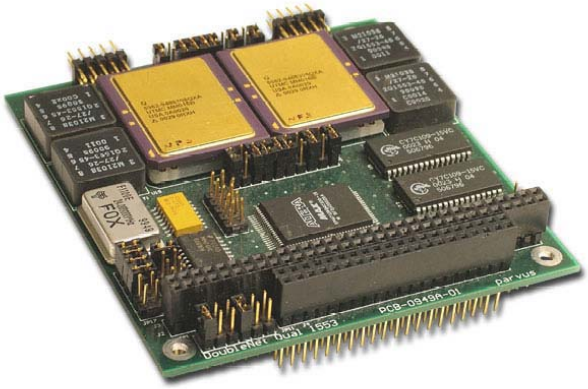
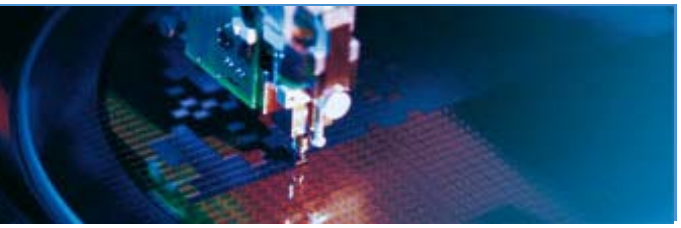
USER MANUAL





COM-1250 / 1251

PC/104 MIL-STD-1553 Interfaces, 1 / 2-Channel (COM-1250-01, COM-1251-00)

MNL-0462-01 Rev E3 REF. ECO-3163 01 Feb 10

RUGGED SOLUTIONS FOR REAL WORLD APPLICATIONS

www.parvus.com

Parvus

Disclaimer

Although the information contained herein has been carefully verified, Parvus Corporation assumes no responsibility for errors that might appear in this document, or for damage to property or persons resulting from improper use of this manual or related software. Parvus reserves the right to change the contents and form of this document, as well as the features and specifications of its products at any time without notice. The information in this publication does not represent a commitment on the part of Parvus. This document contains proprietary information that is protected by copyright.

Parvus Corporation
3222 S. Washington St.
Salt Lake City, Utah, USA 84115
Phone: +1 (801) 483-1533
Toll-Free: +1 (800) 483-3152
Main: +1 (801) 483-1533
Fax: +1 (801) 483-1523
Email:
Sales: sales@parvus.com
Support: tsupport@parvus.com
Web-site: http://www.parvus.com

Send us your comments and feedback: feedback@parvus.com

Parvus is a U.S. subsidiary of the Eurotech Group (www.eurotech.com), a global family of technology companies focused on innovative embedded and high performance computing solutions.

Trademarks

All trademarks both marked and not marked, appearing in this document, are the property of their respective owners.

WEEE

The information below is issued in compliance with the regulations as set out by the 2002/96/CE directive, subsequently superseded by 2003/108/CE, and refers electrical and electronic equipment and the management of their waste (WEEE). When disposing of a device, including all of its components, subassemblies and materials that are an integral part of the product, you should take the WEEE directive into consideration.

This symbol has been attached to the equipment or, in the case that this is not possible, on the packaging, instruction literature and/or the guarantee sheet. By using this symbol it states that the device has been marketed after August 13th 2005, and implies that you must separate all of its components when possible, and dispose of them in accordance with local waste disposal legislations.

- Because of the substances present in the equipment, an improper use or disposal of the refuse can cause damage to human health and to the environment.
- With reference to WEEE, it is compulsory not to dispose of the equipment with normal urban refuse; arrangements should be instigated for separate collection and disposal.
- For more detailed information about recycling of WEEE, please contact your local waste collection body.
- In case of illicit disposal, sanctions will be levied on transgressors.

Table of Contents

Chapter 1 Introduction

The following manual describes the COM-1250 MIL-STD-1553 Single Channel Interface board and COM-1251 Dual Channel Interface board.

Functional Description

The COM-1250 and COM-1251 PC/104 1553 Interface boards provide the systems designer with an intelligent solution to MIL-STD-1553b multiplexed serial data design problems. These Commercial-Off-the-Shelf (COTS) PC/104 modules provides an interface from its one or two dual-redundant 1553 buses to any embedded PC/104 computer system bus. A PC/104 processor can then be used to access the 1553 Dual Bus to configure it for various modes of operation: Remote Terminal (RT) mode, Bus Control (BC) mode, or Monitor Terminal (MT) mode for a variety of 1553 military avionics and ground vehicle applications.

Satisfying all requirements of MIL-STD-1553B Notice II, the 1553 interface utilizes Aeroflex UTMC's SµMMIT MIL-STD-1553 support chips, which offer automatic message handling, message status, general status polling and interrupt capability. The register-based interface architecture provides many programmable functions as well as extensive device maintenance information. Four/Eight 16-bit register locations within the SµMMIT and the Pseudo Dual Ported Memory (PDPM) allow complete access to interface functions through PC/104 I/O space. Software, including programmed I/O instructions (supplied with the board) can be used to access the PDPM and internal registers, and also be used to transfer data to and from the PDPM using Direct Memory Access (DMA). The SµMMIT chip can also cause interrupts to be generated and routed to the PC/104 processor.

This military/avionics databus interface operates in extended temperatures and is specifically designed to work with transformer-coupled stubs in military Local Area Network (LAN) applications, including airplanes, tanks, missiles, satellites and helicopters, as well as space and land-based systems to process data for navigation and altitude, weapon status indications, target tracks and electronic warfare lines.

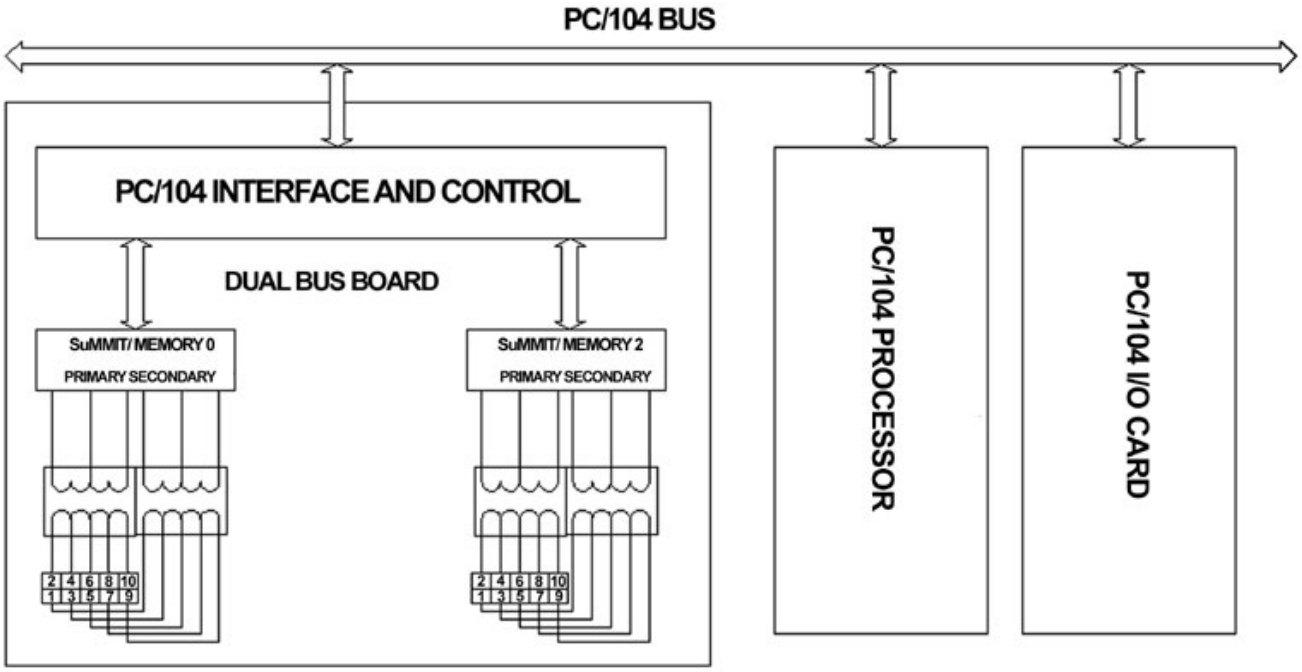
Operating as a *Remote Terminal*, the board provides many features for efficient handling of both bulk and periodic data. Capabilities include sub-address data indexing, double (ping pong) and circular buffering, internal illegalization, user-selectable broadcast command control, programmable interrupts, retries, message information word and time-lag, 16-bit for all transacted messages.

In *Bus Controller* mode, the powerful, field-proven module meets the demands of multi-frame processing with minimal host attention. User-programmable decision-making allows autonomous bus operation until a user-defined event, or series of events occurs. Structuring of independent tasks through the PC/104 host, such as data transfer, service requests and bus diagnostics (initiate BIT), facilitates multiple message processing. Periodic message transactions with multiple remote terminals can be controlled by specifying time between messages. Host instructions can also be given for polling activities. It also offers automatic retry (up to four times per command block).

As a full-featured bus monitor in *Monitor Terminal* mode, the module provides the capability to monitor message validity and terminal health on all, or selected remote stations on the bus with very little host interaction. The board also provides the option of running simultaneously in RT/MT mode, allowing the MT to communicate on the bus as a Remote Terminal. (Note: in this combined mode, the MT can not monitor its own RT address.)

