

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 *DSPCon 9245* at our website: *Click HERE*

User Manual

Model 9245

for Custom VMEbus Systems

First Edition, Revision 5



The Leader in Digital Signal Processing Solutions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receiver and Data Acquisition Systems



Since 1991, our staff of qualified and dedicated systems and software design engineers has been working on the cutting edge of technology by employing innovative designs and aggressive testing techniques to provide a diverse customer base with data acquisition and analysis systems that help to keep them competitive.

DSP*Con* specializes in the development of DSP-based, high-tech systems for real-time data acquisition and analysis applications.

From digital receivers to sound and vibration systems, DSP*Con* has the experience to provide the appropriate products and services to meet your budget and product scheduling needs.

Hardware



DSP*Con* does not just deliver data acquisition systems, we offer consulting services, systems design, DSP board manufacturing and complete standard and custom digital signal processing systems based on COTS technology. DSP*Con* also provides post-purchase training and technical support.

Software



Data Recorders



Whether you are looking for 1024-channel sound and vibration analysis system or a 32-channel receiver system, DSP*Con* works with you and your engineers to design a high-performance solution that meets your needs.

www.dspcon.com

MODEL 9245 USER MANUAL
Edition 1.5—May 2001

A Publication of

DSP*Con*, Incorporated

© MMI

Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced, stored in or transmitted by a retrieval system, or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, without the written permission of DSP*Con*, Inc.

DSP*Con* has worked to verify the accuracy of the information contained in this document as of its publication date; however, such information is subject to change without notice and DSP*Con* is not responsible for any errors that may occur in this document.

Trademarks are acknowledged and are the property of their owners.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

REVISION HISTORY

Revision Date	Revision No.	Revised Section(s)	Comments/Notes
May 2001	1.5	Chapter Two	Table 2.5 “Serial Port Pins” corrected to reflect proper descriptions for RJ-45 pins 4 and 5.
March 2001	1.4	Chapters One, Three, & Four	*Chapter 1*— section entitled “VME Bus Control” modified for greater clarity; *Chapter 3*—Table 3.1 modified/corrected; *Chapter 4*—Table 4.2 added for greater clarity.
February 2001	1.3	Chapter Two	*Chapter 2*—Table 2.3 modified to correct pin names.
January 2001	1.2	Chapters One & Two	*Chapter 1*— updated Option numbers in Table 1.0; *Chapter 2*—modified Figure 2.2 for greater clarification.
December 2000	1.1	Chapter Two	Section on IP connectors expanded to include information on types of connectors for IP Module header J24; Figure 2.2 modified to show corrected aspect of pin numbers.
November 2000	——	——	First Edition

To download the latest version of this document visit the DSP*Con* website @ www.dspcon.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

CHAPTER ONE—INTRODUCTION 2

Using This Manual 2

General Product Description 2
- The Model 9245 Data Engine 3

Model 9245 Interfaces 3
- VMEbus Control 3
- MIX Bus Support 3
- IndustryPack (IP) Bus 4
- TMS320C40 Comm Port Interface 4
- RS-232 Serial Communications Port Interface 5
- Ethernet Interface 5
- VME Interface 5
- MIX Interface 6

System Block Diagram 6
- *Figure 1.1 — Block Diagram 7*

Specifications 8
- *Table 1.1 – System Specifications 8*

Power Consumption 9
- Electrical Specifications 9
  - Power Consumption 9

Model 9245 Peripherals 9
- *Table 1.2– Peripheral Wait States 9*
- Local and Global Static RAM 10
- Dynamic RAM 10
- Flash Chip 10
- Voltage Regulators & Reset Circuitry 10
- Peripheral Speeds 11
  - *Table 1.3 – Peripheral Speeds 11*

CHAPTER TWO—INSTALLATION & CONNECTIONS 14

Introduction 14

Inspection 14
- Unpacking the Unit 14
  - *Figure 2.1 — Circuit Board Showing Jumper Block Locations 15*

Jumper Block Locations 16
- *Table 2.1 – Jumper Block Pin Numbering 16*

Connectors 17
- PMC Module (PCI Mezzanine) Connectors 17
  - PMC Module 1 17

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PMC Module 2 17
IP Module Connectors 17
Table 2.2 – Pin Numbering IP Module *18*
Comm Port Connectors 18
Optional Comm Port Headers 19
Table 2.3 –Pin Numbering Optional Comm Ports 0, 1, 3, 4 ..... *19*
Figure 2.2 – Comm Port Pin Numbering (Options 009/010) *19*
VME P2 Connector 20
Table 2.4 – VME P2 Comm Port Pin Numbering (Rear Panel) *20*
DSPCon Model 9745 21
Emulator Connector 21
Figure 2.3 – Emulator Connector Pinouts *21*
The Front Panel 22
Figure 2.4 – Model 9245 Front Panel Design *22*
Serial Port (UART) Connector 23
Figure 2.5 – Serial Port Pin Numbering *23*
Table 2.5 – Serial Port Pins *23*
Ethernet Port Connector 23
Table 2.6 – Ethernet Port Pins *24*
LED Indicators 24
D1 and D2 24
D13 24
Front Panel Indicators 24
Reset Button 25
Reset Control Logic 25
Expansion Modules Installation 26
MIX Modules Installation 26
Step One 26
Step Two 26
PMC Module Installation 26
Industry Pack Module Installation 27
Figure 2.6 – Module Installation Diagram *27*
Installing the Assembly in the VMEbus Card Cage 28
Removing the Assembly from the VMEbus Card Cage 28

CHAPTER THREE – MEMORY MAPS & REGISTER DESCRIPTIONS 30

TMS320C40 Local Bus Memory Map 30
Table 3.1 – Local Memory Map *30*
'C40 MIX Bus Control Register 31
Table 3.2 – MIX Bus Control Register (R/W) 0x12600000 ..... *31*
Serial Flash Data Transfer (Bits D13-D15) 31
MIX Bus Reset (D0) 31
'C40 Interrupts 32
Table 3.3 – Individual Interrupt Sources *33*
Interrupt Masks 34
Table 3.4 – Interrupt Mask Register #1 IMR1 (W) 0x08000000 ..... *34*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.5 – Interrupt Mask Register #2 IMR2 (W) 0x08000001 35
Interrupt Status 35
Table 3.6 – Interrupt Status Register #1 ISR1 (R) 0x08000002 36
Table 3.7 – Interrupt Status Register #2 ISR1 (R) 0x08000003 37
Reset Control Register 37
Reset Control Logic 37
Table 3.8 – Reset Control Register RCR (W) 0x08000004 38
PCI Interface Registers 39
Table 3.9 – PCI Control Register (R/W) 0x08000005 39
Table 3.10 – PCI Status Register (R) 0x08000006 39
Table 3.11 – PCI Device ID 40
I/O Port Register 40
Table 3.12 – I/O Port (R/W) 0x08000007 40
Table 3.13 – I/O Pin Number Functions 41
TMS320C40 Global Bus Memory Map 41
Table 3.14 – Global Memory Map 41

CHAPTER FOUR—OPERATION 44

Resetting and Booting the Model 9245 44
The Power-up or Push-button Reset 44
Table 4.1 – Register Values 45
Flash Operations 45
Erasing the Flash 45
Writing Data to Flash Memory 47
Downloading Programs to the Model 9245 48
Downloading Programs with the XDS-510 Emulator 48
Downloading Applications with the DSP Load Utility 48
Procedure 49
Mastering the MIX Bus 51
Table 4.2 – MIX Module Memory Regions 51
Table 4.3 – Transfer Times for One Longword (Block Repeat Mode) - 60 MHz 'C40 52
Table 4.4 – Transfer Times for One Longword (C40 DMA Mode) - 60 MHz 'C40 52

APPENDIX A—PROGRAMMING EXAMPLES 54

Introduction 54
VMEbus Interrupts 54
Comm Ports 67
IP Modules and IP Interrupts 83
Linker Command File 99

APPENDIX B—BOOT FLASH SOURCE CODE LISTING 102

Boot Flash Source Code 102

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

End of Section

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter One



Introduction

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Using This Manual

Chapter One
provides an overview of this manual and a brief description of use and content.

Chapter Two
includes inspection procedures, jumper information, and the features of the 9245 Front Panel. It also discusses expansion modules and installation and removal from the VMEbus card cage.

Chapter Three
provides memory maps for the local and global buses and descriptions for the various registers.

Chapter Four
covers system operation and provides specifics on file loading and conversion, program downloads, interrupt handling and bus mastering.

Appendix A
provides programming examples for VMEbus Interrupts, Comm Ports, IP Modules/Interrupts and Linker Command Files.

Appendix B
provides source code listings for Boot Flash.

General Product Description

The DSP*Con* Model 9245 is a powerful platform on which to configure a custom VMEbus system. It incorporates Texas Instrument's TMS320C40 which acts as a

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

controller and performs high-speed signal processing. While powerful enough to operate as a standalone in many applications, the Model 9245's speed and flexibility are enhanced by adding a combination of PMC, IP and/or MIX Bus modules.

The Model 9245 Data Engine

Housed on a dual-sided 6U VME board, the Model 9245's data engine is a high-performance TMS320C40 DSP with Flash Memory, zero-wait-state Local and Global SRAM, EDO DRAM and shared SRAM memory. Its onboard interfaces include IP (IndustryPack), VME, and MIX (Modular Interface eXtension), 10/100 Mbit Ethernet, and RS-232 serial port.

The Model 9245's VME, PCI, IP and MIX Buses combine with its six independent internal, high-speed DMA controllers to form an unparalleled data moving engine.

In addition to its data moving capabilities, the Model 9245's 60-MFLOP DSP core processor provides for a variety of powerful signal-processing applications.

Model 9245 Interfaces

VMEbus Control

The Model 9245 has both master and slave VMEbus capabilities, provided by the TUNDRA UNIVERSE-II chipset. As a bus master, it can read from and write to the entire 32-bit address space of the VMEbus, thus accessing any external slave device. As a host controller, it can act as a bus arbiter for multiple bus masters, and supply all bus arbitration logic as well as clock and reset.

MIX Bus Support

The MIX interface supports high-speed, multi-master control and data transfer to any of the MIX modules. Mapped directly into the 'C40's local bus, this 32-bit data channel includes support for full interrupt handling and generation, to ensure optimum real-time performance.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IndustryPack (IP) Bus

The IP bus supports 8 and 32 MHz transfers and allows interface to industry standard COTS I/O buses supported by a variety of original equipment manufacturers. IP interface is implemented on a programmable logic device (PLD).

Dual PMC (accessed through PLX Technology Part Number PCI 9080) is a 132-Mbyte per second master/slave operator as well as a burst operator. Using the 9245's PMC sites the PCI bus can support any variety of third-party vendor cards – all in a single 6U VME slot. It comes equipped with 512 Kbytes of Bridge Memory that is expandable to up to 2 Mbytes of Bridge Memory.

The 9245's 'C40 interfaces to PCI devices via this chip, which provides the ability to configure and access PCI devices connected to its PCI bus. The PCI bus is arbitrated by a PLD in a "round robin" fashion.

The bridge chip also controls the PMC sites interfaces and the time at which the DSP is allowed to access the memory. Any attempt by the DSP to access SRAM while the bridge device is accessing memory is held off by the bridge chip until the memory becomes available.

Please note that a full discussion of the bridge chip can be found in PLX Technology's User Document.

TMS320C40 Comm Port Interface

The six high-speed Communications Ports of the 'C40 are brought out to convenient connectors—4 to the front and 4 to the back. Any MIX module stacked on the 9245 thus becomes Comm Port compatible. Consequently, the 9245 can act as a Comm Port-based I/O controller for other 'C40 systems.

Please note that a full discussion of the TMS320C40 can be found in the TI User's Guide.

For further information on this topic, refer to Tables 2.4 and 2.5, which describe front and rear comm port access, and the section entitled "The DSPCon Model 9745".

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RS-232 Serial Communications Port Interface

The RS-232 Serial Port Interface (+/- 10V) uses a 16550 UART (Universal Asynchronous Receiver/Transmitter) (EXAR Corporation Part Number ST16C550) capable of operation up to 115.2 Kbps.

All standard RS-232 handshaking signals are brought to the front panel.

The UART's on board status registers provides the error conditions, type and status of the transfer operation being performed; complete modem control capability and a processor interrupt system. The UART provides internal loop-back capability for on board diagnostic testing.

Further discussion of the UART can be found in *Serial Comm Port Connector, Chapter Two* of this manual, and in EXAR Corporation's User Document.

Ethernet Interface

A 10/100 Mbyte Ethernet interface is provided through the use of Ethernet MAC/PHY devices (DEC Part Number 21143-TD) off the embedded PCI bus.

Please note that a full discussion of the 21143 can be found in Digital Equipment Corporation's Data Sheet.

VME Interface

The VME Interface is implemented through a Tundra Universe II device (Tundra Semiconductor Part Number CA91C142) sitting on the embedded PCI bus. The 9245 supports full master/slave and all slot 0 functions.

Please note that a full discussion of the Universe II device can be found in the Tundra User Manual.

Please note that the VME Interface is available as Model 9245 Option 11.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MIX Interface

The Model 9245's MIX interface supports up to three industry-standard Intel interfaces and a complete line of MIX products for high-speed, multi-master control and data transfer to the MIX modules. Mapped directly into the 'C40's local bus, this 32-bit data channel includes support for full interrupt handling and generation, to ensure optimum real-time performance.

System Block Diagram

Figure 1.1 provides a block diagram for the Model 9245.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

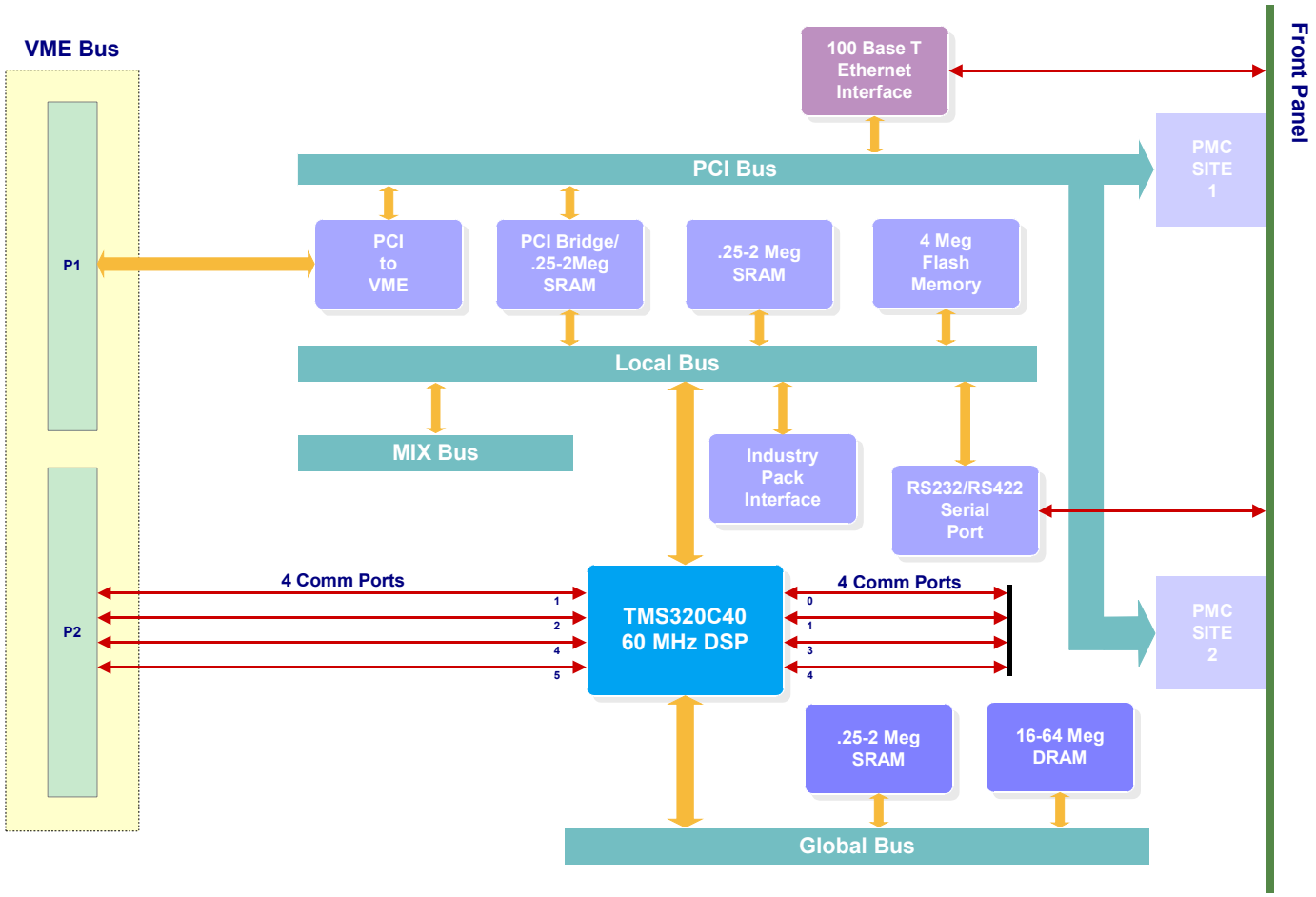


Figure 1.1 — Block Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Specifications

Table 1.1 – System Specifications	
Processor	
Type	Texas Instruments TMS320C40
Clock Speed	
Standard	60 MHz
Address Bus	32 bits
Data Bus	32 bits
VMEbus Interface	
Option 011	VME Bus master/slave access and slot zero controller
DRAM	
Size	
Standard	0 Mbytes
Option 001	16 Mbytes (4M x 32)
Option 002	64 Mbytes (16M x 32)
Option 012	32 Mbytes (8M x 32)
'C40 Access	Memory mapped on global bus, 2 wait states
Local Static RAM	
Size	
Standard	.5 Mbyte (128k x 32)
Option 003	1 Mbyte (256k x 32)
Option 004	2 Mbyte (512k x 32)
'C40 Access	Memory mapped on local bus, 0 wait states
Global Static RAM	
Size	
Standard	.5 Mbyte (128k x 32)
Option 005	1 Mbyte (256k x 32)
Option 006	2 Mbyte (512k x 32)
'C40 Access	Memory mapped on global bus, 0 wait states
PCI Shared RAM	
Size	
Standard	.5 Mbyte (128k x 32)
Option 007	1 Mbyte (256k x 32)
Option 008	2 Mbyte (512k x 32)
'C40 Access	Memory mapped on local bus, 1 wait state from 'C40; 0 wait states from bridge
Flash	
Size	
Standard	512k x 32 (2 Mbyte)
'C40 Access	Memory mapped on local bus, 2 wait states
Comm Ports†	
Option 009*	Comm Port 0 (J3), Comm Port 1 (J5) — Default Output Comm Ports
Option 010*	Comm Port 4 (J6), Comm Port 3 (J8) — Default Input Comm Ports
Option 014	Comm Port 4 available via locking connector in front panel PMC2 site
*Note: *Choosing these options may restrict access to PMC2*	†(Also see Comm Port Sections in Chapter Two)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Power Consumption

Electrical Specifications

Power Consumption

+5 volts ..... 10.5 watts
+12 volts ..... 0 watts
–12 volts ..... 0 watts

The figures for power consumption given here assume that options 004, 006, 008 and 011 are installed on the board. *(Note that these figures represent the worst-case option population for the 9245.)*

Model 9245 Peripherals

Peripherals can be accessed from the DSP at various speeds, which are dependent on their number of "wait states". (Also see Table 1.3—Peripheral Speeds).

Table 1.2– Peripheral Wait States

Peripheral	Bus	# of Wait States
SRAM	Local	0
SRAM	Global	0
DRAM	Global	1
Shared SRAM	Local	1
Interrupt Control Registers	Local	3
Serial Port Interface	Local	3
Flash Chip	Local	3
IP Interface	Local	*
MIX Interface	Local	*
PCI Bridge	Local	*

* Wait states are dependent on device operation & programming.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Local and Global Static RAM

Depending on the options selected, up to 2 Mbytes of SRAM may be provided for the 'C40 on the local bus and global buses. These resources maximize the use of the 'C40's dual-bus architecture and its ability to conduct data and program cycles in parallel on the two buses. Both the local and global SRAMs operate with zero-wait state performance.

The DSP shares access to SRAM via the bridge chip. This memory is arbitrated by glue logic implemented in the PCI bridge memory glue logic PLD.

Dynamic RAM

Up to 64 Mbytes of DRAM provides an extremely powerful structure for passing data and programs between the 'C40 and the VMEbus.

From the 'C40, this resource is mapped directly onto the global bus. The Dynamic RAM works with 2 wait-state performance.

