

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Emerson / Motorola / Force Computers ISCSI-1* at our website: *Click HERE*

SYS68K/ISCSI-1
USER'S MANUAL

Revision No. 0
NOVEMBER 1992

FORCE COMPUTERS Inc./GmbH
All Rights Reserved

This document shall not be duplicated, nor its contents used
for any purpose, unless express permission has been granted.

Copyright by FORCE COMPUTERS©

INTRODUCTION

This page was intentionally left blank

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. FORCE COMPUTERS makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. FORCE COMPUTERS reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

FORCE COMPUTERS assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of FORCE COMPUTERS GmbH/Inc.

FORCE COMPUTERS does not convey to the purchaser of the product described herein any license under the patent rights of FORCE COMPUTERS GmbH/Inc, nor the rights of others.

FORCE COMPUTERS Inc.
3165 Winchester Blvd.
Campbell, CA 95008-6557
U.S.A.

Phone : (408) 370-6300
FAX : (408) 374-1146

FORCE COMPUTERS GmbH
Prof.-Messerschmitt-Str. 1
D-8014 Neubiberg/Munich
West Germany

Phone : (089) 608 14-0
Telex : 524190 forc-d
FAX : (089) 609 77 93

FORCE COMPUTERS FRANCE Sarl
11, rue Casteja
92100 Boulogne
France

Phone : (1) 4620 37 37
Telex : 206 304 forc-f
FAX : (1) 4621 35 19

FORCE Computers UK Ltd.
No. 1 Holly Court
3 Tring Road
Wendover
Buckinghamshire HP22 6PE
England

Phone : (0296) 625456
Telex : 838033
FAX : (0296) 624027

This page was intentionally left blank

Table of Contents

1.0 General Information.....1-1

1.1 Hardware Features of the SYS68K/ISCSI-1.....1-3

1.2 The Hardware Functions.....1-4

1.2.1 The Local 68010 CPU.....1-4

1.2.2 The SCSIbus Interface.....1-5

1.2.3 The Floppy Disk Interface.....1-6

1.2.4 The PI/T 68230.....1-6

1.2.5 The Dual Ported RAM.....1-7

1.2.6 The VMEbus Interface.....1-8

1.2.7 The Bus Interrupter Module.....1-8

1.2.8 The Optional Back Panel.....1-9

2.0 Specification of the SYS68K/ISCSI-1.....2-1

3.0 Ordering Information.....3-1

List of Figures

Figure 1-1: Photo of the SYS68K/ISCSI-1.....1-2

This page was intentionally left blank

1.0 General Information

The SYS68K/ISCSI-1 board is a high performance intelligent SCSIbus controller board which fully supports the SCSI standard.

Up to four floppy drives can be controlled locally, without using the SCSIbus.

The board provides local intelligence with a 68010 CPU, 68450 DMAC, SCSI Bus Controller (SCSIBC) and Floppy Disk Controller (FDC). A 128Kbyte Dual Ported RAM (DPR) is used to store data and to interface the SYS68K/ISCSI-1 board. Highest throughput is guaranteed by using a 10MHz 68450 DMAC and the NCR 5386S SCSIBC.

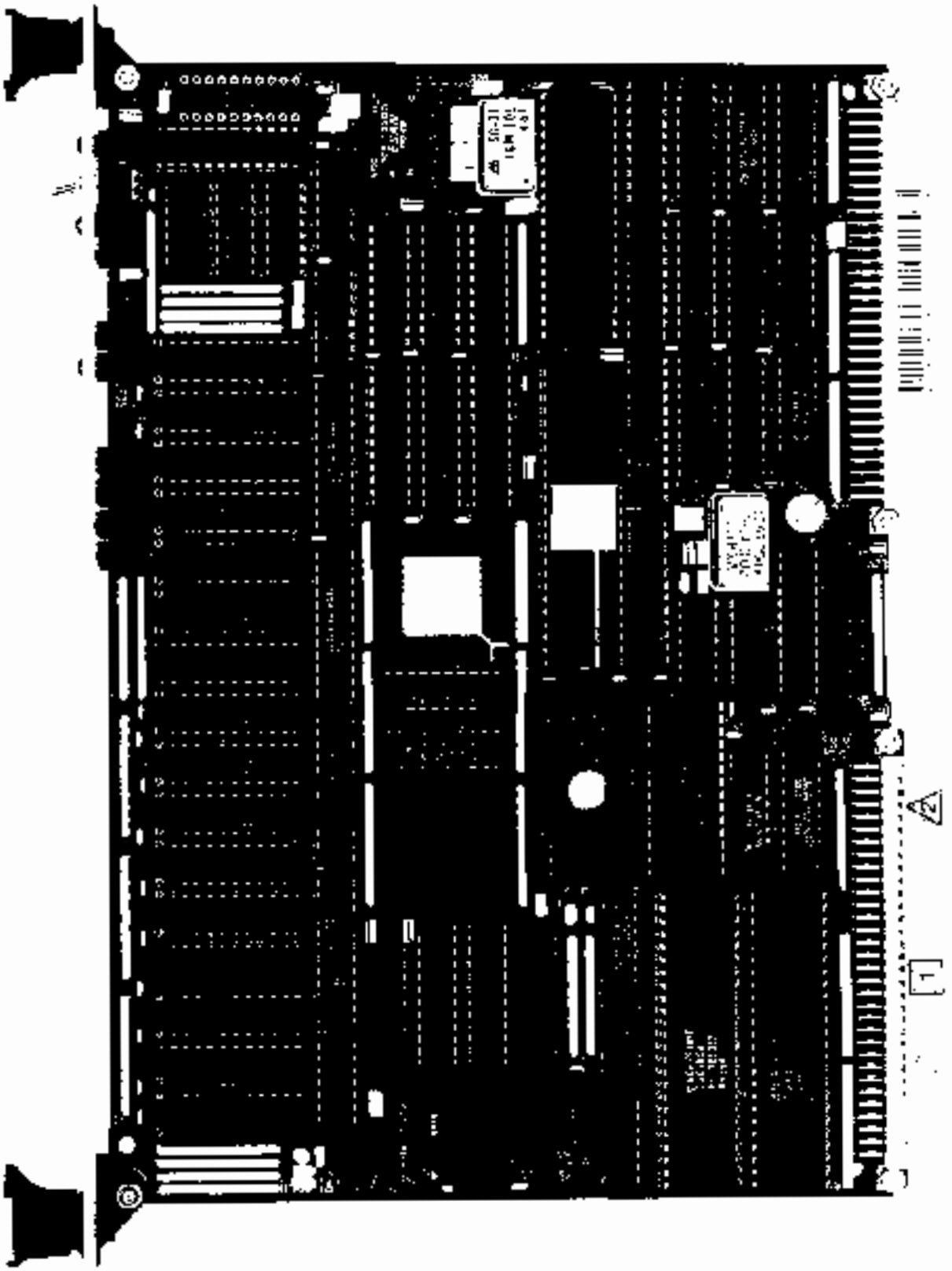
On a single ended SCSIbus, the board works under the powerful firmware (128Kbyte EPROM area) as initiator or as target.

The SYS68K/ISCSI-1's own SCSIbus I.D. is software controlled. Four floppy drives are fully firmware supported, and can also be accessed via the SCSIbus in the target mode.

The SYS68K/ISCSI-1 contains a VMEbus Rev.C/IEEE P1014 compatible interface to communicate to the host CPUs via its 128Kbyte DPR. The access address and the Address Modifier code are jumper selectable. A Bus Interrupter Module - BIM 68153 - is installed on the board to support fully asynchronous operation with the 4 different software programmable interrupt request channels.

The firmware of the SYS68K/ISCSI-1 described in the Firmware User's Manual handles all activities to/from the SCSIbus and the floppy interface. For special applications, the source code of the firmware is optionally available.

Figure 1-1: Photo of the SYS68K/ISCSI-1



1.1 Hardware Features of the SYS68K/ISCSI-1

- 68010 CPU for local control (10MHz)
- 68450 DMA Controller for local transfers (10MHz)
- Dual Ported 128Kbyte 0 wait state static RAM between the VMEbus and the local CPU
- SCSIbus interface built with the NCR 5386S SCSIbus controller. Programmable as an initiator or target
- Transfer rate via the SCSIbus up to 1.5Mbyte/s
- SBUGART compatible floppy interface with the WD1772 FDC. Up to 4 floppy drives can be controlled independent of the SCSIbus
- All I/O signals available on P2 connector
- 4 different interrupt request signals to the VMEbus. Each channel contains a software programmable IRQ level (1 to 7) and vector
- Local parallel interface for controlling and monitoring board functions
- VMEbus Rev.C/IEEE P1014 compatible interface A24:D16, D8
- Watchdog timer controlling correct functions of on-board hard and software
- Status and control LEDs for monitoring local activities
- High level handling firmware for communication, selftest, data caching/hashing and control

1.2 The Hardware Functions

The local CPU reacts on the commands and initialisation parameters within the 128Kbyte DPR. Constant program run times are guaranteed through the special hardware logic providing zero wait state operation from the DPR, independent of the accesses from the VMEbus to the DPR.

The SYS68K/ISCSI-1 consists of self-test functions as well as a hardware watchdog timer which controls the activities of the 68010 CPU running with 10MHz.

User supplied programs can be loaded into the DPR and executed by the local CPU to adapt and extend board functionality.

The local CPU controls the SCSIbus and the floppy disk interface via local interrupts, and communicates to the host CPU via the DPR or via interrupt request to the VMEbus, generated by a Bus Interrupter Module.

The I/O signals to be supported through the NCR 53C65 are terminated on the PC board. The terminators can be removed in order to be able to connect more than 3 SCSI devices.

1.2.1 The Local 68010 CPU

A 10MHz 68010 CPU is installed on the SYS68K/ISCSI-1 to control the data traffic between the serial I/O channels and the VMEbus host CPU(s).

Two EPROMs with a maximum capacity of 128Kbyte are installed on the SYS68K/ISCSI-1 to hold the handling firmware. Constant zero wait state operation from the EPROM guarantees maximum CPU throughput and a fixed program run time.

The 128Kbyte Dual Ported RAM is also accessible without the insertion of wait states by using a CPU clock synchronized arbitration mechanism. The accesses from the CPU to the DPR are not delayed if a VMEbus access is pending or being executed.

A local timer, included in the PI/T, is used to interrupt the CPU for task scheduling, command interpretation and execution.

The CPU and all I/O devices can be RESET through a SYSTEM reset via the SYSRESET signal of the VMEbus or by accessing a dedicated location within the DPR reserved for this function.

1.2.2 The SCSIbus Interface

The SYS68K/ISCSI-1 contains an NCR 5386S SCSIbus controller and an NCR 8310 SCSIbus driver/transceiver. These chips provide the following features:

- Support of the ANSI X3T9.2 SCSI standard
- Asynchronous data transfer to 1.5Mbyte per second
- Support of both initiator and target modes
- Parity generation with optional checking
- Support of arbitration
- Control of all bus signals except RESET
- Doubly-buffered data register
- Versatile MPU bus interface
- Memory or I/O mapped MPU interface
- DMA or programmed I/O transfer
- 24-bit internal transfer counter
- Programmable (re)selection timeouts
- Interrupt of MPU on all bus conditions requiring service
- SCSI pass parity optional with checking
- 48mA driver

The NCR 5386S SCSI Bus Controller communicates with the 68010 CPU and the 68450 DMAC as a peripheral device. The SCSI Bus Controller is controlled by reading and writing the internal registers which are addressed via the local address bus.

Since the SCSI Bus Controller interrupts the MPU when it detects an SCSIbus condition that requires servicing, the MPU is free from polling or controlling any of the SCSIbus signals.

For high speed data transfer, the SCSI Bus Controller communicates directly with the Dual Ported RAM via the DMA Controller. In this mode, the data transfer rate will be a maximum of 1.5Mbyte/s. The minimum guaranteed constant data throughput of the ISCSI-1 board is 1.3Mbyte/s, assuming an equivalent SCSI device data transfer rate.

1.2.3 The Floppy Disk Interface

To provide easy connection from the ISCSI-1 board to floppy drives, the ISCSI-1 board includes a floppy disk interface.

The FDC WD1772 is able to control up to four floppy drives (3", 3 1/2" and 5 1/4").

To allow the connection of all SHUGART-compatible floppy disk drives, the Drive Select signals 0 to 3, the Side Select signal and the single/double density signal are software programmable.

1.2.4 The PI/T 68230

A 68230 Parallel Interface and Timer Chip is installed on the SYS68K/ISCSI-1 to control and display the status of all on-board activities. The PI/T is also used to force and monitor the interrupt request lines to the Bus Interrupter Module, which initiates the interrupts to the VMEbus (under control of the host CPU).

One handshake pin is used to interrupt the local CPU if the host CPU accesses a defined location within the DPR. One output signal is used to force the SYSFAIL signal of the VMEbus if an onboard error has been detected or if the board initializes the DPR after RESET or Power up.

The timer, also included in the PI/T, is the time base for the onboard handling firmware and the scheduler for the macro commands.

A watchdog timer, for processor control, is installed on the board to detect software or hardware errors independent from the onboard CPU. For this purpose, one output of the PI/T is used to retrigger the watchdog timer within defined time frames.

If the onboard CPU does not work properly, or if the hardware isn't working correctly, the timer will not be retriggered, and the SYSFAIL signal of the VMEbus will be activated. The host CPU then can initiate a software controlled RESET for the ISCSI, or start other maintenance activities.

The SCSIbus RESET is controlled by the PI/T. One input of the PI/T indicates the state of the SCSIbus RESET, and one output controls the SCSIbus RESET signal.

The FDC 1772 single/double density selection is made via one PI/T output.

1.2.5 The Dual Ported RAM

128Kbyte of Dual Ported Static RAM with 45ns access time are installed on the SYS68K/ISCSI-1 to service all applications requiring fast operations and large amounts of data areas.

The local 68010 CPU runs without the insertion of wait states out of the DPR, because a CPU clock synchronised arbitration logic and a full buffered and latched VMEbus interface is installed on the SYS68K/ISCSI-1. Between two CPU access cycles, a VMEbus cycle is serviced and completed. On VMEbus Read cycles, the data pattern is latched, and the internal cycle of the DPR is aborted while the VMEbus cycle is decoupled.

A partition of the DPR is reserved for the local CPU for vector storage, the program counter, and temporary buffers. This partition is used from the VMEbus side for programming the BIM and initiating an interrupt, which will be handled from the onboard CPU, or driving a local RESET.

The access address and the Address Modifier code(s) are jumper selectable in 128Kbyte increments within the standard address range (A24:D16,D8). The access times of the DPR depend on the accesses made by the local CPU while the local 68010 has priority over VMEbus accesses.

1.2.6 The VMEbus Interface

A full VMEbus Rev.C/IEEE P1014 compatible interface is installed on the SYS68K/ISCSI-1 to allow an access to the DPR and the Bus Interrupter Module.

The 16-bit data width (D16,D8) of the DPR and the decoding of the standard address range (A24) allows easy installation in all VMEbus environments.

During Power-up and after a RESET has been executed from the local CPU, the SYS68K/ISCSI-1 drives the VMEbus signal SYSFAIL active to signal each board in the VMEbus environment that the board is not ready or has detected a malfunction.

A RESET for the local CPU can be initiated by accessing a dedicated address within the 128Kbyte boundary of the DPR. All local devices as well as the CPU will be reset through this access.

An interrupt to the local CPU can be forced by accessing another location within the DPR, signalling the on-board processor that a command has been given, or that an exception has to be taken.

The Dual Ported RAM can be accessed at least every 640ns because this is the worst case cycle time. The data transfer rate to/from the SYS68K/ISCSI-1 is 3 to 4Mbyte/s including the VMEbus protocol.

	Access Time	Cycle Time
best case	330	400
average	430	500
worst case	560	630

1.2.7 The Bus Interrupter Module

To allow fully asynchronous operation, the SYS68K/ISCSI-1 contains a Bus Interrupter Module - BIM 68153 - providing 4 individually programmable interrupt channels. Each channel is able to force an interrupt request to the VMEbus. For each channel, the IRQ level (1 to 7) as well as the interrupt vector is fully software programmable.

The local CPU forces the requests to the BIM and the host CPU can program the interrupt vector and the level. This allows dynamic change of the interrupt level and vector in multi-processor environments.

1.2.8 The Optional Back Panel

A back panel which can be plugged into the P2 connector of the SYS68K/ISCSI-1 board is optionally available. Included on this board is a 50-pin 2-row connector for the SCSIbus and a 34-pin 2-row connector for the floppy disk interface.

This page was intentionally left blank

2.0 Specification of the SYS68K/ISCSI-1

Local CPU	68010 with 10MHz clock frequency
EPROM	128Kbyte maximum capacity 0 Wait State operation
Dual Ported RAM	128Kbyte capacity using static RAMs 0 Wait State operation from local CPU 330 ns best case VMEbus access time 430 ns average VMEbus access time 560 ns worst case VMEbus access time
SCSIbus Interface	NCR 53685 providing initiator, target mode asynchronous and synchronous modes asynchronous data rate up to 1.5sbyte/s
Floppy Disk Interface	SHUGART-compatible Up to 4 drives (3", 3 1/2", 5 1/4")
VMEbus Interface	Full Rev. C and IEEE P1014 compatible A24:D16,D8 mode 4 IRQs with SW programmable level (1 to7) and vector Access Address jumper selectable in 128Kbyte increments SYSFAIL* supported
Handling Firmware	in EPROM with macro commands for all I/O channels installed
Power Requirements	+5V : 5.6 A (max) (Power Backplane or +12V : 0.0 A (max) power connection -12V : 0.0 A (max) on P2 necessary)
Operating Temperature	0 to 50 Degrees C
Storage Temperature	-50 to +85 Degrees C (non-operating)
Relative Humidity	0 to 90% (non-condensing)
Dimensions	233 x 160mm 9.2" x 6.3"

This page was intentionally left blank

3.0 Ordering Information

SYS68K/ISCSI-1 Part No. 300020	Intelligent SCSIbus Controller board including firmware and documentation.
SYS68K/ISCSI-1BPS Part No. 300021	Back panel for the SYS68K/ISCSI-1 board providing SCSIbus connector and floppy drive connector
SYS68K/ISCSI-1/UM Part No. 800114	User's Manual for the SYS68K/ISCSI-1
SYS68K/ISCSI-1/SC Part No. 800022	Source Code of the SYS68K/ISCSI-1 handling firmware, including documentation

This page was intentionally left blank

INSTALLATION

This page was intentionally left blank

W A R N I N G

TO AVOID MALFUNCTIONS AND COMPONENT DAMAGES, PLEASE READ THE COMPLETE INSTALLATION PROCEDURE BEFORE THE BOARD IS INSTALLED IN A VMEBUS ENVIRONMENT.

This page was intentionally left blank

Table of Contents

1.0 General Overview.....1-1

1.1 The Function Switch Positions.....1-1

1.2 Connection of I/O Devices.....1-2

1.3 Base Address Selection and AM Decoding.....1-4

1.4 Interrupts.....1-4

2.0 Installation in the Rack.....2-1

2.1 Power On.....2-1

2.2 The SYS68K/ISCSI-1 On-Board Selftest.....2-2

List of Tables

Table 1-1 The P2 Pin Assignment.....1-2

Table 1-2 The X1 Pin Assignment.....1-3

Table 1-3 The X2 Pin Assignment.....1-3

This page was intentionally left blank

1.0 General Overview

Easy installation of the SYS68K/ISCSI-1 is provided as the board is shipped in a ready-to-operate default configuration. A selftest is executed on reset by the firmware on the board.

Please read the complete installation procedure before mounting the board into a VMEbus backplane.

1.1 The Function Switch Positions

There is a RUN/LOCAL (R/L) toggle switch installed on the front panel.

The two positions of the switch are defined as "UP" and "DOWN". The switch has to be set to "DOWN" for the first installation.

1.2 Connection of I/O Devices

The SCSIbus signals and the floppy disk interface signals are routed to the P2 connector of the SYS68K/ISCSI-1 board. Table 1-1 shows the pin out of the P2 connector. For a detailed description, please refer the to Hardware User's Manual, Chapter 4.8.

If the optional SYS68K/ISCSI-1BPS is used, it must be connector to the P2 connector of the SYS68K/ISCSI-1 board. The pin out of the SCSIbus connector (X1) is listed in Table 1-2 and the pin out of the floppy disk interface connector (X2) is listed in Table 1-3.

Table 1-1: The P2 Pin Assignment

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	DB 0	VCC	N.C.
2	DB 1	GND	N.C.
3	DB 2	N.C.	Drive Select 0
4	DB 3	N.C.	Index
5	DB 4	N.C.	Drive Select 1
6	DB 5	N.C.	Drive Select 2
7	DB 6	N.C.	Drive Select 3
8	DB 7	N.C.	Motor On
9	DB P	N.C.	Direction In
10	GND	N.C.	Step
11	GND	N.C.	Write Data
12	GND	GND	Write Gate
13	TERMPWR	VCC	Track 00
14	GND	N.C.	Write Protect
15	GND	N.C.	Read Data
16	ATN	N.C.	Side Select
17	GND	N.C.	N.C.
18	BSY	N.C.	N.C.
19	ACK	N.C.	GND
20	RST	N.C.	GND
21	MSG	N.C.	N.C.
22	SEL	GND	GND
23	C/D	N.C.	GND
24	REQ	N.C.	N.C.
25	I/O	N.C.	N.C.
26	N.C.	N.C.	N.C.
27	GND	N.C.	RESERVED
28	N.C.	N.C.	RESERVED
29	RESERVED	N.C.	RESERVED
30	RESERVED	N.C.	RESERVED
31	RESERVED	GND	RESERVED
32	RESERVED	VCC	RESERVED

Table 1-2: The X1 Pin Assignment

Pin No.	Signal Mnemonic	Pin Number	Signal Mnemonic
2	DB 0	1	GND
4	DB 1	3	GND
6	DB 2	5	GND
8	DB 3	7	GND
10	DB 4	9	GND
12	DB 5	11	GND
14	DB 6	13	GND
16	DB 7	15	GND
18	DB P	17	GND
20	GND	19	GND
22	GND	21	GND
24	GND	23	GND
26	TERMPWR	25	N.C.
28	GND	27	GND
30	GND	29	GND
32	ATN	31	GND
34	GND	33	GND
36	BSY	35	GND
38	ACK	37	GND
40	RST	39	GND
42	MSG	41	GND
44	SEL	43	GND
46	C/D	45	GND
48	REQ	47	GND
50	I/O	49	GND

Table 1-3: The X2 Pin Assignment

Pin No.	Signal Mnemonic	Pin Number	Signal Mnemonic
2	N.C.	1	GND
4	N.C.	3	GND
6	Drive Select 0	5	GND
8	Index	7	GND
10	Drive Select 1	9	GND
12	Drive Select 2	11	GND
14	Drive Select 3	13	GND
16	Motor On	15	GND
18	Direction In	17	GND
20	Step	19	GND
22	Write Data	21	GND
24	Write Gate	23	GND
26	Track 000	25	GND
28	Write Protect	27	GND
30	Read Data	29	GND
32	Side Select	31	GND
34	N.C.	33	GND

1.3 Base Address Selection and AM Decoding

The default setup of the SYS68K/ISCSI-1 board is as follows:

Base Address	$A00000
End Address	$A1FFFF

Address Modifiers are set to accept privileged and non-privileged standard data accesses (A24:D16,D8)

If a different setup is required, please refer to the Hardware User's Manual.

1.4 Interrupts

All interrupt levels and vectors are software programmable on the SYS68K/ISCSI-1 so no hardware setup is required.

2.0 Installation in the Rack

The board is configured to be mounted into a VMEbus rack at any one of the slots 2-21.

A reset generator has to be included in another slot.

Caution:

A) The VMEbus rack has to include a J1 and a J2 backplane.

B) Switch power off before installing the board to avoid electrical damages to the components.

C) The board has the be plugged in and the screws of the front panel must be turned on to guarantee proper installation.

D) No connections are allowed on the P2 backplane on rows A and C (i.e. VMXbus or VSBbus connection).

2.1 Power On

If the board is installed correctly, the following sequence will take place:

1) Green RUN LED lights up.

2) During SYSRESET, HALT LED is red.

3) After SYSRESET goes inactive, the HALT LED turns green and the red SYSFAIL LED turns on. At this time the SYSFAIL signal on the VMEbus is also activated.

4) The onboard firmware starts execution of the onboard selftest, which takes about 30 seconds. The status information that is given on the front panel LEDs S1-S4 is described in Chapter 2.2.

5) When the selftest is complete and if no errors have occurred, the SYSFAIL LED turns off and the SYS68K/ISCSI-1 is ready for operation.

If any errors are detected during selftest, the LED S4 turns on and the LEDs S1-S3 show an error code. Please refer to the Software User's Manual for details.

2.2 The SYS68K/ISCSI-1 On-Board Selftest

On power-up or reset, the SYS68K/ISCSI-1 selftest routine is executed in the following manner:

- Test of SCSI controller by write and read back several registers.
- Start SCSI controller self-diagnostic.
- Test the whole local and dual ported memory with read and write bytes, words and long words.
- DMA Controller test with high speed data transfer memory to memory.
- Test the floppy disk controller.

The control of the selftest state and results is provided via the front panel LEDs S1 to S4:

- LED S1 is turned on during the RAM test.
- LED S2 is turned on during the SCSIbus controller test.
- LED S3 is turned on during the floppy disk controller test.
- LED S4 is turned on during the DMA Controller test.

If any error has been found while selftest was active, the LED of the test phase which has generated the error will stay on. After the selftest has been successfully completed, all LEDs are turned off.

Example: After completion of the selftest routine, the LED S3 remains on. This state indicates a hardware error on the WD1772 Floppy Disk Controller.

HARDWARE USER'S MANUAL

This page was intentionally left blank

Table of Contents

1.0 General Information.....1-1

2.0 General Operation.....2-1

3.0 Functional Groups of the SYS68K/ISCSI-1.....3-1
3.1 The Front Panel.....3-2
3.2 The I/O Connector.....3-2

4.0 The Local CPU Hardware.....4-1
4.1 Implementation of the Local Processor 68010.....4-1
4.1.1 Reset and Bootup of the Local CPU.....4-2
4.1.2 Decoding of the Local Address Space.....4-2
4.1.3 The Local Interrupt Structure.....4-3
4.1.4 The Local Bus Error Structure.....4-3
4.2 The EPROMs.....4-4
4.3 The Parallel Interface and Timer 68230 (PI/T).....4-6
4.3.1 Control of the Front Panel LEDs S1-S4.....4-8
4.3.2 Control of the SYSFAIL* Signal and the Watchdog Timer.....4-9
4.3.3 Control of the VMEbus Interrupt Requests.....4-10
4.3.4 The Interrupt Trigger Input.....4-11
4.3.5 Reading the Run/Local Switch.....4-11
4.3.6 The Timer Selection of the PI/T.....4-11
4.3.7 The SCSIbus Reset Control.....4-12
4.3.8 The Single/Double Floppy Density Selection...4-12
4.4 The Direct Memory Access Controller 68450 (DMAC)....4-13
4.4.1 The DMAC Implementation.....4-13
4.4.2 Addressing of the DMAC.....4-13
4.4.3 The DMAC Interrupt Scheme.....4-18
4.4.4 The DMAC Summary.....4-18
4.5 The SCSIbus Controller NCR 5386S (SCSIBC).....4-19
4.5.1 SCSIbus Controller Summary.....4-20
4.6 The Floppy Disk Controller WD 1772 (FDC).....4-21
4.6.1 Floppy Disk Controller Summary.....4-22
4.7 The Control Register.....4-23
4.8 The SYS68K/ISCSI-IBPS.....4-24
4.9 The Dual Ported RAM as Local Memory.....4-27
4.9.1 Local CPU Access to the DPR.....4-27
4.9.2 Local CPU Read-Modify-Write Cycles.....4-28

5.0 The SCSIbus Description.....5-1
5.1 The SCSIbus Configuration.....5-1
5.2 The SCSIbus Description.....5-3
5.3 The SCSIbus Signal Termination.....5-4
5.4 The SCSIbus Termination Power.....5-6

Table of Contents cont'd

6.0 The VMEbus Interface Hardware.....6-1
6.1 VMEbus Access to the Board.....6-1
6.1.1 The Address Modifier Code Selection.....6-2
6.1.2 The Board Base Address Selection.....6-4
6.1.3 The Run/Local Switch.....6-6
6.1.4 The Access LED.....6-6
6.2 The Address Map of the VME Address Range.....6-7
6.3 The Dual Ported RAM (DPR) as VMEbus Memory.....6-8
6.4 The Reset Trigger Call.....6-8
6.5 The Interrupt Trigger Call.....6-8
6.6 The Bus Interrupter Module.....6-9
6.7 The Status Register.....6-10

List of Figures

Figure 1-1 Photo of the SYS68K/ISCSI-1 Board.....1-0

Figure 2-1 Block Diagram of the SYS68K/ISCSI-1.....2-0

Figure 3-1 The Front Panel of the SYS68K/ISCSI-1.....3-3

Figure 4-1 Location Diagram of the Jumperfield B19.....4-5
Figure 4-2 Schematic of the SYS68K/ISCSI-1BP.....4-26

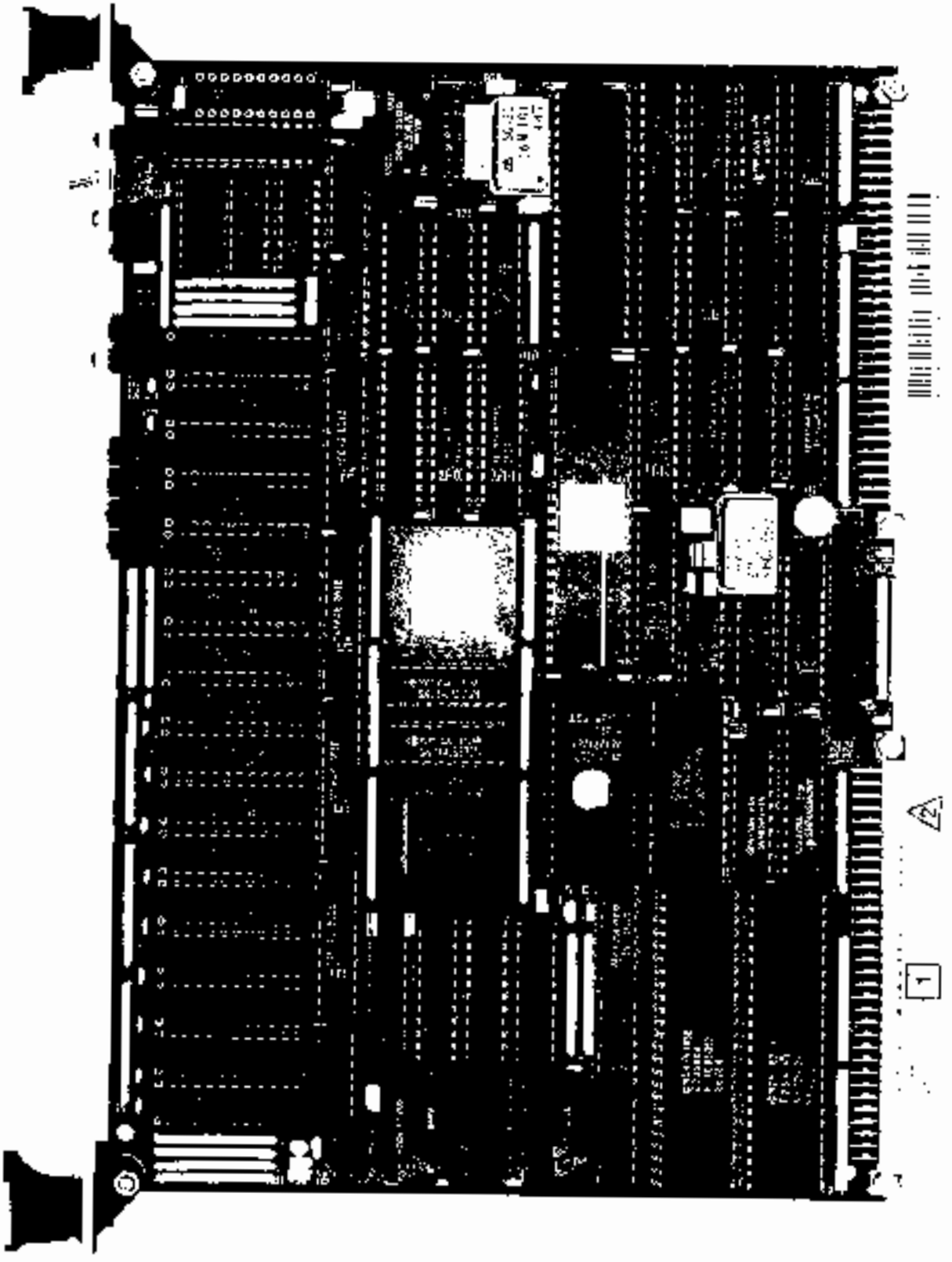
Figure 5-1 SCSI I.D. Bits.....5-1
Figure 5-2 Sample SCSI Configurations.....5-2
Figure 5-3 Location of the SCSIbus Terminators.....5-5
Figure 5-4 Location Diagram of the Jumperfield B24.....5-7

Figure 6-1 Location Diagram of the Jumperfield B22.....6-3
Figure 6-2 Location Diagram of the Jumperfield B21.....6-5

List of Tables

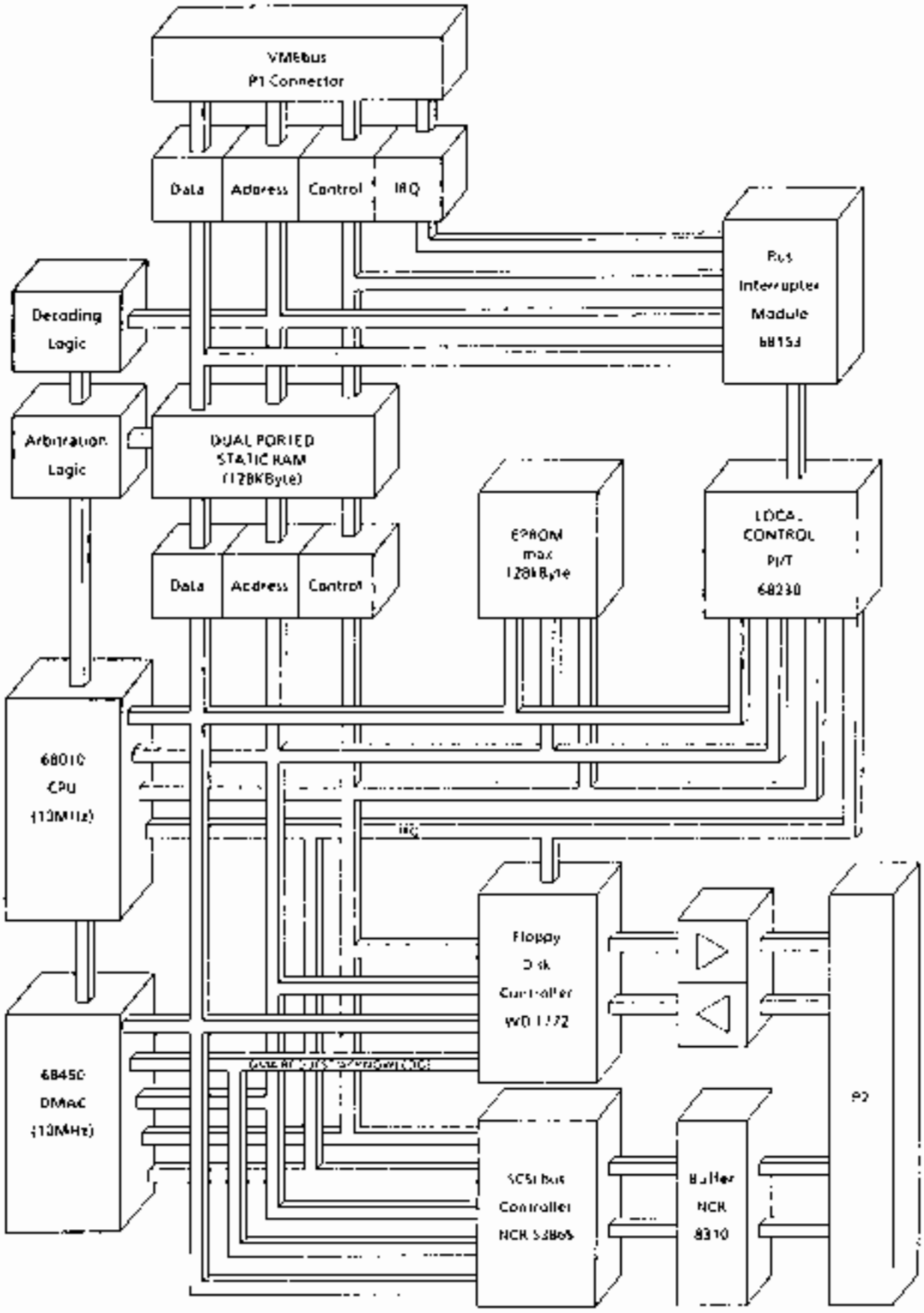
Table 4-1 The PI/T Address Map.....4-7
Table 4-2 Register Model of the DMAC.....4-14
Table 4-3 SCSIBC NCR 5386S Register Address Map.....4-20
Table 4-4 FDC WD 1772 Register Address Map.....4-21
Table 4-5 The ISCSI-1 P2 Pin Assignment.....4-24
Table 4-6 The ISCSI-1BPS X1 Pin Assignment.....4-25
Table 4-7 The ISCSI-1BPS X2 Pin Assignment.....4-25

Figure 1-1: Photo of the SYS69K/ISCSI-1



This page was intentionally left blank

Figure 2-1: Block Diagram of the SYS68K/ISCSI-1



This page was intentionally left blank

2.0 General Operation

The SYS68K/ISCSI-1 is an intelligent VMEbus controller board providing an SCSIbus interface and a floppy disk interface.

The SYS68K/ISCSI-1 board includes 48mA drivers and receivers for the on-board terminated single ended SCSIbus and SHUGART compatible floppy interface. The buffered I/O signals are routed to the P2 backplane connector.

The NCR 53865 SCSIbus Controller supports the asynchronous and synchronous data transfer via the SCSIbus.

The WD 1772 floppy disk controller/formatter provides control of a maximum of four floppy disk drives (3", 3 1/2", 5 1/4").

The 68010 CPU chip, running at 10MHz, executes local software performing the intelligent I/O functions. The local I/O devices can be directly accessed by the CPU. A 68230 PI/T is installed for local control and timer functions.

The four channel 68450 Direct Memory Access Controller (DMAC) controls the high speed I/O data transfer.

The on-board SRAM is a Dual Ported RAM, which is accessible for the local CPU and DMAC and for the VMEbus master. The Dual Ported RAM (DPR) has 128Kbyte of wait-cycle-free memory space for the local CPU, as well as the medium of interface to the VMEbus master.

The SYS68K/ISCSI-1 board includes the function of an interrupter to the VMEbus. Up to four interrupts can be operated independent from each other.

The SYSFAIL* signal of the VMEbus is supported, and status readout and a restart call are included in the hardware.

The general block diagram of the SYS68K/ISCSI-1 is shown in Figure 2-1.

This page was intentionally left blank

3.0 Functional Groups of the SYS68K/ISCSI-1

The SYS68K/ISCSI-1 consists of the following functional groups:

The 68010 CPU with ROM

The 68454 DMAC

Local control and decoding

The SCSIbus interface

The floppy disk interface

VMEbus interface and decoding

Dual Ported RAM (DPR)

Bus Interrupter Module with status register

Reset and interrupt trigger call

The watchdog timer

Interrupt generator

The detailed description has been organised in two chapters: the local CPU hardware is described in Chapter 4 and the VMEbus interface is described in Chapter 5.

The description of the front panel and the P2 connector is as follows:

3.1 The Front Panel

The SYS68K/ISCSI-1 is delivered with a front panel which also includes lever handles.

There is a RUN/LOCAL switch with LED indicators in the top position. When the switch is down, the board can be accessed on the VMEbus and the green RUN LED indicates this state. When the RUN/LOCAL switch is up, the board cannot be accessed from the VMEbus. In this situation the red LED, labelled LOCAL, lights up. The RUN and LOCAL LEDs can only light up one at a time.

The bi-colour LED with the label HALT will show a green light during the normal operation of the local CPU. It turns to red when the local CPU is in the HALT state and also during the RESET of the board.

The yellow LED, labelled SEL for select, lights up each time the SYS68K/ISCSI-1 board is accessed from the VMEbus during data cycles and interrupt acknowledge cycles of the VMEbus.

The yellow FAIL indicator LED reflects the state of the on-board watch-dog timer and blinks during the high-speed DMAC SCSIbus transfers. If the FAIL LED lights up for more that 2 seconds, this means that the watch-dog timer has run out of time.

The yellow status LEDs labelled, S1 - S4, are controlled by the local PI/T device and are fully software controlled. These LEDs are all turned off at the RESET of the board.

Figure 3-1 shows the front panel of the SYS68K/ISCSI-1 with the switches and LEDs.

3.2 The I/O Connector

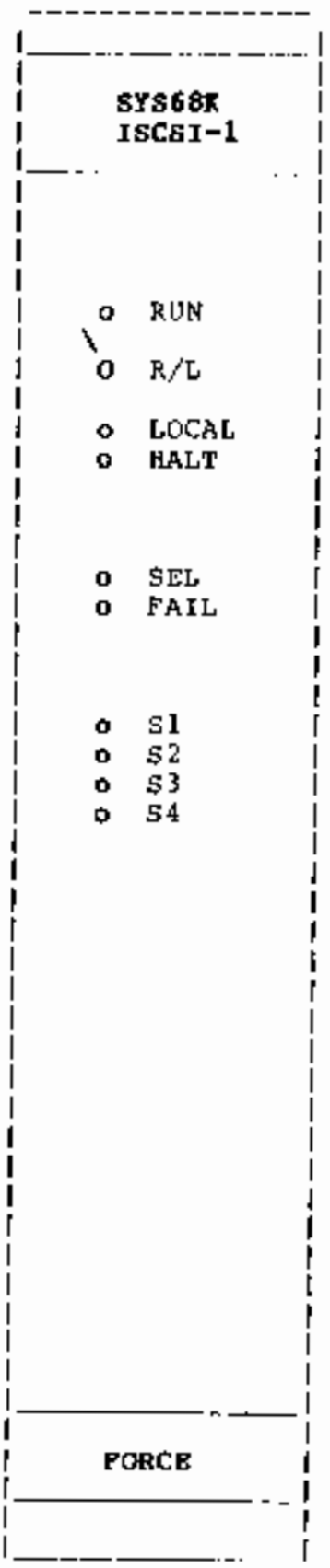
The VMEbus definition of the Double Eurocard form factor includes two 96 pin bus connectors.

The P1 connector is the primary VMEbus interface. The P2 connector carries the extension bus signals and power supply connections via the b row. Rows a and c are for user I/O signals.

The SYS68K/ISCSI-1 board supplies buffered SCSI and floppy disk interface via the P2 connector. There are 25 pins providing the floppy disk I/O signals. 10 pins of the P2 connector are reserved for later applications.

The detailed pin-out is shown in section 4.8.

Figure 3-1 The Front Panel of the SYS68K/ISCSI-1



This page was intentionally left blank

4.0 The Local CPU Hardware

There is a 68010 microprocessor clocked at 10MHz working on the SYS68K/ISCSI-1.

The local CPU chip has an address and data bus totally independent of the VMEbus.

The local structure includes:

- Address decoding
- Reset driving
- Boot-up support
- Interrupt logic
- Bus error time-out control
- The DMAC (68450)
- The EPROMs
- The PI/T (68230)
- The SCSIbus Controller (5386S)
- The FDC (1772)
- The read back register for floppy disk control
- Access to the Dual Ported RAM

4.1 Implementation of the Local Processor 68010

The local structure is built around the microprocessor 68010 which is fast enough to control the SCSIbus, the floppy disk interface and to handle data conversion and transport commands.

The 68010 works at a clock rate of 10MHz, therefore no wait cycles are needed if the Dual Ported RAM or the EPROM are accessed by the processor. This gives maximum yield of the processors power.

The following subchapters describe:

- Reset and bootup
- Decoding
- Interrupt structure
- Bus error

4.1.1 Reset and Bootup of the Local CPU

The local CPU hardware can be reset in two different ways.

If the SYSRESET* signal of the VMEbus is asserted, then the local CPU as well as the local peripheral devices and the VMEbus interface will be reset.

If a byte Read access is performed to the reset trigger address, then the local CPU and the local peripheral devices will be reset. The reset trigger address is $1801 + board base address. Default is: A01801.

For boot-up, upon reset (or restart), the decoding logic is changed, so that the local EPROMs will be selected instead of the RAM. The first Write cycle of the local CPU introduces normal address decoding.

4.1.2 Decoding of the Local Address Space

The local decoding structure is fixed. The address map is as follows:

$000000	-	$01FFFF	read/write	Dual Ported RAM
$C40000	-	$C4001F	byte only	SCSIbus controller
$C80000	-	$C800FF	read/write	DMAC
$CC0000	-	$CC0007	odd byte only	FDC
$CC0009	-	$CC0009	odd byte only	Control register
$D00000	-	$D0003F	odd byte only	PIT
$E00000	-	$E7FFFF	read only	EPROMs

4.1.3 The Local Interrupt Structure

The local interrupt structure is organized as follows:

IRQ Level	IRQ Source	Vector if B41 inserted	Source if B41 removed
1	P3 Pin #13	AV1 Autovector	AV1 Autovector
2	DMAC	DMAC	AV2 Autovector
3	SCSIBC	AV3 Autovector	AV3 Autovector
4	FDC	AV4 Autovector	AV4 Autovector
5	PI/T Timer	PI/T Timer Vector Reg.	PI/T Timer Vector Reg.
6	--	--	--
7	PI/T Port	PI/T Port Vector Reg.	PI/T Port Vector Reg.

Default configuration: B41 jumper removed.

The PI/T port interrupt can be used under software control to cause non-maskable (Level 7) interrupts if the watchdog timer elapses and/or if the VMEbus interrupt trigger call occurs.

4.1.4 The Local Bus Error Structure

There is a time-out counter installed to provide for bus error generation in the case of accessing non-decoded memory locations.

4.2 The EPROMs

The SYS68K/ISCSI-1 is delivered together with firmware, stored in two 27128 EPROMs. These EPROMs must have a maximum access time of 150ns. All EPROM access cycles are without wait states.

Other EPROM sizes are configurable by changes in the jumpering at B19.

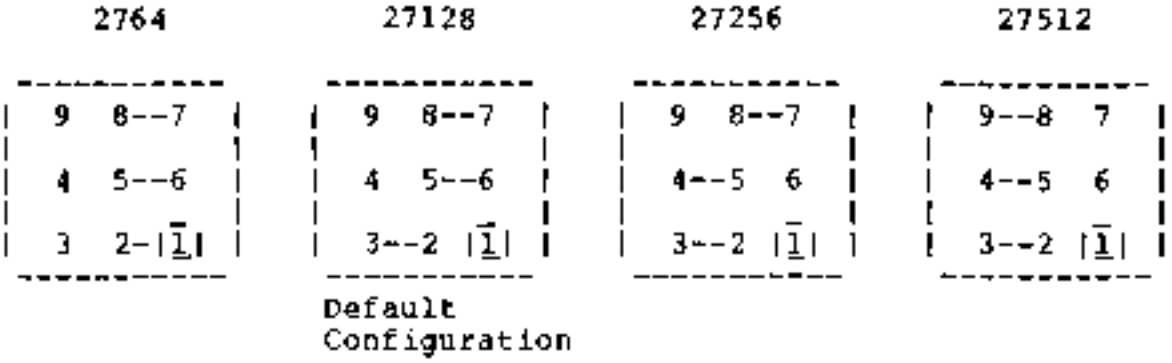
The two EPROMs are stacked above each other, the lower byte (bits 0-7) in the lower socket, the upper byte (bits 8-15) in the upper socket.

The base address of the EPROMs is $E00000, and only read accesses are allowed.

The first two long locations in the EPROM pair must contain the boot-up information as follows:

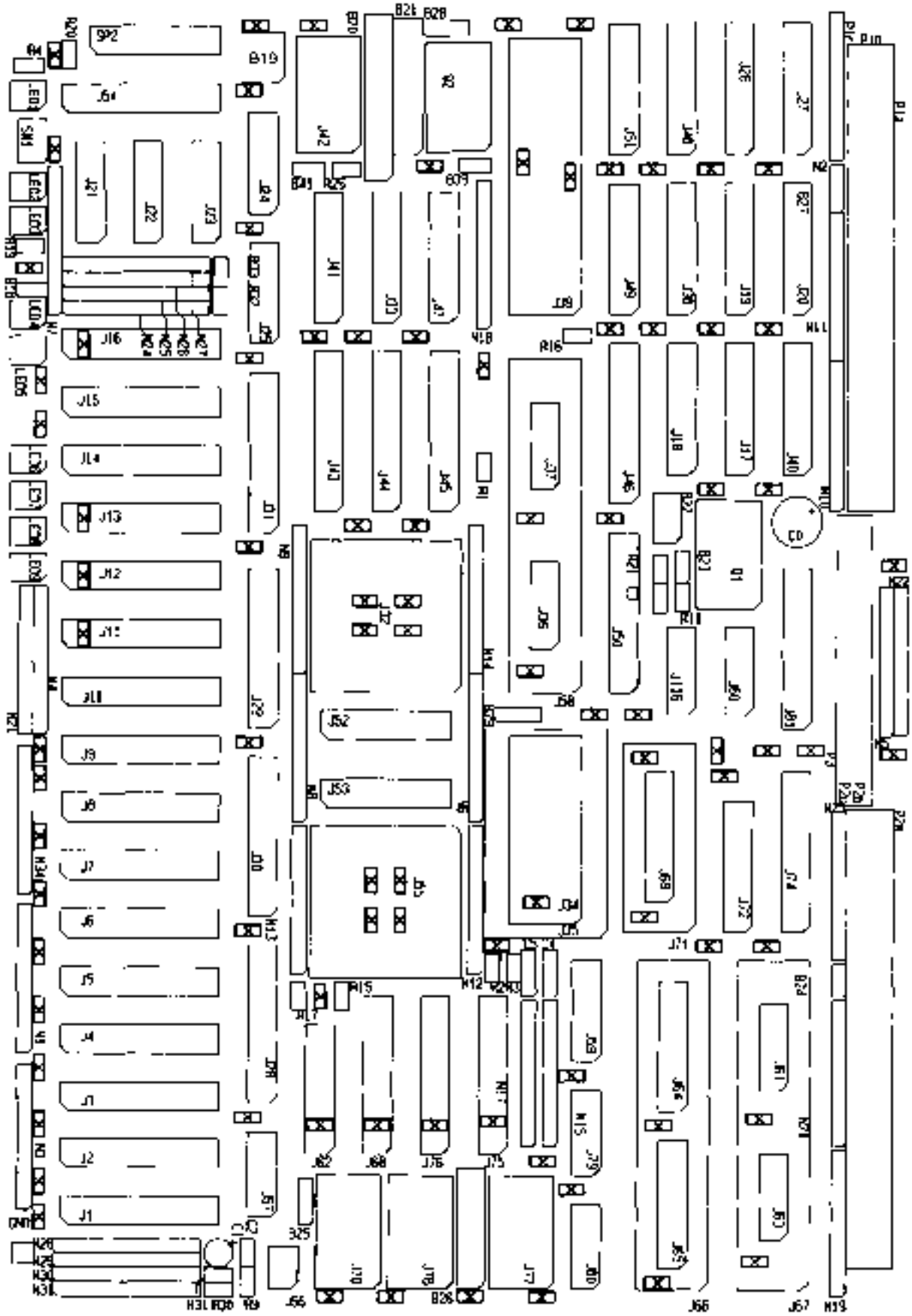
```
E00 000 : Initial Supervisor Stack Pointer
E00 004 : Initial Program Counter
```

Configuration of the B19 jumper area:



Default Configuration

Figure 4-1: Location Diagram of the Jumperfield B19



4.3 The Parallel Interface and Timer 68230 (PI/T)

There is a PI/T device installed for local control on the SYS68K/ISCSI-1.

The PI/T can be accessed only by the local CPU. The address range is $D00001 to $D0003F - odd byte only.

Table 4-1 gives the address map of the PI/T.

The following subchapters describe

- Control of the front panel LEDs
- Control of the SYSFAIL* signal and the watchdog timer
- Control of the VMEbus interrupt requests
- The interrupt trigger input
- Reading the RUN/LOCAL switch
- Reading the B42 jumper selection
- The timer section of the PI/T
- The SCSIbus reset control
- The single or double floppy density selection

The PI/T port interrupt drives level 7 interrupts and the timer is connected to drive level 5 interrupts to the CPU. Both have to be configured (by software) to support interrupt vectors.

Table 4-1: The P1/T Address Map

Base Address : $D00000

Address HEX	Offset	Reset Value	Label	Description
D00001	01	00	PITPGCR	Port General Control Register
D00003	03	00	PITPSRR	Port Service Request Register
D00005	05	00	PITPADDR	Port A Data Direction Register
D00007	07	00	PITPBDDR	Port B Data Direction Register
D00009	09	00	PITPCDDR	Port C Data Direction Register
D0000B	0B	0F	PITPIVR	Port Interrupt Vector Register
D0000D	0D	00	PITPACR	Port A Control Register
D0000F	0F	00	PITPBCR	Port B Control Register
D00011	11	--	PITPADR	Port A Data Register
D00013	13	--	PITPBDR	Port B Data Register
D00015	15	--	PITPAAR	Port A Alternate Register
D00017	17	--	PITPBAR	Port B Alternate Register
D00019	19	--	PITPCDR	Port C Data Register
D0001B	1B	--	PITPSR	Port Status Register
D00021	21	00	PITTCR	Timer Control Register
D00023	23	0F	PITTIVR	Timer Interrupt Vector Reg.
D00025	25	--	PITCPR	Counter Preload Register
D00027	27	--		
D00029	29	--		
D0002B	2B	--		
D0002D	2D	--	PITCNTR	Count Register
D0002F	2F	--		
D00031	31	--		
D00033	33	--		
D00035	35	00	PITTSR	Timer Status Register

4.3.1 Control of the Front Panel LEDs S1-S4

There are four yellow LEDs, labelled S1 to S4, installed on the front panel of the SYS68K/ISCSI-1, that are driven by the PI/T device. These LEDs are fully under control of the local software which can write 1s and 0s to the corresponding PI/T register bits.

LED	Controlled by
S1	Port B Bit 0
S2	Port B Bit 1
S3	Port B Bit 2
S4	Port B Bit 3

For correct operation, Port B has to be programmed to operate in the bit I/O mode, and bits 0, 1, 2 and 3 of the Port B data direction register have to be set (1).

When writing to the Port B data register, bits 0 to 3, a zero causes the light to turn on, and a one causes it to turn off.

4.3.2 Control of the SYSFAIL* Signal and the Watchdog Timer

There is a watchdog timer installed on the SYS68K/ISCSI-1. This is a retriggerable multivibrator with a nominal pulse duration of 20 milliseconds. This device is included to give information should a system breakdown occur.

The output state of the watchdog timer is available on the H3 terminal of the PI/T. The watchdog timer trigger input is connected to the PC4 pin of the PI/T. The high to low transition is the trigger.

If the watchdog is not triggered for more than 20 milliseconds, then its output becomes low (0). Triggering at a rate which is high enough (i.e. every 10 milliseconds), means that the output of the watchdog timer can be kept high (1). In case of a system breakdown, this retrigger does not occur any more, and the watchdog timer becomes low (0).

It is possible to use the H3 terminal of the PI/T not only to read, but also to use as an interrupt input (level 7 to the CPU). This gives an option for system recovery as long as the CPU does not go to HALT.

The SYSFAIL* signal of the VMEbus can be used to broadcast the fact that a board is not functioning because of boot-up, selftest or failure. This signal will not be driven by the ISCSI-1 while SYSRESET* is active. As soon as SYSRESET* is released, SYSFAIL will be asserted (driven to low). This situation can only be changed by software controlled action of the local CPU.

Two output pins of the PI/T device control the SYSFAIL* output: PC1 and PC2. After RESET, both pins are driven high due to pull-up resistors. The same is the case with both pins driven high (1) via software control.

If PC1 is programmed to output low logic level (0), then the SYSFAIL* output to the VMEbus is released. With PC1 = low, SYSFAIL* is never active.

If PC1 is high (1), and PC2 is programmed to output low (0), then the SYSFAIL* output will be asserted if the watchdog timer output is low (0). This should be the runtime configuration. As long as the periodical toggle at the PI/T output pin PC4 occurs every 10 milliseconds, the watchdog timer output stays high (1), and SYSFAIL* will not be asserted by ISCSI-1.

There is one more option:

The watchdog timer output can drive an interrupt request to the VMEbus via channel 3 of the BIM device. If pin PC0 of the PI/T is programmed to output low (0), then the watchdog timer output is connected to set a flip-flop to drive an IRQ to channel 3 of the BIM. The flip-flop will be cleared by hardware during the interrupt acknowledge cycle. In this configuration, the PI/T output PA3 is not available for triggering this IRQ channel.

4.3.3 Control of the VMEbus Interrupt Requests

There is a Bus Interrupter Module 68153 (BIM) included in the SYS68K/ISCSI-1. This device is only accessible from the VMEbus (see chapter 6.6). So, the interrupt levels and vectors can only be programmed by VMEbus masters. The ISCSI-1 local software has no influence on this. The local software decides if and on which channel an interrupt request (IRQ) will be emitted. There are four channels to output an IRQ, while the BIM can distribute them to the seven IRQ level lines of the VMEbus.

The interrupt request signal must be released a very short time after it has been acknowledged. No software is fast enough to do that. Therefore, the IRQ outputs are handled in flip-flops that are asserted by software via a toggle from the flip-flop clock. The state of the four IRQ flip-flops can be read back by the local CPU. This is absolutely necessary, because a second IRQ can only be started to the same channel after the first has been acknowledged.

This affords some dynamic software handling by the local CPU. An IRQ asserted can only be cleared by the interrupt acknowledge cycle or by a SYSRESET* from the VMEbus.

The four IRQ channels are associated and controlled by pins of the PI/T device as follows:

	VMEbus Default Address			
BIM Channel	Control Reg Address	Vector Reg Address	Toggle by PI/T	Readback of IRQ Flip-Flop state via PI/T
0	$A00001	$A00009	PA0	PA4
1	$A00003	$A0000B	PA1	PA5
2	$A00005	$A0000D	PA2	PA6
3	$A00007	$A0000F	PA3	PA7

The PI/T pins PA0/PA3 have to be programmed to outputs, PA4/PA7 are inputs. Before writing to the Port A data direction register, $FF has to be written to the Port A data register to avoid an unwanted output pulse.

4.3.4 The Interrupt Trigger Input

There is an option to trigger a level 7 interrupt to the local CPU from the VMEbus. To enable this option, the PI/T port interrupt (vectored) must be initialized by the local software.

The PI/T must be initialized so that a transition (low to high or high to low) on the H1 interrupt trigger input will cause an interrupt request. This interrupt request is of level 7, that is non-maskable. It must be the interrupt handler software that resets the interrupt. The VMEbus counterparts can be informed of the execution via the Dual Ported RAM or interrupt action via the BIM.

4.3.5 Reading the RUN/LOCAL Switch

The RUN/LOCAL switch controls whether or not the ISCSI-1 board can be accessed from the VMEbus. It would be of interest for the local software to distinguish this situation. The status of the RUN/LOCAL switch can be read by the local CPU via PI/T port PB6.

4.3.6 The Timer Section of the PI/T

The timer of the PI/T device is only of interest in the mode of a periodical interrupter. The timer interrupt has to be programmed so that the PI/T will support vectored interrupt acknowledge cycles.

The associated interrupt request level is level 5 of the CPU. This interrupt level is not subject to user configuration.

If the watchdog timer is used, it makes sense for the board to generate timer interrupts in the range of once every 5 to 10 milliseconds. The interrupt handler software should include some proof of correct system operation and finally the watchdog timer has to be triggered if the correct operation is stated.

The PI/T is clocked at a frequency of 8.0MHz.

4.3.7 The SCSIbus Reset Control

The SCSIbus reset is controlled via the PI/T port B bit #4 and #5. Bit #4 is programmed as an output, bit #5 as an input. To generate a reset on the SCSIbus it is necessary to clear ("0") bit #4 for at least 25 us and then set ("1") again. Bit #5 indicates the state of the SCSI reset.

4.3.8 The Single or Double Floppy Density Selection

The input of the floppy density selection, which selects either single (FM) or double (MFM) density is directly connected to port B bit #7 (programmed as an output). Bit #7 cleared ("0") selects double density, bit #7 set ("1") selects single density.

4.4 The Direct Memory Access Controller 68450 (DMAC)

The SYS68K/ISCSI-1 board contains a 4-channel DMA Controller (68450) with a clock frequency of 10MHz. The 68450 offers a wide variety of programmable transfer options as described in the data sheet in Register 5, Chapter 5.

4.4.1 The DMAC Implementation

The DMAC is an alternative bus master to the CPU, performing read/write cycles on-board.

The base address of the DMAC is $C80000. Byte and word transfers are possible.

The DMAC can interrupt the CPU on IRQ level 2. If jumper B41 is inserted, the DMAC supports a vector which is the contents of one of the eight interrupt vector registers. If jumper B41 is removed, an autovector interrupt is generated. The autovector level is 2.

The DMAC, when master, can access all memory locations accessible to the CPU (except for the DMAC registers).

The DMAC can be requested by the SCSI controller or by the FDC. To provide high speed data transfer on the SCSIbus, the SCSI controller can also be accessed in the "Single Address Mode".

The data sheet of the 68450 DMAC is included in Register 5, Chapter 5.

4.4.2 Addressing of the DMAC

The DMAC can either be programmed in 8 or in 16-bit mode. The data bits D0-D15 can be used for loading the DMAC.

The register model of the DMAC is listed in Table 4-2.

Table 4-2: Register Model of the DMAC

Base Address : $C80000

Address HEX	Offset	Reset Value	Label	Description
				C H A N N E L 0
C80000	00	01	DMACSR	Channel Status Register
C80001	01	00	DMACER	Channel Error Register (Read)
C80004	04	00	DMADCR	Device Control Register
C80005	05	00	DMAOCR	Operation Control Register
C80006	06	00	DMASCR	Sequence Control Register
C80007	07	00	DMACCR	Channel Control Register
C8000A	0A	--	DMAMTC	Memory Transfer Counter
C8000B	0B	--		
C8000C	0C	--	DMAMAR	Memory Address Register
C8000D	0D	--		
C8000E	0E	--		
C8000F	0F	--		
C80014	14	--	DMADAR	Device Address Register
C80015	15	--		
C80016	16	--		
C80017	17	--		
C8001A	1A	--	DMABTC	Base Transfer Counter
C8001B	1B	--		
C8001C	1C	--	DMABAR	Base Address Register
C8001D	1D	--		
C8001E	1E	--		
C8001F	1F	--		
C80025	25	0F	DMANIVR	Normal Interrupt Vector
C80027	27	0F	DMAEIVR	Error Interrupt Vector
C80029	29	07	DMAMFC	Memory Function Codes
C8002D	2D	00	DMACPR	Channel Priority Register
C80031	31	07	DMADFC	Device Function Codes
C80039	39	07	DMABFC	Base Function Codes

cont'd...

Table 4-2 cont'd 68450 DMAC Register Layout

Base Address : $C80000

Address HEX	Offset	Reset Value	Label	Description
				C H A N N E L 1
C80040	40	01	DMACSR1	Channel Status Register
C80041	41	00	DMACER1	Channel Error Register (Read)
C80044	44	00	DMADCR1	Device Control Register
C80045	45	00	DMAOCR1	Operation Control Register
C80046	46	00	DMASCR1	Sequence Control Register
C80047	47	00	DMACCR1	Channel Control Register
C8004A	4A	--	DMAMTC1	Memory Transfer Counter
C8004B	4B	--		
C8004C	4C	--	DMAMAR1	Memory Address Register
C8004D	4D	--		
C8004E	4E	--		
C8004F	4F	--		
C80054	54	--	DMADAR1	Device Address Register
C80055	55	--		
C80056	56	--		
C80057	57	--		
C8005A	5A	--	DMABTC1	Base Transfer Counter
C8005B	5B	--		
C8005C	5C	--	DMABAR1	Base Address Register
C8005D	5D	--		
C8005E	5E	--		
C8005F	5F	--		
C80065	65	0F	DMANIVR1	Normal Interrupt Vector
C80067	67	0F	DMAEIVR1	Error Interrupt Vector
C80069	69	07	DMAMFC1	Memory Function Codes
C8006D	6D	00	DMACPR1	Channel Priority Register
C80071	71	07	DMADFC1	Device Function Codes
C80079	79	07	DMABFC1	Base Function Codes

cont'd...

Table 4-2 cont'd 68450 DMAC Register Layout

Base Address : $C80000

Address HEX	Offset	Reset Value	Label	Description
				C H A N N E L 2
C80080	80	01	DMACSR2	Channel Status Register
C80081	81	00	DMACER2	Channel Error Register (Read)
C80084	84	00	DMADCR2	Device Control Register
C80085	85	00	DMAOCR2	Operation Control Register
C80086	86	00	DMASCR2	Sequence Control Register
C80087	87	00	DMACCR2	Channel Control Register
C8008A	8A	--	DMAMTC2	Memory Transfer Counter
C8008B	8B	--		
C8008C	8C	--	DMAMAR2	Memory Address Register
C8008D	8D	--		
C8008E	8E	--		
C8008F	8F	--		
C80094	94	--	DMADAR2	Device Address Register
C80095	95	--		
C80096	96	--		
C80097	97	--		
C8009A	9A	--	DMABTC2	Base Transfer Counter
C8009B	9B	--		
C8009C	9C	--	DMABAR2	Base Address Register
C8009D	9D	--		
C8009E	9E	--		
C8009F	9F	--		
C800A5	A5	0F	DMANIVR2	Normal Interrupt Vector
C800A7	A7	0F	DMAEIVR2	Error Interrupt Vector
C800A9	A9	07	DMAMFC2	Memory Function Codes
C800AD	AD	00	DMACPR2	Channel Priority Register
C800B1	B1	07	DMADFC2	Device Function Codes
C800B9	B9	07	DMABFC2	Base Function Codes

cont'd...

Table 4-2 cont'd 68450 DMAC Register Layout

Base Address : $C80000

Address HEX	Offset	Reset Value	Label	Description
				C H A N N E L 3
C800C0	C0	01	DMACSR3	Channel Status Register
C800C1	C1	00	DMACER3	Channel Error Register (Read)
C800C4	C4	00	DMADCR3	Device Control Register
C800C5	C5	00	DMAOCR3	Operation Control Register
C800C6	C6	00	DMASCR3	Sequence Control Register
C800C7	C7	00	DMACCR3	Channel Control Register
C800CA	CA	--	DMAMTC3	Memory Transfer Counter
C800CB	CB	--		
C800CC	CC	--	DMAMAR3	Memory Address Register
C800CD	CD	--		
C800CE	CE	--		
C800CF	CF	--		
C800D4	D4	--	DMADAR3	Device Address Register
C800D5	D5	--		
C800D6	D6	--		
C800D7	D7	--		
C800DA	DA	--	DMABTC3	Base Transfer Counter
C800DB	DB	--		
C800DC	DC	--	DMABAR3	Base Address Register
C800DD	DD	--		
C800DE	DE	--		
C800DF	DF	--		
C800E5	E5	0F	DMANIVR3	Normal Interrupt Vector
C800E7	E7	0F	DMAEIVR3	Error Interrupt Vector
C800E9	E9	07	DMAMFC3	Memory Function Codes
C800ED	ED	00	DMACPR3	Channel Priority Register
C800F1	F1	07	DMADFC3	Device Function Codes
C800F9	F9	07	DMABFC3	Base Function Codes
				GLOBAL DEVICE CONTROL
C800FF	FF	00	DMAGCR	DMAC General Control Register

4.4.3 The DMAC Interrupt Scheme

The DMA Controller contains 8 interrupt vector registers which will be read during an interrupt acknowledge cycle of the CPU if jumper B41 is inserted; otherwise, an auto vector is forced.

4.4.4 The DMAC Summary

Default Base Address :	$C80000
Default Access Address :	$C80000 - $C800FF
Access Mode :	Byte and Word Read and Write
Usable Data Bits :	D0-D7, D8-D15
Interrupt Level :	IQ2
Interrupt Vector :	Auto vectors/non-autovector

4.5 The SCSIbus Controller NCR 5386S (SCSIBC)

- The NCR 5386S SCSIbus Controller communicates with the 68010 CPU and the 68450 DMAC as a peripheral device. The SCSIBC is controlled by reading and writing several internal registers which are addressed via the local address bus.

- Since the SCSIBC interrupts the MPU when it detects a SCSIbus condition that requires servicing, the MPU is free from polling or controlling any of the SCSIbus signals.

- The SCSIBC will be programmed and controlled via the local CPU.

- The SCSIBC Interrupt is connected to the CPU interrupt level 3. The generated interrupt is always an auto vector interrupt.

- For high speed data transfer the SCSIBC communicate directly with the Dual Ported RAM under the DMAC control. In this mode the maximum data transfer rate will be 1.5Mbyte(s).

- The ISCSI-1's own SCSI I.D. is software programmable and is stored in the "Control Register" at the local address $CC0009. For more information refer to section 4.7.

- The SCSIBC is connected to the SCSIbus via the SCSIbus driver/receiver chip NCR 8310.

- The data sheet of the SCSIBC is included in Register 5, Chapter 3.

Table 4-3: SCSIBC NCR 5386S Register Address Map

Base Address: $C40000					
Address Hex	Offset	Reset Value	Mode	Label	Description
$C40001	01	--	R/W	SDAT1	Data Register 1
$C40003	03	--	R/W	SCMDR	Command Register
$C40005	05	--	R/W	SCTRLR	Control Register
$C40007	07	--	R/W	SDID	Destination I.D.
$C40009	09	02	R	SASTAT	Auxiliary Status
$C4000B	0B	--	R	SIDR	I.D. Register
$C4000D	0D	01	R	SIDQ	Interrupt Register
$C4000F	0F	--	R	SSID	Source I.D.
$C40011	11	--	R/W	SDAT	Data Register 2
$C40013	13	98	R	SDSTAT	Diagnostic Status
$C40019	19	--	R/W	STCH	Transfer Counter High
$C4001B	1B	--	R/W	STCM	Transfer Counter Mid
$C4001D	1D	--	R/W	STCL	Transfer Counter Low
$C4001F	1F	--	R/W	STST	Reserved for Testability

4.5.1 SCSIbus Controller Summary

Base Address: $C40000

Access Address: $C40001 - $C4001F

Access Mode: Byte
Read and Write

Usable Data bits for programming: D0 - D7

Interrupt Level: IQ Level 3

Interrupt Vector: Auto Vector #3

4.6 The Floppy Disk Controller WD 1772 (FDC)

In order to provide easy connection to floppy drives, the ISCSI-1 board includes a Floppy Disk Controller (FDC)

This FDC, which is able to control up to four floppy drives (3", 3 1/2" or 5 1/4"). The FDC is fully supported by the firmware.

The FDC is programmed and controlled by the on-board CPU. Data transfer is performed via the 68450 DMAC.

The FDC interrupt output is connected to the CPU interrupt level 4. The generated interrupt is an auto vector interrupt.

To select single (FM) or double (MFM) density, bit #7 of the PI/T port B must be cleared or set. Bit #7 cleared ("0") selects double density, bit #7 set ("1") selects single density.

To allow the connection of all SHUGART-compatible floppy disk drives, the Drive Select 0 to 3 signals, the Side Select signal and the single/double density signal are software programmable. The current values of these signals will be written into the Control Register and the PI/T Port B. For more information refer to Chapter 4.7 and Chapter 4.3.8.

The datasheet of the FDC is included in Register 5, Chapter 6.

Table 4-4: The FDC WD 1772 Register Address Map

Base Address : $CC0000					
Address HEX	Offset	Reset Value	Mode	Label	Description
$CC0001	01	7C	R	FSTR	Status Register
$CC0001	01	--	W	FCR	Command Register
$CC0003	03	FF	R/W	FTR	Track Register
$CC0005	05	01	R/W	FSR	Sector Register
$CC0007	07	00	R/W	FDR	Data Register

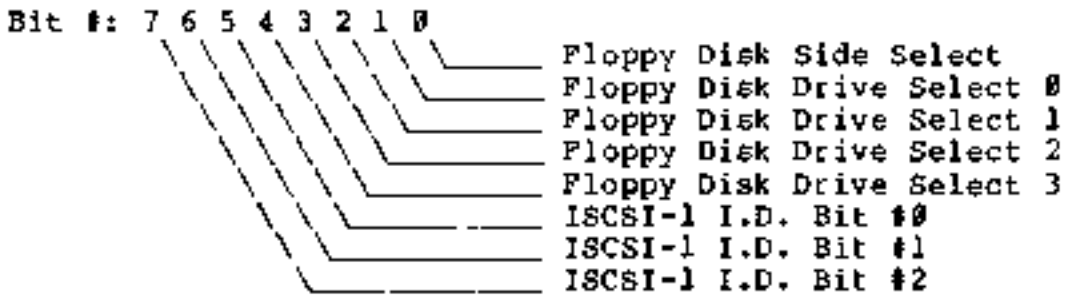
4.6.1 Floppy Disk Controller Summary

Base Address:	$CC0000
Access Address:	$CC0001 - $CC0007
Access Mode:	Byte Read and Write
Usable Data bits for programming:	D0 - D7
Interrupt Level:	IQ Level 4
Interrupt Vector:	Auto Vector #4

4.7 The Control Register

Some I/O and control signals on the board are software programmable, in order to provide easy handling. The on-board Control Register, which handles these signals is designed as a "Read Back Register".

The Control Register bit assignment is shown below:



The control register address is #CC0009.

Access modes available are: Byte
Read and Write

4.8 The SYS68K/ISCSI-1BPS

There is an optional back panel called SYS68K/ISCSI-1BPS available from FORCE Computers.

This board can be plugged into the P2 connector of the ISCSI-1 board. It contains an SCSIbus connector (X1) and a floppy disk interface connector (X2). The pinout of these connectors is shown in Table 4-6 and Table 4-7.

Figure 4-1 shows how the ISCSI-1BPS can be connected to the ISCSI-1.

Table 4-5: The ISCSI-1 P2 Pin Assignment

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	DB 0	VCC	N.C.
2	DB 1	GND	N.C.
3	DB 2	N.C.	Drive Select 0
4	DB 3	N.C.	Index
5	DB 4	N.C.	Drive Select 1
6	DB 5	N.C.	Drive Select 2
7	DB 6	N.C.	Drive Select 3
8	DB 7	N.C.	Motor On
9	DB P	N.C.	Direction In
10	GND	N.C.	Step
11	GND	N.C.	Write Data
12	GND	GND	Write Gate
13	TERMPWR	VCC	Track 000
14	GND	N.C.	Write Protect
15	GND	N.C.	Read Data
16	ATN	N.C.	Side Select
17	GND	N.C.	N.C.
18	BSY	N.C.	N.C.
19	ACK	N.C.	GND
20	RST	N.C.	GND
21	MSG	N.C.	N.C.
22	SEL	GND	GND
23	C/D	N.C.	GND
24	REQ	N.C.	N.C.
25	I/O	N.C.	N.C.
26	N.C.	N.C.	N.C.
27	GND	N.C.	RESERVED
28	N.C.	N.C.	RESERVED
29	RESERVED	N.C.	RESERVED
30	RESERVED	N.C.	RESERVED
31	RESERVED	GND	RESERVED
32	RESERVED	VCC	RESERVED

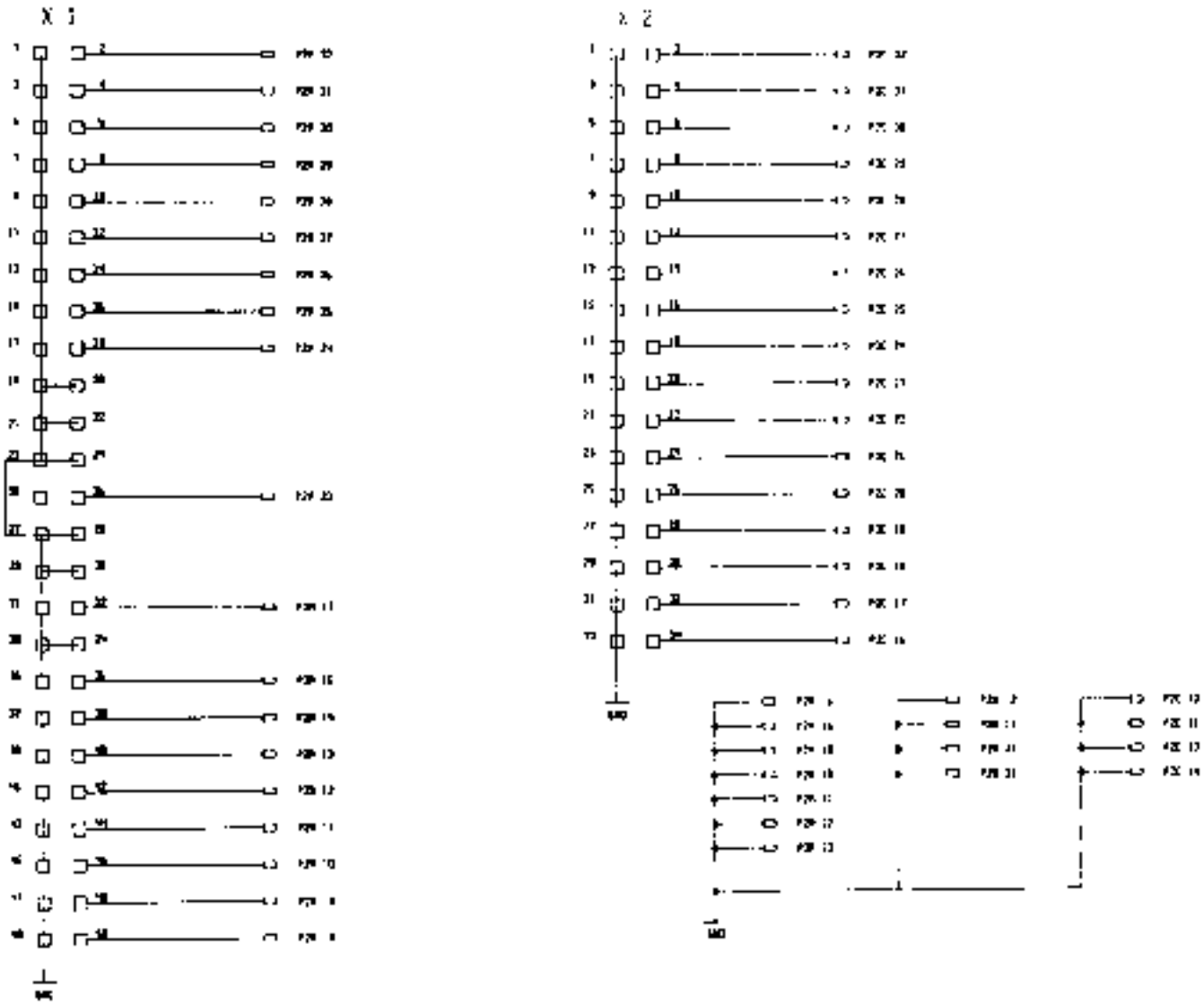
Table 4-6: The ISCSI-1BP X1 Pin Assignment

Pin No.	Signal Mnemonic	Pin Number	Signal Mnemonic
2	DB 0	1	GND
4	DB 1	3	GND
6	DB 2	5	GND
8	DB 3	7	GND
10	DB 4	9	GND
12	DB 5	11	GND
14	DB 6	13	GND
16	DB 7	15	GND
18	DB P	17	GND
20	GND	19	GND
22	GND	21	GND
24	GND	23	GND
26	TERMPWR	25	N.C.
28	GND	27	GND
30	GND	29	GND
32	ATN	31	GND
34	GND	33	GND
36	BSY	35	GND
38	ACK	37	GND
40	RST	39	GND
42	MSG	41	GND
44	SEL	43	GND
46	C/D	45	GND
48	REQ	47	GND
50	I/O	49	GND

Table 4-7: The ISCSI-1BP X2 Pin Assignment

Pin No.	Signal Mnemonic	Pin Number	Signal Mnemonic
2	N.C.	1	GND
4	N.C.	3	GND
6	Drive Select 0	5	GND
8	Index	7	GND
10	Drive Select 1	9	GND
12	Drive Select 2	11	GND
14	Drive Select 3	13	GND
16	Motor On	15	GND
18	Direction In	17	GND
20	Step	19	GND
22	Write Data	21	GND
24	Write Gate	23	GND
26	Track 000	25	GND
28	Write Protect	27	GND
30	Read Data	29	GND
32	Side Select	31	GND
34	N.C.	33	GND

Figure 4-2: Schematic of the SYS68K/ISCSI-1BP



4.9 The Dual Ported RAM as Local Memory

There is 128Kbyte of static memory installed on the SYS68K/ISCSI-1 board. This RAM is built up with a dual port access structure so that it can be read and written by the local CPU and by the VMEbus master.

There is an access control mechanism installed that decides which side performs a memory cycle (Read or Write). This access control mechanism is controlled by the clock of the local CPU, so that the local CPU is always granted an access cycle in a way that no wait cycles are needed. No wait cycles of the local CPU are granted for the case of the maximum access rate from the VMEbus side. This situation allows local program execution times to be calculated.

4.9.1 Local CPU Access to the DPR

The local CPU access range of the DPR is

$000000 - $01FFFF.

This range can be accessed in all addressing modes of the local CPU.

During the boot-up procedure, only the local EPROMs can be selected and the processor reads start-up information from the EPROM even though during normal operation, the corresponding addresses select the DPR area.

The boot-up procedure's special decoding ends with the first Write cycle of the local CPU; from then on, the DPR decoding range is valid.

The address range

$000000 - $001FFF

cannot be accessed from the VMEbus.

The address range

$002000 - $01FFFF

is Dual Ported Memory.

4.9.2 Local CPU Read-Modify-Write Cycles

The local CPU is allowed to perform Read-Modify-Write operations to the Dual Ported RAM.

This operation is nondivisible for the local software, so it can be used to coordinate for example, between several local interrupt driven tasks.

The Dual Ported RAM is not blocked for external access between the Read and Write cycles of the local processor's Read-Modify-Write operation, nor is the local processor's access blocked between the VMEbus RMW access cycles. So the DPR cannot be used with Read-Modify-Write cycles to synchronize between local and VMEbus processes.

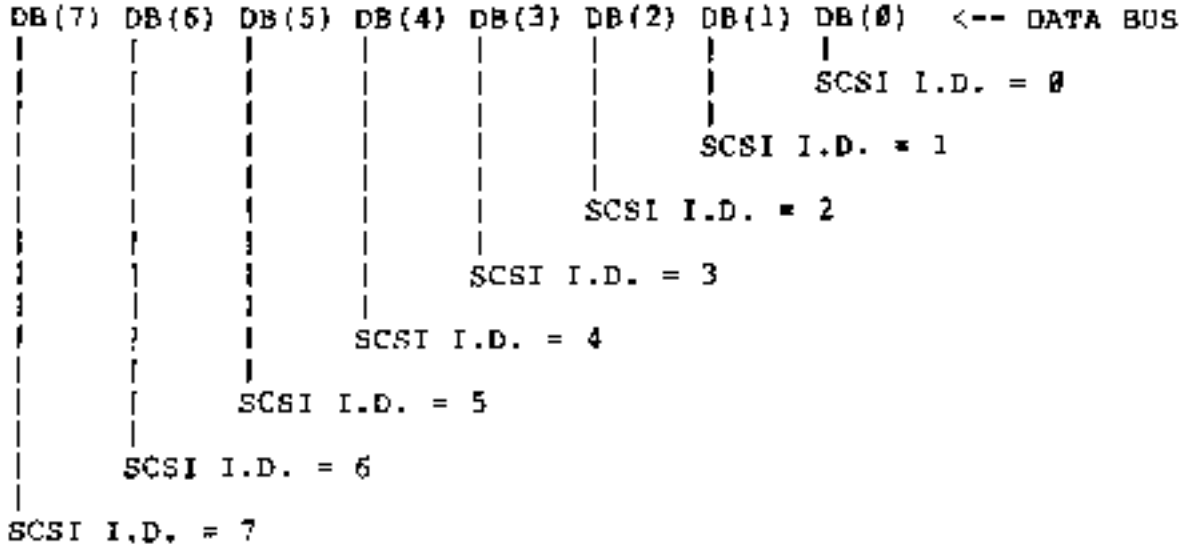
5.0 The SCSIbus Description

5.1 The SCSIbus Configuration

Communication on the SCSIbus is allowed between only two SCSI devices at any given time. There is a maximum of eight SCSI devices. Each SCSI device has an SCSI I.D. bit assigned as shown in Figure 5-1. When two SCSI devices communicate on the SCSIbus, one acts as an initiator, and the target performs the operation. An SCSI device usually has a fixed role as an initiator or target, but some devices may be able to assume either role.

An initiator may address up to eight peripheral devices that are connected to a target. An option allows the addressing of up to 2.048 peripheral devices per target using extended messages. Three sample system configurations are shown in Figure 5-2.

Figure 5-1: SCSI I.D. Bits

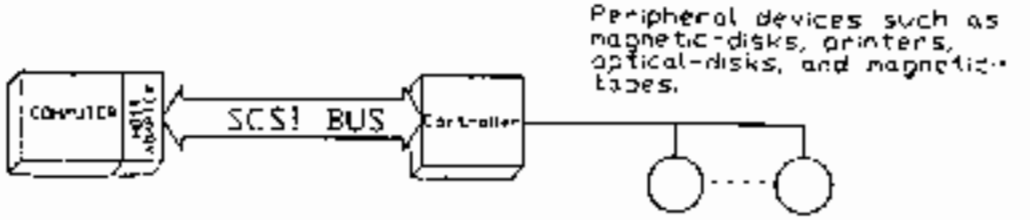


Up to eight SCSI devices can be supported on the SCSIbus. They can be any combination of initiators and targets.

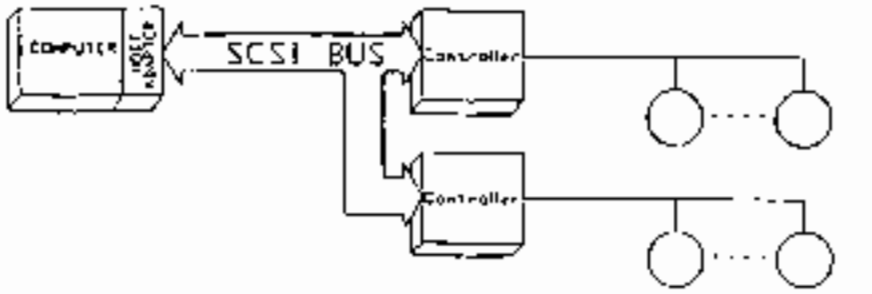
Certain SCSIbus functions are assigned to the initiator and certain SCSIbus functions are assigned to the target. The initiator may arbitrate for the SCSIbus and select a particular target. The target may request the transfer of COMMAND, DATA, STATUS, or other information on the data bus, and in some cases it may arbitrate for the SCSIbus and reselect an initiator for the purpose of continuing an operation.

Information transfers on the data bus are asynchronous and follow a defined REQ/ACK handshake protocol. One byte of information may be transferred with each handshake. A option is defined for synchronous data transfer.

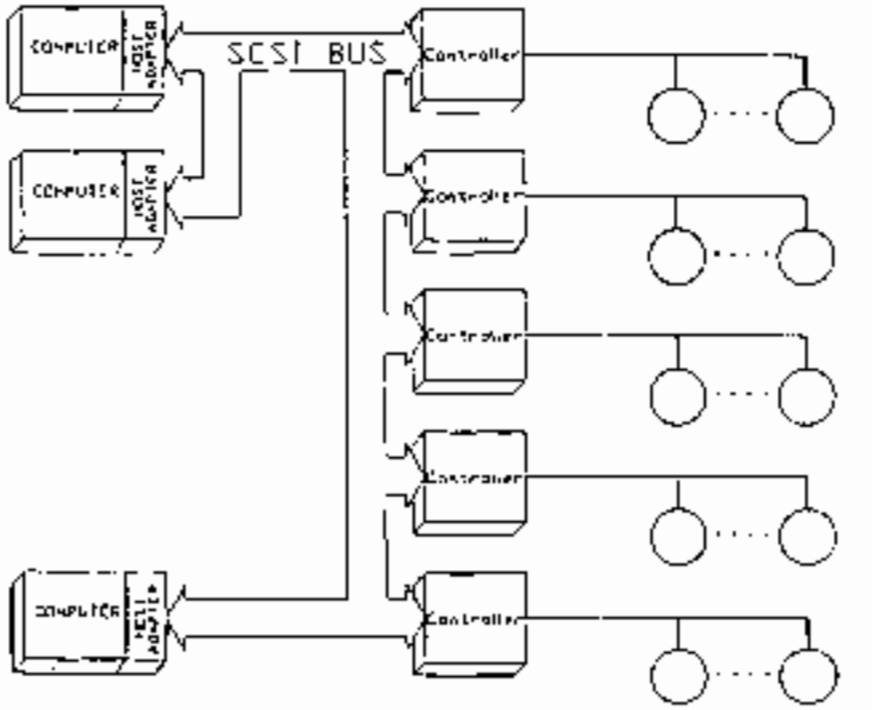
Figure 5-2: Sample SCSI Configurations



SINGLE INITIATOR, SINGLE TARGET



SINGLE INITIATOR, MULTIPLE TARGET



MULTIPLE INITIATOR, MULTIPLE TARGET

5.2 The SCSIbus Signal Description

There are a total of eighteen signals. Nine are used for control, and nine are used for data. (Data signals include the parity signal option). These signals are described as follows:

BSY (BUSY)	An "OR-tied" signal that indicates that the bus is being used.
SEL (SELECT)	A signal used by an initiator to select a target or by a target to reselect an initiator.
C/D (CONTROL/DATA)	A signal driven by a target that indicates whether CONTROL or DATA information is on the data bus. True indicates CONTROL.
I/O (INPUT/OUTPUT)	A signal driven by a target that control the direction of data movement on the data bus with respect to an initiator. True indicates input to the initiator. This signal is also used to distinguish between SELECTION and RESELECTION phases.
MSG (MESSAGE)	A signal driven by a target during the MESSAGE phase.
REQ (REQUEST)	A signal driven by a target to indicate a request for a REQ/ACK data transfer handshake.
ACK (ACKNOWLEDGE)	A signals driven by an initiator to indicate an acknowledgement for a REQ/ACK data transfer handshake.
ATN (ATTENTION)	A signal driven by an initiator to indicate the ATTENTION condition.
RST (RESET)	An "OR-tied" signal that indicates the RESET condition.
DB(7-0,P) (DATA BUS)	Eight data-bit signals, plus a parity-bit signal which together form a data bus. DB(7) is the most significant bit, and has the highest priority during the ARBITRATION phase. Bit number, significance, and priority decrease downward to DB(0). A data bit is defined as one when the signal value is true, and is defined as zero when the signal value is false.

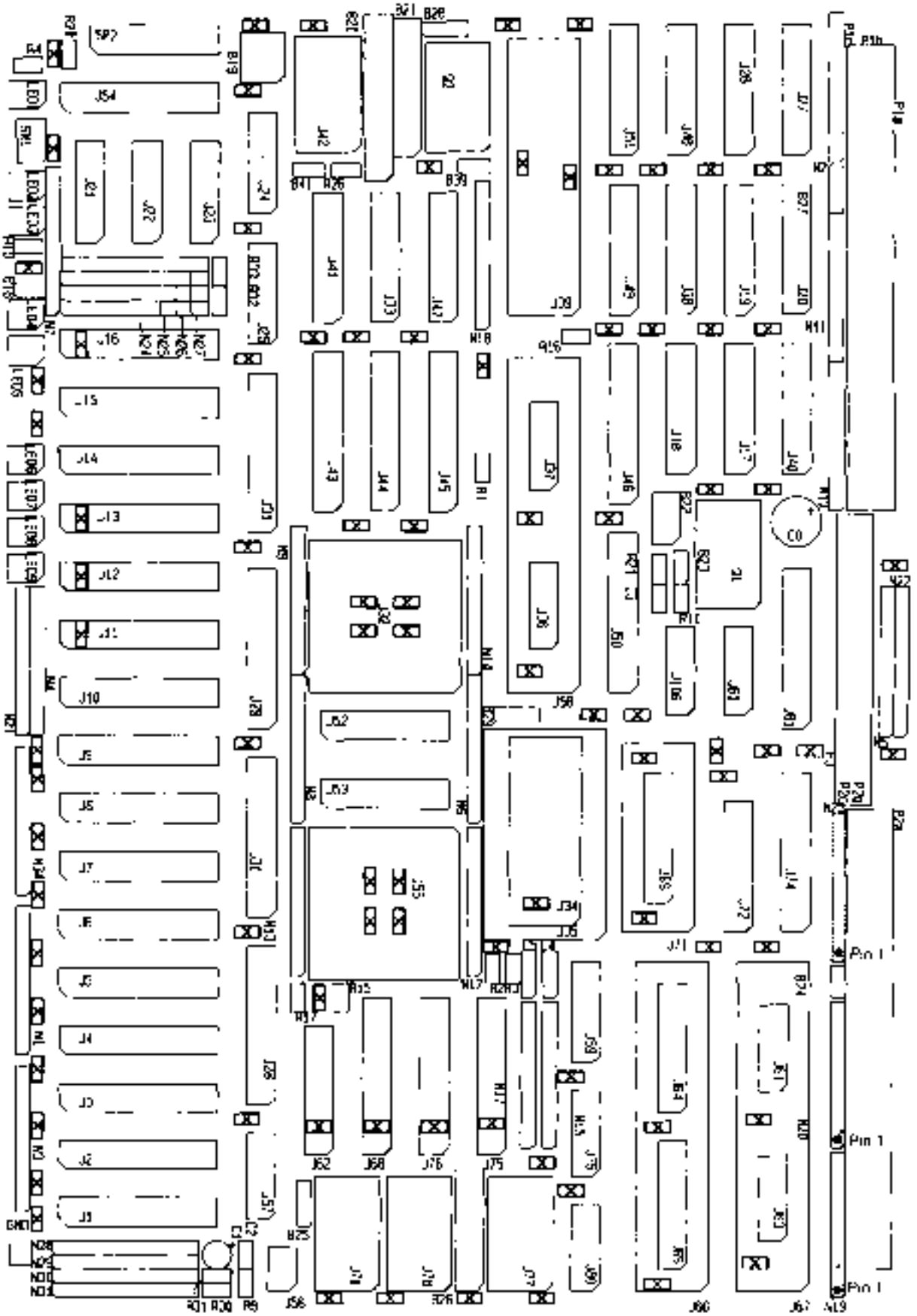
Data parity DP(P) is odd. The use of parity is a system option (i.e. a system is configured so that all SCSI devices on a bus generate parity and have parity detection enabled, or all SCSI devices have parity detection disabled or not implemented). Parity is not valid during the ARBITRATION phase.

5.3 The SCSIbus Signal Termination

Each SCSIbus signal is terminated at the physical start and the physical end of the SCSIbus. Therefore, the terminators must be removable, in order to join one SCSI device to two others.

Figure 5-3 shows the location of the terminators on the SYS68K/ISCSI-1 board. These terminators are plugged into sockets and can be removed if necessary.

Figure 5-3: Location of the SCSIbus Terminators



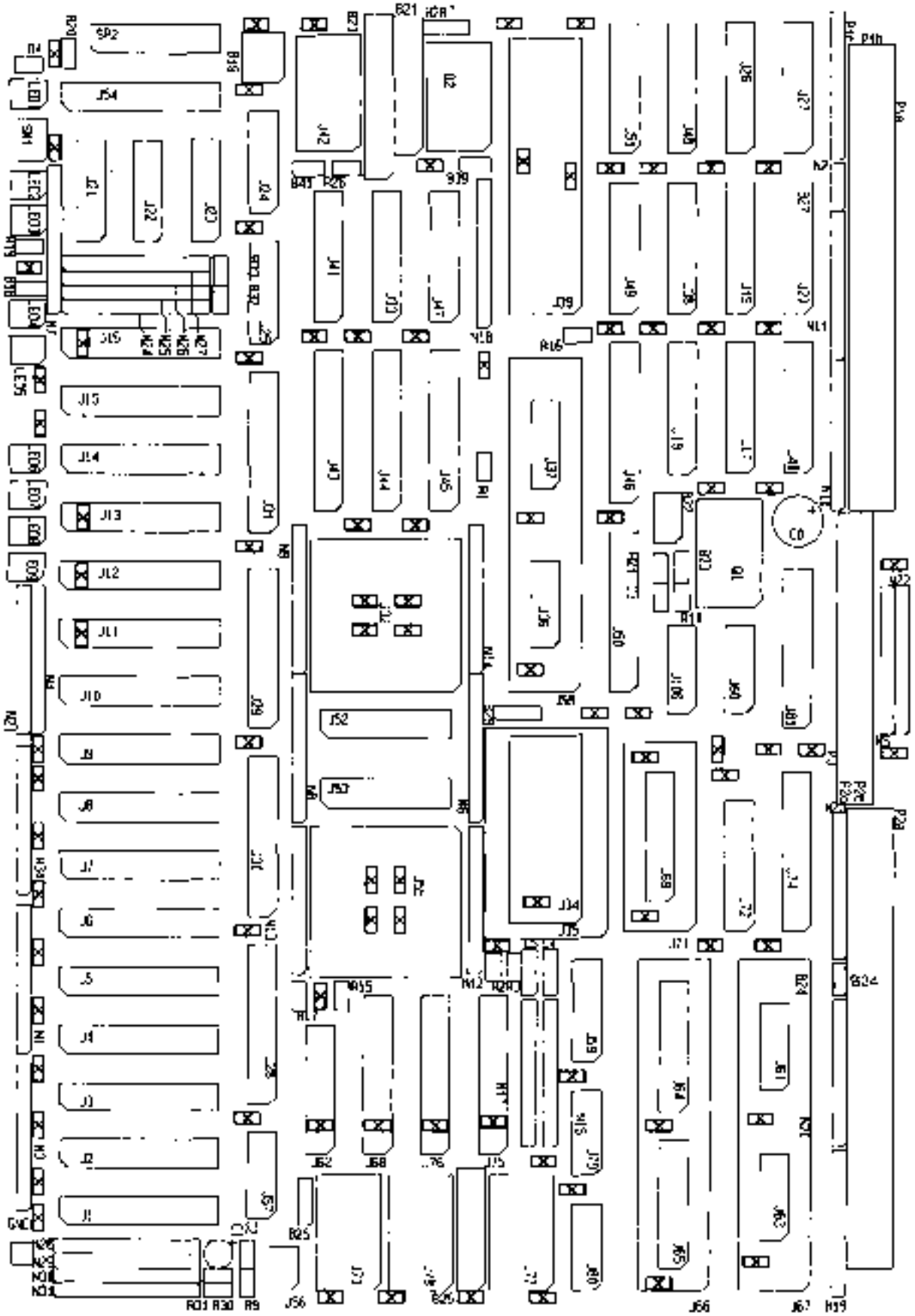
5.4 The SCSIbus Terminator Power

The power for the terminator of any SCSI device can be provided from the SYS68K/ISCSI-1 board, providing the SCSI device supports this optional case.

If jumper B24 is inserted, termination power is delivered from the ISCSI-1 board, and if jumper B24 is not inserted, termination power must be delivered from the SCSI device itself.

The location of the jumper B24 is shown in Figure 5-4.

Figure 5-4: Location of the Jumperfield B24



This page was intentionally left blank

6.0 The VMEbus Interface Hardware

The SYS68K/ISCSI-1 performs the functions of a VMEbus slave and interrupter.

The VMEbus Address Modifier decoding as well as the address range decoding are user configurable via jumpers.

The VMEbus interface is installed to decode A1-A23 of the standard address range. A 16-bit wide data bus is supported.

As an interrupter, the SYS68K/ISCSI-1 can be programmed to drive four interrupt request (IRQ) functions. The IRQ level is individually software programmable to any of IRQ1 to IRQ7.

A front panel switch, labelled R/L, allows the VMEbus decoding of the board to be enabled/disabled. A reset and an interrupt trigger call is supported, as is board status readout.

6.1 VMEbus Access to the Board

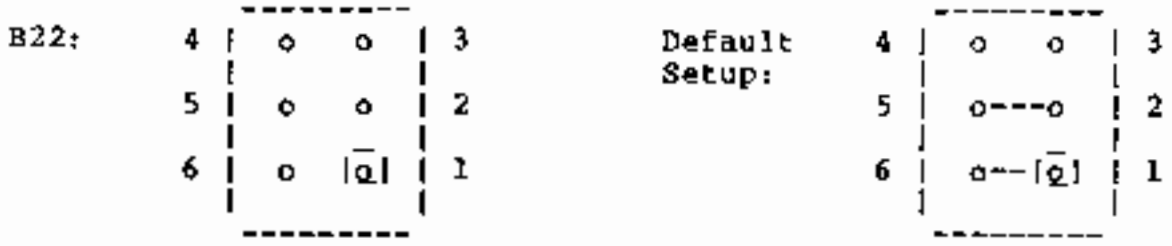
The SYS68K/ISCSI-1 will be accessed from the VMEbus if the current master drives a data transfer cycle under the following conditions:

1) Valid Address Modifier decoding
2) Valid addressing to the board's address range
3) RUN/LOCAL switch (R/L) set to RUN

During each access cycle, the yellow ACCESS LED on the front panel is lit to give optical information.

6.1.1 The Address Modifier Code Selection

The Address Modifier code is defined in the VMEbus specification P1014 of IEEE. The Address Modifier decoding of the board is configured at B22 via jumpers.



More than one condition can be enabled simultaneously. In the default condition with the jumpers 1-6 and 2-5 installed, both supervisory and non-privileged accesses in the standard addressing mode will be recognized as valid (P1014 compatible).

The following selections are installed:

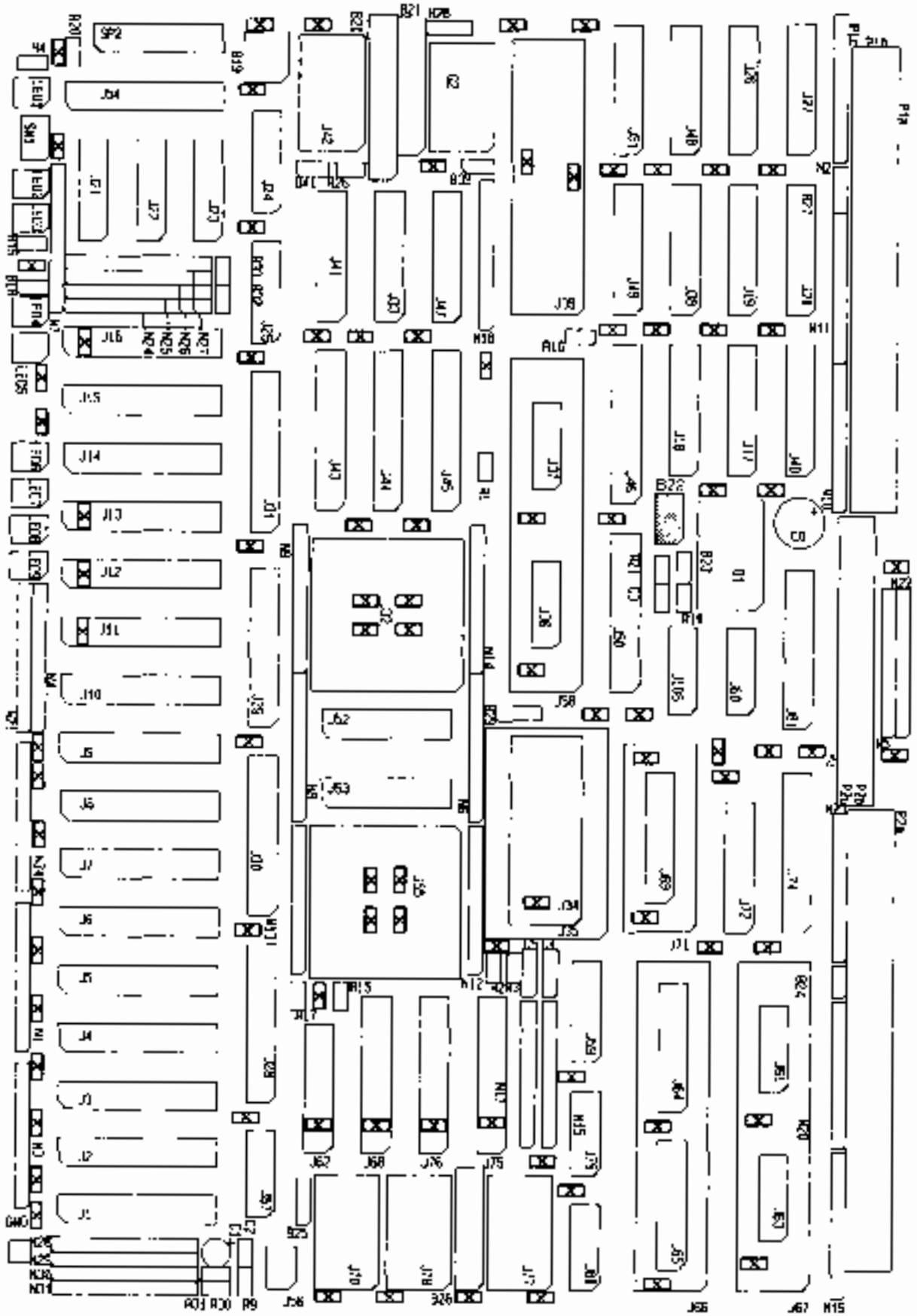
No jumpers : Ignore AM codes (Not P1014 compatible)

1 - 6 : Standard non-privileged program and data access (codes $3A and $39) are enabled

2 - 5 : Standard supervisory program and data access (codes $3E and $3D) are enabled

3 - 4 : Code $3F (not P1014 compatible)

Figure 6-1: Location Diagram of Jumperfield B22

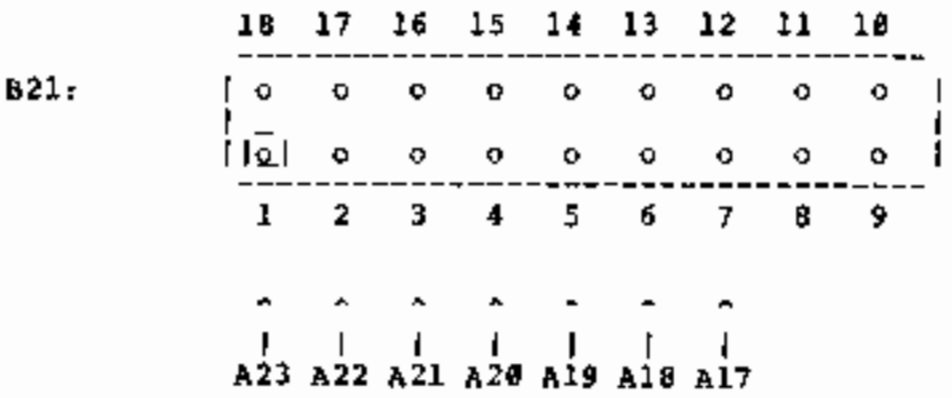


6.1.2 The Board Base Address Selection

The SYS68K/ISCSI-1 board occupies an address range of 128Kbytes of the VMEbus standard address range.

The base address of the board address range can be selected in 128Kbyte steps. The VMEbus address signals A23-A17 are compared to the base address jumper setting at B21. In case of coincidence, the board select condition is met.

The jumperfield B21 is as shown here:



The following jumper connections are allowed:

B21 Connections	Address Signal	Jumper Meaning
1 - 18	A23	in = 0, out = 1
2 - 17	A22	in = 0, out = 1
3 - 16	A21	in = 0, out = 1
4 - 15	A20	in = 0, out = 1
5 - 14	A19	in = 0, out = 1
6 - 13	A18	in = 0, out = 1
7 - 12	A17	in = 0, out = 1

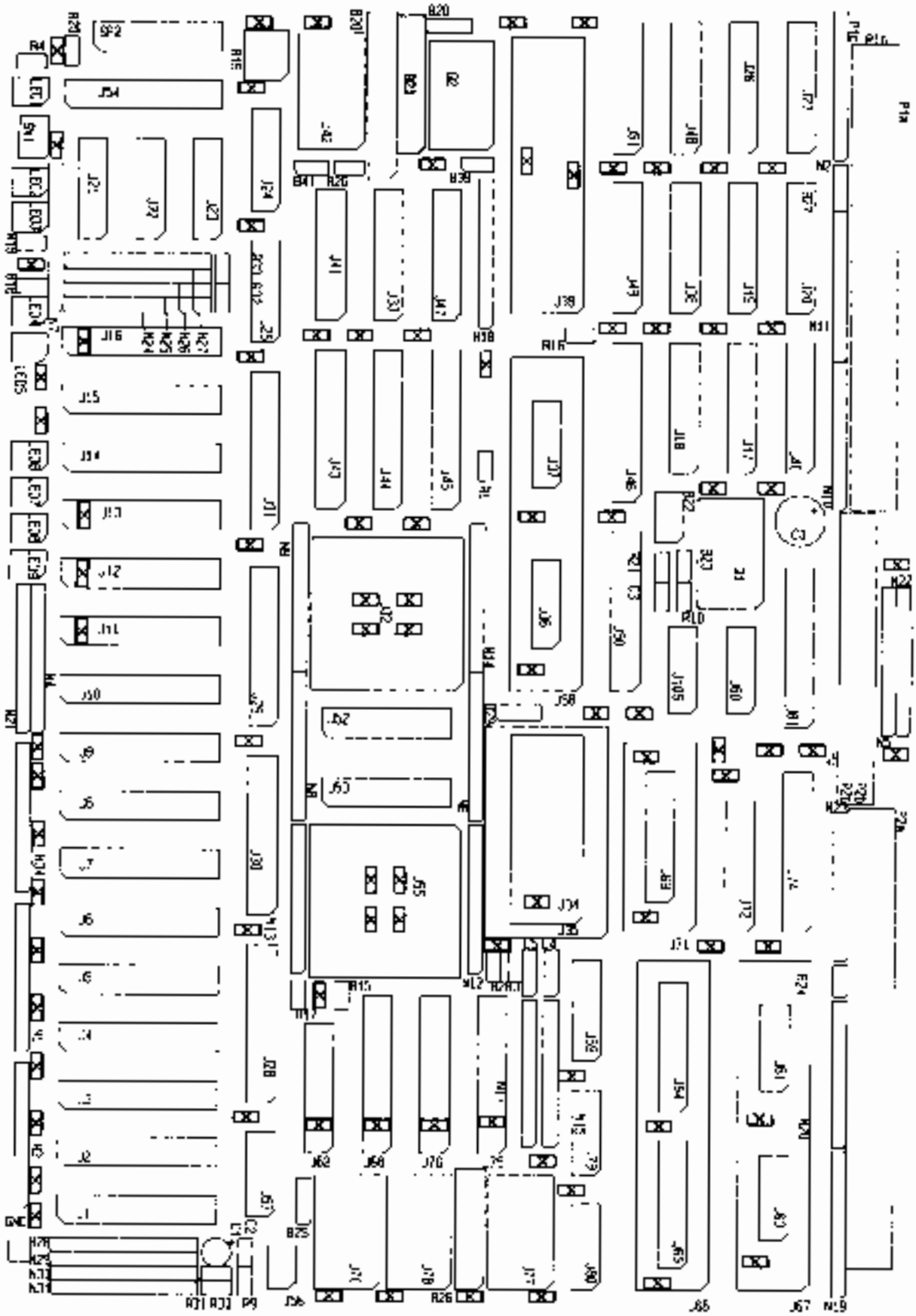
The default base address of the board is $A00000.

B21 default connections:

2 - 17
4 - 15
5 - 14
6 - 13
7 - 12

$A00000
A23 = 1
A22 = 0
A21 = 1
A20 = 0
A19 = 0
A18 = 0
A17 = 0

Figure 6-2: Location Diagram of Jumperfield B21



6.1.3 The RUN/LOCAL Switch

The R/L switch on the front panel allows VMEbus accesses to the ISCSI-1 board to be enabled/disabled.

If the R/L switch is turned on, then the red LOCAL LED is lit and the board cannot be accessed.

If the R/L switch is turned off, then the green RUN LED is lit and the board can be accessed from the VMEbus. An access occurs if a data cycle is driven with the valid AM code and the selected addressing range is hit.

6.1.4 The Access LED

There is a yellow LED, labelled SEL, mounted on the front panel. If a VMEbus access to the board occurs, then this yellow LED is lit. It is also lit during interrupt acknowledge cycles to the board.

6.2 The Address Map of the VMEbus Address Range

There are five addressable objects that can be accessed by the VMEbus master:

1) The BIM

2) The status byte

3) The interrupt trigger

4) The reset call

5) The Dual Ported RAM

The VMEbus address map is as follows:

Offset to Board Base Address	Default Address	Read Access	Write Access	Access Mode	Register Memory
$000000	$A00000	x	--	Word only	Bit 8 = Local RESET state Bit 9 = Local CPU HALT state Bit 10 = Watchdog Timer State
$000001	$A00001	x	x	Byte	BIM Control 0
$000003	$A00003	x	x	Byte	BIM Control 1
$000005	$A00005	x	x	Byte	BIM Control 2
$000007	$A00007	x	x	Byte	BIM Control 3
$000009	$A00009	x	x	Byte	BIM Vector 0
$00000B	$A0000B	x	x	Byte	BIM Vector 1
$00000D	$A0000D	x	x	Byte	BIM Vector 2
$00000F	$A0000F	x	x	Byte	BIM Vector 3
$001001	$A01001	x	--	Byte	Interrupt trigger call
$001801	$A01801	x	--	Byte	Reset trigger call
$002000 to $01FFFF	$A02000 to $A1FFFF	x	x	Byte and Word	Dual Ported RAM

6.3 The Dual Ported RAM (DPR) as VMEbus Memory

The Dual Ported RAM installed on the SYS68K/ISCSI-1 board can be accessed by the local CPU and by a VMEbus master.

The VMEbus access addressing range is by default $A02000-$A1FFFF which is an offset range of $002000-$01FFFF relative to the board base address.

The total size of the RAM is 128Kbyte but the low end 8Kbyte are only local accessible. The actual DPR size is 120Kbyte. The DPR is byte and word mode, read and write accessible for VMEbus masters. Read-Modify-Write accesses are also allowed, but this method can not be used to synchronize between local and VMEbus procedures. The Read-Modify-Write cycles of the local CPU can occur interleaved with the VMEbus access.

6.4 The Reset Trigger Call

A read access from the VMEbus to the byte location with the default address $A01801 that is $001801 offset to the board base address, is reset trigger for the local CPU hardware.

Upon this reset trigger call, the local CPU stops and resets, booting up with the "cold start" in the same way as after the system reset. The call causes a reset pulse on the local side which is long enough for the CPU to reset.

The status of the local RESET signal can be read out via the status register (see Chapter 6.7).

The reset trigger call does not reset the BIM device nor does it reset the interrupt request flip-flops (Chapter 4.3.3).

In a multi-master environment, it is possible that one master triggers reset to the SYS68K/ISCSI-1 and then another master acknowledges an interrupt. Interrupt service routines should begin and end with a check on the SYS68K/ISCSI-1's status register.

6.5 The Interrupt Trigger Call

A non-maskable level 7 interrupt to the local CPU can be triggered by a read access via an addressable location on the SYS68K/ISCSI-1 board. The byte read access has to be addressed by default to $A01001 or to the offset $001001 relative to the board base address. This access does not operate without local software initialization. The trigger signal is routed to the H1 interrupt trigger input of the PI/T device. If software initializes the PI/T to activate H1, to perform IRQ to the CPU and the proper interrupt handling routine is installed, then this interrupt call will be perceived and responded to. The PI/T data sheet includes more details on programming. See also Chapter 4.3.4.

6.6 The Bus Interrupter Module

There is a 68153 Bus Interrupter Module (BIM) installed on the SYS68K/ISCSI-1 board. The device has eight addressable register bytes. The BIM is only accessible from the VMEbus at the address locations which are shown in Chapter 6.2.

The data sheet of the BIM is included in this manual.

If the SYS68K/ISCSI-1 board has to perform interrupt requests to the VMEbus then the corresponding VMEbus master has to program the BIM via register write operations to enable and to drive the IRQ level required. The interrupt vector has to be programmed as well. The four channels of the BIM can be operated by four masters independent of each other.

The description of how the local CPU can drive interrupt requests to the BIM channels is shown in Chapter 4.3.3

There is absolutely no hardware interdependence between the local CPU's interrupt structure and the VMEbus interrupt structure.

All interrupt interfacing has to be built up and needs individual control by local software and by the VMEbus master which needs to be interrupted by the ISCSI-1.

Any sequence that leads the IRQ to the VMEbus has to be executed so that first of all the BIM is programmed by the VMEbus master to the proper vector and interrupt request level. Then the ISCSI-1 software can be triggered to start operations that include a trigger for an interrupt request channel.

Warning: The reset trigger call does not reset interrupt requests pending, only the system reset of the VMEbus can do this. Interrupt request are cleared by the execution of the corresponding interrupt acknowledge cycle.

6.7 The Status Register

There is a status register included in the ISCSI-1 board. The status register is a read only function. If the board base address is accessed in the word mode of addressing, then the data bits 8, 9 and 10 give status information to the ISCSI-1 board.

It is legal to read each of the BIM registers as words together with the status information in bits 8, 9 and 10. The word write access is functionally identical to the corresponding odd byte write to the BIM.

The meaning of the bits on the status register:

Bit 8: If the local CPU's RESET signal is low, then the bit is 0, if not the bit is 1.

This status read is especially important after a reset trigger call, it must be monitored to decide if the board can be operated.

Bit 9: The local CPU's HALT state is monitored via this bit. A 0 means that the CPU's HALT signal is low. HALT is low during the RESET of the CPU, or after a double bus fault condition.

During normal CPU operation, HALT is high and the bit is read as 1. The only way of restarting the board after the CPU is halted is by RESET. This can be a global VME system RESET or a reset trigger call to the ISCSI-1 board.

Bit 10: The state of the watchdog timer on the board can be monitored by this status bit. If the watchdog timer is triggered regularly (e.g. every 10 milliseconds), then this bit is always 1.

If the trigger is not operated (e.g. because of system failure), then the watchdog output turns to low and this bit reads 0.

If the CPU is in the HALT state, then the watchdog timer trigger stops as well. If the watchdog time has elapsed, reporting a malfunction and the local CPU does not go into HALT state, then it is possible to return to the normal operation without resetting the board.

APPENDIX TO THE HARDWARE USER'S MANUAL

This page was intentionally left blank

A P P E N D I C E S

A. Specification of the SYS68K/ISCSI-1

B. Memory Map of the SYS68K/ISCSI-1

C. Address Assignments of the Local Devices

D. Component Part List of the SYS68K/ISCSI-1

E. Description of the Jumperfields on the SYS68K/ISCSI-1

F. Circuit Schematics of the SYS68K/ISCSI-1

G. Connector PIN Assignments of the SYS68K/ISCSI-1

H. Component Part List of the SYS68K/ISCSI-1

I. Circuit Schematics of the SYS68K/ISCSI-1BPS

J. Connector PIN Assignment of the SYS68K/ISCSI-1BPS

K. Glossary of VMEbus Terms (P1014)

L. Literature References

M. Product Error Report

This page was intentionally left blank

Appendix A

Specification of the SYS68K/ISCSI-1

Microprocessor	68010, 10MHz
Control	68230 PI/T for local control and timer function
Interrupter	68153 BIM
RAM	128Kbyte Dual Ported RAM, No Wait States during local access
EPROM	128Kbyte (max) 32Kbyte (default) (JEDEC compatible, 150ns) Only local access: zero wait state
SCSIbus Controller	NCR 5386S
Floppy Interface	WD 1772
VMEbus/IEEE P1014	Fully VMEbus compatible slave and interrupter interface
Power requirements	+ 5V/5.6A (max)
Operating temperature	0 to 50 degrees C
Storage temperature	-50 to 85 degrees C
Relative humidity	0 to 95% (non-condensing)
Board dimensions	Double eurocard 234 x 160mm (9.2" x 6.3")

This page was intentionally left blank

Appendix B

Memory Map of the SYS68K/ISCSI-1

Local CPU Memory Map

$000 000 to $001 FFF	8Kbyte SRAM (Local only)
$002 000 to $01F FFF	120Kbyte SRAM (Dual Ported)
$C00 000 to $CFF FFF	Local Devices
$E00 000 to $E1F FFF	EPROM

VMEbus Memory Map

$A00 000 to $A01 FFF	Control and Status (BIM)
$A02 000 to $A1F FFF	120Kbyte SRAM (Dual Ported)

This page was intentionally left blank

Appendix C

Address Assignment of the On-Board Devices

Local CPU Devices

Base Address:

$C40 000	SCSIbus Controller
$C80 000	Direct Memory Access Controller
$CC0 000	Floppy Disk Controller
$CC0 009	Control Register
$D00 000	PI/T

VMEbus Devices

Base Address:

$A00 000	BIM and Status Register
$A01 001	Interrupt Trigger
$A01 801	Reset Register

This page was intentionally left blank

APPENDIX D
COMPONENT PART LIST

FOR INTERNAL USE ONLY

This page was intentionally left blank

Appendix B

Description of the Jumperfields on the SYS68K/ISCSI-1

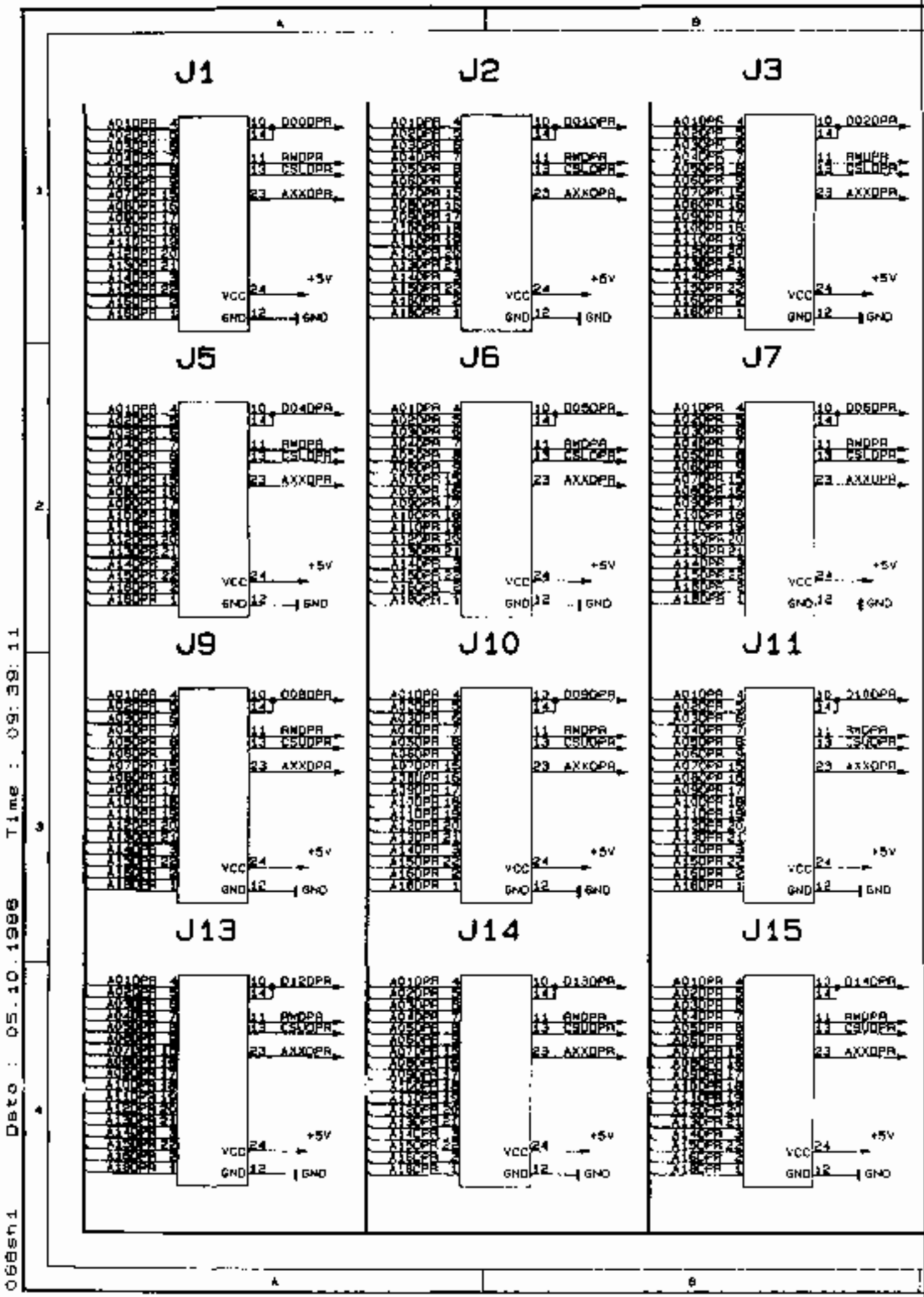
Description	Jumper field	Default Connections	Schematics Sheet	See Page
Reserved	B18	-	1	---
EPROM Size Select	B19	2 - 3 5 - 6 7 - 8	4	4-4
Board Base Address Selection	B21	2 - 17 4 - 15 5 - 14 6 - 13 7 - 12	7	6-4
Address Modifier Code Selection	B22	1 - 6 2 - 5	7	6-2
Test Jumper	B23	1 - 2	8	---
Terminator Power	B24	-	10	5-6
VME Reset	B27	1 - 2	7	---
Test Jumper	B28	1 - 2	12	---
Test Jumper	B29	1 - 2 or 2 - 3	11	---
Test Jumper	B39	1 - 2	5	---
Interrupt Configuration	B41	-	6	4-3

This page was intentionally left blank

A p p e n d i x F

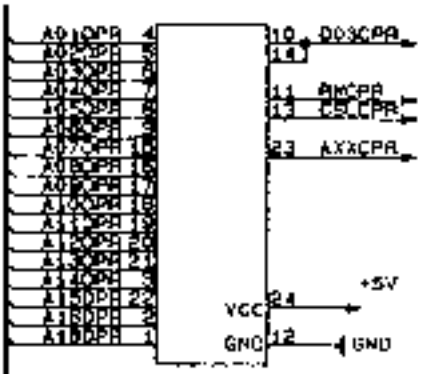
Circuit Schematics of the SYS68K/ISCSI-1

This page was intentionally left blank

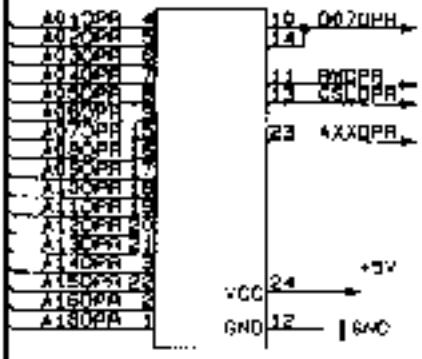

J1
J2
J3
J5
J6
J7
J9
J10
J11
J13
J14
J15
VCC
GND
+5V
Date : 05.10.1988 Time : 09:39:11
068sh1

C D

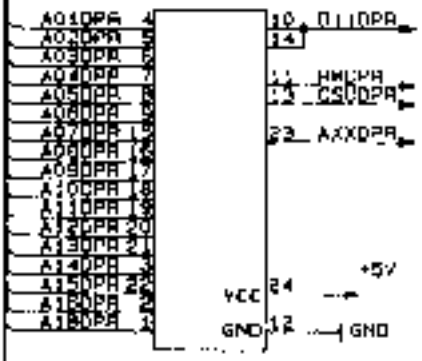
J4



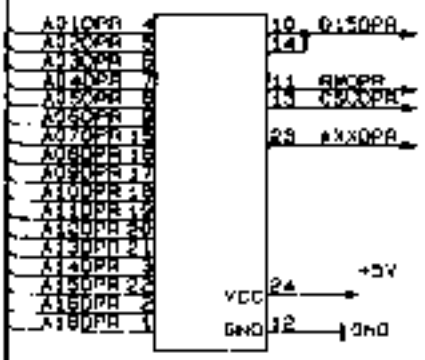
J8



J12

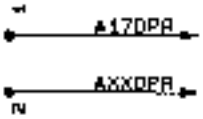


J16

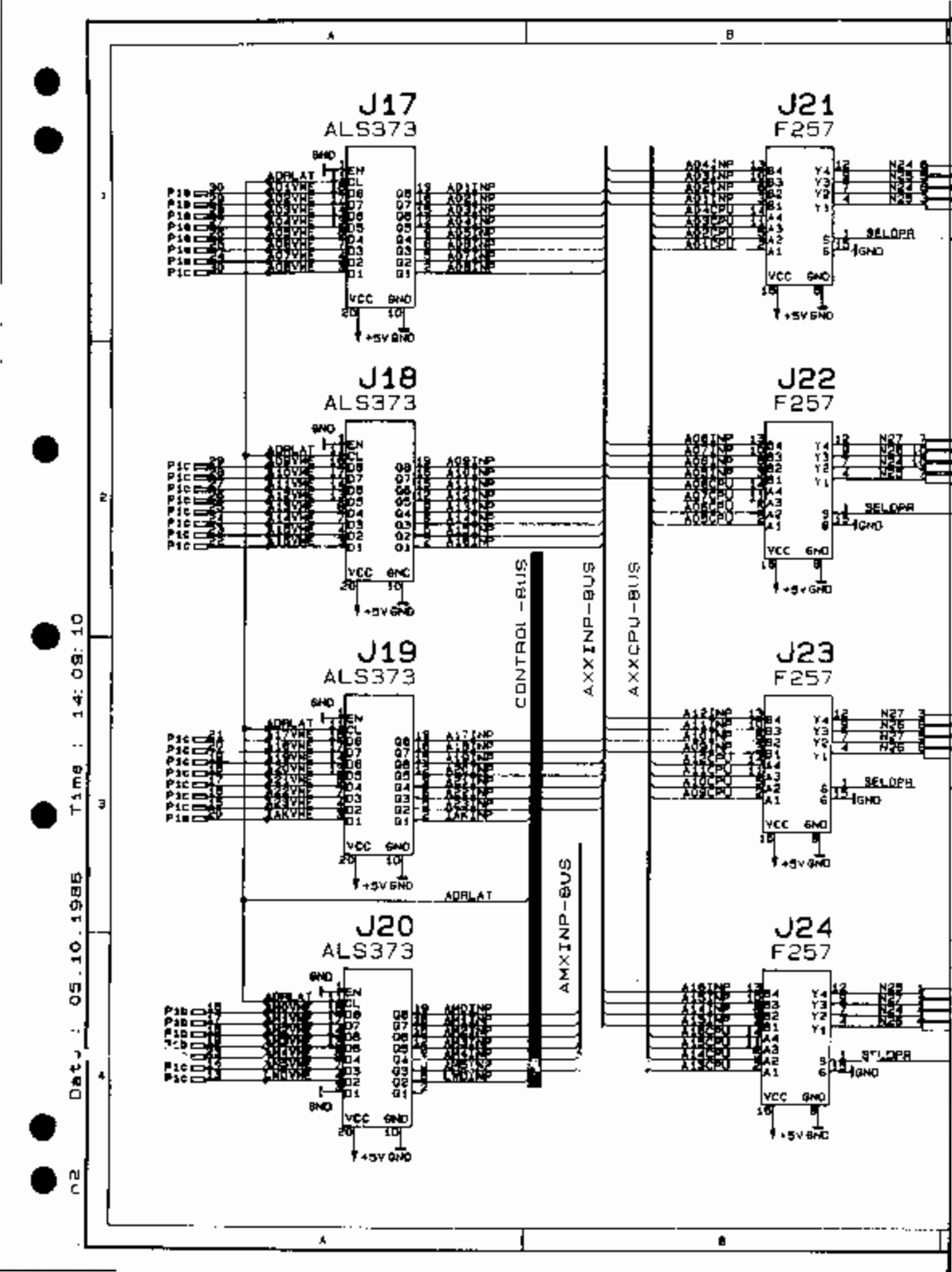


AXXCPR-BUS

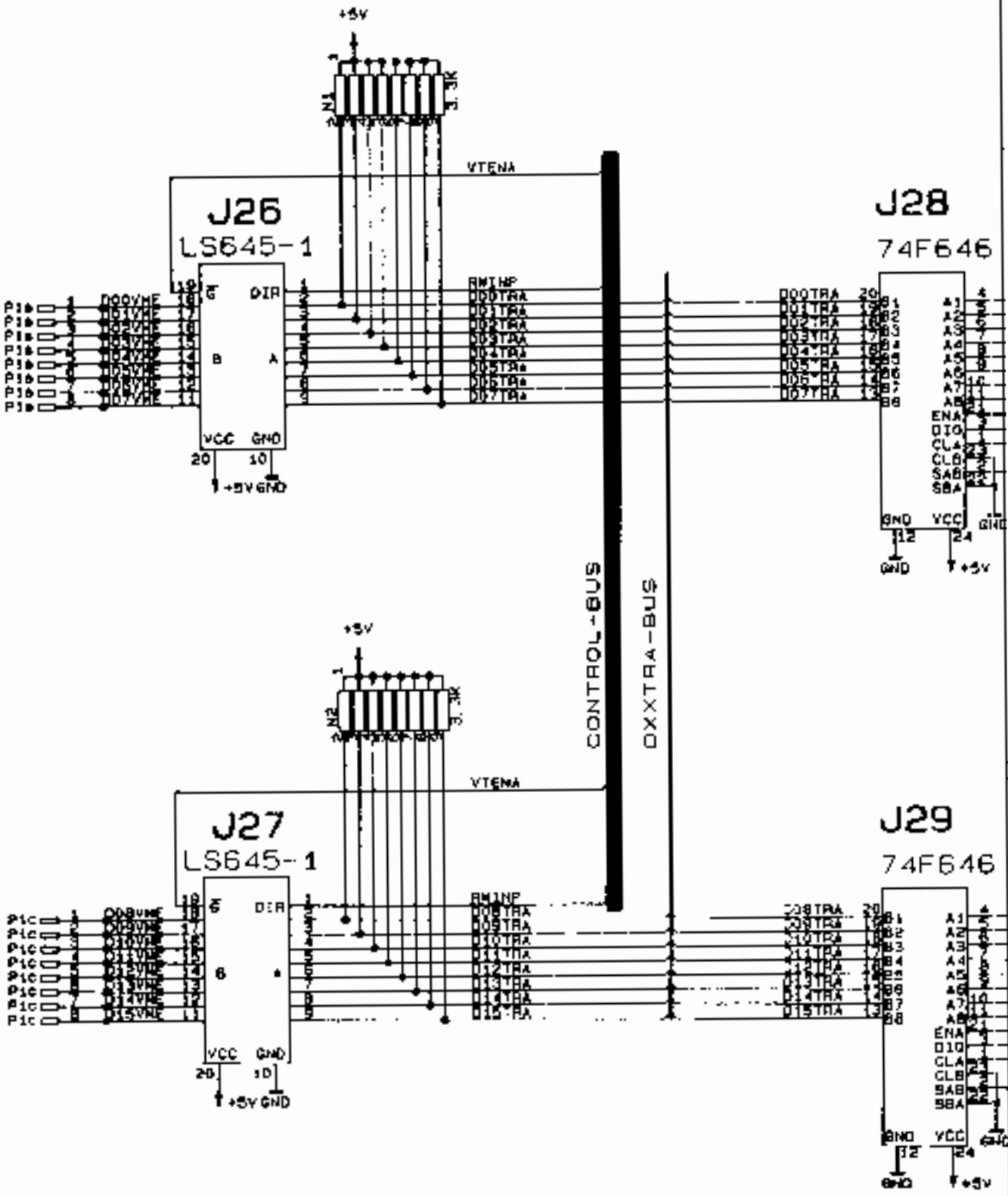
B18



DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		SRAM ARRAY	SHEET : FROM 13 SHEETS

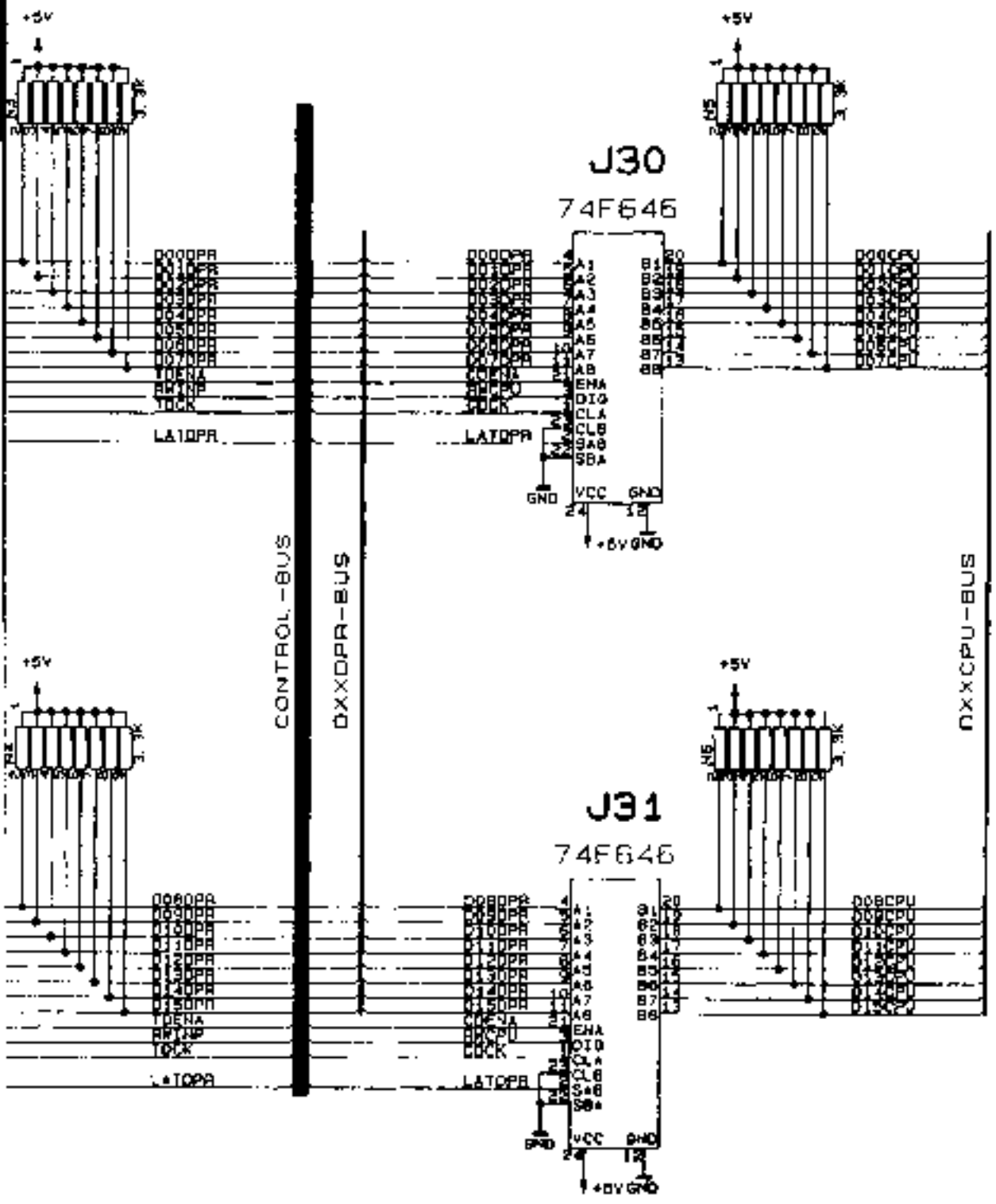

J17
ALS373
J18
ALS373
J19
ALS373
J20
ALS373
J21
F257
J22
F257
J23
F257
J24
F257
CONTROL-BUS
AXXINP-BUS
AXXCPU-BUS
AMXINP-BUS
ADRLAT
SELOPR
SELOPB
VCC
GND
+5V
Date : 05.10.1985 Time : 14:09:10

DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		ADDRESS-BUS	SHEET 2 FROM 13 SHEETS

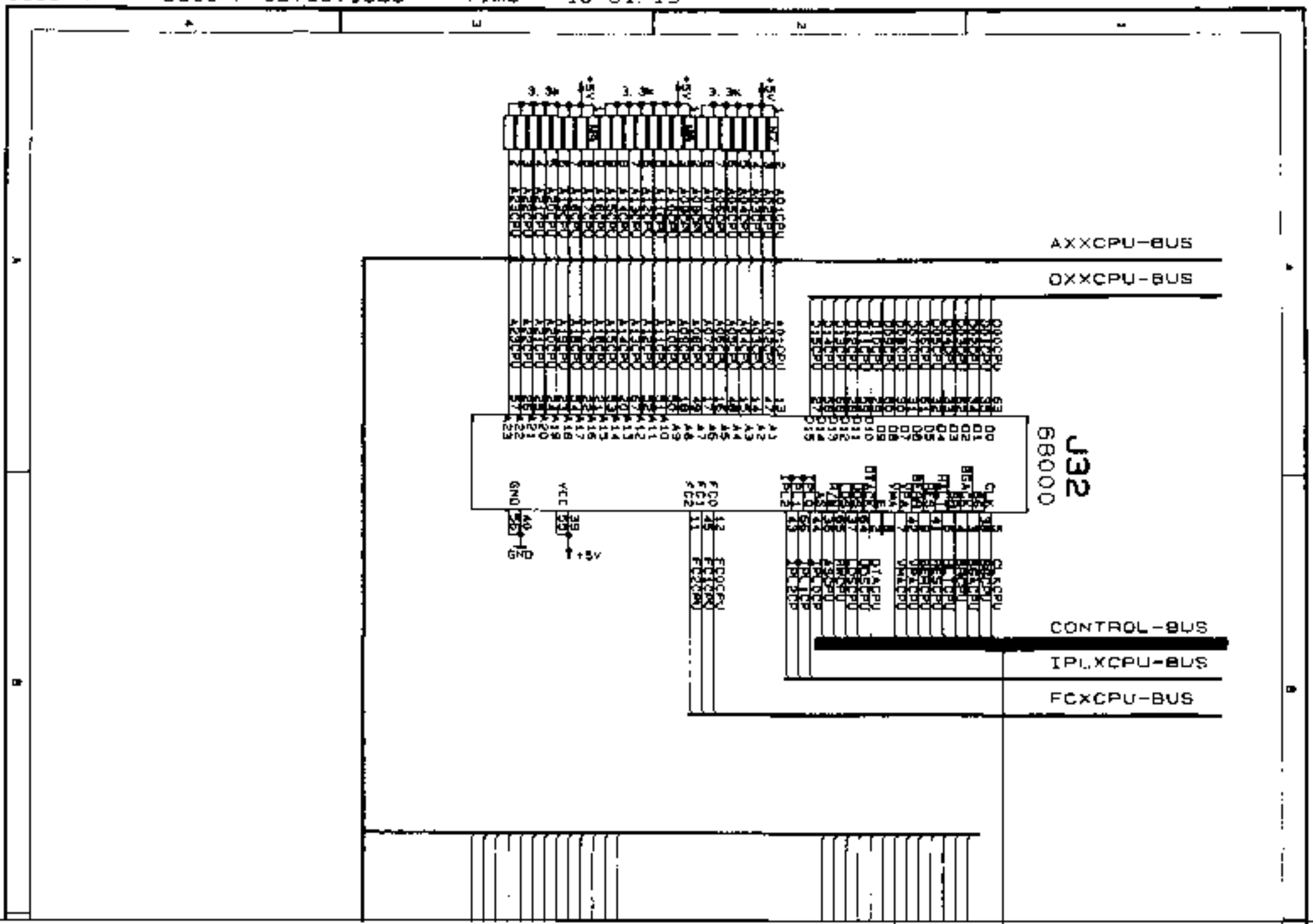


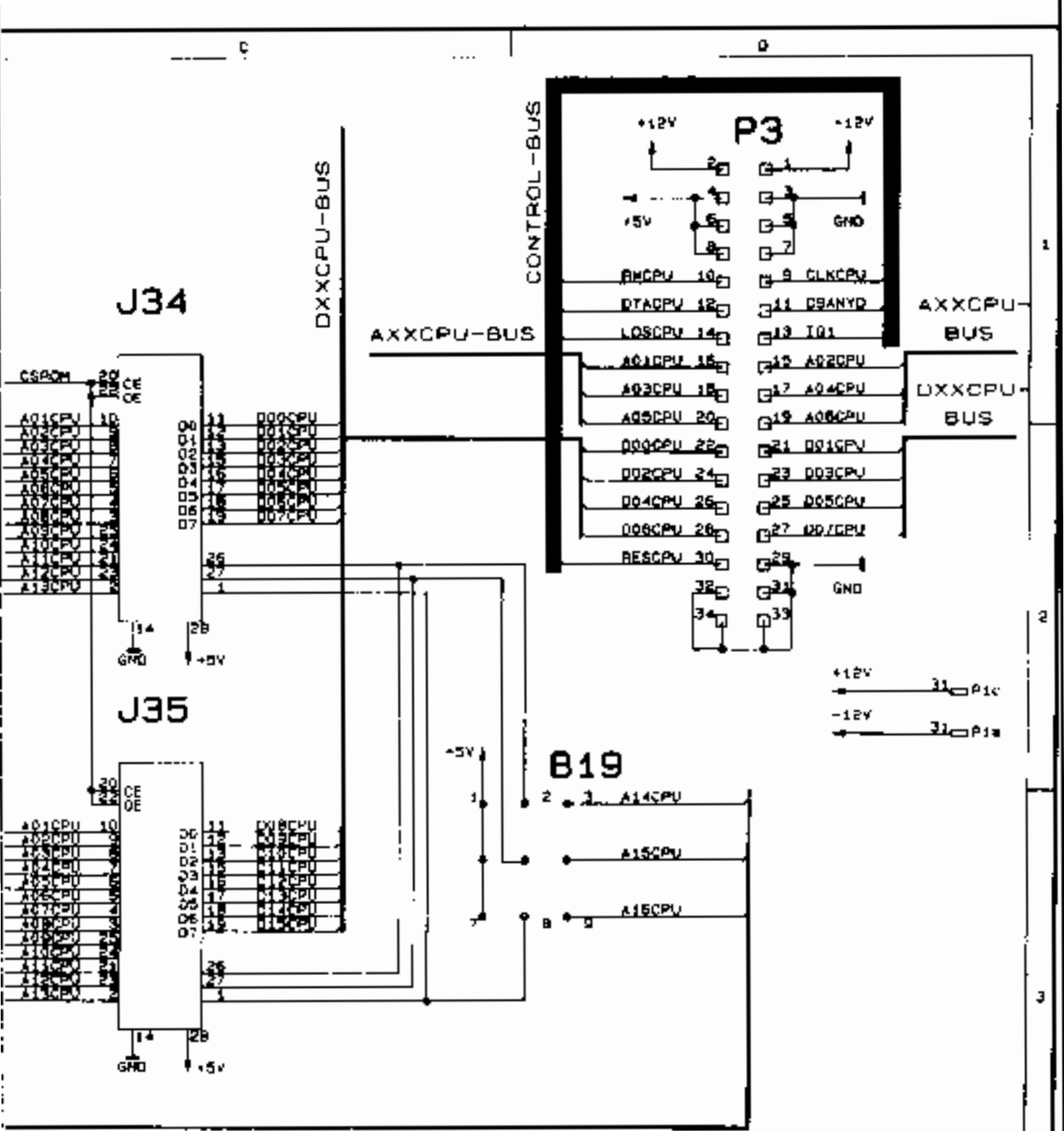
Date: 05.10.1986 Time: 10:18:06

068sh3

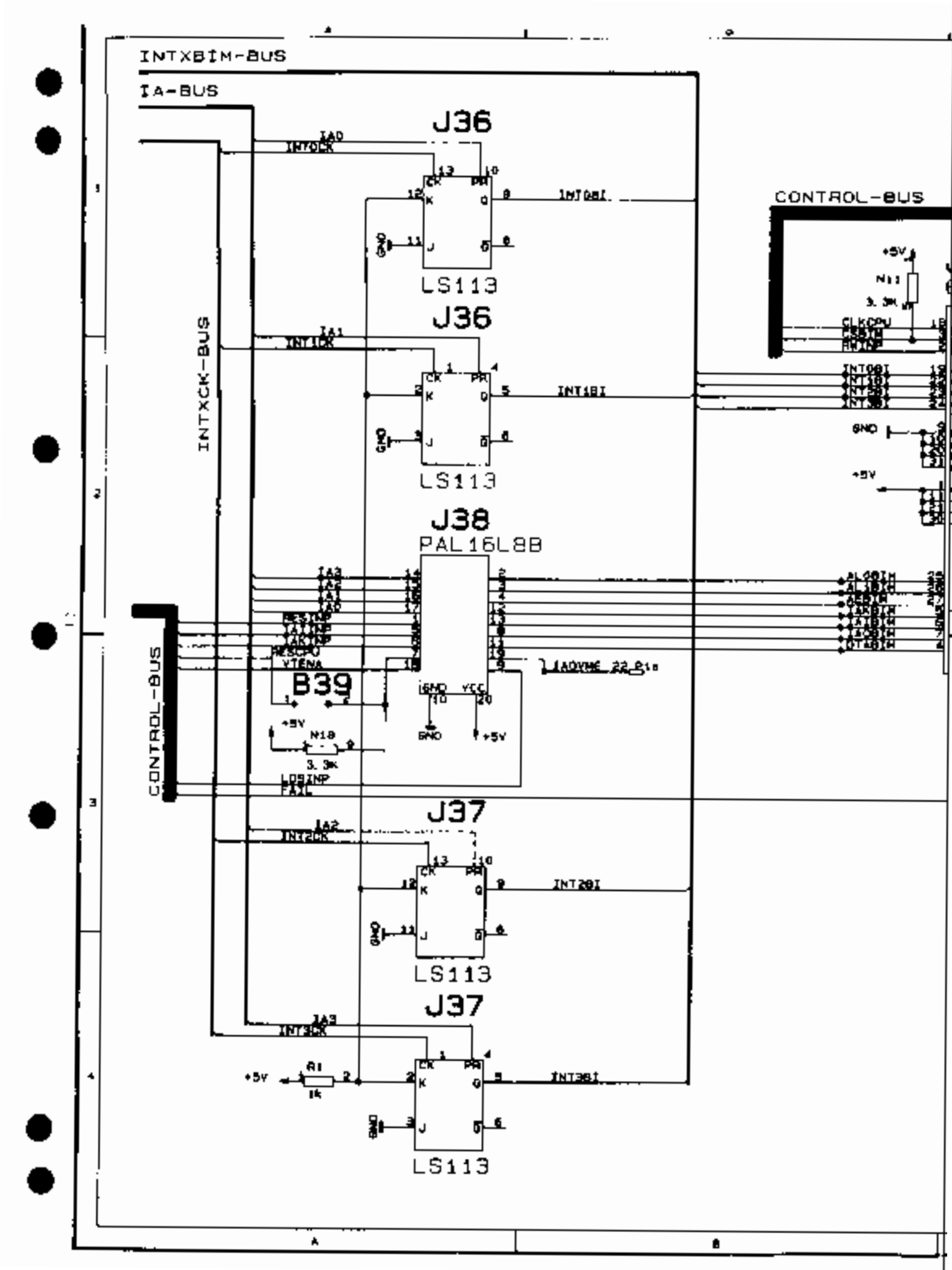


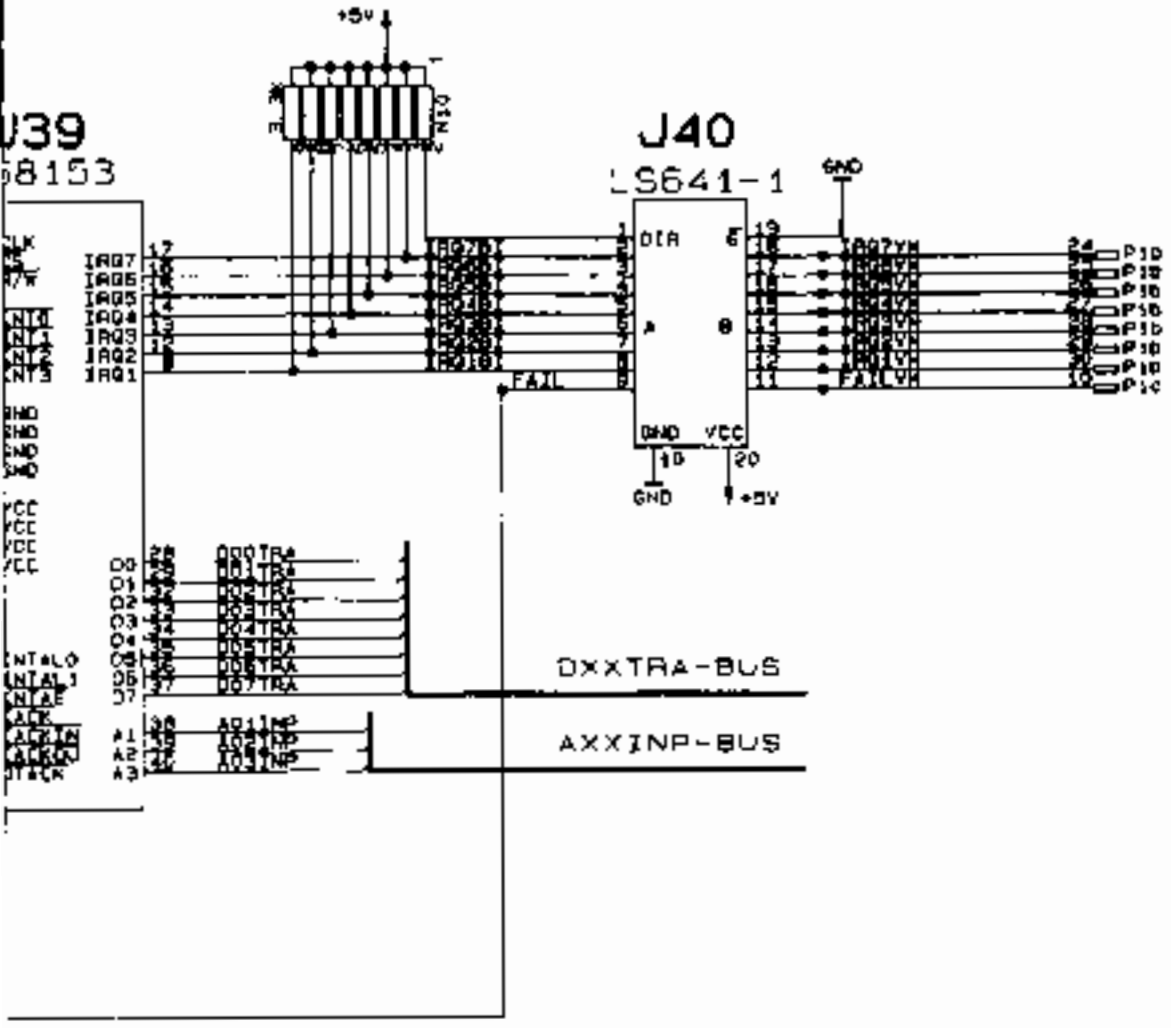
DATE	NAME		REV. 1
17.7.86	J.B.	SYS68K/ISCSI-1	
29.9.86	J.B.		
FORCE COMPUTERS		DATA-BUS	SHEET 3 FROM 13 SHEETS


AXXCPU-BUS
OXXCPU-BUS
J32
68000
CONTROL-BUS
IPLXCPU-BUS
FCXCPU-BUS

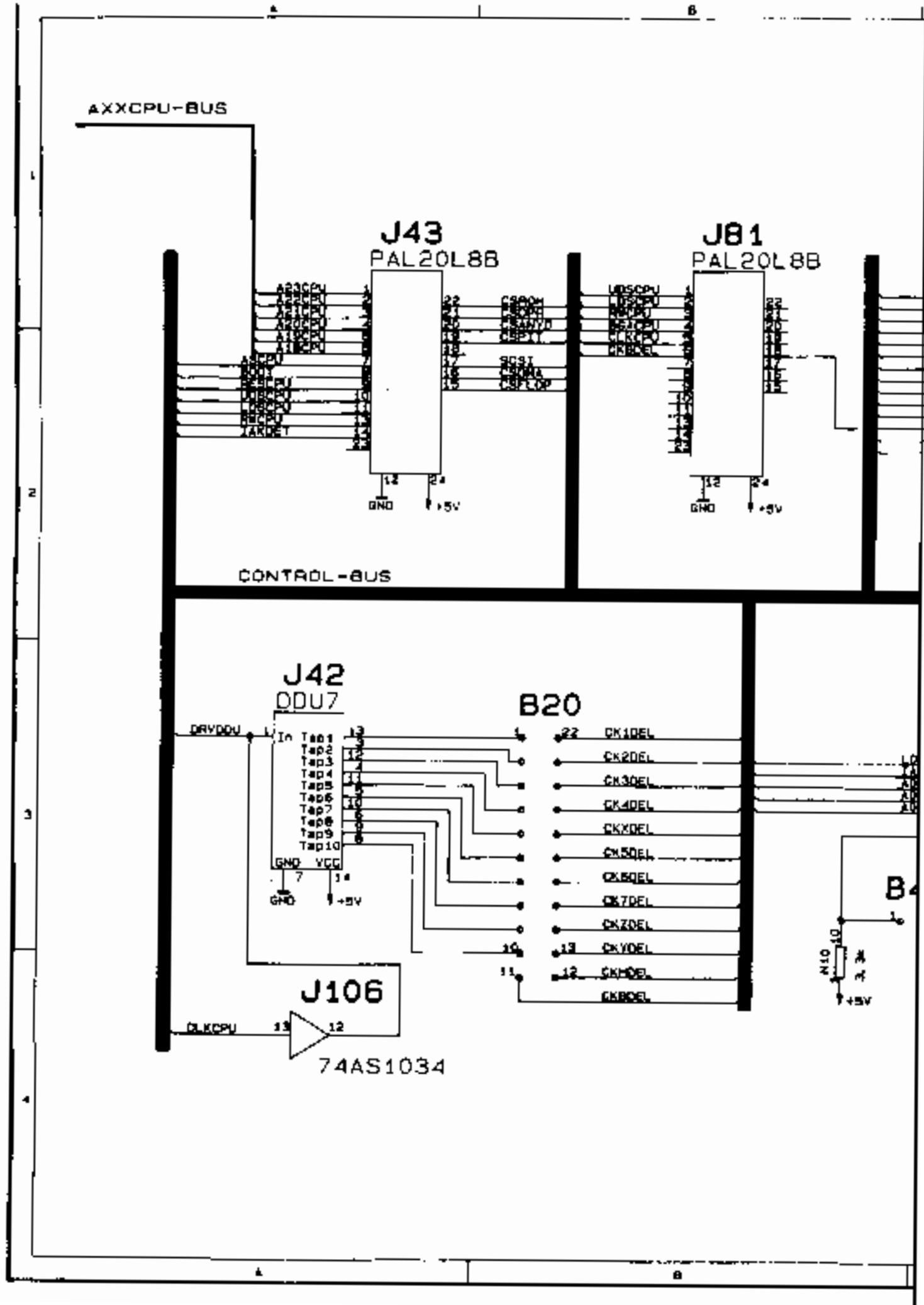


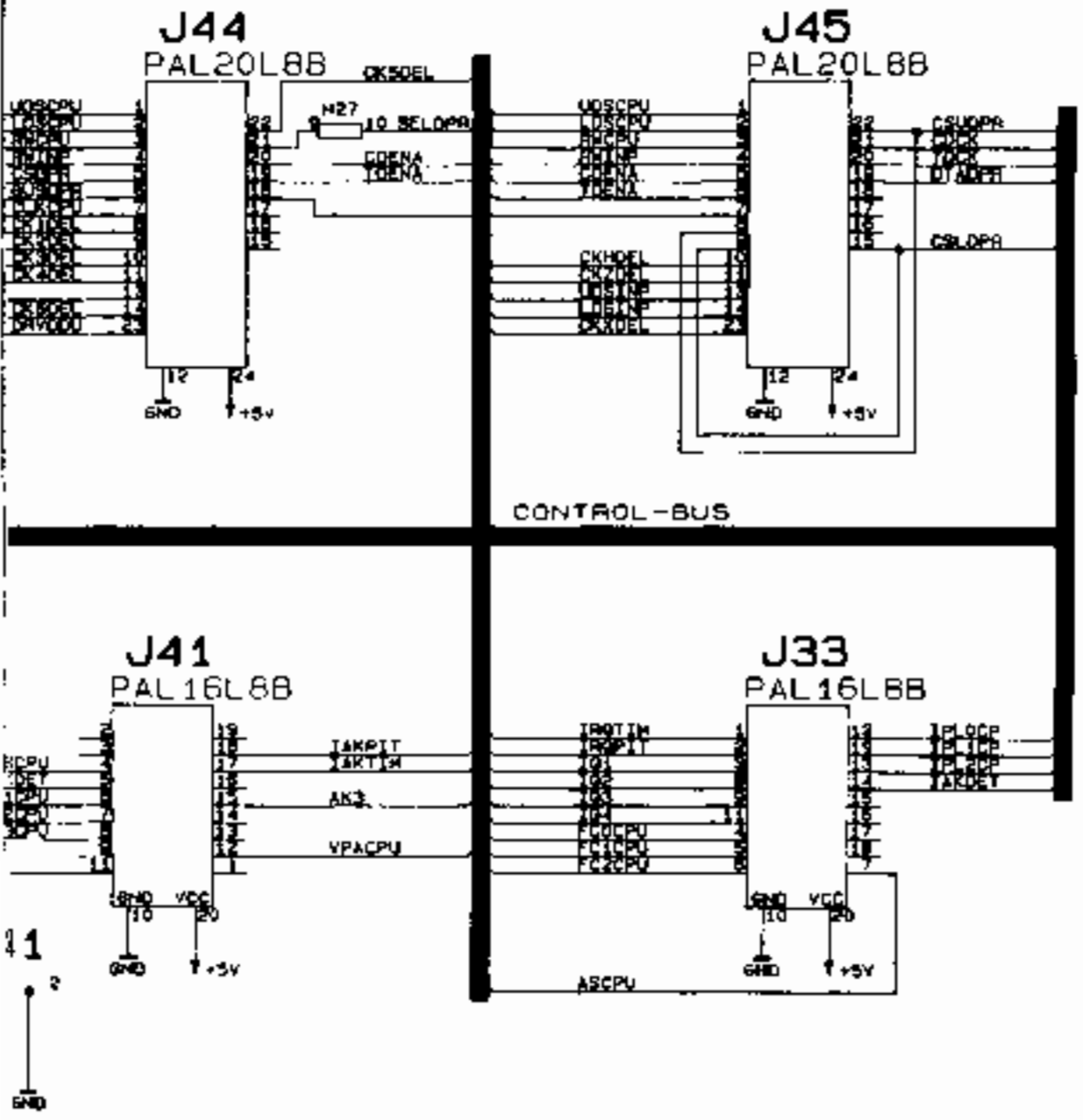
DATE	NAME		REV. 1
17.7.86	J.B.	SYS68K/ISCSI-1	
29.9.86	J.B.		
FORCE COMPUTERS		LOCAL CPU	SHEET 4 FROM 13 SHEETS