Block Diagram

Diagram for COM-1251 (the COM-1250 has only one SuMMIT chip and therefore only one channel on the Dual Bus board):



Features

- 16-bit PC/104 Bus
- Dual Redundant MIL-STD-1553 Bus Operation
- Four/Eight 16-bit register locations
- MIL-Std-1553B protocol
- Standard PC/104 footprint
- Supports remote, monitor, or bus control modes
- Test software provided
- One/Two UTMC SuMMIT Chipset(s)

About PC/104

The PC/104 specification is characterized by its small form-factor (3.550" x 3.775"), stackable 104-pin/socket ISA bus connector, and reduced bus signal drive, making PC/104's size, durability, expandability, reliability, quality, and power consumption ideal for embedded computing. PC/104 technology leverages the same readily available development tools used with personal desktop computers to dramatically improve time-to-market for embedded systems development. The full PC/104 specification can be found at the PC/104 Consortium Web site: http://www.pc104.org/technology/pc104_tech.html

Chapter 2 Quick Start-up

This chapter shows the jumper layout and explains how to set them up.

The user will need to provide 1553 transformer couplers to the Primary and Secondary Lines of the 1553 Bus as shown here in Figure 2.

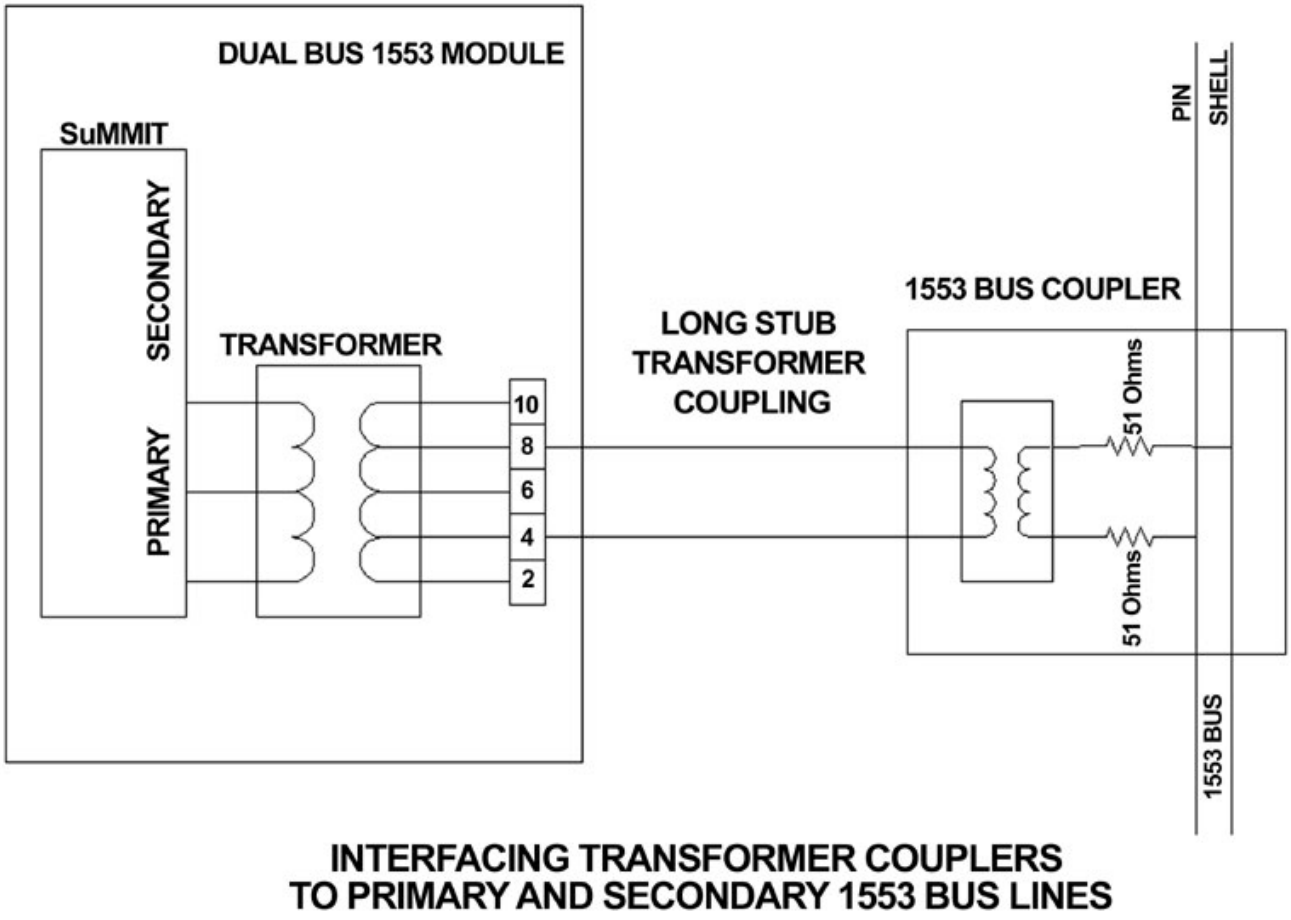


Figure 2: Interfacing with the 1553 Bus

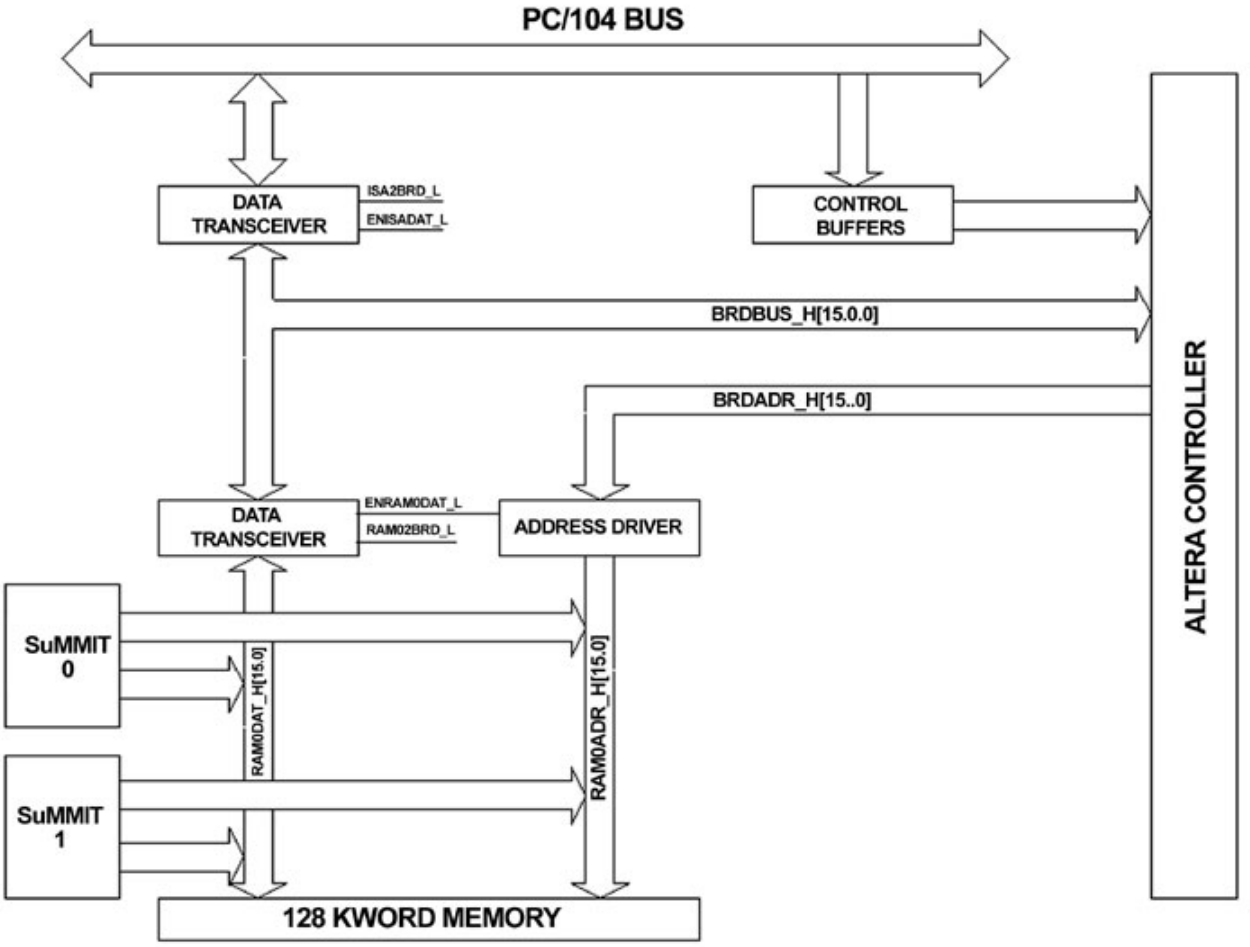


Figure 3: High-Level Functional Partition of the 1553 Bus.

The Primary and Secondary 1553 buses are fed through two line transformers into the SuMMIT support chip manufactured by United Technologies Microelectronics Center, Inc.

The SuMMIT uses internal support registers and an external Pseudo Dual Ported Memory (PDPM). The PDPM is used to buffer the 1553 data and also holds descriptor operating instructions for the SuMMIT.

The processor on the PC/104 can access the SuMMIT registers using programmed I/O instructions to configure it for the various modes of operation. It can also access the PDPM memory using programmed I/O instructions. The card also supports Direct Memory Access (DMA) between the PDPM memory and the PC/104 memory.