Flash Chip

The Model 9245 is equipped with up to 2 Mbytes of Flash memory organized as 512K by 8 bytes for a total of 4-megabit program storage. The Flash Chip (AMD Part #AM29F040B) is mapped as a 'C40 Local Bus resource, and code that is programmed into it may be executed at boot time. Data or application programs can be stored in Flash memory, thus providing true standalone embedded processor operation.

Please note that a full discussion of the Flash Chip can be found in the Advanced Micro Devices User Document.

Voltage Regulators & Reset Circuitry

Onboard devices can be reset by software control, VME bus or manually via the front panel.

A majority of the 9245 circuitry uses the 5-volt supply provided by the VME interface connectors. The +12 and –12 voltages are provided to the PMC and

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

MIX Bus sites and to the IP interface cards for boards that require these voltages. All supply voltages are fuse-protected against over-current conditions.

A reset chip monitors the 5-volt supply and provides a power-up reset pulse to all devices. With the addition of a VMEbus Interface (Option 011), onboard devices may be reset by software control (see *Reset Control Register, Chapter Three*).

Peripheral Speeds

Table 1.3 lists a number of the peripherals that exist on the various TMS320C40 buses and gives their maximum transfer speed from the TMS320C40.

Table 1.3 – Peripheral Speeds

Peripheral	Speed *
SRAM (Local)	60 Mbytes/sec
SRAM (Global)	60 Mbytes/sec
DRAM	24 Mbytes/sec
Shared SRAM	40 Mbytes/sec
Interrupt Control Registers	24 Mbytes/sec
Serial Port Interface	24 Mbytes/sec
Flash Chip	24Mbytes/sec

* Assumes a 32-bit data transfer

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

End of Section

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter Two



Installation and Connections

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introduction

This section contains information and instructions for the configuration and various modes of operation of the Model 9245. These are achieved by setting jumper blocks on the board, before it is installed in the VMEbus card cage. The Model 9245 supports up several expansion modules, which may be attached to its various connectors. Prior to attaching any expansion modules you should successfully install the Model 9245 by itself. This makes access to the jumper blocks and diagnosis of installation problems easier.

Inspection

The operator should conduct a thorough examination of the unit before attempting to put it into use.

Unpacking the Unit

After unpacking the unit, inspect it carefully for any possible damage to connectors or components. Should you discover any damage, report it to DSP*Con* immediately. Full information for reaching DSP*Con* can be found on the first page of this manual.



It is important that you save the original container and all packing material, in the event that return shipment is required.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Figure 2.1 — Circuit Board Showing Jumper Block Locations

Jumper Block Locations

A drawing of the Model 9245's circuit board, showing the location of jumper blocks and connectors referred to in the sections below, is shown in Figure 2.1.

Table 2.1 shows the pin numbering for double-row jumper blocks used in the Model 9245.

Table 2.1 – Jumper Block Pin Numbering

Jumper #	Reserve Status	Jumper Placement
J2	Reserved by the factory	No jumpers should be placed here
J9	Reserved by the factory	No jumpers should be placed here
J10	Reserved by the factory	No jumpers should be placed here
J11	Reserved by the factory	No jumpers should be placed here
J12	Reserved by the factory	No jumpers should be placed here
J15	Reserved by the factory	No jumpers should be placed here
J16	Reserved by the factory	No jumpers should be placed here
J17	*Please refer to the RS-232 Serial Port section for information*	
J18	Reserved by the factory	No jumpers should be placed here
J19	*Please refer to the RS-232 Serial Port section for information*	
J20	Reserved by the factory	No jumpers should be placed here
J21	Reserved by the factory	No jumpers should be placed here
J28	Reserved by the factory	No jumpers should be placed here
J29	Reserved by the factory	The factory jumper should be left here
J30	Reserved by the factory	The factory jumper should be left here
J31	Reserved by the factory	No jumpers should be placed here

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Connectors

PMC Module (PCI Mezzanine) Connectors

The 32-bit PCI bus requires two 64-pin connectors, Pn1/Jn1 and Pn2/Jn2. The 64-bit PCI bus requires three 64 pin connectors, Pn1/Jn1, Pn2/Jn2 and Pn3/Jn3. When I/O is brought off the host's backplane, either the Pn4/Jn4 connector or both Pn4/Jn4 and Pn5/Jn5 connectors are used.

Any combination of connectors may be used on the mezzanine card as well as on the host.

PMC Module 1

9245 connectors PMC1J1 and PMC1J2 provide the connector interface between the 9245 and a PMC module in position 1.

PMC Module 2

9245 connectors PMC2J1 and PMC2J2 provide the connector interface between the 9245 and a PMC module in position 2.

IP Module Connectors

Connectors J22 and J23 provide the connector interface between the Model 9245 and an IP module. IP Module I/O is performed through the header J24. The mating connectors available for header J24 are:

- Samtec FFSD-25-01-N (no attached ribbon cable)
- Samtec FFSD-25-S-24.00-01-N (with attached 24-inch ribbon cable)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2.2 – Pin Numbering IP Module			
Pin Number	Pin Name	Pin Number	Pin Name
49	IPSIG49	50	IPSIG50
47	IPSIG47	48	IPSIG48
45	IPSIG45	46	IPSIG46
43	IPSIG43	44	IPSIG44
41	IPSIG41	42	IPSIG42
39	IPSIG39	40	IPSIG40
37	IPSIG37	38	IPSIG38
35	IPSIG35	36	IPSIG36
33	IPSIG33	34	IPSIG34
31	IPSIG31	32	IPSIG32
29	IPSIG29	30	IPSIG30
27	IPSIG27	28	IPSIG28
25	IPSIG25	26	IPSIG26
23	IPSIG23	24	IPSIG24
21	IPSIG21	22	IPSIG22
19	IPSIG19	20	IPSIG20
17	IPSIG17	18	IPSIG18
15	IPSIG15	16	IPSIG16
13	IPSIG13	14	IPSIG14
11	IPSIG11	12	IPSIG12
9	IPSIG9	10	IPSIG10
7	IPSIG7	8	IPSIG8
5	IPSIG5	6	IPSIG6
3	IPSIG3	4	IPSIG4
1	IPSIG1	2	IPSIG2

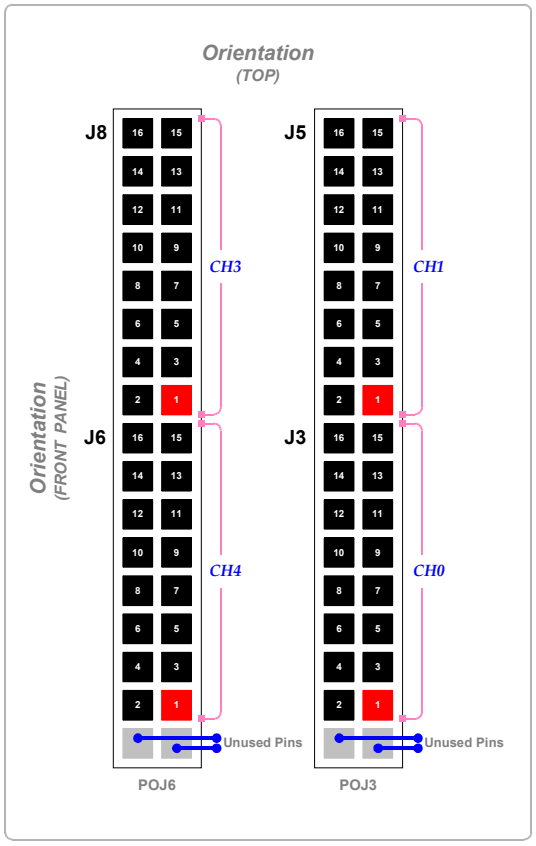
Comm Port Connectors

The six high-speed Communications Ports of the 'C40 are brought out to convenient connectors— 4 to the front (Comm Ports 0, 1, 3, 4) and 4 to the rear (Comm Ports 1, 2, 4, 5). It is important to note that, because of this overlap

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

capability, when the user designates that a Comm Port be used from the front, it cannot concurrently be used from the rear. The reverse is also true.

Optional Comm Port Headers



Comm Ports 0, 1, 3 and 4 on the Model 9245 go to headers J3, 5, 6 and 8. Please note that these headers are accessible to the user ONLY on the condition that no PMC Module is installed in PMC Slot #2. In the illustration below, Headers J3 and J5 (Comm Ports 0 & 1) are available as Option 009, and Headers J6 and J8 (Comm Ports 3 & 4) are available as Option 010.

Table 2.3 –Pin Numbering Optional Comm Ports 0, 1, 3, 4

Pin Number	Pin Name	Pin Number	Pin Name
1	D0	**2**	D1
3	D2	**4**	D3
5	GND	**6**	D4
7	D5	**8**	D6
9	D7	**10**	GND
11	$\overline{\text{REQ}}$	**12**	$\overline{\text{ACK}}$
13	GND	**14**	$\overline{\text{STB}}$
15	$\overline{\text{RDY}}$	**16**	$\overline{\text{GND}}$

Figure 2.2—Comm Port Pin Numbering (Options 009/010)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

VME P2 Connector

Table 2.4—VME P2 Comm Port Pin Numbering (Rear Panel)			
Pin #s	**Pin Name**	**Pin #s**	**Pin Name**
C32	COMM PORT 1 REQ	C6	COMM PORT 4 REQ
C31	COMM PORT 1 ACK	C5	COMM PORT 4 ACK
C30	COMM PORT 1 STRB	C4	COMM PORT 4 STRB
C29	COMM PORT 1 RDY	C3	COMM PORT 4 RDY
C28	COMM PORT 1 D0	C2	COMM PORT 4 D0
C27	COMM PORT 1 D1	C1	COMM PORT 4 D1
C26	COMM PORT 1 D2	A32	COMM PORT 4 D2
C25	COMM PORT 1 D3	A34	COMM PORT 4 D3
C24	COMM PORT 1 D4	A30	COMM PORT 4 D4
C23	COMM PORT 1 D5	A29	COMM PORT 4 D5
C22	COMM PORT 1 D6	A28	COMM PORT 4 D6
C21	COMM PORT 1 D7	A27	COMM PORT 4 D7
C19	COMM PORT 2 REQ	A25	COMM PORT 5 REQ
C18	COMM PORT 2 ACK	A24	COMM PORT 5 ACK
C17	COMM PORT 2 STRB	A23	COMM PORT 5 STRB
C16	COMM PORT 2 RDY	A22	COMM PORT 5 RDY
C15	COMM PORT 2 D0	A21	COMM PORT 5 D0
C14	COMM PORT 2 D1	A20	COMM PORT 5 D1
C13	COMM PORT 2 D2	A19	COMM PORT 5 D2
C12	COMM PORT 2 D3	A18	COMM PORT 5 D3
C11	COMM PORT 2 D4	A17	COMM PORT 5 D4
C10	COMM PORT 2 D5	A16	COMM PORT 5 D5
C9	COMM PORT 2 D6	A15	COMM PORT 5 D6
C8	COMM PORT 2 D7	A14	COMM PORT 5 D7

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DSPCon Model 9745



The DSP*Con* Model 9745 is a standard product that breaks out Comm Port Headers J1, J2, J4 and J5 from the back plane to the front panel and allows the user to avoid manual setup.

Visit the DSP*Con* website or contact your local representative for more information on this product.

Emulator Connector

The Model 9245 has provisions for use with the Texas Instruments XDS510 Emulator. The emulator connector is available on the 9245 in header location J1. This connector is keyed for proper alignment with the socket connector at the end of the emulator cable. Figure 2.3 gives the pinout of the emulator connector. Refer to Texas Instruments TMS320C40 Emulator User's Guide for operating details.

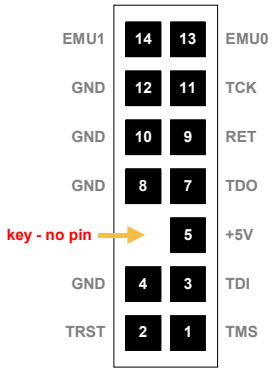
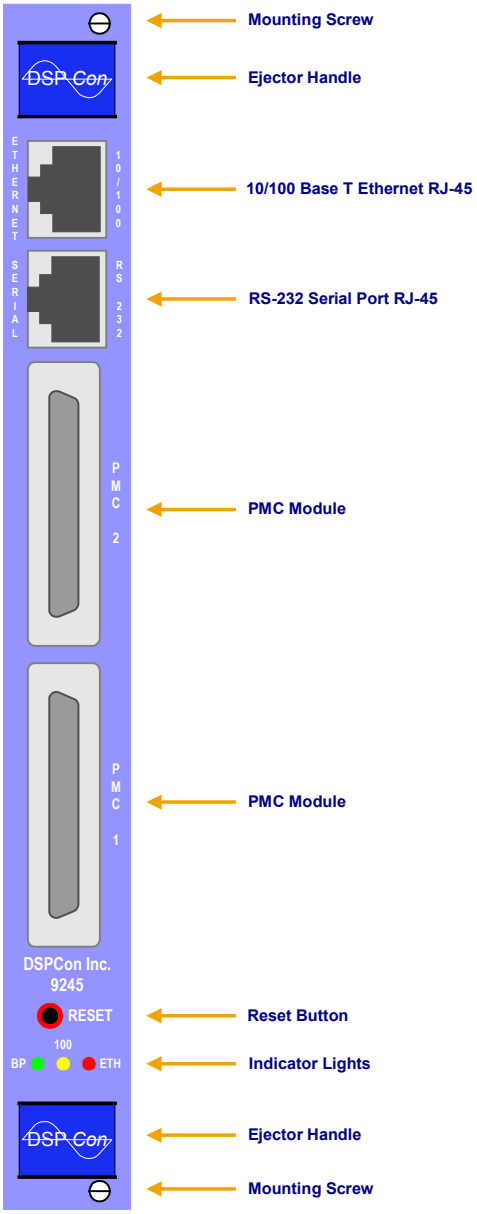


Figure 2.3—Emulator Connector Pinouts

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The Front Panel



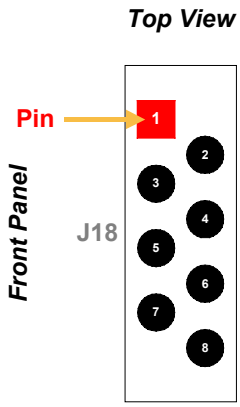
The Front Panel of the Model 9245 makes an RS-232-level Serial Port and a 10/100 Base T Ethernet Interface; two PMC Interface Sites (Modules); three LED indicator lights and a Reset button available to the user.

Each of these resources is discussed in subsequent sections of this chapter, which will give pinouts for the connectors and describe the signals they carry.

Figure 2.4—Model 9245 Front Panel Design

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Serial Port (UART) Connector



The front panel RS-232 Serial Port Connector is of type RJ-45. RS-232 connections to the VME chassis are made through connector J18. The pin-outs for these are listed in Table 2.5, below.

Please note that Pins 2, 4, 5, 7 and 8 are fixed, while Pins 4 and 6 are functions of Jumper #8.

Figure 2.5—Serial Port Pin Numbering

Table 2.5 – Serial Port Pins	
RJ-45 Pin Number	**Description**
1	J17 1-2 Selects Ring Indicator
	J17 3-4 Selects Data Carrier Detect
2	Return to Sender
3	9245 to Ground
4	Receive Data
5	Transmit Data
6	J19 1-2 Selects Data Set Ready
	J19 3-4 Selects Ground
7	Clear to Send
8	Data Terminal Ready

Ethernet Port Connector

The front panel Ethernet 10/100 connector is of type RJ-45. Table 2.6 lists the pins in this connector.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 2.6 – Ethernet Port Pins	
RJ-45 Pin Number	**Description**
1	TX High
2	TX Low
3	RX High
4	75 ohm to ground
5	75 ohm to ground
6	RX Low
7	75 ohm to ground
8	75 ohm to ground

LED Indicators

Note that is helpful to refer to Figure 2.1 when referencing this section.

D1 and D2

LEDs D1 and D2 are driven by bit 0 of the 'C40 VMEbus Interrupt Controller PLD (Interrupt Mask Register) at 'C40 address 0x80000000. Writing a "1" to this bit will turn the LED off and writing a "0" will turn it on.

D13

LED D13 is driven by bit 6 of the PCI Bridge Glue Logic PLD (PCI Control Register) at 'C40 address 0x80000005. Writing a "1" to this bit will turn the LED off and writing a "0" will turn it on.

Front Panel Indicators

Green* (*labeled* BP*) U65
Indicates wake function from the Ethernet link

Yellow* (*labeled* 100*) U65
Indicates TX link from Ethernet link

Red* (*labeled* ETH*) U65

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Indicates link activity from Ethernet link
Further information on the LED indicators, their addresses and usage can be found in the corresponding section of Chapter Three.

Reset Button

Pressing the push-button on the Model 9245's front panel will commence a reset cycle that will, upon completion, return the 9245 to its power-up state.

Reset Control Logic

The 9245 Reset Control Logic individually resets memory-mapped peripherals and system devices. Depressing the front panel reset switch resets each device either automatically on power-up or manually.

- At power-up, a reset generator resets the DSP and discrete logic contained in the CPLDs
- On pressing the front panel reset switch power-up reset is triggered.

In addition, the Reset Control Register (RCR) bits can be used to reset each device individually by pulsing the bits from a logic 0 to a logic 1 and back to a logic 0. Each bit should remain in the logic 1 position for at least 200 milliseconds. Also see the information contained in *Reset Control Register, Chapter Three*.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Expansion Modules Installation

MIX Modules Installation

Please note the following:
On the Model 9245 all MIX modules are assembled on the ***bottom*** face of the board versus typical top-face installation on other types of VME boards. Figure 2.6 illustrates this method.

Step One

Place a standard MIX connector on the 9245 board following the directional arrows on the connector to ensure proper placement. The connector will be "upside down" (see note above).

Place a stiffener plate on the opposite side of the 9245 board and line up its four pre-drilled corner holes directly in line with MIX connector. Attach both units to the 9245 board by screwing them together using the attached #6 pan head screws.

Step Two

Take a MIX Expansion Module and attach it to the MIX connector. Using a second stiffener plate, attach both to the 9245 board with the screws provided.

Please note that connectors must mate evenly. Torque all screws to 47.5 foot pounds.

PMC Module Installation

On the top face of the 9245 board, plug the PMC Module (PCI Mezzanine Card) into the PMC connector sites. Using the screws provided, screw the module securely to the board through the aligned holes.

Please note that connectors must mate evenly. Torque all screws to 47.5 foot pounds.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Industry Pack Module Installation

Attach standard standoffs to all four corners of the IP Module using **NYLON*** flat-head (flush-mount) screws. Then plug the IP Module (standoffs down) into the IP connectors on the top face of the 9245 board. Finally, attach the IP Module to the 9245 board by screwing into the aligned holes on the opposite (bottom) side of the board using the screws provided.

Please note that connectors must mate evenly. Torque all screws to 47.5 foot pounds.

Caution

Failure to use NYLON screws can cause **irreparable damage to the board.*

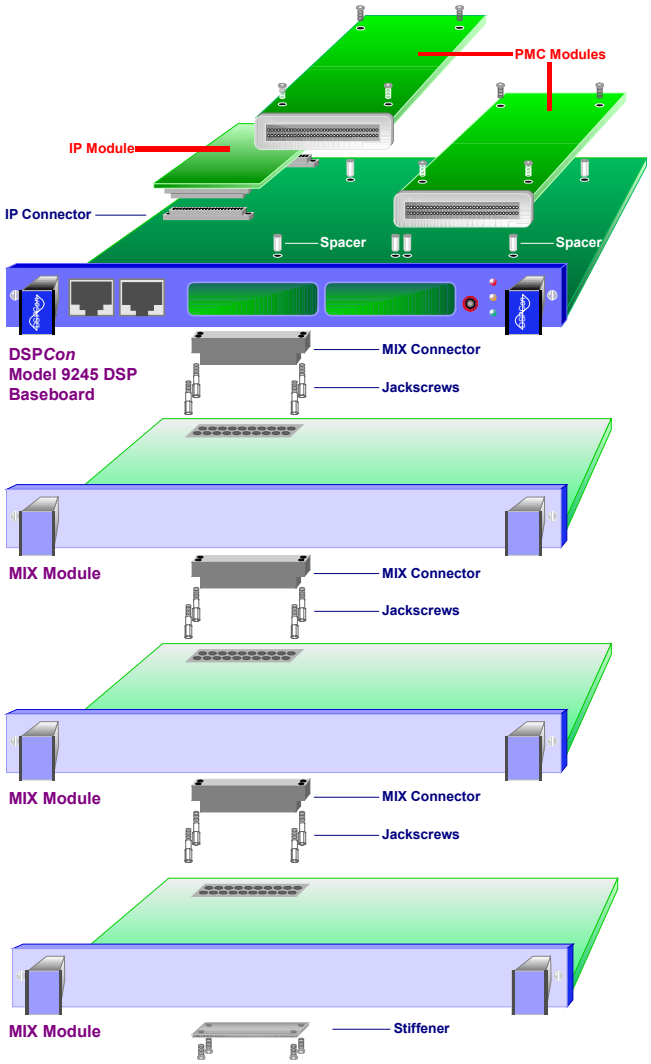


Figure 2.6—Module Installation Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Installing the Assembly in the VMEbus Card Cage



Caution

Turn off all power to the card cage before making an attempt to insert or remove any board.