INTXBIM-BUS
IA-BUS
INTXCK-BUS
CONTROL-BUS
J36
LS113
J36
LS113
J38
PAL16L8B
B39
J37
LS113
J37
LS113
CONTROL-BUS
+5V
GND
3.3K
1K

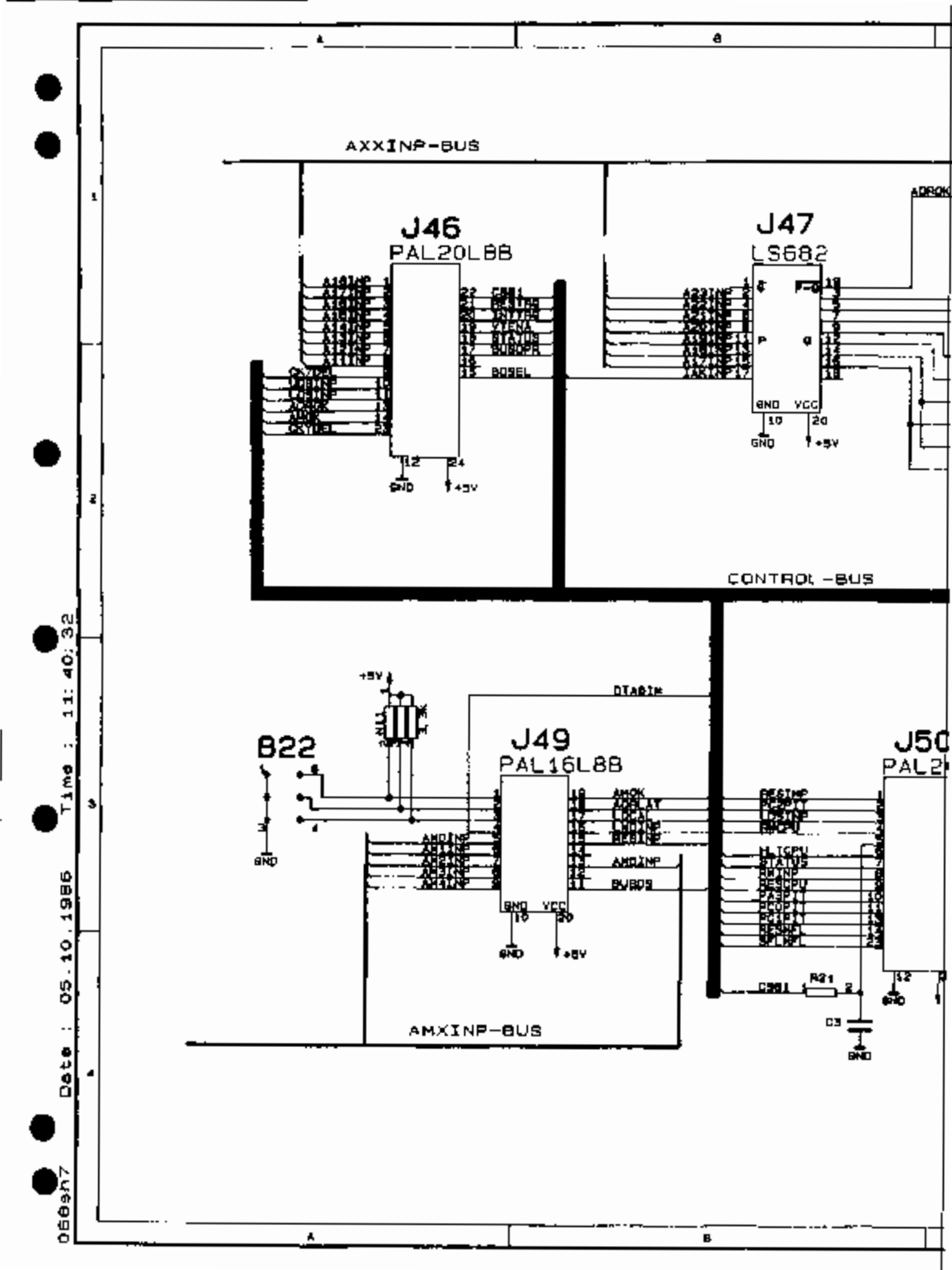


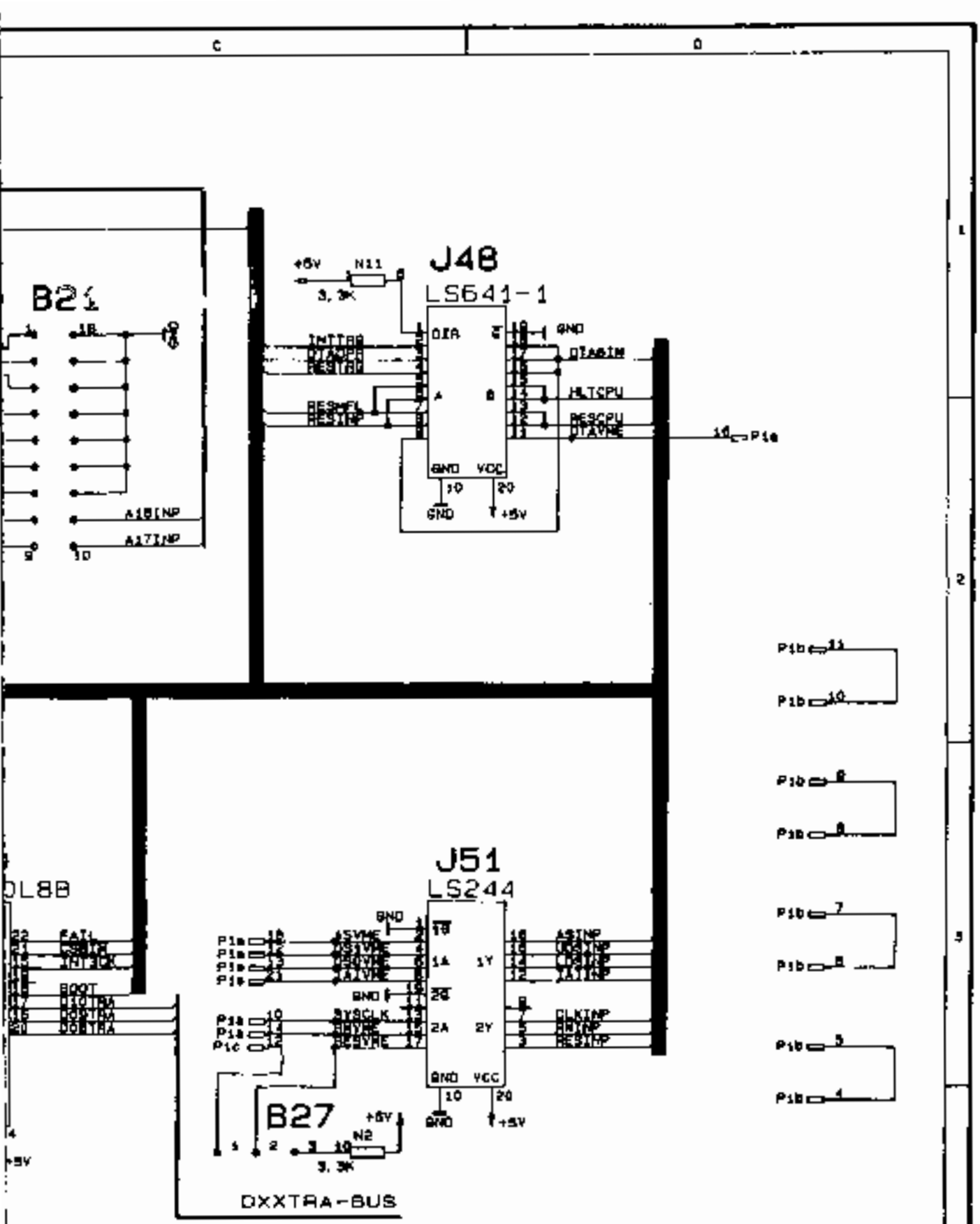
DATE	NAME		REV. 1
17.7.86	J.B.	SYS68K/ISCSI-1	
29.9.86	J.B.		
FORCE COMPUTERS		VME-BUS INTERRUPTER	SHEET 5 FROM 13 SHEETS


AXXCPU-BUS
J43
PAL20L8B
J81
PAL20L8B
GND
+5V
CONTROL-BUS
J42
ODU7
B20
DRVDDU
Tap1
Tap2
Tap3
Tap4
Tap5
Tap6
Tap7
Tap8
Tap9
Tap10
GND
VCC
CK1DEL
CK2DEL
CK3DEL
CK4DEL
CK5DEL
CK6DEL
CK7DEL
CK8DEL
CK9DEL
CK10DEL
J106
CLKCPU
74AS1034

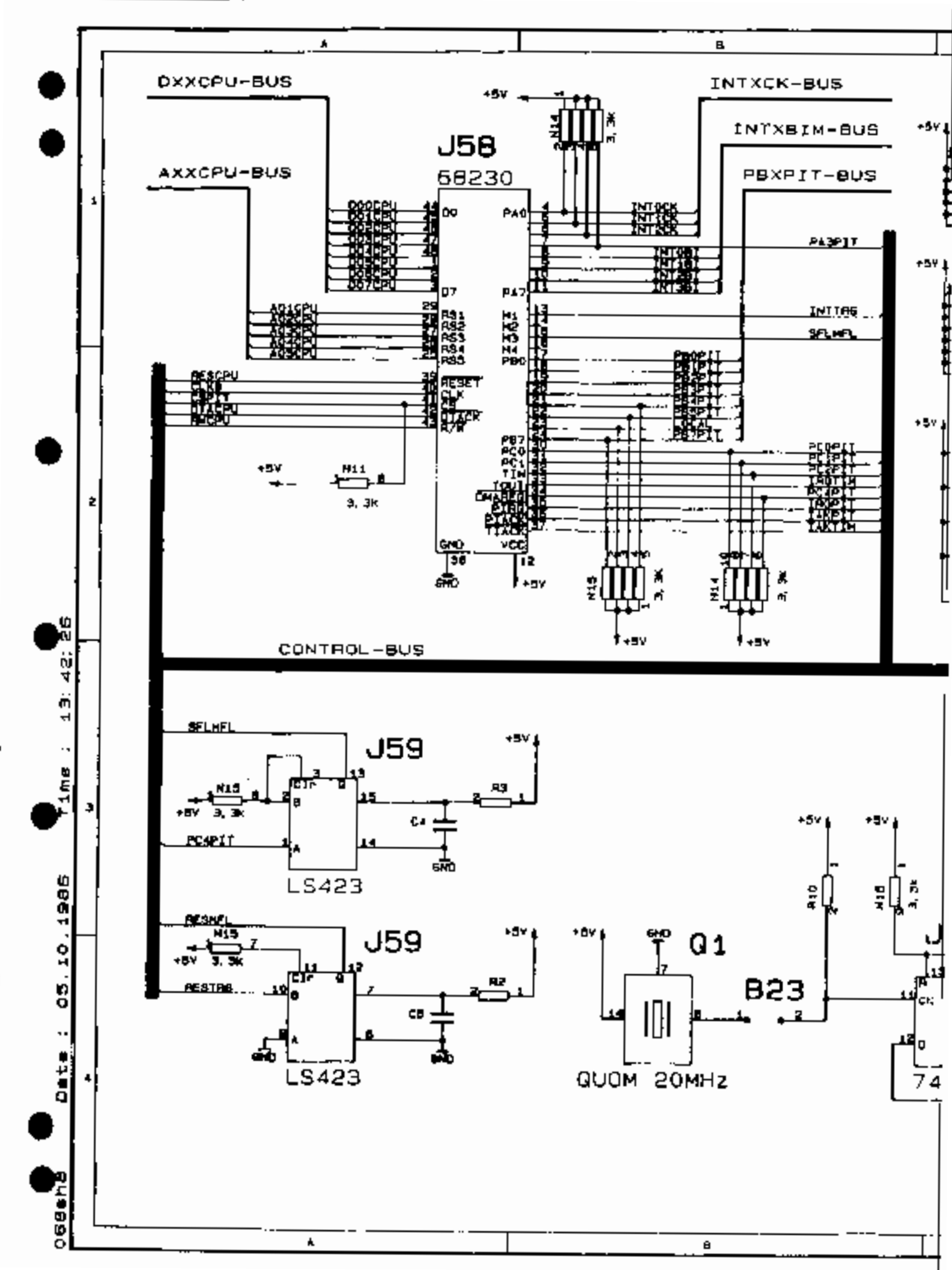


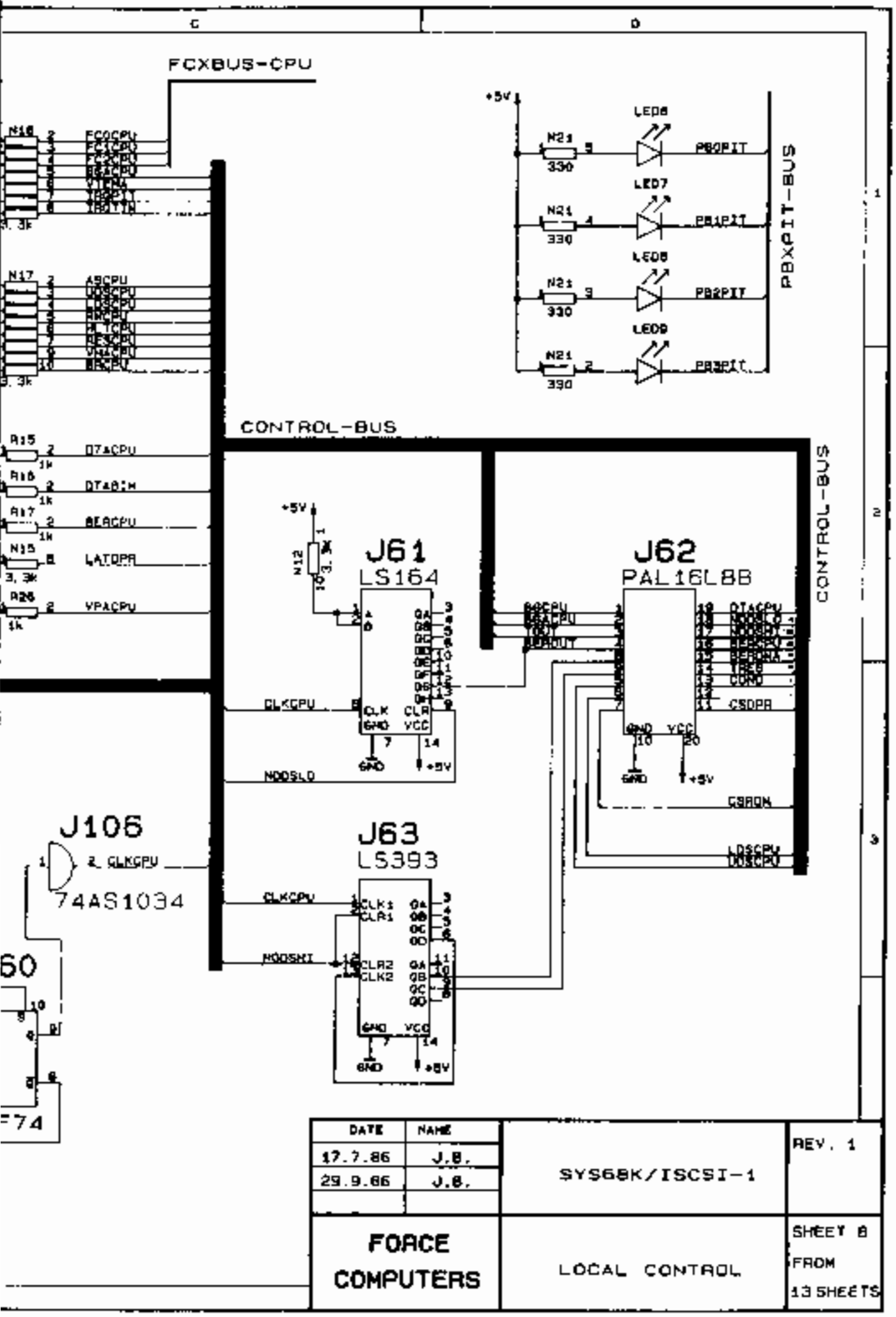
DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		DATA-BUS	SHEET 6 FROM 13 SHEETS


AXXINP-BUS
J46
PAL20L8B
J47
LS682
CONTROL-BUS
B22
J49
PAL16L8B
J50
AMXINP-BUS
Date : 05.10.1986 Time : 11:40:32

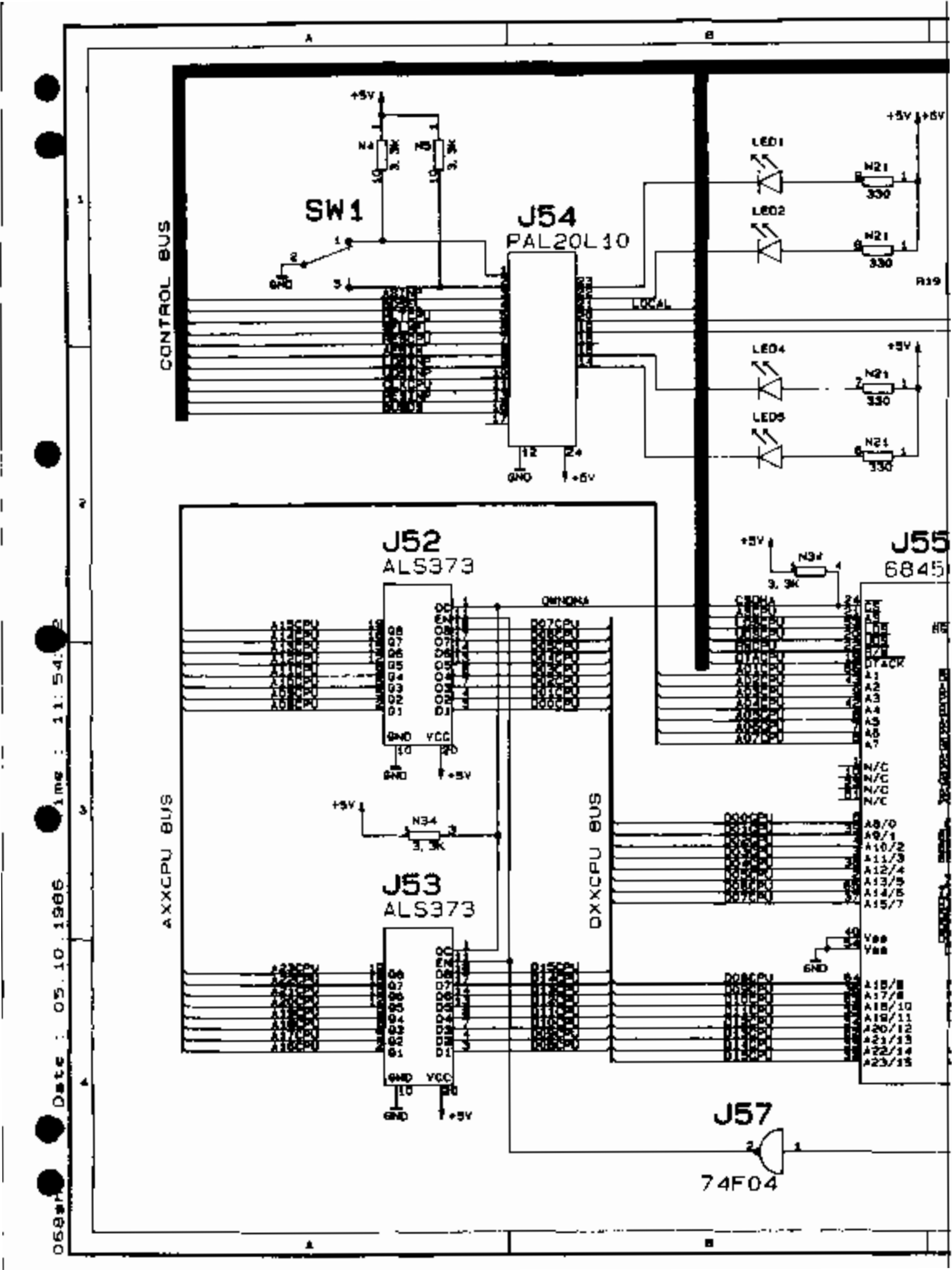


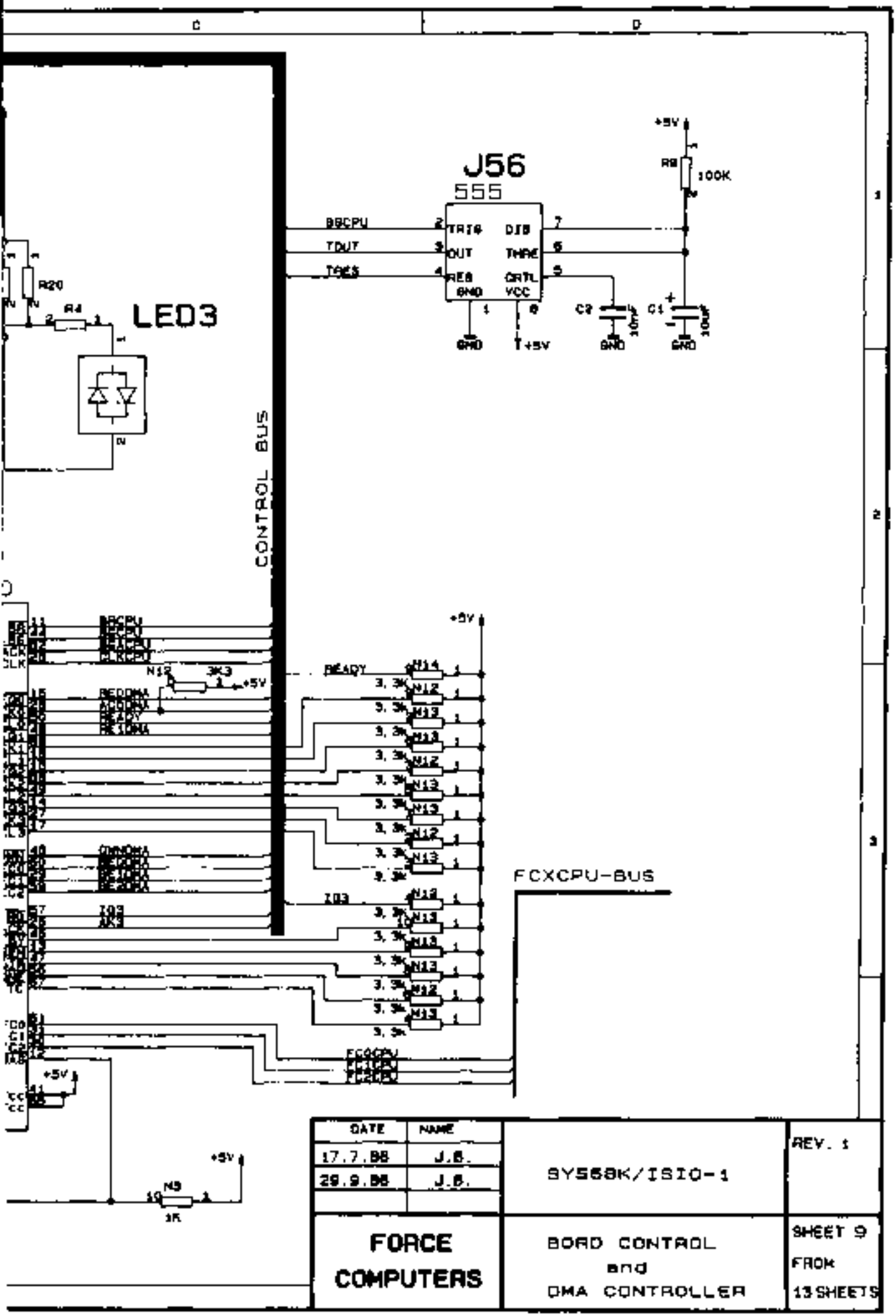
DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		VME ADDRESS DECODING	SHEET 7 FROM 13 SHEETS


DXXCPU-BUS
AXXCPU-BUS
INTXCK-BUS
INTXBIM-BUS
PBXPIT-BUS
J58
68230
CONTROL-BUS
J59
LS423
J59
LS423
Q1
B23
QUOM 20MHz
74
Date : 05.10.1986
Time : 13:42:26

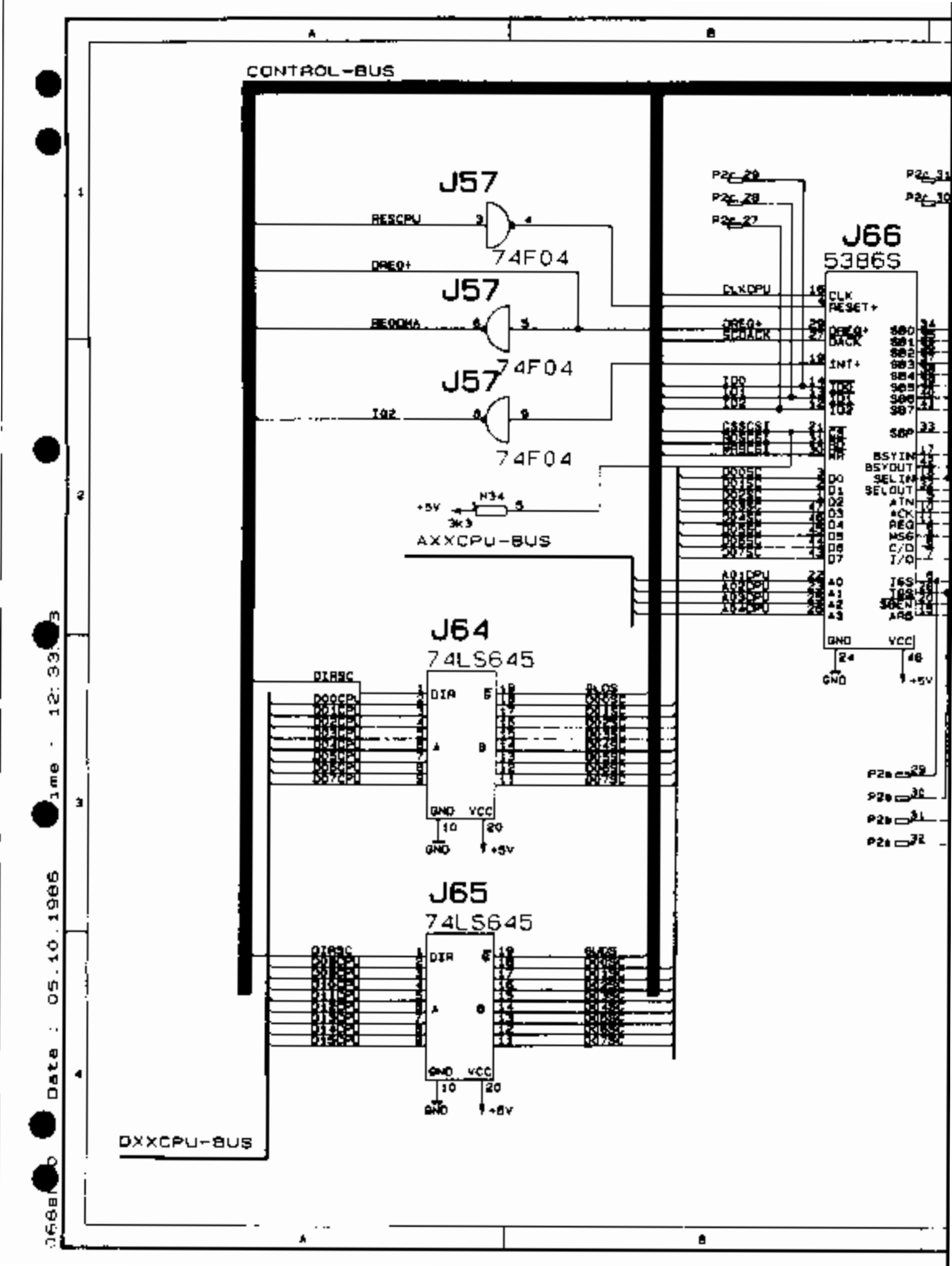


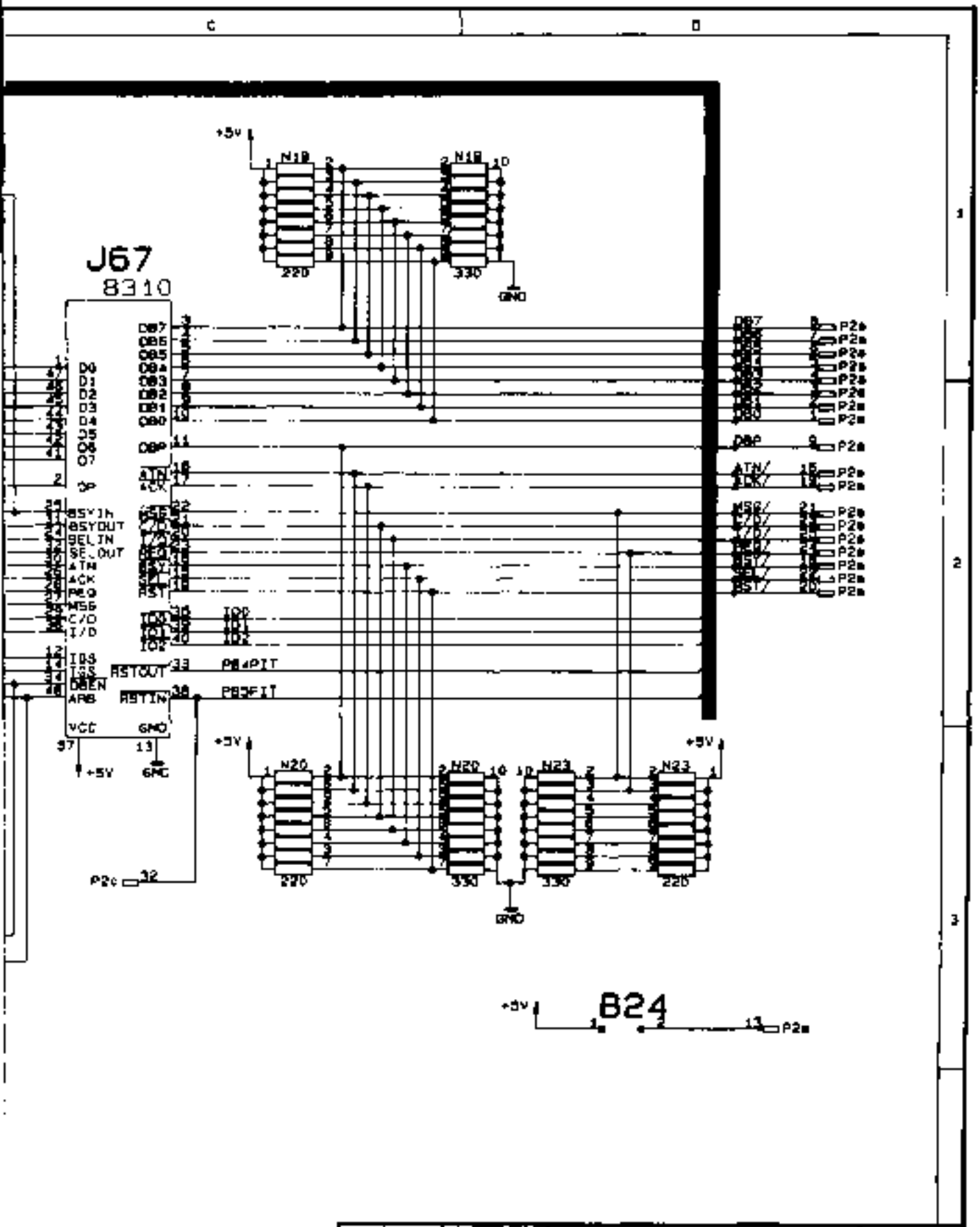
DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		LOCAL CONTROL	SHEET 8 FROM 13 SHEETS


SW1
J54
PAL20L10
CONTROL BUS
LED1
LED2
LED4
LED5
330
LOCAL
J52
ALS373
J53
ALS373
J55
6845
J57
74F04
AXXCPU BUS
DXXCPU BUS
+5V
GND
N34
3, 3K

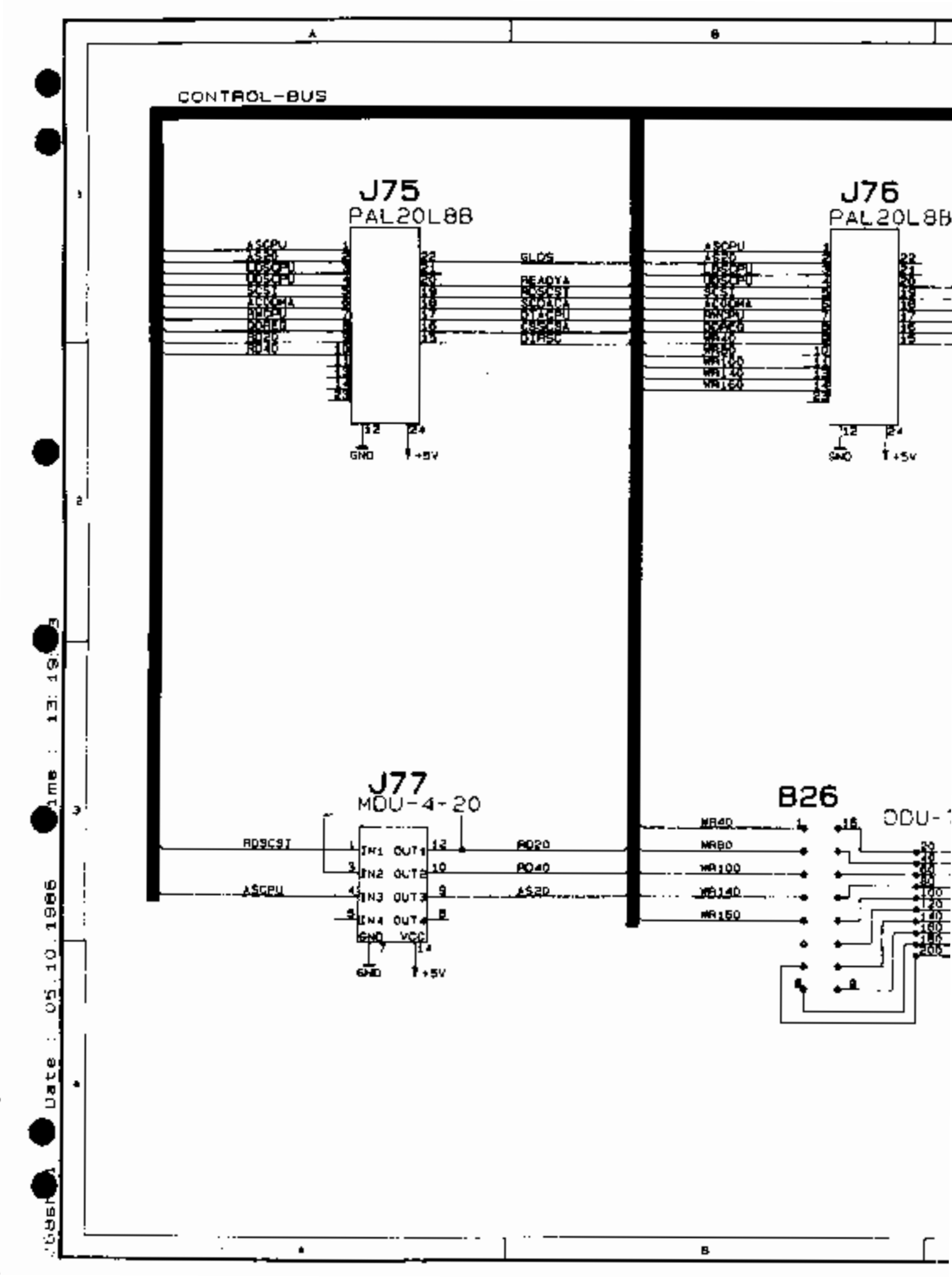


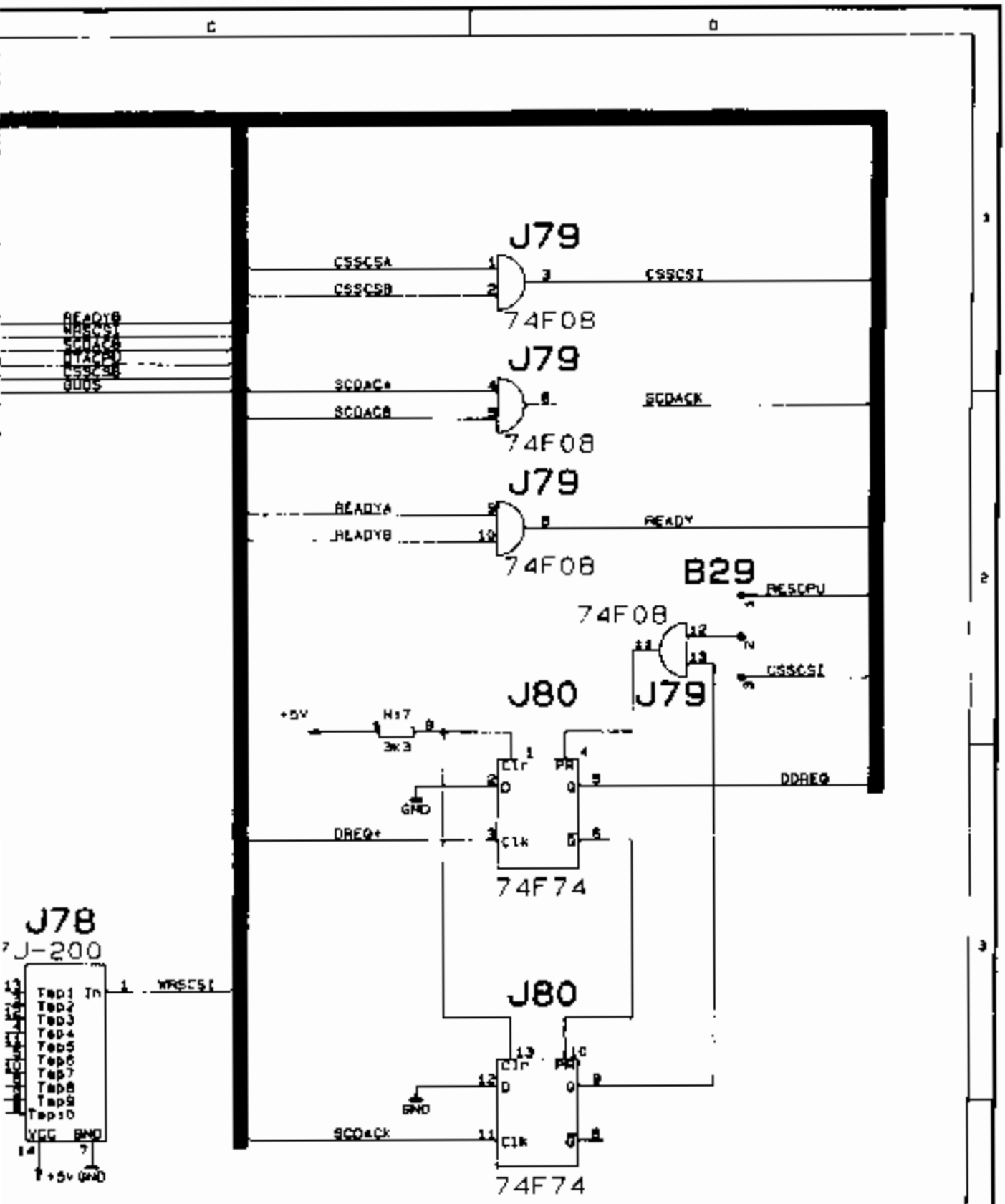
DATE	NAME		
17.7.88	J.B.	SY568K/ISIO-1	REV. 1
29.9.88	J.B.		
FORCE COMPUTERS		BORD CONTROL and DMA CONTROLLER	SHEET 9 FROM 13 SHEETS


CONTROL-BUS
J57
RESCPU
74F04
DREQ+
J57
BEQDMA
74F04
J57
IQ2
74F04
+5V
N34
3K3
AXXCPU-BUS
J66
5386S
CLXCPU
CLK
RESET+
DREQ+
DACK
INT+
IDO
IO1
IO2
CSSCSI
SDP
BSYIN
BSYOUT
SELIN
SELOUT
ATN
ACK
REQ
MSG
C/D
I/O
IGS
IOS
SDEN
ARB
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
GND
VCC
J64
74LS645
DIR
A
B
GND
VCC
J65
74LS645
DXXCPU-BUS
Date : 05.10.1986
Time - 12:33:3

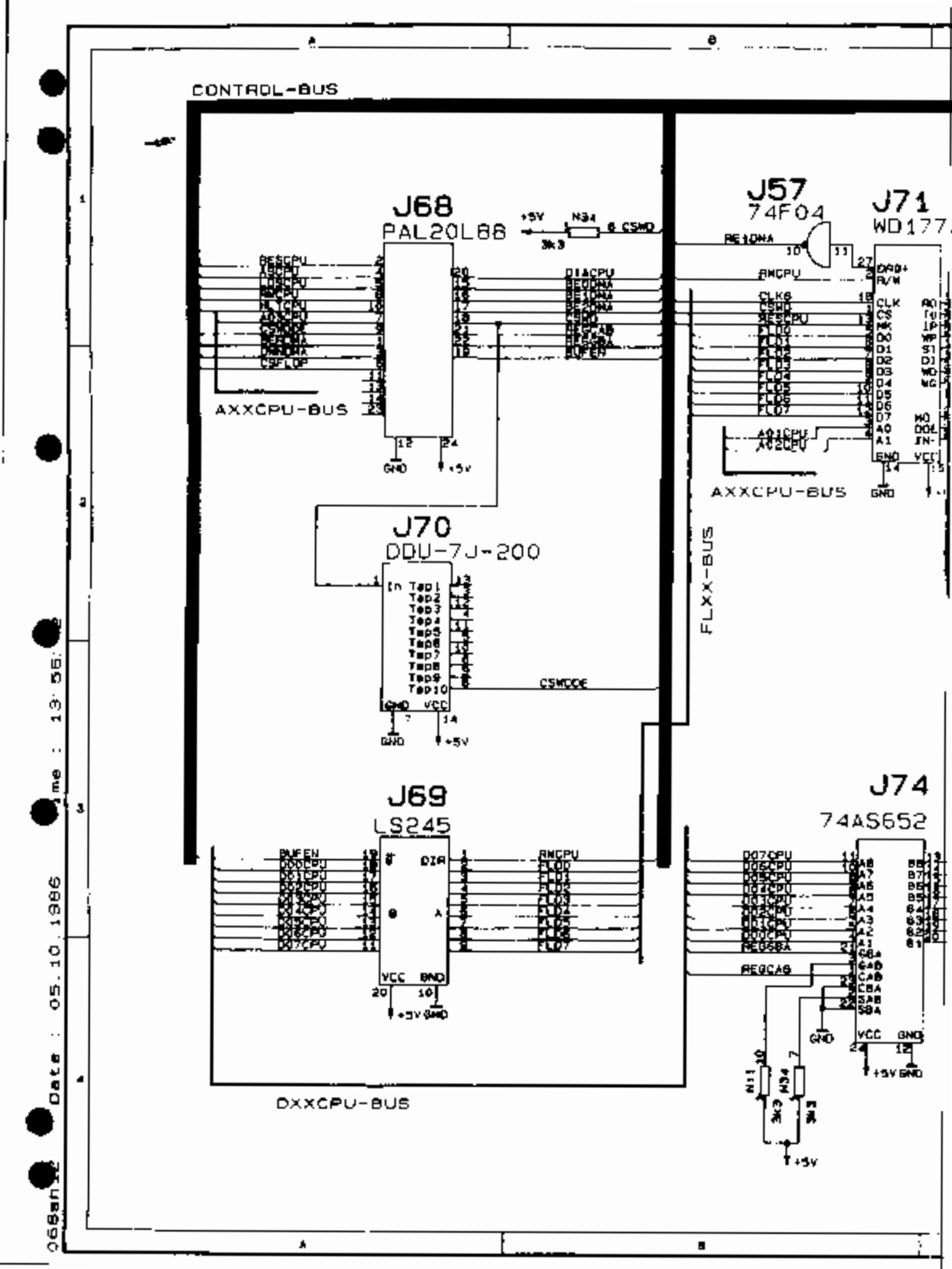


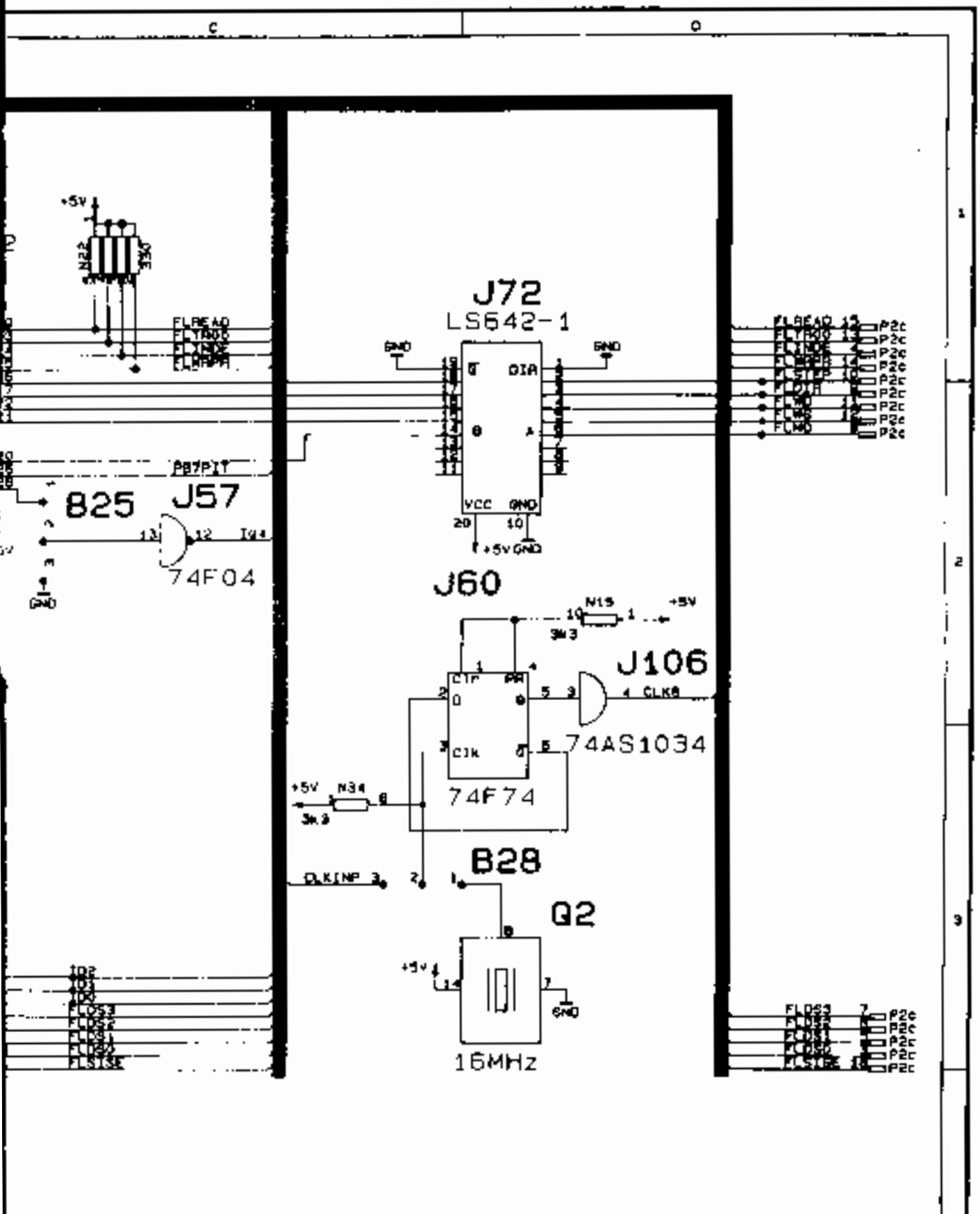
DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		SCSI-CONTROLER	SHEET 10 FROM 13 SHEETS


CONTROL-BUS
J75
PAL20L8B
J76
PAL20L8B
J77
MDU-4-20
B26
GND
+5V
Date : 05.10.1986
Time : 13:19

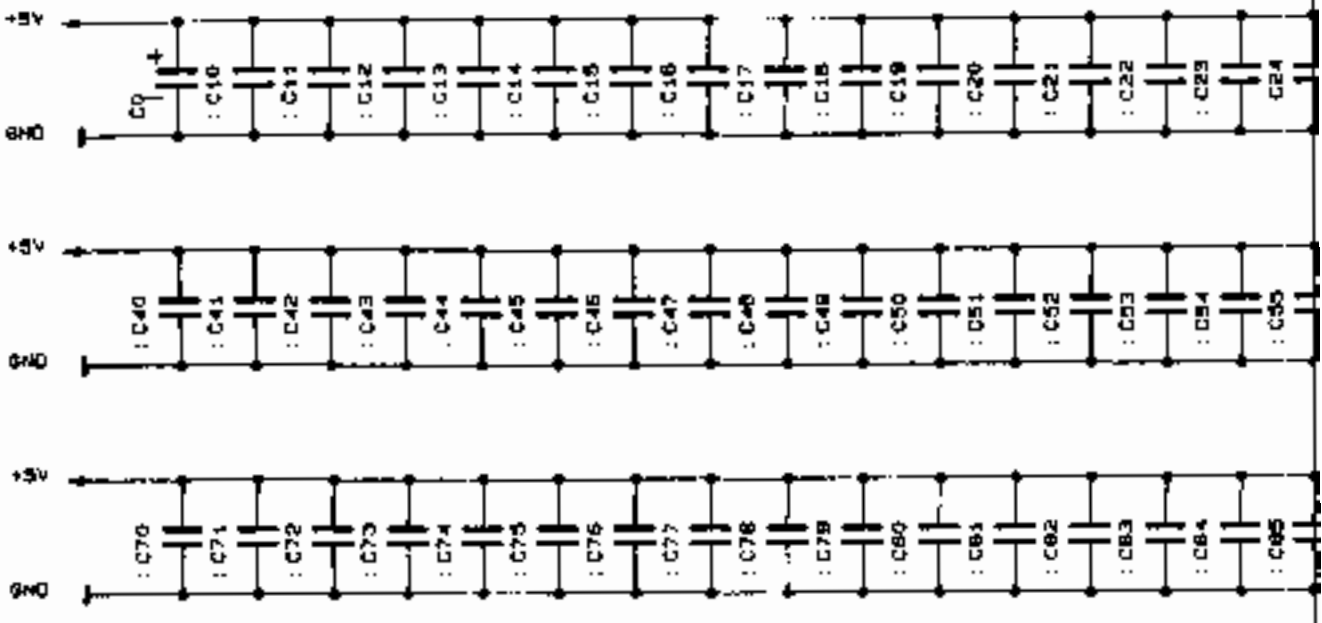


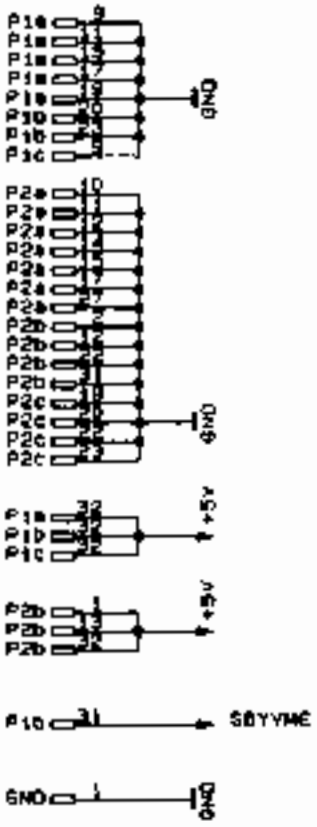
DATE	NAME		REV. 1
17.7.86	J.B.	SYS68K/ISCSI-1	
29.9.86	J.B.		
FORCE COMPUTERS		NCR-CONTROLL-DAL	SHEET 11 FROM 13 SHEETS

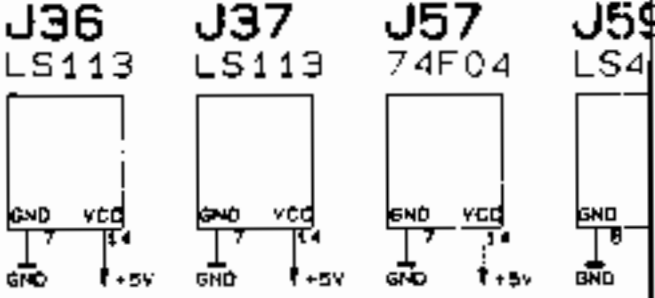

CONTROL-BUS
J68
PAL20L88
+5V
N34
3k3
6 CSWD
AXXCPU-BUS
GND
+5V
J57
74F04
J71
WD177
AXXCPU-BUS
J70
DDU-7J-200
CSWCOE
FLXX-BUS
J69
LS245
DIR
VCC
GND
J74
74AS652
N11
N34
3k3
3k3
+5V
DXXCPU-BUS

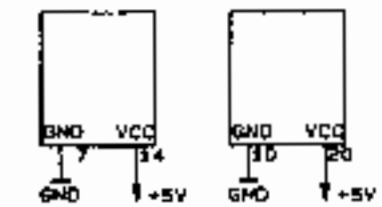


DATE	NAME		REV. 1
17.7.86	J.B.	SYS68K/ISCSI-1	
29.8.86	J.B.		
FORCE COMPUTERS		SCSI-CONTROLER	SHEET 12 FROM 13 SHEETS



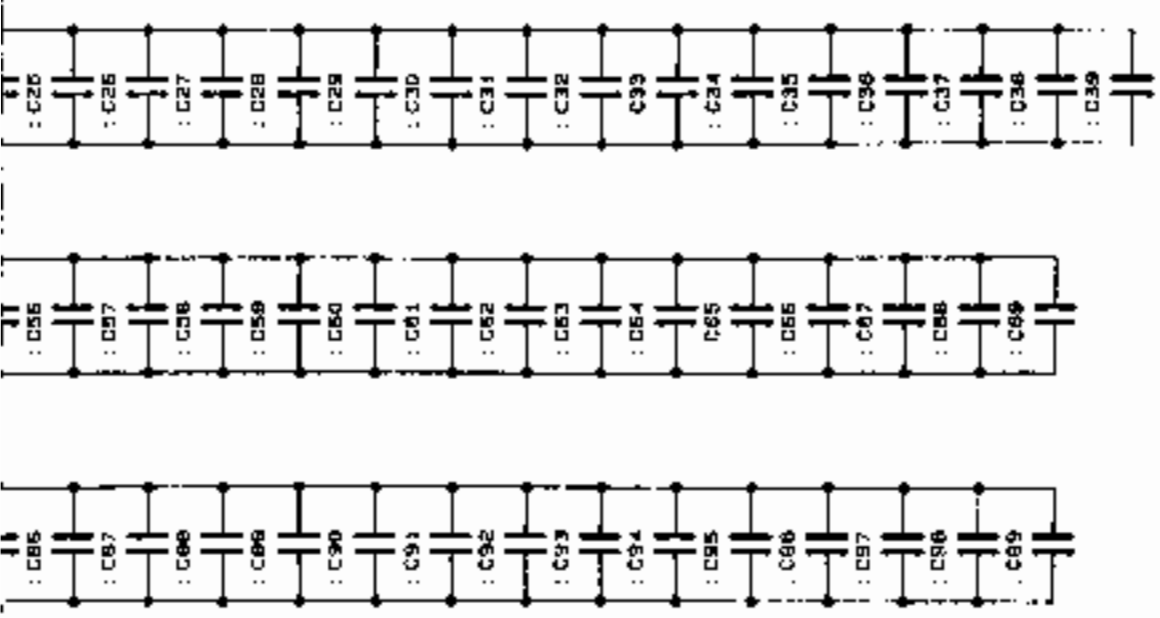


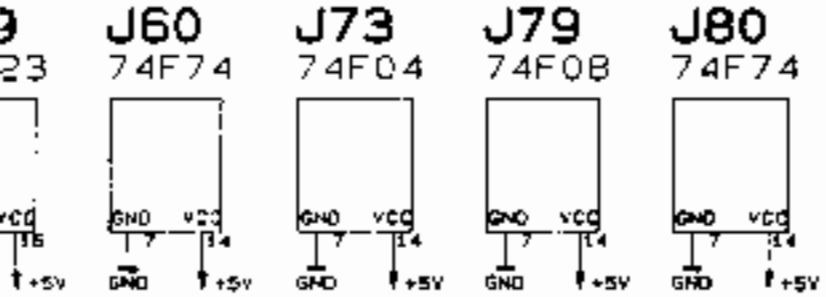




Date : 05.10.1986 Time : 15:25:

68sH





DATE	NAME		
17.7.86	J.B.	SYS68K/ISCSI-1	REV. 1
29.9.86	J.B.		
FORCE COMPUTERS		POWER	SHEET 13 FROM 13 SHEETS

Appendix G

Connector PIN Assignments of the SYS68K/ISCSI-1

Appendix G

Connector Pin Assignments P1

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	D00		D08
2	D01		D09
3	D02		D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	
12	DS1*		SYSRESET*
13	DS0*		LWORD*
14	WRITE*		AM5
15	GND		A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*		A17
22	IACKOUT*		A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V		+12V
32	+5V	+5V	+ 5V

Appendix G

Connector Pin Assignments P2

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
1	DB 0	VCC	N.C.
2	DB 1	GND	N.C.
3	DB 2	N.C.	Drive Select 0
4	DB 3	N.C.	Index
5	DB 4	N.C.	Drive Select 1
6	DB 5	N.C.	Drive Select 2
7	DB 6	N.C.	Drive Select 3
8	DB 7	N.C.	Motor On
9	DB P	N.C.	Direction In
10	GND	N.C.	Step
11	GND	N.C.	Write Data
12	GND	GND	Write Gate
13	TERMPWR	VCC	Track 000
14	GND	N.C.	Write Protect
15	GND	N.C.	Read Data
16	ATN	N.C.	Side Select
17	GND	N.C.	N.C.
18	BSY	N.C.	N.C.
19	ACK	N.C.	GND
20	RST	N.C.	GND
21	MSG	N.C.	N.C.
22	SEL	GND	GND
23	C/D	N.C.	GND
24	REQ	N.C.	N.C.
25	I/O	N.C.	N.C.
26	N.C.	N.C.	N.C.
27	GND	N.C.	RESERVED
28	N.C.	N.C.	RESERVED
29	RESERVED	N.C.	RESERVED
30	RESERVED	N.C.	RESERVED
31	RESERVED	GND	RESERVED
32	RESERVED	VCC	RESERVED

This page was intentionally left blank

APPENDIX H
COMPONENT PART LIST

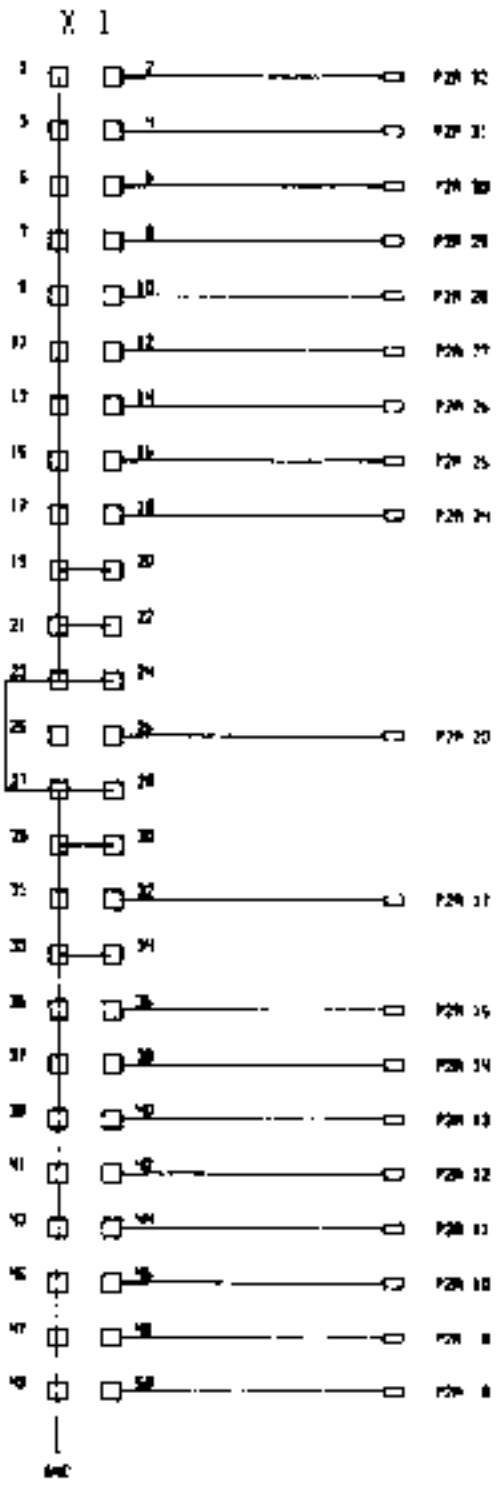
FOR INTERNAL USE ONLY

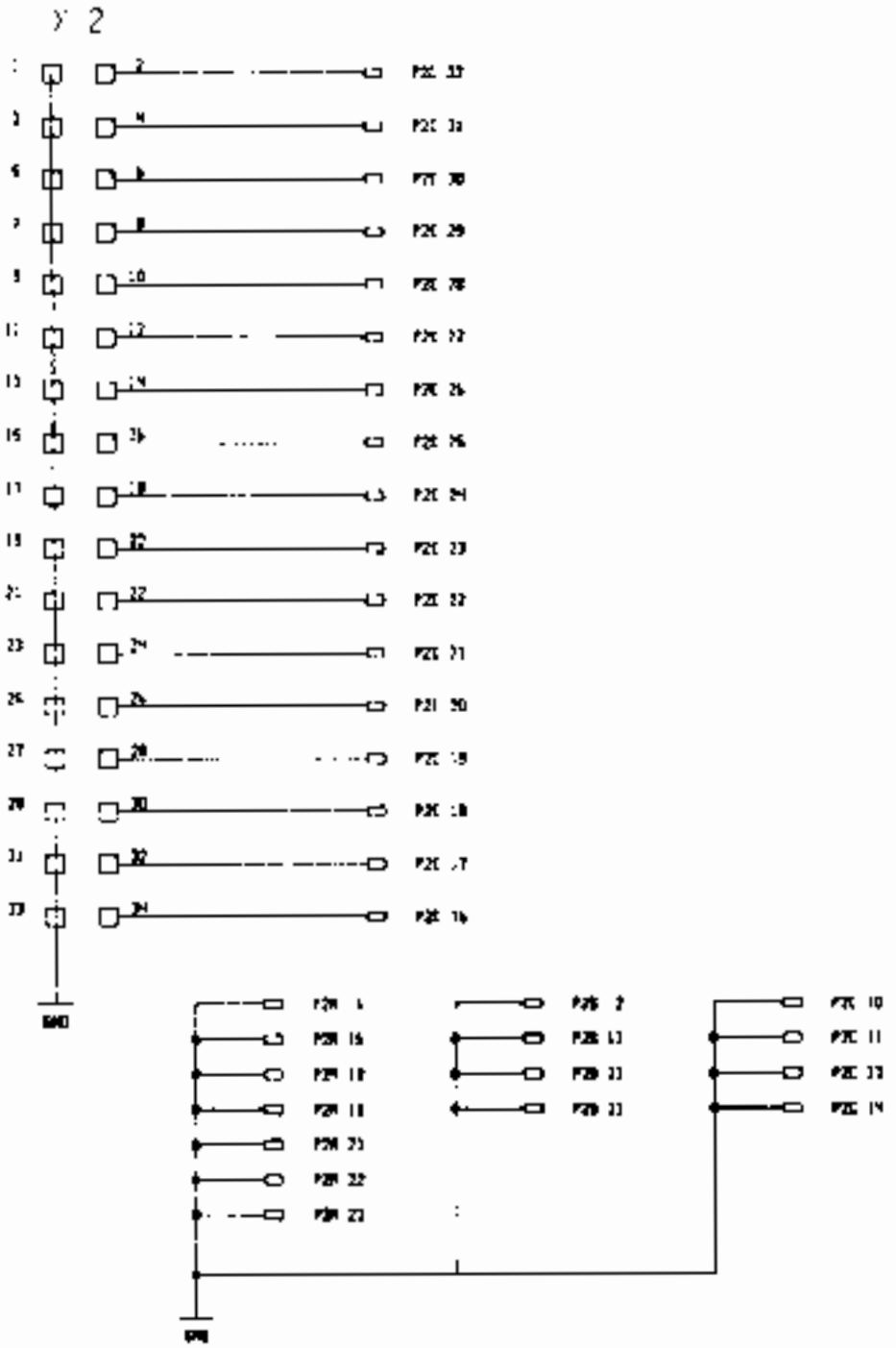
This page was intentionally left blank

Appendix I

Circuit Schematics of the SYS68K/ISCSI-1BPS

This page was intentionally left blank


X 1



DATE	REVD	SYS68K-ISCSI-18PS	REV 0
25.7.96	J.B.		
FORCE COMPUTERS			SHEET 1 FROM 1 SHEETS

Appendix J

Connector PIN Assignments of the SYS68K/ISCSI-1BPS

The ISCSI-1BPS P2 Pin Assignment

Pin Number	Row A Signal Mnemonic	Row B Signal Mnemonic	Row C Signal Mnemonic
32	DB 0	VCC	N.C.
31	DB 1	GND	N.C.
30	DB 2	N.C.	Drive Select 0
29	DB 3	N.C.	Index
28	DB 4	N.C.	Drive Select 1
27	DB 5	N.C.	Drive Select 2
26	DB 6	N.C.	Drive Select 3
25	DB 7	N.C.	Motor On
24	DB P	N.C.	Direction In
23	GND	N.C.	Step
22	GND	N.C.	Write Data
21	GND	GND	Write Gate
20	TERMPWR	VCC	Track 000
19	GND	N.C.	Write Protect
18	GND	N.C.	Read Data
17	ATN	N.C.	Side Select
16	GND	N.C.	N.C.
15	BSY	N.C.	N.C.
14	ACK	N.C.	GND
13	RST	N.C.	GND
12	MSG	N.C.	N.C.
11	SEL	GND	GND
10	C/D	N.C.	GND
9	REQ	N.C.	N.C.
8	I/O	N.C.	N.C.
7	N.C.	N.C.	N.C.
6	GND	N.C.	RESERVED
5	N.C.	N.C.	RESERVED
4	RESERVED	N.C.	RESERVED
3	RESERVED	N.C.	RESERVED
2	RESERVED	GND	RESERVED
1	RESERVED	VCC	RESERVED

Appendix J

The ISCSI-1BPS X1 Pin Assignment

Pin No.	Signal Mnemonic	Pin No.	Signal Mnemonic
2	DB 0	1	GND
4	DB 1	3	GND
6	DB 2	5	GND
8	DB 3	7	GND
10	DB 4	9	GND
12	DB 5	11	GND
14	DB 6	13	GND
16	DB 7	15	GND
18	DB P	17	GND
20	GND	19	GND
22	GND	21	GND
24	GND	23	GND
26	TERMPWR	25	N.C.
28	GND	27	GND
30	GND	29	GND
32	ATN	31	GND
34	GND	33	GND
36	BSY	35	GND
38	ACK	37	GND
40	RST	39	GND
42	MSG	41	GND
44	SEL	43	GND
46	C/D	45	GND
48	REQ	47	GND
50	I/O	49	GND

Appendix J

The ISCSI-1BPS X2 Pin Assignment

Pin No.	Signal Mnemonic	Pin No.	Signal Mnemonic
2	N.C.	1	GND
4	N.C.	3	GND
6	Drive Select 0	5	GND
8	Index	7	GND
10	Drive Select 1	9	GND
12	Drive Select 2	11	GND
14	Drive Select 3	13	GND
16	Motor On	15	GND
18	Direction In	17	GND
20	Step	19	GND
22	Write Data	21	GND
24	Write Gate	23	GND
26	Track 000	25	GND
28	Write Protect	27	GND
20	Read Data	29	GND
32	Side Select	31	GND
34	N.C.	33	GND

This page was intentionally left blank

Appendix K

Glossary of VMEbus Terms (P1014)

A16

A type of module that provides or decodes an address on address line A01 through A15.

A24

A type of module that provides or decodes an address on address lines A01 through A23.

A32

A type of module that provides or decodes an address on address lines A01 through A31.

ARBITRATION

The process of assigning control of the DTB to a REQUESTER.

ADDRESS-ONLY CYCLE

A DTB cycle that consists of an address broadcast, but no data transfer. SLAVES do not acknowledge ADDRESS-ONLY cycles and MASTERS terminate the cycle without waiting for an acknowledgment.

ARBITER

A functional module that accepts bus requests from REQUESTER modules and grants control of the DTB to one REQUESTER at a time.

ARBITRATION BUS

One of the four buses provided by the P1014 backplane This bus allows an ARBITER module and several REQUESTER modules to coordinate use of the DTB.

ARBITRATION CYCLE

An ARBITRATION CYCLE begins when the ARBITER senses a bus request. The ARBITER grants the bus to a REQUESTER, which signals that the DTB is busy. The REQUESTER terminates the cycle by taking away the bus busy signal which causes the ARBITER to sample the bus requests again.

BACKPLANE (P1014)

A printed circuit (PC) board with 96 pin connectors and signal paths that bus the connector pins. Some P1014 systems have a single PC board, called the J1 backplane. It provides the signal paths needed for basic operation. Other P1014 systems also have an optional second PC board called a J2 backplane. It provides the additional 96 pin connectors and signal paths needed for wider data and address transfers. Still others have a single PC board that provides the signal conductors and connectors of both the J1 and J2 backplanes.

Appendix K

BACKPLANE INTERFACE LOGIC
Special interface logic that takes into account the characteristics of the backplane: its signal line impedance, propagation time, termination values, etc. The P1014 specification prescribes certain rules for the design of this logic based on the maximum length of the backplane and its maximum number of board slots.

BLOCK READ CYCLE
A DTB is cycle used to transfer a block of 1 to 256 bytes from a SLAVE to a MASTER. This transfer is done using a string of 1, 2, or 4 byte data transfers. Once the block transfer is started, the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of read cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the data for the next cycle is retrieved from the next higher location.

BLOCK WRITE CYCLE
A DTB cycle used to transfer a block of 1 to 256 bytes from a MASTER to a SLAVE. The block write cycle is very similar to the block read cycle. It uses a string of 1, 2, or 4 byte data transfers and the MASTER does not release the DTB until all of the bytes have been transferred. It differs from a string of write cycles in that the MASTER broadcasts only one address and address modifier (at the beginning of the cycle). Then the SLAVE increments this address on each transfer so that the next transfer is stored on the next higher location.

BOARD
A printed circuit (PC) board, its collection or electronic components, and either one or two 96 pin connectors that can be plugged into P1014 backplane connectors.

BUS TIMER
A functional module that measures how long each data transfer takes on the DTB and terminates the DTB cycle if a transfer takes too long. If the MASTER tries to transfer data to or from a non-existent SLAVE location it might wait forever. The BUS TIMER prevents this by terminating the cycle.

D08(O)
A SLAVE that sends and receives data 8 bits at a time over D00-D07,

or

an INTERRUPT HANDLER that receives 8 bit STATUS/IDs over D00-D07,

or

an INTERRUPTER that sends 8 bit STATUS/IDs over D00-D07.

D08(EO)

A MASTER that sends or receives data 8 bits at a time over either D00-D07 or D08-D15,

or

A SLAVE that sends and receives data 8 bits at a time over either D00-D07 or D08-D15,

or

an INTERRUPT HANDLER that receives 8 bit STATUS/IDs over D00-D07,

or

an INTERRUPTER that sends 8 bit STATUS/IDs over D00-D07.

D16

A MASTER that sends and receives data 16 bits at a time over D00-D15,

or

A SLAVE that sends and receives data 16 bits at a time over D00-D15,

or

an INTERRUPT HANDLER that receives 16 bit STATUS/IDs over D00-D15,

or

an INTERRUPTER that sends 16 bit STATUS/IDs over D00-D15.

D32

A MASTER that sends and receives data 32 bits at a time over D00-D31,

or

A SLAVE that sends and receives data 32 bits at a time over D00-D31,

or

an INTERRUPT HANDLER that receives 32 bit STATUS/IDs over D00-D31,

or

an INTERRUPTER that sends 32 bit STATUS/IDs over D00-D31.

DAISY-CHAIN
A special type of P1014 signal line that is used to propagate a signal level from board to board, starting with the first slot and ending with the last slot. There are four bus grant daisy-chains and one interrupt acknowledge daisy-chain on the P1014.

DATA TRANSFER BUS
One of the four buses provided by the P1014 backplane. The DATA TRANSFER BUS allows MASTERS to direct the transfer of binary data between themselves and SLAVES. (DATA TRANSFER BUS is often abbreviated to DTB).

DATA TRANSFER BUS CYCLE
A sequence of level transitions on the signal lines of the DTB that result in the transfer of an address or an address and data between a MASTER and a SLAVE. There are seven types of data transfer bus cycles.

DTB
An acronym for DATA TRANSFER BUS.

FUNCTIONAL MODULE
A collection of electronic circuitry that resides on one P1014 board and works together to accomplish a task.

IACK DAISY-CHAIN DRIVER
A functional module which activates the interrupt acknowledge daisy-chain whenever an INTERRUPT HANDLER acknowledges an interrupt request. This daisy-chain ensures that only one INTERRUPTER will respond with its STATUS/ID when more than one has generated an interrupt request.

INTERRUPT ACKNOWLEDGE CYCLE
A DTB cycle, initiated by an INTERRUPT HANDLER that reads a "STATUS/ID" from an INTERRUPTER. An INTERRUPT HANDLER generates this cycle when it detects an interrupt request from an INTERRUPTER and it has control of the DTB.

INTERRUPT BUS
One of the four buses provided by the P1014 backplane. The INTERRUPT BUS allows INTERRUPTER modules to send interrupt requests to INTERRUPT HANDLER modules.

INTERRUPTER
A functional module that generates an interrupt request on the INTERRUPT BUS and then provides STATUS/ID information when the INTERRUPT HANDLER requests it.

INTERRUPT HANDLER
A functional module that detects interrupt requests generated by INTERRUPTERS and responds to those requests by asking for STATUS/ID information.

LOCATION MONITOR
A functional module that monitors data transfers over the DTB in order to detect accesses to the locations it has been assigned to watch. When an access occurs to one of these assigned locations, the LOCATION MONITOR generates an on-board signal.

MASTER
A functional module that initiates DTB cycles in order to transfer data between itself and a SLAVE module.

OBO
A SLAVE that sends and receives data 8 bits at a time over D00-D07.

POWER MONITOR MODULE
A functional module that monitors the status of the primary power source to the P1014 system and signals when that power has strayed outside the limits required for reliable system operation. Since most systems are powered by an AC source, the power monitor is typically designed to detect drop-out or brown-out conditions on AC lines.

READ CYCLE
A DTB cycle used to transfer 1, 2, or 4 bytes from a SLAVE to a MASTER. The cycle begins when the MASTER broadcasts and address and an address modifier. Each SLAVE captures this address and address modifier, and checks to see of it is to respond to the cycle. If so, it retrieves the data from its internal storage, places it on the data bus and acknowledges the transfer. Then the MASTER terminates the cycle.

READ-MODIFY-WRITE CYCLE
A DTB cycle that is used to both read from, and write to, a SLAVE location without permitting any other MASTER to access that location. This cycle is most useful in multiprocessing systems where certain memory locations are used to control access to certain systems resources. (For example, semaphore locations.)

REQUESTER
A functional module that resides on the same board as a MASTER or INTERRUPT HANDLER and requests use of the DTB whenever its MASTER or INTERRUPT HANDLER needs it.

SERIAL CLOCK DRIVER
A functional module that provides a periodic timing signal that synchronizes operation of the VMSbus. (Although the P1014 specification defines a SERIAL CLOCK DRIVER for use with the VMSbus, and although it reserves two backplane signal lines for use by that bus, the VMSbus protocol is completely independent of the P1014.)

SLAVE

A functional module that detects DTB cycles initiated by a MASTER and, when those cycles specify its participation, transfers data between itself and the MASTER.

SLOT

A position where a board can be inserted into a P1014 backplane. If the P1014 system has both a J1 and a J2 backplane (or a combination J1/J2 backplane) each slot provides a pair of 96 pin connectors. If the system has only a J1 backplane, then each slot provides a single 96 pin connector.

SUBRACK

A rigid framework that provides mechanical support for boards inserted into the backplane, ensuring that the connectors mate properly and that adjacent boards do not contact each other. It also guides the cooling airflow through the system, and ensures that inserted boards do not disengage themselves from the backplane due to vibration or shock.

SYSTEM CLOCK DRIVER

A functional module that provides a 16 MHz timing signal on the UTILITY BUS.

SYSTEM CONTROLLER BOARD

A board which resides in slot 1 of a P1014 backplane and has a SYSTEM CLOCK DRIVER, a DTB ARBITER, an IACK DAISY CHAIN DRIVER, and a BUS TIMER. Some also have a SERIAL CLOCK DRIVER, a POWER MONITOR or both.

UAT

A MASTER that sends or receives data in an unaligned fashion,

or

a SLAVE that sends and receives data in an unaligned fashion.

UTILITY BUS

One of the four buses provided by the P1014 backplane. This bus includes signals that provide periodic timing and coordinate the power-up and power-down of P1014 systems.

WRITE CYCLE

A DTB cycle used to transfer 1, 2, or 4 bytes from a MASTER to a SLAVE. The cycle begins when the MASTER broadcasts an address and address modifier and places data on the DTB. Each SLAVE captures this address and address modifier, and checks to see if it is to respond to the cycle. If so, it stores the data and then acknowledges the transfer. The MASTER then terminates the cycle.

Appendix L

Literature References

Please refer to the following books for further detailed information:

VMEbus Specification Manual (IEEE P1014/D1.2) PRINTEX

Small Computer System Interface Manual (X3TY.2 Rev 17.B)

This page was intentionally left blank

Appendix M

Product Error Report

DEAR CUSTOMER,

WHILE FORCE COMPUTERS HAS ACHIEVED A VERY HIGH STANDARD OF QUALITY IN OUR PRODUCTS AND DOCUMENTATION, WE CONTINUALLY SEEK SUGGESTIONS FOR IMPROVEMENTS.

WE WOULD APPRECIATE ANY FEEDBACK YOU CARE TO OFFER.

PLEASE USE ATTACHED "PRODUCT ERROR REPORT" FORM FOR YOUR COMMENTS AND RETURN IT TO ONE OF OUR FORCE COMPUTERS OFFICES.

SINCERELY

FORCE COMPUTERS

This page was intentionally left blank

PRODUCT ERROR REPORT

HARDWARE/SOFTWARE/SYSTEMS

PRODUCT

SERIAL NO.

DATE OF PURCHASE

ORIGINATOR

COMPANY

ADDRESS

DATE

TELEPHONE

CONTACT EXT:

THIS BOX TO BE COMPLETED BY FORCE:

DATE:

PR#:

ACTION BY:

- [ ] ENGINEERING
- [ ] MARKETING
- [ ] PRODUCTION

AFFECTED PRODUCT: [ ] SOFTWARE [ ] HARDWARE [ ] SYSTEM

AFFECTED DOCUMENTATION: [ ] SOFTWARE [ ] HARDWARE [ ] SYSTEM

ERROR DESCRIPTION:

PLEASE SEND THIS PRODUCT ERROR REPORT TO ONE OF THE NEAREST FORCE COMPUTERS LOCATIONS:

FORCE COMPUTERS Inc.
3165 Winchester Blvd.
Campbell, CA 95008 6557
U S A

FORCE COMPUTERS GmbH
Prof. Messerschmitt - Str. 1
D 8014 Neubiberg/Munich
West Germany

FORCE COMPUTERS FRANCE Sarl
11, rue Casteja
92100 Boulogne
France

FORCE COMPUTERS UK Ltd.
No. 1 Holly Court
3 Tring Road
Wendover
Buckinghamshire HP22 6PE
England

PRODUCT ERROR REPORT

HARDWARE/SOFTWARE/SYSTEMS

PRODUCT _____

SERIAL NO. _____

DATE OF PURCHASE _____

ORIGINATOR _____

COMPANY _____

ADDRESS _____

_____

_____

_____

DATE _____

TELEPHONE _____

CONTACT _____ EXT: _____

THIS BOX TO BE COMPLETED BY FORCE:

DATE: _____

PR#: _____

ACTION BY:

- [ ] ENGINEERING
- [ ] MARKETING
- [ ] PRODUCTION

AFFECTED PRODUCT: [ ] SOFTWARE [ ] HARDWARE [ ] SYSTEM

AFFECTED DOCUMENTATION: [ ] SOFTWARE [ ] HARDWARE [ ] SYSTEM

ERROR DESCRIPTION:

_____

_____

_____

_____

_____

_____

PLEASE SEND THIS PRODUCT ERROR REPORT TO ONE OF THE NEAREST FORCE COMPUTERS LOCATIONS:

FORCE COMPUTERS Inc.
3165 Winchester Blvd.
Campbell, CA 95008 – 6557
U. S. A.

FORCE COMPUTERS GmbH
Prof. Messerschmitt Str. 1
D 8014 Neubiberg/Munich
West Germany

FORCE COMPUTERS FRANCE Sarl
11, rue Casteja
92100 Boulogne
France

FORCE COMPUTERS UK Ltd.
No. 1 Holly Court
3 Tring Road
Wendover
Buckinghamshire HP22 6PE
England

COPIES OF DATA SHEETS

This page was intentionally left blank

1. BIM 68153

2. PI/T 68230

3. SCSIBC 5386S

4. SCSIBT/R 8310

5. DMAC 68450

6. FDC 1772

This page was intentionally left blank

MOTOROLA

SEMICONDUCTORS

MC68153

Advance Information

TTL

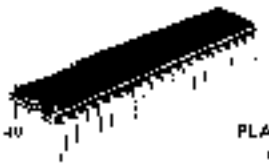
BUS INTERRUPTER MODULE

ADVANCED LOW POWER SCHOTTKY

BUS INTERRUPTER MODULE

The bipolar LSI MC68153 Bus Interrupter interfaces a microcomputer system bus to multiple slave devices requiring interrupt capabilities. It handles up to 4 independent sources of interrupt requests and is fully programmable.

- VERSAbus/VMEbus Compatible
- MC68000 Compatible
- Handles 4 Independent Interrupt Sources
- 6 Programmable Read/Write Registers
- Programmable Interrupt Request Levels
- Programmable Interrupt Vectors
- Supports Interrupt Acknowledge Daisy Chain
- Control Registers Contain Flag Bits
- Single +5.0 Volt Supply
- Total Power Dissipation = 1.5 W Typical
- Temperature Range of 0°C to 70°C
- Chip Access Time = 200 ns Typical with 16 MHz Clock
- 40-Pin Dual-In-Line Package

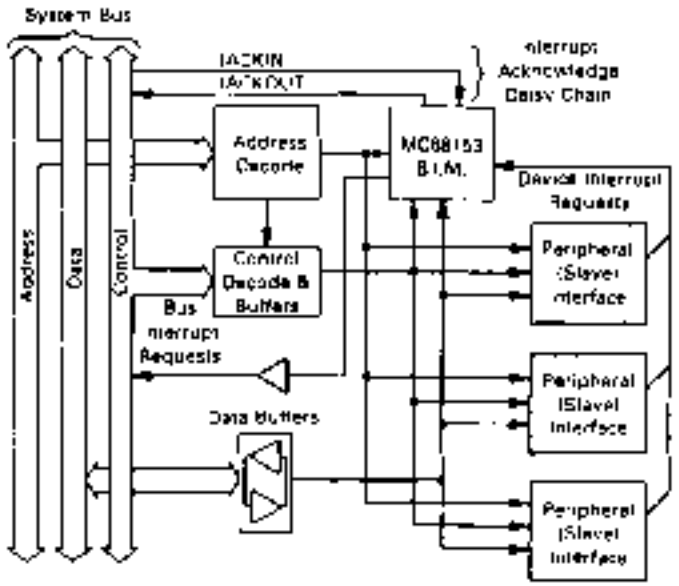


P SUFFIX
PLASTIC PACKAGE
CASE 711-03



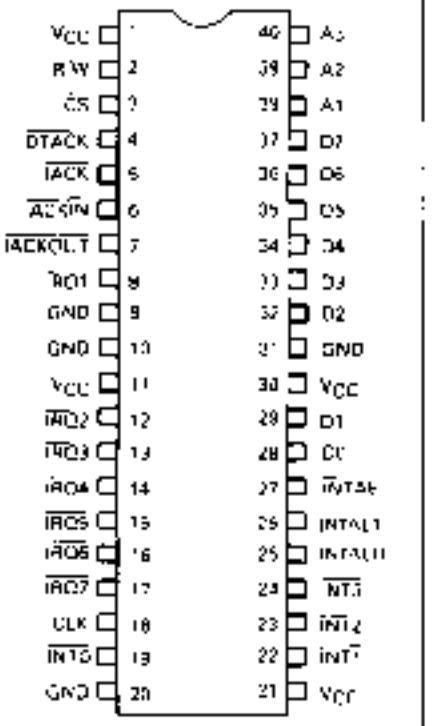
L SUFFIX
CERAMIC PACKAGE
CASE 734-04

FIGURE 1 – MC68153 SYSTEM BLOCK DIAGRAM



VERSAbus is a trademark of Motorola.

PIN ASSIGNMENTS



ADI 1057
(Replacing MBAE)

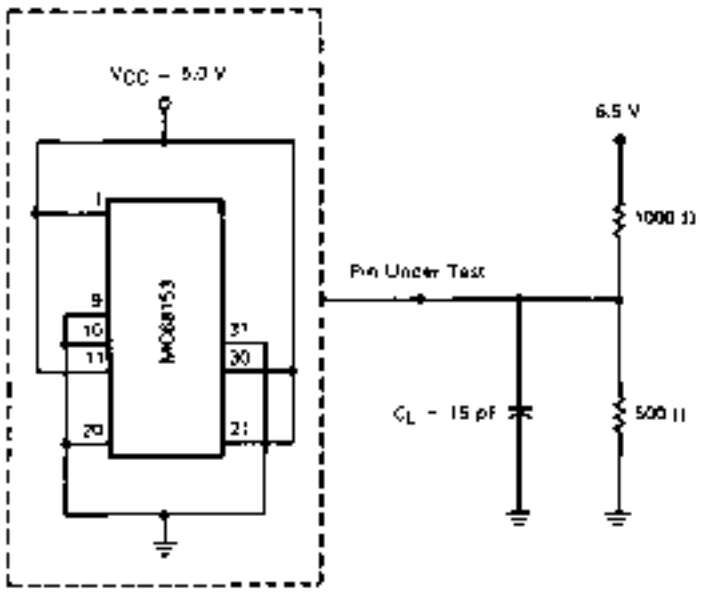
ABSOLUTE MAXIMUM RATINGS (Beyond which useful life may be impaired.)

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.5 to +7.0	V
Input Voltage	V_{in}	−0.5 to +7.0	V
Input Current	I_{in}	−30 to +5.0	mA
Output Voltage	V_{out}	−0.5 to +5.5	V
Output Current	I_{OK}	Twice Rated I_{OL}	mA
Storage Temperature	T_{stg}	−65 to +150	°C
Junction Operating Temperature	T_J	55 to +175	°C

DC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Symbol	Min	Max	Unit	Test Conditions
High Level Input Voltage	V_{IH}	2.0	—	V	
Low Level Input Voltage	V_{IL}	—	0.8	V	
Input Clamp Voltage	V_{IK}		1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
High Level Output Voltage[1]	V_{OH}	2.7	—	V	V_{CC} = MIN, I_{OH} = −800 µA
Low Level Output Voltage	V_{OL}	—	0.4	V	V_{CC} = MIN, I_{OL} = 8.0 mA
Output Short Circuit Current[2]	I_{OS}	15	130	mA	V_{CC} = MAX, V_{OUT} = 0 V
High Level Input Current	I_{IH}	-	20	µA	V_{CC} = MAX, V_{IN} = 2.7 V
Low Level Input Current	I_{IL}	—	−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
Supply Current	I_{CC}	225	365	mA	V_{CC} = MAX
Output Off Current (High)	I_{OZH}	-	20	µA	V_{CC} = MAX, V_{OUT} = 2.4 V
Output Off Current (Low)	I_{OZL}	-	−20	µA	V_{CC} = MAX, V_{OUT} = 0.4 V

AC TEST CIRCUIT — AC Testing of All Outputs



NOTES:
1. Not applicable to open collector outputs.
2. Not more than one output should be shorted at a time for longer than one second.
3. $\overline{CS}$ Low to CLK High (Setup Time) of 15 ns Min must be observed.
4. $\overline{WCR}$ Low to CLK High and $\overline{WCKIN}$ Low to CLK High (Setup Time) of 15 ns Min must be observed.
5. See Table I for additional performance guidelines.

AC ELECTRICAL SPECIFICATIONS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C)

Parameter	Test Number[5]	Max (ns)
CLK High to Data Out Valid (Delay)[3]	1	55
CLK High to $\overline{DTACK}$ Low (Delay)[3]	2	40
$\overline{CS}$ High to $\overline{DTACK}$ High (Delay)	3	35
CLK High to Data Out Valid (Delay)[4]	4	55
CLK High to $\overline{INTAE}$ Low (Delay)[4]	5	40
IACK High to Data Out High Impedance (Delay)	6	60
$\overline{IACK}$ High to $\overline{DTACK}$ High (Delay)	7	45
$\overline{CS}$ High to Data Out High (Delay)	8	45
$\overline{CS}$ High to $\overline{IEO}$ High (Delay)	9	60
IACK High to $\overline{INTAE}$ High (Delay)	10	35

GENERAL DESCRIPTION

The MC68153 Bus Interrupter Module (BIM) is designed to serve as an interrupt requester for peripheral devices in a microcomputer system. Up to 8 independent devices can be interfaced to the system bus by the MC68153. Intended for asynchronous master/slave bus operation, the BIM is compatible with VERSAbus, VMEbus, MC68000 device bus, and other system buses. Figure 1 shows a block diagram of a typical configuration. In this example, three peripheral devices (bus slaves) are connected to the system data bus. Each of these devices could be parallel I/O, serial I/O, or some other function. An interrupt request from any device is routed to the MC68153, and the BIM handles all interface to the system bus. It generates a bus interrupt request as a result of the device interrupt request. When the system interrupt handler or processor responds with an interrupt acknowledge cycle, the MC68153 can answer supplying an interrupt vector and handling all timing.

The functional block diagram of the MC68153 is shown in Figure 2. The device contains circuitry to accept four separate interrupt sources ($\overline{INT0}$ – $\overline{INT3}$). Interface to the system bus includes generation of bus interrupt requests ($\overline{IRQ1}$ – $\overline{IRQ7}$), response to a bus interrupt acknowledge cycle (either supplying a vector or passing on a daisy chain signal), and releasing the bus interrupt request signal at the proper time. The BIM has flexibility provided by eight programmable read/write registers. Four 8-bit vector registers (VR0 – VR3) contain status/address information and supply a byte vector in response to an interrupt acknowledge cycle for the corresponding interrupt source. Four other 8-bit control registers (CR0 – CR3) contain information that oversees operation of the interrupt circuitry. The control information is programmable and includes interrupt request level and interrupt enable and disable. Also contained in the control registers are flag bits. These flags are useful for task coordination, resource management, and interprocessor communication.

SIGNAL DESCRIPTION

Throughout the data sheet, signals are presented using the terms asserted and negated independent of whether the signal is asserted in the high voltage or low voltage state. Active low signals are denoted by a superscript bar.

BIDIRECTIONAL DATA BUS — D0 – D7

Pins D0 – D7 form an 8-bit bidirectional data bus to/from the system bus. These are active high, 3-state pins.

ADDRESS INPUTS — A1 – A3

These active high inputs serve two functions. One function is to select one of the eight possible registers during a read or write cycle. Secondly, during an interrupt acknowledge A1 – A3 show the level of interrupt being acknowledged, and the BIM uses these to determine if a match exists with an internal level.

CHIP SELECT — $\overline{CS}$

$\overline{CS}$ is an active low input used to select the BIM's registers for the current bus cycle. Address strobe, data strobe, and appropriate address bits must be included in the chip select equation.

READ/WRITE — R/$\overline{W}$

The R/$\overline{W}$ input is a signal from the system bus used to determine if the current bus cycle is a read (high) or write (low).

DATA TRANSFER ACKNOWLEDGE — $\overline{DTACK}$

DTACK is an open-collector, active low output that signals the completion of a read, write, or interrupt acknowledge cycle. During read or interrupt acknowledge cycles, $\overline{DTACK}$ is asserted by the MC68153 after data has been provided on the data bus; during write cycles it is asserted after data has been accepted from the data bus. A pullup resistor is required to maintain $\overline{DTACK}$ high between bus cycles.

FIGURE 2 — MC68153 FUNCTIONAL BLOCK DIAGRAM

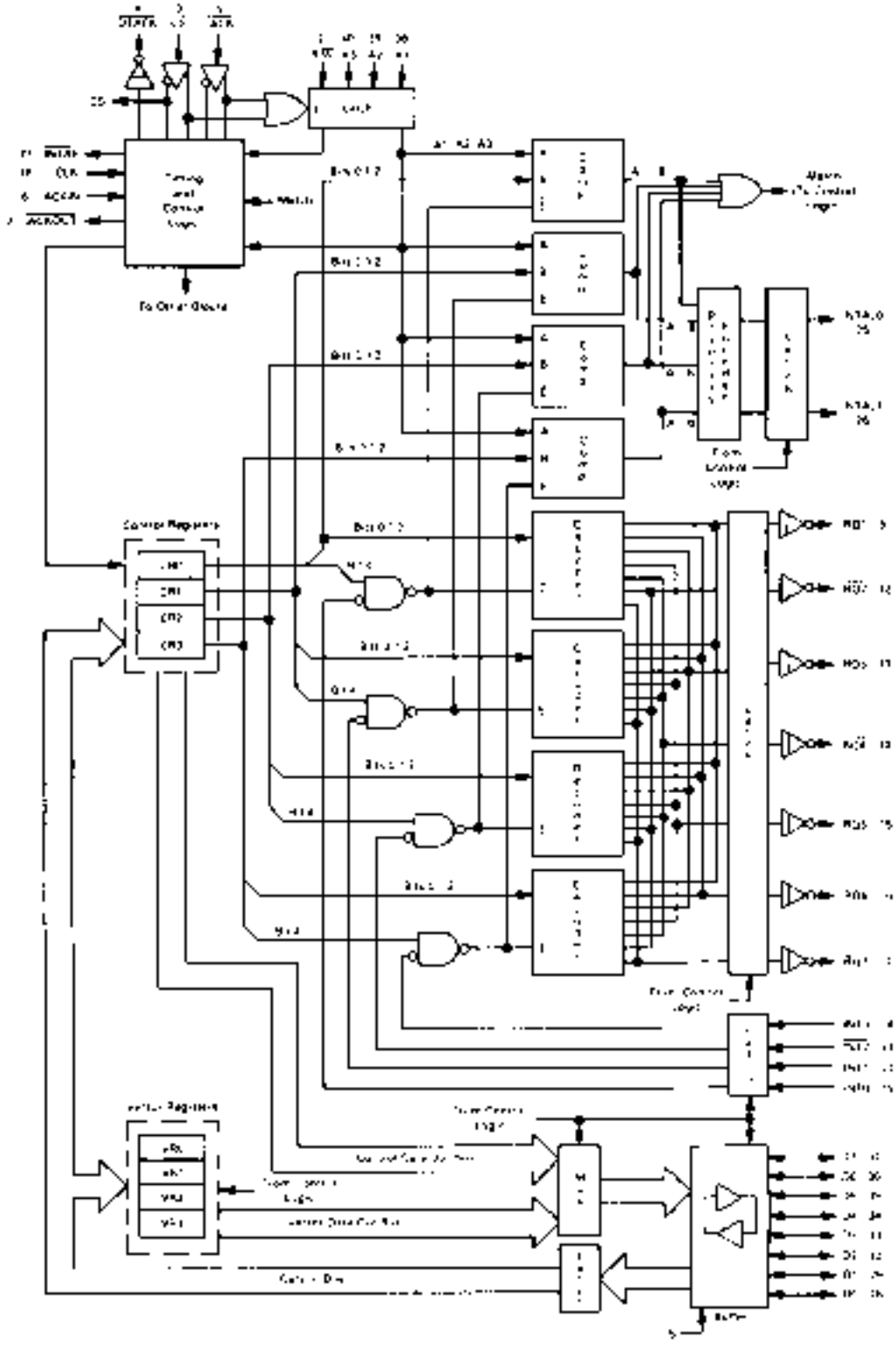
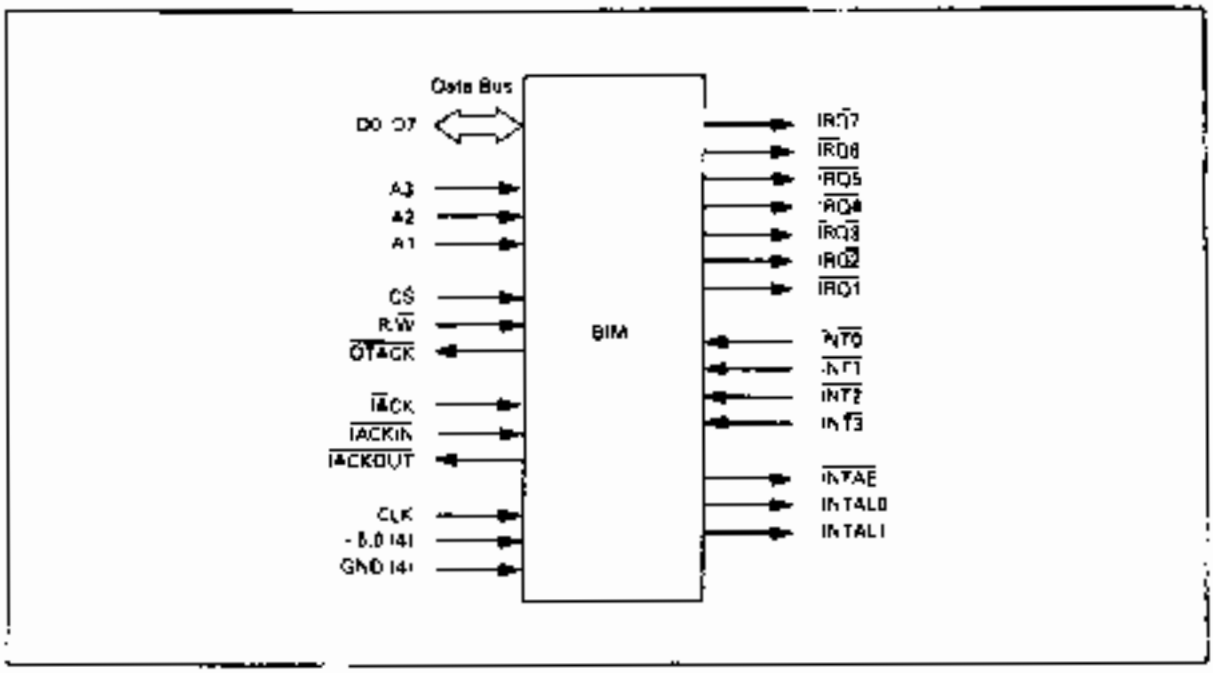


FIGURE 3 — LOGICAL PIN ASSIGNMENT



INTERRUPT ACKNOWLEDGE SIGNALS — IACK, IACKIN, IACKOUT

These three pins support the interrupt acknowledge cycle. A low level on the IACK input indicates an interrupt acknowledge cycle has been initiated. This signal is conditioned externally with Address Strobe and the lower data strobe of an MC68000 type bus. After IACK is asserted the BIM compares the interrupt level presented on address lines A1, A2, and A3 with the current levels generated internally and determines if a match exists. Then, if input IACKIN is asserted (driven low), the BIM will either complete the interrupt acknowledge cycle if a match exists or assert output IACKOUT if no match exists.

IACKIN and IACKOUT form part of a prioritized interrupt acknowledge daisy chain. The daisy chain prioritizes interrupters and guarantees that two or more devices requesting an interrupt on the same level will not respond to the same cycle. The requesting device (or interrupter) must wait until IACKIN is asserted and not pass the signal on (assert IACKOUT) if it is to complete the interrupt acknowledge cycle.

BUS INTERRUPT REQUEST SIGNALS — IRQ1 – IRQ7

These open-collector outputs are low when asserted, indicating a bus interrupt is requested at the corresponding level. An open-collector buffer is normally required for sufficient drive when interfacing to a system bus. A pullup resistor is required to maintain IRQ1 – IRQ7 high between interrupt requests.

DEVICE INTERRUPT REQUEST SIGNALS — INT0 – INT3

INT0 – INT3 are active low inputs used to indicate to the BIM that a device wants a bus interrupt.

INTERRUPT ACKNOWLEDGE ENABLE — INTAE

During an interrupt acknowledge cycle, this output pin is asserted low to indicate that outputs INTAL0 and INTAL1 are valid. These two outputs contain an encoded number (x) corresponding to the interrupt (INTx) being acknowledged. This feature can be used to signal interrupting devices, which supply their own vector, when to respond to the interrupt acknowledge cycle with the vector and a DTACK signal.

INTERRUPT ACKNOWLEDGE LEVEL — INTAL0, INTAL1

These active high outputs contain an encoded number corresponding to the interrupt level being acknowledged. They are valid only when INTAE is asserted low.

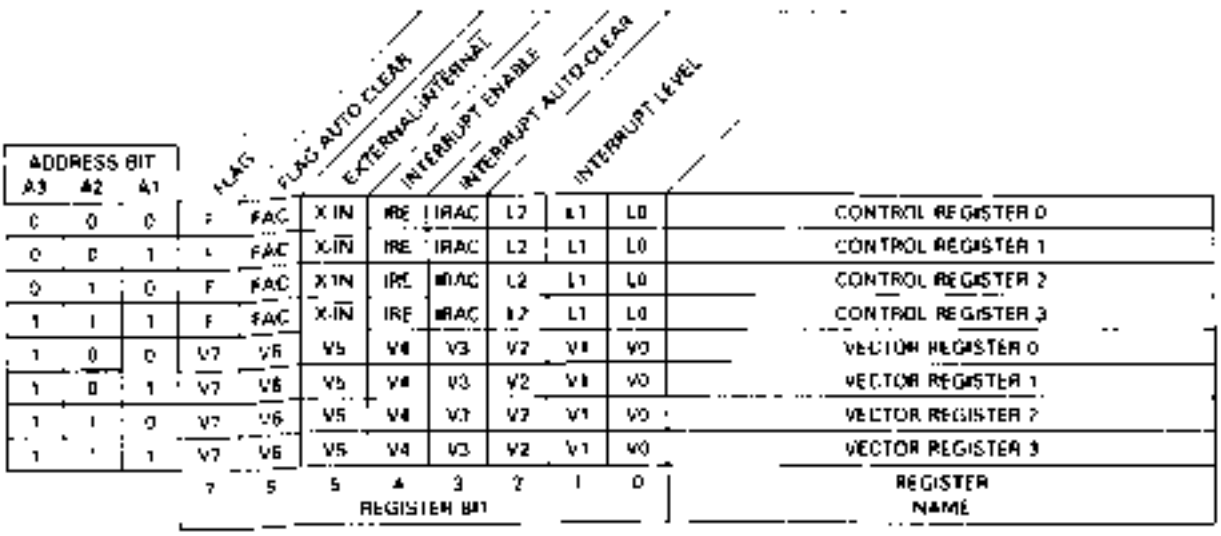
CLOCK — CLK

The CLK input is used to supply the clock for internal operations of the MC68153.

RESET — CS, IACK

Although a reset input is not supplied, an on-board reset is performed if CS and IACK are asserted simultaneously.

FIGURE 4 — MC68153 REGISTER MODEL



REGISTER DESCRIPTION

The MC68153 contains 8 programmable read/write registers. There are four control registers (CR0 – CR3) that govern operation of the device. The other four (VR0 – VR3) are vector registers that contain the vector data used during an interrupt acknowledge cycle. Figure 4 illustrates the device register model.

CONTROL REGISTERS

There is a control register for each interrupt source, i.e., CR0 controls $\overline{INT0}$, CR1 controls $\overline{INT1}$ etc. The control registers are divided into several fields:

1. Interrupt level (L2, L1, L0) — The least significant 3-bit field of the register determines the level at which an interrupt will be generated.

L2	L1	L0	IRQ LEVEL
0	0	0	DISABLED
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

   A value of zero in the field disables the interrupt.

2. Interrupt Enable (IRE) — This field (Bit 4) must be set (high level) to enable the bus interrupt request associated with the control register. Thus, if the $\overline{INTx}$ line is asserted and IRE is cleared, no interrupt request ($\overline{IROX}$) will be asserted.
3. Interrupt Auto-Clear (IRAC) — If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of clearing IRE disables the interrupt request. To re-enable the interrupt associated with this register, IRE must be set again by writing to the control register.
4. External/Internal (X/$\overline{IN}$) — Bit 5 of the control register determines the response of the MC68153 during an interrupt acknowledge cycle. If the X/$\overline{IN}$ bit is clear (low level) the BIM will respond with vector data and a $\overline{DTACK}$ signal, i.e., an internal response. If X/$\overline{IN}$ is set, the vector is not supplied and no $\overline{DTACK}$ is given by the BIM, i.e., an external device should respond.
5. Flag (FI) — Bit 7 is a flag that can be used in conjunction with the test and set instruction of the MC68000. It can be changed without affecting chip operation. It is useful for processor-to-processor communication and resource allocation.
6. Flag Auto-Clear (FAC) — If FAC (Bit 6) is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

VECTOR REGISTERS

Each interrupt input has its own associated vector register. Each register is 8 bits wide and supplies a data byte during its interrupt acknowledge cycle if the associated External/Internal (X/$\overline{IN}$) control register bit is clear. This data can be status identification, or address information depending on system usage. The information is programmed by the system user.

DEVICE RESET

When the MC68153 is reset, the registers are set to a known condition. The control registers are set to all zeros (low). The vector registers are set to $0F. This value is the MC68000 vector for an uninitialized interrupt vector.

FUNCTIONAL DESCRIPTION

SYSTEM OVERVIEW

The MC68153 is compatible with many system buses. however, it is primarily intended for VMEbus. VERSAbus and MC68000 applications. Figure 5 shows a system configuration similar to VMEbus. In the figure only one system Data Transfer Bus (DTB) master is used. The Priority Interrupt structure provides a means for peripheral slave devices to ask for an interrupt of other processor (DTB master) activity and receive service from the processor. The MC68153 BIM acts as an interface device requesting and responding to interrupt acknowledge cycles for up to 4 independent slaves.

In Figure 5, functional modules are identified as Interrupters and an Interrupt Handler. An Interrupter (such as the MC68153) receives slave requests for an interrupt and handles all interface to the system bus required to ask for and respond to interrupt requests. The Interrupt Handler receives the bus interrupt requests, determines when an interrupt acknowledge will occur and at which level, and finally either performs the interrupt acknowledge (IACK) cycle or tells the DTB master to execute the IACK cycle.

The signal lines in the Priority Interrupt structure include (* — indicates active low):

1. IRQ1*–IRQ7* — seven prioritized interrupt request lines.
2. IACK* — signal line that indicates an interrupt acknowledge cycle is occurring.
3. IACKIN*-IACKOUT* — two signals that form part of a daisy chain that prioritizes interrupters.

In addition Data Transfer Bus control signals are involved in the IACK bus cycle:

1. AS* — the Address Strobe asserted low indicates a valid address is on the bus.
2. DS0* — the lower Data Strobe asserted low indicates a data transfer will occur on bus bits D00–D07.
3. WRITE* — the Read/Write is negated indicating the data is to be read from the Interrupter.
4. A01–A03 — Address lines A01–A03 contain the encoded priority level of the IACK cycle.
5. D00–D07 — Data bus lines D00–D07 are used to pass the interrupt vector from the responding Interrupter to the Interrupt Handler.
6. DTACK* — Data Transfer Acknowledge asserted low signals that the Interrupter has put the vector on the data bus.

FIGURE 5 — SIMPLE VMEbus CONFIGURATION

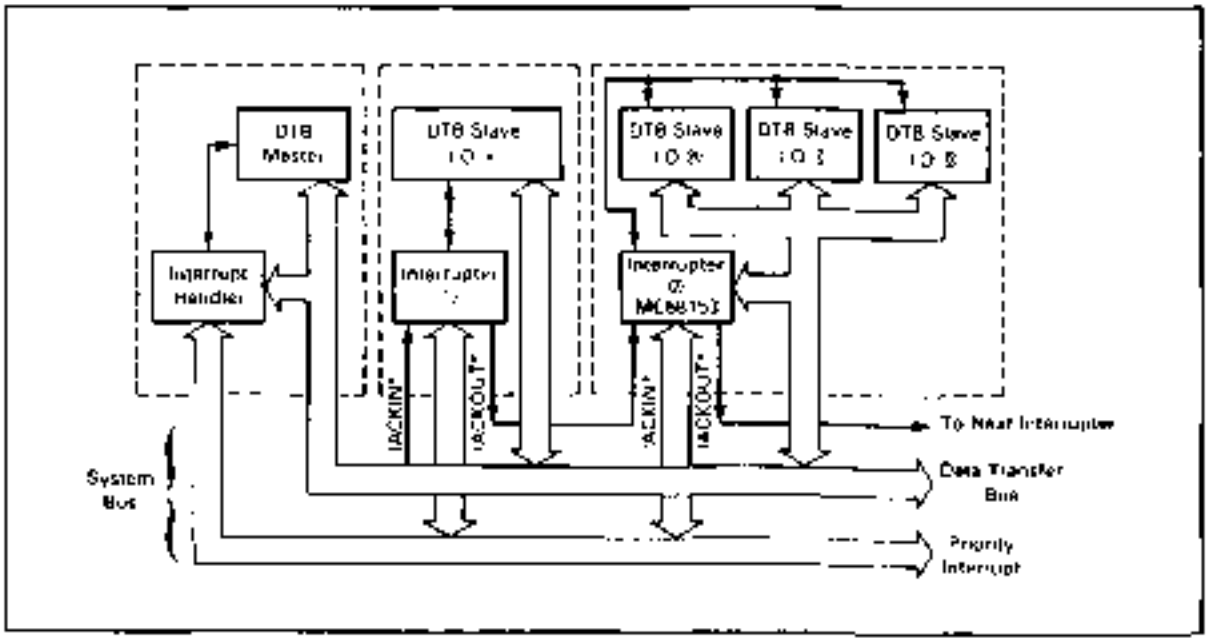
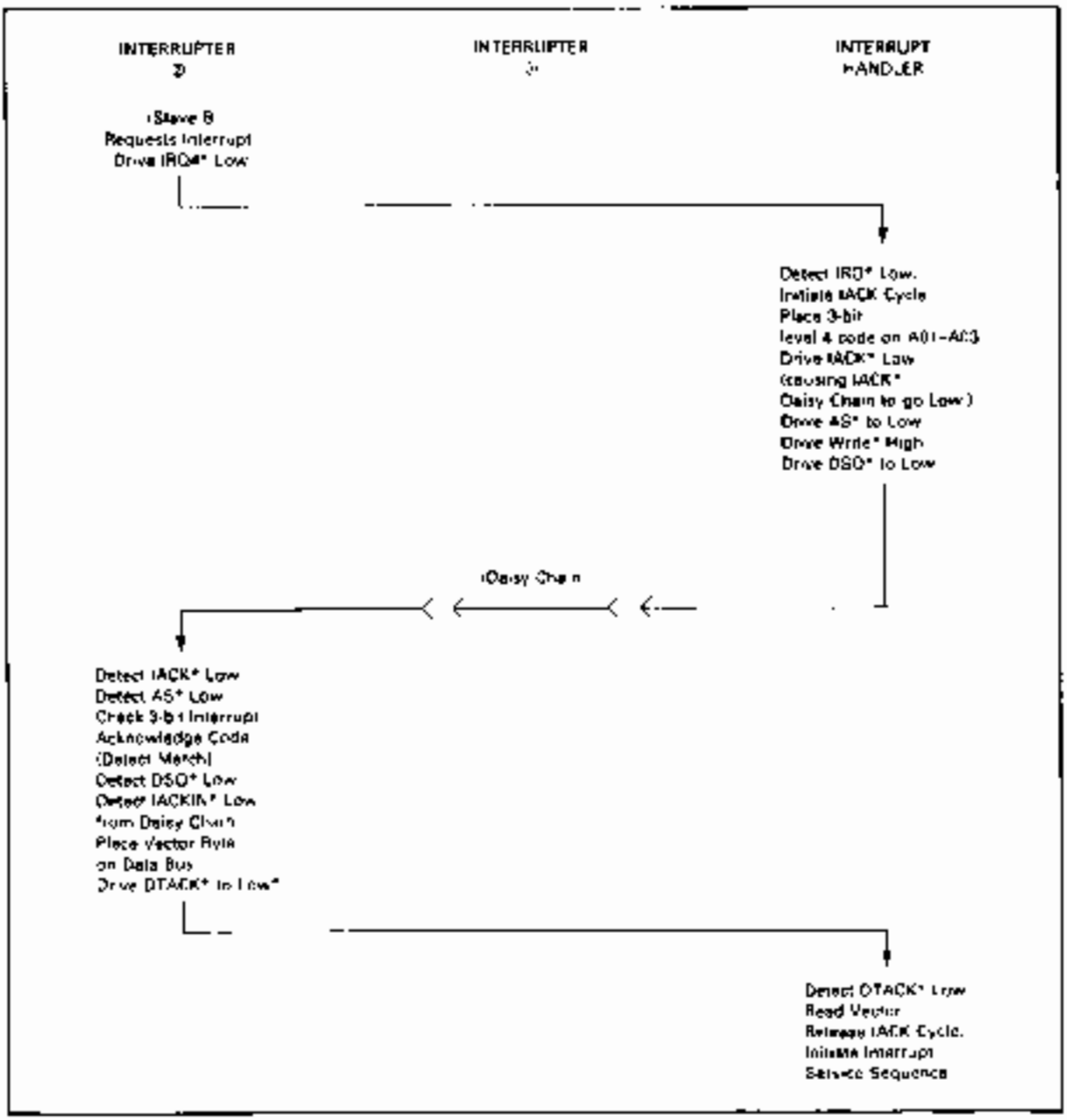


Figure 8 shows a flow diagram of a typical interrupt request and acknowledge operation. Briefly, the sequence of events is first, an Interrupter makes a request, next the Handler responds with an IACK cycle, then the Interrupter passes a vector to the Handler completing the IACK cycle, and finally the Handler uses the vector to determine additional action. Typically, an interrupt service routine is stored in software and the vector points to its starting address.

Note the daisy chain operation. If the IACK level (on A01–A03) does not match the Interrupter's request level or if no request is pending, the Interrupter passes the IACKIN* signal on and asserts IACKOUT*. This sequential action automatically prioritizes. Requesters on the same level (first one in line with a request pending gets serviced) and prevents two or more Interrupters from responding simultaneously.

FIGURE 8 — INTERRUPT REQUEST AND ACKNOWLEDGE OPERATION FLOW DIAGRAM



This discussion is a very cursory look at the bus operation. For more details including situations with multiple bus masters, the user is directed to the VMEbus Specification MVMEBS or VERSAbus Specification M88KVBS. Also, the MC68153 can be used with other buses having similar interrupt structures

BIM BUS INTERFACE

Figure 7 shows a simplified block diagram of the MC68153 interface to VERSAbus or VMEbus. Address Decode and Control Decode are dependent on the application and must be designed to guarantee BIM ac specifications. It is possible in most cases that the decode logic can be shared with the slave devices. Buffers are provided where shown to comply with bus loading and drive specifications. It is also possible that buffers can be shared with the slave bus interface.

READ/WRITE OPERATION

All eight BIM registers can be accessed from the system bus in both read and write modes. The BIM has an asynchronous bus interface, primarily designed for MC68000-like buses. The following signals generate read and write cycles. Chip Select ($\overline{CS}$), Read/Write (R/$\overline{W}$), Address Inputs (A1–A3), Data Bus (D0–D7), and Data Transfer Acknowledge ($\overline{DTACK}$). During read and write cycles the internal registers are selected by A1, A2, and A3 in compliance with the Figure 4 Truth Table

Figure 8 shows the device timing for a read cycle. R/$\overline{W}$ and A1–A3 are latched on the falling edge of $\overline{CS}$ and must meet specified setup and hold times. Chip access time for valid data and $\overline{DTACK}$ are dependent on the clock frequency as shown in the figure

Figure 9 shows the device timing for a write cycle. R/$\overline{W}$, A1–A3, and D0–D7 are latched on the falling edge of $\overline{CS}$ and must meet specified setup and hold times. Chip access time for $\overline{DTACK}$ is dependent on the clock frequency as shown in the figure

FIGURE 7 — VMEbus/VERSAbus INTERFACE BLOCK DIAGRAM

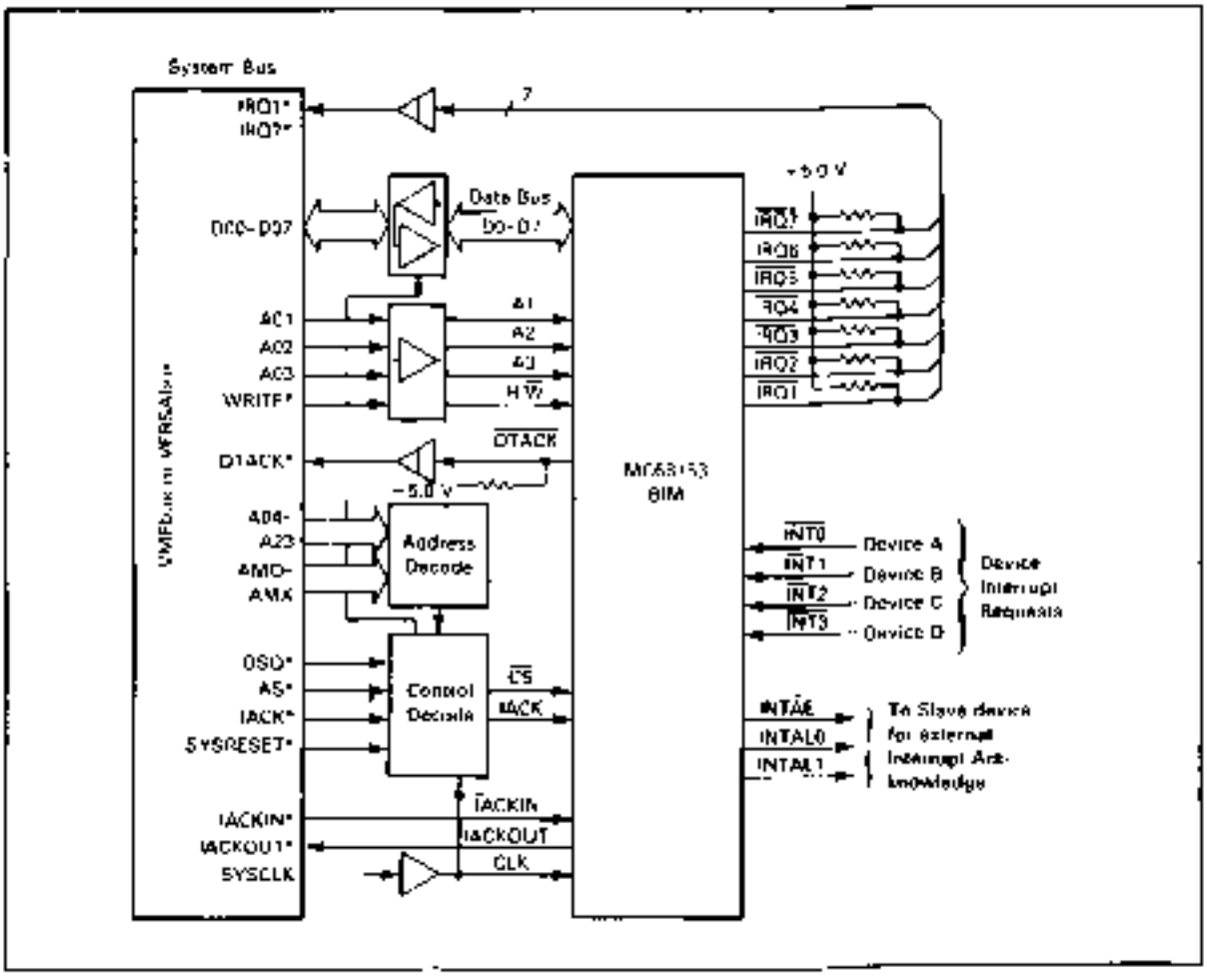


FIGURE 8 — READ CYCLE

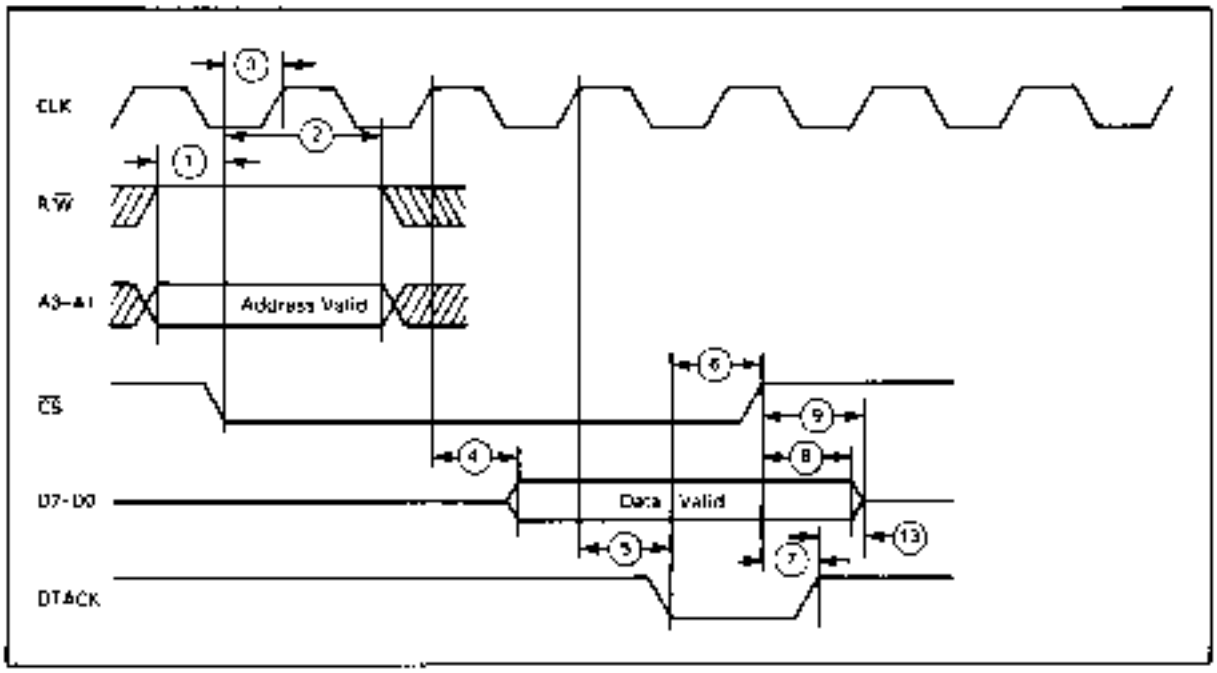
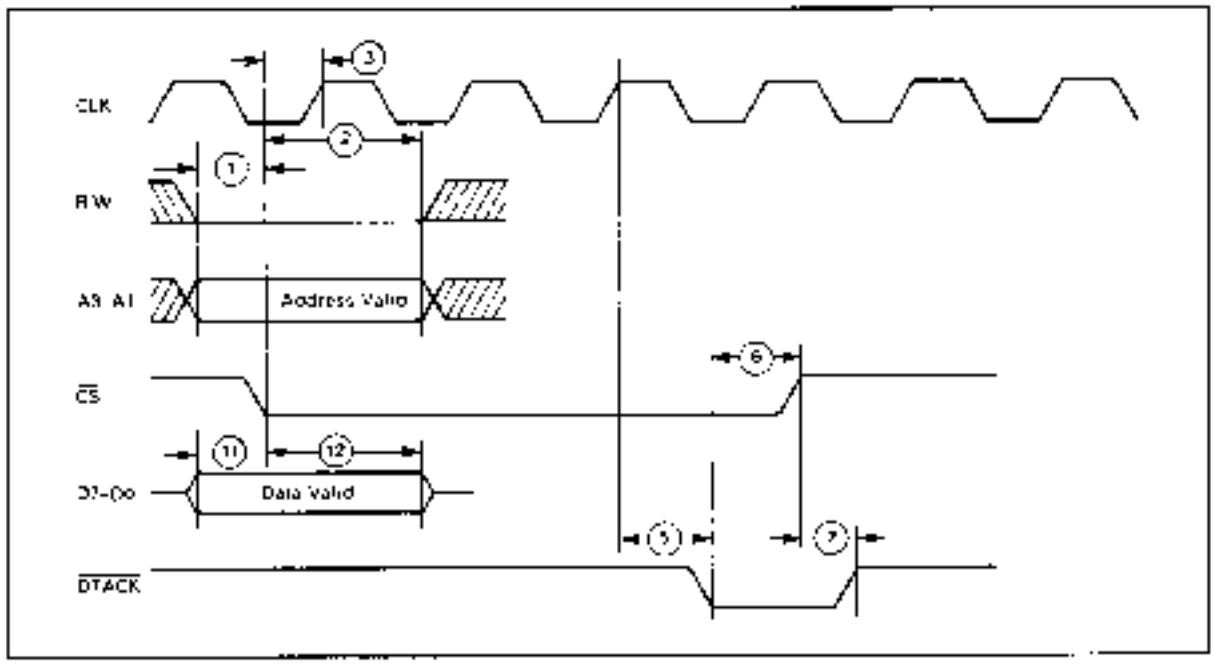


FIGURE 9 — WRITE CYCLE



INTERRUPT REQUESTS

The MC68153 accepts device interrupt requests on inputs INT0, INT1, INT2, and INT3. Each input is regulated by Bit 4 (IRE) of the associated control register (CR0 controls $\overline{INT0}$, CR1 controls $\overline{INT1}$, etc.). If IRE (Interrupt Enable) is set and a device input is asserted, an Interrupt Request open-collector output ($\overline{IRQ0}$–$\overline{IRQ7}$) is asserted. The asserted $\overline{IROX}$ output is selected by the value programmed in Bits 0, 1, and 2 of the control register (L0, L1, and L2). This 3-bit field determines the interrupt request level as set by software.

Two or more interrupt sources can be programmed to the same request level. That $\overline{IROX}$ output will remain asserted until multiple interrupt acknowledge cycles respond to all requests.

If the interrupt request level is set to zero, the interrupt is disabled because there is no corresponding IRQ output.

INTERRUPT ACKNOWLEDGE

The response of an Interrupt Handler to a bus interrupt request is an interrupt acknowledge cycle. The IACK cycle is initiated in the MC68153 by receiving $\overline{IACK}$ low. R/$\overline{W}$, A1, A2, A3 are latched, and the interrupt level on line A1–A3 is compared with any interrupt requests pending in the chip. Further activity can be one of four cases:

1. No further action required — This occurs if $\overline{IACKIN}$ is not asserted. Asserting $\overline{IACK}$ only starts the BIM activity. If the daisy chain signal never reaches the MC68153 ($\overline{IACKIN}$ is not asserted), another Interrupter has responded to the $\overline{IACK}$ cycle. The cycle will end, the chip $\overline{IACK}$ is negated, and no additional action is required.

2. Pass on the interrupt acknowledge daisy chain — For this case, $\overline{IACKIN}$ input is asserted by the preceding daisy chain Interrupter and $\overline{IACKOUT}$ output is in turn asserted. The daisy chain signal is passed on when no interrupts are pending on a matching level or when any possible interrupts are disabled. The Interrupt Enable (IRE) bit of a control register can disable any interrupt requests, and in turn, any possible matches.

3. Respond internally — For this case, $\overline{IACKIN}$ is asserted and a match is found. The MC68153 completes the IACK cycle by supplying an interrupt vector from the proper vector register followed by a DTACK signal asserted. $\overline{IACKOUT}$ is not asserted because the interrupt acknowledge cycle is completed by this device.

   For the MC68153 to respond in this mode of operation, the EXTERNAL/INTERNAL control register bit (X/$\overline{IN}$) must be zero. For each source of interrupt request, the associated control register determines the BIM response to an IACK cycle, and this X/$\overline{IN}$ bit sets this response either internally (X/$\overline{IN}$ = 0) or externally (X/IN = 1).

4. Respond externally — For the final case, $\overline{IACKIN}$ is also asserted, a match is found and the associated control register has X/$\overline{IN}$ bit set to one. The MC68153 does not assert $\overline{IACKOUT}$ and does assert INTAE low. $\overline{INTAE}$ signals that the requesting device must complete the IACK cycle (supplying a vector and $\overline{DTACK}$) and that the 2-bit code contained on outputs INTAL0 and INTAL1 shows which interrupt source is being acknowledged.

These cases are discussed in more detail in the following paragraphs.

Internal Interrupt Acknowledge

For an internal interrupt acknowledge to occur, the following conditions must be met.

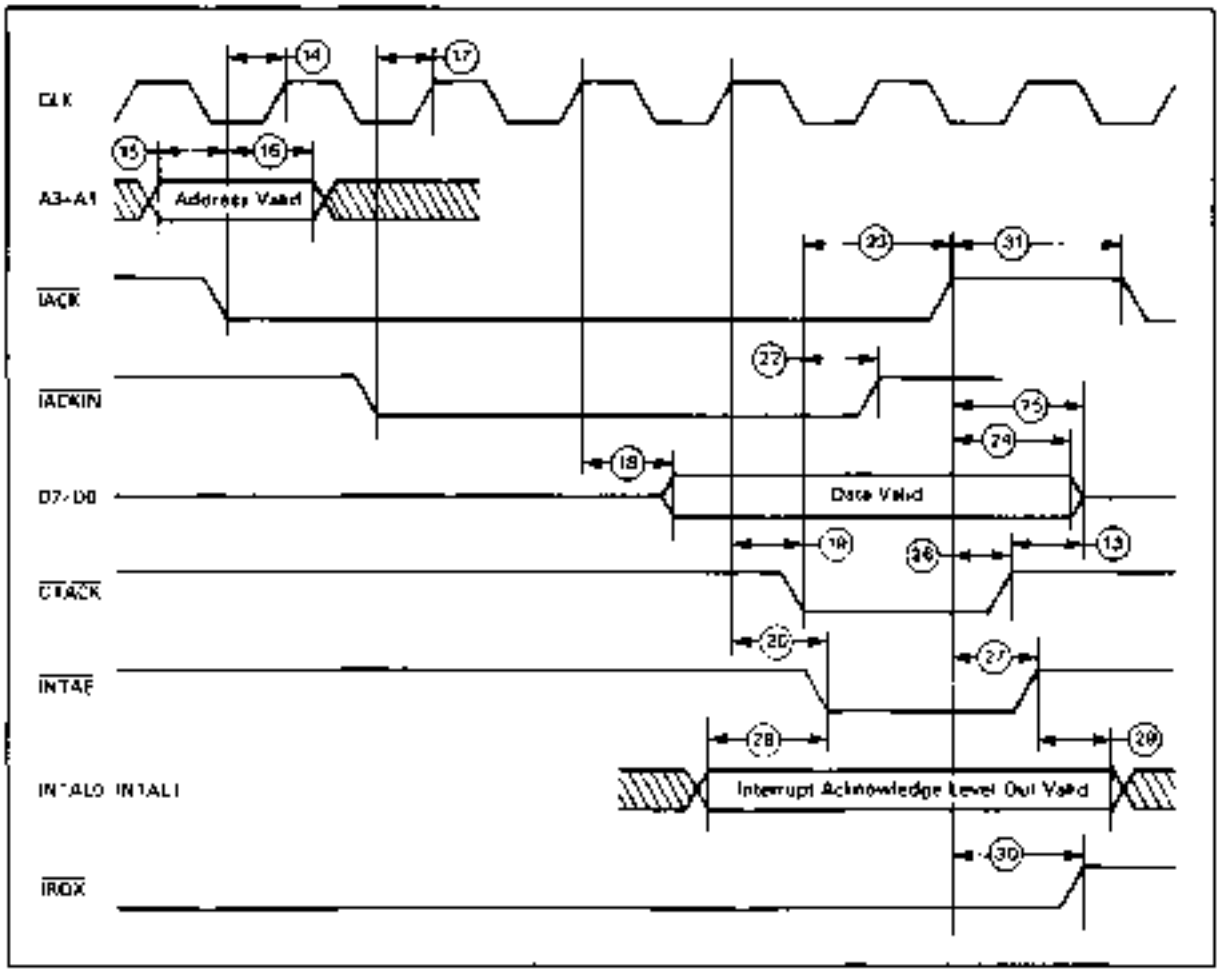
1. One or more device interrupt inputs ($\overline{INT0}$–$\overline{INT3}$) has been asserted and corresponding control bit IRE value is one.
2. IACK asserted.
3. A match exists between (A3, A2, A1) and the (L2, L1, L0) field of an enabled, requesting control register. If two or more devices are requesting at the same interrupt level, preference is given to the highest number requester, that is $\overline{INT3}$ has highest priority and $\overline{INT0}$ has lowest.
4. Control register bit X/$\overline{IN}$ of matching interrupt source must be zero.
5. $\overline{IACKIN}$ asserted.

The internal interrupt acknowledge cycle timing is shown in Figure 10. The 8-bit interrupt acknowledge vector is presented to the data bus and $\overline{DTACK}$ is asserted. Note also that INTAL0 and INTAL1 are valid and $\overline{INTAE}$ is asserted during this cycle although they would normally not be used. The cycle is terminated (data and DTACK released) after $\overline{IACK}$ is negated.

During the IACK cycle, the INTERRUPT AUTO-CLEAR control bit (IRAC) comes into play. If the IRAC = one for the responding interrupt source, the INTERRUPT ENABLE (IRE) bit is automatically cleared during the IACK cycle, thus disabling the associated interrupt input and any $\overline{IROX}$ output asserted due to this interrupt input. Before another interrupt can be requested from this source, IRE must be set to one by writing to the control register.

Note that $\overline{IACKOUT}$ is not asserted because this device is responding to the IACK and does not pass the daisy chain signal on. Also, new device interrupt requests occurring on $\overline{INT0}$–$\overline{INT3}$ after $\overline{IACK}$ is asserted are locked out to prevent any race conditions on the daisy chain.

FIGURE 10 — INTERRUPT ACKNOWLEDGE CYCLE — INTERNAL VECTOR



External Interrupt Acknowledge

For an external interrupt acknowledge, the same conditions as listed above are met with one exception. Control register bit XEN of matching interrupt source must be set to one. The timing is shown in Figure 11. For this cycle, the interrupt vector and DTACK must be supplied by an external device. INTAE is asserted indicating that INTAL0 and INTAL1 are valid. The external device can use these signals to enable the vector and DTACK. The cycle is terminated after IACK is negated.

The IRAC control block acts in the external interrupt acknowledge the same as described for the internal response (see above). Also, IACKOUT is not asserted and new device interrupts are disabled for reasons discussed above.

Pass On IACK Daisy Chain

If the MC68153 has no interrupt request pending at the same level as the interrupt acknowledge, the IACK daisy chain signal is passed on to the next device if IACKIN is asserted. The following conditions are thus met:

1. IACK asserted
2. No match exists between [A3, A2, A1] and the IL2, L1, L0; held at an enabled, requesting control register.
3. IACKIN is asserted.

IACKOUT is asserted if these conditions are valid. This output drives IACKIN of the next interrupter on the daisy chain, passing the signal along. Figure 12 shows the timing for this case. IACKOUT is negated after IACK is negated.

FIGURE 11 — INTERRUPT ACKNOWLEDGE CYCLE — EXTERNAL VECTOR

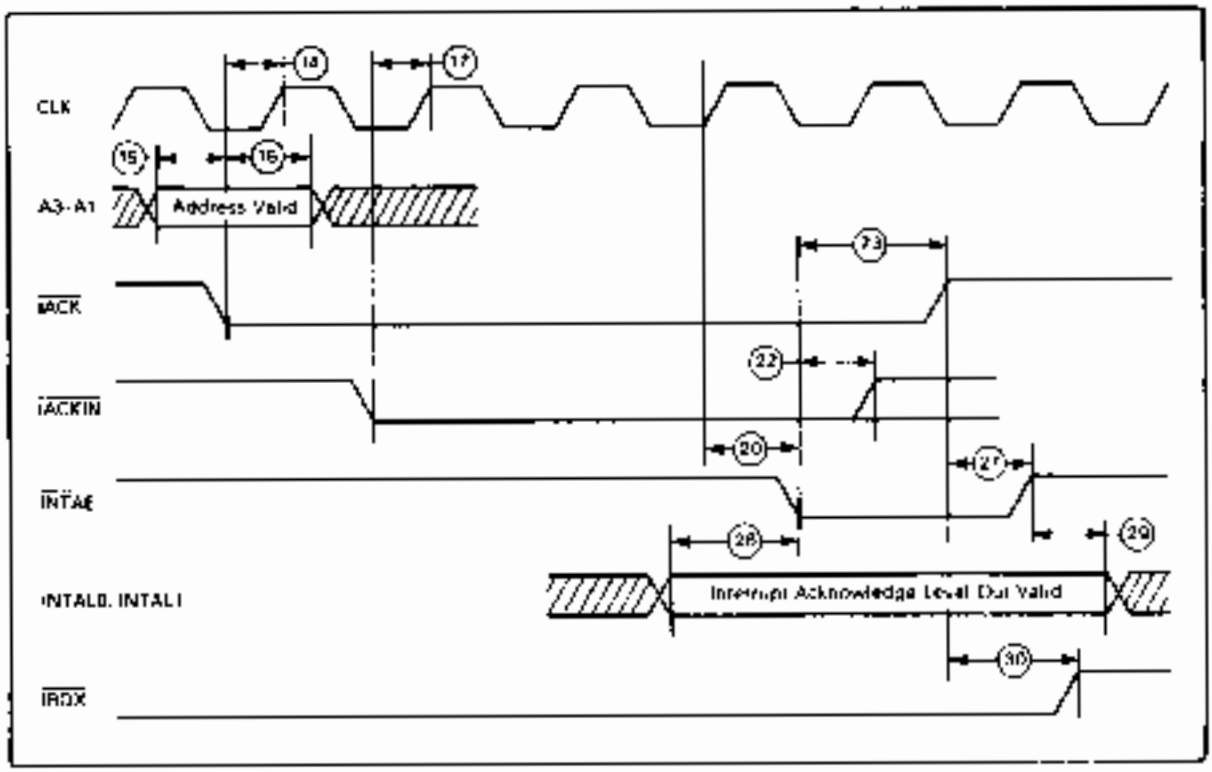


FIGURE 12 - INTERRUPT ACKNOWLEDGE CYCLE - IACKOUT

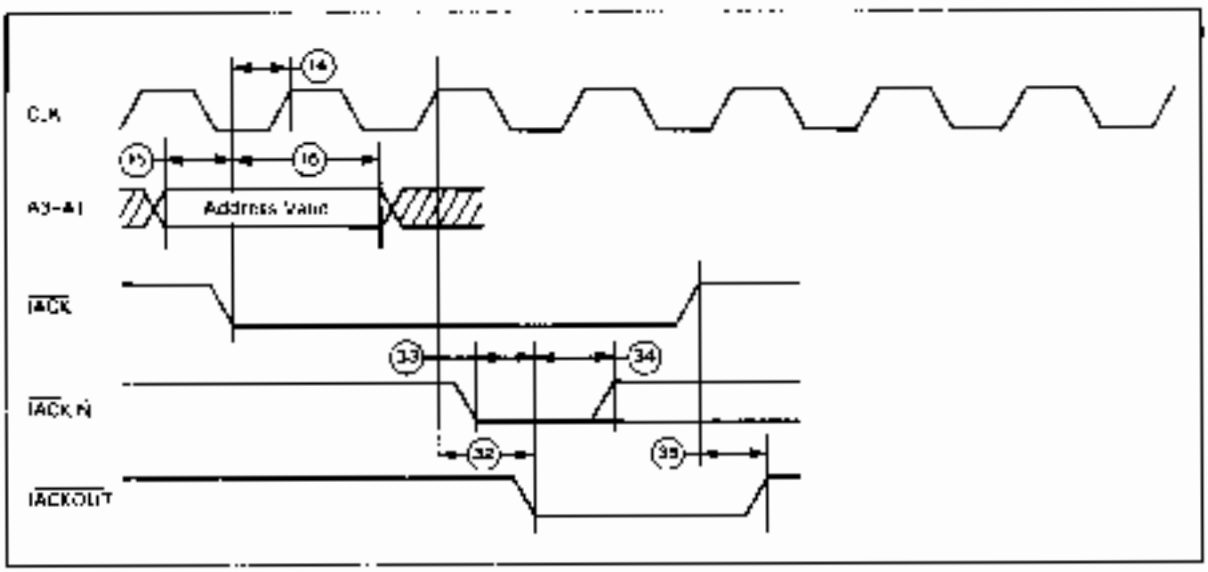


FIGURE 11 — INTERRUPT ACKNOWLEDGE CYCLE — EXTERNAL VECTOR

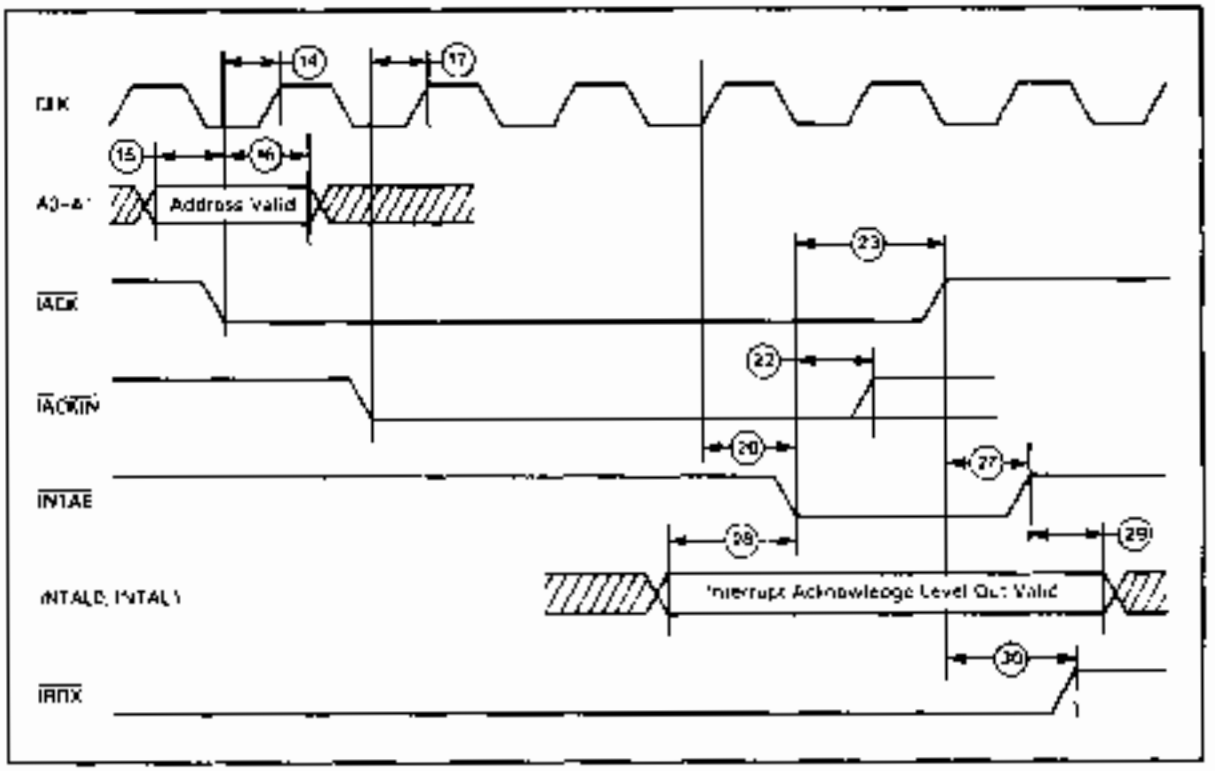
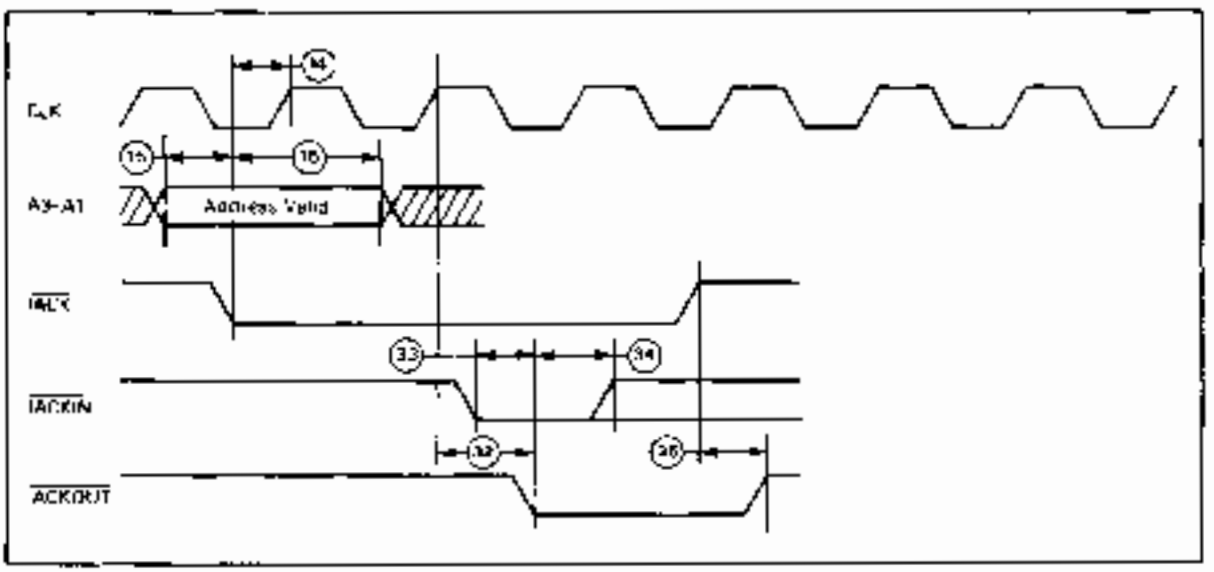


FIGURE 12 — INTERRUPT ACKNOWLEDGE CYCLE — IACKOUT



CONTROL REGISTER FLAGS

Each control register contains a Flag bit (F) and a Flag Auto-Clear bit (FAC). Both bits can be read or altered via a register write without affecting the interrupt operation of the device. The Flag is useful as a status indicator for resource management and as a semaphor in multitasking or multiprocessor systems. Flag (F) is located in bit position 7 and can be used with the MC68000 Test and Set (TAS) instruction.

The Flag Auto-Clear (FAC) is used to manipulate the Flag bit. If the Flag is set to one and the FAC is also one, an interrupt acknowledge cycle to the associated interrupt source clears the Flag bit. This feature is useful in determining the interrupt status and passing messages.

RESET

There is no reset input, however, a chip reset is activated by asserting both $\overline{CS}$ and $\overline{IACK}$ simultaneously (Figure 13). These inputs should be held low for a minimum of two clock cycles for a full reset function. The control registers are reset to all zeroes and the Vector Registers are set to a value of $0F. This vector value is the uninitialized vector for the MC68000. See the MC68000 User's Manual for more details on this vector.

CLOCK

The chip clock is required for internal operation to occur. Typical frequency is 16 MHz in VMEbus and VERSAbus applications derived from the system clock. Any frequency can be used, however, up to 25 MHz (Figure 14).

FIGURE 13 — RESET

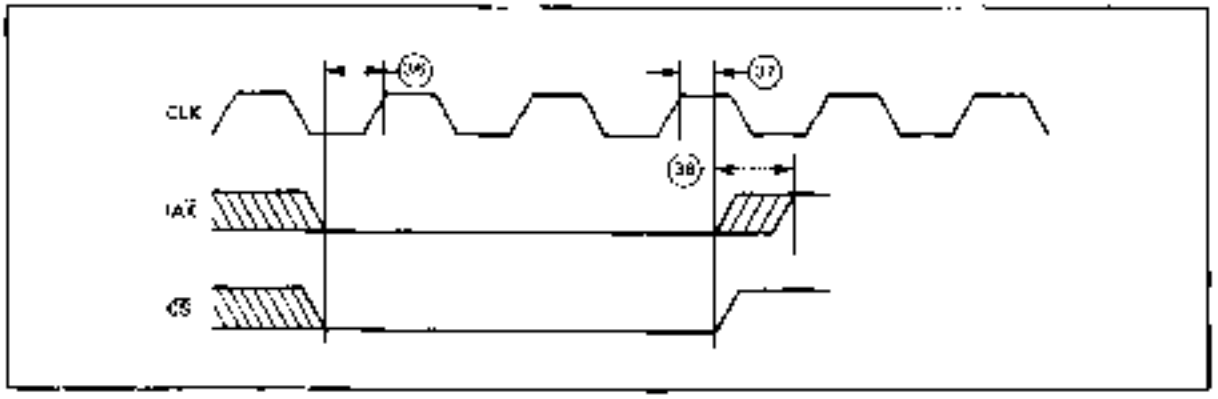


FIGURE 14 — CLOCK WAVEFORM

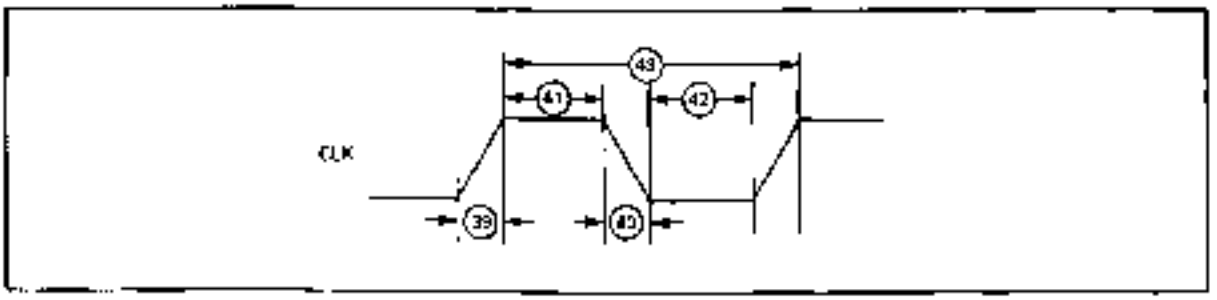


TABLE 1
AC PERFORMANCE SPECIFICATIONS
(V_{CC} = 5.0 V ± 5%, T_A = 0°C to 70°C)

Number	Characteristic	Min	Max	Units	Notes
1	R/W, A1–A3 Valid to CS Low (Setup Time)	10	—	ns	
2	CS Low to R/W, A1–A3 Invalid (Hold Time)	5.0	—	ns	
3	CS Low to CLK High (Setup Time)	15	—	ns	1
4	CLK High to Data Out Valid (Delay)	—	55	ns	2
5	CLK High to DTACK Low (Delay)		40	ns	2
6	DTACK Low to CS High	0		ns	
7	CS High to DTACK High (Delay)	-	35	ns	10
8	CS High to Data Out Invalid (Hold Time)	0	—	ns	
9	CS High to Data Out High-Impedance (Hold Time)	-	50	ns	
10	CS High to CS or IACK Low	20	—	ns	
11	Data In Valid to CS Low (Setup Time)	10	—	ns	
12	CS Low to Data In Invalid (Hold Time)	5.0	-	ns	
13	DTACK High to Data Out High-Impedance	--	25	ns	10
14	IACK Low to CLK High (Setup Time)	15	—	ns	1
15	A1–A3 Valid to IACK Low (Setup Time)	10	—	ns	
16	IACK Low to A1–A3 Invalid (Hold Time)	5.0	—	ns	
17	IACKIN Low to CLK High (Setup Time)	15	—	ns	1, 8
18	CLK High to Data Out Valid (Delay)	—	55	ns	3
19	CLK High to DTACK Low (Delay)	—	40	ns	3
20	CLK High to INTAE Low (Delay)	—	40	ns	3
22	DTACK Low to IACKIN High	0	—	ns	8
23	DTACK Low to IACK High	0	—	ns	
24	IACK High to Data Out Invalid (Hold Time)	0	—	ns	
25	IACK High to Data Out High Impedance (Delay)	—	60	ns	
26	IACK High to DTACK High (Delay)	—	45	ns	10
27	IACK High to INTAE High (Delay)	-	35	ns	
28	INTAL0, INTAL1 Valid to INTAE Low (Setup Time)	1.0	2.0	CLK Per	
29	INTAE High to INTAL0, INTAL1 Invalid (Hold Time)	1.0	2.0	CLK Per	
30	IACK High to IRQx High (Delay)	—	50	ns	7, 10
31	IACK High to IACK or CS Low	20	—	ns	
32	CLK High to IACKOUT Low (Delay)	—	40	ns	5
33	IACKIN Low to IACKOUT Low (Delay)	—	30	ns	4, 8
34	IACKOUT Low to IACKIN, IACK High	0	—	ns	8
35	IACK High to IACKOUT High (Delay)	—	35	ns	
36	IACK and CS both Low to CLK High (Setup Time)	15	—	ns	9
37	CLK High to IACK or CS High (Hold Time)	0	—	ns	
38	IACK or CS High to IACK and CS High (Skew)	--	1.0	CLK Per	6
39	Clock Rise Time	—	10	ns	
40	Clock Fall Time	--	10	ns	
41	Clock High Time	20	—	ns	
42	Clock Low Time	20	—	ns	
43	Clock Period	40	—	ns	

NOTES:

1. This specification only applies if the VBIM had completed all operations initiated by the previous bus cycle when CS or IACK was asserted. Following a normal bus cycle, all operations are completed within 2 clock cycles after CS or IACK have been negated. If IACK or CS is asserted prior to completion of these operations, the new cycle, and hence, DTACK is postponed.

   If the IACK, IACKIN or CS setup time is violated, DTACK may be asserted as shown, or may be asserted one clock cycle later (i.e. IACK will not be recognized until the next rising edge of the clock).

2. Assumes that 3 has been met.

3. Assumes that 14 and 17 have both been met.

4. Assumes that 14 has been met. IACKOUT cannot go low prior to IACKIN going low.

5. Assumes that 14 has been met and IACKIN has been low for at least the amount of time specified by 33.

6. 38 is the minimum skew between the last moment when both IACK and CS are asserted to when both are negated, to ensure that an access cycle is not unintentionally started.

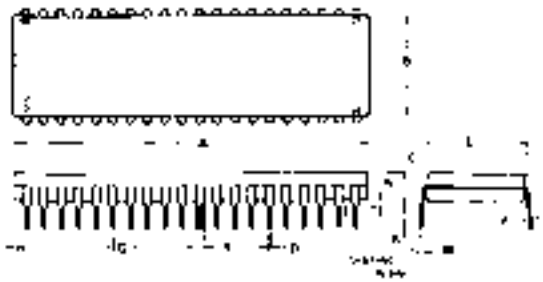
7. Assumes no other INTx input is causing IRQx to be driven low.

8. In non-daisy chain systems, IACKIN may be tied low.

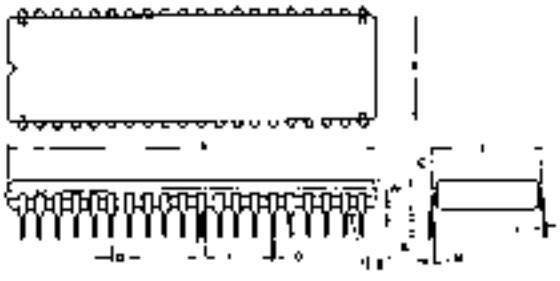
9. Failure to meet this spec causes RESET to be ignored for 1 clock period. It is then recognized. To VBIM these signals low for 3 clock periods, instead of 2.

10. Delay time is specified from input signal to Open-Collector Output pulled High thru 1.0 kΩ resistor to +5.5 V.

OUTLINE DIMENSIONS



CASE 711-03
PLASTIC PACKAGE



CASE 734-04
CERAMIC PACKAGE

TYPICAL THERMAL CHARACTERISTICS

Package	θ_{JA} (Junction to Ambient) Still Air	Junction Temperature Still Air @ 70°C Ambient
L Suffix	40°C/W	147°C
P Suffix	85°C/W	137°C

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and (M) are registered trademarks of Motorola, Inc.