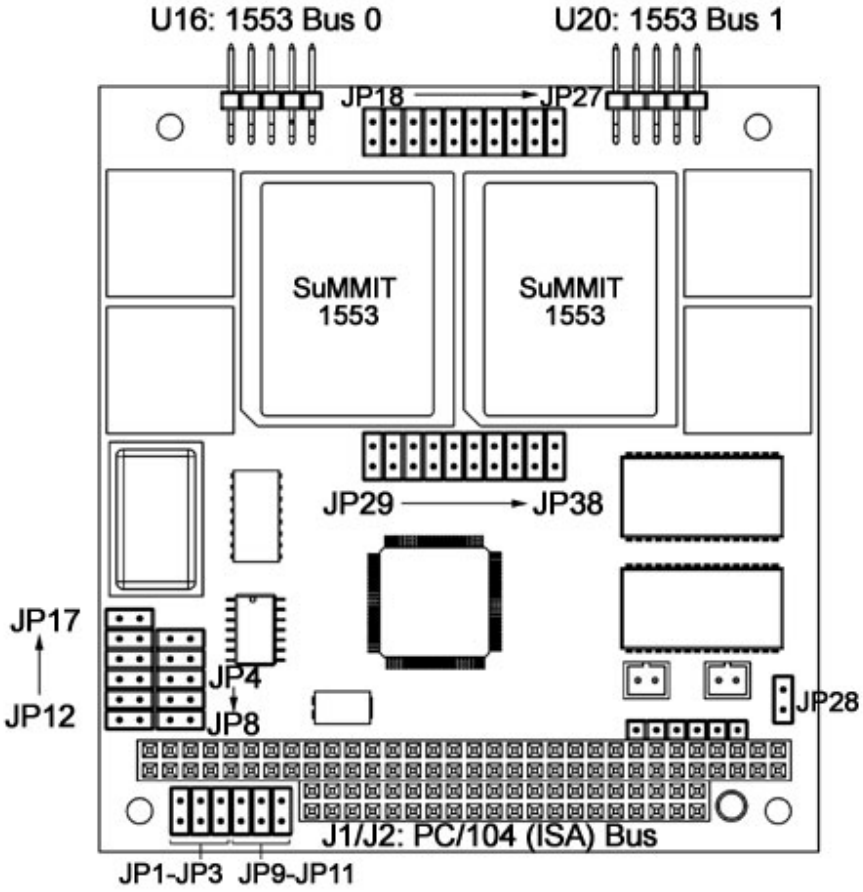
The SuMMIT sources low-priority and high-priority interrupts, which are sent to the processor and handled by the user software. There are various other registers on the 1553/Dual Bus which are used to support DMA access and also for control and status. These registers are described in the *Programmers Guide* section in this manual..

Chapter 3 Connector Description

This chapter includes the pinouts, signal descriptions and connector part numbers for the [Product Name]. Also provided in this section are connector part numbers along with suggested mating connector details as necessary.

Connector Identification

Figure 3.1 Connector Layout



Jumper Locations on the COM-1250/COM1251 board

Connector Pinouts

J1/J2: PC/104 Bus

Pin	Row A	Row B	Row C	Row D
0	--	--	GND	GND
1	/IOck	GND	/SBHE	/MCS16
2	SD7	Rstdrv	LA23	/IOCS16
3	SD6	+5v	LA22	IRQ10
4	SD5	IRQ9	LA21	IRQ11
5	SD4	-5v	LA20	IRQ12
6	SD3	DRQ2	LA19	IRQ15
7	SD2	-12v	LA18	IRQ14
8	SD1	/Exfer	LA17	/DA0
9	SD0	+12v	/MR	DRQ0
10	IOrdy	(key)	/MW	/DA5
11	AEN	/SMW	SD8	DRQ5
12	SA19	/SMR	SD9	/DA6
13	SA18	/IOW	SD10	DRQ6
14	SA17	/IOR	SD11	/DA7
15	SA16	/DA3	SD12	DRQ7
16	SA15	DRQ3	SD13	+5v
17	SA14	/DA1	SD14	/MSTR
18	SA13	DRQ1	SD15	GND
19	SA12	/Rfrsh	(key)	GND
20	SA11	SCLK	--	--
21	SA10	IRQ7	--	--
22	SA9	IRQ6	--	--
23	SA8	IRQ5	--	--
24	SA7	IRQ4	--	--
25	SA6	IRQ3	--	--
26	SA5	/DA2	--	--
27	SA4	TC	--	--
28	SA3	BALE	--	--
29	SA2	+5v	--	--
30	SA1	OSC	--	--
31	SA0	GND	--	--
32	GND	GND	--	--

U16 (Bus 0) & U20 (Bus 1, COM-1251 only) - 1553 Interface

Pin	Bus B	Pin	Bus A
1	Bus B +	2	Bus A +
3	Bus B + (Pin)	4	Bus A + (Pin)
5	Bus B +/-	6	Bus A +/-
7	Bus B - (Shell)	8	Bus A - (Shell)
9	Bus B -	10	Bus A -

The followingsection shows various jumpers which can be used to select operating modes of the COM-1250/Com-1251. The Jumper IN will cause the signal to go LOW or select a DMA or Interrupt Request Level. ONLY ONE JUMPER SHOULD BE INSERTED FOR A DMA OR INTERRUPT REQUEST LEVEL!

Jumpers JP29 and JP38 apply to the COM-1251 only. See the SuMMIT documentation from United Technologies (http://www.utmc.com) for a further description of jumpers JP01 through JP38. Jumper JP28 is a connector which will allow an external source to clock the time tag registers in the SuMMIT chip.

Jumper Descriptions

Jumper	Default	Signal	Description
JP1	IN	ACK7	Acknowledge Level 5 Level 5
JP2	OUT	ACK6	DMA Acknowledge Level 6
JP3	OUT	ACK7	DMA Acknowledge Level 7
JP9	IN	DRQ5	DMA Request Level 5
JP10	OUT	DRQ6	DMA Request Level 6
JP11	OUT	DRQ7	DMA Request Level 7
JP4	IN	IRQ10	Interrupt Request Level 10
JP5	OUT	IRQ11	Interrupt Request Level 11
JP6	OUT	IRQ12	Interrupt Request Level 12
JP7	OUT	IRQ15	Interrupt Request Level 15
JP8	OUT	IRQ14	Interrupt Request Level 14
Address Decode for On Board Control			
JP12	IN	A4_H	Base Address (COM-1250/1251_BA)
JP13	IN	A5_H	A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
JP14	IN	A6_H	-- -- -- -- -- -- -- -- -- --
JP15	IN	A7_H	1 1 0 0 0 0 X X X X - 300
JP16	OUT	A8-H	
JP17	OUT	A9_H	
JP28	OUT	OUT	External Clock Connector

Jumpers for SuMMIT/Bus 0			
JP18	IN	RTPTY _H	- RT Parity
JP19	OUT	RTA0 _H	- RT Address 0
JP20	IN	RTA1_H	- RT Address 1
JP21	IN	RTA2_H	- RT Address 2
JP22	IN	RTA3_H	- RT Address 3
JP23	IN	RTA4_H	- RT Address 4
JP24	OUT	LOCK_L	SuMMIT Configuration LOCK - No LOCK
JP25	IN	A/BSTD	1553A - OUT 1553B - IN
JP26	OUT	MSEL1_H	Mode Select JP26 JP27 IN OUT RT Mode IN IN BC Mode OUT IN MT Mode OUT OUT RT/MT Mode
JP27	IN	MSEL0_H	
JP29	IN	RTPTY _H	- RT Parity
JP33	IN	RTA3_H	- RT Address 3
JP34	IN	RTA4_H	- RT Address 4
JP35	OUT	LOCK_L	SuMMIT Configuration LOCK - No LOCK
JP36	IN	A/BSTD	1553A - OUT 1553B – IN
JP37	OUT	MSEL1_H	Mode Select JP37 JP38 IN OUT RT Mode IN IN BC Mode OUT IN MT Mode OUT OUT RT/MT Mode
JP38	IN	MSEL0_H	

Jumper Configuration for the COM-1250/COM-1251

Chapter 4 Operational Description

Hardware Installation

Procedure

Please refer to the Hardware Operator's Guide for your particular system for detailed instructions and illustrations about the installation and removal of option modules.

Pay careful attention to the precautions, including these:

MAKE CERTAIN THE SYSTEM POWER IS OFF. BEFORE TOUCHING ANYTHING INSIDE THE SYSTEM UNIT OR REMOVE THE CARD FROM ITS ANTI-STATIC BAG, PLACE AN ANTI-STATIC STRAP AROUND YOUR WRIST.

1. If necessary, follow the instructions that came with your PC/104 software to shut down the software.

2. Place the loop on an anti-static wrist strap around your wrist and clip the other end to the metal frame that encloses the system unit.

3. Install the card on the PC/104 stack.

4. The user must solder a 75 Ohm Twisted Shielded wire pair to the points shown in Figure 5. The other ends of these wires need to be soldered to 1553 bus connectors.

Note: The center connector pin for the 1553 primary bus goes to pin 4 on the interface connector and the inner shell pin goes to pin 8. The center connector pin for the 1553 secondary bus goes to pin 3 on the interface connector and the inner shell pin goes to pin 7.

Figure 5 depicts the connection for the two 1553 buses.