After the necessary jumpers on have been set, and the installation of the various expansion modules has been completed, the assembly can be installed in the card cage. The baseboard of the Model 9245 is the only portion of the assembly that actually engages the back plane, so, when inserting and removing the assembly, be careful to use only the front panel ejector handles of the Model 9245.

After inserting the assembly into the card cage, push in on the top and bottom ejector handles of the Model 9245 to fully seat the connectors in the back plane.

Once the assembly is seated, screw the captive panel screws at the top and bottom of each front panel into the top and bottom rails of the card cage.

Removing the Assembly from the VMEbus Card Cage

Loosen all captive panel screws on the top and bottom of each front panel before attempting to remove the assembly. Then, push the ejector handles of the Model 9245 away from the center of the panel. This will eject the assembly from the cage. Once the assembly is disengaged, pull outward on the ejector handles to remove it.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter Three



Memory Maps and Register Descriptions

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TMS320C40 Local Bus Memory Map

The Model 9245's TMS320C40 has two data buses, one Local and one Global.

The Model 9245's TMS320C40 has two data buses, one called the Local Bus and the other called the Global Bus.

Table 3.1, below, maps the resources assigned to Local Memory, as seen by the 'C40 processor.

LSTRB ACTIVE FIELD BITS 24-28 =11101

LSTRB0-0000 0000-3FFF FFFF
LSTRB1-4000 0000-7FFF FFFF

Table 3.1 – Local Memory Map

LSTRB0 Device	Decode	Range	Range Used
Boot ROM (up to 512K)	ROMEN	0030 0000 – 07FF FFFF	0030 0000-0037 FFFF
Interrupt / Control / Status	INTCSN	0800 0000 – 0FFF FFFF	0800 0000-0800 00FF
UART	UARTCSN	1000 0000 – 11FF FFFF	1000 0000-1000 000F
MIX Interface	MIXCSN	1200 0000 – 13FF FFFF	1200 0000 – 127F FFFF
IP 16 Bit Interface	IPCSN	1400 0000 – 15FF FFFF	1400 0000-143F FFFF
IP 8 Lower 8 Bit Interface	IPCSN	1400 0000 – 15FF FFFF	1440 0000 – 147F FFFF
IP 8 High 8 Bit Interface	IPCSN	1400 0000 – 15FF FFFF	1480 0000 – 14BF FFFF
Bridge SRAM (up to 512K)	PCIBMCSN	1800 0000 – 1FFF FFFF	1800 0000 – 1807 FFFF
PLX-9080 Chip Registers	9080CSN	2000 0000 – 2FFF FFFF	2000 0000 – 2FFF FFFF
PCI Master Access Window *	9080CSN	3000 0000 – 3FFF FFFF	3000 0000 – 3FFF FFFF
LSTRB1 Device		**Range**	**Range Used**
SRAM (up to 512K)	LSTRB1	4000 0000-7FFF FFFF	4000 0000 – C007 FFFF

* See PAGING bits contained in 9245 PCI Control Register.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

'C40 MIX Bus Control Register

This 16-bit register is the means by which the 'C40 on the Model 9245 monitors and controls the activities of the expansion modules on the MIX Bus. Table 3.2, below, defines the register's bit structure. Brief descriptions of the functions of the bits follow the table. For more detail, refer to the MIX Bus specification. All bits in this register are cleared to the '0' state at power-up reset, and they may generally be left in that state.

Table 3.2 – MIX Bus Control Register (R/W) 0x12600000

Bit #	D15	D14	D13	D12	D11	D10	D9	D8
Bit Name	MIXSDOUT	MIXSDIN	MIXSCLK	TURBO2	TURBO1	TURBO0	MIXINTMASK2	MIXINTMASK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Bit #	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Name	MIXINTMASK0	MIXINT2	MIXINT1	MIXINT0	N/U	N/U	N/U	MXRST
Read/Write	R	R	R	R	——	——	——	R/W

Serial Flash Data Transfer (Bits D13-D15)

The three least significant bits in this register are concerned with the transfer of serial data to the Flash chip (AMD Part #AM29F040B, a full discussion which can be found in the AMD User Document) that is contained on a MIX module.

However, the D0 bit of this register is a serial data stream to be written to the module's Flash. D2 contains a serial stream read from the module's Flash, and D1 has the clock signal that controls the transfer.

MIX Bus Reset (D0)

All expansion modules on the MIX Bus are held in their power-up reset state while this bit is held high. For normal operation of the modules, this bit should be low.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

'C40 Interrupts

The Model 9245 contains interrupt control logic that groups a number of interrupt sources and maps them to a particular 'C40 interrupt pin. A block of general-purpose registers is mapped in memory space to control interrupt masks and monitor interrupt status. Each interrupt source is individually maskable by writing to the Interrupt Mask Registers (IMR1 and IMR2) in the general purpose I/O block. The status of each interrupt may also be monitored (polled mode) by reading the Interrupt Status Register (ISR).

At a functional level, each DSP interrupt is derived from a gating of the following sources:

INT0 ..... PMC Site # 1 and PMC Site #2 interrupt sources
INT1 .....VME Universe II and 9080 PCI bridge interrupt sources
INT2 .....100 Base T Ethernet + 46550 UART interrupt sources
INT3 .....IP + Mix Interface interrupt sources

Each interrupt is broken down by individual interrupt sources as follows:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.3 – Individual Interrupt Sources		
Interrupt	**Source**	**Operation**
INT0 gated interrupt from→	PMC1INTA#	PMC Site #1 Interrupt
	PMC1INTB#	PMC Site #1 Interrupt
	PMC1INTC#	PMC Site #1 Interrupt
	PMC1INTD#	PMC Site #1 Interrupt
	PMC2INTA#	PMC Site #2 Interrupt
	PMC2INTB#	PMC Site #2 Interrupt
	PMC2INTC#	PMC Site #2 Interrupt
	PMC2INTD#	PMC Site #2 Interrupt
INT1 gated interrupt from→	LINT7N	Universe II Interrupt
	LINT6N	Universe II Interrupt
	LINT5N	Universe II Interrupt
	LINT4N	Universe II Interrupt
	LINT3N	Universe II Interrupt
	LINT2N	Universe II Interrupt
	LINT1N	Universe II Interrupt
	LINT0N	Universe II Interrupt
	PCIB_INTA#	PCI Bridge PCI Interrupt
	PCIB_LINTIN	PCI Bridge Local Bus Interrupt
INT2 gated interrupt from→	EBTINTA#	10/100 BASE T INTERRUPT
	UARTINT	46550 INTERRUPT
INT3 gated interrupt from→	IPINTREQ1N	(I) IP INTERRUPT
	IPINTREQ0N	(I) IP INTERRUPT
	GBLMIXINT	MIX INTERRUPT
NMI Non-maskable Interrupt	NOT USED	—

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Interrupt Masks

By setting the IMR bit to a logic 1, the user can mask each individual interrupt source.

Table 3.4 – Interrupt Mask Register #1 IMR1 (W) 0x08000000		
Bit Number	**Interrupt Mask Bit**	**Operation**
15	MPMC1INTA	Interrupt mask bit for PMC1INTA#
14	MPMC1INTB	Interrupt mask bit for PMC1INTB#
13	MPMC1INTC	Interrupt mask bit for PMC1INTC#
12	MPMC1INTD	Interrupt mask bit for PMC1INTD#
11	MPMC2INTA	Interrupt mask bit for PMC2INTA#
10	MPMC2INTB	Interrupt mask bit for PMC2INTB#
9	MPMC2INTC	Interrupt mask bit for PMC2INTC#
8	MPMC2INTD	Interrupt mask bit for PMC2INTD#
7	MPCIBINTA	Interrupt mask bit for PCIB_INTA#
6	MPCIBLINTI	Interrupt mask bit for PCIB_LINTIN
5	MEBTINTA	Interrupt mask bit for EBTINTA#
4	MGBLMIXINT	Interrupt mask bit for GBLMIXINT
3	MUARTINT	Interrupt mask bit for UARTINT
2	MIPINT1	Interrupt mask bit for IPINTREQ1N
1	MIPINT0	Interrupt mask bit for IPINTREQ0N
0	LED	LED – LED D2 is controlled by this bit 1 = OFF, 0 = ON

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.5 – Interrupt Mask Register #2 IMR2 (W) 0x08000001		
Bit Number	**Interrupt Mask Bit**	**Operation**
15	NU	——
14	NU	——
13	NU	——
12	NU	——
11	NU	——
10	NU	——
9	NU	——
8	NU	——
7	MLINT7	Interrupt mask bit for LINT7N
6	MLINT6	Interrupt mask bit for LINT6N
5	MLINT5	Interrupt mask bit for LINT5N
4	MLINT4	Interrupt mask bit for LINT4N
3	MLINT3	Interrupt mask bit for LINT3N
2	MLINT2	Interrupt mask bit for LINT2N
1	MLINT1	Interrupt mask bit for LINT1N
0	MLINT0	Interrupt mask bit for LINT0N (LED D4 is also controlled by this bit) 1 = OFF, 0 = ON

Interrupt Status

The status of each interrupt may be monitored (polled mode) by reading the Interrupt Status Registers (ISR1 and ISR2). An interrupt is pending (active) if an ISR bit is set to a logic 0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.6 – Interrupt Status Register #1 ISR1 (R) 0x08000002		
Bit Number	**Interrupt Status Bit**	**Operation**
15	SPMC1INTA	Status of PMC1INTA#
14	SPMC1INTB	Status of PMC1INTB#
13	SPMC1INTC	Status of PMC1INTC#
12	SPMC1INTD	Status of PMC1INTD#
11	SPMC2INTA	Status of PMC2INTA#
10	SPMC2INTB	Status of PMC2INTB#
9	SPMC2INTC	Status of PMC2INTC#
8	SPMC2INTD	Status of PMC2INTD#
7	SPCIBINTA	Status of PCIB_INTA#
6	SPCIBLINTIN	Status of PCIB_LINTIN#
5	SEBTINTA	Status of EBTINTA#
4	SGBLMIXINT	Status of GBLMIXINT
3	SUARTINT	Status of UARTINT
2	SIPINT1	Status of IPINTREQ1N
1	SIPINT0	Status of IPINTREQ0N
0	1	——

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.7 – Interrupt Status Register #2 ISR1 (R) 0x08000003		
Bit Number	**Interrupt Status Bit**	**Operation**
15	1	——
14	1	——
13	1	——
12	1	——
11	1	——
10	1	——
9	1	——
8	1	——
7	SLINT7	Status of LINT7N
6	SLINT6	Status of LINT6N
5	SLINT5	Status of LINT5N
4	SLINT4	Status of LINT4N
3	SLINT3	Status of LINT3N
2	SLINT2	Status of LINT2N
1	SLINT1	Status of LINT1N
0	SLINT0	Status of LINT0N

Reset Control Register

Reset Control Logic

The Model 9245 contains reset control logic that can individually reset memory-mapped peripherals and system devices. A reset generator resets the DSP and discrete logic contained in CPLDs at power-up. The front panel reset switch may also be used to trigger the power-up reset. Each device is reset on power-up or by depressing the front panel reset switch.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Pulsing the RCR bits from a logic 0 to a logic 1 and back to a logic 0 resets each device. Each bit should remain in the logic 1 position for at least 200 milliseconds.

The Reset Control Register (RCR) bits can individually reset each device.

Table 3.8 – Reset Control Register RCR (W) 0x08000004

Bit Number	Reset Control Bit	Operation
7	IPRST	IP interface reset
6	UARTRIST	UART reset
5	PMC1RST	PMC Site #1 reset
4	PMC2RST	PMC Site #2 reset
3	EBTRST	100 base T Ethernet reset
2	MIXRST	Mix interface reset
1	PCIBRST	9080 PCI Bridge reset
0	UNIVPWR	Universe II VME Power-up Reset

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PCI Interface Registers

Table 3.9 – PCI Control Register (R/W) 0x08000005		
Bit Number	**PCI Control Bit**	**Operation**
7	PCI PAGE bit	Manipulates the upper most address line on the PCI Bus
6	LED D13	LED D13 – 0 = LED ON, 1 = LED OFF
5	GPIO User IN	General Purpose Output to the PCI Bridge
4	BRQ Norm=0, Force=1	Bus Request Normally Set to a 0, Set to 1 to force Bus Ownership (Not Suggested)
3	LBE3	Local Byte Enable Bits (Default Active Low)
2	LBE2	Local Byte Enable Bits (Default Active Low)
1	LBE1	Local Byte Enable Bits (Default Active Low)
0	LBE0	Local Byte Enable Bits (Default Active Low)

Table 3.10 – PCI Status Register (R) 0x08000006		
Bit Number	**PCI Status Bit**	**Operation**
7	NU	——
6	NU	——
5	NU	——
4	NU	——
3	GPIO User Out LED D12	General Purpose Input controlled by the 9080 GP output. Connected to D12. 0 = LED ON, 1 = LED OFF
2	DMPAF Half Full FIFO	Direct Master Path Almost Full
1	LSEPRN Error Bit	Bridge Error Bit
0	BUS REQ OUT	Status of BUS ownership

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.11 – PCI Device ID		
Signal	**Address Bit**	**Device ID**
Arbiter	A16	5
PMC 1	A17	6
PMC 2	A18	7
Ethernet	A22	14
PCI Bridge	A28	17
Universe	A30	19

I/O Port Register

This I/O port is located on the motherboard J27 microheader.

Table 3.12 – I/O Port (R/W) 0x08000007	
Bit Number	**Operation**
7	Input 3
6	Input 2
5	Input 1
4	Input 0
3	Output 3
2	Output 2
1	Output 1
0	Output 0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 3.13 – I/O Pin Number Functions	
Pin Number	**Function**
15	Output 0
13	Output 1
14	Output 2
9	Output 3
7	Input 0
5	Input 1
3	Input 2
1	Input 3
2,4,6,8,10,12,14,16	GND

TMS320C40 Global Bus Memory Map

The Model 9245's TMS320C40 has two data buses, one called the Local Bus and the other called the Global Bus.

Table 3.14, below, maps the resources assigned to Global Memory, as seen by the 'C40 processor.

GSTRB ACTIVE FIELD BITS 24-28 =11101

GSTRB0 - 8000 0000 - BFFF FFFF
GSTRB1 - C000 0000 - FFFF FFFF

Table 3.14 – Global Memory Map			
GSTRB0 Device	**Decode**	**Range**	**Range Used**
DRAM (up to 64M)	GDRMCSN	8000 0000 - BFFF FFFF	8000 0000 - 83E7 FFFF
GSTRB1 Device	**Decode**	**Range**	**Range Used**
SRAM (up to 512K)	GSTRB1	C000 0000 - FFFF FFFF	C000 0000 - C007 FFFF

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

End of Section

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Chapter Four



Operation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Resetting and Booting the Model 9245

There are three distinct paths by which the 'C40 processor on the Model 9245 may be reset. These are

- the Power-up/Push-button Reset
- the VME System Reset (w/ Option 011)
- the Host Control Reset

There are subtle differences in the results of each of these resets, which will be described in the sections below.

The Power-up or Push-button Reset

When power is initially applied to the Model 9245, the RESET input on the TMS320C40 is held low for a brief period of time by an RC network. This activation of the RESET input places the 'C40 processor in the state described in the Texas Instruments TMS320C4x User's Guide.

Pressing the Reset button on the Model 9245's front panel initiates a reset cycle identical to the power-up reset described above.

The Flash Boot Code initialization routines force all devices to be reset, and initializes the UART device for 9600-N-8-1 operation. If application code was *automatically* loaded from flash, all other devices are reset. If application code was *manually* loaded from flash, or loaded via 'dspload', then the PCI bridge and network interfaces are initialized. However, no assumption about the various devices current operating states can be made; these devices should be reset prior to use. The boot code initializes the DSP on-chip registers to the following values.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4.1 – Register Values		
Description	**Mnemonic**	**Initialization Values in Hexadecimals**
Local Bus Control Register	LMCR	0x3D864040
Global Bus Control Register	GMCR	0x3D864040
Interrupt Enable Register	IIE	0
Interrupt Flags	IIF	0
Global Interrupt Enable	GIE	0

Flash Operations

The Flash Operations support non-volatile storage of programs for boot loading. Other data, such as short tables, may also be stored here. This resource is mapped into the 'C40's Local Bus memory map, beginning at address **0x0030 0000**.

Any data to be stored in this memory region must be in Intel HEX format. A program for converting COFF program files into this format, hex30, is provided by Texas Instruments as a part of their Optimizing 'C' Compiler package.

On the Model 9245, the user's goal is to have the processor boot load the code in the Flash immediately upon power-up or warm reset. The procedures involved in accomplishing that goal will involve all of the operations one might want to perform using the Flash memory. We will therefore describe the procedures for loading bootable HEX code into the Flash, and boot loading into that code, in the sections below.

Erasing the Flash

The Flash memory is erased through software. The example below outlines the steps for erasing a sector of flash memory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EXAMPLE

```
/** flash part initialization **/
* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0xF0;

/** erase flash sector **/
volatile unsigned *sector = 0x00300000;             /** sector 0 **/

* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0x80;
* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;

/** give erase command and wait for verification **/
*sector = 0x30;
while ((*sector & 0x80) == 0);

/** close flash -- we're done **/
* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0xF0;
```



Note

If you have purchased the Model 9245 Board Support Package, you may refer to the ***flash_erase_sector*** *in the Flash Device Support section of the "Model 9245 Application Programmers Guide".*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Writing Data to Flash Memory

Data is written to the Flash through software. The example below outlines the steps for this process.

EXAMPLE

```
/** flash part initialization **/
* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0xF0;

/** write byte to flash **/
volatile unsigned * flash_loc = 0x00300000;  /** base of flash **/
unsigned char val = 0xAB;                    /** value to be written **/

* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0xA0;
* flash_loc = val;

/** strip of msb and verify it was written **/
val &= 0x80;
while (val != (* flash_loc & 0x80));

/** close flash -- we're done **/
* (volatile unsigned *) 0x00305555 = 0xAA;
* (volatile unsigned *) 0x00302AAA = 0x55;
* (volatile unsigned *) 0x00305555 = 0xF0;
```



Note

If you have purchased the Model 9245 Board Support Package, you may use the Flash Device Support Routines documented in the ***Flash Device Support*** *section of the "Model 9245 Application Programmers Guide" to create a flash-bootable program.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Downloading Programs to the Model 9245

Downloading Programs with the XDS-510 Emulator

Users of the Texas Instruments XDS-510 Emulator can transfer 'C40 COFF programs into the 9245 via the front panel JTAG connector. For that code to be properly executed, however, the 9245's register set must be put into its boot state. ***The user must do this***, because the operating environment of the emulator prevents the 9245 from executing its boot code at power-up.

There are also two registers internal to the 'C40 which must be initialized before you try to access the external registers. The Global Memory Interface Control Register, at address 0x0010 0000, must be set to 0x3D9E C000, and the Local Memory Interface Control Register, at address 0x0010 0004, must be set to 0x3DEF 4740. You may either load these values from the emulator's command interface, or place commands to initialize the registers at the beginning of any program you may load. For more information about the use of these registers, see the Texas Instruments TMS320C4x User's Guide.

One method of initializing the external register set is to issue the **RUN** command from the emulator, and then push the reset button, before loading your code. Another method is to initialize the registers explicitly at the beginning of your code, after setting up the internal registers.

Downloading Applications with the DSP Load Utility

The operational firmware is programmed from a standard PC via the network interface, using the *dspload.exe* utility program. Consequently, the 9245 to be programmed and the PC with the executable image to be burned must be connected via Ethernet.

In order to prepare the 9245 to accept a new, executable image via Ethernet, the board's normal boot-up procedure must be interrupted. To accomplish this, connect a terminal to the serial port on the front panel of the 9245 and send a sequence of character commands (see "Procedure" below) via the serial line.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Note

*The DSP Load Utility is included in the Model 9245 package. It is also available in downloadable form on the DSPCon website – **www.dspcon.com**.*

Procedure

- Attach the serial interface cable both to the PC and to the serial interface jack on the target 9245.
- Insert the board into the card cage and attach the network cable to the network interface jack on the target 9245.
- Start a HyperTerm window (a standard application available on most PCs) to communicate with the 9245 via the serial port on the PC. Configure the HyperTerm comm port connection to following settings:

  Bits per seconds..... 9600
  Data bits..... 8
  Parity..... none
  Stop bits..... 1
  Flow control..... none

  Configure HyperTerm ASCII setup to: "Echo typed characters locally".