MOTOROLA GMBH Geschäftsbereich Halbleiter
POSTFACH 1115, MÜNCHNER STRASSE 8, 8043 UNTERFÖHRING, W. GERMANY

MOTOROLA
SEMICONDUCTORS
EAST KILBRIDE SCOTLAND

Advance Information

MC68230

PARALLEL INTERFACE/TIMER (PI/T)

DECEMBER, 1983

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
	Section 1 Introduction	
1.1	Port Mode Description	1-1
1.2	Signal Description	1-2
1.2.1	Bidirectional Data Bus (D0-D7)	1-5
1.2.2	Register Select (RS1-RS0)	1-5
1.2.3	Read (RD) or R/W	1-5
1.2.4	Chip Select ($\overline{CS}$)	1-5
1.2.5	Data Transfer Acknowledge ($\overline{DTACK}$)	1-6
1.2.6	Reset ($\overline{RESET}$)	1-7
1.2.7	Clock (CLK)	1-7
1.2.8	Port A and Port B (PA0-PA7 and PB0-PB7)	1-8
1.2.9	Handshake Lines (H1-H4)	1-8
1.2.10	Port C (PC0-PC7) Alternate Functions	1-9
1.3	Register Model	1-9
1.4	Bus Interface Operation	1-10
1.4.1	Read Cycles	1-10
1.4.2	Interrupt Acknowledge Cycles	1-11
1.4.3	Write Cycles	1-11
	Section 2 Port General Information and Conventions	
2.1	Functional Block Diagram	2-1
2.1	Control of Double-Buffered Data Paths	2-1
2.1.2	Double-Buffered Input Transfers	2-2
2.1.3	Double-Buffered Output Transfers	2-2
2.2	Supporting Bus Master Service	2-3
2.2	Vectored, Prioritized Interrupts	2-3
2.2.1	Automatic Port Interrupts	2-3
2.2.2	DMA Request Operation	2-3
2.3	Direct Monitoring of Peripheral Status	2-3
2.4	Handshake Bus Timing Control	2-3
2.5	Enabling Ports A and B	2-4
2.6	Port A and B Alternate Functions	2-4

TABLE OF CONTENTS
(Continued)

Paragraph Number	Title	Page Number
	Section 3 **Port Modes**	
3.1	Port A Control Register	3-1
3.2	Port B Control Register	3-1
3.3	Mode 0 — Unidirectional 8-Bit Mode	3-2
3.3.1	Submode 00 — Pin Definable Double-Buffered Input or Single-Buffered Output	3-2
3.3.2	Submode 01 — Pin Definable Double-Buffered Output or Non-Latched Input	3-4
3.3.3	Submode 1X — Port C Pins Definable Single-Buffered Output or Non-Latched Input	3-6
3.4	Mode 1 — Unidirectional 16-Bit Mode	3-8
3.4.1	Port A Control Register	3-8
3.4.2	Port B Control Register	3-8
3.4.3	Submode X0 — Pin Definable Double-Buffered Input or Single-Buffered Output	3-9
3.4.4	Submode X1 — Pin Definable Double-Buffered Output or Non-Latched Input	3-12
3.5	Mode 2 — Bidirectional 8-Bit Mode	3-13
3.5.1	Port A Pin Definable Single-Buffered Output or Non-Latching Input	3-14
3.5.2	Port B Double-Buffered I/O	3-14
3.5.2.1	Double-Buffered Input Transfers	3-15
3.5.2.2	Double-Buffered Output Transfers	3-15
3.6	Mode 3 — Bidirectional 16-Bit Mode	3-17
3.6.1	Double-Buffered Input Transfers	3-17
3.6.2	Double-Buffered Output Transfers	3-18
	Section 4 **Programmer's Model**	
4.1	Port Control Command Register (PCCR)	4-1
4.2	Port Status Register (PSR)	4-3
4.3	Port Data Direction Registers	4-4
4.3.1	Port A Data Direction Register (PADDR)	4-4
4.3.2	Port B Data Direction Register (PBDDR)	4-4
4.3.3	Port C Data Direction Register (PCDDR)	4-4
4.4	Port Interrupt Vector Register (PIVR)	4-4
4.5	Port Control Registers (PACR, PBCR)	4-4
4.6	Port Data Registers	4-7
4.6.1	Port A Data Register (PADR)	4-7
4.6.2	Port B Data Register (PBDR)	4-7
4.6.3	Port C Data Register (PCDR)	4-7

TABLE OF CONTENTS
(Concluded)

Paragraph Number	Title	Page Number
4.7	Port Alternate Registers	4-9
4.7.1	Port A Alternate Register (PAAR)	4-9
4.7.2	Port B Alternate Register (PBAR)	4-9
4.8	Port State Register (PSR)	4-9
4.9	Timer Control Register (TCR)	4-9
4.10	Timer Interrupt Vector Register (TIVR)	4-10
4.11	Counter Preload Register H, M, L (CPRH, M, L)	4-11
4.12	Count Register H, M, L (CNTRH, M, L)	4-11
4.13	Timer Status Register (TSR)	4-12
4.14	Register Value After Reset	4-13

Section 5
Timer Operation and Applications Summary

5.1	Timer Operation	5-1
5.1.1	Run/Halt Definition	5-1
5.1.2	Timer Rules	5-1
5.1.3	Timer Interrupt Acknowledge Cycles	5-2
5.2	Timer Applications Summary	5-2
5.2.1	Periodic Interrupt Generator Examples	5-3
5.2.2	Square Wave Generator	5-3
5.2.3	Interrupt After Timeout	5-4
5.2.4	Elapsed Time Measurement Examples	5-5
5.2.4.1	System Clock Example	5-5
5.2.4.2	External Clock	5-6
5.2.5	Device Watchdog	5-6

Section 6
Electrical Specifications

6.1	Maximum Ratings	6-1
6.2	Thermal Characteristics	6-1
6.3	Power Considerations	6-1
6.4	DC Electrical Characteristics	6-2
6.5	AC Electrical Specifications — Clock Timing	6-3
6.6	AC Electrical Specifications	6-4

Section 7
Mechanical Data and Ordering Information

7.1	Pin Assignment	7-1
7.2	Ordering Information	7-2
7.3	Package Dimensions	7-2

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
1-1	Block Diagram	1-2
1-2	Port Mode Layout	1-4
1-3	Typical Pin Assignment	1-5
2-1	Double Buffered Input Transfers Timing Diagram	2-3
2-2	Double Buffered Output Transfers Timing Diagram	2-4
2-3	$\overline{DMARQ}$ Associated with Output Transfers	2-6
2-4	$\overline{DMARQ}$ Associated with Input Transfers	2-6
5-1	Periodic Interrupt Generator Example	5-3
5-2	Square Wave Generator Example	5-4
5-3	Single Interrupt After Timeout Example	5-5
5-4	Elapsed Time Measurement Example	5-6
5-5	Device Watchdog Example	5-7
6-1	Clock Input Timing Diagram	6-3
6-2	Read Cycle Timing Diagram	Foldout 3
6-3	Write Cycle Timing Diagram	Foldout 3
6-4	IACK Timing Diagram	Foldout 4
6-5	Peripheral Input Timing Diagram	Foldouts 5 and 6
6-6	Peripheral Output Timing Diagram	Foldouts 5 and 6

LIST OF TABLES

Table Number	Title	Page Number
1-1	Port Mode Control Summary	1-1
1-2	Signal Summary	1-2
1-3	Register Model	1-7
2-1	Response to Port Interrupt Acknowledge	2-1
3-1	Mode 0 Port Data Paths	3-2
3-2	Mode 1 Port Data Paths	3-4
3-3	Mode 2 Port A Data Paths	3-14
3-4	Mode 2 Port B Data Paths	3-15
3-5	Mode 2 Port A and B Data Paths	3-16
4-1	PCT Register Addressing Assignments	4-1
4-2	PSAB Port Interrupt Priority Control	4-1
4-3	PCDR Hardware Addresses	4-3
5-1	Response to Timer Interrupt Acknowledge	5-1

SECTION 1
INTRODUCTION

The MC68230 parallel interface/timer (PI/T) provides versatile double buffered parallel interfaces and a system oriented timer for M68000 systems. The parallel interfaces operate in unidirectional or bidirectional modes, either 8 or 16 bits wide. In the unidirectional modes, an associated data direction register determines whether each port pin is an input or output. In the bidirectional modes the data direction registers are ignored and the direction is determined dynamically by the state of four handshake pins. These programmable handshake pins provide an interface flexible enough for connection to a wide variety of low, medium, or high speed peripherals or other computer systems. The PI/T ports allow use of vectored or autovectored interrupts, and also provide a DMA request pin for connection to the MC68450 direct memory access controller (DMAC) or a similar circuit. The PI/T timer contains a 24 bit wide counter and a 5-bit prescaler. The timer may be clocked by the system clock (PI/T CLK pin) or by an external clock (TIN pin), and a 5-bit prescaler can be used. It can generate periodic interrupts, a square wave, or a single interrupt after a programmed time period. It can also be used for elapsed time measurement or as a device watchdog.

Features of the PI/T include:

- M68000 Bus Compatible
- Port Modes Include:
  - Bit I/O
  - Unidirectional 8 Bit and 16 Bit
  - Bidirectional 8 Bit and 16 Bit
- Programmable Handshaking Options
- 24-Bit Programmable Timer Modes
- Five Separate Interrupt Vectors
- Separate Port and Timer Interrupt Service Requests
- Registers are Read/Write and Directly Addressable
- Registers are Addressed for MOVEP (Move Peripheral) and DMAC Compatibility

The PI/T consists of two logically independent sections: the ports and the timer. The port section consists of port A (PA0-PA7), port B (PB0-PB7), four handshake pins (H1, H2, H3, and H4), two general input/output (I/O) pins, and six dual-function pins. The dual-function pins can individually operate as a third port (port C) or an alternate function related to either port A, port B, or the timer. The four programmable handshake pins, depending on the mode, can control data transfer to and from the ports, or can be used as interrupt generating inputs or I/O pins. Refer to Figure 1-1.

The timer consists of a 24 bit counter, optionally clocked by a 5-bit prescaler. Three pins provide complete timer I/O: PC2/TIN, PC3/TOUT, and PC7/$\overline{\text{TPACK}}$. Only the ones needed for the given configuration perform the timer function, while the others remain port C I/O.

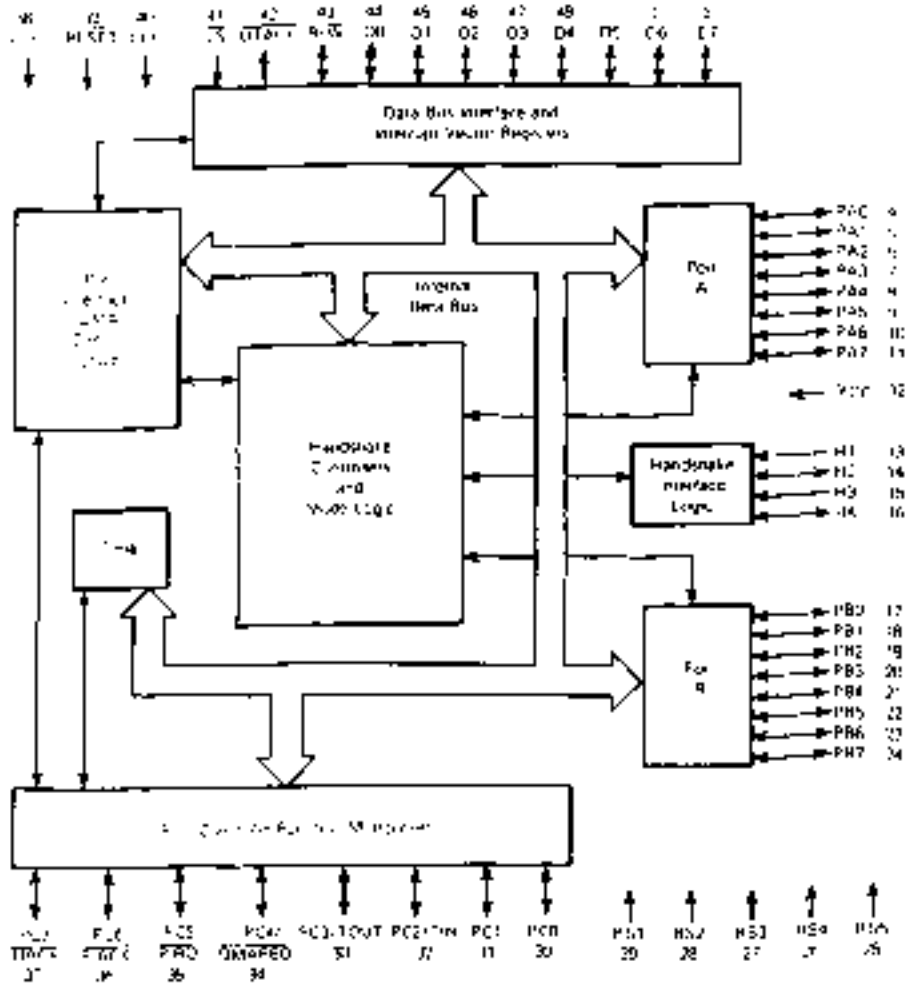


Figure 1-1. Block Diagram

The system bus interface provides for asynchronous transfer of data from the PI/T to a bus master over the data bus (D0-D7). Data transfer acknowledge ($\overline{DTACK}$), register selects (RS1-RS5), interrupt acknowledge ($\overline{ITACK}$), read/write line (R/$\overline{W}$), chip select ($\overline{CS}$), or port interrupt acknowledge ($\overline{PIACK}$) control data transfer between the PI/T and an M68000.

1.1 PORT MODE DESCRIPTION

The primary focus of most applications will be on port A, port B, the handshake pins, the port interrupt pins, and the DMA request pin. They are controlled in the following way: the port general control register contains a 2-bit field that specifies one of four operation modes. These govern the

overall operation of the ports and determine their interrelationships. Some modes require additional information from each port's control register to further define its operation. In each port control register, there is a 2-bit submode field that serves this purpose. Each port mode/submode combination specifies a set of programmable characteristics that fully define the behavior of that port and two of the handshake pins. This structure is summarized in Table 1-1 and Figure 1-2.

Table 1-1. Port Mode Control Summary

Mode 0 (Unidirectional 8-Bit Mode)
- Port A
  - Submode 00 — Pin Definable Double Buffered Input or Single Buffered Output
    - H1 — Latches input data
    - H2 — Status interrupt generating input, general purpose output, or operation with H1 in the interlocked or pulsed handshake protocols
  - Submode 01 — Pin Definable Double Buffered Output or Non Latched Input
    - H1 — Indicates data received by peripheral
    - H2 — Status interrupt generating input, general purpose output, or operation with H1 in the interlocked or pulsed handshake protocols
  - Submode 1X — Pin Definable Single Buffered Output or Non Latched Input
    - H1 — Status interrupt generating input
    - H2 — Status interrupt generating input or general purpose output
- Port B
  - H3 and H4 — Identical in port A with H1 and H2

Mode 1 (Unidirectional 16-Bit Mode)
- Port A — Most Significant Data Byte or Non Latched Input or Single Buffered Output
  - Submode XX — Not used
    - H1 — Status interrupt generating input
    - H2 — Status interrupt generating input or general purpose output
- Port B — Least Significant Data Byte
  - Submode XX — Pin Definable Double Buffered Input or Single-Buffered Output
    - H3 — Latches input data
    - H4 — Status interrupt generating input, general purpose output, or operation with H3 in the interlocked or pulsed handshake protocols
  - Submode X1 — Pin Definable Double Buffered Output or Non Latched Input
    - H2 — Indicates data received by peripheral
    - H4 — Status interrupt generating input, general purpose output, or operation with H3 in the interlocked or pulsed handshake protocols

Mode 2 (Bidirectional 8 Bit Mode)
- Port A — Bit I/O
  - Submode XX — Not used
- Port B — Double Buffered Bidirectional Data
  - Submode XX — Not used
    - H1 — Indicates output data received by the peripheral and controls output drivers
    - H2 — Operation with H1 in the interlocked or pulsed output handshake protocols
    - H3 — Latches input data
    - H4 — Operation with H3 in the interlocked or pulsed input handshake protocols

Mode 3 (Bidirectional 16-Bit Mode)
- Port A — Pin Definable Bidirectional Data - Most Significant Data Byte
  - Submode XX — Not used
- Port B — Double Buffered Bidirectional Data (Least Significant Data Byte)
  - Submode XX — Not used
    - H1 — Indicates output data received by peripheral and controls output drivers
    - H2 — Operation with H1 in the interlocked or pulsed output handshake protocols
    - H3 — Latches input data
    - H4 — Operation with H3 in the interlocked or pulsed input handshake protocols

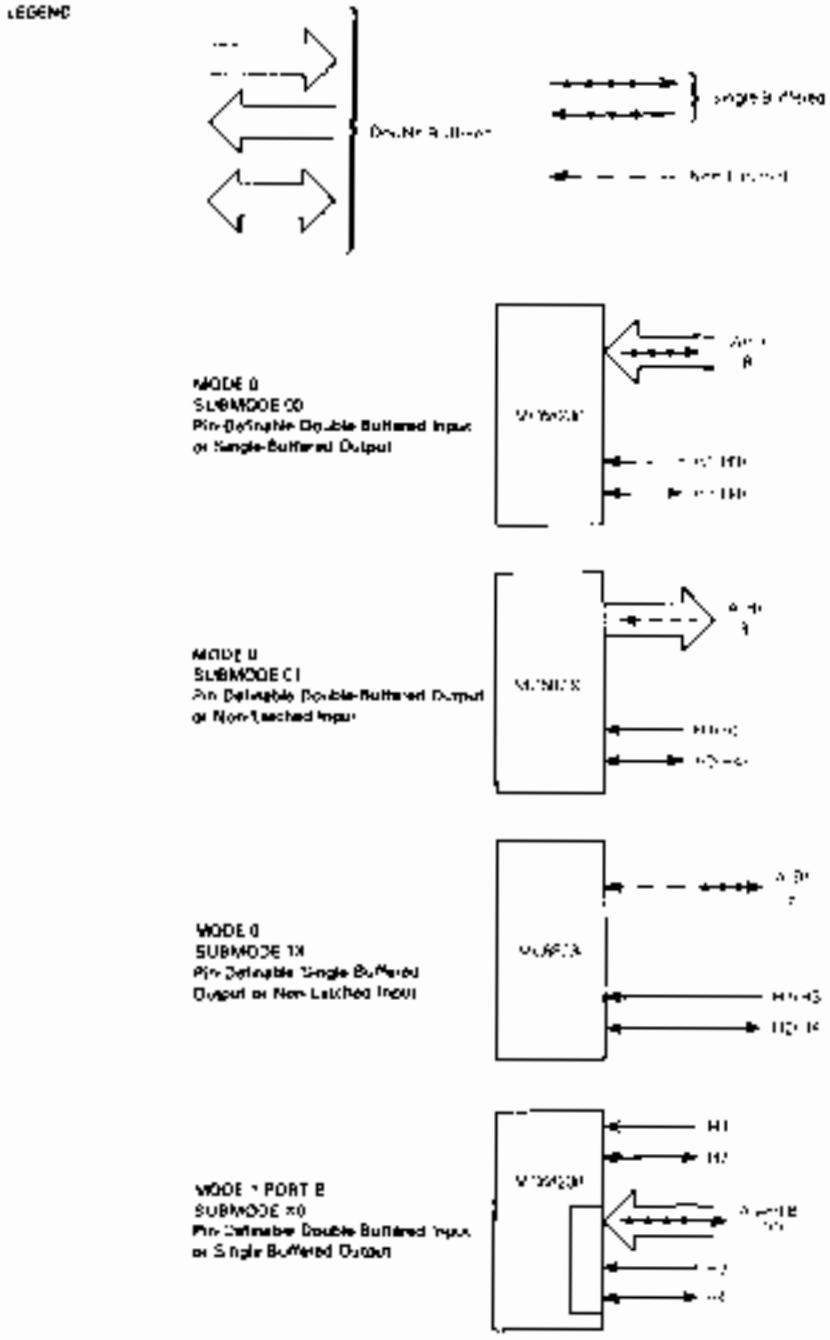


Figure 1-2. Port Mode Layout (Sheet 1 of 2)

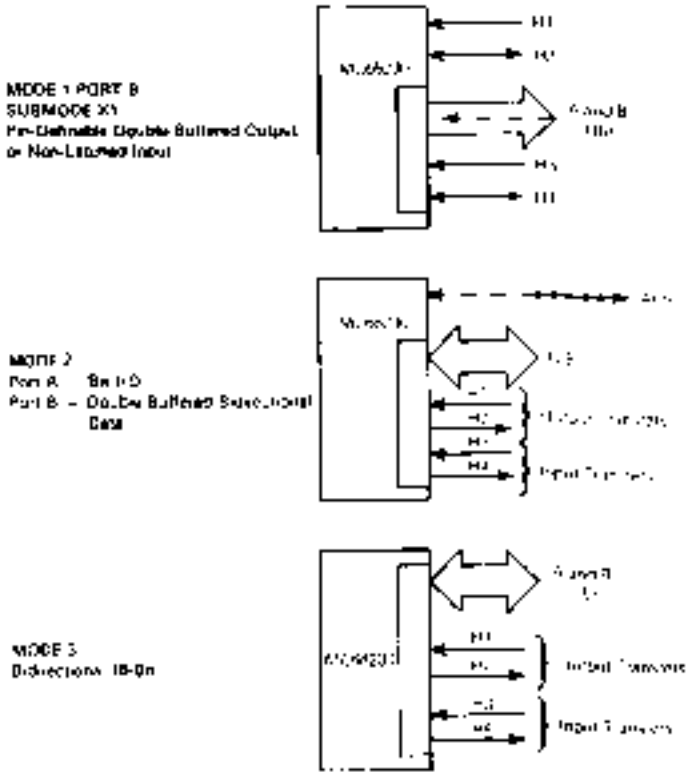


Figure 1-2. Port Mode Layout (Sheet 2 of 2)

1.2 SIGNAL DESCRIPTION

Throughout this data sheet, signals are presented using the terms active and inactive or asserted and negated independent of whether the signal is active in the high-voltage state or low-voltage state. (The active state of each logic pin is given below.) Active low signals are denoted by a superscript bar. $R/\overline{W}$ indicates a write is active low and a read active high. Table 1-1 further describes each pin and the logical pin assignments are given in Figure 1-3.

1.2.1 Bidirectional Data Bus (D0-D7)

The data bus pins, D0-D7, form an 8-bit bidirectional data bus to/from an M68000 bus master. These pins are active high.

1.2.2 Register Selects (RS1-RS0)

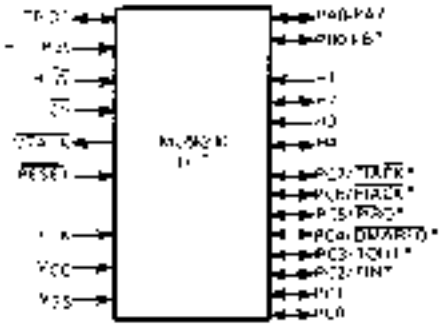
The register select pins, RS1, RS0, are active high inputs that determine which of the 23 internal registers is being selected. They are provided by the M68000 bus master or other bus master.

Table 1-2 Signal Summary

Signal Name	Input/Output	Active State	Edge/Level Sensitive	Output State
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	Level	[illegible]
[illegible]	[illegible]	[illegible]	Level	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

[illegible]



*Individually Programmable Dual Function Pin

Figure 1-3. Logical Pin Assignment

1.2.3 Read/Write (R/$\overline{W}$)

R/$\overline{W}$ is a high impedance read/write input signal from the M68000 bus master indicating whether the current bus cycle is a read (high) or write (low) cycle.

1.2.4 Chip Select ($\overline{CS}$)

$\overline{CS}$ is a high impedance input that selects the PI/T registers for the current bus cycle. Address strobe and the data strobe (upper or lower) of the bus master, along with the appropriate address bits, must be included in the chip select equation. A low level corresponds to an asserted chip select.

1.2.5 Data Transfer Acknowledge (DTACK)

DTACK is an active low output that signals the completion of the bus cycle. During read or interrupt acknowledge cycles, DTACK is asserted after data has been provided on the data bus. during write cycles it is asserted after data has been accepted at the data bus. Data transfer acknowledge is compatible with the MC68000 and with other M68000 bus masters such as the MC68450 direct memory access controller (DMAC). A pullup resistor is required to maintain DTACK high between bus cycles.

1.2.6 Reset (RESET)

RESET is a high-impedance input used to initialize all PI/T functions. All control and data direction registers are cleared and most internal operations are disabled by the assertion of RESET low.

1.2.7 Clock (CLK)

The clock pin is a high impedance TTL compatible signal with the same specifications as the MC68000. The PI/T contains dynamic logic throughout, and hence this clock must not be gated off at any time. It is not necessary that this clock maintain any particular phase relationship with the M68000 system clock. It may be connected to an independent frequency source (faster or slower) as long as all bus specifications are met.

1.2.8 Port A and Port B (PA0-PA7 and PB0-PB7)

Ports A and B are 8-bit ports that may be concatenated to form a 16 bit port in certain modes. The ports may be controlled in conjunction with the handshake pins H1-H4. For stabilization during system power up, ports A and B have internal pullup resistors to VCC. All port pins are active high.

1.2.9 Handshake Pins (H1-H4)

Handshake pins H1-H4 are multi-purpose pins that, depending on the operational mode, may provide an interlocked handshake, a pulsed handshake, an interrupt input (independent of data transfers), or simple I/O pins. For stabilization during system power up, H2 and H4 have internal pullup resistors to VCC. The sense of H1-H4 (active high or low) may be programmed in the port general control register bits 3-0. Independent of the mode, the instantaneous level of the handshake pins can be read from the port status register.

1.2.10 Port C (PC0-PC7/Alternate Function)

This port can be used as eight general purpose I/O pins (PC0-PC7) or any combination of six special function pins and two general purpose I/O pins (PC0-PC1). Each dual function pin can be a standard I/O or a special function independent of the other port C pins. When used as a port C pin, these pins are active high. They may be individually programmed as inputs or outputs by the port C data direction register. The dual-function pins are defined in the following paragraphs.

The alternate functions TIN, TOUT and $\overline{*ACK}$ are other I/O pins. TIN may be used as a rising edge triggered external clock input or an external run/halt control pin (the timer is in the run state if run/halt is high and in the halt state if run/halt is low). TOUT may provide an active low offset interrupt request output or a general-purpose square-wave output, initially high. $\overline{*ACK}$ is an active low high-impedance input used for timer interrupt acknowledge.

Port A and B functions have an independent pair of active low interrupt request (PIRQ) and interrupt acknowledge (PIACK) pins.

The DMAREQ active memory request pin provides an active low signal to request memory access control for three cycles. Compatible with the MC68450 DMAC.

1.3 REGISTER MODEL

A register model that shows the correspondence between register selects is shown in Table 1-3.

Table 1-3 Register Model (Sheet 1 of 2)

Register Select Bits 5 4 3 2 1	7	6	5	4	3	2	1	0	Register Value After RESET (Hex Value)	
0 0 0 0 0	Port A Vector		0 (L)	0	0	0	0	0	0	Port A Vector Register
0 0 0 0 1	*		SYSCLK Select			Port A Interrupt Pulse Control			00	Port A Interrupt Control Register
0 0 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	00	Port A Data Direction Register
0 0 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	00	Port B Data Direction Register
0 0 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	00	Port C Data Direction Register
0 0 1 0 1	Interrupt Vector Number						*	*	0F	Port Interrupt Vector Register
0 0 1 1 0	Port A Commands		H Timers			H2 or Enable	H1 SYCRQ Enable	H0 Start Bit	00	Port A Command Register
0 0 1 1 1	Port B Commands		H Control			H4 or Enable	H3 SYCRQ Enable	H2 Start Bit	00	Port B Command Register
0 1 0 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port A Data Register
0 1 0 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port B Data Register
0 1 0 1 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port A Alternate Register
0 1 0 1 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port B Alternate Register
0 1 1 0 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port A Status Register
0 1 1 0 1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	xx	Port B Status Register
0 1 1 1 0	H Vector		H Data (Low)	H1 (Low)	DFC	H0S	H0S	H Vector	xx	Port H Status Register
0 1 1 1 1	*	*	*	*	*	*	*	*	*	Undefined
1 * * * *	*	*	*	*	*	*	*	*	*	Undefined

* Undefined, reserved for future use
** Active low output (RESET)
*** Undefined, but initialized to zero
**** Undefined, but initialized to one

Table 1-3. Register Model (Sheet 2 of 2)

Register Select Bits														Register Value After RESET (Hex Value)	
5	4	3	2	1	7	6	5	4	3	2	1	0		
*	0	0	0	0	TOUT TRACK Control			2.0 LO	*	Count Control			Timer Enable	00	Timer Control Register
*	0	0	0	1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		00	Timer Interval Select Register
*	0	0	1	0	*	*	*	*	*	*	*	*		00	*Note *
*	0	0	1	1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16		**	Counter Preload Register - High
*	0	1	0	0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8		**	Counter Preload Register - Mid
*	0	1	0	1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		**	Counter Preload Register - Low
*	0	1	1	0	*	*	*	*	*	*	*	*		00	NU
*	0	1	1	1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16		**	Count Register - High
*	1	0	0	0	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8		**	Count Register - Mid
*	1	0	0	1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0		**	Count Register - Low
*	1	0	1	0	*	*	*	*	*	*	*	ZDS		00	Timer Status Register
*	1	0	1	1	*	*	*	*	*	*	*	*		00	NU
*	1	1	0	0	*	*	*	*	*	*	*	*		00	NU
*	1	1	0	1	*	*	*	*	*	*	*	*		00	NU
*	1	1	1	0	*	*	*	*	*	*	*	*		00	NU
*	1	1	1	1	*	*	*	*	*	*	*	*		00	NU

*Unused - read as zero
**Value before RESET

NOTE

Table 1-3 has been duplicated on foldout pages 1 and 2 at the end of this document for your convenience.

1.4 BUS INTERFACE OPERATION

The PI/T has an asynchronous bus interface, primarily designed for use with an M68000 microprocessor. With care, however, it can be connected to synchronous microprocessor buses. This section completely describes the PI/T's bus interface, and is intended for the asynchronous bus designer unless otherwise mentioned.

In an asynchronous system the PI/T clock may operate at a significantly different frequency, either higher or lower, than the bus master and other system components, as long as all bus specifications are met. The MC68230 CLK pin has the same specifications as the MC68000 CLK pin, and must not be gated off at any time.

The following signals generate normal read and write cycles to the PI/T: $\overline{CS}$ (chip select), R/$\overline{W}$ (read/write), RS1-RS5 (five register select pins), D0-D7 (the 8-bit bidirectional data bus), and $\overline{DTACK}$ (data transfer acknowledge). To generate interrupt acknowledge cycles, $\overline{PIACK}$ or $\overline{TIACK}$ is used instead of $\overline{CS}$, and the register select pins are ignored. No combination of the following pin functions may be asserted simultaneously: $\overline{CS}$, $\overline{PIACK}$, or $\overline{TIACK}$.

1.4.1 Read Cycles

The category includes all register reads, except port or timer interrupt acknowledge cycles. When $\overline{CS}$ is asserted, the register select and R/$\overline{W}$ inputs are latched internally. They must meet small setup and hold time requirements with respect to the asserted edge of $\overline{CS}$. (Refer to **6.8 AC ELECTRICAL SPECIFICATIONS** for further information.) The PI/T is not protected against aborted (shortened) bus cycles generated by an address error or bus error exception in which it is addressed.

Certain operations triggered by normal read or write bus cycles are not complete within the time allotted to the bus cycle. One example is transfers to/from the double-buffered latches that occur as a result of the bus cycle. If the bus master's clock is significantly faster than the PI/T's, the possibility exists that, following the bus cycle, $\overline{CS}$ can be negated then re-asserted before completion of these internal operations. In this situation the PI/T does not recognize the re-assertion of $\overline{CS}$ until these operations are complete. Only at that time does it begin the internal sequencing necessary to react to the asserted $\overline{CS}$. Since $\overline{CS}$ also controls the $\overline{DTACK}$ response, this "bus cycle recovery time" can be related to the clock edge on which $\overline{DTACK}$ is asserted for that cycle. The PI/T will recognize the subsequent assertion of $\overline{CS}$ three clock periods after the clock edge on which $\overline{DTACK}$ was previously asserted.

The register select and R/$\overline{W}$ inputs pass through an internal latch that is transparent when the PI/T can recognize a new $\overline{CS}$ pulse (see above paragraph). Since the internal data bus of the PI/T is continuously engaged for read transfers, the read access time (to the data bus buffers) begins when the register selects are stabilized internally. Also, when the PI/T is ready to begin a new bus cycle, the assertion of $\overline{CS}$ enables the data bus buffers within a short propagation delay. This does not contribute to the overall read access time unless $\overline{CS}$ is asserted significantly after the register select and R/$\overline{W}$ inputs are stabilized (as may occur with synchronous bus microprocessors).

In addition to the chip select previously mentioned duties, it controls the assertion of $\overline{DTACK}$ and latching of read data at the data bus interface. Except for controlling input latches and enabling the data bus buffers, all of these functions occur only after $\overline{CS}$ has been recognized internally and synchronized with the internal clock. Chip select is recognized on the falling edge of the clock if the setup time is met, $\overline{DTACK}$ is asserted (low) on the next falling edge of the clock. Read data is

latched at the PI/T's data bus interface at the same time $\overline{DTACK}$ is asserted. It is stable as long as chip select remains asserted independent of other external conditions.

From the above discussion it is clear that if the chip select setup time prior to the falling edge of the clock is met, the PI/T can consistently respond to a new read or write bus cycle every four clock cycles. This fact is especially useful in designing the PI/T's clock in synchronous bus systems not using $\overline{DTACK}$. (An extra clock period is required in interrupt acknowledge cycles, see **1.4.2 Interrupt Acknowledge Cycles**.)

In asynchronous bus systems in which the PI/T's clock differs from that of the bus master, generally there is no way to guarantee that the chip select setup time with respect to the PI/T clock is met. Thus, the only way to determine that the PI/T recognized the assertion of $\overline{CS}$ is to wait for the assertion of $\overline{DTACK}$. In this situation, all latched bus inputs to the PI/T must be held stable until $\overline{DTACK}$ is asserted. These include register select, R/$\overline{W}$, and write data inputs (see below).

System specifications impose a maximum delay from the trailing (negated) edge of $\overline{CS}$ to the negated edge of $\overline{DTACK}$. As system speeds increase this becomes more difficult to meet with a simple pullup resistor tied to the $\overline{DTACK}$ line. Therefore, the PI/T provides an internal active pullup device to reduce the rise time, and a level sensitive circuit that later turns this device off. $\overline{DTACK}$ is negated asynchronously as fast as possible following the rising edge of chip select, then three-stated to avoid interference with the next bus cycle.

The system designer must take care that $\overline{DTACK}$ is negated and three-stated quickly enough after each bus cycle to avoid interference with the next one. With an M68000 this necessitates a relatively fast external path from the data strobe negation to $\overline{CS}$ bus master negation.

1.4.2 Interrupt Acknowledge Cycles

Special internal operations take place on PI/T interrupt acknowledge cycles. The port interrupt vector register or the timer vector register are implicitly addressed by the assertion of PC6/$\overline{PIACK}$ or PC7/$\overline{TIACK}$, respectively. The signals are first synchronized with the falling edge of the clock. One clock period after they are recognized, the data bus buffers are enabled and the vector is driven onto the bus. $\overline{CTACK}$ is asserted after another clock period to allow the vector some setup time prior to $\overline{DTACK}$. $\overline{DTACK}$ is negated, then three-stated, as with normal read or write cycles, when $\overline{PIACK}$ or $\overline{TIACK}$ is negated.

1.4.3 Write Cycles

In many ways, write cycles are similar to normal read cycles. On write cycles, data at the D0-D7 pins must meet the same setup specifications as the register select and R/$\overline{W}$ lines. Like these signals, write data is latched on the asserted edge of $\overline{CS}$, and must meet small setup and hold time requirements with respect to that edge. The same bus cycle recovery normally exist as for normal read cycles. No other differences exist.

NOTE

For mask sets G7J and K01 if the RS lines are selecting the port interrupt vector register (PIVR) or timer interrupt vector register (TIVR) during an interrupt acknowledge bus cycle then those registers may be changed. Four cases exist for this situation, they are:

1. Case — RS lines are addressing PIVR during a port interrupt acknowledge cycle (PIACK asserted)
   Results — Incorrect IACK vector on data lines, bits 0 and 1 are zero. PIVR and TIVR remain the same and are not changed
2. Case — RS lines are addressing TIVR during a port interrupt acknowledge cycle (PIACK asserted)
   Results — Incorrect IACK vector on data lines, PIVR and TIVR are changed
3. Case — RS lines are addressing PIVR during a timer interrupt acknowledge cycle (TIACK asserted)
   Results — Incorrect IACK vector on data lines, PIVR and TIVR are changed
4. Case — RS lines are addressing TIVR during a timer interrupt acknowledge cycle (TIACK asserted)
   Results — Correct IACK vector on data lines, PIVR and TIVR remain the same and are not changed

For MC68000 and MC68010 systems that use A1-A5 for RS lines RS1-RS5 the above cases will never occur. A5 and A4 will remain high during interrupt acknowledge cycles and thus PIVR and TIVR will not be selected as shown below.

	RS5	RS4	RS3	RS2	RS1
MC68000 IACK Cycle	1	1	— Encoded Level —		
MC68230 PIVR Address	0	0	1	0	1
MC68230 TIVR Address	1	0	0	0	1

SECTION 2
PORT GENERAL INFORMATION AND CONVENTIONS

This section introduces concepts that are generally applicable to the PI/T ports independent of the chosen mode and submode. For this reason, no particular port or handshake pins are mentioned; the notation H1:H3 indicates that, depending on the chosen mode and submode, the statement given may be true for either the H1 or H3 handshake pin.

2.1 UNIDIRECTIONAL VS BIDIRECTIONAL

Figure 2-2 shows the configuration of ports A and B and each of the handshake pins in each port mode and submode. In modes 0 and 1, a data direction register is associated with each of the ports. These registers contain one bit for each port pin to determine whether that pin is an input or an output. Modes 0 and 1 are, thus, called unidirectional modes because each pin assumes a constant direction changeable only by a reset condition or a programming change. These modes allow double-buffered data transfers in one direction. This direction, determined by the mode and submode definition, is known as the primary direction. Data transfers in the primary direction are controlled by the handshake pins. Data transfers not in the primary direction are generally unrelated, and single or unbuffered data paths exist.

In modes 2 and 3 there is no concept of primary direction as in modes 0 and 1. Except for port A in mode 2 (all I/O), the data direction registers have no effect. These modes are bidirectional, in that the direction of each transfer (always 8 or 16 bits, double buffered) is determined dynamically by the state of the handshake pins. Thus, for example, data may be transferred out of the ports, followed very shortly by a transfer into the same port pins. Transfers to and from the ports are independent and may occur in any sequence. Since the instantaneous direction is always determined by the external system, a small amount of arbitration logic may be required.

2.1.1 Control of Double-Buffered Data Ports

Generally speaking, the PI/T is a double buffered device. In the primary direction, double buffering allows orderly transfers by using the handshake pins in any of several programmable protocols. (When all I/O is used, double buffering is not available and the handshake pins are used for control or status/interrupt inputs.)

Use of double buffering is most beneficial in situations where a peripheral device and the computer system are capable of transferring data at roughly the same speed. Double buffering allows the high operation of the data transmitter to be overlapped with the slow operation of the data receiver. Thus, throughput measured in bytes or words-per-second may be greatly enhanced. If there is a large mismatch in transfer capability between the computer and the peripheral, little or no benefit is obtained. In these cases there is no penalty in using double buffering.

2.1.2 Double-Buffered Input Transfers

In all modes, the PI/T supports double-buffered input transfers. Data that meets the port setup and hold times is latched on the asserted edge of H1(H3). H1(H3) is edge sensitive, and may assume any duty cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H2S(H4S) status bit is set anytime any input data that has not been read by the bus master is present in the double buffered latches. The action of H2(H4) is programmable. It may indicate whether there is room for more data in the PI/T latches or it may serve other purposes. The following options are available, depending on the mode:

1. H2 (H4) may be an edge sensitive input that is independent of H1(H3) and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by the direct method (refer to **2.3 DIRECT METHOD OF RESETTING STATUS**), the $\overline{\text{RESET}}$ pulse, a write overrun, or when the H12 enable or H34 enable bit of the port general control register is zero.
2. H2(H4) may be a general purpose output pin that is always negated. The H2S (H4S) status bit is always zero.
3. H2 (H4) may be a general purpose output pin that is always asserted. The H2S(H4S) status bit is always zero.
4. H2(H4) may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H1(H3) input. As soon as the input latches become loaded, H2(H4) is again asserted. When both double buffered latches are full, H2(H4) remains negated until data is removed by a read of port A (port B) data register. Thus, anytime the H2(H4) output is asserted, new input data may be entered by asserting H1(H3). At other times, transitions of H1(H3) are ignored. The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2 (H4) is held negated.

5. H2(H4) may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H1(H3) asserted edge occurs before termination of the pulse, H2(H4) is negated asynchronously. Thus, anytime after the trailing edge of the H2(H4) pulse, new data may be entered into the PI/T double buffered input latches. The H2S(H4S) status bit is always zero. When H12 enable (H34 enable) is zero, H2(H4) is held negated.

A sample timing diagram is shown in Figure 2-1. The H2(H4) interlocked and pulse input handshake protocols are shown. The $\overline{\text{DMAREQ}}$ pin is also shown, assuming it is enabled. All handshake pin sense bits are assumed to be zero (refer to **4.1 PORT GENERAL CONTROL REGISTER (PGCR)**), thus the pins are in their low state when asserted. Due to the great similarity between modes, this timing diagram is applicable to all double-buffered input transfers.

2.1.3 Double-Buffered Output Transfers

The PI/T supports double-buffered output transfers in all modes. Data written by the bus master to the PI/T is stored in the port's output latch. This peripheral accepts the data by asserting H1(H3), which causes the next data to be moved to the port's output latch as soon as it is available. The function of H2(H4) is programmable. It may indicate whether new data has been moved to the output latch or it may serve other purposes. The H2S(H4S) status bit may be programmed for two interpretations. First, the status bit is a one when there is at least one latch in the double buffered data path that can accept new data. After writing one data word of data to the port, a second

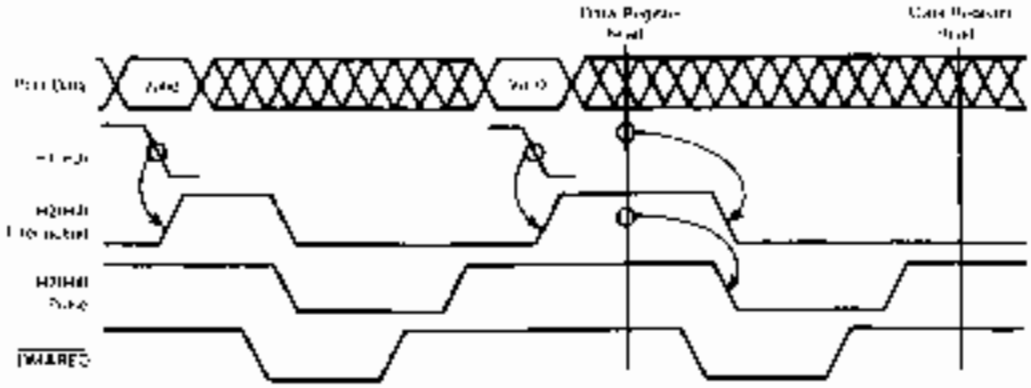


Figure 2-1. **Double-Buffered** Input Transfers Timing Diagram

service routine could check this bit to determine if it could store another byte/word, thus filling both latches. Second, when the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral. The IBF/OBF status bit is set when both output latches are empty. The programmable options of the H2/H4 pin are given below, depending on the mode.

1. H2/H4 may be an edge-sensitive input pin independent of H1/H3 and the transfer of port data. On the asserted edge of H2/H4, the H2S/H4S status bit is set. It is cleared by the direct method (refer to **2.3 DIRECT METHOD OF RESETTING STATUS**), the $\overline{\text{RESET}}$ pin being asserted, or when the H12 enable/H34 enable bit of the port general control register is zero.
2. H2/H4 may be a general purpose output pin that is always negated. The H2S/H4S status bit is always zero.
3. H2/H4 may be a general-purpose output pin that is always asserted. The H2S/H4S status bit is always zero.
4. H2/H4 may be an output pin in the interlocked output handshake protocol. H2/H4 is asserted two clock cycles after data is transferred to the double buffered output latches. The data remains stable and H2/H4 remains asserted until the next asserted edge of the H1/H3 input. At that time, H2/H4 is asynchronously negated. As soon as the next data is available, it is transferred to the output latches and H2/H4 is asserted. When H2/H4 is negated, asserted transitions on H1/H3 have no effect on the data paths. As is assumed also, however, in modes 2 and 3, H1 does control the three state output buffers of the bidirectional port(s). The H2S/H4S status bit is always zero. When H12 enable/H34 enable is zero, H2/H4 is held negated.
5. H2/H4 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked output protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1/H3 asserted edge occurs before termination of the pulse, H2/H4 is negated asynchronously, thus shortening the pulse. The H2S/H4S status bit is always zero. When H12 enable/H34 enable is zero, H2/H4 is held negated.

A sample timing diagram is shown in Figure 2-2. The H2(H4) interlocked and pulsed output handshake protocols are shown. The $\overline{DMARQ0}$ pin is also shown assuming it is enabled. All handshake pins must be programmed to be zero, thus, the pins are in the low state when asserted. Due to the great similarity between modes, this timing diagram is applicable to all double buffered output functions.

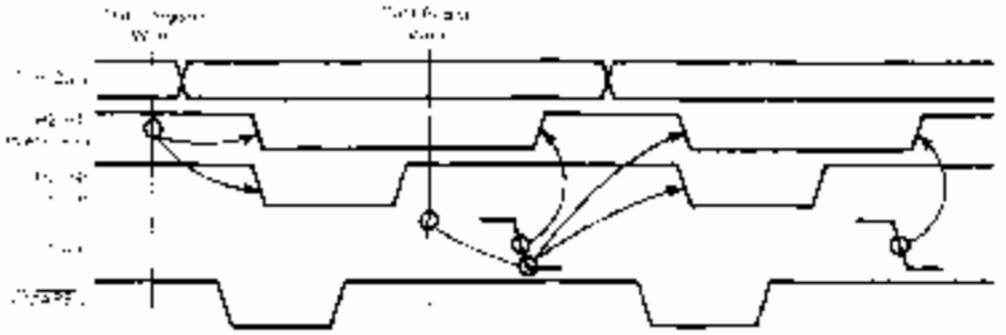


Figure 2-2. Double Buffered Output Transfers Timing Diagram

2.2 REQUESTING BUS MASTER SERVICE

The PI/T has several means of indicating a need for service by a bus master. First, the processor may poll the port status register. It contains a status bit for each handshake pin, plus a level bit that always reflects the instantaneous state of that handshake pin. A status bit is one when the PI/T needs servicing (i.e., generally when the bus master needs to read or write data to the port) or when a handshake pin used as a simple status input has been asserted. The interpretation of these bits is dependent on the chosen mode and submode.

Second, the PI/T may be placed in the processor's interrupt structure. As mentioned previously, the PI/T contains port A and B control registers that configure the handshake pins. Other bits in these registers enable an interrupt associated with each handshake pin. This interrupt is made available through the PC5/$\overline{PIRQ}$ pin, if the $\overline{PIRQ}$ function is selected. Three additional conditions are required for $\overline{PIRQ}$ to be asserted: 1) the handshake pin status bit is set, 2) the corresponding interrupt service request enable bit is set, and 3) DMA requests are not associated with that data transfer (H1 and H3 only). The conditions from each of the four handshake status bits and corresponding status bits are ORed to determine $\overline{PIRQ}$. To clear the interrupt, the proper status bit must be cleared. (See **2.3 DIRECT METHOD OF RESETTING STATUS**.)

The third method of requesting service is via the PC4/$\overline{DMARQ0}$ pin. This pin can be associated with double buffered transfers in each mode. It is used as a DMA controller request. It can initiate requests to keep the PI/T's input (output) double-buffering empty/full as much as possible. It will not overrun the DMA controller. The pin is compatible with the MC68450 direct memory access controller (DMAC).

2.2.1 Vectored, Prioritized Port Interrupts

Use of M68200-compatible vectored interrupts with the PI/T requires the $\overline{PIRQ}$ and $\overline{PIACK}$ pins. When $\overline{PIACK}$ is asserted while $\overline{PIRQ}$ is asserted, the PI/T places an 8-bit vector on the data pins D0-D7. Under normal conditions, this vector corresponds to the highest priority enabled active port interrupt source with which the $\overline{GMACSEC}$ pin is currently associated. The most significant six bits are provided by the port interrupt vector register (PIVR), with the lower two bits supplied by prioritization logic according to conditions present when $\overline{PIACK}$ is asserted. It is important to note that the only effect on the PI/T caused by interrupt acknowledge cycles is that the vector is placed on the data bus. Specifically, no registers, data, status, or other internal states of the PI/T are affected by the cycle.

Several conditions may be present when the $\overline{PIACK}$ input is asserted to the PI/T. These conditions affect the PI/T's response and the termination of the bus cycle. If the PI/T has no interrupt function selected, or is not arbitrating $\overline{PIRQ}$, the PI/T will make no response to $\overline{PIACK}$ ($\overline{DTACK}$ will not be asserted). If the PI/T is asserting $\overline{PIRQ}$ when $\overline{PIACK}$ is received, the PI/T will output the contents of the port interrupt vector register and the prioritization bits. If the PIVR has not been initialized, $0F will be read from this register. These conditions are summarized in Table 2-1.

Table 2-1 Response to Port Interrupt Acknowledge

Conditions	$\overline{PIRQ}$ negated OR interrupt request function not selected	$\overline{PIRQ}$ asserted
PIVR has not been initialized since $\overline{RESET}$	No response from PI/T; no $\overline{DTACK}$	PI/T provides $0F as the interrupt vector
PIVR has been initialized since $\overline{RESET}$	No response from PI/T; no $\overline{DTACK}$	PI/T provides PIVR contents with port priority bits

* Unless an autovector interrupt is also requested, in which case the autovector will be acknowledged.

The vector table entries for the PI/T appear as 4 contiguous blocks of four vector numbers whose common upper-six bits are programmed in the PIVR. The following table pairs each interrupt source with the 2-bit value provided by the prioritization logic when interrupt acknowledge is asserted (see 4.2. PORT SERVICE REQUEST REGISTER (PSRR)).

P1 source = 00	P0 source = 10
P2 source = 01	P3 source = 11

2.2.2 Autovectored Port Interrupts

Autovectored interrupts use only the $\overline{PIRQ}$ pin. The operation of the PI/T with vectored and autovectored interrupts is identical except that no vectors are supplied and the PC6/$\overline{PIACK}$ pin can be used as a port C pin.

2.2.3 DMA Request Operation

The direct memory access request ($\overline{DMAREQ}$) pin, when enabled, is associated with output or input transfers to keep the output and final output latches full or input and final input latch empty respectively. Figures 2-3 and 2-4 show all the possible paths in generating DMA requests. See **4.2 PORT SERVICE REQUEST REGISTER (PSRR)** for programming the operation of the DMA request pin.

$\overline{DMAREQ}$ is generated on the bus side of the MC68230 by the synchronized* chip select. If the conditions of Figures 2-3 or 2-4 are met, an assertion of $\overline{CS}$ will cause $\overline{DMAREQ}$ to be asserted three PI/T clocks plus the delay time from the clock edge after $\overline{CS}$ is synchronized. $\overline{DMAREQ}$ remains asserted three clock cycles plus the delay time from the clock edge) and is then negated.

$\overline{DMAREQ}$ pulses are associated with peripheral transfers or are generated by the synchronized* H1/H3 input. If the conditions of Figures 2-3 or 2-4 are met, an assertion of the H1/H3 input will cause $\overline{DMAREQ}$ to be asserted 2.5 PI/T clock cycles plus the delay time from clock edge) after H1/H3 is synchronized. $\overline{DMAREQ}$ remains asserted three clock cycles plus the delay time from the clock edge) and is then negated.

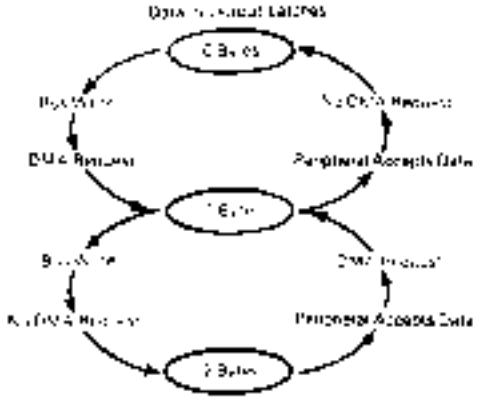


Figure 2-3. $\overline{DMAREQ}$ Associated with Output Transfers

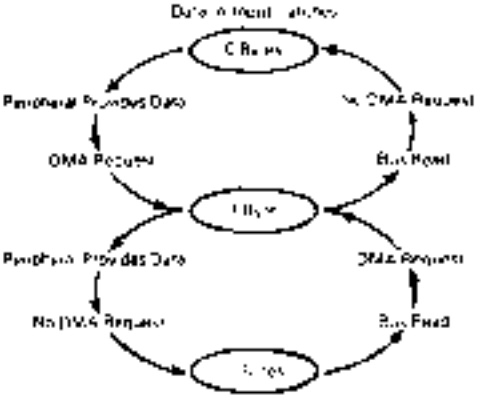


Figure 2-4. $\overline{DMAREQ}$ Associated with Input Transfers

2.3 DIRECT METHOD OF RESETTING STATUS

In certain modes one or more handshake pins can be used as edge sensitive inputs for the sole purpose of setting bits in the port status register. These bits consist of simple flip-flops. They are set to one by the occurrence of the asserted edge of the handshake pin input. Resetting a handshake status bit can be done by writing an 8-bit mask to the port status register. This is called the direct method of resetting. To reset a status bit that is resettable by the direct method, the mask must

*Synchronized means that the appropriate edge (read signal (R/$\overline{W}$), H1, or $\overline{CS}$) has been detected in the [illegible] of the appropriate clock (rising edge on H1/H3 and falling edge for $\overline{CS}$). Refer to **4.4 BUS INTERFACE OPERATION** for the exact conditions. If $\overline{CS}$ is a bus access function of $\overline{CS}$ and a port access assertion of H1/H3 occur at the same time, $\overline{CS}$ will be synchronized in the next cycle. H1/H3 will be synchronized one clock cycle later.

contain a one in the bit position of the port status register corresponding to the desired status bit. For status bits that are not resettable by the direct method in the chosen mode, the data written to the port status register has no effect. For status bits that are resettable by the direct method in the chosen mode, a zero in the mask has no effect.

2.4 HANDSHAKE PIN SENSE CONTROL

The PI/T contains exclusive-OR gates to control the sense of each of the handshake pins, whether used as inputs or outputs. Four bits in the port general control register may be programmed to determine whether the pins are asserted in the low- or high voltage state. As with other control registers, these bits are reset to zero when the $\overline{\text{RESET}}$ pin is asserted, defaulting the asserted level to be low.

2.5 ENABLING PORTS A AND B

Certain functions involved with double-buffered data transfers, the handshake pins, and the status bits may be disabled by the external system or by the programmer during initialization. The port general control register contains two bits, H12 enable and H34 enable, which control these functions. These bits are cleared to the zero state when the $\overline{\text{RESET}}$ pin is asserted, and the functions are disabled. The functions are the following:

1. Independent of other actions by the bus master or peripheral (via the handshake pins), the PI/T's disabled handshake controller is held to the "empty" state, i.e., no data is present in the double-buffered data path.
2. When any handshake pin is used to set a simple status flip-flop, unrelated to double-buffered transfers, these flip-flops are held reset to zero (see Table 1.1).
3. When H2/H4 is used in an interlocked or pulsed handshake with H1/H3, H2(-H4) is held negated, regardless of the chosen mode, submode, and primary direction. Thus, for double-buffered input transfers, the programmer may signal a peripheral when the PI/T is ready to begin transfers by setting the associated handshake enable bit to one.

2.6 PORT A AND B ALTERNATE REGISTERS

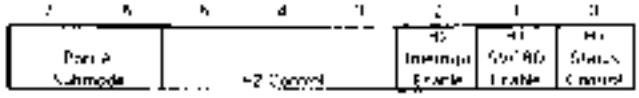
In addition to the port A and B data registers, the PI/T contains port A and B alternate registers. These registers are read only, and simply provide the instantaneous (non-latched) level of each port pin. They have no effect on the operation of the handshake pins, double-buffered transfers, status bits, or any other aspect of the PI/T, and they are mode/submode independent. Refer to **4.7 PORT ALTERNATE REGISTERS** for further information.

SECTION 3
PORT MODES

This section contains information that distinguishes the various port modes and submodes. General characteristics common to all modes are defined in **SECTION 2 PORT GENERAL INFORMATION AND CONVENTIONS**. A description of the port A control register (PACR) and port B control register (PBCR) is given before each mode description. After each submode description, the programmable options are listed for that submode.

3.1 PORT A CONTROL REGISTER

Port A Control Register (PACR)

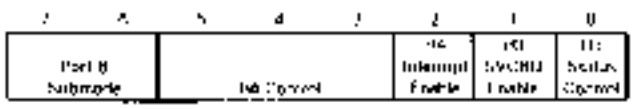


The port A control register, in conjunction with the programmed mode and the port B submode, controls the operation of port A and the handshake pins H1 and H2. The port A control register contains five fields: bits 7 and 6 specify the port A submode; bits 5, 4, and 3 control the operation of the H2 handshake pin and the H2S status bit; bit 2 determines whether an interrupt will be generated when the H2S status bit goes to one; bit 1 determines whether a service request (interrupt request or DMA request) will occur; bit 0 controls the operation of the H1S status bit. The PACR is always readable and writable.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port A submode field is relevant in a mode/submode definition, it must not be altered unless the H12 enable bit in the port general control register is clear (see Table 1-3 located on foldout pages 1 and 2 at the end of this document). Altering these bits will give unpredictable results.

3.2 PORT B CONTROL REGISTER

Port B Control Register (PBCR)



The port B control register specifies the operation of port B and the handshake pins H3 and H4. The port B control register contains five fields: bits 7 and 6 specify the port B submode; bits 5, 4, and 3 control the operation of the H4 handshake pin and H4S status bit; bit 2 determines whether an interrupt will be generated when the H4S status bit goes to a one; bit 1 determines whether a service request (interrupt request or DMA request) will occur; and bit 0 controls the operation of the H3S status bit. The PBCR is always readable and writable. There is never a consequence to reading the register.

All bits are cleared to zero when the $\overline{RESET}$* pin is asserted. When the port B submode field is relevant in a mode/submode definition, it must not be altered unless the H34 enable bit in the port general control register is clear (see Table 1-3 located on foldout pages 1 and 2 at the end of this document).

3.3 MODE 0 — UNIDIRECTIONAL 8-BIT MODE

In mode 0, ports A and B operate independently. Each may be configured in any of its three possible submodes:

Submode 00 — Pin Definable Double-Buffered Input or Single-Buffered Output
Submode 01 — Pin Definable Double-Buffered Output or Non-Latched Input
Submode 1X — Bit I/O (Pin Definable Single-Buffered Output or Non-Latched Input)

Handshake pins H1 and H2 are associated with port A and configured by programming the port A control register. (The H12 enable bit of the port general control register enables port A transfers.) Handshake pins H3 and H4 are associated with port B and configured by programming the port B control register. (The H34 enable bit of the port general control register enables port B transfers.) The port A and B data direction registers operate in all three submodes. Along with the submode, they affect the data read and write of the associated data register according to Table 3-1. They also enable the output buffer associated with each port pin. The $\overline{DMAREQ}$ pin may be associated with either (not both) port A or port B, but does not function if the bit I/O submode (submode 1X) is programmed for the chosen port.

Table 3-1. Mode 0 Port Data Paths

Mode	Read Port A/B Data Register		Write Port A/B Data Register	
	DDR = 0	DDR = 1	DDR = 1	
0 Submode 00	PIL, Q.B	FOL Note 3	PIR, S.R	Note 1
0 Submode 01	Pin	FOL Note 3	IOL/FOL, Q.B	Note 2
0 Submode 1X	Pin	FOL Note 3	FOL, S.B	Note 1

Abbreviations:
IOL = Initial Output Latch — S.B = Single Buffered
FOL = Final Output Latch — D.B = Double Buffered
PIL = Pre-Input Latch — DDR = Data Direction Register

Note 1: Data is latched in the output data registers during output strobe and will be single buffered at the end of the DDR = 1. The output buffers will be turned on if the DDR = 0.
Note 2: Data is latched in the double buffered output data registers. The data in the final output latch will appear on the port pins if the DDR = 1.
Note 3: The output drivers that connect the Pre-Input Latch to the pins are turned off.

3.3.1 Submode 00 — Pin Definable Double-Buffered Input or Single-Buffered Output

In mode 0, double-buffered input transfers of up to eight bits are available by programming submode 00 in the desired port's control register. Data that meets the port setup and hold times is latched on the asserted edge of H1 (H3) and is placed in the initial or final input latch. H1 (H3) is edge sensitive and may assume any duty cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H1S (H3S) status bit is set anytime any input data that has not been read by the bus master is present in the double-buffered latches. The action of H2/H4 is programmable. The following options are available:

1. H2(H4) may be an edge sensitive status input that is independent of H1(H3) and the transfers of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable (H34 enable) bit of the port general register is clear.
2. H2(H4) may be a general-purpose output pin that is always negated. In this case the H2S(H4S) status bit is always clear.
3. H2(H4) may be a general purpose output pin that is always asserted. In this case the H2S(H4S) status bit is always clear.
4. H2(H4) may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H1(H3) input. As soon as the input latches become ready, H2(H4) is again asserted. When the input double-buffered latches are full, H2(H4) remains negated until data is removed. Thus, anytime the H2(H4) output is asserted, new input data may be entered by asserting H1(H3). At other times, transitions on H1(H3) are ignored. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) in the port general control register is clear, H2(H4) is held negated.
5. H2(H4) may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case of a subsequent H1(H3) asserted edge occurring before termination of the pulse, H2(H4) is negated asynchronously. Thus, anytime after the leading edge of the H2(H4) pulse, new data may be entered in the double-buffered input latches. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear, H2(H4) is held negated.

For pins used as outputs, the data path consists of a single latch driving the output buffer. Data written to the port's data register does not affect the operation of any handshake pin or status bit. Output pins may be used independently of the input transfers. However, read bus cycles to the data register do remove data from the port. Therefore, care should be taken to avoid processor instructions that perform unwanted read cycles.

Programmable Options Mode 0 — Port A Submode 00 and Port B Submode 00 (Sheet 1 of 2)

PACR

7 6	Port A Submode
0 0	Submode 00

PACR

5 4 3	H2 Control
0 X X	Input pin — edge-sensitive status input. H2S is set on an asserted edge
1 0 0	Output pin — negated. H2S is always clear
1 0 1	Output pin — asserted. H2S is always clear
1 1 0	Output pin — interlocked input handshake protocol. H2S is always clear
1 1 1	Output pin — pulsed input handshake protocol. H2S is always clear

PACR

2	H2 Interrupt Enable
0	The H2 interrupt is disabled.
1	The H2 interrupt is enabled

Programmable Options Mode 0 — Port A Submode 00 and Port B Submode 00 (Sheet 2 of 2)

PACR
1 — H1 SVCR Enable
0 — The H1 interrupt and DMA requests are disabled
1 — The H1 interrupt and DMA request are enabled

PACR
0 — H1 Status Control
X — The H1S status bit is set anytime input data is present in the dual buffered input port

PBCR
7 6 — Port B Submode
0 0 — Submode 00

PBCR
5 4 3 — H4 Control
0 0 0 — Input pin — edge sensitive status input, H4S is set on an asserted edge
0 0 1 — Output pin — negated, H4S is always cleared
0 1 0 — Output pin — asserted, H4S is always cleared
0 1 1 — Output pin — interlocked input handshake protocol, H4S is always cleared
1 X X — Output pin — pulsed input handshake protocol, H4S is always cleared

PBCR
2 — H4 Interrupt Enable
0 — The H4 interrupt is disabled
1 — The H4 interrupt is enabled

PBCR
1 — H3 SVCRQ Enable
0 — The H3 interrupt and DMA request are disabled
1 — The H3 interrupt and DMA request are enabled

PBCR
0 — H3 Status Control
X — The H3S status bit is set anytime input data is present in the double-buffered input port.

3.3.2 Submode 01 - Pin-Definable Double-Buffered Output or Non-Latched Input

In mode 0, double-buffered output transfers of up to eight bits are available by programming submode 01 in the desired port's control register. The operation of H2 and H4 may be selected by programming the port A and B control registers, respectively. Data written by the bus master to the Port is stored in the port's output latches. The peripheral accepts the data by asserting H1(H3) which causes the next data to be moved to the port's output latch as soon as it is available.

The H1S(H3S) status bit may be programmed for two interpretations.

1. The H1S(H3S) status bit is set when either the port initial or final output latch can accept new data. It is cleared when both latches are full and cannot accept new data.
2. The H1S(H3S) status bit is set when both of the port output latches are empty. It is cleared when at least one latch is full.

The programmable options of the H2(H4) pin are:

1. H2(H4) may be an edge sensitive input pin independent of H1-H3 and the transfer of port data. On the asserted edge of H2(H4), the H2S(H4S) status bit is set. It is cleared by either the

RESET pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H1H2 enable (H3IH4I enable) bit of the port general control register is clear.

2. H2(H4) may be a general-purpose output pin that is always negated. The H2S(H4SI status bit is always clear.
3. H2(H4) may be a general-purpose output pin that is always asserted. The H2S(H4SI status bit is always clear.
4. H2(H4) may be an output pin in the interlocked output handshake protocol. H2(H4) is asserted two clock cycles after data is transferred to the double-buffered output latches. The data remains stable at the port pins and H2(H4) remains asserted until the next asserted edge of the H1(H3) input. At that time, H2(H4) is asynchronously negated. As soon as the next data is available, it is transferred to the output latches. When H2(H4) is negated, asserted transitions of H1(H3) have no affect on data paths. The H2S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear, H2(H4) is held negated.
5. H2(H4) may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1(H3) asserted edge occurs before termination of the pulse, H2(H4) is negated asynchronously shortening the pulse. The H3S(H4S) status bit is always clear. When H12 enable (H34 enable) is clear, H2(H4) is held negated.

For pins used as inputs, data written to the associated data register is double-buffered and passed to the input or final output latch, but, the output buffer is disabled.

Programmable Options Mode 0 — Port A Submode 01 and Port B Submode 01 (Sheet 1 of 2)

PACR

7 6 — Port A Submode

0 1 — Submode 01

PACR

5 4 3 — H2 Control

0 X X	Input pin	edge-sensitive status input. H2S is set on an asserted edge
1 0 0	Output pin	negated. H2S is always clear
1 0 1	Output pin	asserted. H2S is always clear
1 1 0	Output pin	interlocked input handshake protocol. H2S is always clear
1 1 1	Output pin	pulsed input handshake protocol. H2S is always clear

PACR

2 — H2 Interrupt Enable

0 — The H2 interrupt is disabled.
1 — The H2 interrupt is enabled.

PACR

1 — H1 SVCRQ Enable

0 — The H1 interrupt and DMA requests are disabled.
1 — The H1 interrupt and DMA requests are enabled.

Programmable Options Mode 0 — Port A Submode 01
and Port B Submode 01 (Sheet 2 of 2)

PACR
3 — H1 Status Control

0 — The H1S status bit is set when either the port A initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.

1 — The H1S status bit is set when both of the port A output latches are empty. It is clear when at least one latch is full.

PBCR
7 6 — Port B Submode

0 1 — Submode 01

PBCR
5 4 3 — H4 Control

0 X X — Input pin — edge sensitive status input. H4S is set on an asserted edge.
1 0 0 — Output pin — negated. H4S is always cleared.
1 0 1 — Output pin — asserted. H4S is always cleared.
1 1 0 — Output pin — interlocked input handshake protocol. H4S is always cleared.
1 1 1 — Output pin — pulsed input handshake protocol. H4S is always cleared.

PBCR
2 — H4 Interrupt Enable

0 — The H4 interrupt is disabled.
1 — The H4 interrupt is enabled.

PBCR
1 — H3 SVCRQ Enable

0 — The H3 interrupt and DMA request are disabled.
1 — The H3 interrupt and DMA request are enabled.

PBCR
0 — H3 Status Control

0 — The H3S status bit is set when either the port B initial or final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1 — The H3S status bit is set when both of the port B output latches are empty. It is clear when at least one latch is full.

3.3.3 Submode 1X — Bit I/O (Pin-Definable Single-Buffered Output or Non-Latched Input)

In mode 0, simple bit I/O is available by programming submode 1X in the desired port's control register. This submode is intended for applications in which several independent devices must be controlled or monitored. Data written to the associated input/output register is single buffered. If the data direction register bit for that pin is a one (output), the output buffer is enabled. If it is a zero (input), data written is still latched, but is not available at the pin. Data read from the data register is the instantaneous value of the pin or what was written to the data register, depending on the contents of the data direction register. H1/H3I is an edge-sensitive status input pin only and it controls no data related function. The H1S/H3S status bit is set following the asserted edge of the input waveform. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the associated status bit in the port status register (PSR), or when the H/2 enable (H34 enable) bit of the port general control register is clear. H2 may be programmed as:

1. H2/H4I may be an edge sensitive status input that is independent of H1/H3I and the transfer of port data. On the asserted edge of H2/H4I, the H2S/H4S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status

register (PSR), or when the H12 enable (H34 enable) bit of the port general control register is clear.

2. H2/H4 may be a general-purpose output pin that is always negated. In this case the H2S (H4S) status bit is always clear.

3. H2/H4 may be a general-purpose output pin that is always asserted. In this case the H2S (H4S) status bit is always clear.

Programmable Option Mode 0 — Port A Submode 1X and Port B Submode 1X (Sheet 1 of 2)

PACR
7 6 — **Port A Submode**
1 X — Submode 1X

PACR
5 4 3 — **H2 Control**
0 X X — Input pin — edge-sensitive status input. H2S is set on an asserted edge
1 X 0 — Output pin — negated. H2S is always cleared
1 X 1 — Output pin — asserted. H2S is always cleared

PACR
2 — **H2 Interrupt Enable**
0 — The H2 interrupt is disabled
1 — The H2 interrupt is enabled

PACR
1 — **H1 SVCRQ Enable**
0 — The H1 interrupt is disabled
1 — The H1 interrupt is enabled.

PACR
0 — **H1 Status Control**
X — H1 is an edge sensitive status input. H1S is set by an asserted edge of H1

PBCR
7 6 — **Port B Submode**
1 X — Submode 1X

PBCR
5 4 3 — **H4 Control**
0 X X — Input pin — edge-sensitive status input. H4S is set on an asserted edge
1 X 0 — Output pin — negated. H4S is always cleared
1 X 1 — Output pin — asserted. H4S is always cleared

PBCR
2 — **H4 Interrupt Enable**
0 — The H4 interrupt is disabled
1 — The H4 interrupt is enabled

PBCR
1 — **H3 SVCRQ Enable**
0 — The H3 interrupt is disabled
1 — The H3 interrupt is enabled

Programmable Options Mode 0 — Port A Submode 1X and Port B Submode 1X (Sheet 2 of 2)

PBCR	
0	H3 Status Control
X	H3 is an edge-sensitive status input; H2S is set by an asserted edge of H3

3.4 MODE 1 — UNIDIRECTIONAL 16-BIT MODE

In mode 1, ports A and B are concatenated to form a single 16-bit port. The port B submode field controls the configuration of both ports. The possible submodes are:

Port B Submode X0 — Pin-Definable Double-Buffered Input or Single-Buffered Output

Port B Submode X1 — Pin-Definable Double-Buffered Output or Non-Latched Input

Handshake pins H3 and H4, configured by programming the port B control register, are associated with the 16-bit double-buffered transfer. These 16-bit transfers are enabled by setting the H34 enable bit in the port general control register (PGCR). Handshake pins H1 and H2 may be used as simple status inputs not related to the 16-bit data transfer or H2 may be an output. Enabling of the H1 and H2 handshake pins is done by setting the H12 enable bit of the port general control register. The port A and B data direction registers operate in each submode. Along with the submode, they affect the data read and written at the data register according to Table 3-2. The data direction register also enables the output buffer associated with each port pin. The $\overline{SMAREQ}$ pin may be associated only with H3.

Table 3-2. Mode 1 Port Data Paths

Mode	Read Port A/B Register		Write Port A/B Register	
	DDR = 0	DDR = 1	DDR = 0	DDR = 1
1 Port B Submode X0	Port A/B	ICI Note 2	ICI, OLB Note 2	OLB Note 2
1 Port B Submode X1	Port A/B	ICI Note 2	ICI, OLB, ILB Note 3	OLB Note 1

Note 1: Data written to Port A goes to the intermediate latch. When the Port B data register is written, Port A data is transferred to the OLB/PUL.

Note 2: Data is latched in the control data register. The output latch does not change unless the DDR = 0.

Note 3: The output buffers are turned off (high-impedance) if the pin is not an output or

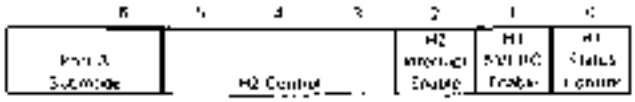
Abbreviations:

OLB	Port A Output Latch
ILB	Port A Input Latch
SB	Single Buffered
DB	Double Buffered
DDR	Data Direction Register

Mode 1 can provide convenient high-speed 16-bit transfers. The port A and port B data registers are addressed for compatibility with the MC68000 move peripheral (MOVEP) instruction and with the MC68450 direct memory access controller (DMAC). To take advantage of this, port A should contain the most-significant byte of data and always be read or written by the bus master first. The interlocked and pulsed handshake protocols, status bits, and $\overline{SMAREQ}$ are keyed to the access of port B data registers in mode 1. Transfers proceed properly with interlocked or pulsed handshakes when the port B data register is accessed last.

3.4.1 Port A Control Register

Port A Control Register (PACR)

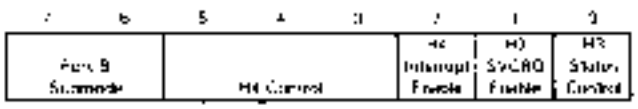


The port A control register, in conjunction with the programmed mode and the port B submode, controls the operation of port A and the handshake pins H1 and H2. The port A control register contains five fields: bits 7 and 6 specify the port A submode, bits 5, 4, and 3 control the operation of the H2 handshake pin and H2S status bit, bit 2 determines whether an interrupt will be generated when the H2S status bit goes to one, bit 1 determines whether a service request (interrupt request or DMA request) will occur, and bit 0 controls the operation of the H1S status bit. The PACR is always readable and writable. There is never a consequence to reading the register.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port A submode field is relevant in a mode/submode definition, it must not be altered unless the H12 enable bit in the port general control register is clear (see Table 1-3 located on foldout pages 1 and 2 at the end of this document). Altering these bits may give unpredictable results if the H12 enable bit in the PGCR is set.

3.4.2 Port B Control Register

Port B Control Register (PBCR)



The port B control register specifies the operation of port B and the handshake pins H3 and H4. The port B control register contains five fields: bits 7 and 6 specify the port B submode, bits 5, 4, and 3 control the operation of the H4 handshake pin and H4S status bit, bit 2 determines whether a service request (interrupt request or DMA request) will occur, and bit 0 controls the operation of the H3S status bit. The PBCR is always readable and writable.

All bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted. When the port B submode field is relevant in a mode/submode definition, it must not be altered unless the H34 enable bit in the port general control register is clear (see Table 1-3 located on foldout pages 1 and 2 at the end of this document). Altering these bits may give unpredictable results if the H12 enable bit in the PGCR is set.

3.4.3 Submode X0 — Pin-Definable Double-Buffered Input or Single-Buffered Output

In mode 1 submode X0, double-buffered input transfers of up to 16 bits may be obtained. The level of each pin is asynchronously latched with the asserted edge of H3 and placed in the initial input latch or the final input latch. The processor may check the H3S status bit to determine if new data is present. The $\overline{\text{DMAREQ}}$ pin may be used to signal a DMA controller to empty the input buffers. Regardless of the bus master, port A data should be read first and port B data should be read last. The operation of the internal handshake controller, the H3S bit, and the $\overline{\text{DMAREQ}}$ are keyed to the reading of the port B data register. The MC68450 DMAC can be programmed to perform the exact transfers needed for compatibility with the PI/T. H4 may be programmed as

1. H4 may be an edge-sensitive status input that is independent of H3 and the transfer of port data. On the asserted edge of H4, the H4S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H34 enable bit of the port general control register is clear.
2. H4 may be a general-purpose output pin that is always negated. In this case the H4S status bit is always clear.
3. H4 may be a general-purpose output pin that is always asserted. In this case the H4S status bit is always clear.
4. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H5 input. As soon as the input latches become ready, H4 is again asserted. When the input double buffered latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. At other times transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable of the port general control register is clear, H4 is held negated.
5. H4 may be an output pin in the pulsed-input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H4 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double buffered input latches. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.

For pins used as outputs, the data path consists of a single latch driving the output buffer. Data written to the port's data register does not affect the operation of any handshake pin, status bit, or any other aspect of the PI/T. Thus, output pins may be used independently of the input transfer.

The programmable options of the H2 pin are:

1. H2 may be an edge-sensitive input pin independent of H1 and the transfer of port data. On the asserted edge of H2, the H2S status bit is set. It is cleared by either the $\overline{\text{RESET}}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable bit of the port general control register is clear.
2. H2 may be a general-purpose output pin that is always negated. The H2S status bit is always clear.
3. H2 may be a general-purpose output pin that is always asserted. The H2S status bit is always clear.

Programmable Options Mode 1 - Port A Submode XX and Port B Submode X0

PACR
7 6 — Port A Submode
0 0 — Submode XX

PACR
5 4 3 — H2 Control
0 X X — Input pin — edge sensitive status input, H2S is set on an asserted edge
1 X 0 — Output pin — negated, H2S is always cleared
1 X 1 — Output pin — asserted, H2S is always cleared

PACR
2 — H2 Interrupt Enable
0 — The H2 interrupt is disabled
1 — The H2 interrupt is enabled

PACR
1 — H1 SVCRQ Enable
0 — The H1 interrupt is disabled
1 — The H1 interrupt is enabled

PACR
0 — H1 Status Control
X — H1 is an edge sensitive status input. H1S is set by an asserted edge of H1

PBCR
7 6 — Port B Submode
X 0 — Submode X0

PBCR
5 4 3 — H4 Control
0 X X — Input pin — edge-sensitive status input, H4S is set on an asserted edge
1 0 0 — Output pin — negated, H4S is always cleared
1 0 1 — Output pin — asserted, H4S is always cleared
1 1 0 — Output pin — interlocked input handshake protocol
1 1 1 — Output pin — pulsed input handshake protocol

PBCR
2 — H2 Interrupt Enable
0 — The H4 interrupt is disabled
1 — The H4 interrupt is enabled

PBCR
1 — H3 SVCRQ Enable
0 — The H3 interrupt and DMA requests are disabled
1 — The H3 interrupt and DMA requests are enabled

PBCR
0 — H3 Status Control
X — The H3S status bit is set anytime input data is present in the double buffered input path

3.4.4 Submode X1 — Pin Definable Double-Buffered Output or Non-Latched Input

In mode 1 submode X1, double-buffered output transfers of up to 16 bits may be obtained. Data is written by the bus master (processor or DMA controller) in two bytes. The first byte (most significant) is written to the port A data register. It is stored in a temporary latch until the next byte is written to the port B data register. Then all 16 bits are transferred to one of the output latches of ports A and B. The DMAREQ pin may be used to signal a DMA controller to transfer another word to the port output latches. (The MC68M450 DMAC can be programmed to perform the exact transfers needed for compatibility with the PI/T.) H4 may be programmed as:

1. H4 may be an input sensitive to the status of the port, that is independent of H3 and the transfer of port data. On the asserted edge of H4, the H4S status bit is set. It is cleared by either the $\overline{RESET}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H34 enable bit of the port general control register is clear.
2. H4 may be a general-purpose output pin that is always negated. In this case the H4S status bit is always clear.
3. H4 may be a general-purpose output pin that is always asserted. In this case the H4S status bit is always clear.
4. H4 may be an output pin in the interlocked output handshake protocol. H4 is asserted two clock cycles after data is transferred to the double-buffered output latches. The data remains stable at the port pins and H4 remains asserted until the next asserted edge of the H3 input. At that time, H4 is asynchronously negated. As soon as the next data is available, it is transferred to the output latches. When H4 is negated, asserted transitions of H3 have no effect on both paths. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.
5. H4 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked protocol above, but will never remain asserted longer than four clock cycles. Therefore, a four clock pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously shortening the pulse. The H4S status bit is always cleared. When H34 enable is clear, H4 is held negated.

The H3S status bit may be programmed for two interpretations:

1. The H3S status bit is set when either the port A or B output latch can accept new data. It is clear when both latches are full and cannot accept new data.
2. The H3S status bit is set when both of the port output latches are empty. It is clear when at least one latch is full.

The programmable options of the H2 pin are:

1. H2 may be an edge sensitive input pin independent of H1 and the transfer of port data. On the asserted edge of H2, the H2S status bit is set. It is cleared by either the $\overline{RESET}$ pin being asserted, writing a one to the particular status bit in the port status register (PSR), or when the H12 enable bit of the port general control register is clear.
2. H2 may be a general-purpose output pin that is always negated. The H2S status bit is always clear.
3. H2 may be a general purpose output pin that is always asserted. The H2S status bit is always clear.

For pins used as inputs, data written to either data register is double buffered and passed to the latch, or final output latch, as usual, but the output buffer is disabled (refer to **3.3.2 Submode 01 – Pin-Definable Double-Buffered Output or Non-Latched Input**).

Programmable Options Mode 1 Port A Submode XX and Port B Submode X1 (Sheet 1 of 2)

PACR
7 6 — **Port A Submode**
0 0 — Submode XX

PACR
5 4 3 — **H2 Control**
0 X X — Input pin — edge-sensitive status input, H2S is set on an asserted edge.
1 X 0 — Output pin — negated, H2S is always cleared
1 X 1 — Output pin — asserted, H2S is always cleared

PACR
2 — **H2 Interrupt Enable**
0 — The H2 interrupt is disabled
1 — The H2 interrupt is enabled

PACR
1 — **H1 $VCRQ Enable**
0 — The H1 interrupt is disabled
1 — The H1 interrupt is enabled.

PACR
0 — **H1 Status Control**
X — H1 is an edge sensitive status input. H1S is set by an asserted edge of H1.

PBCR
7 6 — **Port B Submode**
0 0 — Submode X1

PBCR
5 4 3 — **H4 Control**
0 X X — Input pin — edge sensitive status input, H4S is set on an asserted edge
1 0 0 — Output pin — negated, H4S is always cleared
1 0 1 — Output pin — asserted, H4S is always cleared
1 1 0 — Output pin — interlocked input handshake protocol
1 1 1 — Output pin — pulsed input handshake protocol

PBCR
2 — **H4 Interrupt Enable**
0 — The H4 interrupt is disabled
1 — The H4 interrupt is enabled

Programmable Options Mode 1 — Port A Submode XX
and Port B Submode X1 (Sheet 2 of 2)

PBCR

1	H3 SVCRQ Enable
0	The H3 interrupt and DMA request are disabled
1	The H3 interrupt and DMA request are enabled

PBCR

0	H3 Status Control
0	The H3S status bit is set when either the initial or final output latch of ports A and B can accept new data. It is clear when both latches are full and cannot accept new data.
1	The H3S status bit is set when both the initial and final output latches of ports A and B are empty. The H3S status bit is clear when at least one set of output latches is full.

3.5 MODE 2 — BIDIRECTIONAL 8 BIT MODE

In mode 2, port A is used for bi-directional I/O with no associated handshake pins. Port B is used for bidirectional 8-bit double buffered transfers. H1 and H2, enabled by the H12 enable bit in the port general control register, control output transfers, while H3 and H4, enabled by the port general control register bit H34 enable, control input transfers. The instantaneous direction of the data is determined by the H1 handshake pin. The port B data direction register is not used. The port A and port B submode fields do not affect PI/T operation in mode 2.

3.5.1 Port A 8-bit I/O (Pin-Definable Single-Buffered Output or Non-Latched Input)

Mode 2, port A performs simple 8-bit I/O with no associated handshake pins. This configuration is intended for applications in which several independent devices must be controlled or monitored. Data written to the port A data register is single buffered. If the port A data direction register bit for that pin is set (output), the output buffer is enabled. If it is zero (input), data written is still latched but not available at the pin. Data read from the data register is either the instantaneous value of the pin (if data is strobed from CS asserted to DTACK asserted, data on these pins will be guaranteed valid in the data register or what was written to the data register, depending on the contents of the port A data direction register. This is summarized in Table 3-3.

Table 3-3. Mode 2 Port A Data Paths

Mode	Read Port A Data Register		Write Port A Data Register	
	DDR = 0	DDR = 1	DDR = 0	DDR = 1
2	Pins	FEC	FEC	FEC

Abbreviations:
DDR — Port A Data Direction Register
FEC — Port A Data Register
Pins — Input Pins

3.5.2.1 **DOUBLE-BUFFERED INPUT TRANSFERS.** Port B input data (Port mode, the port setup and hold times is latched on the asserted edge of H3 and placed in the initial input latch. H3's edge valid true, and may assume any duty-cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose IBF status bit is set and indicates input data that has not been read by the bus master is present in the double buffered latches. The action of H4 is programmable and can be programmed as:

1. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H3 input. As soon as the input latches become ready, H4 is again asserted. When the input double-buffering latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. At other times transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable in the port general control register is clear, H4 is held negated.
2. H4 may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double buffered input latches. The H4S status bit is always clear. When H34 enable is clear, H4 is held negated.

3.5.2.2 **DOUBLE-BUFFERED OUTPUT TRANSFERS.** Data written by the bus master to the PI/T is stored in the port's output latch. The peripheral accepts the data by asserting H1, which causes the next data to be removed to the port's output latch as soon as it is available. The H1S status bit in the port status register may be programmed for two interpretations. Normally the status bit is a one when there is at least one latch in the double-buffered data path that can accept new data. After writing one byte of data to the port, an interrupt service routine could check this bit to determine if it could store another byte, thus filling both latches. When the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral. The H1S status control bit of the port A control register provides this flexibility. The H1S status bit is set when both output latches are empty. The programmable options for H2 are:

1. H2 may be an output pin in the interlocked output handshake protocol. It is asserted when the port output latches are ready to transfer new data. It is negated asynchronously following the asserted edge of the H1 input. As soon as the output latches become ready, H2 is again asserted. When the output double-buffered latches are full, H2 remains asserted until data is removed. Thus, anytime the H2 output is asserted, new output data may be transferred by asserting H1. At other times transitions on H1 are ignored. The H2S status bit is always clear. When H12 enable in the port general control register is clear, H2 is held negated.
2. H2 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H1 asserted edge occurs before termination of the pulse, H2 is negated asynchronously. Thus, anytime after the leading edge of the H2 pulse, new data may be transferred to the double-buffered output latches. The H2S status bit is always clear. When H12 enable is clear, H2 is held negated.

The DMAREQ pin may be associated with either input transfers (H3) or output transfers (H1), but not both. Refer to Table 3-4 for a summary of the port B data register responses in mode 2.

Table 3-4. Mode 2 Port B Data Paths

Mode	Read Port B Data Register	Write Port B Data Register
2	[illegible]	[illegible]
[illegible]	D P	Double B mode

Programmable Options Mode 2 - Port A Submode XX and Port B Submode XX (Sheet 1 of 2)

PACR
7 6 Port A Submode
X X [illegible]

PACR
5 4 3 H2 Control
X X 0 Output pin — interlocked output handshake protocol. H2S is always cleared
X X 1 Output pin — pulsed output handshake protocol. H2S is always cleared

PACR
2 H2 Interrupt Enable
0 The H2 interrupt is disabled
1 The H2 interrupt is enabled

PACR
1 H1 SVCRQ Enable
0 The H1 interrupt and SVA request are disabled
1 The H1 interrupt and SVA request are enabled

PACR
0 H1 Status Control
0 The H1 status bit is set when either the port B input or the output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1 The H1S status bit is set when both of the port B output latches are empty. It is clear when at least one latch is full.

PBCR
7 6 Port B Submode
X X Submode XX

PBCR
5 4 3 H4 Control
X X 0 Output pin — interlocked input handshake protocol. H4S is always cleared
X X 1 Output pin — pulsed input handshake protocol. H4S is always cleared

PBCR
2 H4 Interrupt Enable
0 The H4 interrupt is disabled
1 The H4 interrupt is enabled

Programmable Options Mode 2 — Port A Submode XX and Port B Submode XX (Sheet 2 of 2)

PBCR	
1	H3 SVCRQ Enable
0	The H3 interrupt and DMA request are disabled
1	The H3 interrupt and DMA request are enabled

PBCR	
0	H3 Status Control
X	The H3S status bit is set anytime input data is present in the double-buffered input path

3.6 MODE 3 — BIDIRECTIONAL 16-BIT MODE

In mode 3, ports A and B are used for bidirectional 16 bit double buffered transfers. H1 and H2 control output transfers, while H3 and H4 control input transfers. H1 and H2 are enabled by the H12 enable bit while H3 and H4 are enabled by the H34 enable bit of the port general control register. The instantaneous direction of data is determined by the H1 handshake pin. Thus, the data direction registers are not used and have no affect. The port A and port B submode fields do not affect PI/T operation in mode 3. Port A and port B output buffers are controlled by the level of H1. When H1 is negated, the output buffers (A 16) are enabled and the pins drive the bidirectional port bus. Generally a peripheral will negate H1 in response to an asserted H2, which indicates that new output data is present in the double-buffered latches. Following acceptance of the data, the peripheral asserts H1, disabling the output buffers. Other than controlling the output buffers, H1 is edge sensitive as in other modes. The port A and port B data direction registers are not used.

3.6.1 Double-Buffered Input Transfers

Port A and B input data that meets the port setup and hold times is latched on the asserted edge of H3 and placed in the initial input latch or the final input latch. H3 is edge-sensitive, and may assume any duty-cycle as long as both high and low minimum times are observed. The PI/T contains a port status register whose H3S status bit is set anytime any input data is present in the double buffered latches (not has not been read by the bus master). The action of H4 is programmable and can be programmed as:

1. H4 may be an output pin in the interlocked input handshake protocol. It is asserted when the port input latches are ready to accept new data. It is negated asynchronously following the asserted edge of the H3 input. As soon as the input latches become ready, H4 is again asserted. When the input double-buffered latches are full, H4 remains negated until data is removed. Thus, anytime the H4 output is asserted, new input data may be entered by asserting H3. All other transitions on H3 are ignored. The H4S status bit is always clear. When H34 enable in the port general control register is clear, H4 is held negated.
2. H4 may be an output pin in the pulsed input handshake protocol. It is asserted exactly as in the interlocked input protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock cycle pulse is generated. But in the case that a subsequent H3 asserted edge occurs before termination of the pulse, H4 is negated asynchronously. Thus, anytime after the leading edge of the H4 pulse, new data may be entered in the double-buffered input latches. The H4 status bit is always clear. When H34 enable is clear, H4 is held negated.

3.6.2 Double-Buffered Output Transfers

Data written by the bus master to the PI/T, is stored in the port's output latch. The peripheral accepts the data by asserting H1, which causes the new data to be moved to the port's output latch as soon as it is available. The H1S status bit in the port status register may be programmed for two interpretations. Normally the status bit is a one when there is at least one latch in the double-buffered data path that can accept new data. After writing one byte of data to the ports, an interrupt service routine could check this bit to see if it could store another byte, thus filling both latches. When the bus master is finished, it is often useful to be able to check whether all of the data has been transferred to the peripheral. The H1S status control bit of the port A control register provides this flexibility. The H1S status bit is set when both output latches are empty. The programmable options for H2 are:

1. H2 may be an output pin in the interlocked output handshake protocol. It is asserted when the port output latches are ready to transfer new data. It is negated asynchronously following the asserted edge of the H1 input. As soon as the output latches become ready, H2 is again asserted. When the output double-buffered latches are full, H2 remains asserted until data is removed. Thus, anytime the H2 output is asserted, new output data may be transferred by asserting H1. At other times transitions on H1 are ignored. The H2S status bit is always clear. When H12 enable in the port general control register is clear, H2 is held negated.
2. H2 may be an output pin in the pulsed output handshake protocol. It is asserted exactly as in the interlocked output protocol above, but never remains asserted longer than four clock cycles. Typically, a four clock pulse is generated. But in the case that a subsequent H1 asserted edge occurs before termination of the pulse, H2 is negated asynchronously shortening the pulse. The H2S status bit is always zero. When H12 enable is zero, H2 is held negated.

When 3 can provide convenient high speed 16-bit transfers. The port A and B data registers are addressed for compatibility with the MC68000's move peripheral (MOVEP) instruction and with the MC68450 DMAC. To take advantage of this, port A should contain the most significant data and always be read or written by the bus master first. The interlocked and pulsed handshake protocols, status bits, and $\overline{DMAREQ}$ are keyed to the access of port B data register in mode 3. If it is accessed last, the 16-bit double buffered transfer proceeds smoothly.

The $\overline{DMAREQ}$ pin may be associated with either input transfers (H1) or output transfers (H2), but not both. Refer to Table 3-5 for a summary of the port A and B data paths in mode 3.

Table 3-5. Mode 3 Port A and B Data Paths

Mode	Read Port A and B Data Register	Write Port A and B Data Register
3	[illegible] D0	[illegible] D0 Note

Note 1. Data written to Port B goes to the peripheral latch. While the Port B data register is being written, Port A data is transferred to [illegible]

Abbreviations:
[illegible] = Input Data Register
[illegible] = Input Latch
[illegible] = Output Data Latch
[illegible] = Output Latch
S = Single Buffered
D = Double Buffered

Programmable Options Mode 3 — Port A Submode XX
and Port B Submode XX

PACR
7 6 **Port A Submode**
X X Submode XX

PACR
5 4 3 **H2 Control**
X X 0 Output pin = interlocked output handshake protocol, H2S status always cleared
X X 1 Output pin = pulsed output handshake protocol, H2S status always cleared

PACR
2 **H2 Interrupt Enable**
0 The H2 interrupt is disabled
1 The H2 interrupt is enabled

PACR
1 **H1 SVCRQ Enable**
0 The H1 interrupt and DMA requests are disabled
1 The H1 interrupt and DMA requests are enabled

PACR
0 **H1 Status Control**
0 The H1 status bit is set whenever the port B output final output latch can accept new data. It is clear when both latches are full and cannot accept new data.
1 The H1S status bit is set when both of the port B output latches are empty. It is clear when at least one latch is full.

PBCR
7 6 **Port B Submode**
X X Submode XX

PBCR
5 4 3 **H4 Control**
X X 0 Output pin = interlocked input handshake protocol, H4S is always clear
X X 1 Output pin = pulsed input handshake, H4S is always clear

PBCR
2 **H4 Interrupt Enable**
0 The H4 interrupt is disabled
1 The H4 interrupt is enabled

PBCR
1 **H3 SVCRQ Enable**
0 The H3 interrupt and DMA request are disabled
1 The H3 interrupt and DMA request are enabled

PBCR
0 **H3 Status Control**
X The H3S status bit is set anytime input data is present in the double buffered input path

SECTION 4
PROGRAMMER'S MODEL

This section describes the internal accessible register organization as represented in Table 1-3 located on foldout pages 1 and 2 at the end of this document and in Table 4-1. Address space within the address map is reserved for future expansion.

Table 4-1. PI/T Register Addressing Assignments

Register		Register Select Bits					Accessible	Affected by Reset	Affected by Read Cycle
		5	4	3	2	1			
Port General Control Register	(PGCR)	0	0	0	0	0	R/W	Yes	No
Port Service Request Register	(PSRR)	0	0	0	0	1	R/W	Yes	No
Port A Data Direction Register	(PADDR)	0	0	0	1	0	R/W	Yes	No
Port B Data Direction Register	(PBDDR)	0	0	0	1	1	R/W	Yes	No
Port C Data Direction Register	(PCDDR)	0	0	1	0	0	R/W	Yes	No
Port Interrupt Vector Register	(PIVR)	0	0	1	0	1	R/W	Yes	No
Port A Control Register	(PACR)	0	0	1	1	0	R/W	Yes	No
Port B Control Register	(PBCR)	0	0	1	1	1	R/W	Yes	No
Port A Data Register	(PADR)	0	1	0	0	0	R/W	No	Yes
Port B Data Register	(PBDR)	0	1	0	0	1	R/W	No	Yes
Port A Alternate Register	(PAAR)	0	1	0	1	0	R	No	No
Port B Alternate Register	(PBAR)	0	1	0	1	1	R	No	No
Port C Data Register	(PCDR)	0	1	1	0	0	R/W	No	No
Port Status Register	(PSR)	0	1	1	0	1	R/W*	Yes	Yes
Timer Control Register	(TCR)	1	0	0	0	0	R/W	Yes	No
Timer Interrupt Vector Register	(TIVR)	1	0	0	0	1	R/W	Yes	No
Counter Preload Register High	(CPRH)	1	0	0	1	0	R/W	Yes	No
Counter Preload Register Middle	(CPRM)	1	0	0	1	1	R/W	Yes	No
Counter Preload Register Low	(CPRL)	1	0	1	0	0	R/W	Yes	No
Count Register High	(CNTRH)	1	0	1	0	1	R	Yes	No
Count Register Middle	(CNTRM)	1	0	1	1	0	R	Yes	No
Count Register Low	(CNTRL)	1	0	1	1	1	R	Yes	No
Timer Status Register	(TSR)	1	1	0	1	0	R/W*	Yes	No

* A write to this register may perform a special status resetting operation.
** Mode dependent.

R = Read
W = Write

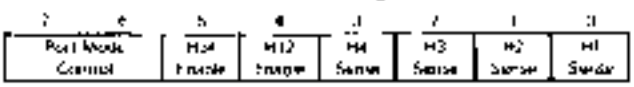
Throughout this section the following conventions are maintained:

1. A read from a reserved location in the map results in a read from the "null register." The null register returns all zeros for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle, but written data is ignored.
2. Unused bits of a defined register are denoted by "0" and are read as zeros, unless data is altered.

3. Bits that are unused in the chosen mode/submode but are used in others are denoted by "X", and are readable and writable. Their content, however, is ignored in the chosen mode/submode.
4. All registers are addressable as 8-bit quantities. To facilitate operation with the MOVEP instruction and the DMAC, addresses are ordered such that certain sets of registers may also be accessed as words (two bytes) or long words (four bytes).

4.1 PORT GENERAL CONTROL REGISTER (PGCR)

Port General Control Register (PGCR)



The port general control register controls many of the functions that are common to the overall operation of the ports. The PGCR is composed of three major fields: bits 7 and 6 define the operational mode of ports 4 and 5 and affect operation of the handshake pins and status bits, bits 5 and 4 allow a software-controlled disabling of particular hardware associated with the handshake pins of each port, and bits 3-0 define the sense of the handshake pins. The PGCR is always readable and writable.

All bits are reset to zero when the $\overline{\text{RESET}}$ pin is asserted.

The port mode control field should be altered only when the H12 enable and H34 enable bits are zero. Except when mode zero is desired (submode 0X), the port general control register should be written once to establish the mode with the H12 and H34 bits clear. Any other necessary control registers can then be programmed, after which H12 or H34 is set. In order to enable the respective operation(s), the port general control register should be written again.

PGCR
7-6 **Port Mode Control**

0 0 Mode 0: Unidirectional 8-Bit Mode
0 1 Mode 1: Unidirectional 16-Bit Mode
1 0 Mode 2: Bidirectional 8-Bit Mode
1 1 Mode 3: Bidirectional 16-Bit Mode

PGCR
5 **H34 Enable**

0 Disabled
1 Enabled

PGCR
4 **H12 Enable**

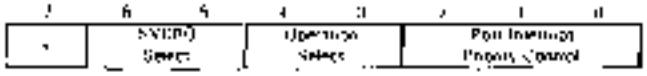
0 Disabled
1 Enabled

PGCR
3-0 **Handshake Pin Sense**

0 The associated pin is at the high-voltage level when negated and at the low-voltage level when asserted.
1 The associated pin is at the low-voltage level when negated and at the high-voltage level when asserted.

4.2 PORT SERVICE REQUEST REGISTER (PSRR)

Port Service Request Register (PSRR)



The port service request register controls other functions that are common to the overall operation of the ports. It is composed of four major fields. Bit 7 is unused and is always read as zero, bits 6 and 5 define whether interrupt or DMA requests are generated from activity on the H1 and H3 handshake pins, bits 4 and 3 determine whether two dual function pins operate as port C or port interrupt request/acknowledge pins, and bits 2, 1, and 0 control the priority among all port interrupt sources. Since bits 2, 1, and 0 affect interrupt operation, it is recommended that they be changed only when the affected interrupt(s) is (are) disabled or known to remain inactive. The PSRR is always readable and writable.

All bits are reset to zero when the $\overline{RESET}$ pin is asserted.

PSRR
6 5 — SVCRQ Select

0 0 — The PC4/$\overline{DMAREQ}$ pin carries the PC4 function, DMA is not used.

PSRR — SVCRQ Select

1 1 — The PC4/$\overline{DMAREQ}$ pin carries the $\overline{DMAREQ}$ function and is associated with double-buffered transfers controlled by H1. H1 is removed from the PI/T's interrupt structure, and thus, does not cause interrupt requests to be generated. To obtain $\overline{DMAREQ}$ pulses, port A control register bit 1 (H1 SVCRQ enable) must be a one.

1 0 — The PC4/$\overline{DMAREQ}$ pin carries the $\overline{DMAREQ}$ function and is associated with double-buffered transfers controlled by H3. H3 is removed from the PI/T's interrupt structure, and thus, does not cause interrupt requests to be generated. To obtain $\overline{DMAREQ}$ pulses, port B control register bit 1 (H3 SVCRQ enable) must be one.

PSRR
4 3 — Interrupt Pin Function Select

0 0 — The PC5/$\overline{PIRQ}$ pin carries the PC5 function, no interrupt support.
The PC6/$\overline{PIACK}$ pin carries the PC6 function, no interrupt support.

0 1 — The PC5/$\overline{PIRQ}$ pin carries the $\overline{PIRQ}$ function, supports autovectored interrupts.
The PC6/$\overline{PIACK}$ pin carries the PC6 function, supports autovectored interrupts.

1 0 — The PC5/$\overline{PIRQ}$ pin carries the PC5 function.
The PC6/$\overline{PIACK}$ pin carries the $\overline{PIACK}$ function.

1 1 — The PC5/$\overline{PIRQ}$ pin carries the $\overline{PIRQ}$ function, supports vectored interrupts.
The PC6/$\overline{PIACK}$ pin carries the $\overline{PIACK}$ function, supports vectored interrupts.

Bits 2, 1, and 0 determine port interrupt priority. The priority as shown in Table 4-2 is in descending order left to right.

Table 4-2. PSRR Port Interrupt Priority Control

2 1 0	Highest			Lowest
0 0 0	H3S	H3S	H1S	H1S
0 0 1	H3S	H3S	H2S	H1S
0 1 0	H3S	H2S	H1S	H3S
0 1 1	H2S	H1S	H3S	H3S

2 1 0	Highest			Lowest
1 0 0	H3S	H1S	H2S	H2S
1 0 1	H2S	H3S	H1S	H1S
1 1 0	H1S	H2S	H3S	H3S
1 1 1	H1S	H2S	H3S	H3S

4.3 PORT DATA DIRECTION REGISTERS

The following paragraphs describe the port data direction registers.

4.3.1 Port A Data Direction Register (PADDR)

The port A data direction register determines the direction and buffering characteristics of each of the port A pins. One bit in the PADDR is assigned to each pin. A zero bit indicates that the pin is used as a input, while a one indicates it is used as an output. The PADDR is always readable and writable. This register is ignored in mode 3.

All bits are reset to the zero (input) state when the $\overline{\text{RESET}}$ pin is asserted.

4.3.2 Port B Data Direction Register (PBDDR)

The PBDDR is identical to the PADDR for the port B pins and the port B data register, except that this register is ignored in modes 2 and 3.

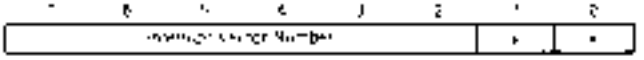
4.3.3 Port C Data Direction Register (PCDDR)

The port C data direction register specifies whether each dual-function pin that is chosen for port C operation is an input (zero) or an output (one) pin. The PCDDR, along with bits that determine the respective pin's function, also specify the exact hardware to be accessed at the port C data register address. See **4.6.3 Port C Data Register (PCDR)** for more details. The PCDDR is an 8-bit register that is readable and writable in all modes. Its operation is independent of the chosen PI/T mode.

These bits are cleared to zero when the $\overline{\text{RESET}}$ pin is asserted.

4.4 PORT INTERRUPT VECTOR REGISTER (PIVR)

Port Interrupt Vector Register (PIVR)



The port interrupt vector register contains the upper order six bits of the four port interrupt vectors. The contents of this register may be read two ways: by an ordinary read cycle, or by a port interrupt acknowledge bus cycle. The exact data read depends on how the cycle was terminated and other factors. Behavior during a port interrupt acknowledge cycle is summarized in Table 2-1.

From a normal read cycle, there is never a consequence to reading this register. Following negation of the $\overline{\text{RESET}}$ pin, but prior to writing to the PIVR, a $0F will be read. After writing to the register, the upper six bits may be read and the lower two bits are forced to zero. No prioritization computation is performed.

4.5 PORT CONTROL REGISTERS (PACR, PBCR)

The port A and B control registers (PACR and PBCR) are described in **SECTION 3 PORT MODES**. The description is organized such that for each mode/submode all programmable options of each pin and status bit are given.

4.6 PORT DATA REGISTERS

The following paragraphs describe the port data registers.

4.6.1 Port A Data Register (PADR)

The port A data register is a holding register for moving data to and from the port A pins. The port A data direction register determines whether each pin is an input (zero) or an output (one), and is used in configuring the actual data paths. The data paths are described in **SECTION 3 PORT MODES**.

This register is readable and writable at all times. Depending on the chosen mode/submode, reading or writing may affect the double-buffered handshake mechanism. The port A data register is not affected by the assertion of the $\overline{\text{RESET}}$ pin.

4.6.2 Port B Data Register (PBDR)

The port B data register is a holding register for moving data to and from port B pins. The port B data direction register determines whether each pin is an input (zero) or an output (one), and is used in configuring the actual data paths. The data paths are described in **SECTION 3 PORT MODES**.

This register is readable and writable at all times. Depending on the chosen mode/submode, reading or writing may affect the double-buffered handshake mechanism. The port B data register is not affected by the assertion of the $\overline{\text{RESET}}$ pin.

4.6.3 Port C Data Register (PCDR)

The port C data register is a holding register for moving data to and from each of the eight port C/alternate function pins. The exact hardware accessed is determined by the type of bus cycle (read or write) and individual conditions affecting each pin. These conditions are: 1) whether the pin is used for the port C or alternate function, and 2) whether the port C data direction register indicates the input or output direction. The port C data register is single buffered for output pins and non-latched for input pins. These conditions are summarized in Table 4-3.

Table 4-3. PCDR Hardware Accesses

Operation	Port C Function		Alternate Function	
	PCDDR = 0	PCDDR = 1	PCDDR = 0	PCDDR = 1
Read Port C Data Register	Pin	Output Register	Pin	Output Register
Write Port C Data Register	Output Register Buffer Cleared	Output Register, Buffer Enabled	Output Register	Output Register

Note that two additional useful benefits result from this structure. First, it is possible to directly read the state of a dual-function pin while used for the non-port C function. Second, it is possible to generate program-controlled transitions on alternate function pins by switching back to the port C function and writing to the PCDR.

This register is readable and writable at all times and operation is independent of the chosen PI/T mode. The port C data register is not affected by the assertion of the $\overline{\text{RESET}}$ pin.

4.7 PORT ALTERNATE REGISTERS

The following paragraphs describe the port alternate registers.

4.7.1 Port A Alternate Register (PAAR)

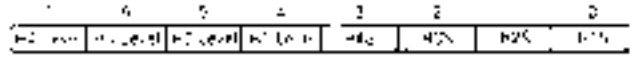
The port A alternate register is an alternate register for reading the port A pins. It is a read-only address and no other PI/T condition is affected. In all modes, the instantaneous pin level is read and no input latching is performed except at the data bus interface. Writes to this address are answered with $\overline{DTACK}$, but the data is ignored.

4.7.2 Port B Alternate Register (PBAR)

The port B alternate register is an alternate register for reading the port B pins. It is a read-only address and no other PI/T condition is affected. In all modes, the instantaneous pin level is read and no input latching is performed except at the data bus interface. Writes to this address are answered with $\overline{DTACK}$, but the data is ignored.

4.8 PORT STATUS REGISTER (PSR)

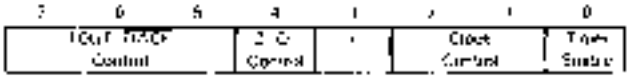
Port Status Register (PSR)



The port status register contains information about handshake pin activity. Bits 7-4 show the instantaneous level of the respective handshake pin, and are independent of the handshake pin sense bits in the port general control register. Bits 3-0 are the respective status bits referred to throughout this document. Their interpretation depends on the programmed mode/submode of the PI/T. For bits 3-0 a one is the active or asserted state.

4.9 TIMER CONTROL REGISTER (TCR)

Timer Control Register (TCR)



The timer control register (TCR) determines all operations of the timer. Bits 7-6 configure the PC3/TOUT and PC0/$\overline{TACK}$ pins for port C, square wave, vectored interrupt, or autovectored interrupt operation, bit 4 specifies whether the counter acquires data from the counter preload register or continues counting when zero detect is reached, bit 3 is unused and is read as zero; bits 2 and 1 configure the path from the CLK and TIN pins to the counter controller; and bit 0 enables the timer. The register is readable and writable at all times. All bits are cleared to zero when the $\overline{RESET}$ pin is asserted.

TCR 7 6 5	TOUT/$\overline{TACK}$ Control
0 0 X	The dual-function pins PC3/TOUT and PC0/$\overline{TACK}$ carry the port C function.
0 1 X	The dual-function pin PC3/TOUT carries the TOUT function. In the low state it is used as a square wave output and is toggled on zero detect. The TOUT pin is high-value in the low state. The dual-function pin PC0/$\overline{TACK}$ carries the PC0 function.

1 0 0 The dual-function pin PC3/TOUT carries the TOUT function. In the run or halt state it is used as a timer interrupt request output. The timer interrupt is disabled, thus, the pin is always three-stated. The dual-function pin PC7/$\overline{TIACK}$ carries the $\overline{TIACK}$ function, however, since the interrupt is disabled, the PIT produces no response to the data or DTACK in an asserted $\overline{TIACK}$. Refer to **5.1.3 Timer Interrupt Acknowledge Cycles** for details.

1 0 1 The dual-function pin PC3/TOUT carries the TOUT function and is used as a timer interrupt request output. The timer interrupt is enabled, thus, the pin is low when the timer ZDS status bit is one. The dual-function pin PC7/$\overline{TIACK}$ carries the $\overline{TIACK}$ function and is used as a timer interrupt acknowledge input. Refer to the **5.1.3 Timer Interrupt Acknowledge Cycles** for details. This combination supports vectored timer interrupts.

1 1 0 The dual-function pin PC3/TOUT carries the TOUT function. In the run or halt state it is used as a timer interrupt request output. The timer interrupt is disabled, thus, the pin is always three-stated. The dual-function pin PC7/$\overline{TIACK}$ carries the PC7 function.

1 1 1 The dual-function pin PC3/TOUT carries the TOUT function and is used as a timer interrupt request output. The timer interrupt is enabled, thus, the pin is low when the timer ZDS status bit is one. The dual-function pin PC7/$\overline{TIACK}$ carries the PC7 function and a prioritized interrupt is supported.

TCR
4 **Zero Detect Control**

0 The counter is loaded from the counter preload register on the first clock to the 24-bit counter after zero detect, then resumes counting.

1 The counter rolls over on zero detect, then continues counting.

TCR
3 Unused and is always read as zero.

TCR
2 1 **Clock Control**

0 0 The PC2/TIN input pin carries the port C function, and the CLK pin and prescaler are used. The prescaler is decremented on the falling transition of the CLK pin, the 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from $00 to $1F. The timer enable bit determines whether the timer is in the run or halt state.

0 1 The PC2/TIN pin serves as a timer input, and the CLK pin and prescaler are used. The prescaler is decremented on the falling transition of the CLK pin, the 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from $00 to $1F. The timer is in the run state when the timer enable bit is one and the TIN pin is high, otherwise, the timer is in the halt state.

1 0 The PC2/TIN pin serves as a timer input and the prescaler is used. The prescaler is decremented following the rising transition of the TIN pin after being synchronized with the internal clock. The 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers when the prescaler rolls over from $00 to $1F. The timer enable bit determines whether the timer is in the run or halt state.

1 1 The PC2/TIN pin serves as a timer input and the prescaler is not used. The 24-bit counter is decremented, rolls over, or is loaded from the counter preload registers following the rising edge of the TIN pin after being synchronized with the internal clock. The timer enable bit determines whether the timer is in the run or halt state.

TCR
0 **Timer Enable**

0 Disabled

1 Enabled

4.10 TIMER INTERRUPT VECTOR REGISTER (TIVR)

The timer interrupt vector register contains the 8-bit vector supplied when the timer interrupt acknowledge pin ($\overline{TIACK}$) is asserted. The register is readable and writable at all times, and the same value is always obtained from a normal read cycle or a timer interrupt acknowledge bus cycle ($\overline{TIACK}$). When the $\overline{RESET}$ pin is asserted the value $0F is loaded into the register. Refer to 5.1.3 Timer Interrupt Acknowledge Cycles for more details.

4.11 COUNTER PRELOAD REGISTER H, M, L (CPRH-L)

Counter Preload Register H, M, L (CPRH-L)

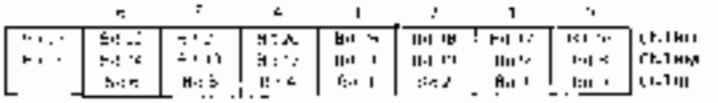


The counter preload registers are a group of three 8-bit registers used for storing data to be transferred to the counter. Each of the registers is individually addressable, or the group may be accessed with the MOVEP L or the MOVEP W instructions. The address is 12 locations less than the address of CPRH - the holding register and is arranged so that zeros are read in the upper eight bits of the destination data register when a MOVEP instruction is used. Data written to this address is ignored.

These registers are readable and writable at all times. A read cycle proceeds independently of any transfer to the counter, which may be occurring simultaneously. To ensure proper operation of the PIT timer, a value of $0000 00 must be stored in the counter preload registers for use with the counter. The $\overline{RESET}$ pin does not affect the contents of these registers.

4.12 COUNT REGISTER H, M, L (CNTRH-L)

Count Register H, M, L (CNTRH-L)



The count registers are a group of three 8-bit addresses at which the counter can be read. The contents of the counter are not latched during a read bus cycle; thus, the data read at these addresses is not guaranteed if the timer is in the count state. Write operations to these addresses result in a normal bus cycle, but the data is ignored.

Each of the registers is individually addressable, or the group may be accessed with the MOVEP L or the MOVEP W instructions. The address one less than the address CNTRH, is the holding register and is reserved so that zeros are read in the upper eight bits of the destination data register when a MOVEP L is used. Data written to this address is ignored.

4.13 TIMER STATUS REGISTER (TSR)

Timer Status Register (TSR)



The timer status register contains one bit from which the zero detect status can be determined. The ZDS status bit (bit 0) is an edge-sensitive flip-flop that is set to one when the 24-bit counter decrements from $000001 to $000000. The ZDS status bit is cleared to zero following the direct read operation or when the timer is halted. Note that when the $\overline{RESET}$ pin is asserted the timer is halted, and thus enters the halt state.

This register is always readable without consequence. A write access performs a direct read operation and bit 0 of the written data is one. Following that, the ZDS bit is zero.

This register is constructed with a reset dominant S-R flip-flop so that all clearing conditions prevail over the possible zero detect condition.

Bits 7-1 are unused and are read as zero.

4.14 REGISTER VALUE AFTER RESET

Table 1.3, located on foldout pages 1 and 2 at the end of this document, shows the values that remain or are changed after a reset. Note that interrupt vector registers are initialized to $0F. For the port interrupt vector register, the only time that bits 0 and 1 are set is after reset.

SECTION 5
TIMER OPERATION AND APPLICATIONS SUMMARY

This section describes the programmable options available, capabilities, and restrictions that apply to the timer. Programming of the timer control register is outlined with several examples given.

5.1 TIMER OPERATION

The MC68230 timer can provide several facilities needed by M68000 operating systems. It can generate periodic interrupts, a square wave, or a single interrupt after a programmed time period. Also, it can be used for elapsed time measurement or as a device watchdog.

The PI/T timer contains a 24-bit synchronous down counter that is loaded from three 8-bit counter preload registers. The 24-bit counter may be clocked by the output of a 5-bit divide-by-32 prescaler or by an external timer input (TIN). If the prescaler is used, it may be clocked by the system clock (CLK pin) or by the TIN external input. The counter signals the occurrence of an event primarily through zero detection. A zero is when the counter of the 24-bit timer is equal to zero. This sets the zero detect status (ZDS) bit in the timer status register. It may be checked by the processor or may be used to generate a timer interrupt. The ZDS bit can be reset by writing a one to the timer status register in that bit position independent of timer operation.

The general operation of the timer is flexible and easily programmable. The timer is fully configured and controlled by programming the 8-bit timer control register (refer to **4.8 TIMER CONTROL REGISTER (TCR)** for additional information). It controls: 1) the choice between the port C operation and the timer operation of three timer pins, 2) whether the counter is loaded from the counter preload register or rolls over when zero detect is reached, 3) the clock input, 4) whether the prescaler is used, and 5) whether the timer is enabled.

5.1.1 Run/Halt Definition

The overall operation of the timer is described in terms of the run or halt states. The control of the current state is determined by programming the timer control register. When in the halt state, all of the following occur:

1. The prior content of the counter is not altered and is reliably readable via the count registers.
2. The prescaler is forced to $11, whether or not it is used.
3. The ZDS status bit is forced to zero, regardless of the possible zero contents of the 24-bit counter.

The run state is characterized by:

1. The counter is clocked by the source programmed in the timer control register.
2. The counter is not reliably readable.
3. The prescaler is allowed to decrement if programmed for use.
4. The ZDS status bit is set when the 24-bit counter transitions from $000001 to $000000.

5.1.2 Timer Rules

The following is a set of rules that allow easy application of the timer.

1. Refer to 5.1.1 **Run/Halt Definition**
2. When the $\overline{RESET}$ pin is asserted, all bits of the timer control register are cleared, configuring the dual function pins as port C inputs.
3. The contents of the counter preload registers and counter are not affected by the $\overline{RESET}$ pin.
4. The count registers provide a direct read data path from each portion of the 24-bit counter, but data written to their addresses is ignored. (This results in a normal bus cycle.) These registers are readable at any time, but their contents are never altered. Unpredictable data may be read when the timer is in the run state.
5. The counter preload registers are readable and writable at any time and this occurs independently of any timer operation. No protection mechanisms are provided against ill-timed writes.
6. The input frequency to the 24-bit counter from the TIN pin or prescaler output must be between zero and the input frequency at the CLK pin divided by eight, regardless of the configuration chosen.
7. For configurations in which the prescaler is used (with the CLK pin or TIN pin as an input), the contents of the counter preload register (CPR) is transferred to the counter the first time that the prescaler passes from $00 to $1F (rolls over) after entering the run state. Thereafter, the counter decrements, rolls over, or is loaded from the counter preload register each time the prescaler rolls over.
8. For configurations in which the prescaler is not used, the contents of the counter preload registers are transferred to the counter on the first asserted edge of the TIN input after entering the run state. On subsequent asserted edges the counter decrements, rolls over, or is loaded from the counter preload registers.
9. The smallest value allowed in the counter preload register for use with the counter is $000007.

5.1.3 Timer Interrupt Acknowledge Cycles

Several conditions may be present when the timer interrupt acknowledge pin ($\overline{TIACK}$) is asserted. These conditions affect the PITS response and the termination of the bus cycle (see Table 5-1).

Table 5-1. Response to Timer Interrupt Acknowledge

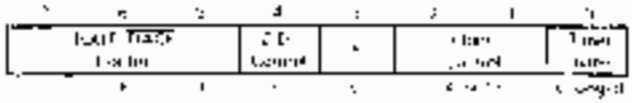
PC3/TOUT Function	Response to Asserted $\overline{TIACK}$
PITS = Port C Pin	No Response No $\overline{CTACK}$
TOUT = Square Wave	No Response No $\overline{CTACK}$
TOUT = Periodic Timer Interrupt Request	No Response No $\overline{CTACK}$
TOUT = Periodic Timer Interrupt Request	Vector number is placed on Data Bus ($\overline{CTACK}$ Asserted)

5.2 TIMER APPLICATIONS SUMMARY

The following paragraphs outline programming of the timer control register for several typical examples.

5.2.1 Periodic Interrupt Generator Example

Periodic Interrupt Generator Example



In this configuration the timer generates a periodic interrupt. The TOUT pin is connected to the system's interrupt request circuitry and the $\overline{TIACK}$ pin may be used as an interrupt acknowledge input to the timer. The TIN pin may be used as a clock input.

The processor loads the counter preload register (CPR) and timer control register (TCR) and then enables the timer. When the 24 bit counter passes from $000001 to $000000, the ZDS status bit is set and the TOUT interrupt request pin is asserted. At the next clock to the 24 bit counter, the CPR is again loaded with the contents of the CPRs and the counter decrements. In normal operation, the processor must direct clear the status bit to negate the interrupt request (see Figure 5-1).

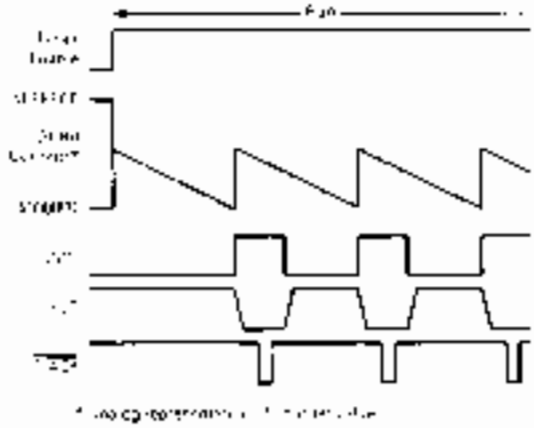
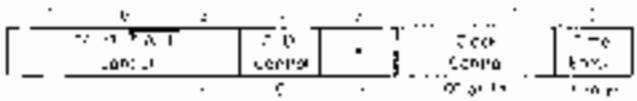


Figure 5-1 Periodic Interrupt Generator Example

5.2.2 Square Wave Generator

Square Wave Generator



This configuration of the timer produces a square wave at the TOUT pin. The TOUT pin is connected to the system's interrupt circuitry and the $\overline{TIACK}$ pin is not used. The TIN pin may be used as a clock input.

The processor loads the counter preload register and timer control register and then enables the timer. When the 24-bit counter passes from $000001 to $000000, the ZDS status bit is set and the

TC/T square wave output pin is toggled. At the next clock to the 24-bit counter it is again loaded with the contents of the CPRs, and thereafter decrements. In this application there is no need for the processor to direct clear the ZDS status bit. However, it is possible for the processor to sync itself with the square wave by clearing the ZDS status bit, then polling it. The processor may also read the TCOUT level at the port C address.

Note that the PC3/TCOUT pin functions as PC3 following the negation of $\overline{RESET}$. If used in the square wave configuration, a pullup resistor may be required to keep a known level prior to programming. Prior to enabling the timer, TCOUT is high (see Figure 5-2).

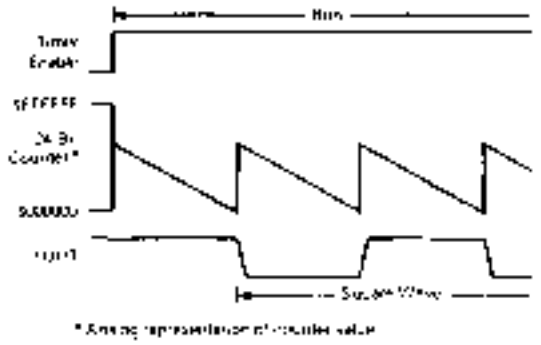
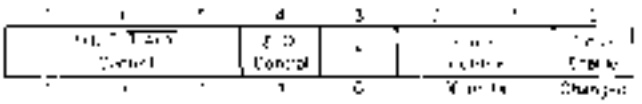


Figure 5-2. Square Wave Generator Example

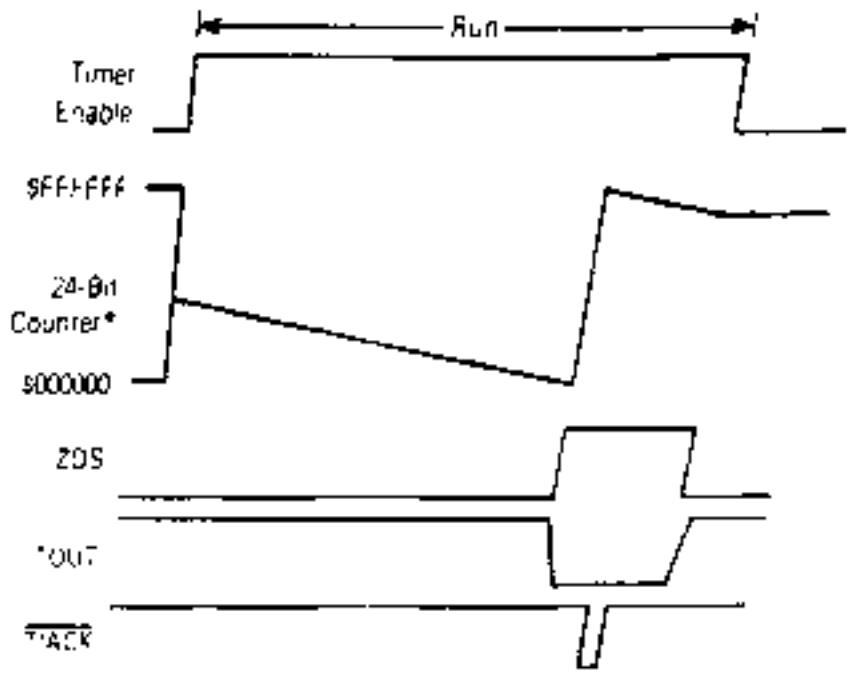
5.2.3 Interrupt After Timeout

Interrupt After Timeout



In this configuration the timer generates an interrupt after a programmed time period has expired. The TCOUT pin is connected to the system's interrupt request circuitry and the $\overline{IACK}$ pin may be an interrupt acknowledge input to the timer. The TIN pin may be used as a clock input.

This configuration is similar to the periodic interrupt generator except that the zero detect control bit is set. This forces the counter to roll over after zero detect is reached, rather than reloading from the CPRs. When the processor takes the interrupt it can halt the timer, read the counter and calculate the time from the interrupt request to entering the service routine. Accurate knowledge of the interrupt latency may be useful in some applications (see Figure 5-3).



* Analog representation of counter value

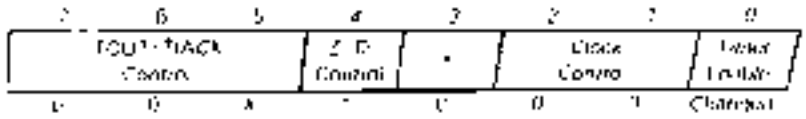
Figure 5-3. Single Interrupt After Timeout Example

5.2.4 Elapsed Time Measurement Examples

Elapsed time measurement takes several forms; two forms are described in the following paragraphs.

5.2.4.1 **SYSTEM CLOCK EXAMPLE.** This configuration allows time interval measurement by software. The TIN pin may be used as an external timer enable if desired.

System Clock Example



The processor loads the counter preload registers (generally with all ones), loads the timer control register, and then enables the timer. The counter is allowed to decrement until the zero/stop event takes place. When it is desired to read the time interval, the processor must halt the timer and then read the counter. If TIN is used as an enable, the start and stop counter functions are controlled externally.

For applications in which the interval may exceed the programmed time interval, zero detection can be counted by polling the status register or through interrupts to simulate additional timer bits. Note that the ZDS bit is latched and should be cleared after each detection of zero. At this time, the value can be halted and read (see Figure 5-4).

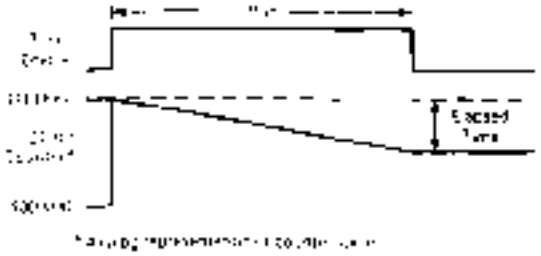
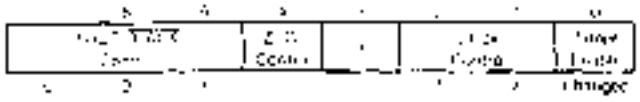


Figure 5-4. Elapsed Time Measurement Example

5.2.4.2 EXTERNAL CLOCK This configuration is basically measurement (counting) of the number of input pulses occurring in an interval in which the counter is enabled. The TIN input pin provides the input pulses. Generally, the TOUT and $\overline{TACK}$ pins are not used.

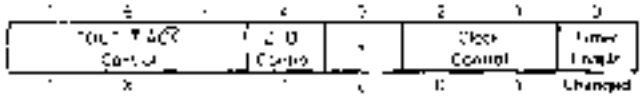
External Clock



This configuration is similar to the elapsed time measurement system clock configuration except that the TIN pin is used to provide the input frequency. It can be connected to a simple oscillator and the same methods could be used. Alternately, it could be gated off and on externally and the number of cycles occurring while in the run state can be counted. However, minimum pulse width high and low specifications must be met.

5.2.5 Device Watchdog

Device Watchdog Example



This configuration provides the watchdog function needed in many systems. The TIN pin is the timer input whose period at the high level is to be checked. Once allowed by the processor, the TIN input pin controls the run/halt mode. The TOUT pin is connected to external circuitry requiring notification when the TIN pin has been asserted longer than the programmed time. The $\overline{TACK}$ pin (timer interrupt acknowledge) is only needed if the TOUT pin is connected to the interrupt circuitry.

The processor reads the counter preload register and timer control register, and then enables the timer. When the TIN input is asserted (pre-night) the timer transfers the contents of the counter preload register to the counter and begins counting. If the TIN input is negated before zero detect is reached, the TOUT output and the ZDS status bit is not negated. If zero detect is reached while the TIN input is still asserted, the ZDS status bit is set and the TOUT output is asserted. The counter rolls over and keeps counting.

In either case, when the TIN input is negated the ZDS status bit is zero, the TOUT output is negated, the counting stops, and the prescaler is forced to all ones (see Figure 5-5).

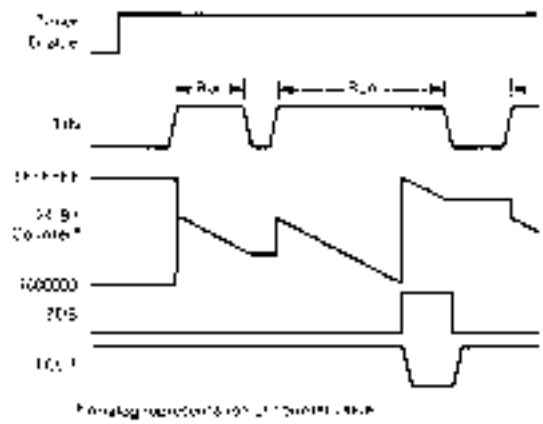


Figure 5-5. Device Watchdog Example

SECTION 6
ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the MC68230.

6.1 MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

6.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Rating
Thermal Resistance Ceramic Plastic	θ_{JA}	35 60	°C/W

6.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J in °C, can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

Where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected):

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

6.4 DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ± 5%, T_A = 0 to 70°C, unless otherwise noted)

Characteristics	Symbol	Min	Max	Unit

6.5 AC ELECTRICAL SPECIFICATIONS — CLOCK TIMING (See Figure 6-1)

Characteristic	Symbol	8 MHz		10 MHz		12.5 MHz		Unit
		Min	Max	Min	Max	Min	Max	
Frequency of Operation		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
Clock Pulse Width	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
Cycle-to-Cycle Jitter	[illegible]		[illegible]		[illegible]		[illegible]	[illegible]
	[illegible]		[illegible]		[illegible]		[illegible]	

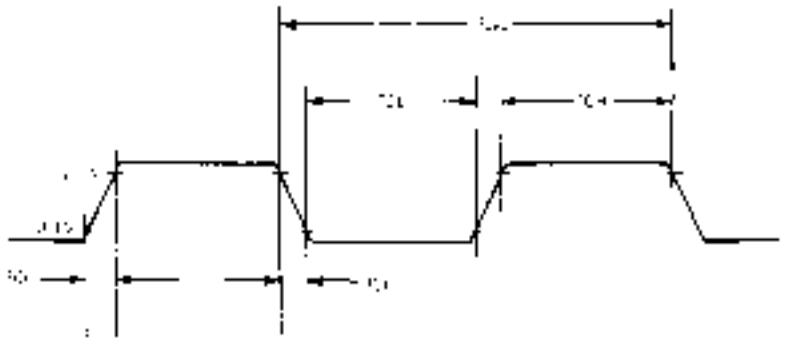
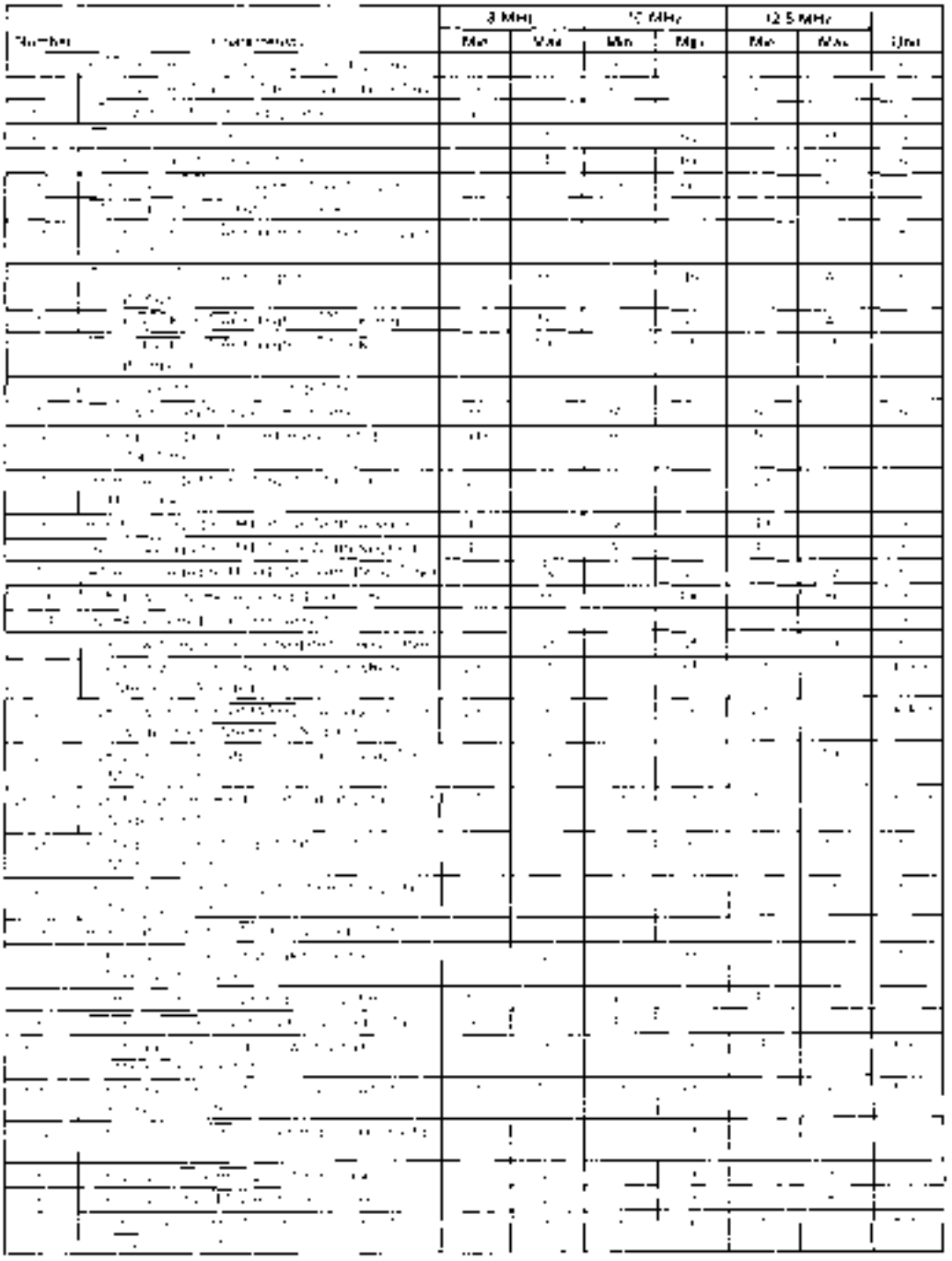


Figure 6-1. Clock Input Timing Diagram

6.6 AC ELECTRICAL SPECIFICATIONS



6.6 AC ELECTRICAL SPECIFICATIONS (Continued)

Number	Characteristic	8 MHz		10 MHz		12.5 MHz		Unit
		Min	Max	Min	Max	Min	Max	
[illegible]	Synchronized $\overline{CS}$ to CLK Low on which $\overline{PEC}$ is High (operation)	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	CLK Per
[illegible]	CLK Low to PEC Low or High Impedance	0	250	0	200	0	200	ns
[illegible]	CLK Frequency (External Clock — Prescaler Used)	0	8	0	10	0	12.5	f_{CLK} (MHz) (6)
41	CLK Frequency (External Clock — Prescaler Not Used)	0	16	0	16	0	16	f_{CLK} (MHz) (6)
42	CLK Pulse Width High or Low (External Clock)	55	–	45	–	45	–	ns
43	CLK Pulse Width Low (Run, High Current)	1	–	1			–	CLK Per
44	CLK Low to TIACF High, Low or High Impedance	0	250	0	200	0	200	ns
45	$\overline{CS}$ TIACF or TIACF High to CS TIACK or TIACF Low	50	–	30	–	20	–	ns

NOTES:

1. This specification only applies if the PIU had completed all operations initiated by the previous bus cycle when $\overline{CS}$ was asserted. Following a normal read or write bus cycle, all operations are complete within three clocks after the falling edge of the CLK period when $\overline{DTACK}$ was asserted. If $\overline{CS}$ is asserted prior to completion of these operations, the new bus cycle will be held off until the operations are completed.

   If all operations of the previous bus cycle were complete when $\overline{CS}$ was asserted, the specification is met only if the $\overline{CS}$ TIACK was asserted with respect to the falling edge of the CLK period shown in the timing diagrams. For an application requiring the part to run the $\overline{CS}$ setup time is reduced, $\overline{DTACK}$ may be asserted as shown or may be asserted one clock cycle later.

2. Assuming the PIU CLK is stable at the time reset is released.

3. This parameter imposes a lower bound on $\overline{CS}$ low time, guaranteeing that $\overline{CS}$ will be low for at least 1 CLK period.

4. This period can be extended by the modification of the asserted edge of HIRQ1.

5. For some devices, the $\overline{CS}$ setup time is defined as the time required for the proper operation of the internal synchronization circuitry. If the setup time is met, the number of clocks required for the assertion of TIACK will be minimized.

6. Refers to the accuracy of the CLK input. Particular baud rates may require specific frequencies.

7. The minimum time the CLK signal must remain at a high or low level. This value is derived from the basic clock period and may be used to calculate the width of any CLK pulse.

8. This value applies to the falling edge of the signal. It is measured from the point where the CLK signal reaches 1.5 volts to the point where the signal reaches 0.8 volts. The rising edge of the signal is measured from 0.8 volts to 2.0 volts. This value is derived from the basic clock period and may be used to calculate the width of any CLK pulse.

   Timing is guaranteed when the rise and fall times are 10 ns or less. The duty cycle of the clock may be anywhere between 40-60% as long as the minimum and maximum limits are observed. The frequency of the CLK signal is derived from a crystal or external source.

9. These specifications are derived from the same parameters as the two edges of the signal, but they determine different delays. The minimum specification on one edge will always be greater than the minimum specification on the other. For example, if a 74S04 is used to generate the signal and the input to the AND function is the clock and a control signal, the [illegible] will be generated by the AND function if the clock and control signal

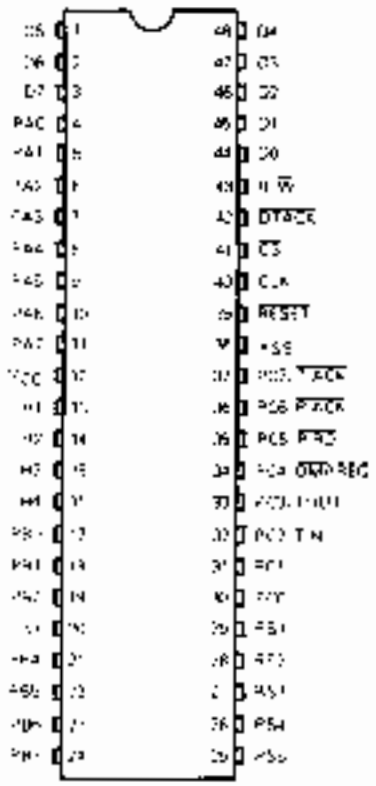
10. [illegible] peripheral [illegible] asserted and has driven $\overline{DT}$ low, the PIU [illegible] appropriate edge of the clock [illegible] $\overline{CS}$ [illegible] (see 6.4 BUS INTERFACE OPERATION for the exception to this).

Timing diagrams (Figures 6-2, 6-3, 6-4, 6-5, and 6-6) are detailed on layout pages 3, 4, 5, and 6 at the end of this document.

SECTION 7
MECHANICAL DATA AND ORDERING INFORMATION

This section contains the pin assignments and package dimensions of the VCA820. In addition, ordering information is provided to be used as a guide when ordering.

7.1 PIN ASSIGNMENT

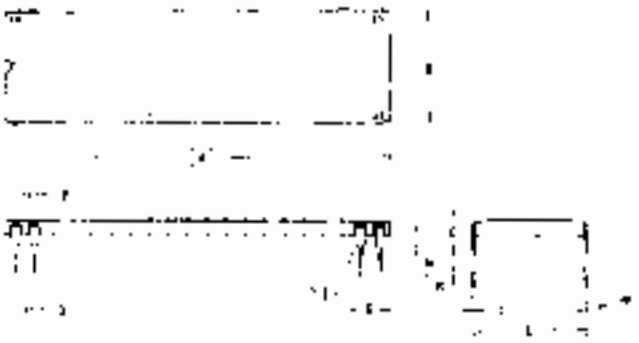


7.2 ORDERING INFORMATION

Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic	8.0	0°C to 70°C	MC68230L8
L Suffix	10.0	0°C to 70°C	MC68230L10
	12.5	0°C to 70°C	MC68230L12
Plastic	8.0	0°C to 70°C	MC68230G8
G Suffix	10.0	0°C to 70°C	MC68230G10
	12.5	0°C to 70°C	MC68230G12

7.3 PACKAGE DIMENSIONS

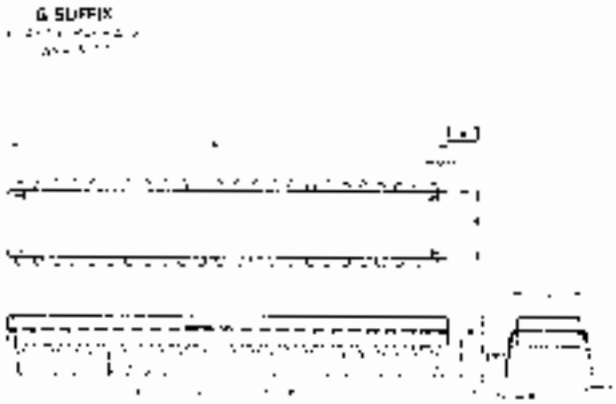
L SUFFIX
CERAMIC PACKAGE
CASE 740-03



NOTES
1. DIMENSION A IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS
   (4 PLACES): 0.25 (0.010) M T A S
3. T IS SEATING PLANE.
4. DIMENSION "L" TO CENTER OF
   LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING
   PER ANSI Y14.5, 1973.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	39.35	41.91	1.549	1.650
B	14.62	15.24	0.576	0.600
C	3.05	4.32	0.120	0.170
D	0.381	0.533	0.015	0.021
F	0.762	1.397	0.030	0.055
G	2.54 BSC		0.100 BSC	
J	0.203	0.330	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.99	15.65	0.590	0.616
M	0°	10°	0°	10°
N	1.016	1.524	0.040	0.060

G SUFFIX

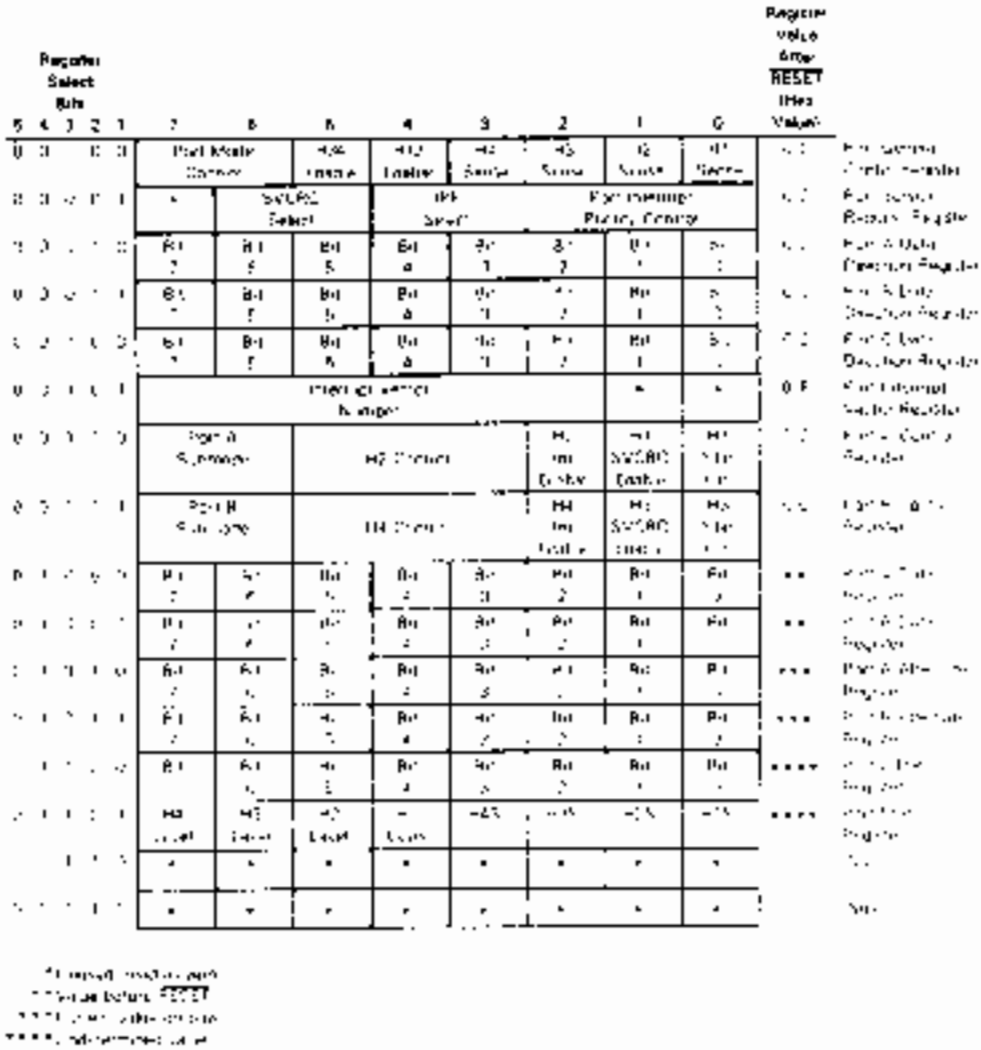


NOTES

1. [D] IS END OF PACKAGE DATUM PLANE
   [T] IS BOTH A DATUM AND SEATING PLANE
2. POSITIONAL TOLERANCE FOR LEADS 24 & 25
   [⌀0.35+0.014|T|B|C|R]
   POSITIONAL TOLERANCE FOR LEAD PATTERN
   [⌀|0.25+0.010|T|B|C]
3. DIM B DOES NOT INCLUDE MOLD FLASH.
4. D.M.L IS TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5 1973.

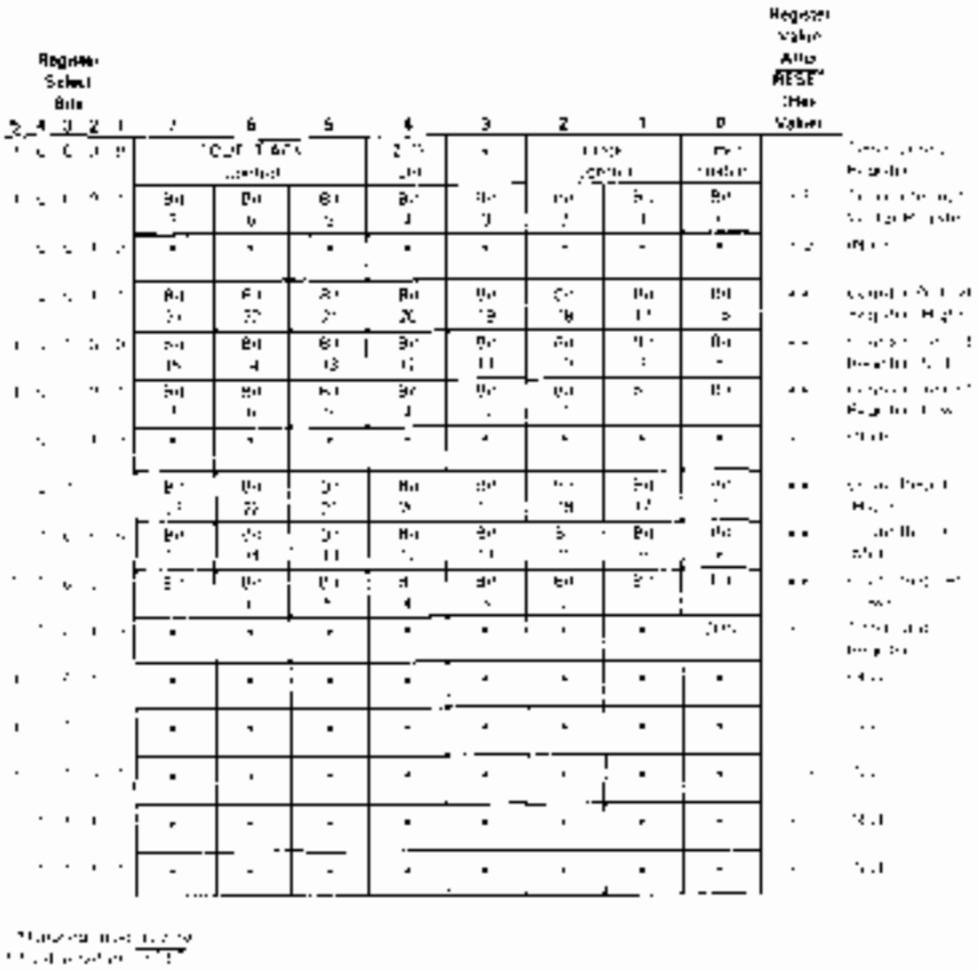
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	61.07	61.77	2.405	2.432
B	13.92	14.42	0.548	0.567
C	4.83	5.08	0.190	0.200
D	0.38	0.50	0.015	0.020
F	1.22	1.34	0.048	0.053
G	2.54 BSC		0.100 BSC	
J	0.25	0.30	0.010	0.012
K	3.23	3.37	0.127	0.133
L	15.24 BSC		0.600 BSC	
M	0°	10°	0°	10°
N	0.64	0.89	0.025	0.035
U	1.19 BSC		0.070 BSC	

Table 1-3. Register Model (Sheet 1 of 2)



Folded

Table 1-3. Register Model (Sheet 2 of 2)



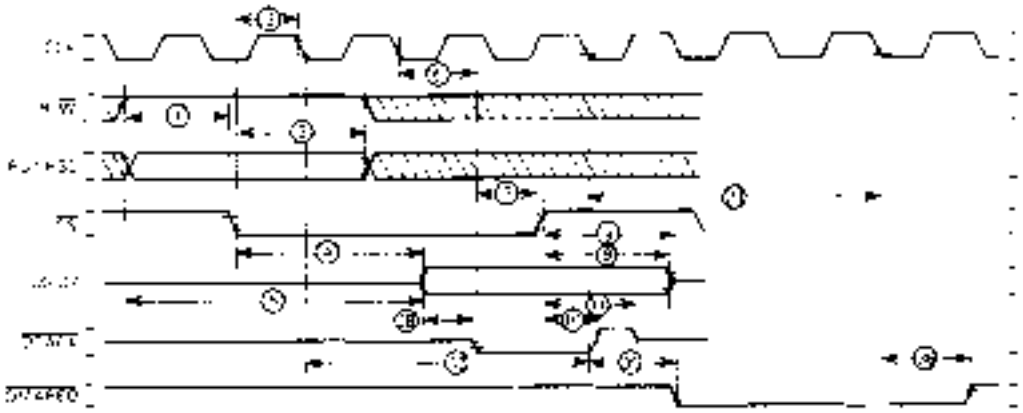


Figure 6-2. Read Cycle Timing Diagram

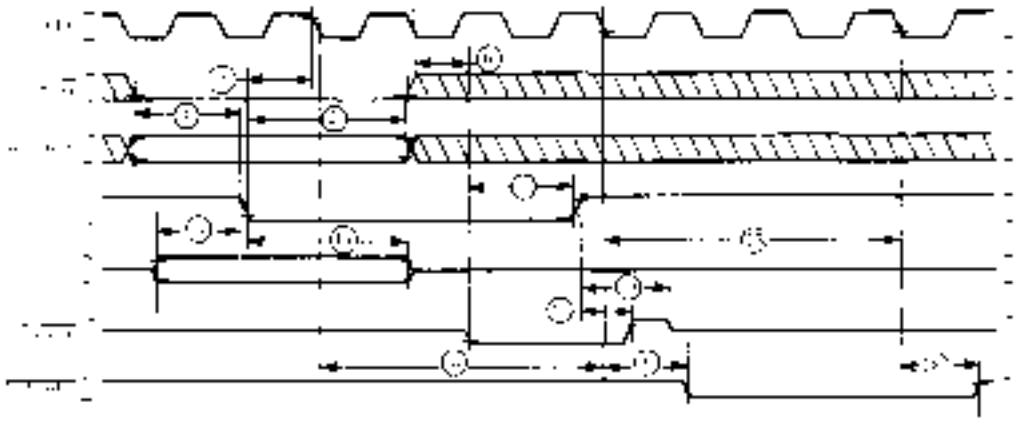


Figure 6-3. Write Cycle Timing Diagram

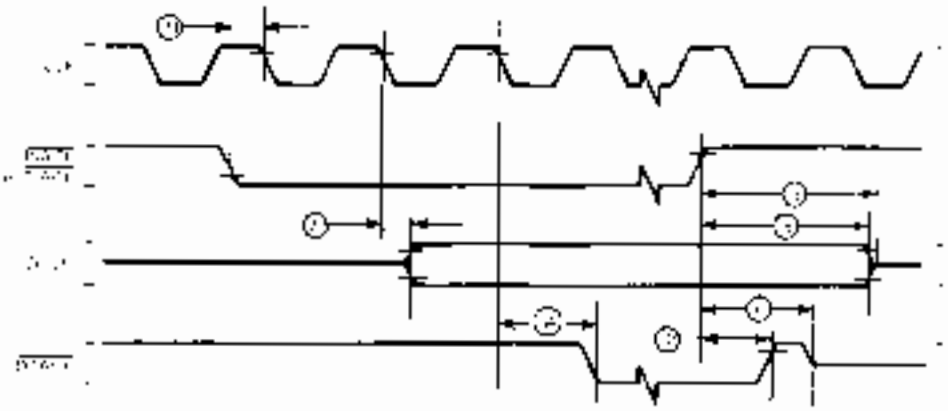


Figure 5-4. IACK Timing Diagram

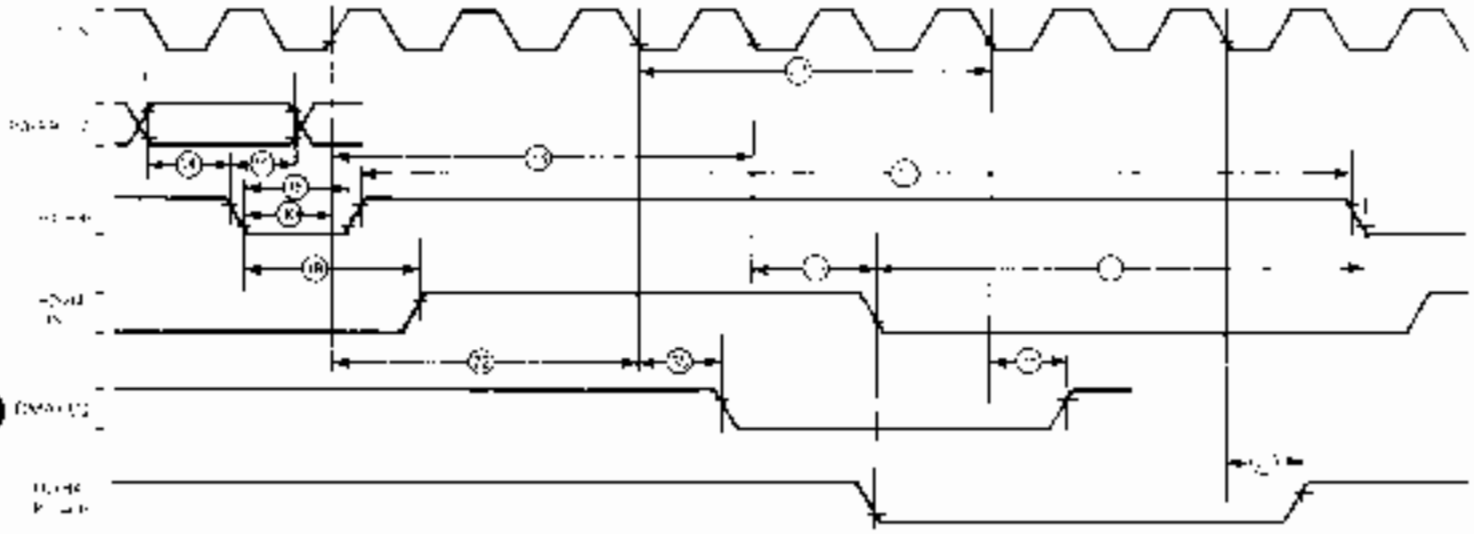


Figure 6 5. Peripheral Input Timing Diagram

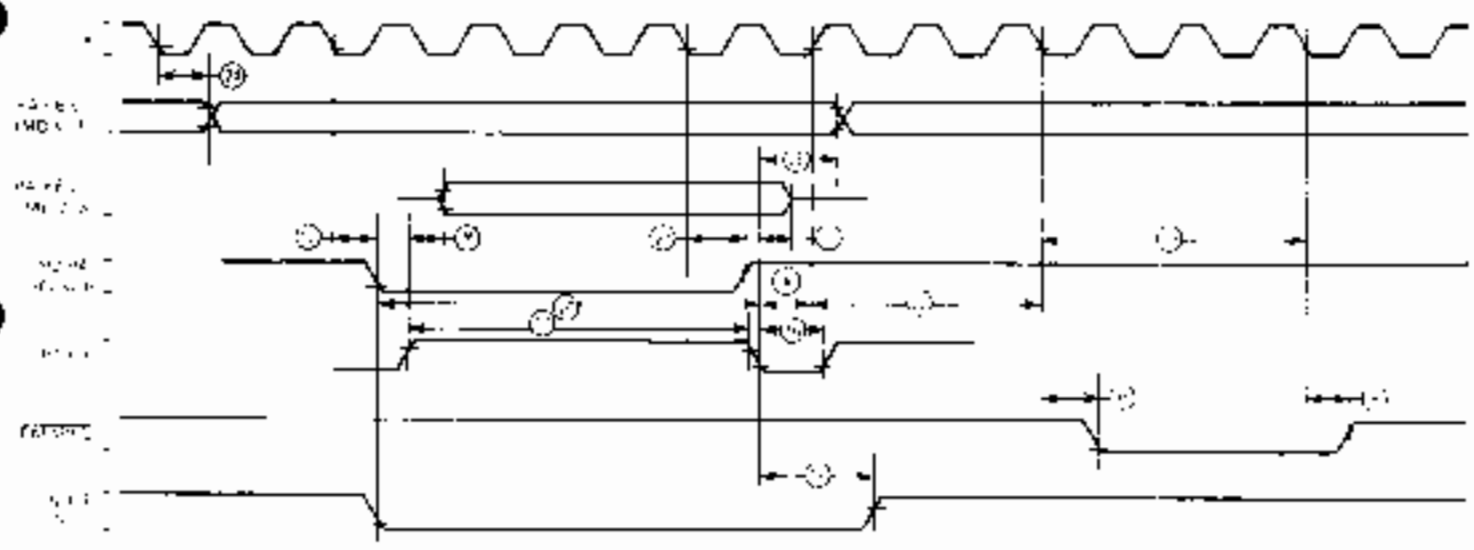


Figure 6 6. Peripheral Output Timing Diagram

This page was intentionally left blank

NCR 5386 SCSI Protocol Controller Data Sheet

NCR

NCR Microelectronics Division, Colorado Springs

Copyright © 1985, by NCR Corporation
Dayton, Ohio
All Rights Reserved Printed in U.S.A.