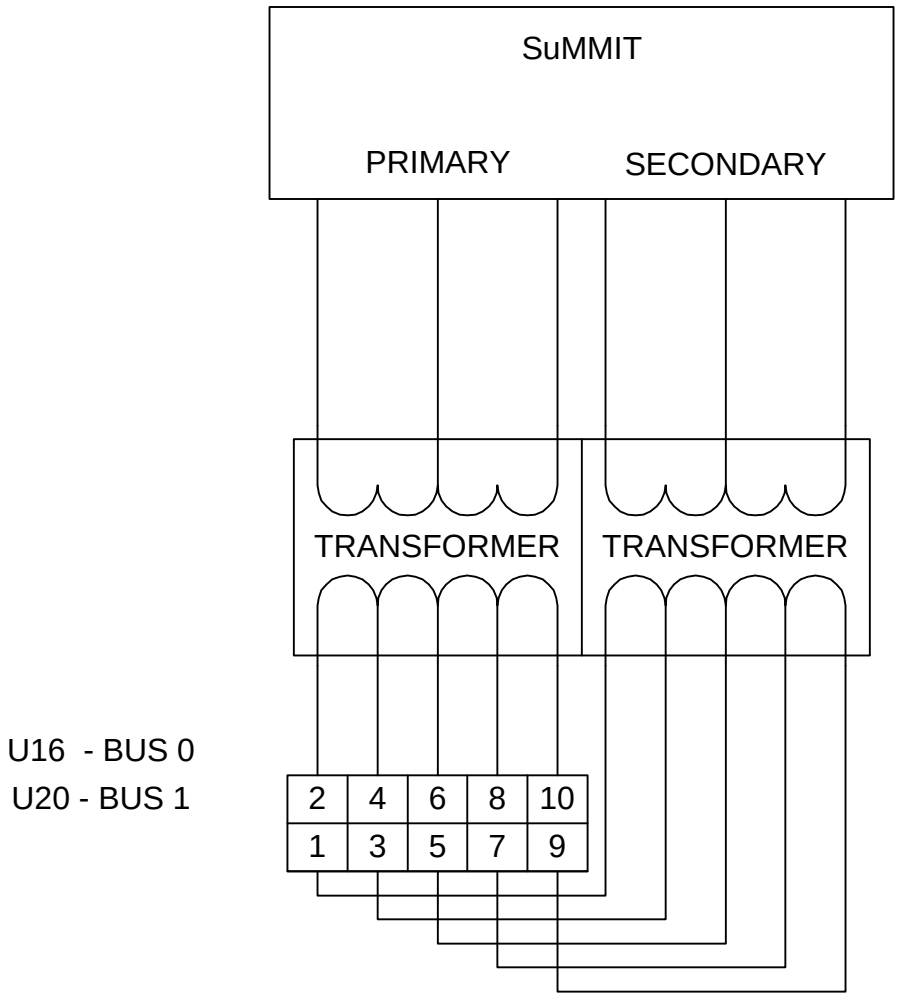


Figure 5: Connecting to the 1553 Buses

Testing the Hardware Installation

Test Routines

To be sure that your new COM-1250/COM-1251 module has been installed correctly and that it is working properly, you will want to run a test.

We have provided three test routines on the software diskette. These test routines will allow the card to act as a simple monitor, Bus Controller or Remote Terminal. Copy the files on the floppy disk to a hard disk, if desired. The user can run the RT.EXE program to test the board as an Remote Terminal (RT). The user can also run the MT.EXE program to test the board as a Monitor (MT) or the BC.EXE program to test the board as a Bus Controller. We have provided the source code for these programs as user examples.

To run tests on the unit, the user needs to have a 1553 test device that can provide bus traffic, or act as a Bus Controller. The user can run the RT or Monitor programs to verify the correct operation of the 1553 Busses unit using the test device to generate traffic or commands.

Programmers Guide

Parvus has developed a application called p1553.exe for Windows 95/98/NT/2000/XP which provides access to all its functionality. A .dll file is available as well as source code for this application. Platform independent source code written in C is also available. Go to http://parvus.com/support to download the software.

The card uses four/eight 16 bit register locations which allow complete access to the functions of the interface.

The card also supports DMA accesses to the PDPM. DMA operations will transfer partial images of the PDPM to and from the computer's memory. The DMA takes 16 bit words and places them on 16 bit boundaries in the computer and vice-versa.

CONTROL, STATUS & I/O REGISTERS

Table 2 shows the 1553 Interface bit map for the four/eight user registers which are mapped into the PC/104 register space. The Register Space address is selected by Jumpers JP12 - JP17.

The following Control, Status and I/O Registers are mapped into the I/O space of the PC/104 bus for controlling the 1553 Board. Each Register is 16 bits wide. There are four/eight registers on the card.
Bus 0 and Bus 1 share a common Control and Status Register (CSR). The COM-1250 only uses Bus 0, but on the COM-1251 the CSR can be read or written from either of the two addresses. Bus 0 and Bus 1 each have three unique register locations for setting the address and accessing the SuMMIT chips or the Memory.

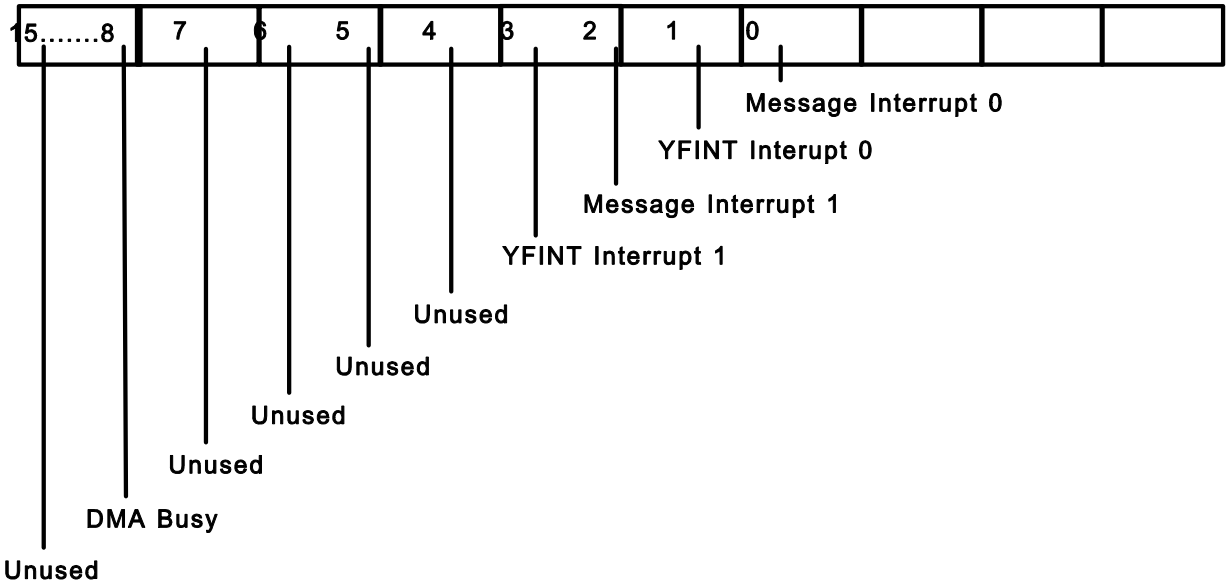
Shared Control and Status Reg	Base Address + 0
Bus 0 Address Register	Base Address + 2
Bus 0 SuMMIT Indirect Access	Base Address + 4
Bus 0 MEMORY Indirect Access	Base Address + 6
Shared Control and Status Reg	Base Address + 8
Bus 1 Address Register	Base Address + 10
Bus 1 SuMMIT Indirect Access	Base Address + 12
Bus 1 MEMORY Indirect Access	Base Address + 14

On board Registers and Addresses

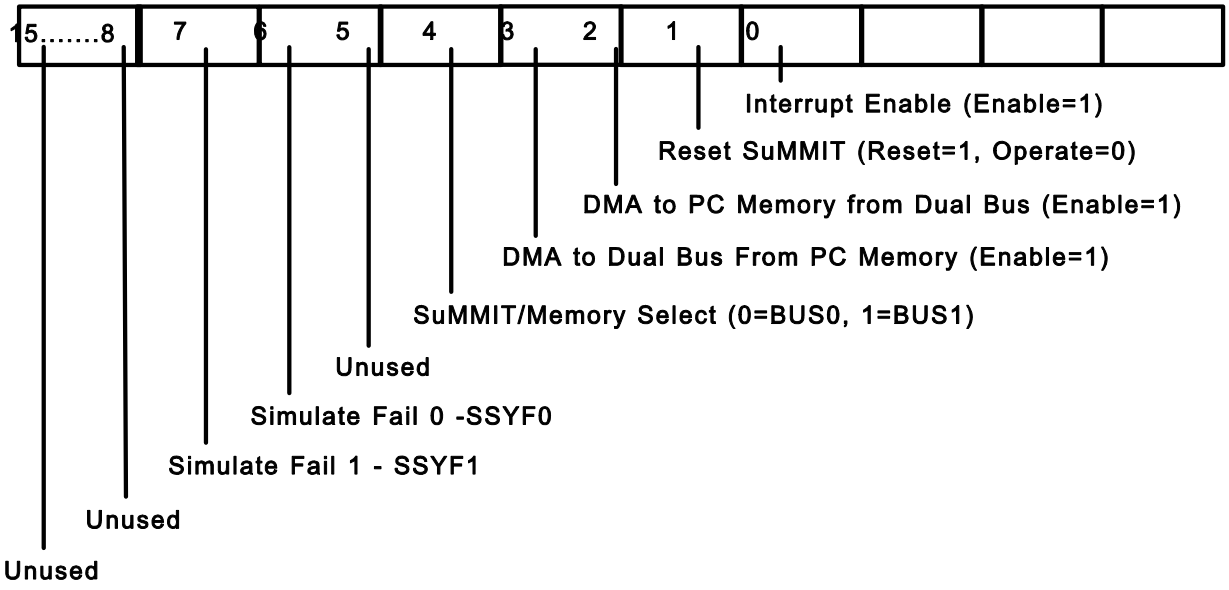
The following is a description of the bits within each of the COM-1250/COM-1251 board locations:

Both Buses/SuMMIT=s share a common board CSR as follows:
Base Address = Dual Bus_BA = 0x300 (also Found at Dual Bus_BA+8 0x308) (Factory Default)

Dual Bus CSR Description for Reads from the Dual Bus CSR:



Dual Bus CSR Description for Writes to the Dual Bus CSR:



[Dual Bus_BA + 2 (0x302) Bus 0] [Dual Bus_BA +8 (0x30A) Bus1]
DMA and Memory/SuMMIT Address Register
for Indirect I/O - Read/Write

|15 |14 |13 |12 |11 |10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

[Dual Bus_BA + 4 (0x304) Bus 0] [Dual Bus_BA + 12 (0x30C) Bus 1]
Read/Write SuMMIT at Indirect Address

|15 |14 |13 |12 |11 |10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

[Dual Bus_BA + 6 (0x306) Bus 0] [Dual Bus_BA + 14 (0x30E) Bus 1]
Read/Write Memory at Indirect Address

|15 |14 |13 |12 |11 |10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Control and Status Register (CSR)

Dual Bus_BA = 0x300, 0x308 (Factory Default)

CSR Register Read:

Bit 0,1:
Message Interrupt - These bits will be set when the SuMMIT chip sources a Message interrupts are enabled, the COM-1250/COM-1251 card will interrupt the processor at the interrupt level selected by jumpers JP4 through JP8. The SuMMIT chip requires special programming for this signal to assert.

Bit 4,5:
YF Interrupt - These bits will be set when the SuMMIT chip sources a YF Interrupt. If the interrupts are enabled, the COM-1250/COM-1251 will interrupt the processor at the interrupt level selected by jumpers JP4 through JP8. The SuMMIT chip requires special programming for this signal to assert.

Bit 8:
DMA Busy - This bit will be set when the DMA starts to take place. When the number of words has been transferred by a DMA transaction to or from the PDPM the bit will be cleared. The software can monitor this bit to poll for end-of-DMA. No Interrupt is generated.

Bits 2,3,6,7,9-15:
These bits are undefined.

CSR Register Write:

Bit 0:
Interrupt Enable - Writing this bit will cause the COM-1250/COM1251 to issue an interrupt whenever a Message, or YF interrupt asserts. The interrupt will occur on the level selected by jumpers JP4 through JP8. The COM-1250/Com-1251 will re-arm its interrupt level when the software clears this bit by writing a zero to bit 0. Therefore, an interrupt service routine must clear, then set this bit after servicing the interrupt. The Interrupt support functions on the processor board must also be properly programmed for interrupts to be recognized by the processor.

Bit 1:
Reset SuMMIT - Writing a 1 to this bit will cause a hardware reset to both SuMMIT chips. The user must write a 1 then write a 0 to this bit to affect the reset. After a 1 is written, the software should wait 1 millisecond before writing a 0.

Bit 2:
DMA to PC Memory from Dual Bus - Enable - Writing this bit with a 1 will cause the COM-1250/COM-1251 to commence a DMA transfer of data from the PDPM to the processor memory. The user software needs to program the DMA engine on the processor to cause the DMA to work correctly. The data will be transferred from the memory associated with the selected SuMMIT chip as specified in bit 4.

Bit 3:
DMA to Dual Bus From PC Memory - Enable - Writing this bit with a 1 will cause COM-1250/COM-1251 card to commence a DMA transfer of data to the PDPM from the processor memory. The user software needs to program the DMA engine on the processor to cause the DMA to work correctly. The data will be transferred to the Memory associated with the selected SuMMIT chip as specified in bit 4.

Bit 4:
This bit selects which SuMMIT/MEMORY is selected for the DMA operation. 0 - Bus 0, 1 - Bus 1

Bit 5:
Not Used

Bits 6 - 7:
These bits allow the user to simulate a Sub-System Failure to the Bus Controller. Bit 6 corresponds to Bus 0 and Bit 7 to Bus 1.

Bits 8 - 15:
These bits are not used.

DMA and Memory/SuMMIT Address Register

for Indirect I/O - Read/Write Dual Bus_BA + 2 Bus 0
Dual Bus_BA + 10 Bus 1

These registers are used to hold the beginning 16 bit address of the Pseudo Dual-Ported Memory (PDPM). It is also used to hold the address for access to the SuMMIT devices. The software must first write the address register associated with the selected bus and then can access the memory or SuMMIT device through the indirect register. This registers automatically increment after each access. The address will wrap-around to zero if an access causes the address to increment beyond a 16 bit value.

Read/Write SuMMIT at Indirect Address

Dual Bus_BA + 4 Bus 0
Dual Bus_BA + 12 Bus 1

The software uses these 16 bit locations to communicate with the two SuMMIT chips. These register address causes the hardware to access the SuMMIT chip at the selected address. The address register will auto-increment after each access to this location. To access a block of registers in the SuMMIT chip, the software must first write the beginning address of the block, followed by repeated access to this location for access to successive registers within the SuMMIT device.

Read/Write Memory at Indirect Address

Dual Bus_BA + 6 Bus 0
Dual Bus_BA + 14 Bus 1

The software uses this 16 bit location to communicate with the Pseudo Dual Ported Memory (PDPM) associated with each SuMMIT device. The address of the location in the PDPM is taken from the DMA and Memory/SuMMIT Address Registers. The address register will auto-increment after each access to this location. To access a block of memory in the PDPM, the software must first write the beginning address of the block, followed by repeated access to this location for access to successive locations within the PDPM. The PDPM has 64K 16 bit words. This register also acts as the beginning DMA address for a DMA transfer between the PDPM and the processor's memory.

SOFTWARE PROCEDURES

The following procedures indicate the programming steps which cause DMA transactions to occur between the Pseudo Dual-Ported Memory (PDPM) and the processor's memory.

DMA From the processor's Memory to the Pseudo Dual-Ported Memory.

1. Select the desired PDPM memory in the CSR register.

2. Write the PDPM memory DMA address register with the beginning location of the DMA transaction.

3. Program the PC/104 DMA Engine with the processor's memory address and necessary control bits.

4. Set bits 3 and 0 in the Control and Status Register (Dual Bus_BA = 0x300 Factory Default). This will cause the DMA to start and the DMA finish bit to assert when the DMA is finished.

5. The software should clear bits 3 and 0 in the Control and Status Register after the DMA is complete.

DMA to the computers Memory from the Pseudo Dual-Ported Memory.

1. Select the desired PDPM memory in the CSR register.

2. Write the PDPM memory DMA address register with the beginning location of the DMA transaction.

3. Program the PC/104 DMA Engine with the processor's memory address and necessary control bits.

4. Set bits 2 and 0 in the Control and Status Register (Dual Bus_BA = 0x300 Factory Default). This will cause the DMA to start and the DMA finish bit to assert when the DMA is finished.

5. The software should clear bits 2 and 0 in the Control and Status Register after the DMA is complete.

Windows Software

This software utility is provided to give the user a functional tool that will operate with the PC/104 1553 board. This is the same program used by Parvus to test the product during manufacturing. This tool allows the user to verify the functionality of the board prior to, and anytime during, development of custom software. In most case the implementation of these cards into a system will require the user to create their own custom application software.
This software utility and source code can be downloaded from http://parvus.com/support

Installation

To install the Windows software, simply extract the zip file and run Setup.exe. Then follow the onscreen procedure to install the software.

Software Description

A screen shot of the running program is shown below.

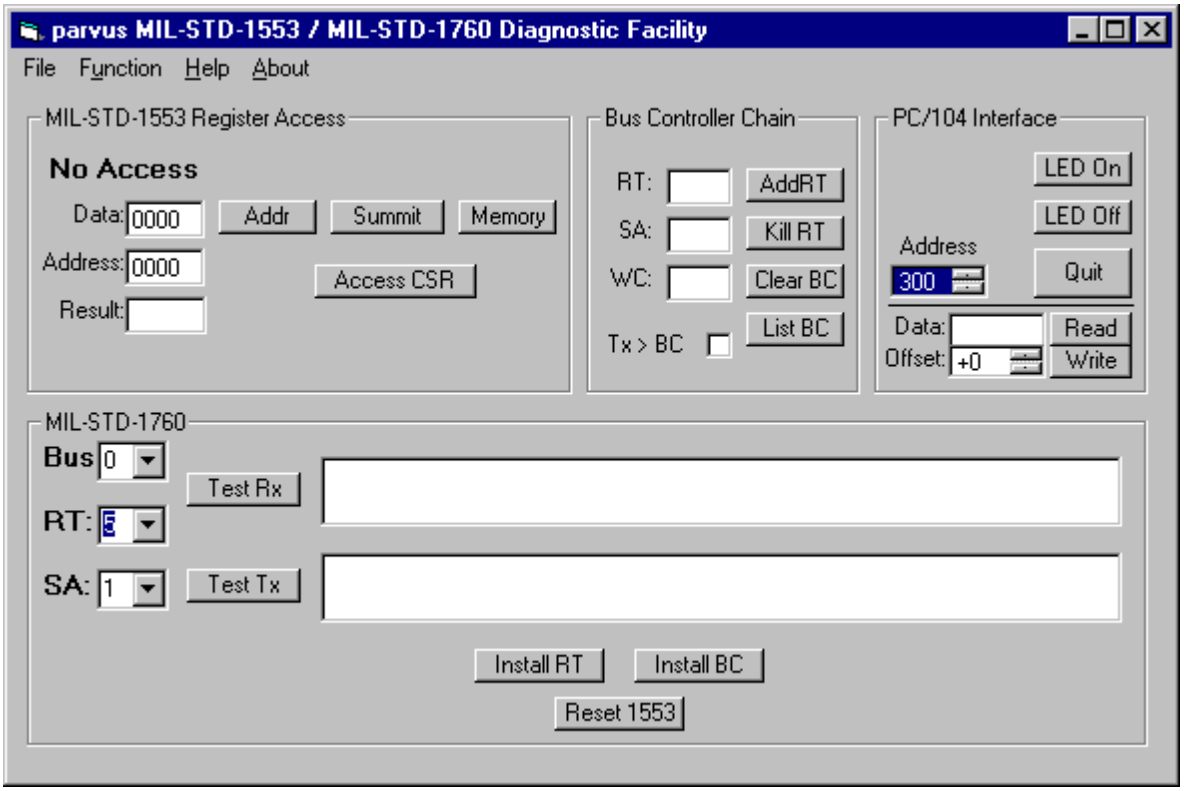


Figure 6 – The Running Application

The program window is divided into four sections: MIL-STD-1553 Register Access,Bus Controller Chain, MIL-STD-1760, and PC/104 Interface.

MIL-STD-1553 Register Access

This section provides direct register access.

Address - The “Addr” button is pressed to have direct access to the address register. When “Addr” is pressed any data in the Address text field is written to the address register (base + 2), read back and displayed as result. If the Address text field is left blank, then only the value is read back from the address register and displayed as result.

Summit – The “Summit” button is pressed to have direct access to the Summit register. When “Summit” is pressed any data in the Address text field is written to the address register (base + 2 or base + 10) and any data in the Data Text Field to the Summit Register (Base + 4 or Base + 12). If the Address text field and the Data Text field are left blank, then only the value is read back from the Summit register and displayed as result.

Memory - The “Memory” button is pressed to have direct access to the memory register. When “Memory” is pressed any data in the Address text field is written to the address register (base + 2 or base + 10) and any data in the Data Text Field to the Memory Register (Base + 4 or Base + 12). If the Address text field and the Data Text field are left blank, then only the value is read back from the Summit register and displayed as result.

CSR - CSR is a control status register value. The “CSR” button is pressed to have direct access to the CSR register. When “CSR” is pressed any data in the Data text field is written to the CSR register (base + 0) and read back and displayed as result. If Data text field is left blank, then only the value is read back from the CSR register and displayed as result.

Bus Controller Chain

To add a new RT for the BC to the BC chain, enter the RT, SA and the WC (word count) values and press “Add RT”. An existing RT in the bus chain can be removed by pressing “Kill RT”. “List BC” will show all the existing RT entries in the BC chain (See Figure 6 below). The “Tx > BC” check box is used to select “Tx” or “Rx”. While adding or removing, enabling the checkbox will add or remove the “Tx” (transfer mode), for the particular RT in the in the BC chain. Disabling it will add or remove the “Rx” mode.

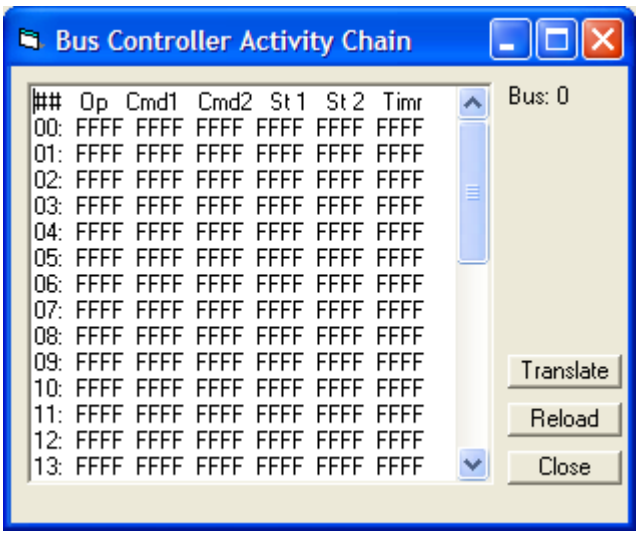


Figure 7 – Bus Controller Activity Chain
Op: Opcode Cmd1: Command Word 1 Cmd2: Command Word 2
St1: Status Word 1 St2: Status Word 2 Timr: Timer

MIL-STD 1760

This section provides the ability to send and receive packets from one system to another. For example, two Computers, A and B, are up and running with the board mounted and connected through the 1553 bus. Start the driver software on the both the computers. One computer will act as RT and the other as BC. Let A be the RT and B be the BC.

On 'A' under the MIL-STD-1760 frame choose the bus=0, RT=5 and SA=1. Press the button "Install RT". Similarly, on 'B' under the MIL-STD-1760 frame, choose the same bus=0, RT=5 and SA=1. Press "Install BC". In this test scenario, by default, on pressing "Install RT" the remote terminal sends 30 words starting from 1300-1330 to the BC.

Testing Tx:
On 'B', press "Test Tx" see the test data sent from RT to the BC, in the text box adjacent to it. Now on 'A', press "Test Tx", and 'No Data' message will appear in the text box. Now Left click once on the text box to see 0000… appear on the text field. Overwrite any of the values on 'A' and press "Test Tx". Now on 'B' press "Test Tx" and the data transferred from 'A' should appear on the test field.

Testing Rx:
Press "Test Rx" on 'A'. In the text filed adjacent to the button, you should see the first two words (non zero command words) and the rest 0000. Next press "Test Rx" on 'B' to see 0000 in the corresponding text field. Click on the test field, and you will see all zeroes displayed. Overwrite the values and hit "Test Rx". Now hit "Test Rx" on 'A', to see the data sent by the BC.

MIL-STD 1553 Windows Software DLL function Description

Common Routines:

Name: **vbapp_Open**
Entry: Channel - Address Group Base, 0-300
Exit: Status of Open
0 = Already Open
1 = Fault Upon Open
2 = Successful Open
Globals: None
Prototype: long WINAPI vbapp_Open(long Channel);
Description: Performs and open of the channel between the board and the application software. Checks for a proper installation of the DLL and WinDriver components. This routine must be called to start the channel. The address group base changes with each card type.

Name: **vbapp_Close**
Entry: None
Exit: None
Globals: None
Prototype: void WINAPI vbapp_Close(void);
Description: Closes and open channel. Performs the close only if the channel is open, performs a no-op if the channel was already closed or not open.

Name: **vbapp_LedOn**
Entry: None

Exit: None
Globals: None
Prototype: void WINAPI vbapp_LedOn(void);
Description: Turns the diagnostic LED located on the PC/104 board on. The diagnostic LED is used as a simple test to see that the board is operational and that the DLL is attached to it. The normal state of the LED is on.

Name: **vbapp_LedOff**
Entry: None
Exit: None
Globals: None
Prototype: void WINAPI vbapp_LedOff(void);
Description: Turns the diagnostic LED located on the PC/104 board off

Name: **vbapp_Read**
Entry: Offset from base address of byte to read in I/O space
Exit: Value read from supplied offset
Globals: None
Prototype: long WINAPI vbapp_Read(long offset);
Description: This is a general-purpose routine for direct access to the board. It works by reading from the base address plus the supplied offset. The returned value is a byte read from the board. This can be used for creating a uniform interface set, diagnostic purposes, or for accessing features potentially missing from the DLL command set.

Name: **vbapp_Write**
Entry: Offset from base address of byte to read in I/O space
Value to write to board
Exit: None
Globals: None
Prototype: void WINAPI vbapp_Write(long offset, long value);
Description: This is a general-purpose routine for direct access to the board. It works by writing to the base address plus the supplied offset. The value is a byte written to the board. This can be used for creating a uniform interface set, diagnostic purposes, or for accessing features potentially missing from the DLL command set.

Name: **vbapp_Test**
Entry: A test value, 0-255
Exit: The test value times 2
Globals: None
Prototype: long WINAPI vbapp_Test(long value);
Description: This routine is used to assure that the DLL is properly attached and that parameters can be passed back and forth. It simply multiplies the supplied value by 2 and returns it.

Name: **vbapp_Version**
Entry: None
Exit: Returns a long value of version, 10102 translates to 1.01.02
Globals: None
Prototype: long WINAPI vbapp_Version(void);
Description: This is used to version lock DLLs to critical applications as well as to simply return the version of the DLL during test.

Name: vbapp_**CardRangesL**
Entry: These are all pointers to values
int* nRanges -

int* nIo - Number of available I/O Ranges
int* nMem - Number of available Memory Ranges
int* ioBase - First element in I/O Range Array
int* ioBytes - First element in I/O Range Size Array
int* ioOffset - Reserved, empty pointer required
int* memBase - First element in Memory Range Array
int* memBytes - First element in Memory Range Size Array
int* memOffset - reserved, empty pointer required
Exit: All pointer values are filled
Globals: None
Prototype: void WINAPI vbapp_CardRangesL (unsigned int* nRanges,
unsigned int* nIo,
unsigned int* nMem,
unsigned int* ioBase,
unsigned int* ioBytes,
unsigned int* ioOffset,
unsigned int* memBase,
unsigned int* memBytes,
unsigned int* memOffset)
Description: By calling this routine at the beginning of the application it is possible to get all of the I/O and memory ranges available for the board.

Application specific Routines:

Name: **vbapp_s1553CSR**
Entry: value
Exit: Returns the value read from the register.
Globals: None
Prototype: long vbapp_s1553CSR (long value)
Description: Control Status Register. If value is -1 then the contents of the CSR register are returned. Else, 'value' is written to the CSR register and read back.

Name: **vbapp_s1553Reset**
Entry: value
Exit: None
Globals: None
Prototype: void vbapp_s1553Reset(int bus);
Description: Reset the Summit chip for the specified bus (0 or 1);

Name: **vbapp_s1553Addr**
Entry: bus, value
Exit: status or value
Globals: None
Prototype: long vbapp_s1553Addr(long bus, long value);
Description: Direct Access of Address Registers. Bus can be selected as a 0 or a 1. If value is -1 then the contents read from the address register are returned, else value is written to the address register and read back. Returns –1 if operation unsuccessful else the value read.

Name: **vbapp_s1553Summit**
Entry: bus,address,value.
Exit: status
Globals: status or value
Prototype: long vbapp_s1553Summit(long bus, long address, long value) ;

Description: Direct Access of Summit Registers. Bus can be selected by passing 0 or 1 as argument. If address is not -1, address is written to the address register. If value is not -1 value is written to the summit register. Value read from the Summit register is returned. -1 is returned if operation was unsuccessful.

Name: **vbapp_s1553SummitWrite**
Entry: bus,value.
Exit: None
Globals: None
Prototype: void vbapp_s1553SummitWrite(long bus, long value);
Description: Direct Write of Summit Registers.

Name: **vbapp_s1553Memory**
Entry: bus,address,value
Exit: status or value read.
Globals: None
Prototype: int vbapp_s1553Memory(long bus, long address, long value);
Description: Direct Access to Memory. Bus can be selected by passing 0 or 1 as argument. If address is not -1 'address' is written to the address register. If value is not -1 'value' is written to the memory location. Returns –1 if operation unsuccessful else the value read.

Name: **vbapp_s1553MemoryWrite**
Entry: bus,value
Exit: None
Globals: None
Prototype: void vbapp_s1553MemoryWrite(long bus, long value);
Description: Direct Write to Memory.

Name: **vbapp_s1553MemoryRead**
Entry: value
Exit: Status
Globals: None
Prototype: long vbapp_s1553MemoryRead(long bus);
Description: Direct Read from Memory. Returns -1 if operation unsuccessful else the value read.

Name: **vbapp_s1553BufferBCTx**
Entry: bus,sa,buffer
Exit: status or change.
Globals: None
Prototype: long vbapp_s1553BufferBCTx(long bus, long sa, long * buffer);
Description: Read buffer from RT to BC, indicate if changed. Buffer must be 2x32 because age data is stored in upper section. Data starting at sub address 'sa' is read into the buffer. Age data is copy of the data stored. It is used to check if data in the location has changed. Returns 1 if data has changed, 0 otherwise and -1 if operation unsuccessful.

Name: **vbapp_s1553BufferRTRx**
Entry: bus,sa,buffer
Exit: status or change
Globals: None
Prototype: long vbapp_s1553BufferRTRx(long bus, long sa, long *buffer);
Description: Read buffer from BC to RT, indicate if changed. Buffer must be 2x32 because age data is stored in upper section. Data starting at sub address 'sa' is read into the Buffer .Buffer[0] and buffer[1] will contain control word and time lag respectively. Returns if data has changed, 0 otherwise and -1 if operation unsuccful.

Name: **vbapp_s1553BufferBCRx**
Entry: bus,sa,buffer,numwords
Exit: status
Globals: None

Prototype: long vbapp_s1553BufferBCRx(long bus, long sa, long* buffer, long numwords);
Description: Send buffer to RT from BC. Buffer length must be 32 even if numwords is less than 32. Buffer contents are written to the memory starting at the sub address 'sa'. Returns 0 on successful operation -1 otherwise

Name: **vbapp_s1553BufferRTTx**
Entry: bus,sa,buffer,numwords
Exit: status
Globals: None
Prototype: long WINAPI vbapp_s1553BufferRTTx(long bus, long sa, long* buffer, long numwords);
Description: Send buffer from RT to BC. Returns 0 on successful operation -1 otherwise. Buffer length must be 32 even if numwords is less than 32.

Name: **vbapp_s1553LoadBC**
Entry: bus,rt
Exit: None
Globals: None
Prototype: void vbapp_s1553LoadBC(long bus, long rt1);
Description: 1553 File transfer protocol, BC Side;

Name: **vbapp_s1553LoadRT**
Entry: bus,rt,sa
Exit: None
Globals: None
Prototype: void vbapp_s1553LoadRT(long bus, long rt, long sa);
Description: 1553 File transfer protocol, RT Side.

Name: **vbapp_s1553IgnoreBC**
Entry: bus,rt,sa,tr
Exit: status
Globals: None
Prototype: long vbapp_s1553IgnoreBC(long bus, long rt1, long sa1, long tr);
Description: Ignore previously assigned RT in BC chain. Returns 0 on successful operation -1 otherwise.

Name: **vbapp_s1553ClearBC**
Entry: bus
Exit: status
Globals: None
Prototype: long vbapp_s1553ClearBC(long bus);
Description: Clear all managed RTs in BC chain. Returns 0 on successful operation -1 otherwise.

Name: **vbapp_s1553AddBC**
Entry: bus,rt,sa,wc,tr
Exit: status
Globals: None
Prototype: long vbapp_s1553AddBC(long bus, long rt1, long sa1, long wc1, long tr);
Description: Add RT to BC chain. Returns 0 on successful operation -1 otherwise.

Name: **vbapp_s1553ListBC**
Entry: bus, buffer
Exit: status
Globals: None
Prototype: long vbapp_s1553ListBC(long bus, long* buffer);
Description: Reads status of the BC Chain. Updates the buffer, and returns 0 on successful operation -1 otherwise.

Linux Software

Mil-Std-1553 Linux Remote Terminal Driver

The 1553 Linux Remote Terminal driver is designed specially for implementing 1553 remote terminal function on a one Summit 1553 chipset Parvus pc/104 1553 interface board. The driver also includes a RT test program "rttest", which implements remote terminal buffer receive and buffer transmit. This is intended to be compiled together with the files "rttest.c" and kbhit.c. This remote terminal code is intended to be used with a higher level layer to access the low level device driver, e.g. on board I/O or memory. Therefore, an application program to use this must begin with a following code.

```
iopl(3);
```

The program must be run as root to enable I/O.

A make file is included to allow the user to compile and link this driver. It will compile and generate the shared library object code and link to files with extension so.

```
[username]$make
```

To load the shared object code to a library dynamically, use following command:

```
[username]$LD_LIBRARY_PATH=$(pwd)
```

To execute, the program must be run as root. Use following commands:

```
[username]$su
[username]$<password>
[root]#./rttest bus_number
```

Note: Bus number is 0 is the first bus on either single board or dual board. Bus number 1 is the second bus on dual boards only.

After a successful compile and link, the user can run a test program "rttest". It has a menu to let user select operations for a remote terminal (RT) mode. In RT mode, it has three commands. First is a RT node received a command from BC which its address was matched and indicates the RT node to receive data that follows up. Second is a RT node transmitting buffered data to BC after receiving a BC transmit command which indicates the RT address matches to it. Third operation is tests the 1553 board LED command to toggle the on board LED in order to test the board is on line. Please follow the select menu instruction to select and run the examples.

The user can also compile and link the code by command line as below example. It must be compiled with -O for outb to be inlined, otherwise a link error will be generated.

```
gcc -fPIC -g -O -Wall -c m53rt.c -o m53rt.o
gcc -g -O -Wall -shared -Wl,-soname,m53rt.1 -o m53rt.so.1.0.0 m53rt.o -lc
ln -s m53rt.so.1.0.0 m53rt.so.1
ln -s m53rt.so.1.0.0 m53rt.so
```

Mil-Std-1553 Linux Driver Library

The 1553 Linux driver implements the Mil-Std-1553 functions on a Parvus PC/104 1553 single interface board. This board has either one Summit 1553 chipset or a dual interface board with two Summit 1553 chipsets. The driver also encloses a test program "libtest", which can implement either Bus Controller mode or Remote Terminal mode to test the library functionalities. This library is intended to be compiled together with the files "libtest.c" and kbhit.c. The kbhit.c is only used in libtest program to a formatted screen output. This library is also intended to be used from a user mode level to access the low level I/O device port, e.g. on board I/O or memory. Therefore, an application program must be run as root to enable I/O, and it must begin with a following code.

```
iopl(3);
```

or

```
ioperm(BASE, 16, 1);
```

A make file is included. The 1553 device driver will be compiled and generate a shared library object code and link to files with extension so. To compile and link the driver, use following command.

```
[user]$make
```

After compiling and linking, then the shared library code can be dynamically loaded into the Linux libraries. For executing a program, use following command lines;

```
[user]$LD_LIBRARY_PATH=$(pwd)
[user]$super
[user]$<password>
[root]#./libtest bus
```

Note: The bus number 0 is the first bus either on single board or dual board. The bus number 1 is the second bus on a dual chip board.

The program displays a menu to let user select different modes: Bus Controller (BC) mode or remote terminal (RT) mode. BC mode has three commands: first BC sends data to a RT node, second BC instructs a RT node to transmit data to BC which BC will receive the data from RT node, third it commands two RT nodes that one for transmitter and another for receiver. Besides these operation, there is also a command to toggle a LED on board in order to test the board is on line.

In a RT mode, there are two commands to implement RT functions that are transmit data or receive data from BC. It also has a LED test command as same as BC does. Please follow the select menu instruction to select and run the examples.

User also can compile and link the library by command line as below example. IT must be compiled with -O for outb to be inlined, otherwise a link error will be generated.

```
gcc -fPIC -g -O -Wall -c m53lib.c -o m53lib.o
gcc -g -O -Wall -shared -Wl,-soname,m53lib.1 -o m53lib.so.1.0.0 m53lib.o -lc
ln -s m53lib.so.1.0.0 m53lib.so.1
ln -s m53lib.so.1.0.0 m53lib.so
```

Mil-Std-1553 Linux Bus Controller Driver

This 1553 Linux Bus Controller (BC) driver is designed specially for implementing 1553 bus controller function on a one Summit 1553 chipset Parvus pc/104 1553 interface board. The driver also includes a BC test program "bctest", which implements bus controller commands. This is intended to be compiled together with the files "bctest.c" and kbhit.c. This bus controller code is intended to be used with a higher level layer to access the low level device driver, e.g. on board I/O or memory. Therefore, an application program to use the code must begin with a following code.

```
iopl(3);
```

The program must be run as root to enable I/O.

A make file is included to allow the user to compile and link this driver. It will compile and generate the shared library object code and link to files with extension so.

```
[username]$make
```

To load the shared object code to a library dynamically, use following command:

```
[username]$LD_LIBRARY_PATH=$(pwd)
```

To execute, the program must be run as root. Use following commands:

```
[username]$su
[username]$<password>
[root]#./bctest bus_number
```

Note: Bus number is 0 is the first bus on either single board or dual board. Bus number 1 is the second bus on dual boards only.

After a success compile and link, the user can run a test program "bctest". The program has a menu to let user select operations for a bus controller (BC) test. BC mode has three commands. First is BC send out a Tx command with a RT address and a sub address to a specific RT node, which instructs the selected RT node to transfer a certain number of data to BC. Second is BC send out a Rx command with a RT address and a sub address to instructs the specific RT to receive a certain number data from the 1553 bus. The third command is BC send out two RT addresses and sub addresses for two RT nodes, which one RT is going to receive data from the 1553 bus and another RT is going to transfer a certain amount of data. There is also a test LED command that the 1553 board LED can be toggled on board in order to test the board is on line. Please follow the select menu instruction to select and run the examples.

User also can compile and link bus controller code by command line as in the example below. It must be compiled with -O for outb to be inlined, otherwise a link error will be generated.

```
gcc -fPIC -g -O -Wall -c m53bc.c -o m53bc.o
gcc -g -O -Wall -shared -Wl,-soname,m53bc.1 -o m53bc.so.1.0.0 m53bc.o -lc
ln -s m53bc.so.1.0.0 m53bc.so.1
ln -s m53bc.so.1.0.0 m53bc.so
```

Chapter 5 Specifications

This chapter provides the specifications for the COM-1250 / 1251.

Technical Specification

Electrical

Requirements

- +5V +/ 0.25 Volts DC
- 600 Milliamps receiving
- 1.6 Amps w/ both buses transmitting

Environmental Specifications

This section describes the environmental standards that the COM-1250 / 1251 has either been designed to meet or tested to.

	Operating	Storage
Temperature	-40 to +85	
Humidity	10 to 80%, nc	10 to 95%, nc

Reliability (MTBF)

The following table shows the predicted values for reliability of the COM-1250 / 1251 as a Mean Time Between Failures (MTBF).

Test Standard	MTBF (Hours)
Ground Benign, Controlled GB, GC	856,433
Airborne Rotary Winged, ARW	60,164
Airborne Inhabit Fighter, AIF	157,971

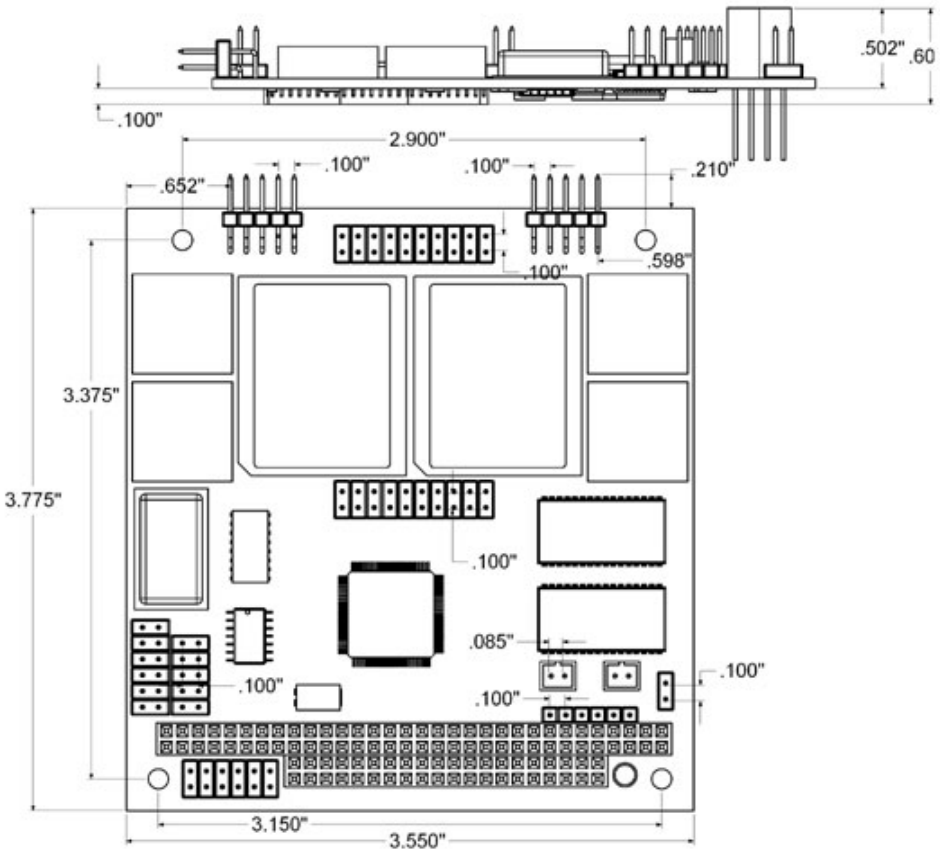
Mechanical

This section provides details related to the mechanical construction of the COM-1250 / 1251.

Dimensions

- 3.550” x 3.775”

The following diagrams provide the physical dimensions of the COM-1250 / 1251.



Dimensions are in millimeters

Chapter 6 Troubleshooting

Technical Assistance

If you have a technical question or if you cannot isolate a problem with your product, please call or e-mail the Parvus Technical Support team:

- Email: tsupport@parvus.com
- Phone: +1 (801) 433-4322
- Fax: +1 (801) 483-1523

Returning For Service

Before returning any Parvus product, please fill out a Returned Material Authorization (**RMA**) request form available for download from the following website under the support section:

www.parvus.com

Email this form to the email address listed above to receive authorization for shipment. An RMA number will be emailed back to you as soon as possible.

Note. You must have the RMA number in order to return any product for any reason.

Pack the module in an anti-static material and ship it in a sturdy cardboard box with enough packing material to adequately cushion it.

Warning! Any product returned to Parvus improperly packed will immediately void the warranty for that particular product!

Chapter 7 Contact Info

Main Phone: +1 (801) 483-1533
Fax: +1 (801) 483-1523

Sales
+1(800) 483-3152 or (801) 483-1533,
sales@parvus.com

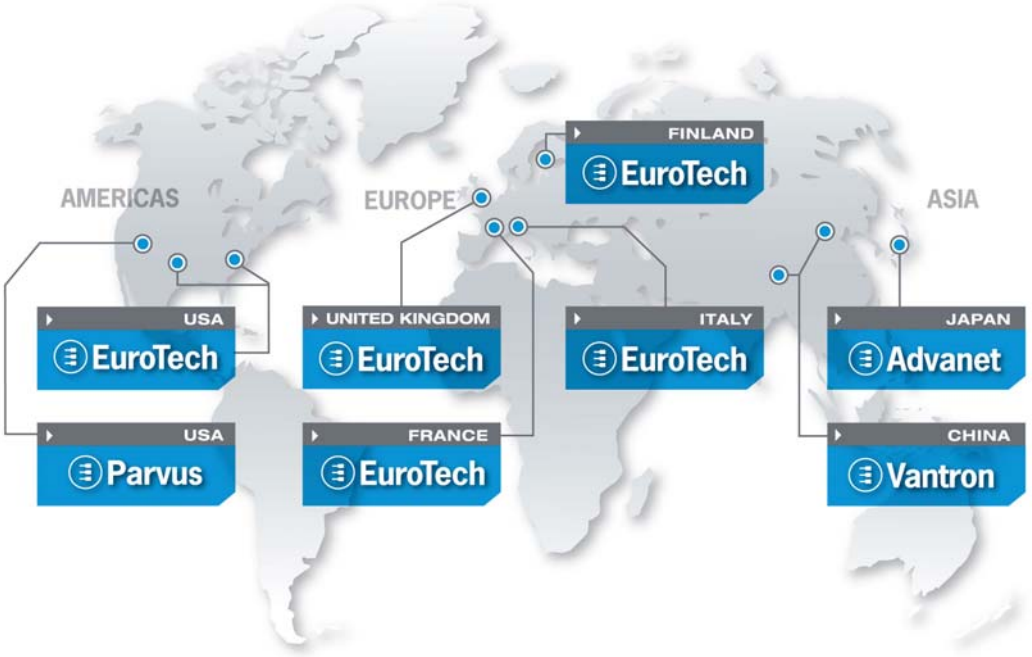
Product Technical Support
+1 (801) 433-6322,
tsupport@parvus.com

Customer Feedback
feedback@parvus.com

Company contact info:

Parvus® Corporation
3222 S. Washington St.
Salt Lake City, Utah, USA 84115
(801) 483-1533, FAX (801) 483-1523
Web-site: http://www.parvus.com

Eurotech Group Worldwide presence



AMERICAS

USA

EUROTECH
Toll free +1 888.941.2224
Tel. +1 301.490.4007
Fax +1 301.490.4582
E-mail: sales.us@eurotech.com
E-mail: support.us@eurotech.com
Web: www.eurotech-inc.com

PARVUS
Tel. +1 800.483.3152
Fax +1 801.483.1523
E-mail: sales@parvus.com
E-mail: tsupport@parvus.com
Web: www.parvus.com

EUROPE

Italy

EUROTECH
Tel. +39 0433.485.411
Fax +39 0433.485.499
E-mail: sales.it@eurotech.com
E-mail: support.it@eurotech.com
Web: www.eurotech.com

United Kingdom

EUROTECH
Tel. +44 (0) 1223.403410
Fax +44 (0) 1223.410457
E-mail: sales.uk@eurotech.com
E-mail: support.uk@eurotech.com
Web: www.eurotech.com

France

EUROTECH
Tel. +33 04.72.89.00.90
Fax +33 04.78.70.08.24
E-mail: sales.fr@eurotech.com
E-mail: support.fr@eurotech.com
Web: www.eurotech.com

Finland

EUROTECH
Tel. +358 9.477.888.0
Fax +358 9.477.888.99
E-mail: sales.fi@eurotech.com
E-mail: support.fi@eurotech.com
Web: www.eurotech.com

ASIA

Japan

ADVANET
Tel. +81 86.245.2861
Fax +81 86.245.2860
E-mail: sales@advanet.co.jp
E-mail: tsupport@advanet.co.jp
Web: www.advanet.co.jp

China

VANTRON
Tel. +86 28.85.12.39.30
Fax +86 28.85.12.39.35
E-mail: sales@vantrontech.com.cn
E-mail: support.cn@eurotech.com
Web: www.vantrontech.com.cn

Parvus

www.parvus.com