  Note: If changes are made to the HyperTerm setting, save the configuration and restart the HyperTerm program.

- While pressing the Enter key in the HyperTerm window, reset the target 9245. A startup banner message should appear in the HyperTerm window, followed by a prompt character of the form ">". This indicates that the 9245 is running its boot monitor program. Repeatedly pressing the Enter key should produce additional prompt characters.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Note

The parameters that follow are maintained by the 9245 boot monitor for the use of application firmware, but the boot monitor itself does NOT use these parameters.

- Press the "**c**" bootstrap monitor command to change boot parameters and then click Enter.

IP Address..... an available network address (e.g., 10.44.2.72)
IP Net Mask 255.255.255.0
Def Gateway[1] ..... 000.000.000.000 (default)
VME Mem Size[2] ..... 0x000000 (default)
VME Mem Addr[3] ..... off (default)
VME Reg Addr[4] ..... off (default)
Processor Id[5] ..... 0 (default)
Flash Boot ..... Y (for yes)

[1]**Def Gateway**
This is the IP address of a default network gateway. Application firmware may employ this value in configuring network access.

[2]**VME Mem Size**
This is the size of the VME slave memory window, expressed in bytes. Application firmware may employ this value in configuring the VME slave memory.

[3]**VME Mem Addr**
This is the address of the VME slave memory window, exposed in VME space. This address may be in A16, A24, or A32 space and is specified as ASS:XXXX where SS is either 16, 24, or 32, and XXXX is the address expressed in hexadecimal notation. Application firmware may employ this value in configuring the VME slave memory.

[4]**VME Reg Addr**
This is the address of the VME controller (TUNDRA UNIVERSE) registers, exposed in VME space. This address may be in A16, A24, or A32 space and is specified as ASS:XXXX where SS is either 16, 24, or 32, and XXXX is the address expressed in hexadecimal notation. Application firmware may employ this value in configuring the VME slave memory.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

[5]**Processor Id**
This is a parameter that may be employed by application firmware in an application specific manner to uniquely identify multiple 9245 units within a system.

- Start a standard MS-DOS window, and use the "***cd dirName***" command to move to the directory that contains the executable image to be loaded.
- Use the "***dspload xx.xx.xx.xx fln***" command to load the firmware image file, substituting the network IP address of the target 9245 for the *xx.xx.xx.xx* string and the file name of the executable image to be loaded for the fln string. (Note that fln typically has the following form (e.g., *f_yyyy.x45*).
- Verify that the HyperTerm window displays progress messages during programming operation; operation should take less than 10 seconds to complete. Upon successful completion, the HyperTerm displays a final message (the word "success"), which indicates successful operation.
- Remove power from the target board, and detach all cables.

Mastering the MIX Bus

The 'C40 on the Model 9245 accesses MIX modules from its Local bus. Each module position on the MIX stack is assigned an 8M-Longword region of 'C40 memory space (see Table 4.2, below). The region between 0x12000000 and 0x125FFFFF is referred to as the 'C40's MIX Bus window.

Table 4.2 — MIX Module Memory Regions

Name	Range
Module 0	0x12000000 to 0x121FFFFF
Module 1	0x12200000 to 0x123FFFFF
Module 2	0x12400000 to 0x125FFFFF

Mastering MIX transactions from the 9245 is simply a matter of reading from or writing to the proper area of the 'C40 MIX Bus window. The 9245 also supports mastering of the MIX Bus by processor modules in the MIX stack, called Upper MIX Bus Masters, or UMBMs. The 9245 does not support MIX Bus mastering by standard MIX modules.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 4.3 — Transfer Times for One Longword (Block Repeat Mode) - 60 MHz 'C40							
From	To →	C40 Int. RAM	Local SRAM	Global SRAM	Global DRAM*	MIX (wait)	VMEbus*
C40 Int. RAM		33 nsec	66 nsec	66 nsec	99 nsec	266 nsec	133 nsec
Local SRAM		33 nsec	99 nsec	66 nsec	99 nsec	299 nsec	166 nsec
Global SRAM		33 nsec	66 nsec	99 nsec	166 nsec	266 nsec	133 nsec
Global DRAM*		66 nsec	99 nsec	133 nsec	199 nsec	266 nsec	133 nsec
MIX (no wait)		233 nsec	300 nsec	233 nsec	233 nsec	500 nsec	466 nsec
VMEbus **		400 nsec	433 nsec	400 nsec	400 nsec	666 nsec	566 nsec
* DRAM timing includes refresh time of 120 nsec every 1 msec - VMEbus transfers are all to or from DRAM ** For high-speed VMEbus transfers DMA is suggested. All timing results are given as average for 1 longword transfer based on 4000 longword transfers							
VME Timing							
DS to DTACK (assertion) 150 nsec		DTACK to DS ↑ (response) 10 nsec		DS ↑ to DTACK ↑ (de-assertion) 20 nsec (max)			

Table 4.4 — Transfer Times for One Longword (C40 DMA Mode) - 60 MHz 'C40							
From	To →	C40 Int. RAM	Local SRAM	Global SRAM	Global DRAM*	MIX (wait)	VMEbus*
C40 Int. RAM		33 nsec	66 nsec	66 nsec	99 nsec	266 nsec	133 nsec
Local SRAM		33 nsec	99 nsec	66 nsec	99 nsec	299 nsec	166 nsec
Global SRAM		33 nsec	66 nsec	99 nsec	166 nsec	266 nsec	133 nsec
Global DRAM*		66 nsec	99 nsec	133 nsec	199 nsec	266 nsec	133 nsec
MIX (no wait)		233 nsec	300 nsec	233 nsec	233 nsec	500 nsec	466 nsec
VMEbus **		400 nsec	433 nsec	400 nsec	400 nsec	666 nsec	566 nsec
* DRAM timing includes refresh time of 120 nsec every 1 msec - VMEbus transfers are all to or from DRAM ** For high-speed VMEbus transfers DMA is suggested. All timing results are given as average for 1 longword transfer based on 4000 longword transfers							
VME Timing							
DS to DTACK (assertion) 150 nsec		DTACK to DS ↑ (response) 10 nsec		DS ↑ to DTACK ↑ (de-assertion) 20 nsec (max)			

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix A



Programming Examples

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Introduction

This section provides programming examples for:

- Issuing and handling VMEbus Interrupts
- Using the Comm Ports
- Working with IP Modules and IP Interrupts
- Linker Command File

VMEbus Interrupts

```
/*****************************************************************************
  DSPCON SCSI TEST MONITOR

               DSPCon, Inc.
               380 FootHill Road
               Bridgewater, NJ 08807

               e-mail: info@dspcon.com

  (C) Copyright 2000 DSPCon, Inc.  All rights reserved.
  Use of copyright notice is precautionary and does not imply publication
*****************************************************************************/

/*****************************************************************************
  T_MON.H
  $Revision:   1.0  $
*****************************************************************************/

#ifndef t_mon_DEFINED
#define t_mon_DEFINED


typedef (*CmdFxn) (int argc, char** argv);

typedef struct CmdInfo
{ /*-------------------------------------------------------------------------*/
  char* cmd;                        // command string
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  CmdFxn fxn;                       // command routine

  char* help;                       // command help text
} /*---------------------------------------------------------------------*/
CmdInfo;


typedef struct CmdModule
{ /*---------------------------------------------------------------------*/
  struct CmdModule* next;           // next command module

  char* name;                       // command module name
  CmdInfo* cmdVec;                  // command vector
  int cmdCnt;                       // command count
} /*---------------------------------------------------------------------*/
CmdModule;


extern struct CmdModule* init_app (void);


#endif /** DEFINED **/


/*****************************************************************************
  T_MON.C
  $Revision:   1.0  $
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <spoxknl.h>

#include <ctype.h>
#include <stdio.h>

#include "t_mon.h"



/*===========================================================================
 *
 *                   CONSOLE MONITOR ROUTINES
 *
============================================================================*/

/**
*** getline
**/

static
char* getline (char* buffer, int maxchar)
{
  char* ptr;
  int ch;

  // get line of input
  fgets (ptr = buffer, maxchar, stdin);
  putchar ('\n');

  // skip leading white space
  while (ch = *ptr,  ch && isspace(ch))  ++ptr;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // return pointer
  return (ptr);
}


/*==========================================================================
 *
 *                  THE DEFAULT COMMAND TABLE
 *
==========================================================================*/

static void cmd_help ();
static void cmd_dump ();
static void cmd_modify ();
static void cmd_trace ();


#define  DEF_CMD_COUNT  (sizeof(defCmdTab) / sizeof(CmdInfo))

static CmdInfo defCmdTab [] = {
  { "?",       (CmdFxn) cmd_help,     "?                    display this help text" },
  { "d",       (CmdFxn) cmd_dump,     "d ADDR LEN           display memory or register" },
  { "m",       (CmdFxn) cmd_modify,   "m ADDR VAL           modify memory or register" },
  { "t",       (CmdFxn) cmd_trace,    "t ADDR LEN           display trace buffer" },
};


static CmdModule defModule = {
  0,                  // next
  "DEFAULT COMMANDS",  // module name
  defCmdTab,           // module command table
  DEF_CMD_COUNT               // module command count
};


/*==========================================================================
 *
 *                   DEFAULT COMMAND ROUTINES
 *
==========================================================================*/

/**
*** dump
**/

void cmd_dump (int nargs, char **argv)
{
  static unsigned *addr = 0;
  static unsigned size = 0x10;

  if (nargs == 2) {
    if (strcmp (argv[1],"die") == 0) {
      printf ("die: %08X\n", MX40_getDIE());
      return;
    }
    else if (strcmp (argv[1],"iie") == 0) {
      printf ("iie: %08X\n", MX40_getIIE());
      return;
    }
    else if (strcmp (argv[1],"iif") == 0) {
      printf ("iif: %08X\n", MX40_getIIF());
      return;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    }
    else if (strcmp (argv[1],"st") == 0) {
      printf ("st: %08X\n", MX40_getST());
      return;
    }
  }


  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);


  { unsigned* ptr = addr;
    int i,j;

    if ((unsigned) ptr < 0x100000)
      ptr += 0xC0000000;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++)
        printf("%08X  ", *ptr++);

      putchar ('\n');
    }
  }
}


/**
*** help -- display help text
**/

void cmd_help (int argc, char** argv)
{
  CmdModule* module = &defModule;
  int k;

  do {
    puts (module->name);

    for (k=0; k < module->cmdCnt; k++) {
      CmdInfo* cinfo = module->cmdVec + k;
      puts (cinfo->help);
    }

    putchar ('\n');
  } while (module = module->next,  module);
}


/**
*** modify
**/

void cmd_modify (int nargs, char **argv)
{
  unsigned val;

  if (nargs < 3)
    return;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  val = (unsigned) strtoul (argv[2], 0, 0);

  if (strcmp (argv[1],"die") == 0) {
    printf ("die: %08X\n", MX40_getDIE());
  }
  else if (strcmp (argv[1],"iie") == 0) {
    printf ("iie: %08X\n", MX40_getIIE());
  }
  else if (strcmp (argv[1],"iif") == 0) {
    printf ("iif: %08X\n", MX40_getIIF());
  }
  else if (strcmp (argv[1],"st") == 0) {
    printf ("st: %08X\n", MX40_getST());
  }
  else {
    unsigned* addr;

    addr = (unsigned*) strtoul (argv[1], 0, 16);
    *addr = val;

    printf ("%08X:  %08X\n", addr, val);
  }
}


/**
*** trace
**/

void cmd_trace (int nargs, char **argv)
{
  extern int PUTCBEG;         // defined typcially in linker command file
  extern int PUTCEND;         // defined typically in linker command file

  #define LNKCMD_VAL(x)              ((unsigned) &(x))

  static unsigned* addr = 0;
  static unsigned size = 0;

  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  if (addr == 0  ||  size == 0) {
    addr = (unsigned*) LNKCMD_VAL(PUTCBEG);
    size = LNKCMD_VAL(PUTCEND) - LNKCMD_VAL(PUTCBEG) +1;
  }

  { unsigned* ptr = addr;
    int cntr = size;
    int ch;

    while (cntr--  &&  (ch = *ptr++,  ch != 0))  putchar (ch);
    putchar ('\n');
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=========================================================================
 *
 *                    THE COMMAND PROCESSOR
 *
=========================================================================*/


void do_cmd_proc (void)
{
#define HIST_SIZE  10
#define LINE_SIZE  79

  while (1) {
    /*-----------------------------------------------------------------------
      perform console monitor forever
    -----------------------------------------------------------------------*/
    static char buffer [HIST_SIZE+1][LINE_SIZE+1];
    static int curline = 0;

    CmdModule* cmodule;
    CmdInfo* cinfo;

    char argbuf [LINE_SIZE+1];
    char* argp;

    char* argvec [10];
    int argcnt;

    char* ptr;
    int ch;
    int k;

    // display prompt
    fputs ("> ", stdout);
    fflush (stdout);

    // get line of input
    ptr = getline (buffer[curline], LINE_SIZE);
    if (ch = *ptr,  ch == 0)  continue;

    // handle line history references
    if (ch == '!') {
      int hindx = strtol (ptr+1, 0, 0);
      int kline;

      if ((unsigned) hindx > HIST_SIZE) continue;
      if (hindx == 0)  hindx = 1;

      kline = curline - hindx;
      if (kline < 0)  kline += HIST_SIZE+1;

      ptr = strcpy (buffer[curline], buffer[kline]);
      while (ch = *ptr,  ch && isspace(ch))  ++ptr;

      if (ch == 0)  continue;
      printf ("!%d > %s", hindx, ptr);
    }

    // parse input into word list
    argp = argbuf;
    argcnt = 0;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    do {
      argvec [argcnt++] = argp;

      // skip over non-space characters
      while (ch != 0  &&  !isspace(ch))  *argp++ = ch,  ch = *++ptr;
      *argp++ = 0;

      // skip over trailing space characters
      while (ch != 0  &&  isspace(ch))  ch = *++ptr;
    } while (ch != 0);

    argvec [argcnt] = 0;

    // locate command entry
    cinfo = 0;

    for (cmodule = &defModule;  !cinfo && cmodule;  cmodule = cmodule->next) {
      for (k=0;  !cinfo  &&  k < cmodule->cmdCnt;  k++) {
        cinfo = cmodule->cmdVec + k;

        if (strcmp (cinfo->cmd, argvec[0]) != 0)
          cinfo = 0;
      }
    }

    // execute command
    if (cinfo) {
      cinfo->fxn (argcnt, argvec);
      if (++curline == HIST_SIZE+1)  curline = 0;
    }
    else {
      printf ("unknown command\n");
    }
  }
}


/*============================================================================
 *
 *                    THE MAIN ENTRY POINT
 *
============================================================================*/


void main (void)
{
  { /*-------------------------------------------------------------------
      create console interface
    -------------------------------------------------------------------*/
    freopen ("/hblock:" "/uart9245,9600,8,n,1", "r", stdin);
    freopen ("/uart9245,9600,8,n,1", "w", stdout);

    TSK_setpri (TSK_self(), 12);
  }


  { /*-------------------------------------------------------------------
      initialize operational environment and start command processor
    -------------------------------------------------------------------*/
    defModule.next = init_app ();
    do_cmd_proc ();
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*****************************************************************************
  T_VME.C
  $Revision:   1.0  $
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <spoxknl.h>

#include <ctype.h>
#include <stdio.h>

#include <9245/pcilib.h>
#include <9245/vmelib.h>

#include "hw.h"

#include "t_mon.h"



/*===========================================================================
 *
 *                        GLOBAL VARIABLES
 *
===========================================================================*/


/*===========================================================================
 *
 *                    INTERRUPT SERVICE ROUTINES
 *
===========================================================================*/

unsigned vme_service_cnt = 0;


/*===========================================================================
  vme_service

  VME interrupt service routine
===========================================================================*/

void vme_service (void)
{
  unsigned stat;

  // acknowledge interrupts and retrieve status
  stat = vme_ackLocIntr ();

  // increment count if MBOX0
  if (stat & BIT(vme_LocIntr_MBOX0))  ++vme_service_cnt;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=======================================================================
 *
 *                  APPLICATION COMMAND ROUTINES
 *
=======================================================================*/

/**
*** vme_peek16
**/

void cmd_peek16 (int nargs, char **argv)
{
  static unsigned addr = 0;
  static unsigned size = 0x10;

  if (nargs >= 2)  addr = (unsigned) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned ptr = addr;
    int i,j;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++) {
        printf("%08X  ", vme_peekWord ((void*) ptr, 0x2D));
        ptr += 4;
      }

      putchar ('\n');
    }
  }
}


/**
*** vme_peek24
**/

void cmd_peek24 (int nargs, char **argv)
{
  static unsigned addr = 0;
  static unsigned size = 0x10;

  if (nargs >= 2)  addr = (unsigned) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned ptr = addr;
    int i,j;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++) {
        printf("%08X  ", vme_peekWord ((void*) ptr, 0x3D));
        ptr += 4;
      }

      putchar ('\n');
    }
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/**
*** vme_peek32
**/

void cmd_peek32 (int nargs, char **argv)
{
  static unsigned addr = 0;
  static unsigned size = 0x10;

  if (nargs >= 2)  addr = (unsigned) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned ptr = addr;
    int i,j;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++) {
        printf("%08X  ", vme_peekWord ((void*) ptr,0x0D));
        ptr += 4;
      }

      putchar ('\n');
    }
  }
}


/**
*** vme_poke16
**/

void cmd_poke16 (int nargs, char **argv)
{
  unsigned addr;
  unsigned val;

  if (nargs < 3)
    return;

  addr = (unsigned) strtoul (argv[1], 0, 16);
  val = (unsigned) strtoul (argv[2], 0, 0);

  vme_pokeWord ((void*) addr, 0x2D, val);
  printf ("%08X:  %08X\n", addr, val);
}


/**
*** vme_poke24
**/

void cmd_poke24 (int nargs, char **argv)
{
  unsigned addr;
  unsigned val;

  if (nargs < 3)
    return;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  addr = (unsigned) strtoul (argv[1], 0, 16);
  val = (unsigned) strtoul (argv[2], 0, 0);

  vme_pokeWord ((void*) addr, 0x3D, val);
  printf ("%08X:  %08X\n", addr, val);
}


/**
*** vme_poke32
**/

void cmd_poke32 (int nargs, char **argv)
{
  unsigned addr;
  unsigned val;

  if (nargs < 3)
    return;

  addr = (unsigned) strtoul (argv[1], 0, 16);
  val = (unsigned) strtoul (argv[2], 0, 0);

  vme_pokeWord ((void*) addr, 0x0D, val);
  printf ("%08X:  %08X\n", addr, val);
}


/**
*** cmd_i
**/

void cmd_i (int nargs, char **argv)
{
  vme_genBusIntr ();
}


/**
*** cmd_s
**/

void cmd_s (int nargs, char **argv)
{
  printf ("%u\n", vme_service_cnt);
}


/*=========================================================================
 *
 *                  APPLICATION COMMAND TABLE
 *
=========================================================================*/

#define  APP_CMD_COUNT  (sizeof(appCmdTab) / sizeof(CmdInfo))

static CmdInfo appCmdTab [] = {
  { "peek16",  (CmdFxn) cmd_peek16,   "peek16 [addr [nwords]]  perform A16D32 peek" },
  { "peek24",  (CmdFxn) cmd_peek24,   "peek24 [addr [nwords]]  perform A24D32 peek" },
  { "peek32",  (CmdFxn) cmd_peek32,   "peek32 [addr [nwords]]  perform A32D32 peek" },
  { "poke16",  (CmdFxn) cmd_poke16,   "poke16 addr value       perform A16D32 poke" },
  { "poke24",  (CmdFxn) cmd_poke24,   "poke24 addr value       perform A24D32 poke" },
  { "poke32",  (CmdFxn) cmd_poke32,   "poke32 addr value       perform A32D32 poke" },
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  { "i",       (CmdFxn) cmd_i,              "i                        generate interrupt" },
  { "s",       (CmdFxn) cmd_s,              "s                        show interrupts" },
};


static CmdModule appModule = {
  0,                  // next
  "APP COMMANDS",     // module name
  appCmdTab,          // module command table
  APP_CMD_COUNT               // module command count
};


/*============================================================================
 *
 *                  APPLICATION INITIALIZATION
 *
============================================================================*/


CmdModule* init_app (void)
{
  { /*-------------------------------------------------------------------
      display startup banner
    -------------------------------------------------------------------*/
    printf ("\n"
            "\n"
            "DSPCon VME TEST MONITOR\n"
            "\n");
  }


  { /*-------------------------------------------------------------------
      initialize application context
    -------------------------------------------------------------------*/

    // wait for UNIVERSE-II to complete reset
    TSK_sleep (1000);

    // initialize PCI bus interface
    pci_init ();

    // initialize MIX/IP bus interface
    hw_init ();


    /*
     * configure VME master and slave access
     */


    // initialize VME bus interface
    vme_init (0);

    // map UNIVERSE-II control registers into A16
    vme_mapSlaveReg (VME_A16_SLAVE_REG, 0xC000);

    // map shared memory in A24
    // vme_mapSlaveWin (0, VME_A24_SLAVE_MEM, 0x0C0000, pci_address(0), 0x010000);
    vme_mapSlaveWin (0, VME_A32_SLAVE_MEM, 0x01000000, pci_address(0), 0x020000);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /*
     * configure VME interrupt generation toward VME
     */


    // configure status/ID for interrupt generation toward VME
    vme_cfgBusIntr (0xC0);

    // map SW-INT interrupt toward VME on desired LEVEL
    vme_mapBusIntr (vme_BusIntr_SW_INT, 4);


    /*
     * configure VME/MBOX interrupt generation toward DSP
     */


    // map MBOX0 interrupt toward PCI on local INT0
    vme_mapLocIntr (vme_LocIntr_MBOX0, 0);

    // enable PCI local INT0
    vme_enableLocIntr (BIT (vme_LocIntr_MBOX0));

    // enable 9245 UNIVERSE-II local INT0
    D9245_IMR2 &= ~D9245_INT_LINT0;

    // define UNIVERSE-II properties as ISR #1
    hwISR_tab[1] .iieMask = 0;
    hwISR_tab[1] .iifMask = IIOF_INT_ENB (1);
    hwISR_tab[1] .fxn = (HW_ISR) vme_service;
    hwISR_tab[1] .arg = 0;

    // bind ISR #1 to UNIVERSE-II interrupt handler
    MX40_plugIVT (IIOF_INT_VEC (1), hwISR_fxn[1]);

    // set external interrupt enabled
    MX40_enableIIF (IIOF_INTR_LEVEL (1) | IIOF_INT_ENB (1));
  }


  { /*-------------------------------------------------------------------------
      return handle to our cmd modules
    -------------------------------------------------------------------------*/
    return &appModule;
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Comm Ports