This document contains the latest information available at the time of publication. However, NCR reserves the right to modify the contents of this material at any time. Also, all features, functions and operations described herein may not be marketed by NCR in all parts of the world. Therefore, before using this document, consult your NCR representative or NCR office for the information that is applicable and current.

TABLE OF CONTENTS

SECTION			PAGE
1.		GENERAL DESCRIPTION	1
2.		PIN DESCRIPTION	3
	2.1	Microprocessor Interface Signals	3
	2.2	SCSI Interface Signals	4
3.		ELECTRICAL CHARACTERISTICS	6
4.		INTERNAL REGISTERS	7
	4.0	General	7
	4.1	Data Registers	7
	4.2	Command Register	7
	4.3	Control Register	8
	4.4	Destination ID Register	9
	4.5	Auxiliary Status Register	9
	4.6	ID Register	11
	4.7	Interrupt Register	12
	4.8	Source ID Register	14
	4.9	Data Register 2	14
	4.10	Diagnostic Status Register	15
	4.10.1	Self-Diagnostic Status Code Summary	16
	4.11	Transfer Counter	16
5		COMMANDS	17
	5.1	Command Format	17
	5.2	Command Type	18
	5.3	Invalid Commands	19
	5.4	Command Summary	19
	5.5	Command Definitions	20
	5.5.1	Chip Reset	20
	5.5.2	Disconnect	20
	5.5.3	Pause	20
	5.5.4	Set ATN	21
	5.5.5	Message Accepted	21
	5.5.6	Chip Disable	21
	5.5.7	Select w/ATN	22
	5.5.8	Select w/o ATN	22
	5.5.9	Reselect	23
	5.5.10	Diagnostic Data Turnaround	24
	5.5.11	Receive Commands	25
	5.5.12	Send Commands	26
	5.5.13	Transfer Info	27
	5.5.14	Transfer Pad	28

6 BUS INITIATED FUNCTIONS 29
6.1 Selection 29
6.2 Reselection 29
7. INITIALIZATION 30
8. EXTERNAL CHIP TIMINGS 31
8.1 Microprocessor Interface 31
8.1.1 Clock 31
8.1.2 Reset 31
8.1.3 MPU Write 32
8.1.4 MPU Read 32
8.1.5 DMA Write 33
8.1.6 DMA Read 33
8.1.7 Interrupt 34
8.2 SCSI Interface 35
8.2.1 Selection (Initiator) 35
8.2.2 Selection (Target) 37
8.2.3 Reselection (Initiator) 38
8.2.4 Reselection (Target) 39
8.2.5 Information Transfer Phase Input (Initiator) 41
8.2.6 Information Transfer Phase Input (Target) 42
8.2.7 Information Transfer Phase Output (Initiator) 43
8.2.8 Information Transfer Output (Target) 44
8.2.9 Bus Release from Selection (Initiator) 45
8.2.10 Bus Release From Reselection (Target) 46
8.2.11 Bus Release From Information Phase (Initiator) 47
8.2.12 Bus Release From Information Phase (Target) 48

SECTION 1
GENERAL DESCRIPTION

The NCR SCSI Protocol Controller (SPC) is designed to accommodate the Small Computer Systems Interface (SCSI) as defined by the ANSI X3T9.2 committee. The SPC operates in both the Initiator and Target roles and can therefore be used in host adapter and control unit designs. This device supports arbitration, including reselection, and is intended to be used in systems that require either open collector or differential pair transceivers.

The NCR 5386 SCSI Protocol Controller communicates with the system microprocessor as a peripheral device. The chip is controlled by reading and writing several internal registers which may be addressed as standard or memory mapped I/O. A 24-bit Transfer Counter and the appropriate handshake signals accomodate large DMA transfers with minimal processor intervention. Since the NCR 5386 interrupts the MPU when it detects a bus condition that requires servicing, the MPU is freed from polling or controlling any of the SCSI bus signals.

Below is a list of important features:

SCSI INTERFACE

*Supports ANSI X3T9.2 SCSI Standard

- Asynchronous data transfers to 2.0 MBPS
- Supports both Initiator and Target roles
- Parity generation with optional checking
- Supports arbitration
- Controls all bus signals except Reset
- Doubly-buffered Data Register

MPU INTERFACE

*Versatile MPU Bus Interface

- Memory or I/O mapped MPU interface
- DMA or programmed I/O transfers
- 24-bit Internal Transfer Counter
- Programmable (Re) Selection Timeouts
- Interrupts MPU on all bus conditions requiring service
- SCSI pass parity optional with checking

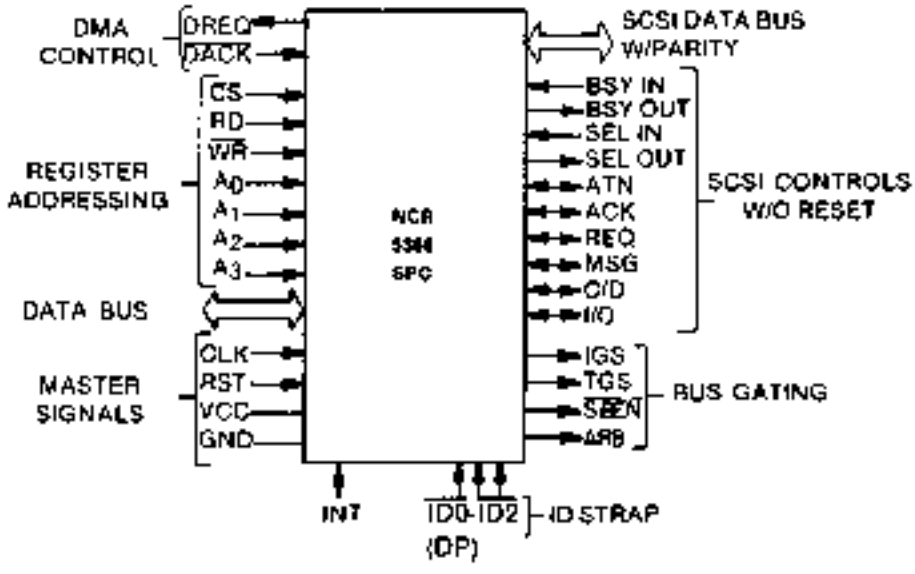


Figure 1.1
Functional Pin Grouping

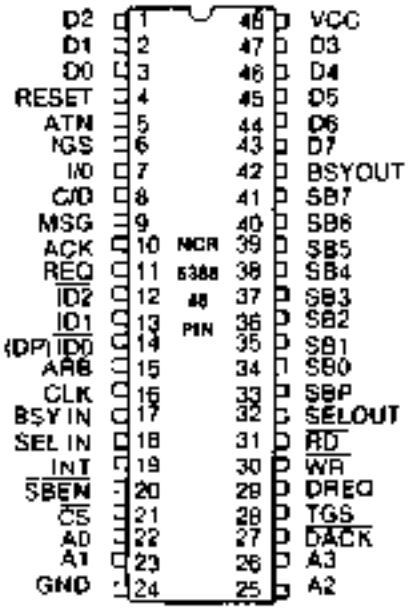


Figure 1.2
Pinout

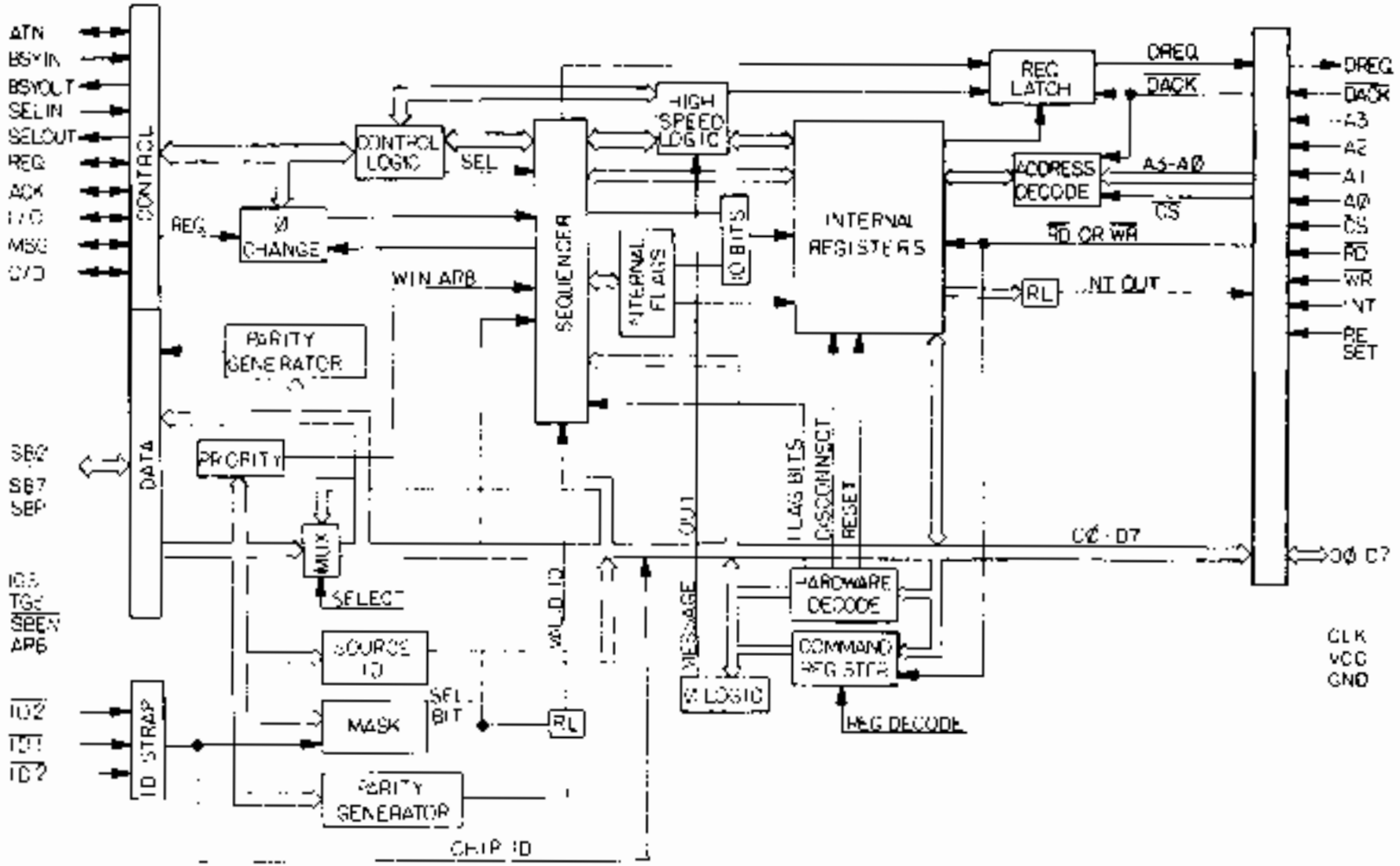


Figure 1.3
NCR 5386 Block Diagram

SECTION 2
PIN DESCRIPTION

2.1 MICROPROCESSOR INTERFACE SIGNALS

CLK	16	Symmetrical square wave signal which generates internal chip timing. Maximum frequency is 10 MHz.
RESET	4	When high (1), this signal forces the chip into a reset state. All current operations are terminated, internal storage elements are cleared and self-diagnostics are performed.
D0-D7	3-1 47-43	These signals comprise an active high data bus. It is intended that these signals be connected to the microprocessor data bus.
INT	19	This signal is used to interrupt the microprocessor for various bus conditions that require service. INT is set high for request and cleared when the chip is reset or the Interrupt Register is read.
$\overline{WR}$	30	Write pulse (active low) is used to strobe data from the data bus into an internal register which has been selected.
$\overline{RD}$	31	Read pulse (active low) is used to read data from an internal register that has been selected. The contents of the register are strobed onto the data bus
$\overline{CS}$	21	When low (0), this signal enables reading from or writing to the internal register which has been selected.
A0-A3	22, 23, 25, 26	These signals are used in conjunction with $\overline{CS}$, to address all the internal registers.
DREQ	29	Data request. When high (1), this signal indicates that the internal Data Register has a byte to transfer (inputting from the SCSI bus) or needs a byte to transfer (outputting to the SCSI bus). This signal becomes active only if the DMA mode bit in the Command Register is on. It is cleared when $\overline{DACK}$ becomes active
$\overline{DACK}$	27	Data acknowledge. When low (0), this signal resets DREQ and selects the Data Register for input or output. $\overline{DACK}$ acts as a chip select for the Data Register when in the DMA mode. $\overline{DACK}$ and $\overline{CS}$ must never be active at the same time.

2.2 SCSI INTERFACE SIGNALS

$\overline{ID0}$-$\overline{ID2}$ (DP)	14-12	These active low signals determine the three-bit code of the SCSI bus ID assigned to the chip. External pullup resistors are required only if tied to switches or straps. (Optionally, pin 14 may be used as the data bus parity signal. In this case, the device ID is programmed by the system processor and pins 12 and 13 are not used. Refer to Section 4.6, "ID Register," for a detailed description.)
SB0-SB7, SBP	34-41, 33	Active high data bus. These signals comprise the SCSI data bus and are intended to be connected to the external SCSI bus transceivers.
BSYIN	17	When high (1), this signal indicates to the chip that the SCSI BSY signal is active.
BSYOUT	42	When high (1), the chip is asserting the BSY signal to the SCSI bus.
SELIN	18	When high (1), this signal indicates to the chip that the SCSI SEL signal is active.
SELOUT	32	When high (1), the chip is asserting the SEL signal to the SCSI bus.
ATN	5	INITIATOR ROLE: The chip asserts this signal when the microprocessor requests the attention condition or a parity error has been detected in a byte received from the SCSI bus. TARGET ROLE: This signal is an input which indicates the state of the ATN signal on the SCSI bus.
ACK	10	INITIATOR ROLE: The chip asserts this signal in response to REQ for a byte transfer on the SCSI bus. TARGET ROLE: This signal is an input which, when active, indicates a response to the REQ signal.
REQ	11	INITIATOR ROLE: This signal is an input which, when active, indicates that the Target is requesting a byte transfer on the SCSI bus. TARGET ROLE: Asserted by the chip to request a byte transfer on the SCSI bus.
MSG, C/D, I/O	9, 8, 7	INITIATOR ROLE: These signals are inputs which indicate the current SCSI bus phase. TARGET ROLE: The chip drives these signals to indicate the current bus phase.
IGS	6	Initiator Group Select. When high (1), this signal indicates to the external SCSI drivers that the chip is controlling in the Initiator role. Its purpose is to enable the external drivers for ATN and ACK. When low (0), ATN and ACK should be received.
TGS	28	Target Group Select. When high (1), this signal indicates to the external SCSI drivers that the chip is controlling in the Target role. Its purpose is to enable the external drivers for REQ, MSG, C/D, and I/O. When low (0), REQ, MSG, C/D, and I/O should be received.

$\overline{SBEN}$	20	SCSI data Bus Enable. When low (0), this signal directly enables the external SCSI data bus drivers.
ARB	15	Arbitration phase. When high (1), this signal enables the external circuitry to place the ID bit on the SCSI bus for the Arbitration phase.

POWER SIGNALS

VCC	48	+5 V input
GND	24	Signal reference input

SECTION 3
ELECTRICAL CHARACTERISTICS

PRELIMINARY

OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN	MAX	UNITS
Supply Voltage	V_{DD}	4.75	5.25	Vdc
Supply Current	I_{DD}		300	mA
Ambient Temperature	T_A	0	70	°C

INPUT SIGNAL REQUIREMENTS

PARAMETER	CONDITIONS	MIN	MAX	UNITS
High-level Input, V_{IH}		2.0	5.25	Vdc
Low-level Input, V_{IL}		-0.3	0.8	Vdc
High-level Input Current, I_{IH}	V_{IH} = 5.25V		10	μA
Low-level Input Current, I_{IL}	V_{IL} = 0V		-10	μA

OUTPUT SIGNAL REQUIREMENTS (Except $\overline{SBEN}$, IGS, and TGS)

PARAMETER	CONDITIONS	MIN	MAX	
High-level Output Voltage, V_{OH}	V_{DD} = 4.75V @ I_{OH} = -400 μA	2.4	—	Vdc
Low-level Output Voltage, V_{OL}	V_{DD} = 4.75V @ I_{OL} = 2.0mA	—	0.4	Vdc

$\overline{SBEN}$, IGS, and TGS SIGNALS

PARAMETER	CONDITIONS	MIN	MAX	UNITS
High-level Output Voltage, V_{OH}	V_{DD} = 4.75V @ I_{OH} = -400 μA	2.4	—	Vdc
Low-level Output Voltage, V_{OL}	V_{DD} = 4.75V @ I_{OL} = 4.0mA	—	0.4	Vdc

SECTION 4
INTERNAL REGISTERS

4.0 GENERAL

The NCR SCSI Protocol Controller has a set of internal registers which are used by the microprocessor to direct the operation of the SCSI bus. These registers are read (written) by activating $\overline{CS}$ with an address on A3-A0 and then issuing a $\overline{RD}$/($\overline{WR}$) pulse. They can be made to appear to a microprocessor as standard I/O ports or as memory-mapped I/O ports depending on the external circuitry that controls $\overline{CS}$. The following sections describe the operation of these internal registers.

REGISTER SUMMARY

A3	A2	A1	A0	R/W	REGISTER NAME
0	0	0	0	R/W	Data Register
0	0	0	1	R/W	Command Register
0	0	1	0	R/W	Control Register
0	0	1	1	R/W	Destination ID
0	1	0	0	R	Auxiliary Status
0	1	0	1	R	ID Register
0	1	1	0	R	Interrupt Register
0	1	1	1	R	Source ID
1	0	0	1	R	Diagnostic Status
1	1	0	0	R/W	Transfer Counter (MSB)
1	1	0	1	R/W	Transfer Counter (2nd BYTE)
1	1	1	0	R/W	Transfer Counter (LSB)
1	1	1	1	R/W	Reserved for Testability

4.1 DATA REGISTERS

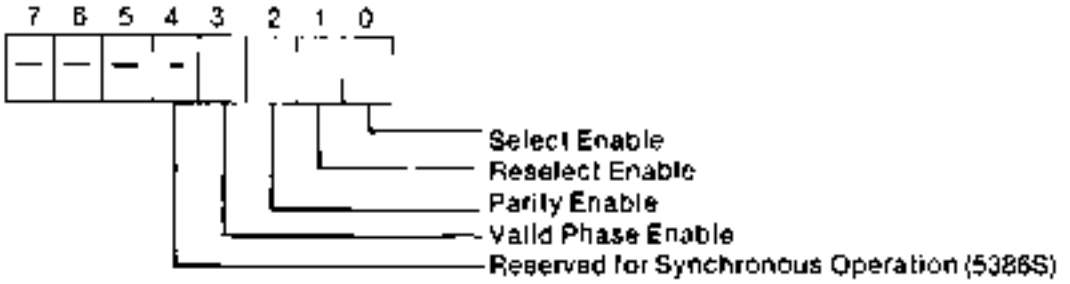
The Data Registers are used to transfer SCSI commands, data, status and message bytes between the microprocessor data bus and the SCSI bus. These are eight-bit registers which are doubly-buffered in order to support maximum throughput. In the non-DMA mode, the microprocessor reads from (writes to) Data Register 1 by activating $\overline{CS}$ with A3-A0 = 0000 and issuing a $\overline{RD}$/($\overline{WR}$) pulse. Two bits have been provided in the Auxiliary Status Register to indicate the status of both Data Register 1 and Data Register 2. In the DMA mode, the DMA logic reads from (writes to) Data Register 1 by responding to DREQ with $\overline{DACK}$ and issuing a $\overline{RD}$/($\overline{WR}$) pulse. The SCSI bus reads from or writes to Data Register 2 when the chip is connected as an Initiator or Target and the bus is in one of the Information Transfer Phases.

4.2 COMMAND REGISTER

The Command Register is an eight-bit register used to give commands to the SCSI chip. The microprocessor can write to (read from) the Command Register by activating $\overline{CS}$ with A3-A0 = 0001 and issuing a $\overline{WR}$/($\overline{RD}$) pulse. Writing to the Command Register causes the chip to execute the command that is written. The Command Register can be read; however, the chip resets the Command Register when the SPC sets an Interrupt. Therefore, one cannot guarantee that the data in the register will be correct after loading an interrupting command or enabling selection or reselection. To be safe, a copy of the last command issued should be stored in the microprocessor's memory. Immediate commands are not stored.

4.3 CONTROL REGISTER

This eight-bit read/write register is used for enabling certain modes of operation for the SCSI Protocol Controller. The microprocessor reads from (writes to) the Control Register by activating $\overline{CS}$ with A3-A0 = 0010 and issuing a $\overline{RD}$ ($\overline{WR}$) pulse.

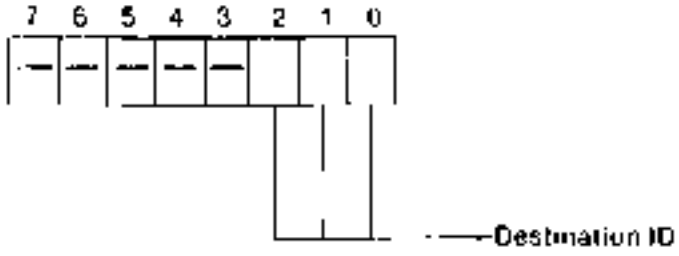


BIT	7-4	Reserved	
BIT	3	Valid Phase Enable	When this bit is a "1", the chip generates an interrupt for phase changes only when REQ becomes active. When bit 3 is a "0", the chip generates interrupts for phase changes that occur even though REQ may not be valid
BIT	2	Parity Enable	When the parity enable bit is a "1", the chip generates and checks parity on all transfers on the SCSI bus. When the parity enable bit is a "0", the chip generates but does not check parity on bus transfers.
BIT	1	Reselect Enable	When this bit is a "1", the chip will respond to any attempt by a Target to reselect it. When the bit is a "0", the chip will ignore all attempts to reselect it.
BIT	0	Select Enable	When this bit is a "1", the chip will respond to any attempts to select it as a Target. When it is a "0", the chip will ignore all selections.

NOTE: After being reset and completing self-diagnostics, the control register will contain all zeros.

4.4 DESTINATION ID REGISTER

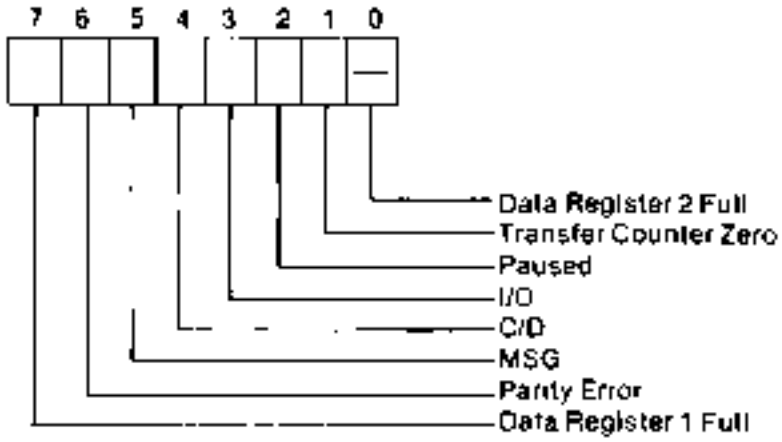
The Destination ID Register is an eight-bit register that is used to program the SCSI bus address of the destination device prior to issuing a Select or Reselect command to the chip. Bits 0-2 specify the address and bits 3-7 are always zeroes. The ID register is written (read) by activating $\overline{CS}$ with A3-A0 equal to "0011" and then pulsing $\overline{WR}$ ($\overline{RD}$).



4.5 AUXILIARY STATUS REGISTER

The Auxiliary Status Register is an eight-bit read-only register. It contains bits which indicate the status of the chip's operational condition. Some of these bits are used to determine the reason for interrupts. Therefore, the Auxiliary Status Register should always be read prior to reading the Interrupt Register when servicing interrupts. After the Interrupt Register is read, the Auxiliary Status Register bits needed to service the interrupt may change.

The Auxiliary Status Register is read by activating $\overline{CS}$ with A3-A0 = 0100 and then pulsing $\overline{RD}$. The individual bits of the Auxiliary Status Register are defined below.



BIT 7 Data Register 1 Full

This bit indicates the status of Data Register 1 and must be monitored by the microprocessor during non-DMA mode commands that use Data Register 1. When the DMA mode bit in the Command Register is off (0) and the command being executed is one of Send, Receive or Transfer Info commands (refer to Section 5.0, COMMANDS), data is transferred to (from) the chip by writing (reading) Data Register 1. Data Register 1 Full is set on (1) when data is written and turned off (0) when data is read. Therefore, Data Register 1 Full should be on before taking data from the chip, and off when sending data to the chip.

The Data Register 1 Full bits is always reset (to 0) at the time an interrupting type command is loaded into the Command Register. Therefore, when issuing such commands, the Command Register should be loaded prior to loading the Data Register 1 and monitoring the Data Register 1 Full flag.

BIT 6 Parity Error

When this bit is one, it indicates that the chip has detected a parity error on a byte of data received across the SCSI bus. It can be set when the chip is executing one of the Receive commands or the Transfer Info command (when the transfer is an input). This bit is reset after the Interrupt Register is read.

In Pass Parity mode, this bit will indicate whether MPU data had a parity error before being sent onto the SCSI bus. This parity error will not generate an interrupt to the MPU, but the receiving SCSI device will indicate an error was detected. The flag is useful in identifying where the error occurred.

BIT 3-5 I/O, C/D, MSG

These bits indicate the status of the SCSI I/O, C/D, and MSG signals at all times. They define the Information Phase type being requested by the Target. These signals are significant when servicing interrupts and the chip is logically connected to the bus in the Initiator role. An interrupt will occur with any phase change. This allows the Initiator to prepare for the next phase of data transfer. These bits are only held while INT is active. The bits are coded as follows:

I/O	C/D	MSG	BUS PHASE
0	0	0	Data Out
0	0	1	Unspecified Info Out
0	1	0	Command
0	1	1	Message Out
1	0	0	Data In
1	0	1	Unspecified Info In
1	1	0	Status
1	1	1	Message In

BIT	2	**Paused**	When on (1), this bit indicates that the chip has aborted the command being executed in response to the Pause command. It is turned off when the Interrupting type command code is loaded into the Command Register.
BIT	1	Transfer Counter Zero	This bit is provided to indicate the status of the 24-bit Transfer Counter. When on (1), it indicates that the Transfer Counter is equal to zero. It is intended to facilitate interrupt servicing. This bit is also set when the Single-byte bit in the Command Register is active.
BIT	0	Data Register 2 Full	Since the 5386 design includes a doubly-buffered data register, this bit is used to indicate if data has been transferred into the second data register. If bit 0 is a "1", Data Register 2 is full; if this bit is a "0", Data Register 2 is empty.

NOTE The Auxiliary Status Register will contain the following pattern after a Reset and self-diagnostics: 00xxx010.

4.6 ID REGISTER

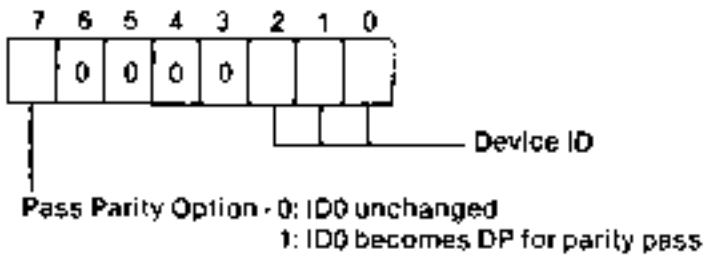
The ID Register operates in two configurations, the "strapped ID" mode or the "programmed ID" mode. If the ID register is written before the Control Register is written, the NCR 5386 assumes the "programmed ID" mode. In the "programmed ID" mode, pin 14 becomes the data bus parity signal (DP) and pins 12 and 13 are not used. If the Control Register is initialized and the ID register has not been written previously, then the device will assume the "strapped ID" mode which is identical to the NCR 5385E operation.

Strapped ID Configuration

The ID Register is an eight-bit read-only register in the "strapped ID" mode, which indicates the logical SCSI bus address occupied by the chip. Bits 0-2 directly reflect the logical inversion of the chip ID input signals ID0-ID2; the ID Register is active high whereas the ID input signals are active low. The ID Register allows the microprocessor to read the chip's SCSI bus address which would normally be strapped in hardware. Bits 3-6 of the ID Register will always be zeros. The ID Register is read by activating CS with A3-A0 equal to 0101 and then pulsing RD.

Programmed ID Configuration

The ID Register is an eight-bit read/write register in the "programmed ID" mode, which is intended to be programmed by the system MPU. Bits 0-2 directly reflect the logical SCSI device ID. Bit 7 indicates if pass parity feature is enabled. In the "programmed ID" mode, pin 14 becomes the parity signal for the data bus interface (DP) only if bit 7 is a "1", and pins 12 and 13 are not used.

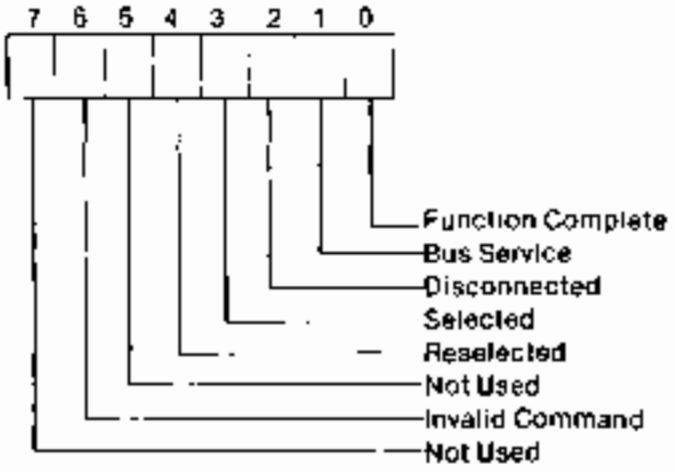


4.7 INTERRUPT REGISTER

The Interrupt Register is an eight-bit read-only register. It is used in conjunction with the Auxiliary Status Register to determine the reason for an interrupt condition. This register is read by activating $\overline{CS}$ with A3-A0 = 0110 and then pulsing $\overline{RD}$. When the Interrupt Register is read, it automatically resets itself (after the read is complete) and enables the chip for a new interrupt condition. Since the Parity Error bit in the Auxiliary Status Register is reset after a read of the Interrupt Register, and since I/O, C/D, and MSG are only held while INT is active, the Auxiliary Status Register should always be read prior to reading the Interrupt Register.

If a Selected or Reselected interrupt occurs after issuing a command that would normally cause an interrupt, the chip will ignore the last command issued. This allows the microprocessor to service the Selected or Reselected interrupt prior to proceeding with the other operation. An example of this situation is when the microprocessor issues a command to select a Target at about the same time another Target reselects the chip. If the chip sees the reselection first, the microprocessor will receive an interrupt for the reselection, and the chip will ignore the Select command, which would now be invalid since the chip is now logically connected on the SCSI bus to another device.

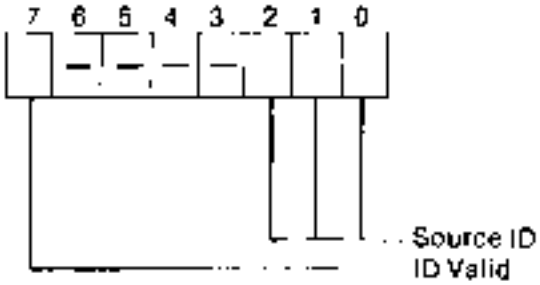
Individual interrupt conditions are described below. (Note: that for all cases, an interrupt condition is on, when the corresponding bit is a one (1), and off when zero (0).)



BIT 7	Not Used	May be either (1) or (0)
BIT 6	Invalid Command	When on (1), this bit indicates that the last command loaded into the Command Register is not valid.
BIT 5	Not Used	(Reserved)
BIT 4	Reselected *	This interrupt will be on (1) when the chip has been reselected by another SCSI device. After setting this interrupt, the chip is logically connected to the bus in an Initiator role and is waiting for the Target to send REQ or disconnect from the bus
BIT 3	Selected *	This interrupt will be on (1) whenever the chip has been selected by another SCSI device. After setting this interrupt, the chip is logically connected to the bus in the Target role and is waiting for a command to be loaded into the Command Register. * The chip will become selected (reselected) only if the ID data byte put on the SCSI bus during the Selection (Reselection) Phase has good parity and not more than one ID other than the chip's own ID is on.
BIT 2	Disconnected	This interrupt will be set on (1) when the chip is connected to the bus in the Initiator role and the Target disconnects or when the chip is executing a Select or Reselect command and the destination device does not respond before the Transfer Counter times out.
BIT 1	Bus Service	When the chip is logically connected to the bus in the Initiator role, this bit will be set on (1) whenever the Target sends a REQ which the chip cannot automatically handle. This happens when the first REQ for connection is received or when the chip is executing a Transfer Info or Transfer Pad command and either the Transfer Counter is zero or the Target changes the Information Phase type. If the valid phase enable bit is not set, Bus Service interrupt may be set if a phase change occurs *before* REQ is seen; see "valid phase enable," section 4.3. This early notification will allow the Initiator extra time to prepare for a phase change in some systems. (Note: the chip may Generate Bus Service Interrupts for phases that never request transfers. This is not an error condition, merely transitional status of I/O, C/D, and MSG.) If the chip is logically connected in the Target role, this bit will be set on (1) whenever the Initiator asserts ATN. When indicating ATN the Bus Service Interrupt may occur with a Selected Interrupt or with a Function Complete interrupt.
BIT 0	Function Complete	When this bit is on (1), it indicates that the last interrupting command has completed. It is the normal successful completion interrupt for Select, Reselect, Send and Receive commands (Refer to Section 5.0, COMMANDS). During any of the Receive commands, it is set on (1) along with the parity error bit as soon as a parity error is detected. A Bus Service Interrupt may also occur simultaneously with the Function Complete if an ATN signal was activated during a Send or a Receive command. The Function Complete interrupt is also generated at the end of a Message In phase for a Transfer Info command. (See TRANSFER INFO command for details.)

4.8 SOURCE ID REGISTER

The Source ID Register is an eight-bit read-only register which contains the three-bit encoded ID of the last device which Selected or Reselected the chip. The following is the format of the Source ID Register



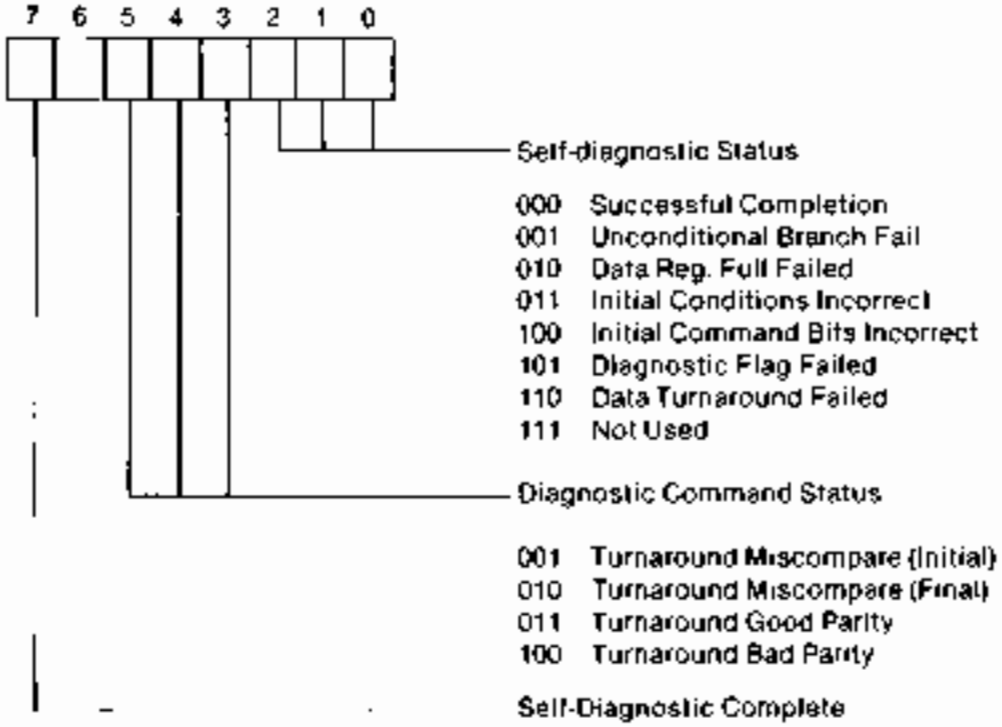
The ID Valid bit indicates that the source device placed its own ID bit on the SCSI bus during the Selection Phase. The SPC chip has encoded the source ID and placed it in bits 2-0. This information remains valid until the chip disconnects from the SCSI bus, at this time the ID Valid bit is reset.

4.9 DATA REGISTER 2

Data Register 2 is an eight-bit read-only register which may contain data that has been sent to the chip but not transferred across the bus. Bit 0 in the Auxiliary Status Register is used to determine if this register is full. The microprocessor can read Data Register 2 by activating $\overline{CS}$ with A3-A0 equal to 1000 and issuing a $\overline{RD}$ pulse.

4.10 DIAGNOSTIC STATUS REGISTER

The Diagnostic Status Register is an eight-bit read-only register which indicates the result of self-diagnostics and the last diagnostic command issued to the chip. The format of the Diagnostic Status Register is shown below



Bit 7 = 1 indicates that self-diagnostics have been completed. (NOTE: A reset will clear bits 6-3 if possible) After a reset to the chip, the microprocessor should make sure that the Diagnostic Status Register contains the following pattern before attempting any commands: 10000000. This code indicates self-diagnostics are complete and no errors were detected. After a diagnostic command has been executed, bits 6-3 will contain the resulting status, but bit 7 and bits 2-0 are not affected.

The microprocessor may read the Diagnostic Status Register by activating $\overline{CS}$ with A3-A0 = 1001 and issuing a $\overline{RD}$ pulse.

If an error is detected during self-diagnostics, the proper status is loaded into the Diagnostic Status Register and the chip halts until a Reset command or a Reset signal is asserted. Refer to the Self-Diagnostic Status Code Summary for an explanation of the individual codes.

When a diagnostic command is issued to the chip, the chip will attempt to perform the function, load a status into bits 6-3, and initiate a Function Complete Interrupt

4.10.1 SELF-DIAGNOSTIC STATUS CODE SUMMARY

000 - Successful Completion. The chip executed all self-diagnostics following a reset and detected no errors.

001 - Unconditional Branch Failed. The chip's internal sequencer attempted an unconditional branch and failed to reach the desired location.

010 - Data Register Full Failed. The chip attempted to set and reset the Data Register Full status bit in the Interrupt Register and failed.

011 - Initial Conditions Incorrect. The chip detected one of its internal initial conditions in the wrong state.

100 - Initial Command Bits Incorrect. The chip tested bits 6,4,2,1 and 0 of the Command Register and found at least one was not zero.

101 - Diagnostic Flag Failed. The chip failed in its attempt to set and reset its internal diagnostic flag.

110 - Data Turnaround Failed. During self-diagnostics the chip attempts to flush several bytes of data through its internal data paths. It also attempts to set and reset the Parity Error bit in the Interrupt Status Register. This status indicates that one of these operations failed.

4.11 TRANSFER COUNTER (THREE EIGHT-BIT COUNTERS)

The Transfer Counter is comprised of three, eight-bit register/counters. It is used by the chip for Send, Receive and Transfer commands that require more than a single byte of data to be transferred. It may also be used with Select and Reselect commands to set a timeout for no response. To write to (read from) the Transfer Counter, $\overline{CS}$ is activated with A3-A0 selecting a byte and then pulsing $\overline{WR}$ ($\overline{RD}$). The Transfer Counter is addressed as shown below.

A3	A2	A1	A0	SELECTED BYTE
1	1	0	0	Most Significant Byte
1	1	0	1	Middle Byte
1	1	1	0	Least Significant Byte

For Send, Receive and Transfer commands with single-byte not specified, the Transfer Counter specifies to the chip the maximum number of bytes to be sent or received before interrupting. The Transfer Counter must be loaded prior to issuing the command. When single-byte is specified, the chip neither uses nor alters the Transfer Counter. To facilitate servicing interrupts for commands that use the Transfer Counter, a bit is provided in the Auxiliary Status Register to indicate when the Transfer Counter is zero.

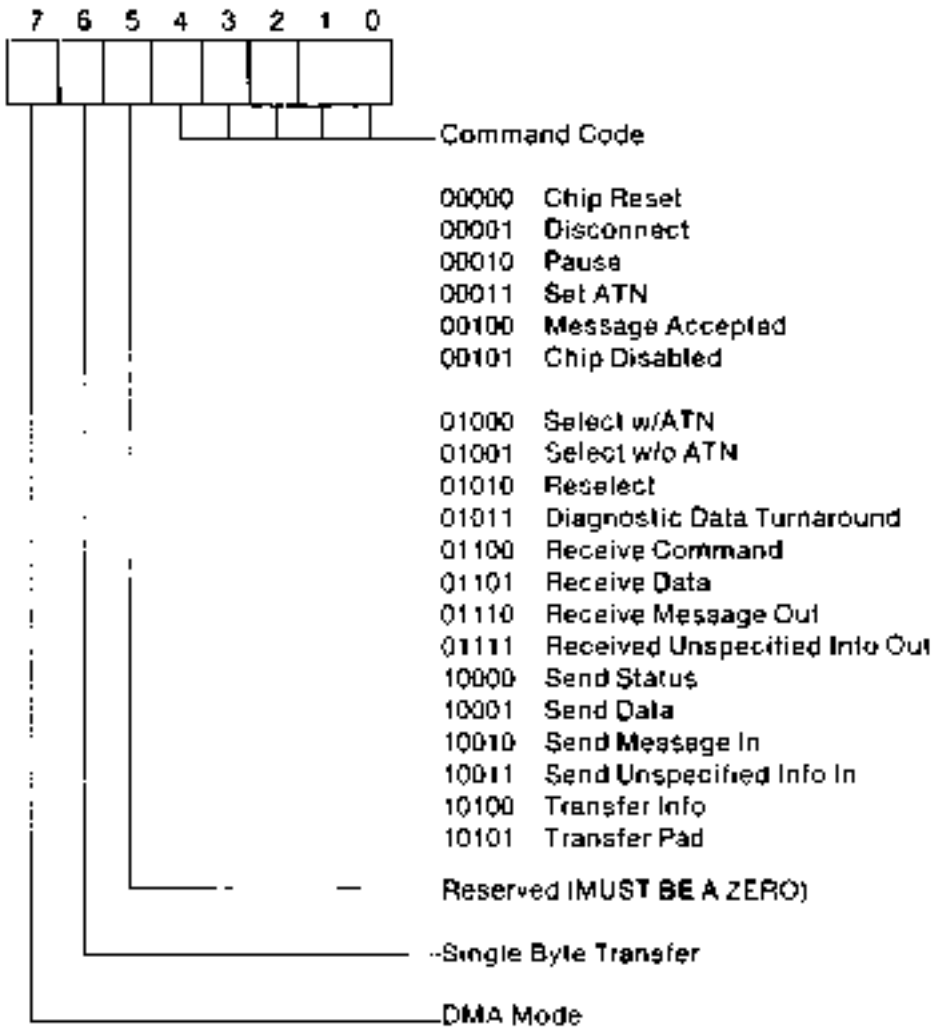
For Select and Reselect commands, the Transfer Counter specifies the number of time intervals (1024 CLK periods) that the chip will wait before automatically aborting the command due to no response (BSY) from the destination device. The Transfer Counter must be loaded prior to issuing the command. If the Transfer Counter is loaded with all zeroes, the timeout logic in the chip will be disabled, and the chip will not automatically abort the command due to no response.

SECTION 5
COMMANDS

This section defines command format, types, codes and operation. Commands are given to the chip by loading the Command Register.

5.1 COMMAND FORMAT

The bits in the Command Register are defined as follows.



BIT	7	DMA Mode	This bit is applicable only for commands that use the Data Register. When this bit is on (1), it indicates that data will be transferred to (from) the Data Register using the DMA signals DREQ and $\overline{DACK}$. When it is off (0), the microprocessor must monitor the state of the Data Register Full flag in the Auxillary Status Register. Data is then transferred by using the appropriate input/output command.
BIT	6	Single Byte Transfer	When on (1), this bit indicates that only one byte of data is to be transferred for this command. The Transfer Counter will not be used or altered by the chip. Therefore, for common single byte message and status transfers, the Transfer Counter does not need to be loaded prior to issuing a command with this bit set. When this bit is off (0), the Transfer Counter is used by the chip to determine the length of the transfer for the command.
BIT	5	Reserved	This bit is not used and should always be programmed off (0).
BIT	4-0	Command Code	These bits are used to specify the command to be executed.

5.2 COMMAND TYPES

There are two types of commands; Immediate and Interrupting. All of the immediate commands, except for Pause, cause immediate results within three clock cycles from the time the Command Register is loaded. The Pause command is explained in a later section (See PAUSE). Interrupting commands do not result in immediate action. Their completion is always flagged by an interrupt.

Command codes 00000-00111 specify immediate commands. Immediate commands that are listed as reserved, will be ignored if issued to the SPC chip. Command codes 01000-10101 specify interrupting commands. When one of these codes is loaded into the Command Register, a second interrupting command code should not be loaded until after the interrupt has occurred for the first command. However, an immediate type command may be loaded before the interrupt for an interrupting command occurs. If a reserved interrupting command code is issued, the chip will respond with an Invalid Command interrupt.

5.3 INVALID COMMANDS

The user of the chip can be in one of three states at any particular time: Disconnected, connected as an Initiator, or connected as a Target. Commands are valid only in specified states. If an invalid Immediate command is issued, the chip will ignore the command. If an Interrupting command is issued in an invalid state, or a reserved Interrupting command code is issued, an Invalid Command interrupt will result. The exceptions are described below:

The microprocessor must never issue any interrupting type command when the chip is not expecting such a command. Unpredictable results will occur in this case. The following is a list of user states in which the chip is not expecting an interrupting command:

1. The chip is currently processing an Interrupting type command and has not yet set the interrupt to signal the completion.
2. The chip is currently processing an Interrupting type command, a Pause command has been issued but the Paused bit in the Auxiliary Status Register has not been set.
3. The chip is connected as an Initiator, but the Target has not yet requested an Information Transfer.
4. The chip has completed a Transfer Info or Transfer Pad command and the Target has not requested additional Information or has not changed the Information Phase.

In user states three and four, described above, the microprocessor must wait for a Bus Service, Disconnected, or Function Complete interrupt.

If an interrupting command is illegitimately issued in these states, no interrupt will occur for it, and it is likely that the current function will be altered.

5.4 COMMAND SUMMARY

Below is a summary that lists all commands. In the table the following abbreviations are used.

INT = INTERRUPTING D = DISCONNECTED I = CONNECTED AS AN INITIATOR
IMM = IMMEDIATE T = CONNECTED AS A TARGET

COMMAND CODE	COMMAND	TYPE	VALID STATES
00000	Chip Reset	IMM	D,I,T
00001	Disconnect	IMM	I,T
00010	Paused	IMM	D,T
00011	Set ATN	IMM	I
00100	Message Accepted	IMM	I
00101	Chip Disable	IMM	D,I,T
00110-00111	Reserved	IMM	
01000	Select w/ATN	INT	D
01001	Select w/o ATN	INT	D
01010	Reselect	INT	D
01011	Diagnostic	INT	D
01100	Receive Command	INT	T
01101	Receive Data	INT	T
01110	Receive Message Out	INT	T
01111	Receive Unspecified Info Out	INT	T
10000	Send Status	INT	T
10001	Send Data	INT	T
10010	Send Message In	INT	T
10011	Send Unspecified Info In	INT	T
10100	Transfer Info	INT	I
10101	Transfer Pad	INT	I
10110-11111	Reserved	INT	

5.5 COMMAND DEFINITIONS

5.5.1 CHIP RESET

Chip Reset immediately stops any chip operation and resets all registers, counters, etc. on the chip. It performs the same operation as the hardware "reset" input.

5.5.2 DISCONNECT

Upon receipt of this command, the chip immediately releases all SCSI bus signals and returns to a Disconnected idle state. For the Initiator role, this is the normal method of disconnecting from the bus when a transfer is complete. For the Initiator role, Disconnect may be used to release the bus signals as a result of a timeout condition. In this case, the chip ignores the Target and is left in the Disconnected state. For the Disconnected state, it is not valid to issue a Disconnect command. If issued, the chip will ignore this command.

5.5.3 PAUSE

Pause is an Immediate command that is valid in the Disconnected state or when logically connected to the bus as a Target device. Pause is not valid when connected as an Initiator.

When connected as a Target, the Pause command provides a means of halting a Send or Receive command without having to wait for the transfer to complete. When Pause is issued, it immediately sets a flag in the chip. Within one byte transfer cycle, the chip recognizes the flag, aborts the Send or Receive operation, and then sets the Paused status bit in the Auxiliary Status Register. At this time, the chip is still connected to the bus in the Target role, and it is waiting for another command.

The Pause command stops the Send or Receive command in an orderly manner leaving the Transfer Counter in a valid state that indicates the remaining number of bytes to be transferred. Also no REQ or ACK is asserted on the bus and no data is left in the chip waiting to be transferred. An operation that is paused may be resumed, if desired, simply by reloading the original command into the Command Register. (Note: after issuing the Pause while executing Send or Receive, it is necessary to continue transferring data with the chip (due to double-buffering) until the Paused status bit is set or an interrupt occurs.)

When in the disconnected state, Pause may be issued to abort a Select or Reselect command. After a Select or Reselect command is issued and before an interrupt occurs, a Pause command may be issued to abort the operation. The Pause command immediately sets an internal flag. If the chip has not yet won arbitration, it sets the Paused bit in the Auxiliary Status Register and waits in the disconnected state for another command. If the chip has won arbitration, it releases the bus by dropping the two ID bits with SELOUT on for a minimum of 100 μs, checks for no BSYIN, and then releases the bus. After this procedure, it sets the Paused bit in the Auxiliary Status Register and waits for another command in the Disconnected state.

Since Pause is an Immediate command, it does not cause an interrupt. As previously noted, the chip sets the Paused status bit to indicate that is has been executed. If an interrupt-causing event occurs before the chip sees the pause flag set, the chip will set the Interrupt. In this case, the Paused status bit is not set by the chip either before or after the Interrupt. In all cases, an interrupt-causing event will take precedence over Pause. For example, in the Target role if ATN is on when Pause is issued, a Bus Service interrupt will occur and the Paused status bit will not be set.

If the Pause command is issued when the chip is Disconnected, the Paused status bit will be set by the chip, provided it has not already detected a Selection or Reselection.

5.5.4 SET ATN

The Set ATN command causes ATN to be asserted immediately if the chip is connected as an Initiator. This command is invalid and ignored if issued when the chip is Disconnected or is operating in a Target role. The ATN signal is de-asserted in a Message Out phase when the transfer count becomes zero or one byte has been transferred (in a one-byte transfer command) during the execution of a Transfer Info command.

The chip automatically sets ATN in two cases:

1. If a Select w/ATN command is issued and arbitration is won.
2. If a parity error is detected on an input byte during execution of a Transfer Info command.

5.5.5 MESSAGE ACCEPTED

The Message Accepted command is an Immediate command that is valid only when connected as an Initiator. It is used after a Transfer Info or Pad command (See pages 29, 30 TRANSFER INFO and TRANSFER PAD) to indicate to the chip that ACK can be de-asserted for the last byte.

When an Initiator receives a message, a Transfer command is used. If the transfer is an input (I/O = 1) and the information is a message (MSG = 1, C/D = 1), the chip interrupts after receiving the last byte with a Function Complete interrupt. For this one special case, the chip also leaves ACK asserted on the bus. By interrupting and leaving ACK asserted, the chip gives the microprocessor a chance to interpret the message and set ATN, prior to ACK being de-asserted. This allows the chip to properly request a Message Out phase if the Initiator wants to send a "Reject Message" to the Target.

Message Accepted must always be issued after a Transfer Info for a Message In phase, whether or not Set ATN is issued, in order to have the chip de-assert ACK. If the Initiator wants to reject the message, Set ATN would be issued first followed by Message Accepted. If the message is not to be rejected, only Message Accepted is issued. (Note: until Message Accepted is issued, the Target will not send another REQ since ACK is still asserted.)

5.5.6 CHIP DISABLE

Chip Disable immediately stops all chip operations and logically disconnects it from the circuit. All outputs will be placed in a high impedence state and the chip will not respond to any commands (other than chip reset). The chip will also not respond to any activity on the SCSI bus. The only way to exit this condition is to activate the "reset" input or issue a Reset command.

5.5.7 SELECT w/ATN

This command causes the chip to attempt to select a Target. It may only be used if the SPC is in the Disconnected state. Any attempt to issue this command in another state will result in an Invalid Command interrupt. Before issuing this command, the microprocessor must load the Transfer Counter for a timeout on the Target's response. This value is computed according to the following formula:

$$\text{Transfer Counter} = \text{Desired Timeout} / (1024 \times \text{Clock Period})$$

If the Transfer Counter is loaded with the value zero, the chip will wait indefinitely for a response from the Target being selected.

The microprocessor must also load the Destination ID Register with the three-bit code of the Target to be selected before issuing the Select w/ATN command.

When the chip detects the Select w/ATN command, it begins by attempting to arbitrate for control of the SCSI bus. If, at any time during arbitration the chip becomes selected or reselected, the Select w/ATN is aborted and forgotten and the chip will interrupt with one of the following conditions:

1. Selected
2. Selected and Bus Service
3. Reselected

If arbitration is won, the chip places the SCSI bus in the Selection phase with ATN asserted, and uses the Destination ID Register to identify the desired Target. At the same time, the chip begins a timer based on the value computed above. If the Target does not respond within the timeout period, the chip will disconnect from the bus and interrupt with the Disconnected flag set in the Interrupt Register. (Note: The microprocessor should never monitor the Transfer Counter Zero flag in the Auxiliary Status Register to determine when a timeout has occurred.) If the Target responds within the allotted time, the chip will interrupt with a Function Complete status. Control of the SCSI bus then belongs to the selected Target and after the interrupt status has been read, another interrupt may occur indicating either that the Target has disconnected or is requesting a transfer.

If the timeout is disabled and the Target does not respond, or if arbitration is not won, the only way to abort the Select w/ATN command is to issue the Pause command. After the Pause command is issued, it is still possible that the Function Complete or Disconnect interrupts may occur. This happens if one of the interrupts get set before the chip detects the Pause command, or if the Target responds while the chip is sequencing off the SCSI bus in a timeout condition. If the chip does not set either interrupt, it will set the Paused bit in the Auxiliary Status Register. If the microprocessor detects this bit after issuing the Pause command, then it is assured that the chip aborted the Select w/ATN command and no connection exists.

5.5.8 SELECT w/o ATN

The Select w/o ATN is identical to the Select w/ATN command except that the ATN signal is not asserted during the Selection phase.

5.5.9 RESELECT

This command causes the chip to attempt to reselect an Initiator. It may only be used if the microprocessor is in the Disconnected state. Any attempt to issue this command in another state will result in an Invalid Command interrupt. Before issuing this command, the microprocessor must load the Transfer Counter for a timeout on the Initiator's response. This value is computed according to the following formula:

$$\text{Transfer Counter} = \text{Desired Timeout} / (1024 \times \text{Clock Period})$$

If the Transfer Counter is loaded with the value zero, the chip will wait indefinitely for a response from the Initiator being reselected.

The microprocessor must also load the Destination ID Register with the three-bit code of the Initiator to be reselected before issuing the Reselect command.

When the chip detects the Reselect command, it begins by attempting to arbitrate for control of the SCSI bus. If, at any time during arbitration, the chip becomes selected or reselected, the Reselect is aborted and forgotten and the chip will interrupt with one of the following conditions:

1. Selected
2. Selected and Bus Service
3. Reselected

If arbitration is won, the chip places the SCSI bus in the Reselection phase using the Destination ID Register to identify the desired Initiator. At the same time, the chip begins a timer based on the value computed above. If the Initiator does not respond within the timeout period, the chip will disconnect from the bus and interrupt with the Disconnected flag set in the Interrupt Register. (Note: The microprocessor should never monitor the Transfer Counter Zero flag in the Auxiliary Status Register to determine when a timeout has occurred.) If the Initiator responds within the allotted time, the chip will interrupt with a Function Complete status. The chip (acting as the Target) is then in control of the SCSI bus, and waits for the Interrupt Register to be read by the microprocessor. After it has been read, the chip waits for a command from the microprocessor or ATN from the Initiator. If the ATN occurs, the chip will set the Bus Service interrupt. This interrupt may happen immediately after a command has been issued due to internal timing. In this case, the chip waits for the Interrupt Register to be read and the command is ignored. The chip then waits for a new command.

If the timeout is disabled and the Initiator does not respond, or if arbitration is not won, the only way to abort the Reselect command is to issue the Pause command. After the Pause command is issued, it is still possible that the Function Complete or Disconnected interrupts may occur. This happens if one of the interrupts get set before the chip detects the Pause command, or if the Initiator responds while the chip is sequencing off the SCSI bus in a timeout condition. If the chip does not set either interrupt, it will set the Paused bit in the Auxiliary Status Register. If the microprocessor detects this bit after issuing the Pause command, then it is assured that the chip aborted the Reselect command and no connection exists.

5.5.10 DIAGNOSTIC (DATA TURNAROUND)

This Interrupting command causes the chip to attempt to turn a data byte around through its internal data paths. When the command is loaded into the Command Register the Data Register Full bit is reset. The microprocessor then writes one byte into the Data Register. The chip moves the byte to another register and compares the contents of the Data Register. The byte is then moved to a third register (the SCSI output register) and good parity is generated if bit 6 of the command is off (0); bad parity is generated if bit 6 is on (1). Finally, the chip moves the byte back to the Data Register and compares it with the contents of the second register. Based on these comparisons and parity checking, the chip stores a result into the Diagnostic Status Register and sets the Function Complete interrupt. After reading the Interrupt Register, the microprocessor should make sure the Data Register Full bit is on (1) and read the contents of the Data Register. If the Data Register Full bit is not on (0), then an error has occurred. The following is a list of codes which are loaded into bits 6-3 of the Diagnostic Status Register as a result of this command.

BIT 6543	RESULT
0001	Data Miscompare (INITIAL)
0010	Data Miscompare (FINAL)
0011	Good Parity Detected
0100	Bad Parity Detected

5.5.11 RECEIVE COMMANDS

The Receive commands are interrupting commands that are valid only when connected as a Target device. They are used by the Target to receive commands, data, and message information from an Initiator.

The Receive commands transfer data; therefore, the Single Byte Transfer and DMA mode bits in the Command Register are valid for these commands. If the Single Byte Transfer bit is off (0), the Transfer Counter must be loaded before a Receive command is issued to the chip. In this case, the chip uses the Transfer Counter to determine the number of bytes to receive.

When a Receive command is issued, the chip immediately resets the Data Register Full bit in the Auxiliary Status Register. The chip then drives the I/O, C/D, and MSG outputs for the proper information phase as follows:

COMMAND NAME	I/O	C/D	MSG
Receive Command	0	1	0
Receive Data	0	0	0
Receive Message Out	0	1	1
Receive Unspecified Info Out	0	0	1

The chip then proceeds to request and receive the specified number of information bytes. The DMA mode bit in the Command Register determines how the chip transfers these bytes from its Data Register to the microprocessor.

When a Receive command is terminated, the chip generates an interrupt. The following two events can cause termination:

1. The operation completes successfully; the Transfer Counter is zero. This event results in a Function Complete interrupt with the Parity Error bit in the Auxiliary Status Register off (0). If the Initiator activated ATN during the operation, the Bus Service bit will also be on.
2. A Parity Error occurs. The last byte transferred is the byte that caused the error. This event causes a Function Complete interrupt with the Parity Error bit in the Auxiliary Status Register on (1). If the Initiator activated ATN during the operation, the Bus Service bit will also be on.

After any of the interrupts, the chip is always left in the connected Target state. The Transfer Counter indicates the number of bytes remaining to be transferred (zero if completed successfully), and the Data Register is empty (the last byte received is sent to the microprocessor). Also, ACK and REQ are inactive on the bus.

(Note: if a Bus Service interrupt alone occurs after issuing a Receive command, the Initiator activated ATN before the chip began executing the command. In this case, the command is ignored by the chip.)

A Receive command may be stopped prior to an interrupt causing event by issuing a Pause command. Operation of the Pause command is explained in an earlier section. In the event the Initiator does not respond, or stops responding, the chip is left in a state where it cannot respond to a Pause command. For this case, a Disconnect command can be used to abort the command and the connection. The Disconnect command is explained in an earlier section.

5.5.12 SEND COMMANDS

The Send commands are interrupting commands that are valid only when connected to the bus in the Target role. They are used by a Target to send status, data, and message information to an Initiator.

The Send commands transfer data, and therefore, the Single Byte Transfer and DMA mode bits in the Command Register are valid for these commands. If the Single Byte Transfer bit is off(0), the Transfer Counter must be loaded before a Send command is issued to the chip. In this case, the chip uses the Transfer Counter to determine the number of bytes to send.

When a Send command is issued, the chip immediately resets the Data Register Full bit in the Auxiliary Status Register. Therefore, the first byte of data for the transfer cannot be put into the Data Register until after a Send command is loaded into the Command Register.

In executing a Send command, the chip drives the I/O, C/D, and MSG outputs for the proper information phase. These lines are logically driven for each Send command as shown below.

COMMAND NAME	I/O	C/D	MSG
Send Status	1	1	0
Send Data	1	0	0
Send Message In	1	1	1
Send Unspecified Info In	1	0	1

After resetting Data Register Full and driving I/O, C/D, and MSG, the chip then proceeds to monitor Data Register Full, take the data from the Data Register, and send it to the Initiator. The DMA mode bit in the Command Register specifies how the data is loaded into the chip.

After interrupting, the chip is left in the connected Target state, and ACK and REQ are inactive on the bus. When the transfer is complete, the chip interrupts with a Function Complete Interrupt. If the Initiator activated ATN during the transfer, a Bus Service bit will also be set by the chip.

(NOTE: If a Bus Service interrupt alone occurs after issuing a Send command, the Initiator activated ATN before the chip began executing the command. In this case, the command is ignored by the chip.)

A Send command may be stopped prior to an interrupt causing event by issuing a Pause command. Operation of the Pause command is explained in an earlier section. In the event the Initiator does not, or stops responding, the chip is left in a state where it cannot respond to a Pause command. For this case, a Disconnect command can be used to abort the command and the connection. The Disconnect command is explained in a earlier section.

5.5.13 TRANSFER INFO

The Transfer Info command is an interrupting command that is valid only when connected to the bus in the Initiator role. It is used by the Initiator for all information transfers across the SCSI bus.

A Transfer Info command is issued by an Initiator in response to a Bus Service Interrupt. The Bus Service interrupt, as explained in a previous section, is received by the connected Initiator upon the following conditions: receiving the first REQ from a Target, a previous command has completed and the Target changes phases, the Target changes phases before termination, or when a previous command has completed and the Target is requesting more information. It is not valid to issue a Transfer Info command without having a Bus Service interrupt, because the Target requests and controls all transfers. The chip will only permit one Transfer Info or Transfer Pad per Bus Service interrupt.

After an Initiator receives a Bus Service interrupt, and prior to issuing a Transfer Info command, the I/O, C/D, and MSG bits from the Auxiliary Status Register (read prior to reading the Interrupt) should be examined to determine the type of information phase and the direction of transfer requested by the Target. The Initiator then prepares for the transfer. If the Single Byte Transfer bit is not going to be set in the Command Register, the Transfer Counter must be loaded prior to issuing the Transfer Info command. This is done in order to specify to the chip the maximum number of bytes to be transferred.

When a Transfer Info is issued, the chip immediately resets the Data Register Full bit in the Auxiliary Status Register. For this reason, the first byte of data for an output operation cannot be loaded into the Data Register until after the command is loaded into the Command Register. The chip then proceeds with the transfer, expecting data to be read from (input), or written to (output), its Data Register as indicated by the DMA Mode bit in the Command Register. The chip automatically detects the direction of the transfer from the I/O bit which is stored in the Auxiliary Status Register.

The chip continues a transfer until an interrupt causing event occurs. The following four events will cause the chip to terminate and interrupt.

1. The maximum number of bytes specified have been transferred and the Target activated REQ or the Information Phase changed. This event results in a Bus Service Interrupt. Either single byte transfer was specified or the Transfer Counter is zero as indicated by a bit in the Auxiliary Status Register. The Target may or may not have changed the information phase type. The I/O, C/D, and MSG bits in the Auxiliary Status Register need to be examined at the time of the interrupt to determine what phase the Target is requesting.

   (NOTE: Due to early notification of the phase change, a phase may be selected spuriously and not transfer any data. The microprocessor should not consider this an error condition. The early notification can be blocked with "valid phase bit".)

2. The Target changes the information phase type before the maximum number of bytes are transferred. This event also causes a Bus Service interrupt. The new information phase may be determined by examining the I/O, C/D, and MSG bits in the Auxiliary Status Register. The Transfer Counter may be read at the time of the interrupt to determine the number of bytes remaining to be transferred. When this interrupt occurs for an output transfer, the chip may take one more byte from the microprocessor than it transfers, because of pre-fetching. However, the Transfer Counter still reflects the number of bytes remaining to be transferred.

3. The Target releases the bus by dropping BSY. This event results in a Disconnected interrupt. Following this interrupt, the chip is no longer in the Initiator role. It now remains in the Disconnected state.

4. The last byte of a Message Input phase has been received. This event results in a Function Complete interrupt. For this case, ACK is left active on the bus to allow the microprocessor to Set ATN for the purpose of rejecting the message. After this interrupt is received and a Set ATN is issued (if desired), a Message Accepted must be issued to turn off ACK for the last byte of the Message In phase.

For Input transfers (I/O = 1), the chip checks parity for each byte received if the Parity Enable bit in the Control Register is on. When checking parity and the parity error occurs, the chip activates ATN prior to deactivating ACK for the byte that causes the error. It also turns on the Parity Error bit in the Auxiliary Status Register. The parity error, however, does not result in an interrupt. The chip waits for one of the four events listed above before interrupting. Therefore, the Parity Error bit should be examined when servicing any interrupt after issuing Transfer Info command for an Input transfer.

If ATN is asserted by the chip, either because of a parity error or because a SET ATN command is issued, the ATN will remain asserted until the end of the connection, or until a Message Out is transferred. Therefore, during each cycle of a Transfer Info operation for output, the chip checks for a message phase (C/D = 1, MSG = 1) and also either a single byte transfer or the Transfer Counter set at zero. If these conditions exist, the chip turns off ATN prior to activating ACK for the last byte of the message.

As previously stated, a Transfer Info normally terminates with an interrupt. If a Transfer Info command must be aborted, possibly because of a timeout violation, either a Chip Reset or a Disconnect command can be used. It is noted, however, that although these commands will force the chip into a disconnected state, the Target device is left on the bus. A SCSI bus reset, which is not a chip function, is the only way an Initiator can force a Target to disconnect.

5.5.14 TRANSFER PAD

The Transfer Pad command is an interrupting command that is valid only when connected to the bus as an Initiator. It is similiar to the Transfer Info command except that the data transfer between the chip and the microprocessor bus will be different.

Transfer Pad can be used by an Initiator to continue handshaking with a Target without giving data to, or taking data from, the chip. This may be useful if the Target requests an invalid Information Transfer Phase. The chip operates in the same manner as it does for a Transfer Info command, except that for output transfers it takes only one byte of data from the microprocessor and sends the same byte repeatedly until the transfer terminates. For input transfers, it accepts data from the SCSI bus but does not check parity or send it to the microprocessor. Though data is not exchanged with the microprocessor bus, the Transfer Counter is still used by the chip so that a maximum number of pad bytes can be specified.

Protocol for using a Transfer Pad command is the same as the Transfer Info except that the DMA Mode bit has significance only for output transfers. The Transfer Pad terminates because of the same four events that cause a Transfer Info command to terminate. Also, similar to the Transfer Info command, Chip Reset and Disconnect can be used to abort the command.

SECTION 6
BUS INITIATED FUNCTIONS

6.1 SELECTION

If the Select Enable bit in the Control Register is on, the chip may be selected by another SCSI device to be a TARGET for an I/O operation. Selection occurs in the chip only if all the following conditions exist: SELOUT = 0, BSYIN = 0, SELIN = 1, I/O = 0, the chip's ID bit is asserted by the selecting device on the data bus, no more than one other ID bit (the Initiator's) is asserted on the data bus and data bus parity is good.

When all of these conditions exist, the chip is selected. It then encodes the Initiator's ID and loads it into bits 2-0 of the Source ID Register. The chip also detects whether or not the Initiator asserted its ID during selection, and either sets or resets the ID Valid bit in the Source ID Register.

The chip then asserts BSYOUT, waits for SELIN to turn off, and proceeds to take one of the following actions as a result of being selected:

1. If ATN is not asserted by the Initiator during selection, the chip generates a Selected interrupt indicating that the chip is connected as a Target.

2. If ATN is asserted, the chip simultaneously generates Selected, and Bus Service interrupts, indicating that the chip is connected as a Target and ATN is asserted.

6.2 RESELECTION

If the Reselect Enable bit in the Control Register is on, the chip may be reselected by a SCSI Target device. Reselection occurs only if SELOUT = 0, SELIN = 1, BSYIN = 0, I/O = 1, the chip's ID bit and the Target's ID bit are asserted on the data bus, no other ID bits are asserted, and data bus parity is good.

When all of these conditions exist, the chip is reselected. It then encodes the Target's ID and loads it into the Source ID Register. The chip also sets the ID Valid bit in the Source ID Register.

The chip then asserts BSYOUT and waits for SELIN to be released by the Target. When the chip detects SELIN = 0, it de-asserts BSYOUT and then generates a Reselected interrupt.

Reselection is now complete and the chip is in the connected Initiator state.

SECTION 7
INITIALIZATION

The SCSI device may be initialized by asserting RST for a period of at least 100ns, or by issuing a Chip Reset command to the device. The NCR 5385 will respond to the RST pulse or the Chip Reset command, by immediately disconnecting from the SCSI bus, initializing all storage elements and executing an internal self-diagnostic program. The self-diagnostic is explained in a previous section (See Section 4.10, Diagnostic Status Register). The following table lists the status of all registers after the initialization procedure.

	7	6	5	4	3	2	1	0
Data Register 1	x	x	x	x	x	x	x	x
Command Register	0	0	0	0	0	0	0	0
Control Register	0	0	0	0	0	0	0	0
Destination ID Register	0	0	0	0	0	0	0	0
Auxiliary Status Register	0	0	x	x	x	0	1	0
ID Register	0	0	0	0	0	x	x	x
Interrupt Register	0	0	0	0	0	0	0	0
Source Register	0	0	0	0	0	1	1	1
Data Register 2	x	x	x	x	x	x	x	x
Diagnostic Status Register	1	x	x	x	x	x	x	x
Transfer Counter (MSB)	0	0	0	0	0	0	0	0
Transfer Counter (2nd)	0	0	0	0	0	0	0	0
Transfer Counter (LSB)	0	0	0	0	0	0	0	0

x = Unknown

Table 7.1
Register Initialization

The controlling processor should loop on reading the Diagnostic Status Register until the Self-Diagnostic Complete bit (bit 7) is on (1). This should take approximately 350 clock cycles after reset occurs. If Programmed IO Mode is used, the processor should then check the remaining bits in this register for all zeros (no errors), and then load the ID Register. Following this, the Control Register should be programmed to enable the proper bits for SCSI operation. The SCSI Protocol Controller is now connected to the SCSI bus in a disconnected state. It is ready to receive commands from the controlling processor or respond to (re) selection attempts.

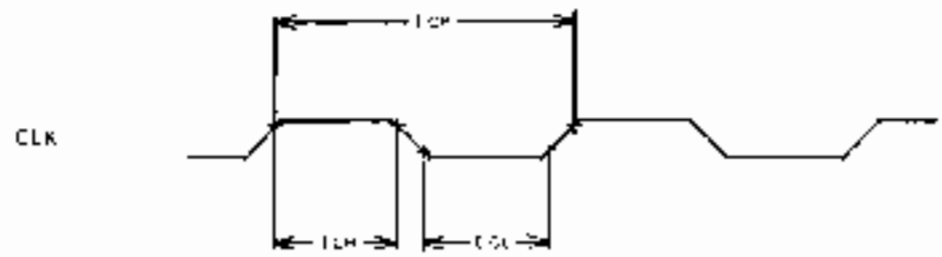
SECTION 8
EXTERNAL CHIP TIMING

Timing requirements must be over the operating temperature (0-70°C) and voltage (4.75 to 5.25V) ranges. Loading for all output signals, except $\overline{SBEN}$, is assumed to be four low-power Schottky inputs, including 50 pF capacitance. Loading for $\overline{SBEN}$ is assumed to be ten low-power Schottky inputs, including 100 pF capacitance.

8.1 MICROPROCESSOR INTERFACE — PRELIMINARY

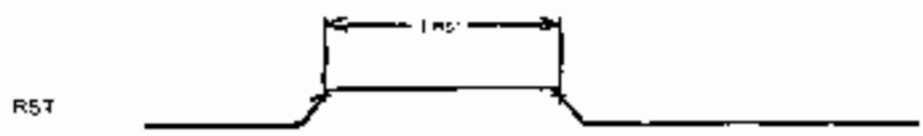
8.1.1 CLK

NAME	DESCRIPTION	MIN	MAX	UNITS
t_{CP}	Clock Period	100	200	ns
t_{CH}	Clock High	.45 t_{CP}	.55 t_{CP}	
t_{CL}	Clock Low	.45 t_{CP}	.55 t_{CP}	



8.1.2 RESET

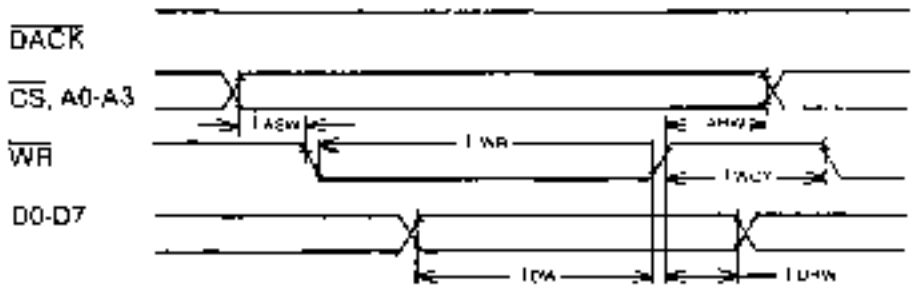
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{RST}	Reset Pulse Width	100			ns



8.1.3 MPU WRITE

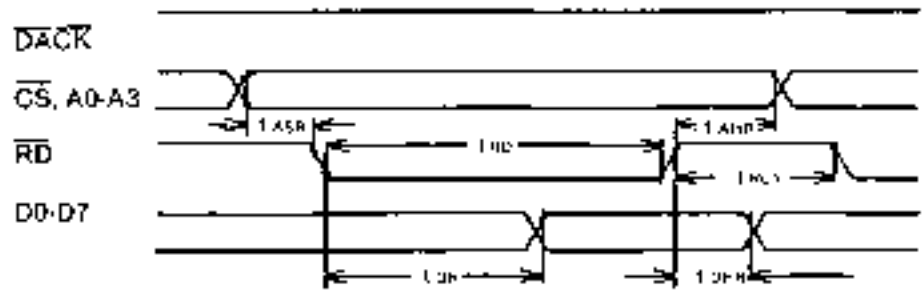
PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tASW	Address Set-up Time	0			ns
tWR	WR Pulse Width	95			ns
tDW	Data-to WR High	50			ns
tAHW	Address Hold Time	0			ns
tDHW	Data Hold Time	20			ns
tWCY	WR Off to WR or RD On	125			ns



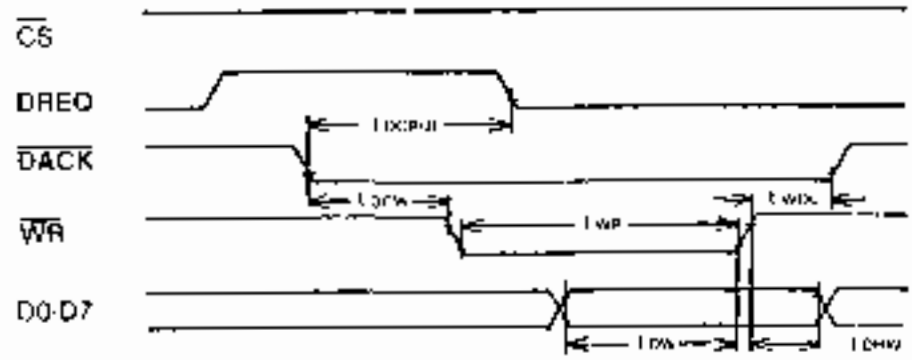
8.1.4 MPU READ

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tASR	Address Set-up Time to RD	0			ns
tRD	RD Pulse Width	125			ns
tDR	RD to Data			90	ns
tAHR	Address Hold Time	0			ns
tDHR	Data Hold Time	10		65	ns
tRCY	RD Off to WR or RD On	125			ns



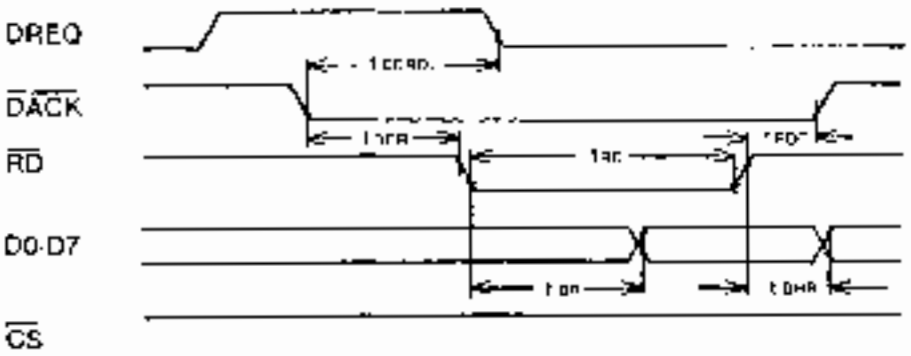
8.1.5 DMA WRITE

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tDCROL	$\overline{DACK}$ to DREQ Low	0		40	ns
tDCW	$\overline{DACK}$ to $\overline{WR}$	0			ns
tWR	$\overline{WR}$ Pulse Width	70			ns
tWDC	$\overline{WR}$ High to $\overline{DACK}$ High	0			ns
tDHW	Data Hold Time	20			ns
tDW	Data to $\overline{WR}$ High	50			ns



8.1.6 DMA READ

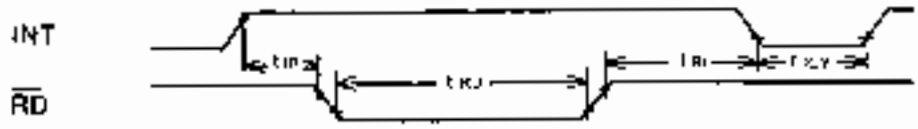
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tDCROL	$\overline{DACK}$ to DREQ Low	0		40	ns
tDCR	$\overline{DACK}$ to $\overline{RD}$	0			ns
tRD	$\overline{RD}$ Pulse Width	95			ns
tRDC	$\overline{RD}$ High to $\overline{DACK}$ High	0			ns
tDHR	Data Hold Time	10		65	ns
tDR	$\overline{RD}$ to Data			75	ns



8.1.7 INTERRUPT

PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{IR}	INT to $\overline{RD}$	0			ns
t_{RD}	$\overline{RD}$ Pulse Width	95			ns
t_{RI}	$\overline{RD}$ High to INT Low			125	ns
t_{ICY}	INT Off to INT On	125			ns



8.2.1 SELECTION (INITIATOR)

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{BF}	Bus Free	385			ns
$t_{BIA(5)}$	BSYIN low to ARB high	1.2		2.6	us
t_{SLA}	SELOUT high to ARB low & ID bit Disabled	3.2			us
$t_{BIBO\ (5)}$	BSYIN low to BSYOUT high	1.2		2.8	us
t_{BCD}	Bus Clear Delay			225	ns
t_{AD}	Arbitration Delay	3.0			us
t_{PC}	Priority check to SELOUT	0			ns
$t_{BID\ (5)}$	BSYIN low to ID bit high	1.2		2.9	us
t_{ADV}	Arbitration Data Valid to Priority Check	0			ns
t_{SI}	SELOUT to IGS	2.0			us
t_{IDBL}	Target ID high to BSYOUT low	1.1			us
t_{BOBI}	BSYOUT low to BSYIN low	0		400	ns
t_{BSL}	BSYIN high to SELOUT low	800			ns
t_{DID}	SBEN active to Bus enabled	150			ns

NOTES:

1. The chip ensures that the bus remains free (BSYIN and SELIN inactive) for t_{BF} before attempting arbitration.

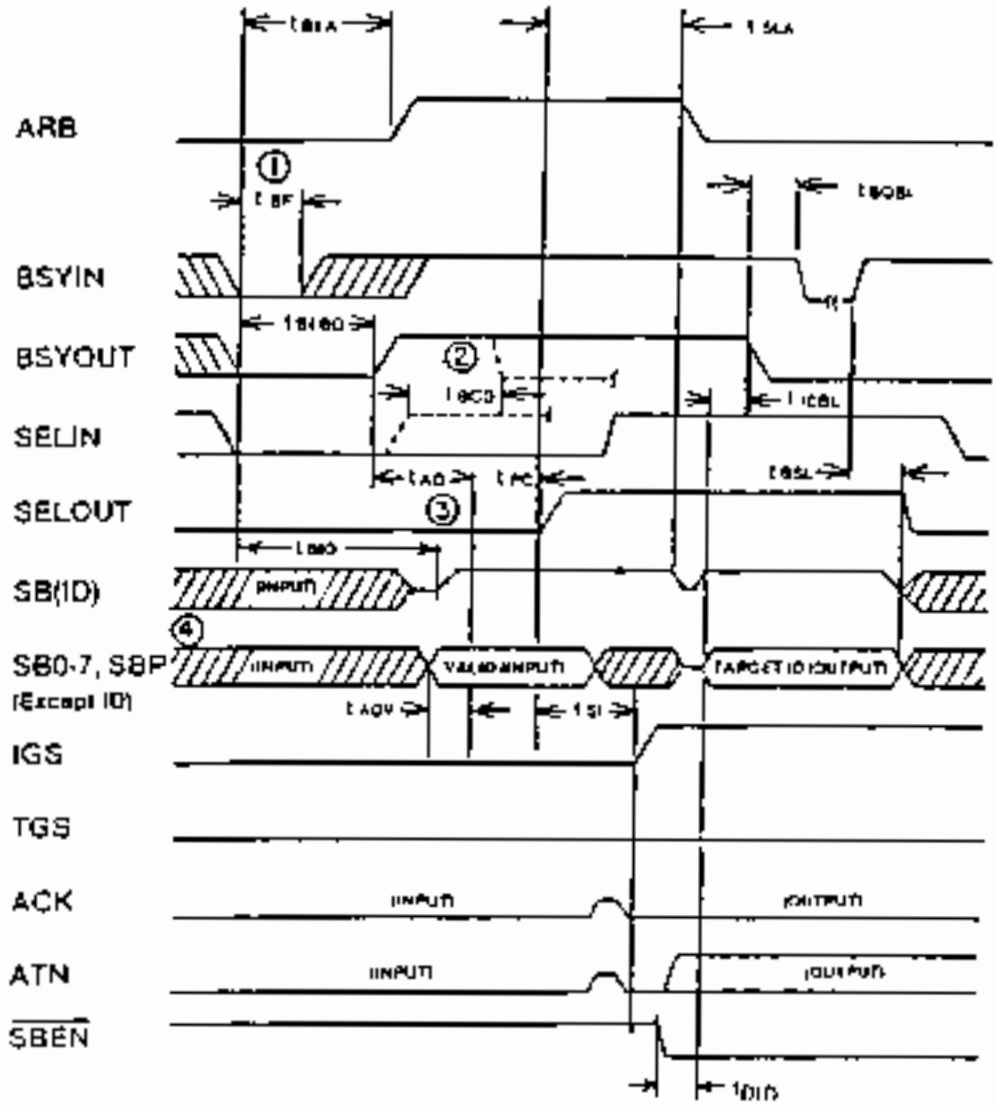
2. If SELIN becomes active at any time during arbitration, the chip must deassert BSYOUT within t_{BCD}.

3. The chip waits (t_{AD}), and then checks to see if arbitration is won (t_{PC}). The chip then asserts SELOUT if arbitration is won.

4. One of the data bits is assigned as an ID bit by the ID0-ID2 signals. During Bus Free, the chip places all of the data bits, including ID, in a high impedance state. During arbitration the chip enables its ID bit and drives it high, but the remainder of the data bits remain in the high impedance state for reading.

5. To verify these timings in a test environment, the user must allow a minimum of 45 clock cycles after the select command has been issued before the device begins to check for BSYIN low.

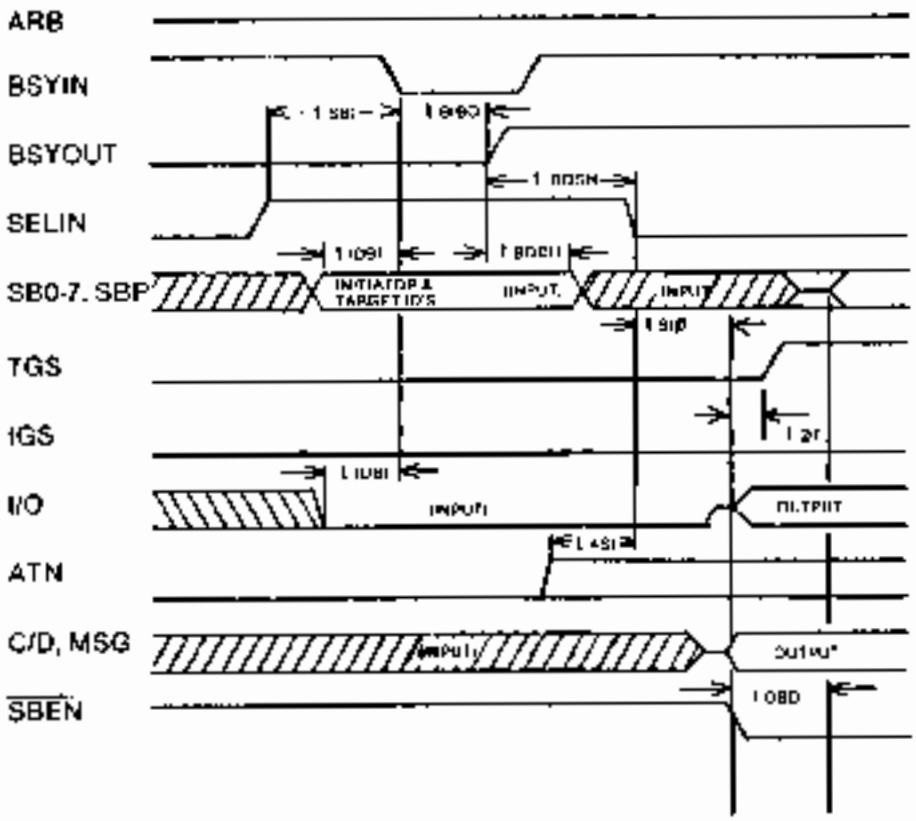
8.2.1 SELECTION (INITIATOR)



8.2.2 SELECTION (TARGET)

PRELIMINARY

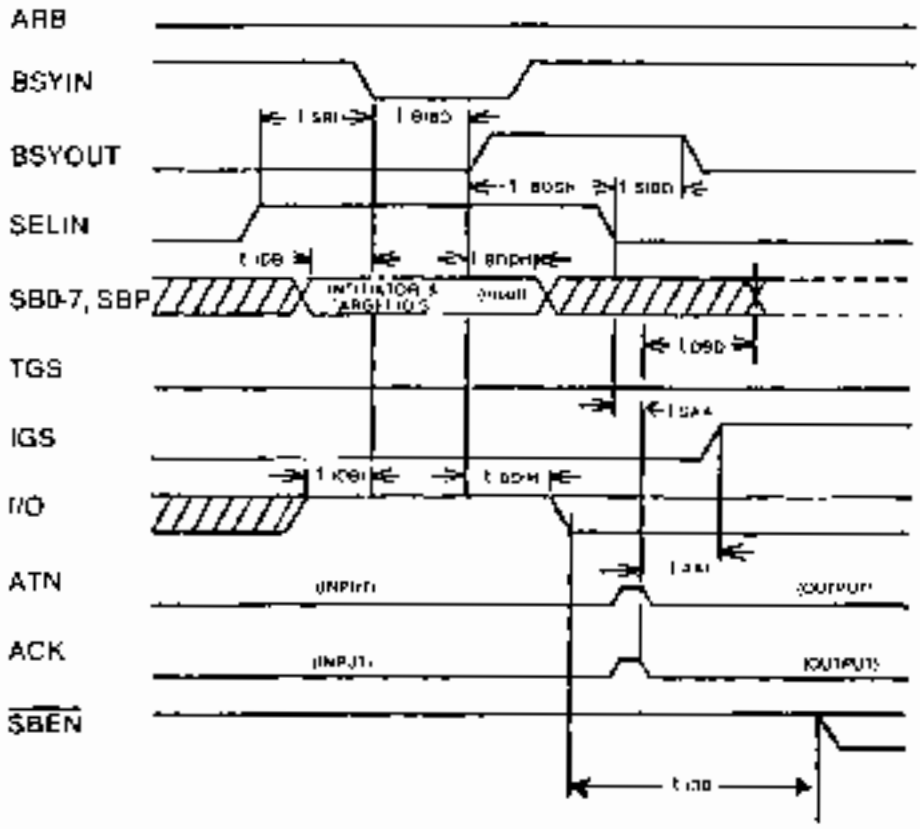
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
T_{SBI}	SELIN high to BSYIN low	50			ns
T_{IDBI}	ID's valid to BSYIN low	0			ns
T_{IOBI}	I/O low to BSYIN low	0			ns
T_{BIBO}	BSYIN low to BSYOUT high	0		2.0	us
T_{BODH}	BSYOUT high Data Hold	0			ns
T_{BOSH}	BSYOUT high SELIN Hold	0			ns
T_{ASI}	ATN high to SELIN low	0			ns
$T_{SI\emptyset}$	SELIN low to Phase signals Enabled	150			ns
T_{OT}	Phase signals enabled to TGS High	150			ns
T_{DBD}	SBEN low to Data Bus Enabled	150			ns



8.2.3 RESELECTION (INITIATOR)

PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tSBI	SELIN high to BSYIN low	50			ns
tIDBI	IO s valid to BSYIN low	0			ns
tIOBI	I/O high to BSYIN low	0			ns
tBIBO	BSYIN low to BSYOUT high	0		2.0	µs
tBODH	BSYOUT high Data Hold	0			ns
tBOSH	BSYOUT high SELIN Hold	0			ns
tBOIH	BSYOUT high I/O hold	0			ns
tSIBO	SELIN low to BSYOUT low	0			ns
tSAA	SELIN low to ACK & ATN enabled	750			ns
tAAI	ACK & ATN enabled to IGS high	150			ns
tIDB	I/O low to $\overline{SBEN}$ low	0			ns
tDBD	$\overline{SBEN}$ low to Data Bus Enabled	150			ns



8.2.4 RESELECTION (TARGET)

PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
^tBF	Bus Free	385			ns
^tBIA (5)	BSYIN low to ARB high	1.2		2.6	us
^tSLA	SELOUT high to ARB low & ID bit Disabled	3.2			us
^tBIBO (5)	BSYIN low to BSYOUT high	1.2		2.8	us
^tBCD	Bus Clear Delay			225	ns
^tAD	Arbitration Delay	3.0			us
^tPC	Priority check to SELOUT	0			ns
^tBID (5)	BSYIN low to ID bit high	1.2		2.9	us
^tADV	Arbitration Data Valid to Priority Check	0			ns
$^tS0$	SELOUT Phase signals Enabled & SBEN Low	2.4			us
^t0T	Phase Signals Enabled to TGS High	150			ns
^tOID	$\overline{SBEN}$ low to Bus Enabled	150			ns
^tIDBL	INITIATOR ID high to BSYOUT low	2.7			us
^tBOBI	BSYOUT low to BSYIN	0		400	ns
^tBIBO	BSYIN high to BSYOUT high	0.7		2.0	us
^tBSL	BSYOUT high to SELOUT low	450			ns

NOTES:

1. The chip ensures that the bus remains free (BSYIN and SELIN inactive) for ^tBF before attempting arbitration.

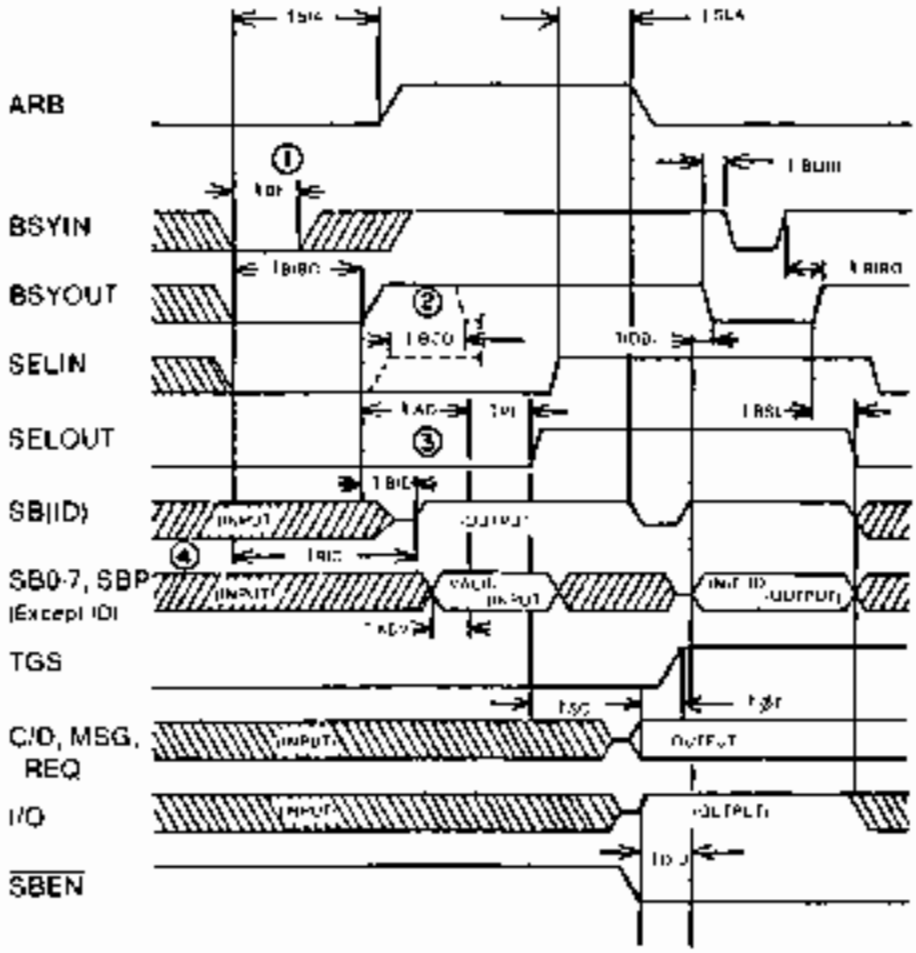
2. If SELIN becomes active at any time during arbitration, the chip must deassert BSYOUT within ^tBCD

3. The chip waits (^tAD), and then checks to see if arbitration is won (^tPC). The chip then asserts SELOUT if arbitration is won

4. One of the data bits is assigned as an ID bit by the IDO-ID2 signals. During Bus Free, the chip places all of the data bits, including ID, in a high impedance state. During arbitration the chip enables its ID bit and drives it high, but the remainder of the data bits remain in the high impedance state for reading.

5. To verify these timings in a test environment, the user must allow a minimum of 45 clock cycles after the select command has been issued before the device begins to check for BSYIN low.

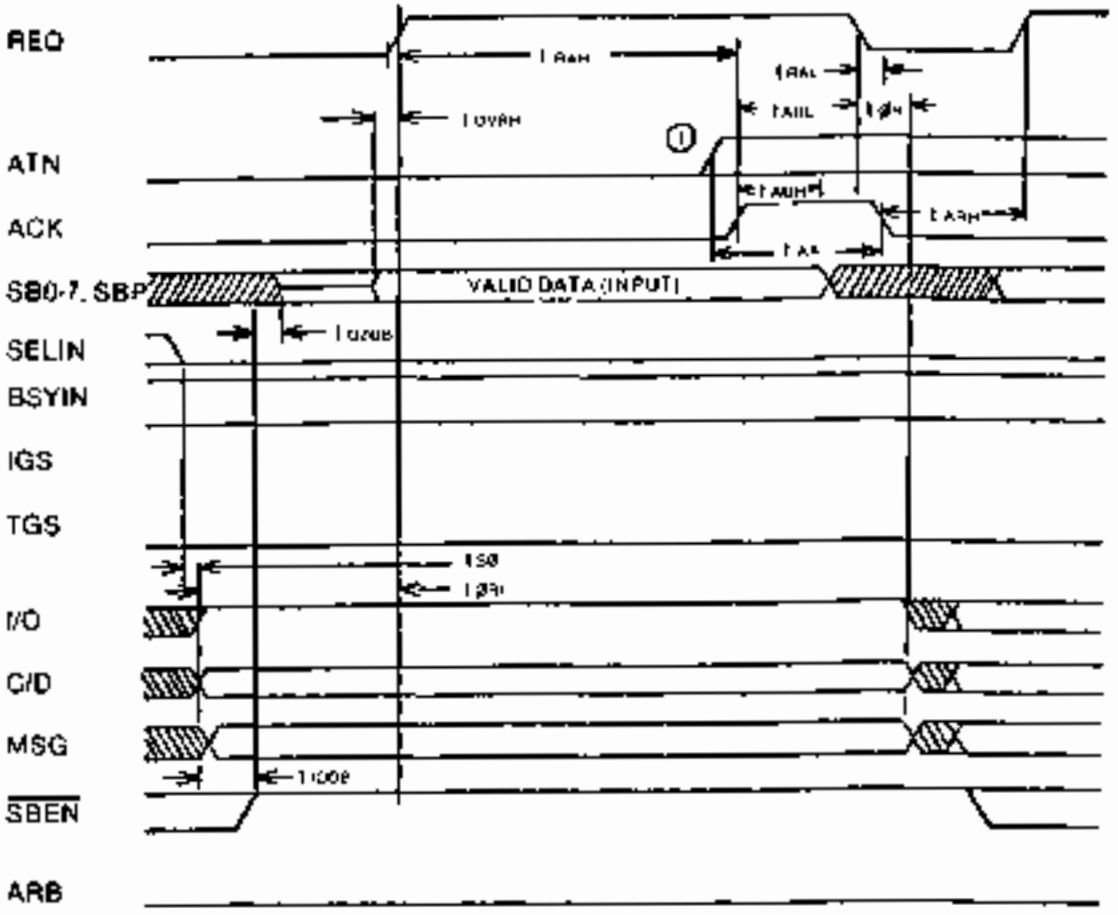
8.2.4 RESELECTION (TARGET)



8.2.5 INFORMATION TRANSFER PHASE INPUT (INITIATOR) PRELIMINARY

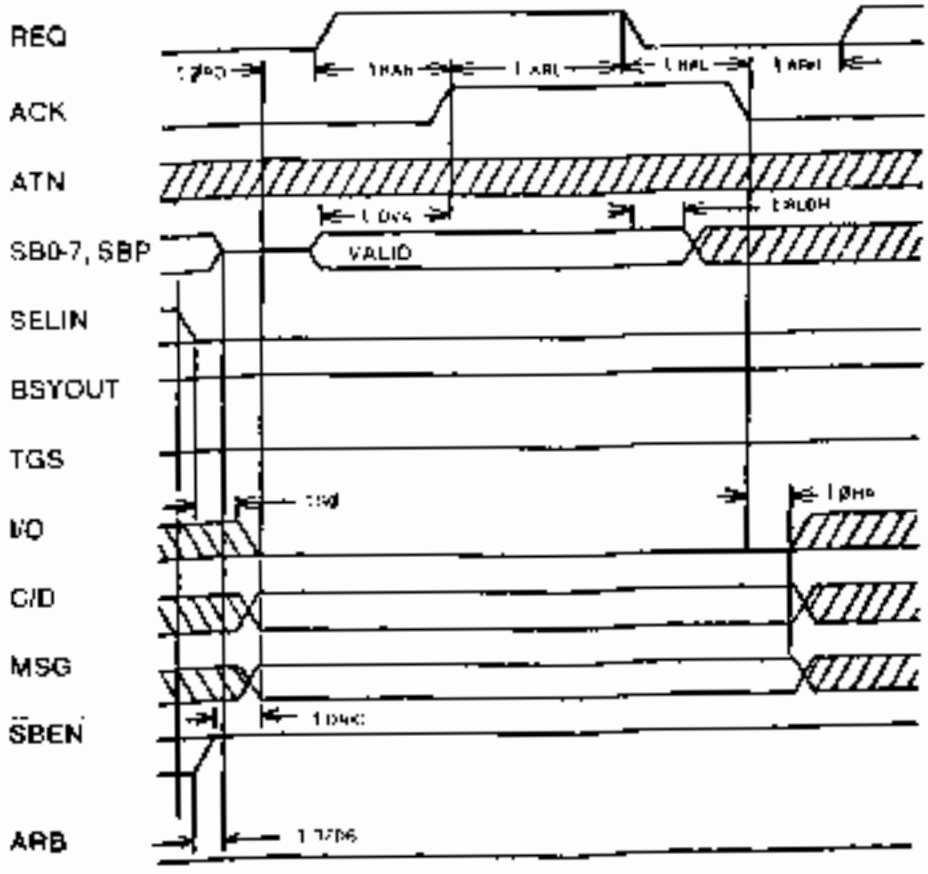
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{DVRH}	Data Valid to REQ high	0			ns
$t_{\phi RH}$	Phase Valid to REQ high	100			ns
t_{RAH}	REQ high to ACK high	0			ns
t_{RAL}	REQ low to ACK low	0			ns
t_{AA}	ATN high to ACK low	100			ns
$t_{S\phi}$	SELIN low to Phase change	0			ns
$t_{\phi H}$	Phase hold from ACK low	20			ns
t_{ADH}	Data hold from ACK high	0			ns
t_{ARL}	ACK high to REQ low	35			ns
t_{IODB}	I/O high to $\overline{SBEN}$ high			50	ns
t_{DZDB}	Data Bus disable from $\overline{SBEN}$ high			10	ns
t_{ARH}	ACK Low to REQ High	35			ns

NOTE 1: If the chip detects a parity error it must assert ATN at least t_{AA} before it de-asserts ACK.



8.2.6 INFORMATION TRANSFER PHASE INPUT (TARGET) PRELIMINARY

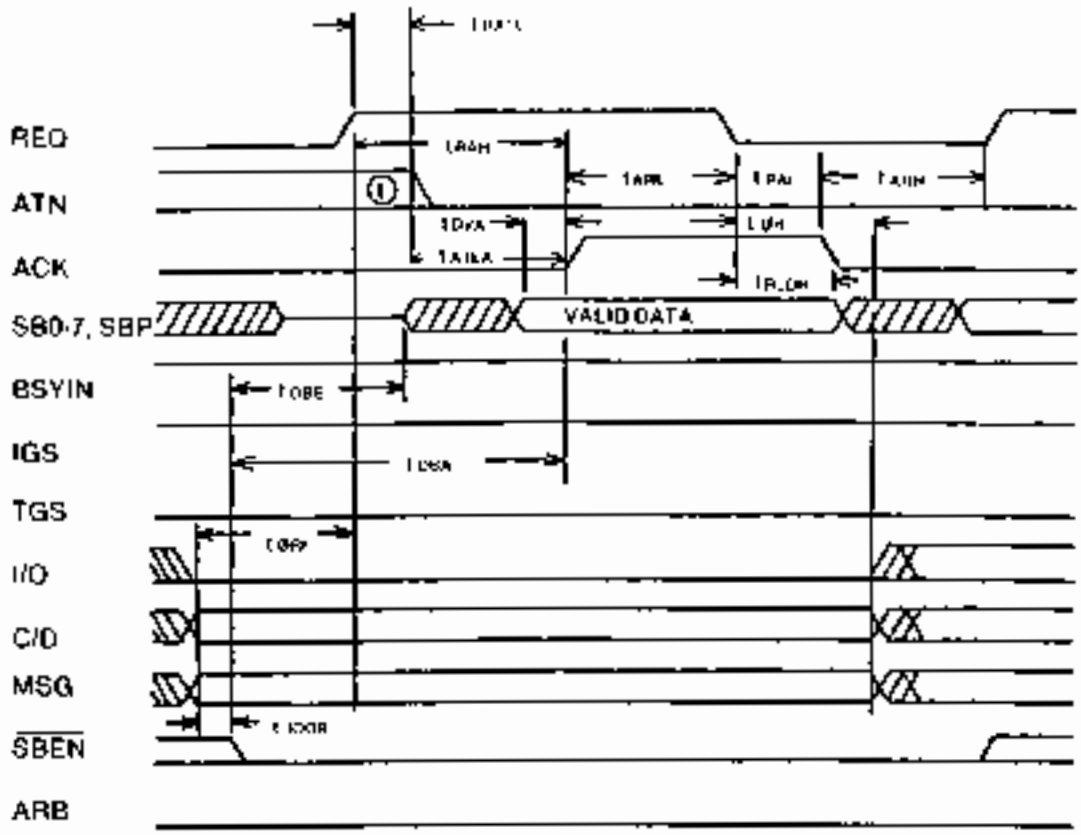
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
$t_{SØ}$	SELIN low to Phase Change	0			ns
$t_{ØRQ}$	Phase Change to REQ out	500			ns
t_{RAH}	REQ high to ACK high	35			ns
t_{ARL}	ACK high to REQ low	0			ns
t_{DVA}	Data Valid to ACK high	0			ns
t_{RAL}	REQ low to ACK low	35			ns
t_{ARH}	ACK low to REQ high	0			ns
t_{RLDH}	REQ low Data Hold	0			ns
$t_{ØHA}$	Phase Hold from ACK low	0			ns
t_{DBIO}	SBEN high to I/O low	0			ns
t_{DZDB}	Data Bus disable to SBEN high	0			ns



8.2.7 INFORMATION TRANSFER PHASE OUTPUT (INITIATOR) PRELIMINARY

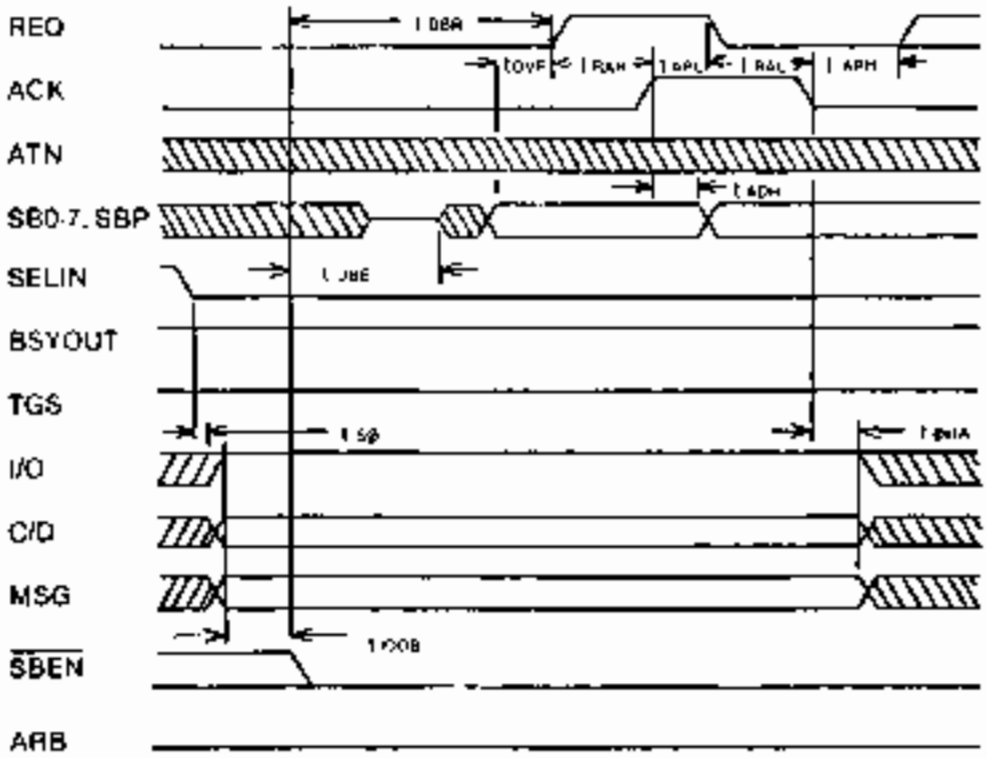
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tϕRI	Phase Valid to REQ high	100			ns
tRAH	REQ high to ACK high	35			ns
tRAL	REQ low to ACK low	0			ns
tDVA	Data Valid to ACK high	100			ns
tRLDH	REQ low Data hold	0			ns
tϕH	Phase hold from ACK low	20			ns
tARL	ACK high to REQ low	0			ns
tIODB	I/O low to $\overline{SBEN}$ low	0			ns
tDBE	$\overline{SBEN}$ low to Data Bus Enable	85			ns
tDBA	$\overline{SBEN}$ low to ACK high	185			ns
tRATL	REQ High to ATN low	0			ns
tATLA	ATN Low to ACK High	25			ns
tARH	ACK Low to REQ High	35			ns

NOTE 1: ATN is only de-asserted in this manner during the last byte of a Message Out Phase.



8.2.8 INFORMATION TRANSFER PHASE OUTPUT (TARGET) PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tSØ	SELIN low to Phase Change	0			ns
tIODB	I/O high to SBEN low	500			ns
tDBR	SBEN low to REQ out	185			ns
tDVA	Data Valid to REQ high	100			ns
tRAH	REQ high to ACK high	0			ns
tARL	ACK high to REQ low	0			ns
tRAL	REQ low to ACK low	0			ns
tARH	ACK low to REQ high	0			ns
tØHA	Phase hold from ACK low	0			ns
tADH	Data hold from ACK low	0			ns
tDBE	SBEN low to Data Bus Enabled	85			ns

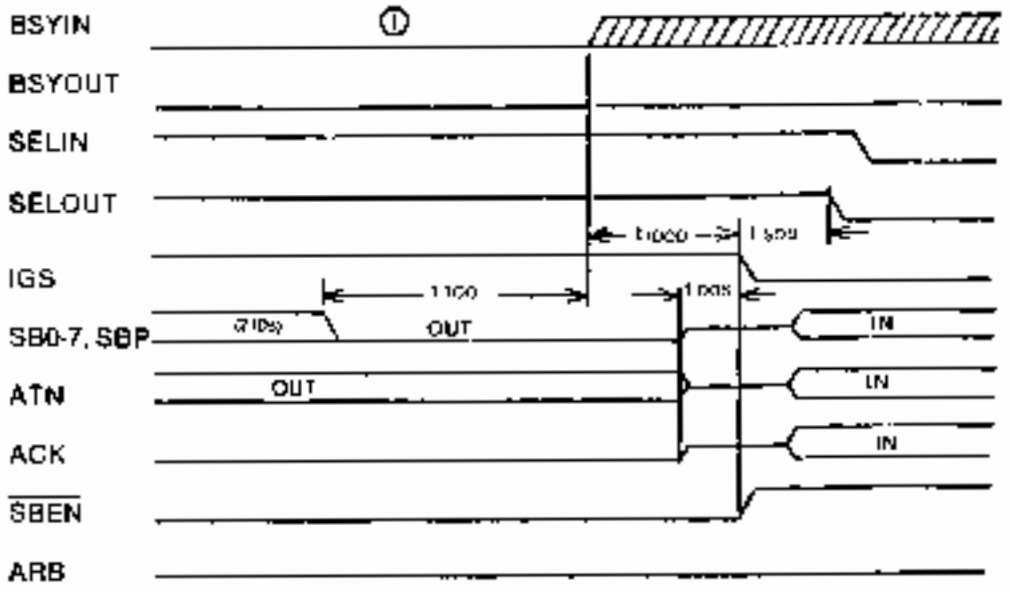


8.2.9 BUS RELEASE FROM SELECTION (INITIATOR)

PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{TOD}	Bus Release Timeout Delay	100			µs
t_{IDOD}	IGS & $\overline{SBEN}$ Turn-off Delay	0			ns
t_{SOD}	SELOUT Turn-off Delay	0			ns
t_{DOS}	Driver Turn-off set-up to IGS & $\overline{SBEN}$ off	0			ns

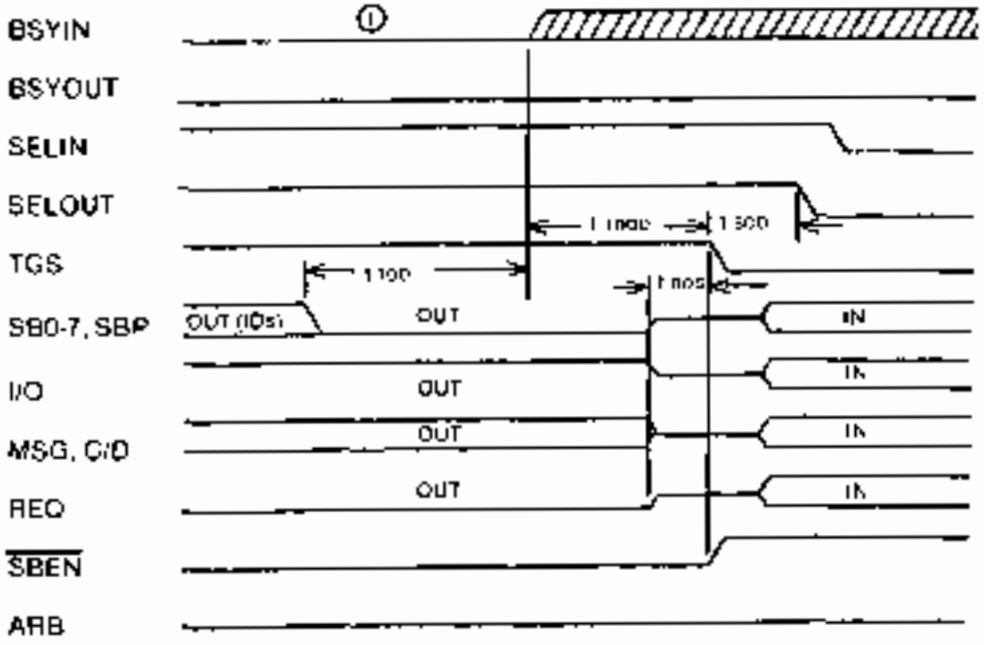
NOTE 1: If the chip detects BSYIN active by the end of the timeout delay, the bus release sequence shall be aborted since selection has been successful.



8.2.10 BUS RELEASE FROM RESELECTION (TARGET) PRELIMINARY

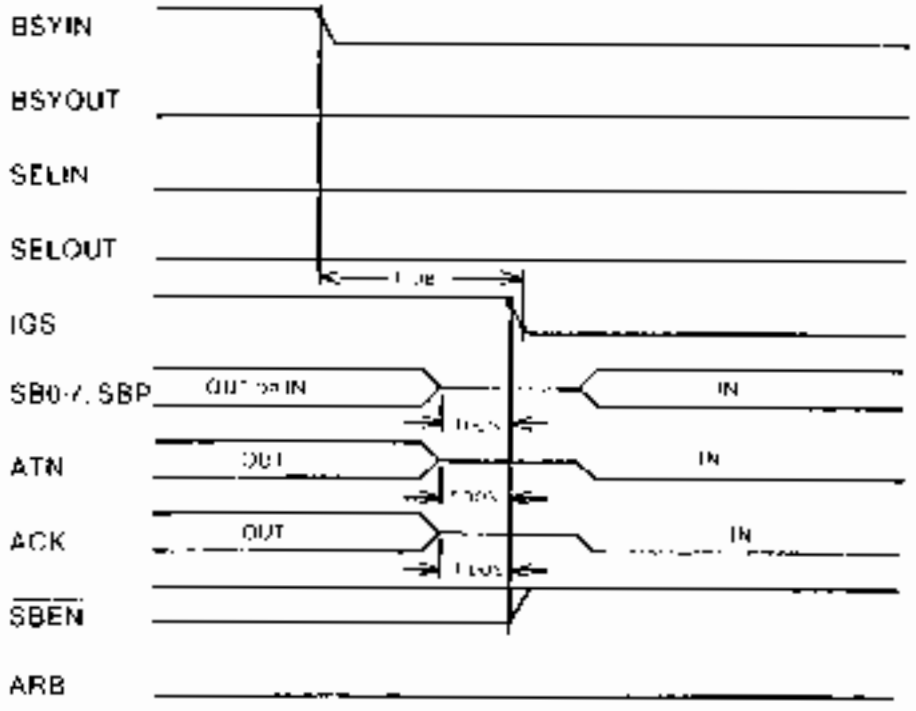
NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tTOD	Bus Release Timeout Delay	100			uS
tDOD	TGS & $\overline{SBEN}$ Turn-off Delay	0			ns
tSOD	SELOUT Turn-off Delay	0			ns
tDOS	Driver Turn-off set-up to TGS & $\overline{SBEN}$ off	0			ns

NOTE 1: If the chip detects BSYIN active by the end of the timeout delay, the bus release sequence shall be aborted since selection has been successful



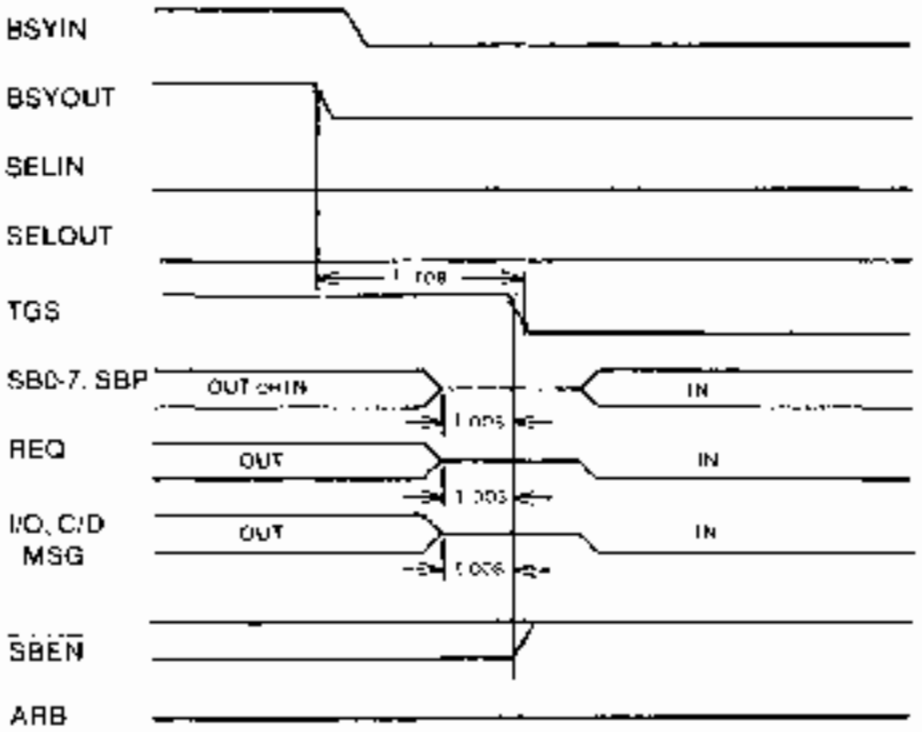
8.2.11 BUS RELEASE FROM INFORMATION PHASE (INITIATOR) PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
1IDB	IGS & SBEN Turn-off Delay from BSYIN off			225	ns
1DOS	Driver Turn-off set-up to IGS off	0			ns



8.2.12 BUS RELEASE FROM INFORMATION PHASE (TARGET) PRELIMINARY

NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
tTDB	TGS & $\overline{SBEN}$ Turn off from BSYOUT off			225	ns
tDOS	Driver Turn-off set-up to TGS off	0			ns





NCR Microelectronics Division
1635 Aeroplaza Drive
Logic Products Marketing
Colorado Springs, CO 80916
(303) 596-5812 or (800) 525-2252
TELEX 45-2457

MC-704 1285

NCR 8310 GENERAL PURPOSE DRIVER

RECEIVER CHIP

PRELIMINARY DATA SHEET

NCR

MICROELECTRONICS DIVISION, COLORADO SPRINGS

08/07/85

PRELIMINARY

Copyright 1985, by NCR Corporation
Dayton, Ohio
All Rights Reserved Printed in U.S.A.

This document contains the latest information available at the time of publication. However, NCR reserves the right to modify the contents of this material at any time. Also, all features, functions and operations described herein may not be marketed by NCR in all parts of the world. Therefore, before using this document, consult your NCR representative or NCR office for the information that is applicable and current.

TABLE OF CONTENTS

SECTION
1. GENERAL DESCRIPTION
2. PIN DESCRIPTION
   2.1 SCSI Interface Signals
   2.2 Transceiver Interface Signals
   2.3 Power Signals
3. ELECTRICAL CHARACTERS
   3.1 Operating Conditions
   3.2 Input Signal Requirements
   3.3 Output Signal Requirements
4. CHIP TIMING

SECTION 1
GENERAL DESCRIPTION

1.0 The NCR 8310 General Purpose Driver/Receiver Chip, a 48-pin NMOS device, is designed as a 48 mA bus transceiver chip for all of the Small Computer System Interface (SCSI) bus signals. It contains high-current single-ended drivers for the SCSI bus. The NCR 8310 is specifically intended to be used with the NCR 5385E and NCR 5386 SCSI Protocol Controller chips and interfaces directly to the above referenced chips. Additionally, it may be used with other interfaces where a general purpose 48 mA driver/receiver chip is required.

Figure 1.1 shows the pinout for the NCR 8310, with signals labeled in parenthesis referenced in Figure 1.2 interfacing directly to the NCR 5385E. Signals not enclosed by parenthesis are referenced to Figure 1.3 and refer to the most general application of the NCR 8310. The prefix "H" indicates "high current." The 48 mA data bus, HD0-HD7/ and microprocessor bus, D0-D7, are controlled by the Data Bus Enable (DBEN/), and port HB0-HB3 and B0-B3 are controlled by PBEN. Similarly, ports A, E, F and G are controlled likewise.

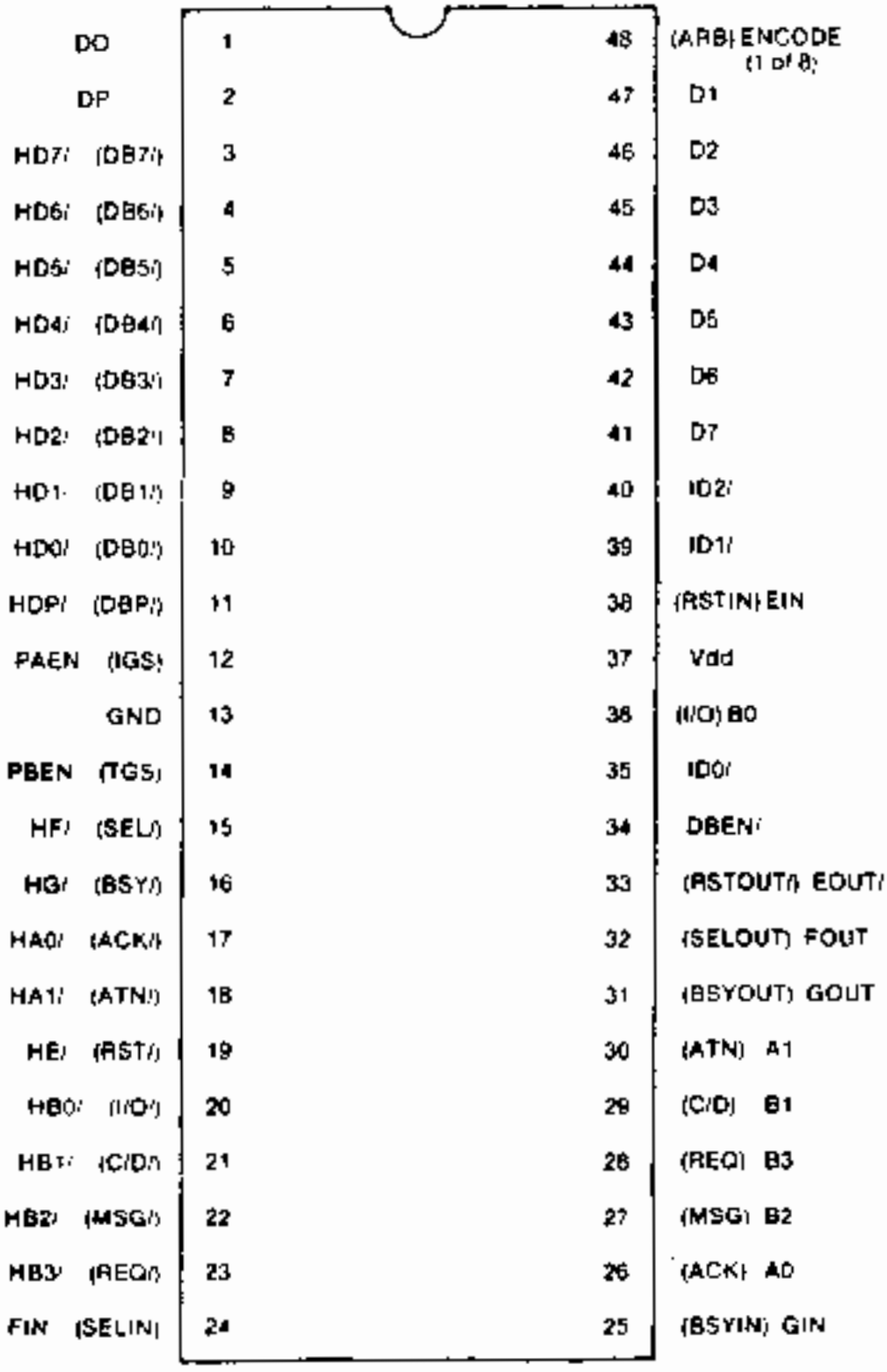


FIGURE 1.1 PINOUT

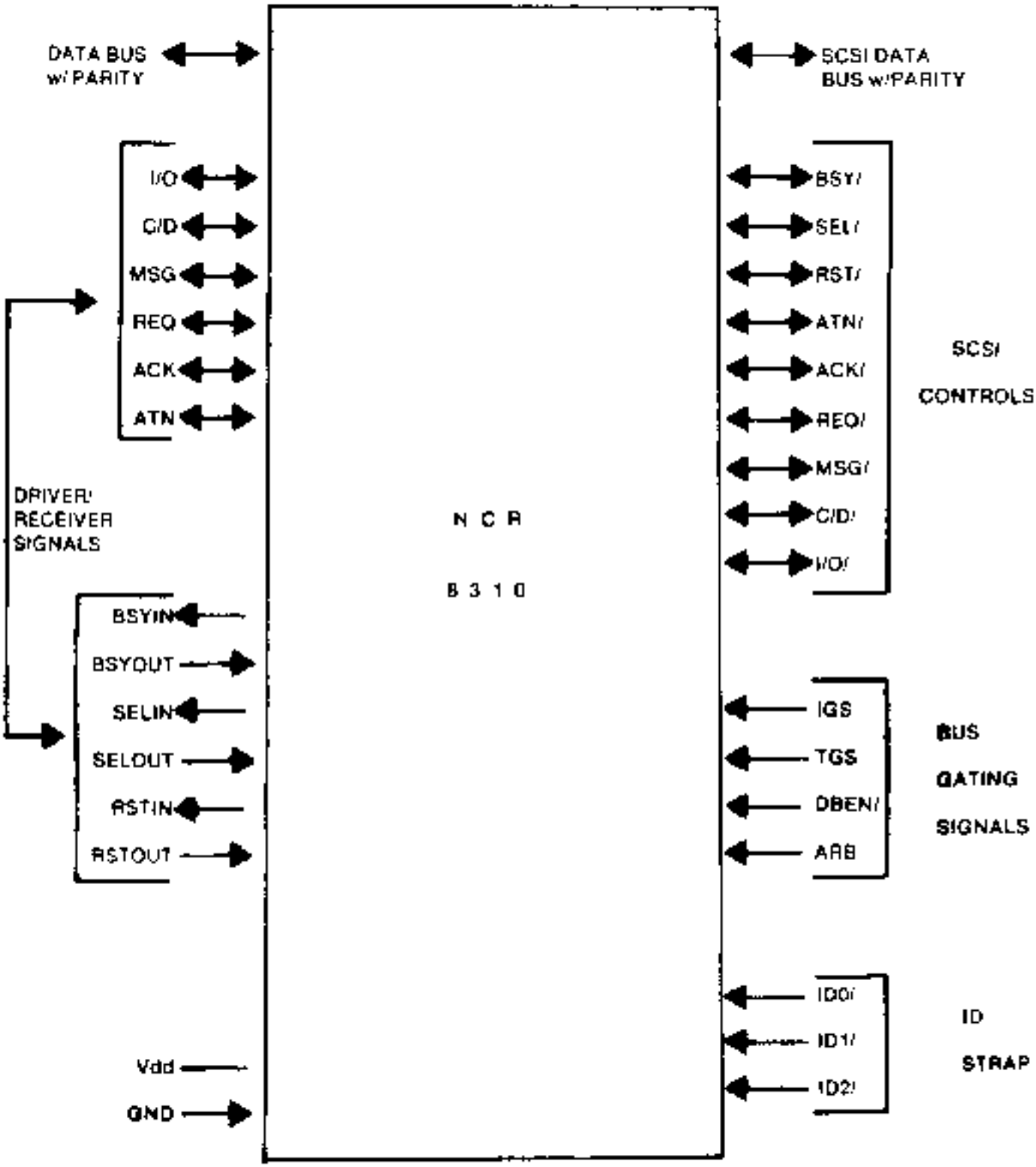


FIGURE 1.2
FUNCTIONAL PIN GROUPING

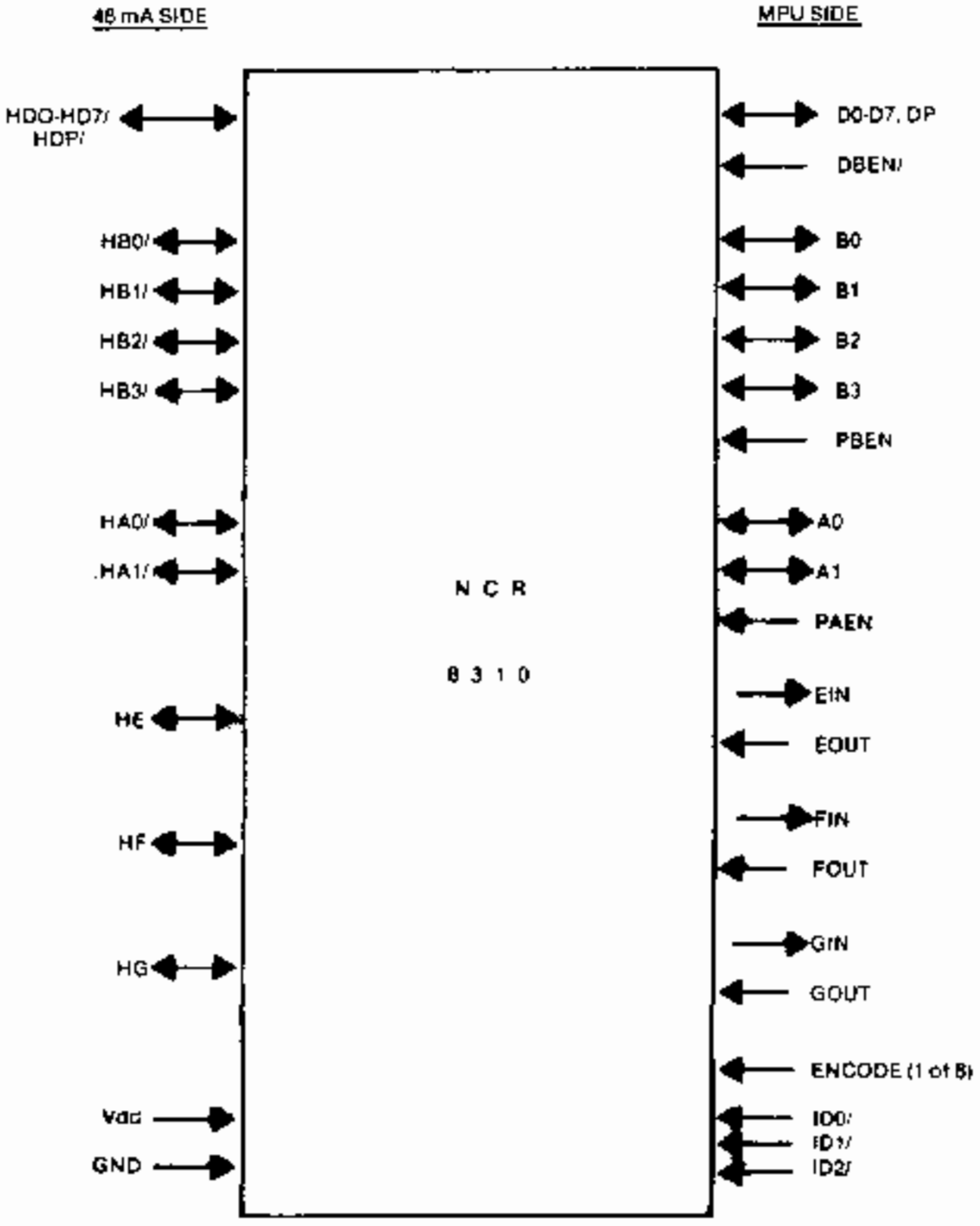


FIGURE 1.3
GENERAL PURPOSE FUNCTIONAL PIN GROUPING

Figure 1.4 shows the interface between the NCR 5385E SCSI Protocol Controller and the NCR 8310 Driver/Receiver Chip. With the implementation of this suggested interface, a two-chip solution is possible.

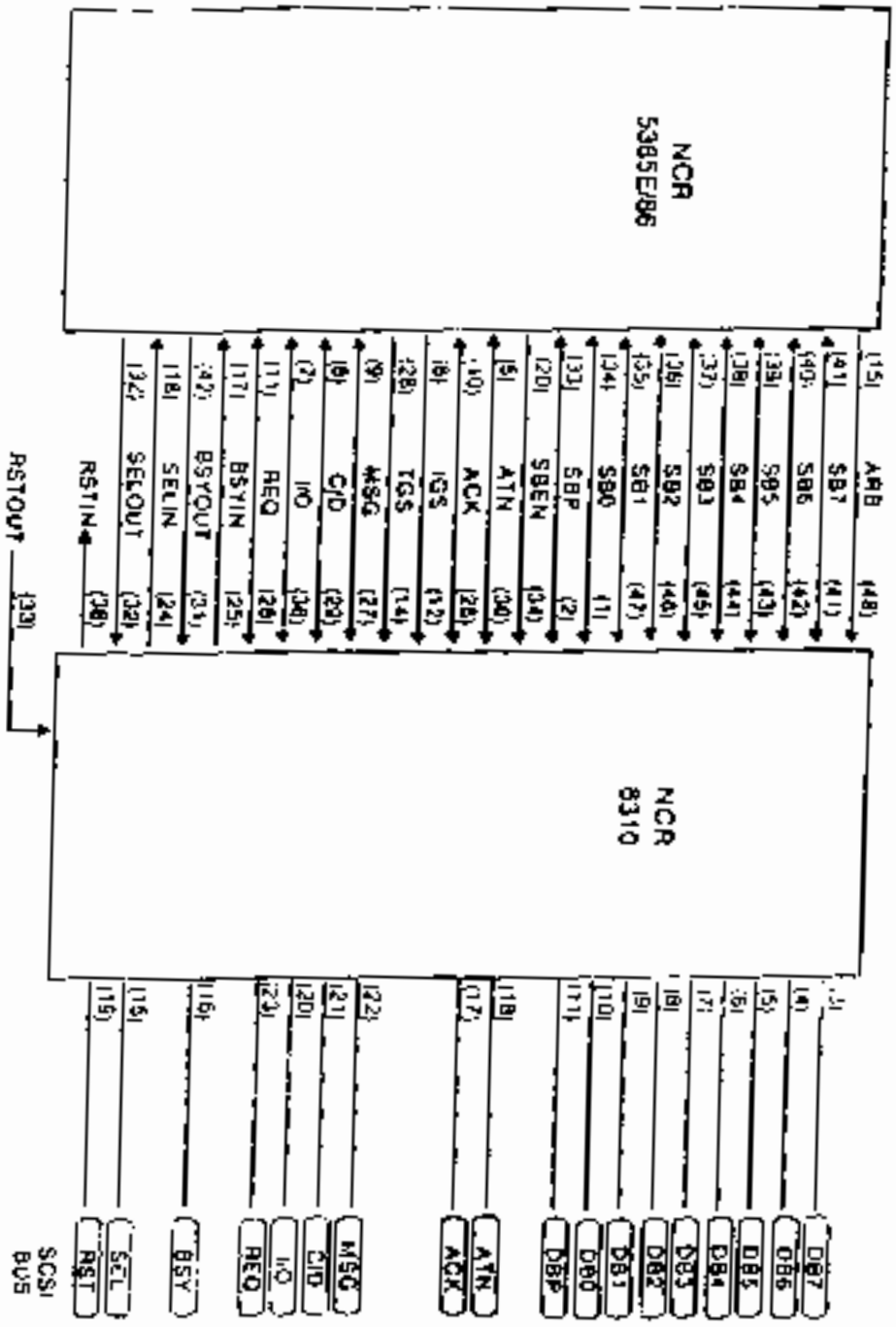


FIGURE 1.4
SINGLE-ENDED INTERFACE USING THE NCR 5385E/5386 SCSI PROTOCOL CONTROLLER AND THE NCR 8310 GENERAL PURPOSE DRIVER/RECEIVER CHIP

Figure 1.5 shows the Equivalent Circuit for the NCR 8310.

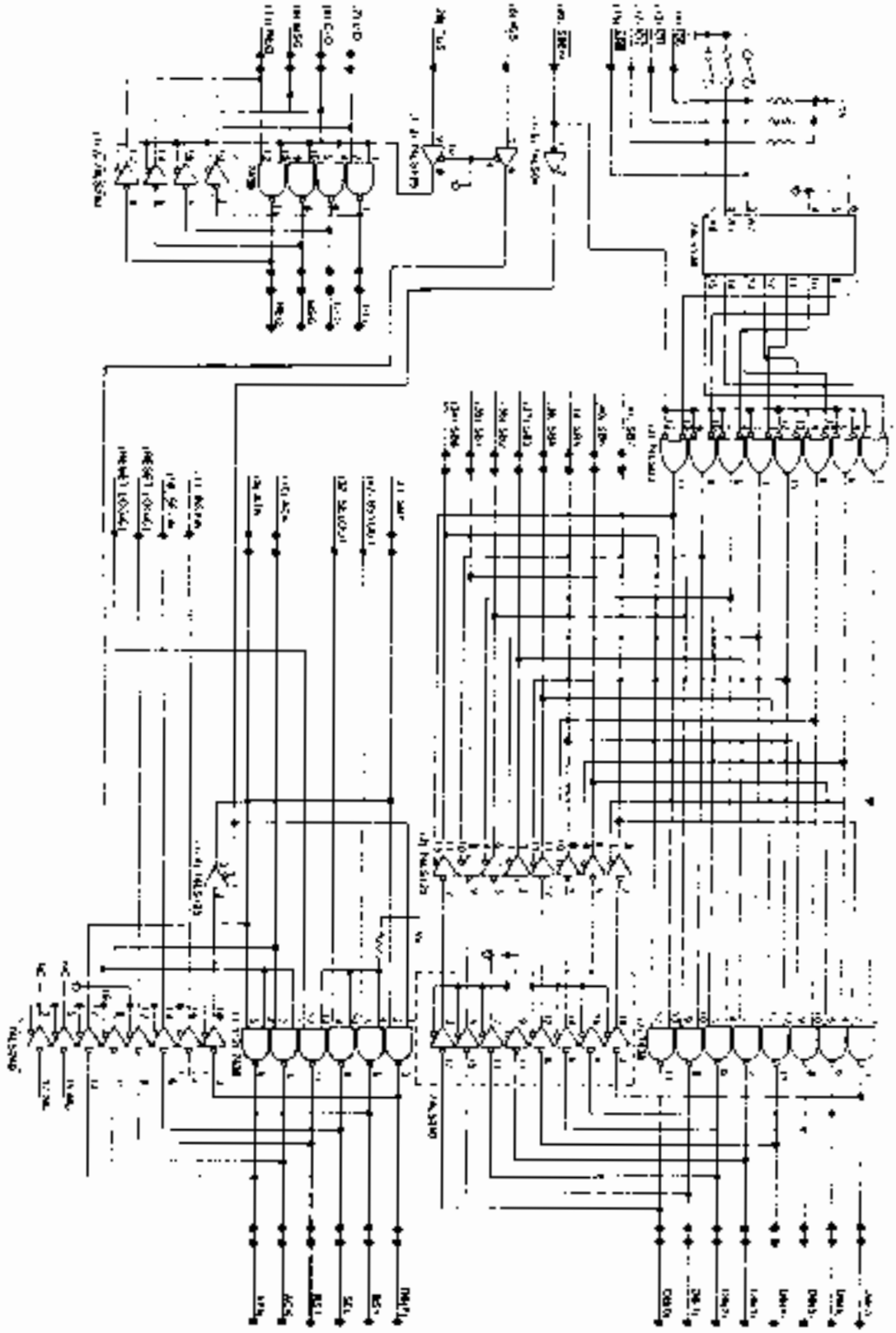


FIGURE 1.5
NCR 8310 GENERAL PURPOSE DRIVER/RECEIVER EQUIVALENT CIRCUIT

SECTION 2
PIN DESCRIPTION

2.1 SCSI INTERFACE SIGNALS

The following signals are all bi-directional, active low, open collector signals. With 48 mA sink capability, all pins interface directly with the SCSI bus.

SIGNAL	PIN	DESCRIPTION
HD0...HD7/ (DB0...DB7/)	10...3	These eight data bits (DB0-DB7/) plus a parity bit (DBP/) form the data bus. DB7/ is the most significant bit and has the highest priority during the Arbitration phase. Data parity is odd. Parity is not valid during Arbitration
HDP/ (DBP/)	11	
HB0/ (I/O/)	20	I/O/ is a signal driven by a target which controls the direction of data movement on the SCSI bus. True indicates input to the initiator. This signal is also used to distinguish between Selection and Reselection phases.
HB1/ (C/D/)	21	A signal driven by the target. C/D/ indicates Control of Data information is on the data bus. This signal is received by the initiator.
HB2/ (MSG/)	22	MSG/ is a signal driven by the target during the Message phase. This signal is received by the initiator.
HB3/ (REQ/)	23	Driven by a target, REQ/ indicates a request for a REQ/ACK data transfer handshake. This signal is received by the initiator
HA0/ (ACK/)	17	Driven by an initiator, ACK/ indicates an acknowledgement for a REQ/ACK data transfer handshake. In the target role, ACK/ is received as a response to the REQ/ signal.
HA1/ (ATN/)	18	Driven by an initiator, ATN/ indicates an attention condition. This signal is received in the target role.
HG/ (BSY/)	16	This signal indicates that the SCSI bus is being used and can be driven by both the initiator and the target device.
HF/ (SEL/)	15	SEL/ is used by an initiator to select a target or by a target to reselect an initiator.
HE/ (RST/)	19	The RST/ signal indicates an SCSI bus RESET condition.

2.2 TRANSCEIVER INTERFACE SIGNALS

SIGNAL	PIN	DESCRIPTION
D0-D7	1,47-41	(I/O) This bidirectional data bus is connected to the NCR 5385E signals SB0-SB7.
DP	2	(I/O) This bidirectional parity signal is connected to the NCR 5385E signal SBP
DBEN/	34	(INPUT) This signal enables the SCSI bus drivers for DB0-DB7/ and DBP/. It is connected to the NCR 5385E signal SBEN/.
PBEN (TGS)	14	(INPUT) This signal enables the SCSI bus drivers for I/O/, C/D/, MSG/ and REQ/. It is connected to the NCR 5385E signal TGS/.
PAEN (IGS)	12	(INPUT) This signal enables the SCSI bus drivers for ACK/ and ATN/. It is connected to the NCR 5385E signal IGS.
ENCODE (1 of 8) (ARB)	48	(INPUT) This signal enables decode of ID0-ID2/ and asserts the selected SCSI data bus ID. It is connected to the NCR 5385E signal ARB.
ID0-ID2/	35, 39-40	(INPUT) These ID signals are used during arbitration to select the correct DB(n)/ line. They are connected to the NCR 5385E signals ID0-ID2/
B0 (I/O)	36	(I/O) INPUT/OUTPUT - This pin is used to drive or receive the SCSI signal I/O/. It is connected to the NCR 5385E signal I/P.
B1 (C/D)	29	(I/O) CONTROL/DATA - This pin is used to drive or receive the SCSI signal C/D/. It is connected to the NCR 5385E signal C/D.
B2 (MSG)	27	(I/O) MESSAGE — This pin is used to drive or receive the SCSI signal MSG/. It is connected to the NCR 5385E signal MSG.
B3 (REQ)	28	(I/O) REQUEST - This pin is used to drive or receive the SCSI signal REQ/. It is connected to the NCR 5385E signal REQ.
A0 (ACK)	26	(I/O) ACKNOWLEDGE - This pin controls the SCSI signal ACK/. It is connected to the NCR 5385E signal ACK.
A1 (ATN)	30	(I/O) ATTENTION - This pin controls the SCSI signal ATN/. It is connected to the NCR 5385E signal ATN.
GIN (BSYIN)	25	(OUTPUT) This signal indicates the received state of the SCSI signal BSY/. It is connected to the NCR 5385E signal BSYIN.
GOUT (BSYOUT)	31	(INPUT) This signal drives the SCSI signal BSY/. It is connected to the NCR 5385E signal BSYOUT.
FIN (SELIN)	24	OUTPUT) This signal indicates the received state of the SCSI signal SEL/. It is connected to the NCR 5385E signal SELIN
FOUT (SELOUT)	32	(INPUT) This signal drives the SCSI signal SEL/. It is connected to the NCR 5385E signal SELOUT.
EIN (RSTIN)	38	(OUTPUT) This signal indicates the received state of the SCSI signal RST/. It is not connected to the NCR 5385E
EOUT/ (RSTOUT/)	33	(INPUT) This signal drives the SCSI signal RST/. It is not connected to the NCR 5385E

2.3 POWER SIGNALS

SIGNAL	PIN	DESCRIPTION
Vdd	37	5 Volts DC
GND	13	Ground

SECTION 3
ELECTRICAL CHARACTERISTICS

3.1 OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN	MAX	UNITS
Supply Voltage	Vdd	4.75	5.25	VDC
Supply Current	Idd		145	mA
Ambient Free Air Temperature	Ta	0	70	degrees C

3.2 INPUT SIGNAL REQUIREMENTS

PARAMETER	CONDITIONS	MIN	MAX	UNITS
High-level Input Voltage Vih		2.0	5.75	VDC
Low-level Input Voltage Vil		-0.3	0.8	VDC
SCSI: High-level Input Current Iih	Vih=5.25V		50	uA
Other Pins High-level Input Current Iih	Vih=5.25V		10	uA
DP Low-level Input Current Iil	Vil=0V		-2	mA
SCSI: Low-level Input Current Iil	Vil=0V		-50	uA
All Other Pins: Low-level Input Current Iil	Vil=0V		-10	uA

3.3 OUTPUT SIGNAL REQUIREMENTS

PARAMETER	CONDITIONS	MIN	MAX	UNITS
High-level Output Voltage Voh	Vdd = 4.75V Ioh = 7.0mA	2.4		VDC
Low-level Output Voltage Vol	Vdd = 4.75V Iol = 7.0mA		0.5	VDC
SCSI. Low-level Output Voltage Vol	Vdd = 4.75V Iol = 48.0mA		0.5	VDC

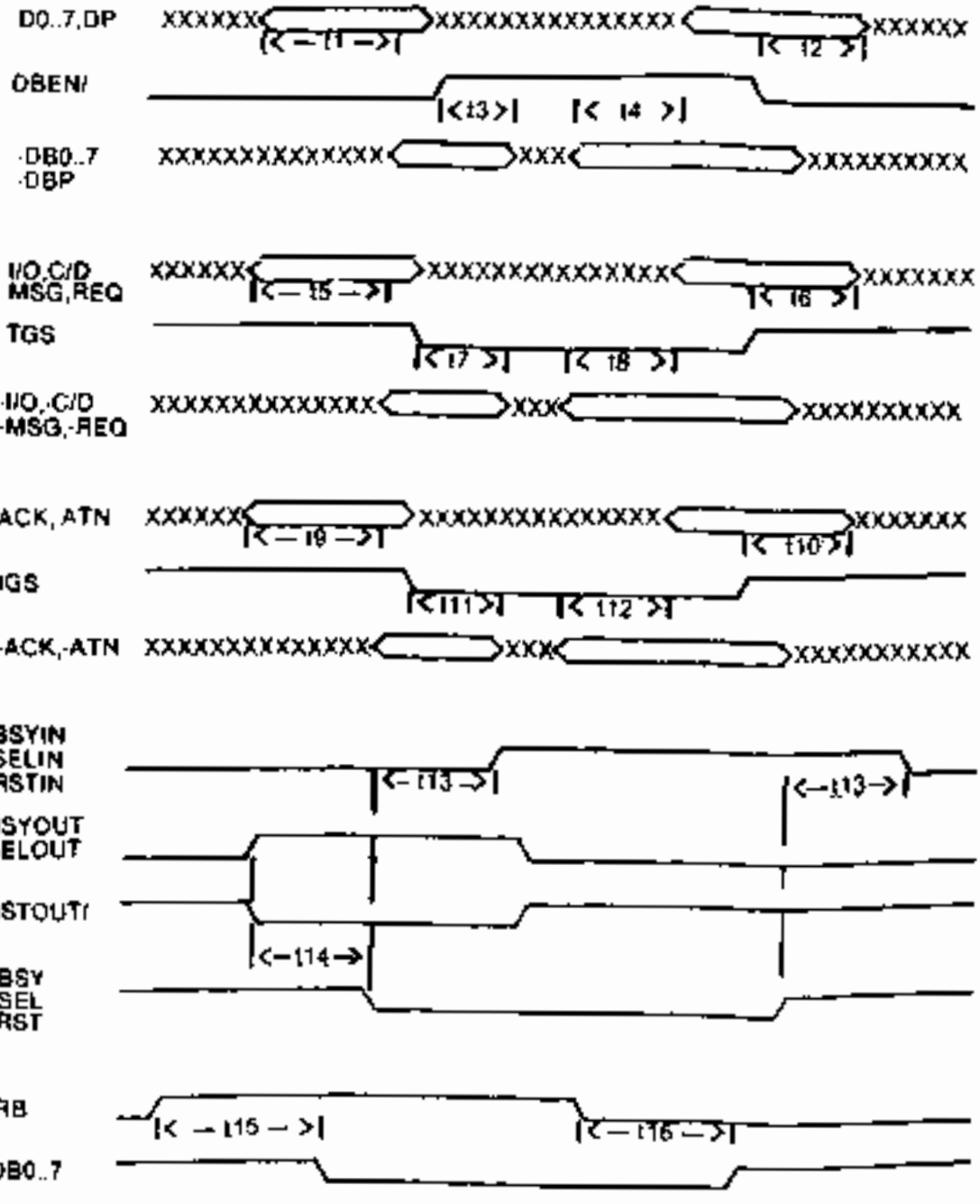
PRELIMINARY

Notice: This is not a final specification.
Some parametric limits are subject to change.

SECTION 4

CHIP TIMING

DRIVER/RECEIVER MODE



NAME	DESCRIPTION	MIN	TYP	MAX	UNITS
T1	Data bus to SCSI bus delay			90	nsec.
T2	DBEN true to Data bus tri-state			90	nsec.
T3	DBEN false to SCSI bus release			100	nsec
T4	SCSI bus to Data bus delay			60	nsec.
T5	Control signal delay to SCSI bus			70	nsec.
T6	TGS true to input release			50	nsec
T7	TGS false to SCSI bus release			70	nsec.
T8	SCSI bus receiver delay			50	nsec
T9	Control signal delay to SCSI bus			70	nsec.
T10	IGS true to input release			50	nsec.
T11	IGS false to SCSI bus release			70	nsec.
T12	SCSI bus receiver delay			50	nsec.
T13	Receiver delay BSY RST SEL			50	nsec.
T14	Driver delay BSY RST SEL			70	nsec.
T15	ARB true to DB(id)/ true			70	nsec.
T16	ARB false to DB(id)/ false			70	nsec.

PRELIMINARY

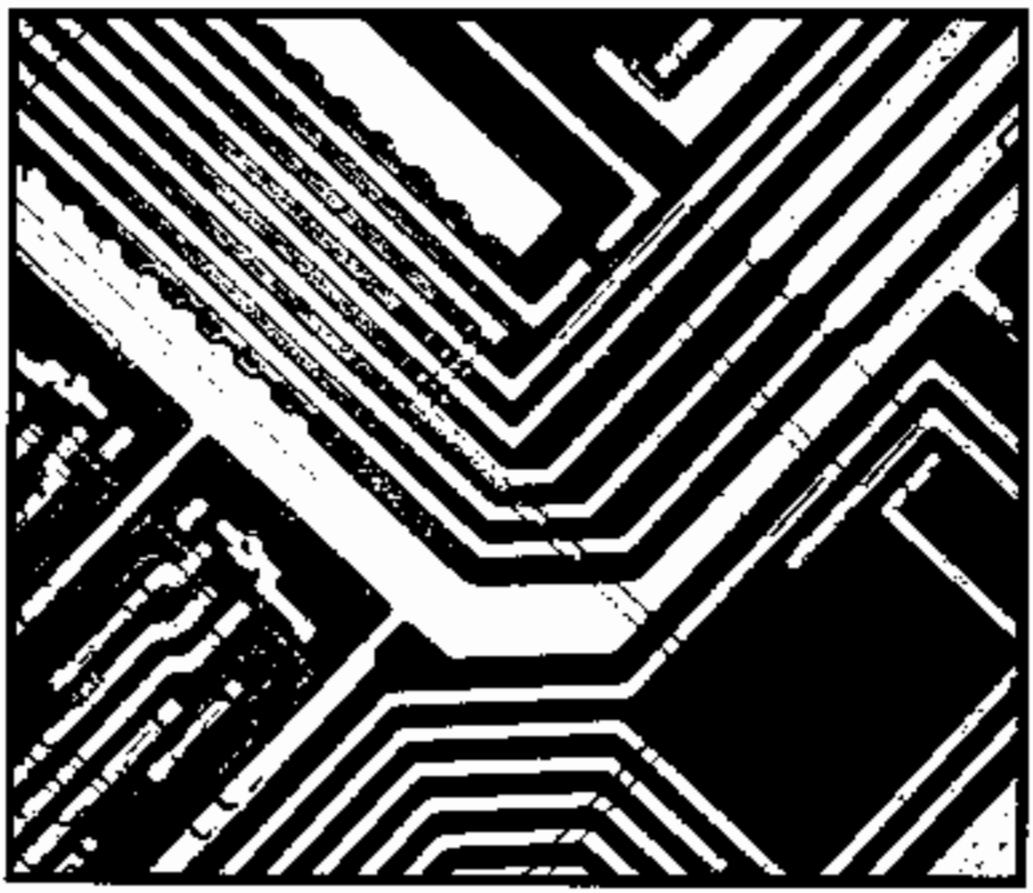
Notice: This is not a final specification.
Some parametric limits are subject to change.

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC68450

Advance Information

Direct Memory Access Controller (DMAC)



MOTOROLA INC. 1985

MOTOROLA

AD 1016

TABLE OF CONTENTS

Paragraph Number	Title	Page Number
	Section 1 Introduction	
1.1	General Operation	1-1
1.2	Operand Transfer Modes	1-4
1.3	Channel Operating Modes	1-5
1.4	Interrupt Operation	1-7
1.5	Channel Priority	1-8
1.6	Request Modes	1-8
1.7	Registers	1-8
	Section 2 Signal Description	
2.1	Signal Organization	2-1
2.1.1	Address, Data Bus (AB/D0 through A23/D15)	2-1
2.1.2	Lower Address Bus (A1 through A7)	2-2
2.1.3	Function Codes (FC0 through FC2)	2-2
2.1.4	Asynchronous Bus Control	2-2
2.1.4.1	Chip Select ($\overline{CS}$)	2-2
2.1.4.2	Address Strobe ($\overline{AS}$)	2-2
2.1.4.3	Read/Write (R/$\overline{W}$)	2-2
2.1.4.4	Upper and Lower Data Strobe ($\overline{UDS}$ and $\overline{LDS}$)	2-2
2.1.4.5	Data Transfer Acknowledge ($\overline{DTACK}$)	2-2
2.1.4.6	Bus Exception Control ($\overline{BEC0}$ through $\overline{BEC2}$)	2-3
2.1.5	Multiplex Control	2-3
2.1.5.1	Own ($\overline{OWN}$)	2-4
2.1.5.2	Upper Address Strobe ($\overline{UAS}$)	2-4
2.1.5.3	Data Buffer Enable ($\overline{DBEN}$)	2-4
2.1.5.4	Data Direction ($\overline{DDIR}$)	2-4
2.1.5.5	High Byte ($\overline{HIBYTE}$)	2-4
2.1.6	Bus Arbitration Control	2-4
2.1.6.1	Bus Request ($\overline{BR}$)	2-4
2.1.6.2	Bus Grant ($\overline{BG}$)	2-4
2.1.6.3	Bus Grant Acknowledge ($\overline{BGACK}$)	2-4
2.1.7	Interrupt Control	2-4
2.1.7.1	Interrupt Request ($\overline{IRQ}$)	2-4
2.1.7.2	Interrupt Acknowledge ($\overline{IACK}$)	2-4
2.1.8	Device Control	2-5
2.1.8.1	Request ($\overline{REQ0}$) through $\overline{REQ3}$)	2-5
2.1.8.2	Acknowledge ($\overline{ACK0}$ through $\overline{ACK3}$)	2-5
2.1.8.3	Peripheral Control Line (PCL0) through PCL3)	2-5
2.1.8.4	Data Transfer Complete ($\overline{DTC}$)	2-5

TABLE OF CONTENTS
(CONTINUED)

Paragraph Number	Title	Page Number
2.1.8.5	Done (DONE)	2-5
2.1.9	Clock (CLK)	2-5
2.2	Signal Summary	2-6
	Section 3 **Register Description**	
3.1	Address Registers (MAR, DAR, and BAR)	3-1
3.2	Function Code Registers (MFCR, DFCR, and BFCR)	3-2
3.3	Transfer Count Registers (MTCR and BTCR)	3-3
3.4	Interrupt Vector Registers (NIVR and EIVR)	3-3
3.5	Channel Priority Register (CPR)	3-3
3.6	Control Registers	3-3
3.6.1	Device Control Register (DCR)	3-4
3.6.1.1	External Request Mode	3-4
3.6.1.2	Device Type	3-5
3.6.1.3	Device Port Size	3-5
3.6.1.4	Peripheral Control Line	3-5
3.6.2	Operation Control Register (OCR)	3-5
3.6.2.1	Transfer Direction	3-6
3.6.2.2	Operand Size	3-6
3.6.2.3	Chaining Operation	3-6
3.6.2.4	Request Generation Method	3-6
3.6.3	Sequence Control Register (SCR)	3-6
3.6.4	Channel Control Register (CCR)	3-7
3.7	Status Register	3-8
3.7.1	Channel Status Register (CSR)	3-9
3.7.2	Channel Error Register (CER)	3-9
3.8	General Control Register (GCR)	3-10
3.9	Register Summary	3-11
3.10	Reset Operation Results	3-11
	Section 4 **Bus Operation**	
4.1	Reset Operation	4-1
4.2	MPU Mode Operation	4-2
4.2.1	MPU Read Cycles	4-2
4.2.2	MPU Write Cycles	4-3
4.2.3	Interrupt Acknowledge Cycles	4-3
4.3	DMA Mode Operation	4-4
4.3.1	Dual Address Transfers	4-4
4.3.1.1	Dual Address Read, M68000 Type	4-4
4.3.1.2	Dual Address Write, M68000 Type	4-6
4.3.1.3	Dual Address Read, M6800 Type Device	4-8
4.3.1.4	Dual Address Write, M6800 Type Device	4-10

TABLE OF CONTENTS
(CONTINUED)

Paragraph Number	Title	Page Number
4.3.2	Single Address Transfers	4-12
4.3.2.1	Single Address Read	4-12
4.3.2.2	Single Address Write	4-14
4.3.2.3	High-Byte Operation	4-17
4.4	Bus Exception Control	4-18
4.4.1	$\overline{BEC0}$-$\overline{BEC2}$ Synchronization	4-19
4.4.2	Bus Exception Control Functions	4-20
4.4.2.1	Normal Termination	4-21
4.4.2.2	Halt	4-21
4.4.2.3	Bus Error	4-21
4.4.2.4	Retry	4-21
4.4.2.5	Relinquish and Retry	4-21
4.4.2.6	Undefined $\overline{BEC}$ Codes	4-26
4.4.2.7	Reset	4-26
4.4.3	Bus Exception Control Summary	4-26
4.5	Bus Arbitration	4-27
4.5.1	Requesting and Receiving the Bus	4-27
4.5.2	Bus Overhead Time	4-28
4.5.2.1	Front-End Overhead	4-28
4.5.2.2	Back End Overhead	4-30
4.5.2.3	Inter-Cycle Overhead	4-30
4.6	Asynchronous Versus Synchronous Operation	4-43
4.6.1	Asynchronous Operation	4-43
4.6.2	Synchronous Operation	4-44

Section 5
Operational Description

5.1	General Operating Sequence	5-1
5.2	Channel Initialization	5-2
5.2.1	Memory and Operand Description	5-2
5.2.2	Device Description	5-3
5.2.3	Operation Description	5-3
5.2.3.1	Transfer Direction	5-4
5.2.3.2	Address Sequencing	5-4
5.2.3.3	Transfer Request Generation	5-5
5.2.3.3.1	Internal, Maximum Rate	5-5
5.2.3.3.2	Internal, Limited Rate	5-5
5.2.3.3.3	External, Burst Mode	5-7
5.2.3.3.4	External, Cycle Steal	5-7
5.2.3.3.5	External, Cycle Steal with Hold	5-7
5.2.3.4	Channel Priority	5-7
5.2.3.5	Interrupt Operation	5-8
5.3	Channel Start-Up	5-8
5.4	Data Transfers	5-9

TABLE OF CONTENTS
(CONTINUED)

Paragraph Number	Title	Page Number
5.4.1	Bus Cycle Sequence	5-9
5.4.1.1	Single Address Transfers	5-9
5.4.1.2	Dual Address Transfers	5-10
5.4.1.3	Transfer Count Synchronization	5-11
5.4.2	Effects of Channel Priority	5-11
5.4.3	Exceptional Bus Conditions	5-11
5.4.3.1	Reset	5-12
5.4.3.2	Halt	5-12
5.4.3.3	Bus Error	5-12
5.4.3.4	Retry Operations	5-12
5.4.3.4.1	Retry	5-12
5.4.3.4.2	Relinquish and Retry	5-12
5.4.4	External Request Recognition	5-13
5.4.5	Software Control	5-13
5.4.5.1	Software Halt	5-13
5.4.5.2	Software Abort	5-13
5.4.6	Hardware Abort	5-14
5.5	Base Register Operation	5-14
5.5.1	Continue Operation	5-14
5.5.2	Sequential Array Chaining	5-15
5.5.3	Linked Array Chaining	5-17
5.5.4	BTCR Value Restrictions	5-19
5.6	Channel Termination	5-19
5.6.1	Transfer Count Exhausted	5-19
5.6.2	Device Termination	5-19
5.6.3	Error Termination	5-20
5.6.3.1	Internal Errors	5-20
5.6.3.1.1	Configuration Error	5-20
5.6.3.1.2	Operation Timing Error	5-21
5.6.3.1.3	Address Error	5-21
5.6.3.1.4	Count Error	5-21
5.6.3.2	External Errors	5-21

Section 6
Electrical Specifications

6.1	Maximum Ratings	6-1
6.2	Thermal Characteristics	6-1
6.3	Power Considerations	6-2
6.4	DC Electrical Characteristics	6-3
6.5	AC Electrical Specifications — Clock Timing	6-3
6.6	AC Electrical Specifications — Read and Write Cycles	6-4

TABLE OF CONTENTS
(CONCLUDED)

Paragraph Number	Title	Page Number
	Section 7 **Ordering Information**	
	Section 8 **Mechanical Data**	
8.1	Pin Assignments	8-1
8.1.1	64-Pin Dual in Line Packages	8-1
8.1.2	68 Terminal Pin Grid Array	8-2
8.2	Package Dimensions	8-3

LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
1-1	Typical M68000 Based System Configuration	1-2
1-2	Typical System Configuration with M6800 Type Peripheral Devices	1-3
1-3	Implicitly Addressed Device Interface	1-4
1-4	Dual-Address Transfer Sequence	1-4
1-5	Single Block Transfer	1-5
1-6	Array Chain Transfer	1-6
1-7	Linked Array Chain Transfer	1-7
1-8	DMAC Programmer's Model	1-8
2-1	Signal Organization	2-1
2-2	Demultiplex Logic	2-3
3-1	Register Memory Map	3-2
3-2	Register Summary	Foldout 1
4-1	Relationship Between External and Internal Signals	4-1
4-2	Reset Operation Timing Diagram	4-2
4-3	MPU Read Cycle Timing Diagram	4-2
4-4	MPU Write Cycle Timing Diagram	4-3
4-5	Interrupt Acknowledge Cycle Timing Diagram	4-4
4-6	M68000 Type Dual Address Read Cycle Timing Diagram	4-5
4-7	M68000 Type Dual Address Write Cycle Timing Diagram	4-7
4-8	M6800 Type Read Cycle Timing Diagram	4-9
4-9	M6800 Type Write Cycle Timing Diagram	4-11
4-10	Single Address Read Cycle Timing Diagram	4-13
4-11	Single Address Write Cycle Timing Diagram	4-15
4-12	$\overline{\text{READY}}$ Circuit for Device with Acknowledge and Ready	4-17
4-13	Implicitly Addressed 8-Bit Device with 16-Bit Memory Data Flow	4-18
4-14	Example $\overline{\text{BEC}}$ Signal Generation Circuit	4-19
4-15	$\overline{\text{BEC}}$ Synchronizer Functional Diagram	4-19
4-16	$\overline{\text{BEC}}$ Debouncer Timing Diagram	4-20
4-17	Halt Operation Timing Diagram	4-22
4-18	Bus Error Operation	4-23
4-19	Retry Operation Timing Diagram	4-24
4-20	Relinquish and Retry Operation Timing Diagram	4-25
4-21	Undefined $\overline{\text{BEC}}$ Operation Timing Diagram	4-26
4-22	Bus Exception Control Flow Diagram	4-27
4-23	Bus Arbitration Timing Diagram (Best Case)	4-29
4-24	Front End Overhead (Worst Case)	4-29
4-25	DMA Operation Timing Table	4-31
4-26	Idle-Cycle Overhead Timing Diagram	4-38

LIST OF ILLUSTRATIONS (CONTINUED)

Figure Number	Title	Page Number
5-1	Memory Data Formats	5-2
5-2	Limited-Rate Auto-Request Operation Timing Parameter Relationships	5-5
5-3	Continue Mode Operation Flowchart	5-15
5-4	Sequential Array Chaining Mode Example	5-16
5-5	Linked Array Chaining Mode Example	5-18
5-6	$\overline{\text{READY}}$ Circuit for Device with Acknowledge and Ready	5-19
6-1	Test Loads	6-1
6-2	Clock Input Timing Diagram	6-2
6-3	MPU Read/Write Timing Diagram	Foldout 2
6-4	Bus Arbitration Timing Diagram	Foldout 2
6-5	DMA Read/Write Timing Diagram (Single Cycle)	Foldout 3
6-6	DMA Read/Write Timing Diagram (Dual Cycle)	Foldout 3
6-7	DMA Read/Write Timing Diagram (Single Cycle with PCL as $\overline{\text{READY}}$)	Foldout 4
6-8	DONE Input Timing Diagram	Foldout 4
7-1	68 Pin Grid Array	7-1

LIST OF TABLES

Table Number	Title	Page Number
2-1	Signal Summary	2-9
3-1	Reset Operation Results	3-11
4-1	$\overline{\text{BEC}}$ Encoding Definitions	4-18
5-1	Device and Operand Size Combinations	5-3
5-2	Address Register Sequencing Rules	5-5
5-3	Limited-Rate Auto-Request Timing Parameter Value Combinations	5-6
5-4	Bus Cycles per Operand Transfer Dual Address	5-10
5-5	Chaining Mode Address Count Information	5-17
5-6	Channel Error Condition Summary	5-20

SECTION 1
INTRODUCTION

M68000 microprocessors utilize state-of-the-art MOS technology to maximize performance and throughput. The MC68450 direct memory access controller (DMAC) is designed to complement the performance and architectural capabilities of M68000 Family microprocessors by moving blocks of data in a quick, efficient manner with minimum intervention from a processor. The DMAC performs memory-to-memory, memory-to-device, and device-to-memory data transfers by utilizing the following features:

- Four Independent DMA Channels with Programmable Priority
- Asynchronous M68000 Bus Structure with a 24-Bit Address and a 16-Bit Data Bus
- Fast Transfer Rates: Up to 5 Megabytes per Second at 10 MHz, No Wait States
- Fully Supports all M68000 Bus Options such as Halt, Bus Error, and Retry
- FIFO Lookahead Stage Support with Device Transfer Complete Signal
- Flexible Request Generation
  - Internal, Maximum Rate
  - Internal, Limited Rate
  - External, Cycle Steal (With or Without Hold)
  - External, Burst
  - Mixed Internal and External
- Programmable 8-Bit or 16-Bit Peripheral Device Types:
  - Explicitly Addressed, M68000 Type
  - Explicitly Addressed, M6800 Type
  - Implicitly Addressed:
    - Device with Request and Acknowledge
    - Device with Request, Acknowledge, and Ready
- Pin and Register Compatible Functional Superset of the MC68440 DDMA

1.1 GENERAL OPERATION

The main purpose of a direct memory access (DMA) controller in any system is to transfer data at very high rates, usually much faster than a microprocessor under software control can handle. The term DMA is used to refer to the ability of a peripheral device to access memory in a system in the same manner as a microprocessor does. DMA operation can occur concurrently with other operations that the system processor needs to perform, thus greatly boosting overall system performance.

Figure 1-1 illustrates a typical system configuration using a DMA interface to a high-speed disk storage device. In a system such as this, the DMAC will move blocks of data between the peripheral controller and memory at rates approaching the limits of the memory bus, since the simple function of data movement is implemented in high-speed MOS hardware. A block of data consists of a sequence of byte, word, or long-word operands starting at a specific address in memory with the length of the block determined by a transfer count. A single channel operation may involve the transfer of several blocks of data between the memory and a device.

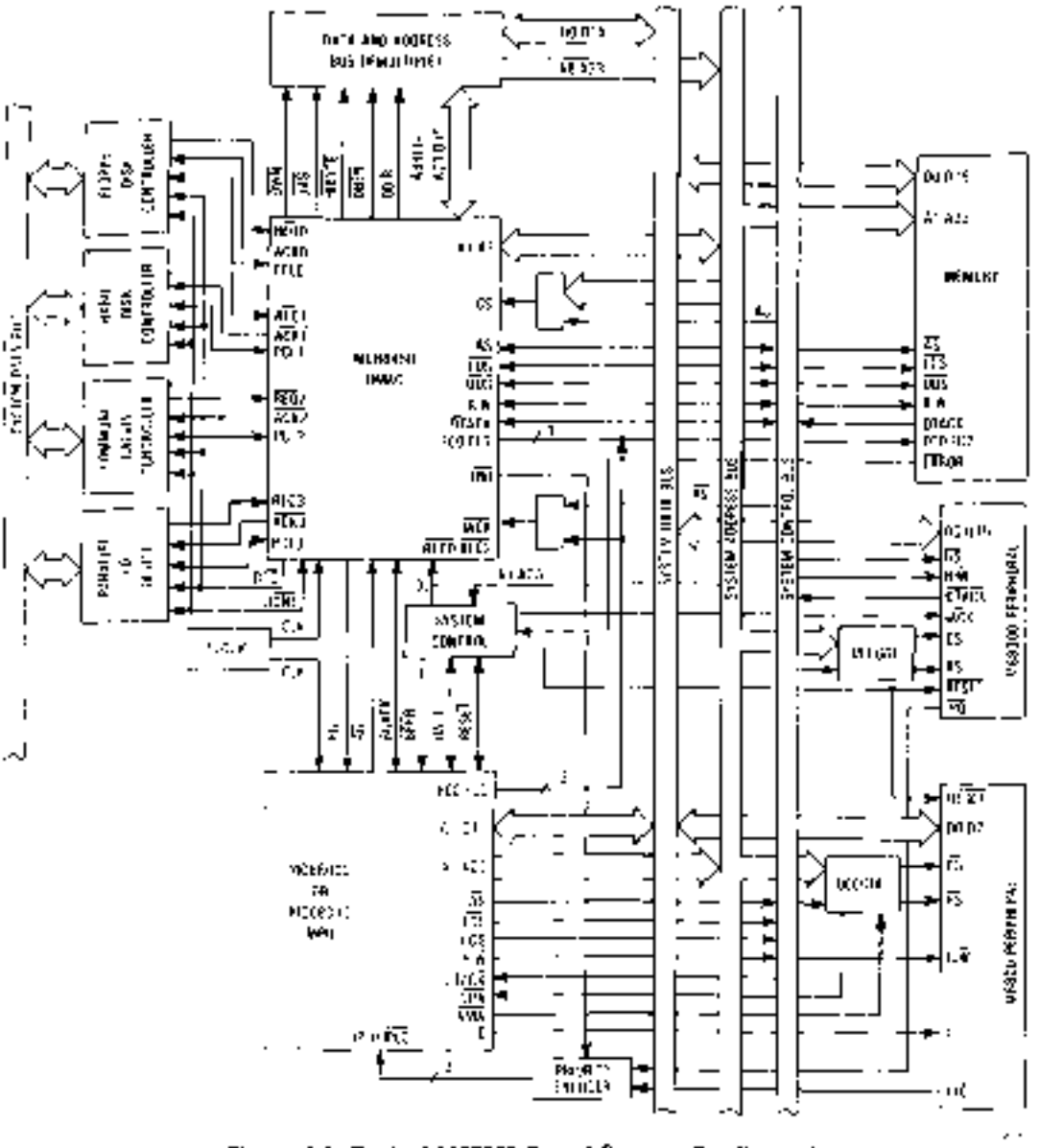


Figure 1-1. Typical M68000 Based System Configuration

Any operation involving the DMAC will follow the same basic steps: channel initialization by the system processor, data transfer, and block termination. In the initialization phase, the host processor loads the registers of the DMAC with control information, address pointers, and transfer counts and then starts the channel. During the transfer phase, the DMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The termination phase occurs after the operation is complete, when the DMAC indicates the status of the operation in the status register. During all phases of a data transfer operation, the DMAC will be in one of three operating modes.

IDLE This is the state that the DMAC assumes when it is reset by an external device and waiting for initialization by the system processor or an operand transfer request from a peripheral.

MPU This is the state that the DMAC enters when it is chip selected by another bus master in the system (usually the main system processor). In this mode, the DMAC internal registers are written or read, to control channel operation or check the status of a block transfer.

DMA This is the state that the DMAC enters when it is acting as a bus master to perform an operand transfer.

In addition to fully supporting the M68000 Family bus, the DMAC also supports transfers to or from M6800 Family peripheral devices. Figure 1-2 illustrates a typical system configuration utilizing an M6800 type peripheral device.

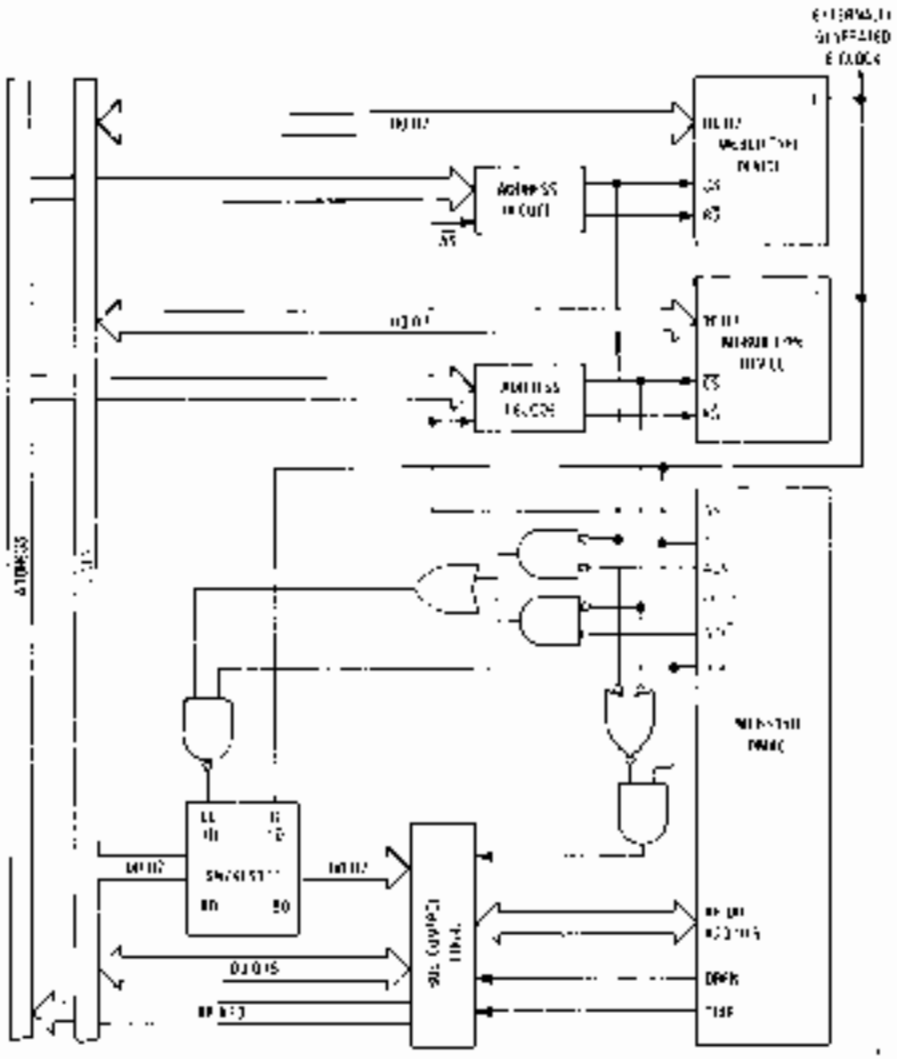


Figure 1-2. Typical System Configuration with M6800-Type Peripheral Devices

1.2 OPERAND TRANSFER MODES

The DMAC can perform one of eight explicit address data transfers using any of the following protocols:

1) Explicitly addressed, MC68000 compatible, dual-cycle
2) Explicitly addressed, MC6800 compatible device
3) Implicitly addressed, device with acknowledge, and
4) Implicitly addressed, device with acknowledge and ready.

In the first two protocols, data is transferred from the source to an internal DMAC holding register, and then on the next bus cycle moved from the holding register to the destination. Protocols 3 and 4 require only one bus cycle for data transfer, since only one device needs to be addressed. With these protocols, communication is performed using a DAC signal and three signal handshake, respectively.

Implicitly addressed devices do not require the generation of a device data register address for a data transfer. Such a device is controlled by a five signal device control interface on the DMAC during implicit address transfers as shown in Figure 1-3. Since only memory is addressed during such a data transfer, this method is called the single address method.

Explicitly addressed devices require that a data register within the peripheral device be addressed. No signals other than the M68000 asynchronous bus control signals are needed to interface with such a device, although any of the five device control signals may also be used. Because the address bus is used to access the peripheral, the data cannot be directly transferred to/from memory since memory also requires addressing. Therefore, data is transferred from the source to an internal holding register in the DMAC and then transferred to the destination during a second bus transfer as shown in Figure 1-4. Since both memory and the device are addressed during such a data transfer, this method is called the dual address method.

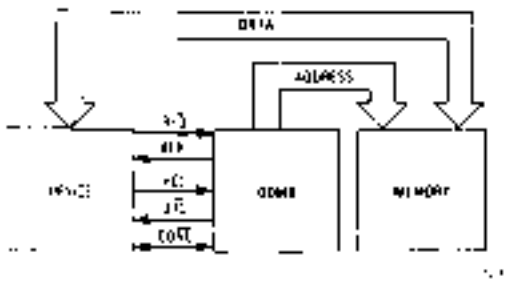


Figure 1-3. Implicitly Addressed Device Interface

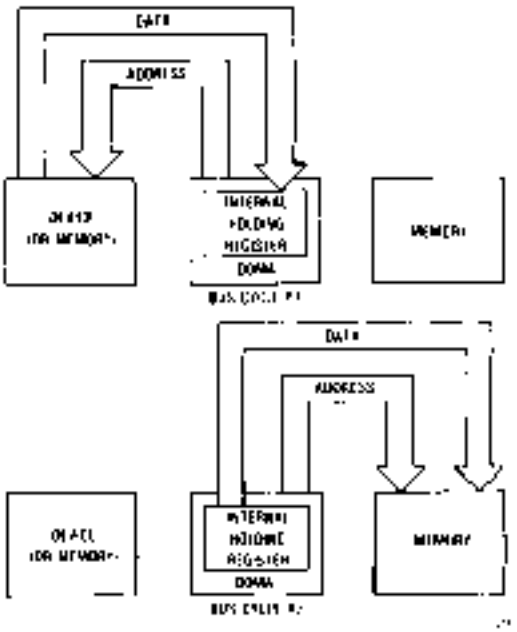


Figure 1-4 Dual-Address Transfer Sequence

1.2 CHANNEL OPERATING MODES

There are three types of channel operations: 1) single block transfers, 2) continued operation, and 3) chained operations. The first two modes utilize on-chip registers while the last mode uses an on-chip address register to point to address and count parameters stored in system memory.

When transferring single blocks of data, the memory address and device address registers are initialized by the user to specify the source and destination of the transfer. Also initialized is the memory transfer count register to count the number of operands transferred in a block. Repeated transfers are possible with the continue mode of operation, where the memory address and transfer count registers are automatically loaded from internal registers upon completion of a block transfer. See Figure 1-5

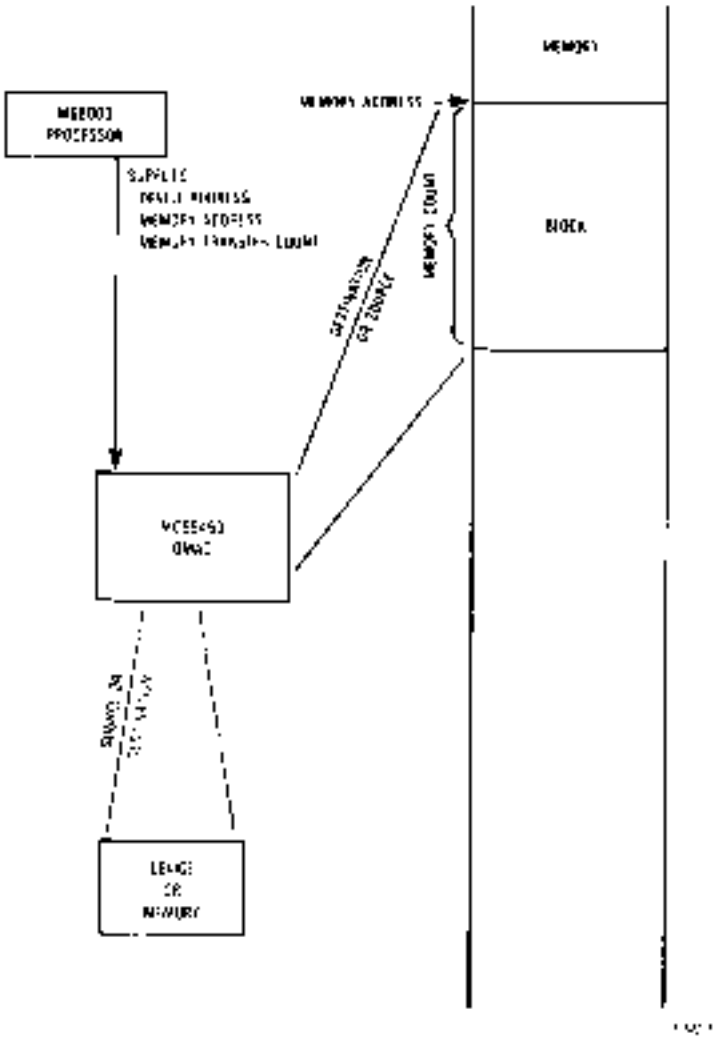


Figure 1-5. Single Block Transfer

The two chaining modes are array chaining and linked array chaining. The array chaining mode operates from a contiguous array in memory consisting of memory addresses and transfer counts. The base address register and base transfer count register are initialized to point to the beginning address of the array and the number of array entries, respectively. As each block transfer is completed, the next entry is fetched from the array, the base transfer count is decremented, and the base address is incremented to point to the next array entry. When the base transfer count reaches zero, the entry just fetched is the last block transfer defined in the array. See Figure 1-6

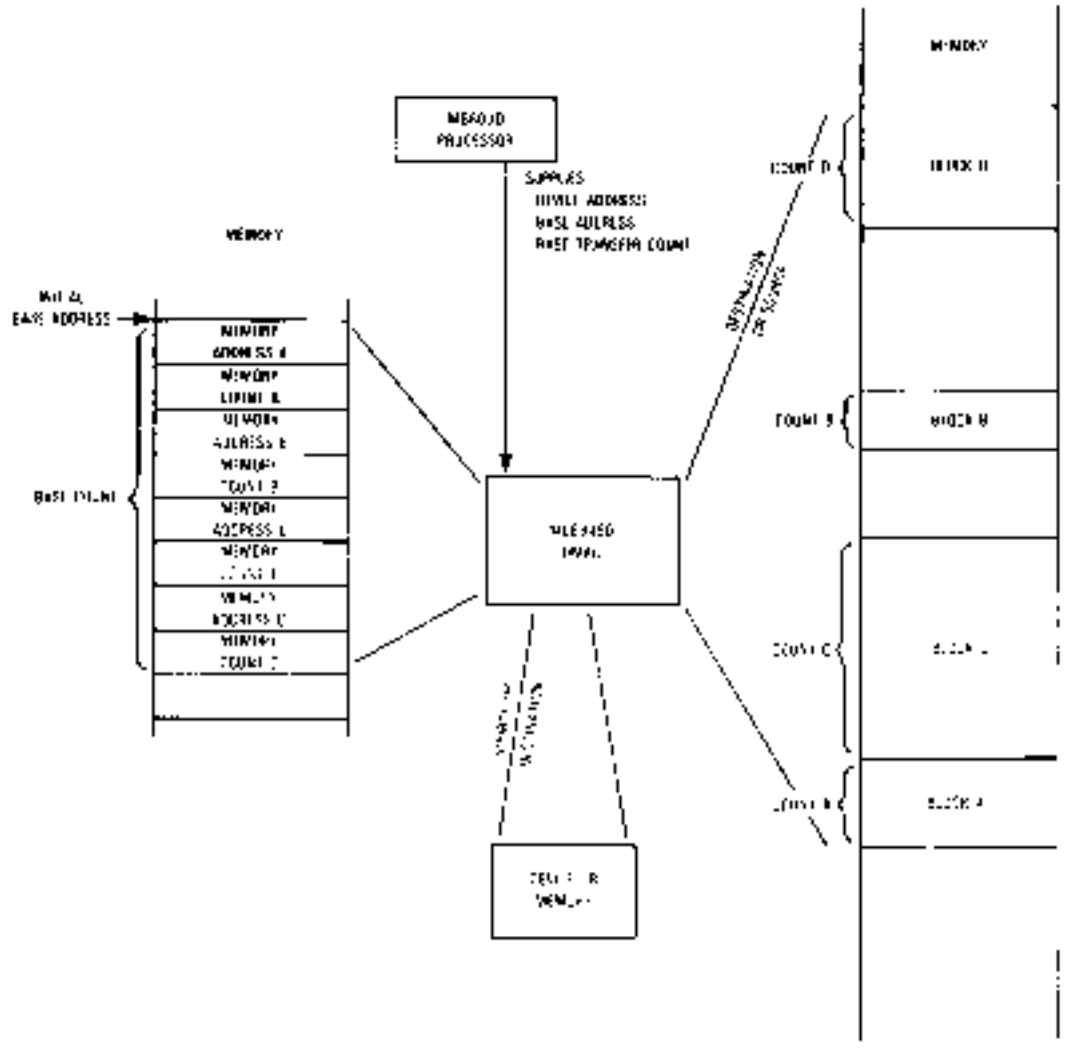


Figure 1-6 Array Chain Transfer

The linked array chaining mode is similar to the array chaining mode, except that each entry in the memory array also contains a link address which points to the next entry in the array. This allows a non contiguous memory array. The last entry contains a link address set to zero. No base transfer count register is needed in this mode. The base address register is initialized to the address of the first entry in the array. The link address is used to update the base address register at the beginning of each block transfer. This chaining mode allows array entries to be easily moved or inserted without having to reorganize the array into sequential order. Also, the number of entries in the array need not be specified to the DMAC. See Figure 1-7.

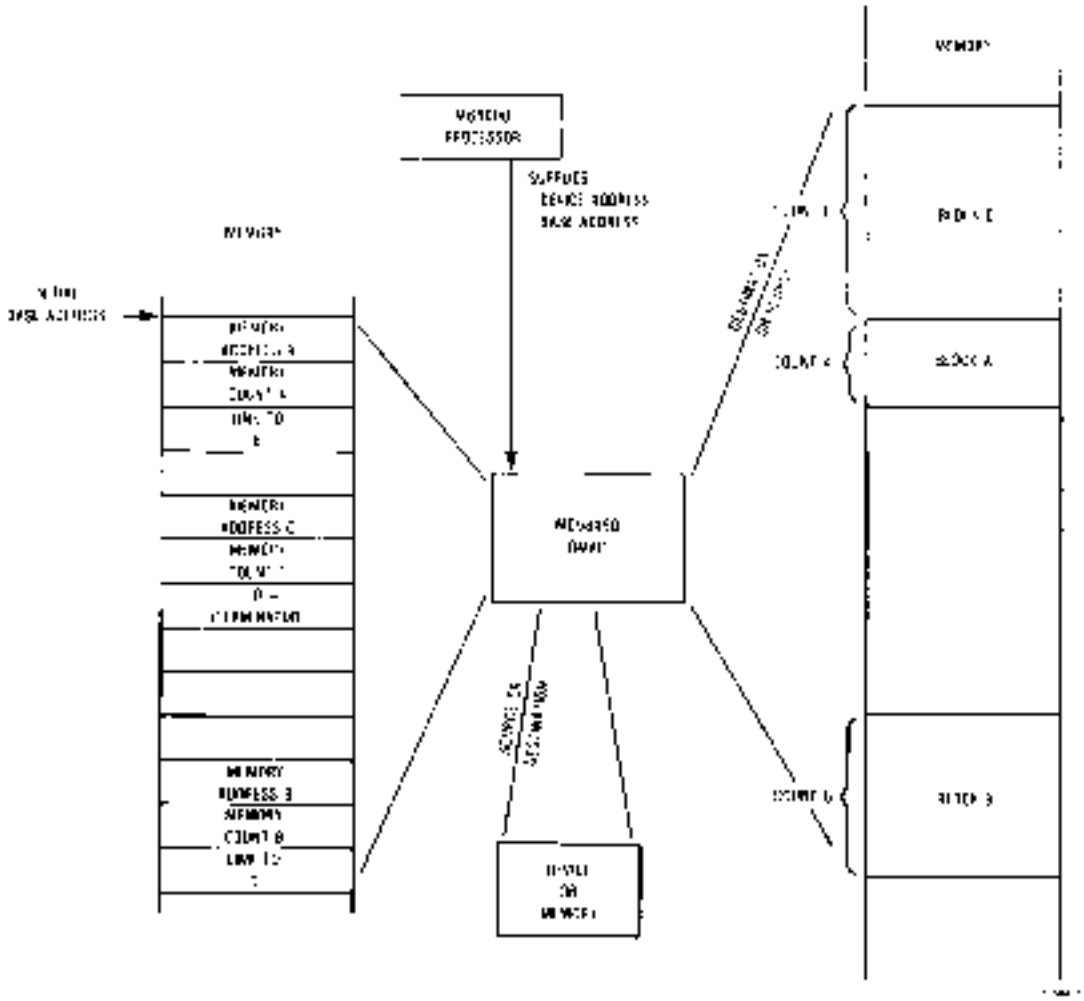


Figure 1-7. Linked Array Chain Transfer

1.4 INTERRUPT OPERATION

The DMAC will interrupt the MPU for a number of event occurrences such as the completion of a DMA operation, or at the request of a peripheral device using a PCL line. The user must write interrupt vectors into an on-chip vector register, for use in the M68000 vectored interrupt structure. Two vector registers are available for each channel.

1.5 CHANNEL PRIORITY

Each channel may be given a priority level of 0, 1, 2, or 3. If several channel requests occur at the same priority level, a round-robin mode is entered automatically.

1.6 REQUEST MODES

Requests may be externally generated by a device or internally generated by the auto-request mechanism of the DMAC. Auto-requests may be generated either at the maximum rate, where the channel always has a request pending, or at a limited rate determined by selecting a portion of the bus bandwidth to be available for DMA activity. External requests can be either burst requests or cycle steal requests that are generated by the request signal associated with each channel.

1.7 REGISTERS

The DMAC contains 17 registers for each of the four channels plus one general control register, all of which are under complete software control. The user programmer's model of the registers is shown in Figure 1-8.

The DMAC registers contain information about the data transfers such as the source and destination address and function codes, transfer count, operand size, device port size, channel priority, continuation address and transfer count, and the function of the peripheral control line. One register also provides status and error information on channel activity, peripheral inputs, and various events which may have occurred during a DMA transfer. A general control register selects the bus utilization factor to be used in limited rate auto-request DMA operations.

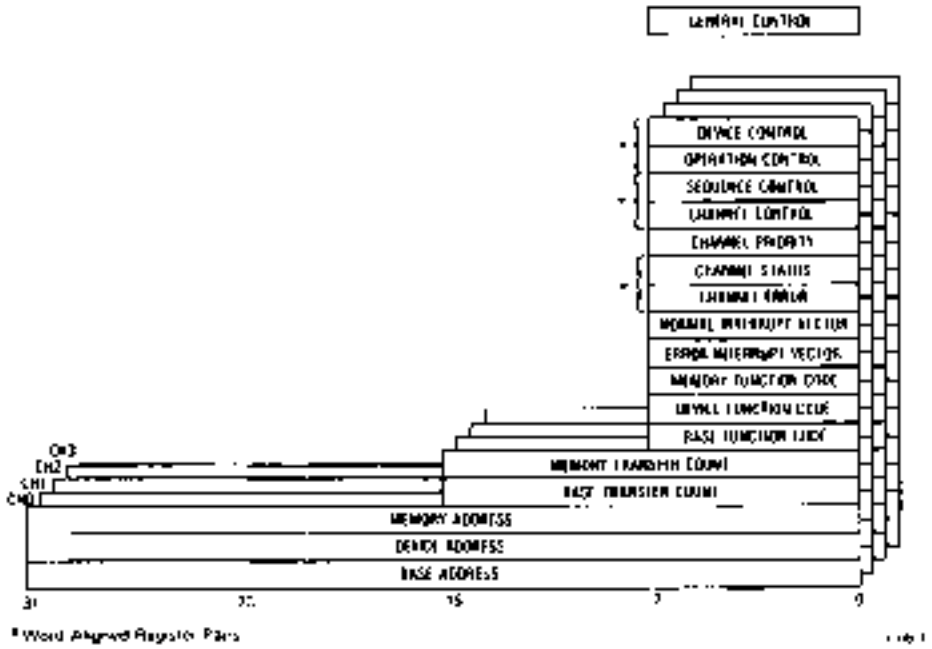


Figure 1-8. DMAC Programmer's Model

SECTION 2
SIGNAL DESCRIPTION

This section contains a brief description of the DMAC input and output signals. Included at the end of the functional description of the signals is a table describing the electrical characteristics of each pin (i.e., the type of driver used).

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active low" and "active high" signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

2.1 SIGNAL ORGANIZATION

The input and output signals can be functionally organized into the groups shown in Figure 2-1. The function of each signal or group of signals is discussed in the following paragraphs.

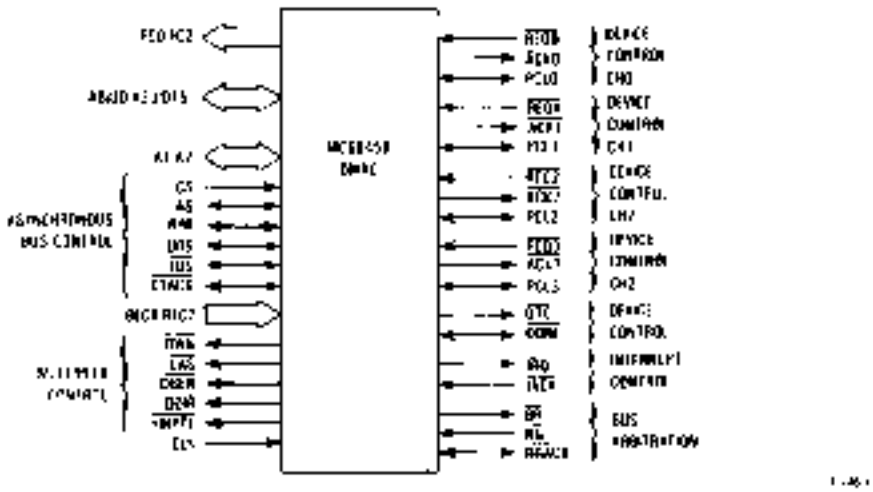


Figure 2-1. Signal Organization

2.1.1 Address/Data Bus (AB/D0 through A23/D15)

This 16-bit bus is time multiplexed to provide address outputs during the DMA mode of operation and is used as a bidirectional data bus to input data from an external device during an MPU write or DMA read

or to output data to an external device (during an MPU read or a DMA write). This is a three-state bus and is demultiplexed using external latches and buffers controlled by the multiplex control lines (refer to **2.1.5 Multiplex Control**).

2.1.2 Lower Address Bus (A1 through A7)

These bidirectional three-state lines are used to address the DMAC internal registers in the MPU mode and to provide the lower seven address outputs in the DMA mode.

2.1.3 Function Codes (FC0 through FC2)

These three-state output lines are used in the DMA mode to further qualify the value on the address bus to provide eight separate address space that may be defined by the user. The value placed on these lines is taken from one of the internal function code registers, depending on the register that provides the address used during a DMA bus cycle.

2.1.4 Asynchronous Bus Control

Asynchronous data transfers are handled using the following control signals: chip select, address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are described in the following paragraphs.

2.1.4.1 CHIP SELECT ($\overline{CS}$). This input signal is used to select the DMAC for an MPU bus cycle. When $\overline{CS}$ is asserted, the address on A1-A7 and the data strobes (or A0 when using an 8-bit bus) select the internal DMAC register that will be involved in the transfer. $\overline{CS}$ should be generated by qualifying an address decode signal with the address and data strobes.

2.1.4.2 ADDRESS STROBE (AS). This bidirectional signal is used as an output in the DMA mode to indicate that a valid address is present on the address bus. In the MPU or IDLE modes, it is used as an input to determine when the DMAC can take control of the bus (if the DMAC has requested and been granted use of the bus).

2.1.4.3 READ/WRITE (R/$\overline{W}$). This bidirectional signal is used to indicate the direction of a data transfer during a bus cycle. In the MPU mode, a high level indicates that a transfer is from the DMAC to the data bus and a low level indicates a transfer from the data bus to the DMAC. In the DMA mode, a high level indicates a transfer from the addressed memory or device to the data bus, and a low level indicates a transfer from the data bus to the addressed memory or device.

2.1.4.4 UPPER AND LOWER DATA STROBE ($\overline{UDS}$ AND $\overline{LDS}$). These bidirectional lines indicate when data is valid on the bus and what portions of the bus should be involved in a transfer.

During any bus cycle, $\overline{UDS}$ is asserted if data is to be transferred over data lines D8-D15 and $\overline{LDS}$ is asserted if data is to be transferred over data lines D0-D7. $\overline{UDS}$, $\overline{LDS}$ are asserted by the DMAC when operating in the DMA mode and by another bus master when in the MPU mode.

2.1.4.5 DATA TRANSFER ACKNOWLEDGE ($\overline{DTACK}$). This bidirectional line is used to signal that an asynchronous bus cycle may be terminated. In the MPU mode, this output indicates that the DMAC has accepted data from the MPU or placed data on the bus for the MPU. In the DMA mode, this input is monitored by the DMAC to determine when to terminate a bus cycle. As long as $\overline{DTACK}$ remains negated,

the DMAC will insert wait cycles into a bus cycle and when $\overline{DTACK}$ is asserted, the bus cycle will be terminated (except when PCL is used as a ready signal, in which case both signals must be asserted before the cycle is terminated).

2.1.4.6 BUS EXCEPTION CONTROL ($\overline{BEC0}$ THROUGH $\overline{BEC2}$). These input lines provide an encoded signal that indicates an abnormal bus condition such as a bus error or reset. Refer to **4.4.2 Bus Exception Control Functions** for a detailed description of the function of these pins.

2.1.5 Multiplex Control

These signals are used to control external multiplex/demultiplex devices to separate the address and data information on the A8/D8–A23/D15 lines and to transfer data between the upper and lower halves of the data bus during certain DMA bus cycles.

Figure 2-2 shows the five external devices needed to demultiplex the address/data pins and the interconnection of the multiplex control signals. The SN74LS245 that may connect the upper and lower halves of the data bus is needed only if an 8-bit device is used during single address transfers.

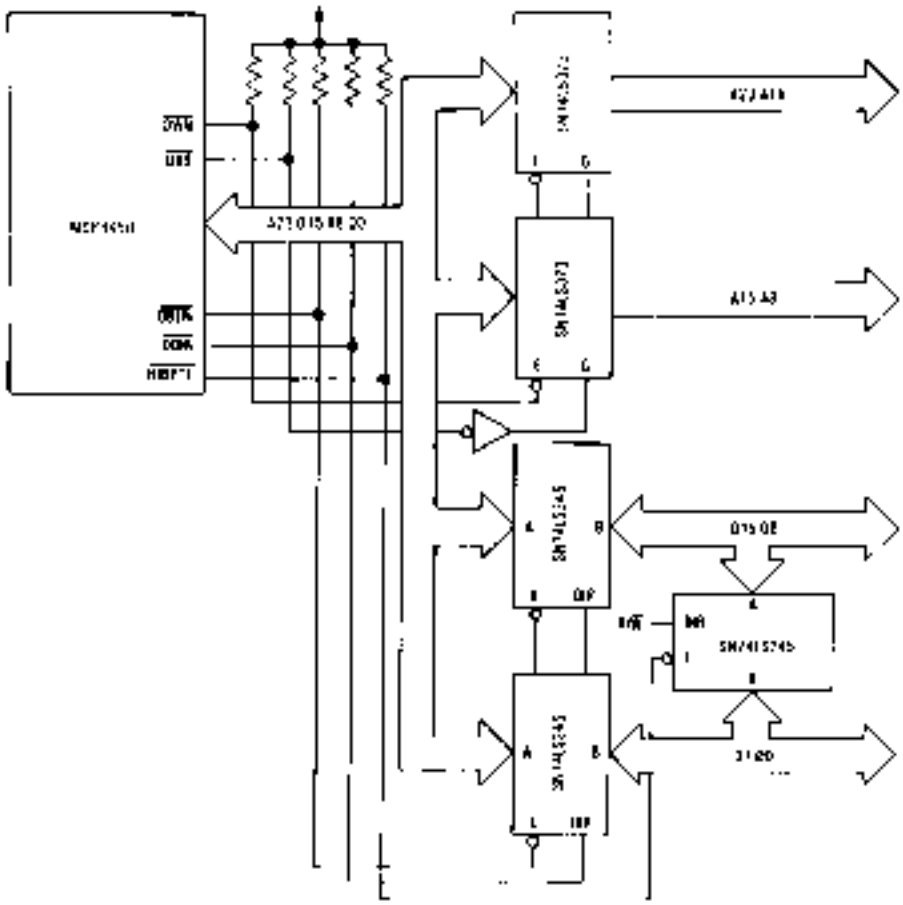


Figure 2-2. Demultiplex Logic

2.1.5.1 OWN (OWN). This three-state output indicates that the DMAC is controlling the bus. It is used as the enable signal to turn on the external address drivers and control signal buffers.

2.1.5.2 UPPER ADDRESS STROBE (UAS). This three state output is used as the gate signal to the transparent latches that capture the value of A8-A23 on the multiplexed address/data bus.

2.1.5.3 DATA BUFFER ENABLE (DBEN). This three state output is used as the enable signal to the external bidirectional data buffers.

2.1.5.4 DATA DIRECTION (DDIR). This three-state output controls the direction of the external bidirectional data buffers. If DDIR is high, the data transfer is from the DMAC to the data bus. If DDIR is low, the data transfer is from the data bus to the DMAC.

2.1.5.5 HIGH BYTE (HIBYTE). This three-state output indicates that data will be present on data lines D8-D15 that must be transferred to data lines D0-D7, or vice versa, through an external buffer during a single address transfer between an 8-bit device and memory (refer to **4.3.2 Single Address Transfers**).

2.1.6 Bus Arbitration Control

These three signals form a bus arbitration circuit used to determine which device in a system will be the current bus master.

2.1.6.1 BUS REQUEST (BR). This output is asserted by the DMAC to request control of the bus.

2.1.6.2 BUS GRANT (BG). This input is asserted by an external bus arbiter to inform the DMAC that it may assume bus mastership as soon as the current bus cycle is completed. The DMAC will not take control of the bus until CS, IACK, AS, and BGACK are all negated.

2.1.6.3 BUS GRANT ACKNOWLEDGE (BGACK). This bidirectional signal is asserted by the DMAC to indicate that it is the current bus master. BGACK is monitored as an input to determine when the DMAC can become bus master and if a bus master other than the system MPU is a master during limited rate auto-request operation.

2.1.7 Interrupt Control

These two signals form an interrupt request/acknowledge handshake circuit with an MPU.

2.1.7.1 INTERRUPT REQUEST (IRQ). This output is asserted by the DMAC to request service from the MPU.

2.1.7.2 INTERRUPT ACKNOWLEDGE (IACK). This input is asserted by the MPU to acknowledge that it has received an interrupt from the DMAC. In response to the assertion of IACK, the DMAC will place a vector on D0-D7 that will be used by the MPU to fetch the address of the proper DMAC interrupt handler routine.

2.1.8 Device Control

These eight lines perform the interface between the DMAC and four peripheral devices. Four sets of three lines are dedicated to a single DMAC channel and its associated peripheral; the remaining two lines are global signals shared by all channels.

2.1.8.1 REQUEST ($\overline{REQ0}$ THROUGH $\overline{REQ3}$). These inputs are asserted by a peripheral device to request an operand transfer between that peripheral device and memory. In the cycle steal request generation mode, these inputs are edge sensitive; in burst mode, they are level sensitive.

2.1.8.2 ACKNOWLEDGE ($\overline{ACK0}$ THROUGH $\overline{ACK3}$). These outputs are asserted by the DMAC to signal to a peripheral that an operand is being transferred in response to a previous transfer request.

2.1.8.3 PERIPHERAL CONTROL LINE (PCL0 THROUGH PCL3). These bidirectional lines are multipurpose signals that may be programmed to function as ready, abort, reload, status, interrupt, or enable clock inputs or as start pulse outputs.

2.1.8.4 DATA TRANSFER COMPLETE ($\overline{DTC}$). This output is asserted by the DMAC during any DMAC bus cycle to indicate that data has been successfully transferred (i.e., the bus cycle was not terminated abnormally).

2.1.8.5 DONE ($\overline{DONE}$). This bidirectional signal is asserted by the DMAC or a peripheral device during DMA bus cycle to indicate that the data being transferred is the last item in a block. The DMAC will assert this signal during a bus cycle when the memory transfer count register is decremented to zero.

2.1.9 Clock (CLK)

The clock input is a TTL compatible signal that is internally buffered for development of the internal clocks needed by the DMAC. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

2.2 SIGNAL SUMMARY

Table 2-1 is a summary of the signals discussed in the previous paragraphs.

Table 2-1. Signal Summary

Signal Name	Direction	Active State	Driver Type	Synchronizing Clock Edge
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

[illegible]

[illegible]

SECTION 3
REGISTER DESCRIPTION

This section contains a description of the DMAC internal registers and the control bit assignments within each register. Figure 3-1 shows the memory mapped locations of the registers for each channel. Figure 3-2 (found on a foldout page at the end of this document) shows the register summary and may be used for a quick reference to the bit definitions within each register.

The register memory map for the MC68450 DMAC is identical to the register memory map for the MC68440 Dual Channel DMA Controller (DDMAC), including the individual bit assignments within the registers. However, not all functional options available on the DMAC are available on the DDMA and vice versa. If any programmable options labeled "MC68440 Reserved" or "Undefined, Reserved" are programmed into a DMAC channel, a configuration error will occur when the MPU attempts to start that channel.

The description given in this section is for a single channel, but applies to all channels since they are identical. When the operation of a register affects the four channels differently or is common to all channels, that fact will be included in the register description.

All registers within the DMAC are always accessible as bytes or words by the MPU (assuming that the MPU can gain control of the DMA bus); however, some registers may not or should not be modified while a channel is actively transferring data. If a register may not be modified during operation and an attempt is made to write to it, an operation timing error will be signaled and the channel operation aborted. Refer to **5.3.1 Programming Sequence** for the proper order to use when writing to channel registers in order to start an operation.

3.1 ADDRESS REGISTERS (MAR, DAR, AND BAR)

These three 32-bit registers contain addresses that are used by the DMAC to access memory while it is a bus master. Since the DMAC only has 24 address lines available externally, the upper eight bits of these registers are truncated; however, the upper byte is supported for reads and writes for compatibility with future expanded address capabilities. The upper eight bits will also be incremented if necessary, for example, an address register will count from $00FFFFFF to $01000000. If such an increment occurs, the external address bus will "roll over" to $000000 and the DMAC will continue normal operation without any error indication.

The memory address register (MAR) contents are used whenever the DMAC is running a bus cycle to fetch or store an operand part in memory. Likewise, the device address register (DAR) contents are used when the DMAC is running a bus cycle to transfer an operand part to or from a peripheral device during a dual address transfer (refer to **4.3 DMA MODE OPERATION**). The base address register (BAR) contents are used when the DMAC is performing table accesses in the chained modes of operation. The BAR is used by the DMAC in the continue mode of operation to hold the address to be transferred to the MAR in response to a continue operation.

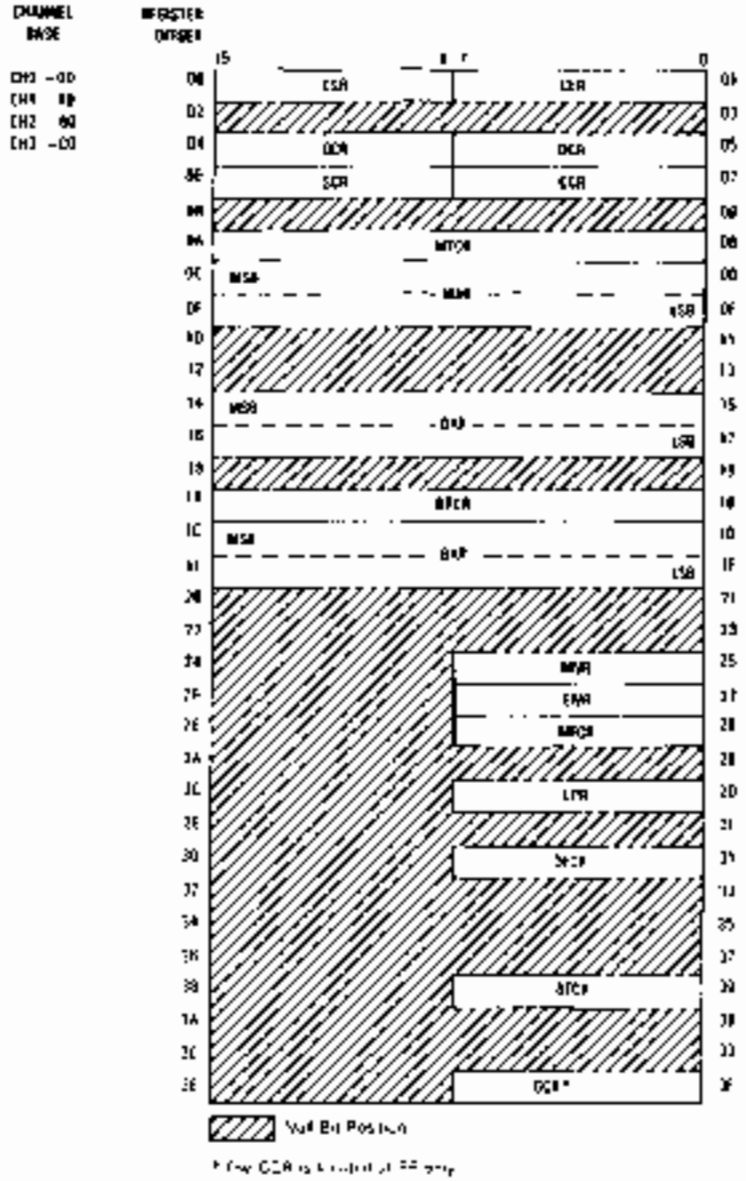


Figure 3-1. Register Memory Map

3.2 FUNCTION CODE REGISTERS (MFCR, DFCR, AND BFCR)

These 4-bit registers contain function code values that are used by the DMAC in conjunction with the three address registers during a DMA bus cycle. The address space value on the function code lines may be used by an external memory management unit (MMU) or other memory protection device to translate the DMAC logical addresses into the proper physical addresses. The value programmed into the memory function code register (MFCR), the device function code register (DFCR), or the base function code register

(BFCR) is placed on pins FC2-FC0 during a bus cycle to further qualify the address bus value, which will be the contents of the MAR, DAR, or BAR respectively. The BFCR is used by the DMAC in the continue mode of operation to hold the function code to be transferred to the MFCR in response to a continue operation request. Note that these are 4 bit registers, with bit 3 readable and writeable for future compatibility; however, it is not available as an external signal. Bits 4-7 of the function code registers will always read as zeros and are not affected by writes.

3.3 TRANSFER COUNT REGISTERS (MTCR AND BTCR)

These 16-bit registers are programmed with the operand count for a block transfer. The memory transfer count register (MTCR) is decremented each time the DMAC successfully transfers an operand between memory and a device until it is zero, at which time the channel will terminate the operation. The base transfer count register (BTCR) is used in the sequential array chained mode to count the number of block transfers to perform during a channel operation. The BTCR is used in the continue mode of operation to hold the operand count to be transferred to the MTCR in response to a continue operation request.

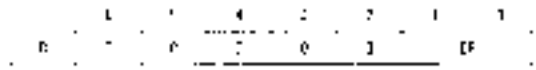
3.4 INTERRUPT VECTOR REGISTERS (NIVR AND EIVR)

These two 8-bit registers are used by the DMAC during interrupt acknowledge bus cycles. When the $\overline{IACK}$ signal is asserted to the DMAC, the contents of the normal interrupt vector register (NIVR) or the error interrupt vector register (EIVR) for the highest priority channel with an interrupt pending will be placed on A8-D0-A15-D7. If the ERR bit in the corresponding channel status register is clear when $\overline{IACK}$ is asserted, the contents of the NIVR will be used; otherwise, the contents of the EIVR will be used. Both of these registers are set to the uninitialized vector number ($0F) by a reset operation.

3.5 CHANNEL PRIORITY REGISTER (CPR)

This register determines the priority level of a channel. When simultaneous operand transfer requests are pending in two or more channels, the highest priority channel will be serviced first. When two or more channels are at the same priority level, a round-robin service order is used to resolve simultaneous requests. The same priority scheme is used to resolve simultaneous pending interrupts during an interrupt acknowledge cycle; however, two independent circuits within the DMAC perform the priority arbitration for operand transfer requests and interrupt requests.

Channel Priority Register (CPR)



0 Not Used

CP Channel Priority
- 00 Priority 0 (Highest)
- 01 Priority 1
- 10 Priority 2
- 11 Priority 3 (Lowest)

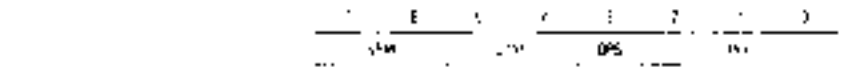
3.6 CONTROL REGISTERS

These four registers control the operation of the DMA channels by selecting various options for operand size, device characteristics, and address sequencing.

3.6.1 Device Control Register (DCR)

This register defines the characteristics of the device interface, including the device transfer request type, bus interface, port size, and the function of the PCL line.

Device Control Register (DCR)



XRM External Request Mode
- 00 Burst Transfer Mode
- 01 (Undefined, Reserved)
- 10 Cycle Steal Without Hold
- 11 Cycle Steal With Hold

DTYP Device Type
- 00 M68000 Compatible, Explicitly Addressed
- 01 M6800 Compatible, Explicitly Addressed
- 10 Device with ACK, Implicitly Addressed
- 11 Device with ACK and RDY, Implicitly Addressed

DPS Device Port Size
- 0 8-Bit Port
- 1 16-Bit Port

PCL Peripheral Control Line Function
- 000 Status Input (Level Read in CSR)
- 001 Status Input with Interrupt
- 010 Start Pulse Output
- 011 Abort Input
- 100 (MC68440 Reserved)
- 101, 110, 111 (Undefined, Reserved)

3.6.1.1 EXTERNAL REQUEST MODE. This field determines the method used by the device to request operand transfers. In the burst transfer mode, the $\overline{REQ}$ pin is a level sensitive input that generates transfer requests as long as it remains asserted. In the cycle steal modes, the $\overline{REQ}$ pin is an edge sensitive input; an asserted edge must occur to request each operand transfer. When the cycle steal with hold mode is selected, the DMAC will retain ownership of the bus for a programmable time period after an operand transfer occurs in order to wait for the next operand transfer request, thus reducing bus arbitration overload and response latency. For more detailed information, refer to **5.2.3.3 TRANSFER REQUEST GENERATION**

3.6.1.2 DEVICE TYPE. This field selects one of four available device transfer protocols. For M68000 type devices, the DMAC will use a dual address transfer protocol by running M68000 type bus cycles to transfer data to or from the device registers and a second bus cycle to complete the operand transfer from or to memory (note that this device type also allows memory-to-memory block transfers). If the device is of the M6800 type, dual address transfers are also used; but the DMAC uses synchronous M6800 type bus cycles when accessing the device. In the remaining two device protocols, the DMAC asserts the acknowledge signal to implicitly address the device during a single address transfer while it is explicitly addressing a memory location. For a device with acknowledge only, the DMAC will assume that the device will place data onto or accept data from the bus in order to meet the timing requirements of a minimum length bus cycle and will terminate all bus cycles when the memory asserts **DTACK**. For a device with acknowledge and ready, the PCL line is used as a ready signal that is similar to the DTACK signal. During a write to memory, the DMAC will not assert the data strobes to memory until the device has asserted ready. During a read from memory, the DMAC will not terminate the bus cycle until the device has asserted ready and memory has asserted DTACK. Note that when the DTYP field is programmed for device with acknowledge and ready, the PCL function field in the DCR and the PCT and PCS bits in the CSR will be ignored.

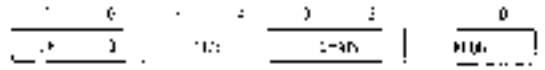
3.6.1.3 DEVICE PORT SIZE. This field indicates how wide the device port is, either 8 or 16 bits. Refer to Table 5-1 for legal port and operand size combinations.

3.6.1.4 PERIPHERAL CONTROL LINE. If the **DTYP** field is not programmed for device with acknowledge and ready or explicit M6800, this field will define the function of the PCL line. The level on the PCL line can always be read from the PCS bit in the channel status register and can also be used to generate an interrupt when a high-to-low transition occurs on the pin. If programmed as an abort input, it is a high-to-low edge-sensitive signal that will cause the DMAC to abort the channel. The PCL line may also be programmed as a start pulse output, in which case it is asserted for eight clock cycles when a channel operation is started.

3.6.2 Operation Control Register (OCR)

This register defines the type of operation that will be performed by a channel. The parameters programmed in this register are the direction of the transfer, the operand size to be used, whether the operation is chained, and the type of request generation to be used.

Operation Control Register (OCR)



DIR — Transfer Direction
- 0 — From Memory to Device
- 1 — From Device to Memory

0 — Not Used

SIZE — Operand Size
- 00 — Byte
- 01 — Word
- 10 — Long Word
- 11 — Byte, No Packing

CHAIN Chain Operation

00	Chaining Disabled
01	(Undefined, Reserved)
10	Array Chaining Enabled
11	Linked Array Chaining Enabled

REQG Request Generation

00	Internal Request at Limited Rate
01	Internal Request at Maximum Rate
10	External Request
11	Auto-Start, External Request

3.6.2.1 **TRANSFER DIRECTION** The field controls the direction of the block transfer. A zero indicates that the transfer is from memory to the device, a one indicates that the transfer is from the device to memory.

3.6.2.2 **OPERAND SIZE** This field defines the size of the data item to be transferred in response to each operand transfer request. The DMAC supports byte, word, and long word operand sizes; refer to Table 5-1 for legal port, memory, and operand size combinations.

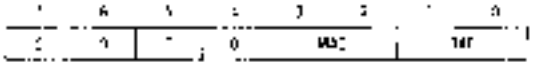
3.6.2.3 **CHAINING OPERATION.** This field is used to program a channel for a chained operation. When chaining is enabled, the DMAC starts an operation by reading a data block descriptor, consisting of a memory address and transfer count, from a table pointed to by the BFCR and BAR. Array chaining uses a linear, sequential table of descriptors and the BTCR is used to count the number of descriptors remaining in the table. Linked array chaining uses a linked list table structure, with each descriptor containing a pointer to the next descriptor.

3.6.2.4 **REQUEST GENERATION METHOD.** This field defines the method used by the DMAC to detect operand transfer requests. Two types of internal request generation methods are available, requests may be externally generated, or the first request of a channel operation can be internally generated with subsequent requests generated externally. If one of the internal request generation methods is programmed, the XRM field in the DCR is ignored. Refer to **3.8 GENERAL CONTROL REGISTER (GCR)** for more details on programming the time intervals for the internal request at a limited rate method of request generation.

3.6.3 Sequence Control Register (SCR)

This register controls the manner in which the memory and device address registers are modified during transfer operations. Each address register may be programmed to remain unchanged during an operand transfer, or to be incremented or decremented after each operand is successfully transferred. The value by which a register is incremented or decremented after each DMA bus cycle depends on the operand size and the device port size. For more details on address sequencing, refer to 5.2.3.2 **ADDRESS SEQUENCING.**

Sequence Control Register



0 Not Used

MAC Memory Address Count
- 00 MAR Does Not Count
- 01 MAR Counts Up
- 10 MAR Counts Down
- 11 (Undefined, Reserved)

DAC Device Address Count
- 00 DAR Does Not Count
- 01 DAR Counts Up
- 10 DAR Counts Down
- 11 (Undefined, Reserved)

3.5.4 Channel Control Register (CCR)

This register is used to control the operation of the channel. By writing to this register, a channel operation may be started, set to be continued, halted, and aborted. Also, the channel can be enabled to issue interrupts when an operation is completed.

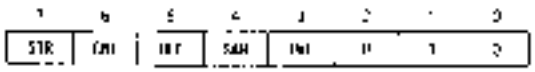
The STR and CNT bits are used to start the channel and to program it for continued operation. These bits can only be set by writing to them and are cleared internally by the DMAC. The STR bit will be cleared by the DMAC one clock after being set and the CNT bit is cleared when the MTCR is decremented to zero or by an error termination of the channel operation. The CNT bit must not be set at the beginning of a write cycle to the CCR to set the STR bit or an operation timing error will occur (the STR and CNT bits may be set by the same write cycle, however). Also, once the STR bit is set, none of the other control register may be modified or an operation timing error will occur. The CCR must be written with a byte operation when the STR bit is set or an operation timing error will occur (i.e., the SCR may not be written simultaneously with the CCR when the STR bit is set).

The HLT bit may be used to suspend the channel operation at any time by writing a one to it and the operation may then be continued by clearing HLT. If it is desired to stop the channel and not continue at a later time, the SAB bit may be set at any time to abort a channel operation.

The INT bit controls the generation of interrupts by the DMAC. If the INT bit is set and the COC, BTC, or PCT (with PCL programmed as a status with interrupt input) bits are set in the CSR, then the $\overline{IRQ}$ signal will be asserted. If the PCL line is not programmed as an interrupt input, the PCT bit will be set by a high-to-low transition, but will not generate an interrupt.

If a one is written to the STR bit position, the one will not be stored, but the CSR will reflect the status of the channel after the start operation is executed; the ACT bit in the CSR will be set if no error occurs. The STR bit always reads as zero, and writing a zero to this position will have no effect on the channel operation.

Channel Control Register



STR Start Operation
- 0 No Effect
- 1 Start Channel

CNT Continue Operation
- 0 No Continuation is Pending
- 1 Continue Operation at End of Block

HLT Halt Operation
- 0 Normal Channel Operation
- 1 Halt Channel Operation

SAB Software Abort
- 0 Normal Channel Operation
- 1 Abort Channel Operation

INT Interrupt Enable
- 0 Channel Will Not Generate Interrupts
- 1 Channel May Generate Interrupts

0 Not Used

3.7 STATUS REGISTER

These two registers reflect the status of a channel operation. If an error is indicated by the channel status register (CSR), a more detailed description of the first error that occurred during an operation will be encoded in the lower five bits of the channel error register (CER).

3.7.1 Channel Status Register (CSR)

This register reflects the status of a channel operation. The status information supplied indicates if the channel is active or complete, whether an error has occurred, and whether or not the PCL signal has been asserted during the operation.

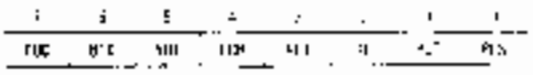
The ACT bit will be set when the channel is started by writing to the STR bit in the channel control register and will be cleared when the channel operation terminates. The COC, BTC, and NDT bits indicate the manner by which an operation terminated. COC is set when the channel is terminated, either successfully or for an error. BTC is set when the MTCR is decremented to zero and the CNT bit is set in the channel control register to indicate that a continued operation is complete and the DMAC is now transferring the next block. The NDT bit is set when the device terminates an operation by asserting the $\overline{\text{DONE}}$ signal. The ERR bit will be set any time that an error has occurred in the channel and indicates that the non-zero value in the channel error register reflects the type of the first error that has occurred since the channel was last started.

The PCT and PCS bits are associated with the function of the PCL signal. The PCT bit will be set anytime that a high-to-low transition occurs on the PCL line, regardless of its programmed function. The PCS bit reflects the instantaneous level of the PCL line; if this bit is set, the PCL line is at the high-voltage level and if this bit is clear, the PCL line is at the low voltage level.

Once they have been set by the DMAC, the COC, BTC, NDT, ERR, and PCT bits must be cleared either by a reset or by writing to the CSR. In order to clear a status bit, a one (1) is written to the bit position corresponding to that status bit. If a zero (0) is written to a status bit position, that bit will be unaffected; also, the ACT and PCS status bits are unaffected by all write operations to the CSR. In order to start channel operation, the COC, NDT, and ERR bits must be cleared; in order to perform a continued operation,

the BTC bit must be cleared before the CNT bit is set in the channel control register. Note that when the ERR bit is cleared, either by reset or by writing to it, the channel error register will also be cleared.

Channel Status Register



CGC Channel Operation Complete
- 0 Channel Operation Not Complete
- 1 Channel Operation Has Completed

BTC Block Transfer Complete
- 0 Block Transfer Not Complete
- 1 Continued Block Transfer Has Completed

NDT Normal Device Termination
- 0 Operation Not Terminated by Device
- 1 Device Terminated Operation with DONE

ERR Error
- 0 No Error
- 1 Error Has Occurred

ACT Channel Active
- 0 Channel Not Active
- 1 Channel Is Active

0 Not Used

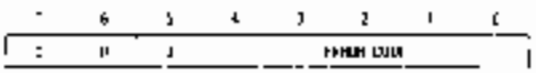
PCT PCL Transition
- 0 No PCL Transition Has Occurred
- 1 High-to-Low PCL Transition Occurred

PCS PCL State
- 0 PCL Line is Low
- 1 PCL Line is High

3.7.2 Channel Error Register (CER)

This register indicates the cause of the first error that has occurred since the ERR bit in the channel status register was last cleared. When the ERR bit in the CSR is set, bits 0-4 indicate the type of error that occurred. Refer to **5.6.3 Error Termination** for more information on the cause of the errors reported by the CER.

Channel Error Register



0 Not Used

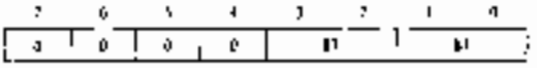
Error	00000	No Error
Code	00001	Configuration Error
	00010	Operation Timing Error
	00011	(Undefined, Reserved)
	001rr	Address Error
	010rr	Bus Error
	011rr	Count Error
	10000	External Abort
	10001	Software Abort
	10010	
	•	
	•	(Undefined, Reserved)
	•	
	11111	

rr Register Code
- 00 (MC68440 Reserved)
- 01 MAR or MTCR
- 10 DAR
- 11 BAR or BTCH

3.8 GENERAL CONTROL REGISTER (GCR)

This is a global register that may affect the operation of all channels. If a channel is programmed for internal timing, the GCR is used to determine the time constants for the limited-rate timing. Refer to **5.2.3.2 Internal, Limited Rate** for more details on limited rate auto-request operation. If a channel is programmed for cycle steal with hold request generation, the DMAC will retain ownership of the bus until the end of the next full sample period (as defined in 5.2.3.3.2) after an operand transfer is completed. Refer to **5.4.4 External Request Recognition** for more details on cycle steal with hold operation.

General Control Register



0 Not Used

BT Burst Transfer Time
- 00 16 Clocks
- 01 32 Clocks
- 10 64 Clocks
- 11 128 Clocks

BR Bandwidth Available to DMAC
- 00 50.00%
- 01 25.00%
- 10 12.50%
- 11 6.25%

3.9 REGISTER SUMMARY

A summary of the bit definitions for each of the channel registers along with the address of each register within the DMAC register memory map is provided in Figure 3-2.

3.10 RESET OPERATION RESULTS

When the DMAC is reset, either during a system power up sequence or to re-initialize the DMAC, many of the registers will be affected and will be set to known values. Table 3-1 shows the hexadecimal value that will be placed in each register by a reset operation.

Table 3-1. Reset Operation Results

Register	Value	Comments
MARc	XXXXXXXX	
DARc	XXXXXXXX	
BARc	XXXXXXXX	
MFCRc	X	Not Affected
DFCRc	X	
BFCRc	X	
MTCRc	XXXX	
BRCRc	XXXX	
NIVR	0F	Uninitialized Vector
EIVR	0F	Uninitialized Vector
CPRc	0C	
DCRc	0C	
DDRc	40	
SCRc	0C	
CCRc	0C	Channel Not Active,
		Interrupts Disabled
CSRc	30 or 01	(Depending on the level of RQ in)
CERc	00	No Errors
CPR	00	

X = Indicates an unknown value or the previous value of the register.
c = the channel number (c = 0, 1, 2, or 3)

SECTION 4
BUS OPERATION

This section describes the bus operation of the DMAC in all operating modes. All bus operations are described in relation to the DMAC CLK signal, with all signal activity taking place during a certain period of the CLK input. The terminology and conventions used are identical to the bus description for the MC68000 16-bit microprocessor. Functional timing diagrams are included to assist in the definition of signal timings; however, these diagrams are not intended as parametric timing definitions. For detailed timing relationships between various signals, refer to **SECTION 6 ELECTRICAL SPECIFICATIONS**.

The term "synchronization delay" will be used repeatedly when discussing bus operation. This delay is the time period required for the DMAC to sample an external asynchronous input signal, determine whether it is high or low, and synchronize the input to the DMAC internal clocks. Figure 4-1 shows the relationship between the CLK signal, an external input, and its associated internal signal that is typical for all of the asynchronous inputs. Note that some input signals are synchronized on the rising edge of the CLK signal as shown, while others are synchronized on the falling edge of the CLK signal (in which case the EXT and INT signals below would be shifted to the left or right by one-half clock cycle). Refer to Table 2-1 for the synchronizing edge for each input signal.

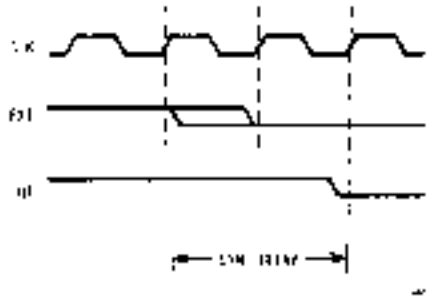


Figure 4-1. Relationship Between External and Internal Signals

4.1 RESET OPERATION

The DMAC can be reset to the IDLE mode by an external device at any time by asserting the reset encoding on $\overline{BEC0}$-$\overline{BEC2}$. Two types of reset operations may be performed by external hardware, a power-on reset and a system reset after power has been stable for a long period of time. During power-on reset, the reset signal must be asserted to the DMAC for at least 100 milliseconds after V_{CC} has reached its nominal operating level, as shown in Figure 4-2. After power has been stable for a long period of time, reset must be asserted for at least ten clock cycles to reset the DMAC.

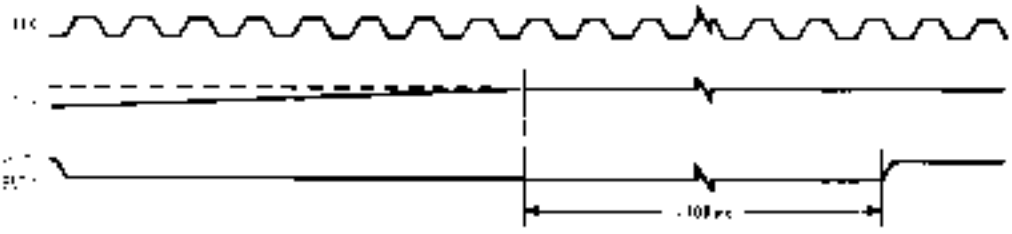


Figure 4-2. Reset Operation Timing Diagram

4.2 MPU MODE OPERATION

In the MPU mode, the DMAC acts as a peripheral slave to another bus master. When the $\overline{CS}$ and $\overline{UDS}$ or $\overline{LDS}$ signals are asserted, the DMAC will enter the MPU mode from the idle mode (if the DMAC is in the DMA mode and $\overline{CS}$ is asserted, an error will be signaled). During MPU mode operations, the DMAC will accept data from or place data on the data bus according to the level of the R/$\overline{W}$ pin. The data transferred will either be placed into or come from the internal register(s) that is selected by the encoding on A1-A7, $\overline{UDS}$, and $\overline{LDS}$. Once the data transfer has been completed, the DMAC will assert $\overline{DTACK}$ to terminate the bus cycle and return to the IDLE mode.

During MPU mode operations, the DMAC will synchronize input signals to the CLK input and all outputs will occur in relation to the CLK signal; however, the CLK does not need to be synchronous with the bus master's clock. Thus, the DMAC will appear to be completely asynchronous to the bus master unless they both use the same clock signal. In the functional diagrams showing MPU operations, the bus master is assumed to be an MC68000 or MC68010 with a clock signal identical to the DMAC CLK signal.

4.2.1 MPU Read Cycles

During MPU read cycles, the DMAC will place data on the data bus and assert DTACK to indicate to the bus master that the data from the register(s) selected is available to it. Figure 4-3 shows the functional timing for a word read cycle. The timing for even and odd byte MPU reads is identical with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte.

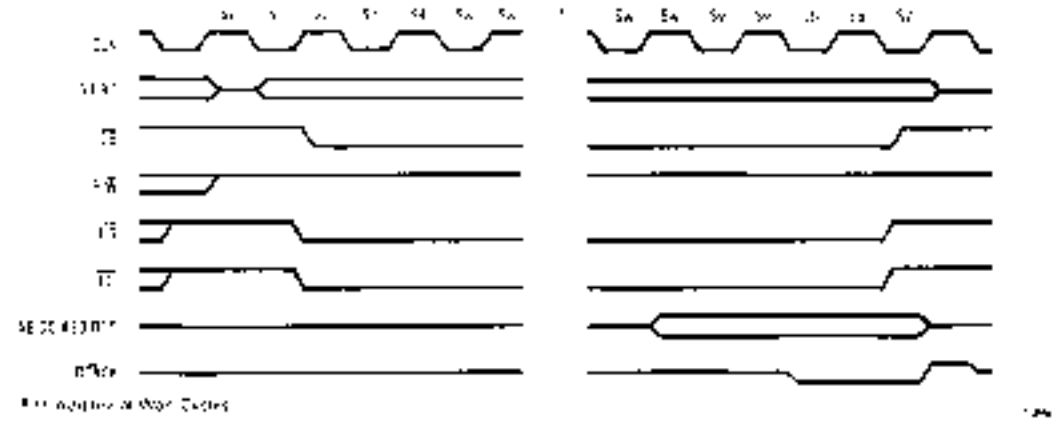


Figure 4-3. MPU Read Cycle Timing Diagram

The DMAC starts an MPU read operation when $\overline{CS}$ is asserted and synchronized internally with the R/$\overline{W}$ line high. The DMAC responds to the $\overline{CS}$ by decoding A1-A7, $\overline{UDS}$, and $\overline{LDS}$ and placing the data from the selected register(s) on the appropriate address/data pins, driving $\overline{DDIR}$ high, asserting $\overline{DBEN}$ and asserting $\overline{DTACK}$. The DMAC will then wait until $\overline{CS}$ is negated and three state the address data pins, $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

4.2.2 MPU Write Cycles

During MPU write cycles, the DMAC accepts data from the data bus and places it into the register(s) selected by the bus master. Figure 4-4 shows the functional timing for a word write cycle. The timing for even and odd byte MPU writes is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte.

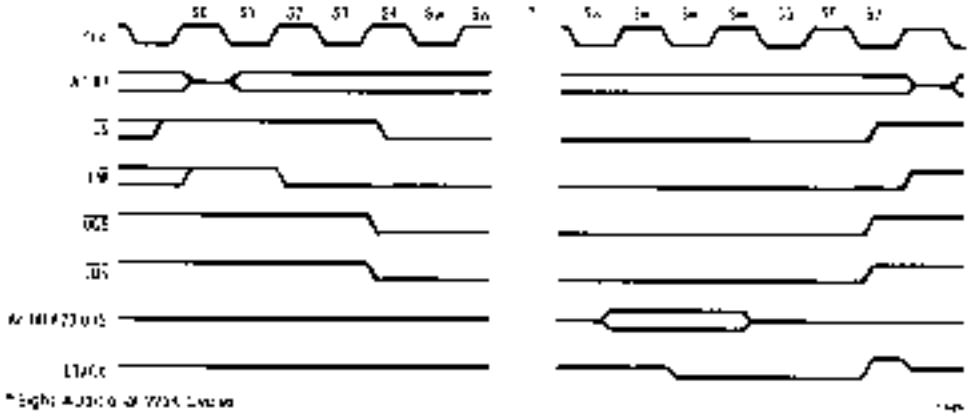


Figure 4-4. MPU Write Cycle Timing Diagram

The DMAC starts an MPU write operation when $\overline{CS}$ is asserted and synchronized internally with the R/$\overline{W}$ line low. Note that input data should be valid at the data buffers when $\overline{CS}$ is asserted; thus, the $\overline{CS}$ equation should include the data strobes. The DMAC responds to the $\overline{CS}$ by decoding A1-A7, $\overline{UDS}$, and $\overline{LDS}$, driving $\overline{DDIR}$ low and asserting $\overline{DBEN}$. Data is then taken from the appropriate address/data pins and placed into the selected register(s) and $\overline{DTACK}$ is asserted. The bus master should hold the data valid until $\overline{CS}$ is negated, at which time the DMAC will three state $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

4.2.3 Interrupt Acknowledge Cycles

A special case of the MPU mode of operation is the interrupt acknowledge cycle during which the system processor is responding to an interrupt request from the DMAC. The timing of an interrupt acknowledge cycle is identical to an odd byte read cycle via $\overline{CS}$, except that it is started by the assertion of the $\overline{IACK}$ signal rather than $\overline{CS}$. $\overline{IACK}$ and $\overline{CS}$ are mutually exclusive signals and must not be asserted at the same time. Also, $\overline{IACK}$ should not be asserted while the DMAC is acting as a bus master, or an address error will occur for the channel that is being serviced when the $\overline{IACK}$ signal is asserted. Figure 4-5 shows the functional timing for an interrupt acknowledge cycle.

The DMAC starts an interrupt acknowledge operation when $\overline{IACK}$ is asserted and synchronized internally. The DMAC responds to the IACK by placing a vector number on A8-D0 A15/D7, driving $\overline{DDIR}$ high and asserting $\overline{DBEN}$ and $\overline{DTACK}$. The A1-A7, $\overline{UDS}$, and $\overline{LDS}$ signals are all ignored during an interrupt acknowledge cycle. The vector number placed on the bus will be taken from either the NIVR or EIVR of the highest priority channel with an interrupt pending. The vector number will remain valid until the $\overline{IACK}$ signal is negated, at which time the DMAC will three state the address/data bus and negate $\overline{DDIR}$, $\overline{DBEN}$, and $\overline{DTACK}$.

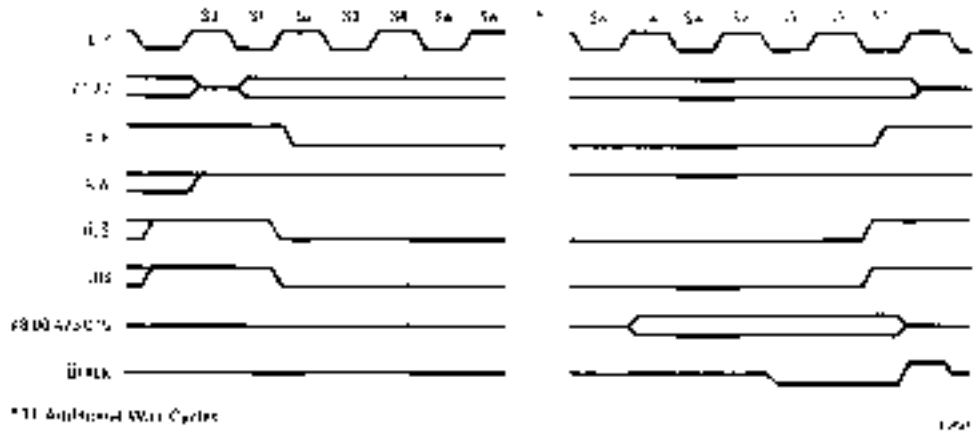


Figure 4-5. Interrupt Acknowledge Cycle Timing Diagram

4.3 DMA MODE OPERATION

In the DMA mode of operation the DMAC is acting as a bus master and performs data transfer operations between a device and memory in response to operand transfer requests. There are three classes of DMA bus cycles that may be run by the DMAC: M68000 cycles that transfer data between a device or memory and the DMAC internal holding register, M6800 cycles that transfer data between an M6800 family device and the DMAC internal holding register, and cycles that transfer data directly between a device and memory without going through the DMAC. The first two classes of cycles are referred to as dual address cycles, since one operand transfer takes place in two bus cycles, one where a device is addressed and one where memory is addressed. The second class of cycle is referred to as a single address transfer since the operand transfer takes place in one bus cycle where only the memory is explicitly addressed. Within each class of DMA bus cycles, the DMAC may execute a read or a write operation. The following paragraphs describe the six DMA cycle types according to the signal transitions during each clock cycle of a bus operation.

4.3.1 Dual Address Transfers

Dual address transfer signal timing closely resembles the signal timing of an M68000 processor. The DMAC starts a bus cycle by presenting function code, address, direction, and data size information to an external device and then waits for the addressed device to present or accept data and terminate the bus cycle. Since the bus structure used is asynchronous, the memory or device access time can be dynamically changed according to the needs of the current bus cycle. The data that is transferred during a dual address operation is either read from the data bus into the DMAC internal holding FIFO or written from that FIFO to the data bus. All cycle terminating options of the M68000 bus are supported, such as halt, cycle retry, or bus error. Additionally, the M6800 synchronous bus protocol is supported for device accesses.

Note that several combinations of device and operand sizes may be used with dual address transfers, including combinations where the memory and device width are different. When an 8-bit device is used, the device may be placed on the most significant (even address) or least significant (odd address) half of the data bus.

4.3.1.1 DUAL ADDRESS READ, M68000 TYPE. During this type of a DMA cycle, the DMAC will read data from a device or memory into an internal holding FIFO, the MAR, MTCR, or BAR. Figure 4-6 shows the functional timing for a word read. The timing for an even or odd byte read is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte. The signal protocol is as follows:

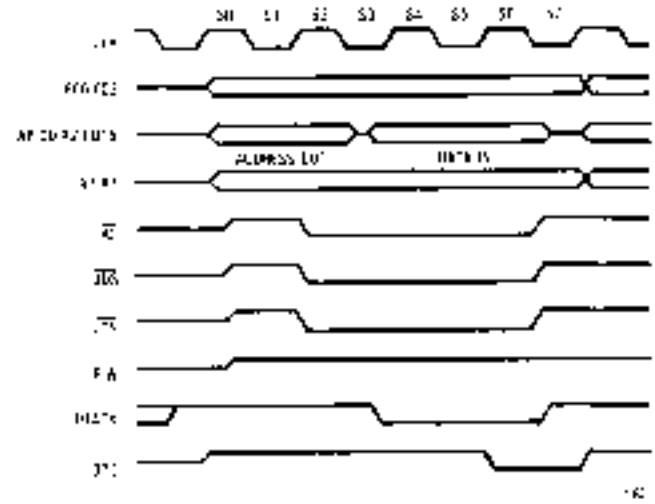


Figure 4-6. M68000 Type Dual Address Read Cycle Timing Diagram

S0 The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the function code and address registers (the MFCR and MAR, DFCR and DAR, or BFCR and BAR) of the current channel, $\overline{UAS}$ is asserted to enable the external address latches and R/$\overline{W}$ is driven high.

S1 No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. $\overline{UDS}$ and/or $\overline{LDS}$ will also assert to indicate that data should be placed on the bus and $\overline{DDIR}$ will be driven low to prepare the external data buffers/demultiplexers to drive the address/data pins when enabled. If the current bus cycle is to a device, $\overline{ACK}$ will also be asserted to the device and $\overline{DONE}$ will be asserted if this is the last transfer to the device for the current data block.

S3 The address/data pins are three stated in preparation to receive input data and $\overline{DBEN}$ is asserted to enable the data buffer/demultiplexer. The DMAC also starts sampling the $\overline{DTACK}$ and BEC0-BEC2 pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC may continue into S4 immediately. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, two wait cycles will be inserted between S3 and S4. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC will insert wait cycles after S3 until a termination signal is asserted, and then proceed to S4.

S4 No signals change during S4. The memory or device access is taking place at this time and data is allowed to propagate through external buffers, etc. The DMAC is also internally synchronizing the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ signals.

S5 No signals change during S5. The memory or device access terminates and the DMAC completes synchronization of the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins.

S6 Data is sampled during this period, with the data bus value latched at the end of S6. Thus, valid data must be present at the external data buffers/demultiplexers such that it may propagate through those devices and be valid at the DMAC address/data pins within the setup time to the falling edge of S6. For byte read operations, the data lines for the byte not being read will be ignored. $\overline{DTC}$ is asserted to indicate that the bus cycle has been successfully terminated. $\overline{DTC}$ will not be asserted if a $\overline{BEC0}$-$\overline{BEC2}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 $\overline{AS}$ is negated to indicate that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ and $\overline{DBEN}$ will also negate to disable external data buffers/demultiplexers. The memory or peripheral device that was addressed during the cycle may then negate $\overline{DTACK}$ and/or $\overline{BEC0}$-$\overline{BEC2}$.

S7 + 1 This may be S0 of the following cycle, if the DMAC is ready to immediately execute another bus cycle. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three-stated. $\overline{DDIR}$ is negated to turn the data buffers/demultiplexers away from the DMAC. $\overline{ACK}$, $\overline{DTC}$, and $\overline{DONE}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed.

4.3.1.2 DUAL ADDRESS WRITE, M68000 TYPE. During this type of a DMA cycle, the DMAC will write data to a device or memory from an internal holding register. Figure 4-7 shows the functional timing for a word write on a 16-bit bus. The timing for an even or odd byte write is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper byte. The signal protocol is as follows:

S0 The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23-D15 pins with values from the function code and address registers (either the MFCR and MAR or DFCR and DAR) of the current channel and $\overline{UAS}$ is asserted to enable the external address latches. $\overline{DDIR}$ will be driven high to prepare the external data buffers/demultiplexers to drive the external data bus when enabled.

S1 No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. R/$\overline{W}$ is also driven low to indicate that this cycle is a write.

S3 The address/data pins change from address to data and $\overline{DBEN}$ is asserted to allow data to propagate through the external data buffers/demultiplexers. For byte write operations, the value of the data lines for the byte not being written is undefined. If the cycle is accessing a device, $\overline{ACK}$ will be asserted at this time and $\overline{DONE}$ will be asserted if this is the last transfer to the device for the current block operation. The DMAC also begins to sample the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC will 'skip' S5 and S7 and proceed directly from S5 to S8. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC will execute S5 and S7.

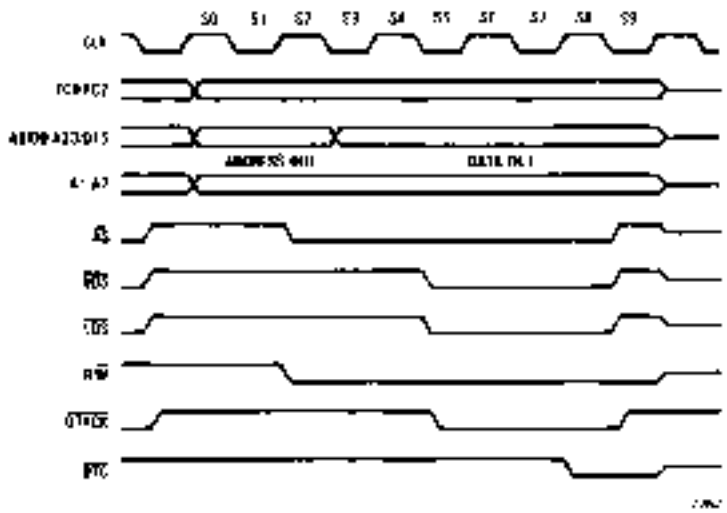


Figure 4-7. M68000 Type Dual Address Write Cycle Timing Diagram

S4 No signals change during S4. Data from the DMAC is allowed to propagate to the device or memory through external buffers, etc.

S5 $\overline{UDS}$ and/or $\overline{LDS}$ is asserted at this time to indicate that valid data is present on the bus and may be latched. The DMAC continues to sample the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S5, then the DMAC will continue into S6 immediately. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S5, two wait cycles will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S5, then the DMAC will insert wait cycles after S5 until a termination signal is asserted and then proceed to S6.

S6 No signals change during S6. The DMAC is internally synchronizing the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins.

S7 No signals change during S7. The DMAC continues to synchronize the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins.

S8 $\overline{DTC}$ is asserted to indicate that the bus cycle has been successfully terminated. $\overline{DTC}$ will not be asserted if a $\overline{BEC0}$-$\overline{BEC2}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S9 $\overline{AS}$ is negated to indicate that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ are negated to disable the memory or device data buffers. The memory or device that was addressed during the cycle may then negate $\overline{DTACK}$ and/or $\overline{BEC0}$-$\overline{BEC2}$.

S9 + 1 This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC1 and A1-A7 pins will all be changed on their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. $\overline{DREN}$ is negated and $\overline{DDIR}$ and R/$\overline{W}$ are driven high to prepare the data bus for the next cycle. ACK, DTC, and $\overline{DONE}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed.

4.3.1.3 DUAL ADDRESS READ, M6800 TYPE DEVICE During this type of a DMA cycle, the DMAC will read data from a device into an internal holding register, using an M6800 type synchronous bus cycle. Figure 4-8 shows the functional timing for a read from an M6800 type device. Note that $\overline{LDS}$ is asserted and $\overline{UDS}$ is negated during the cycle shown, since M6800 peripherals are eight bits wide and will normally be placed on the lower half of the data bus. The DMAC also will allow an M6800 type device to be accessed on the upper half of the data bus, and will allow word accesses using the M6800 synchronous bus interface. The signal protocol is as follows.

S0 1 The DMAC samples the E clock input (the PCL pin) at the end of the state prior to the start of the M6800 bus cycle, and stores the level for later use. Throughout the remainder of this cycle, the E clock will be sampled on each rising edge of the clock and the level that is latched will be compared with the level latched on the previous rising edge of the clock. If the two sampled levels are different, then the DMAC will take action appropriate for the type of transition (high to low or low to high) that has been detected on the E input.

S0 The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the DFCR and DAR of the current channel. $\overline{UAS}$ is asserted to enable the external address latches and R/$\overline{W}$ is driven high.

S1 No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and the bus is in use. $\overline{UDS}$ and/or $\overline{LDS}$ are asserted to indicate that data should be placed on the bus and $\overline{DDIR}$ is driven low to prepare the external data buffers/demultiplexers to drive the address/data pins when enabled. $\overline{DONE}$ will also be asserted if this is the last operand transfer for the block operation. If a high-to-low transition on E was detected during S1, then $\overline{ACK}$ will be asserted as a valid memory address (VMA) strobe to indicate the start of an M6800 peripheral access; otherwise, the DMAC will insert wait cycles after S2 until a high to low transition of E is detected before asserting $\overline{ACK}$.

S3 The address/data pins are three stated in preparation to receive input data and $\overline{DREN}$ is asserted to enable the data buffer demultiplexer.

Sw After the end of S3 and before the start of S4, the DMAC will insert wait cycles (in two wait states each) until $\overline{ACK}$ is asserted (following a high-to-low transition on E) and one low-to-high (followed by one high-to-low transition occurs on the E clock. No output signals change during this synchronization period, and the M6800 device access is performed during the E clock high period.

S4 No signals change during S4. The DMAC is internally synchronizing the second high to low transition of the E clock.

S5 No signals change during S5. The DMAC continues to synchronize the second high-to-low transition of the E clock.

S6 Data is sampled during this period, with the data bus value latched at the end of S6. For byte read operations, the data lines for the byte not being read will be ignored. $\overline{DTC}$ is asserted to indicate that the bus cycle has been successfully terminated.

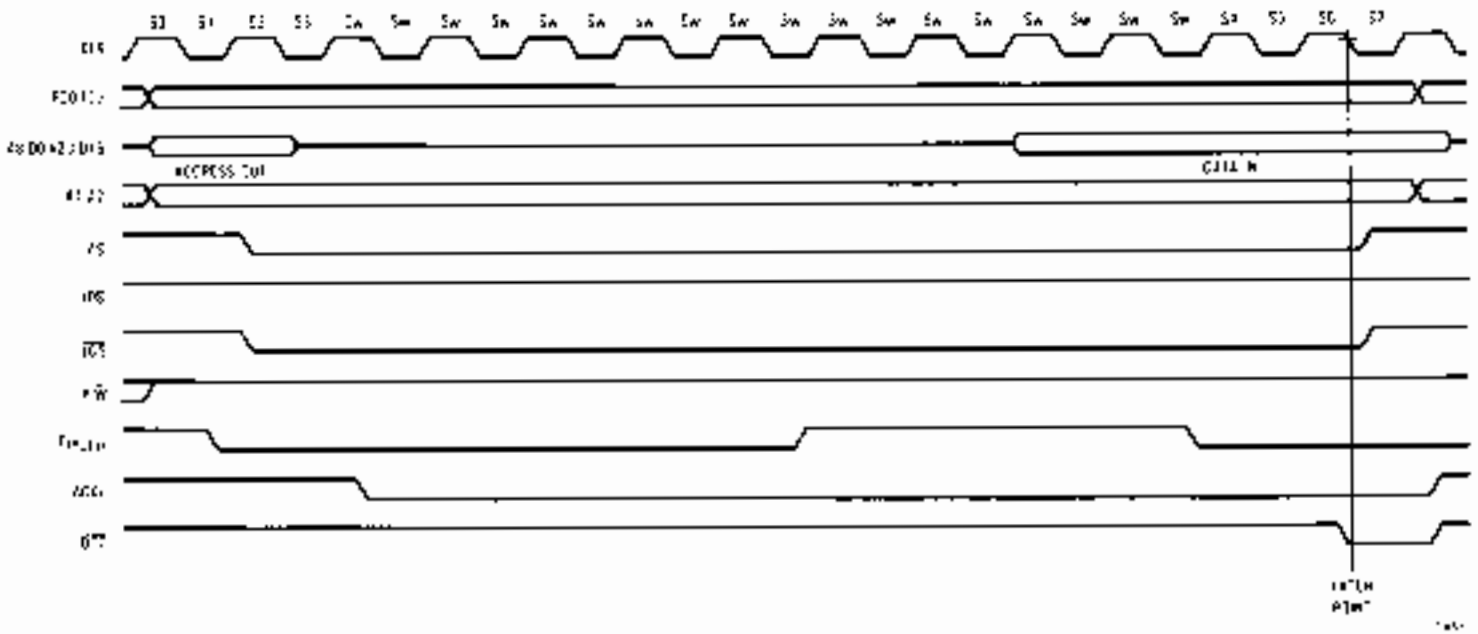


Figure 4-8 M6800 Type Read Cycle Timing Diagram

S7 — $\overline{AS}$ is negated to indicate that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ and $\overline{DBEN}$ will also negate to disable external data buffers/demultiplexers.

S7 → 1 — This may be S0 of the following cycle if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three-stated. $\overline{DDIR}$ is negated to turn the data buffers/demultiplexers away from the DMAC. $\overline{ACK}$ (VMA), $\overline{DTC}$ and $\overline{DONE}$ (if it was asserted) are all negated to indicate to the peripheral that the cycle has completed.

NOTE

As indicated in Figure 4-8, the DMAC does not latch data until after the E clock is low at the end of the cycle. Since M6800 peripherals remove data from the bus when E goes low, a latch must be provided to hold data valid at the DMAC until $\overline{AS}$ goes high.

4.3.1.4 DUAL ADDRESS WRITE, M6800 TYPE DEVICE. During this type of a DMA cycle, the DMAC will write data to a device from an internal holding register, using an M6800 type synchronous bus cycle. Figure 4-9 shows the functional timing for a write to an M6800 type device. Note that $\overline{LDS}$ is asserted and $\overline{UDS}$ is negated during the cycle shown, since M6800 peripherals are eight bits wide and will normally be placed on the lower half of the data bus. The DMAC also will allow an M6800 type device to be accessed on the upper half of the data bus, and will allow word accesses using the M6800 synchronous bus interface. The signal protocol is as follows:

S0-1 — The DMAC samples the E clock input (the PCI line) at the end of the state prior to the start of the M6800 bus cycle, and stores the level for later use. Throughout the remainder of this cycle, the E clock will be sampled on each rising edge of the clock, and the level that is latched will be compared with the level latched on the previous rising edge of the clock. If the two sampled levels are different, then the DMAC will take action appropriate for the type of transition (high-to-low or low-to-high) that has been detected on the E input.

S0 — The DMAC drives the FC0-FC2, A1-A7, and AR-D0-A23/D15 pins with values from the DFCR and DAR of the current channel and $\overline{UAS}$ is asserted to enable the external address latches. $\overline{DDIR}$ will be driven high to prepare the external data buffers/demultiplexers to drive the external data bus when enabled.

S1 — No signals change and the upper address information is allowed to propagate through the external address latches.

S2 — $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and the bus is in use. R/$\overline{W}$ is also driven low to indicate that this cycle is a write.

S3 — The address/data pins change from address to data and $\overline{DBEN}$ is asserted to allow data to propagate through the external data buffers/demultiplexers. For byte write operations, the value of the data lines for the byte not being written is undefined. $\overline{DONE}$ will be asserted if this is the last transfer to the device for the current block operation.

S4 — No signals change during S4. Data from the DMAC is propagated to the device or memory through external buffers, etc. If a high-to-low transition on E was detected during S1 or

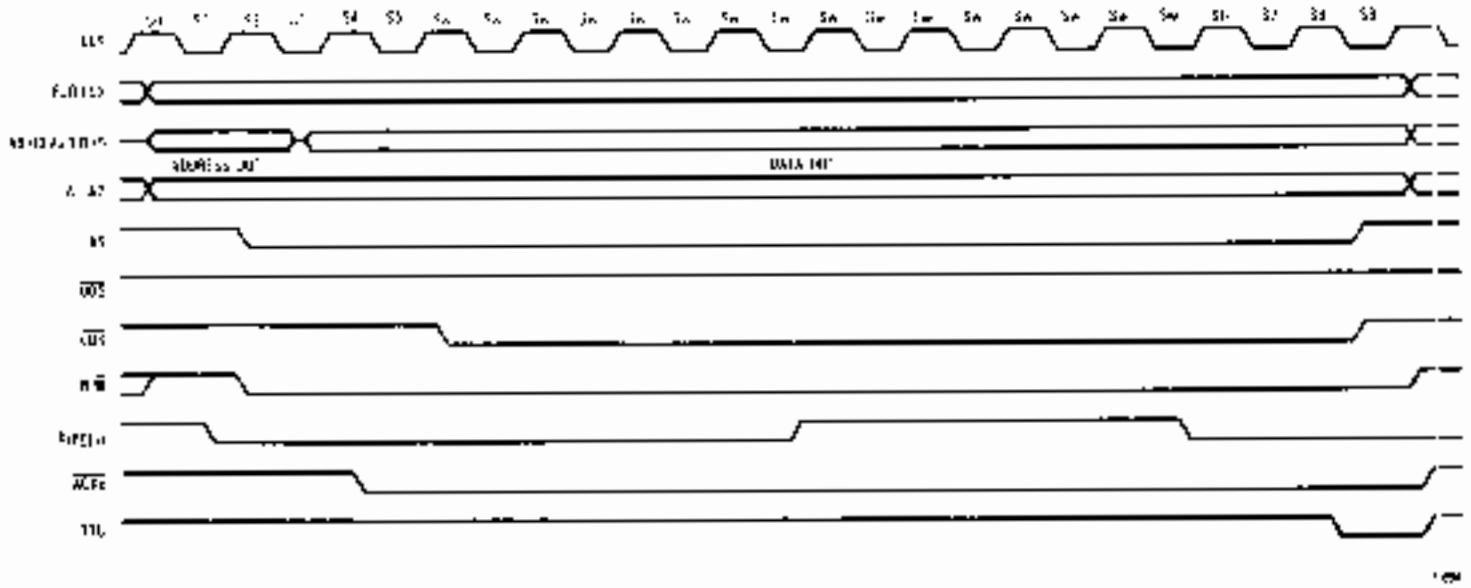


Figure 4-9 M6800 Type Write Cycle Timing Diagram

S3, then $\overline{ACK}$ will be asserted as a valid memory address (VMA) strobe to indicate the start of an M6800 peripheral access; otherwise, the DMAC will insert wait cycles after S0 until a high-to-low transition of E is detected before asserting $\overline{ACK}$.

S5 — $\overline{UDS}$ and/or $\overline{LDS}$ is asserted to indicate that valid data is present on the bus.

Sw — After the end of S5 and before the start of S6, the DMAC will insert wait cycles in two wait states each until $\overline{ACK}$ is asserted (following a high-to-low transition on E) and one low-to-high followed by one high-to-low transition occurs on the E clock. No output signals change during this synchronization period, and the M6800 device access is performed during the E clock high period, with data latched by the device on the falling edge of E.

S6 — No signals change during S6. The DMAC is internally synchronizing the second high-to-low transition of the E clock.

S7 — No signals change during S7. The DMAC continues to synchronize the second high-to-low transition of the E clock.

S8 — $\overline{DTC}$ is asserted to indicate that the bus cycle has been successfully terminated.

S9 — $\overline{AS}$, $\overline{UDS}$, and/or $\overline{LDS}$ are negated to indicate that the cycle has terminated.

S9 + 1 — This may be S0 of the following cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 lines will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory or peripheral location. If the DMAC is not ready to start a bus cycle, these lines will be three-stated. DBEN is negated, and $\overline{DDIR}$ and R/$\overline{W}$ are driven high to prepare the data bus for the next cycle. $\overline{ACK}$ (VMA), $\overline{DTC}$, and $\overline{DONE}$ (if it was asserted) are all negated to indicate to the peripheral that the cycle has completed.

4.3.2 Single Address Transfers

Single address transfer signal timing is similar to the timing for dual address transfers, except data flows directly between a device and memory without passing through the DMAC. There are two types of single address transfer protocols that are used to support different device types. A device with acknowledge uses a two signal handshake ($\overline{REQ}$ and $\overline{ACK}$) to perform transfers. A device with acknowledge and ready uses a three signal handshake ($\overline{REQ}$, $\overline{ACK}$, and PCL) to perform transfers.

When the DMAC is executing a single address transfer bus cycle, it will drive the address bus and the control signals for both the address and data bus, but will not put data onto or read data from, the data bus. The device control signals $\overline{ACK}$, PCL (for a device with ready), and $\overline{DTC}$ are used to synchronize the device to the memory so that the data can be transferred directly between them. In this way, the combination of the DMAC and the device appears to execute bus cycles in the same manner as an MPU as far as the memory is concerned.

4.3.2.1 SINGLE ADDRESS READ. During this type of DMA cycle, the DMAC will control the transfer of data from memory to a device. Figure 4-10 shows the functional timing for an even byte read to an 8-bit device. The timing for a word or odd byte read to an 8 bit or 16 bit device is identical, with the encoding of $\overline{UDS}$ and $\overline{LDS}$ selecting the proper bytes(s) and $\overline{HIBYTE}$ and R/$\overline{W}$ controlling the flow of data from D8-D15 to D0-D7 if necessary (which will only occur for the case shown in Figure 4-10).

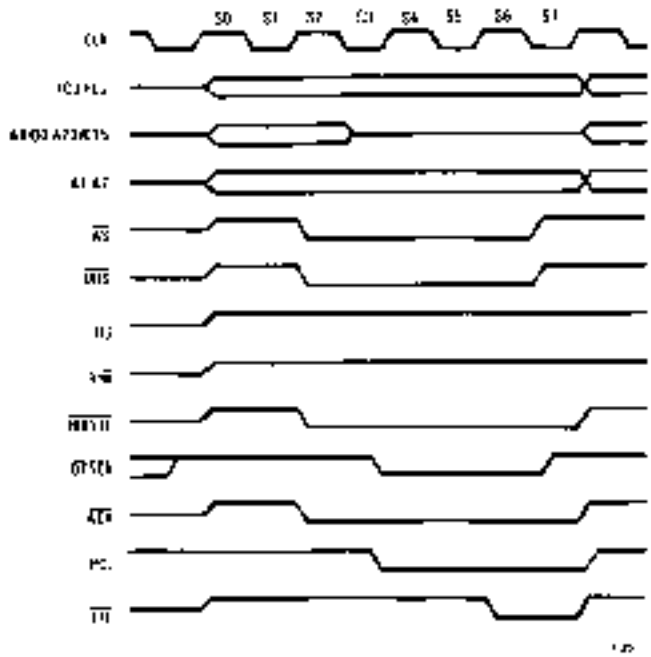


Figure 4-10. Single Address Read Cycle Timing Diagram

S0 The DMAC drives the FC0-FC2, A1-A7, and AB:D0-A23:D15 pins with values from the MFCR and MAR of the current channel, $\overline{UAS}$ is asserted to enable the external address latches, and R/$\overline{W}$ is driven high. $\overline{DBEN}$ will be driven high and remain high through the end of the cycle so that the data buffers/multiplexers will not interfere with the data transfer.

S1 No signals change and the upper address information is allowed to propagate through the external address latches.

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. $\overline{UDS}$ and/or $\overline{LDS}$ will assert to indicate that the memory should place data on the bus. $\overline{ACK}$ is also asserted to the device to indicate that the cycle is beginning, and $\overline{DONE}$ will be asserted if this is the last transfer for the current block operation. If the device port size is eight bits and the address is even, $\overline{HIBYTE}$ will be asserted to cause the memory data from D8-D15 to be driven onto D0-D7 through an external buffer. In this case, $\overline{UDS}$ will be asserted and $\overline{LDS}$ will be negated.

S3 The address/data pins are three-stated and the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins will be sampled to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC may continue into S4 immediately. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, two wait cycles will be inserted between S3 and S4. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC will insert wait cycles after S3 until a termination signal is asserted and then proceed to S4.

If the device protocol for the selected device is acknowledge with ready, the PCL input will also be sampled as a ready signal from the device. In order for the DMAC to terminate the bus cycle, the memory must present a termination signal and the device must drive PCL low. In this way, the device can monitor the DTACK signal from the memory to determine when data is available, latch it, and terminate the bus cycle when access time requirements have been met. If either the memory or device has not asserted its termination signal by the end of S3, wait cycles will be inserted until both termination conditions are met and then the DMAC will proceed to S4.

S4 — No signals change during S4. The memory may be placing data on the bus at this time. The DMAC is internally synchronizing the DTACK and BEC0-BEC2 pins in both protocols and the PCL pin for the device with ready protocol.

S5 — No signals change during S5, but data should be valid at the device port by the end of S5. The DMAC continues to synchronize the DTACK, BEC0-BEC2, and PCL signals.

S6 — DTC is asserted to the device at this time to indicate that the cycle is being terminated normally and that valid data may be latched from the bus. DTC will not be asserted if a BEC0-BEC2 coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S7 — AS is negated to indicate to the memory that the cycle has terminated. UDS and/or LDS will also negate in this time and the memory may respond by negating DTACK and/or BEC0-BEC2.

S7 / 1 — This may be S0 of the next bus cycle, if the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DMAC is not ready to start a bus cycle, these lines will be three-stated. ACK, DTC, DONE, and HIBYTE are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed, and the device may negate the ready (PCL) input if the acknowledge with ready protocol was used.

4.3.2.2 SINGLE ADDRESS WRITE. During this type of a DMA cycle, the DMAC will control the transfer of data from a device to memory. Figure 4-11 shows the functional timing for an even byte write from an 8-bit device. The timing for a word or odd byte write from an 8-bit or 16-bit device is identical, with the encoding of UDS and LDS selecting the proper byte(s), and HIBYTE and R/W controlling the flow of data from D0-D7 to D8-D15 if necessary (which will only occur for the case shown in Figure 4-11).

S0 — The DMAC drives the FC0-FC2, A1-A7, and A8/D0-A23/D15 pins with values from the MFCR and MAR of the current channel and AS is asserted to enable the external address latches. DBEN will be driven high and remain high through the end of the cycle so that the data buffers/multiplexers will not interfere with the data transfer.

S1 — No signals change and the upper address information is allowed to propagate through the external address latches. If the device with acknowledge and ready protocol is used, PCL will first be sampled on the rising edge at the end of S1; thus, if PCL is low by the end of S1, SD0 and SD1 will not occur and the timing will be the same as for the device with acknowledge-only protocol.

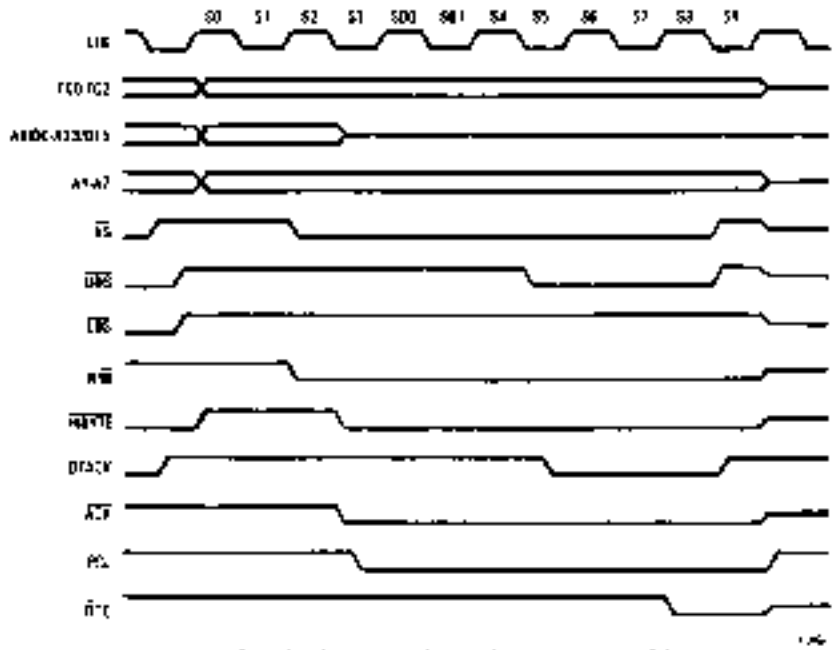


Figura 4-11. Single Address Write Cycle Timing Diagram

S2 $\overline{UAS}$ is negated to latch the upper address information and $\overline{AS}$ is asserted to signal that the address is valid and decoding may begin. R/$\overline{W}$ is asserted to indicate that data will flow to the memory from the device

S3 The address/data pins are three stated and $\overline{ACK}$ is asserted to indicate that the device should place data on the bus. If the device port size is eight bits and the address is even, $\overline{HIBYTE}$ will be asserted to cause the device data on D0-D7 to be driven onto D8-D15 through an external buffer (in this case, $\overline{UDS}$ will be asserted and $\overline{LDS}$ will be negated). If this is the last transfer in a block operation, the $\overline{DONE}$ signal will be asserted by the DMAC.

If the protocol being used is a device with acknowledge only, then S4 will immediately follow S3. If the protocol being used is device with acknowledge and ready, then the PCL pin will be sampled to determine when the DMAC should assert the data strobe(s). If PCL is low by the end of S3, then the DMAC will insert one delay cycle (SD0, SD1) and continue into S4. If PCL is high at the end of S3, then the DMAC will insert additional delay cycles (two delay states named SDw) after S3 until a ready signal is asserted and then proceed to SD0.

The DMAC also begins to sample the $\overline{DTACK}$ and $\overline{BEC0}$ $\overline{BEC2}$ pins to determine when the cycle should be terminated. If $\overline{DTACK}$ is asserted by the end of S3, then the DMAC will 'skip' S6 and S7 and proceed directly from S5 to S8. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S3, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC may execute S6 and S7 (see SD1 below).

SD0 This state is not executed for the device with acknowledge-only protocol. No signals change during SD0. The device is preparing to drive data onto the bus at this time and the DMAC is internally synchronizing PCL.

SD1 This state is not executed for the device with acknowledge only protocol. No signals change during SD1, but the device continues to prepare to drive data onto the bus. The DMAC continues to synchronize the PCL signal. If $\overline{DTACK}$ is asserted by the end of SD1, then the DMAC will 'skip' S6 and S7 and proceed directly from S5 to S8. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of SD1, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$-$\overline{BEC2}$ are coded to normal at the end of S3, then the DMAC will execute S6 and S7.

S4 No signals change during S4. Data from the device should be valid on the bus at the end of S4.

S5 $\overline{UDS}$ and/or $\overline{LDS}$ is asserted at this time to indicate that valid data is present on the bus and may be latched by the memory. If $\overline{DTACK}$ is asserted by the end of S5, then the DMAC may continue into S6 immediately. If $\overline{BEC0}$-$\overline{BEC2}$ are encoded with the bus error, retry, or relinquish and retry code by the end of S5, one wait cycle will be inserted between S5 and S6. If $\overline{DTACK}$ is negated and $\overline{BEC0}$ $\overline{BEC2}$ are coded to normal at the end of S5, then the DMAC will insert wait cycles after S5 until a termination signal is asserted and then proceed to S6.

S6 No signals change during S6. The DMAC is internally synchronizing the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins.

S7 No signals change during S7. The DMAC continues to synchronize the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ pins.

S8 $\overline{DTC}$ is asserted to indicate to the device that the bus cycle has been successfully terminated. $\overline{DTC}$ will not be asserted if a $\overline{BEC0}$-$\overline{BEC2}$ coding of bus error, retry, relinquish and retry, or reset was used to terminate the cycle.

S9 $\overline{AS}$ is negated to indicate to the memory that the cycle has terminated. $\overline{UDS}$ and/or $\overline{LDS}$ is negated to disable the device to memory data buffers. The memory that was addressed during the cycle may then negate $\overline{DTACK}$ and/or $\overline{BEC0}$ $\overline{BEC2}$.

S9 + 1 This may be S0 of the following cycle. If the DMAC is ready to execute another bus cycle immediately. The FC0-FC2 and A1-A7 pins will all be changed to their next states. If the DMAC is ready to start the next bus cycle, the next state will be the address of the next memory location. If the DMAC is not ready to start a bus cycle, these lines will be three stated. R/$\overline{W}$ and $\overline{HIBYTE}$ are driven high to prepare the data bus for the next cycle. $\overline{ACK}$, $\overline{DTC}$, and $\overline{DONE}$ are all negated (if they were asserted) to indicate to the peripheral that the cycle has completed. The peripheral may respond by negating the ready signal at this time (for the device with acknowledge and ready protocol).

Note that although the basic timing for a single address write using the device with acknowledge and ready protocol is six clock cycles, if the ready and $\overline{DTACK}$ signals are asserted for one half clock before $\overline{ACK}$ and the data strobes are asserted respectively, then the timing will be reduced to four clock cycles. There are also two timing combinations that give a five clock cycle transfer: one for an 'early' ready and

one for an 'early' $\overline{DTACK}$. In all cases, the system designer must be certain that data setup and hold times are met for the memory with respect to the data strobes.

If a channel is programmed for the device with acknowledge and ready protocol and one of the chaining modes, then the ready signal (PCLI) for that channel must be asserted during accesses to the chain control table. This can be accomplished with a circuit such as the one shown in Figure 4-12, which asserts ready (PCLx) whenever $\overline{ACK}$ for this channel is negated and $\overline{DTACK}$ is asserted. Even though this circuit causes ready (PCLc) to be asserted during DMAC bus cycles executed for other channels, this will not cause improper behavior, since the PCLc input is ignored when other channels are active.

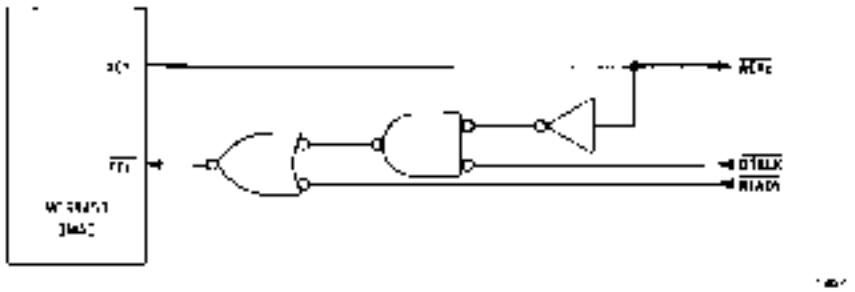


Figure 4-12. $\overline{READY}$ Circuit for Device with Acknowledge and Ready

Also note that when using the device with acknowledge and ready protocol, the assertion of $\overline{DTACK}$ should not precede the assertion of the data strobe(s). This is due to the fact that if $\overline{DTACK}$ is asserted before ready (PCLc) is asserted, then the DMAC will terminate the cycle in the normal fashion, but the data strobe(s) will not be asserted. Thus, if the falling or rising edge of the data strobe(s) is used by memory to latch data, the data transfer will not execute properly.

4.3.2.3 HIGH-BYTE OPERATION. During single-address transfers between an 8-bit device and 16-bit memory, $\overline{HIBYTE}$ is used to control a bidirectional buffer as shown in Figure 2-2. $\overline{HIBYTE}$ is asserted when the memory address is even, in which case $\overline{UDS}$ will be asserted and $\overline{LDS}$ will be negated. R/$\overline{W}$ is used to determine the direction of the data transfer and the data flow through the external buffer. This method of operation allows the 8-bit device to be placed on D0-D7 so that it may properly return an interrupt vector number to an M68000 system processor, while preventing buffer contention on the data lines.

There are four possible cases for data flow during a single-address transfer between an 8-bit device and memory; even and odd read, and even and odd write cycles. Figure 4-13 shows the data flow for the four cases.

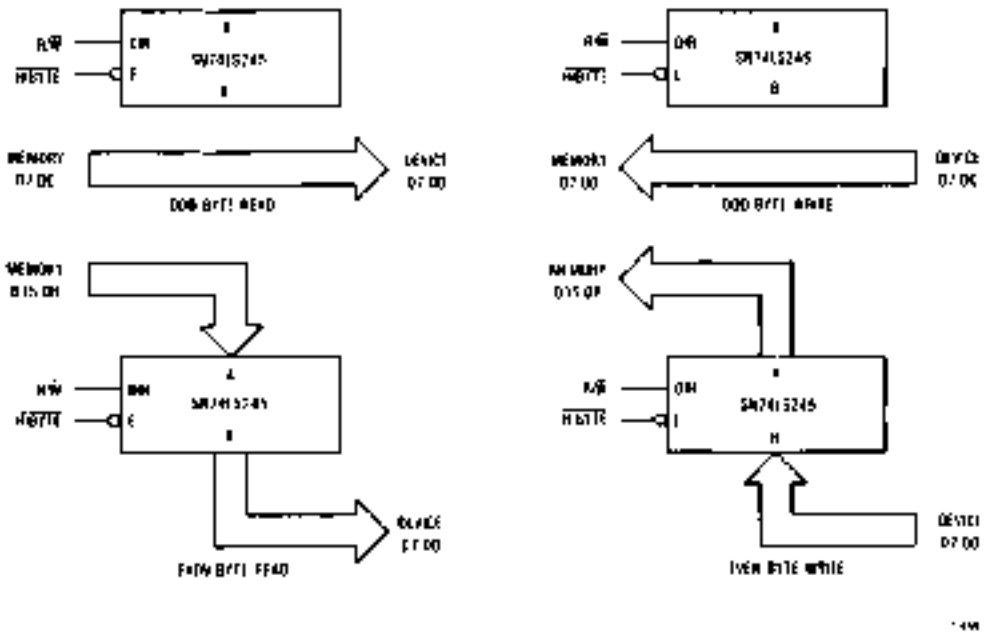


Figure 4-13. Implicitly Addressed 8-Bit Device with 16-Bit Memory Data Flow

4.4 BUS EXCEPTION CONTROL

In order to fully support the M68000 bus architecture, the DMAC uses three encoded inputs, $\overline{BEC0}$-$\overline{BEC2}$, to detect abnormal bus cycle termination conditions. These three lines function in a similar manner to the $\overline{RESET}$, $\overline{HALT}$, and $\overline{BERR}$ signals on an M68000 processor, but have different functionality than the processor counterparts. Table 4-1 shows the definition for each encoding of the $\overline{BEC0}$-$\overline{BEC2}$ pins and Figure 4-14 illustrates how the three inputs might be generated from signals normally present in an M68000 system.

Table 4-1. $\overline{BEC}$ Encoding Definitions

$\overline{BEC2}$	$\overline{BEC1}$	$\overline{BEC0}$	Definition
H	H	H	No Exception
H	H	L	Halt and Release Bus
H	L	H	Bus Error
H	L	L	Retry
L	H	H	Relinquish and Retry
L	H	L	Undefined, Reserved
L	L	H	Undefined, Reserved
L	L	L	Reset

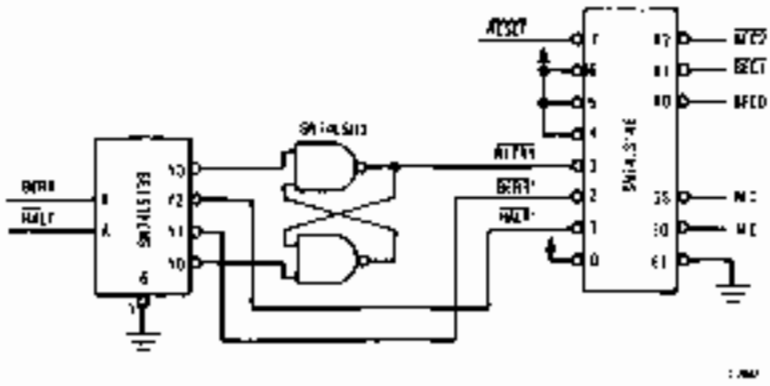


Figure 4-14. Example $\overline{BEC}$ Signal Generation Circuit

4.4.1 $\overline{BEC0}$-$\overline{BEC2}$ Synchronization

The bus exception control inputs are synchronized in the same manner as all asynchronous inputs, but an additional debounce delay is included in order to assure valid operation in a system. The $\overline{BEC0}$ $\overline{BEC2}$ pins are latched on each rising edge of CLK and are internally synchronized on the next rising edge of CLK. This synchronizer can be represented as a shift register as shown in Figure 4-15. On the $\overline{BEC}$n pins, an additional shift register stage is added to perform the debounce function. The internally synchronized signals BECn' and $\overline{BEC}$n" are continuously compared and if they are the same, then the level is accepted as valid and the DMAC may take action on it. If the two encodings are different, the DMAC will wait one more clock cycle and perform the comparison again. In this way, an SN74LS148 may be used to generate the $\overline{BEC0}$-$\overline{BEC2}$ inputs and transitional encodings will not cause erroneous operation. This debouncing also means that an encoding must be stable for at least two clock cycles to guarantee that it is latched at the same level on two successive rising clock edges.

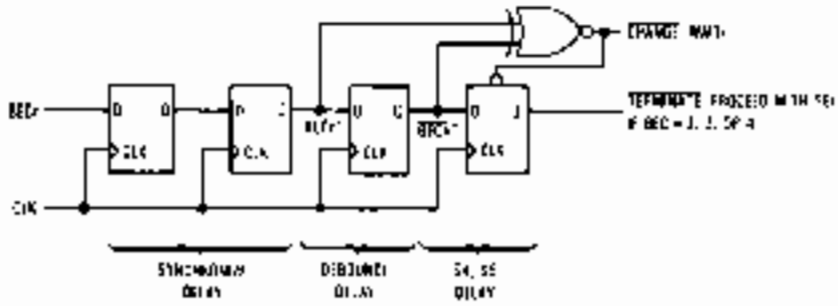


Figure 4-15. $\overline{BEC}$ Synchronizer Functional Diagram

As an example of the debouncer operation, consider the timing diagram in Figure 4-16. When the SN74LS148 outputs change from halt to bus error, they may transition through retry at the time when the DMAC is latching the $\overline{BEC0}$-$\overline{BEC2}$ pins. The internal debounce circuit will ignore the retry code and instead terminate the cycle with a bus error. Note that the delay from the first valid bus error, retry, or relinquish and retry encoding to when the bus cycle is ended is one synchronization delay (one to two clocks), plus one debounce delay (one clock), plus one bus controller delay (one clock) needed for the DMAC to switch from a normal termination to a bus exception type termination. This results in a bus cycle that is two clock cycles longer than it had been terminated by a $\overline{DTACK}$ signal. Also, note that if $\overline{DTACK}$ and bus error, retry, or relinquish and retry are asserted at the same time and the asynchronous input setup time is met for the $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ signals, then the bus cycle will be two clock cycles longer than if it had been terminated with $\overline{DTACK}$ alone and $\overline{DTACK}$ will be ignored. This behavior is different from what an M68000 processor will exhibit since the DMAC must suppress the assertion of $\overline{DTC}$ and prevent the internal address counters from incrementing if necessary.

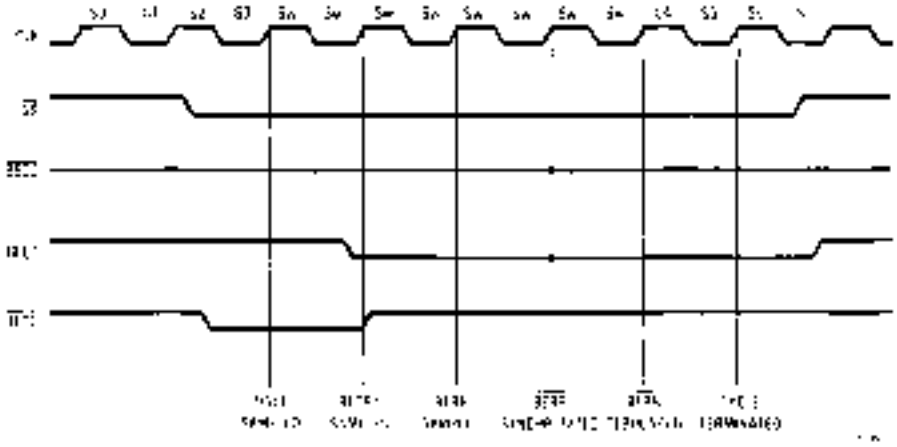


Figure 4-16 $\overline{BEC}$ Debouncer Timing Diagram

4.4.2 Bus Exception Control Functions

The three bus exception control signals allow eight different bus termination conditions to be signaled to the DMAC as shown in Table 4-1. The function of each encoding is described below. In describing the encoding of the $\overline{BEC0}$-$\overline{BEC2}$ pins, $\overline{BEC0}$ is the least significant bit and $\overline{BEC2}$ is the most significant bit of a binary coded digit from zero (0) to seven (7) with the encoding HHH equal to zero.

Whenever a non-zero value is encoded on the $\overline{BEC}$ inputs, the DMAC will enter the idle mode of DMA wait-state mode as soon the current bus cycle is terminated and remain in that state until the $\overline{BEC}$ encoding returns to normal. Another system bus master may access the DMAC registers while it is in the idle mode if it can gain mastership of the DMAC bus. In this manner, the DMAC can be halted by external hardware to enable the system processor to modify the DMAC registers and abort or monitor channel operations if necessary.

4.4.2.1 NORMAL TERMINATION. If a zero (0) is encoded on the $\overline{BEC}$ pins, the DMAC will operate in the normal mode and $\overline{DTACK}$ is used to terminate bus cycles.

4.4.2.2 HALT. If a one (1) is encoded on the $\overline{BEC}$ pins, the DMAC will be halted when the current bus cycle is terminated by the assertion of $\overline{DTACK}$. In order to preempt the bus cycle following the current cycle, the halt encoding must be valid one synchronization delay plus one debounce delay prior to when S0 of the next cycle would normally start. When the DMAC halts in response to the halt input, it will release ownership of the bus and enter the idle state until the $\overline{BEC}$ coding is returned to normal, at which time it will arbitrate for the bus and continue DMA operations if necessary. Figure 4-17 shows the timing of the halt operation. Note that if the halt encoding is asserted on the $\overline{BEC}$ pins at the same time that $\overline{DTACK}$ is asserted, the cycle will terminate two clock cycles later than if $\overline{DTACK}$ alone were asserted due to the debounce delay on the $\overline{BEC}$ inputs. Also, when the $\overline{BEC}$ pins are returned to normal coding, the DMAC will wait three clock cycles before asserting $\overline{BR}$ to re-arbitrate for the bus.

4.4.2.3 BUS ERROR. If a two (2) is encoded on the $\overline{BEC}$ pins, the DMAC will abort the current bus cycle and associated channel operation, and log an error in the channel status register and channel error register. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will execute an internal error recovery sequence and then may relinquish ownership of the bus or start a bus cycle for the other channel if it has an operand transfer request pending. The error recovery sequence requires 25 clock cycles if the aborted cycle is a single address transfer or a dual address read, and 29 clock cycles if the aborted bus cycle is a dual address write. Figure 4-18 shows the timing of a bus error operation.

If a bus cycle is aborted with a bus error, $\overline{DTC}$ will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR.

4.4.2.4 RETRY. If a three (3) is encoded on the $\overline{BEC}$ pins, the DMAC will terminate the current bus cycle, enter the DMA waiting mode, and $\overline{BGACK}$ will remain asserted so that the DMAC retains ownership of the bus. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will wait for three clocks and re-run the same bus cycle, using the same function code and address values. Figure 4-19 shows the functional timing of a retry operation.

If a bus cycle is terminated with a retry, $\overline{DTC}$ will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR and the internal data holding register will not be affected.

4.4.2.5 RELINQUISH AND RETRY. If a four (4) is encoded on the $\overline{BEC}$ pins, the DMAC will terminate the current bus cycle and relinquish ownership of the bus. When the $\overline{BEC}$ encoding is returned to normal, the DMAC will wait for three clocks, re-arbitrate for the bus, and re-run the same bus cycle using the same function code and address values. If an operand transfer request is internally asserted for a higher priority channel than the channel being retried before the DMAC regains control of the bus, then the retry and any remaining bus cycles required to complete the current operand transfer will be executed before the higher priority request will be serviced. Figure 4-20 shows the functional timing of a relinquish and retry operation.

If a bus cycle is terminated with a relinquish and retry, DTC will not be asserted so that a peripheral device will not increment any operand counters. The DMAC will not increment or decrement the MAR, MTCR, or DAR and the internal data holding register will not be affected.

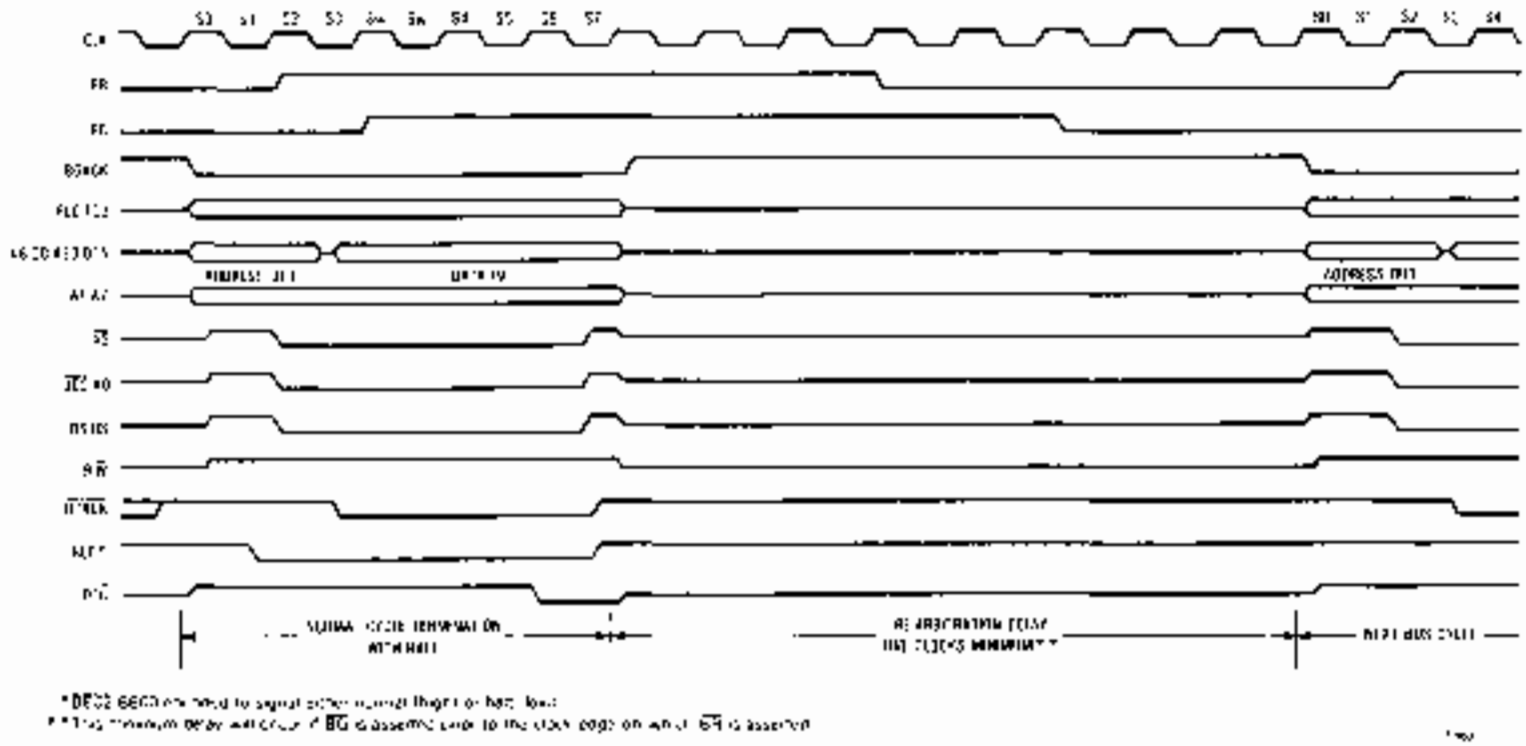


Figure 4-17. Halt Operation Timing Diagram

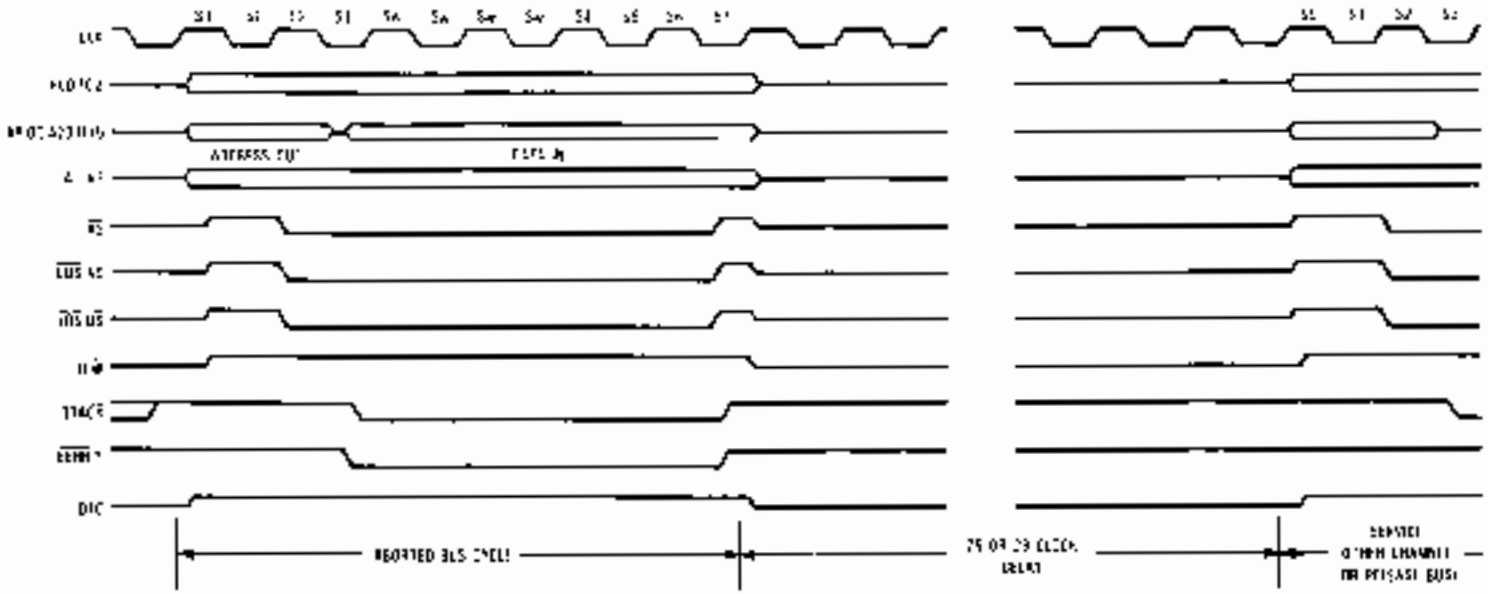


Figure 4-18. Bus Error Operation

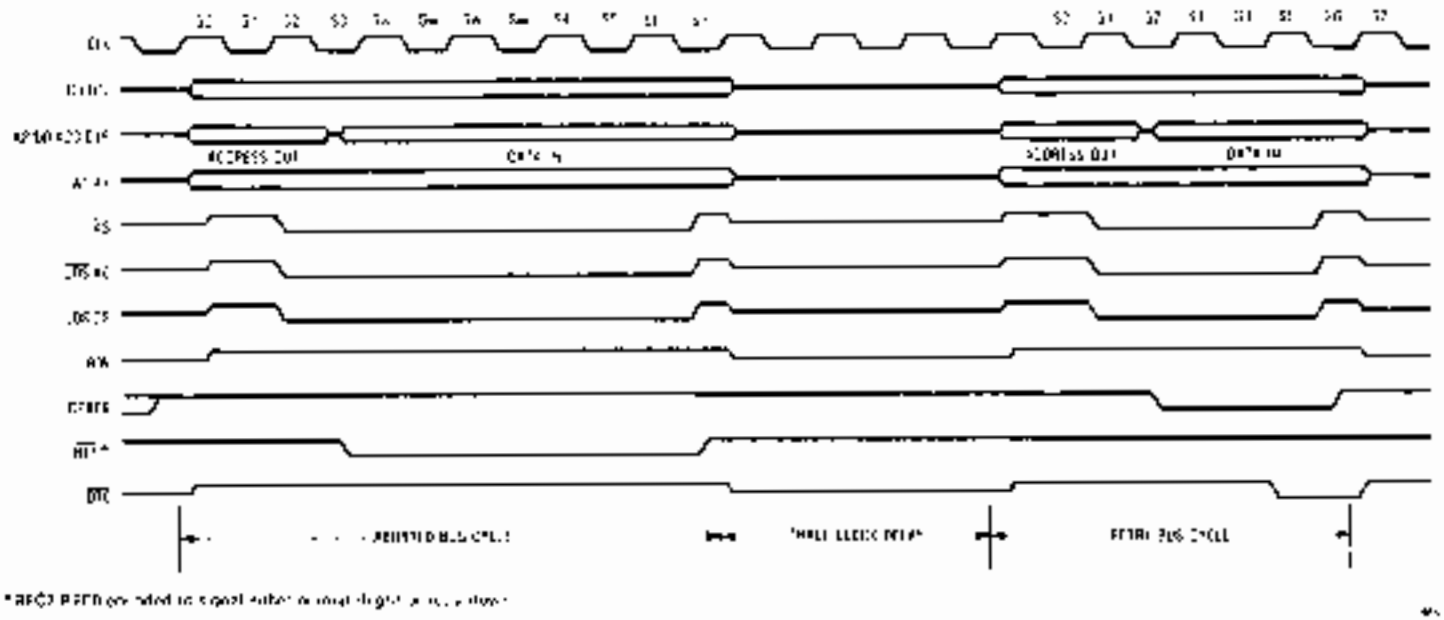


Figure 4-19 Retry Operation Timing Diagram

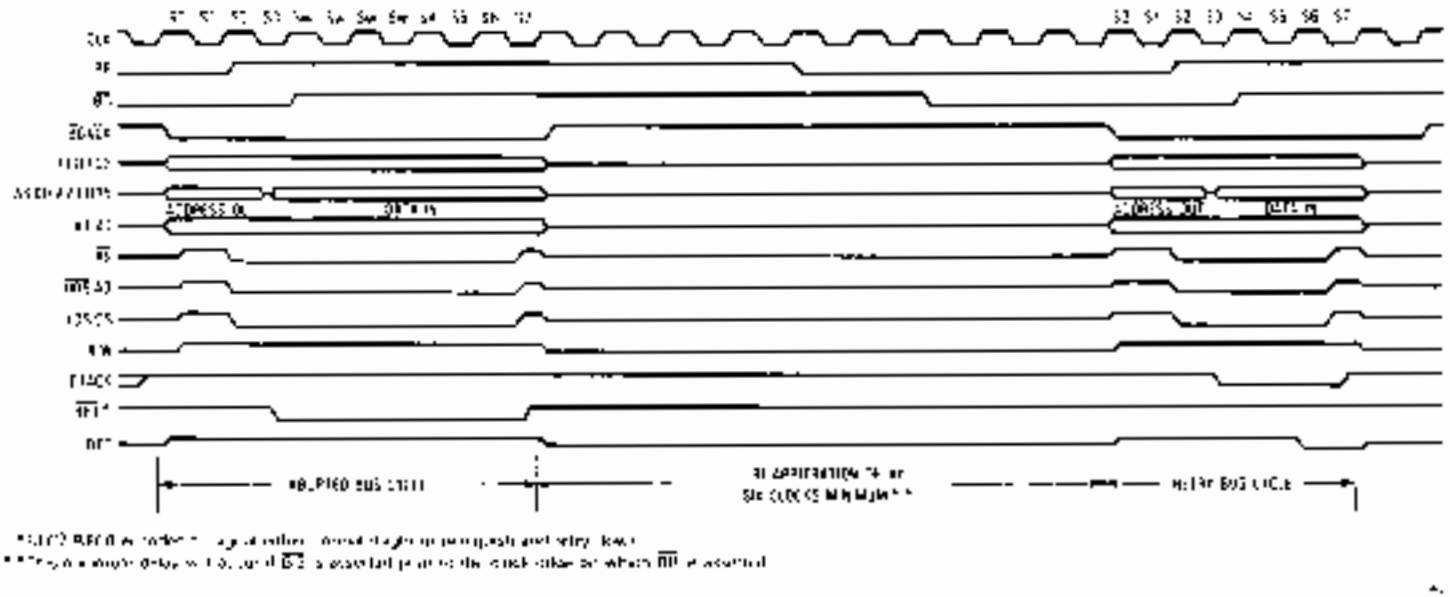
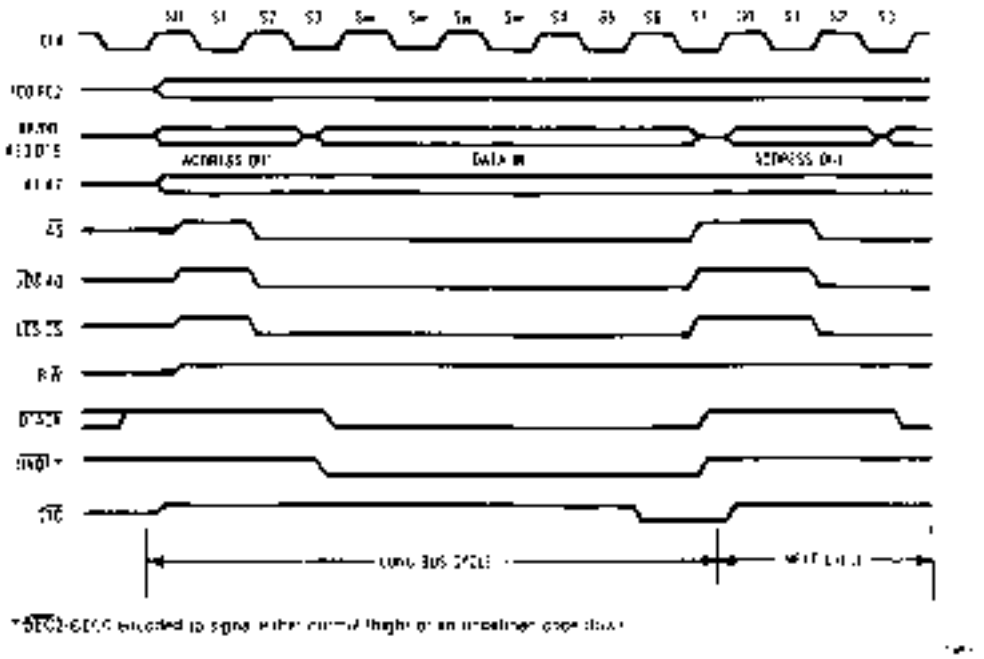


Figure 4-20 Relinquish and Retry Operation Timing Diagram

4.4.2.6 UNDEFINED $\overline{BEC}$ CODES. If a five (5) or six (6) is encoded on the $\overline{BEC}$ pins, the DMAC will synchronize and debounce the inputs as usual, but no action will be taken. One effect that these encodings will have is to lengthen a bus cycle until the $\overline{BEC}$ pins are returned to the normal encoding, regardless of the level on the $\overline{DTACK}$ pin. Furthermore, the DMAC will not continue or start to service any further operand transfer sequests until one synchronization delay plus one debounce delay after the $\overline{BEC}$ pins are returned to the normal coding. Figure 4-21 shows the functional timing of an undefined $\overline{BEC}$ operation.



*$\overline{BEC}$=ECC encoded to signal either normal (high) or an illegal (low) operation

Figure 4-21. Undefined $\overline{BEC}$ Operation Timing Diagram

4.4.2.7 RESET. If a seven (7) is encoded on the $\overline{BEC}$ pins, the DMAC will execute an internal reset sequence and enter the idle mode after one synchronization delay plus one debounce delay. Refer to **3.10 RESET OPERATION RESULTS** for a description of the effect that a reset operation has on the internal registers.

4.4.3 Bus Exception Control Summary

Figure 4-22 shows the bus exception control flow diagram for the DMAC in the idle and DMA modes of operation.

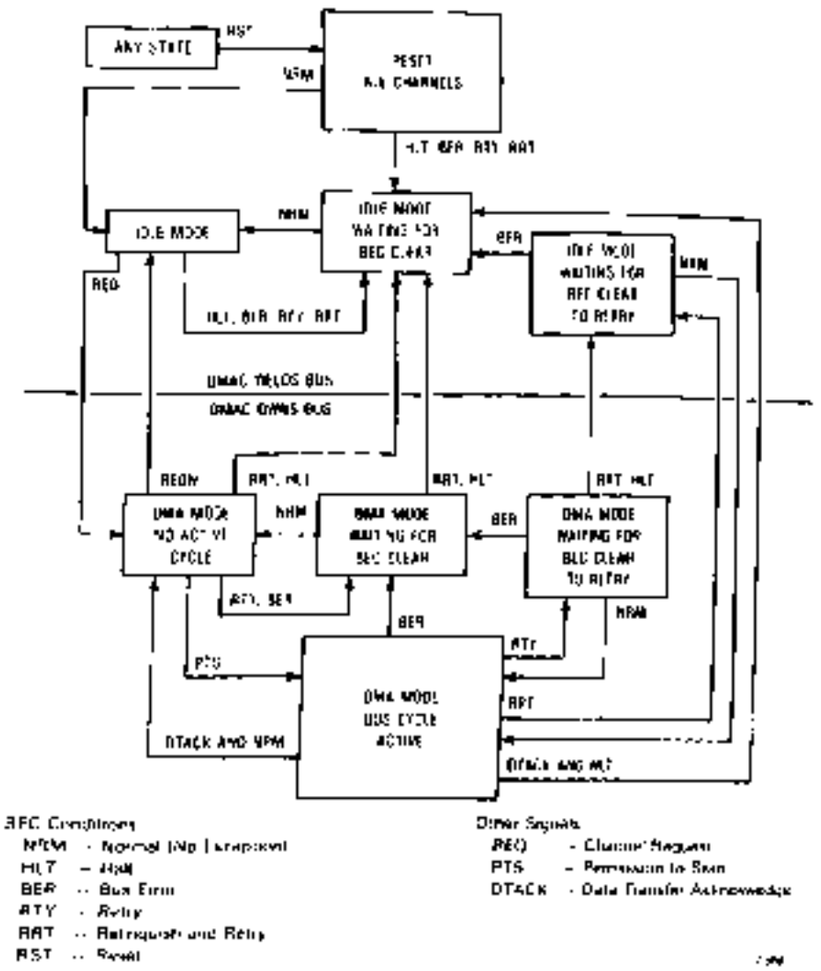


Figure 4-22. Bus Exception Control Flow Diagram

4.6 BUS ARBITRATION

Once the system processor has initialized and started a DMAC channel and an operand transfer request has been made pending, either by an external device or the internal request generator, the DMAC will use the M68000 bus arbitration protocol to request bus mastership before entering the DMA mode of operation. The following paragraphs describe the arbitration protocol used by the DMAC and several special cases of operation relative to bus arbitration. Figure 4-23 shows a functional timing diagram of a typical bus arbitration sequence.

4.6.1 Requesting and Receiving the Bus

When an active channel has an operand transfer request pending, the DMAC will request bus mastership by asserting the $\overline{BR}$ signal. An external bus arbiter (either a separate unit such as the MC68452 Bus

Arbitration Module or the arbiter built into an M68000 processor) will then assert BG to indicate that bus mastership will belong to the DMAC as soon as the current bus master has released the bus. The DMAC then monitors the $\overline{AS}$ and $\overline{BGACK}$ signals to determine when it may assume mastership of the bus; these two signals must be negated to indicate that the previous bus cycle is complete and the previous bus master has released the bus. When this condition is met, the DMAC will assert OWN and then assert $\overline{BGACK}$ one-half clock cycle later to indicate that it has taken control of the bus. One clock after $\overline{BGACK}$ is asserted, $\overline{BR}$ will be negated to allow the external arbiter to begin arbitration for the next bus master. Once all operand transfer requests have been serviced, the DMAC will release control of the bus by negating $\overline{BGACK}$ and will then negate $\overline{OWN}$ one half clock cycle later. Note that $\overline{OWN}$ may be used to control a bidirectional buffer for $\overline{BGACK}$ since it is asserted before and negated after $\overline{BGACK}$ is asserted and negated, respectively.

Note that if the DMAC is requesting the bus and $\overline{CS}$ or $\overline{IACK}$ is asserted, $\overline{BR}$ will be negated without waiting for the assertion of $\overline{BG}$. After the MPU cycle is completed, $\overline{BR}$ will be reasserted to continue the DMA operation.

4.5.2 Bus Overhead Time

In asynchronous bus systems such as those defined for the M68000 Family, a certain amount of time is used to synchronize incoming signals and is thus 'wasted', since no data transfer activity can take place during those periods. In many applications, the synchronization overhead time required to switch bus masters and channels within the DMAC must be known to predict system behavior. The following paragraphs describe three types of overhead periods: those associated with the DMAC taking control of the bus, those in between successive operand transfers, and those that occur when the DMAC is releasing control of the bus to another bus master. In all of these discussions, the system is assumed to consist of a single M68000 processor and a single DMAC using the same clock so that all arbitration delays are controlled by those two devices and can be predicted, as shown in Figure 4-23.

4.5.2.1 FRONT-END OVERHEAD. This is the delay that will occur from the time that the MPU terminates a bus cycle by negating $\overline{AS}$ (in S7 of the MPU cycle) to when the DMAC starts a bus cycle by placing function code and address information on the bus (in S0 of the DMA cycle). It is assumed that $\overline{BG}$ is asserted and $\overline{BGACK}$ is negated prior to the negation of $\overline{AS}$ by the MPU so that no additional synchronization delays are introduced by those signals.

The amount of idle time between a bus cycle executed by another bus master and the start of a DMA cycle is dependent on the point at which the DMAC receives a request in relation to the bus cycle being executed by the other bus master (the MPU in this case). The minimum time for the recognition of a request from a device to when the DMAC will start a bus cycle is 12 clock cycles. A portion of these 12 clock cycles will normally overlap with the execution of a bus cycle by another bus master, as shown in Figure 4-23. This figure shows the best case overlap, which results in a front-end overhead of five clocks. Figure 4-24 shows the worst case overlap, which results in an overhead of eight clocks. The difference in the two cases is due to the fact that for the best case $\overline{BR}$ is recognized, by the MPU, as asserted during S6 of a bus cycle, which is too late to preempt the next MPU cycle; thus the next bus cycle overlaps the DMAC bus arbitration sequence. For the worst case, the $\overline{BR}$ assertion is recognized by the MPU during S4 of a bus cycle, which is early enough to preempt the next MPU bus cycle, and only one and one-half clocks of the current MPU bus cycle overlap with the bus arbitration.