```
/*****************************************************************************
  T_MON.C
  $Revision:   1.0  $
*****************************************************************************/

#ifndef t_mon_DEFINED
#define t_mon_DEFINED


typedef (*CmdFxn) (int argc, char** argv);

typedef struct CmdInfo
{ /*-------------------------------------------------------------------------*/
  char* cmd;                        // command string
  CmdFxn fxn;                       // command routine

  char* help;                       // command help text
} /*-------------------------------------------------------------------------*/
CmdInfo;


typedef struct CmdModule
{ /*-------------------------------------------------------------------------*/
  struct CmdModule* next;           // next command module

  char* name;                       // command module name
  CmdInfo* cmdVec;                  // command vector
  int cmdCnt;                       // command count
} /*-------------------------------------------------------------------------*/
CmdModule;


extern struct CmdModule* init_app (void);


#endif /** DEFINED **/

/*****************************************************************************
  T_MON.C
  $Revision:   1.0  $
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <spoxknl.h>

#include <ctype.h>
#include <stdio.h>

#include "t_mon.h"
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=========================================================================
 *
 *                   CONSOLE MONITOR ROUTINES
 *
=========================================================================*/

/**
*** getline
**/

static
char* getline (char* buffer, int maxchar)
{
  char* ptr;
  int ch;

  // get line of input
  fgets (ptr = buffer, maxchar, stdin);
  putchar ('\n');

  // skip leading white space
  while (ch = *ptr,  ch && isspace(ch))  ++ptr;

  // return pointer
  return (ptr);
}


/*=========================================================================
 *
 *                  THE DEFAULT COMMAND TABLE
 *
=========================================================================*/

static void cmd_help ();
static void cmd_dump ();
static void cmd_modify ();
static void cmd_trace ();


#define  DEF_CMD_COUNT  (sizeof(defCmdTab) / sizeof(CmdInfo))

static CmdInfo defCmdTab [] = {
  { "?",       (CmdFxn) cmd_help,     "?                  display this help text" },
  { "d",       (CmdFxn) cmd_dump,     "d ADDR LEN         display memory or register" },
  { "m",       (CmdFxn) cmd_modify,   "m ADDR VAL         modify memory or register" },
  { "t",       (CmdFxn) cmd_trace,    "t ADDR LEN         display trace buffer" },
};


static CmdModule defModule = {
  0,                   // next
  "DEFAULT COMMANDS",  // module name
  defCmdTab,           // module command table
  DEF_CMD_COUNT               // module command count
};
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=========================================================================
 *
 *                    DEFAULT COMMAND ROUTINES
 *
=========================================================================*/

/**
*** dump
**/

void cmd_dump (int nargs, char **argv)
{
  static unsigned *addr = 0;
  static unsigned size = 0x10;

  if (nargs == 2) {
    if (strcmp (argv[1],"die") == 0) {
      printf ("die: %08X\n", MX40_getDIE());
      return;
    }
    else if (strcmp (argv[1],"iie") == 0) {
      printf ("iie: %08X\n", MX40_getIIE());
      return;
    }
    else if (strcmp (argv[1],"iif") == 0) {
      printf ("iif: %08X\n", MX40_getIIF());
      return;
    }
    else if (strcmp (argv[1],"st") == 0) {
      printf ("st: %08X\n", MX40_getST());
      return;
    }
  }


  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);


  { unsigned* ptr = addr;
    int i,j;

    if ((unsigned) ptr < 0x100000)
      ptr += 0xC0000000;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++)
        printf("%08X  ", *ptr++);

      putchar ('\n');
    }
  }
}


/**
*** help -- display help text
**/

void cmd_help (int argc, char** argv)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
{
  CmdModule* module = &defModule;
  int k;

  do {
    puts (module->name);

    for (k=0; k < module->cmdCnt; k++) {
      CmdInfo* cinfo = module->cmdVec + k;
      puts (cinfo->help);
    }

    putchar ('\n');
  } while (module = module->next,  module);
}


/**
*** modify
**/

void cmd_modify (int nargs, char **argv)
{
  unsigned val;

  if (nargs < 3)
    return;

  val = (unsigned) strtoul (argv[2], 0, 0);

  if (strcmp (argv[1],"die") == 0) {
    printf ("die: %08X\n", MX40_getDIE());
  }
  else if (strcmp (argv[1],"iie") == 0) {
    printf ("iie: %08X\n", MX40_getIIE());
  }
  else if (strcmp (argv[1],"iif") == 0) {
    printf ("iif: %08X\n", MX40_getIIF());
  }
  else if (strcmp (argv[1],"st") == 0) {
    printf ("st: %08X\n", MX40_getST());
  }
  else {
    unsigned* addr;

    addr = (unsigned*) strtoul (argv[1], 0, 16);
    *addr = val;

    printf ("%08X:  %08X\n", addr, val);
  }
}


/**
*** trace
**/

void cmd_trace (int nargs, char **argv)
{
  static unsigned* addr = (unsigned*) 0xC007FF00;
  static unsigned size = 256;

  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned* ptr = addr;
    int cntr = size;
    int ch;

    while (cntr--  &&  (ch = *ptr++,  ch != 0))  putchar (ch);
    putchar ('\n');
  }
}


/*=========================================================================
 *
 *                    THE COMMAND PROCESSOR
 *
=========================================================================*/


void do_cmd_proc (void)
{
#define HIST_SIZE  10
#define LINE_SIZE  79

  while (1) {
    /*-----------------------------------------------------------------
      perform console monitor forever
    -----------------------------------------------------------------*/
    static char buffer [HIST_SIZE+1][LINE_SIZE+1];
    static int curline = 0;

    CmdModule* cmodule;
    CmdInfo* cinfo;

    char argbuf [LINE_SIZE+1];
    char* argp;

    char* argvec [10];
    int argcnt;

    char* ptr;
    int ch;
    int k;

    // display prompt
    fputs ("> ", stdout);
    fflush (stdout);

    // get line of input
    ptr = getline (buffer[curline], LINE_SIZE);
    if (ch = *ptr,  ch == 0)  continue;

    // handle line history references
    if (ch == '!') {
      int hindx = strtol (ptr+1, 0, 0);
      int kline;

      if ((unsigned) hindx > HIST_SIZE) continue;
      if (hindx == 0)  hindx = 1;

      kline = curline - hindx;
      if (kline < 0)  kline += HIST_SIZE+1;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
      ptr = strcpy (buffer[curline], buffer[kline]);
      while (ch = *ptr,  ch && isspace(ch))  ++ptr;

      if (ch == 0)  continue;
      printf ("!%d > %s", hindx, ptr);
    }

    // parse input into word list
    argp = argbuf;
    argcnt = 0;

    do {
      argvec [argcnt++] = argp;

      // skip over non-space characters
      while (ch != 0  &&  !isspace(ch))  *argp++ = ch,  ch = *++ptr;
      *argp++ = 0;

      // skip over trailing space characters
      while (ch != 0  &&  isspace(ch))  ch = *++ptr;
    } while (ch != 0);

    argvec [argcnt] = 0;

    // locate command entry
    cinfo = 0;

    for (cmodule = &defModule;  !cinfo && cmodule;  cmodule = cmodule->next) {
      for (k=0;  !cinfo  &&  k < cmodule->cmdCnt;  k++) {
        cinfo = cmodule->cmdVec + k;

        if (strcmp (cinfo->cmd, argvec[0]) != 0)
          cinfo = 0;
      }
    }

    // execute command
    if (cinfo) {
      cinfo->fxn (argcnt, argvec);
      if (++curline == HIST_SIZE+1)  curline = 0;
    }
    else {
      printf ("unknown command\n");
    }
  }
}


/*=========================================================================
 *
 *                    THE MAIN ENTRY POINT
 *
=========================================================================*/


void main (void)
{
  { /*-------------------------------------------------------------------
      create console interface
    -------------------------------------------------------------------*/
    freopen ("/hblock:" "/uart9245,9600,8,n,1", "r", stdin);
    freopen ("/uart9245,9600,8,n,1", "w", stdout);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    TSK_setpri (TSK_self(), 12);
  }


  { /*-------------------------------------------------------------------
      initialize operational environment and start command processor
    -------------------------------------------------------------------*/
    defModule.next = init_app ();
    do_cmd_proc ();
  }
}

/*****************************************************************************
  T_COMM.C
  $Revision:   1.0  $
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <spoxknl.h>

#include <ctype.h>
#include <stdio.h>

#include <c40/c40_comm.h>
#include <c40/c40_dma.h>
#include <c40/c40_tmr.h>

#include "t_mon.h"


/*===========================================================================
 *
 *                       GLOBAL VARIABLES
 *
===========================================================================*/

#define RX_BUF_BASE           (0xC0000000)
#define TX_BUF_BASE           (0xC0010000)

#define IO_BUF_SIZE           (0x10000)
#define IO_DEF_SIZE           (0x80)


#define dmaRX_CTRL            (DMA_AUTO_NOWAIT | DMA_SYNC_SRC | DMA_SHARE | DMA_TCINT)
#define dmaTX_CTRL            (DMA_AUTO_NOWAIT | DMA_SYNC_DST | DMA_SHARE | DMA_TCINT)


enum PATN {
  PATN_RAMP,
  PATN_SIZE
};


volatile DmaRegs* rx_dma;
DmaRegs rx_autoA, rx_autoB;
int rx_comm = 3;

volatile DmaRegs* tx_dma;
DmaRegs tx_autoA, tx_autoB;
int tx_comm = 0;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
volatile unsigned rx_cntr, rx_count;
unsigned rx_buf_size = IO_DEF_SIZE;

volatile unsigned tx_cntr, tx_count;
unsigned tx_buf_size = IO_DEF_SIZE;

unsigned tx_buf_patn = PATN_RAMP;


/*=============================================================================
 *
 *                    TIMER SERVICE ROUTINE
 *
=============================================================================*/

#define tmr_service c_int90

static
void tmr_service (void)
{
  if (rx_dma) {
    unsigned count = rx_buf_size - rx_dma -> xferCnt;
    rx_count = rx_cntr + count;
    rx_cntr = -count;
  }
  else {
    rx_count = rx_cntr = 0;
  }

  if (tx_dma) {
    unsigned count = tx_buf_size - tx_dma -> xferCnt;
    tx_count = tx_cntr + count;
    tx_cntr = -count;
  }
  else {
    tx_count = tx_cntr = 0;
  }
}


/*=============================================================================
 *
 *                  DMA DATA SERVICE ROUTINES
 *
=============================================================================*/

#define rx_service c_int92
#define tx_service c_int93

static
void rx_service (void)
{
  rx_cntr += rx_buf_size;
}

static
void tx_service (void)
{
  tx_cntr += tx_buf_size;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=========================================================================
 *
 *                       PRIVATE ROUTINES
 *
=========================================================================*/


/*=========================================================================
 *
 *                  APPLICATION COMMAND ROUTINES
 *
=========================================================================*/

/*=========================================================================
  cmd_show

  show comm transfer statistics
=========================================================================*/

void cmd_show (int nargs, char **argv)
{
  unsigned rx_dma_cnt;
  unsigned tx_dma_cnt;

  if (rx_dma)  rx_dma_cnt = rx_buf_size - rx_dma -> xferCnt;
  if (tx_dma)  tx_dma_cnt = tx_buf_size - tx_dma -> xferCnt;

  if (rx_dma) {
    printf (" RX[%d] word cnt: %6d of %6d at %10.6f Mbyte/sec\n",
            rx_comm, rx_dma_cnt, rx_buf_size, (float) rx_count * 4 / 1E6);
  }
  else {
    printf (" RX flow off\n");
  }

  if (tx_dma) {
    printf (" TX[%d] word cnt: %6d of %6d at %10.6f MByte/sec\n",
            tx_comm, tx_dma_cnt, tx_buf_size, (float) tx_count * 4 / 1E6);
  }
  else {
    printf (" TX flow off\n");
  }
}


/**
*** cmd_lon
**/

void cmd_lon (int nargs, char **argv)
{
  /*
  ** parse arguments
  **/

  if (nargs >= 2) {
    rx_comm = (unsigned) strtoul (argv[1], 0, 0);
    if (rx_comm > 5)  rx_comm = 3;
  }

  if (nargs >= 3) {
    tx_comm = (unsigned) strtoul (argv[2], 0, 0);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  if (tx_comm > 5)  tx_comm = 0;
}

if (nargs >= 4) {
  unsigned size;

  size = (unsigned) strtoul (argv[3], 0, 0);
  if (size > IO_BUF_SIZE)  size = IO_BUF_SIZE;

  rx_buf_size = tx_buf_size = size;
}

/*
** stop existing RX DMA if necessary
*/

if (rx_dma) {
  rx_dma -> control = dmaRX_CTRL | DMA_STOP;
  rx_dma = 0;
}

/*
** stop existing TX DMA if necessary
*/

if (tx_dma) {
  tx_dma -> control = dmaTX_CTRL | DMA_STOP;
  tx_dma = 0;
}

/*
** configure RX DMA auto-init
*/

rx_autoA.control = dmaRX_CTRL | DMA_START;

rx_autoA.srcAddr = (int*) COMM_REG_ADDR (rx_comm) + COMM_IN_REG;
rx_autoA.srcIndx = 0;
rx_autoA.dstAddr = (int*) RX_BUF_BASE + 0;
rx_autoA.dstIndx = 1;
rx_autoA.xferCnt = rx_buf_size;

rx_autoA.linkPtr = (void*) &rx_autoB;

rx_autoB.control = dmaRX_CTRL | DMA_START;

rx_autoB.srcAddr = (int*) COMM_REG_ADDR (rx_comm) + COMM_IN_REG;
rx_autoB.srcIndx = 0;
rx_autoB.dstAddr = (int*) RX_BUF_BASE + rx_buf_size;
rx_autoB.dstIndx = 1;
rx_autoB.xferCnt = rx_buf_size;

rx_autoB.linkPtr = (void*) &rx_autoA;

/*
** configure RX DMA
*/

rx_dma = (volatile DmaRegs*) DMA_REG_ADDR (rx_comm);

MX40_disableDIE (DMA_DIE_RD_MASK (rx_comm));
MX40_enableDIE (DMA_DIE_RD_COMM (rx_comm));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  MX40_plugIVT (DMA_INT_VEC (rx_comm), rx_service);
  MX40_enableIIE (DMA_INT_ENB (rx_comm));

  /*
  ** configure TX DMA auto-init
  */

  tx_autoA.control = dmaTX_CTRL | DMA_START;

  tx_autoA.srcAddr = (int*) RX_BUF_BASE + tx_buf_size;
  tx_autoA.srcIndx = 1;
  tx_autoA.dstAddr = (int*) COMM_REG_ADDR (tx_comm) + COMM_OUT_REG;
  tx_autoA.dstIndx = 0;
  tx_autoA.xferCnt = tx_buf_size;

  tx_autoA.linkPtr = (void*) &tx_autoB;

  tx_autoB.control = dmaTX_CTRL | DMA_START;

  tx_autoB.srcAddr = (int*) RX_BUF_BASE + 0;
  tx_autoB.srcIndx = 1;
  tx_autoB.dstAddr = (int*) COMM_REG_ADDR (tx_comm) + COMM_OUT_REG;
  tx_autoB.dstIndx = 0;
  tx_autoB.xferCnt = tx_buf_size;

  tx_autoB.linkPtr = (void*) &tx_autoA;

  /*
  ** configure TX DMA
  */

  tx_dma = (volatile DmaRegs*) DMA_REG_ADDR (tx_comm);

  MX40_disableDIE (DMA_DIE_WR_MASK (tx_comm));
  MX40_enableDIE (DMA_DIE_WR_COMM (tx_comm));

  MX40_plugIVT (DMA_INT_VEC (tx_comm), tx_service);
  MX40_enableIIE (DMA_INT_ENB (tx_comm));

  /*
  ** start RX DMA
  */

  rx_dma -> control = dmaRX_CTRL | DMA_RESET;
  rx_dma -> xferCnt = 0;
  rx_dma -> linkPtr = (void*) &rx_autoA;

  rx_dma -> control = dmaRX_CTRL | DMA_START;

  /*
  ** start TX DMA
  */

  tx_dma -> control = dmaTX_CTRL | DMA_RESET;
  tx_dma -> xferCnt = 0;
  tx_dma -> linkPtr = (void*) &tx_autoA;

  tx_dma -> control = dmaTX_CTRL | DMA_START;
}


/**
*** cmd_loff
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
**/

void cmd_loff (int nargs, char **argv)
{
  if (tx_dma) {
    tx_dma -> control = dmaTX_CTRL | DMA_STOP;
    tx_dma = 0;
  }

  if (rx_dma) {
    rx_dma -> control = dmaRX_CTRL | DMA_STOP;
    rx_dma = 0;
  }
}


/**
*** cmd_ron
**/

void cmd_ron (int nargs, char **argv)
{
  /*
  ** parse arguments
  */

  if (nargs >= 2) {
    rx_comm = (unsigned) strtoul (argv[1], 0, 0);
    if (rx_comm > 5)  rx_comm = 3;
  }

  if (nargs >= 3) {
    rx_buf_size = (unsigned) strtoul (argv[2], 0, 0);
    if (rx_buf_size > IO_BUF_SIZE)  rx_buf_size = IO_BUF_SIZE;
  }

  /*
  ** stop existing RX DMA if necessary
  */

  if (rx_dma) {
    rx_dma -> control = dmaRX_CTRL | DMA_STOP;
    rx_dma = 0;
  }

  /*
  ** configure RX DMA auto-init
  */

  rx_autoA.control = dmaRX_CTRL | DMA_START;

  rx_autoA.srcAddr = (int*) COMM_REG_ADDR (rx_comm) + COMM_IN_REG;
  rx_autoA.srcIndx = 0;
  rx_autoA.dstAddr = (int*) RX_BUF_BASE;
  rx_autoA.dstIndx = 1;
  rx_autoA.xferCnt = rx_buf_size;

  rx_autoA.linkPtr = (void*) &rx_autoA;

  /*
  ** configure RX DMA
  */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  rx_dma = (volatile DmaRegs*) DMA_REG_ADDR (rx_comm);

  MX40_disableDIE (DMA_DIE_RD_MASK (rx_comm));
  MX40_enableDIE (DMA_DIE_RD_COMM (rx_comm));

  MX40_plugIVT (DMA_INT_VEC (rx_comm), rx_service);
  MX40_enableIIE (DMA_INT_ENB (rx_comm));

  /*
  ** start RX DMA
  */

  rx_dma -> control = dmaRX_CTRL | DMA_RESET;
  rx_dma -> xferCnt = 0;
  rx_dma -> linkPtr = (void*) &rx_autoA;

  rx_dma -> control = dmaRX_CTRL | DMA_START;
}


/**
*** cmd_roff
**/

void cmd_roff (int nargs, char **argv)
{
  if (rx_dma) {
    rx_dma -> control = dmaRX_CTRL | DMA_STOP;
    rx_dma = 0;
  }
}