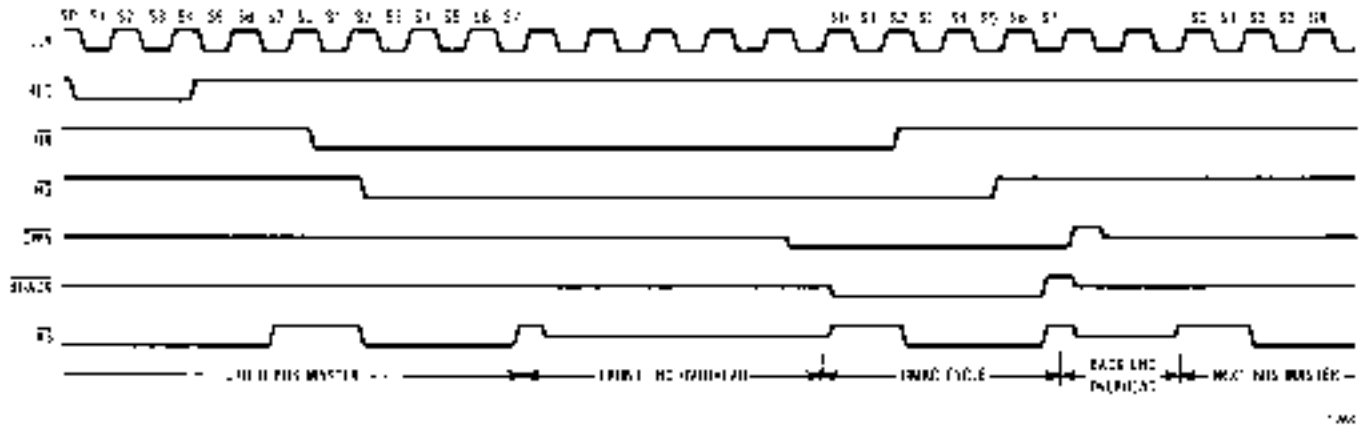


Figure 4-23. Bus Arbitration Timing Diagram (Best Case)

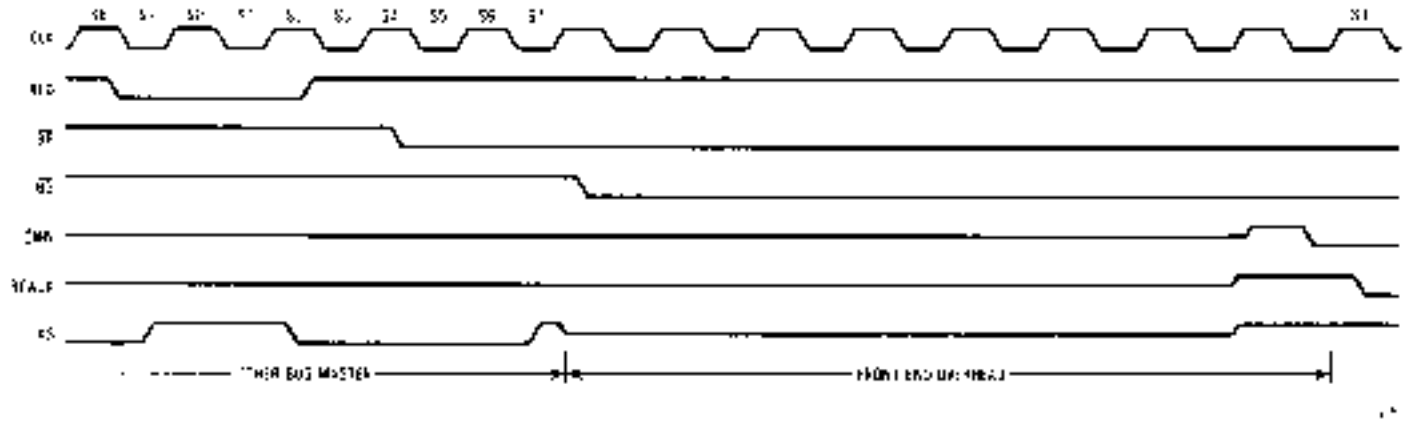


Figure 4-24. Front End Overhead (Worst Case)

4.5.2.2 BACK-END OVERHEAD. This overhead time is the delay between when the DMAC has completed all pending operand transfers and releases the bus, plus the synchronization of $\overline{BGACK}$ negated by the MPU. As the DMAC completes the last pending bus cycle in S7 for a DMA read or S9 for a DMA write, it may negate $\overline{BGACK}$ at the same time as $\overline{AS}$. The MPU will then wait one synchronization delay plus one-half clock cycle before S0 of the next MPU cycle. Thus, the best case back end overhead time is two clock cycles, and the worst case is determined by the mode of operation as described below.

4.5.2.3 INTER-CYCLE OVERHEAD. This is the overhead delay that will occur between successive DMA bus cycles for any of the channels, or at the end of an operand transfer, while the DMAC maintains ownership of the bus.

Figure 4-25 shows the timing, in clock cycles, for various operations that the DMAC may perform while it owns the bus (i.e., when $\overline{BGACK}$ is asserted). The operations described include block transfer initiation and termination times for chained operations, all types of operand transfers, and channel shut down sequences. Each operation described by one box in this figure is independent of the number of channels that are active, and there is no overlap of execution between channels (i.e., idle termination time for one channel will not be concurrent with the idle initiation time for another channel). To use this table, select all of the individual operations that will be executed during one period of bus ownership (referred to as a burst period) and add the total number of clocks for each operation to get the total burst period time in clocks. Also, the total number of read and write cycles and the total number of overhead clocks can be calculated. In addition, the appropriate front end and back end overhead must also be included in the calculation.

In addition to the overhead times that may be introduced during the operations described in Figure 4-25, the timing of a cycle steal request assertion can also affect bus overhead times. Figure 4-26 indicates the number of idle clocks that will occur between the current operation (which is for a channel using cycle steal requests) and the start of the next operation (for any channel), depending on when a new request is asserted by a device. The intervals shown in this figure are in relation to the CLK signal, such that the start or end of an interval is slightly before a rising clock edge. With this assumption, an asserted edge of $\overline{REQ}$ is guaranteed to meet the asynchronous input setup time to a rising edge of CLK within that interval and the behavior of the DMAC can be predicted. If $\overline{REQ}$ is asserted in a given interval shown in Figure 4-26, the number of idle clock cycles labeled in that interval will occur before the next operation. If $\overline{REQ}$ is asserted during an interval labeled "Ignore Requests", meets the two clock minimum pulse width, and is negated before the end of that interval, then the channel will not recognize the request. However, if $\overline{REQ}$ is asserted during such an interval and remains asserted for two clock cycles into the next interval, then a new request will be recognized.

THE FOLLOWING NOTES APPLY TO FIGURE 4-25 (SHEETS 1 THROUGH 7) WHICH APPEARS ON THE PAGES 4-31 THROUGH 4-37

NOTES

1. Each tic mark on a line represents the end of a two clock period.
2. All read cycles are assumed to take four clocks and all write cycles are assumed to take five clocks. In a system that does not meet these assumptions, the number of additional clock cycles for each bus cycle is added to the total to obtain the actual operation timing.
3. Except where noted, the request generation method does not affect operation timing.
4. The only operation that will follow operations marked with an asterisk (*) will be the appropriate termination operation.

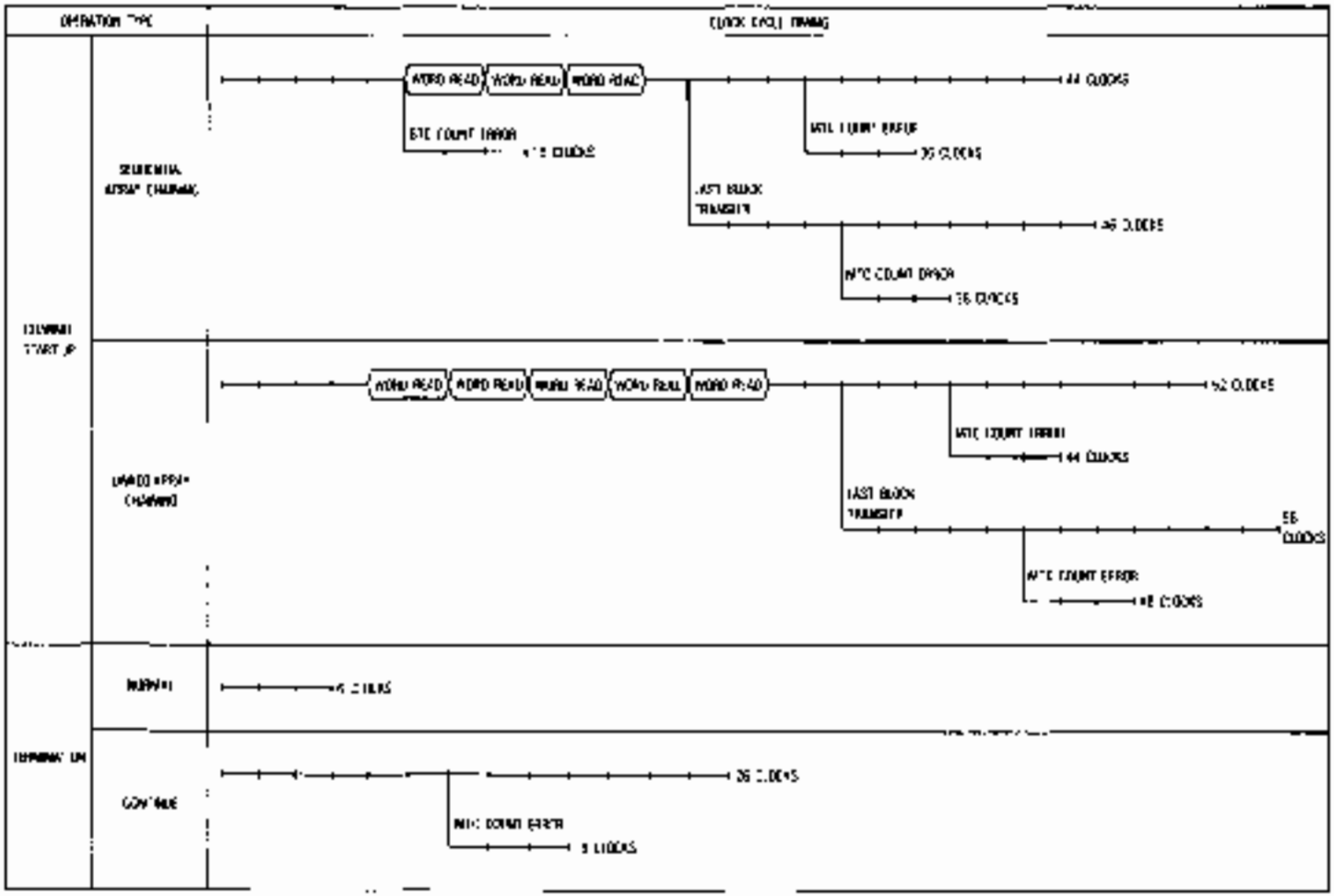


Figure 4-25. DMA Operation Timing Table (Sheet 1 of 7)

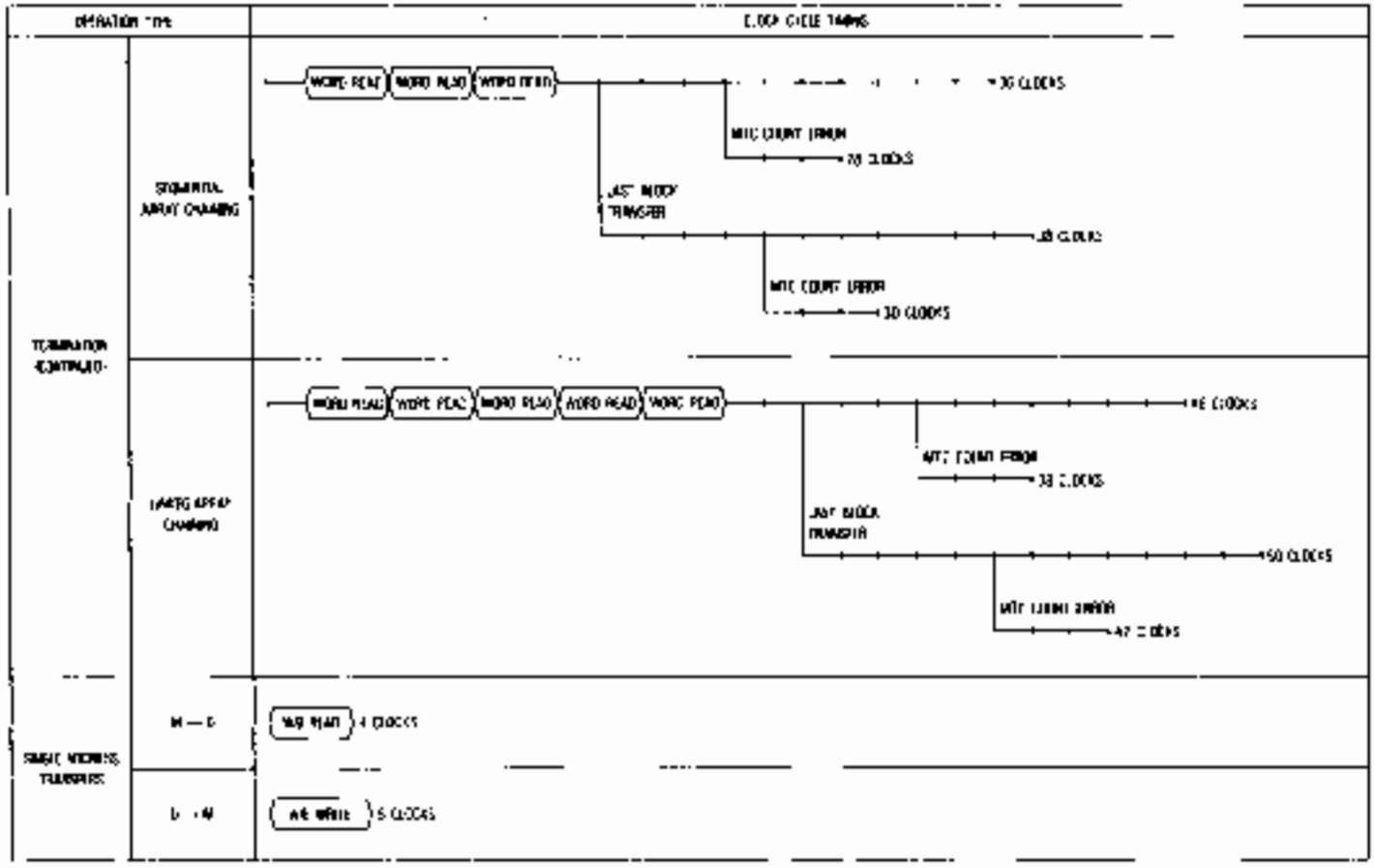


Figure 4-25. DMA Operation Timing Table (Sheet 2 of 7)

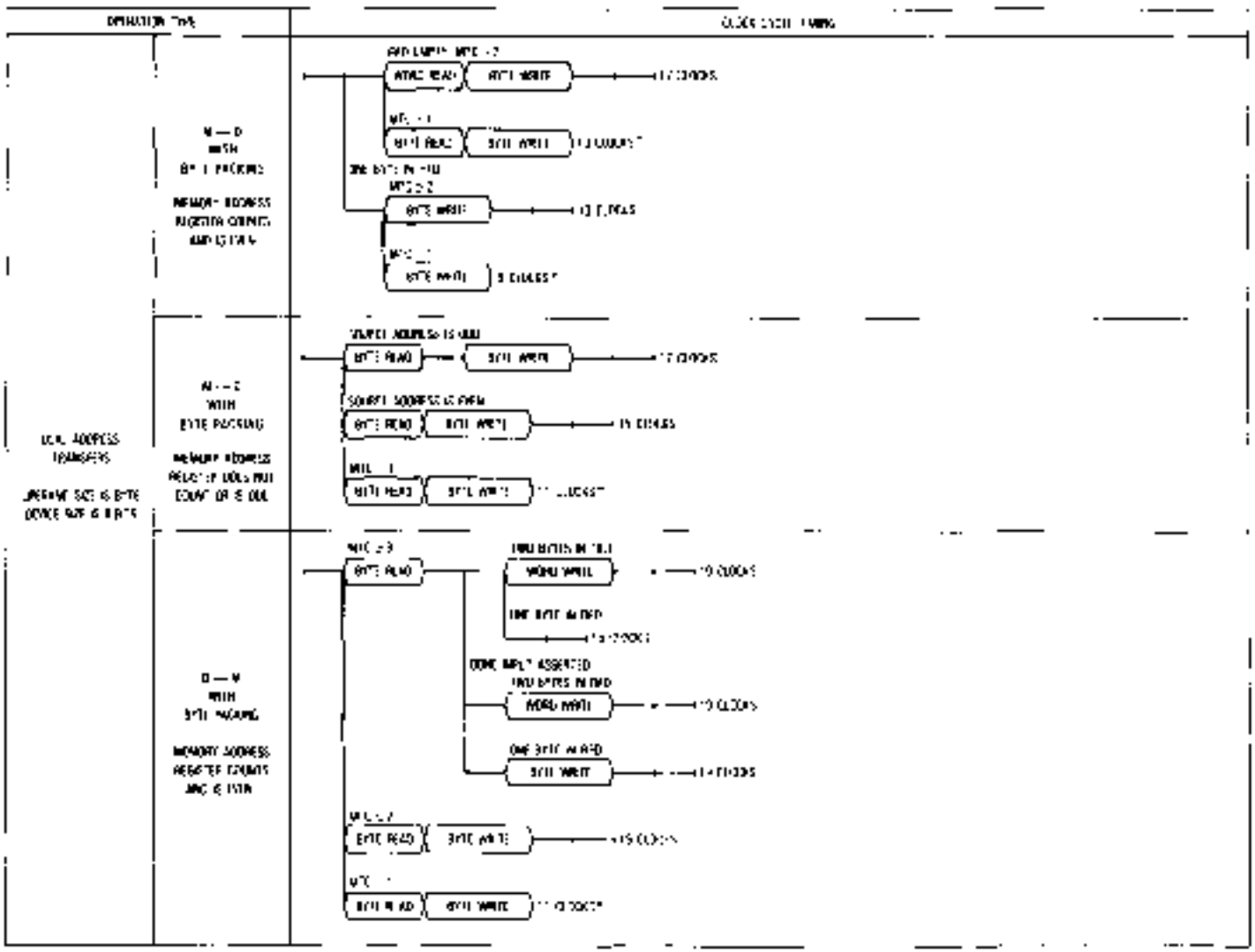


Figure 4-25 DMA Operation Timing Table (Sheet 3 of 7)

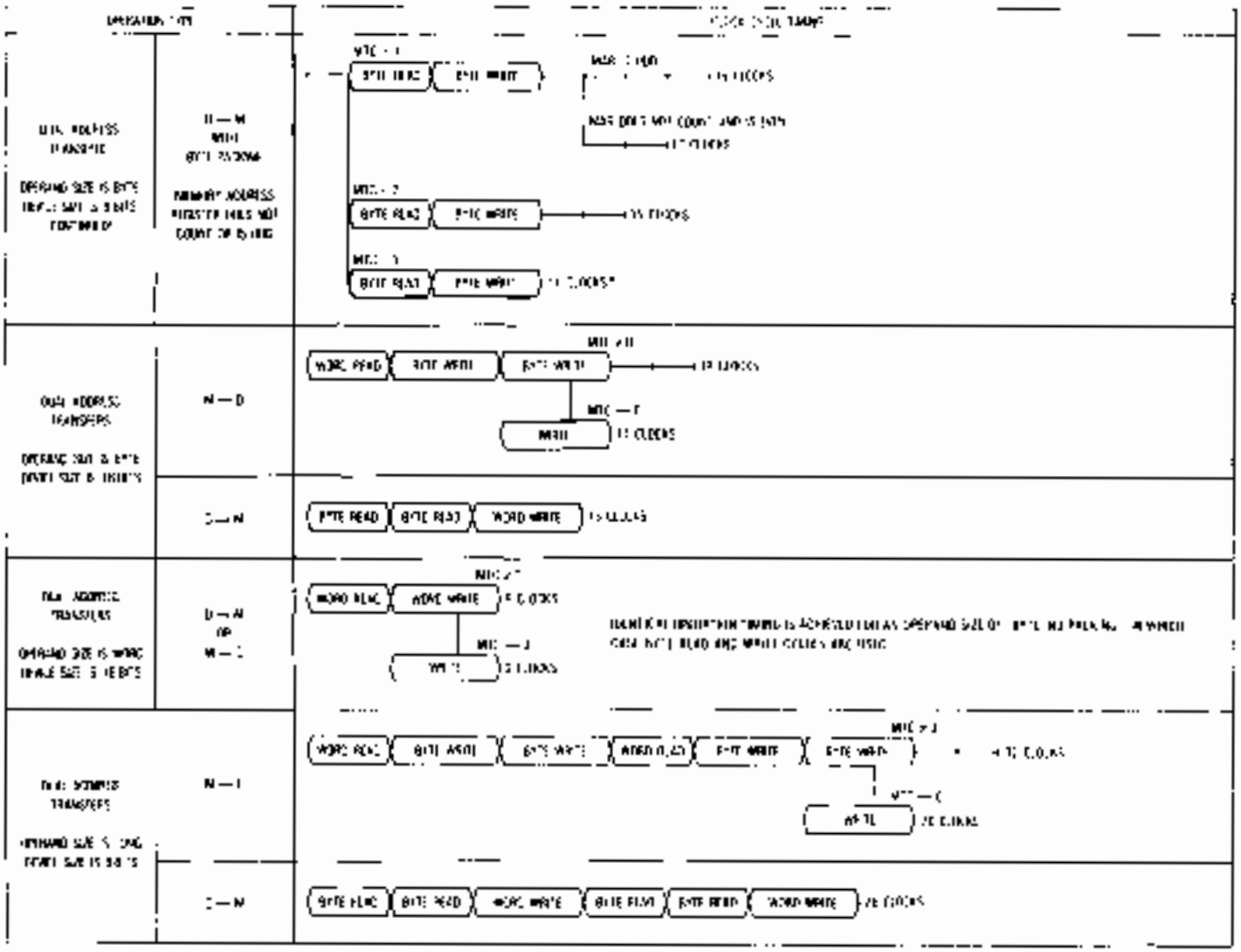


Figure 4-25. DMA Operation Timing Table (Sheet 4 of 7)

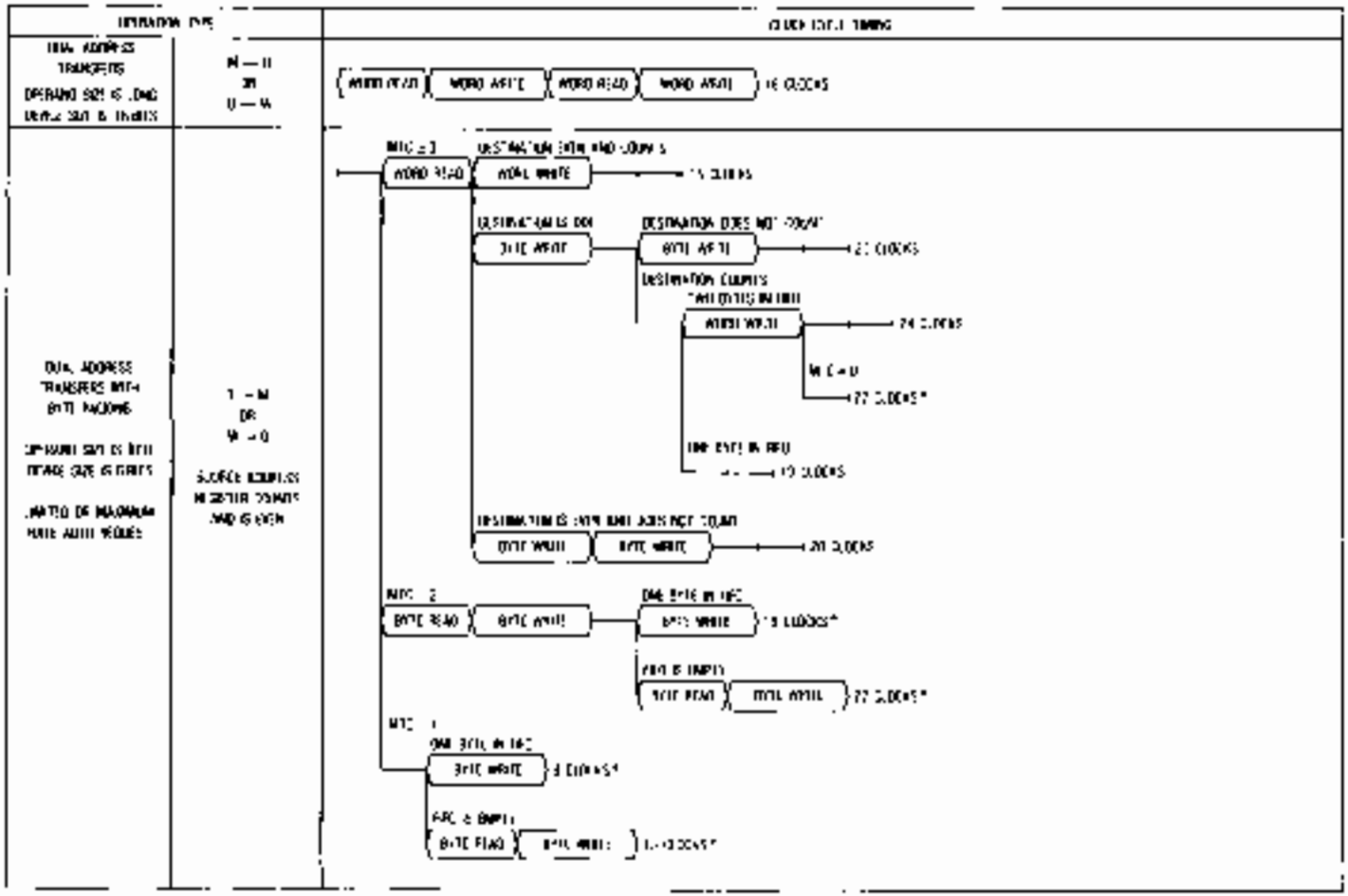


Figure 4-25. DMA Operation Timing Table (Sheet 5 of 7)

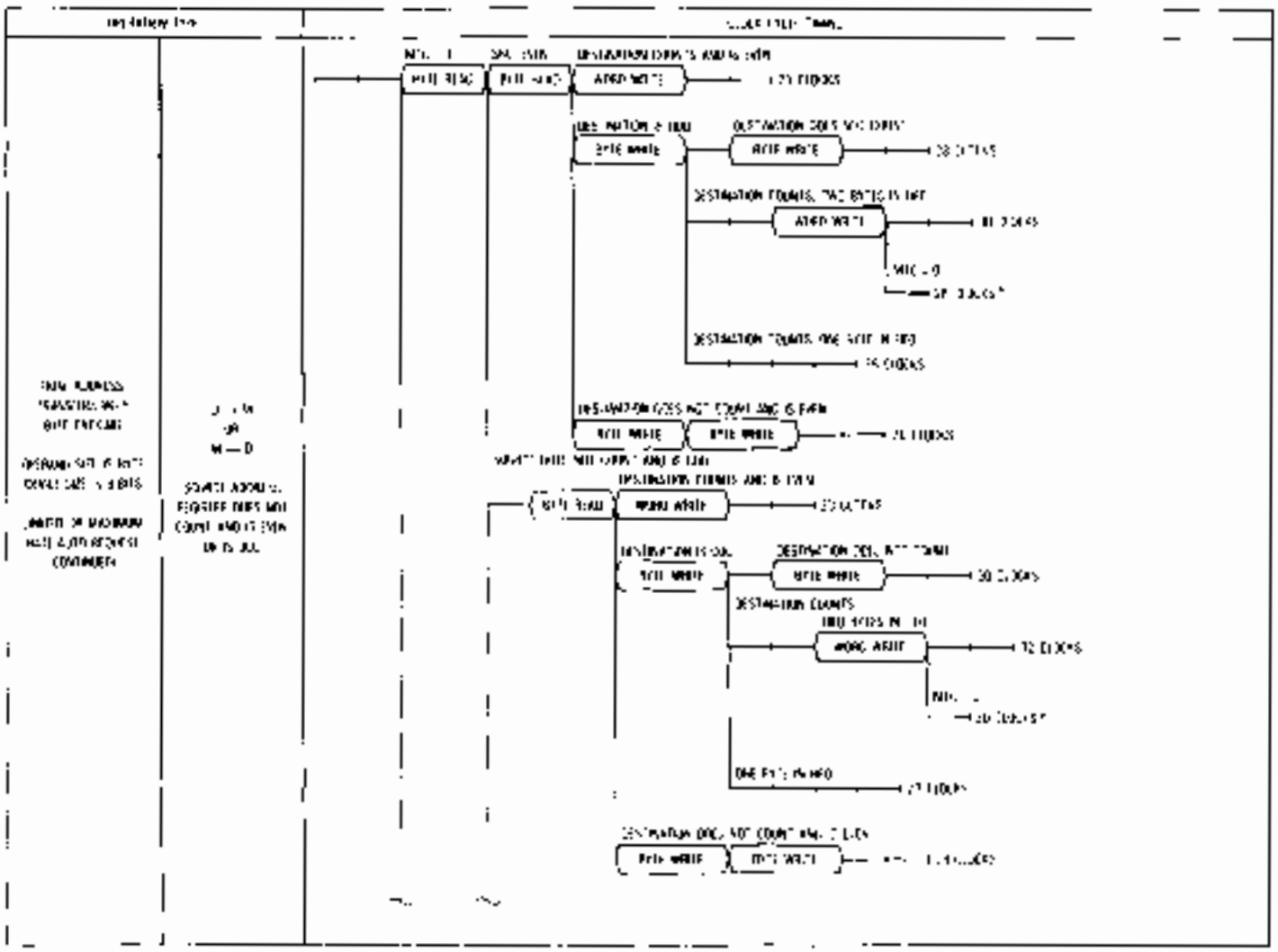
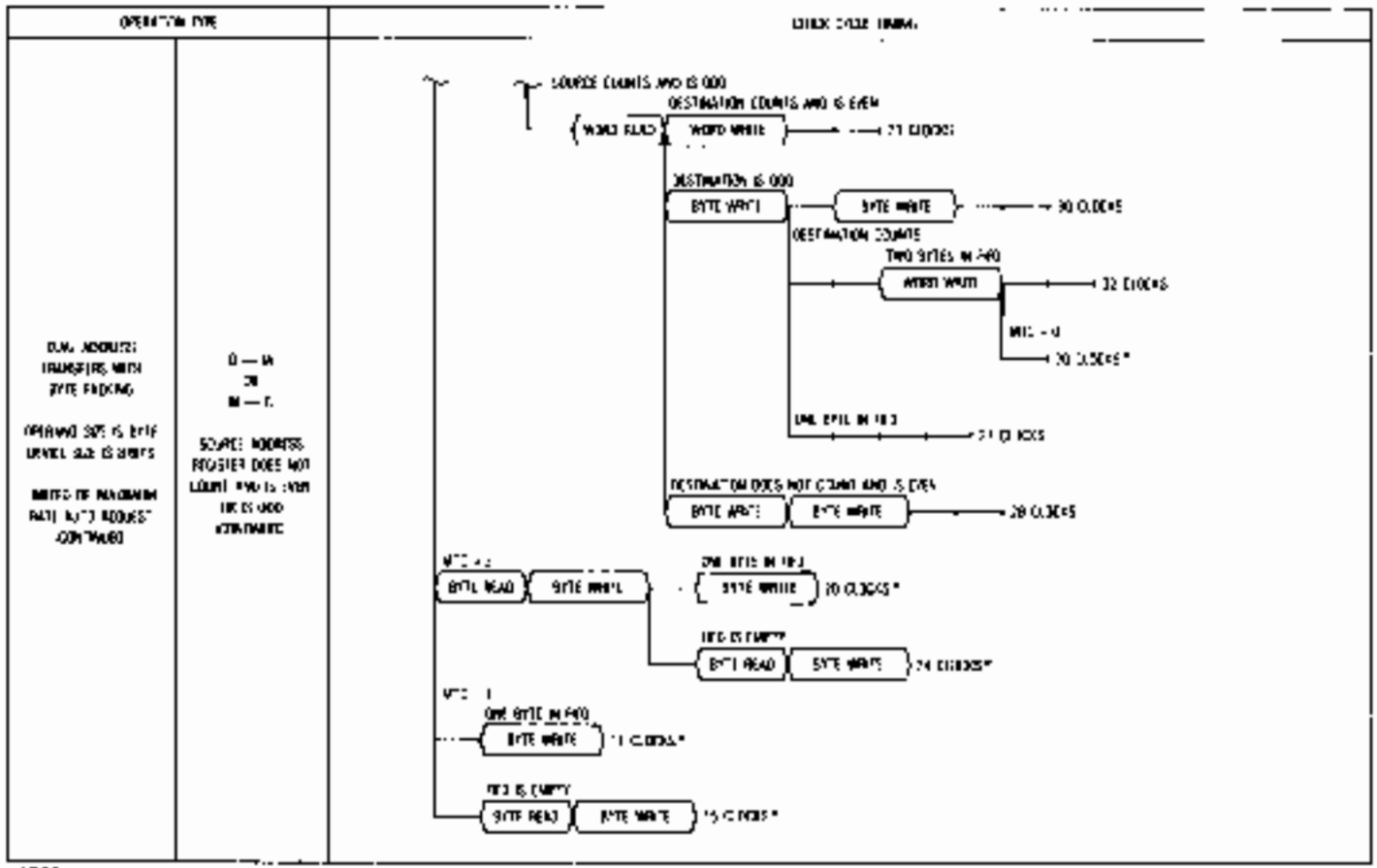


Figure 4-25. DMA Operation Timing Table (Sheet 5 of 7)



NOTES

1. Each "n" mark on a data operation is the end of a clock high period.
2. All read cycles are assumed to take four clocks and all write cycles are assumed to take two clocks. If a system that does not meet these assumptions, the number of wait-state clocks for each bus cycle is added to the total to obtain the actual operation timing.
3. External wait control. The request/acknowledge method does not affect operation timing.
4. The dual-operation text will follow operations marked with an asterisk (*) in the appropriate destination operation.

Figure 4-25 DMA Operation Timing Table (Sheet 7 of 7)

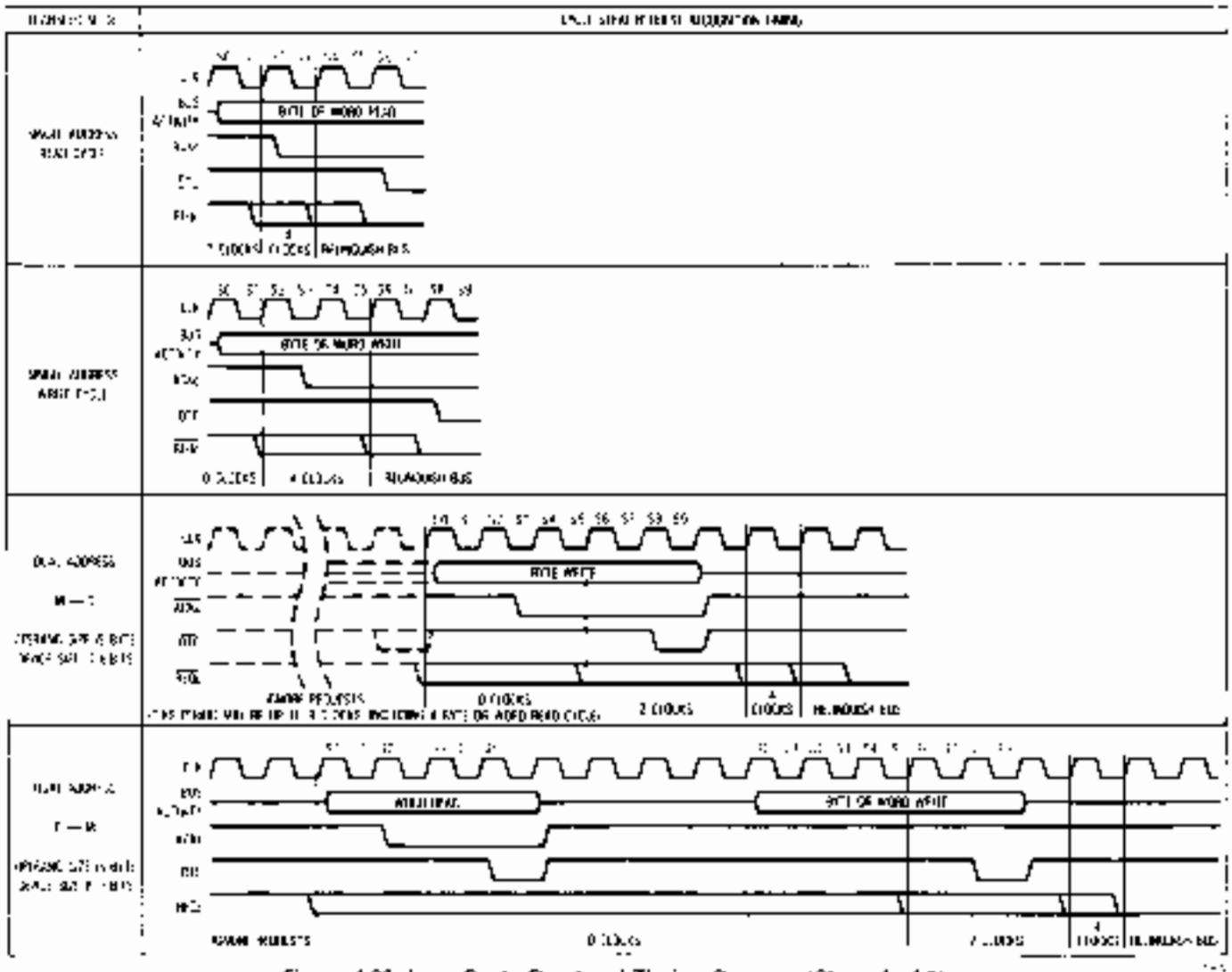


Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 1 of 5)

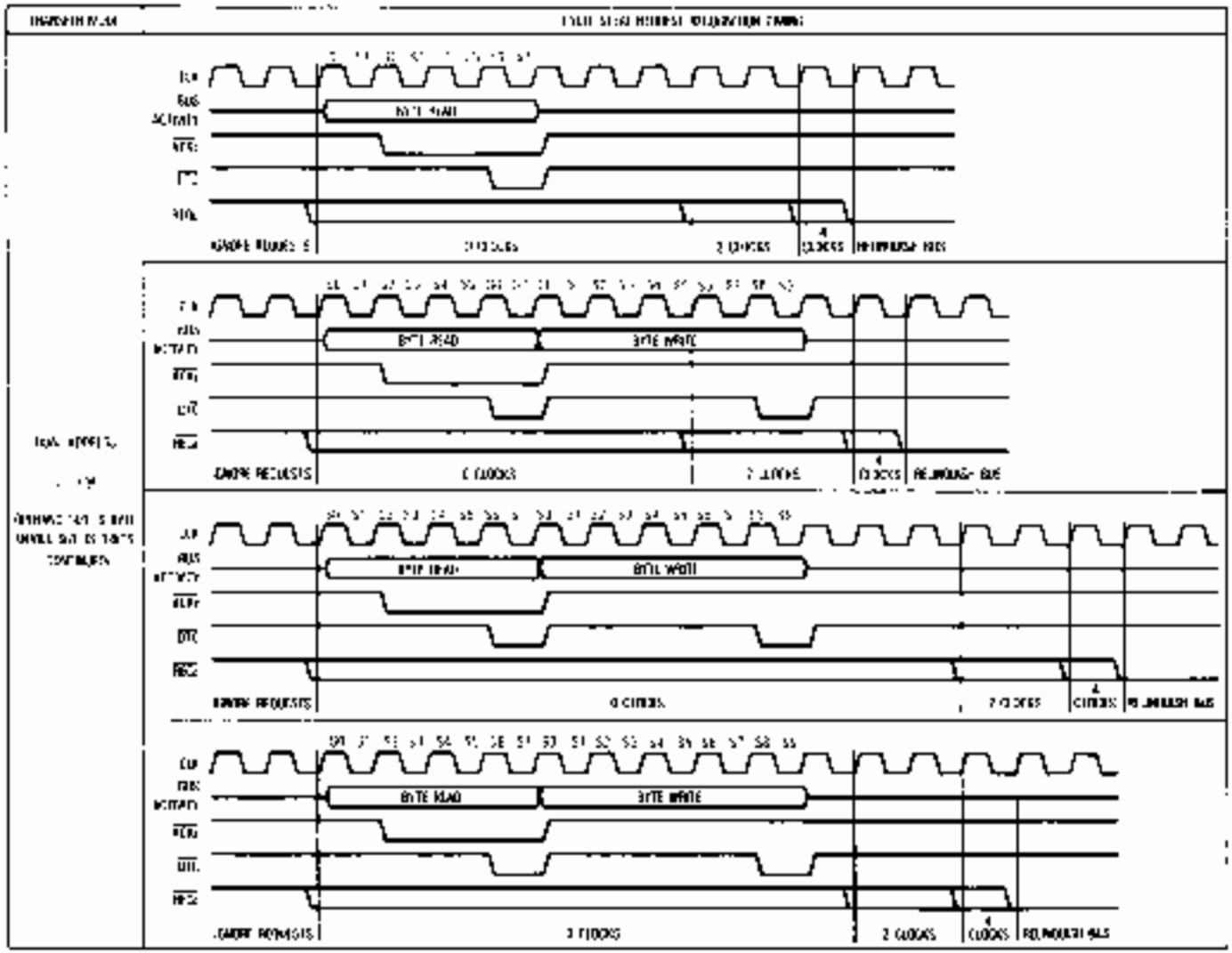


Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 2 of 5)

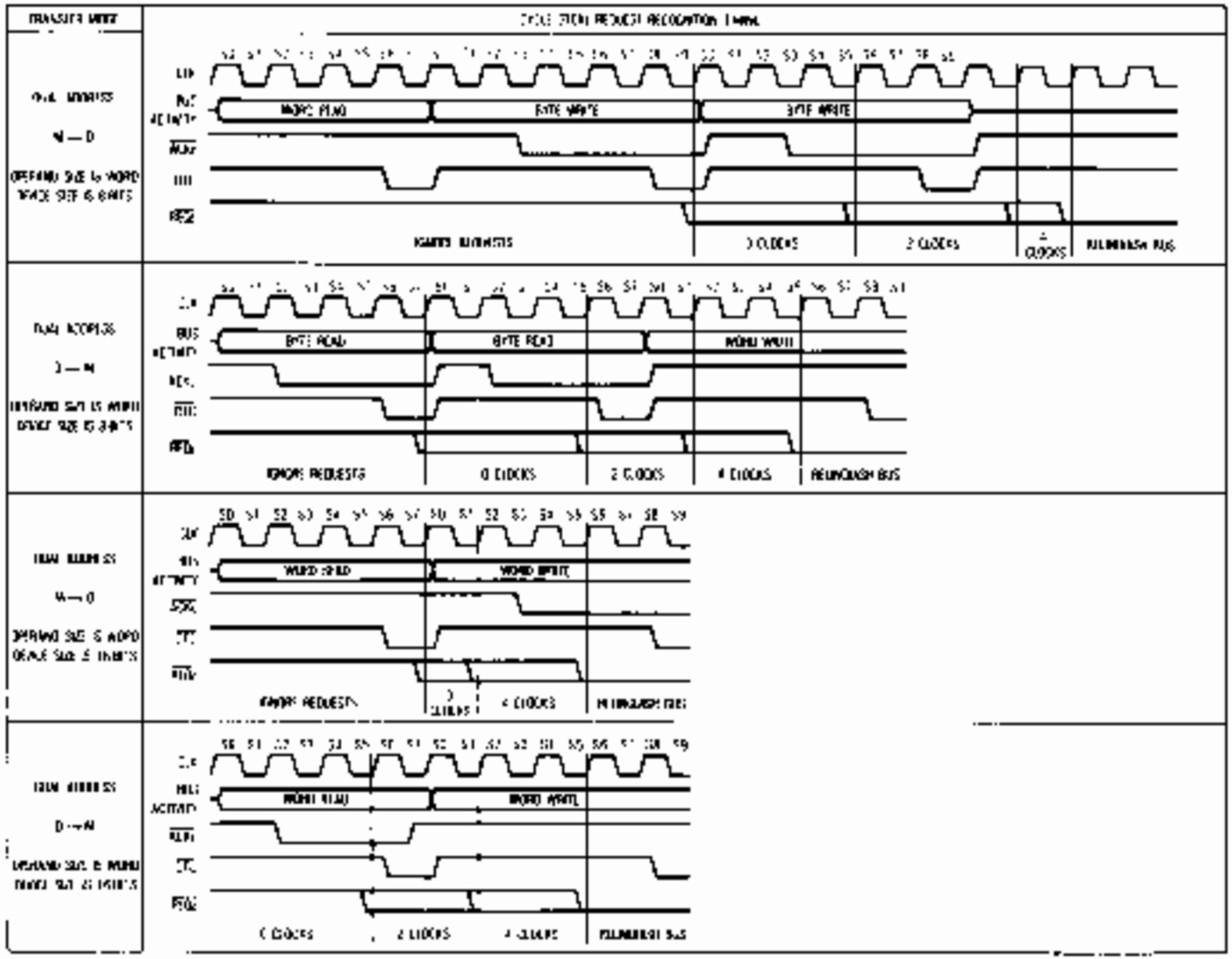


Figure 4-25. Inter-Cycle Overhead Timing Diagram (Sheet 3 of 5)

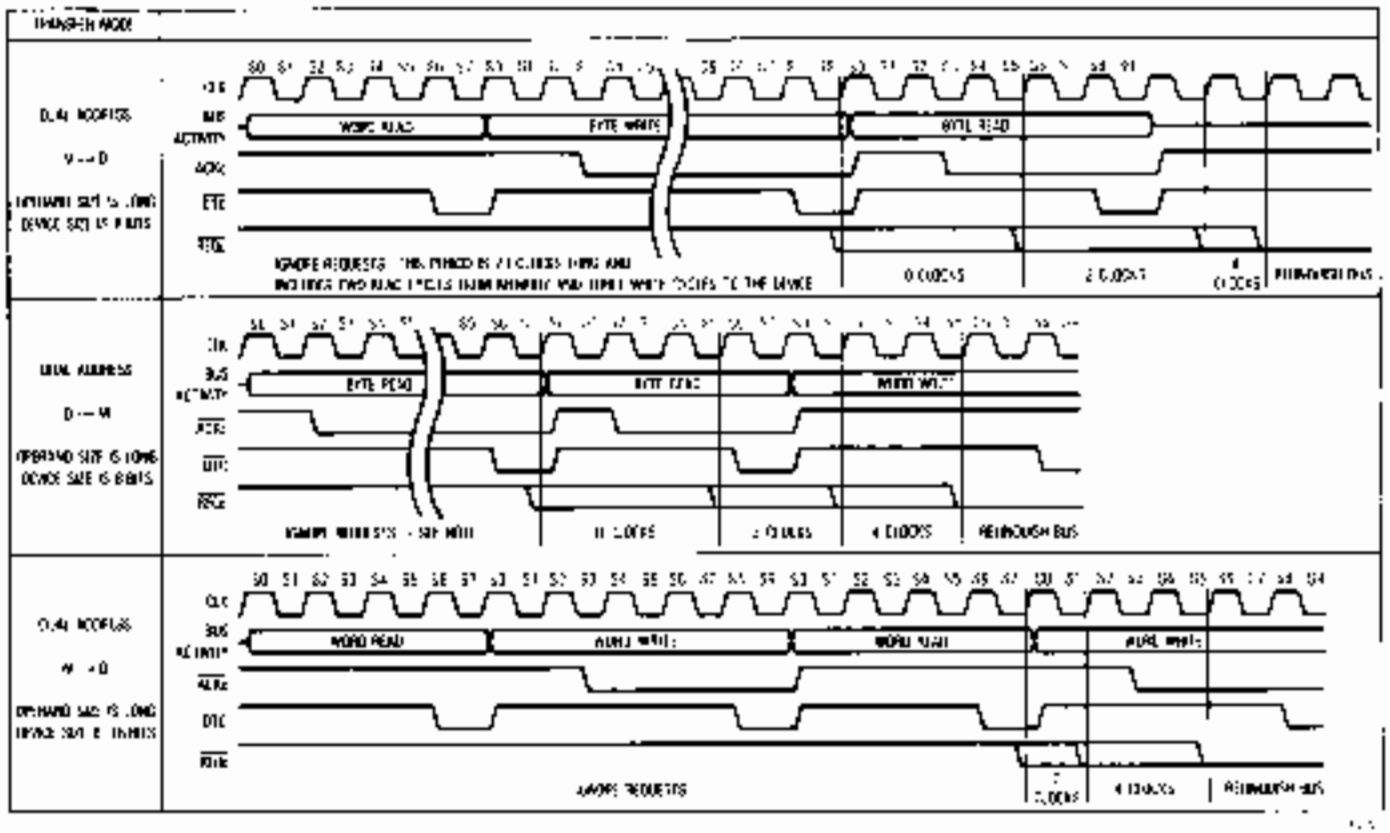


Figure 4-25. Inter-Cycle Overhead Timing Diagram (Sheet 4 of 5)

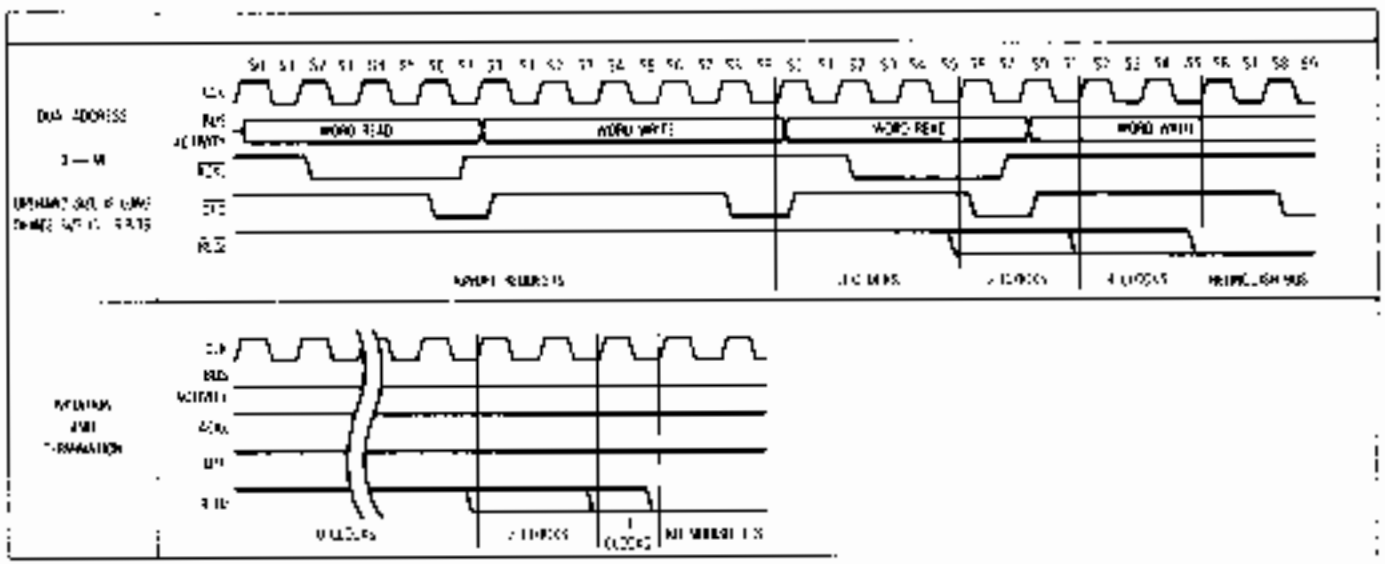


Figure 4-26. Inter-Cycle Overhead Timing Diagram (Sheet 5 of 5)

4.6 ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

Because the M68000 Family of microprocessors uses an asynchronous bus protocol, precise timing considerations are not necessary in order for system components to communicate properly. By using such an asynchronous bus interface, peripherals in the system do not need to have any knowledge of the clock(s) used by bus masters in the system. However, even though the external system bus is asynchronous, any bus master that uses the bus is synchronous internally and must synchronize incoming handshake signals. If it is desired to insure exact timing relationships between a bus master and a peripheral during a bus cycle, it is necessary for the peripheral interface to use the same clock that the bus master uses. In this case, the points at which the bus master asserts outputs and recognizes the assertion of inputs will be precisely known. To a bus master, the type of peripheral interface used is not important, since it will always operate in the same manner; but the design requirements of a peripheral subsystem and the performance of the overall system can be affected by the type of interface that is used.

For a bus master such as the DMAC, there can be two distinct asynchronous or synchronous interfaces operating simultaneously. One is the M68000 Family bus interface to memory or an explicitly addressed peripheral. The second is the device control interface to an implicitly addressed peripheral using the device with acknowledge and ready protocol. An implicitly addressed peripheral using the device with acknowledge only protocol must always operate in synchronization with the DMAC.

4.6.1 Asynchronous Operation

To achieve clock signal independence at a system level, the DMAC can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, and $\overline{BEC0}$-$\overline{BEC2}$) and, as needed, the device control lines ($\overline{RECc}$, $\overline{ACKc}$, $\overline{PCLc}$, $\overline{DTC}$, and $\overline{DONE}$). Using this method, $\overline{AS}$ (and $\overline{ACKc}$ during device accesses) signals the start of a bus cycle and the data strobes (possibly delayed until ready is asserted for single address writes) are used as a condition for valid data on a write cycle. The slave device(s) then responds by placing the requested data on the data bus or latching data from the data bus and asserting $\overline{DTACK}$ (and possibly read for single address reads). The DMAC then acknowledges the successful termination of the bus cycle by asserting $\overline{DTC}$ for one clock cycle and negating $\overline{AS}$, $\overline{ACKc}$, and the data strobes. If no slave responds or an access is invalid, external control logic uses the bus exception control signals to abort or re-run the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle (either single or dual address). The length of time that $\overline{DTACK}$ may precede data during a dual address read is given as parameter #8 and it must be met in any asynchronous system to insure that valid data is latched into the processor. This delay is allowed due to the manner in which the DMAC synchronizes the $\overline{DTACK}$ signal before asserting $\overline{DTC}$ and latching data. The length of time that $\overline{DTACK}$ may precede data during a single address read is dependent on the system design and manner in which the peripheral device latches the read data. In a like manner, the ready signal ($\overline{PCLc}$) can be asserted during a single address read before the peripheral will latch data.

The ready signal ($\overline{PCLc}$) is allowed to be asserted before the data from a peripheral device is valid on a single address write cycle. The length of time that ready may precede data is dependent on the system design such that data will be valid when the memory system latches it. In order to operate in the same manner as an M68000 Family processor, data must be valid prior to the assertion of the data strobe(s).

If a bus exception control condition other than normal is used to terminate a bus cycle, the $\overline{BEC0}$-$\overline{BEC2}$ levels must be valid for specification #96 prior to the assertion of $\overline{DTACK}$ in order for the normal termination to be preempted and the exception action taken. Since a peripheral system will normally generate a single signal that is used by the DMAC interface logic to generate the $\overline{BEC0}$-$\overline{BEC2}$ signals, the delay from the

assertion of the signal by the peripheral to the assertion of the proper $\overline{BEC}$ encoding must be accounted for in order to insure that the setup time to the assertion of $\overline{DTACK}$ is met.

4.6.2 Synchronous Operation

If it is desired to precisely determine the behavior of a system, the relationship between a bus master's clock signal and asynchronous input signals to that bus master must be known. Thus, the system clock may be used in addition to the bus master control signals to generate pseudo-synchronous bus slave handshake signals. In order to allow the system designer to predict when the DMAC will recognize the assertion or negation of input signals, the asynchronous input setup time is given as parameter #6. If an asynchronous input signal makes a transition from one level to the other and is stable for the setup time prior to a rising edge of the clock, that level is guaranteed to be recognized and is valid internally on the next rising edge of the clock. However, the converse is not true — if the input signal changes states and is not stable for the full setup time, the new level is not guaranteed not to be recognized. This means that if a signal is asserted (or negated) inside the setup time window prior to the rising edge of the clock, it may be recognized as negated (or asserted) by the DMAC, depending on several factors such as individual device characteristics, clock waveform and frequency, ambient temperature, supply voltage, and system age.

In order to allow the DMAC to execute minimum length bus cycles while giving the longest possible memory or peripheral access times, $\overline{DTACK}$ and data should be valid in precise relation to the DMAC clock. If $\overline{DTACK}$ is asserted prior to the setup time from the rising edge of S3 for a dual address read cycle, the DMAC will latch the value of the data bus on the falling edge of S6. Thus, if the $\overline{DTACK}$ and data setup times are both guaranteed to be met by the system, then specification #9 may be exceeded.

For single address read cycles, a peripheral device should not latch data before $\overline{DTC}$ is asserted or after the data strobes are negated and thus might use the falling edge of S6 as a latch trigger. Additionally, if the peripheral asserts ready prior to the setup time for the rising edge of S3 and $\overline{DTACK}$ is also asserted by that time, then the falling edge of S6 will occur one and one-half clock cycles later. If it is not known whether or not $\overline{DTACK}$ will meet the setup time to the end of S3, then the peripheral should wait until $\overline{DTACK}$ has been asserted for at least one clock before asserting ready prior to the setup time for the next rising edge of the clock (which will be a wait cycle) so that the rising edge of S6 can be correctly predicted.

During single address write cycles from a device with acknowledge and ready, the device may assert ready ($\overline{PCLK}$) before valid data is on the bus. If ready is asserted prior to the setup time to the rising edge of the clock (either S3 or SDW), then the DMAC will assert the data strobe(s) one and one-half clocks later.

If an implicitly addressed peripheral is used with the device with acknowledge-only protocol, it may operate in a pseudo-asynchronous manner; however, the timing constraints placed on the device will be closely related to the DMAC clock. During a read cycle, the device must be prepared to accept input data when $\overline{DTC}$ is asserted and cannot require a data hold time past the negation of the data strobe(s) one half clock later. During a write cycle, the device must be able to place data onto the bus when the DMAC asserts $\overline{ACK}_x$, with some setup time prior to the data strobe(s) assertion one clock later.

If a BEC encoding other than normal is used to terminate a bus cycle in which $\overline{DTACK}$ is also asserted, then the $\overline{BEC}$ encoding must be recognized as valid on the same clock edge that $\overline{DTACK}$ is first recognized as asserted. Thus, if $\overline{DTACK}$ and $\overline{BEC0}$-$\overline{BEC2}$ are all valid prior to the setup time from the same rising edge of the clock (S3 for a minimum length cycle) the normal termination is guaranteed to be overridden, and the exceptional action is taken.

SECTION 5
OPERATIONAL DESCRIPTION

This section describes the programmable channel functions available on the DMAC, the manner in which data transfer operations are executed, and behavior under exceptional conditions. Also included are descriptions of the general sequence of a channel operation, the organization of operands in memory, and the proper programming sequence used to start channel operations.

Throughout this section references will be made to the DMAC internal registers and bits within those registers. It is suggested that the register summary foldout (Figure 3-2) at the end of this book be kept extended for reference while reading the operational description for the first time to quickly gain familiarity with the register functions.

5.1 GENERAL OPERATING SEQUENCE

Any channel operation will follow the same basic sequence of steps outlined below. The following subsections explain each of these steps in more detail, including how the channel registers should be programmed to select and control various operating modes.

1. Initialization of channel registers by the system processor.

2. Channel Start Up. This includes setting the STR bit in the CCR and recognizing the first operand transfer request (either internally or externally generated). For the chained modes of operation, the first array entry will be fetched to initialize the MAR and MTCR (and to update the BAR for the linked array chaining mode).

3. Transfer Data Block. After a channel is started, it will transfer one operand in response to each request until an entire data block has been transferred (for exceptions to this sequence, refer to **5.4.1 Bus Cycle Sequences**).

4. Chained Operation. If a channel is programmed for one of the chained operating modes, the channel operation may not terminate at the end of a block transfer. In the middle of operation, when a data block transfer is completed, the channel will read an entry in a command table to reprogram the MAR and MTCR for another block transfer, and will optionally read a new value for the BAR that points to the next chain element. Either the value of the BTCR or value of the BAR is used to determine when a channel operation should be terminated.

5. Continue Operation. In unchained operations, if a channel is initialized for the continue mode, a channel may deviate from a strictly sequential block transfer. In the continue mode, when the MTCR decrements to zero, the MFCR, MAR, and MTCR will be loaded from the BFCR, BAR, and BTCR before the next operand transfer request is honored and the channel will continue operation (returning to step 3).

6. Channel Termination. A channel operation can be terminated for several reasons: a peripheral device asserts the $\overline{DONE}$ signal during an operand transfer, the MTCR is decremented to zero, a bus cycle is terminated with a bus error, the system processor sets the SAB bit in the CCR, or an external device asserts an abort input. There are also several illegal operations that may terminate a channel operation.

5.2 CHANNEL INITIALIZATION

In order to start a block transfer operation, the system processor must initialize channel registers with information describing the data block, device type, request generation method, and other special control options. It may be necessary to write into as many as 17 registers to initialize a channel; however, after a channel has been initialized once, only three or four registers may need to be written to start subsequent transfer operations.

5.2.1 Memory and Operand Description

Most DMA transfer operations involve the movement of data into or out of a random-access memory array (although the DMAC also supports device to device transfer operations) and thus, the DMAC must be programmed with the location and size of the data block to be moved. Three registers are used to hold this information: the MAR and MFCR, which provide a 24-bit address and a 3-bit function code to locate data anywhere in one of eight 16 megabyte linear address spaces, and the MTCR which is a 16-bit transfer counter that defines a block size of up to 65 535 operands (bytes, words, or long words). The MTCR is decremented by the DMAC after each operand transfer until it reaches zero, and the MAR may be programmed to be decremented or incremented after each operand transfer.

Although the DMAC does not place any format requirements on the data that is transferred during a channel operation (other than the format of the command tables used during chained operations), it does use the M68000 conventions for operand alignment and byte significance. As shown in Figure 5-1, bytes are individually addressable with the high-order byte having an even address the same as the word. The low-order byte has an odd address that is one count higher than the word address. Word and long-word

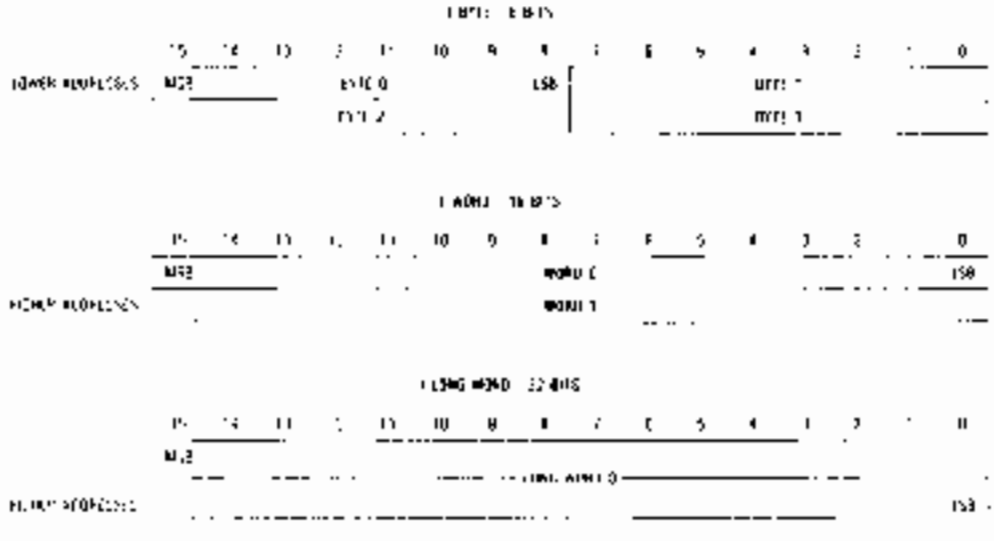


Figure 5-1. Memory Data Formats

data is accessed only on word (even byte) boundaries, with the lowest address byte being most significant. The size field in the DCR is used to select the operand size for a block transfer.

5.2.2 Device Description

Four types and two sizes of peripheral devices are supported by the DMAC. The types supported are explicitly addressed M68000 bus compatible, explicitly addressed M6800 bus compatible, implicitly addressed with acknowledge only, and implicitly addressed with acknowledge and ready. The type and size of a device is programmed in the DCR, and the address and address space of an explicitly addressed device is programmed into the DAR and DFCR.

A device port may be 8 or 16 bits wide. If a 16 bit wide, explicitly addressed device is to support byte operand sizes, the data port must be accessible as two bytes as well as a word. For all other legal combinations of port and operand sizes, the device port will always be accessed full width (i.e., an 8-bit port accessed as a byte and a 16-bit port accessed as a word). If a memory-to-memory transfer is to be executed by the DMAC, the device is programmed as M68000 compatible, with a 16 bit port size.

The functional definition of the PCL input is also part of the device description. If the transfer protocol selected is implicitly addressed with acknowledge and ready, the PCL input is used as an active low ready signal. If the device type selected is M6800, the PCL input is used as the E clock for the synchronous bus cycles. Otherwise, the DCR may be programmed to select one of four other functions for the PCL line: a status, interrupt, or abort input or a start pulse output. As a status input, the level of the PCL line can be read at anytime through the PCS bit in the CSR, regardless of any other function associated with it. Also, in all modes, a high-to-low transition on the input will set the PCL transition bit (PCT) in the CSR and that bit can be read at any time. If the status with interrupt function is selected, the PCT bit is set and interrupts are enabled, the DMAC will assert an interrupt request to the system processor. If the abort function is selected, the channel operation will be terminated when the PCT bit is set and the CSR will reflect the aborted operation. As a start pulse output, the PCL line will be driven low for eight clocks when the channel is started, and will remain high otherwise.

The DMAC supports peripheral device sizes of 8 or 16 bits using either single or dual address transfers of byte, word, or long word operands. When all combinations of the above are considered, 12 different transfer operations are possible, however, four of those operations are not legal. Table 5-1 shows a matrix of the possible device, operand size, and transfer type combinations, and whether or not each is legal.

Table 5-1. Device and Operand Size Combinations

	Dual Address				Single Address	
	Byte Operand	Word Operand	Long Operand	Byte Operand	Word Operand	Long Operand
8 Bit Device	*	*	*	*	X	X
16 Bit Device	*	*	*	*	X	X

X Denotes Illegal Combinations

*Illegal Only when Internal Memory Buffer Location Is Accessed by Auto-Request Operation

5.2.3 Operation Description

A wide variety of block transfer operations are supported by the DMAC by choosing from the available device sizes, bus transfer protocols, request generation methods, and special operating modes. The following paragraphs discuss the channel operation functions that can be programmed to facilitate various block transfers.

5.2.3.1 TRANSFER DIRECTION. The DIR bit in the OCR is used to indicate the direction of the block transfer, either from memory to the device or vice versa. During single address transfers, the R/$\overline{W}$ signal will be driven high if the direction is from memory to device and driven low if the direction is device to memory. This is identical to a normal read or write cycle for the memory, but exactly opposite for the peripheral, since a high level indicates a write to the peripheral and a low level indicates a read from the peripheral.

During dual-address transfers, the DIR bit determines whether the MAR or DAR is used as the source pointer or destination pointer. For memory-to-device transfers, the value in the MAR (and MFCR) will be used during read cycles and the value in the DAR (and DFCR) will be used during write cycles. The opposite relationship is true if the DIR bit indicates a device-to-memory transfer.

5.2.3.2 ADDRESS SEQUENCING. The manner in which the MAR and DAR are incremented or decremented during a transfer operation depends on the programming of the SCR, operand size, device port size, and transfer protocol used. Both the MAR and DAR can be individually programmed to be incremented or decremented after a successful operand transfer or to remain unchanged.

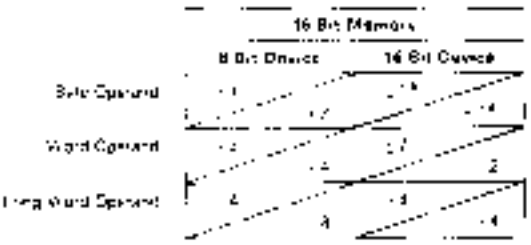
When using the single-address transfer protocol, the operand size must always be equal to the device port size. Thus, all operands can be transferred in one bus cycle and only the MAR is incremented or decremented (the DAR is not used). The amount by which the MAR is incremented or decremented will be one or two for byte or word operands respectively (if the MAC field in the SCR is programmed for increment or decrement). If no increment has been programmed for the MAR, all bus cycles will use the same address.

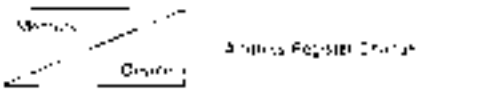
When using the dual-address transfer protocol, the DMAC will always run at least two, and up to six bus cycles to transfer each operand. When the operand size is larger than either the memory width or device port size, the DMAC will run multiple bus cycles to transfer operand parts until the entire operand has been transferred. If the source (memory or device) is smaller than the operand size, two or four read cycles will be executed to fetch each operand; and, if the destination (device or memory) is smaller than the operand size, two or four writes cycles will be executed to store each operand.

One exception to the above rules is when the combination of a 16-bit device and byte operand size is used with internal request generation. In this case, the DMAC maximizes bus utilization by transferring two operands with each bus cycle if possible. In other words, most bus cycles during such a transfer will be word read or write cycles to transfer two operands (bytes) at a time.

If the SCR is programmed to increment or decrement for either the MAR or DAR, the manner in which the address register(s) is incremented or decremented will depend on whether the access is to memory or the device, and the operand-port size combination. Table 5-2 shows how the MAR and DAR will be incremented for all combinations of operand and device sizes in the dual-address mode. It should be noted that when an 8-bit device is used, the DAR will be incremented or decremented by two, four, or eight for byte, word, or long-word operands respectively, and consecutive odd or even addresses will be accessed. This is consistent with the M68000 MOVEP instruction, so that an 8-bit device may be placed on one half of a 16-bit bus and successive bytes within the device will be accessed. If no increment is selected for the DAR, and word or long-word operands are to be transferred to or from an 8-bit device, successive bytes will be accessed in the same manner as just described, but the value of the DAR will not be changed; thus each operand transfer will start at the same address.

Table 5-2. Address Register Sequencing Rules





If an exceptional condition occurs during an operand transfer that preempts normal termination, no address registers will be incremented. Also, any data that was previously read from the source (memory or device) into the internal holding register will be lost when a bus error occurs. Thus, if it is desired to continue a channel operation after a write cycle is aborted due to a bus error (for example, after a page fault in a virtual memory system), the system processor may simply restart the channel and the aborted operand transfer will be attempted again. The MTCR will not be changed until all bus cycles required to transfer an operand have completed normally, so it will not need to be modified to continue the operation.

5.2.3.3 TRANSFER REQUEST GENERATION. In order for a peripheral to control the transfer of operands to or from memory in an orderly manner, it uses an input signal to the DMAC as a service request. If a memory-to-memory transfer or a peripheral with no request signal is used, the DMAC can internally generate operand transfer requests. If an internal request generation method and dual-address transfer protocol is used, the DMAC will not assert $\overline{ACK}$ when performing device accesses. Thus, a channel can be used both for transfers between a device (either implicitly or explicitly addressed) and memory using external request generation, and for memory to memory transfers using internal request generation without conflicting with the operation of the device.

The request generation method used for a channel is programmed in the DCR. If an external request method is used, the type is further defined by the DCR. If the internal limited rate method is used, the BCR is used to determine the timing constants for bus utilization calculations. If the auto-start external request generation method is used, the first request is generated internally when the channel is started; subsequent requests are generated using the selected external request method.

5.2.3.3.1 Internal, Maximum Rate. The simplest method of internal request generation is to always have a transfer request pending for a channel until the transfer count is exhausted. If this method is used, as soon as a channel is started the DMAC will arbitrate for the bus and begin to transfer data when it becomes bus master. If no exceptional conditions occur, all operands in the data block will be transferred in one burst, so that the DMAC will use 100% of the available bus bandwidth.

5.2.3.3.2 Internal, Limited Rate. In order to guarantee that the DMAC will not use all of the available system bus bandwidth during a transfer, internal requests can be generated according to the amount of bus bandwidth allocated to the DMAC. This method of request generation is called limited rate auto-request (LRAR) and uses three constants programmed via the DCR to monitor bus activity and allow the DMAC to use a portion of the bus bandwidth.

One constant is a sample period measured in clock cycles. During each sample period the DMAC monitors the $\overline{RGACK}$ signal as an input and counts the number of clocks during which it is asserted. After a sample period has ended, the number of clocks during which $\overline{BGACK}$ was asserted by any bus master (including the DMAC) is compared to a second programmable constant called the burst time. If the previous sample count is less than the burst time, then the DMAC will generate internal requests for any channel programmed for LRAR during the burst time for the current sample period. In this manner, if the DMAC or other bus master asserts $\overline{BGACK}$ for more than the burst time during any sample period, no limited rate requests will be generated during the next sample period so that the system processor will be able to use the bus. (This assumes that the system processor does not assert $\overline{BGACK}$ when it owns the bus.) The third programmable constant, which determines the amount of the bus bandwidth available to DMA devices, is the ratio of the sample period length to the burst time length.

Of the three constants used in the LRAR equation, two are programmed directly into the GCR and the third is calculated from the other two. The fraction of the bus bandwidth available to the DMAC is equal to $2^{-(BR-1)}$, the burst time is equal to $2^{(BT+BR+5)}$. Figure 5-2 depicts graphically how the three constants are used to determine bus utilization by the DMAC. Table 5-3 lists all combinations of BT and BR and the resulting MPU, burst and sample period lengths for each.

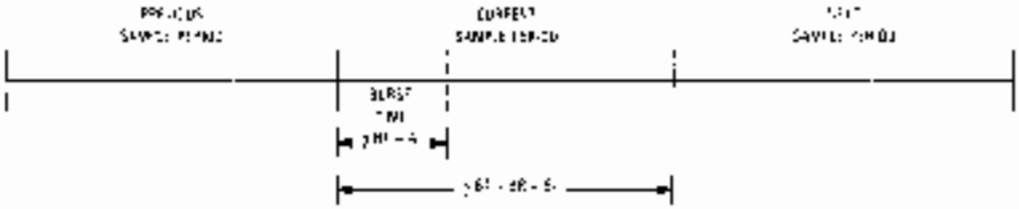


Figure 5-2. Limited-Rate Auto-Request Operation Timing Parameter Relationships

Table 5-3 Limited-Rate Auto-Request Timing Parameter Value Combinations

BR	BT	Burst Time*	MPU Period*	DMA Ratio	Sample Period*
00	00	16	16	50.00%	32
00	01	32	32	50.00%	64
00	10	64	64	50.00%	128
00	11	128	128	50.00%	256
01	00	16	48	25.00%	64
01	01	32	96	25.00%	128
01	10	64	192	25.00%	256
01	11	128	384	25.00%	512
10	00	16	112	12.50%	128
10	01	32	224	12.50%	256
10	10	64	448	12.50%	512
10	11	128	896	12.50%	1024
11	00	16	240	6.25%	256
11	01	32	480	6.25%	512
11	10	64	960	6.25%	1024
11	11	128	1920	6.25%	2048

*Maximum [illegible] DMA clock cycles

5.2.3.3.3 External, Burst Mode. For peripheral devices that require very high data transfer rates, the burst request mode allows the DMAC to use all of the bus bandwidth under control of the device. In the burst mode, the $\overline{REQ}$ input to the DMAC is level sensitive and sampled at certain points to determine when a valid request is asserted by the peripheral. The device requests service by asserting $\overline{REQ}$ and leaving it asserted. In response, the DMAC arbitrates for the system bus and begins to perform an operand transfer. During each operand transfer, the DMAC will assert $\overline{ACK}$ to indicate to the device that a request is being serviced. If $\overline{REQ}$ is asserted when the DMAC asserts $\overline{ACK}$ and remains asserted until the rising edge of the clock on which DTC is asserted, then a valid request for another operand transfer is recognized and the DMAC will service that request immediately if a higher priority channel does not have a pending request. If $\overline{REQ}$ is negated before the rising edge of the clock one clock cycle before DTC is asserted, a new request will not be recognized and other channels may be serviced, or the DMAC will release ownership of the bus and return to the idle state.

5.2.3.3.4 External, Cycle Steal. For devices that generate a pulsed signal for each operand to be transferred, the cycle steal request generation mode uses the $\overline{REQ}$ pin as a falling edge sensitive input. The $\overline{REQ}$ pulse generated by the device must be asserted during two consecutive falling edges of the DMAC clock to be recognized as valid; thus, if the peripheral generates it asynchronously, it must be at least two clock periods long. The DMAC will respond to cycle steal requests in the same manner as all other requests; however, if subsequent $\overline{REQ}$ pulses are generated before $\overline{ACK}$ is asserted in response to each request, they will be ignored. If $\overline{REQ}$ is asserted after the DMAC asserts $\overline{ACK}$ for the previous request and before DTC is asserted, then the new request will be serviced before bus ownership is released. If a new request has not been generated for any channel by the time DTC is asserted, the bus may be released to the next master.

5.2.3.3.5 External, Cycle Steal with Hold. In order to reduce the latency time required for the DMAC to respond to a cycle steal request, the DMAC can retain bus ownership for a short time after an operand transfer in anticipation of a new request within that time period. This eliminates bus arbitration delays, at the expense of wasting some bus bandwidth due to idle "hold" periods.

This request generation method is identical to external, cycle steal (refer to **5.2.3.3.4 External, Cycle Steal**) with the exception that the DMAC will retain bus mastership for one full sample period (as defined by the values programmed into the GCR, refer to **5.2.3.3.2 Internal, Limited Rate**) and then relinquish the bus if a new request has not been received. The bus hold time is one full sample period. Thus, the DMAC will wait until the end of two sample periods before giving up the bus. If a request is recognized on a higher priority channel during the hold time, that channel will be serviced and then the DMAC will relinquish the bus if a new request is not present for the cycle steal with hold channel. If a request is received during the hold time, a bus cycle to service the request will start within six clocks.

Note that the DMAC will hold the bus for one full sample period after the last operand transfer of a channel operation, even though no further requests can be serviced.

5.2.3.4 CHANNEL PRIORITY. Each channel has a priority level, which is determined by the contents of the channel priority register (CPR). The priority of a channel is a number from zero to three, with zero being the highest priority level. When multiple requests are pending on the DMAC, the channel with the highest priority receives first service. The priority of a channel is independent of the device protocol or the request mechanism for that channel. If there are several requesting channels at the highest priority level, a round-robin arbitration scheme is used, such that the least recently serviced channel will be the next one to be serviced. This priority scheme is also used when multiple channels have pending interrupts and the MPU executes an interrupt acknowledge.

5.2.3.6 INTERRUPT OPERATION In order for the system processor to determine the status of a channel, it may read the CSR at any time (assuming that it can gain control of the DMAC bus) or the DMAC can be programmed to generate an interrupt to inform the processor of certain events. Four registers take part in the interrupt operation and response structure of the DMAC: the GCR, CSR, NIVR, and EIVR.

The interrupt bit in the CCR is used to enable a channel to generate interrupts via the $\overline{IRQ}$ signal. If a channel is enabled to generate interrupts and the COC, BTC, or PCT (with PCL programmed as a status with interrupt input) bit is set in the CSR, then the $\overline{IRQ}$ signal will be asserted. As long as a valid interrupt condition is present for any channel, the $\overline{IRQ}$ signal will be asserted, and it is the responsibility of the system processor to clear the appropriate bit(s) to negate the interrupt (refer to **3.7.1 Channel Status Register**).

In response to the assertion of $\overline{IRQ}$, an M68000 Family processor will execute an interrupt acknowledge bus cycle to read a vector number that is used to locate the interrupt handler routine. External hardware may be used to return a vector number to the processor during the interrupt acknowledge cycle; however, the DMAC efficiently supports the M68000 interrupt structure by supporting two separate vector numbers for each channel and using internal priority and status information to select the proper value to be returned.

In order to cause the DMAC to return one of its vectors to the processor, the $\overline{IACK}$ input is asserted during the interrupt acknowledge cycle. If the DMAC is asserting $\overline{IRQ}$ when $\overline{IACK}$ is asserted, it will return a vector number from the highest priority channel that has an interrupt condition present. The vector number that is returned depends further on the error (ERR) bit in the CSR of the highest priority channel. If ERR is clear, then the value in the NIVR will be returned; otherwise, the value of the EIVR will be returned. If $\overline{IRQ}$ is negated when $\overline{IACK}$ is asserted, the $\overline{IACK}$ will be ignored.

5.3 CHANNEL START-UP

Once a channel has been initialized with all parameters required for a transfer operation, it is started by writing a one with a zero write to the STR bit in the CCR. This write cycle will not leave a one in the STR bit position, but the ACT bit in the CSR will be set to indicate that the channel has been started. After the channel has been started, any register that describes the current operation may not be modified or an operation timing error will occur. The registers that may not be modified are the DCR, OCR, SCR, MTCR, MAR, MFCR, DAR, and DFCR. All of the other channel registers may be modified if desired during a channel operation; thus, status bits can be cleared in the CSR, the operation can be controlled via the setting bits in the CCR, and the channel priority level and interrupt vector numbers may be modified through the CPR, NIVR, and EIVR. The GCR can always be modified so that DMA bus utilization may be dynamically controlled.

In order to assure that status information from previous transfer operations has been read by the system processor before starting a new operation, certain status bits in the CSR must be clear when the STR bit is set. The status bits that must be clear are the COC, BTC, NDT, and ERR bits. Thus, part of the normal initialization and start-up procedure is to clear the CSR before the STR bit is set. If any of the above mentioned bits are set when an attempt is made to set the STR bit, an operation timing error will occur.

If the channel operation being started is the first block of a transfer using the continue mode, the CNT bit in the CCR may be set by the same write cycle that sets the STR bit; however, CNT must not be set prior to setting STR. Also, the BTC bit in the CSR must be clear before the CNT bit is set, for the first or subsequent block(s) in the transfer operation.

Once the STR bit has been set, the channel will become active and accept operand transfer requests. When the first valid request is recognized, it will enter the DMA mode of operation by

arbitrating for the bus. It should be noted that the $\overline{REQ}$ input is ignored until the channel is started, so that the channel will not recognize transfer requests from the device until it is made active. The PCL line can be programmed as a start pulse output that will signal an external device when a channel is ready to accept requests. The PCL input, however, is not ignored before the channel is started. If the PCL line is programmed as a status input and a high-to-low transition occurs on it at any time, the PCT bit in the CSR will be set. Thus, an interrupt may be generated even when the channel is not active.

In order to begin a data transfer operation, the following general sequence for programming the DMAC should be followed:

1. Write a one to the SAB bit and a zero to the INT bit in the CCR. This will cause any previous operation to be terminated and prevent the channel from generating an interrupt request. If the previous state of a channel is known, this operation is not necessary.

2. Load the channel registers with parameters describing the memory, device, and operation to be performed. If interrupts are to be generated by the channel operation, the MIVR and EIVR should be loaded with the appropriate vector numbers and the corresponding locations in the exception vector table should be loaded with the start address of the interrupt handler routines. If the continue or chaining modes of operation are to be used, the **BFCR**, **BAR**, and **BTCR** must also be loaded.

3. Clear any previous status information by writing a $F6 (or $FF) to the CSR.

4. Start the channel by writing a one to the STR bit of the CCR. The **CNT** and **INT** bits may also be set by the same write cycle, if desired, in order to start a continued operation and enable interrupts. Also note that the **SAR** and **HLT** bits should be written as zero in order to avoid an immediate abort or halt operation.

5. Write to the peripheral device control registers as necessary to enable the device to generate requests and begin the transfer operation. This operation may be performed before the **DMAC** channel is started if internal request generation is used, depending on the requirements of the device.

5.4 DATA TRANSFERS

When the DMAC is performing a data transfer operation, several factors affect the order in which bus cycles will be run and when each cycle starts in relationship to other operations. The following paragraphs describe the bus cycle sequence for each of the transfer protocols and the effects of simultaneous channel operations, bus exceptions, and special operations on that sequence.

5.4.1 Bus Cycle Sequence

The order in which the **DMAC** executes bus cycles during a transfer operation depends primarily on the transfer protocol used and the device and operand size combination. The following descriptions of the sequence used by each protocol assumes that only one channel is active during the operation.

5.4.1.1 SINGLE ADDRESS TRANSFERS The sequence used for this protocol is very simple. When a request is recognized internally, the DMAC will arbitrate for the bus if necessary and execute one bus cycle in response to each request. Since the operand size must match the device port size for single address transfers and only one bus cycle is run for each request, the number of normally terminated bus cycles

executed during a transfer operation will always be equal to the value programmed into the MTCR (unless the transfer is terminated early by the peripheral device). The sequencing of the address bus will follow the programming of the MAC field in the SCR. If programmed for no increment, all of the bus cycles will access the same, initial address. If programmed to count up or down, each successive bus cycle will access the address one operand size beyond or before the last cycle in a linear fashion.

5.4.1.2 DUAL ADDRESS TRANSFERS. Each operand transfer in the dual address mode will require at least two and as many as six bus cycles in response to each operand transfer request. Table 5-4 shows the number of bus cycles required to transfer each operand for the eight memory, device, and operand combinations.

Table 5-4. Bus Cycles per Operand Transfer-Dual Address

	16-Bit Memory	
	8 Bit Device	16 Bit Device
Byte Operand	2*	2*
Word Operand	3	2
Long Word Operand	6	4

*Packing will be performed such that two operands will be transferred in 2 cycles.
**Operand size of byte is used for FIFO mode (packing option must be disabled).

For the combinations that require two or four bus cycles per operand, an equal number of cycles will access memory and the device, for the three and six bus cycle cases, two cycles will access whichever port is smaller and one cycle will access the larger port. The bus cycle sequence will always follow the pattern of: read a byte or word using as many bus cycles as are required and then write that operand part, again using as many bus cycles as are required; this process is then repeated for long-word operands.

The address register and transfer count register sequencing during a dual address transfer will follow the rules discussed previously for increment/decrement amounts except for the case discussed in the next paragraph. An address register will be incremented or decremented only after the entire operand has been successfully transferred. The MTCR will be decremented by one when the last bus cycle of the operand transfer has been completed successfully.

For most combinations of memory device, and operand size, the number of bus cycles run for each operand transfer count will be two, three, or four, according to Table 5-4, regardless of the operand request generation method used or transfer count value. One combination that is not consistent with this rule is when an operand size of byte is used to transfer data between memory and a 16 bit device using minimal request generation. In this case, the DMAC will perform the transfer in order to best utilize the available bus bandwidth by performing word read and write cycles when possible.

Once the DMAC has started a dual address operand transfer, it must complete that transfer before releasing ownership of the bus or servicing requests for another channel of equal or higher priority, unless one of the bus cycles during the transfer is terminated with a relinquish and retry operation. When a bus cycle is terminated with a halt, the DMAC will halt operation and release ownership of the bus immediately following the termination of that bus cycle. When the $\overline{BEC}$ encoding is returned to normal and bus ownership is returned to the DMAC, any operand transfer previously suspended must then be completed, regardless of pending requests for the other channels. The same rule applies to cycles terminated with retry or relinquish and retry; the same cycle that was terminated with the retry will be run again before the other channels can be serviced. If a bus cycle is terminated with a bus error, the channel operation will be terminated and any portion of the operand that was a read but not yet written to the destination will be lost.

5.4.1.3 TRANSFER COUNT SYNCHRONIZATION. An important consideration in any system where two devices (the DMAC and a peripheral) are transferring a fixed number of data items is that both devices count the same number of operand transfers under all conditions. The DMAC facilitates this count synchronization with the $\overline{DTC}$ signal in two ways. First, $\overline{DTC}$ always indicates to a device that a bus cycle to transfer an operand or part of an operand has been successfully completed so that the device may update its transfer pointer (for an internal FIFO, for example). Second, if the device counts the number of assertions of $\overline{DTC}$ and knows how many device bus cycles are required to transfer each operand, it can properly decrement its transfer counter in step with the DMAC MTCR. A device should never use $\overline{ACK}$ by itself to clock pointer and counter registers if it is desired to support exceptional bus conditions. This is due to the fact that $\overline{ACK}$ will be asserted at the beginning of a bus cycle to the device, but if a bus cycle is terminated with a bus error, retry, or relinquish and retry, $\overline{DTC}$ will not be asserted.

5.4.2 Effects of Channel Priority

When all the channels of the DMAC are active, there exists the possibility that operand transfers will be interleaved due to the priority scheme used to resolve simultaneous pending requests. The effects of channel priority on the operand transfer sequence during simultaneous channel operations is independent of the request generation method or transfer protocol used. In all cases, the following rules apply to the operand transfer order used:

1. Once a channel has started the transfer of an operand, the entire operand must be successfully transferred or the channel operation aborted before the other channels may be serviced, regardless of priority relationships. The halt, retry, and relinquish and retry exceptional bus conditions will not affect the order in which operands are transferred, only the order of bus cycles during an operand transfer or the delay between successive bus cycles. For example, if a lower priority channel is being serviced and a bus cycle is terminated with a relinquish and retry, when the DMAC regains bus ownership, the lower priority operand transfer will be completed even if a higher priority request is pending. This is to preserve the contents of the internal data holding register.

2. If two channels are at different priority levels, the lower priority channel will not be serviced as long as higher priority channels have a request being serviced or pending.

3. If two or more channels are at the same priority level and have pending requests, the last channel serviced will be the last channel serviced again. For example, if two channels use internal maximum rate request generation and are active simultaneously, successive operand transfers will service alternate channels.

4. When the memory transfer counter (MTC) is exhausted, the channel operation will continue if the channel is set to a chaining mode and the chain is not exhausted. The reading of the next descriptor from the array is treated as an operand transfer and must be completed before any higher priority requests can be serviced.

5.4.3 Exceptional Bus Conditions

In any computer system, there always exists the possibility that an error will occur during a bus cycle due to a hardware failure, random noise, or because an improper access is being attempted. When an asynchronous bus structure, such as the one supported by the MC68000 Family, is used in a system, it is very simple to make provisions to allow bus masters to detect and respond to errors during a bus cycle. The DMAC uses a three bit, binary encoded input bus ($\overline{BEC0}$-$\overline{BEC2}$) to allow external hardware to indicate when an abnormal condition has occurred. The following paragraphs describe the exceptional conditions

that can be detected by the DMAC and their effects on a data transfer operation. Refer to **4.4.2 Bus Exception Control Functions** for a detailed description of the operation of the $\overline{BEC}$ pins and the encodings used to indicate various conditions.

5.4.3.1 RESET. In the event of a catastrophic system failure, including recovery from a power-down state, the DMAC may be returned to an uninitialized state by asserting the reset encoding on the $\overline{BEC}$ pins. Regardless of any current operation being performed, the DMAC will immediately abort all channel operations and return to the idle state. In the event that a bus cycle is in progress when reset is detected, it will be terminated in an orderly fashion, the control and address/data pins will be three-stated and bus ownership released.

5.4.3.2 HALT. Channel transfer operations may be temporarily suspended at any time by asserting halt to the DMAC. In response, any DMA bus cycle that is in progress will be completed (when $\overline{DTACK}$ and/or ready is asserted) and bus ownership will be released. No further bus cycles will be started as long as halt remains asserted. The system processor may access the DMAC internal registers to determine channel status or alter operations. When halt is negated, if the DMAC was in the middle of an operand transfer when halted or if there is a transfer request pending, it will arbitrate for the bus and continue normal operation.

5.4.3.3 BUS ERROR. When a fatal error occurs during a bus cycle, bus error is used to abort the cycle and terminate a channel operation. When bus error is asserted during a bus cycle, it will terminate in an orderly fashion, but the channel that was being serviced by that bus cycle will have the current operation terminated with an error signalled in the CSR. When the bus cycle is terminated, the DMAC will retain ownership of the bus as long as bus error remains asserted and will not start any new bus cycles to service the other channel. When bus error is negated, a bus cycle will be started to service the other channel, if a transfer request is pending; otherwise, bus ownership will be released.

5.4.3.4 RETRY OPERATIONS. If a correctable error occurs during a bus cycle, it is often desirable to cause the bus cycle to be terminated and then re-executed at a later time. The DMAC supports two such operations that perform the same function, but in different ways: retry, and relinquish and retry. When either retry signal is asserted during any DMA bus cycle, that cycle is terminated in an orderly fashion, but no data transfer operation takes place. When the retry signal is later negated, the same bus cycle that was terminated with the retry will be executed again, using the same address, data, and control information. If the other channel has a higher priority request pending, it will not be serviced before the retry has been executed and the entire operand for the current channel has been transferred.

5.4.3.4.1 Retry. When the retry encoding is asserted during a bus cycle, the DMAC will terminate the cycle and suspend any further bus cycles until retry is negated, while retaining ownership of the bus. When retry is negated, the bus cycle will be executed again and normal operations will be continued.

5.4.3.4.2 Relinquish and Retry. When the relinquish and retry encoding is asserted during a bus cycle, the DMAC will terminate the cycle, release ownership of the bus, and suspend any further operations until relinquish and retry is negated. During this time, the system processor may access the DMAC internal registers to check channel status or modify the operation. When relinquish and retry is negated, the DMAC will then arbitrate for the bus, re-execute the previous bus cycle, and continue normal operations.

5.4.4 External Request Recognition

When an external generation method is used to inform the DMAC that a peripheral device needs service, only one request can be pending for each channel at any time. In order to guarantee that every peripheral request is recognized by the DMAC, the protocol described previously must be followed, i.e., the peripheral must wait until $\overline{ACK}$ has been asserted once for each successive transfer request before issuing a new request. Furthermore, the assertion of $\overline{DTC}$ must be used by the peripheral to determine if a bus cycle terminated normally in order to properly hand e a bus error, retry, or relinquish and retry termination. Thus, $\overline{ACK}$ indicates that the DMAC is starting to service a request, while $\overline{DTC}$ indicates that the request has been successfully serviced. If a request is pending for a channel at the end of a bus cycle terminated with retry or relinquish and retry, further requests will not be recognized until the retry has been successfully completed and the DMAC has begun to service the pending request.

When single address transfers are used, this protocol is very easy to follow since $\overline{ACK}$ will only be asserted once for each operand transfer. However, during dual address transfers, $\overline{ACK}$ may be asserted up to four times for each operand transfer and thus the DMAC will only recognize requests at certain points during such a transfer.

5.4.5 Software Control

During a data transfer operation, the system processor may interrogate the CSR (and other registers if desired) to monitor a channel operation. It is assumed that the system processor may gain ownership of the DMAC bus in order to perform the MPU accesses to the DMAC. If it is determined that an operation should be suspended or stopped, the system processor may also write to the CCR to cause such action to be taken by the DMAC. The following paragraphs describe the software control functions available and the DMAC response to them.

5.4.5.1 SOFTWARE HALT. The system processor may temporarily suspend channel operation at any time by setting the HLT bit in the CCR. In response, the channel will remain active, but no further operand transfer requests will be serviced for that channel as long as the HLT bit remains set. If no requests are pending when the HLT bit is set, the channel will recognize a transition or assertion of $\overline{REQ}$ as a valid request, but only one request will be made pending while the channel is halted. When the processor clears the HLT bit, the channel will then resume normal operation and may service a pending request.

The software halt functions similarly to the hardware halt operation in that a channel may be halted at any point during a block transfer, but the system processor will not normally have access to the DMAC registers in the middle of an operand transfer operation. One exception to this is the case when a dual address transfer read cycle (or the first of two write cycles) is terminated with a hardware halt or relinquish and retry; the system processor will have access to the CCR to set the HLT bit before the next bus cycle. In this case, when the $\overline{BEC}$ pins return to normal, the DMAC will rearbitrate for the bus and complete the dual address operand transfer before the software halt will be honored. It should also be noted that pending transfer requests for the other channels can not be serviced until the system processor has returned bus ownership to the DMAC and the operand transfer for the halted channel is complete.

5.4.5.2 SOFTWARE ABORT. The system processor may terminate a channel operation at any time by setting the SAB bit in the CCR. When this bit is set, the channel operation will be terminated immediately, the ACT bit will be cleared, and the COC and ERR bits will be set in the CSR. A software abort will be flagged in the CER to indicate that the abort has been honored.

The software abort operation may be executed at any point during a channel operation, but the system processor will normally not have access to the DMAC registers in the middle of an operand transfer operation. One exception to this is the case when a dual address transfer read cycle (or the first of two write cycles) is terminated with a hardware halt or relinquish and retry, then the system processor will have access to the CCR to set the SAB bit before the next bus cycle. In this case, the DMAC will remain idle until the REQ pins return to normal, but the remainder of the operand transfer will not be executed and the data that was read from the source will be lost.

5.4.6 Hardware Abort

In order to allow external hardware to abort a channel operation at any time, the PCL signal may be programmed as an abort input. This input operates in a manner similar to the software abort just described by setting the PCT bit in the CSR when a high-to-low transition occurs on PCL. The DMAC responds to PCT being set with PCL programmed as an abort input by terminating the channel operation, including any bus cycle that is being executed. If a bus cycle is in progress when the abort signal is asserted, an orderly termination of the bus cycle will occur in the same manner as if a bus error exception code were used to terminate the cycle.

The PCL signal may be asserted at any time to set the PCT bit. It should be noted that if PCL is programmed as an abort input and the PCT bit is set when a channel is started, it will be aborted before the first operand is transferred. This is true whether the PCT bit was set during a previous operation and was never cleared or by a transition of PCL before the channel was started.

NOTE

The contents of the MAR, DAR, and MTCR may not be valid after an external abort.

5.5 BASE REGISTER OPERATIONS

Three special operating modes are available to reduce the system processor overhead required for certain operations, and to allow non-contiguous data blocks to be transferred to or from a device or a contiguous data block(s) during one channel operation. These modes are the continue and chaining modes of operation and all three utilize the BFCR, BAR, and BTCR as temporary holding registers to be transferred to the MFCR, MAR, and MTCR at certain points during a channel operation, or as pointers into an array of data block descriptors. All of these modes of operation are independent of the transfer protocol or request generation method used and do not affect the DFCR or DAR.

5.5.1 Continue Operation

To initiate the continue mode of operation, the system processor loads the BFCR, BAR, and BTCR with the address and size of the next block to be transferred after the completion of the current block (described by the MFCR, MAR, and MTCR) and sets the CNT bit in the CCR. The only limitation on when this operation may be performed is that the CNT bit may not be set before a channel is active (it may be set at the same time that the channel is made active) and the BTC bit in the CSR may not be set prior to an attempt to set CNT. When the channel decrements the MTCR to zero and the CNT bit is set, it will not terminate the operation, but instead remain active (although the $\overline{\text{DONE}}$ signal is asserted to signal the end of each block). The BFCR, BAR, and BTCR will then be transferred to the MFCR, MAR, and MTCR, the CNT bit will be cleared and the BTC bit will be set. The channel will then resume normal operation by responding to operand transfer requests to transfer the new block of data. If the channel completes the transfer of the new block of data and the CNT bit is not set, the channel operation will be terminated in the normal fashion.

If the interrupt bit in the CCR is set when the channel completes a block transfer with the CNT bit set, the system processor will receive an interrupt when the BTC bit is set to indicate that the DMAC has completed the previous block transfer and is starting on the next (or the processor may poll the DMAC). At this time, the interrupt handler routine executed by the processor can load the BFCR, BAR, and BTCR with information describing the next data block if necessary, clear the BTC bit, and set the CNT bit to repeat the operation described above as many times as desired. Figure 5-3 shows a flowchart depicting the use of the continue mode.

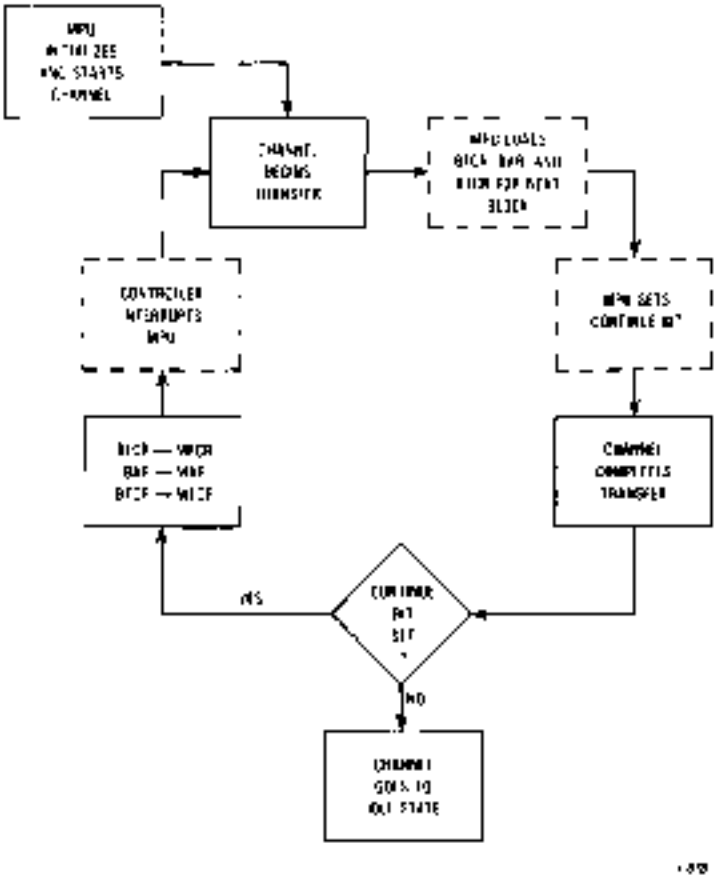


Figure 5-3. Continue Mode Operation Flowchart

5.5.2 Sequential Array Chaining

This type of a chaining operation uses an array of data block descriptors, organized sequentially in memory, to define the data to be transferred during a single channel operation. The data block descriptor array is defined by the BFCR, BAR, and BTCR, and each data block descriptor defines the location and size of a data block. Figure 5-4 shows the format of the data block descriptor array and the data blocks described by that array.

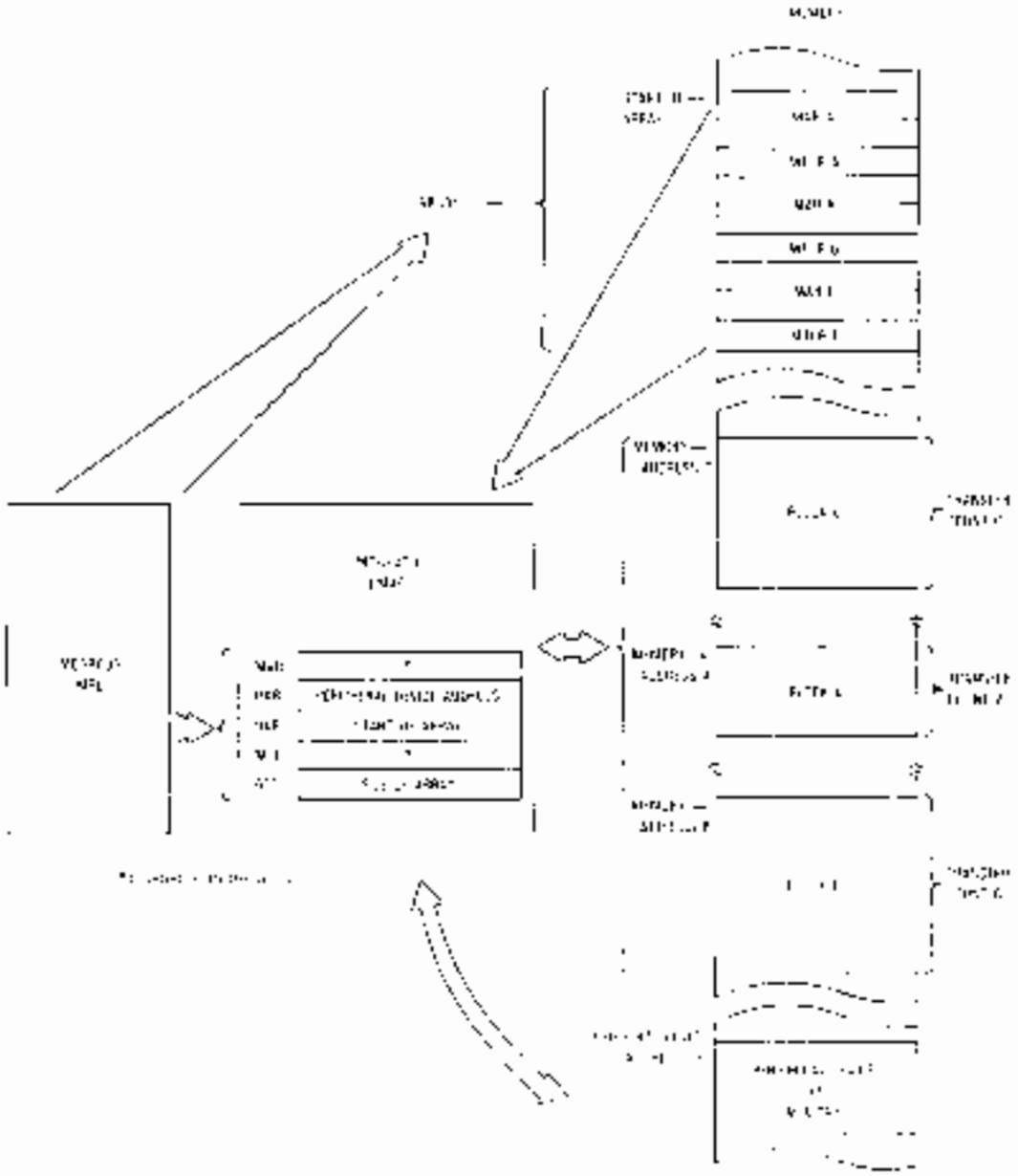


Figure 5-4 Sequential Array Chaining Mode Example

To utilize this mode of operation, the BFCR and BAR are loaded with the starting address of the data block descriptor array and the BTCR is loaded with the number of descriptors in the array. The MFCR must be loaded with the address space of all of the data blocks, since that information is not included in a data block descriptor. The channel is then started, and the DMAC will read the first descriptor into the MAR and MTCR, increment the BAR by six, decrement the BTCR by one, and begin to transfer the first data block by servicing transfer requests. When the MTCR is decremented to zero, the DMAC will assert DONE to signal the completion of a block transfer and read the next data block descriptor from the array. This process continues until the DMAC completes a block transfer and the BTCR value is zero, at which time the channel operation is terminated.

NOTE

The data block descriptor array must start at an even address or an address error will occur when the channel is started.

5.5.3 Linked Array Chaining

This mode of operation is very similar to sequential array chaining, except for the organization of the data block descriptor array. In this case, each descriptor not only describes a data block, but also contains a pointer to the next descriptor. This organization allows for easier editing of the descriptor array in order to modify the order of transfer for several data blocks. Figure 5-5 illustrates the organization of a descriptor array for this mode of operation.

To utilize this mode of operation, the BFCR and BAR are loaded with the address of the first descriptor in the array, the MFCR is loaded with the address space for all of the data blocks, and the channel is started. The DMAC then reads the first descriptor block into the MAR and MTCR, and loads the link address from the descriptor into the BAR. The data block is then transferred in the normal manner, and when the MTCR is decremented to zero, the DMAC asserts DONE to indicate the end of a data block transfer, and checks the value in the BAR. If the value is not the NULL pointer (all zeros), then it is used as the address of the next descriptor and the whole sequence is repeated. The channel operation will continue until a block transfer completes and the BAR contains a value of zero.

NOTE

Each of the data block descriptors must start at an even address or an address error will occur.

A comparison of both chaining modes is shown in Table 5-5.

Table 5-6. Chaining Mode Address/Count Information

Chaining Mode	Base Address Register	Base Transfer Counter	Completion When
Sequential Array Chaining	Address of the Array	Number of Data Blocks to be Transferred	BTC = 0
Linked Array Chaining	Address of the First Link Array Descriptor	Unused	BAR = 0

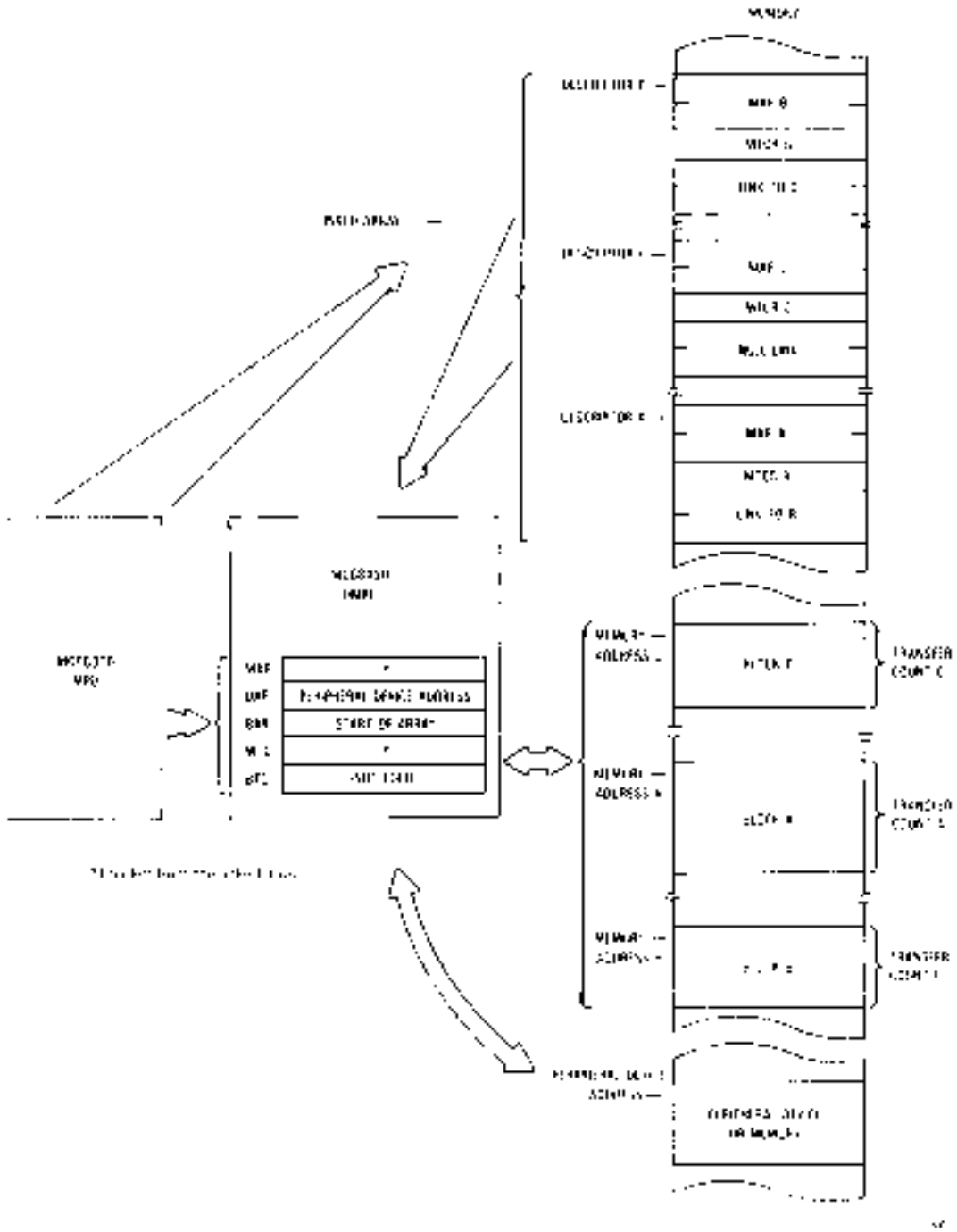


Figure 5-5 Linked Array Chaining Mode Example

NOTE

The chaining modes of operation can be used with any transfer mode or request generation method. However, if the implicitly addressed, with ACK and RDY protocol is used, then the RDY signal must be asserted during the DMA read cycles from the descriptor array. The circuit shown below can be used to provide the correct RDY signal timing for this operating mode. With this circuit, notice that RDY is asserted during every cycle that is not acknowledging the chained channel; however, this does not cause incorrect operation since RDY is ignored when the channel is not active.

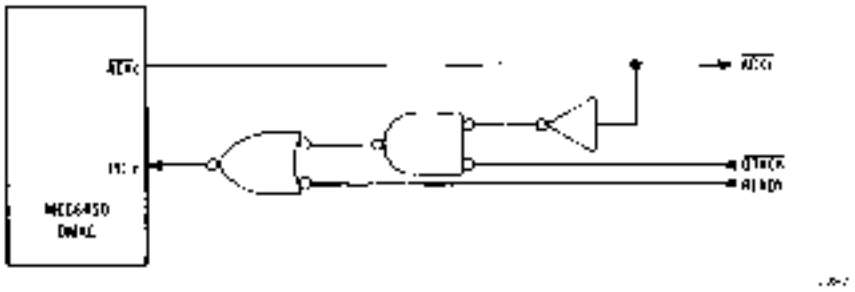


Figure 5-6. $\overline{\text{READY}}$ Circuit for Device With Acknowledge and Ready

5.5.4 BTCR Value Restrictions

In the continue or sequential array chaining modes of operation, the BTCR should not be loaded with a value of zero by the MPU or a count error will occur. If a continue or chaining operation occurs and the MTCH is loaded with zeros by the operation, a count error will occur.

5.6 CHANNEL TERMINATION

Once a channel operation has been completed normally or terminated due to an error, the DMAC will reflect the final status of the operation in the CSR and optionally interrupt the system processor. The following paragraphs describe the occurrences that cause a channel operation to be terminated and the status information affected by each.

5.6.1 Transfer Count Exhausted

When the DMAC begins an operand transfer, if the current value of the MTCR is one and an external request generation method is being used, then the $\overline{\text{DONE}}$ signal will be asserted during the last bus cycle in the device to indicate that the channel operation will be terminated when the current operand transfer is successfully completed. When the operand transfer is completed and the MTCR is decremented to zero, the channel operation will be terminated, the ACT bit will be cleared, and the COC bit will be set. The MAR and/or DAR will also be incremented in the normal fashion.

5.6.2 Device Termination

A transfer operation may be terminated before the MTCR is decremented to zero by the device if desired. If an external request generation method is used and the $\overline{\text{DONE}}$ signal is asserted when $\overline{\text{DTC}}$ is asserted during a device access, then the channel operation will be terminated. Also, $\overline{\text{DONE}}$ must be sampled as asserted on two consecutive rising clock edges, so if it is generated asynchronously, it must be asserted for two DMAC clock periods.

If the dual address transfer protocol is used, the $\overline{DONE}$ input will be recognized during any bus cycle of an operand transfer. If $\overline{DONE}$ is asserted during the first read or write cycle of a word or long-word operand from or to an 8-bit device, the channel operation will be terminated after the entire operand transfer is completed. This means that several more bus cycles may be executed, after $\overline{DONE}$ is recognized, to write the byte to the memory or a device, such that a full size operand transfer will be executed.

Once the operand transfer is completed, the channel will be terminated, the ACT bit will be cleared, and the COC and NDT bits will be set. The MTCR will also be decremented and the MAR and/or DAR will be incremented in the normal fashion. If $\overline{DONE}$ is asserted by the device at the same time that the DMAC is asserting it, the channel will be terminated, but the device termination will not be recognized and the NDT bit will not be set.

5.6.3 Error Termination

There are two basic types of errors that can cause a channel operation to be terminated. Internal errors are those generated by the DMAC either during start up by the system processor or while it is bus master. External errors are those generated by external hardware in response to abnormal conditions. When a channel operation is terminated due to any of these errors, the COC and ERR bits in the CSR will be set and the ERROR CODE field in the CER will indicate what error caused the termination. Table 5-6 summarizes all of the error conditions that can cause channel termination and the source of each error.

Table 5-6 Channel Error Condition Summary

Error Type	Cause
Configuration Error	A function code other than 001, 002, or 005 is loaded into the MTCR or DFC register, or an illegal device/operand size combination is programmed into a channel and an attempt is made to set the STR bit.
Operation Timing Error	An attempt is made to set STR with ACT, COC, BCC, NDT, or ERR set. An attempt is made to set CNT without setting STR simultaneously, and ACT is not set. An attempt is made to set CNT with STR and ACT set. An attempt is made to write to the DCR, OCR, SCR, MAR, MFCR, MTCR, DAR, or BTCR with STR or ACT set.
Address Error	$\overline{CS}$ or $\overline{IACK}$ is asserted when the DMAC is acting as bus master. A word or long-word bus cycle is attempted to an odd address. This may occur during the following operations: Any word or long-word operand transfer. An access to the descriptor array during channel operation.
Bus Error	A DMAC bus cycle is terminated with a bus error exception.
Count Error	The MTCR contains 0000 and an attempt is made to start the channel while not in a chaining mode. 0001 is loaded into the MTCR by a decrement of zero operation. The BTCR contains 0000 and an attempt is made to start the channel in the sequential array chaining mode.
External Abort	The PCL line is programmed as an abort input and the PCL pin is set with the ACT bit set.
Software Abort	The SAB bit is set by the MPU.

5.6.3.1 INTERNAL ERRORS. There are five types of internal errors that can occur: configuration errors, operation timing errors, address errors, count errors, and software abort. Software abort has already been discussed (refer to **6.4.5.2 SOFTWARE ABORT**), and the following paragraphs describe each of the other four.

5.6.3.1.1 Configuration Error. When a channel is configured incorrectly and the STR bit is set by the system processor, a configuration error will occur and the channel will not be started.

5.6.3.1.2 Operation Timing Error. When a write access is attempted to certain registers or bits within registers while a channel is active or when certain status bits are set, an operation timing error will occur and the channel operation will be aborted if the channel is active.

5.6.3.1.3 Address Error. When the DMAC is acting as the bus master and $\overline{CS}$ or $\overline{IACK}$ is asserted, or a word or long word operation is attempted to an odd address, an address error will be signaled and the channel operation will be aborted. If $\overline{CS}$ or $\overline{IACK}$ is asserted during a DMA access, no internal registers will be modified by the access and the operation of the other channel will not be affected. If a word access on an odd address is attempted, the address error will be detected before the DMAC executes an operand transfer and no DMAC internal registers will be affected (other than the CSR).

5.6.3.1.4 Count Error. When a channel is started or a continue operation is performed and the MTCR value is zero, a count error will be signaled and the channel operation terminated. If the MTCR is zero when the STR bit is set, the count error will be detected before the channel is started and no transfer requests will be recognized. If the BTCR is zero when a block transfer is terminated with the CNT bit set, a count error is detected, and the channel operation is terminated whether or not a transfer request is pending. In the sequential array chaining mode of operation, if the BTCR contains zero when the channel is started, a count error will be detected and the operation terminated before the array is accessed. If the MTCR is loaded with zero by a chaining operation, the count error will cause the channel operation to be terminated before the BAR and BTCR are updated.

5.6.3.2 EXTERNAL ERRORS. There are two types of error conditions that are generated by external hardware: bus error and hardware abort. The behavior of the DMAC when either of these conditions is signaled has been previously discussed (refer to **5.4.3.3 BUS ERROR** and **5.4.6 HARDWARE ABORT**).

SECTION 6
ELECTRICAL SPECIFICATIONS

This section contains electrical specifications and associated timing information for the MC68450 direct memory access controller.

6.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range	T_A	0 to 70	°C
Storage Temperature	T_{stg}	−55 to 150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{CC}).

6.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Symbol	Value	Unit
Thermal Resistance	θ_{JA}		θ_{JC}		°C/W
Ceramic (L, CC)		30		15*	
Plastic (P)		30		15*	
Pin Grid Array (RP, RC)		35		15	

*Estimated

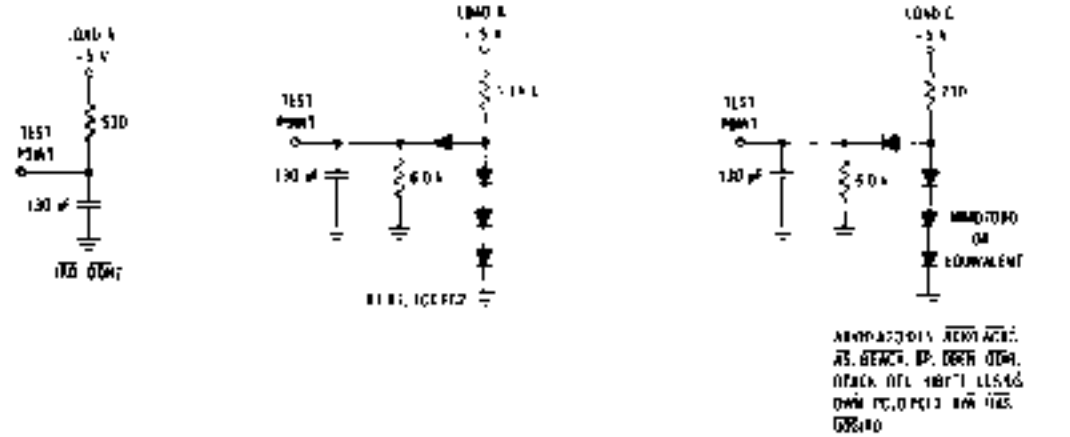


Figure 6-1. Test Loads

6.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T_A = Ambient Temperature, °C

θ_{JA} = Package Thermal Resistance, Junction to Ambient, °C/W

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins, Watts — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The total thermal resistance of a package (θ_{JA}) can be separated into two components, θ_{JC} and θ_{CA}, representing the barrier to heat flow from the semiconductor junction to the package (case) surface (θ_{JC}) and from the case to the outside ambient (θ_{CA}). These terms are related by the equation:

$$\theta_{JA} = \theta_{JC} + \theta_{CA} \quad (4)$$

θ_{JC} is device related and cannot be influenced by the user. However, θ_{CA} is user dependent and can be minimized by such thermal management techniques as heat sinks, ambient air cooling and thermal convection. Thus, good thermal management on the part of the user can significantly reduce θ_{CA} so that θ_{JA} approximately equals θ_{JC}. Substitution of θ_{JC} for θ_{JA} in equation (1) will result in a lower semiconductor junction temperature.

Values for thermal resistance presented in this document, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843, "Thermal Resistance Measurement Method for MC68XX Microcomponent Devices," and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User-derived values for thermal resistance may differ

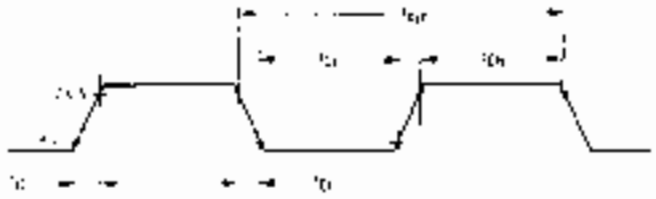
6.4 DC ELECTRICAL CHARACTERISTICS

(V_{CC} = 5 V ±5%, GND = 0 V, T_A = 0 to 70°C, unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage	V_{IH}	2.0	V_{CC}	V
Input Low Voltage	V_{IL}	GND – 0.3	0.8	V
Input Leakage Current — $\overline{CS}$, $\overline{IACK}$, $\overline{DS}$, CLK, $\overline{REC0}$, REC1, REC2, $\overline{RE0}$–$\overline{RE3}$, $\overline{RE4}$	I_{in}	–	±10	µA
Three-State (Off State) Input Current — A1–A7, D0–A3, D15–A20, AS, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, DTACK, $\overline{BGACK}$, CIWV, DTC, HIBYTE, $\overline{DCYR}$, $\overline{DSEN}$, FC0–FC2, PCL0–PCL3	I_{TSI}	–	±20	µA
Input Capacitance (V_{in} = 0 V, T_A = 25°C, f = 1 MHz)	C_{in}	–	15	pF
Open-Drain (Off State) Input Current — $\overline{BG}$, $\overline{SCINT}$	I_{OD}	–	20	µA
Output High Voltage — A1–A7, D0–A3, D15–A23, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, DTACK, $\overline{BR}$, $\overline{DWN}$, $\overline{DTC}$, HIBYTE, $\overline{ODR}$, $\overline{DREN}$, $\overline{AVEC}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DWN}$, FC0–FC2, PCL0–PCL3, $\overline{LIR}$–$\overline{LIR2}$ — I_{OH} = –400 µA	V_{OH}	2.4	–	V
Output Low Voltage — I_{OL} = 3.2 mA: A1–A7, FC0–FC2; I_{OL} = 5.2 mA: D0–A3, D15–A23, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, DTACK, $\overline{BR}$, $\overline{DWN}$, $\overline{DTC}$, HIBYTE, $\overline{ODR}$, $\overline{DREN}$, $\overline{AVEC}$, $\overline{IACK}$, $\overline{RE0}$–$\overline{RE3}$, $\overline{DAS}$, PCL0–PCL3, $\overline{BGACK}$; I_{OL} = 3.8 mA: $\overline{BG}$, $\overline{SCINT}$	V_{OL}	–	0.5	V
Power Dissipation (25°C, Frequency = 8 MHz)	P_D	–	2.5	W

6.5 AC ELECTRICAL SPECIFICATIONS — CLOCK TIMING (See Figure 6-2)

Parameter	Symbol	8 MHz Min	8 MHz Max	10 MHz Min	10 MHz Max	Unit
Frequency of Operation	f	2	8	2	10	MHz
Clock Period	t_{cyc}	125	500	100	500	ns
Clock Width Low	t_{CL}	58	250	45	250	ns
Clock Width High	t_{CH}	57	250	45	250	ns
Clock Fall Time	t_{CF}	–	10	–	10	ns
Clock Rise Time	t_{CR}	–	10	–	10	ns



NOTE

Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted. The actual voltage at the point of measurement is dependent on the rise and fall times and the input characteristics of the device under test.

Figure 6-2. Clock Input Timing Diagram

6.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES

(V_{CC} = 5.0 V ±5%, GND = 0 V, T_A = 0°C to 70°C, unless otherwise noted)

(See Figures 6-4 through 6-9)

			8 MHz		10 MHz		
No.	Parameter	Symbol	Min	Max	Min	Max	Unit
1	Clock Period	t_{CYC}	125	500	100	500	ns
2	Clock Width (High)	t_{CL}	56	282	45	226	ns
3	Clock Width (Low)	t_{CH}	59	277	47	224	ns
4	Clock Rise Time	t_{Cr}		10		10	ns
5	Clock Fall Time	t_{Cf}		10		10	ns
6	Address Valid to Clock High (ASL) Setup	t_{ASL}	30		25		ns
7	Clock High to DBEN Low	t_{CHDEL}	0		0		ns
8	DBEN Low to Data In Valid (Read)	t_{DELID}		0		0	ns
9	Address Valid to AS Low	t_{ASL}	0		0		ns
10	AS, R/W, Data Out Valid to Clock High	t_{ASL+DS}	7		7		ns
11	Clock High to DSR High	t_{CHSRH}		70		60	ns
12	Clock High to DSR Low	t_{CHSRL}		70		60	ns
13	DS or HDS Low to DSR High (Write)	t_{DSHDRH}		190		160	ns
14	Clock Low to DTACK Low	t_{CLDTL}		70		60	ns
15	DS or HDS Low to DTACK Low	t_{DSHDTL}		60		50	ns
16	DS or HDS Low to DTACK High	t_{DSHDTH}		210		180	ns
17	Data Out Valid to DSR Low	t_{DVDRL}	10		10		ns
18	DS or HDS High to DTACK, DSR, Data Out Invalid	t_{DSHDVZ}		80		70	ns
19	DS or HDS High to Data Out Invalid	t_{DSHDVZ}	0		0		ns
20	DS or HDS High to Data High Impedance	t_{DSHDZ}		120		110	ns
21	DTACK Low to Data In Valid	t_{DTLDSH}		50		40	ns
22	DS or HDS Low to DTACK High	t_{DSHDTH}		210		180	ns
23	Data In Valid to Clock Low (Data Setup)	t_{DICL}	10		10		ns
24	DS or HDS High to DTACK, DSR, Data In Invalid (Hold)	t_{DSHDIZ}		80		70	ns
25	DTACK Low to Data In Invalid (Hold)	t_{DTLDSH}	0		0		ns
26	R/W Low to DS or HDS Low	t_{RWDSL}	82		82		ns
27	R/W Valid to Clock High (Setup)	t_{RWCH}	70		60		ns
28	Clock High to R/W High	t_{CHRH}		70		60	ns
29	Clock High to R/W Low	t_{CHRL}		70		60	ns
30	R/W Low to DS or HDS Low	t_{RWDSL}	82		82		ns
31	R/W Valid to AS High or DS or HDS Low	t_{RWASH}	70		60		ns
32	R/W Low to Data Out Valid	t_{RWDV}	12		12		ns
33	Clock Low to R/W, AS, DS or HDS High	t_{CLRH}		70		60	ns
34	Clock High to AS, DS or HDS High	t_{CHSH}		70		60	ns
35	Clock Low to AS, DS or HDS High	t_{CLSH}		80		70	ns
36	Clock High to Address, FC, Data High Impedance	t_{CHAZ}		80		70	ns
37	Clock High to Address Valid	t_{CHAV}	0		0		ns
38	Clock High to AS, DS or HDS High	t_{CHSH}		70		60	ns
39	AS, DS or HDS High to Address Invalid	t_{SHAZ}	30		30		ns
40	AS High to DS or HDS Low	t_{ASDSL}	60		50		ns
41	Clock High to AS High	t_{CHSH}		70		60	ns
42	AS High to Clock High	t_{SHCH}	30		30		ns
43	Clock Low to AS, DS or HDS High	t_{CLSH}		80		70	ns
44	AS High to DS or HDS Low	t_{ASDSL}	60		50		ns
45	Address Valid to AS, DS or HDS Low (Setup)	t_{AVSL}	30		25		ns
46	AS, DS or HDS High to Address Invalid (Hold)	t_{SHAZ}	30		30		ns
47	Address Valid to Clock High	t_{AVCH}	30		25		ns
48	Clock High to Address Invalid	t_{CHAZ}	0		0		ns
49	Clock Low to Address Invalid	t_{CLAZ}	0		0		ns
50	Clock Low to Address Valid	t_{CLAV}		80		70	ns

6.8 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

No	Parameter	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
47	Clock Low to $\overline{AS}$, $\overline{DS}$ High	t_{CLSH}		35		30	ns
48	Clock Low to $\overline{AS}$, $\overline{DS}$ High Impedance	t_{CLSZ}		80		70	ns
49	$\overline{AS}$ Width Low	t_{ASL}	255		135	–	ns
50	$\overline{DS}$ Width Low	t_{DSL}	250		190		ns
51	$\overline{AS}$, $\overline{DS}$ Width High	t_{SH}	170		105		ns
52	Address, FC Valid to $\overline{AS}$, $\overline{DS}$ Low (Read)	t_{AVSL}	70	–	30	–	ns
53	$\overline{AS}$, $\overline{DS}$ High to Address, FC Data Invalid	t_{SHAZ}	30	–	30	–	ns
54	Clock High to R/$\overline{W}$ Low	t_{CHRL}	–	50	–	60	ns
55	Clock High to R/$\overline{W}$ High	t_{CHRH}		50		60	ns
56	Clock Low to R/$\overline{W}$ High Impedance	t_{CLRZ}		80		70	ns
57	Address, FC Valid to R/$\overline{W}$ Low	t_{AVRL}	30		10		ns
58	R/$\overline{W}$ Low to $\overline{DS}$ Low (Write)	t_{RLSL}	120		80		ns
59	$\overline{DS}$ High to R/$\overline{W}$ High	t_{SHRH}	40		20		ns
60	Clock Low to $\overline{DWN}$ Low	t_{CLDL}		70		60	ns
61	Clock Low to $\overline{DWN}$ High	t_{CLDH}		70		60	ns
62	Clock High to $\overline{DWN}$ High Impedance	t_{CHDZ}		80		70	ns
63	$\overline{DWN}$ Low to $\overline{BGACK}$ Low	t_{DLBL}	70		30		ns
64	$\overline{BGACK}$ High to $\overline{DWN}$ High	t_{BHDH}	60		40		ns
65	$\overline{DWN}$ Low to $\overline{IAS}$ Low	t_{DLIL}	30		20		ns
66	Clock High to $\overline{ACK}$ Low (Read)	t_{CHAL}		70		60	ns
67	Clock Low to $\overline{ACK}$ Low (Write)	t_{CLAL}		70	–	60	ns
68	Clock High to $\overline{ACK}$ High	t_{CHAH}		70	–	60	ns
69	$\overline{ASP}$ Low to $\overline{DS}$ Low	t_{ASLDSL}	100		80		ns
70	$\overline{DS}$ High to $\overline{ASP}$ High	t_{DSHASH}	30		30		ns
71	Clock High to $\overline{HIBYTE}$ Low	t_{CHHL}		70		60	ns
72	Clock Low to $\overline{HIBYTE}$ Low	t_{CLHL}		70		60	ns
73	Clock High to $\overline{HIBYTE}$ High	t_{CHHH}		70		60	ns
74	Clock Low to $\overline{HIBYTE}$ High Impedance	t_{CLHZ}		80		70	ns
75	Clock High to $\overline{DTC}$ Low	t_{CHDTL}		70		60	ns
76	Clock High to $\overline{DTC}$ High	t_{CHDTH}		70		60	ns
77	Clock Low to $\overline{DTC}$ High Impedance	t_{CLDTZ}		80		70	ns
78	$\overline{DTC}$ Width Low	t_{DTCL}	105		80		ns
79	$\overline{DTC}$ Low to $\overline{DS}$ High	t_{DTLDH}	90		70		ns
80	Clock High to $\overline{ZONE}$ Low	t_{CHZOL}	–	70		60	ns
81	Clock Low to $\overline{ZONE}$ Low	t_{CLZOL}		70		60	ns
82	Clock High to $\overline{ZONE}$ High	t_{CHZOH}		140		120	ns
83	Clock Low to $\overline{ZOIR}$ High Impedance	t_{CLZOZ}		80		70	ns
84	Clock Low to $\overline{IPEN}$ High Impedance	t_{CLIPZ}		80		70	ns
85	$\overline{DSR}$ Low to $\overline{DREN}$ Low	t_{DRLSL}	40		30	–	ns
86	$\overline{DSES}$ High to $\overline{DDR}$ High	t_{DSHDH}	30	–	20	–	ns
87	$\overline{DSES}$ Low to Address, Data High Impedance	t_{DLAZ}		10		10	ns
88	Clock Low to $\overline{PCL}$ Low (1 Clock)	t_{CLPL}		70		60	ns
89	Clock Low to $\overline{PCL}$ High (1 Clock)	t_{CLPH}		70		60	ns
90	$\overline{PCL}$ Width Low (1 Clock)	t_{PCL}	240		190		CLK Per
91	$\overline{DTACK}$ Low to Data In Setup Time	t_{DALDI}	–	150		120	ns
92	$\overline{DS}$ High to Data Invalid (Hold Time)	t_{SHDI}	0		0		ns

6.6 AC ELECTRICAL SPECIFICATIONS — READ AND WRITE CYCLES (Continued)

No.	Parameter	Symbol	8 MHz		10 MHz		Unit
			Min	Max	Min	Max	
53	DS high to DTACK high	tSHDAH	0	120	0	90	ns
54	R/W valid to DS low	tRVSL	40		40		ns
55	Data-in valid to CLOCK low (Setup Time)	tDICL	10		10		ns
56	DS low to CLOCK low	tSLCL	50		40		ns
57	DS width low	tSLSH	240		180		ns Per
58	Clock high to AS low	tCHAL		80		60	ns
59	Clock high to AS high (maintain)	tCHAH		120		120	ns
60	DS low to READY low in DTC (MC68450 Read)	tSLRDL	145		120		ns
61	Clock low to READY low in DTC (MC68450 Write)	tCLRDL	205		155		ns
62	DS high to DTACK, BERR high	tSHDAH	0	120	0	90	ns
63	DTACK low to CLOCK low	tDALCL	50		30		ns
64	DS high to DTACK high	tSHDOH	0	120	0	90	ns

Timing Diagrams (Figures 6-4 through 6-9) are located on foldout pages at the end of this document.

SECTION 7
ORDERING INFORMATION

The following information should be used as a guide when ordering the MC68450.

Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic	8.0	0°C to 70°C	MC68450L8
L Suffix	10.0	0°C to 70°C	MC68450L10
	12.5	0°C to 70°C	MC68450L12
Plastic	8.0	0°C to 70°C	MC68450P8
P Suffix	10.0	0°C to 70°C	MC68450P10
	12.5	0°C to 70°C	MC68450P12
Pin Grid Array	8.0	0°C to 70°C	MC68450R8
(with Standoffs)	10.0	0°C to 70°C	MC68450R10
R Suffix	12.5	0°C to 70°C	MC68450R12
(See Figure 7-1)			
Pin Grid Array	8.0	0°C to 70°C	MC68450RC8
with Gold Lead	10.0	0°C to 70°C	MC68450RC10
Frame (without	12.5	0°C to 70°C	MC68450RC12
Standoffs)			
RC Suffix			

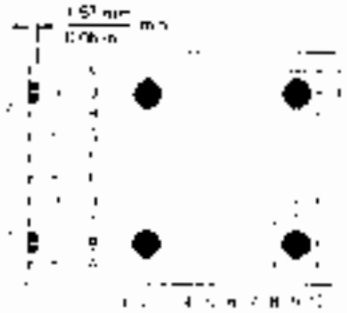


Figure 7-1. 68-Pin Grid Array
(Pins Soldered Dipped with Plastic Standoff)

SECTION 8
MECHANICAL DATA

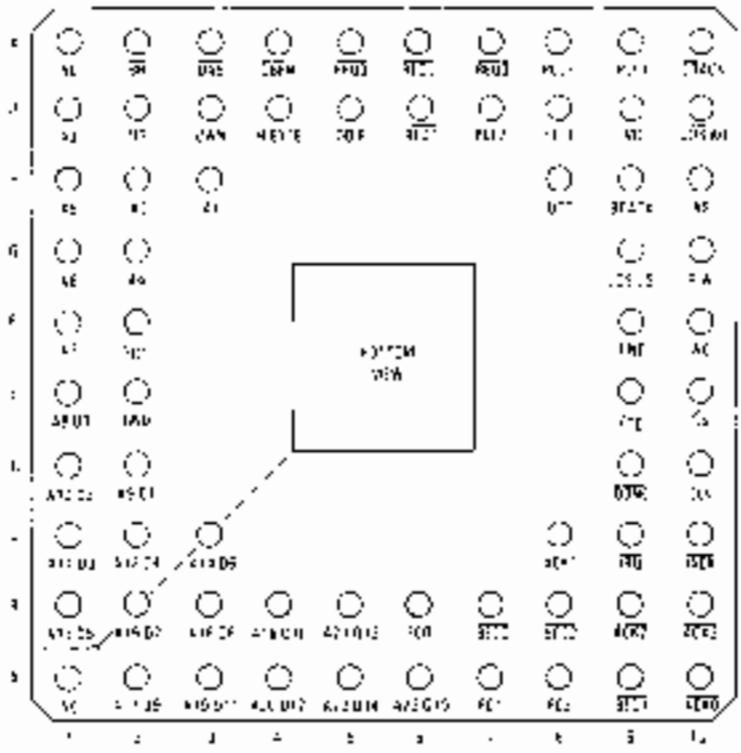
This section contains the pin assignments and package dimensions for the MC68450

8.1 PIN ASSIGNMENTS

8.1.1 68 Pin Dual In-Line Package



8.1.2 68-Terminal Pin-Grid Array



8.2 PACKAGE DIMENSIONS

L SUFFIX
CERAMIC PACKAGE
CASE 740-0



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
4. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
5. 740-0 OBSOLETE, NEW STANDARD 740-02.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	80.52	82.03	3.170	3.230
B	22.35	22.86	0.880	0.900
C	3.05	4.32	0.120	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.41	0.030	0.055
G	2.54 BSC		0.100 BSC	
J	0.20	0.33	0.008	0.013
K	2.54	4.13	0.100	0.163
L	22.61	23.11	0.890	0.910
M	0°	10°	0°	10°
N	1.02	1.52	0.040	0.060

P SUFFIX
PLASTIC PACKAGE
CASE 711-02



NOTES:
1. DIMENSIONS A AND B ARE DATUMS.
2. T IS SEATING PLANE.
3. POSITIONAL TOLERANCE FOR LEADS (DIMENSION D):
   ⌀ 0.25 (0.010) M T A M B M
4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION B DOES NOT INCLUDE MOLD FLASH.
6. DIMENSIONS A AND B INCLUDE MENISCUS PER ANSI Y14.5 1982.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	81.15	81.91	3.195	3.225
B	20.17	20.57	0.790	0.810
C	4.81	5.84	0.190	0.230
D	0.33	0.53	0.013	0.021
F	1.27	1.77	0.050	0.070
G	2.54 BSC		0.100 BSC	
J	0.20	0.38	0.008	0.015
K	3.05	3.55	0.120	0.140
L	22.86 BSC		0.900 BSC	
M	0°	15°	0°	15°
N	0.51	1.01	0.020	0.040

RC SUFFIX

P SUFFIX

NOTES



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	20.47	21.46	0.806	0.845
B	6.40	7.49	0.252	0.295
C	3.43	3.68	0.135	0.145
D	0.51	0.63	0.020	0.025
G	2.54 BSC		0.100 BSC	
K	2.92	4.14	0.115	0.163
M	10°	20°	10°	20°
R	15.24	15.95	0.600	0.628
U	15.24	15.95	0.600	0.628
V	1.14	2.79	0.045	0.110

Figure 3-2. Register Summary

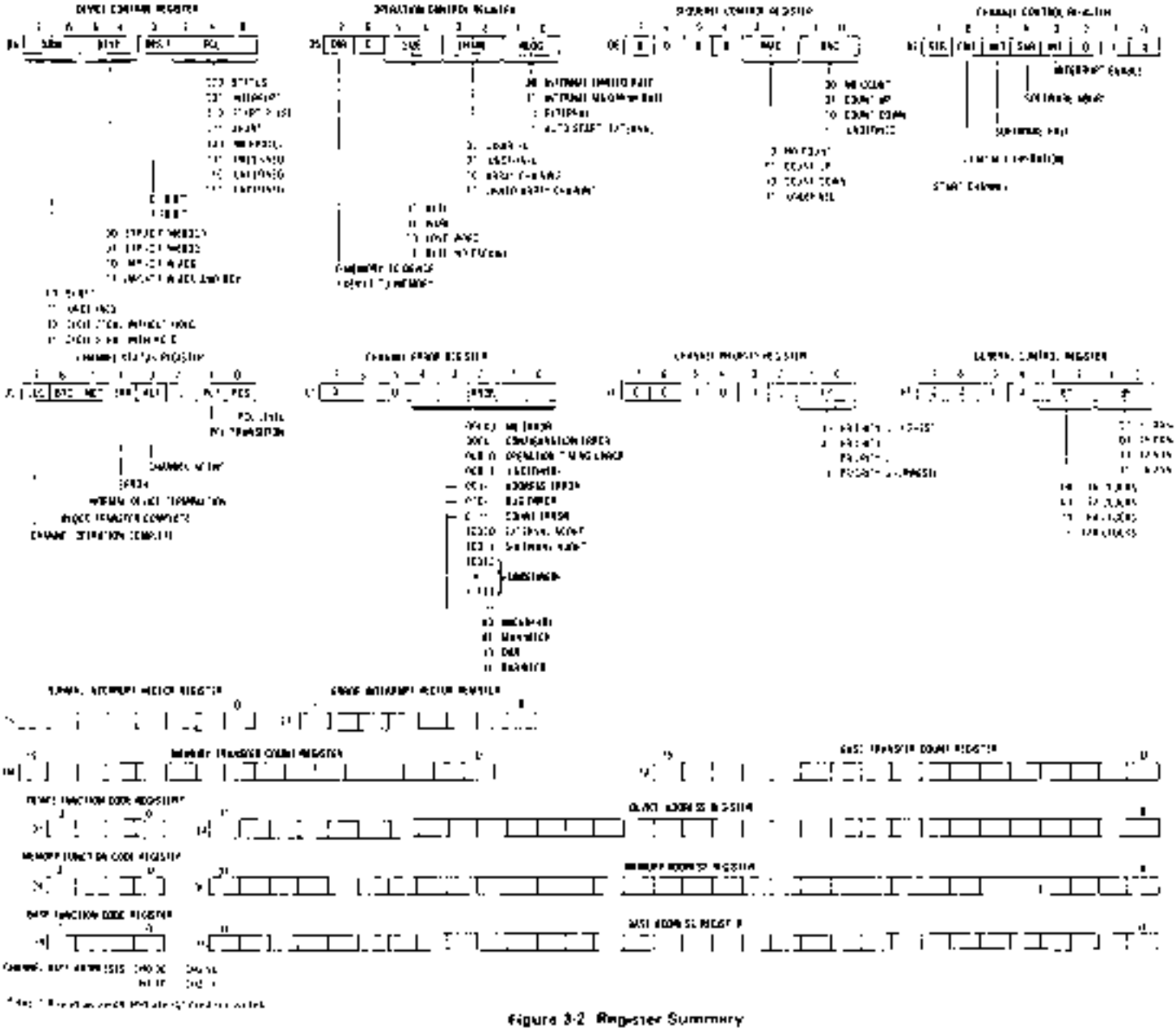
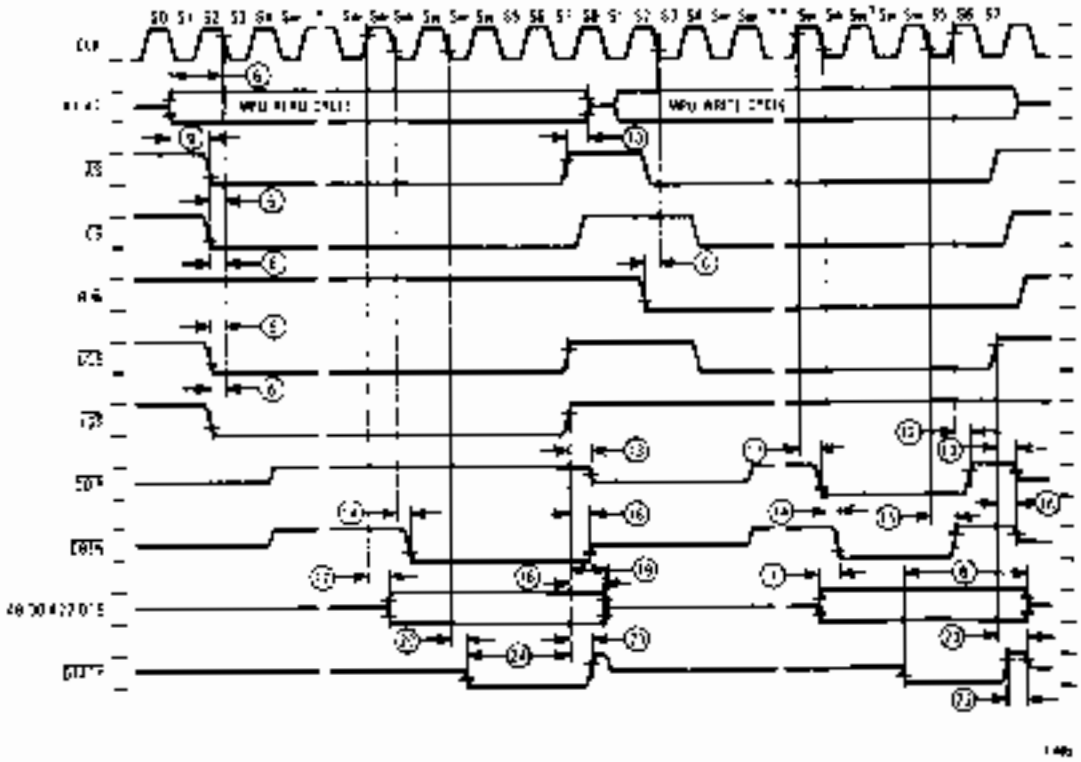


Figure 3-2 Register Summary

Figure 6-3. MPU Read/Write Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



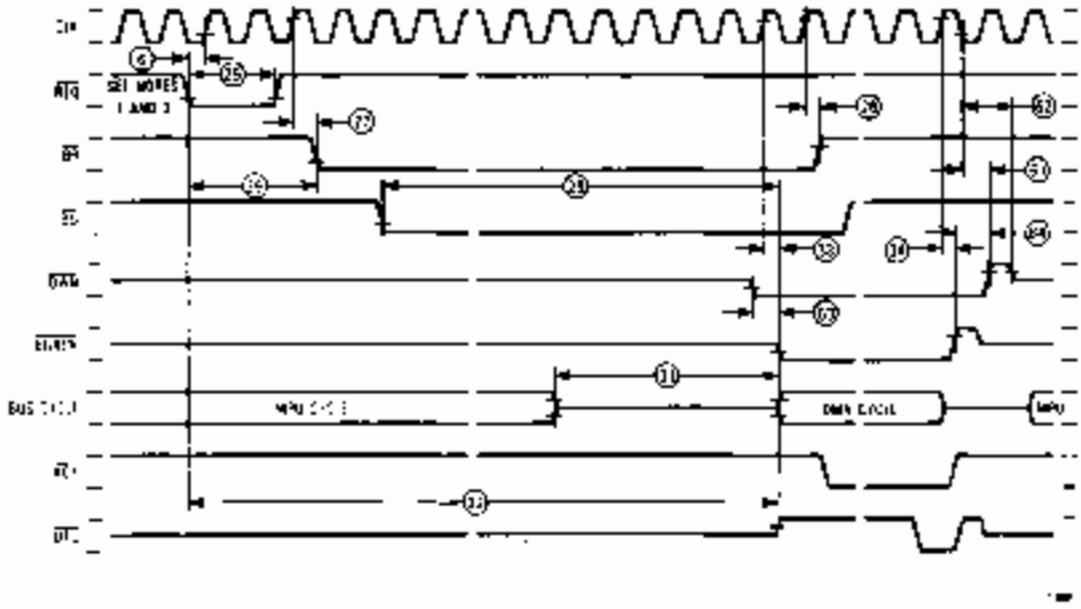
* 10 Additional Wait Cycles
** 42 Additional Wait Cycles

NOTES:
1. Data is latched at the end of this clock.
2. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-3. MPU Read/Write Timing Diagram

Figure 6-4. Bus Arbitration Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

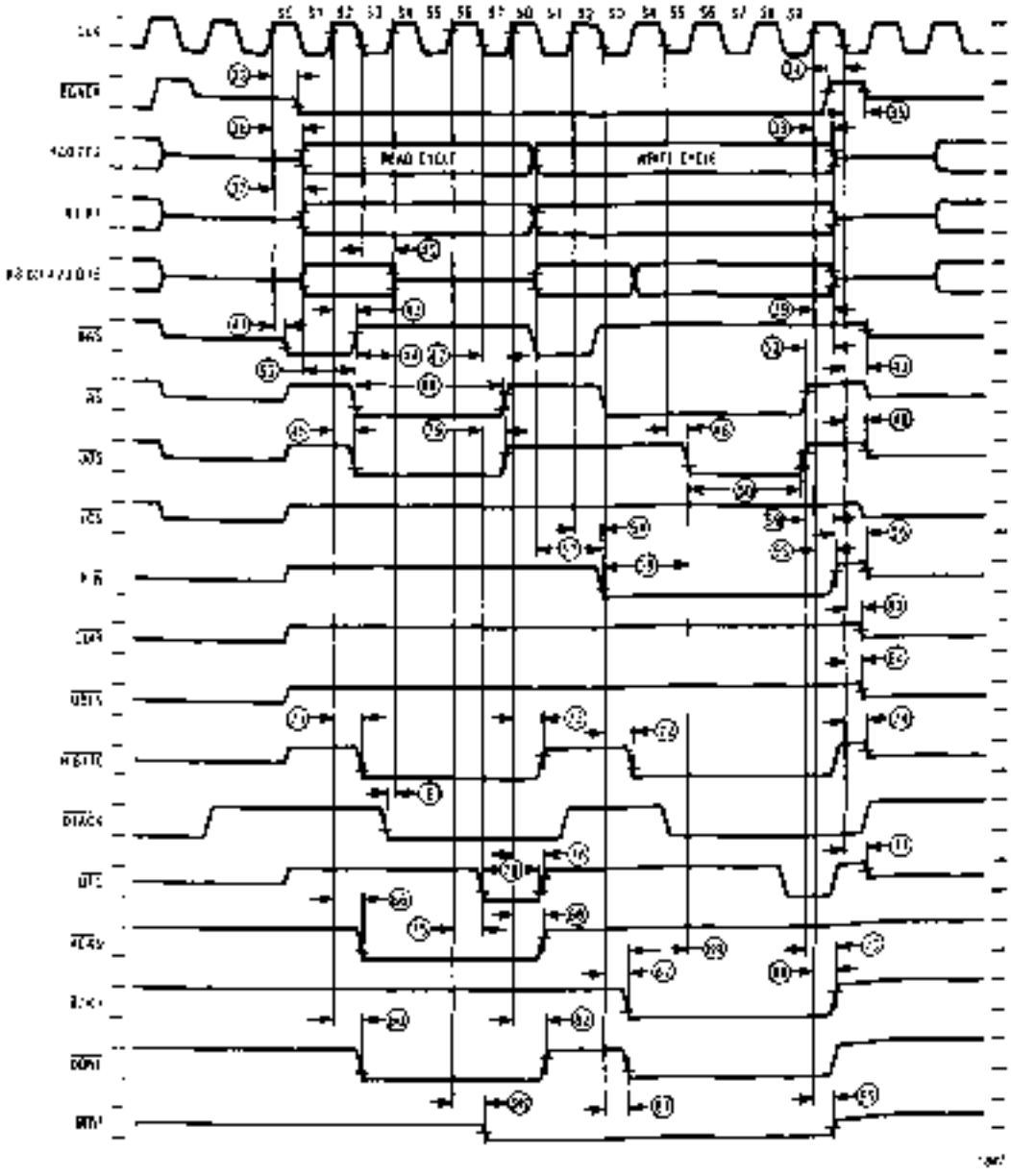


NOTES:
1. $\overline{RES}$ is sampled on the rising edge of CLK in mode start and halt modes.
2. $\overline{BR}$ will not be asserted while any $\overline{BG}$ assertion conditions exist, or when $\overline{CS}$ or $\overline{IACK}$ is asserted.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-4. Bus Arbitration Timing Diagram

Figure 6-6. DMA Read/Write Timing Diagram (Single Cycle)

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



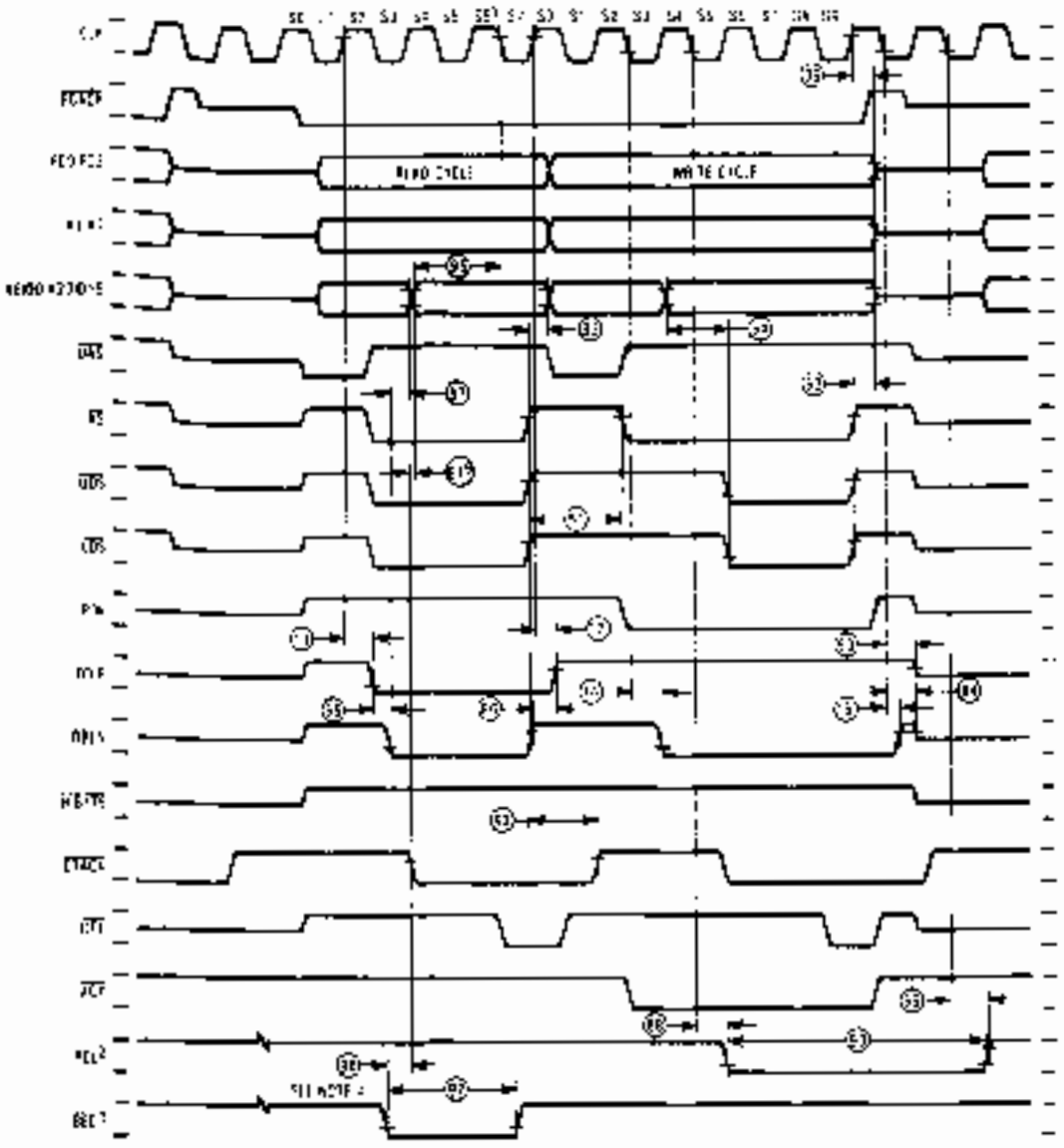
NOTES

1. The timing is not direct, relative to the DMA read/write cycle, but is relative to the E clock.
2. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-5. DMA Read/Write Timing Diagram (Single Cycle)

Figure 6-6. DMA Read/Write Timing Diagram (Dual Cycle)

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



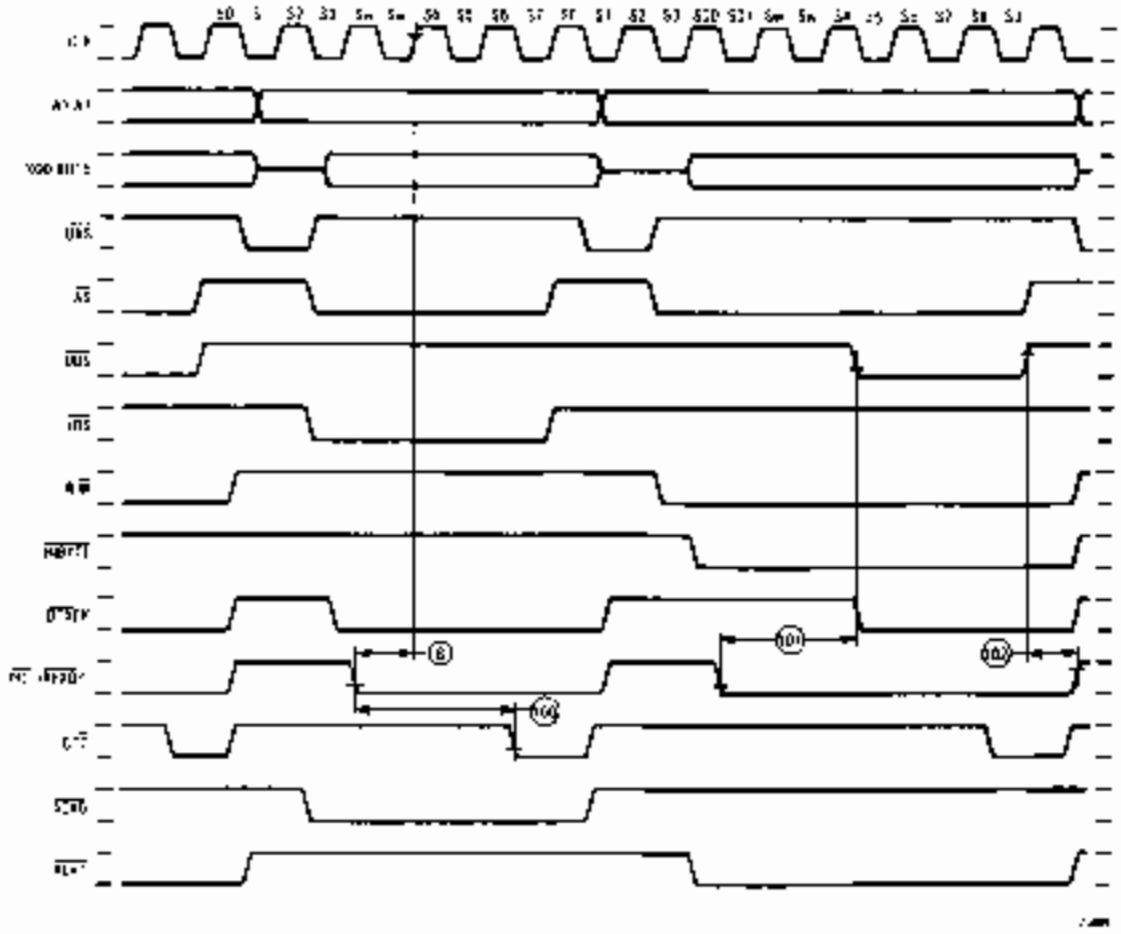
NOTES:

1. Data is latched at the end of clock 5f.
2. Timing is not directly related to the DMA read write cycle. Dual cycle sequence and is only applicable when the asynchronous mode is selected.
3. Timing is applicable when a bus error occurs.
4. A specification number 9 is valued for both $\overline{DTACK}$ and $\overline{BEC}$. Specification number 9 may be 0 nanoseconds.
5. If the propagation delay of the external synchronous buffers is less than 12 nanoseconds, a conflict may occur between the address output of the DMAC and the system data bus. In this case, the output of $\overline{DREN}$ must be delayed externally.
6. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-6. DMA Read/Write Timing Diagram (Dual Cycle)

Figure 5-7. DMA Read/Write Timing Diagram (Single Cycle with PCL as $\overline{\text{READY}}$)

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



NOTE

1. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-7 DMA Read/Write Timing Diagram (Single Cycle with PCL as $\overline{\text{READY}}$)

Figure 5-8. DONE Input Timing Diagram

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

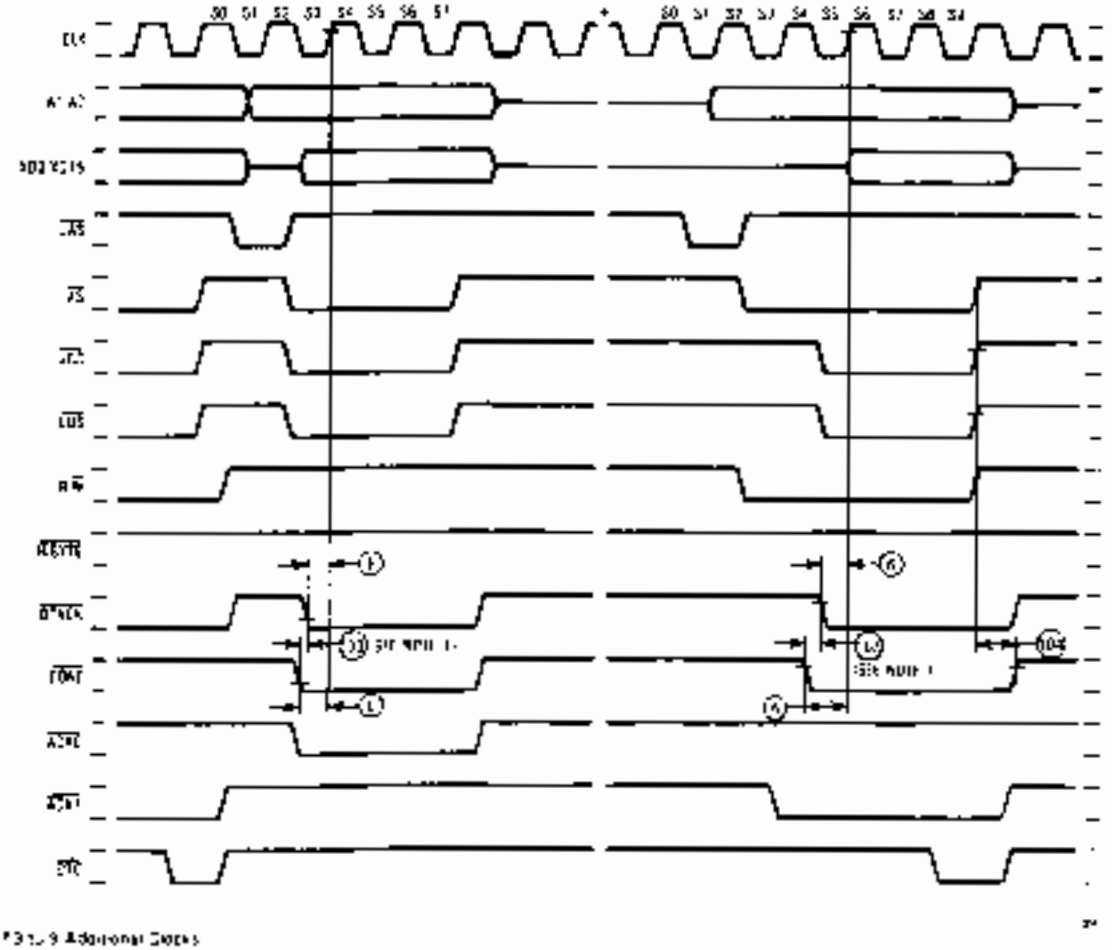


Figure 6-9 Additional Clocks

NOTES:

1. If speed option number 6 is selected for pins DTACK and DONE, specification number 103 may be 0 nanoseconds.
2. The setup time for asynchronous inputs RESET BGACK CS IACK AS UDS LDS and R/W guarantees their recognition at the next falling edge of the clock. The setup time for SIZ0 SIZ1 AS00-AS01 FC00 FC01 DTACK and DONE guarantees their recognition at the next rising edge of the clock.
3. Timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

Figure 6-8. DONE Input Timing Diagram

Literature Distribution Center

MOTOROLA

WESTERN DIGITAL
CORPORATION

WD1772-02 Floppy Disk Formatter/Controller

FEATURES

- 28 PIN DIP
- SINGLE 5V SUPPLY
- HIGH PERFORMANCE DPLL BUILT-IN DIGITAL DATA SEPARATOR
- BUILT-IN WRITE PRECOMPENSATION, INCREASED TO 187 NS
- SINGLE AND DOUBLE DENSITY
- MOTOR CONTROL
- 128, 256, 512 OR 1024 SECTOR LENGTHS
- TTL COMPATIBLE
- 8-BIT BI-DIRECTIONAL DATA BUS
- 100% PIN COMPATIBLE WITH WD1770-00 AND WD1772-00
- ENHANCED STEPRATES 2,3,6,12 MS



PIN DESIGNATION

DESCRIPTION

The WD1772-02 is a MOS/LSI device which performs the functions of a Floppy Disk Formatter/Controller. It is similar to its predecessor, the FD179X, but also contains a digital data separator and write precompensation circuitry. The drive side of the interface needs no additional logic except for buffers/receivers. Designed for single (FM) or double (MFM) density operation, the device contains a programmable Motor On signal.

The WD1772-02 is implemented in NMOS silicon gate technology and is available in a 28 pin dual-in-line.

The WD1772-02 is a low cost version of the FD179X Floppy Disk Controller/Formatter. It is similar to the FD179X, but has a built-in digital data separator and write precompensation circuits.

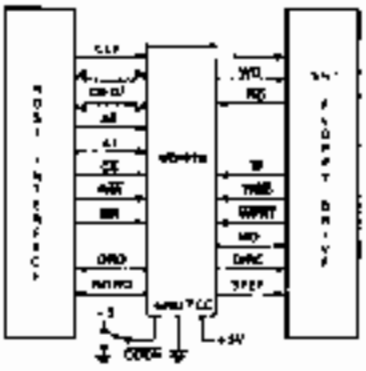
A single read line (RD, Pin 19) is the only input required to recover serial FM of MFM data from the disk drive. The device is designed for control of floppy disk drives with data rates of 125 KBits/Sec (single density) and 250 KBits/Sec (double density). In addition, write precompensation of 187 nsec from nominal is enabled at any point through simple software commands. Another programmable feature, Motor On, enables the spindle motor automatically prior to operating a selected drive.

The WD1772-02 offers stepping rates of 2, 3, 6 and 12 msec. The processor interface consists of an 8-bit bi-directional bus for transfer of status, data, and commands. All Host communication with the drive occurs through these lines. They are capable of driving one standard TTL load or three LS loads.

PIN NUMBER	MNEMONIC	SIGNAL NAME	I/O	FUNCTION
1	$\overline{CS}$	$\overline{CHIP\ SELECT}$	I	A logic low on this input selects the chip and enables Host communication with the device.
2	R/$\overline{W}$	READ/$\overline{WRITE}$	I	A logic high on this input controls the placement of data on the D0-D7 lines from a selected register, while a logic low causes a write operation to a selected register.
3,4	A0,A1	ADDRESS 0,1	I	These two inputs select a register to Read/Write data:
5-12	DAL0-DAL7	DATA ACCESS LINES 0 THROUGH 7	I/O	Eight-bit bi-directional bus used for transfer of data, control, or status. This bus is enabled by $\overline{CS}$ and R/$\overline{W}$. Each line will drive one TTL load.
13	$\overline{MR}$	$\overline{MASTER\ RESET}$		A logic low pulse on this line resets the device and initializes the Status Register (internal pull-up).
14	GND	GROUND		Ground.
15	V_{CC}	POWER SUPPLY	I	+5V ±5% power supply input.
16	STEP	STEP	O	The Step output contains a pulse for each step of the drive's R/W head.
17	DIRC	DIRECTION	O	The Direction output is high when stepping in towards the center of the diskette, and low when stepping out.
18	CLK	CLOCK	I	This input requires a free-running 50% duty cycle clock (for internal timing) at 8 MHz ±0.1%.
19	$\overline{RD}$	$\overline{READ\ DATA}$	I	This active low input is the raw data line containing both clock and data pulses from the drive.
20	MO	MOTOR ON	O	Active high output used to enable the spindle motor prior to read, write or stepping operations.
21	WG	WRITE GATE	O	This output is made valid prior to writing on the diskette.
22	WD	WRITE DATA		FM or MFM clock and data pulses are placed on this line to be written on the diskette.
23	$\overline{TR00}$	$\overline{TRACK\ 00}$	I	This active low input informs the WD1772-02 that the drive's R/W heads are positioned over Track zero.
24	$\overline{IP}$	$\overline{INDEX\ PULSE}$	I	This active low input informs the WD1772-02 when the physical index hole has been encountered on the diskette.
25	$\overline{WPRT}$	$\overline{WRITE\ PROTECT}$	I	This input is sampled whenever a Write Command is received. A logic low on this line will prevent any Write Command from executing (internal pull up).

$\overline{CS}$	A1	A0	R/$\overline{W}$ = 1	R/$\overline{W}$ = 0
0	0	0	Status Reg	Command Reg
0	0	1	Track Reg	Track Reg
0	1	0	Sector Reg	Sector Reg
0	1	1	Data Reg	Data Reg

PIN NUMBER	MNEMONIC	SIGNAL NAME	I/O	FUNCTION
26	$\overline{DDEN}$	$\overline{DOUBLE}$ DENSITY ENABLE	I	This input pin selects either single (FM) or double (MFM) density. When $\overline{DDEN}$ = 0, double density is selected (internal pull-up).
27	DRQ	DATA REQUEST	O	This active high output indicates that the Data Register is full (on a Read) or empty (on a Write operation).
28	INTRQ	INTERRUPT REQUEST	O	This active high output is set at the completion of any command, is reset by a read of the Status Register.



WD1772-02 SYSTEM BLOCK DIAGRAM

ARCHITECTURE

The primary sections of the Floppy Disk Formatter are the parallel processor interface and the Floppy disk interface.

Data Shift Register - This 8-bit register assembles serial data from the Read Data input ($\overline{RD}$) during Read operations and transfers serial data to the Write Data output during Write operations.

Data Register – This 8-bit register is used as a holding register during Disk Read and Write operations. In disk Read operations, the assembled data byte is transferred in parallel to the Data Register from the Data Shift Register. In Disk Write operations, information is transferred in parallel from the Data Register to the Data Shift Register.

When executing the Seek Command, the Data Register holds the address of the desired Track position. This register is loaded from the DAL and gated onto the DAL under processor control.

Track Register - This 8-bit register holds the track number of the current Read/Write head position. It is incremented by one every time the head is stepped in and decremented by one when the head is stepped out (towards track 00). The contents of the register are compared with the recorded track number in the ID field during disk Read, Write, and Verify operations. The Track Register can be loaded from or transferred to the DAL. This Register is not loaded when the device is busy.

Sector Register (SR) – This 8-bit register holds the address of the desired sector position. The contents of the register are compared with the recorded sector number in the ID field during disk Read or Write operations. The Sector Register contents can be loaded from or transferred to the DAL. This register is not loaded when the device is busy.

Command Register (CR) - This 8-bit register holds the command presently being executed. This register is not loaded when the device is busy unless the new command is a forced interrupt. The Command Register is loaded from the DAL, but not read onto the DAL.

Status Register (STR) – This 8-bit register holds device Status information. The meaning of the Status bits is a function of the type of command previously executed. This register is read onto the DAL, but not loaded from the DAL.

CRC Logic - This logic is used to check or to generate the 16-bit Cyclic Redundancy Check (CRC). The polynomial is:

$$G(x) = x^{16} + x^{12} + x^5 + 1.$$

The CRC includes all information starting with the address mark and up to the CRC characters. The CRC Register is preset to ones prior to data being shifted through the circuit.

Arithmetic/Logic Unit (ALU) - The ALU is a serial comparator, incrementer, and decrementer and is used for register modification and comparisons with the disk recorded ID field.

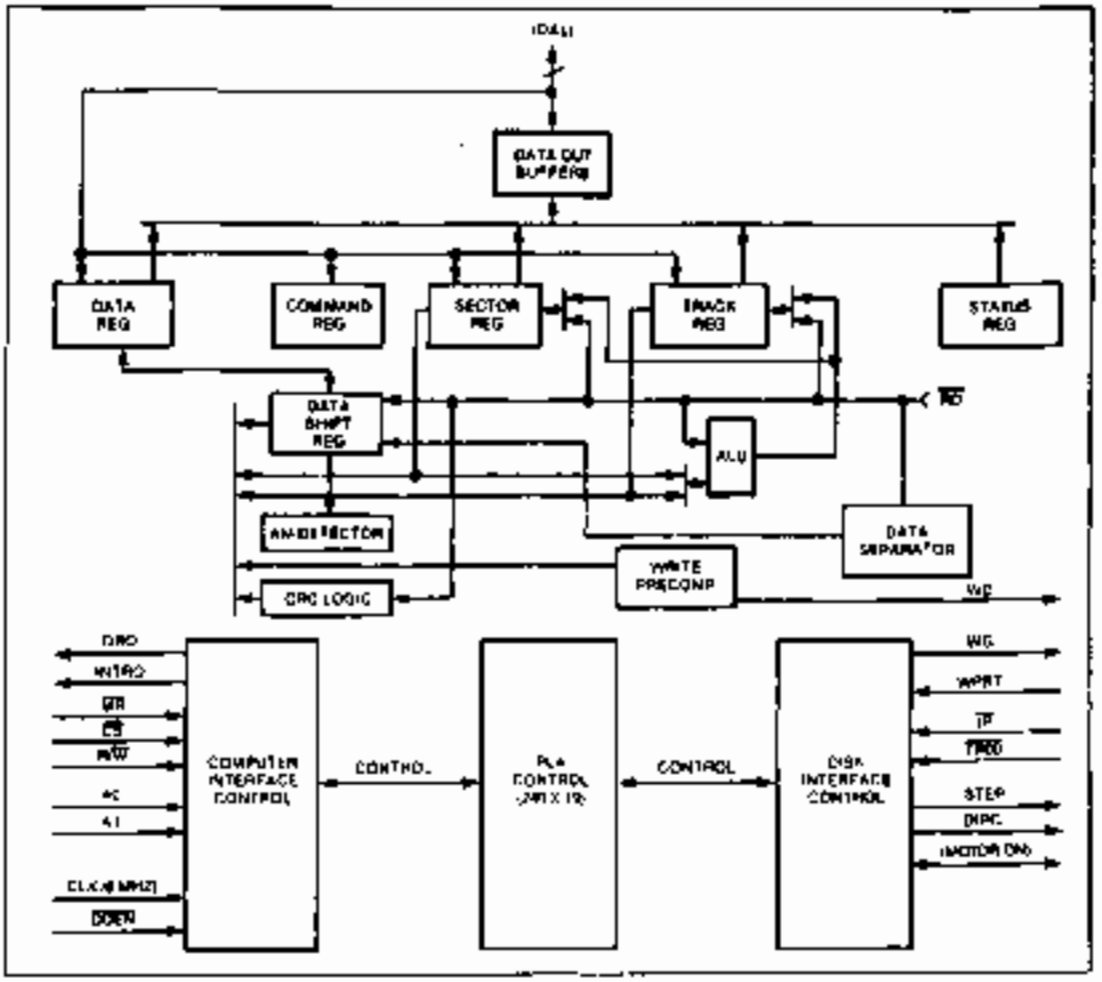


FIGURE 1. WD1772-02 BLOCK DIAGRAM

Timing and Control - All computer and Floppy Disk interface controls are generated through this logic. The internal device timing is generated from an external crystal clock. The WD1772-02 has two different modes of operation according to the state of DDEN.

When $\overline{\text{DDEN}}$ = 0, double density (MFM) is enabled. When DDEN = 1, single density is enabled.

AM Detector – The address mark detector detects ID, data, and index address marks during read and write operations.

Data Separator – A digital phase lock loop (DPLL) of type 2, second order performs the data separator function. DPLL has a filter transfer function used to remove jitter effects thereby achieving adequate window margin. The algorithm used gives performance equal to second order analog designs.

DPLL performance specifications are as follows:

Fc capture range ±8% (min)
TI lock response 4 bytes 00H (max)
W window tolerance 50% for 10E-9 error rate

PROCESSOR INTERFACE

The interface to the processor is accomplished through the eight Data Access Lines (DAL) and associated control signals. The DAL are used to transfer Data, Status, and Control words out of, or into the WD1772-02. The DAL are three state buffers that are enabled as output drivers when $\overline{\text{CS}}$ and R/$\overline{\text{W}}$ = 1 are active or act as input receivers when $\overline{\text{CS}}$ and R/$\overline{\text{W}}$ = 0 are active.

When transfer of data with the Floppy Disk Controller is required by the Host processor, the device address is decoded and $\overline{\text{CS}}$ is made low. The address bits A1 and A0, combined with the signal R/$\overline{\text{W}}$ during a Read operation or Write operation are interpreted as selecting the following registers:

A1	A0	READ (R/$\overline{W}$ = 1)	WRITE (R/$\overline{W}$ = 0)
0	0	Status Register	Command Register
0	1	Track Register	Track Register
1	0	Sector Register	Sector Register
1	1	Data Register	Data Register

After any register is written to, the same register cannot be read from until 16 μsec in MFM or 32 μsec in FM have elapsed.

During Direct Memory Access (DMA) types of data transfers between the Data Register of the WD1772-02 and the processor, the Data Request (DRQ) output is used in Data Transfer control. This signal also appears as status bit 1 during Read and Write operations.

On Disk Read operations, the Data Request bit is activated (set high) when an assembled serial input byte is transferred in parallel to the Data Register. This bit is cleared when the Data Register is read by the processor. If the Data Register is read after one or more characters are lost, by having new data transferred into the register prior to processor readout, the Lost Data bit is set in the Status Register. The Read operations continue until the end of sector is reached.

On Disk Write operations, the Data Request bit is activated when the Data Register transfers its contents to the Data Shift Register, and requires a new data byte. It is reset when the Data Register is loaded with new data by the processor. If new data is not loaded at the time the next serial byte is required by the Floppy Disk, a byte of zeroes is written on the diskette and the Lost Data bit set in the Status Register.

At the completion of every command, an INTRQ is generated. INTRQ is reset by either reading the Status Register or by loading the Command Register with a new command. In addition, INTRQ is generated if a Force Interrupt Command condition is met.

The WD1772-02 has two modes of operation according to the state DDEN. When DDEN = 1, single density is selected. In other case, the CLK input is at 8 MHz.

GENERAL DISK READ OPERATIONS

Sector lengths of 128, 256, 512 or 1024 are obtainable in either FM or MFM formats. For FM, DDEN is placed to logical 1. For MFM formats, DDEN is placed to a logical 0. Sector lengths are determined at format time by the fourth byte in the ID field.

SECTOR LENGTH TABLE	
SECTOR LENGTH FIELD (HEX)	NUMBER OF BYTES IN SECTOR (DECIMAL)
00	128
01	256
02	512
03	1024

The number of sectors per track for the WD1772-02 are from 0 to 244. The number of tracks for the WD1772-02 are 0 to 244.

GENERAL DISK WRITE OPERATION

When writing on the diskette, the WG output is activated, allowing current to flow into the Read/Write head. As a precaution to erroneous writing, the first data byte is loaded into the Data Register in response to a Data Request from the device before the WG is activated.

Writing is inhibited when the $\overline{WPRT}$ input is asserted, in which case any Write Command is immediately terminated, an interrupt is generated and the Write Protect Status bit is set.

For Write operations, the WD1772-02 provides WG to enable a Write condition, and WD which consists of a series of active high pulses. These pulses contain both Clock and Data information in FM and MFM. WD provides the unique missing clock patterns for recording Address Marks.

On the WD1772-02, the Precomp Enable bit in Write Commands allows automatic Write precompensation to take place. The outgoing Write Data stream is delayed or advanced from nominal by 187 nsec according to the following table:

PATTERN				MFM	FM
X	1	1	0	Early	N/A
X	0	1	1	Late	N/A
0	0	0	1	Early	N/A
1	0	0	0	Late	N/A

Next Bit to be sent
Current Bit sending
Previous Bits sent

Precompensation is typically enabled on the innermost tracks where bit shifts usually occur and bit density is at its maximum.

COMMAND DESCRIPTION

The WD1772-02 accepts 11 commands. Command words are only loaded in the Command Register when the Busy Status bit is off (Status bit 0). The one exception is the Force Interrupt Command. Whenever a command is being executed, the Busy Status bit is set. When a command is completed, an interrupt is generated and the Busy status bit is reset. The Status Register indicates whether the completed command encountered an error or was fault free. Commands are divided into four types and are summarized in the following pages.

COMMAND SUMMARY

TYPE	COMMAND	BITS 7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step-in	0	1	0	u	h	V	r_1	r_0
I	Step-out	0	1	1	u	h	V	r_1	r_0
II	Read Sector	1	0	0	m	h	E	0	0
II	Write Sector	1	0	1	m	h	E	P	a_0
III	Read Address	1	1	0	0	h	E	0	0
III	Read Track	1	1	1	0	h	E	0	0
III	Write Track	1	1	1	1	h	E	P	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

FLAG SUMMARY

TYPE I COMMANDS

h = Motor On Flag (Bit 3)

h = 0, Enable Spin-up Sequence
h = 1, Disable Spin-up Sequence

V = Verify Flag (Bit 2)

V = 0, No Verify
V = 1, Verify on Destination Track

r_1, r_0 = Stepping Rate (Bits 1,0)

r_1	r_0	WD1772-02
0	0	6 ms
0	1	12 ms
1	0	2 ms
1	1	3 ms

u = Update Flag (Bit 4)

u = 0, No Update
u = 1, Update Track Register

TYPE II & III COMMANDS

m = Multiple Sector Flag (Bit 4)

m = 0, Single Sector
m = 1, Multiple Sector

H = Motor On Flag (Bit 3)

H = 0, Enable Spin Up Sequence
H = 1, Disable Spin Up Sequence

a_0 = Data Address Mark (Bit 0)

a_0 = Write Normal Data Mark
a_0 = 1, Write Deleted Data Mark

E = 15ms Settling Delay (Bit 2)

E = 0, No Delay
E = 1, Add 15ms Delay

P = Write Precompensation (Bit 1)

P = 0,Enable Write Precomp
P = 1,Disable Write Precomp

TYPE IV COMMANDS

I_3-I_0 Interrupt Condition (Bits 3-0)

I_0 = 1, Not Used
I_1 = 1, Not Used
I_2 = 1, Interrupt on Index Pulse
I_3 = 1, Immediate Interrupt
I_3-I_0 = 0, Terminate without Interrupt

TYPE I COMMANDS

The Type I Commands include the Restore, Seek, Step, Step-In, and Step-Out Commands. Each of the Type I Commands contains a rate field (r_0,r_1), which determines the stepping motor rate.

A 4 µsec (MFM) or 8 µsec (FM) pulse is provided as an output to the drive. For every step pulse issued, the drive moves one track location in a direction determined by the direction output. The chip steps the drive in the same direction it last stepped unless the command changes the direction.

The Direction signal is active high when stepping in and low when stepping out. The Direction signal is valid 24 µsec before the first stepping pulse is generated.

After the last directional step, an additional 15 msec of head settling time takes place if the Verify flag is set in Type I Commands. There is also a 15 msec head settling time if the E flag is set in any Type II or III Command.

When a Seek, Step, or Restore Command is executed, an optional verification of Read/Write head position can be performed by setting bit 2 (V = 1) in the command word to a logic 1. The verification operation begins at the end of the 15 msec settling time after the head is loaded against the media. The track number from the first encountered ID Field is compared against the contents of the Track Register. If the track numbers compare and the ID Field CRC is correct, the verify operation is complete and an INTRQ is generated with no errors. If there is a match but not a valid CRC, the CRC error status bit is set (Status Bit 3), and the next encountered ID Field is read from the disk for the verification operation.

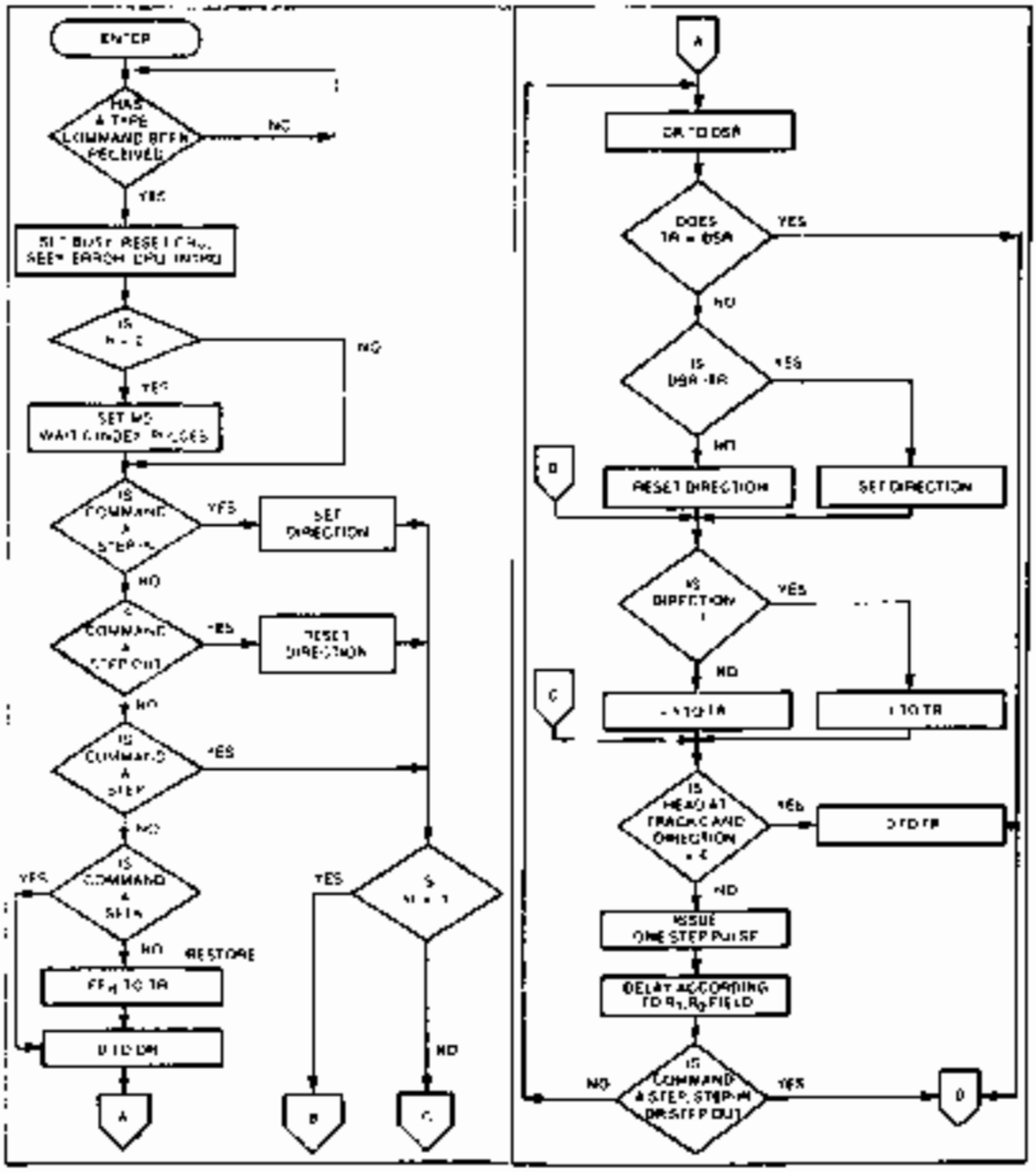
The WD1772-02 finds an ID Field with correct track number and correct CRC within 5 revolutions of the media, or the seek error is set and an INTRQ is generated. If V = 0, no verification is performed.

On the WD1772-02, all commands, except the Force Interrupt Command, are programmed via the h Flag to delay for spindle motor start up time. If the h Flag is not set and the MO signal is low when a command is received, the WD1772-02 forces MO to a logic 1 and waits 6 index pulses before executing the command. At 300 RPM, this guarantees a one second spindle start up time. If after finishing the command, the device remains idle for 9 revolutions, the MO

signal goes back to a logic 0. If a command is issued while MO is high, the command executes immediately, defeating the 5 revolution start up. This feature allows consecutive Read or Write commands without waiting for motor start up each time; the WD1772-02 assumes the spindle motor is up to speed.

RESTORE (SEEK TRACK 0)

Upon receipt of this command, the Track 00 ($\overline{TR00}$) input is sampled. If $\overline{TR00}$ is active low, indicating the Read/Write head is positioned over track 0, the Track Register is loaded with zeroes and an interrupt is generated. If $\overline{TR00}$ is not active low, stepping pulses at a rate specified by the r_1r_0 field are issued until the $\overline{TR00}$ input is activated.



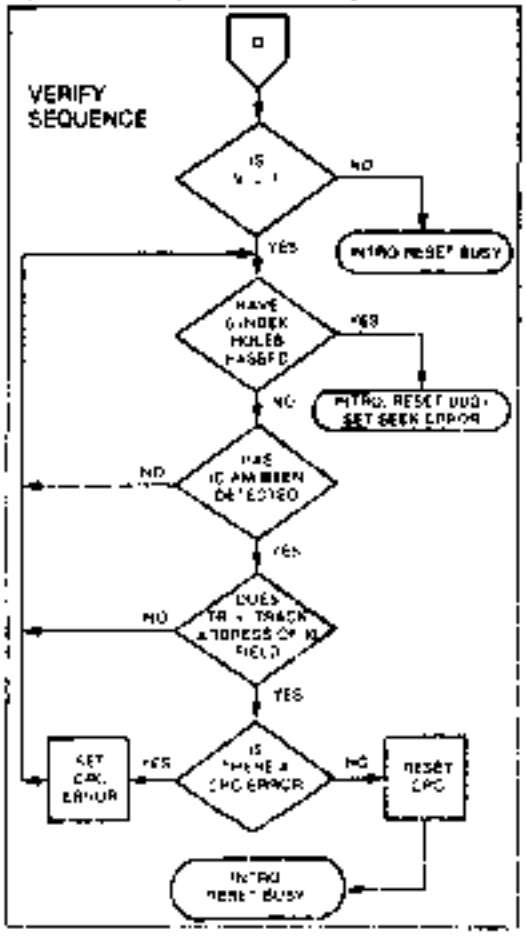
TYPE I COMMAND FLOW

TYPE I COMMAND FLOW

At this time, the Track Register is loaded with zeroes and an interrupt is generated. If the $\overline{TR00}$ input does not go active low after 255 stepping pulses, the WD1772-02 terminates operation, interrupts, and sets the Seek Error status bit, providing the V flag is set. A verification operation also takes place if the V flag is set. The h bit allows the Motor On option at the start of a command.

SEEK

This command assumes that the Track Register contains the track number of the current position of the Read/Write head and the Data Register contains the desired track number. The WD1772-02 updates the Track Register and issues stepping pulses in the appropriate direction until the contents of the Track Register are equal to the contents of the Data Register (the desired track location). A verification operation takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command. Note: When using multiple drives, the Track Register is updated for the drive selected before seeks are issued.



TYPE I COMMAND FLOW

STEP

Upon receipt of this command, the WD1772-02 issues one Stepping Pulse to the disk drive. The stepping motor direction is the same as in the previous step command. After a delay determined by the r_1,r_0 field, a verification takes place if the V flag is on. If the U flag is on, the Track Register is updated. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

STEP-IN

Upon receipt of this command, the WD1772-02 issues one Stepping Pulse in the direction towards the inner most track. If the U flag is on, the Track Register is incremented by one. After a delay determined by the r_1,r_0 field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

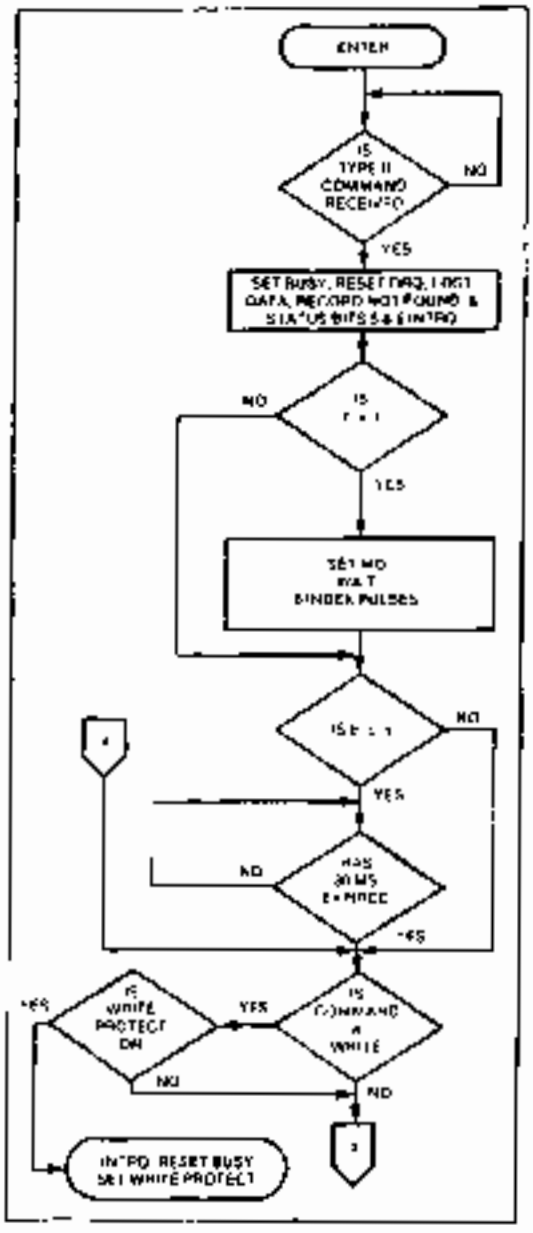
STEP-OUT

Upon receipt of this command, the WD1772-02 issues one stepping pulse in the direction towards track 0. If the U flag is on, the Track Register is decremented by one. After delay determined by the r_1,r_0 field, a verification takes place if the V flag is on. The h bit allows the Motor On option at the start of the command. An interrupt is generated at the completion of the command.

TYPE II COMMANDS

The Type II Commands are the Read Sector and Write Sector commands. Prior to loading the Type II Command into the Command Register, the computer loads the Sector Register with the desired sector number. Upon receipt of the Type II command, the Busy Status bit is set. If the E flag = 1, the command executes after a 15 msec delay.

(When an ID field is located on the disk, the WD1772-02 compares the Track Number on the ID field with the Track Register. If there is not a match, the next encountered ID field is read and a comparison is again made). If there is a match, the Sector Number of the ID field is compared with the Sector Register. If there is no Sector match, the next encountered ID field is read off the disk and comparisons again made. If the ID field CRC is correct, the data field is located and is either written into or read from, depending upon the command.



TYPE II COMMAND

The WD1772-02 finds an ID field with a Track number, Sector number, and CRC within 5 revolutions of the disk, or, the Record Not Found Status bit is set (Status Bit 4) and the command is terminated with an INTRQ.

Each of the Type II Commands contains an m flag which determines if multiple records (sectors) are read or written, depending upon the command. If m = 0 a single sector is read or written and an interrupt is generated at the completion of the command. If m = 1, multiple records are read or written with the Sector Register internally updated so that an address verification occurs on the next record. The WD1772-02 continues to read or write multiple records and updates the Sector Register in numerical ascending sequence until the Sector Register exceeds the number of sectors on the track or until the Force Interrupt Command is loaded into the Command Register, which terminates the command and generates an interrupt.

For example: If the WD1772-02 is instructed to read sector 27 and there are only 26 on the track, the Sector Register exceeds the number available. The WD1772-02 searches for 5 disk revolutions, interrupts out, resets Busy and sets the Record Not Found Status Bit.

READ SECTOR

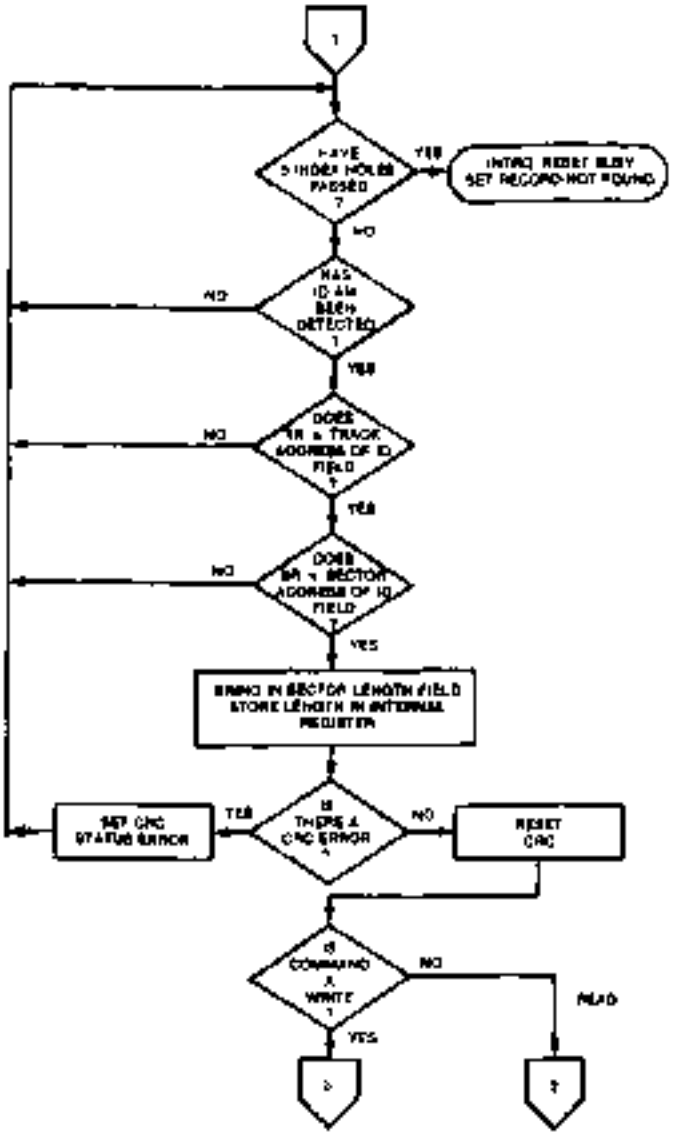
Upon receipt of the Read Sector Command, the Busy Status Bit is set, then when an ID field is encountered that has the correct track number, correct sector number, and correct CRC, the data field is presented to the computer. The Data Address Mark of the data field is found with 30 bytes in single density and 43 bytes in double density of the last ID field CRC byte. If not, the ID field is searched for and verified again followed by the Data Address Mark search. If, after five revolutions the DAM is not found, the Record Not Found Status Bit is set and the operation is terminated. When the first character or byte of the data field is shifted through the DSR, it is transferred to the DR, and DRQ is generated. When the next byte is accumulated in the DSR, it is transferred to the DR and another DRQ is generated. If the computer has not read the previous contents of the DR before a new character is transferred, that character is lost and the Lost Data Status Bit is set. This sequence continues until the complete data field is inputted to the computer. If there is a CRC error at the end of the data field, the CRC Error Status bit is set, and the command is terminated (even if it is a multiple record command).

At the end of the Read operation, the type of Data Address Mark encountered in the data field is recorded in the Status Register (Bit 5) as shown.

STATUS BIT 5	
1	Deleted Data Mark
0	Data Mark

WRITE SECTOR

Upon receipt of the Write Sector Command, the Busy Status Bit is set. When an ID field is encountered that has the correct track number, correct sector number, and correct CRC, a DRQ is generated. The WD1772-02 counts off 11 bytes in single density and 22 bytes in double density from the CRC field and the WG

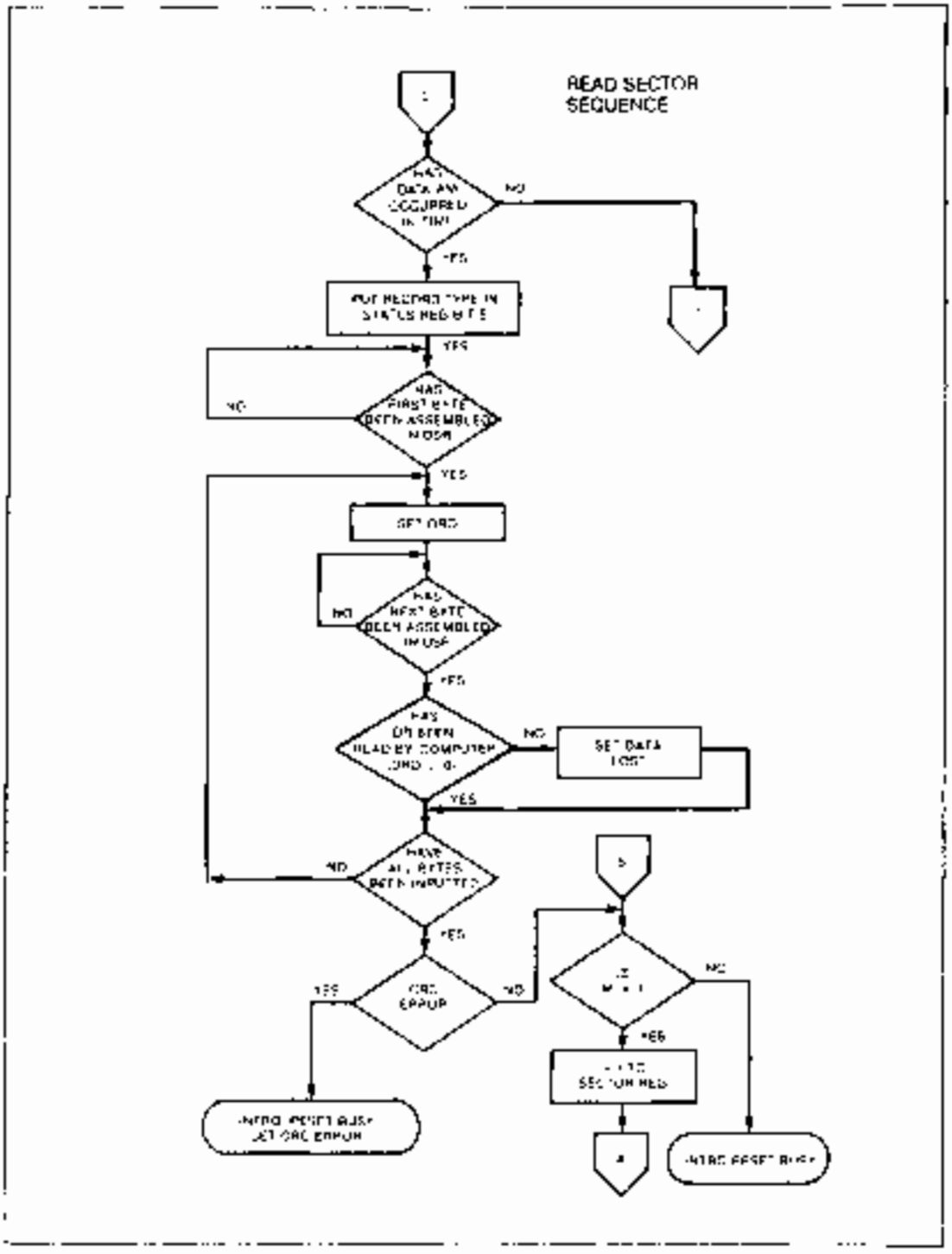


TYPE II COMMAND

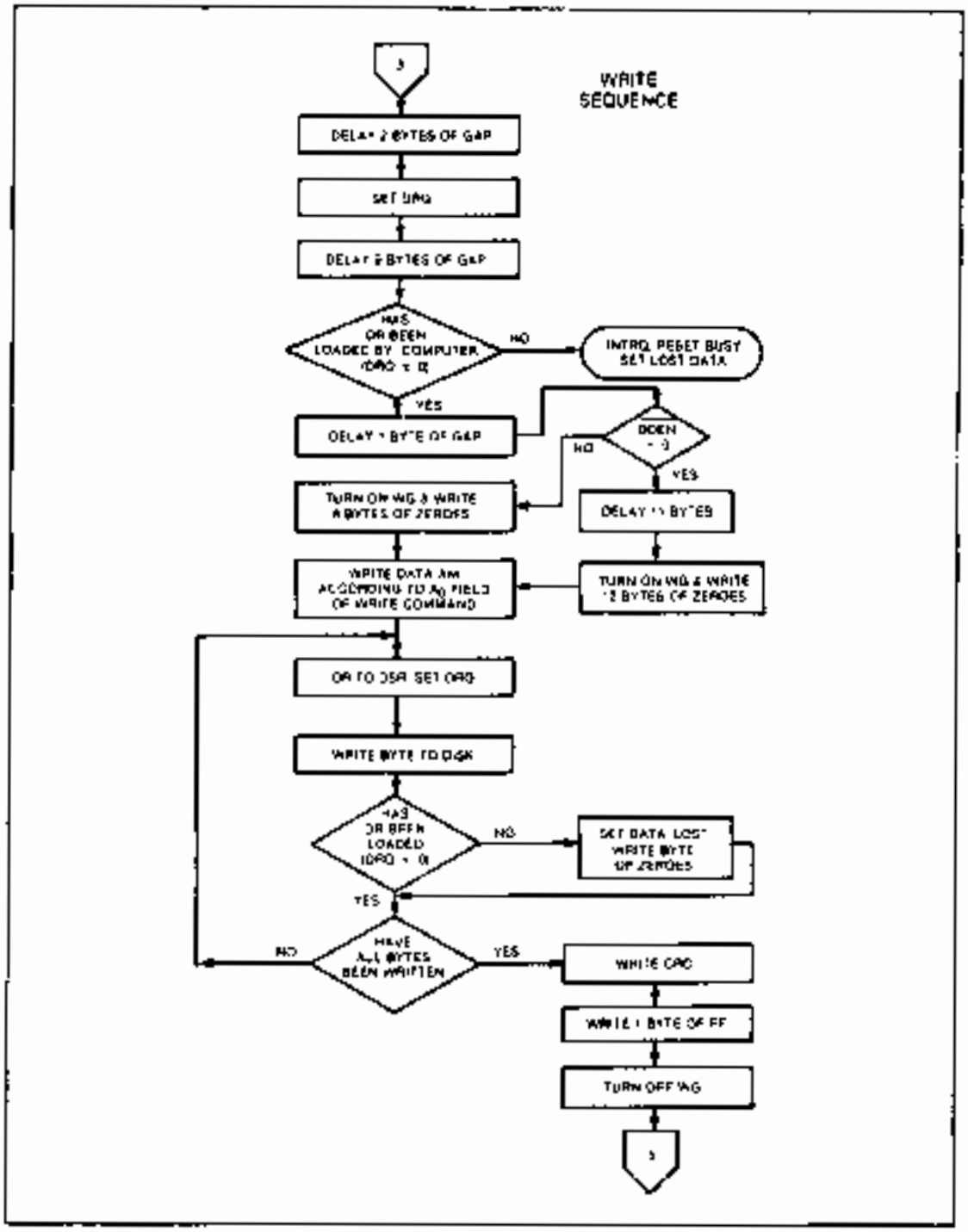
output is made active if the DRQ is serviced (i.e., the DR is loaded by the computer). If DRQ is not serviced, the command is terminated and the Lost Data Status Bit is set. If the DRQ is serviced, the WG is made active and six bytes of zeroes in single density and 12 bytes in double density are written on the disk. The Data Address Mark is then written on the disk as determined by the a_0 field of the command as shown:

a_0	DATA ADDRESS MARK (BIT 0)
1	Deleted Data Mark
0	Data Mark

The WD1772-02 writes the data field and generates DRQ's to the computer. If the DRQ is not serviced in time for continuous writing, the Lost Data Status



TYPE II COMMAND



TYPE II COMMAND

Bit is set and a byte of zeroes is written on the disk. The command is not terminated. After the last data byte is written on the disk, the two-byte CRC is computed internally and written on the disk, followed by one byte of logic ones in FM or in MFM. The WG output is then deactivated. INTRQ sets 24 μsec (MFM) after the last CRC byte is written. For partial sector writing, the proper method is to write data and fill the balance with zeroes.

TYPE III COMMANDS

Read Address

Upon receipt of the Read Address Command, the Busy Status Bit is set. The next encountered ID field is then read in from the disk, and six data bytes of the ID field are assembled and transferred to the DR, and a DRQ is generated for each byte. The six bytes of the ID field are shown:

TRACK ADDR	SIDE NUMBER	SECTOR ADDR	SECTOR LENGTH	CRC 1	CRC 2
1	2	3	4	5	6

Although the CRC characters are transferred to the computer, the WD1772-02 checks for validity and the CRC error status bit is set if there is a CRC error. The Track Address of the ID field is written into the sector register so that a comparison can be made by the user. At the end of the operation, an interrupt is generated and the Busy Status is reset.

Read Track

Upon receipt of the Read Track Command, the head is loaded and the Busy Status bit is set. Reading starts with the leading edge of the first encountered index pulse and continues until the next index pulse. All Gap, Header, and data bytes are assembled and transferred to the data register and DRQ's are generated for each byte. The accumulation of bytes is synchronized to each address mark encountered. An interrupt is generated at the completion of the command.

This command has several characteristics which make it suitable for diagnostic purposes. These characteristics are: no CRC checking is performed, gap information is included in the data stream; and the Address Mark Detector is on for the duration of the command. Because the AM detector is always on, write splices or noise may cause the chip to look for an AM.

The ID AM, ID field, ID CRC bytes, DAM, Data, and Data CRC Bytes for each sector are correct. The Gap Bytes may be read incorrectly during write-splice time because of synchronization.

WRITE TRACK FORMATTING THE DISK

Data and gap information are provided at the computer interface. Formatting the disk is accomplished by positioning the head over the desired track number and issuing the Write Track Command.

Upon receipt of the Write Track Command, the Busy Status Bit is set. Writing starts with the leading edge of the first encountered index Pulse and continues until the next Index Pulse, at which time the interrupt is activated. The Data Request is activated immediately upon receiving the command, but writing does not start until after the first byte is loaded into the Data Register (if the DR is not loaded within three byte times, the operation is terminated making the device Not Busy, the Lost Data Status Bit is set, and the interrupt is activated). If a byte is not present in the DR when needed, a byte of zeroes is substituted.

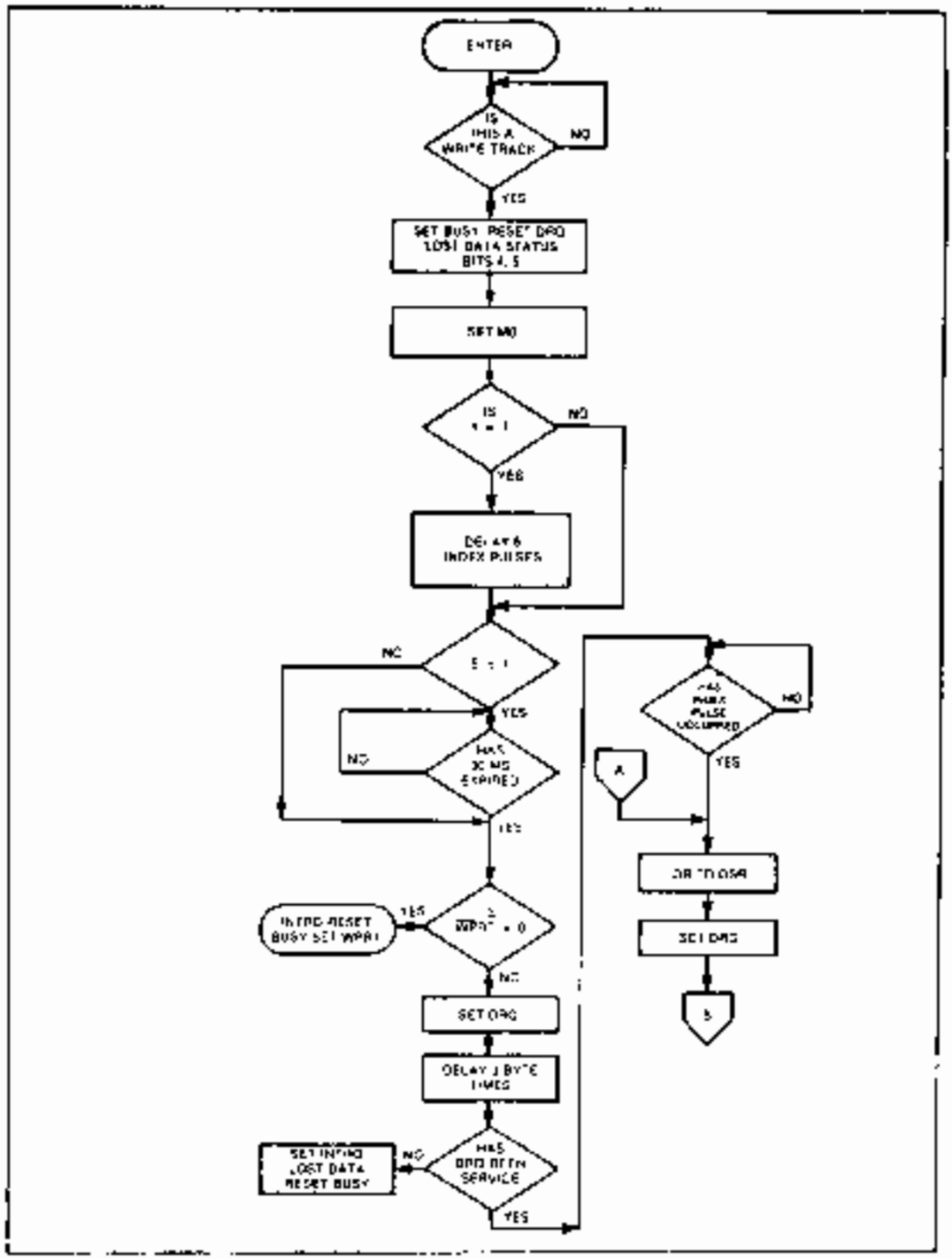
This sequence continues from one Index Pulse to the next. Normally, whatever data pattern appears in the Data Register is written on the disk with a normal clock pattern. However, if the WD1772-02 detects a data pattern of F5 through FE in the Data Register, this is interpreted as Data Address Marks with missing clocks or CRC generation.

The CRC generator is initialized when any data byte from F8 to FE is transferred from the DR to the DSR in FM or by receipt of F5 in MFM. An F7 pattern generates two CRC characters in FM or MFM. As a consequence, the patterns F5 through FE do not appear in the gaps, data field, or ID fields. Also, CRC's are generated by an F7 pattern.

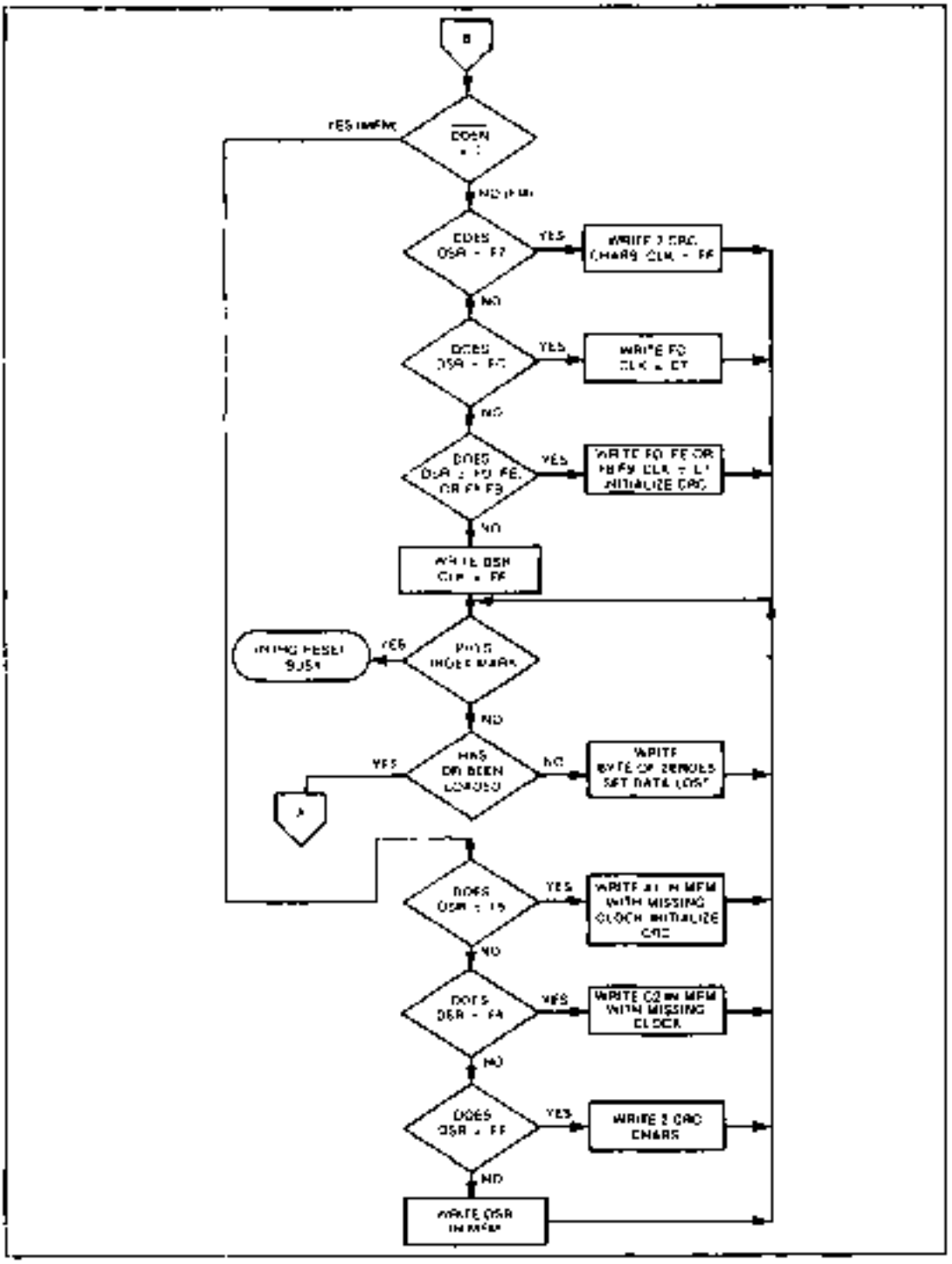
Disks are formatted in IBM 3740 or System 34 formats with sector lengths of 128, 256, 512, or 1024 bytes.

DATA PATTERN IN DR (HEX)	IN FM ($\overline{DDEN}$ = 1)	IN MFM ($\overline{DDEN}$ = 0)
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* in MFM, Preset CRC
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, CLK = C7, Preset CRC	Write F8 thru FB, in MFM
FC	Write FC with CLK = D7	Write FC in MFM
FD	Write FD with CLK = FF	Write FD in MFM
FE	Write FE, CLK = C7, Preset CRC	Write FE in MFM
FF	Write FF with CLK = FF	Write FF in MFM

*Missing clock transition between bits 4 and 5.
**Missing clock transition between bits 3 and 4.



TYPE III COMMAND WRITE TRACK



TYPE III COMMAND WRITE TRACK

TYPE IV COMMANDS

The Forced Interrupt Command is used to terminate a multiple sector read or write command or to ensure Type I status in the Status Register. This command is loaded into the Command Register at any time. If there is a current command under execution, (Busy Status Bit set), the command is terminated and the Busy Status Bit reset.

The lower four bits of the command determine the conditional interrupt as follows:

I_0 = Not used

I_1 = Not Used

I_2 = Every Index Pulse

I_3 = Immediate Interrupt

The conditional interrupt is enabled when the corresponding bit positions of the command (I_3-I_0) are set to a 1. When the condition for interrupt is met, the INTRQ line goes high signifying that the condition specified has occurred. If I_3-I_0 are all set to zero (Hex D0), no interrupt occurs but any command presently under execution is immediately terminated. When using the immediate interrupt condition (I_3 = 1) an interrupt is immediately generated and the current command terminated. Reading the status or writing to the command register does not automatically clear the interrupt. The Hex D0 is the only command that enables the immediate interrupt (Hex D8) to clear on a subsequent load Command Register or Read Status Register operation. Follow a Hex D8 with D0 command.

Wait 16 μsec (double density) or 32 μsec (single density) before issuing a new command after issuing a forced interrupt. Loading a new command sooner than this nullifies the forced interrupt.

Forced interrupt stops any command at the end of an internal micro-instruction and generates INTRQ when the specified condition is met. Forced interrupt waits until ALU operations in progress are complete (CRC calculations, compares, etc.).

Status Register

Upon receipt of any command, except the Force Interrupt Command, the Busy Status Bit is set and the rest of the status bits are updated or cleared for the new command. If the Force Interrupt Command is received when there is a current command under execution, the Busy Status Bit is reset, and the rest of the status bits are unchanged. If the Force Interrupt Command is received when there is not a current command under execution, the Busy Status Bit is reset and the rest of the status bits are updated or cleared. In this case, Status reflects the Type I commands.

The user has the option of reading the Status Register through program control or using the DRQ line with DMA or interrupt methods. When the Data Register is read, the DRQ bit in the Status Register and the DRQ line are automatically reset. A write to the Data Register also causes both DRQ's to reset.

The Busy Bit in the status may be monitored with a user program to determine when a command is complete, in lieu of using the INTRQ line. When using the INTRQ, a Busy Status check is not recommended because a read of the Status Register to determine the condition of busy resets the INTRQ line.

The format of the Status Register is shown below:

(Bits)							
7	6	5	4	3	2	1	0
S7	S6	S5	S4	S3	S2	S1	S0

Because of internal sync cycles, certain time delays are observed when operating under programmed I/O, as shown:

Operation	Next Operation	Delay Req'd. FM	Delay Req'd. MFM
Write to Command Reg.	Read Busy Bit (Status Bit 0)	48μsec	24μsec
Write to Command Reg	Read Status Bits 1-7	64μsec	32μsec
Write Register	Read Same Register	32μsec	16μsec

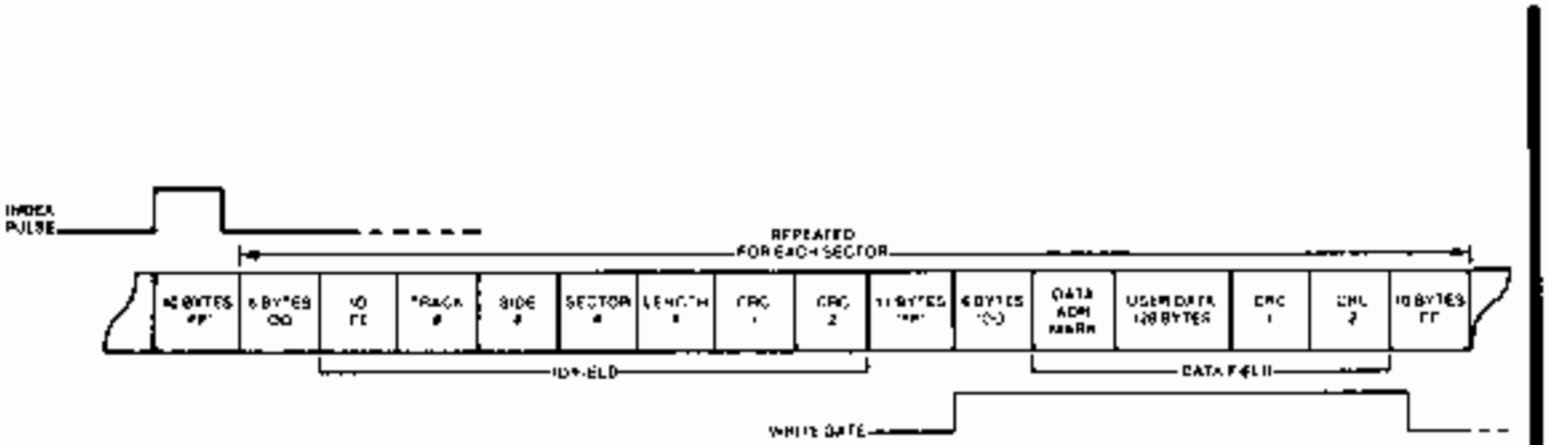
RECOMMENDED - 128 BYTES/SECTOR

The recommended single-density format with 128 bytes/sector is shown below. In order to format a diskette, the user issues the Write Track Command, and loads the Data Register with the following values. For every byte to be written, there is one Data Request.

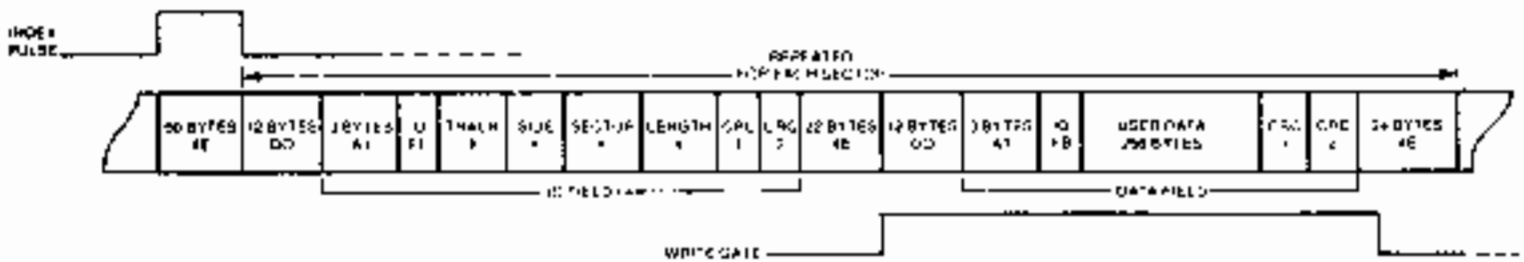
NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF (or 00)
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 or 01)
1	Sector Number (1 thru 1A)
1	00 (Sector Length)
1	F7 (2 CRC's written)
11	FF (or 00)
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
10	FF (or 00)
368**	FF (or 00)

*Write bracketed field 16 times.

**Continue writing until WD1772-02 interrupts out. Approx. 369 bytes.



SINGLE DENSITY FORMAT



DOUBLE DENSITY FORMAT

WD1772-02

256 BYTES/SECTOR

Shown above is the recommended dual-density format with 256 bytes/sector. In order to format a diskette, the user issues the Write Track Command and loads the Data Register with the following values. For every byte to be written, there is one Data Request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F5 (Writes A1)
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01 (Sector Length)
1	F7 (2 CRC's written)
22	4E
12	00
3	F5 (Writes A1)
1	FB (Data Address Mark)
256	DATA
1	F7 (Data Address Mark)
24	4E
668**	4E

*Write bracketed field 16 times.
**Continue Writing until WD1772-02 interrupts out
Approx 666 bytes

1. Non-Standard Formats

Variations in the recommended formats are possible to a limited extent if the following requirements are met:

1) Sector size must be 128, 256, 512 or 1024 bytes.
2) Gap 2 cannot be varied from the recommended format.
3) 3 bytes of A1 must be used in MFM.

In addition the Index Address Mark is not required for operation by the WD1772-02. Gap 1, 3 and 4 lengths are as short as 2 bytes for WD1772-02 operation, however PLL lock up time, motor speed variation, write splice area, etc. adds more bytes to each gap to achieve proper operation. For highest system reliability, use the recommended format.

	FM	MFM
Gap I	16 bytes FF	32 bytes 4E
Gap II	11 bytes FF	22 bytes 4E
	6 bytes 00	12 bytes 00
		3 bytes A1
Gap III**	10 bytes FF	24 bytes 4E
	4 bytes 00	8 bytes 00
		3 bytes A1
Gap IV	16 bytes FF	16 bytes 4E

*Byte counts must be exact.
**Byte counts are minimum, except exactly 3 bytes of A1 must be written.

STATUS REGISTER DESCRIPTION

BIT NAME	MEANING
S7 MOTOR ON	This bit reflects the status of the Motor On output.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Track: Not Used. On any Write: It indicates a Write Protect. This bit is reset when updated.
S5 RECORD TYPE/SPIN-UP	When set, this bit indicates that the Motor Spin-Up sequence has completed (5 revolutions) on Type I commands. Type 2 & 3 commands, this bit indicates record Type: 0 = Data Mark, 1 = Deleted Data Mark.
S4 RECORD NOT FOUND (RNF)	When set, it indicates that the desired track, sector, or side were not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, an error is found in one or more ID fields; otherwise it indicates error data field. This bit is reset when updated.
S2 LOST DATA/ BYTE	When set, it indicates the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated. On Type I commands, this bit reflects the status of the TR00 signal.
S1 DATA REQUEST/ INDEX	This bit is a copy of the DRQ output. When set, it indicates the DR is full on a Read Operation or the DR is empty on a Write operation. This bit is reset to zero when updated. On Type 1 commands, this bit indicates the status of the IP signal.
S0 BUSY	When set, command is under execution. When reset, no command is under execution.

DC ELECTRICAL CHARACTERISTICS

MAXIMUM RATINGS

Storage Temperature . . . . . = 55°C(−67°F) to +125°C (257°F)
Operating Temperature . . . . . 0°C(32°F) to 70°C (158°F) Ambient
Maximum Voltage to Any Input with Respect to V_{SS} . . . . . +7V to −0.5V

NOTE

Maximum limits indicate where permanent device damage occurs. Continuous operation at these limits is not intended and should be limited to those conditions specified in the DC Operating Characteristics.

DC OPERATING CHARACTERISTICS

T_A = 0°C(32°F) to 70°C (158°F), V_{SS} = 0V, V_{CC} = +5V ± .25V

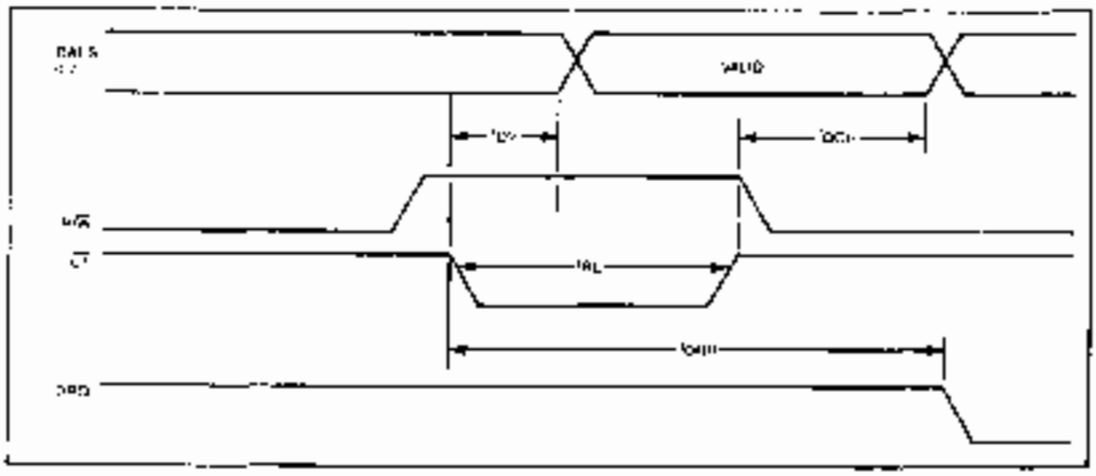
SYMBOL	CHARACTERISTIC	MIN	MAX	UNITS	CONDITIONS
I_{IL}	Input Leakage		10	µA	V_{IN} V_{CC}
I_{OL}	Output Leakage		10	µA	V_{OUT} V_{CC}
V_{IH}	Input High Voltage	2.0		V	
V_{IH}	Input Low Voltage		0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_O = −100 µA
V_{OL}	Output Low Voltage		0.40	V	I_O = 1.6 mA
P_D	Power Dissipation		.75	W	
I_{PU}	Internal Pull-Up	100	1700	µA	V_{IN} 0V
I_{CC}	Supply Current	75(Typ)	150	mA	

DPLL SPECIFICATION

F_C Capture Range ± 8% (min)
T_l Lock Response 4 bytes 00 hex (max)
WI Window Tolerance 50% for 10E-9 Error Rate (min)

AC TIMING CHARACTERISTICS

TA = 0°C (32°F) to 70°C (158°F), V_{SS} = 0V, V_{CC} = +5V ± .25V

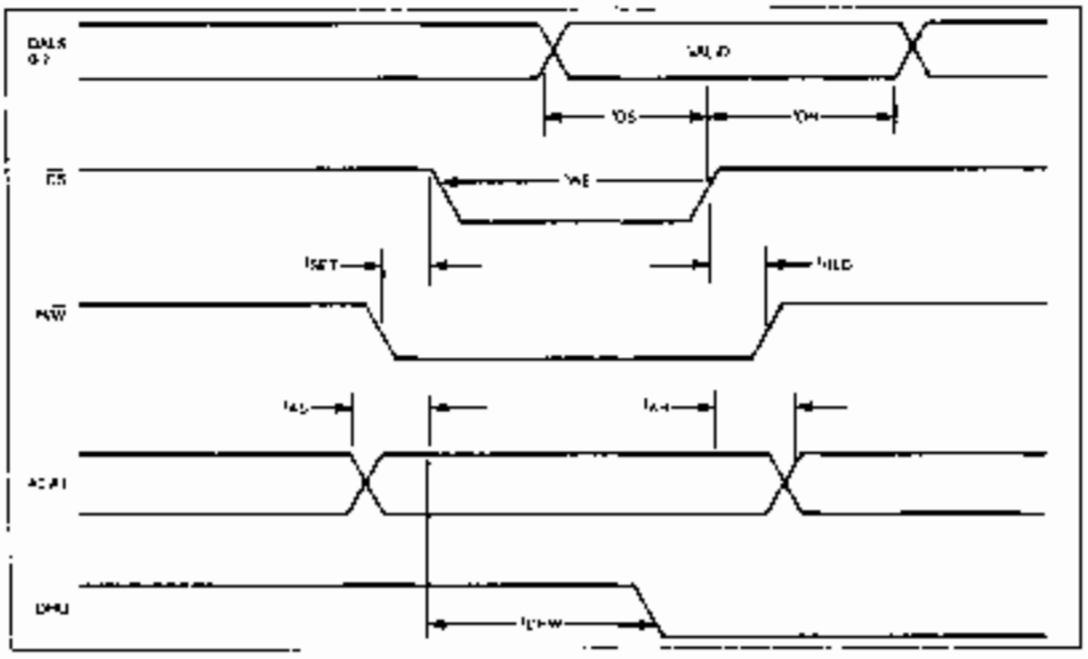


READ ENABLE TIMING

READ ENABLE TIMING - $\overline{RE}$ such that, R/$\overline{W}$ = 1, CS = 0

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
t_{RE}	$\overline{RE}$ Pulse Width of $\overline{CS}$	200			nsec	C_L = 50 pf
t_{DRR}	DRQ Reset from $\overline{RE}$		200	300	nsec	
t_{DV}	Data Valid from $\overline{RE}$		100	200	nsec	C_L = 50 pf
t_{DRH}	Data Hold from $\overline{RE}$	20		150	nsec	C_L = 50pf
	INTRQ Reset from $\overline{RE}$			6	µsec	

Note: Worst case service time for DRQ is 23.5 usec for MFM and 47.5 usec for FM



WRITE ENABLE TIMING

WRITE ENABLE TIMING – $\overline{WE}$ such that R/$\overline{W}$ = 0, $\overline{CS}$ = 0

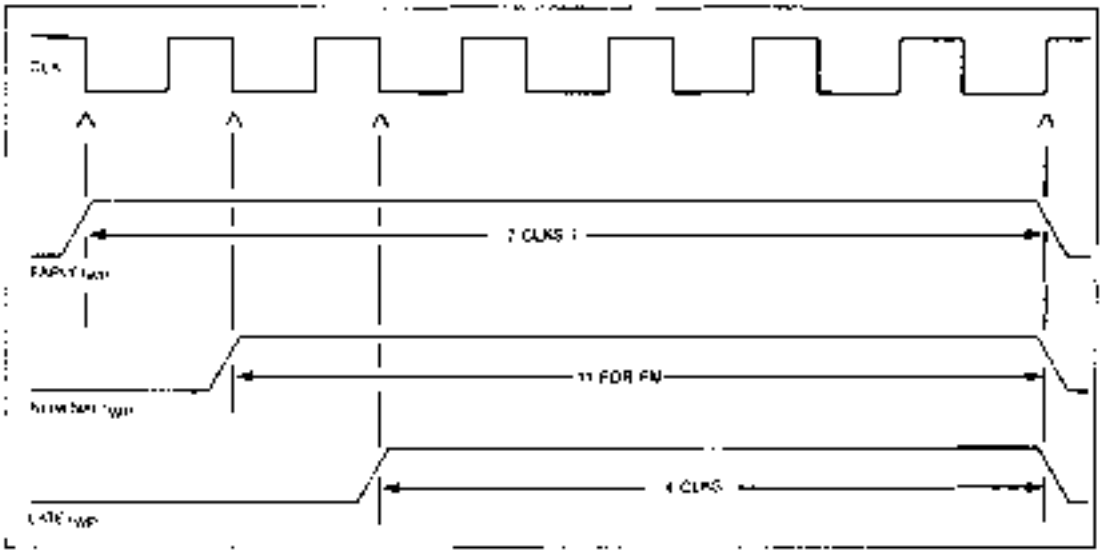
SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
t_{AB}	Setup ADDR to $\overline{CS}$	50			nsec	
t_{SET}	Setup R/$\overline{W}$ to $\overline{CS}$	0			nsec	
t_{AH}	Hold ADDR from $\overline{CS}$	10			nsec	
t_{HLD}	Hold R/$\overline{W}$ from $\overline{CS}$	0			nsec	
t_{WE}	$\overline{WE}$ Pulse Width	200			nsec	
t_{DRW}	DRQ Reset from $\overline{WE}$		100	200	nsec	
t_{DS}	Data Setup to $\overline{WE}$	150			nsec	
t_{DH}	Data Hold from $\overline{WE}$	0			nsec	
	INTRQ Reset from $\overline{WE}$			8	µsec	

READ DATA TIMING:

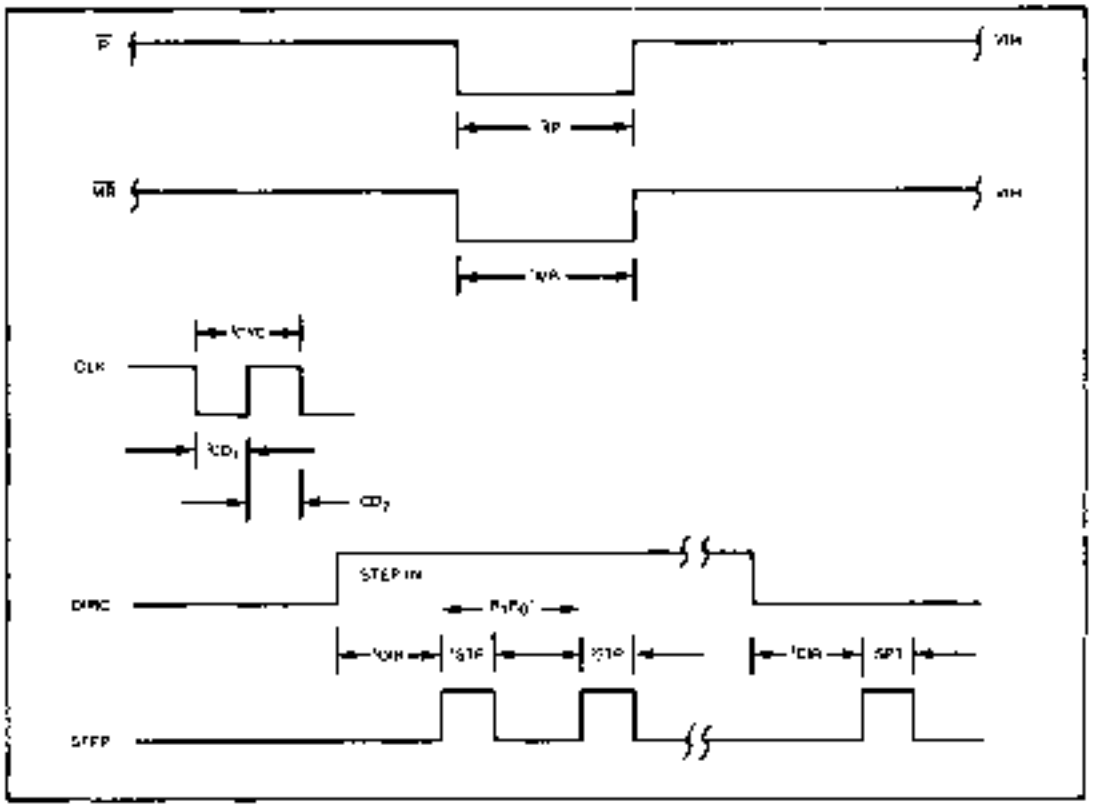
CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
$\overline{Raw\ Read}$ Pulse Width	200		3	µsec	MFM
	400		3		FM
$\overline{Raw\ Read}$ Cycle Time	3			µsec	

WRITE DATA TIMING:

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
	Write Gate to Write Data		4		µsec	FM
			2		µsec	MFM
	Write Data Cycle Time		4,6,8		µsec	
	Write Gate off from WD		4		µsec	FM
			2		µsec	MFM
t_{WP}	Write Data Pulse Width		820		nsec	Early MFM
			890		nsec	Nominal MFM
			570		nsec	Late MFM
			1.38		µsec	FM



WRITE DATA TIMING



MISCELLANEOUS TIMING

MISCELLANEOUS TIMING:

SYMBOL	CHARACTERISTIC	MIN	TYP	MAX	UNITS	CONDITIONS
t_{CL}	Clock Duty (low)	50	67		nsec	
t_{CH}	Clock Duty (high)	50	67		nsec	
t_{STP}	Step Pulse Output		4		µsec	MFM
			8			FM
t_{DIR}	Dir Setup to Step		24		µsec	MFM
			48			FM
t_{MR}	Master Reset Pulse Width	50			µsec	
t_{IP}	Index Pulse Width	20			µsec	

WESTERN DIGITAL
CORPORATION

WD1772-02 Floppy Disk Formatter/Controller Family Application Notes

INTRODUCTION

To meet the demand for a low cost compact LSI Floppy Disk Controller device, Western Digital has developed the WD1772-02. The WD1772-02 is a NMOS Floppy Disk Controller device that incorporates the FD179X, a digital data separator and write precompensation circuitry all in a single chip. The device offers soft sector formatting, selectable stepping rates, automatic track seek with verify, and variable sector lengths. The WD1772-02 comes in a 28-pin dual-in-line package or quad pack and operates from a single 5 volt only power supply.

APPLICATIONS

The Mini-Floppy Controller is targeted for the low cost sector of the disk drive market, where digital data separation is preferred over analog phase lock loop. Included in this market are Personal Computers, Portable Computers and Small Business Computers.

HOST INTERFACING

Interfacing to a Host processor is accomplished through the eight bit bi-directional Data Access Lines (DAL) and associated control lines. The DAL is used to transfer data, status and control words out of or into WD1772-02. The DAL having three states enabled as an output when Chip Select (CS) is active low and Read/Write (R/W) is high or as input receiver when CS and R/W is low. When transfer of data with the device is required by the Host CS is made low. The address bits A0 and A1 combined with the R/W line select the register and the direction of data.

During Direct Memory Access (DMA) data transfers between the WD1772-02 and Host Memory, the Data Request (DRQ) line is used in Data Transfer Control. This signal also appears as status bit 1 during Read/Write operations. On Disk Read operations the DRQ is active when an assembled byte is present in the Data Register, then reset when read by the Host. If the Host fails to read the Data Register before the following byte is assembled in the Data Register, the lost data bit is set in Status Register.

At the completion of every command INTRQ is asserted. INTRQ is de-asserted by either reading the status or by loading the Command Register.

DISKETTE DRIVE INTERFACING

The WD1772-02 has two modes of operation depending on the state of DDEN, regardless of the state DDEN the CLK input remains at 8 MHz. Disk Reads with sector lengths of 128, 256, 512 and 1024 byte sector in both FM or MFM from diskettes is accomplished via the internal digital data separator. Disk Write operation in MFM on inner tracks may require write precompensation. Write precompensation is enabled when bit 4 = 0, in the Write command and a precompensation value of 187 nsec is produced.

The diskettes spindle motor is controlled by bit 3 of any Type I, II or III command, upon receiving a command with bit 3 = 0, the spin up sequence is enabled.

GENERAL INFORMATION

A +5 volt supply ±5% is used as V_{CC}, and the clock input requires a free running 50% duty cycle at 8 MHz ± 0.1%.

ISCSI-1 FIRMWARE

USER'S MANUAL

Fifth Edition
April 1988

FORCE COMPUTERS Inc./GmbH
All Rights Reserved

This document shall not be duplicated, nor its contents used for any purpose unless written permission has been given.

Copyright by FORCE Computers

This page was intentionally left blank

TABLE OF CONTENTS

1. GENERAL DESCRIPTION OF THE ISCSI-1 FIRMWARE.....1-1

1.1 Introduction and Global Notes.....1-1

1.2 Features of the ISCSI-1 Handling Firmware.....1-2

1.3 Theory of Operation.....1-3

1.3.1 Global Notes.....1-3

1.3.2 The Single Command Mode.....1-3

1.3.3 The Command Chain Mode.....1-3

1.3.4 The SCSIbus Target Mode.....1-3

1.3.5 Data Hashing and Blocking.....1-4

1.4 The Power Up Procedure.....1-6

1.5 The ISCSI-1 Onboard Self-test.....1-6

1.6 ISCSI-1 Default Setups.....1-7

1.7 Memory Layout.....1-8

2. THE ISCSI-1 SOFTWARE INTERFACE.....2-1

2.1 General Information.....2-1

2.2 The Command RAM Layout.....2-1

2.3 Command Chaining.....2-2

2.4 Special Ram Locations.....2-3

2.5 Floppy Disk Support.....2-3

3. ISCSI-1 COMMAND DESCRIPTION.....3-1

3.1 The Command Structure and Philosophy.....3-1

3.2 Command Overview.....3-2

TABLE OF CONTENTS

3.3 Detailed Command Description.....3-5

- SETOFFS - Set VMEbus address offset.....3-5
- DEFOFFS - Define physical block offset.....3-7
- RESET - Reset the firmware to power up state.....3-8
- SUNPARM - Set logical units parameter.....3-9
- BUFFWR - Enable/disable buffering for write.....3-12
- WRPARAL - Enable/disable write to parallel disks...3-13
- MAINIT - Global controller initialization.....3-14
- SFTIME - Set flush buffers time interval.....3-16
- DEBLOCK - Set sector deblocking mode.....3-18
- GETCHR - Read one byte from logical block.....3-19
  PUTCHR - Write one byte to logical block.....3-21
- GETBLOC - Read a logical block from unit.....3-23
- PUTBLOC - Write a logical block to unit.....3-25
- GETSTR - Read a delimited string from a block.....3-27
- PUTSTR - Write a delimited string to a block.....3-29
- GETCNT - Read a counted string from a block.....3-31
  PUTCNT - Write a counted string to a block.....3-33
  CTLSTAT - Return general controller status.....3-35
- CHKTARG - Check for and test existing targets.....3-37
- GUNPARM - Get unit parameters.....3-39
- EXPROG - Execute user program in DPR.....3-41
- BACKUP - Backup logical unit.....3-43
- FLUSH - Flush modified buffers.....3-45
- FORMAT - Format a logical unit.....3-47
- FTRACK - Format a track on local floppy disk.....3-49

TABLE OF CONTENTS

- COMPARE - Compare data between two logical units...3-52
- LOCKUN - Lock local unit for SCSIbus access.....3-54
- FREEUN - Free local unit for SCSIbus access.....3-55
- COPY - Copy data between logical units.....3-56
- CHAIN - Enter command chaining mode.....3-58
- SEARCH - Search data pattern on a logical unit....3-59
- CLRHASH Clear hashing buffer list.....3-61
- TRSPMOD - Handle SCSI command in transparent mode..3-62
- SCSIRST - Perform SCSIbus reset.....3-63
- TARGMOD - Enable/disable controller target mode....3-64
- TARGINT - Set target interrupt mode.....3-65
- ABORT - Abort command execution.....3-66

3.5 ISCSI-1 Target Mode Description.....3 67

3.5.1 Overview.....3-67

3.5.2 Detailed Description of the Supported Commands...3-68

- TEST UNIT READY.....3-68
- REQUEST SENSE.....3-69
- FORMAT UNIT.....3-70
- READ/RECEIVE.....3-71
- WRITE/SEND.....3-72
- INQUIRY.....3-73

4. THE ISCSI-1 INTERRUPT STRUCTURE.....4-1

This page was intentionally left blank

LIST OF TABLES

Table 1-1: Default Setup.....1-6

Table 1-2: ISCSI-1 Memory Layout from local Sight.....1-7

Table 1-3: ISCSI-1 Memory Layout from System/VMEbus Sight....1-8

Table 3-1: Command Overview.....3-3

LIST OF FIGURES

Figure 1-1: Logical Block Diagram of the ISCSI-1 Firmware....1-5

Figure 2-1: Command chaining diagram.....2-4

LIST OF PROGRAMMING EXAMPLES

SETOFFS.....3-6

RESET.....3-8

SPARM1.....3-11

MAINIMIT.....3-15

SETIME.....3-17

GETCHR.....3-20

PUTCHR.....3-22

RDBLOCK.....3-24

WRBLOCK.....3-26

GETSTR.....3-28

PUTSTR.....3-30

GETCNT.....3-32

PUTCNT.....3-34

CTLSTAT.....3-36

CHKTARG.....3-38

This page was intentionally left blank

LIST OF PROGRAMMING EXAMPLES

GUNPARM 3-40

EXPROC 3-42

BACKUP 3-44

FLUSH 3-46

FORMAT 3-48

FTRACK 3-51

COMPARE 3-53

COPY 3-57

ABORT 3-66

This page was intentionally left blank

1. GENERAL DESCRIPTION OF THE ISCSI-1 FIRMWARE

1.1 INTRODUCTION

The ISCSI-1 Handling Firmware is an EPROM resident package which controls all ISCSI-1 features and supports execution of powerful macro I/O and initialization commands. The Handling Firmware supports the full SCSI standard, either as an initiator or as a target.

The ISCSI-1 Handling Firmware consists of:

- I/O initialization and ISCSI-1 self-test routines.

  Modules which support SCSI initiator and target modes.

- Command chaining mode.

- Logical Winchester partitioning routines.

- Block buffering and hashing routines.

- Physical sector deblocking routines.

  Handling for up to four floppy disk drives

  Command execution routines for:

  SCSIbus control

  - Board and firmware initialization
  - System, I/O and device status information
  - Byte, string and block I/O
  - Starting and executing user programs

All programming examples in this manual are written in M680000 Assembler language under the PDOS operating system.

1.2 Features of the ISCSI-1 Handling Firmware

The SCSIbus interface:

- Full support of the SCSI standard, either as initiator or as target.
- Emulation of the SCSI commands COPY, COMPARE and SEARCH.
- High speed data transfer of up to 1.42 Mbytes/second.
- BACKUP command and automatic run time backup to a secondary unit.
- Automatic handling of REQUEST SENSE.
- Transparent mode to the SCSI interface for vendor-unique commands or software debugging.
- Easy system interconnection via the SCSIbus.

Global ISCSI-1 features:

- Supports multiprocessor access via double software interface.
- Hashing buffers are installed to upgrade the performance
- Up to five logical units are under local control and can be accessed via the host interface or the SCSIbus.
  - LUN #0 = the processor unit (CPU and DMAC)
  - LUN #1 to #4 - floppy disk drives
- 68000 programs can be loaded into the DPR and executed under local control.
- Sector deblocking mode (1024 bps-->512 bps=-256 bps).
- Logical Winchester partitioning by block address offset definition.

Floppy Disk Drives:

- Supports FORMAT and FORMAT TRACK commands.
- Disk parameters and format interleaves are all changeable.

1.3 Theory of Operation

1.3.1 Global Notes

After executing the start-up routine, the firmware main loop is entered. The power up procedure and self-test are described in sections 1.4 and 1.5 of this manual. The main routine polls the CMDRAMs for an executable command. When a valid command is found in one of the CMDRAMs, the command will be decoded, all parameters fetched, and the command executed.

On completing command execution, the status and command dependent parameters are returned in the CMDRAM. Optionally, an interrupt will be generated.

1.3.2 The Single Command Mode

The SINGLE COMMAND MODE works exactly as described above. For more information, please refer to chapters 2 and 3 of this manual, where a detailed description of all ISCSI-1 commands is given.

1.3.3 The Command Chaining Mode

The COMMAND CHAINING MODE is a powerful feature for building command macros and relieving the host from disk I/O jobs. It is entered from the single command mode with a special command and requests subsequent sequential CMDRAMs in the command chaining buffer. Each CMDRAM in the chain buffer is handled and executed as in the single command mode. For a detailed description of the command chaining mode, please refer to section 2.3.

1.3.4 The SCSIbus Target Mode

The SCSIbus TARGET MODE can be enabled by an initialization command. In this mode, the ISCSI-1 controller is selectable from the SCSIbus as a target (i.e. an SCSIbus slave). This mode is useful for system interconnection. Please refer to chapter 3 of this manual for a detailed description of the SCSIbus Target Mode.

1.3.5 Data Hashing and Blocking

NOTE: "Hashing" is the buffering of data read from a logical unit. "Blocking" is the buffering of data to be written to a logical unit.

For increased performance on ISCSI-1 read/write operations, data buffering and hashing are done as follows:

- Hashing is enabled by default for all SCSI devices on any READ operation. It can be disabled/enabled for each target seperately.

- For floppy disk drives, hashing is not supported.

- For all WRITE operations, blocking is disabled and can only be globally enabled for all units.

- On system shut down, the flush modified buffers command should be used to ensure that all data is stored to the disks!

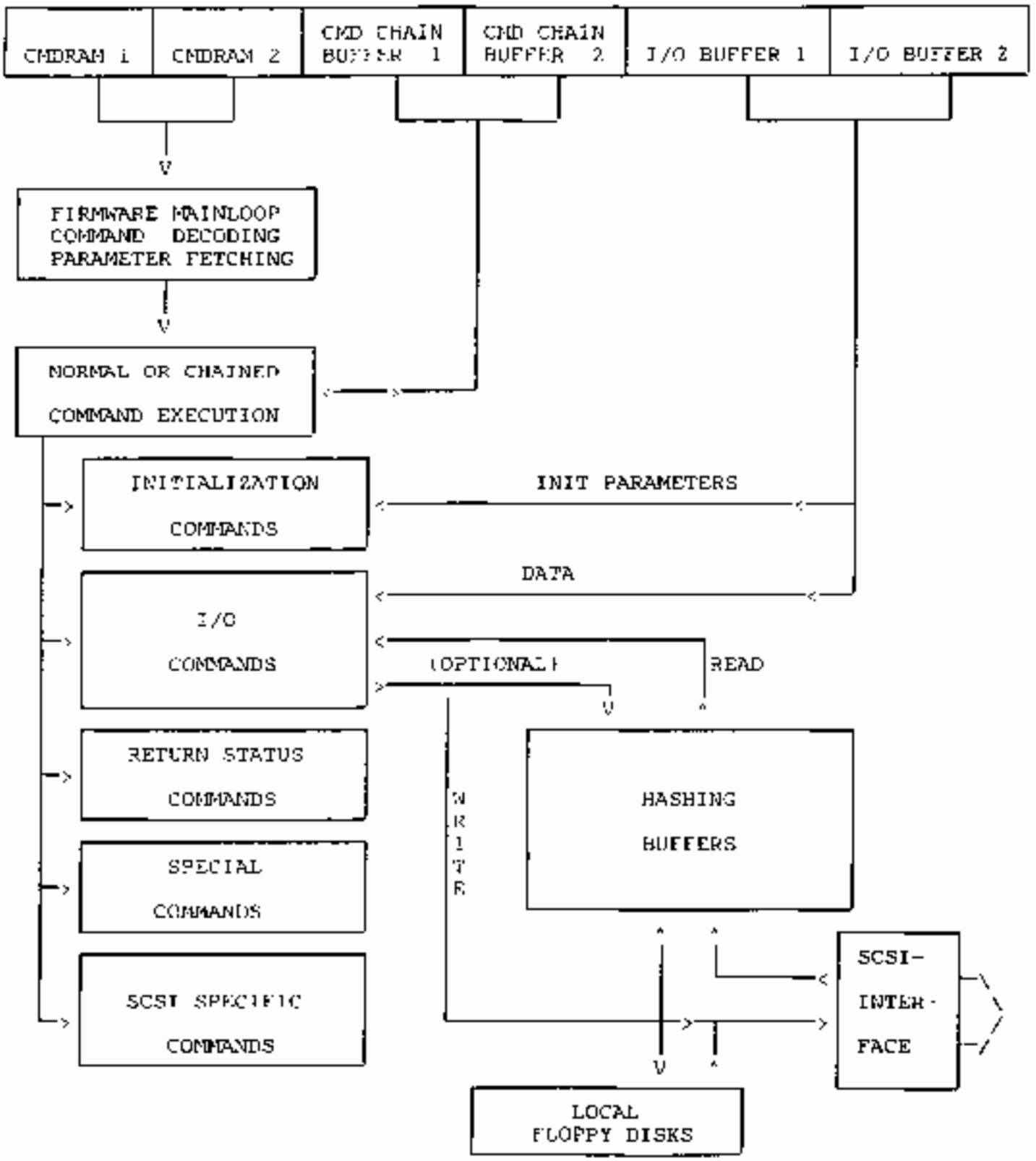
The hashing/blocking buffers are seven 16 Kbyte sized memory areas beginning at board address $4000. On read operations while hashing is enabled, the firmware searches first in a special list for a buffer which contains the requested block. If the buffer is found, the read is completed. If the buffer is nonexistent, the list is scanned for an unused buffer. When an unused buffer is found, it is marked as used and the firmware fills the buffer with blocks from the desired logical unit and completes the operation. If no unused buffer exists, the least recently called buffer is used for the read. If blocking is enabled and the buffer contains any changed blocks, they will first be flushed to the logical unit.

When blocking is enabled, a write to a logical unit will result in a read block operation. The procedure described above is executed and will return a pointer to the requested block in the buffers. The block(s) are then transferred from the I/O buffer to the hashing buffer and the firmware completes the operation.

NOTE: When hashing is enabled, only one block can be requested for any read operation. The hashing buffers must not necesarily be contiguous in memory; therefore, we cannot give <u>one</u> pointer to the data when the number of blocks will exceed the hashing buffer.

If blocking is enabled, changed buffers will be flushed to the disk(s) by default every 20 seconds. Therefore, if a system hangup occurs, the user should wait at least 20 seconds to ensure that all changed buffers are flushed.

Figure 1-1: Logical Block Diagram of the ISCSI-1 Firmware



1.4 The Power Up Procedure

After power up or reset, the firmware EPROM is down mapped to address location 0. The CPU fetches the initial stack pointer and start address and starts the firmware program.

First, the ISCSI-1 firmware performs a local self-test, which tests the local and dual ported memory, the DMA Controller, the SCSI controller, and the floppy disk controller. After the self-test, the system RAM is cleared and initialized, and the firmware sets all on-board interrupt vectors. The firmware enters the main routine and is ready to accept commands from the host.

1.5 The ISCSI-1 On-Board Self-test

On power up or reset, the ISCSI-1 self-test routine is executed in the following manner:

- The SCSI controller is tested by write and read back several registers.
- The entire local and dual ported memory is tested with read and write bytes, words, and long words.
- A DMA controller test is performed with memory to memory high speed data transfer.
- The floppy disk controller is tested.

Control of the self-test status and results of the self-test are provided via the front panel LEDs S1 through S4:

- The LED S1 is illuminated during the RAM test.
- The LED S2 is illuminated during the SCSIbus controller test.
- LED S3 is illuminated during the floppy disk controller test.
- LED S4 is illuminated during the DMA controller test.

NOTE: If any error is found while the self-test is active, the LED of the test phase which has generated the error will remain illuminated! When the self-test is successfully completed, all LEDs are turned off.

EXAMPLE:

After completion of the self-test routine, LED S3 remains illuminated. This state indicates a hardware problem with the WD1772 Floppy Disk Controller.

1.6 ISCSI-1 Default Setups

Table 1-1: ISCSI-1 Default Setups

Global Definitions:	ISCSI-1 SCSIbus ID	7
	Interfacing Sector Size	256 Bytes
	VMEbus Address Offset	0
	Data buffering for Write	disabled
SCSIbus Definitions:	Target Mode	disabled
	Sector Deblocking Mode	disabled
	Auto Backup Mode	disabled
	Hashing for Read	enabled for all targets
	Target interrupts	disabled
	Physical Block Address Offset	0
	Auto Flush Time Interval	20 seconds
Floppy Disk Drives:	Locked/reserved Units	none
	Sides per Disk	2
	Cylinders per Disk	80
	Sectors per Cylinder	32
	Bytes per Sector	256
	Disk Density	double/80 track drive
	Sector Offset (BIAS)	0
	Format Pattern	$E5
	Sector Interleave for Format	0
	Step rate	0 = 6ms

1.7 ISCSI-1 Memory Layout

Table 1-2 : Memory Layout from Local Sight

==========	==========	BOARD BASE	==========
$0000 - $0400	1 KB	VECTOR AREA	
$0400 - $2000	3 KB	FIRMWARE WORK AREA	
==========	==========	DUAL PORTED MEMORY	==========
$2000	1 W	HOLDS THE BOARD AND FIRMWARE REVISION CODE	
$20F0 - $20FA	5 W	TARGET MESSAGE INFO AREA	
$2100 - $210F	16 B	CMDRAM #1	
$2110 $22FF	496 B	CMD CHAIN AREA #1 = 31 CMDRAMS	
$2300 - $28FF	2 KB	I/O BUFFER #1	
$2B00 - $32FF	2 KB	I/O BUFFER #2	
$3300 - $34FF	512 B	CMD CHAIN AREA #2 = 31 CMDRAMS	
$3500 - $350F	16 B	CMDRAM #2	
$3510 - $3FFF	3 KB	RESERVED	
$4000 $1FFFF	112 KB	HASHING BUFFERS (7 BUFFERS, 16KB/BUFFER)	
==========	==========	END OF RAM	==========

Table 1-3 : Memory Layout from System/VMEbus Sight

========== ISCSI-1 BASE ADDRESS ==========			
$A00000 - $A00FFF			Reading a WORD from an EVEN address such as $A00000 will return RESET/SYSFAIL status information to bits 8, 9 and 10. Reading or writing a BYTE from an ODD address will access the ISCSI-1 BIM registers as described below.
$A00001	1	B	BIM control register 0
$A00003	1	B	BIM control register 1
$A00005	1	B	BIM control register 2
$A00007	1	B	BIM control register 3
$A00009	1	B	BIM vector register 0
$A0000B	1	B	BIM vector register 1
$A0000D	1	B	BIM vector register 2
$A0000F	1	B	BIM vector register 3
This structure is continued to address $A00FFF			
$A01001-$A017FF	1	B	Reading performs a local interrupt (odd addresses only)
$A01801-$A01FFF	1	B	Reading performs a local reset (odd addresses only)
======== ========== BEGIN OF DPR ==========			
$A02000	1	W	HOLDS THE BOARD AND THE FIRMWARE REVISION CODE
$A020F0 - $A020FA	5	W	TARGET MESSAGE INFO AREA
$A02100 - $A0210F	16	B	CMDRAM #1
$A02110 - $A022FF	496	B	CMD CHAIN AREA #1 - 31 CMDRAMS
$A02300 - $A02AFF	2	KB	I/O BUFFER #1
$A02B00 - $A032FF	2	KB	I/O BUFFER #2
$A03300 - $A034FF	512	B	CMD CHAIN AREA #2 = 31 CMDRAMS
$A03500 - $A0350F	16	B	CMDRAM #2
$A03510 - $A03FFF	3	KB	RESERVED
$A04000 - $A1FFFF	112	KB	HASHING BUFFERS (7 BUFFERS, 16KB/BUFFER)
========== END OF RAM AREA ==========			

This page was intentionally left blank

2. THE ISCSI-1 SOFTWARE INTERFACE

2.1 General Information

External software can access and use the ISCSI-1 firmware via DPR located RAM structures. These structures are known as Command RAM areas (CMDRAM), command chaining buffers, and general purpose I/O and program buffers. The CMDRAMs are described in chapter 2.2, command chaining buffers are described in chapter 2.3, and the general purpose I/O buffers in chapter 2.4.

For easy multiprocessor system support, all firmware interface structures exist twice. In addition, in each CMDRAM, a word location is reserved for interprocessor communication (access with the M68000 'TAS' instruction).

2.2 The Command RAM Layout

Each of the two CMDRAMs is defined as a 16 byte DPR-located structure. Parameters pass through the CMDRAM on input, status or error returns, and return parameters pass after command completion.

CMDRAM Layout for Input:

Offset	Size	Meaning
00	W	ISCSI-1 Command and Target ID
02	W	Command Dependent Parameter
04	L	Pointer to Data or Parameter block
08	L	Logical Block address
12	W	Logical Unit
14	W	Reserved for interprocessor communication

CMDRAM Layout on Command Completion:

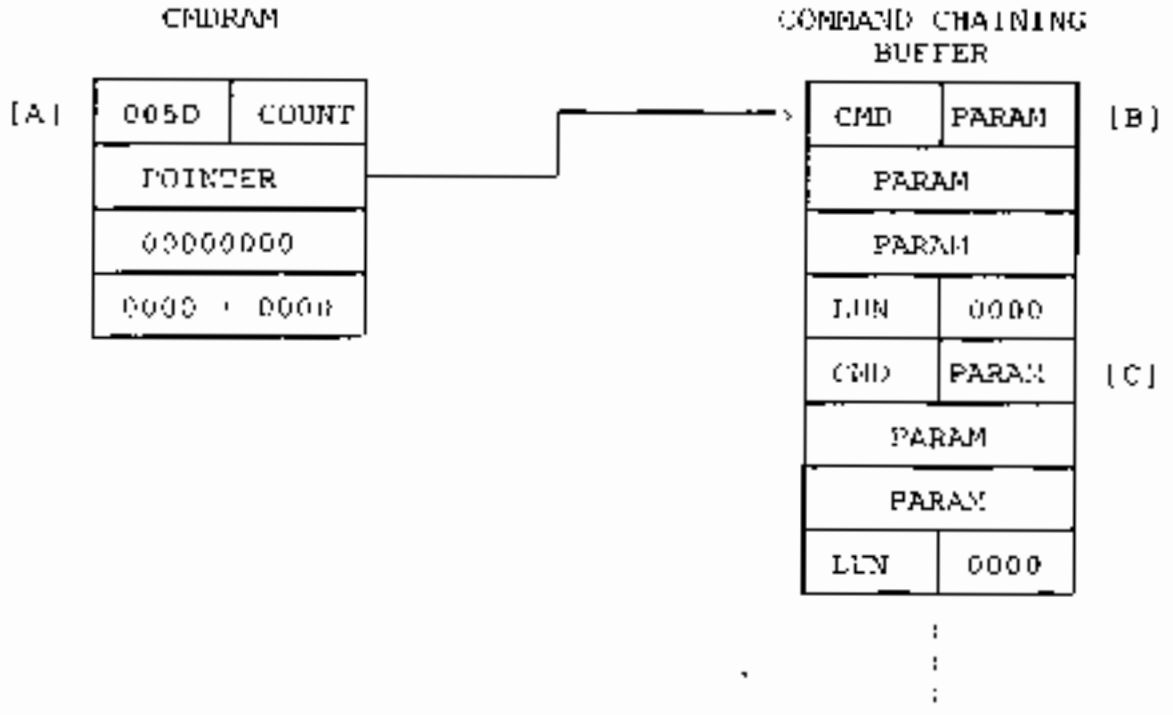
Offset	Size	Meaning
00	W	Status/Error Return Code
02	W	Extended Error Message or Data after Byte I/O
04	L	Pointer to Data or Return Values
08	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

2.3 Command Chaining

A multiple command execution option is built into the ISCSI-1 firmware. All CMDRAM blocks to be executed must be entered into a command queue. The command queue location is requested in the command chaining buffer of the corresponding channel. The queue can be longer than the size of the buffer. Each one of the commands in the queue is executed in the same way as in the single command mode. The only difference is the return value handling.

Error and completion codes are placed into the current command RAM in the queue. If an error occurs, an error code is placed in the root command RAM and a pointer to the error command RAM is returned. After execution of each command in the queue, and when the whole sequence has been completed, optional interrupts can be generated.

Figure 2-1: Command Chaining



[A] - CHAIN command

[B] - First CMDRAM to chain

[C] - Second CMDRAM to chain

2.4 Special RAM Locations

Board Type and Firmware Version Number:

At the board relative address $2000 (VMEbus $A02000), the board version and the firmware revision are stored in coded form.

first byte : ISCSI version - 11 = ISCSI-1
12 = ISCSI-2

second byte : firmware revision - 10 = REV 1.0
11 = REV 1.1
and so on.

Target Message Info Area:

This area is located at board relative address $20F0 (VMEbus $A020F0) and contains information about any data received from an initiator during the board is selected as an SCSI target. The message area has the following format:

Offset	Size	Meaning
0	W	contains zero if no message was received becomes negative after receiving data
2	W	contains the initiator ID
4	L	pointer to the received data
8	W	number of received 256 byte blocks

2.5 Floppy Disk Support

The ISCSI-1 firmware supports up to four floppy disk drives which are under local, not SCSI, control. The following points are important for proper use of the floppy disk drives with the firmware.

- During a floppy disk format, the defined sector interleave table (refer to the SUNPARM command) is used to number the sectors. The start sector of the interleave table is the lowest physical sector number on each track (i.e., 0 or 1).

- During a read or write operation, the lowest sector number is taken from the SUNPARM parameters.

- The physical sector layout of a floppy which was formatted with the ISCSI-1 firmware is as follows:

  side 1 : sector 0 or 1 ..... sector n
  side 2 : sector 0 or 1 ..... sector n

- Block hashing for read and data buffering for write are not supported for the floppy disks!

- The character and string I/O commands are not supported for floppy disks!

This page was intentionally left blank

3. ISCSI-1 COMMAND DESCRIPTION

3.1 The Command Structure and Philosophy

The ISCSI-1 commands are structured as general purpose command words which are compatible to other FORCE intelligent I/O controllers (i.e., ISIO-1). This supports easy design and installation of global I/O drivers.

Each command is a word sized code of the following format:

Bit#	Name	Explanation
15	Command/Status Flag	This flag is reset if the code is a command, and set if it is a status/error return code.
14-13		Reserved for future use
12	Interrupt enable	When the bit is set, the ISCSI-1 firmware generates an interrupt after operation complete. The vectors for normal and error interrupts can be set by programming the BIM.
11-9	Target ID	These three bits hold the SCSIbus Target ID (can be 0 - 7).
8		"don't care" (only for compatibility with the ISIO-1/2 command set).
7-4	Command Groups	These bits classify up to 16 groups of commands. Only seven of them are implemented: - Initialization commands [0] - Byte I/O commands [1] - Block I/O commands [2] - String I/O commands [3] - Status commands [4] - Special commands [5] - SCSI specific commands [6] - Reserved [7] to [15]
3-0	Commands per group	A maximum of 16 commands per group is defined.

3.2 Command Overview

Mnemonic	Hexcode	Function
SETOFFS	0003	Set VMEbus address offset.
	1003	Same function but complete with interrupt.
DEFOFFS	0008	Define physical block address offset.
	1008	Same function but complete with interrupt.
RESET	0009	Reset the firmware to power-on state.
SUNPARM	0X0A	Install logical unit 4 SCSI and local floppy disk).
	1X0A	Same function but complete with interrupt.
BUFFWR	000B	Enable/disable buffering for write operations.
	100B	Same function but complete with interrupt.
WRPARAL	000C	Enable/disable runtime backup to parallel logical unit.
	100C	Same function but complete with interrupt.
MAININIT	000D	Controller main initialization and installation command.
	100D	Same function but complete with interrupt.
SFTIME	000E	Set auto flush time interval.
	100E	Same function but complete with interrupt.
DEBLOCK	000F	Define sector deblocking mode.
	100F	Same function but complete with interrupt.
GETCHR	0X10	Read one byte from a logical block.
	1X10	Same function but complete with interrupt.
PUTCHR	0X14	Write one byte to a logical block.
	1X14	Same function but complete with interrupt.
GETBLOC	0X20	Read fix sized blocks/sectors from a logical unit.
	1X20	Same function but complete with interrupt.
PUTBLOC	0X22	Write fix sized blocks/sectors to a logical unit.
	1X22	Same function but complete with interrupt.
GETSTR	0X30	Read a delimited string from a logical block.
	1X30	Same function but complete with interrupt.

Mnemonic	Hexcode	Function
PUTSTR	0X32	Write a delimited string to a logical block.
	1X32	Same function but complete with interrupt.
GETCNT	0X33	Read a counted string from a logical block
	1X33	Same function but complete with interrupt.
PUTCNT	0X35	Write a counted string to a logical block.
	1X35	Same function but complete with interrupt.
CTLSTAT	0044	Return controller status information.
	1044	Same function but complete with interrupt.
CHKTARG	0045	Check SCSI targets for existence and device type.
	1045	Same function but complete with interrupt.
GUNPARM	0X46	Return logical units parameter.
	1X46	Same function but complete with interrupt.
EXPROG	0050	Execute user program in local DPR.
	1050	Same function but complete with interrupt.
BACKUP	0055	Backup a logical unit to another.
	1055	Same function but complete with interrupt.
FLUSH	0056	Flush all modified buffers.
	1056	Same function but complete with interrupt.
FORMAT	0X57	Format a logical unit.
	1X57	Same function but complete with interrupt.
FTRACK	0E58	Format a physical track on a floppy disk drive.
	1E58	Same function but complete with interrupt.
COMPARE	0059	Compare data between two logical units.
	1059	Same function but completes with interrupt.
LOCKUN	005A	Lock local unit against access from SCSIbus initiator.
	105A	Same function but complete with interrupt.
FREEUN	005B	Free local unit for SCSIbus access.
	105B	Same function but complete with interrupt.
COPY	005C	Copy data from one logical unit to another or between logical blocks on the same logical unit.
	105C	Same function but complete with interrupt.
CHAIN	005D	Enter command chaining mode.
	105D	Same function but complete with interrupt.

Mnemonic	Hexcode	Function
SEARCH	0X5E	Search for data on a logical unit.
	1X5E	Same function but complete with interrupt.
CLRHASH	005F	Clears the hashing buffer list entries.
	105F	Same function but complete with interrupt.
TRSPMOD	0060	Send a command in transparent mode to SCSIbus.
	1060	Same function but complete with interrupt.
SCSIRST	0061	Perform SCSIbus reset.
	1061	Same function but complete with interrupt.
TARGMOD	0062	Enable/disable target mode.
	1062	Same function but complete with interrupt.
TARGINT	0063	Enable/disable target message interrupts.
	1063	Same function but complete with interrupt.

3.3 Detailed Command Description

Mnemonic: **SETOFFS** - Set VMEbus Address Offset

Hex Code: $0003 Command completion without interrupt.
$1003 Command completion with interrupt.

DESCRIPTION:

This command sets an address offset which is always added to the ISCSI's local pointer operations. Usually, the pointers passed in the CMDRAM must be board relative. This means that the default address offset is zero.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Address Offset
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

EXAMPLE:

Set the VMEbus address offset to the system base address of the ISCSI-1 controller.

```
SETOFFS LEA.L   ISCSI+CMDRAM,A0 ;GET COMMAND RAM ADDRESS
        MOVE.L  #ISCSI,4(A0)    ;OFFSET = BASE ADDRESS
        MOVE.W  #SETADDR,(A0)   ;COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        XEXT                    ;EXIT TO SYSTEM
*
        END
```

Mnemonic: **DEFOFFS** - Define Physical Block Offset

Hex Code: $0008
$1008

DESCRIPTION:

This command sets an offset of physical data blocks local to the CMDRAM, which is used on SCSI read/write operations. It is useful for switching between several partitions on a hard disk, or for using different parts of the disk for the two CMDRAMs.

NOTE: The defined physical block offset is used for all SCSI devices accessed via the Firmware Read/Write commands, but not for floppies or devices accessed via the transparent mode command.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Physical block offset
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **RESET** - Reset the Firmware to the Power On State

Hex Code: $0009

DESCRIPTION:

This command restarts the firmware and reinitializes the I/O devices. During the RESET procedure, SYSFAIL is active and can be tested at address BASE+$0000;EVEN.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

NONE

EXAMPLE:

We restart the firmware:

```
RESET   LEA.L   ISCSI+CMDRAM,A0  ;GET CMDRAM ADDRESS
        LEA.L   ISCSI,A1          ;BASE ADDRESS
        MOVE.W  #RESTART,(A0)     ;COMMAND
@00:    MOVE.B  (A1),D0           ;WAIT UNTIL SYSFAIL IS RESETTED
        CMP.B   #$FF,D0
        BNE.S   @001
        XEXT                      ;EXIT TO SYSTEM
*
        END
```

Mnemonic: **SUNPARM** - Set Unit Parameters

Hex Code: $0X0A Command completion without interrupt.
$1X0A Command completion with interrupt.

DESCRIPTION:

This command enables the user to install different SCSIbus devices and to change the parameters of the local floppy disk drives.

Changeable options are:

- Hashing enable/disable for each logical SCSI unit
- Floppy disk parameters

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Unused/reserved
4	L	Pointer to Parameter Block
8	L	Unused/reserved
12	W	Logical Unit Number
14	W	Reserved for interprocessor communication

PARAMETER BLOCK

The parameter block must be located in the I/O buffer area. To install a floppy disk, the target ID must be the ID of the ISCSI-1 controller (default = 7).

1.) SCSI DEVICE INSTALLATION

Offset	Size	Meaning
00	L	Reserved
04	L	Reserved
08	B	Reserved
09	B	Reserved
10	B	Hashing mode 00 = disable hashing 01 = enable hashing

2.) LOCAL FLOPPY DISK INSTALLATION

Offset	Size	Meaning
00	B	Number of Heads
01	B	Number of Cylinders
02	B	Sectors per Cylinder
03	B	Sector offset to first used sector
04	B	Bytes per sector 0 = 128 bps 1 = 256 bps 2 = 512 bps 3 = 1024 bps
05	B	Disk density 00 = single density-80 track drive 01 - double density-80 track drive 02 - single density-40 track drive 03 - double density-40 track drive
06	B	Format Pattern
07	B	Sector interleave table 0-7 or new table 8
08	B	Lowest sector number on each physical track (0 or 1)
09	B	Step Rate Code: 0 = 6ms (default) 1 = 12ms 2 = 20ms 3 = 30ms Please refer to section 1-6 in this manual for the default setup.
10	B	Start of sector interleave table when new table is asserted. (The length depends on the number of sectors)

Predefined Sector Interleave Tables:

0 : 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
1 : 1,3,5,7,9,11,13,15,2,4,6,8,10,12,14,16
2 : 1,4,7,10,13,16,3,5,8,11,14,3,6,9,12,15
3 : 1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16
4 : 1,6,11,16,5,10,15,2,7,12,3,8,13,4,9,14
5 : 1,7,13,2,8,14,3,9,15,4,10,16,5,11,6,12
6 : 1,8,15,2,9,16,3,10,4,11,5,12,6,13,7,14
7 : 1,9,7,10,3,11,4,12,5,13,6,14,7,15,8,16

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
8002, 9002 = Address error

EXAMPLE:

```
LOCAL FLOPPY DISK INSTALLATION

SPARM1  LEA.L   SCSI+CMDRAM,A0   ;GET COMMAND RAM ADDRESS
        LEA.L   SCSI+IOBUF1,A1   ;GET I/O BUFFER #1
        LEA.L   FDCDEF(PC),A2    ;NEW FLOPPY DISK DEFINITIONS
        MOVE.L  #TABLEN,D0       ;GET TABLE LENGTH
@001    MOVE.B  (A2)+,(A1)+      ;TRANSFER DEFINITIONS
        SUBQ.L  #1,D0
        BNE.S   @001
        MOVE.L  #IOBUF1,4(A0)    ;POINTER TO DEFINITIONS
        MOVE.W  #1,12(A0)        ;FLOPPY DISK #1
        MOVE.W  #SUNPARM,(A0)    ;COMMAND, ID = 7 ($0E0A)
@002    TST.W   (A0)             ;WAIT UNTIL READY
        BPL.S   @002
        XEXT                     ;DONE
*
FDCDEF  DC.B    2                ;TWO HEADS
        DC.B    80               ;80 CYLINDERS
        DC.B    16               ;16 SECTORS PER CYLINDER
        DC.B    16               ;SECTOR OFFSET TO FIRST TRACK
        DC.B    2                ;512 BYTES PER SECTOR
        DC.B    1                ;DOUBLE DENSITY/80 TRACK DRIVE
        DC.B    $E5              ;FORMAT PATTERN
        DC.B    8                ;OWN INTERLEAVE
        DC.B    1                ;FIRST SECTOR NUMBER
        DC.B    3                ;STEP RATE = 3ms
        DC.B    1,4,7,2,5,8,3,6  ;SECTOR INTERLEAVE TABLE
TABLEN  EQU     18               ;TABLE LENGTH
        END
```

Mnemonic: **BUFFWR** - Enable/disable Buffering Mode

Hex Code: $000B Command completion without interrupt.
$100B Command completion with interrupt.

DESCRIPTION:

This command enables or disables (depending on the current state) sector buffering for write operations. When buffering is disabled (default), the command enables the buffering mode. If it is enabled, the command causes a 'flush modified buffers' operation and buffering is disabled for further write operations.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	BUFFWR Status: 1 = On 0 = Off
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error

Mnemonic: **WRPARAL** - Write to Parallel Disks

Hex Code: $000C Command completion without interrupt.
$100C Command completion with interrupt.

DESCRIPTION:

This command enables/disables (depending on the current state) the automatic run time backup of a logical unit. When enabled, each write operation goes to the specified logical unit and its parallel logical unit.

Restriction: The logical unit and the parallel logical unit must physically be the same devices, which means they must have the same number of blocks and an identical command set. Both drives should contain the same data when this command is executed. The parallel target should never be directly written to. This may confuse the whole data structure of the device!

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Target ID in lower byte, parallel Target ID in upper byte
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Extended Error Code
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **MAININIT** - Global ISCSI-1 Initialization

Hex Code: $000D Command completion without interrupt.
$100D Command completion with interrupt.

DESCRIPTION:

The MAININIT command is used to install and change global ISCSI-1 parameters and setups.

Installable items are:

- ISCSI-1's SCSIbus ID, (default = 7)
- The interfacing sector size between the ISCSI-1 and the host., (default = 256 bytes/sector)
- The standard string delimiter for string I/O functions (default = $00).

NOTE: The "interfacing sector size" is the global work block/sector size for all SCSIbus Read/Write operations.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Pointer to Parameter Block
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

PARAMETER_BLOCK

The parameter block must be located in the I/O buffer area.

Offset	Size	Meaning
0	B	ISCSI-1 own SCSIbus ID
1	B	Interfacing Sector size 1 = 256 bytes/sector 2 = 512 bytes/sector 4 =1024 bytes/sector 8 =2048 bytes/sector
2	B	Standard String Delimiter

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error
8002, 9002 - Address error

EXAMPLE:

Set the SCSI-1 ID to 6, the interfacing sector size to 256 bytes and the string delimiter to '$'.

```
MAINIT          LEA.L   SCSI+CMDRAM,A0   ;GET COMMAND RAM ADDRESS
                LEA.L   SCSI+IOBUF1,A1   ;GET I/O BUFFER #1
                MOVE.W  #$0601,(A1)      ;DEFINITIONS
                MOVE.B  #'$',2(A1)
                MOVE.L  #IOBUF1,4(A0)    ;POINTER TO DEFINITIONS
                MOVE.W  #MINIT,(A0)      ;COMMAND
@002            TST.W   (A0)             ;WAIT UNTIL READY
                BPL.S   @002
                XEXT                     ;DONE
*
*
                END
```

Mnemonic: **SFTIME** - Set Flush Buffers Time Interval

Hex Code: $000E Command completion without interrupt.
$100E Command completion with interrupt.

DESCRIPTION:

In the buffered write mode the iSCSI-1 firmware flushes out the modified buffers after a specified timeout. By default, this timeout is 20 seconds. The SFTIME command is used to change this interval, or to disable the function. A time interval of '0' is expanded to $FFFFFFFF.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	time out interval (a value of 500 is one second)
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

EXAMPLE:

Set the auto flush timeout to 10 seconds.

```
SETIME  LEA.L   15C51+CMDRAM,A0         ;GET COMMAND RAM
        MOVE.L  #5000,4(A0)             ;10 SECONDS TIMEOUT
        MOVE.W  #STIME,(A0)             ;COMMAND CODE
@001    TST.W   (A0)                    ;WAIT FOR COMPLETION
        BPL.S   @001
        XEXT
        END
```

Synonym: DEBLOCK - Enable/disable Sector Deblocking

Hex Code: $000F Command completion without interrupt.
$100F Command completion with interrupt.

DESCRIPTION:

This command enables or disables sector deblocking. This means operating with different sector sizes between the host and the ISCSI-1, and the SCSIbus and the ISCSI-1; but only if the sector size between the host and the ISCSI-1 is smaller that the SCSI sector size. The factor for this difference can be passed in the CMDRAM. It is a positive number from which the real deblocking factor is calculated internal as (n^2).

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	0=disable/1=enable
4	L	Deblocking factor (0,1,2...)
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **GETCHR** - Read One Byte from a Logical Block

Hex Code: $0X10 Command completion without interrupt.
$1X10 Command completion with interrupt.

DESCRIPTION:

The GETCHR command returns a data byte from a logical block to the CMDRAM. The byte is addressed by the logical block address and an offset from the block base to the byte.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Unused/reserved
4	L	Logical Block Address
8	L	Offset from Block base to the Byte
12	W	Logical Unit
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Data byte in the lower byte or extended error message
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800F, 900F = Extended error message, (refer to Appendix A for detailed information)

EXAMPLE:

Read the character at position 122 from logical block 2635 on logical unit 1, which is connected to target controller 4. The programming example is designed as a subroutine.

```
                .
                .
                .
        MOVE.W  #4,D0           ;TARGET ID
        MOVE.W  #1,D1           ;LUN#
        MOVE.L  #2635,D2        ;BLOCK ADDRESS
        MOVE.L  #122,D3         ;CHARACTER POSITION
        BSR.L   GETCHR          ;CALL THE ROUTINE
                .
                .
                .
*
**************************************************************
*
*       IN :    D0.W = TARGET ID
*               D1.W = LOGICAL UNIT
*               D2.L = LOGICAL BLOCK ADDRESS
*               D3.L = CHARACTER OFFSET
*
*       OUT:    D0.B = CHARACTER
*
*
GETCHR  MOVE.L  A0,-(A7)        ;SAVE A0
        LEA.L   ISCSI+CMDRAM,A0 ;GET COMMAND RAM POINTER
        MOVE.L  D2,4(A0)        ;LOGICAL BLOCK
        MOVE.W  D1,12(A0)       ;LOGICAL UNIT
        MOVE.L  D3,8(A0)        ;CHARACTER OFFSET
        LSL.W   #8,D0           ;ADJUST ID
        ORI.W   #GCHAR,D0       ;GETCHR COMMAND
        MOVE.W  D0,(A0)
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        MOVE.W  2(A0),D0        ;GET THE CHARACTER
        MOVE.L  (A7)+,A0        ;RESTORE REGISTER
        RTS                     ;RETURN TO CALLER
*
*
        END
```

Mnemonic: PUTCHR - Write One Byte to a Logical Block

Hex Code: S0X14 Command completion without interrupt.
S1X14 Command completion with interrupt.

DESCRIPTION:

This command writes a data byte to a logical block. The location at which the byte is written is addressed with the logical block address and an offset to the write position in the logical block.

NOTE: This command works only when the blocking mode is enabled.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Data Byte(s)
4	L	Logical Block Address
8	L	Offset from Block Base to write Position
12	W	Logical Unit
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Extended Error Message
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800F, 900F = Extended error message, (Refer to Appendix A for detailed information)

EXAMPLE:

Write a byte to position 100 from logical block 2000 on logical unit 1, which is connected to target controller 4. The programming example is designed as a subroutine.

```
                .
                .
                .
        MOVE.W  #4,D0           ;TARGET ID
        MOVE.W  #1,D1           ;LUN#
        MOVE.L  #2000,D2        ;BLOCK ADDRESS
        MOVE.L  #100,D3         ;CHARACTER POSITION
        MOVE.W  #$00FF,D4       ;DATA
        BSR.L   PUTCHR          ;CALL THE ROUTINE
                .
                .
                .
*
*****************************************************************
*
*       IN :    D0.W = TARGET ID
*               D1.W = LOGICAL UNIT
*               D2.L = LOGICAL BLOCK ADDRESS
*               D3.L = CHARACTER OFFSET
*               D4.W = CHARACTER(S)
*
*       OUT:    D0.L = RETURN CODE
*
PUTCHR  MOVE.L  A0,-(A7)        ;SAVE A0
        LEA.L   ISCSI+CMDRAM,A0 ;GET COMMAND RAM POINTER
        MOVE.L  D2,4(A0)        ;LOGICAL BLOCK
        MOVE.W  D1,12(A0)       ;LOGICAL UNIT
        MOVE.L  D3,8(A0)        ;CHARACTER OFFSET
        MOVE.W  D4,2(A0)        ;DATA BYTE(S)
        LSL.W   #8,D0           ;ADJUST ID
        ORI.W   #PCHAR,D0       ;PUTCHR COMMAND
        MOVE.W  D0,(A0)
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        MOVE.L  (A0),D0         ;GET RETURN STATUS
        MOVE.L  (A7)+,A0        ;RESTORE REGISTER
        RTS                     ;RETURN TO CALLER
*                                       .
*
        END
```

Mnemonic: **GETBLOC** - Read Logical Blocks from a Unit

Hex Code: $0X20 Command completion without interrupt.
$1X20 Command completion with interrupt.

DESCRIPTION:

This function reads a number of blocks from the specified logical unit and returns a pointer to the first byte of the data. The blocks which should be read are requested as contiguous on the medium.

NOTE: When hashing is enabled (default), only one block can be requested for any read operation. The hashing buffers must not necesarily be contiguous in memory; therefore, we cannot give one pointer to the data when the number of blocks will exceed the hashing buffer.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Number of Blocks to read (max - 256)
4	L	Block address
8	L	[Buffer pointer] (only necessary for floppy access or other unhashed operations.)
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[Extended Error Code]
4	L	Pointer to first byte of data
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F = Extended error message (Refer to Appendix A for detailed information)

EXAMPLE:

The program reads a logical block from the target with the ID = 0, logical unit number 0. The block address is $100.

```
RDBLOCK MOVEA.L #ISCSI+CMDRAM,A0        ;COMMAND RAM POINTER
        MOVE.W  #1,2(A0)                ;NUMBER OF BLOCKS
        MOVE.L  #$100,4(A0)             ;BLOCK ADDRESS
        MOVE.W  #0,12(A0)               ;LOGICAL UNIT NUMBER
        MOVE.W  #RBLOCK,(A0)            ;COMMAND/TARGET ID = 0
@001    TST.W   (A0)                    ;READY?
        BPL.S   @001                    ;N, WAIT
        TST.B   1(A0)                   ;Y, ERROR?
        BEQ.S   @002                    ;N
        BRA.S   ERROR                   ;Y
@002    MOVEA.L 4(A0),A0                ;GET DATA POINTER
        ADDA.L  #ISCSI,A0               ;ADD BASE ADDRESS
        RTS
*
ERROR   .
        .
        .
*
        END
```

Mnemonic: **PUTBLOC** - Write Logical Blocks to a Unit

Hex Code: 80X22 Command completion without interrupt.
81X22 Command completion with interrupt.

DESCRIPTION:

This function writes a number of contiguous blocks to a logical unit. The blocks must reside in the ISCSI-1 on-board Dual Ported RAM (in the I/O buffers). When an error occurs during the write, the number of remaining blocks to write is reported in the command RAM. The data pointer points to the first remaining block.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Number of blocks to write (max - 256)
4	L	Block address of first block
8	L	Pointer to first byte of data
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Number of blocks which were written
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F - Extended error message (Refer to Appendix A for detailed information)

EXAMPLE:

The program writes out four logical blocks, starting with block address $100 to target 0, logical unit 0.

```
WRBLOCK MOVEA.L #ISCSI+CMDRAM,A0          ;COMMAND RAM POINTER
        MOVE.W  =4,2(A0)                  ;NUMBER OF BLOCKS
        MOVE.L  =$100,4(A0)               ;BLOCK ADDRESS
        MOVE.L  #IO_BUF1,8(A0)            ;DATA POINTER
        MOVE.W  #0,12(A0)                 ;LOGICAL UNIT NUMBER
*
        LEA.L   BUFFER(PC),A1             ;GET DATA
        MOVEA.L #ISCSI+IO_BUF1,A2         ;GET DPR ADDRESS
        MOVE.L  #512,D0                   ;BLOCK SIZE
@001    MOVE.L  (A1)+,(A2)+               ;TRANSFER DATA
        SUBQ.L  #1,D0
        BNE.S   @001
*
        MOVE.W  #WBLOCK,(A0)              ;COMMAND/ID = 0
@002    TST.W   (A0)                      ;READY?
        BPL.S   @002                      ;N, WAIT
        TST.B   1(A0)                     ;Y, ERROR?
        BEQ.S   @003                      ;N
        BRA.S   ERROR                     ;Y
@003    RTS
*
ERROR   .
        .
        .
*
        END
```

Mnemonic: **GETSTR** - Read a Delimited String from a Logical Block

Hex Code: $0X30 Command completion without interrupt.
$1X30 Command completion with interrupt.

DESCRIPTION:

The GETSTR command copies a delimited string from a unit's logical block to the I/O buffer. If the string delimiter is not found, a maximum string of the size of one block is returned. At offset 2 in the CMDRAM the string delimiter can be specified. If this parameter is set to zero, the standard delimiter (installed with MAININIT, default is $00) is used. The parameter at offset 4 in CMDRAM holds the offset from the logical block base to the first byte of the requested string.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Optional String Delimiter
4	L	Logical Block Address
8	L	First Byte Location in Block
12	W	Logical Unit Number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Extended Error Message
4	L	Pointer to String
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F = Extended error message (Refer to Appendix A for detailed information)

EXAMPLE:

Read a delimited string from an SCSI Winchester drive:

```
GETSTR  LEA.L   ISCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W  #$0,2(A0)       ;STRING DELIMITER IS NULL
        MOVE.L  #$20,4(A0)      ;OFFSET TO START BYTE IN BLOCK
        MOVE.L  #$1000,8(A0)    ;BLOCK ADDRESS
        MOVE.W  #$0,12(A0)      ;LUN = 0
        MOVE.W  #GSTRING,(A0)   ;COMMAND AND TARGET ID = 0
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE.L   ERRHDL          ;YES, INDIVIDUAL ERROR HANDLER
        MOVEA.L 4(A0),A0        ;NO, GET STRING POINTER
                :
                :
                :
```

Mnemonic: **PUTSTR** Write a Delimited String to a Logical Block

Hex Code: $0X32 Command completion without interrupt.
$1X32 Command completion with interrupt.

DESCRIPTION:

The PUTSTR command copies a delimited string from the I/O buffer to a specified block of a logical unit. When the string delimiter is not found, a maximum string size of one block is copied. At offset 2 in the CMDRAM, the string deliniter can be specified. If this parameter is set to 0, the standard delimiter (installed with MAININIT, default is $00) is used. The parameter at offset 8 in CMDRAM holds the offset from the logical block base to the first byte of the destination area. The string source location is requested at the lowest byte of one of the I/O buffers.

NOTE: This command works only when the blocking mode is enabled.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Optional string delimiter
4	L	Destination block address
8	L	Offset in destination block to the first string location
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error

EXAMPLE:

Assume that blocking mode is enabled!!!

```
PUTSTR  LEA.L   ISCSI+CMDRAM,A0 ;GT CMDRAM POINTER
        LEA.L   ISCSI+IO_BUF,A1 ;GET I/O BUFFER POINTER
        LEA.L   STRING(PC),A2   ;GET STRING DEFINITION
        MOVEQ.L #0,D0           ;CLEAR THIS
        MOVE.B  (A2)+,D0        ;GET STRING LENGTH
@001    MOVE.B  (A2)+,(A1)+     ;TRANSFER STRING TO I/O BUFFER
        SUBQ.L  #1,D0           ;DECREMENT COUNTER
        BNE.S   @001            ;LOOP UNTIL DONE
        MOVE.W  #$0,2(A0)       ;STRING DELIMITER IS NULL
        MOVE.L  #$20,8(A0)      ;FIRST STRING LOCATION IN BLOCK
        MOVE.L  #$1000,4(A0)    ;BLOCK ADDRESS
        MOVE.W  #$0,12(A0)      ;LUN = 0
        MOVE.W  #PSTRING,(A0)   ;COMMAND AND TARGET ID
@002    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @002
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, INDIVIDUAL ERROR HANDLING
                :
                :
                :
```

Mnemonic: **GETCNT - Read a Counted String from a Logical Block**

Hex Code: $0X33 Command completion without interrupt.
$1X33 Command completion with interrupt.

DESCRIPTION:

The GETCNT command copies a counted string from a logical block of a unit to the I/O buffer. The parameter at offset 8 in CMDRAM holds the offset from the logical block base to the first byte of the requested string.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	String length
4	L	Logical Block Address
8	L	First Byte Location in Block
12	W	Logical Unit Number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

0	W	Status/Error return code
2	W	Extended Error Message
4	L	Pointer to String
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

9000, 9000 = Successful, no error
900B, 900B = Address error

EXAMPLE:

```
GETCNT  LEA.L   $A02100,A0         ;GET CMDRAM POINTER
        MOVE.W  #$40,2(A0)         ;GET 64 BYTES
        MOVE.L  #$1000,4(A0)       ;FROM BLOCK $1000
        MOVE.L  #$0,8(A0)          ;OFFSET 0
        MOVE.W  #$0,12(A0)         ;LUN = 0
        MOVE.W  #$0233,14(A0)      ;COMMAND AND TARGET ID
@001    TST.W   (A0)               ;WAIT UNTIL READY
        BEL.S   @001
        TST.B   1(A0)              ;ANY ERRORS?
        BNE.L   ERRHDL             ;YES, ERROR HANDLER!
        MOVEA.L 4(A0),A0           ;NO, GET STRING POINTER
                :
                :
                :
```

Mnemonic: **PUTCNT** - Write a Counted String to a Logical Block

Hex Code: $0X35 Command completion without interrupt.
$1X35 Command completion with interrupt.

DESCRIPTION:

The PUTCNT command writes a counted string to a logical block. The first byte location of the source string is requested at the beginning of I/O Buffer 1 (or I/O Buffer 2 for the second command RAM).

NOTE: This command works only when the blocking mode is enabled.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	String length
4	L	Block Address
8	L	First Byte Location in the Block
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error

EXAMPLE:

Assume that blocking mode is enabled:

```
PUTCNT  LEA.L   ISCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        LEA.L   ISCSI+IO_BUF,A1 ;GET I/O BUFFER POINTER
        LEA.L   STRDEF(PC),A2   ;GET STRING DEFINITION
        MOVE.L  #$40,D0         ;STRING LENGTH
@001    MOVE.B  (A2)+,(A1)+     ;TRANSFER STRING
        SUBQ.L  #1,D0           ;DECREMENT COUNTER
        BNE.S   @001            ;LOOP UNTIL DONE
        MOVE.W  #$40,2(A0)      ;STORE STRING LENGTH
        MOVE.L  #$0,8(A0)       ;OFFSET IN BLOCK
        MOVE.L  #$1000,4(A0)    ;BLOCK ADDRESS
        MOVE.W  #$0,12(A0)      ;LUN - 0
        MOVE.W  #PCOUNT,(A0)    ;COMMAND AND TARGET ID
@002    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @002
        TST.W   2(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER!
                ;
                ;
                ;
```

Mnemonic: **CTLSTAT** - Return General Controller Status

Hex Code: $0044 Command completion without interrupt.
$1044 Command completion with interrupt.

DESCRIPTION:

This command returns the current status of the ISCSI 1 controller as a block of parameters.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Pointer to the parameter block
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

RETURN PARAMETER BLOCK

Offset	Size	Meaning
00	B	ISCSI-1 own ID
01	B	Interfacing sector size 01 - 256 bytes 02 - 512 bytes 04 - 1024 bytes 08 = 2048 bytes
02	B	Standard string delimiter
03	B	SCSI status 00 = disconnected 01 = connected as initiator 02 - selected as target
04	B	Target mode 00 = disabled 01 = enabled

05	B	Auto backup mode 00 = disabled 01 = enabled
06	B	Auto reserve/release unit 00 = disabled 01 = enabled
07	B	Data buffering for write 00 = disabled 01 = enabled
08	B	Locked units bits 0-4 - if 0 then unlock if 1 then locked
09	B	Fill byte
10	L	Defined address offset
14	L	Defined block offset

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

EXAMPLE:

Get current ISCSI-1 state:

```
CTLSTAT LEA.L   ISCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W  #CSTATUS,(A0)   ;COMMAND CODE
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
        MOVEA.L 4(A0),A0        ;NO, GET POINTER TO INFO
                :
                :
                :
```

Mnemonic: **CHKTARG** - Check for and Test Existing Targets

Hex Code: $0045 Command completion without interrupt.
$1045 Command completion with interrupt.

DESCRIPTION:

This command scans the SCSIbus for all possible IDs and returns information about the existence and the type of the corresponding devices.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error Return Code
2	W	Unused/reserved
4	L	Pointer to return parameters
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

RETURN PARAMETER BLOCK

- The command always returns an eight byte block which contains the ID status information.
- The block starts with the information at ID 0 and ends with ID 7.
- The status information byte can hold one of the following values:

$00 - direct access device
$01 - sequential access device
$02 - printer device
$03 - processor device
$04 - write once/read multiple device
$05 - read only/direct access device
$7F - device does not exist
$80 - unknown device type
$FF - ISCSI-1 own ID

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error

EXAMPLE:

```
CHECK   XPCL                            ;CR/LF
        XPCL                            ;CR/LF
        XPMC    MSTART                  ;START MESSAGE
        MOVEA.L #ISCSI+CMDRAM,A0        ;CMDRAM
@001    TST.W   (A0)                    ;BUSY?
        BPL.S   @001                    ;Y, WAIT
        MOVE.W  #CHKTARG,(A0)           ;COMMAND
@002    TST.W   (A0)                    ;BUSY?
        BPL.S   @002                    ;Y, WAIT
        MOVEA.L 4(A0),A3                ;GET POINTER
        ADDA.L  #ISCSI,A3               ;ADD BOARD BASE
        MOVE.L  #0,D1                   ;COUNTER
@100    MOVE.B  (A3)+,D2                ;GET DATA
        XPCL                            ;CR/LF
        XBIN                            ;BIN TO DEC ASCII
        XPRL                            ;WRITE ID
        CMP.B   #$FF,D2                 ;OWN?
        BNE.S   @101                    ;N
        XPMC    MOWNID                  ;Y
        BRA.S   @300
@101    CMP.B   #$80,D2                 ;UNKNOWN DEVICE?
        BNE.S   @102                    ;N
        XPMC    MUNKN                   ;Y
        BRA.S   @300
@102    CMP.B   #$7F,D2                 ;NON EXISTING?
        BNE.S   @103                    ;N
        XPMC    MNEXIS                  ;Y
        BRA.S   @300
@103    CMP.B   #0,D2                   ;MAGNETIC DISK?
        BNE.S   @104                    ;N
        XPMC    MMAGN                   ;Y
        BRA.S   @300
@104    XPMC    MTAPE                   ;SHOULD BE STREAMER TAPE
@300    ADDQ.L  #1,D1                   ;NEXT ID
        CMP.B   #8,D1                   ;DONE?
        BLT.S   @100                    ;N, CONTINUE
        XPCL                            ;CR/LF
        XEXT                            ;EXIT TO PDOS
*
MOWNID  DC.B    '      ISCSI-1',0
MUNKN   DC.B    '      UNKNOWN DEVICE TYPE',0
MMAGN   DC.B    '      MAGNETIC DISK',0
MNEXIS  DC.B    '      NON EXISTING',0
MTAPE   DC.B    '      STREAMER TAPE',0
MSTART  DC.B    'ID    DEVICE',10,13
        DC.B    '-------------------------    ',0
        EVEN
        END
```

Mnemonic: **GUNPARM** - Get Unit Parameters

Hex Code: $0X46 Command completion without interrupt.
$1X46 Command completion with interrupt.

DESCRIPTION:

This command is the opposite of the SUNPARM (set units parameters) command and returns the installed values for the specified logical unit.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID If the ID is the ISCSI-1 ID, the values of the floppy disk units are returned.
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Pointer to return parameters
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

RETURN PARAMETER BLOCK

The parameter block format is identical to the format described at the SUNPARM command. Refer to that description for more information.

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

EXAMPLE:

Get the installation parameters of the floppy disk #1:

```
GETPARM LEA.L    $SCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W   #$1,12(A0)      ;LUN = 1
        MOVE.W   #GPARAM,(A0)    ;COMMAND, ID = 7
@001    TST.W    (A0)            ;WAIT UNTIL READY
        BPL.S    @001
        TST.B    1(A0)           ;ANY ERROR??
        BNE.L    ERRHDL          ;YES, ERROR HANDLER!
        MOVEA.L  4(A0),A0        ;NO, GET POINTER
                 :
                 :
                 :
```

Mnemonic: **EXPROG** - Execute User Program in DPR

Hex Code: $0050 Command completion without interrupt.
$1050 Command completion with interrupt.

DESCRIPTION:

This command executes a program which must be loaded into the ISCSI-1 DPR area. The program should be located only in the I/O buffers and must complete with a RTS instruction.

NOTE: The program must not access any nonlocal addresses because the ISCSI 1 CPU cannot become a VMEbus master!

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Program start address
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error

EXAMPLE:

Execute a program under local firmware control:

- First load the program with the PDOS LO command.
- Then start the execution.

```
>LO PROG,SA02300
```

The program PROG is now loaded to the I/O buffer 1.

```
EXPROG  LEA.L   ISCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.L  #IO_BUF,4(A0)   ;START ADDRESS OF PROG
        MOVE.W  #XPROG,(A0)     ;COMMAND CODE
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER

                ;
                ;
                ;
```

NOTE: The last executed instruction must be an RTS.

Mnemonic: **BACKUP** - Backup a Logical Unit

Hex Code: $0055 Command completion without interrupt.
$2055 Command completion with interrupt.

DESCRIPTION:

This command processes a backup from one logical unit to another. Before the backup starts, a flush modified buffers is done if the ISCSI-1 is in the blocking mode. The hashing buffers are cleared because they will be used as data storage areas during the command execution.

NOTE: The BACKUP command is only allowed for Winchester SCSI units!

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Source unit low word = logical unit number high word = target ID
8	L	Destination unit low word = logical unit number high word = target ID
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F = Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

Backup the target 0 to target 1:

```
BACKUP  LEA.L   ISCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W  #$0,2(A0)       ;CLEAR THIS
        MOVE.W  #$0,4(A0)       ;SOURCE TARGET ID
        MOVE.W  #$0,6(A0)       ;SOURCE TARGET LUN
        MOVE.W  #$1,8(A0)       ;DESTINATION TARGET ID
        MOVE.W  #$0,10(A0)      ;DESTINATION TARGET LUN
        MOVE.W  #$0,12(A0)      ;CLEAR THIS
        MOVE.W  #DOBACK,(A0)    ;COMMAND CODE
@001    TST.W   (A0)            ;WAIT UNTIL READY
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
                :
                :
                :
```

Mnemonic: **FLUSH** - Flush Modified Buffers

Hex Code: $0056 Command completion without interrupt.
$1056 Command completion with interrupt.

DESCRIPTION:

This command only affects the buffered write mode and is used to save and update all modified buffers to the devices.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error
800F, 900F - Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

```
FLUSH   LEAL    TSCSI+CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W  #DOFLUSH,(A0)   ;COMMAND CODE
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERRORS?
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
                :
                :
                :
```

Mnemonic: **FORMAT** - Format a Logical Unit

Hex Code: $0X57 Command completion without interrupt.
$1X57 Command completion with interrupt.

DESCRIPTION:

This command does a physical format of a logical unit. The installed parameters (SUNPARM) are used to format the unit. For SCSIbus units, please refer to the example included in Appendix E.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Logical unit number
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800F, 900F = Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

Format the hard disk target ID 1 and the floppy disk LUN 2

```
FORMAT  LEA.L   2SCSI-CMDRAM,A0 ;GET CMDRAM POINTER
        MOVE.W  #$0,12(A0)      ;TARGET LUN = 0
        MOVE.W  #HDFRMT,(A0)    ;COMMAND AND ID = 1 ($0257)
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
        MOVE.W  #2,12(A0)       ;NO, SET DISK LUN
        MOVE.W  #FDFRMT,(A0)    ;COMMAND AND ID = 7 ($0E57)
@002    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @002
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
                :
                :
                :
```

Mnemonic: FTRACK - Format a Track on a Floppy Disk Drive

Hex Code: $0X58 Command completion without interrupt.
$1X58 Command completion with interrupt.

DESCRIPTION:

This command formats a single track on a floppy disk with the parameters given in the input parameter block. The target ID must be set to the ISCSI-1 ID.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Track number
4	L	Pointer to parameter block
8	L	Unused/reserved
12	W	Floppy disk number
14	W	Reserved for interprocessor communication

PARAMETER BLOCK

Offset	Size	Meaning
00	B	Number of Heads
01	B	Number of Cylinders
02	B	Sectors per Track
03	B	Dummy byte
04	B	Bytes per sector 0 = 128 bps 1 = 256 bps 2 = 512 bps 3 = 1024 bps
05	B	Disk density 00 = single density, 80 track drive 01 = double density, 80 track drive 02 = single density, 40 track drive 03 = double density, 40 track drive
06	B	Format Pattern
07	B	Sector interleave table 0-7 or new table 8

PARAMETER BLOCK cont.

Offset	Size	Meaning
08	B	Lowest sector number on each physical track (0 or 1)
09	B	Step rate
10	B	Start of sector interleave table when new table is asserted. (The length depends on the number of sectors)

OUT: CHDRAM

Offset	Size	Meaning
0	W	Status/Error return code .
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F = Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

Format floppy disk 1, track 0, both sides:

```
FTRACK  LEA.L   ISCSI-CMDRAM,A0 ;GET CMDRAM POINTER
        LEA.L   ISCSI+IO_BUF,A1 ;GET I/O BUFFER POINTER
        LEA.L   PARAM(PC),A2    ;GET PARAMETER BLOCK
        MOVE.L  #10,D0          ;PARAMETER LENGTH
@001    MOVE.B  (A2)+,(A1)+     ;TRANSFER PARAMETERS
        SUBQ.L  #1,D0           ;DECREMENT COUNTER
        BNE.S   @001            ;LOOP UNTIL DONE
        MOVE.W  #0,2(A0)        ;FORMAT TRACK 0
        MOVE.L  #IO_BUF,4(A0)   ;POINTER TO PARAM
        MOVE.W  #1,12(A0)       ;DISK - 1
        MOVE.W  #TRFRMT,(A0)    ;COMMAND CODE ($0E58)
@002    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @002
        TST.B   1(A0)           ;ANY ERROR??
        BNE.L   ERRHDL          ;YES, ERROR HANDLER
                :
                :
                ;

PARAM   DC.B    $02,$50,$10,$00,$01,$01,$E5,$00,$01,$00
```

NUMBER OF HEADS
NUMBER OF CYLINDERS
SECTORS PER TRACK
DUMMY BYTE
BYTES PER SECTOR (256)
DISK DENSITY CODE (DOUBLE)
FORMAT PATTERN
SECTOR INTERLEAVE
LOWEST SECTOR NUMBER ON TRACK
STEP RATE

Mnemonic: **COMPARE** Compare Data Between Two Logical Units

Hex Code: 50059 Command completion without interrupt.
51059 Command completion with interrupt.

DESCRIPTION:

This command compares data between two logical units or between different logical blocks on the same logical unit.

NOTE: The COMPARE command is only allowed for SCSI Winchester units!

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Pointer to parameter block
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

PARAMETER BLOCK

Offset	Size	Meaning
00	B	First unit - ID
01	B	First unit - logical unit number
02	L	First unit - start block address
06	B	Second unit - ID
07	B	Second unit - logical unit number
08	L	Second unit - start block address
12	W	Number of blocks to compare

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
800F, 900F = Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

```
ISCSI   EQU     $A00000         ;ISCSI 1 BASE ADDRESS
CR_1    EQU     $2100           ;COMMAND RAM 1
BUF_1   EQU     $2400           ;I/O BUFFER 1
BUF_2   EQU     $2B00           ;I/O BUFFER 2
*
        OPT     ALT
*
* SETUP ISCSI-1 BASE ADDRESS
SETUP   LEA.L   ISCSI+CR_1,A0   ;GET COMMAND RAM
        MOVE.L  #ISCSI,4(A0)    ;SET BASE ADDRESS
        MOVE.W  #$0003,(A0)     ;SETOFFS COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE     ERROR           ;YES
*
* INIT ISCSI 1
INIT    LEA.L   ISCSI+BUF_1,A1  ;GET I/O BUFFER #1
        MOVE.L  #$07040000,(A1) ;MAININIT PARAMETERS
        MOVE.L  #ISCSI+BUF_1,4(A0) ;POINTER TO PARAMS
        MOVE.W  #0,2(A0)        ;CLEAR THIS
        MOVE.W  #0,12(A0)       ;LUN = 0
        MOVE.W  #$000D,(A0)     ;MAININIT COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE     ERROR           ;YES
*
* COMPARE 1000 BLOCKS FROM 0 TO 20000 ON ID 0
COMP    LEA.L   ISCSI+BUF_1,A1  ;PARAMETER POINTER
        MOVE.W  #0,(A1)+        ;SOURCE : ID = 0, LUN = 0
        MOVE.L  #0,(A1)+        ;         START BLOCK ADDRESS
        MOVE.W  #0,(A1)+        ;DEST   : ID = 0, LUN = 0
        MOVE.L  #20000,(A1)+    ;         START BLOCK ADDRESS
        MOVE.W  #1000,(A1)+     ;NUMBER OF BLOCKS TO COMPARE
        MOVE.L  #ISCSI+BUF_1,4(A0) ;PARAMETER POINTER
        MOVE.W  #$0059,(A0)     ;COMPARE COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE.S   ERROR           ;YES
        XEXT                    ;NO, EXIT TO PDOS
*
* ISCSI-1 ERRORS COME HERE
ERROR   XPMC    ERMSG           ;PRINT MESSAGE
        MOVE.L  (A0),D1         ;GET ERROR CODE
        XCBH                    ;CONVERT
        XPEL                    ;PRINT
        XEXT
*
ERMSG   DC.B    10,13,'*** ISCSI-1 ERROR - ',0
        EVEN
BUFFER  DS.B    1024
        END
```

Mnemonic: **LOCKUN - Lock Local Unit for SCSIbus Access**

Hex Code: $005A Command completion without interrupt.
$105A Command completion with interrupt.

DESCRIPTION:

The command allows the specified floppy disks to lock for SCSIbus access. If any initiator selects one of these locked units, a "unit not ready" error will be returned.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Lock unit flags bit# meaning 0 lock all units 1 lock floppy disk 1 2 lock floppy disk 2 3 lock floppy disk 3 4 lock floppy disk 4 When a flag is set (1) the corresponding unit is locked
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **FREEUN** - Free Local Unit for SCSIbus Access

Hex Code: $005B Command completion without interrupt.
$105B Command completion with interrupt.

DESCRIPTION:

This command frees the specified floppy disks for SCSIbus access.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unlock unit flags bit# meaning 0 free all units 1 free floppy disk 1 2 free floppy disk 2 3 free floppy disk 3 4 free floppy disk 4 When a flag is set (1) the corresponding unit is unlocked.
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error

Mnemonic: **COPY** - Copy Data from One Logical Unit to Another

Hex Code: $005C Command completion without interrupt.
$105C Command completion with interrupt.

DESCRIPTION:

This command copies data between two logical units or between different logical blocks on the same logical unit.

NOTE: The COPY command is only allowed for SCSI Winchester units!

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Pointer to parameter block
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

PARAMETER BLOCK

Offset	Size	Meaning
00	B	First unit - ID
01	B	First unit - logical unit number
02	L	First unit - start block address
06	B	Second unit - ID
07	B	Second unit - logical unit number
08	L	Second unit - start block address
12	W	Number of blocks to copy

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 - Successful, no error
800B, 900B - Address error
800F, 900F = Extended error message (refer to Appendix A for detailed information)

EXAMPLE:

```
ISCSI   EQU     $A00000         ;ISCSI-1 BASE ADDRESS
CR_1    EQU     $2100           ;COMMAND RAM 1
BUF_1   EQU     $2400           ;I/O BUFFER 1
BUF_2   EQU     $2B00           ;I/O BUFFER 2
*
        OPT     ALT,NOWARN
*
* SETUP ISCSI-1 BASE ADDRESS
SETUP   LEA.L   ISCSI+CR_1,A0   ;GET COMMAND RAM
        MOVE.L  #ISCSI,4(A0)    ;SET BASE ADDRESS
        MOVE.W  #$0003,(A0)     ;SETOFFS COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE     ERROR           ;YES
*
* INIT ISCSI 1
INIT    LEA.L   ISCSI+BUF_1,A1  ;GET I/O BUFFER #1
        MOVE.L  #$07040000,(A1) ;MAININIT PARAMETERS
        MOVE.L  #ISCSI+BUF_1,4(A0) ;POINTER TO PARAMS
        MOVE.W  #0,2(A0)        ;CLEAR THIS
        MOVE.W  #0,12(A0)       ;LUN = 0
        MOVE.W  #$000D,(A0)     ;MAININIT COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE     ERROR           ;YES
*
* COPY 1000 BLOCKS FROM 0 TO 20000 ON ID 0
COPY    LEA.L   ISCSI+BUF_1,A1  ;PARAMETER POINTER
        MOVE.W  #0,(A1)+        ;SOURCE : ID = 0, LUN = 0
        MOVE.L  #0,(A1)+        ;         START BLOCK ADDRESS
        MOVE.W  #0,(A1)+        ;DEST   : ID = 0, LUN = 0
        MOVE.L  #20000,(A1)+    ;         START BLOCK ADDRESS
        MOVE.W  #1000,(A1)+     ;NUMBER OF BLOCKS TO COPY
        MOVE.L  #ISCSI+BUF_1,4(A0) ;PARAMETER POINTER
        MOVE.W  #$005C,(A0)     ;COPY COMMAND
@001    TST.W   (A0)            ;WAIT UNTIL DONE
        BPL.S   @001
        TST.B   1(A0)           ;ANY ERROR?
        BNE.S   ERROR           ;YES
        XEXT                    ;NO, EXIT TO PDOS
* ISCSI-1 ERRORS COME HERE
ERROR   XPMC    ERMSG           ;PRINT MESSAGE
        MOVE.L  (A0),D1         ;GET ERROR CODE
        XCBH                    ;CONVERT
        XPEL                    ;PRINT
        XEXT
*
ERMSG   DC.B    10,13,'*** ISCSI-1 ERROR -> ',0
        EVEN
BUFFER  DS.B    1024
        END
```

Mnemonic: **CHAIN** - Enter Command Chaining Mode

Hex Code: $005D Command completion without interrupt.
$105D Command completion with interrupt.