/**
*** cmd_xon
**/

void cmd_xon (int nargs, char **argv)
{
  /*
  ** parse arguments
  */

  if (nargs >= 2) {
    tx_comm = (unsigned) strtoul (argv[1], 0, 0);
    if (tx_comm > 5)  tx_comm = 0;
  }

  if (nargs >= 3) {
    tx_buf_size = (unsigned) strtoul (argv[2], 0, 0);
    if (tx_buf_size > IO_BUF_SIZE)  tx_buf_size = IO_BUF_SIZE;
  }

  if (nargs >= 4) {
    unsigned patn = (unsigned) strtoul (argv[3], 0, 0);
    tx_buf_patn = (patn < PATN_SIZE) ? patn : PATN_RAMP;
  }

  /*
  ** stop existing TX DMA if necessary
  */

  if (tx_dma) {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    tx_dma -> control = dmaTX_CTRL | DMA_STOP;
    tx_dma = 0;
  }

  /*
  ** fill TX pattern
  */

  switch (tx_buf_patn) {
    int k;

  case PATN_RAMP:
    { /*
       * RAMP PATTERN
       */

      for (k=0; k < tx_buf_size; k++) {
        unsigned val = k & 0x7F;
        *(unsigned*) (TX_BUF_BASE + k) = (val << 9) | (val << 1) | 0x00000100;
      }
    }

    break;
  }

  /*
  ** configure TX DMA auto-init
  */

  tx_autoA.control = dmaTX_CTRL | DMA_START;

  tx_autoA.srcAddr = (int*) TX_BUF_BASE;
  tx_autoA.srcIndx = 1;
  tx_autoA.dstAddr = (int*) COMM_REG_ADDR (tx_comm) + COMM_OUT_REG;
  tx_autoA.dstIndx = 0;
  tx_autoA.xferCnt = tx_buf_size;

  tx_autoA.linkPtr = (void*) &tx_autoA;

  /*
  ** configure TX DMA
  */

  tx_dma = (volatile DmaRegs*) DMA_REG_ADDR (tx_comm);

  MX40_disableDIE (DMA_DIE_WR_MASK (tx_comm));
  MX40_enableDIE (DMA_DIE_WR_COMM (tx_comm));

  MX40_plugIVT (DMA_INT_VEC (tx_comm), tx_service);
  MX40_enableIIE (DMA_INT_ENB (tx_comm));

  /*
  ** start TX DMA
  */

  tx_dma -> control = dmaTX_CTRL | DMA_RESET;
  tx_dma -> xferCnt = 0;
  tx_dma -> linkPtr = (void*) &tx_autoA;

  tx_dma -> control = dmaTX_CTRL | DMA_START;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/**
*** cmd_xoff
**/

void cmd_xoff (int nargs, char **argv)
{
  if (tx_dma) {
    tx_dma -> control = dmaTX_CTRL | DMA_STOP;
    tx_dma = 0;
  }
}


/*=========================================================================
 *
 *                  APPLICATION COMMAND TABLE
 *
=========================================================================*/

#define  APP_CMD_COUNT  (sizeof(appCmdTab) / sizeof(CmdInfo))

static CmdInfo appCmdTab [] = {
  { "lon",     (CmdFxn) cmd_lon,     "lon RXPORT TXPORT SIZE  loop RXPORT to TXPORT using frames
of SIZE" },
  { "loff",    (CmdFxn) cmd_loff,    "loff                    loop mode off" },
  { "ron",     (CmdFxn) cmd_ron,     "ron  RXPORT  SIZE           recv  on  RXPORT  using  frame  of
SIZE" },
  { "roff",    (CmdFxn) cmd_roff,    "roff                    recv mode off" },
  { "xon",     (CmdFxn) cmd_xon,     "xon  TXPORT  SIZE           xmit  on  TXPORT  using  frame  of
SIZE" },
  { "xoff",    (CmdFxn) cmd_xoff,    "xoff                    xmit mode off" },
  { "s",       (CmdFxn) cmd_show,    "s                       show comm statistics" },
};


static CmdModule appModule = {
  0,                    // next
  "APP COMMANDS",       // module name
  appCmdTab,            // module command table
  APP_CMD_COUNT                 // module command count
};


/*=========================================================================
 *
 *                  APPLICATION INITIALIZATION
 *
=========================================================================*/


CmdModule* init_app (void)
{
  { /*-----------------------------------------------------------------------
      display startup banner
    -----------------------------------------------------------------------*/
    printf ("\n"
            "\n"
            "DSPCon COMM TEST MONITOR\n"
            "Version 1.000\n"
            "\n");
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  { /*-------------------------------------------------------------------
      initialize application context
    -------------------------------------------------------------------*/
    volatile TmrRegs* tmr = (volatile TmrRegs*) TMR_REG_ADDR(1);

    /*
     * clear variables
     */

    rx_count = rx_cntr = 0;
    rx_dma = 0;

    tx_count = tx_cntr = 0;
    tx_dma = 0;

    /*
     * start timer with 1000 msec period
     */

    tmr->control = TMR_FUNC_ICLK_TMR | TMR_RESET;
    tmr->period = 1000 * 15000;

    MX40_plugIVT (TMR_INT_VEC(1), tmr_service);
    MX40_enableIIE (TMR_INT_ENB(1));

    tmr->control = TMR_FUNC_ICLK_TMR | TMR_START;
  }


  { /*-------------------------------------------------------------------
      return handle to our cmd modules
    -------------------------------------------------------------------*/
    return &appModule;
  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IP Modules and IP Interrupts

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; d9810_io.s40
;;; DSP9810 DATA I/O UTILITIES
;;;
;;; $Revision:   1.0  $
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .version 40



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; WRITE_DELAY
;;
;; Macro to inject required write delay cycles
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

WRITE_DELAY  .macro

        .loop 6
          bu $+1
        .endloop

    .endm


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; void* d9810_aux_rdfifo (dst, src, length, blksize)
;;
;; Read and pack LENGTH sample pairs from even-numbered channel of
;; specified fifo into a circular buffer of size BLKSIZE.  Data is
;; packed such that first sample is in lower half-word.
;;
;; Returns pointer to next location in circular buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .def    _d9810_aux_rdfifo
        .text

_d9810_aux_rdfifo

        lda     R2, AR1                 ; prepare source pointer
        ldi     RC, BK                  ; prepare circular block size

        subi    1, R3, RC               ; prepare loop counter
        rptb    end_aux_rdfifo -1       ; start repeat block

beg_aux_rdfifo
        ldi     *AR1, R2                ; get first good sample
        sti     R2, *AR2                ; dummy write to clear bus

        ldi     *AR1, R0                ; get first waste sample
        sti     R2, *AR2                ; dummy write to clear bus

        lwl2    *AR1, R2                ; merge second good sample
        sti     R2, *AR2                ; dummy write to clear bus
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
        lwl2    *AR1, R0                ; merge second waste sample
        sti     R2, *AR2++(1)%          ; write valid data
end_aux_rdfifo

        ldi     AR2, R0                 ; return next pointer

        rets



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; void* d9810_aux_wrfifo (src, dst, length, blksize)
;;
;; Unpack and write LENGTH sample pairs to even-numbered channel of
;; specified fifo into from circular buffer of size BLKSIZE.  Data is
;; unpacked such that first sample is in lower half-word.
;;
;; Returns pointer to next location in circular buffer.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        .def    _d9810_aux_wrfifo
        .text

_d9810_aux_wrfifo

        lda     R2, AR1                 ; prepare source pointer
        ldi     RC, BK                  ; prepare circular block size

        subi    1, R3, RC               ; prepare loop counter
        rptb    end_aux_wrfifo -1       ; start repeat block

beg_aux_wrfifo
        lh0     *AR2, R2                ; get first good sample
**      LDI     55AAh, R2
        sti     R2, *AR1                ; write data to fifo
        WRITE_DELAY                     ; add delay cycles

        stik    0, *AR1                 ; write zero to fifo
        WRITE_DELAY                     ; add delay cycles

        lh1     *AR2++(1)%, R2          ; get second good sample
**      LDI     55AAh, R2
        sti     R2, *AR1                ; write data to fifo
        WRITE_DELAY                     ; add delay cycles

        stik    0, *AR1                 ; write zero to fifo
        WRITE_DELAY                     ; add delay cycles
end_aux_wrfifo

        ldi     AR2, R0                 ; return next pointer

        rets


/*****************************************************************************
.. TESTMON.C

  DSPCon 9810 TEST MONITOR
*****************************************************************************/

#include <stdlib.h>
#include <string.h>
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#include <spoxknl.h>

#include <ctype.h>
#include <stdio.h>

#include <c40/c40_tmr.h>


/*============================================================================
                          STATIC VARIABLES
============================================================================*/

#define DISP_BUF_BASE         (0xC0000000)

#define CHAN0_BUF_BASE        (0x40070000)
#define CHAN2_BUF_BASE        (0x40078000)
#define CHAN4_BUF_BASE        (0x40076000)
#define CHAN6_BUF_BASE        (0x4007C000)


static unsigned* chanBufBase [8] = {
  (unsigned*) CHAN0_BUF_BASE,
  (unsigned*) 0x4007FF00,
  (unsigned*) CHAN2_BUF_BASE,
  (unsigned*) 0x4007FF00,
  (unsigned*) CHAN4_BUF_BASE,
  (unsigned*) 0x4007FF00,
  (unsigned*) CHAN6_BUF_BASE,
  (unsigned*) 0x4007FF00,
};


/*============================================================================
 *
 *                        FRONT PANEL LED
 *
============================================================================*/

#define tmr_service c_int90

static
void tmr_service (void)
{
  *(volatile int*) 0x8000005 ^= 0x40; // toggle LED D13
}



/*============================================================================
 *
 *                    CONSOLE MONITOR ROUTINES
 *
============================================================================*/

/**
*** getline
**/

static
char* getline (char* buffer, int maxchar)
{
  char* ptr;
  int ch;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // get line of input
  fgets (ptr = buffer, maxchar, stdin);
  putchar ('\n');

  // skip leading white space
  while (ch = *ptr,  ch && isspace(ch))  ++ptr;

  // return pointer
  return (ptr);
}


/*******
** Console Command Routines
*******/

/**
*** dump
**/

void cmd_dump (int nargs, char **argv)
{
  static unsigned *addr = 0;
  static unsigned size = 0x10;

  if (nargs == 2) {
    if (strcmp (argv[1],"die") == 0) {
      printf ("die: %08X\n", MX40_getDIE());
      return;
    }
    else if (strcmp (argv[1],"iie") == 0) {
      printf ("iie: %08X\n", MX40_getIIE());
      return;
    }
    else if (strcmp (argv[1],"iif") == 0) {
      printf ("iif: %08X\n", MX40_getIIF());
      return;
    }
    else if (strcmp (argv[1],"st") == 0) {
      printf ("st: %08X\n", MX40_getST());
      return;
    }
  }


  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);


  { unsigned* ptr = addr;
    int i,j;

    if ((unsigned) ptr < 8)
      ptr = chanBufBase [(unsigned) ptr];

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++)
        printf("%08X  ", *ptr++);

      putchar ('\n');
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    }
  }
}


/**
*** help -- display help text
**/

void cmd_help (int argc, char** argv)
{
  puts ("?            help");
  puts ("d            dump system memory");
  puts ("m            modify system memory");
  puts ("t            display trace buffer");
  putchar ('\n');

  puts ("disp         display channel data");
  puts ("kill         kill aux i/o");
  puts ("run          run aux i/o");
  puts ("sel          select aux output");
  putchar ('\n');
}


/**
*** modify
**/

void cmd_modify (int nargs, char **argv)
{
  unsigned val;

  if (nargs < 3)
    return;

  val = (unsigned) strtoul (argv[2], 0, 0);

  if (strcmp (argv[1],"die") == 0) {
    printf ("die: %08X\n", MX40_getDIE());
  }
  else if (strcmp (argv[1],"iie") == 0) {
    printf ("iie: %08X\n", MX40_getIIE());
  }
  else if (strcmp (argv[1],"iif") == 0) {
    printf ("iif: %08X\n", MX40_getIIF());
  }
  else if (strcmp (argv[1],"st") == 0) {
    printf ("st: %08X\n", MX40_getST());
  }
  else {
    unsigned* addr;

    addr = (unsigned*) strtoul (argv[1], 0, 16);
    *addr = val;

    printf ("%08X:  %08X\n", addr, val);
  }
}


/**
*** trace
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
**/

void cmd_trace (int nargs, char **argv)
{
  static unsigned* addr = (unsigned*) 0xC007FF00;
  static unsigned size = 256;

  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned* ptr = addr;
    int cntr = size;
    int ch;

    while (cntr--  &&  (ch = *ptr++,  ch != 0))  putchar (ch);
    putchar ('\n');
  }
}


/**
*** aux_disp
**/

void cmd_aux_disp (int nargs, char **argv)
{
  static int* addr = 0;
  static unsigned size = 0x400;


  if (nargs >= 2)  addr = (int*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);


  { int* ptr = addr;
    int cntr = size;

    if ((unsigned) ptr < 8)
      ptr = (int*) chanBufBase [(unsigned) ptr];

    fmemcpy (DISP_BUF_BASE, ptr, cntr);
    ptr = (int*) DISP_BUF_BASE;

    while (cntr--) {
      printf ("  %d\n", ((*ptr << 16) >> 16));
      printf ("  %d\n", (*ptr++ >> 16));
    }

    putchar ('\n');
  }
}


/**
*** aux_kill
**/

void cmd_aux_kill (int nargs, char** argv)
{
  aux_kill ();
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/**
*** aux_run
**/

void cmd_aux_run (int nargs, char** argv)
{
  aux_run ();
}


/**
*** aux_sel
**/

void cmd_aux_sel (int nargs, char** argv)
{
  static int selChan = 0;

  if (nargs < 2) {
    printf ("selected %d\n", selChan);
  }

  else {
    int val;

    if (val = strtol (argv[1],0,0),  (unsigned) val >= 8) {
      printf ("invalid argument: %s\n", argv[1]);
    }
    else {
      aux_select (selChan = val);
    }
  }
}


/**
*** command table
**/

typedef (*CmdFxn) (int argc, char** argv);

typedef struct {
  const char* cmd;
  CmdFxn fxn;
} CmdInfo;


#define NUM_CMDS  (sizeof(cmdTab) / sizeof(CmdInfo))

static CmdInfo cmdTab [] = {
  { "?",              (CmdFxn) cmd_help },
  { "d",              (CmdFxn) cmd_dump },
  { "m",              (CmdFxn) cmd_modify },
  { "t",              (CmdFxn) cmd_trace },

  { "disp",           (CmdFxn) cmd_aux_disp },
  { "kill",           (CmdFxn) cmd_aux_kill },
  { "run",            (CmdFxn) cmd_aux_run },
  { "sel",            (CmdFxn) cmd_aux_sel },
};
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/**
*** main
**/

void main (void)
{
  volatile TmrRegs* tmr = (volatile TmrRegs*) TMR_REG_ADDR(1);

  { /*-------------------------------------------------------------------
      configure status/LED timer: normal blink
    -------------------------------------------------------------------*/
    tmr->control = TMR_FUNC_ICLK_TMR | TMR_RESET;
    tmr->period = 250 * 15000;

    MX40_plugIVT (TMR_INT_VEC(1), tmr_service);
    MX40_enableIIE (TMR_INT_ENB(1));

    tmr->control = TMR_FUNC_ICLK_TMR | TMR_START;
  }


  { /*-------------------------------------------------------------------
      create console interface
    -------------------------------------------------------------------*/
    freopen ("/hblock:" "/uart9245,9600,8,n,1", "r", stdin);
    freopen ("/uart9245,9600,8,n,1", "w", stdout);

    TSK_setpri (TSK_self(), 12);
  }


  { /*-------------------------------------------------------------------
      display startup banner
    -------------------------------------------------------------------*/
    puts ("\n"
          "\n"
          "DSPCon 9810 TEST MONITOR\n"
          "\n");
  }


  { /*-------------------------------------------------------------------
      initialize aux i/o module
    -------------------------------------------------------------------*/
    aux_init ();
  }


  while (1) {
    /*-------------------------------------------------------------------
      perform console monitor forever
    -------------------------------------------------------------------*/
#define HIST_SIZE  5
#define LINE_SIZE  79

    static char buffer [HIST_SIZE+1][LINE_SIZE+1];
    static int curline = 0;

    int kcmd;

    char argbuf [LINE_SIZE+1];
    char* argp;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
char* argvec [10];
int argcnt;

char* ptr;
int ch;

// display prompt
fputs ("> ", stdout);
fflush (stdout);

// get line of input
ptr = getline (buffer[curline], LINE_SIZE);
if (ch = *ptr,  ch == 0)  continue;

// handle line history references
if (ch == '!') {
  int hindx = strtol (ptr+1, 0, 0);
  int kline;

  if ((unsigned) hindx > HIST_SIZE) continue;
  if (hindx == 0)  hindx = 1;

  kline = curline - hindx;
  if (kline < 0)  kline += HIST_SIZE+1;

  ptr = strcpy (buffer[curline], buffer[kline]);
  while (ch = *ptr,  ch && isspace(ch))  ++ptr;

  if (ch == 0)  continue;
  printf ("!%d > %s", hindx, ptr);
}

// parse input into word list
argp = argbuf;
argcnt = 0;

do {
  argvec [argcnt++] = argp;

  // skip over non-space characters
  while (ch != 0  &&  !isspace(ch))  *argp++ = ch,  ch = *++ptr;
  *argp++ = 0;

  // skip over trailing space characters
  while (ch != 0  &&  isspace(ch))  ch = *++ptr;
} while (ch != 0);

argvec [argcnt] = 0;

// locate command entry
for (kcmd=0; kcmd < NUM_CMDS; kcmd++) {
  if (strcmp (cmdTab[kcmd].cmd, argvec[0]) == 0)  break;
}

// execute command
if (kcmd < NUM_CMDS) {
  cmdTab[kcmd].fxn (argcnt, argvec);
  if (++curline == HIST_SIZE+1)  curline = 0;
}
else {
  printf ("unknown command\n");
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  }
}


/*****************************************************************************
..AUXIO.C
  DSP9810 AUX I/O MODULE
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <std.h>
#include <sys.h>

#include <c40/c40_iiof.h>
#include <c40/c40_util.h>

#include <9245/d9245_hw.h>


/*===========================================================================
 *
 * BUFFER DEFINITIONS
 *
============================================================================*/

#define DATA_BASE             (0x40070000)
#define DATA_SIZE             (0x10000)

#define ESM_BUF_BASE          (DATA_BASE + 0x0000)
#define ESM_BUF_SIZE          (0x6000)

#define TCG_BUF_BASE          (DATA_BASE + 0x8000)
#define TCG_BUF_SIZE          (0x3600)

#define ICS_BUF_BASE          (DATA_BASE + 0x6000)
#define ICS_BUF_SIZE          (0x1200)

#define DIC_BUF_BASE          (DATA_BASE + 0xC000)
#define DIC_BUF_SIZE          (0x3000)



/*===========================================================================
 *
 * AUX A/D SERVICE INFO
 *
============================================================================*/

typedef struct AuxServiceInfo
{ /*-------------------------------------------------------------------------*/
  int esmSize;                /*-- ESM channel -------------------------*/
  int dicSize;                /*-- DIC channel -------------------------*/
  int tcgSize;                /*-- TCG channel -------------------------*/
  int icsSize;                /*-- ICS channel -------------------------*/
} /*-------------------------------------------------------------------------*/
AuxServiceInfo;


#define AuxSrcServTabSize 8

static AuxServiceInfo auxSrcServTab [AuxSrcServTabSize] = {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  { 128, 66, 54, 16 },
  { 128, 68, 56, 18 },
  { 128, 68, 56, 18 },
  { 128, 68, 56, 18 },
  { 128, 70, 56, 20 },
  { 128, 70, 56, 20 },
  { 128, 70, 56, 20 },
  { 128, 70, 58, 20 }
};


static unsigned auxMonSelTab [4] = {
  BIT(3), BIT(1), BIT(0), BIT(2)
};



/*============================================================================
 *
 * WORKING CONTEXT
 *
============================================================================*/

typedef struct AuxInfo
{ /*-------------------------------------------------------------------------*/
  volatile unsigned* src;     /*-- source data pointer -----------------*/
  volatile unsigned* snk;     /*-- sink data pointer -------------------*/
} /*-------------------------------------------------------------------------*/
AuxInfo;


typedef struct AuxContext
{ /*-------------------------------------------------------------------------*/
  int active;                 /*-- active flag -------------------------*/

  AuxInfo esm;                /*-- ESM channel info --------------------*/
  AuxInfo tcg;                /*-- TCG channel info --------------------*/
  AuxInfo ics;                /*-- ICS channel info --------------------*/
  AuxInfo dic;                /*-- DIC channel info --------------------*/

  int monitorId;              /*-- id of monitor source channel --------*/
  int monitorSelId;           /*-- id of selected monitor channel ------*/

  int srcServIndx;            /*-- index into source service tab -------*/
} /*-------------------------------------------------------------------------*/
AuxContext;


AuxContext aux;



/*============================================================================
 *
 * DUMP CONTEXT
 *
============================================================================*/

#define DUMP_BASE  (0x7007FF00)
#define DUMP_SIZE  (0x100)

volatile unsigned* DUMP;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*=========================================================================
 *
 * DSP9810 DEFINITIONS
 *
=========================================================================*/

#define d9810_ESM_AD   (0)
#define d9810_TCG_AD   (2)
#define d9810_ICS_AD   (4)
#define d9810_DIC_AD   (6)

#define d9810_ACO1_DA  (0)
#define d9810_ACO2_DA  (1)
#define d9810_ACO3_DA  (2)
#define d9810_ACOM_DA  (3)
#define d9810_TCG_DA   (4)
#define d9810_AUXM_DA  (6)


#define d9810_ESM_FIFO ((void*) d9810_REG_ADDR (d9810_ESM_AD >> 1))
#define d9810_TCG_FIFO ((void*) d9810_REG_ADDR (d9810_TCG_AD >> 1))
#define d9810_ICS_FIFO ((void*) d9810_REG_ADDR (d9810_ICS_AD >> 1))
#define d9810_DIC_FIFO ((void*) d9810_REG_ADDR (d9810_DIC_AD >> 1))

#define d9810_AUXM_OUT ((void*) d9810_REG_ADDR (d9810_AUXM_DA >> 1))


#define d9810_REG(n)          (* (volatile unsigned*) d9810_REG_ADDR(n))
#define d9810_REG_ADDR(n)     (0x15000000 + (n))


/**
*** peek with a four cycle delay due to function call return
**/

static
unsigned ip_peek (int indx)
{
  unsigned val = d9810_REG(indx);
  return (val);
}

/**
*** poke with a twenty-four cycle delay due to inserted nop's and return
**/

static
void ip_poke (int indx, unsigned val)
{
  d9810_REG(indx) = val;

  asm (" .loop 5  ");
  asm ("  BU $+1  ");
  asm (" .endloop ");
}



/*=========================================================================
 *
 * INTERRUPT SERVICE ROUTINE
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
 *
============================================================================*/

void* d9810_aux_rdfifo ();


#define  ip_service  c_int95

void ip_service (void)
{
  unsigned status;

  while (status = ~ip_peek(5) & 0x20,  status != 0) {

    if (status & 0x20) {
      /*-------------------------------------------------------------------
       A/D SERVICE
      -------------------------------------------------------------------*/
      AuxServiceInfo* servInfo = auxSrcServTab + aux.srcServIndx;
      volatile unsigned* ptr;

      // indicate we are here
      D9245_DIO ^= D9245_DIO_OUT0;
      ++DUMP[0x00];

      // consume ESM data
      aux.esm.src = d9810_aux_rdfifo (ptr = aux.esm.src, d9810_ESM_FIFO,
                                      servInfo->esmSize >> 1, ESM_BUF_SIZE);

      if (aux.monitorId == d9810_ESM_AD)
        d9810_aux_wrfifo (ptr, d9810_AUXM_OUT, servInfo->esmSize >> 1, ESM_BUF_SIZE);

      // consume TCG data
      aux.tcg.src = d9810_aux_rdfifo (ptr = aux.tcg.src, d9810_TCG_FIFO,
                                      servInfo->tcgSize >> 1, TCG_BUF_SIZE);

      if (aux.monitorId == d9810_TCG_AD)
        d9810_aux_wrfifo (ptr, d9810_AUXM_OUT, servInfo->tcgSize >> 1, TCG_BUF_SIZE);

      // consume ICS data
      aux.ics.src = d9810_aux_rdfifo (ptr = aux.ics.src, d9810_ICS_FIFO,
                                      servInfo->icsSize >> 1, ICS_BUF_SIZE);

      if (aux.monitorId == d9810_ICS_AD)
        d9810_aux_wrfifo (ptr, d9810_AUXM_OUT, servInfo->icsSize >> 1, ICS_BUF_SIZE);

      // consume DIC data
      aux.dic.src = d9810_aux_rdfifo (ptr = aux.dic.src, d9810_DIC_FIFO,
                                      servInfo->dicSize >> 1, DIC_BUF_SIZE);

      if (aux.monitorId == d9810_DIC_AD)
        d9810_aux_wrfifo (ptr, d9810_AUXM_OUT, servInfo->dicSize >> 1, DIC_BUF_SIZE);

      // re-select aux monitor clock if necessary
      if (aux.monitorId != aux.monitorSelId) {
        ip_poke (0xA, auxMonSelTab [aux.monitorId >> 1]);
        aux.monitorSelId = aux.monitorId;
      }

      // bump service index
      aux.srcServIndx = (aux.srcServIndx +1) & (AuxSrcServTabSize -1);
    }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  }
}


/*=========================================================================
 *
 * PRIVATE ROUTINES
 *
=========================================================================*/


/*=========================================================================
 *
 * PUBLIC ROUTINES
 *
=========================================================================*/


/*=========================================================================
  aux_init

  Initialize aux i/o module
=========================================================================*/

void aux_init (void)
{

  /*
   * set initial working context
   */

  aux.monitorSelId = aux.monitorId = d9810_ESM_AD;
  aux.active = FALSE;

  memset ((void*) DUMP_BASE, 0, DUMP_SIZE);
  DUMP = (void*) DUMP_BASE;


  /*
   * perform one-time initialization of DSP9810
   */

  // enable all A/D and D/A clocks
  ip_poke (0x8, 0x1FF);

  // select ESM clock for aux monitor
  ip_poke (0xA, 0x8);

  // assert D/A logic reset
  ip_poke (0x9, 0x01F);
  ip_poke (0x9, 0x000);

  // assert D/A chip reset
  ip_poke (0x7, 0x000);
  spox_delay (100);
  ip_poke (0x7, 0x01F);


  /*
   * bind interrupt handler
   */

  MX40_disableIIF (IIOF_INT_MASK (IIOF3));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  MX40_plugIVT (IIOF_INT_VEC (IIOF3), ip_service);

  MX40_enableIIF (IIOF_INTR_LEVEL (IIOF3));
  D9245_IMR1 &= ~D9245_INT_IP_INT0;
}


/*============================================================================
  aux_kill

  Kill aux i/o process, if already running.
============================================================================*/

void aux_kill (void)
{
  if (aux.active) {

    /*
     * indicate not active
     */

    // disable C40 interrupt
    MX40_disableIIF (IIOF_INT_ENB (IIOF3));

    // clear flag
    aux.active = FALSE;
  }
}


/*============================================================================
  aux_run

  Run aux i/o process, if not already running.
============================================================================*/

void aux_run (void)
{
  if (! aux.active) {

    /*
     * initialize working context
     */

    memset ((void*) DATA_BASE, 0x55AA55AA, DATA_SIZE);

    aux.esm.src = (void*) (ESM_BUF_BASE);
    aux.esm.snk = (void*) (ESM_BUF_BASE + 3*(ESM_BUF_SIZE >> 2));
    memset ((void*) ESM_BUF_BASE, 0, ESM_BUF_SIZE);

    aux.tcg.src = (void*) (TCG_BUF_BASE);
    aux.tcg.snk = (void*) (TCG_BUF_BASE + 3*(TCG_BUF_SIZE >> 2));
    memset ((void*) ESM_BUF_BASE, 0, ESM_BUF_SIZE);

    aux.ics.src = (void*) (ICS_BUF_BASE);
    aux.ics.snk = (void*) (ICS_BUF_BASE + 3*(ICS_BUF_SIZE >> 2));
    memset ((void*) ESM_BUF_BASE, 0, ESM_BUF_SIZE);

    aux.dic.src = (void*) (DIC_BUF_BASE);
    aux.dic.snk = (void*) (DIC_BUF_BASE + 3*(DIC_BUF_SIZE >> 2));
    memset ((void*) ESM_BUF_BASE, 0, ESM_BUF_SIZE);

    aux.srcServIndx = 0;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    /*
     * initialize A/D and D/A channels
     */

    // clear interrupt mask
    ip_poke (0x6, 0x000);

    // set A/D enable high but keep D/A chip reset high
    ip_poke (0x7, 0x1FF);

    // assert A/D and D/A logic reset
    ip_poke (0x9, 0x1FF);
    ip_poke (0x9, 0x000);

    // wait a brief amount of time
    spox_delay (1000);

    // prime D/A fifos
    d9810_aux_wrfifo (aux.esm.src, d9810_AUXM_OUT, 256 >> 1, ESM_BUF_SIZE);

    // set A/D enable low but keep D/A chip reset high
    ip_poke (0x7, 0x01F);

    // set interrupt mask
    ip_poke (0x6, 0x20);



    /*
     * indicate active
     */

    // set flag
    aux.active = TRUE;

    // enable C40 interrupt
    MX40_enableIIF (IIOF_INT_ENB (IIOF3));
  }
}


/*============================================================================
  aux_select

  Select aux monitor channel.
============================================================================*/

void aux_select (int chan)
{
  aux.monitorId = chan;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Linker Command File

```
/*****************************************************************************
  DSPCon 9245 Linker Command File
  PLAIN-C
*****************************************************************************/

-c

-stack 0x100
-heap  0x0



/**
 **  SPECIFY STANDARD LIBRARIES
 **/


/** TI C Run Time Support Library **/
-lrts40r.lib



/**
 **  SPECIFY MEMORY MODEL
 **/

MEMORY
{
    CRAM    (RWIX): org = 0x002ff800, len = 0x00000800
    LSRAM   (RWIX): org = 0x40000000, len = 0x00080000
    GSRAM   (RWIX): org = 0xC0000000, len = 0x00080000
    DRAM    (RWIX): org = 0x80000000, len = 0x00100000
}


/**
 **  SPECIFY SECTIONS ALLOCATION
 **/

SECTIONS
{

    GROUP > CRAM
    {
        .traps:
        .stack:

        .bss:     {}

        .text:     {}
        .const:    {}
        .data:     {}

        .cinit:    {}
    }


}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

End of Section

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Appendix B



Boot Flash Source Code Listing

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Boot Flash Source Code

Note that the Boot Code resides at the base of the Flash (0x0030 0000) and consumes approximately 100 Kbytes (102236 bytes) of the Flash memory.

```
/*****************************************************************************
  DSPCon NETWORK BOOT MONITOR
*****************************************************************************/

#include <stdlib.h>
#include <string.h>

#include <spoxknl.h>
#include <spnet.h>

#include <ctype.h>
#include <stdio.h>

#include <c40/c40_tmr.h>

#include "flashInfo.h"


/*============================================================================
                    NETWORK COMMAND REFERENCE
============================================================================*/

#define BOOTMON_LOAD              (0x0011)
#define BOOTMON_JUMP              (0x0012)

#define BOOTMON_BUFSIZ            (0x110)


/*============================================================================
                       STATIC CONSTANTS
============================================================================*/

#define BOOTMON_VERSION           "1.12"
#define BOOTMON_SIZE              (0x020000)

#define FLASH_BOOT_ADDR(f) (0x00300000 + ((f) << 16))
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#define FLASH_WORK_BASE          (0x40000000)
#define FLASH_WORK_SIZE          (0x04000)

#define IMAGE_WORK_BASE          (0xC0000000)
#define IMAGE_WORK_SIZE          (0x18000)


/*=========================================================================
                      STATIC VARIABLES
=========================================================================*/

DSP9245_FlashInfo flashInfo;



/*=========================================================================
 *
 *                     FRONT PANEL LED
 *
=========================================================================*/

#define tmr_service c_int90

static
void tmr_service (void)
{
  *(volatile int*) 0x8000005 ^= 0x40;   // toggle LED D13
}



/*=========================================================================
 *
 *                  NETWORK MONITOR ROUTINES
 *
=========================================================================*/

/**
*** network_mon
**/

int network_mon (int sock)
{
  unsigned buffer [BOOTMON_BUFSIZ];
  unsigned cmd, size;
  int nr;

  struct sockaddr_in client;
  int client_size;

  // capture command frame from socket
  nr = spnet_recvfrom (sock, buffer, BOOTMON_BUFSIZ*4, 0,
                    (struct sockaddr*) &client, &client_size);

  if (nr <= 0)  return (-1);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // parse command word into cmd and size fields
  cmd = buffer[0] >> 16;
  size = buffer[0] & 0xFFFF;

  switch (cmd) {
  case BOOTMON_LOAD:
    { /*-------------------------------------------------------------------
       LOAD IMAGE DATA
      -------------------------------------------------------------------*/
      unsigned addr;

      // validate minimum size
      if (size < 1)  return (0);

      // capture copy address
      addr = buffer[1];

      // copy data to desired address
      fmemcpy (addr, buffer+2, size-1);
    }

    break;
  case BOOTMON_JUMP:
    { /*-------------------------------------------------------------------
       JUMP TO ADDRESS
      -------------------------------------------------------------------*/
      Fxn fxn;

      // validate packet size
      if (size != 1)  return (0);

      // capture jump address
      fxn = (Fxn) buffer[1];

      // disable global interrupts
      asm(" andn 2000h,ST ");
      asm(" nop ");
      asm(" nop ");
      asm(" nop ");

      // disable all interrupt sources
      asm(" and 0,IIE ");
      asm(" and 0,IIF ");

      // jump to address
      fxn ();
    }
  }


  // send acknowledgement
  spnet_sendto (sock, buffer, 2*4, 0, (struct sockaddr*) &client, client_size);
  return (0);
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/**
*** network_daemon
**/

static
int network_daemon (void)
{
  int servSock;


  // WAIT SOME TIME
  TSK_sleep (2500);


  { /*---------------------------------------------------------------------
      initialize network interface
    ---------------------------------------------------------------------*/
    SPNET_Attr spAttr;
    SPNET_DSP9245_Attr tulipAttr;

    SPNET_IntfAttr* ifAttr [2];
    int ifCnt = 0;

    /*-- dsp9245 tulip interface ----------------------------------------*/

    ifAttr [ifCnt++] = (SPNET_IntfAttr*) &tulipAttr;

    tulipAttr.intf.class = SPNET_INTF_DSP9245;
    tulipAttr.intf.ipAddr = flashInfo.net.ipAddr;
    tulipAttr.intf.netMask = flashInfo.net.ipNetMask;

    tulipAttr.ethAddr[0] = 0x91;
    tulipAttr.ethAddr[1] = 0x92;
    tulipAttr.ethAddr[2] = 0x93;
    tulipAttr.ethAddr[3] = 0x94;
    tulipAttr.ethAddr[4] = 0x95;
    tulipAttr.ethAddr[5] = (flashInfo.net.ipAddr & 0xFF);

    tulipAttr.ethType = SPNET_DSP9245_ETH_10MBIT;

    /*-- spnet attributes -----------------------------------------------*/

    spAttr.netTskPri = 10;
    spAttr.netNumBufs = 64;
    spAttr.netSegId = 0;

    if (SPNET_init (&spAttr, ifAttr, ifCnt) < 0) {
      puts ("!! network intitialization failed\n");
      return (-1);
    }
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  { /*---------------------------------------------------------------------
      create server socket
    ---------------------------------------------------------------------*/
    struct sockaddr_in server;

    if (servSock = spnet_socket (AF_INET, SOCK_DGRAM, 0),  servSock < 0) {
      puts ("!! network socket create failed\n");
      return (-2);
    }

    memset (&server, 0, sizeof (server));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = 7777;

    if (spnet_bind (servSock, (struct sockaddr*) &server, sizeof(server)) != 0) {
      puts ("!! network socket bind failed\n");
      return (-3);
    }
  }


  { /*---------------------------------------------------------------------
      process packets forever
    ---------------------------------------------------------------------*/
    while (TRUE) {
      if (network_mon (servSock) < 0)
       puts ("!! network transaction failed\n");
    }
  }


  return (0);
}



/*=========================================================================
 *
 *                  CONSOLE MONITOR ROUTINES
 *
=========================================================================*/

/***
 *** Flash Info Support Routines
 ***/

/**
*** computeFlashInfoChecksum
**/

unsigned computeFlashInfoChecksum (DSP9245_FlashInfo* info)
{
  unsigned cksum = 0;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  int cntr = DSP9245_FlashInfo_VERS_SIZE(info->versionId);
  unsigned* mptr = (unsigned*) info;

  while (cntr--)  cksum += *mptr++;

  return (cksum);
}


/***
 *** Console Support Routines
 ***/

/**
*** getline
**/

char* getline (char* buffer, int maxchar)
{
  char* ptr;
  int ch;

  // get line of input
  fgets (ptr = buffer, maxchar, stdin);
  putchar ('\n');

  // skip leading white space
  while (ch = *ptr,  ch && isspace(ch))  ++ptr;

  // return pointer
  return (ptr);
}


/**
*** parse_uint
**/

int parse_uint (char* buffer, int radix, unsigned* val)
{
  *val = strtoul (buffer, &buffer, radix);

  if (!isspace (*buffer)) {
    printf ("INVALID: must be numeric argument expressed in %s notation.\n\n",
           (radix == 16) ? "hexadecimal" : (radix == 10) ? "decimal" : "unknown");

    return (-1);
  }

  return (0);
}


/**
*** prep_net_ipaddr
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
**/

char* prep_net_ipaddr (char* buffer, unsigned val)
{
  sprintf (buffer, "%03d.%03d.%03d.%03d",
          (val >> 24) & 0xFF,
          (val >> 16) & 0xFF,
          (val >>  8) & 0xFF,
          (val >>  0) & 0xFF);

  return (buffer);
}


/**
*** parse_net_ipaddr
**/

int parse_net_ipaddr (char* buffer, unsigned* val)
{
  unsigned char byVal[4];

  byVal[0] = strtoul (buffer, &buffer, 10);
  if (byVal[0] >= 0x100  ||  *buffer++ != '.')  goto parseFailed;

  byVal[1] = strtoul (buffer, &buffer, 10);
  if (byVal[1] >= 0x100  ||  *buffer++ != '.')  goto parseFailed;

  byVal[2] = strtoul (buffer, &buffer, 10);
  if (byVal[2] >= 0x100  ||  *buffer++ != '.')  goto parseFailed;

  byVal[3] = strtoul (buffer, &buffer, 10);
  if (byVal[3] >= 0x100  ||  !isspace (*buffer))  goto parseFailed;

  *val = (byVal[0] << 24) | (byVal[1] << 16) | (byVal[2] << 8) | byVal[3];
  return (0);

 parseFailed:
  printf ("INVALID: must be network address expressed in standard dot notation.\n\n");
  return (-1);
}


/**
*** prep_vme_addr
**/

char* prep_vme_addr (char* buffer, unsigned val)
{
  switch (DSP9245_FlashVme_ADDR_SPACE(val)) {
  case DSP9245_FlashVme_OFF:
    sprintf (buffer, "off");
    break;

  case DSP9245_FlashVme_A16:
    sprintf (buffer, "A16: 0x%04X", DSP9245_FlashVme_ADDR_VALUE(val));
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    break;

  case DSP9245_FlashVme_A24:
    sprintf (buffer, "A24: 0x%06X", DSP9245_FlashVme_ADDR_VALUE(val));
    break;

  case DSP9245_FlashVme_A32:
    sprintf (buffer, "A32: 0x%08X", DSP9245_FlashVme_ADDR_VALUE(val));
    break;
  }

  return (buffer);
}


/**
*** parse_vme_addr
**/

int parse_vme_addr (char* buffer, unsigned wsize, unsigned* val)
{
  unsigned space;
  unsigned value;

  if (wsize == 0) {
    // window size zero, therefore address must be OFF
    *val = DSP9245_FlashVme_ADDR (DSP9245_FlashVme_OFF, 0);
    return (0);
  }

  if (tolower (buffer[0]) == 'o'   &&   tolower (buffer[1]) == 'f'   &&   tolower
(buffer[2]) == 'f') {
    // keyword OFF, therefore address must be OFF
    *val = DSP9245_FlashVme_ADDR (DSP9245_FlashVme_OFF, 0);
    return (0);
  }

  if (tolower (buffer[0]) == 'a') {
    // parse initial 'Axx:' specifier
    value = strtoul (buffer+1, &buffer, 10);
    if (*buffer++ != ':')  return (-1);

    switch (value) {
    case 16:  space = DSP9245_FlashVme_A16; break;
    case 24:  space = DSP9245_FlashVme_A24; break;
    case 32:  space = DSP9245_FlashVme_A32; break;

    default:  goto parseFailed;
    }
  }

  else {
    // assume initial 'A32:' specifier
    space = DSP9245_FlashVme_A32;
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // parse address value
  value = strtoul (buffer, &buffer, 16);
  if (!isspace (*buffer))  goto parseFailed;

  // validate address value within address space
  if (space == DSP9245_FlashVme_A16  &&  value > 0xFFFF) {
    printf ("INVALID: the address must be valid for A16 space.\n\n");
    return (-1);
  }

  // validate address value within address space
  if (space == DSP9245_FlashVme_A24  &&  value > 0xFFFFFF) {
    printf ("INVALID: the address must be valid for A24 space.\n\n");
    return (-1);
  }

  // validate address value aligned on window size boundary
  if (value & (wsize-1)) {
    printf ("INVALID: the address must be on 0x%X byte boundary.\n\n", wsize);
    return (-1);
  }

  // set VME address
  *val = DSP9245_FlashVme_ADDR (space, value);
  return (0);

 parseFailed:
  printf ("INVALID: must be in the form 'SPACE: VALUE' where SPACE may be
A16,A24,A32\n"
         "and VALUE must be numeric argument expressed in hexadecimal notation.\n\n");

  return (-1);
}


/**
*** prep_vme_size
**/

char* prep_vme_size (char* buffer, unsigned val)
{
  sprintf (buffer, "0x%06X", val);
  return (buffer);
}


/**
*** parse_vme_size
**/

int parse_vme_size (char* buffer, unsigned* val)
{
  unsigned wsize;

  // parse window size
  if (parse_uint (buffer, 16, val) != 0)  return (-1);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // verify window size is non-zero power of two
  if (wsize = *val,  wsize != 0) {
    while ((wsize & 1) == 0)  wsize >>= 1;

    if (wsize != 1) {
      printf ("INVALID: the size must be an integral power of two.\n\n");
      return (-1);
    }
  }

  return (0);
}


/***
 *** Console Command Routines
 ***/

void  cmd_change    (int argc, char** argv);
void  cmd_display   (int argc, char** argv);
void  cmd_dump            (int argc, char **argv);
void  cmd_help            (int argc, char** argv);
void  cmd_modify    (int argc, char** argv);
void  cmd_run       (int argc, char** argv);
void  cmd_trace           (int argc, char** argv);


/**
*** change -- change boot parameters
**/

void cmd_change (int argc, char** argv)
{
  DSP9245_FlashInfo info = flashInfo;


  { /*-------------------------------------------------------------------------
      query parameter values
    -------------------------------------------------------------------------*/
    char buffer[80];
    char* ptr;

    int redo;

    /*
     * IP Address
     */

    do {
      prep_net_ipaddr (buffer, info.net.ipAddr);

      printf ("IP Address      %-20s ", buffer);
      fflush (stdout);

      if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
   unsigned val;

   if (redo = parse_net_ipaddr (ptr, &val),  !redo) {
     info.net.ipAddr = val;
   }
  }
} while (redo);

/*
 * IP NetMask
 */

do {
  prep_net_ipaddr (buffer, info.net.ipNetMask);

  printf ("IP Net Mask     %-20s ", buffer);
  fflush (stdout);

  if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
   unsigned val;

   if (redo = parse_net_ipaddr (ptr, &val),  !redo) {
     info.net.ipNetMask = val;
   }
  }
} while (redo);

/*
 * IP Default Gateway
 */

do {
  prep_net_ipaddr (buffer, info.net.ipDefGateway);

  printf ("Def Gateway     %-20s ", buffer);
  fflush (stdout);

  if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
   unsigned val;

   if (redo = parse_net_ipaddr (ptr, &val),  !redo) {
     info.net.ipDefGateway = val;
   }
  }
} while (redo);

/*
 * VME Memory Size
 */

do {
  prep_vme_size (buffer, info.vme.memSize);

  printf ("VME Mem Size    %-20s ", buffer);
  fflush (stdout);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
   unsigned val;

   if (redo = parse_vme_size (ptr, &val),  !redo) {
     info.vme.memSize = val;
   }
  }
} while (redo);

/*
 * VME Memory Address
 */

do {
  prep_vme_addr (buffer, info.vme.memAddr);

  printf ("VME Mem Addr    %-20s ", buffer);
  fflush (stdout);

  if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
   unsigned val;

   if (redo = parse_vme_addr (ptr, info.vme.memSize, &val),  !redo) {
     info.vme.memAddr = val;
   }
  }
} while (redo);

/*
 * VME Register Address
 */

do {
  prep_vme_addr (buffer, info.vme.regAddr);

  printf ("VME Reg Addr    %-20s ", buffer);
  fflush (stdout);

  if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
   unsigned val;

   if (redo = parse_vme_addr (ptr, 4096, &val),  !redo) {
     info.vme.regAddr = val;
   }
  }
} while (redo);

/*
 * Processor Id
 */

do {
  sprintf (buffer, "%d", info.procId);

  printf ("Processor Id    %-20s ", buffer);
  fflush (stdout);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
     unsigned val;

     if (redo = parse_uint (ptr, 10, &val),  !redo) {
       info.procId = val;
     }
    }
  } while (redo);

  /*
   * Flash Boot
   */

  do {
    sprintf (buffer, "%c", (info.bootImage ? 'y' : 'n'));

    printf ("Flash Boot      %-20s ", buffer);
    fflush (stdout);

    if (ptr = getline (buffer, 80),  redo = (*ptr != 0),  redo) {
     int val = tolower (*ptr);

     if (val == 'y')  redo = FALSE,  info.bootImage = TRUE;
     else if (val == 'n')  redo = FALSE,  info.bootImage = FALSE;
    }
  } while (redo);
}


{ /*-----------------------------------------------------------------------
    commit changes to flash
  -----------------------------------------------------------------------*/
  printf ("\n"  "updating flash parameters...\n");

  // copy flash parameters
  flashInfo = info;

  // compute flash info checksum
  flashInfo.checksum = 0;
  flashInfo.checksum = computeFlashInfoChecksum (&flashInfo);

  // read existing image from flash
  flash_read_words (FLASH_WORK_BASE, DSP9245_FLASH_BOOT_BASE, FLASH_WORK_SIZE);

  // replace flash parameters
  *(DSP9245_FlashInfo*) (FLASH_WORK_BASE + DSP9245_FLASH_INFO_OFFSET) = flashInfo;

  // erase flash sector
  printf ("...erasing flash...\n");
  flash_erase_sector (0);

  // write image to flash
  printf ("...writing flash...\n");
  flash_write_words (FLASH_WORK_BASE, DSP9245_FLASH_BOOT_BASE, FLASH_WORK_SIZE);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    // verify flash image
    printf ("...verifying...\n");
    flash_compare_words (FLASH_WORK_BASE, DSP9245_FLASH_BOOT_BASE, FLASH_WORK_SIZE);

    putchar ('\n');
    cmd_display (0,0);
  }
}


/**
*** display
**/

void cmd_display (int argc, char** argv)
{
  char buffer[80];

  printf ("IP Address      %s\n", prep_net_ipaddr (buffer, flashInfo.net.ipAddr));
  printf ("IP Net Mask     %s\n", prep_net_ipaddr (buffer, flashInfo.net.ipNetMask));
  printf    ("Def    Gateway                    %s\n",    prep_net_ipaddr    (buffer,
flashInfo.net.ipDefGateway));

  printf ("VME Mem Size    %s\n", prep_vme_size (buffer, flashInfo.vme.memSize));
  printf ("VME Mem Addr    %s\n", prep_vme_addr (buffer, flashInfo.vme.memAddr));
  printf ("VME Reg Addr    %s\n", prep_vme_addr (buffer, flashInfo.vme.regAddr));

  printf ("Processor Id    %d\n", flashInfo.procId);
  printf ("Flash Boot      %c\n", flashInfo.bootImage ? 'y' : 'n');
  printf ("\n");
}


/**
*** dump
**/

void cmd_dump (int nargs, char **argv)
{
  static unsigned *addr = 0;
  static unsigned size = 1;

  if (nargs == 2) {
    if (strcmp (argv[1],"die") == 0) {
      printf ("die: %08X\n", MX40_getDIE());
      return;
    }
    else if (strcmp (argv[1],"iie") == 0) {
      printf ("iie: %08X\n", MX40_getIIE());
      return;
    }
    else if (strcmp (argv[1],"iif") == 0) {
      printf ("iif: %08X\n", MX40_getIIF());
      return;
    }
    else if (strcmp (argv[1],"st") == 0) {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
      printf ("st: %08X\n", MX40_getST());
      return;
    }
  }


  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);


  { unsigned* ptr = addr;
    int i,j;

    for (i=0; i < size; ) {
      printf("%08X:  ", ptr);

      for (j=0;  i < size && j < 4;  i++, j++)
       printf("%08X  ", *ptr++);

      putchar ('\n');
    }
  }
}


/**
*** help -- display help text
**/

void cmd_help (int argc, char** argv)
{
  puts ("?            help");
  puts ("@            run program from flash");
  puts ("c            change system parameters");
  puts ("d            dump system memory");
  puts ("m            modify system memory");
  puts ("p            display system parameters");
  puts ("t            display trace buffer");
  putchar ('\n');
}


/**
*** modify
**/

void cmd_modify (int nargs, char **argv)
{
  unsigned val;

  if (nargs < 3)
    return;

  val = (unsigned) strtoul (argv[2], 0, 0);

  if (strcmp (argv[1],"die") == 0) {
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    printf ("die: %08X\n", MX40_getDIE());
  }
  else if (strcmp (argv[1],"iie") == 0) {
    printf ("iie: %08X\n", MX40_getIIE());
  }
  else if (strcmp (argv[1],"iif") == 0) {
    printf ("iif: %08X\n", MX40_getIIF());
  }
  else if (strcmp (argv[1],"st") == 0) {
    printf ("st: %08X\n", MX40_getST());
  }
  else {
    unsigned* addr;

    addr = (unsigned*) strtoul (argv[1], 0, 16);
    *addr = val;

    printf ("%08X:  %08X\n", addr, val);
  }
}


/**
*** run -- run loaded image
**/

void cmd_run (int argc, char** argv)
{
  unsigned char* image = (void*) FLASH_BOOT_ADDR(2);

  unsigned entryPt;

  unsigned loadAddr;
  unsigned loadCnt;

  // parse optional 'where' argument if specified
  if (argc >= 2) {
    if (toupper( *argv[1] ) == 'F') {
      // parse flash section number (0-7)
      image = (void*) FLASH_BOOT_ADDR( strtol(argv[1] +1, 0, 10) );
    }
    else {
      // parse absolute address
      image = (void*) strtoul (argv[1], 0, 16);
    }
  }

  // verify header info
  if (flash_get_word (image) != 0x0008)  return;

  // notify operator
  puts ("loading program from flash...");

  // skip header info
  image += 4*4;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  // get entry point
  entryPt = flash_get_word (image);
  image += 2*4;

  // process load records
  while (loadCnt = flash_get_word (image),  loadCnt != 0) {
    image += 4;

    loadAddr = flash_get_word (image);
    image += 4;

    flash_read_words (loadAddr, image, loadCnt);
    image += loadCnt * 4;
  }

  // notify operator
  puts ("executing program.");

  // disable global interrupts
  asm(" andn 2000h,ST ");
  asm(" nop ");
  asm(" nop ");
  asm(" nop ");

  // disable all interrupt sources
  asm(" and 0,IIE ");
  asm(" and 0,IIF ");

  // jump to entry point
  ((Fxn) entryPt) ();
}


/**
*** trace
**/

void cmd_trace (int nargs, char **argv)
{
  static unsigned* addr = (unsigned*) 0xC007FF00;
  static unsigned size = 256;

  if (nargs >= 2)  addr = (unsigned*) strtoul (argv[1], 0, 16);
  if (nargs >= 3)  size = (unsigned) strtoul (argv[2], 0, 0);

  { unsigned* ptr = addr;
    int cntr = size;
    int ch;

    while (cntr--  &&  (ch = *ptr++,  ch != 0))  putchar (ch);
    putchar ('\n');
  }
}


/**
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
*** command table
**/

typedef (*CmdFxn) (int argc, char** argv);

typedef struct {
  const char* cmd;
  CmdFxn fxn;
} CmdInfo;


#define NUM_CMDS  (sizeof(cmdTab) / sizeof(CmdInfo))

static CmdInfo cmdTab [] = {
  { "?",            (CmdFxn) cmd_help },
  { "@",            (CmdFxn) cmd_run },
  { "c",            (CmdFxn) cmd_change },
  { "d",            (CmdFxn) cmd_dump },
  { "h",            (CmdFxn) cmd_help },
  { "m",            (CmdFxn) cmd_modify },
  { "p",            (CmdFxn) cmd_display },
  { "t",            (CmdFxn) cmd_trace },
};



/**
*** main
**/

void main (void)
{
  volatile TmrRegs* tmr = (volatile TmrRegs*) TMR_REG_ADDR(1);

  { /*-------------------------------------------------------------------
      configure status/LED timer: normal blink
    -------------------------------------------------------------------*/
    tmr->control = TMR_FUNC_ICLK_TMR | TMR_RESET;
    tmr->period = 100 * 15000;

    MX40_plugIVT (TMR_INT_VEC(1), tmr_service);
    MX40_enableIIE (TMR_INT_ENB(1));

    tmr->control = TMR_FUNC_ICLK_TMR | TMR_START;
  }


  { /*-------------------------------------------------------------------
      create console interface
    -------------------------------------------------------------------*/
    freopen ("/hblock:" "/uart9245,9600,8,n,1", "r", stdin);
    freopen ("/uart9245,9600,8,n,1", "w", stdout);

    TSK_setpri (TSK_self(), 12);
  }
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
{ /*-------------------------------------------------------------------------
    read flash information table
  -------------------------------------------------------------------------*/
  flash_read_words (&flashInfo, (void*) DSP9245_FLASH_INFO_BASE, sizeof(flashInfo));
}


{ /*-------------------------------------------------------------------------
    display startup banner and wait for activity
  -------------------------------------------------------------------------*/
  puts ("\n"
       "\n"
       "DSPCon 9245 NETWORK BOOT MONITOR\n"
       "Version "  BOOTMON_VERSION  "\n"
       "\n");

  cmd_display (0,0);

  if (flashInfo.bootImage) {
    unsigned timo = 1 * 100;

    puts ("press any key to abort flash boot operation...");

    while (--timo  &&  !UART9245_rxCount ())  TSK_sleep (10);
    if (!UART9245_rxCount ())  cmd_run (0,0);
  }
}


{ /*-------------------------------------------------------------------------
    configure pci bus interface
  -------------------------------------------------------------------------*/
  pci_init ();
  dpr_alloc (BOOTMON_SIZE << 2);  // reserve memory occupied by boot monitor
}


if (flashInfo.net.ipAddr  &&  flashInfo.net.ipAddr != (unsigned)(-1)) {
  /*-------------------------------------------------------------------------
    invoke network monitor daemon
  -------------------------------------------------------------------------*/
  TSK_Attrs attr = TSK_ATTRS;
  char buffer [40];

  printf ("starting network monitor at IP address %s...\n",
         prep_net_ipaddr (buffer, flashInfo.net.ipAddr));

  attr.priority = 9;
  TSK_create ((Fxn) network_daemon, &attr);
}


while (1) {
  /*-------------------------------------------------------------------------
    perform console monitor forever
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    ------------------------------------------------------------------------*/
#define HIST_SIZE  5
#define LINE_SIZE  79

    static char buffer [HIST_SIZE+1][LINE_SIZE+1];
    static int curline = 0;

    int kcmd;

    char argbuf [LINE_SIZE+1];
    char* argp;

    char* argvec [10];
    int argcnt;

    char* ptr;
    int ch;

    // display prompt
    fputs ("> ", stdout);
    fflush (stdout);

    // get line of input
    ptr = getline (buffer[curline], LINE_SIZE);
    if (ch = *ptr,  ch == 0)  continue;

    // handle line history references
    if (ch == '!') {
      int hindx = strtol (ptr+1, 0, 0);
      int kline;

      if ((unsigned) hindx > HIST_SIZE) continue;
      if (hindx == 0)  hindx = 1;

      kline = curline - hindx;
      if (kline < 0)  kline += HIST_SIZE+1;

      ptr = strcpy (buffer[curline], buffer[kline]);
      while (ch = *ptr,  ch && isspace(ch))  ++ptr;

      if (ch == 0)  continue;
      printf ("!%d > %s", hindx, ptr);
    }

    // parse input into word list
    argp = argbuf;
    argcnt = 0;

    do {
      argvec [argcnt++] = argp;

      // skip over non-space characters
      while (ch != 0  &&  !isspace(ch))  *argp++ = ch,  ch = *++ptr;
      *argp++ = 0;

      // skip over trailing space characters
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
      while (ch != 0  &&  isspace(ch))  ch = *++ptr;
    } while (ch != 0);

    argvec [argcnt] = 0;

    // locate command entry
    for (kcmd=0; kcmd < NUM_CMDS; kcmd++) {
      if (strcmp (cmdTab[kcmd].cmd, argvec[0]) == 0)  break;
    }

    // execute command
    if (kcmd < NUM_CMDS) {
      cmdTab[kcmd].fxn (argcnt, argvec);
      if (++curline == HIST_SIZE+1)  curline = 0;
    }
    else {
      printf ("unknown command\n");
    }

  }
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Warranty

DSP*Con*, Inc. warrants that all products manufactured by DSP*Con*, Inc. conform to published DSP*Con* specifications and are free from defects in materials and workmanship for a period of one (1) year from the date of delivery when used under normal conditions and within the service conditions for which they were furnished.

The obligation upon DSP*Con*, Inc. arising from a warranty claim shall be limited to repairing, or, at its option, replacing without charge, any product which, in DSP*Con*, Inc.'s sole opinion, proves to be defective within the scope of this warranty.

DSP*Con*, Inc. must be notified in writing of the defect or nonconformity within the warranty period, and the affected product must be returned to DSP*Con*, Inc. within thirty (30) days after discovery of such defect or nonconformity.

The buyer shall prepay shipping charges, taxes, duties and insurance for products returned to DSP*Con*, Inc. for warranty service. DSP*Con*, Inc. shall pay for the return of products to buyer except for products returned to another country or from outside the forty-eight contiguous United States.

DSP*Con*, Inc. shall have no responsibility for any defect or damage caused by improper installation, unauthorized modification, misuse, neglect, inadequate maintenance, accident or for any product that has been repaired or altered by anyone other than DSP*Con*, Inc. or its authorized representatives.

The warranty described above is the buyer's sole and exclusive remedy and no other warranty, whether oral or written, is expressed or implied. DSP*Con*, Inc. specifically disclaims fitness for a particular purpose. Under no circumstances shall DSP*Con*, Inc. be liable for any direct, indirect, special, incidental or consequential damages, expenses, losses or delays (including loss of profits) based on contract, tort, or any legal theory.

This warranty is valid for a period of one (1) year from the date of shipment. DSP*Con*, Inc. extends this warranty to the party that first purchases a new unit from the distributor, and extends the validity of the warranty to one year from the date of sale by distributor, as evidenced by the distributor's invoice. However, this limited extended warranty period shall not be more than 18 months from the date of shipment from DSP*Con*'s factory.

Upon warranty expiration, all services on DSP*Con* hardware, software and systems are subject to a current hourly rate, plus travel expenses. As an option, DSP*Con*, Inc. offers extended warranties when purchased within ten (10) days of receipt of product.

Extended Warranties

Hardware

DSP*Con*, Inc. offers an extended warranty on its hardware at a cost of 4% of the list price per month of coverage, or 20% of the current list price per year of coverage.

This coverage is available for DSP*Con* hardware only. All third-party hardware is covered by the manufacturer's warranty.

Software

DSP*Con*, Inc. offers an extended warranty on its software for one, two and three year periods, at a cost of 15%, 28% and 40%, respectively, of the current list price.

This coverage is available for DSP*Con* software only. All third party software is covered by the manufacturer's warranty.

Systems

DSP*Con*, Inc. offers an extended warranty on its systems at a cost of 20% of the current list price, per year of coverage.

This coverage includes all hardware and software used in system integration.

Service & Repair

Before returning a product for service and repair, please contact DSP*Con*'s corporate offices at (908) 722-5656 to obtain a Return Material Authorization (RMA) number. Have available the Model and Serial numbers of the item, a description of the problem, your company name, and the name and address of the person returning the product.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



DSP*Con*, Incorporated
380 Foothill Rd.
Bridgewater, NJ 07828

Phone **908-722-5656**
Phone **888-DSP-CON8**
Fax **908-722-3259**
E-mail **info@dspcon.com**
Website **www.dspcon.com**

A Publication of
DSP*Con*, Incorporated

© MMI

Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced, stored in or transmitted by a retrieval system, or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, without the written permission of DSP*Con*, Inc.

DSP*Con* has worked to verify the accuracy of the information contained in this document as of its publication date; however, such information is subject to change without notice and DSP*Con* is not responsible for any errors that may occur in this document.

Trademarks are acknowledged and are the property of their owners.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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