DESCRIPTION:

This command enters the chain mode and starts executing the chain commands which are placed in the current chain buffer. For a detailed description of the chain mode refer to section 2.4 of this manual.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Number of chained commands
4	L	Pointer to first CMDRAM
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Pointer to error command RAM on chain buffer.
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800F, 900F = Extended Error Message

Mnemonic: **SEARCH** - Search Data on a Logical Unit

Hex Code: 80X5E Command completion without interrupt.
81X5E Command completion with interrupt.

DESCRIPTION:

This command scans a specified number of blocks on a logical unit for a matching data pattern to the given data pattern. The return parameters are the the block number and the offset of the block for the first found match.

NOTE: The SEARCH command is only allowed for SCSI Winchester units!

PARAMETER PASSING

IN: CMDRAM

Offset	Size	Meaning
0	W	Command/Target ID
2	W	Unused/reserved
4	L	Pointer to parameter block
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

PARAMETER BLOCK

Offset	Size	Meaning
00	B	Target ID
01	B	Target LUN
02	L	Start block address
06	W	Number of blocks to search
08	W	Length of data pattern
0A	B	First byte of a data pattern
:		
:		
nn	B	Last byte of data pattern

OFF: CMDRAC4

Offset	Size	Meaning
0	W	Status/Error return code
2	W	[extended error message]
4	L	Block address of the pattern match
8	L	Offset into the block
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES

8000, 9000 = Successful, no error
800F, 900F - Extended error message

Mnemonic: **CLRHASH** - Clear Internal Hashing Buffer List

Hex Code: $005F Command completion without interrupt.
$105F Command completion with interrupt.

DESCRIPTION:

This command clears the list of the internal hashing buffers. It means the buffer contents are set to invalid and the next read block operation (with hashing enabled) first fills one of the buffers before the block is searched in the buffer.

The command is useful when the hashing buffer area needs to be used as I/O buffer for high speed or big data SCSIbus operation.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **TRSPMOD** - Handle SCSI Command in Transparent Mode

Hex Code: $0060 Command completion without interrupt.
$1060 Command completion with interrupt.

DESCRIPTION:

The TRSPMOD command transfers a SCSI command from the I/O buffer to the specified target. The resulting parameters and the SCSIbus state are returned in CMDRAM. Data and extended messages are returned in the I/O buffer area.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Data transfer length (bytes)
4	L	Pointer to the Command in Buffer
8	L	Pointer to data area
12	W	Target ID
14	W	Reserved for interprocessor communication

NOTE: A preceding byte which contains the SCSI command length is required at the command block.

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	(extended error message)
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error
800B, 900B = Address error
8006, 9006 = Missing or invalid parameter
800F, 900F = Extended error message (Please refer to Appendix A of this manual for detailed information.)

Mnemonic: **SCSIRST** - Reset All SCSIbus Devices

Hex Code: $0061 Command completion without interrupt.
$1061 Command completion with interrupt.

DESCRIPTION:

This command drives out the SCSIbus reset signal and resets all SCSI devices.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **TARGMOD** - Enable/disable Controller Target Mode

Hex Code: $0067 Command completion without interrupt.
$1062 Command completion with interrupt.

DESCRIPTION:

This command enables/disables (depending upon the current mode) the ISCSI-1 target mode. This means that when the target mode is enabled, the ISCSI-1 is selectable as a target via the SCSIbus. The target mode can be used to connect two host systems via the SCSIbus or to allow SCSIbus access to the floppy disks.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **TARGINT** - Set Target Interrupt Mode

Hex Code: $0063 Command completion without interrupt.
$1063 Command completion with interrupt.

DESCRIPTION:

When the SCSI target mode is enabled and a message is received from an initiator, optionally an interrupt can be generated. The message info is returned at address $20F0 (board relative) in the following format:

Offset	Size	Meaning
0	W	$0000 if no message was received, negative if a message was received from an initiator
2	W	Initiator ID
4	L	Pointer to the message block (board relative)
8	W	Number of 256 byte blocks

NOTE: The interrupt is always triggered through channel 0 of the BIM.

PARAMETER PASSING:

IN: CMDRAM

Offset	Size	Meaning
0	W	Command
2	W	0 = disable target interrupts 1 = enable target interrupts
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

OUT: CMDRAM

Offset	Size	Meaning
0	W	Status/Error return code
2	W	Unused/reserved
4	L	Unused/reserved
8	L	Unused/reserved
12	W	Unused/reserved
14	W	Reserved for interprocessor communication

POSSIBLE RETURN CODES:

8000, 9000 = Successful, no error

Mnemonic: **ABORT** - Abort Command Execution

Hex Code: NONE!

DESCRIPTION:

The ABORT function is not a normal ISCSI command, but can be used to abort any command execution. The ABORT function is executed as an interrupt service routine which handles the PI/T port interrupt at level 7. The interrupt can be triggered by reading the address BASE + $1001 from the ISCSI-1. The firmware terminates any command execution, resets the SCSIbus and restarts the main loop which polls the command RAMs.

EXAMPLE:

```
ABORT   LEA.L   ISCSI+ABRT,A0      ;GET ABORT ADDRESS
        MOVE.B  (A0),D0            ;THIS TRIGGERS THE INTERRUPT
        LEA.L   ISCSI+CMDRAM,A0    ;GET CMDRAM POINTER
@001    TST.W   (A0)               ;WAIT FOR ISCSI-1 IDLE
        BPL.S   @001
                :
                :
                :
```

3.5 Target Mode Description

3.5.1 Overview

The ISCSI-1 supports a subset of the SCSI Common Command Set (CCS), which is defined in the ANSI X3T9.2 standardization note. The commands are:

COMMAND	OPCODE
TEST UNIT READY	$00
REQUEST SENSE	$03
FORMAT UNIT	$04
READ/RECEIVE	$08
WRITE/SEND	$0A
INQUIRY	$12

The ISCSI-1 supports up to five logical units accessable from the SCSIbus:

UNIT #	DEVICE
0	ISCSI-1 Processor Device
1	Floppy Disk Drive 1
2	Floppy Disk Drive 2
3	Floppy Disk Drive 3
4	Floppy Disk Drive 4

The ISCSI-1 supports intersystem connection using the target message area to inform the host about received data. Optional interrupts can be generated when the data is transferred from the initiator. (See the TARGINT command description.)

Address	Size	Meaning
$20F0	W	Contains 0 if no data has been received from LUN #0. Becomes negative when data is received from initiator.
$20F2	W	Contains the initiator's ID.
$20F4	L	This is a pointer to the received data.
$20F8	W	Contains the number of 256 byte blocks received.

3.5.2 Detailed Description of the Supported CCS Commands

TEST UNIT READY

The TEST UNIT READY command returns a status byte which reflects the current state of the target.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	0	0	0	0	0	OPCODE
L	U	N	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

RETURNED STATUS BYTE:

$s00 - Good return, unit ready
$02 - Check condition

REQUEST SENSE

The REQUEST SENSE command requests that the target transfer sense data to the initiator.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	0	0	0	1	1	OPCODE
L	U	N	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	1	0	0	ALLOCATION LENGTH
0	0	0	0	0	0	0	0	

SENSE DATA:

The REQUEST SENSE command always returns a block of four bytes of information (nonextended sense data format).

BYTE	MEANING	
00	Sense key, error information	
01	Logical block address (MSB)	Always set to zero
02	Logical block address	
03	Logical block address (LSB)	

SENSE KEYS	
$00	NO SENSE
$01	RECOVERED ERROR
$02	NOT READY
$03	MEDIUM ERROR
$04	HARDWARE ERROR
$05	ILLEGAL REQUEST
$06	not used
$07	not used
$08	not used
$09	not used
$0A	not used
$0B	ABORT COMMAND
$0C	not used
$0D	VOLUME OVERFLOW
$0E	not used
$0F	reserved

FORMAT UNIT

The FORMAT UNIT command can be used to format one of the logical units of the ISCSI-1. If the LUN number in the command is 1 through 4, the corresponding floppy disk drive will be formatted. If the LUN number is 0, the ISCSI-1 will perform a restart of the firmware.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	0	0	1	0	0	OPCODE
L	U	N	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

READ/RECEIVE

The READ/RECEIVE command requests a data transfer from the target to the initiator. The READ/RECEIVE affects the ISCSI-1 in the following ways.

- If the logical unit number is set to zero, the ISCSI-1 memory contents are returned in blocks of 256 bytes, starting with the memory address which is specified as the block address.
- If the logical unit number is one through four, the specified blocks/sectors of the corresponding disk drive are returned.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	0	1	0	0	0	OPCODE
L	0	N	X	X	X	X	X	BLOCK ADDRESS (MSB)
X	X	X	X	X	X	X	X	BLOCK ADDRESS
X	X	X	X	X	X	X	X	BLOCK ADDRESS (LSB)
X	X	X	X	X	X	X	X	NUMBER OF BLOCKS
0	0	0	0	0	0	0	0	

WRITE/SEND

The WRITE/SEND command requests a data transfer from the initiator to the target. The WRITE/SEND affects the ISCSI-1 in the following ways.

- If the logical unit number is set to zero, the data is transferred to the memory address of the ISCSI 1 which is specified with the block address. The block size for this operation is 256 bytes.

  If the logical unit number is 1 through 4, then the specified blocks/sectors are written to the corresponding floppy disk drive number.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	0	1	0	1	0	OPCODE
L	0	M	X	X	X	X	X	BLOCK ADDRESS (MSB)
X	X	X	X	X	X	X	X	BLOCK ADDRESS
X	X	X	X	X	X	X	X	BLOCK ADDRESS (LSB)
X	X	X	X	X	X	X	X	NUMBER OF BLOCKS
0	0	0	0	0	0	0	0	

INQUIRY

The INQUIRY command requests that the information, regarding parameters of the target and its attached peripheral device(s) be sent to the initiator.

COMMAND BLOCK:

7	6	5	4	3	2	1	0	BIT #
0	0	0	1	0	0	1	0	OPCODE
L.	U	N	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	1	0	0	1	0	1	ALLOCATION LENGTH
0	0	0	0	0	0	0	0	

INQUIRY DATA:

BYTE	MEANING
00	Peripheral device type
01	Not used, always 0
02	Not used, always 0
03	Reserved
04	Additional data length
05 - 31	Vendor-unique parameter bytes

This page was intentionally left blank

4. THE ISCSI-1 INTERRUPT STRUCTURE

On command complete, the ISCSI-1 can generate an interrupt if the command interrupt bit is set (refer to section 3.1 in this manual). The interrupt is sent to the VMEbus via the on-board BIM. Four interrupts are supported:

BIM	Interrupt
0	Command successfully completed on channel 1
1	Command completed on channel 1, error was detected
2	Command successfully completed on channel 2
3	Command completed on channel 2, error was detected

The interrupt vectors and levels are programmable by programming the BIM. For detailed information how to program the BIM, please refer to the Hardware User's Manual.

The target message interrupts are always sent via BIM channel 0.

This page was intentionally left blank

APPENDIX TO THE
FIRMWARE USER'S MANUAL

This page was intentionally left blank

TABLE OF CONTENTS

Error Messages Summary.....APPENDIX A

ISCSI-1 Default Setups.....APPENDIX B

ISCSI-1 Memory Layout.....APPENDIX C

ISCSI-1 Command Summary.....APPENDIX D

Advanced Programming Examples.....APPENDIX E

Sample of C Routines.....APPENDIX F

This page was intentionally left blank

APPENDIX A

ISCSI-1 Error Messages Summary

$8001 SCSIbus Parity Error
$8006 Missing or invalid parameter(s)
$8007 Illegal use of an ISCSI-1 Command
$800B Address Error, Invalid Pointer or Address Parameter
$800C An exception was generated during command execution. The command RAM contains additional information:

Offset	Size	Meaning
0	W	Error code
2	W	Special format word (68010)
4	L	Program counter (error address)
8	L	Pointer to a system stack copy (This pointer is always board relative!!!)
12	W	Unused/reserved
14	W	Reserved for interprocessor communic.

$800E Data compare mismatch
$800F Extended Error Message. The Error Message extension is located in CMDRAM at offset $02.
$8FFF Unknown Command

Extended Error Messages

SCSI ERRORS

$800F0101 ISCSI-1 timeout error
$800F0102 Logical unit is not ready
$800F0103 SCSIbus read error
$800F0104 SCSIbus write error
$800F0105 Specified logical block not found
$800F0106 Target reservation conflict
$800F0107 Format error
$800F0108 Undefined/unknown SCSI command
$800F0109 Target did not respond during selection phase

OR

Target returned fife times CHECK CONDITION state If this error occurs, an automatic REQUEST SENSE is isssued by the firmware and the return values are store in the DPR. A pointer to this parameters is returned at offset 4 in the command ram.

FLOPPY DISK ERRORS

$800F0302 Floppy disk not ready
$800F0303 FDC read error
$800F0304 FDC write error
$800F0305 Floppy disk record not found
$800F0306 Floppy disk write protected
$800F0307 Floppy disk format error

This page was intentionally left blank

APPENDIX - B

ISCSI-1 Default Setups

Global Definitions:

```
ISCSI 1 SCSIbus ID.............7
Interfacing Sector Size.......256 Bytes
VMEbus Address Offset.........0
Data buffering for Write......disabled
```

SCSIbus Definitions:

```
Target Mode...................disabled
Sector Deblocking Mode .......disabled
Auto Backup Mode..............disabled
Hashing for Read..............enabled
Target interrupts.............disabled
Physical Block Address Offset..0
Auto Flush Time Interval......20 seconds
```

Floppy Disk Drives:

```
Locked/reserved Units.........none
Sides per Disk................2
Cylinders per Disk............80
Sectors per Cylinder..........32
Bytes per Sector..............256
Disk Density..................double/80 track drive
Sector Offset (BIAS)..........0
Format Pattern................$E5
Sector Interleave for Format...0
Step Rate.....................0 = 6ms
```

This page was intentionally left blank

APPENDIX C

ISCSI-1 MEMORY LAYOUT FROM LOCAL SIGHT

==========	=======	BOARD BASE	==========
$0000 - $0400	1 KB	VECTOR AREA	
$0400 - $2000	3 KB	FIRMWARE WORK AREA	
==========	==========	DUAL PORTED MEMORY	==========
$2000	1 W	HOLDS THE BOARD AND THE FIRMWARE REVISION CODE	
$20F0 - $20FA	5 W	TARGET MESSAGE INFO AREA	
$2100 - $210F	16 B	CMDRAM #1	
$2110 - $22FF	496 B	CMD CHAIN AREA #1 = 31 CMDRAMS	
$2300 - $2AFF	2 KB	I/O BUFFER #1	
$2B00 - $32FF	2 KB	I/O BUFFER #2	
$3300 - $34FF	512 B	CMD CHAIN AREA #2 = 31 CMDRAMS	
$3500 - $350F	16 B	CMDRAM #2	
$3510 - $3FFF	3 KB	RESERVED	
$4000 - $1FFFF	112 KB	HASHING BUFFERS (7 BUFFERS, 16KB/BUFFER)	
==========	==========	END OF RAM	==========

ISCSI-1 MEMORY LAYOUT FROM SYSTEM/VMEBUS SIGHT

========== ISCSI-1 BASE ADDRESS ==========

A00000 - A00FFF Reading a WORD from an EVEN address such as A00000 will return RESET/SYSFAIL status information in the bits 8,9 and 10. Read or write a BYTE at an ODD address will access the ISCSI-1 BIM registers as described below.

A00001	1	B	BIM control register 0
A00003	1	B	BIM control register 1
A00005	1	B	BIM control register 2
A00007	1	B	BIM control register 3
A00009	1	B	BIM vector register 0
A0000B	1	B	BIM vector register 1
A0000D	1	B	BIM vector register 2
A0000F	1	B	BIM vector register 3

This structure is continued to address A00FFF

A01001-A017FF	1	B	Reading performs a local interrupt (odd addresses only)
A01801-A01FFF	1	B	Reading performs a local reset (odd addresses only)

========== BEGIN OF DPR ==========

$A02000		1 W	HOLDS THE BOARD AND THE FIRMWARE REVISION CODE
$A020F0	$A020FA	5 W	TARGET MESSAGE INFO AREA
$A02100 -	$A0210F	16 B	CMDRAM #1
$A02110	$A022FF	496 B	CMD CHAIN AREA #1 = 31 CMDRAMs
$A02300 -	$A02AFF	2 KB	I/O BUFFER #1
$A02B00 -	$A032FF	2 KB	I/O BUFFER #2
$A03300 -	$A034FF	512 B	CMD CHAIN AREA #2 = 31 CMDRAMs
$A03500 -	$A0350F	16 B	CMDRAM #2
$A03510 -	$A03FFF	3 KB	RESERVED
$A04000 -	$A1FFFF	112 KB	HASHING BUFFERS (7 BUFFERS, 16KB/BUFFER)

========== END OF RAM AREA ==========

APPENDIX - D

Mnemonic	Hexcode	Function
SETOFFS	0003	Set VMEbus address offset.
	1003	Same function but complete with interrupt.
DEFOFFS	0008	Define physical block address offset.
	1008	Same function but complete with interrupt.
RESET	0009	Reset the firmware to power-on state.
SUNPARM	0X0A	Install logical unit (SCSI and local floppy disk).
	1X0A	Same function but complete with interrupt.
BUFFWR	000B	Enable/disable buffering for write operations.
	100B	Same function but complete with interrupt.
WRPARAL	000C	Enable/disable runtime backup to parallel logical unit.
	100C	Same function but complete with interrupt.
MAININIT	000D	Controller main initialization and installation command.
	100D	Same function but complete with interrupt.
SFTIME	000E	Set auto flush time interval.
	100E	Same function but complete with interrupt.
DEBLOCK	000F	Define sector deblocking mode.
	100F	Same function but complete with interrupt.
GETCHR	0X10	Read one byte from a logical block.
	1X10	Same function but complete with interrupt.
PUTCHR	0X14	Write one byte to a logical block.
	1X14	Same function but complete with interrupt.
GETBLOC	0X20	Read fix sized blocks/sectors from a logical unit.
	1X20	Same function but complete with interrupt.
PUTBLOC	0X22	Write fix sized blocks/sectors to a logical unit.
	1X22	Same function but complete with interrupt.
GETSTR	0X30	Read a delimited string from a logical block.
	1X30	Same function but complete with interrupt.

Mnemonic	Hexcode	Function
PUTSTR	0X32	Write a delimited string to a logical block.
	1X32	Same function but complete with interrupt.
GETCNT	0X33	Read a counted string from a logical block
	1X33	Same function but complete with interrupt.
PUTCNT	0X35	Write a counted string to a logical block.
	1X35	Same function but complete with interrupt.
CTLSTAT	0044	Return controller status information.
	1044	Same function but complete with interrupt.
CHKTARG	0045	Check SCSI targets for existence and device type.
	1045	Same function but complete with interrupt.
GUNPARM	0X46	Return logical units parameter.
	1X46	Same function but complete with interrupt.
EXPROG	0050	Execute user program in local DPR.
	1050	Same function but complete with interrupt.
BACKUP	0055	Backup a logical unit to another.
	1055	Same function but complete with interrupt.
FLUSH	0056	Flush all modified buffers.
	1056	Same function but complete with interrupt.
FORMAT	0X57	Format a logical unit.
	1X57	Same function but complete with interrupt.
FTRACK	0E58	Format a physical track on a floppy disk drive.
	1E58	Same function but complete with interrupt.
COMPARE	0059	Compare data between two logical units.
	1059	Same function but completes with interrupt.
LOCKUN	005A	Lock local unit against access from SCSIbus initiator.
	105A	Same function but complete with interrupt.
FREEUN	005B	Free local unit for SCSIbus access.
	105B	Same function but complete with interrupt.
COPY	005C	Copy data from one logical unit to another or between logical blocks on the same logical unit.
	105C	Same function but complete with interrupt.
CHAIN	005D	Enter command chaining mode.
	105D	Same function but complete with interrupt.

Mnemonic	Hexcode	Function
SEARCH	0X5E	Search for data on a logical unit.
	1X5E	Same function but complete with interrupt.
CLRHASH	005F	Clears the hashing buffer list entries.
	105F	Same function but complete with interrupt.
TRSPMOD	0060	Send a command in transparent mode to SCSIbus.
	1060	Same function but complete with interrupt.
SCSIRST	0061	Perform SCSIbus reset.
	1061	Same function but complete with interrupt.
TARGMOD	0062	Enable/disable target mode.
	1062	Same function but complete with interrupt.
TARGINT	0063	Enable/disable target message interrupts.
	1063	Same function but complete with interrupt.

This page was intentionally left blank

APPENDIX E

ISCSI-1 Advanced Programming Examples

All programming examples in this section are written under the PDOS Operating System in M68000 Assembler language. They can be assembled with the PDOS macro assembler MASM and are executable under PDOS. Appended to some of the programs is an execution example of the program.

The examples are:

- **REPORT** This program expects as command line parameter the ID number of an SCSI target and reports the commented device parameters as they are returned by the MODE SENSE command.

  BUS_CHK This program uses the ISCSI-1 command CHKTARG, checks SCSI targets for existence and device type and displays the results as a table.

- **IDDUMP** This example is an ISCSI-1 disk dump utility. Refer to the listing and execution example to see how it works.

- **CONNECT** This is an ISCSI-1 target mode example which shows how to connect two ISCSI-1 boards together.

- **GETIBM** This example shows how to read IBM-PC floppy disks on the ISCSI-1, under the PDOS operating system.

EXAMPLE 1: REPORT

```
*****************************************************************
***                  ISCSI-1 PROGRAMMING EXAMPLE                ***
***                                                             ***
***                REPORT DISKS CURRENT PARAMETERS              ***
***                                                             ***
*****************************************************************
INCLUDE ISCSI:IN
*
OPT     ALT,NOWARN
*
* FIRST WE GET THE ID FROM THE COMMAND LINE
*
REPORT  XGNP                        ;GET ID FROM COMMAND LINE
        MOVE.L  A1,A0               ;STORE POINTER
        XCDB                        ;CONVERT TO BINARY
        MOVE.L  D1,D5               ;SAVE IT FOR LATER USE
        XPCL                        ;CR/LF
        XPMC    STRTMSG             ;HEADLINE
        MOVE.L  A0,A1               ;DISPLAY ID
        XPEL
        XPCL                        ;<NL>
        BSR     INQU                ;IDENTIFY DRIVE
*
* FIRST TIME MODE SENSE
* IT IS ONLY USED TO GET THE REAL TRANSFER LENGTH
*
        LEA.L   ISCSI+CR_1,A0       ;CMDRAM
        LEA.L   ISCSI+BUF_1,A1      ;CMD BUFFER
        LEA.L   MODSENS(PC),A2      ;MODE SENSE COMMAND
        MOVE.W  D5,D1               ;TARGET ID
        MOVE.W  #77,D2              ;TRANSFER COUNTER
        BSR     INIT                ;INIT CMDRAM AND COMMAND BUFFER
        BSR     WAIT                ;START AND WAIT UNTIL DONE

* LOOK FOR TRANSFER LENGTH AND DO MODE SENSE AGAIN
*
        LEA.L   ISCSI+CR_1,A0       ;CMDRAM
        LEA.L   ISCSI+BUF_1,A1      ;COMMAND BUFFER
        LEA.L   MODSENS(PC),A2      ;INQUIRY COMMAND
        MOVE.W  D5,D1               ;TARGET ID
        LEA.L   ISCSI+BUF_2,A3      ;DATA BUFFER
        MOVE.B  (A3),D2             ;TRANSFER COUNTER
        MOVE.B  D2,5(A2)            ;SET ALLOCATION LENGTH
        BSR     INIT                ;INIT CMDRAM AND COMMAND BUFFER
        BSR     WAIT                ;START AND WAIT UNTIL DONE
```

EXAMPLE 1 continued

```
*
* NOW WE DECODE THE RETURN PARAMETERS
* AND DISPLAY THEM TOGETHER WITH COMMENTS
*
          LEA.L    BUFFER(PC),A1     ;STRING BUFFER
          LEA.L    ISCSI+BUF_2,A2    ;GET DATA BUFFER
          MOVEQ.L  #0,D1             ;CLEAR IT
          XPMC     MSG00             ;DISPLAY MESSAGE
          MOVE.B   (A2)+,D1          ;IGNORE THIS
          MOVE.B   (A2)+,D1
          CMP.B    #0,D1
          BNE.S    @600
          XPMC     DMSG0
          BRA.S    @610
@600      CMP.B    #1,D1
          BNE.S    @601
          XPMC     DMSG1
          BRA.S    @610
@601      XPMC     DMSGU
@610      MOVE.B   (A2)-,D1
          MOVE.B   (A2)+,D1
          TST.B    D1
          BEQ      @100
*
* BLOCK DESCRIPTOR
*
          XPMC     MSG01
          XPMC     MSG07
          MOVE.B   (A2)+,D1
          XCBX
          XPEL
          XPMC     MSG03
          MOVE.B   (A2)+,D1
          LSL.L    #8,D1
          MOVE.B   (A2)+,D1
          LSL.L    #8,D1
          MOVE.B   (A2)+,D1
          XCBX
          XPEL
          MOVE.B   (A2)+,D1
          CLR.L    D1
          XPMC     MSG04
          MOVE.B   (A2)+,D1
          LSL.L    #8,D1
          MOVE.B   (A2)+,D1
          LSL.L    #8,D1
          MOVE.B   (A2)+,D1
          XCBX
          XPEL
          BSR      GETKEY            ;STOP DISPLAY AND WAIT FOR A KEY
```

EXAMPLE 1 continued

```
*
* ERROR RECOVERY PARAMETERS = PAGE 01
*
@100    MOVE.B  (A2)+,D1
        ANDI.L  #$1F,D1
        CMP.B   #1,D1
        BNE     @200
        XPMC    MSG05
        XPMC    MSG06
        MOVE.B  (A2)+,D1
        MOVE.B  (A2)+,D1
        MOVE.B  #'%',(A1)
        BTST    #7,D1
        BEQ.S   @700
        MOVE.B  #'1',1(A1)
        BRA.S   @701
@700    MOVE.B  #'0',1(A1)
@701    BTST    #6,D1
        BEQ.S   @702
        MOVE.B  #'1',2(A1)
        BRA.S   @703
@702    MOVE.B  #'0',2(A1)
@703    BTST    #5,D1
        BEQ.S   @704
        MOVE.B  #'1',3(A1)
        BRA.S   @721
@704    MOVE.B  #'0',3(A1)
@721    BTST    #4,D1
        BEQ.S   @705
        MOVE.B  #'1',4(A1)
        BRA.S   @706
@705    MOVE.B  #'0',4(A1)
@706    BTST    #3,D1
        BEQ.S   @707
        MOVE.B  #'1',5(A1)
        BRA.S   @708
@707    MOVE.B  #'0',5(A1)
@708    BTST    #2,D1
        BEQ.S   @709
        MOVE.B  #'1',6(A1)
        BRA.S   @710
@709    MOVE.B  #'0',6(A1)
@710    BTST    #1,D1
        BEQ.S   @711
        MOVE.B  #'1',7(A1)
        BRA.S   @712
@711    MOVE.B  #'0',7(A1)
@712    BTST    #0,D1
        BEQ.S   @713
        MOVE.B  #'1',8(A1)
        BRA.S   @714
@713    MOVE.B  #'0',8(A1)
```

EXAMPLE 1 continued

```
@714    MOVE.B  #$0,$14(A1)
        XPEL
        XPMC    MSG07
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        XPMC    MSG08
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        XPMC    MSG09
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        XPMC    MSG10
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        XPMC    MSG11
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        BSR     GETKEY
*
* DISCONNECT/RECONNECT PARAMETERS = PAGE 02
*
@200    MOVE.B  (A2)+,D1
        ANDI.L  #$1F,D1
        CMP.B   #2,D1
        BNE     @300
        MOVE.B  (A2)+,D1
        XPMC    MSG12
        XPMC    MSG13
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG14
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG15
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
```

EXAMPLE 1 continued

```
          XPMC    MSG16
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
          XPEL
          CLR.L   D1
          MOVE.B  (A2)+,D1
          MOVE.B  (A2)+,D1
          BSR     GETKEY
*
* DIRECT ACCESS DEVICE FORMAT PARAMETERS = PAGE 03
*
@300      MOVE.B  (A2)+,D1
          ANDI.L  #$1F,D1
          CMP.B   #3,D1
          BNE     @400
          MOVE.B  (A2)+,D2
          XPMC    MSG17
          XPMC    MSG18
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
          XPEL
          CLR.L   D1
          XPMC    MSG19
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
          XPEL
          CLR.L   D1
          XPMC    MSG20
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
          XPEL
          CLR.L   D1
          XPMC    MSG21
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
          XPEL
          CLR.L   D1
          XPMC    MSG22
          MOVE.B  (A2)+,D1
          LSL.L   #8,D1
          MOVE.B  (A2)+,D1
          XCBX
```

EXAMPLE 1 continued

```
        XPEL
        CLR.L   D1
        XPMC    MSG23
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG24
        MOVE.B  (A2)-,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG25
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG26
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)-,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG27
        MOVE.B  (A2)+,D1
        MOVE.B  #'4',(A2)
        BTST    #7,D1
        BEQ.S   @800
        MOVE.B  #'1',1(A1)
        BRA.S   @801
@800    MOVE.B  #'0',1(A1)
@801    BTST    #6,D1
        BEQ.S   @802
        MOVE.B  #'1',2(A1)
        BRA.S   @803
@802    MOVE.B  #'0',2(A1)
@803    BTST    #5,D1
        BEQ.S   @804
        MOVE.B  #'1',3(A1)
        BRA.S   @821
@804    MOVE.B  #'0',3(A1)
@821    BTST    #4,D1
        BEQ.S   @805
        MOVE.B  #'1',4(A1)
        BRA.S   @806
```

EXAMPLE 1 continued

```
@805    MOVE.B  #'0',4(A1)
@806    BTST    #3,D1
        BEQ.S   @807
        MOVE.B  #'1',5(A1)
        BRA.S   @808
@807    MOVE.B  #'0',5(A1)
@808    BTST    #2,D1
        BEQ.S   @809
        MOVE.B  #'1',6(A1)
        BRA.S   @810
@809    MOVE.B  #'0',6(A1)
@810    BTST    #1,D1
        BEQ.S   @811
        MOVE.B  #'1',7(A1)
        BRA.S   @812
@811    MOVE.B  #'0',7(A1)
@812    BTST    #0,D1
        BEQ.S   @813
        MOVE.B  #'1',8(A1)
        BRA.S   @814
@813    MOVE.B  #'0',8(A1)
@814    MOVE.B  #$0,9(A1)
        XPEL
        CLR.L   D1
        MOVE.B  (A2)+,D1
        MOVE.B  (A2)+,D1
        MOVE.B  (A2)-,D1
        BSR     GETKEY
*
* RIGID DISK DRIVE GEOMETRY PARAMETERS - PAGE 04
*
@400    MOVE.B  (A2)+,D1
        ANDI.L  #$1F,D1
        CMP.B   #4,D1
        BNE     @500
        MOVE.B  (A2)+,D1
        XPMC    MSG28
        XPMC    MSG29
        MOVE.B  (A2)+,D1
        CLR.L   D1
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG30
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
```

EXAMPLE 1 continued

```
        XPMC    MSG31
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG32
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG33
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
        XPMC    MSG34
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        LSL.L   #8,D1
        MOVE.B  (A2)+,D1
        XCBX
        XPEL
        CLR.L   D1
*
@500    XEXT
*
**************************************
*
* INIT COMMAND BUFFER AND COMMAND RAM
*
INIT    MOVE.L  #7,D0           ;7 BYTES TO COPY
@001    MOVE.B  (A2)+,(A1)+     ;TRANSFER COMMAND
        SUBQ.L  #1,D0
        BNE.S   @001
        MOVE.W  D1,12(A0)       ;SET TARGET ID
        MOVE.W  D2,2(A0)        ;DATA TRANSFER LENGTH
        MOVE.L  #BUF_1,4(A0)    ;SET BUFFER POINTERS
        MOVE.L  #BUF_2,8(A0)
        RTS
```

EXAMPLE 1 continued

```
*
************************************
*
* START EXECUTION AND WAIT UNTIL DONE
*
WAIT    MOVE.W  #$60,(A0)       ;TRSPMOD COMMAND
@001    TST.W   (A0)            ;WAIT FOR COMPLETION STATE
        BPL.S   @001
        RTS
*
************************************
*
* WAIT FOR A KEYBOARD INPUT
*
GETKEY  MOVE.L  D0,-(A7)
        XPMC    KEYMSG
        XGCP
        MOVE.L  (A7)+,D0
        RTS
*
************************************
*
* GET INQUIRY MESSAGE FROM TARGET
*
INQU    MOVEM.L D0-A6,-(A7)
        LEA.L   ISCSI+CR_1,A0   ;CMDRAM POINTER
        LEA.L   ISCSI+BUF_1,A1  ;COMMAND BUFFER
        LEA.L   INQUIRY(PC),A2  ;INQUIRY COMMAND
        MOVE.W  D5,D1           ;TARGET ID
        MOVE.W  #$0,D2          ;TRANSFER COUNTER
        BSR     INIT            ;INIT CMDRAM AND CMD BUFFER
        BSR     WAIT            ;START AND WAIT UNTIL DONE
        LEA.L   ISCSI+BUF_2,A1  ;GET DATA BUFFER
        ADD.L   #8,A1           ;IGNORE PRECEDING BYTES
        MOVE.B  #0,27(A1)       ;SET 0 DELIMITER
        XPEL                    ;DISPLAY INQUIRY MESSAGE
        MOVEM.L (A7)+,D0-A6
        RTS
*
************************************
*
* DATA AREA
*
* SCSI COMMANDS
*
INQUIRY DC.B    6,$12,0,0,0,$0,0        ;INQUIRY COMMAND
MODSENS DC.B    6,$1A,0,$3F,0,77,0      ;MODE SENSE COMMAND
```

EXAMPLE 1 continued

```
*
* MESSAGES
*
STRTMSG DC.B    10,13,'DEVICE MODE REPORT, TARGET ID = ',0
MSG00   DC.B    10,13,'DEVICE TYPE = ',0
MSG01   DC.B    10,10,13,'BLOCK DESCRIPTOR',0
MSG02   DC.B    10,13,'  DENSITY ..............................',0
MSG03   DC.B    10,13,'  NUMBER OF ............................',0
MSG04   DC.B    10,13,'  BLOCK ................................',0
MSG05   DC.B    10,13,'ERROR RECOVERY PARAMETERS',0
MSG06   DC.B    10,13,'  FLAG .................................',0
MSG07   DC.B    10,13,'  RETRY ................................',0
MSG08   DC.B    10,13,'  CORRECTION SPAN.......................',0
MSG09   DC.B    10,13,'  HEAD OFFSET COUNT.....................',0
MSG10   DC.B    10,13,'  DATA STROBE OFFSET COUNT..............',0
MSG11   DC.B    10,13,'  RECOVERY TIME LIMIT...................',0
MSG12   DC.B    10,13,'DISCONNECT/RECONNECT PARAMETERS',0
MSG13   DC.B    10,13,'  BUFFER FULL RATIO.....................',0
MSG14   DC.B    10,13,'  BUS INACTIVITY LIMIT..................',0
MSG15   DC.B    10,13,'  DISCONNECT TIME LIMIT.................',0
MSG16   DC.B    10,13,'  CONNECT TIME LIMIT....................',0
MSG17   DC.B    10,13,'DIRECT ACCESS DEVICE PARAMETERS',0
MSG18   DC.B    10,13,'  TRACKS PER ZONE.......................',0
MSG19   DC.B    10,13,'  ALTERNATE SECTORS PER ZONE............',0
MSG20   DC.B    10,13,'  ALTERNATE TRACKS PER ZONE.............',0
MSG21   DC.B    10,13,'  ALTERNATE TRACKS PER VOLUME...........',0
MSG22   DC.B    10,13,'  SECTORS PER TRACK.....................',0
MSG23   DC.B    10,13,'  DATA BYTES PER SECTOR.................',0
MSG24   DC.B    10,13,'  INTERLEAVE VALUE......................',0
MSG25   DC.B    10,13,'  TRACK SKEW............................',0
MSG26   DC.B    10,13,'  CYLINDER SKEW.........................',0
MSG27   DC.B    10,13,'  FLAG BYTE.............................',0
MSG28   DC.B    10,13,'RIGID DRIVE GEOMETRY PARAMETERS',0
MSG29   DC.B    10,13,'  MAXIMUM NUMBER OF CYLINDERS...........',0
MSG30   DC.B    10,13,'  MAXIMUM NUMBER OF HEADS...............',0
MSG31   DC.B    10,13,'  STARTING CYLINDER-WRITE PRECOMP.......',0
MSG32   DC.B    10,13,'  STARTING CYLINDER-REDUCED WR CURRENT..',0
MSG33   DC.B    10,13,'  DRIVE STEP RATE.......................',0
MSG34   DC.B    10,13,'  LANDING ZONE CYLINDER.................',0
KEYMSG  DC.B    10,13,'  .....................................',0
DMSG0   DC.B    'MAGNETIC DISK',0
DMSG1   DC.B    'STREAMER TAPE',0
DMSGU   DC.B    'UNKNOWN DEVICE',0
*
        EVEN
*
BUFFER  DS.B    40                      ;STRING BUFFER
*
*
        END
```

PROGRAM RUNNING EXAMPLE:

```
>REPORT 3

DEVICE MODE REPORT, TARGET ID = 3
MICROP  1370              A05
DEVICE TYPE = MAGNETIC DISK
BLOCK DESCRIPTOR
      DENSITY CODE.....0
      NUMBER OF BLOCKS.....276896
      BLOCK LENGTH.....512
ERROR RECOVERY PARAMETERS
      FLAG BYTE.....%00100100
      RETRY COUNT.....10
      CORRECTION SPAN.....11
      HEAD OFFSET COUNT.....0
      DATA STROBE OFFSET COUNT.....0
      RECOVERY TIME LIMIT.....0
DISCONNECT/RECONNECT PARAMETERS
      BUFFER FULL RATIO.....0
      BUS INACTIVITY LIMIT.....5
      DISCONNECT TIME LIMIT.....0
      CONNECT TIME LIMIT.....0
DIRECT ACCESS DEVICE PARAMETERS
      TRACKS PER ZONE.....1
      ALTERNATE SECTORS PER ZONE.....1
      ALTERNATE TRACKS PER ZONE.....0
      ALTERNATE TRACKS PER VOLUME.....24
      SECTORS PER TRACK.....35
      DATA BYTES PER SECTOR.....512
      INTERLEAVE VALUE.....0
      TRACK SKEW.....0
      CYLINDER SKEW.....0
      FLAG BYTE.....%01000000
RIGID DRIVE GEOMETRY PARAMETERS
      MAXIMUM NUMBER OF CYLINDERS.....262152
      MAXIMUM NUMBER OF HEADS.....0
      STARTING CYLINDER - WRITE PRECOMP.....0
      STARTING CYLINDER - REDUCED WR CURRENT...0
      DRIVE STEP RATE.....0
      LANDING ZONE CYLINDER.....0
```

EXAMPLE 2: BUS_CHK

```
*****************************************************************
***                                                           ***
***              ISCSI-1 PROGRAMMING EXAMPLE                   ***
***                                                           ***
***     CHECK TARGETS FOR EXISTENCE AND DEVICE TYPE           ***
***                                                           ***
*****************************************************************

INCLUDE ISCSI:IN

OPT     ALT,NOWARN

 BUS_CHK XPCL                          ;CR/LF
         XPCL                          ;CR/LF
         XPMC    MSTART                ;START MESSAGE
         MOVEA.L #ISCSI+CR_1,A0        ;CMDRAM
@001     TST.W   (A0)                  ;BUSY?
         BPL.S   @001                  ;Y, WAIT
         MOVE.W  #$45,(A0)             ;COMMAND
@002     TST.W   (A0)                  ;BUSY?
         BPL.S   @002                  ;Y, WAIT
*
         MOVEA.L 4(A0),A3              ;GET POINTER
         ADDA.L  #ISCSI,A3             ;ADD BOARD BASE
         MOVE.L  #0,D1                 ;COUNTER
@100     MOVE.B  (A3)+,D2              ;GET DATA
         XPCL                          ;CR/LF
         XCBD                          ;BIN TO DEC ASCII
         XPEL                          ;WRITE ID
         CMP.B   #$FF,D2               ;OWN?
         BNE.S   @101                  ;N
         XPMC    MOWNID                ;Y
         BRA.S   @300
@101     CMP.B   #$80,D2               ;UNKNOWN DEVICE?
         BNE.S   @102                  ;N
         XPMC    MUNKN                 ;Y
         BRA.S   @300
@102     CMP.B   #$7F,D2               ;NON EXISTING?
         BNE.S   @103                  ;N
         XPMC    MNEXIS                ;Y
         BRA.S   @300
@103     CMP.B   #$03,D2               ;PROCESSOR DEVICE?
         BNE.S   @105                  ;N
         XPMC    XPROC                 ;Y
         MOVE.L  D1,D5
         BSR     INQU
         BRA.S   @300
@105     CMP.B   #0,D2                 ;MAGNETIC DISK?
         BNE.S   @104                  ;N
         XPMC    MMAGN                 ;Y
         MOVE.L  D1,D5
         BSR     INQU
```

EXAMPLE 2 continued

```
        BSR     CAPACIT
        BRA.S   @300
@104    XPMC    MTAPE           ;SHOULD BE STREAMER TAPE
@300    ADDQ.L  #1,D2           ;NEXT ID
        CMP.B   #8,D1           ;DONE?
        BLT.S   @100            ;N, CONTINUE
*
        XPCL                    ;CR/LF
        XEXT                    ;EXIT TO PDOS
*
INQU    MOVEM.L D0-A6,-(A7)
        LEA.L   ISCSI+CR_1,A0           ;CMDRAM
        LEA.L   ISCSI+BUF_1,A1
        LEA.L   INQUIRY(PC),A2          ;INQUIRY COMMAND
        MOVE.W  D5,D1
        MOVE.W  #50,D2                  ;TRANSFER COUNTER
        BSR     INIT
        BSR     WAIT
*
        LEA.L   ISCSI+BUF_2,A1
        MOVE.B  #0,(A1,D2.W)
        ADD.L   #8,A1
        MOVE.B  #0,27(A1)
        XPMC    BLANK
        XPEL
        BSR     CLEAR
*
        MOVEM.L (A7)+,D0-A6
        RTS
*
*
CAPACIT MOVEM.L D0-A6,-(A7)
        XPMC    MSG03
        LEA.L   ISCSI+CR_1,A0           ;CMDRAM
        LEA.L   ISCSI+BUF_1,A1
        LEA.L   CAPAC(PC),A2            ;CAPACITY COMMAND
Strike any key...

        MOVE.W  D5,D1                   ;TARGET ID
        MOVE.W  #8,D2                   ;TRANSFER COUNTER
        BSR     INIT1
        BSR     WAIT1
*
        LEA.L   ISCSI+BUF_2,A0
        LEA.L   BUFFER(PC),A1
        MOVE.L  (A0)+,D3
        MOVE.L  (A0),D4
        MOVE.L  D4,D5
        DIVU    #1000,D3
        ANDI.L  #$0FFFF,D3
        MULU    D4,D3
        DIVU    #1000,D3
```

EXAMPLE 2 continued

```
          ANDI.L   #$0FFFF,D3
          MOVE.L   D3,D1
          XCBX
XPEL
XPMC      MSG02
MOVE.L    D5,D1
          XPMC     MSG01
          LEA.L    BUFF_2(PC),A1
          XCBX
          XPEL
*
          MOVEM.L  (A7)+,D0 A6
          RTS
*
*
INIT     MOVE.L   #7,D0
@001     MOVE.B   (A2)+,(A1)+
          SUBQ.L   #1,D0
          BNE.S    @001
          MOVE.W   D1,12(A0)
          MOVE.W   D2,2(A0)
          MOVE.L   #BUF 1,4(A0)
          MOVE.L   #BUF_2,8(A0)
          RTS
*
*
WAIT1    MOVE.W   #$60,(A0)
@001     TST.W    (A0)
          BPL.S    @001
          RTS
*
*
CAPAC    DC.B     10,$25,0,0,0,0,0,0,0,0
*
          EVEN
*
BUFFER   DS.B     40
BUFF_2   DS.B     40
*
MSG01    DC.B     10,13,'          BLOCKSIZE = ',0
MSG04    DC.B     ' BYTES',0
MSG02    DC.B     ' MBYTE',0
MSG03    DC.B     10,13,'          CAPACITY = ',0
*
          EVEN
```

EXAMPLE_2_continued

```
*
CLEAR   LEA.L   ISCSI+BUF_2,A4
        MOVE.L  #10,D7
@001    CLR.L   (A4)+
        SUBQ.L  #1,D7
        BNE.S   @001
        RTS
INIT    MOVE.L  #7,D0
@001    MOVE.B  (A2)+,(A1)+
        SUBQ.L  #1,D0
        BNE.S   @001
        MOVE.W  D1,12(A0)
        MOVE.W  D2,2(A0)
        MOVE.L  #BUF_1,4(A0)
        MOVE.L  #BUF_2,8(A0)
        RTS
*
WAIT    MOVE.W  #$60,(A0)
@001    TST.W   (A0)
        BPL.S   @001
        RTS
*
TURDY   DC.B    6,0,0,0,0,0,0
INQUIRY DC.B    6,$12,0,0,0,50,0
BLANK   DC.B    10,13,'                ',0
*
*
NOWNID  DC.B    '      ISCSI 1',0
NUNKN   DC.B    '      UNKNOWN DEVICE TYPE',0
NMAGN   DC.B    '      MAGNETIC DISK',0
NNEXIS  DC.B    '             ',0
NTAPE   DC.B    '      STREAMER TAPE',0
NPROC   DC.B    '      PROCESSOR DEVICE',0
MSTART  DC.B    'ID    DEVICE',10,13
        DC.B    '       ----------------   ',0
*
        EVEN
        END
```

PROGRAM RUNNING EXAMPLE:

```
/BUS CHK

ID     DEVICE
----------------------------
0      -----
1      -----
2      MAGNETIC DISK
           QUANTUM Q280Z022003-REV00
           CAPACITY = 79 MBYTE
           BLOCKSIZE = 512
3      MAGNETIC DISK
           MICROP  1370              A05
           CAPACITY = 141 MBYTE
           BLOCKSIZE = 512
4      -----
5
6
7      ISCSI 1
```

EXAMPLE 3: IDDUMP

```
*****************************************************************
***                                                           ***
***                  ISCSI-1 PROGRAMMING EXAMPLE              ***
***                                                           ***
***                   DISK DUMP UTILITY                       ***
***                                                           ***
*****************************************************************
*
        INCLUDE ISCSI:IN
*
        OPT     ALT,NOWARN
*
IDDUMP  XCLS                        ;CLEAR SCREEN
        XFMC    SMSG                ;DISPLAY MESSAGE
        XFMC    IMSG
        LEA.L   PBUFF(PC),A1
        XGLB
        BEQ     EXIT
        XCDB
        BLT     EXIT
        MOVE.L  D1,D7
        LSL.W   #1,D7
        LSL.W   #8,D7
        OR.W    #$20,D7
        XFCL
        XFMC    IMSG
        LEA.L   PBUFF(PC),A1
        XGLB
        BEQ     @800
        XCDB
        BLT     EXIT
        MOVE.L  D1,D6
        BRA.S   @801
@800    MOVE.L  #0,D6
        XFMC    NULL
@801    XFCL
        MOVE.L  #-1,D5
START   XFMC    PMSG
        LEA.L   PBUFF(PC),A1            ;GET SECTOR #
        XGLB
        BEQ     @000
        XCDB                            ;CONVERT TO BINARY
        BLT     L002
        MOVE.L  D1,D5                   ;SAVE IT FOR LATER USE
        XFCL
        BRA.S   @700
@000    ADD.L   #1,D5
        MOVE.L  D5,D1
        LEA.L   PBUFF(PC),A1
        XCBX
        XFEL
        XFCL
```

EXAMPLE 3 continued

```
          @700      CMP.W    #$0E20,D7
          BNE.S     @300
          MOVEA.L   #ISCSI+CR_1,A0
@010      MOVE.W    #1,2(A0)
          MOVE.L    D5,4(A0)
          MOVE.L    #$4000,8(A0)
          MOVE.W    D6,12(A0)
          MOVE.W    D7,(A0)
@020      TST.W     (A0)
          BPL.S     @020
*
          TST.B     1(A0)
          BNE       ERROR
*
          MOVEA.L   #ISCSI+CR_1,A0
          MOVEA.L   4(A0),A0
          ADDA.L    #ISCSI,A0
          MOVE.L    #$10,D4
          BSR.L     MDUMP
          BRA.L     START
*
@300      MOVEM.L   D5/D7,-(A7)
          LEA.L     $A02301,A3
          MOVE.B    #$06,(A3)+
          OR.L      #$08000000,D5
          MOVE.L    D5,(A3)+
          MOVE.W    #$0100,(A3)+
          MOVEA.L   #ISCSI+CR_1,A0
          MOVE.L    #$2301,4(A0)
          MOVE.L    #$2400,8(A0)
          MOVE.W    #$200,2(A0)
          LSR.W     #1,D7
          LSR.W     #8,D7
          MOVE.W    D7,12(A0)
          MOVEM.L   (A7)+,D5/D7
          MOVE.W    #$0060,(A0)
*
@301      TST.W     (A0)
          BPL.S     @301
*
          TST.B     1(A0)
          BNE       ERROR
          MOVEA.L   #ISCSI+CR_1,A0
          MOVEA.L   8(A0),A0
          ADDA.L    #ISCSI,A0
          MOVE.L    #$10,D4
          BSR.S     MDUMP
          XPCL
          XPMC      KMSG
          XGCP
          XPCL
          MOVE.L    #$10,D4
          BSR.S     MDUMP
          BRA.L     START
```

EXAMPLE 3 continued

```
*
EXIT    XEXT
*
L002    LEA.L   PBUFF(PC),A1
        CMP.B   #'?',(A1)
        BEQ.S   @001
        CMP.B   #'.',(A1)
        BEQ     IDDUMP
        BRA     START
@001    XPCL
        MOVEM.L D1/D7/A1,-(A7)
        LEA.L   PBUFF(PC),A1
        XPMC    IMSG
        LSR.W   #1,D7
        LSR.W   #8,D7
        MOVE.L  D7,D1
        XCBX
        XPEL
        XPMC    LMSG
        MOVE.L  D6,D1
        XCBX
        XPEL
        XPMC    PMSG
        MOVE.L  D5,D1
        XCBX
        XPEL
        XPCL
        MOVEM.L (A7)+,D1/D7/A1
        BRA     START
*
ERROR   XPMC    EMSG
        BRA.S   EXIT
*
**********************************
*
* MEMORY DUMP
*
*       IN : A0.L = MEMORY ADDRESS
*       OUT: 256 BYTES ARE DISPLAYED
*            A0 IS UPDATED
*
MDUMP   MOVEM.L D0-D3/A1,-(A7)
        LEA.L   WBUFF(PC),A1
        MOVE.L  D4,D1
@001    MOVE.W  #$0A0D,(A1)+
        LEA.L   ABUFF(PC),A2
        MOVE.L  #4,D2
        MOVE.L  A0,-(A7)
        BSR.L   HEXAS
        CLR.L   (A7)+
        MOVE.W  #': ',(A1)+
```

EXAMPLE 3 continued

```
@002    MOVE.L  (A0),-(A7)
        MOVE.L  (A0),(A2)+
        BSR.L   HEXAS
        CLR.L   (A7)+
        MOVE.W  #'  ',(A1)+
        ADDA.L  #4,A0
        SUBQ.L  #1,D2
        BNE.S   @002
*
        MOVE.L  #16,D2
        LEA.L   ABUFF(PC),A2
@003    MOVE.B  (A2)+,D3
        CMP.B   #$20,D3
        BLT.S   @004
        CMP.B   #$7F,D3
        BGT.S   @004
        MOVE.B  D3,(A1)+
        BRA.S   @005
@004    MOVE.B  #'.',(A1)+
@005    SUBQ.L  #1,D2
        BNE.S   @003
*
        SUBQ.L  #1,D1
        BNE.S   @001
*
        MOVE.L  #$0A0D0000,(A1)+
        LEA.L   WBUFF(PC),A1
        XFEL
        MOVEM.L (A7)+,D0-D3/A1
        RTS
*
************************************
* TRANSFORM HEX TO ASCII
*
*       IN : 4(A7) = HEX NUMBER
*            A1.L = BUFFER
*
HEXAS   MOVEM.L D0-D2,-(A7)
        MOVE.L  16(A7),D0               ;GET HEX NUMBER
        MOVE.L  #8,D2
@001    ROL.L   #4,D0
        MOVE.B  D0,D1
        ANDI.L  #$0F,D1
        CMP.B   #$0A,D1
        BLT.S   @002
        ADD.B   #$37,D1
        BRA.S   @003
@002    ADD.B   #$30,D1
@003    MOVE.B  D1,(A1)+
        SUBQ.L  #1,D2
        BNE.S   @001
        MOVEM.L (A7)+,D0-D2
        RTS
```

EXAMPLE 3 continued

```
*
MSG01   DC.B    10,13,'SECTOR DUMP...',10,13,0
PMSG    DC.B    10,13,'BLOCK NUMBER : ',0
SMSG    DC.B    10,13,'TSC5I-1 DISK DUMP UTILITY!'
        DC.B    10,13,'====================',10,13,0
IMSG    DC.B    10,13,10,'TARGET ID     : ',0
LMSG    DC.B    10,13,'UNIT NUMBER   : ',0
EMSG    DC.B    10,13,10,'***** READ ERROR *****',10,13,0
KMSG    DC.B    'strike any key to continue...',0
NULL    DC.B    '0',0
*
        EVEN
*
WBUFF   DS.B    4000
ABUFF   DS.B    400
PBUFF   DS.L    20
*
        END
```

PROGRAM RUNNING EXAMPLE:

```
>IDDUMP

ISCSI-1 DISK DUMP UTILITY!
==========**********

TARGET ID    : 3

UNIT NUMBER  : 0

BLOCK NUMBER : 9

00A02400: 41534D00  00000000  00000000  8004024C  ASM.............L
00A02410: 00000000  00000016  1034AB06  1034AB06  .........4...4..
00A02420: 53554E50  41524D00  53520001  028001DD  SUNPARM.SR......
00A02430: 00000016  00160083  0A16AB57  0F16AB57  ............W...W
00A02440: 424C4F43  4B494E47  4F420065  20000398  BLOCKINGOB.e ...
00A02450: 00000001  0001004F  101FAB21  130DAD49  .......O...!...I
00A02460: 424C4F43  4B494E47  53520001  02000125  BLOCKINGSR.....%
00A02470: 0000000B  000B0086  0E07AB1D  1238AD49  .............8.I
00A02480: 42554646  57520000  53520002  02040486  BUFFWR..SR......
00A02490: 00000009  000900C9  0B17AB16  0E26AD47  .............&.G
00A024A0: 43484149  4E000000  4F420065  2004039D  CHAIN...OB.e ...
00A024B0: 00000000  00000066  1020AB21  1020AB21  .......f. .!. .!
00A024C0: 43484149  4E000000  53520001  020402CA  CHAIN...SR......
00A024D0: 00000006  000400A4  1217AB07  0B20AB15  ............. ..
00A024E0: 43484149  4E4D0000  4F420065  2000039E  CHAINM..OB.e ...
00A024F0: 0000000D  000D0007  1020AB21  0E1BAD47  ......... .!...G
00A02500: 43484149  4E4D0000  53520001  0200029E  CHAINM..SR......
00A02510: 00000013  001200F5  1210AB07  0F30AB23  .............0.#
00A02520: 434D4452  414D0000  53520032  0204006D  CMDRAM..SR.2...m
00A02530: 00000006  00050018  100EAAEF  0A13AB0B  ................
00A02540: 434D4453  45540000  53520032  020002E1  CMDSET..SR.2....
00A02550: 00000013  00130006  0C02AD08  0B20AD46  ............. .F
00A02560: 434D4454  424C0000  494E0003  028400BD  CMDTBL..IN......
00A02570: 00000026  001B00BB  0B11AADA  1008AB57  ...&...........W
00A02580: 434F4D50  4C455400  53520002  02000074  COMPLET.SR.....t
00A02590: 00000009  000900C9  1106AAFF  1108AD54  ...............T
00A025A0: 43544C53  54415400  53520001  020001BF  CTLSTAT.SR......
00A025B0: 00000011  001100F0  0E33AB28  0F13AB28  .........3.(...(
00A025C0: 44454255  47000000  4F420065  200404C3  DEBUG...OB.e ...
00A025D0: 0000001C  001C0081  1101AB21  130BAD49  ...........!...I
00A025E0: 44454654  41420000  53520032  028002B0  DEFTAB..SR.2....
00A025F0: 00000016  001500FB  111FAB1C  1012AB57  ...............W

BLOCK NUMBER : 9
TARGET ID    : 3
UNIT NUMBER  : 0
BLOCK NUMBER : 9

BLOCK NUMBER : .
```

```
ISCSI-1 DISK DUMP UTILITY
=========================

TARGET ID     : 2

UNIT NUMBER   : 0

BLOCK NUMBER : 0

00A02400: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02410: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02420: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02430: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02440: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02450: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02460: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02470: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02480: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02490: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024A0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024B0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024C0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024D0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024E0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A024F0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02500: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02510: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02520: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02530: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02540: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02550: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02560: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02570: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02580: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A02590: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025A0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025B0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025C0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025D0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025E0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................
00A025F0: E5E5E5E5  E5E5E5E5  E5E5E5E5  E5E5E5E5  ................

BLOCK NUMBER : <ESC> (QUITS TO SYSTEM)
```

EXAMPLE 4: CONNECT

```
*************************************************************
***                                                       ***
***              ISCSI-1 PROGRAMMING EXAMPLE              ***
***                                                       ***
*************************************************************
***                                                       ***
***     CONNECT TWO ISCSI-1 BOARDS TOGETHER. ONE AS AN    ***
***     INITIATOR, THE OTHER AS A TARGET.                 ***
***                                                       ***
***     THE FIRST ISCSI-1 SHOULD BE THE INITIATOR         ***
***     AND HAS THE BASE ADDRESS $A00000.                 ***
***     THE SECOND ISCSI-1 HAS TO BE THE TARGET AND HAS   ***
***     THE BASE ADDRESS $A80000.                         ***
***                                                       ***
***     WHEN INITIALIZATION IS DONE WE CAN USE THE        ***
***     OTHER UTILITIES TO ACCESS THE TARGET BOARD.       ***
***                                                       ***
*************************************************************
*
        OPT     ALT
*
* FIRST WE INITIALIZE THE INITIATOR BOARD
*
INIT    LEA.L   $A00000,A0              ;GET BASE ADDRESS
*
* SET ADDRESS OFFSET
*
        MOVE.W  #$0003,D3               ;SETOFFS COMMAND
        SWAP    D3                      ;COMMAND IN HIGH WORD
        MOVE.W  #0,D3                   ;FIRST PARAMETER
        MOVE.L  A0,D2                   ;BASE ADDRESS OFFSET
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
*
* GET ISCSI-1 STATUS
*
        MOVE.W  #$0044,D3               ;CTLSTAT COMMAND
        SWAP    D3                      ;COMMAND IN HIGH WORD
        MOVE.W  #$0,D3                  ;FIRST PARAMETER
        MOVEQ.L #0,D2                   ;SECOND PARAMETER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
        MOVEA.L D2,A1                   ;GET PARAMETER POINTER
```

EXAMPLE 4 continued

```
        MOVE.B  (A1),D4                 ;CONTROLLER ID
        TST.B   7(A1)                   ;TEST TARGET MODE
        BEQ.S   @001                    ;DISABLED, CONTINUE
*
* DISABLE TARGET MODE
*
        MOVE.L  #$00620000,D3           ;TARGMOD COMMAND
        MOVEQ.L #0,D2                   ;SECOND PARAMETER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
*
* CHECK AND SET ID IF NECESSARY
*
@001    CMP.B   #7,D4                   ;ID = 7?
        BEQ.S   @002                    ;YES, CONTINUE
        MOVE.L  #$07000000,$2300(A0)    ;SET PARAMETERS FOR
                                         MAININIT
        MOVE.L  #$000D0000,D3           ;MAININIT COMMAND
        MOVE.L  A0,D2                   ;BASE ADDRESS
        ADD.L   #$2300,D3               ;PARAMETER POINTER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
*
* NOW WE INITIALZE THE SECOND CONTROLLER WHICH IS THE TARGET
*
@002    LEA.L   $A80000,A0              ;GET BASE ADDRESS
*
* SET ADDRESS OFFSET
*
        MOVE.W  #$0003,D3               ;SETOFFS COMMAND
        SWAP    D3                      ;COMMAND IN HIGH WORD
        MOVE.W  #0,D3                   ;FIRST PARAMETER
        MOVE.L  A0,D2                   ;BASE ADDRESS OFFSET
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
```

EXAMPLE 4 continued

```
*
* GET ISCSI-1 STATUS
*
        MOVE.W  #$0044,D3               ;CTLSTAT COMMAND
        SWAP    D3                      ;COMMAND IN HIGH WORD
        MOVE.W  #$0,D3                  ;FIRST PARAMETER
        MOVEQ.L #0,D2                   ;SECOND PARAMETER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI 1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.L   ERROR                   ;YES
        MOVEA.L D2,A1                   ;GET PARAMETER POINTER
        MOVE.B  (A1),D4                 ;CONTROLLER ID
        TST.B   7(A1)                   ;TEST TARGET MODE
        BNE.S   @003                    ;ENABLED, CONTINUE
*
* ENABLE TARGET MODE
*
        MOVE.L  #$00620000,D3           ;TARGMOD COMMAND
        MOVEQ.L #0,D2                   ;SECOND PARAMETER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI 1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.S   ERROR                   ;YES
*
* CHECK AND SET ID IF NECESSARY
*
@003    CMP.B   #6,D4                   ;ID = 6?
        BEQ.S   @002                    ;YES, CONTINUE
        MOVE.L  #$06000000,$2300(A0)    ;SET PARAMETERS FOR
                                         MAININIT
        MOVE.L  #$000D0000,D3           ;MAININIT COMMAND
        MOVE.L  A0,D2                   ;BASE ADDRESS
        ADD.L   #$2300,D3               ;PARAMETER POINTER
        MOVEQ.L #0,D1                   ;THIRD PARAMETER
        MOVEQ.L #0,D0                   ;FOURTH PARAMETER
        BSR.L   COMMAND                 ;ISSUE ISCSI-1 COMMAND
        SWAP    D3                      ;GET ERROR CODE
        TST.B   D3                      ;ERROR?
        BNE.S   ERROR                   ;YES
        XEXT                            ;NO, EXIT TO PDOS
```

EXAMPLE 4 continued

```
*
* ERROR HANDLING
*
ERROR   XPMC    EMSG                            ;PRINT MESSAGE
        XEXT                                    ;EXIT TO PDOS
*
EMSG    DC.B    10,13,'ISCSI-1 ERROR',10,13,0
*
        EVEN
*
* SUBROUTINE
*       ISSUE ISCSI-1 COMMAND
*
COMMAND MOVE.L  A0,-(A7)                        ;SAVE BASE ADDRESS
        ADDA.L  #$2110,A0                       ;GET COMMAND RAM TOP
        MOVEM.L D0-D3,-(A0)                     ;SETUP CMDRAM
@001    TST.W   (A0)                            ;WAIT UNTIL READY
        BPL.S   @001
        MOVEM.L (A0)+,D0-D3                     ;GET RETURN VALUES
        MOVE.L  (A7)+,A0                        ;RESTORE BASE ADDRESS
        RTS                                     ;RETURN TO CALLER
*
*
        END
```

EXAMPLE 5: GET1BM

```
*
*-------------------------------------
* READ 1BM FLOPPIES ON AN ISCSI 1
* UNDER PDOS
*-------------------------------------
*
        OPT     ALT,NOWARN
*
ISCSI   EQU     $FCA00000       ;ISCSI-1 BASE ADDRESS
CMD_1   EQU     ISCSI+$2100     ;COMMAND RAM #1
IOB_1   EQU     ISCSI+$2300     ;I/O BUFFER #1
HBUF    EQU     ISCSI+$4000     ;HASHING BUFFER BASE
*
FLUSH   EQU     $0056           ;FLUSH ALL BUFFERS COMMAND
SUNPARM EQU     $0E0A           ;SET UNIT PARAMETER COMMAND
RDBLOCK EQU     $0E20           ;READ BLOCK/SECTOR COMMAND
CLRHASH EQU     $005F           ;CLEAR HASHING TABLES COMMAND
*
FLUN    EQU     $0003           ;FLOPY DISK UNIT #
SSIZE   EQU     $200            ;SECTOR SIZE
*
*-------------------------------------
* GET THE SECTOR COUNT FROM THE
* COMMAND LINE
*-------------------------------------
*
GETCNT  XGNP                    ;GET PARAMETER
        BLE.S   @001            ;NO PARAMETER
        XCDB                    ;CONVERT TO BINARY NUMBER
        BLT.S   @001            ;NO NUMBER!
        MOVE.L  D1,D5           ;SAVE COUNT
        XGNP                    ;GET START SECTOR
        BLE.S   @001            ;NO PARAMETER
        XCDB                    ;CONVERT TO BINARY NUMBER
        BLT.S   @001            ;NO NUMBER
        BRA.S   DOFLUSH         ;GO...
@001    XPMC    USMSG           ;USAGE!
        XEXT                    ;EXIT TO PDOS
*
*-------------------------------------
* WE FLUSH ALL HASHING BUFFERS
* BECAUSE WE DESTROY THEM
*-------------------------------------
*
DOFLUSH LEA.L   CMD_1,A0        ;CMDRAM POINTER
        MOVE.W  #FLUSH,(A0)     ;COMMAND
@001    TST.W   (A0)            ;WAIT FOR ISCSI
        BPL.S   @001
        TST.B   1(A0)           ;ERROR??
        BNE.L   ERROR           ;YES
```

```
*
*---- --  ------------------------
* WE INIT FLOPPY #3 FOR IBM
* FORMAT
*--------------------------
*
INIT    LEA.L   IOB_1,A1        ;I/O BUFFER POINTER
        LEA.L   INITAB(PC),A2   ;FLOPPY DEFINITIONS
        MOVE.L  #19,D0          ;TABLE LENGTH
@001    MOVE.B  (A2)+,(A1)+     ;COPY TABLE TO ISCSI-1 DPR
        SUBQ.L  #1,D0
        BNE.S   @001
*
* INIT COMMAND RAM
*
        MOVE.L  #IOB_1,4(A0)    ;TABLE POINTER
        MOVE.W  #FLUN,12(A0)    ;FLOPPY LUN
        MOVE.W  #SUNPARM,(A0)   ;COMMAND
@002    TST.W   (A0)            ;WAIT FOR ISCSI
        BPL.S   @002
        TST.B   1(A0)           ;ERROR??
        BNE.L   ERROR           ;YES
*
*-------------------------------
* NOW WE READ 32 BLOCKS FROM
* THE IBM FLOPPY
*------- ----  ------- --------
*
FREAD   XPCL                    ;CR/LF
        MOVE.L  #HBUF,D7        ;GET BUFFER BASE
@100    MOVE.W  #1,2(A0)        ;SECTOR COUNT
        MOVE.L  D1,4(A0)        ;SECTOR NUMBER
        MOVE.L  D7,8(A0)        ;DATA BUFFER
        MOVE.W  #FLUN,12(A0)    ;FLOPPY LUN
        MOVE.W  #RDBLOCK,(A0)   ;COMMAND
@001    TST.W   (A0)            ;WAIT FOR ISCSI
        BPL.S   @001
        TST.B   1(A0)           ;ERROR??
        BNE.L   ERROR           ;YES
        XCBD
        XPLC
        XPCL
        ADDQ.L  #1,D1           ;NEXT SECTOR
        ADD.L   #$0128,D7       ;POINTS TO NEXT BUFFER
        SUBQ.L  #1,D5           ;DECREMENT COUNTER
        BNE.S   @100
```

```
*
*-----------------------
* WE HAVE TO CLEAR THE HASHING
* TABLES
*-----------------------------
*
CLRTAB  MOVE.W  #CLRHASH,(A0)    ;COMMAND
@001    TST.W   (A0)             ;WAIT FOR ISCSI
        BPL.S   @001
        TST.W   1(A0)            ;ERROR??
        BNE.L   ERROR            ;YES
        BSR.L   GETPDOS          ;RESTORE PDOS FLOPPIES
*
*-------------------------------
* PRINT MESSAGE AND EXIT
*-------------------------------
*
        XPMC    GOODMSG          ;SUCCESS MESSAGE
        XPMC    ADDRMSG          ;BUFFER ADDRESSES
        MOVE.L  #HIBUF,D1        ;START ADDRESS
        XCBH                     ;TO HEX
        XPLC
        MOVE.W  #'T ',D0
        XPCC
        MOVE.W  #' 0',D0
        XPCC
        MOVE.L  D7,D1            ;END ADDRESS
        XCBH                     ;TO HEX
        XPLC
        XPCL
        XEXT                     ;EXIT TO PDOS
*
*-----------------------------
* ISCSI 1 ERRORS COME HERE
*-----------------------------
*
ERROR   XPMC    ERMSG            ;PRINT ERROR MESSAGE
        MOVE.L  (A0),D1          ;GET ERROR CODE
        ANDI.L  #$FFFFF,D1       ;MASK UNNECESARY
        XCBH                     ;CONVERT TO HEX
        XPLC                     ;PRINT ERROR CODE
        XPCL                     ;CR/LF
        MOVE.W  #CLRHASH,(A0)    ;CLEAR HASHING BUFFERS COMMAND
@001    TST.W   (A0)             ;WAIT FOR ISCSI
        BPL.S   @001
        BSR.S   GETPDOS          ;RESTORE PDOS FLOPPIES
        XEXT                     ;EXIT TO PDOS
```

```
*
*..  ..     --------------------
* RESTORE PDOS FLOPPIES
*  ---     --------------------
*
GETPDOS LEA.L   IOB_1,A1          ;I/O BUFFER POINTER
        LEA.L   PDOSTAB(PC),A2    ;FLOPPY DEFINITIONS
        MOVE.L  #10,D0            ;TABLE LENGTH
@001    MOVE.B  (A2)+,(A1)+       ;COPY TABLE TO ISCSI 1 DPR
        SUBQ.L  #1,D0
        BNE.S   @001
        MOVE.L  #IOB_1,4(A0)      ;TABLE POINTER
        MOVE.W  #FLUN,12(A0)      ;FLOPPY LUN
        MOVE.W  #SUNPARM,(A0)     ;COMMAND
@002    TST.W   (A0)              ;WAIT FOR ISCSI
        BPL.S   @002
        RTS                       ;RETURN TO CALLER
*
*       ------ .. ----------  ---
* MESSAGES AND DATA DEFINITIONS
*        ------- ---------- ----
*
USMSG   DC.B    10,13,'USAGE : '
        DC.B    10,13,'<SECTOR COUNT>,<START SECTOR>',10,13,0
GOODMSG DC.B    10,13,'SUCCESSFULLY COMPLETED!',10,13,0
ADDRMSG DC.B    10,13,'DATA IN BUFFER FROM ',0
ERMSG   DC.B    10,13,'ISCSI 1 ERROR >',0
        EVEN
INITAB  DC.B    $02     ;NUMBER OF HEADS ..................2
        DC.B    $28     ;NUMBER OF CYLINDERS..............40
        DC.B    $12     ;NUMBER OF SECTORS PER CYLINDER..18
        DC.B    $00     ;SECTOR OFFSET.....................0
        DC.B    $02     ;BYTES PER SECTOR................512
        DC.B    $01     ;DISK DENSITY................DOUBLE
        DC.B    $E5     ;FORMAT PATTERN..................$E5
        DC.B    $08     ;SECTOR INTERLEAVE TABLE.......USER
        DC.B    $01     ;LOWEST PHYSICAL SECTOR..........0
        DC.B    $00     ;STEP RATE.....................6MS
*
        DC.B    $01     ;START OF SECTOR INTERLEAVE TABLE
        DC.B    $02
        DC.B    $03
        DC.B    $04
        DC.B    $05
        DC.B    $06
        DC.B    $07
        DC.B    $08
        DC.B    $09
PDOSTAB DC.B    2,80,32,0,1,1,$E5,5,1,0         ;PDOS FLOPPY
        EVEN
*
        END
```

APPENDIX F

SAMPLE OF C ROUTINES

```
/*
  ************************************
  *                                  *
  *     C header file for the        *
  *     FORCE ISCSI-1 board          *
  *                                  *
  ************************************
*/

/*---------------------- address definitions ---------- --- -- -----*/

#define ISCSI1     0xfca00000     /* base address of the ISCSI 1 board */

/*------ ---------- ISCSI-1 command definitions ----- ------ --------*/

#define SETOFFS   0x0003          /* address offset definition */
#define DEFOFFS   0x0008          /* define physical block offset */
#define RESET     0x0009          /* firmware reset */
#define SUNPARM   0x000a          /* set unit parameter */
#define BLOCKING  0x000b          /* toggle blocking mode */
#define WRPARAL   0x000c          /* toggle parallel write mode */
#define MAININIT  0x000d          /* global setup */
#define SFTIME    0x000e          /* set flush time interval */
#define DEBLOCK   0x000f          /* enable/disable sector deblocking */
#define GETCHR    0x0010          /* get single character */
#define PUTCHR    0x0014          /* output one or two chars */
#define GETBLOCK  0x0020          /* get fixed sized block */
#define PUTBLOCK  0x0022          /* output fixed sized block */
#define GETSTR    0x0030          /* get delimited string */
#define PUTSTR    0x0032          /* output delimited string */
#define GETCNT    0x0034          /* get counted string */
#define PUTCNT    0x0035          /* output counted string */
#define CTLSTAT   0x0044          /* get controller status */
#define CHKTARG   0x0045          /* check existing targets */
#define GUNPARM   0x0046          /* get unit parameters */
#define EXPROG    0x0050          /* execute user program */
#define BACKUP    0x0055          /* backup winni to winni */
#define FLUSH     0x0056          /* flush modified buffers */
#define FORMAT    0x0057          /* format logical unit */
#define FTRACK    0x0e58          /* format floppy track */
#define COMPARE   0x0059          /* compare data winni to winni */
#define LOCKUN    0x005a          /* lock local units */
#define FREEUN    0x005b          /* unlock local units */
#define COPY      0x005c          /* copy data blocks */
#define CHAIN     0x005d          /* enter command chain mode */
#define SEARCH    0x005e          /* search data on lun */
#define CLRHASH   0x005f          /* clear hashing buffers */
#define TRSPMOD   0x0060          /* SCSIbus transparent mode */
#define TARGMOD   0x0061          /* toggle SCSIbus target mode */
#define TARGINT   0x0062          /* enable/disable target mode ints */
#define INTFLAG   0x1000          /* complete with interrupt flag */
```

```
/*---------------------- ISCSI-1 structures ---------    ----. -------*/

struct BIM_ODD {                                /* BIM and the status register *
/* 0000 */      unsigned short  bimcr1;
/* 0002 */      unsigned short  bimcr2;
/* 0004 */      unsigned short  bimcr3;
/* 0006 */      unsigned short  bimcr4;
/* 0008 */      unsigned short  bimvr1;
/* 000A */      unsigned short  bimvr2;
/* 000C */      unsigned short  bimvr3;
/* 000E */      unsigned short  bimvr4;
};

Struct CMDRAM {                         /* CMDRAM for intelligent controllers */
/* 0000 */      unsigned int  cmd;      /* command */
/* 0002 */      unsigned int  parm1;    /* first word param */
/* 0004 */      unsigned long  ptr1;    /* first pointer parameter */
/* 0008 */      unsigned long _ptr2;    /* second pointer parameter */
/* 000C */      unsigned int _parm2;    /* second word parameter */
/* 000E */      unsigned int _ipres;    /* res. for interprocess. commun
};

struct IABORT {                         /* abort interrupt location */
/* 0000 */      char  res3;             /* unused even byte */
/* 0001 */      char _abint;            /* abort int location */
};

struct IRESET {                         /* reset board and firmware */
/* 0000 */      char  res4;             /* unused even byte */
/* 0001 */      char _reset;            /* reset location */
};

struct TMSG {                           /* ISCSI-1 target message area */
/* 0000 */      unsigned int _valid;    /* valid message available */
/* 0002 */      unsigned int  initid;   /* initiator id */
/* 0004 */      char *_mptr;            /* message pointer */
/* 0008 */      unsigned int _numbl;    /* number of received blocks */
};

struct ISCSI {
/* 0000 */      struct BIM_ODD  bim[0x100];     /* bim structure */
/* 1000 */      struct IABORT  _iabort[0x0400]; /* abort location */
/* 1800 */      struct IRESET _ireset[0x0400]; /* reset location */
/* 2000 */      unsigned int _version;  /* board and revision type */
/* 2002 */      char _unus1[0x00ee];    /* unused/reserved */
/* 20F0 */      struct TMSG _targmsg;   /* target message area */
/* 20FA */      char _unus2[0x0006];    /* unused/reserved */
/* 2100 */      struct CMDRAM _cmd1;    /* command RAM #1 */
/* 2110 */      char _chn1[0x01f0];     /* command chain area #1 */
/* 2300 */      char  iob1[0x0800];     /* I/O buffer #1 */
/* 2B00 */      char  iob2[0x0800];     /* I/O buffer #2 */
/* 3300 */      char  chn2[0x0200];     /* command chain area #2 */
/* 3500 */      struct CMDRAM  cmd2;    /* command RAM #2 */
/* 3510 */      char  unus3[0x0af0];    /* unused/reserved */
/* 4000 */      char  hbuff[0x1c000];   /* hashing buffers */
};
```

```
/*---  ------------- pointer definitions -       ---------------*/

struct ISCSI *iscsi = ISCSI1;  /* define pointer to iscsi board */

/*---     -----    ------   end of header ----------- ---------*/


/*
  ************************************
  *                                  *
  * SAMPLE OF ISCSI 1 FUNCTIONS      *
  *                                  *
  ************************************
*/


 setoffs(offs)
long offs;
{
int icmd;
    icmd = SETOFFS;
    if (int flag) icmd |= INTFLAG;
    iscsi-> cmd1._ptr1 = offs;
    iscsi-> cmd1._cmd = icmd;
    if (!int_flag)
    {
        while (iscsi-> cmd1._cmd > 0);
        return(iscsi-> cmd1._cmd & 0x0f);
    }
    else return(0);
}

_defoffs(offs)
long offs;
{
int icmd;
    icmd = DEFOFFS;
    if (int flag) icmd |= INTFLAG;
    iscsi-> cmd1._ptr1 = offs;
    if (!int flag)
    {
        while (iscsi-> cmd1._cmd > 0);
        return(iscsi-> cmd1._cmd & 0x0f);
    }
    else return(0);
}

_reset()
{
long timeout;
    iscsi-> cmd1._cmd = RESET;
    while (((iscsi-> bim[0]. bimctl & 0xff00) >> 8) == 0xff);
    while (((iscsi-> bim[0]. bimctl & 0xff00) >> 8) != 0xff);
    for(timeout = 0; timeout < 100000; timeout++);
    return(0);
}
```

```
 sunparm(id,lun,ptr)
int id,lun;
char *ptr;
{
int len,i,icmd;
    icmd = (id << 9) | SUNPARM;
    if (int_flag) icmd |= INTFLAG;
    if ((id == 7) && (ptr[7] == 8)) len = 2;
    else len = 1;
    for (i = 0; i < len; i++) iscsi->iobi[i] = ptr[i];
    iscsi->_cmdl._ptrl = &iscsi->_iobi[0];
    iscsi->_cmdl._parm2 = lun;
    iscsi->_cmdl._cmd = icmd;
    if (!int_flag)
    {
        while (iscsi->_cmdl._cmd > 0);
        return(iscsi->_cmdl._cmd & 0x0f);
    }
    else return(0);
}


 blocking()
{
int icmd;
    icmd = BLOCKING;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmdl._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmdl._cmd > 0);
        return(iscsi->_cmdl._cmd & 0x0f);
    }
    else return(0);
}


 wrparal(id_1,id_2)
int id_1,id_2;
{
int icmd;
    icmd = WRPARAL;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmdl._parm2 = (id_1 << 8) | id_2;
    iscsi->_cmdl._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmdl._cmd > 0);
        return(iscsi->_cmdl._cmd & 0x0f);
    }
    else return(0);
}
```

```
maininit(ownid,ssize,delim)
char ownid,ssize,delim;
{
int icmd;
    icmd = MAININIT;
    if (int flag) icmd |= INTFLAG;
    iscsi->iobl[0] = ownid;
    iscsi->iobl[1] = ssize;
    iscsi->iobl[2] = delim;
    iscsi->cmdl.ptr1 = &iscsi->iobl[0];
    iscsi->cmdl.cmd = icmd;
    if (!int flag)
    {
        while(iscsi->cmdl.cmd > 0);
        return(iscsi->cmdl.cmd & 0x0f);
    }
    else return(0);
}


sftime(seconds)
int seconds;
{
int icmd;
    icmd   SFTIME;
    if (int flag) icmd    INTFLAG;
    iscsi->cmdl.ptr1 = (long)(seconds * 500);
    iscsi->cmdl.cmd = icmd;
    if (!int flag)
    {
        while(iscsi->cmdl.cmd > 0);
        return(iscsi->cmdl.cmd & 0x0f);
    }
    else return(0);
}


deblock(factor,flag)
long factor;
int flag;
{
int icmd;
    icmd = DEBLOCK;
    if (int flag) icmd |= INTFLAG;
    iscsi->cmdl.ptr1 = factor;
    iscsi->cmdl.parm1 = flag;
    if (!int flag)
    {
        while (iscsi->cmdl.cmd > 0);
        return(iscsi->cmdl.cmd & 0x0f);
    }
    else return(0);
}
```

```
 getch(bladdr,offs,id,lun,c)
long bladdr,offs;
int id,lun,*c;
{
int icmd;
    icmd = (id << 9) | GETCHR;
    if (int flag) icmd |= INTFLAG;
    iscsi->_cmd1._ptr1 = bladdr;
    iscsi->_cmd1._ptr2 = offs;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        *c = iscsi->_cmd1._parm1;
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}


_putch(bladdr,offs,id,lun,c)
long bladdr,offs;
int id,lun,c;
{
int icmd;
    icmd = (id << 9) | PUTCHR;
    if (int flag) icmd |= INTFLAG;
    iscsi->_cmd1._parm1 = c;
    iscsi->_cmd1._ptr1 = bladdr;
    iscsi->_cmd1._ptr2 = offs;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}
```

```
 getbloc(bladdr,num,id,lun,ptr)
long bladdr,*ptr;
int num,id,lun;
{
int icmd;
    icmd = (id << 9) | GETBLOCK;
    if (int flag) icmd |= INTFLAG;
    iscsi->_cmd1._parm1 = num;
    iscsi->_cmd1._ptr1 = bladdr;
    iscsi->_cmd1._ptr2 = *ptr;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        *ptr = iscsi->_cmd1._ptr1;
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}



 putbloc(bladdr,num,id,lun,ptr)
long bladdr;
char *ptr;
int num,id,lun;
{
int icmd,i;
    icmd = (id << 9) | PUTBLOCK;
    if (int flag) icmd |= INTFLAG;
    for (i = 0; i < bisize; i++) iscsi->_iobl[i] = ptr[i];
    iscsi->_cmd1._parm1 = num;
    iscsi->_cmd1._ptr2 = &iscsi->_iobl[0];
    iscsi->_cmd1._ptr1 = bladdr;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}
```

```
 getstr(bladdr,offs,id,lun,ptr)
long bladdr,offs,*ptr;
int id,lun;
{
int icmd;
    icmd = (id << 9) | GETSTR;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmdl._parm1 = 0;
    iscsi->_cmdl._ptr1 = offs;
    iscsi->_cmdl._ptr2 = bladdr;
    iscsi->_cmdl._parm2 = lun;
    iscsi->_cmdl._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmdl._cmd > 0);
        *ptr = iscsi->_cmdl._ptr1;
        return(iscsi->_cmdl._cmd & 0x0f);
    }
    else return(0);
}


_putstr(bladdr,offs,id,lun,ptr)
long bladdr,offs;
int id,lun;
char *ptr;
{
int icmd,i;
    icmd = (id << 9)   PUTSTR;
    if (int_flag) icmd |= INTFLAG;
    i = 0;
    while(ptr[i])
    {
        iscsi->_iobl[i] = ptr[i];
        i++;
    }
    iscsi->_iobl[i] = 0;
    iscsi->_cmdl._parm1 = 0;
    iscsi->_cmdl._ptr1 = offs;
    iscsi->_cmdl._ptr2 = bladdr;
    iscsi->_cmdl._parm2 = lun;
    iscsi->_cmdl._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmdl._cmd > 0);
        return(iscsi->_cmdl._cmd & 0x0f);
    }
    else return(0);
}
```

```
 getcnt(bladdr,offs,id,lun,len,ptr)
long bladdr,offs,*ptr;
int id,lun,len;
{
int icmd;
    icmd = (id << 9) | GETCNT;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmd1._parm1 = len;
    iscsi->_cmd1._ptr1 = offs;
    iscsi->_cmd1._ptr2 = bladdr;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        *ptr = iscsi->_cmd1._ptr1;
        return(iscsi->_cmd1._cmd & 0x01);
    }
    else return(0);
}



 putcnt(bladdr,offs,id,lun,len,ptr)
long bladdr,offs;
int id,lun,len;
char *ptr;
{
int icmd,i;
    icmd = (id << 9) | PUTCNT;
    if (!int_flag) icmd |= INTFLAG;
    for (i = 0; i < len; i++) iscsi->iob[i] = ptr[i];
    iscsi->_cmd1._parm1 = len;
    iscsi->_cmd1._ptr1 = offs;
    iscsi->_cmd1._ptr2 = bladdr;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x01);
    }
    else return(0);
}
```

```
 ctlstat(ptr)
long *ptr;
{
int icmd;
    icmd = CTLSTAT;
    if (int flag) icmd |= INTFLAG;
    iscsi-> cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        *ptr = iscsi-> cmd1. ptr1;
        return(iscsi-> cmd1. cmd & 0x0f);
    }
    else return(0);
}


_chktarg(ptr)
long *ptr;
{
int icmd;
    icmd = CHKTARG;
    if (int flag) icmd |  INTFLAG;
    iscsi > cmd1._cmd = icmd;
    if (!int flag)
    {
        while(iscsi > cmd1. cmd > 0);
        *ptr = iscsi-> cmd1. ptr1;
        return(iscsi > cmd1. cmd & 0x0f);
    }
    else return(0);
}



 gunparm(id,lun,ptr)
int id,lun;
long *ptr;
{
int icmd;
    icmd = (id << 9) | GUNPARM;
    if (int_flag) icmd |= INTFLAG;
    iscsi ._cmd.. parm2 = lun;
    iscsi-> cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi-> cmd1. cmd > 0);
        *ptr = iscsi-> cmd1. ptr1;
        return(iscsi->_cmd1. cmd & 0x0f);
    }
    else return(0);
}
```

```
 exprog(start)
char *start;
{
int icmd;
    icmd = EXPROG;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmd1._ptr1 = start;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}


 backup(sid,slun,did,dlun)
int sid,slun,did,dlun;
{
int icmd;
    icmd = BACKUP;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmd1._ptr1 = sid;
    iscsi->_cmd1._ptr1 <<= 16;
    iscsi->_cmd1._ptr1 |= slun;
    iscsi->_cmd1._ptr2 = did;
    iscsi->_cmd1._ptr2 <<= 16;
    iscsi->_cmd1._ptr2  = dlun;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}



_flush()
{
int icmd;
    icmd = FLUSH;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}
```

```
 format(id,lun)
int id,lun;
{
int icmd;
    icmd = (id << 9) | FORMAT;
    if (int_flag) icmd |= INTFLAG;
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}


 ftrack(track,lun,ptr)
int track,lun;
char *ptr;
{
int icmd,i,len;
    icmd = FTRACK;
    if (int_flag) icmd |= INTFLAG;
    if (ptr[7] == 8) len = 27;
    else len = 11;
    for (i = 0; i < len; i++) iscsi->_iobl[i] = ptr[i];
    iscsi->_cmd1._ptr1 = &iscsi->_iobl[0];
    iscsi->_cmd1._parm2 = lun;
    iscsi->_cmd1._parm1 = track;
    iscsi->_cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi->_cmd1._cmd > 0);
        return(iscsi->_cmd1._cmd & 0x0f);
    }
    else return(0);
}


/* the following structure is needed for parameter passing by the*/
/* compare and the copy command */

struct CPARM {          /* compare, copy parameter block */
    char _sid;          /* source target id */
    char _slun;         /* source lun */
    long _saddr;        /* source start blockaddress */
    char _did;          /* destination target id */
    char _dlun;         /* destination lun */
    long _daddr;        /* destination start block address */
    int  _count;        /* number of blocks */
};
```

```
struct CPARM * parm;

 compare(sid,slun,saddr,did,dlun,daddr,len)
int sid,slun,did,dlun,len;
long saddr,daddr;
{
int icmd;
    _parm = &iscsi->_iobl[0];
    icmd = COMPARE;
    if (!int_flag) icmd |= INTFLAG;
    _parm-> sid = sid;
     parm-> slun = slun;
     parm-> saddr = saddr;
    _parm-> did = did;
     parm-> dlun = dlun;
    _parm-> daddr = daddr;
    _parm-> count = len;
    iscsi-> cmd1._ptr1 = _parm;
    iscsi-> cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi-> cmd1._cmd > 0);
        return(iscsi-> cmd1._cmd & 0x0f);
    }
    else return(0);
}


 lockun(mask)
int mask;
{
int icmd;
    icmd = LOCKUN;
    if (int_flag) icmd |= INTFLAG;
    iscsi-> cmd1._parm1 = mask;
    iscsi-> cmd1._cmd = icmd;
    if (!int_flag)
    {
        while(iscsi-> cmd1._cmd > 0);
        return(iscsi-> cmd1._cmd & 0x0f);
    }
    else return(0);
}
```

```
freeun(mask)
int mask;
{
int icmd;
    icmd = FREEUN;
    if (int flag) icmd |= INTFLAG;
    iscsi-> cmd1. parm1 = mask;
    iscsi-> cmd1. cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmd1. cmd > 0);
        return(iscsi-> cmd1. cmd & 0x0f);
    }
    else return(0);
}


copy(sid,slun,saddr,did,dlun,daddr,len)
int sid,slun,did,dlun,len;
long saddr,daddr;
{
int icmd;
    parm = &iscsi-> inbi[0];
    icmd   COPY;
    if (int_flag) icmd |= INTFLAG;
    parm-> sid = sid;
    parm-> slun = slun;
    parm-> saddr = saddr;
    parm-> did = did;
    parm-> dlun = dlun;
    parm-> daddr = daddr;
    parm-> count = len;
    iscsi-> cmd1. ptr1 = parm;
    iscsi-> cmd1. cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmd1. cmd > 0);
        return(iscsi-> cmd1. cmd & 0x0f);
    }
    else return(0);
}


chain()
{
int icmd;
    icmd = CHAIN;
    if (int flag) icmd |= INTFLAG;
    iscsi-> cmd1. cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmd1. cmd > 0);
        return(iscsi-> cmd1. cmd & 0x0f);
    }
    else return(0);
}
```

```
 search(id,lun,plen,ptr,bladdr,offs)
int id,lun,plen;
char *ptr;
long *bladdr,*offs;
{
int icmd,i;
    icmd = (id << 9) | SEARCH;
    if (int flag) icmd |= INTFLAG;
    for (i = 0; i < plen; i++) iscsi->_iobl[i] = ptr[i];
    iscsi-> cmdl._parm1 = plen;
    iscsi-> cmdl._ptr1 = &iscsi->_iobl[0];
    iscsi-> cmdl._parm2 = lun;
    iscsi-> cmdl._cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmdl._cmd > 0);
        *bladdr = iscsi->_cmdl._ptr1;
        *offs = iscsi->_cmdl._ptr2;
        return(iscsi-> cmdl._cmd & 0x0f);
    }
    else return(0);
}
```

```
 trspmod(id,cmd,dlen,data)
int id,dlen;
char *cmd;
long *data;
{
int icmd,i;
char *p;
    icmd = TRSPMOD;
    if (int flag) icmd |= INTFLAG;
    for (i = 0; i <= cmd[0]; i++) iscsi-> cmdl[i] = cmd[i];
    if (*data)
    {
        p = *data;
        for(i = 0; i < dlen; i++) iscsi-> iobl[i] = p[i];
    }
    iscsi -> cmdl. parm1 = dlen;
    iscsi -> cmdl. ptr1 = &iscsi-> cmdl[0];
    iscsi -> cmdl. ptr2 = &iscsi-> iobl[0];
    iscsi-> cmdl. parm2 = id;
    iscsi-> cmdl. cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmdl. cmd > 0);
        *data = iscsi-> cmdl. ptr2;
        return(iscsi -> cmdl. cmd & 0x0f);
    }
    else return(0);
}



 targmod()
{
int icmd;
    icmd = TARGMOD;
    if (int flag) icmd  = INTFLAG;
    iscsi-> cmdl. cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmdl. cmd > 0);
        return(iscsi-> cmdl. cmd & 0x01);
    }
    else return(0);
}
```

```
 targint(flag)
int flag;
{
int icmd;
    icmd = TARGINT;
    if (int flag) icmd |= INTFLAG;
    iscsi->cmdi._parm1 = flag;
    iscsi->cmdi._cmd = icmd;
    if (!int flag)
    {
        while(iscsi-> cmdi._cmd > 0);
        return(iscsi-> cmdi._cmd & 0x0f);
    }
    else return(0);
}


_hreset()
{
char c;
long i;
    c   iscsi->_ireset[0]._reset;
    while (((iscsi->_bim[0]._bimerl & 0xff00) >> 8) == 0xff);
    while (((iscsi->_bim[0]._bimerl & 0xff00) >> 8) != 0xff);
    for(i = 0; i < 400000; i++);
    return(0);
}


 abort()
{
char d;
long to;
    d = iscsi-> iabort[0]. abint;
    for (to = 0; to < 500000; to++);
}
```

This page was intentionally left blank

This page was intentionally left blank

APPLICATIONS

This page was intentionally left blank

```
*****************************************************************
*
* DRIVER FOR FORCE ISCSI-1 DISK CONTROLLER
* THIS CONTROLLER IS CALLED FROM THE STANDARD FORCE DISK CONTROLLER
* HEADER.
*
* UPDATE SCHEDULE:
*  04-FEB-87  1.0  M.S.  INITIAL VERSION
*  16-FEB-87  1.1  M.S.  Reduced default number of tracks on Micropolis
*                        winchester and the number of floppy partitions
*                        supported by default.
*  10-MAR-87  1.2  M.S.  Wait at INIT for SYSFAIL to disappear in
*                        the ISCSI-1 status register
*
*
        IFUDF   RETRY  :RETRY   EQU  0 ; DO NO RETRIES ON ERROR
*****************************************************************
*         --- ISCSI-1 CODE ---
*
*         D0.B - SELECT CODE BYTE       (TARGET ID | LUN)
*         D1.L = LOGICAL SECTOR #       (BLOCK NUMBER)
*         D2.L = READ/WRITE COMMAND     (0/1)
*         (A2) = DATA ADDRESS
*         A5   = B$$RAM ADDRESS
*
ISCSINIT CLR.W  -(A7)           ;SET .NE. FOR ENTRY FROM W$XDIT
*
* BUILD COMMAND IN D2.L: $2000=READ, $2200=WRITE
*
ISCS100 ADD.W   #$20,D2         ;MAKE 20,21
        CMPI.B  #$20,D2         ;READ ?
          BEQ.S @000            ;Y
        ADDQ.L  #1,D2           ;N, MAKE WRITE
@000    LSL.L   #8,D2           ;MAKE $2000/$2200
*
* BUILD COMMANDRAM IN D-REGISTERS
*    D3.L = CMD | NUMBER OF BLOCKS(=1)
*    D4.L = BLOCK ADDRESS
*    D5.L = DATA ADDRESS IN ISCSI DPR
*    D6.L = LUN | 00
*
        LEA.L   ISCCMD1,A1      ;GET ISCSI-1 BASE ADDRESS
        CLR.W   $E(A1)          ;RESET RETRY FLAG (MULTIPROC)
        MOVEQ.L #1,D3           ;1 BLOCK
        SWAP    D3              ;COUNT | 00
        MOVE.W  D2,D3           ;GET COMMAND: COUNT | COMMAND
        LSR.W   #8,D3           ;GET ISCSI COMMAND
        BTST.B  #ISCSI_PSINTF   ;USING ISCSI INTS?
          BEQ.S @010            ;N, SKIP EVENT RESET
        OR.W    #$1000,D3       ;SET INTERRUPT BIT
        BCLR.B  #^W$EVNT,W$EVNT/8+EVTB.+S$$RAM ;RESET EVENT 119
@010    CLR.L   D6
        MOVE.B  D0,D6           ;GET ID AND LUN
        ROR.L   #4,D6
        ROR.W   #7,D6           ;GET LLLL.0000.0000.0000|0000.III0.0000.0000
        OR.W    D6,D3           ;PUT ID INTO COMMAND
        SWAP    D6
        ROL.W   #4,D6           ;D6.W = 0000LLLL
        MOVE.L  D1,D4
        LEA.L   ISCIOB1,A4
        MOVE.L  A4,D5           ;GET I/O BUFFER ADDRESS
*
* COMMAND BLOCK IS COMPLETE NOW, GO TO READ OR WRITE COMMAND
*
        MOVE.W  #64-1,D7        ;GET LONG WORD COUNT
@020    TST.B   (A1)            ;WAIT HERE IF CONTROLLER BUSY
```

```
          BPL.S @020              ;CONTROLLER BUSY (??)
        CMPI.B  #$20,D3           ;READ OR WRITE ?
          BEQ.S @050              ;IT IS READ
*
* IT IS WRITE -- CHECK DMA AND TRANSFER DATA
*
@040    BTST.B  #DSC51,P$INTF     ;USE DMA FOR TRANSFER ?
          BEQ.S @048              ;N
*
* WE USE DMA
*
        MOVE.L  FSDMAC,A3         ;POINT TO CHIP
        MOVE.B  #$FF,DCSR(A3)     ;RESET DMA
        MOVE.L  A2,DMAR(A3)       ;SET MEMORY ADDRESS
        MOVE.L  D5,DDAR(A3)       ;SET DEVICE ADDRESS
        MOVE.W  #128,DMTC(A3)     ;DO 256 BYTES TRANSFER
        MOVE.B  #$11,DOCR(A3)     ;DO MEM TO DEV ON MAX SPEED
        MOVE.B  #$80,DCCR(A3)     ;GO !!
@044    BTST.B  #7,DCSR(A3)       ;POLL FOR COMPLETION
        BEQ.S   @044              ;WAIT
        BTST.B  #4,DCSR(A3)       ;ERROR ?
        BEQ.S   @050              ;N
        MOVEQ.L #107,D0           ;ERROR 107 = DMA ERROR
        BRA     ISCOUT
*
* CPU DOES THE TRANSFER
*
@048    MOVE.L  (A2)+,(A4)+       ;OUTPUT DATA TO CONTROLLER
          DBF   D7,@048           ;LOOP
*
* DMA AND POLLING,  WRITE, COME HERE AFTER DATA TRANSFER
*
@050    SWAP    D3
        MOVE.W  D6,$C(A1)         ;SET LUN
        MOVEM.L D3-D5,(A1)        ;LOAD UP COMMAND RAM
@054    BTST.B  #ISCSI,P$INTF     ;USE ISCSI INTS?
          BEQ.S @060              ;N, JUST POLL BUSY
        BSR     SUSPEND           ;Y, SUSPEND ON EV 119, TIMEOUT?
          BEQ.S @080              ;N, GET STATUS AND DROP INTO FINISH
        BCLR.B  #ISCSI,P$INTF     ;Y, DISABLE INTS FOR NEXT TIME
*
@080    CLR.L   D1
@082    MOVE.W  (A1),D1           ;GET STATUS
          BPL.S @082              ;LOOP, STILL BUSY
        SWAP    D3                ;GET COMMAND
        TST.B   D1                ;ERROR ??
*
* THE FOLLOWING LINE IS USED IF RETRY IS DISABLED
*
        IFEQ    RETRY
          BNE.S ISCSIE            ;Y
        ENDC
*
* THE FOLLOWING CODE IS USED FOR RETRY
*
        IFNE    RETRY
          BEQ.S @088              ;N
        MOVE.W  D1,D0
        AND.W   #$0E00,D0         ;MASK
        CMPI.W  #$E00,D0          ;ID 7 ?
          BNE.S ISCSIE            ;Y, DO ERROR HANDLING
        CMPI.B  #$F,D1            ;EXTENDED ERROR
          BNE.S ISCSIE
        CMPI.W  #$101,2(A1)       ;SCSI TIME OUT ?
          BNE.S ISCSIE            ;N
        TAS.B   $E(A1)            ;Y, RETRY ?
```

```
          BNE.S ISCSIE            ;N
          SWAP  D3                ;Y
          MOVE.L D3,(A1)          ;DO COMMAND AGAIN
          BRA.S @054
          ENDC
*
@088      CMPI.B #$20,D3          ;WAS IT A READ?
            BNE.S @100            ;N, TO COMMON
*
* IT IS READ -- CHECK DMA AND TRANSFER DATA
*
@090      BTST.B #DSCSI,PSINTF    ;USE DMA FOR TRANSFER ?
            BEQ.S @098            ;N
*
* WE USE DMA
*
          MOVE.L PSDMAC,A3        ;POINT TO CHIP
          MOVE.B #$FF,DCSR(A3)    ;RESET DMA
          MOVE.L A2,DMAR(A3)      ;SET MEMORY ADDRESS
          MOVE.L 4(A1),DDAR(A3)   ;SET DEVICE ADDRESS FROM RETURN CODE
          MOVE.W #128,DMTC(A3)    ;DO 256 BYTES TRANSFER
          MOVE.B #$91,DOCR(A3)    ;DO DEV TO MEM ON MAX SPEED
          MOVE.B #$80,DCCR(A3)    ;GO !!
@094      BTST.B #7,DCSR(A3)      ;POLL FOR COMPLETION
          BEQ.S  @094             ;WAIT
          BTST.B #4,DCSR(A3)      ;ERROR ?
          BEQ.S  @100             ;N
          MOVEQ.L #107,D0         ;ERROR 107 = DMA ERROR
          BRA.S  ISCOUT
*
* CPU DOES THE TRANSFER
*
@098      MOVEA.L 4(A1),A4        ;GET DATA ADDRESS
@099      MOVE.L  (A4)+,(A2)+     ;READ FROM BUFFER
            DBF   D7,@099         ;LOOP
*
* COMMON TERMINATION CODE
*
@100      ADDQ.W  #4,(A7)         ;SET .EQ.
*
* RETURN WITH ERROR IN D0
*
ISCOUT    RTR
*
* ISCSI-1 ERROR HANDLER
*
ISCSIE    CMPI.B  #$F,D1          ;EXTENDED ERROR ??
            BNE.S @010            ;N
          MOVE.W  2(A1),D1        ;Y, GET EXTENDED ERROR NUMBER
          CLR.L   D0
          MOVE.B  D1,D0
          LSR.W   #8,D1
          CMPI.B  #3,D1           ;FLOPPY ERROR ?
            BNE.S @002            ;N, ERROR 1-7
          ADD.W   #10,D0          ;Y, ERROR 11-17

@002      ADD.W   #160,D0         ;SCSI - 110-128
          BRA.S   ISCOUT
*
@010      ADD.W   #146,D1         ;STANDARD ERROR
          MOVE.L  D1,D0
          BRA.S   ISCOUT
          PAGE
*****************************************************************************
*         SYS68K/SCSI-1 DISK INIT
*
```

```
*  IN: (A4) = DISK 0 PARAMTER TABEL
*      (A5) = P$WPARM AREA
*      (A6) = P$FPARM AREA
*
INITSCSI
*
* INIT DMAC
*
        TST.W   P$DMAC          ;DO WE HAVE A DMA ??
         BEQ.S  @000            ;N
        MOVE.L  P$DMAC,A1       ;Y, GET BASE
        MOVE.B  #$FF,DCSR(A1)   ;RESET CSR
        MOVE.B  #$08,DDCR(A1)   ;SET BURST, 8-BIT 68000 DEV (ALSO IF 16-BIT!!!!
        MOVE.B  #5,DSCR(A1)     ;MEM AND DEVICE COUNTS UP
        MOVE.B  #5,DMFC(A1)     ;SUPERVISOR DATA ACCESS
        MOVE.B  #5,DDFC(A1)     ;SUPERVISOR DATA ACCESS
        MOVE.B  #0,DBFC(A1)     ;BFC IS DON'T CARE
*
* INIT BIM
*
@000    LEA.L   ISCBIM,A0       ;POINT TO BIM
        MOVE.B  #WSEVNT,ISCBV0(A0) ;SET COMPLETE VECTOR
        MOVE.B  #WSEVNT,ISCBV1(A0) ;ERROR VECTOR IS SAME
        MOVE.B  #$54,ISCBC0(A0) ;LEVEL 4
        MOVE.B  #$54,ISCBC1(A0) ;ERROR IS SAME
*
* SET OFFSET TO BASE ADDRESS
*
        MOVE.L  A0,D0           ;GET ADDRESS OF BOARD
        AND.L   #$FFFE0000,D0   ;MASK IT
        LEA.L   ISCCMD1,A0
@010    MOVE.W  ISCSIB,D1       ;READ STATUS REGISTER
        EOR.W   #8,D1
        BTST    #2,D1           ;TEST SYSFAIL FLAG, WAIT HERE IF SET
        BEQ.S   @010
@011    TST.B   (A0)            ;WAIT HERE IF BUSY
         BPL.S  @011
        MOVE.L  D0,4(A0)        ;SET BASE ADDRESS
        MOVE.W  #$3,(A0)        ;WRITE COMMAND
@012    TST.B   (A0)            ;WAIT UNTIL DONE
         BPL.S  @012
*
* INIT 3 SCSI WINIS (ID 0,1,2)
*
        LEA.L   INIDAT(PC),A1   ;POINT TO DATA
        LEA.L   ISCIOD1,A2      ;POINT TO I/O BUFFER
        MOVE.L  (A1)+,(A2)+     ;3 LONGS
        MOVE.L  (A1)+,(A2)+
        MOVE.L  (A1)+,(A2)+
        LEA.L   INIINS(PC),A1   ;GET INIT INSTRUCTIONS
@020    MOVEM.L (A1)+,D3-D6     ;GET THEM
        MOVE.W  D6,$C(A0)       ;SET CON
        MOVEM.L D3-D5,(A0)      ;WRITE TO ISCSI CMD2
@022    TST.B   (A0)            ;WAIT FOR COMMAND TO COMPLETE
         BPL.S  @022
        TST.W   (A1)            ;END OF TABLE ?
         BPL.S  @020            ;N
*
* UNIT PARMS ARE SET NOW FOR 3 WINIS, FLOPPIES USE THE DEFAULT
* NOW WE SET THE FLOPPY PARTITIONS AND TRY TO READ THE HEADER
* INFORMATION OF THE WINIS.
*
        MOVE.W  #$0373,D6       ;CNTR | DSEL FOR FLOPPY
        LEA.L   FLPARM(PC),A1   ;GET FLOPPY TABLE
        MOVEA.L A1,A2           ;GET A DUMMY POINTER
        EXG     A5,A6           ;A5 = PSFPARM
```

```
        BSR     DOIT            ;LOADUP FLOPPY PARMS
        ADDQ.L  #1,D6           ;SECOND FLOPPY DRIVE
        LEA.L   FLIPARM(PC),A1  ;GET FLOPPY TABLE FOR SECOND DRIVE
        BSR     DOIT            ;LOADUP PARMS OF SECOND FLOPPY
        EXG     A5,A6           ;GET WINI PARM AREA
*
        MOVE.W  #$300,D6        ;CNT / ID
@040    LEA.L   PSBLOCK,A2      ;DISK DATA SHOULD GO THERE
        MOVE.B  D6,D0           ;WE START WITH ID 0
        MOVEQ.L #0,D1           ;READ BLOCK 0
        MOVEQ.L #$0,D2          ;READ COMMAND
        MOVEM.L D6/A2/A4/A5,-(A7)       ;SAVE THESE PARMS
        CLR.L   (A2)            ;DESTROY MESSAGE
        LEA.L   BSSRAM,A5
        BSR     ISCSINIT        ;READ BLOCK 0
        MOVEM.L (A7)+,D6/A2/A4/A5
          BNE.S @054            ;SKIP IF READ ERROR
        LEA.L   WIPARM(PC),A1   ;GET DEFAULT WINI PARMS
        BSR     DOIT            ;LOAD UP PARMS AS DRIVE FOUND
@054    ADD.W   #$10,D6         ;INCREMENT LUN
        CMPI.B  #$30,D6         ;DONE 3 WINIS ?
          BNE.S @040            ;N
*
@050
        TST.W   P$DMAC          ;DO WE HAVE A DMA ?
        BEQ.S   @052            ;N
        BSET    #DSCSI,P$INTF   ;Y, USE IT
@052    BSET    #ISCSI,P$INTF   ;ENABLE INTS
        RTS
        PAGE
*****************************************************************************
*       ISCSI-1 DISK FORMATTER
* THIS FUNCTION DOES A WINCHESTER FORMAT WITH THE INSTALLED UNIT
* PARAMETERS.
*
* IN:   0(A7).L - RETURN ADDRESS
*       4(A7).W - ID | LUN
*   OR AT ENTRY ADDRESS + 4:
*       D0.W    = ID| LUN
*
* OUT:  D0 = ERROR CODE OR 0 IF NO ERROR
*       SR = .NE.       OR    .EQ.
*
* NO REGISTERS ARE DESTROYED (EXCEPT D0 FOR STATUS RETURN)
*
SCSIFORM MOVE.W 4(A7),D0        ;GET PARMS
SCSIF1  MOVEM.L D1/D2/A0,-(A7)  ;SAVE REGISTERS
        LEA.L   ISCCMD1,A0      ;GET CONTROLLER ADDRESS
        CLR.W   $E(A0)          ;CLEAR FLAG
        MOVE.W  D0,D2           ;GET A COPY
@000    MOVE.W  D2,D0           ;GET ID | LUN
        MOVE.W  D0,D1
        AND.W   #$0FF,D0        ;D0.W = LUN
        AND.W   #$0F00,D1       ;D1.W = ID | 00
        ROL.W   #1,D1           ;POSITION ID
        OR.W    #$57,D1         ;OR-IN FORMAT COMMAND
*
@002    TST.B   (A0)            ;BUSY ?
          BPL.S @002            ;Y, WAIT
        MOVE.W  D0,$C(A0)       ;WRITE LUN
        MOVE.W  D1,(A0)         ;WRITE COMMAND AND ID
@006    MOVE.W  (A0),D0         ;GET STATUS, BUSY ?
          BPL.S @006            ;Y ,WAIT
        AND.W   #$0FF,D0        ;MASK
          BEQ.S @020            ;NO ERROR
        TAS.B   $E(A0)          ;ERROR BEFORE ??
```

```
          BEQ.S @000              ;N, RETRY
          CMPI.W  #$0F,D0         ;EXTENDED ERROR ?
            BNE.S @020            ;N
          MOVE.W  2(A0),D0        ;Y, GET ERROR
@020      MOVEM.L (A7)+,D1/D2/A0
          RTS
          PAGE
*****************************************************************
*         DATA FOR UNIT PARAMETERS
*
INIDAT    DC.L    0
          DC.L    0               ; USE DEFAULT FOR THOSE
          DC.B    0               ; HARD DISK ASSUMED
          DC.B    0               ; 256 BYTES/SECTOR
          DC.B    1               ; HASHING ENABLED
          DC.B    0               ; DUMMY FOR EVEN ADDRESS
*
*         COMMANDS TO SET UNIT PARAMETERS FOR 3 UNITS
*
INIINS    DC.B    0               ; ID 0
          DC.B    $0A             ; OPCODE
          DC.W    0               ; NOT USED
          DC.L    ISCIOB1         ; ADDRESS OF FIRST I/O BUFFER
          DC.L    0               ; NOT USED
          DC.W    0               ; LUN 0
          DC.W    0               ; NOT USED
*
          DC.B    2               ; ID 1
          DC.B    $0A             ; OPCODE
          DC.W    0               ; NOT USED
          DC.L    ISCIOB1         ; ADDRESS OF FIRST I/O BUFFER
          DC.L    0               ; NOT USED
          DC.W    0               ; LUN 0
          DC.W    0               ; NOT USED
*
          DC.B    4               ; ID 2
          DC.B    $0A             ; OPCODE
          DC.W    0               ; NOT USED
          DC.L    ISCIOB1         ; ADDRESS OF FIRST I/O BUFFER
          DC.L    0               ; NOT USED
          DC.W    0               ; LUN 0
          DC.W    0               ; NOT USED
*
          DC.W    $FFFF           ; END OF TABLE
*
          PAGE
*****************************************************************
*
*         DEFAULT ISCSI-1 HEADER PARTITIONS
*
* MOST OF THE VALUES IN THE TABLE ARE DUMMIES AS THE ACTUAL
* NUMBER OF CYLS, HEADS, SECTORS/TRACK ARE NOT USED ON SCSI
* WINCHESTER DRIVES.
* THE FOLLOWING IS VALID FOR THE MICROPOLIS WINCHESTER 1375. THIS
* DEVICE HAS MORE THAN 513000 BLOCKS WHEN FORMATTED WITH 256 BYTES
* PER SECTOR. THE TRANSLATION FOR PDOS USES 512000 BLOCKS.
*
*
WIPARM    DC.W    16              ; HEDS
          DC.W    1000            ; CYLS
          DC.W    32              ; SPT
          DC.W    256             ; BPB
          DC.W    0               ; SKIP
          DC.W    32              ; SPT
          DC.W    22              ; # OF PARTS
          DC.W    0               ; BAD TRACKS, NOT USED !!
```

```
        DC.W    $0300           ;CNTN & DSEL
        DC.W    0               ;STEP
        DC.W    0               ;REDU WRT CURR
        DC.W    0               ;WRT PERCOMP
        DC.W    2,0,1499        ;WINI PART 1 ...
        DC.W    3,1500,2999
        DC.W    4,3000,4499
        DC.W    5,4500,5999
        DC.W    6,6000,7499
        DC.W    7,7500,8999
        DC.W    9,9000,10499
        DC.W    10,10500,11999
        DC.W    11,12000,13499
        DC.W    12,13500,14999  ;..WINI PART 10
        DC.W    13,15000,15079  ;FLOPPY PART 1 ...
        DC.W    14,15080,15159
        DC.W    15,15160,15239
        DC.W    16,15240,15319
        DC.W    17,15320,15399
        DC.W    18,15400,15479
        DC.W    19,15480,15559
        DC.W    20,15560,15639
        DC.W    21,15640,15719
        DC.W    22,15720,15799
        DC.W    23,15800,15879
        DC.W    24,15880,15959  ;..FLOPPY PART 12
        DC.W    0
*
        PAGE
************************************
*       FAKE FLOPPY PARTITION TABLE
*
FLPARM  DC.W    2               ;FLOPPY HEDS
        DC.W    80              ;CYLS
        DC.W    16              ;PHYS SPT BLOCKS PER TRACK
        DC.W    256             ;PHYS BPB BYTES PER BLOCK
        DC.W    0               ;SKIP
        DC.W    16              ;# PDOS SPT SECTORS PER TRACK
        DC.W    1               ;# PARTS
        DC.W    0               ;# BAD TRKS
        DC.W    $0173           ;CNTN & DSEL
        DC.W    0               ;STEP
        DC.W    0               ;REDU WRT CURR
        DC.W    0               ;WRT PERCOMP
        DC.W    0               ;DISK #0
        DC.W    0,159           ;PARTITION 0
        DC.W    0
*
FLIPARM DC.W    2               ;FLOPPY HEDS
        DC.W    80              ;CYLS
        DC.W    16              ;PHYS SPT BLOCKS PER TRACK
        DC.W    256             ;PHYS BPB BYTES PER BLOCK
        DC.W    0               ;SKIP
        DC.W    16              ;# PDOS SPT SECTORS PER TRACK
        DC.W    1               ;# PARTS
        DC.W    0               ;# BAD TRKS
        DC.W    $0174           ;CNTN & DSEL
        DC.W    0               ;STEP
        DC.W    0               ;REDU WRT CURR
        DC.W    0               ;WRT PERCOMP
        DC.W    1               ;DISK #1
        DC.W    0,159           ;PARTITION 1
        DC.W    0
*
********** END OF ISCSI-1 DRIVER **********
```

This page was intentionally left blank

```
/*
 * Stollmann & Co Driver for the Force ISCSI-1
 * Copyright (1987) Stollmann , Hamburg
 * author:      um
 *
 * For use of the ISCSI to control Floppy disks and Disk/Tape Storage Media onl
 */

#define VOLUMES 7                               /* max. Number of volumes per driver */
#define DT(x)           (x & 0xF)               /* get pure disktype */
#define SCSIID(x)       ((x) & 0x70) >> 4)      /* Get Target id */
#define SECOFF(x)       ((x) & 0x7001 >> 8)     /* Sectoroffset. */
#define SCSIUNIT(x)     ((x) & 0x7000) >> 12)   /* Target submnit */
#define TAPE(x)         (DT(x)==1)              /* 1 == Tape */
#define FLOP(x)         (DT(x)>1)               /* 0 Winny, 2.. Floppy */


#define P(x)    (minor(x))
#define limit(d) ((minor(d)&7!=7 |.!(Thd_olim))
                                                /* provent off limit write      */

#include "sys/types.h"
#include "sys/sysmacros.h"
#include "sys/param.h"
#include "sys/dir.h"
#include "sys/signal.h"
#include "sys/utsname.h"
#include "sys/buf.h"
#include "sys/conf.h"
#include "sys/user.h"
#include "sys/elog.h"
#include "sys/erec.h"
#include "sys/errno.h"
#include "sys/iobuf.h"
#include "sys/systm.h"
#include "sys/disk.h"
#include "sys/iotmode.h"



/*
 * Definiton for the Configuration pool
 *
 */

#define CONFMAGIC       0x120387

extern  struct {
        int             cf_magic;
        char            cf_name[8];
        char            cf_prod[16];
        struct dkldev   cf_ldev[8];
        } hdconf[];

typedef unsigned char devreg;

/*
 * global ISCSI Data used during maininit
 *
 */

struct iscsi_global     {
        devreg  i_busid;                /* scsi own busid */
        devreg  i_secsize;              /* interlacing secsize */
        devreg  i_strdelim;             /* string delimiter, not used here */
```

```
        }       iscsi_global = { 0x07, 0x04, '\0' };

struct  bimctl  {
        devreg  bimf;   devreg  ctlreg;
        };

struct  bimvec  {
        devreg  bimf;   devreg  vecreg;
        };

struct  iscsi_bim       {
        struct bimctl   bimctl[4];
        struct bimvec   bimvec[4];
        };

/*
 * used for input and return parameters from ISCSI.
 */

struct  cmdram  {
        ushort  icmd;                   /* Command and target ID  / Status */
        ushort  icmdpar;                /* Command dependent Paramter / Error */
        uint    iparblock;              /* Parameter Block or data offset */
        uint    iblockno;               /* Logical Block no. */
        ushort  iunit;                  /* Logical Unit */
        ushort  ires;                   /* reserved */
        };

#define Cmd_intenable   0x1000

#define Cmd_target(x)   ((x) << 9)
#define Cmd_pure(x)     ((x) & 0xFF)

#define Cmd_reset       0x09            /* Reset firmware */
#define Cmd_sunparm     0x0a            /* Set unit parameters */
#define Cmd_maininit    0x0d            /* ISCSI Initialization */
#define Cmd_readb       0x20            /* Read Blocks from unit */
#define Cmd_writeb      0x22            /* Write Blocks to unit */
#define Cmd_cristat     0x44            /* Get controller status */
#define Cmd_chklarg     0x45            /* Check for and test targets */
#define Cmd_gunparm     0x46            /* Get unit parameters */
#define Cmd_format      0x57            /* Format entire Unit */
#define Cmd_ftrack      0x58            /* Format track (floppy only) */
#define Cmd_trApmod     0x60            /* Transparent mode */

/* Commands used in transparent mode to streamer */

#define Cmd_treadb      0x100A
#define Cmd_twriteb     0x100A
#define Cmd_twritefmark 0x1010
#define Cmd_tnoremoval  0x101E
#define Cmd_tremoval    0x201E
#define Cmd_trewind     0x1001
#define Cmd_tload       0x101B
#define Cmd_tunload     0x201B


struct  iscsi_dpr       {
        ushort          rev;            /* ISCSI and firmware revision */
        devreg          dfill[0x100-2];
        struct  cmdram  cmdram1;        /* First command ram */
        struct  cmdram  cmdchain1[31];  /* For command chaining */
        devreg          iobuf1[2*1024]; /* 2k IO-Buffer 1 */
        devreg          iobuf2[2*1024]; /* 2k IO-Buffer 2 */
        struct  cmdram  cmdchain2[32];
        struct  cmdram  cmdram2;        /* Second cmdram */
```

```
        };

extern  int     hd_ivec[];
extern  int     hd_addr[];              /* This MUST contain the address of the
                                         * DPR area of the ISCSI
                                         */

#define DPROFFSET       0x2000
#define ISCSIDPR        ((struct iscsi_dpr *) hd_addr[0])
#define ISCSIBIM        ((struct iscsi_bim *) (hd_addr[0] - DPROFFSET))
#define ISCSIBBASE      ((devreg *) (hd_addr[0] - DPROFFSET))
#define IOBUF1          ((int) ((char *) ISCSIDPR->iobuf1   (char *) ISCSIBIM))
#define IOBUF2          ((int) ((char *) ISCSIDPR->iobuf2 - (char *) ISCSIBIM))
#define CMDCHAIN2       ((int) ((char *) ISCSIDPR->cmdchain2-(char *) ISCSIBIM)


#if BSIZE==1024
#define  blkno b_sector
#endif

struct  iostat  hdstat[VOLUMES];
struct  iobuf hdtab = tabinit(BLKHD, hdstat);

extern  struct  dkvol   hdvol[];        /* volume infos                 */
extern  struct  dkldev  hdldev[];       /* logical device infos         */


/*
 * disktype dependent data
 */

struct floppyparm       {
        devreg  f_heads;
        devreg  f_cyl;
        devreg  f_sptrack;
        devreg  f_firstsec;
        devreg  f_bps;
        devreg  f_dens;
        devreg  f_formpattern;
        devreg  f_interleave;
        devreg  f_secoff;
        devreg  f_hashing;
        devreg  f_filler[1];
        };

struct scsiparm         {
        long    s_comset0;
        long    s_comset1;
        devreg  s_devtype;
        devreg  s_logbsize;
        devreg  s_hashing;
        };


#define DTNOFLOP        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
#define DTNOSCSI        { 0, 0, 0, 0, 0 }
#define DT0DATA { 0xFF, 0xFF, 0x00, 0x01, 0x00 },  DTNOFLOP, 0  /* Winnie */
#define DT1DATA         DTNOSCSI, DTNOFLOP, 1    /* Streamer */
#define DT1024DATA      DTNOSCSI, {0x02,80,0x0a,0,3,1,0xE5,0,1,0,0}, 0
#define DT256DATA       DTNOSCSI, {0x02,80,0x20,0,1,1,0xE5,0,1,0,0}, 2
#define DT512DATA       DTNOSCSI, {0x02,80,0x12,0,2,1,0xE5,0,1,0,0}, 2

struct hd_dt_dat
        {
        struct  scsiparm        scsiparm;
```

```
        struct  floppyparm      floppyparm;
        devreg  dt_shift;                       /* To Blocksize */
        } hd_dt_dat[] = { DT0DATA, DT1DATA, DT256DATA, DT512DATA, DT1024DATA };


/* Transparent mode Parameter Block definition */

struct  hd_tmode        {
        devreg  t_len;
        devreg  t_cmd;
        devreg  t_cdp;                  /* command dependent parameter */
        devreg  t_par[10];              /* Some are dummy .. */
        };

#define TMODCMD ((struct hd_tmode *) (ISCSIDPR->cmdchain3))
                /* Put then in cmdchain2 cause iobuf1 must be left free */


/***    data for current io-request at work                     ***/

struct hd_dt_dat        *hd_dtdat;      /* disktype spec. data          */
unsigned        hd_iocmd;       /* instruction code data transfer cmd   */
unsigned char   hd_dev;         /* deviceno major,minor                 */
int             hd_cnt;         /* sectors to perform                   */
unsigned        hd_iosec;       /* sectorno                             */
caddr_t         hd_iomem;       /* adress for data transfer             */
struct buf      *hd_ibp;        /* buffer to work on                    */
struct dkvol    *hd_vol;        /* volume to work on                    */
short           hd_gsec;        /* sectors to perform in track          */
#define qsta hdtab.b_active     /* main state: hdstart / hdint / chain  */

int             hdunits;        /* number of allowed physical units     */
struct  buf     hdcbuf;         /* for commands used by hdcmd           */
char    hdwork[BSIZE];          /* used by ioctl; avoid simult. access  */
int     hd_tapewritten;         /* true if EOF mark should be written   */
int     hd_tapebno;             /* to ensure sequential read/writes     */
devreg  hd_sense;               /* error code as reported by last sense */
                                /* Reset in hdtrack, analysed in hddone */
int     hd_sensebno;            /* logical block no of disk error       */
```

```
hdopen(dev)
                                /***    public procedure                *
                                 *      recalibrate the drive           */
register dev_t dev;
{       int punit; punit = physical(dev);
        if (Thd_misc) printf("###hd%d: open \n",P(dev));
        if (punit >= hdunits) u.u_error = ENXIO;

        if (FLOP(hdvol[physical(dev)].disktype)) {
                if (Thd_misc)
                        printf("###hd%d: Floppy init\n", P(dev));
                hdcmd(dev, Cmd_sunparm, 0, 1, 0); /* Setup parameters for LUN *
                }
        if (TAPE(hdvol[physical(dev)].disktype)) {
                if (Thd_misc)
                        printf("###hd%d: Tape init\n", P(dev));
                hdcmd(dev, Cmd_tload, 0, 1, 0);
                hdcmd(dev, Cmd_tnoremoval, 0, 1, 0); /* Transparent mode
                                                        disallow removal */
                hdcmd(dev, Cmd_trewind, 0, 1, 0); /* rewind the tape */
                }
        }



hdcmd(dev, cmd, bn, bcnt, addr)
                                /***    pass command to driver          *
                                 *      called by ioctl, format         *
                                 *      calls   strategy                */
dev_t dev; daddr_t addr;
{       int s;
        s=spl6();
        while (hdcbuf.b_flags & B_BUSY) {
                hdcbuf.b_flags |= B_WANTED;
                sleep((caddr_t)&hdcbuf, PRIBIO+1);}
        hdcbuf.b_flags |= B_BUSY;
        splx(s);
        if (Thd_cmd) printf("###hd%d cmd = %x bn = %x addr = %x\n",
                        P(dev), cmd, bn, addr);
        hdcbuf.b_dev =          dev;            /* pass cmd to hdstart() */
        hdcbuf.b_bcount =       bcnt * BSIZE;
        hdcbuf.b_resid =        (unsigned)cmd;  /* strategy sorts this  */
        hdcbuf.b_blkno =        bn;
        hdcbuf.b_un.b_addr =    (caddr_t)addr;
        hdRstrategy(&hdcbuf);
        iowait(&hdcbuf);
        s=spl6();
        if (hdcbuf.b_flags & B_WANTED)
                wakeup((caddr_t) &hdcbuf);
        hdcbuf.b_flags = 0;
        splx(s);
}

hdread(dev)
    dev_t dev;
    {   physio(hdRstrategy, 0, dev, B_READ); }

hdwrite(dev)
    dev_t dev;
    {   physio(hdRstrategy, 0, dev, B_WRITE); }

hdclose(dev) {
        if (TAPE(hdvol[physical(dev)].disktype) && hd_tapewritten) {
                /* write EOF Mark */
                if (Thd_misc)
```

```
                printf("###hd%d: Tape EOF\n", P(dev));
            hdcmd(dev, Cmd_writefmark, 0, 1, 0); /* Transparent mode
                                                    write filemark */
            /*hdcmd(dev, Cmd_unload, 0, 1, 0);*/
            hdcmd(dev, Cmd_removal, 0, 1, 0);    /* Transparent mode
                                                    allow removal */
            hd_tapewritten = 0;
            }
        }

hdprint(dev,str) char *str;
    {   printf("###hd%d %s\n", P(dev), str); }
```

```
hdstrategy(bp)
#if BSIZE==1024
    register struct buf *bp;
{       bp->Blkno = bp->b_blkno >> 1;
        if (bp->b_bcount & (BSIZE-1) || (bp->b_blkno & 1))
                 printf("###hd%d bad request blk=%d count=%d\n",
                        P(bp->b_dev), bp->b_blkno, bp->b_bcount);
        hdStrategy(bp); }

hdStrategy(bp)
    register struct buf *bp;
{       bp->Blkno = bp->b_blkno ;
        if (bp->b_bcount & (BSIZE-1))
                printf("###rhd%d bad sectorsize %d\n",P(bp->b_dev),bp->b_bcount
        hdStrategy(bp); }

hdStrategy(bp)
#endif          /***    enter buffer in queue, possibly wakeup driver   *
                 *      called by physio, cmd, init                     *
                 *      calls init, start                               */
register struct buf *bp;
{       register unsigned punit, i, s;
        register struct buf *;
        punit = physical(bp->b_dev);
        if (hd_stra) printf("###hd%d strategy flag=%x cnt=%x bno=%d mem=%x\n",
           P(bp->b_dev), bp->b_flags, bp->b_bcount, bp->Blkno, bp->b_un.b_addr)
        bp->av_forw = (struct buf *)NULL;
        bp->b_start = lbolt;
        if (bp == &hdcbuf) {
                hdstat[punit].io_misc++;         /* errlog: */
                s=spl6();
                if (hdtab.b_actf == (struct buf *)NULL)
                        hdtab.b_actf = bp;
                else    hdtab.b_actl->av_forw = bp;
                hdtab.b_actl = bp;
        } else {
                hdstat[punit].io_ops++;          /* errlog: */
                i = bp->Blkno + hd_dev[minor(bp->b_dev)].bl_offset;
                bp->b_resid = i ;               /* sort by track-no */
                s=spl6();
                if (hdtab.b_actf == (struct buf *)NULL) {
                        hdtab.b_actf = bp;
                        hdtab.b_actl = bp; }
                else    disksort(&hdtab, bp);
        }
        if (qsta == 0) hdchain();               /* trigger the chain */
        splx(s);
}
```

```
hdioctl(dev, cmd, addr, flag)
dev_t dev;
caddr_t addr;
{       switch (cmd) {
                case SIOCFORMAT:                /* format volume        */
                    hdformat(dev, addr); break;
                case SIOCNDBK:                  /* set badblock illegal */
                    break;
                case SIOCBD00:                  /* set badblock none    */
                    break;
                case SIOCSETP:                  /* set hd test          */
                    if      ((int)addr&Test_set) hd_test |= (int)addr;
                    else if ((int)addr&Test_res) hd_test&=~(int)addr;
                    else                         hd_test = (int)addr;
                    break;
                case SIOCGETP:                  /* get volume info      */
                    { struct dkinfo *dk = (struct dkinfo *)addr;
                    if (copyout(&hdvol[physical(dev)], &dk->vol,
                            sizeof(struct dkvol))) u.u_error = EFAULT;
                    if (copyout(&hdldev[minor(dev)], &dk->ldev,
                            sizeof(struct dkldev))) u.u_error = EFAULT;
                    }
                    break;
                default:        u.u_error = EINVAL;
        }
}

hdformat(dev, addr)

dev_t dev; caddr_t addr;        {

        struct dkvol *hd_vol;
        struct hd_dt_dat *hd_dtdat;
        int cyl, hd, sec, bias=0, bn=0;
        if (limit(dev)) {u.u_error=EINVAL; return; }
        hd_vol = &hdvol[physical(dev)];
        hd_dtdat = &hd_dt_dat[DT(hd_vol->disktype)];

        hdcmd(dev, Cmd_format, 0, 1, 0);
        }
```

```
hdinit()        {
                                /***    initial startups        ****/

        int     i,index,k,size;

        if (bprobe( &ISCSIDPR->rev, -1))        {
                printf("NO ISCSI! ");
                return;
                }

#define BIM_LEVEL       0x01
#define BIM_IRAC        0x08
#define BIM_IRE         0x10
#define BIM_EXTRN       0x20
#define BIM_FAC         0x40
#define BIM_FLAG        0x80

        for (i=2; i < 4; i++)   {
                ISCSIBIM->bimctl[i].ctlreg = BIM_LEVEL | BIM_IRE;
                ISCSIBIM->bimvec[i].vecreg = hd_ivec[0] >> 2;
                }

        printf("iscsi ");
        hdunits = VOLUMES;
        hdcmd(0, &md_maininit, 0, 1, 0);
                /* Now do the Configuration */

        for (i=0; i<7; i++)     {
                struct {
                        devreg  hdifl[8];
                        char    hdivendor[8];
                        char    hdiproduct[16];
                        char    hdif2[4];
                } inq;

                if ((hdldev[i*8].bl_offset == -1) &&
                    (hdldev[i*8].bl_offset == -1)) {
                        if (hdinquire(SCSIID(hdvol[i].disktype),&inq)==-1)
                          if (hdinquire(SCSIID(hdvol[i].disktype),&inq)==-1)
                                {
                                printf("hd: Cannot inquire on %d\n", i);
                                break;
                                }
                        hdstrip(inq.hdivendor, sizeof(inq.hdivendor));
                        hdstrip(inq.hdiproduct, sizeof(inq.hdiproduct));
                        if (Thd_misc)
                                printf("hdinit : :%s: :%s:,\n", inq.hdivendor,
                                        inq.hdiproduct);
                        index=-1;
                        k=0;
                        while (hdconf[k].cf_magic == CONFMAGIC) {
                          if (hdstrcmp(hdconf[k].cf_name, inq.hdivendor) &&
                              hdstrcmp(hdconf[k].cf_prod, inq.hdiproduct))
                                index = k;
                          k++;
                          }
                        if (index == -1)        /* oops */
                                panic("hd: Autoconfig failed");
                        printf("%s %s ", hdconf[index].cf_name,
                                hdconf[index].cf_prod);
                        bcopy(hdconf[index].cf_ldev, &hdldev[i*8],
                                8*sizeof(struct dkldev));
                        }
                i++;
                }
```

This page was intentionally left blank

```
/*************************************************************************

this completes the Unix - interface to the driver
now:
the driver at the lower level.
activated by interrupt or by strategy if asleep
works only on 'current io request data' as defined above
Note: On interrupt-level: donot use events: wakeup / sleep.
        donot manipulate user-structure, there is none !

*************************************************************************/


                /* chain is called by strategy if someth in queue
                 * but iscsi is inactive.
                 * priv is already set to 6!            */


hdintr()
                                /* called from kernel on scsi interrupt */
{       if (qsta==0) printf("###hd: interrupt while inactive \n");
        hdchain();                      /* completion interrupt */
        }


hdchain()
{       register i;
        for (;;){
                switch(qsta){
                        case  0 : if ((i=hdstart())==0) return;
                        case 't': i=hdtrack();  break;
                        case 'd': i=hddone();   break;
                        default : printf("###hd: chain bad state\n"); }
                if (i)  { return; ;             /* just wait for interrupt */
                }                               /* else loop around     */
}
```

```
hdstart()
{       struct dkldev *minp;
        if ((hd_ibp=hdtab.b_actf)==(struct buf *)NULL) return(0);
        blkacty     |= (1<<BLKHD);      /* setup request vector */
        hd_iomem     = hd_ibp->b_un.b_addr;
        hd_dev       = hd_ibp->b_dev;
        hd_vol       = &hdvol[physical (hd_dev)];
        minp         = &hdldev[minor(hd_dev)];
        hd_dtdat     = &hd_dt_dat[DT(hd_vol->disktype)];

        hd_cnt       = hd_ibp->b_bcount >> (BSHIFT - hd_dtdat->dt_shift);
                                        /* no of blocks        */
        hd_iosec     = hd_ibp->Blkno << hd_dtdat->dt_shift;
        if (hd_ibp==&hdcbuf) hd_iocmd = hdcbuf.b_resid;
          else hd_iocmd = (hd_ibp->b_flags&B_READ ? Cmd_readb : Cmd_writeb );
        if (TAPE[hd_vol->disktype] && (hd_ibp != &hdcbuf))
                hd_iocmd = (hd_ibp->b_flags&B_READ) ? Cmd_treadb : Cmd_twriteb;
        hd_ibp->b_resid = hd_ibp->b_bcount;
        if ((hd_iosec+hd_cnt > (minp->bl_size << hd_dtdat->dt_shift))
                        && limit[hd_dev])
                 hd_cnt = (minp->bl_size << hd_dtdat->dt_shift) - hd_iosec;
                        /* error! just cut      */
        hd_iosec        += (minp->bl_offset << hd_dtdat->dt_shift);
        qsta         = 't';
        hd_sense     = 0;                       /* no error */
        return (1);                             /* chain to hdtrack     */
}


hdtrack()
{       register unsigned i, tno, bno, maxtr;
        register struct hd_dt_dat *tosdt = &hd_dt_dat[DT(hd_vol->disktype)];
        if (hd_rwop) {
                printf("###hd%d: op %x cnt=%x bno=%d mem=%x",
                        P(hd_dev),hd_iocmd,hd_cnt,hd_ibp->Blkno,hd_iomem);
                printf(" SCSI/Unit =%x %x",
                        SCSIID(hd_vol->disktype), SCSIUNIT(hd_vol->disktype));
                }
        hd_qsec = 0;
        if (hd_cnt<=0) return (hdgood());
        hd_qsec = hd_cnt;                       /* compute qsec         */
        if (hd_iocmd == Cmd_readb)
                maxtr = (16   -((hd_iosec >> tosdt->dt_shift) & 15)) <<
                        tosdt->dt_shift;
                /* up to next 16k boundary for read */
        else
                maxtr = (2 << tosdt->dt_shift);
        hd_qsec = (hd_cnt > maxtr) ? maxtr : hd_cnt;
        if (hd_rwop)
                printf(" LBlock %x Lcount %x\n", hd_iosec, hd_cnt);
                /* transfer from hd_iosec hd_qsec blocks. */
        switch (hd_iocmd) {
                case Cmd_readb:
                        ISCSIDPR->cmdram2.icmdpar = hd_qsec;
                        ISCSIDPR->cmdram2.iparblock = hd_iosec;
                        ISCSIDPR->cmdram2.iblockno = IOBUF2;
                                        /* undoc., for nonhashed dev */
                        ISCSIDPR->cmdram2.iunit = SCSIUNIT(hd_vol->disktype);
                        ISCSIDPR->cmdram2.icmd = Cmd_readb |
                                        Cmd_intenable |
                                        Cmd_target(SCSIID(hd_vol->disktype));
                        break;
                case Cmd_writeb:
                        ISCSIDPR->cmdram2.icmdpar = hd_qsec;
                        ISCSIDPR->cmdram2.iparblock = hd_iosec;
```

```
        ISCSIDPR->cmdram2.iblockno = IOBUF2;
        ISCSIDPR->cmdram2.iunit = SCSIUNIT(hd_vol->disktype);
        bcopy(hd_iomem, ISCSIDPR->iobuf2,
                hd_qsec << (BSHIFT - hd_dtdat->dt_shift));
        ISCSIDPR->cmdram2.icmd = Cmd_writeb |
                        Cmd_intenable |
                        Cmd_target(SCSIID(hd_vol->disktype));
        break;
case Cmd_format:
        if (DT(hd_vol->disktype) == 0)  { /* winnie */
          if (mselect(hd_vol))
            printf("###hd Format error (mode sel)\n");
          }
        ISCSIDPR->cmdram2.iunit = SCSIUNIT(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmd  = Cmd_format |
                        Cmd_intenable |
                        Cmd_target(SCSIID(hd_vol->disktype));
        break;
case Cmd_sunparm:
        ISCSIDPR->cmdram2.iparblock     = IOBUF2;
        if (FLOP(hd_vol->disktype)) {
                bcopy(&tosdh->floppyparm, ISCSIDPR->iobuf2,
                        sizeof(struct floppyparm));
                if (SECOFF(hd_vol->disktype) == 0) {
                  int i;
                  ((struct floppyparm *)
                        (ISCSIDPR->iobuf2))->f_secoff = 0;
                  ((struct floppyparm *)
                        (ISCSIDPR->iobuf2))->f_interleave = 8;
                  for (i=0; i<20; i++)
                    ((struct floppyparm *)
                        (ISCSIDPR->iobuf2))->f_fill[i] = i;
                  }
                }
        else
                bcopy(&tosdh->scsiparm, ISCSIDPR->iobuf2,
                        sizeof(struct scsiparm));
        ISCSIDPR->cmdram2.iunit = SCSIUNIT(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmd  = Cmd_sunparm |
                          Cmd_intenable |
                          Cmd_target(SCSIID(hd_vol->disktype));
        break;
case Cmd_maininit:
        ISCSIDPR->cmdram2.iparblock = IOBUF2;
        bcopy(&iscsi_global, ISCSIDPR->iobuf2,
                        sizeof(struct iscsi_global));
        ISCSIDPR->cmdram2.icmd  = Cmd_maininit |
                          Cmd_intenable |
                          Cmd_target(SCSIID(hd_vol->disktype));
        break;
case Cmd_twriteb:
        if (hd_tapebno != hd_iosec) {
                qsta = 'd';       /* do not allow nonseq. */
                return 0;
                }
        bcopy(hd_iomem, ISCSIDPR->iobuf2,
                hd_qsec << (BSHIFT - hd_dtdat->dt_shift));
        TMODCMD->t_len = 6;
        TMODCMD->t_cmd = Cmd_pure(Cmd_twriteb);
        TMODCMD->t_cdp = 1;
        TMODCMD->t_par[0] = (hd_qsec >> 16) & 0xff;
        TMODCMD->t_par[1] = (hd_qsec >> 8) & 0xff;
        TMODCMD->t_par[2] = hd_qsec & 0xff;
        TMODCMD->t_par[3] = 0;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
```

```
        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmdpar   = hd_qsec <<
                                (BSHIFT-hd_dtdat->dt_shift);
        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod |
                                      Cmd_intenable;
        hd_tapewritten = 1;
        break;
case Cmd_treadb:
        if (hd_tapebno != hd_iosec) {
                qsca ='d';      /* do not allow nonseq. */
                return 0;
                }
        TMODCMD->t_len = 6;
        TMODCMD->t_cmd = Cmd_pure(Cmd_treadb);
        TMODCMD->t_cdp = 1;
        TMODCMD->t_par[0] = (hd_qsec >> 16) & 0xff;
        TMODCMD->t_par[1] = (hd_qsec >> 8) & 0xff;
        TMODCMD->t_par[2] = hd_qsec & 0xff;
        TMODCMD->t_par[3] = 0;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmdpar   = hd_qsec <<
                                (BSHIFT-hd_dtdat->dt_shift);
        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod |
                                      Cmd_intenable;
        break;
case Cmd_twritefmark:
        TMODCMD->t_len = 6;
        TMODCMD->t_cmd = Cmd_pure(Cmd_twritefmark);
        TMODCMD->t_cdp = 0;
        TMODCMD->t_par[0] = 0;
        TMODCMD->t_par[1] = 0;
        TMODCMD->t_par[2] = 1;
        TMODCMD->t_par[3] = 0;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmdpar   = 0;
        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod |
                                      Cmd_intenable;
        break;
case Cmd_tload:
case Cmd_tunload:
        TMODCMD->t_len = 6;
        TMODCMD->t_cmd = Cmd_pure(Cmd_tload);
                                /* same code */
        TMODCMD->t_cdp = 0;
        TMODCMD->t_par[0] = 0;
        TMODCMD->t_par[1] = 0;
        TMODCMD->t_par[2] = (hd_iocmd==Cmd_tload) ? 1 : 0;
        TMODCMD->t_par[3] = 0;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
        ISCSIDPR->cmdram2.icmdpar   = 0;
        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod |
                                      Cmd_intenable;
case Cmd_tnoremoval:
case Cmd_tremoval:
        TMODCMD->t_len = 6;
        TMODCMD->t_cmd = Cmd_pure(Cmd_tremoval);
                                /* same code */
        TMODCMD->t_cdp = 0;
        TMODCMD->t_par[0] = 0;
        TMODCMD->t_par[1] = 0;
```

```
                        TMODCMD->t_par[2] = (hd_iocmd==Cmd_tnoremoval) ? 1 : 0;
                        TMODCMD->t_par[3] = 0;
                        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
                        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
                        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
                        ISCSIDPR->cmdram2.icmdpar   = 0;
                        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod |
                                                      Cmd_intenable;
                        break;
                case Cmd_trewind:
                        TMODCMD->t_len = 6;
                        TMODCMD->t_cmd = Cmd_pure|Cmd_trewind;
                                                /* same code */
                        TMODCMD->t_cdp = 0;
                        TMODCMD->t_par[0] = 0;
                        TMODCMD->t_par[1] = 0;
                        TMODCMD->t_par[2] = 0;
                        TMODCMD->t_par[3] = 0;
                        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
                        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
                        ISCSIDPR->cmdram2.iunit     = SCSIID(hd_vol->disktype);
                        ISCSIDPR->cmdram2.icmdpar   = 0;
                        ISCSIDPR->cmdram2.icmd      = Cmd_trspmod
                                                      Cmd_intenable;
                        break;
                }
        qsta = 'd';
        return(1);
        }


char *hdstatusdecode();

hddone()        {

        if (hdstatusdecode()) return(hdfatal(0));
        switch (hd_iocmd) {
          case Cmd_readb:
                bcopy(ISCSIBBASE + (ISCSIDPR->cmdram2.iparblock), hd_iomem,
                        hd_qsec << (BSHIFT - hd_dtdat->dt_shift));
                if (!hd_rwop)
                        printf("DPR Rd. %x\n",
                                ISCSIBBASE+(ISCSIDPR->cmdram2.iparblock));
                break;
          case Cmd_treadb:
                bcopy(ISCSIDPR->iobuf2, hd_iomem,
                        hd_qsec << (BSHIFT - hd_dtdat->dt_shift));
                if (!hd_rwop)
                        printf("DPR tpr Rd. %x\n", ISCSIDPR->iobuf2);
                hd_tapebno += hd_qsec;
                break;
          case Cmd_twriteb:
                hd_tapebno += hd_qsec;
                break;
          case Cmd_trewind:
                hd_tapebno = 0;
                break;
          }
        return (hdgood());
        }

hdfatal(ow)
   {
        hderrlog(hdstatusdecode());             /* Error wfc            */
        hd_bp->b_flags |= B_ERROR;              /* mark error           */
        logberr(&hdtab,1);                      /* log: unrecovered     */
```

```
        return (hdend());
        }

hdgood()
    {
        if (hd sense)                       /* This means down here: end of tape */
                return(hdend());
        hd_ibp->b_resid  = hd_qsec << (BSHIFT - hd_dtdat->dt_shift);
        if ((hd_cnt -= hd_qsec) > 0)    {
                hd_iorem += hd_qsec << (BSHIFT - hd_dtdat->dt_shift);
                hd_iosec += hd_qsec;            /* more to come          */
                return(hd(track());
                }
        return (hdend());
        }

hdend()
    {   qsta = 0;
        hikacty &= ~(1<<BLKHD);
        hdtab.b_actf = hd_ibp->av_forw;
        iodone(hd_ibp);
        if (!hd_fake) printf("%c",(hdtab.b_actf?'-':' '));
        return(0);
        }
```

```
/***    HD  SUBROUTINES ***/

devreg hdreassb[ ] = { 0x07, 0x00, 0x00, 0x00, 0x00, 0x00 };

hdreassign(bno) {

        /* use the REASSIGN BLOCK command for the spec. block. */

        devreg *list;

        bcopy(hdreassb, ISCSIDPR->cmdchain2, sizeof(hdreassb));
        ISCSIDPR->cmdram2.icmdpar    = 8;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = CMDCHAIN2+256;
        list = ((devreg *) ISCSIDPR->cmdchain2 + 256);
        list[0] = 0; list[1] = 0; list[2] = 0; list[3] = 4;
        list[5] = (bno >> 24) & 0xFF;
        list[6] = (bno >> 16) & 0xFF;
        list[7] = (bno >> 8) & 0xFF;
        list[8] = bno & 0xFF;
        ISCSIDPR->cmdram2.icmd = Cmd_trspmod;
        while ((ISCSIDPR->cmdram2.icmd & 0x8000) == 0)
                ;
        if (ISCSIDPR->cmdram2.icmd != 0x8000)
                return -1;
        }


char *hdstatusdecode()  {

        char *reqsense();

        switch (hd_iocmd) {
                case Cmd_readb:
                case Cmd_writeb:
                        if (hd_iosec != hd_tapebno)
                                return "Nonsequential tape access";
                        break;
                }
        if (hd_sense)                   /* This is the second call */
                return reqsense();
        switch(ISCSIDPR->cmdram2.icmd & 0xff)   {
                case 0x00:      return 0;
                case 0x02:      return "Address error";
                case 0x06:      return "Missing or invalid parameter";
                case 0x07:      return "Illegal Command";
                case 0x0B:      return "Invalid Parameter";
                case 0x0D:      return "Controller in Target Mode";
                case 0xFF:      return "Unknown Command";
                case 0x0F:      switch(ISCSIDPR->cmdram2.icmdpar) {
                                        case 0x0101: return "Timeout";
                                        case 0x0102: return "Log. unit !ready";
                                        case 0x0103: return "SCSIbus read err";
                                        case 0x0104: return "SCSIbus write err"
                                        case 0x0105: return "Log. Block !found"
                                        case 0x0106: return "Target res.confl."
                                        case 0x0107: return "Format error";
                                        case 0x0109: return reqsense();
                                        case 0x0302: return "Floppy not ready";
                                        case 0x0303: return "FDC read error";
                                        case 0x0304: return "FDC write error";
                                        case 0x0305: return "FDC record !found"
                                        case 0x0307: return "FCD format error";
                                        }
                }
```

```
        return ("Unknown error");
        }


hderrlog(ecd)
    char *ecd;
{       short qblock[2];
        qblock[0]=(hd_sense) ? hd_sense : ISCSIDPR->cmdram2.icmd;
        hdtab.io_addr = (physadr) qblock;
        hdtab.io_nreq = 2;
        fmtberr( &hdtab, hd_iosec);
        if (hd_sense)
                printf("###hd%d: error sense=%x %s cmd=%x\n",
                        P(hd_ibp->b_dev), hd_sense, ecd, hd_iocmd);
        else
                printf("###hd%d: error sta=%x %s cmd=%x\n", P(hd_ibp->b_dev),
                        ISCSIDPR->cmdram2.icmd, ecd, hd_iocmd);
}


/* Dump driver is a no op */

hddump() {}

/* Some teststuff for SEM */

showscsi()      {}

char    hdmodesense[] = {  0x06, 0x1a, 0x00, 0x3f, 0x00, 0x40, 0 };
char    hdmodeselect[] = { 0x06, 0x15, 0x01, 0x00, 0x00, 0x40, 0 };

struct  msr     {
        devreg  d1;
        devreg  mt;
        devreg  res1;
        devreg  bd1;

        devreg  dens;
        devreg  lb2;
        devreg  lb1;
        devreg  lb0;
        devreg  res2;
        devreg  lb12;
        devreg  lb11;
        devreg  lb10;

        devreg  fill1[8];       /* page code 1 */

        devreg  fill2[12];      /* page code 2 */

        devreg  fill3[10];      /* page code 3 */
        ushort  spt;
        ushort  bps;
        devreg  fill4[10];      /* page code 3 continued */

        devreg  fill5[20];      /* page code 4 */

        };

msense(unit)    {

        int     bcount,bsize;
        struct msr *got = (struct msr *) (ISCSIDPR->iobuf2);

        bcopy(hdmodesense, ISCSIDPR->cmdchain2, sizeof(hdmodesense));
```

```
        ISCSIDPR->cmdram2.icmdpar       = 76;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = unit;
        ISCSIDPR->cmdram2.icmd = Cmd_trspmod;
        while ((ISCSIDPR->cmdram2.icmd & 0x8000) == 0)
                ;
        if (Thd_misc)
                printf("hd msense [unit %d code %x)\n",
                        unit, ISCSIDPR->cmdram2.icmd);
        if (ISCSIDPR->cmdram2.icmd != 0x8000) {
                return  -1;
                }
        bcount = (got->lb2 << 16) + (got->lb1 << 8) + got->lb0;
        bsize = (got->lb12 << 16) + (got->lb11 << 8) + got->lb10;
        return  (bcount*bsize)/1024;
        }

mselect(hdvptr)

struct dkvol *hdvptr;   {

        struct msr *got = (struct msr *) (ISCSIDPR->iobuf2);
        register unit = SCSIID(hdvptr->disktype);

        if (Thd_misc)
                printf("hd mselect (unit %d spt %d)\n",
                        unit, hdvptr->sec_p_track);
                                /* get the current data to IOBUF2       */
        if ((msense(unit)==-1) && (msense(unit)==-1))   {
                printf("hd: mode sense errorunit=%d\n", unit);
                return -1;
                }
        got->bps = 0x400;       /* select 1k sectorsize (physical)      */
        got->lb12 = 0x00;
        got->lb11 = 0x04;
        got->lb10 = 0x00;       /* select 1k sectorsize (logical)       */
        got->spt  = hdvptr->sec_p_track;        /* spt (io.h)           */
                /* Reset some things in the mode sense data */
        got->dl = 0;
        got->fill2[0] &= 0x3F;
        got->fill2[0] &= 0x3F;
        got->fill3[0] &= 0x3F;
        got->fill5[0] &= 0x3F;
        bcopy(hdmodeselect, ISCSIDPR->cmdchain2, sizeof(hdmodeselect));
        ISCSIDPR->cmdram2.icmdpar       = 76;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = unit;
        ISCSIDPR->cmdram2.icmd = Cmd_trspmod;
        while ((ISCSIDPR->cmdram2.icmd & 0x8000) == 0)
                ;
        if (ISCSIDPR->cmdram2.icmd != 0x8000) {
                return  -1;
                }
        return 0;
        }

char    hdcinquire[] = {  0x06, 0x12, 0x00, 0x00, 0x00, 36, 0 };

char hdgetsense[] = { 0x06, 0x03, 0x00, 0x00, 0x00, 11, 0x00 };

hdinquire(unit, dptr)

int     unit;
unsigned char *dptr;    {
```

```
        bcopy(hdcinquire, ISCSIDPR->cmdchain2, sizeof(hdcinquire));
        ISCSIDPR->cmdram2.icmdpar       = 36;
        ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
        ISCSIDPR->cmdram2.iblockno  = IOBUF2;
        ISCSIDPR->cmdram2.iunit     = unit;
        ISCSIDPR->cmdram2.icmd = Cmd_trspmod;
        while ((ISCSIDPR->cmdram2.icmd & 0x8000) == 0)
                ;
        if (Thd_misc)
                printf("hd inquire (unit %d code %x)\n",
                        unit, ISCSIDPR->cmdram2.icmd);
        if (ISCSIDPR->cmdram2.icmd != 0x8000) {
                return  -1;
                }
        bcopy(ISCSIDPR->iobuf2, dptr, 36);
        }

char *reqsense()        {

        int     i;
        struct  rqsense {
                devreg  rs_code;
                devreg  rs_lbh;
                devreg  rs_lbm;
                devreg  rs_lbl;
                } *rqptr;

        if (hd_sense==0) {
                if (TAPE(hd_vol->disktype))     {
                  bcopy(hdgetsense, ISCSIDPR->cmdchain2, sizeof(hdgetsense));
                  ISCSIDPR->cmdram2.icmdpar    = 11;
                  ISCSIDPR->cmdram2.iparblock = CMDCHAIN2;
                  ISCSIDPR->cmdram2.iblockno  = CMDCHAIN2+256;
                  ISCSIDPR->cmdram2.icmd = Cmd_trspmod;
                  while ((ISCSIDPR->cmdram2.icmd & 0x8000) == 0)
                        ;
                  if (ISCSIDPR->cmdram2.icmd != 0x8000)
                        return "hd: reqsense error";
                  hd_sense = *((devreg *) ISCSIDPR->cmdchain2 + 256 + 2);
                  }
                else    {
                        rqptr = (struct rqsense *) ISCSIDPR->cmdram2.iparblock
                                        + ISCSIBBASE;
                        if (rqptr->rs_code & 0x80) /* valid bno */
                          .     hd_sensebno = ((rqptr->rs_lbh & 0x1F) << 16) +
                                        (rqptr->rs_lbm << 8) +
                                        (rqptr->rs_lbl);
                        else
                                hd_sensebno = -1;
                        hd_sense = rqptr->rs_code;
                        if ((hd_sensebno != -1) &&
                                (((hd_sense & 0x0F) == 0x10) || /*error code*/
                                 ((hd_sense & 0x0F) == 0x11)) ) {
                          if (hdreassign(hd_sensebno) == -1)
                            printf("###hd%d: Cannot reassign id\n",
                                PInd ibp->b_dev), (hd_sensebno -
                                  hdidev[minor(hd_ibp->b_dev)].bl_offset));
                                /* make block numbers device relative ! */
                          else
                            printf("###hd%d: Reassigned Block %d (see doc)\n",
                                PInd_ibp->b_dev), (hd_sensebno -
                                  hdidev[minor(hd_ibp->b_dev)].bl_offset));
                                /* make block numbers device relative ! */
                          }
                        }
```

```
                ;
        if (TAPE!hd_vol->disktype):     {
          if (hd_sense & 0xF0)
                return 0;               /* No true error, but end of op */
          switch (hd_sense & 0x0F)      {
                case 0x00:      hd_sense=0; return 0;
                case 0x01:      hd_sense=0; return 0; /* Recovered */
                case 0x02:      return "Not Ready";
                case 0x03:      return "Media Error";
                case 0x04:      return "Hardware Error";
                case 0x05:      return "Illegal Request";
                case 0x06:      hd_sense=0; return 0; /* Ignore this one */
                case 0x07:      return "Write Protect";
                case 0x08:      return 0;       /* End of media */
                default:        printf ("hd: Unknown error %x", hd_sense);
                                return "Unknown";
                ;
          }
        else
          return "Hard disk error";
        return  0;
        }

hdstrip(str, len)

int     len;
char    *str;   {
        while ((*str != ' ') && len)    {
                str++;
                len--;
                }
        if (*str == ' ')
                *str = 0;
        }

hdstrcmp(s1, s2)

char    *s1, *s2;       {
        while ((*s1 == *s2) && (*s1 != '\0') && (*s2 != '\0'))
                { s1++; s2++; }
        return (*s1 - *s2);
        }
```

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP