



Blue Wave Systems

DBV46

TMS320C4x Carrier Board

Technical Reference Manual

Version 1.03 June 1998

Blue Wave Systems is the result of a merger of Loughborough Sound Images Ltd and Mizar Inc. Any remaining references to Loughborough Sound Images or Mizar in this manual (and/or the product it refers to) can be regarded as referring to Blue Wave Systems.



All Blue Wave Systems hardware must be protected against static discharge. Appropriate precautions, such as wearing the anti-static wrist strap provided, must be taken at all times when handling the hardware.

This product conforms to the Electromagnetic Compatibility (EMC) Regulations as stated by EC Directive 89/336/EEC and the amending Directive 92/31/EEC. Testing has been performed with the product inserted in EMC compliant equipment (PC or VME rack) with all panels in place. The Declaration of Conformity for this product is held by Blue Wave Systems.

Note that in order to conform to the EMC regulations when using any of the front panel communication ports, the communication port cabling must be kept inside the EMC enclosure. Screened cables should be used if external cabling is required.



Blue Wave Systems work to procedures which have been designed to comply with the requirements of ISO9001 and TickIT. Blue Wave have been assessed against these procedures and registered as meeting the requirements of the standard (Certificate No. 95/5633).

All trademarks and registered trademarks are acknowledged.

© 1998 Blue Wave Systems

This publication is issued to provide outline information only, which (unless agreed by the company in writing) may not be used to form part of any order or be regarded as a representation relating to products or services concerned. Blue Wave Systems reserve the right to alter, without notice, the specification, design, price or conditions of supply of any product or service.

Contents	Page
1 Introduction	1
1.1 Conventions	2
1.2 Using This Manual	3
1.3 Overview	4
1.3.1 Processing Nodes B and C: On-Board C40 Processors	6
1.3.2 Processing Nodes A and D: TIM-40 Modules	7
1.3.3 Processor Global Memory Organisation	8
1.3.4 VMEbus Master/Slave Interface	9
1.3.5 Shared Bus Architecture	10
1.3.6 Parallel Expansion - C4x Communication Ports	11
1.3.7 PIM Expansion	11
1.3.8 JTAG Emulation System	12
2 Board Layout and Installation	13
2.1 Layout	13
2.2 Installation	16
2.3 Booting	18
3 Processing Nodes B and C: On-Board C40 Processors	19
3.1 Introduction	19
3.2 Local Memory Interface	20
3.2.1 Local SRAM	21
3.2.2 PEROM	21
3.3 Global Memory Interface	21
3.3.1 Global SRAM	22
3.3.2 Shared Bus Resources	22
3.4 Optimising Performance	24
3.5 Memory Interface Control Registers	24
3.5.1 Local Memory Interface Control Register	25
3.5.2 Global Memory Interface Control Register	26
3.6 Booting	27
3.6.1 Boot Options	27
3.6.2 Blue Wave System's PEROM Bootcode	29
3.6.3 Bootcode Corruption	30
3.7 Programming the 32K PEROM Device from the C40	32
3.7.1 Software Data Protection Algorithm	33
3.8 Interrupts	36

4	Processing Nodes A and D: TIM-40 Modules	37
4.1	Introduction	37
4.2	Global Memory Interface	38
4.2.1	Global SRAM	38
4.2.2	Shared Bus Resources	39
4.2.3	Global Memory Interface Control Register	41
4.3	Optimising Performance	43
4.4	Booting	44
4.5	Interrupts	44
5	Shared Bus Architecture	45
5.1	Introduction	45
5.2	DBV46 Control Registers	48
5.2.1	System Reset Register	49
5.2.2	System Control Register	50
5.2.3	LED Control Register	52
5.2.4	Communication Port Control Register	53
5.2.5	System Status Register	54
5.2.6	Node Identification Register	55
5.2.7	Shared Bus Lock Register	56
5.2.8	Timeout Error Register	57
5.3	DBV46 Interrupt Registers	57
5.3.1	Interrupt Enable Registers	58
5.3.2	Interrupt Pending Registers	60
5.3.3	Interrupt Line Status Registers	62
5.3.4	Interrupt Condition Status Registers	64
5.4	SCV64 Register Set	66
5.5	TBC Register Set	69
5.6	DBV46 Shared Memory Resources	70
5.6.1	Shared DRAM	70
5.6.2	Shared PEROM	71
5.7	Private Global Memory Resources	75
5.8	Shared Bus Arbitration	77
5.8.1	Arbitration	77
5.8.2	Timeout	78
5.8.3	Bus Locking	79
5.8.4	Deadlocks	80
6	VMEbus Slave Interface	81
6.1	VME Slave Access Configuration	82
6.1.1	Supported Transfers	84
6.2	The SCV64 Location Monitor	86
6.3	VME Interrupts to DBV46	87
6.4	Access Protection	89

7	VMEbus Master Interface	91
7.1	Overview	91
7.2	DBV46 System Control Register	94
7.3	Setting Address Modifiers on VME Master Cycles	95
7.4	Interrupts	101
7.4.1	Interrupts to the VMEbus	101
7.4.2	SCV64 Interrupts	102
7.4.3	VME Bus Error	102
7.5	System Controller Facilities	102
8	Parallel Communication Ports	103
8.1	Port Configuration	105
8.1.1	Front Panel Communication Port Connectors	106
8.1.2	Setting Up a Multi-Board System	108
8.2	Port to Port Communication	110
8.2.1	Program Controlled Transfers	110
8.2.2	DMA Controlled Transfers	110
8.2.3	Bidirectional Communication	111
9	PIM-3 Interface	113
9.1	Shared Bus Access	116
9.1.1	Access to the PIM-3 Interface	116
9.1.2	Access to Shared Bus Resources	117
9.2	Interrupts	119
10	Interrupts	121
10.1	Interrupt Implementation	121
10.2	Interrupt Configuration	124
11	JTAG Emulation System	127
11.1	JTAG Emulation System Connectors	128
11.2	Single Board Emulation System	129
11.3	Multi-Board Emulation System	129
11.4	JTAG Scan Path Routing	131
12	Reset Sources	133
12.1	/SYSRESET	133
12.2	Reset Switch	133
12.3	DBV46 System Reset Register	134

Figures	Page
Figure 1.1: DBV46 Block Diagram	5
Figure 2.1: Board Layout and Front Panel	15
Figure 3.1: On-Board C40 Local Memory Map	20
Figure 3.2: Boot Method Selection - B_BOOT and C_BOOT	27
Figure 3.3: DBV46 Boot Protection Feature - SW3	30
Figure 3.4: Software Data Protection Algorithm	34
Figure 3.5: Flow Chart For Programming the 32K PEROM	35
Figure 5.1: Shared Bus Memory Map	47
Figure 5.2: SLK1 and SLK2	55
Figure 5.3: Software Data Protection Algorithm	73
Figure 5.4: Flow Chart For Programming the Shared PEROM	74
Figure 5.5: Private Global Memory Resources Memory Map	76
Figure 6.1: DBV46 VMEbus Slave Memory Map	83
Figure 6.2: VME IACK Space	88
Figure 7.1: Default DBV46 VMEbus Master Memory Map	92
Figure 7.2: DSP and VME Master Interconnection	93
Figure 8.1: On and Off-Board Routing of Communication Ports	104
Figure 8.2: Front Panel Communication Port Connector Orientation	106
Figure 8.3: Multi-Board System	109
Figure 9.1: PIM-3 Interface Slave Memory Map	116
Figure 9.2: PIM-3 Master Memory Map	118
Figure 10.1: IIOF line Configuration	125
Figure 11.1: JTAG_IN and JTAG_OUT Connector Pinouts	128
Figure 11.2(i):Multi-Board Emulation System Using On-Board TBC	130
Figure 11.2(ii):Multi-Board Emulation System Using External Emulator	130
Figure 11.3: JTAG Scan Path Routing	132

Listings	Page
Listing 7.1: VME Master Read and Write Cycles	96

Tables	Page
Table 1.1: Shared Bus Master/Slave Access	10
Table 2.1: SW2 Switch Functions	13
Table 2.2: Default LED Functions	14
Table 3.1: On-Board C40 Global Memory Map	23
Table 3.2: Local Memory Interface Control Register	25
Table 3.3: Global Memory Interface Control Register	26
Table 4.1: TIM-40 Module Global Memory Map	40
Table 4.2: Global Memory Interface Control Register	42
Table 5.1: Shared Bus Master/Slave Access	46
Table 5.2: DBV46 Control Registers	48
Table 5.3: System Reset Register	49
Table 5.4: System Control Register	51
Table 5.5: LED Control Register	52
Table 5.6: LED Source Selection	52
Table 5.7: LED Function Description	53
Table 5.8: Communication Port Control Register	53
Table 5.9: System Status Register	54
Table 5.10: Node Identification Register	55
Table 5.11: Shared Bus Lock Register	56
Table 5.12: Timeout Error Register	57
Table 5.13: DBV46 Interrupt Registers	58
Table 5.14: Interrupt Enable Registers	59
Table 5.15: Interrupt Pending Registers	61
Table 5.16: Interrupt Line Status Registers	62
Table 5.17: Interrupt Condition Status Registers	65
Table 5.18: SCV64 Register Set	67
Table 5.19: SCV64 Default Register Values	68
Table 5.20: TBC Register Set	69
Table 6.1: SCV64 MODE Register - RXATOM and FIFOBEN	84
Table 6.2: SCV64 MODE and APBR Registers	89
Table 7.1: VME Master Address Modifiers	95
Table 7.2: SCV64 VINT and IVECT Registers	101
Table 8.1: Communication Port Control Register	105
Table 8.2: Front Panel Port Connector Pinout	106

Tables	Page
Table 9.1: PIM-3 Connector Pinout	114
Table 9.2: PIM P2 Connector Pinout	115
Table 9.3: PIM Global Connector Pinout	115
Table 9.4: PIM-3 Space Access Times	117
Table 10.1: DBV46 Interrupt Registers	122
Table 12.1: System Reset Register	134

1 Introduction

The DBV46 carrier board is part of Blue Wave Systems range of VME boards based around Texas Instruments' TMS320C4x family of Digital Signal Processors (DSPs). It provides a modular, systems level solution to a diverse range of real-time data acquisition and signal processing applications. DBV46's features include:

- Two TIM-40 module sites,
- One or two on-board TMS320C40 processors with two banks of zero wait state SRAM per processor,
- Twelve communication ports for multi-board expansion, providing maximum flexibility and reliable system optimisation,
- Large capacity shared DRAM accessible from both TIM-40 modules, both on-board C40 processors and from the VMEbus,
- VMEbus access to the global bus of both TIM-40 modules and both on-board C40 processors via a shared bus,
- Custom system expansion facilities via the PIM-3 module expansion interface,
- VMEbus master/slave interface, including full VMEbus arbiter and Slot 1 functions. DBV46 is compliant with the VME64 revision of the VMEbus Specification (Rev. C.1) and therefore provides for A64 and D64 accesses. The ability to interface to the VMEbus as a bus master allows DBV46 to control and interface with any compliant VME board. The Slot 1 functions mean that many VME systems will not require any other system controller.
- Software development for multi-board and multi-processor systems via the JTAG control scan path and shared bus data transfer mechanism,
- Extensive software support for development and target systems.

1.1 Conventions

This manual uses the following conventions:

- This Technical Reference Manual applies to both the single and dual on-board processor variants of the DBV46 board. If you have purchased the single processor variant of the board, the C40 C processor is NOT provided. All references to this processor and its associated resources can simply be ignored. Also note that the parallel communication ports to/from C40C are not utilised on this variant of the board.
- Hexadecimal numbers are followed by 'h'; for example, the address 1000 0000h.
- Names of internal C4x processor registers and register bits are given in italics to differentiate between them and those of DBV46; for example, the C4x processor *Global Memory Interface Control Register*.
- Throughout this manual certain register bits are stated as being 'Reserved, read as 0'. Reading these bits as '0' is recommended for software purposes since the actual state of these register bits can be either '0' or '1'.
- The VME memory map is byte wide and can be relocated by changing its base address, whereas the DSP memory map is 32 bit wide with fixed addresses. To avoid confusion between VME and DSP references, DSP addresses are given in italics; for example, the space *8900 0000h* to *8BFF FFFFh* in the DSP global memory map is reserved for shared DRAM.
- Active low signals are indicated by '/' preceding the signal name; for example, /CONFIG.
- Throughout this manual references are made to the 1993 edition of Texas Instruments' TMS320C4x User's Guide.



Text shown in this format highlights useful or important information



Text shown in this format is a warning to the user, it describes a situation that could potentially damage your equipment. Please read each warning carefully.

1.2 Using This Manual

This Technical Reference Manual (TRM) describes the features and functions of Blue Wave Systems' DBV46 board:

- **Chapter 1** gives a general overview of the board's features.
- **Chapter 2** explains how to install the board and describes the board's layout.
- Chapters **3** and **4** detail the facilities available via the on-board C40 processors and the TIM-40 module sites.
- **Chapter 5** details each of the resources available via the DBV46 shared bus.
- Chapters **6** and **7** describe DBV46's VME slave and master interfaces respectively.
- Chapters **8** to **12** expand upon particular features of DBV46.

Note that this TRM provides detailed reference information on DBV46. If you wish to get up and running quickly with your hardware and software, the C4x VME QuickStart Guide provided with your DBV46 board gives all the information you require to configure and install your carrier board.

Any software support packages that you have purchased with your DBV46 board are documented in separate User Guides.

A separate User Manual is provided for each type of TIM-40 module and PIM module purchased from Blue Wave.

Your distributor can provide you with up to date information on Blue Wave's range of related hardware and software products.

1.3 Overview

DBV46 is one of a family of Blue Wave carrier boards designed around Texas Instruments' TMS320C4x generation of floating-point parallel Digital Signal Processors (pDSP) to meet the needs of fast parallel processing and real-time embedded applications. Fully scalable systems architectures are available based on the intrinsic design features of the C4x. The board can support up to two TIM-40 modules containing TMS320C4x processors and also provides one or two on-board C40 processors. The modules and on-board C40 processors are referred to as 'processing nodes'.

DBV46 provides a shared bus accessible by the TIM-40 modules, on-board C40s, PIM module and VMEbus. DBV46 also provides shared global memory resources (on-board DRAM and PEROM) that can be accessed via the shared bus.

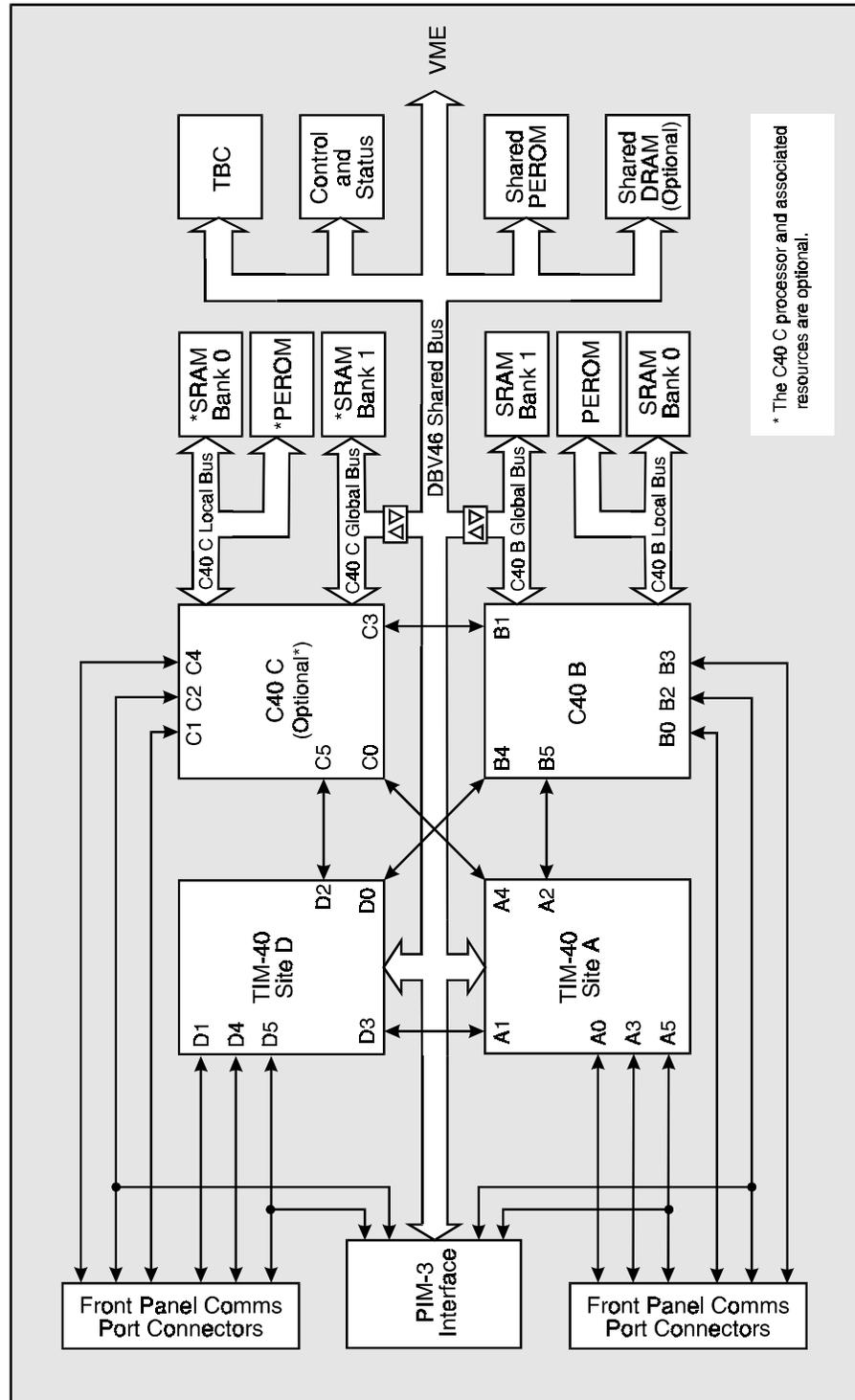
DBV46 is designed to closely match the requirements of both application and development users by providing a high degree of customisation. The Peripheral Interface Module (PIM-3) expansion interface provides for custom system expansion. Blue Wave's range of PIM modules includes support for the VME Subsystem Bus (VSB) via the VMEbus P2 connector.

Three parallel communication ports from each processing node are brought out to high density connectors on the DBV46 front panel. This provides an extremely flexible system as various combinations of carrier boards and modules can be connected together with little need for hardware development or prototyping. The remaining three ports from each processing node are interconnected between nodes to provide rapid processor to processor communication without the need to go off-board. One port from each node can be optionally routed to the PIM module site.

DBV46 provides a master/slave interface to the VMEbus including full VMEbus arbiter and Slot 1 functions, using the industry standard Newbridge Microsystems' SCV64 device. DBV46 is compliant with the VME64 revision of the VMEbus Specification (Rev. C.1) and therefore provides for A64 and D64 accesses.

A block diagram of the board's functionality is given in [Figure 1.1](#) and discussed briefly in the following subsections.

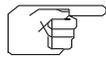
Figure 1.1: DBV46 Block Diagram



1.3.1 Processing Nodes B and C: On-Board C40 Processors

The TMS320C40 is the first of Texas Instruments' TMS320C4x generation of floating-point processors. It has two 32 bit address and data buses and six high speed parallel communication ports. Operating from a 50 MHz clock, a performance of 25 Million Instructions Per Second and 275 Million Operations Per Second can be achieved (30 MIPS and 330 MOPS at 60 MHz).

DBV46 provides one or two on-board 50 MHz or 60 MHz C40 processors, B and C, with two banks of zero wait state SRAM (Static RAM) per processor, Bank 0 and Bank 1. For each processor, Bank 0 is connected to the processor's local memory port and Bank 1 is connected to the global memory port. The global bus of each processor is also buffered onto the DBV46 shared bus, allowing both processors to access all DBV46 global resources.



If you have purchased the single processor variant of the board, the C40 C processor is NOT provided. All references to this processor and its associated resources can simply be ignored. Also note that the parallel communication ports to/from C40C are not utilised on this variant of the board.

Each on-board processor also has a PEROM to provide the Identification ROM (ID ROM) in accordance with Texas Instruments' TIM-40 Specification. This is required to hold data such as hardware dependent set-up values and it is accessed by the TMS320C40 processor during bootstrap. The PEROM can also hold bootcode and each processor can be configured independently to boot from the PEROM. Each 32K x 8 bit device can be programmed via its associated TMS320C40 processor to allow the bootcode and/or ID ROM data to be updated in situ.

The support software supplied with Blue Wave software support packages provides a comprehensive set of PEROM programming tools. These tools allow you to program each PEROM with ID ROM contents or bootcode. This software is detailed in the C4x VME Support Manual accompanying the support software. If you have not purchased a software support package, please contact your distributor for more information. If you do not have this software or wish to perform a PEROM programming function not covered by the tools, then the method for doing this is detailed in [Chapter 3](#) of this TRM.

1.3.2 Processing Nodes A and D: TIM-40 Modules

The DBV46 board has two single width TIM-40 sites, A and D, that meet Texas Instruments' TIM-40 specification. Blue Wave Systems supply a range of suitable single and double width modules that adhere to Texas Instruments' TMS320C4x Module Specification (TIM-40). This specification document can be obtained from Texas Instruments so that you can design a module to meet your specific application.

A TIM-40 module is a standard hardware platform typically comprising a Texas Instruments' TMS320C4x processor, some memory and peripherals. The C4x processors currently implemented on Blue Wave TIM-40 modules include the TMS320C40, as provided on DBV46, and the TMS320C44. While similar in core functionality to the TMS320C40 processor, the TMS320C44 processor provides four communication ports (as opposed to six) and two 24 bit external address buses (as opposed to 32 bit). This allows the C44 to be implemented in a smaller package, which in turn increases the space available for other on-module facilities such as memory.

The memory implemented on the module may be of any type, such as fast SRAM, DRAM, EDRAM or VRAM. The TIM-40 specification places few restrictions on the amount of memory or its distribution on either the local or global buses.

All TIM-40 modules have two standard connectors which carry the signals required by the module to function in a distributed memory environment. The TIM-40 specification also allows for an optional global bus connector. Both TIM-40 sites on DBV46 provide for this connector. A TIM-40 module with a global bus connector in either site has access to all DBV46 global resources.

For up to date details on Blue Wave's range of TIM-40 modules please consult your distributor.



Any TIM-40 module located on DBV46 must meet two requirements. Firstly, the C4x processor must be of the appropriate speed for the DBV46 board (50 MHz C4x on a 50 MHz DBV46, or 60 MHz C4x on a 60 MHz DBV46). Secondly, the module must be configured to take its clock from the carrier board, the module's User Manual will describe how to do this.

1.3.3 Processor Global Memory Organisation

The global bus from each processing node is buffered onto DBV46's shared bus. This means that all processing nodes have direct access to the global resources available via the shared bus:

- **Shared global memory resources**

Up to 16M x 32 (64M bytes) shared DRAM and 128K x 32 (512K bytes) shared PEROM are supplied on DBV46 as standard.

- **Private global memory resources**

The global SRAM of each on-board C40 processor is accessible via the shared bus. If you have Blue Wave MDC40S TIM-40 modules located on DBV46, the global SRAM available on these types of module is also accessible via the shared bus.

- **The VMEbus**

A64 D64 master accesses are supported to the VMEbus via the SCV64 device. Memory-mapped control registers are also provided allowing the features of SCV64 to be programmed if required.

- **Global peripheral facilities**

Blue Wave's PIM-3 interface. Note that C4x communication port access is also provided to this interface.

Note that the shared bus can also be accessed from the VMEbus and PIM-3 module interface. The shared bus architecture is described in [Chapter 5](#).

1.3.4 VMEbus Master/Slave Interface

DBV46 provides a full master/slave interface to the VMEbus that complies with the VME64 Specification. This means that DBV46 supports A64 and D64 master and slave accesses. The VMEbus interface is implemented using Newbridge Microsystems' SCV64 single chip VME interface.

The VMEbus slave interface provides access to the following:

- A control interface, which provides access to various features such as board reset.
- An on-board Test Bus Controller (TBC), which can be used to control the TMS320C4x JTAG-based scan path circuitry.
- Access to shared DRAM which provides a data transfer system to/from the processing nodes.
- Access to shared PEROM.
- Access to private global memory resources of each processing node.

DBV46's VMEbus slave interface is discussed in [Chapter 6](#).

All processing nodes have shared master access to the VMEbus. The ability to act as VMEbus master allows DBV46 to control and communicate with any compliant VME board, such as Blue Wave's DBV44 board which is a slave VME board with four TIM-40 sites. DBV46 can also act as the Slot 1 controller for your VME system. DBV46's VMEbus master interface is discussed in [Chapter 7](#).

1.3.5 Shared Bus Architecture

DBV46 provides a shared bus accessible by the TIM-40 modules, on-board C40s, PIM module and VMEbus. DBV46 also provides shared global memory resources (on-board DRAM and PEROM) as well as various control and status registers that can be accessed via the shared bus.

The master/slave capabilities of each of the above shared bus resources are shown in [Table 1.1](#) below.

Table 1.1: Shared Bus Master/Slave Access

Master → Slave ↓	TIM-40 A	C40 B	C40 C	TIM-40 D	PIM-3	VMEbus
TIM-40 A Global Memory	X	✓	✓	✓	✓	✓
C40 B Global Memory	✓	X	✓	✓	✓	✓
C40 C Global Memory	✓	✓	X	✓	✓	✓
TIM-40 D Global Memory	✓	✓	✓	X	✓	✓
PIM-3 Interface	✓	✓	✓	✓	X	X
VMEbus Interface	✓	✓	✓	✓	✓	X
Shared DRAM	✓	✓	✓	✓	✓	✓
Shared PEROM	✓	✓	✓	✓	✓	✓
DBV46 Register Set	✓	✓	✓	✓	✓	✓
SCV64 Register Set	✓	✓	✓	✓	✓	✓
TBC Registers (JTAG)	X	X	X	X	X	✓
VME IACK Space	✓	✓	✓	✓	✓	X
PIM Interrupt Space	✓	✓	✓	✓	✓	✓

✓ = Access X = No Access

1.3.6 Parallel Expansion - C4x Communication Ports

DBV46's parallel processor system can support optimum performance by distributing tasks between two or more processors. High performance multi-processing requires rapid transfer of data between processors. The DBV46 board achieves this by utilising all six of the high speed parallel communication ports available from each processing node.

Of the six ports available for processor-to-processor communication from each node, two ports are brought out to front panel connectors so that the user can interconnect ports on and off-board. Another port from each node can be routed to either the front panel or the PIM-3 interface. The three remaining ports from each node are interconnected to provide a dedicated communication link between nodes.

The on-board and off-board routing of the parallel communication ports is discussed in [Chapter 8](#).

1.3.7 PIM Expansion

DBV46 can also accommodate a Peripheral Interface Module (PIM) adjacent to TIM-40 Site A. DBV46 incorporates Blue Wave's standard PIM-3 interface which provides a general purpose expansion capability to DBV46. PIM-3 is an extension of Blue Wave's PIM interface and is compatible with all Blue Wave PIM modules.

A PIM module has up to three connectors:

- **PIM Global Connector**
This connector allows direct access to a PIM module from the shared bus.
- **PIM P2 Connector**
This connector allows direct access to/from the user-defined I/O pins of the VMEbus P2 connector.
- **PIM-3 Connector**
This connector allows direct access to the shared bus from a PIM module. This connector also allows bidirectional access to the four C4x parallel communication ports which can be optionally routed to the PIM site.

By implementing one or more of the above connectors, a PIM module can provide:

- Global memory expansion for a processing node.
- A C4x communication port link via P2.
- Access to/from a P2 I/O subsystem bus, such as the VME Subsystem Bus (VSB).
- Access to/from any custom interface, such as a data acquisition interface, either memory-mapped, via a communication port or via the VMEbus P2 connector.

The TIM-40 connectors for Site A and the adjacent PIM connectors are arranged so that DBV46 can support a double width PIM module that also incorporates all the facilities of a TIM-40 module.

Blue Wave's current range of PIM modules includes PIM-VSB2 which provides a memory-mapped interface from the processing nodes to the VME Subsystem Bus (VSB). The module provides a full master/slave VSB interface with on-board DPRAM (Dual-Port RAM) and DMA controller and can also act as the Slot 1 controller for your VSB system.

DBV46's PIM-3 interface is discussed in [Chapter 9](#). For information on Blue Wave's current range of PIM modules and the PIM-3 specification, please contact your distributor.

1.3.8 JTAG Emulation System

A JTAG emulation system can be set up to download and develop parallel multi-processor software. Single and multi-board systems can be controlled over the VMEbus interface using the on-board Test Bus Controller (TBC) or via an external controller, such as Blue Wave's XDSC40 or Texas Instruments' XDS510 system. See [Chapter 11](#) for more information.

2 Board Layout and Installation

2.1 Layout

The layout of the DBV46 board is given in [Figure 2.1](#). Before installing the board in your system you may wish to change the configuration of switch SW2. The functions and default settings are listed in [Table 2.1](#) below.

! **Appropriate anti-static precautions, such as wearing the wrist strap provided, must be taken at all times when handling the board.**

The front panel of the board has four LEDs which provide the user with information about the board's status, without having to access the board. The function of each LED is programmable via the LED Control Register, see [Chapter 5](#). [Table 2.2](#) below gives their default functions.

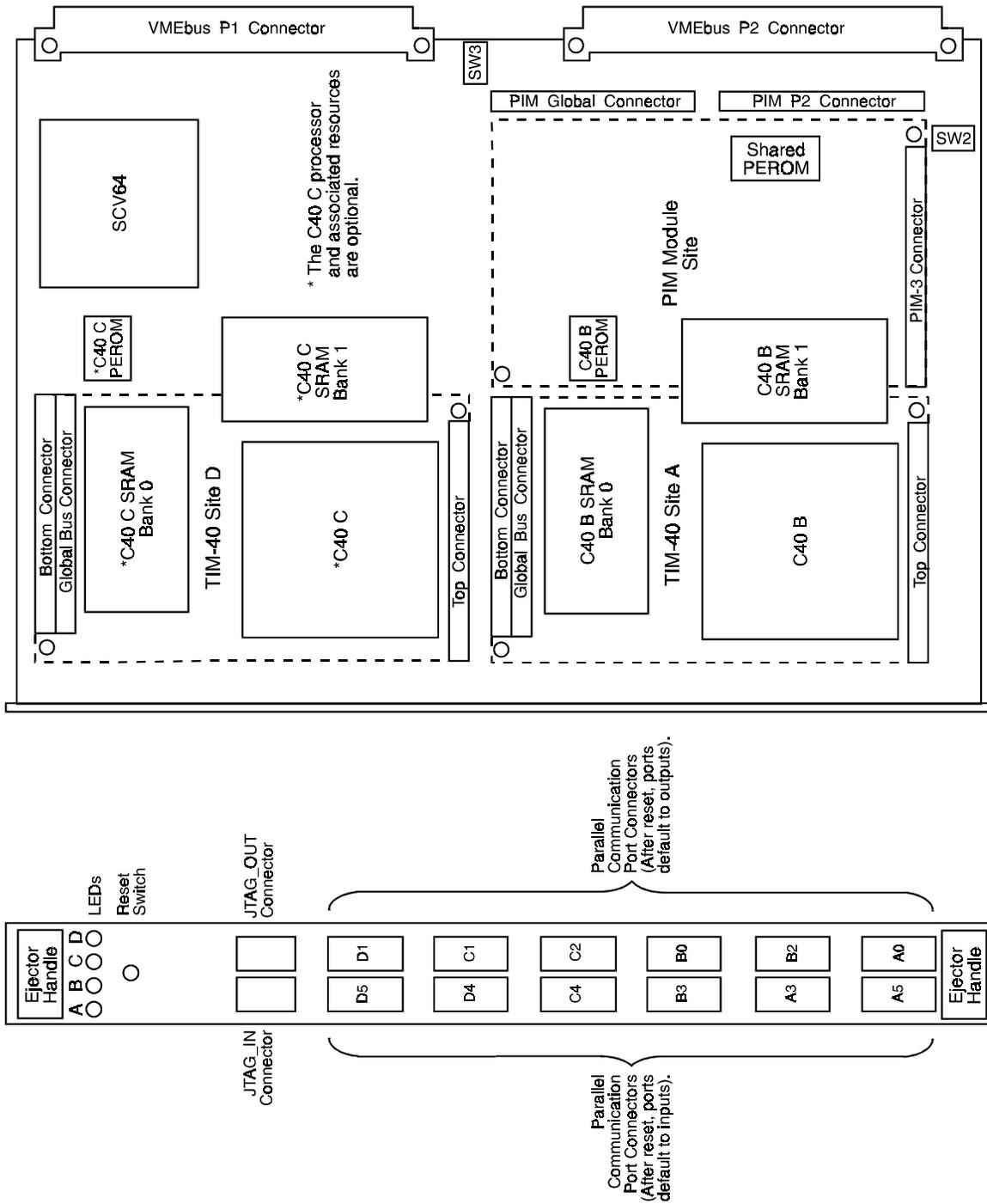
Table 2.1: SW2 Switch Functions

Switch	Function	Default	Reference
1	B_BOOT - C40 B Boot Method Select ON - Boot from PEROM OFF - Boot from communication port	ON	Section 3.6.1
2	C_BOOT - C40 C Boot Method Select ON - Boot from PEROM OFF - Boot from communication port	OFF	Section 3.6.1
3	* SLK1 ON - Set /SLK1 bit to '0'. OFF - Set /SLK1 bit to '1'.	ON	Section 5.2.5
4	* SLK2 ON - Set /SLK2 bit to '0'. OFF - Set /SLK2 bit to '1'.	ON	Section 5.2.5
* SLK1 and SLK2 set the user defined /SLK1 and /SLK2 bits of the DBV46 System Status Register.			

Table 2.2: Default LED Functions

LED	Default Function and Status
A	+5 V Power LED ON - DBV46 receiving VMEbus +5 V supply.
B	SYSFAIL LED ON - DBV46 has asserted /SYSFAIL.
C	RESET LED ON - C40 B on DBV46 is in reset. LED is cleared when C40 B is no longer in reset.
D	CONFIG LED ON - At least one module/processor in your system is pulling the global configuration line low. LED is cleared when all modules/processors in your system have released this line. Reflects the status of bit 3 of the System Status Register, see Section 5.2.5 .

Figure 2.1: Board Layout and Front Panel



2.2 Installation



All components of the DBV46 and your host system must be protected against static discharge while this installation procedure is carried out. Appropriate precautions, such as wearing the anti-static wrist strap supplied, must be taken at all times when handling the hardware.

DBV46 is a double Eurocard (6U) board which occupies a single slot in the VMEbus backplane. Your host VMEbus system must comply with the VMEbus Specification Rev C.1. Note that if A64 and/or D64 accesses are required, your host VMEbus system must also comply with the VME64 Specification.

The DBV46 board typically draws 3A from the +5 V VME supply rail. The additional current drawn from the +5 V and ± 12 V rails by any modules sited on the board must also be taken into account. This is detailed in the appropriate User Manuals.

Any TIM-40 modules that have been ordered with DBV46 will already be sited on the board. If you wish to move or add any modules, care must be taken to ensure that the correct module orientation is maintained. All TIM-40 modules should have a triangle printed on the corner of the PCB adjacent to the top primary connector. A module is installed by aligning its connectors to the appropriate site on DBV46 and simply push-fitting the module to the board.



Double-check the installation of each module. Failure to align and push home the connectors correctly may damage both the module and the DBV46 board.

There are two locating holes for each TIM-40 site and we strongly recommend that the locating bolts provided with DBV46 are used to secure each module to the board.



Any TIM-40 module located on DBV46 must meet two requirements. Firstly, the C4x processor must be of the appropriate speed for the DBV46 board (50 MHz C4x on a 50 MHz DBV46, or 60 MHz C4x on a 60 MHz DBV46). Secondly, the module must be configured to take its clock from the carrier board, the module's User Manual will describe how to do this.

Before installing DBV46 you must ensure that the IACK, BSREQ and BSGNT daisychains on the VMEbus backplane are complete.

To install DBV46:

- Ensure that power to the VMEbus system is turned off at all appropriate mains power outlets.
- Gently, but firmly, press-fit the DBV46's VMEbus 96-way male connectors into the VMEbus connectors on the backplane.
- Once the board is firmly in place secure the two retaining screws and power up your system.

If you need to remove the DBV46 board from your VMEbus system, ensure that the power is turned off, undo the retaining screws and use the ejector handles on the front panel.



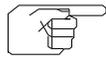
To avoid the danger of port contention and data corruption, do not connect any front panel C4x communication ports whilst your VME system is switched on. Note that DBV46 is shipped with protective covers on all front panel connectors. These should be left in place on unused connectors.

2.3 Booting

The C40 B on-board processor is configured to boot from PEROM by default. The PEROM will contain code that will automatically load default values into the SCV64 registers at boot-up. These values will correctly configure the VME memory map for your board. [Section 5.4](#) lists the complete SCV64 register set and gives the default settings of the registers affected by the C40 B bootcode. The PEROM bootcode also initialises the C40 B processor including configuration of the internal C40 *Local* and *Global Memory Interface Control Registers* with values suitable for DBV46 and C40 B. More detailed information on booting the on-board processors is given in [Section 3.6](#).

The C40 C processor and all TIM-40 modules ordered with DBV46 will be configured to boot from communication port and the PEROM will only contain ID ROM contents specific to DBV46 and the processor/module type. Blue Wave have designed several PC and VME carrier boards that can accommodate our range of TIM-40 modules. Therefore, if you order any modules separately, these will be configured to boot from communication port and the PEROM will only contain ID ROM contents for the module only.

Blue Wave supply a comprehensive set of tools to program the PEROM with bootcode or with module and carrier board ID ROM contents. These are supplied with all DBV46 development support packages. Please contact your distributor for further information.



If you wish to set up a multi-DBV46 system, you must reprogram the PEROMs on all but one board. The default VME memory map needs to be changed on these boards to prevent contention. This can be done using the PEROM programming tools.

3 Processing Nodes B and C: On-Board C40 Processors

3.1 Introduction

DBV46 provides one or two on-board 50 MHz or 60 MHz C40 processors, B and C, operating from the DBV46 on-board clock oscillator. These processors, and the resources available to each, essentially provide the same functionality as a standard TIM-40 module and can therefore be used in the same way.



If you have purchased the single processor variant of the board, the C40 C processor is NOT provided. All references to this processor and its associated resources can simply be ignored. Also note that the parallel communication ports to/from C40C are not utilised on this variant of the board.

Each processor has two banks of zero wait state SRAM (Static RAM), Bank 0 is connected to the processor's local memory port and Bank 1 is connected to the global memory port. Each processor also has a PEROM on the local memory port and the global bus of each processor is buffered onto the DBV46 shared bus, allowing both processors to access the DBV46 global resources.

The C40 external local and global memory ports can be considered separately as they do not interact in any way. The local port covers the address range 0 to $7FFF\ FFFFh$ and the global port covers the range $8000\ 0000h$ to $FFFF\ FFFFh$ inclusive. Sections [3.2](#) to [3.4](#) below cover the memory resources available to the on-board C40 processors.

3.2 Local Memory Interface

There are two internal C40 strobe signals that operate over the local memory interface, /LSTRB0 and /LSTRB1. These strobe signals control different areas of local memory external to the C40 and are configured via the internal C40 *Local Memory Interface Control Register*. The configuration of this register required for the on-board processors is described in [Section 3.5.1](#).

The local memory external to each C40 comprises one bank of SRAM and a PEROM. /LSTRB0 is used to access the local SRAM and /LSTRB1 is used to access the PEROM. Note that the C40 accesses the PEROM during bootstrap using /LSTRB0. The following subsections describe these memory areas. [Figure 3.1](#) below shows the C40 local memory map. Note that the memory map is the same for both on-board processors.

Figure 3.1: On-Board C40 Local Memory Map

C40 Longword Address	
0000 0000h	Internal Bootstrap
0000 1000h	Reserved
0010 0000h	Internal Peripherals
0010 0100h	Reserved
0020 0000h	Reserved
002F F800h	Internal RAM Block 0
002F FC00h	Internal RAM Block 1
0030 0000h	External SRAM Bank 0 128K x 32 or 512K x 32
	Reflections of External SRAM Bank 0
4000 0000h	PEROM 32K x 8
4000 8000h	Reflections of PEROM
7FFF FFFFh	

3.2.1 Local SRAM

The local memory external to each C40 includes one bank of zero wait state SRAM, Bank 0. There are two factory-fitted size options:

- DBV46S1 - Bank 0 = 128K x 32.
- DBV46S2 - Bank 0 = 512K x 32.

Bank 0 is located from address *0030 0000h* in the C40 local memory map, see [Figure 3.1](#).

The local RAM of each processor (Bank 0 plus the internal C40 RAM Blocks 0 and 1) makes up one contiguous block in the memory map. This avoids the inconvenience of working with separated blocks where, for example, a programmer would have to ensure that their program jumped from one block to another if the program was larger than any one block.

3.2.2 PEROM

A 32K x 8 PEROM (Programmable Erasable ROM) is present on the local port of each C40 between addresses *4000 0000h* and *7FFF FFFFh*. This 8 bit wide device is connected to the TMS320C40 local data lines LD0 to LD7. Note that data lines LD8 to LD31 are not used by the PEROM and are undefined for reads. It can be used as either a bootstrap or identification ROM and can also be used to store additional data as required. Booting is described in [Section 3.6](#).

Each PEROM can be programmed via its associated C40 processor to allow the bootcode and/or ID ROM data to be updated in situ. The procedure for programming a PEROM is explained in [Section 3.7](#).

3.3 Global Memory Interface

There are two internal C40 strobe signals that operate over the global memory interface, */STRB0* and */STRB1*. These strobe signals control different areas of global memory and are configured via the internal C40 *Global Memory Interface Control Register*. The configuration of this register required for the on-board processors is described in [Section 3.5.2](#).

The global memory of each C40 comprises one bank of SRAM per processor and the memory-mapped shared bus. */STRB0* is used to access the global SRAM and */STRB1* is used to access the shared bus resources. The following subsections describe these memory areas and [Table 3.1](#) below shows the C40 global memory map. Note that the memory map is the same for both on-board processors.

3.3.1 Global SRAM

The global memory of each C40 includes one bank of zero wait state SRAM, Bank 1. There are two factory-fitted size options:

- DBV46S1 - Bank 1 = 128K x 32.
- DBV46S2 - Bank 1 = 512K x 32.

Bank 1 is located from address *8000 0000h* in the C40 global memory map (see [Table 3.1](#)).

Bank 1 is accessed via the associated processor's private global bus. However, Bank 1 can also be accessed via the shared bus by other shared bus masters, such as the TIM-40 modules and VMEbus. DBV46 includes automatic arbitration logic to avoid allowing a C40 to access its global SRAM at the same time as another shared bus master. Shared bus access and arbitration are described in detail in [Chapter 5](#).

Note that the C40 B bootcode and Blue Wave's DBV46 software support packages makes use of the upper 2K x 32 of C40 B global SRAM. Therefore, if you wish to use a Blue Wave software support package, do not access this area of C40 B's global memory.

3.3.2 Shared Bus Resources

The global bus of each processor is buffered onto the DBV46 shared bus, allowing both processors to access the DBV46 global resources. Each of these resources is described in detail in [Chapter 5](#) along with a full memory map. Note that:

- A C40 cannot access its own global SRAM (Bank 1) via the shared bus.
- The speed of Bank 1 is not affected by the speed of the shared bus resources because the wait states are generated independently, see [Section 3.5](#). Similarly, the speed of local memory is independent of the speed of global memory.

Table 3.1: On-Board C40 Global Memory Map

C40 Longword Address	Address Space	Reference
8000 0000h	Bank 1 SRAM (128K or 512K x 32) plus Reflections	Section 3.3.1
8800 0000h	No Access	
8804 0000h	DBV46 Control Registers	Section 5.2
8808 0000h	SCV64 Register Set	Section 5.4
880C 0000h	Reserved	
8820 0000h	VME IACK Space	Section 6.3
8830 0000h	Reserved	
8840 0000h	DBV46 Interrupt Registers	Section 5.3
8850 0000h	Shared PEROM (128K x 8)	Section 5.6.2
8860 0000h	Reserved	
8870 0000h	PIM Interrupt Space	Section 9.2
8880 0000h	Reserved	
8888 0000h	TIM-40 A Global Memory	See Note Below
8890 0000h	C40 B Global Memory	
8898 0000h	Nodes A and B Global Memory	
88A0 0000h	C40 C Global Memory	
88A8 0000h	Nodes A and C Global Memory	
88B0 0000h	No Access	
88C0 0000h	TIM-40 D Global Memory	
88C8 0000h	Nodes A and D Global Memory	
88D0 0000h	Nodes B and D Global Memory	
88D8 0000h	Nodes A, B and D Global Memory	
88E0 0000h	Nodes C and D Global Memory	
88E8 0000h	Nodes A, C and D Global Memory	
88F0 0000h	No Access	
8900 0000h	Shared DRAM	Section 5.6.1
8C00 0000h	Reserved	
9000 0000h	PIM-3 Interface	Chapter 9
A000 0000h	Reserved	
C000 0000h FFFF FFFFh	VMEbus Address Space	Chapter 7

These memory areas allow access to the 'private' global memory of each processing node. The first address of each space relates directly to address 8000 0000h in the processing node's global memory map. Where more than one node is indicated, these spaces are write only and provide a broadcast write facility. For example, a write to address 88C8 0000h will write to address 8000 0000h of both TIM-40 modules (Sites A and D). These memory areas and the broadcast write facility are described in detail in [Chapter 5](#). Note that an attempt to access a global memory resource which is not available will result in a shared bus error.

3.4 Optimising Performance

Accesses to the SRAM of each on-board processor are performed at zero wait state. However, a number of points are provided below to help you optimise the performance of your memory resources further.

- Restrict data to local memory and executable code to global memory. This allows the C40 to use a 'dedicated' bus for data and code accesses.
- When transferring data to/from DBV46 global DRAM, use the C40's local memory to store the data. This allows the C40 to use a 'dedicated' bus for read and write accesses.
- Make effective use of the DMA controllers to isolate the C40 from memory accesses, see Chapter 9 of Texas Instruments' TMS320C4x User's Guide.
- Make effective use of the C40 instruction cache, see Section 3.5 of Texas Instruments' TMS320C4x User's Guide.
- Use the C40's internal memory (Block 0 and 1) wherever possible. Note that since this is only 2K x 32 some applications may find this prohibitively small. However, nearly all programs should be successful in finding some use for the two internal memory buses which can take frequently used location accesses away from the local and global buses.
- In general, take care when deciding how the various memory resources are used and allocated.

3.5 Memory Interface Control Registers

Both the local and global buses have to be configured to operate with the memory connected to them. These settings are contained in the internal C40 *Local and Global Memory Interface Control Registers*. These two registers are identical in layout and are described in detail in Chapter 7 of Texas Instruments' TMS320C4x User's Guide.

These registers must be set to reflect the number of software wait states to be generated, the active ranges of the two memory strobes (/LSTRB0 and 1 for local memory and /STRB0 and 1 for global memory) and the memory page sizes within the strobe range. Note that Blue Wave's bootcode contained in the PEROM of C40 B, will automatically configure the *Local and Global Memory Interface Control Registers* of C40 B with the correct values.

3.5.1 Local Memory Interface Control Register

The local memory interface of the C40 processor has two sets of control signals, /LSTRB0 and /LSTRB1. The internal C40 *Local Memory Interface Control Register* defines the page sizes for the two strobes, when the strobes are active and the mode of wait state generation. This 32 bit register is accessed at the internal C40 address *0010 0004h* and is fully documented in Chapter 7 of Texas Instruments' TMS320C4x User's Guide.

Table 3.2 below details the configuration of this register required for each on-board processor. The values given in the table below equate to a value of 3D74 2050h for DBV46S1 and 3D74 A050h for DBV46S2.

Table 3.2: Local Memory Interface Control Register

D31 - D30	D29	D28 - D24	D23 - D19	D18 - D14	D13 - D11	D10 - D8	D7 - D6	D5 - D4	D3 - D0
Reserved	<i>LSTRB SWITCH</i>	<i>LSTRB ACTIVE</i>	<i>LSTRB1 PAGESIZE</i>	<i>LSTRB0 PAGESIZE</i>	<i>LSTRB1 WTCNT</i>	<i>LSTRB0 WTCNT</i>	<i>LSTRB1 SWW</i>	<i>LSTRB0 SWW</i>	-

Bit	Name	Function and Configuration
3 - 0	-	Read only, write '0000'.
5 - 4	<i>LSTRB0 SWW</i>	Mode of wait state generation for /LSTRB0 accesses. Set to '01' for internal wait states.
7 - 6	<i>LSTRB1 SWW</i>	Mode of wait state generation for /LSTRB1 accesses. Set to '01' for internal wait states.
10 - 8	<i>LSTRB0 WTCNT</i>	Software wait state count for /LSTRB0 accesses - three bit range from zero to seven. Set to '000' for zero wait states.
13 - 11	<i>LSTRB1 WTCNT</i>	Software wait state count for /LSTRB1 accesses - three bit range from zero to seven. Set to '100' for four wait states.
18 - 14	<i>LSTRB0 PAGESIZE</i>	Page size of /LSTRB0 accesses. Set to '10000' for 128K page size. Set to '10010' for 512K page size.
23 - 19	<i>LSTRB1 PAGESIZE</i>	Page size of /LSTRB1 accesses. Set to '01110' for 32K page size.
28 - 24	<i>LSTRB ACTIVE</i>	Specifies the address ranges over which /LSTRB0 and /LSTRB1 are active. Set to '11101' for /LSTRB0 active from 0 to 3FFF FFFFh and /LSTRB1 active from 4000 0000h to 7FFF FFFFh.
29	<i>LSTRB SWITCH</i>	Must be set to '1', this will insert a single cycle between back-to-back reads that switch from /LSTRB0 to /LSTRB1 (and vice versa).
31 - 30	Reserved	Write as '00'.

3.5.2 Global Memory Interface Control Register

The global memory interface of the C40 processor has two sets of control signals, /STRB0 and /STRB1. The internal C40 *Global Memory Interface Control Register* defines the page sizes for the two strobes, when the strobes are active and the mode of wait state generation. This 32 bit register is accessed at the internal C40 address *0010 0000h* and is fully documented in Chapter 7 of Texas Instruments' TMS320C4x User's Guide.

Table 3.3 below details the configuration of this register required for each on-board processor. The values given in the table below equate to a value of 3A4C 0000h for DBV46S1 and 3A4C 8000h for DBV46S2.

Table 3.3: Global Memory Interface Control Register

D31 - D30	D29	D28 - D24	D23 - D19	D18 - D14	D13 - D11	D10 - D8	D7 - D6	D5 - D4	D3 - D0
Reserved	<i>STRB SWITCH</i>	<i>STRB ACTIVE</i>	<i>STRB1 PAGESIZE</i>	<i>STRB0 PAGESIZE</i>	<i>STRB1 WTCNT</i>	<i>STRB0 WTCNT</i>	<i>STRB1 SWW</i>	<i>STRB0 SWW</i>	-

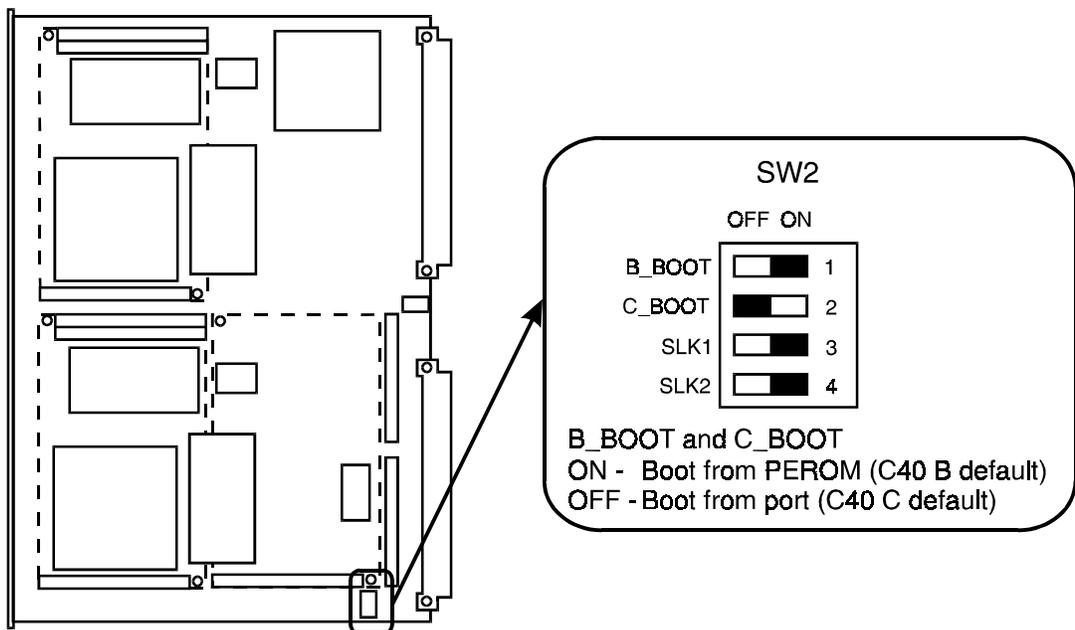
Bit	Name	Function and Configuration
3 - 0	-	Read only, write '0000'.
5 - 4	<i>STRB0 SWW</i>	Mode of wait state generation for /STRB0 accesses. Set to '00' for external wait states.
7 - 6	<i>STRB1 SWW</i>	Mode of wait state generation for /STRB1 accesses. Set to '00' for external wait states.
10 - 8	<i>STRB0 WTCNT</i>	Not used, set to '000'.
13 - 11	<i>STRB1 WTCNT</i>	Not used, set to '000'.
18 - 14	<i>STRB0 PAGESIZE</i>	Page size of /STRB0 accesses. Set to '10000' for 128K page size. Set to '10010' for 512K page size.
23 - 19	<i>STRB1 PAGESIZE</i>	Page size of /STRB1 accesses. Set to '01001' for 1K page size.
28 - 24	<i>STRB ACTIVE</i>	Specifies the address ranges over which /STRB0 and /STRB1 are active. Set to '11010' for /STRB0 active from <i>8000 0000h</i> to <i>87FF FFFFh</i> and /STRB1 active from <i>8800 0000h</i> to <i>FFFF FFFFh</i> .
29	<i>STRB SWITCH</i>	Set to '1' to insert a single cycle between back-to-back reads that switch from /STRB0 to /STRB1 (and vice versa).
31 - 30	Reserved	Write as '00'.

3.6 Booting

3.6.1 Boot Options

There are two methods to boot each C40 processor on DBV46. One is to use the processor's PEROM and the other to boot from one of the six parallel communication ports on the processor. Switches 1 and 2 (B_BOOT and C_BOOT) of SW2 must be configured to enable the method you wish to use for C40 B and C40 C respectively. Each processor is configured independently and [Figure 3.2](#) below gives the location and setting of these switches.

Figure 3.2: Boot Method Selection - B_BOOT and C_BOOT



B_BOOT is configured to enable C40 B to boot from PEROM and C_BOOT is configured to enable C40 C to boot from communication port, as default. Blue Wave's PEROM bootcode and the booting process is described in [Section 3.6.2](#) below.

Each PEROM is located from address *4000 0000h* which is one of the designated boot addresses in the C40's memory map (see Section 13.7 of Texas Instruments' TMS320C4x User's Guide). The 'module' is configured at reset to cause the boot loader of each C40 to operate from address *4000 0000h*. DBV46 pulls the IIOF0 and IIOF1 interrupt lines high after reset during bootstrap. This is necessary because the C40s use these lines during bootstrap. The ROMEN pin of the C40 processor is tied high and if B_BOOT/C_BOOT is ON, IIOF2 is pulled low during bootstrap which causes the C40 processor to boot from PEROM at address *4000 0000h*. If B_BOOT/C_BOOT is OFF, IIOF2 is pulled high during bootstrap, which causes the C40 processor to boot from one of the communication ports.

A specific boot load sequence must be followed and this is described in detail with examples in Chapter 13 of the Texas Instruments' TMS320C4x User's Guide (these are also detailed in the TIM-40 Specification).

The general mechanism for booting from a byte wide ROM is detailed in Table 13-4 of Texas Instruments' TMS320C4x User's Guide. This consists of a data stream, which contains headers that define the width of the memory and the *Global* and *Local Memory Interface Control Register* settings. Following this is the size of the first program block to be loaded and the address where the program block is to be loaded to, next comes the user program. This input data sequence is the same for the communication port boot loader except that there is no need to define the source memory width (because this is fixed for the communication ports).

It is possible to load a number of programs in this way. The complete load sequence must be terminated by a full 32 bit word of zeros. Two entries in the stream then define locations for the interrupt and trap vector tables. The circuitry external to the C40 processor (i.e. the module and carrier board) uses the /IACK signal from the processor to detect that booting is complete. Thus the final entry in the stream is an external memory location for an IACK instruction to generate an active low pulse on the /IACK signal. The processor then begins execution at the first word of the first block that was downloaded.

When booting from one of the communication ports, the C40 processor automatically scans all six ports until the correct protocol is observed, this is transparent to the user. The C40 then boots from this port. Note that at least one processing node on DBV46 must initialise the SCV64 register set to configure the VMEbus interface.

Further details may be found in Section 13.7 of Texas Instruments' TMS320C4x User's Guide.

3.6.2 Blue Wave System's PEROM Bootcode

The C40 B on-board processor is configured to boot from PEROM as default. The basic operations performed by the C40 B PEROM bootcode are detailed below:

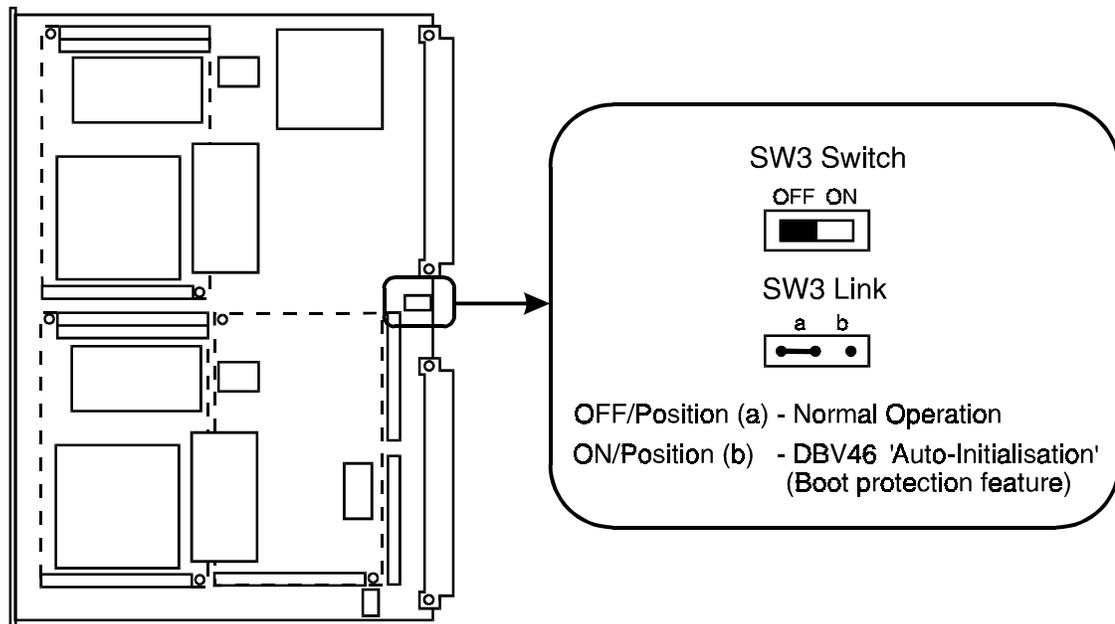
- The C40 B PEROM bootcode first initialises the processor, including configuration of the *Local* and *Global Memory Interface Control Registers* with values suitable for DBV46 and the on-board processors. The /CONFIG line for the processor is then released, see [Section 5.2.5](#).
- Default values are then loaded into the SCV64 registers. These values will correctly configure the VME memory map for your board. [Section 5.4](#) lists the complete SCV64 register set and gives the default settings of the registers affected by the C40 B bootcode.
- The bootcode then checks for the presence of any bootcode in the shared PEROM. If any bootcode is located in this PEROM, C40 B will boot from this code.
- If no bootcode is found in the shared PEROM, the C40 B bootcode will emulate communication port booting. That is, the C40 processor will automatically scan all six ports until the correct protocol is observed. The C40 then boots from this port. Note that during the communication port scan, the front panel LEDs are flashed in a rippling pattern and intermittently display the system status information as described in [Table 2.2](#).
- As well as scanning the communication ports, the bootcode also checks the status of the SCV64 location monitor (described in [Section 6.2](#)). This is used by Blue Wave's DBV46 software support packages, to perform certain initialisation routines. Therefore, in general, if you wish to use a Blue Wave software support package do not use the SCV64 location monitor. If you wish to make use of the SCV64 location monitor from your host or DSP code, your software support package User Guide will describe how to do this.
- Note that the C40 B bootcode and Blue Wave's DBV46 software support packages also make use of interrupts via the C40 B IIOF0 and IIOF2 lines. Therefore, in general, if you wish to use an Blue Wave software support package do not alter any register settings associated with C40 B IIOF0 and IIOF2 (such as the Interrupt Vector Table Pointer, IIF Register etc.). If you wish to make use of these interrupt lines from your DSP or host code, your software support package User Guide will describe how to do this.

3.6.3 Bootcode Corruption

As described above, the C40 B bootcode performs the SCV64 initialisation required to correctly configure the VME memory map for your board. However, if this bootcode is corrupted in any way, for example by an incorrect PEROM programming operation, DBV46 will not appear in the VME memory map. Therefore, DBV46 incorporates a protection feature to avoid this situation.

The DBV46 'boot protection feature' is implemented in hardware using the SW3 device. In normal operation this device can simply be ignored. However, if C40 B bootcode corruption occurs, SW3 can be configured to initialise the DBV46 VME slave image. In this situation, a VME base address of 1000 0000h and image size of 128M bytes (A32 space) is used at power-up. The location and setting of SW3 is shown in [Figure 3.3](#) below. The steps you should perform for DBV46 recovery are also listed below.

Figure 3.3: DBV46 Boot Protection Feature - SW3



Step 1 **Switch off your VME system**

If C40 B bootcode corruption occurs, you should first switch off power to the DBV46 board. This enables you to reconfigure the switches on DBV46 as described in the steps below before attempting to reprogram the PEROM bootcode.

Step 2 **Reconfigure SW3**

Configure SW3 to initialise the DBV46 VME slave image at power-up with a VME base address of 1000 0000h and image size of 128M bytes (A32 space), see [Figure 3.3](#) above.

Step 3 **Reconfigure SW2**

Set C40 B to boot from communication port via SW2 (B_BOOT - OFF), see [Section 3.6.1](#). This will prevent the corrupted bootcode from performing erroneous initialisation routines.

Step 4 **Reprogram the bootcode**

Once you have configured SW2 and SW3 as described above, you can power up your system. You will now have access to the DBV46 TBC register set which will enable you to reprogram the C40 B PEROM bootcode. This can be performed in one of two ways:

- Blue Wave supply a comprehensive set of tools to program the PEROM with bootcode. These are supplied with all DBV46 development support packages and are described in the C4x VME Support Manual.
- If you have not purchased a DBV46 development support package, you must use the debugger (DB40 or TDB) to download your own DSP code to reprogram the PEROM. The method for programming this device is described in [Section 3.7](#) below.

3.7 Programming the 32K PEROM Device from the C40

The support software supplied with Blue Wave carrier board software development packages contains a PEROM programming tool for use with DBV46. This tool enables you to program a PEROM with ID ROM data or bootcode. Details of this software are given in the C4x VME Support Manual accompanying the support software. If you have not purchased a software support package, please contact your distributor for more information. If you do not have this software or wish to perform a PEROM programming function not covered by the tools, then the method for doing this is detailed in the sections below. Note that the C40 B PEROM supplied on DBV46 is already programmed with Blue Wave's bootcode and valid ID ROM data and C40 C is already programmed with valid ID ROM data.

Each PEROM is an ATMEL AT29C256 device and resides on the local memory bus of its corresponding C40 between addresses *4000 0000h* and *7FFF FFFFh*. This 8 bit wide device is connected to the C40 local data lines LD0 to LD7, with LD8 to LD31 undefined. The device is 32K bytes in size, so there are a number of images of the same device in this space.

To protect the PEROM from accidental programming, there are two protection features that must be overcome before programming the PEROM is possible:

- The decode logic of DBV46 is designed to prevent writes to the PEROM from */LSTRB1*, the recommended memory strobe for reading the PEROM. Consequently, it is necessary to reconfigure the *Local Memory Interface Control Register* to allow accesses using */LSTRB0*. This is done by changing the *STRB ACTIVE* field to extend the range of */LSTRB0* to cover all addresses up to *7FFF FFFFh* and by increasing the number of wait states to five. Note that whereas only four wait states are required to read the PEROM, five are required to write to it. Consequently the value required for the *Local Memory Interface Control Register* is *3E73 A550h* for DBV46S1 and DBV46S2.
- The second protection feature is a software data protection algorithm which is present on the PEROM. The PEROM is supplied with this algorithm enabled, therefore you must first disable the algorithm if you wish to write to the PEROM.

The data protection algorithm and the method for programming the PEROM is explained in [Section 3.7.1](#) below.



Note that each PEROM on DBV46 is programmed separately and can only be accessed by its corresponding C40.

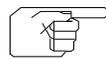
3.7.1 Software Data Protection Algorithm

The software data protection algorithm, which is enabled on the supplied PEROM as default, consists of a series of three program commands to specific addresses with specific data (see [Figure 3.4\(i\)](#)). After the software data protection is enabled the same three program commands must begin each program cycle in order for programming to occur. Once set, the software data protection feature remains active unless its disable command is issued. Power transitions will not reset the software data protection feature, however the software feature will guard against inadvertent program cycles during power transitions. In order to disable this algorithm so that you can write to the PEROM the programming steps detailed in [Figure 3.4\(ii\)](#) must be performed.

The device is programmed on a page basis, each page being 64 bytes in length. If a single byte is to be changed, the whole of that page has to be written. Any location that is not loaded during the programming of the page is erased to read FFh. Writes in each page must be performed within 150 μ s of each other. Note that back-to-back writes to the PEROM must be prevented. To do this it is recommended that you perform a dummy read, after each PEROM write, from the address that has just been written to.

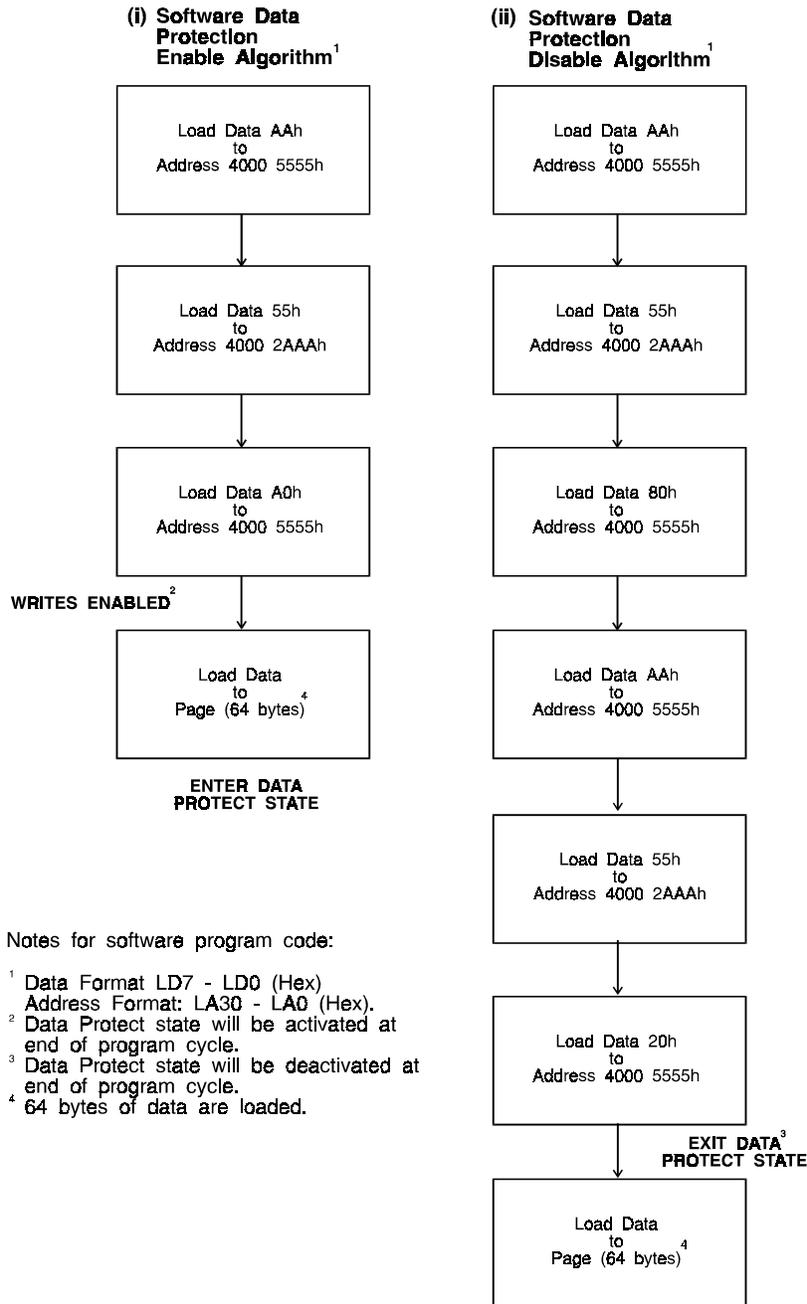
Once the page has been written to the device there is a period where those values are programmed into the storage matrix. Data polling allows the C40 to attempt to read the last value written to the data page and will return the complement of the data loaded on bit 7 of the byte. This byte may be polled and when the programming cycle has completed bit 7 will reflect the true value of the data written to it. This should take no longer than 10 ms.

[Figure 3.5](#) gives a general flow chart which shows the steps necessary to allow you to write to the PEROM.



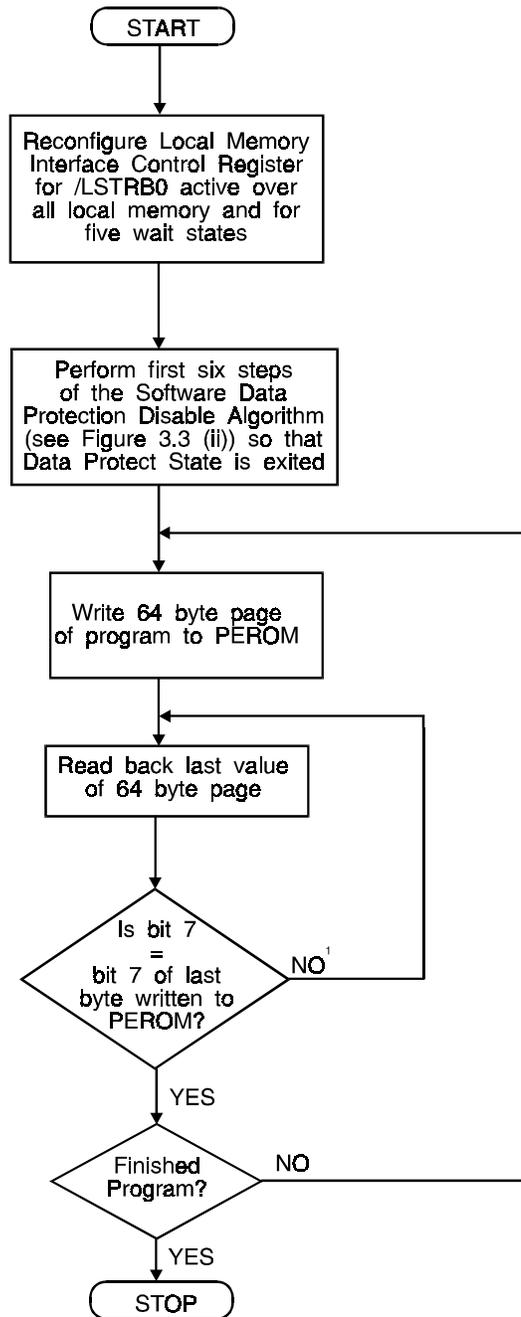
Due to the properties of the PEROM device, data integrity can only be assured up to 1000 programming cycles.

Figure 3.4: Software Data Protection Algorithm



To prevent back-to-back writes to the PEROM, perform a dummy read after each write from the same address that has just been written to.

Figure 3.5: Flow Chart For Programming the 32K PEROM



¹This loop should take no longer than 10 ms.

3.8 Interrupts

Three of the four IIOF lines (IIOF0, 1 and 2) provided by each on-board C40 can be configured to receive or generate interrupts on DBV46. IIOF3 is used to control the /CONFIG line.

The source and direction of interrupts for both processors are configured via the DBV46 Interrupt Registers. These registers are described in detail in [Section 5.3](#) and allow you to enable interrupts to/from a number of different sources on various interrupt conditions. The IIOF lines are controlled via the C40 processor *IIOF Flag Register*, a summary of this register is given in Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide. [Chapter 10](#) of this TRM details the configuration of interrupts to/from the processing nodes and provides example step by step procedures.

! **If you require any of the IIOF lines to be configured as outputs you must ensure that they are not also driven by another source, as this may cause damage to the C40 processor.**

Note that if you wish to use the IIOF lines of an individual C40, an IACK instruction must be performed after that processor has been reset. The C40 boot loader in the on-chip ROM performs this IACK instruction automatically at the end of the booting procedure. However, if you configure a C40 to boot from communication port, the boot loader will NOT perform this operation if valid bootcode is not received (such as when using a debugger). In this case, you must execute the IACK instruction to a valid external memory address in the C40 memory map yourself.

4 Processing Nodes A and D: TIM-40 Modules

4.1 Introduction

DBV46 has two single width TIM-40 sites, A and D. Both sites have a global bus connector as well as the standard top and bottom connectors. The global bus of each TIM-40 site is buffered onto the DBV46 shared bus, allowing both TIM-40 sites to access the DBV46 global resources.

Table 4.1 below gives the global memory maps for the C4x processor(s) on Site A and Site D. Each TIM-40 module global memory map is a combination of the DBV46 global resources and any global memory sited on the module itself. The size, type and organisation of on-module global memory is specific to the TIM-40 module type and is detailed separately in the TIM-40 Module User Manual that accompanies each module. Therefore, the DBV46 global memory map must be merged with that of the TIM-40 module to obtain the full picture.

Section 4.2 below covers the DBV46 global resources available to TIM-40 Sites A and D.



Any TIM-40 module located on DBV46 must meet two requirements. Firstly, the C4x processor must be of the appropriate speed for the DBV46 board (50 MHz C4x on a 50 MHz DBV46, or 60 MHz C4x on a 60 MHz DBV46). Secondly, the module must be configured to take its clock from the carrier board, the module's User Manual will describe how to do this.

Note that it is possible to locate a TIM-40 module which incorporates a 60 MHz C4x processor, on a 50 MHz DBV46 board. This option is possible since the module/processor must be configured to use the 50 MHz DBV46 clock oscillator as indicated above. However, it should be noted that the C4x will therefore operate at 50 MHz performance.

4.2 Global Memory Interface

There are two internal C4x strobe signals that operate over the global memory interface, */STRB0* and */STRB1*. These strobe signals control different areas of global memory and are configured via the internal C40 *Global Memory Interface Control Register*. The configuration of this register required for the on-board processors is described in [Section 4.2.3](#).

The global memory of each TIM-40 module is a combination of the global memory sited on the module itself and the DBV46 global resources available via the memory-mapped shared bus. */STRB0* is used to access any on-module global SRAM and */STRB1* is used to access the shared bus resources. The following subsections describe these memory areas.

4.2.1 Global SRAM

The size, type and organisation of on-module global memory is specific to the TIM-40 module type and is detailed separately in the TIM-40 Module User Manual that accompanies each module.

On-module global memory is accessed via the associated processor's private global bus. However, if you have Blue Wave MDC40S TIM-40 modules located on DBV46, the global SRAM available on these types of module is also accessible via the shared bus by other shared bus masters, such as the DBV46 on-board processors and the VMEbus. DBV46 includes automatic arbitration logic to avoid allowing a module's C4x processor to access its on-module global memory at the same time as another shared bus master. Shared bus access and arbitration are described in detail in [Chapter 5](#).

Note that in order to access the on-module global memory of an MDC40S TIM-40 module via the shared bus, you must set the *STRB0 SWW* bits of the C4x *Global Memory Interface Control Register* to '00' for external wait states, see [Section 4.2.3](#) below.

4.2.2 Shared Bus Resources

The global bus of each TIM-40 site is buffered onto the DBV46 shared bus, allowing both sites to access the DBV46 global resources. Each of these resources is described in detail in [Chapter 5](#) along with a full memory map. Note that:

- A TIM-40 site cannot access its own global memory via the shared bus.
- In order to access any of the DBV46 global resources via the shared bus, the TIM-40 module must have a global bus connector.
- The speed of on-module global memory is not affected by the speed of the shared bus resources because the wait states are generated independently, see [Section 4.2.3](#). Similarly, the speed of local memory is independent of the speed of global memory.

Table 4.1: TIM-40 Module Global Memory Map

C4x Longword Address	Address Space	Reference
8000 0000h	Reserved for On-Module Global Memory	TIM-40 Module User Manual
8800 0000h	No Access	
8804 0000h	DBV46 Control Registers	Section 5.2.
8808 0000h	SCV64 Register Set	Section 5.4
880C 0000h	Reserved	
8820 0000h	VME IACK Space	Section 6.3
8830 0000h	Reserved	
8840 0000h	DBV46 Interrupt Registers	Section 5.3
8850 0000h	Shared PEROM (128K x 8)	Section 5.6.2
8860 0000h	Reserved	
8870 0000h	PIM Interrupt Space	Section 9.2
8880 0000h	Reserved	
8888 0000h	TIM-40 A Global Memory	See Note Below
8890 0000h	C40 B Global Memory	
8898 0000h	Nodes A and B Global Memory	
88A0 0000h	C40 C Global Memory	
88A8 0000h	Nodes A and C Global Memory	
88B0 0000h	Nodes B and C Global Memory	
88B8 0000h	Nodes A, B and C Global Memory	
88C0 0000h	TIM-40 D Global Memory	
88C8 0000h	No Access	
88D0 0000h	Nodes B and D Global Memory	
88D8 0000h	No Access	
88E0 0000h	Nodes C and D Global Memory	
88E8 0000h	No Access	
88F0 0000h	Nodes B, C and D Global Memory	
88F8 0000h	No Access	
8900 0000h	Shared DRAM	Section 5.6.1
8C00 0000h	Reserved	
9000 0000h	PIM-3 Interface	Chapter 9
A000 0000h	Reserved	
C000 0000h FFFF FFFFh	VMEbus Address Space	Chapter 7

These memory areas allow access to the private global memory of each processing node. The first address of each space relates directly to address 8000 0000h in the processing node's global memory map. Where more than one node is indicated, these spaces are write only and provide a broadcast write facility. For example, a write to address 88B0 0000h will write to address 8000 0000h of both on-board processors (C40 B and C40 C). These memory areas and the broadcast write facility are described in detail in [Chapter 5](#). Note that an attempt to access a global memory resource which is not available will result in a shared bus error.

4.2.3 Global Memory Interface Control Register

The user must configure the internal C4x *Local* and *Global Memory Interface Control Registers* for each TIM-40 module on DBV46. The configuration of the global register for modules with a global bus connector is discussed below. The configuration of this register for modules with no global bus connector and the configuration of the local register is specific to the TIM-40 module, this is discussed in the TIM-40 module's User Manual.

The *Global Memory Interface Control Register* defines the page sizes for /STRB0 and /STRB1, when the strobes are active and the mode of wait state generation. This 32 bit register is accessed at the internal C4x address *0010 0000h* and is fully documented in Chapter 7 of Texas Instruments' TMS320C4x User's Guide.

Table 4.2 below details the configuration of this register required for a module located on DBV46. The values given in the table below equate to a value of 3A4C 0000h. Note that this value assumes certain TIM-40 module specific settings.

Table 4.2: Global Memory Interface Control Register

D31 - D30	D29	D28 - D24	D23 - D19	D18 - D14	D13 - D11	D10 - D8	D7 - D6	D5 - D4	D3 - D0
Reserved	<i>STRB SWITCH</i>	<i>STRB ACTIVE</i>	<i>STRB1 PAGESIZE</i>	<i>STRB0 PAGESIZE</i>	<i>STRB1 WTCNT</i>	<i>STRB0 WTCNT</i>	<i>STRB1 SWW</i>	<i>STRB0 SWW</i>	-

Bit	Name	Function and Configuration
3 - 0	-	Read only, write '0000'.
5 - 4	<i>STRB0 SWW</i>	Mode of wait state generation for /STRB0 accesses. Configuration depends upon TIM-40 module. Note that if access is required to the on-module global SRAM of an MDC40S TIM-40 module via the shared bus, <i>STRB0 SWW</i> must be set to '00' for external wait states.
7 - 6	<i>STRB1 SWW</i>	Mode of wait state generation for /STRB1 accesses. Set to '00' for external wait states.
10 - 8	<i>STRB0 WTCNT</i>	Software wait state count for /STRB0 accesses - three bit range from zero to seven. Configuration depends upon TIM-40 module. For example, set to '000' for zero wait states.
13 - 11	<i>STRB1 WTCNT</i>	Not used, set to '000'.
18 - 14	<i>STRB0 PAGESIZE</i>	Page size of /STRB0 accesses. Configuration depends upon TIM-40 module. For example, set to '10000' for 128K page size.
23 - 19	<i>STRB1 PAGESIZE</i>	Page size of /STRB1 accesses. Set to '01001' for 1K page size.
28 - 24	<i>STRB ACTIVE</i>	Specifies the address ranges over which /STRB0 and /STRB1 are active. Set to '11010' for /STRB0 active from 8000 0000h to 87FF FFFFh and /STRB1 active from 8800 0000h to FFFF FFFFh.
29	<i>STRB SWITCH</i>	Set to '1' to insert a single cycle between back-to-back reads that switch from /STRB0 to /STRB1 (and vice versa).
31 - 30	Reserved	Write as '00'.

For Blue Wave TIM-40 modules, refer to your module User Manual for the recommended settings of *STRB0 SWW*, *WTCNT* and *PAGESIZE* bits.

4.3 Optimising Performance

A number of points are provided below to help you optimise the performance of your TIM-40 module when using the module with DBV46:

- Restrict data to local memory and executable code to global memory. This allows the C4x to use a 'dedicated' bus for data and code accesses.
- When transferring data to/from DBV46 shared DRAM, use the C4x processor's local memory to store the data. This allows the C4x to use a 'dedicated' bus for read and write accesses.
- Make effective use of the DMA controllers to isolate the C4x from memory accesses, see Chapter 9 of Texas Instruments' TMS320C4x User's Guide.
- Make effective use of the C4x instruction cache, see Section 3.5 of Texas Instruments' TMS320C4x User's Guide.
- Use the C4x processor's internal memory (Block 0 and 1) wherever possible. Note that since this is only 2K x 32 some applications may find this prohibitively small. However, nearly all programs should be successful in finding some use for the two internal memory buses which can take frequently used location accesses away from the local and global buses.
- In general, take care when deciding how the various memory resources are used and allocated.

4.4 Booting

Any TIM-40 module ordered with DBV46 will be configured to boot from communication port. This configuration is required when using Blue Wave's interface libraries supplied with DBV46 software support packages.

If you are not using Blue Wave's software support packages, each module can be configured to boot from its associated PEROM as required. Your TIM-40 module User Manual will describe how to configure the boot method for your module. Note that if a module is configured to boot from PEROM, the PEROM must contain valid bootcode. It is up to you to provide this code. Useful information can be found in Section 13.7 of Texas Instruments' TMS320C4x User's Guide.

4.5 Interrupts

Three of the four IIOF lines (IIOF0, 1 and 2) provided by each module site can be configured to receive or generate interrupts on DBV46. IIOF3 is used to control the /CONFIG line.

The source and direction of interrupts for both sites are configured via the DBV46 Interrupt Registers. These registers are described in detail in [Section 5.3](#) and allow you to enable interrupts to/from a number of different sources on various interrupt conditions. The IIOF lines are controlled via the C40 processor *IIOF Flag Register*, a summary of this register is given in Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide. [Chapter 10](#) of this TRM details the configuration of interrupts to/from the processing nodes and provides example step by step procedures.



If you require any of the IIOF lines to be configured as outputs you must ensure that they are not also driven by another source, as this may cause damage to the C4x processor.

Note that if you wish to use the IIOF lines of an individual C40, an IACK instruction must be performed after that processor has been reset. The C4x boot loader in the on-chip ROM performs this IACK instruction automatically at the end of the booting procedure. However, when the processor is configured to boot from communication port and Blue Wave's interface library is not being used or the JTAG subsystem is implemented (such as when using a debugger) the boot loader will NOT perform this operation. In this case, you must execute the IACK instruction to a valid external memory address in the C4x memory map yourself.

5 Shared Bus Architecture

5.1 Introduction

DBV46 provides a shared bus allowing access from/to a number of memory-mapped facilities. These facilities are listed and described below.

- **On-Board C40 Processors**

Each of the on-board C40 processors (C40 B and C40 C) has access to the DBV46 shared bus facilities. The shared bus is mapped to the global memory map of each processor from address *8800 0000h*, see [Chapter 3](#). The global SRAM of each processor, Bank 1, can also be accessed via the shared bus. The only restriction is that an on-board C40 processor cannot access its own global SRAM via the shared bus.

- **TIM-40 Modules**

Each of the TIM-40 sites (Site A and Site D) has access to the DBV46 shared bus facilities. The shared bus is mapped to the global memory map of each module from address *8800 0000h*, see [Chapter 4](#). The global SRAM of a Blue Wave MDC40S module can also be accessed via the shared bus, see [Section 4.2.1](#). The only restriction is that an MDC40S module cannot access its own global memory via the shared bus.

- **PIM Module**

The PIM module site has access to the DBV46 shared bus facilities. The shared bus is mapped from address *0800 0000h* in the PIM-3 master memory map. The PIM module itself can also be accessed via the shared bus. A separate PIM Interrupt Space is also mapped to the shared bus for interrupt generation to the PIM module site. The PIM-3 interface is described in detail in [Chapter 9](#).

- **VMEbus**

DBV46 provides a master/slave interface to the VMEbus. The slave interface provides access to the DBV46 shared bus resources. The VMEbus master memory map is mapped to the shared bus allowing the on-board C40 processors and the TIM-40 modules to make VMEbus master accesses. The VMEbus slave interface is described in [Chapter 6](#) and the master interface is described in [Chapter 7](#). Note that a VME IACK space is also provided, used to perform a VMEbus interrupt acknowledge, see [Chapters 6 and 7](#).

- **On-Board Shared DRAM (Optional)**

Shared DRAM can be supplied on DBV46, accessible via the shared bus. This DRAM can be used to implement a shared memory architecture between the on-board processors, TIM-40 modules and VMEbus. DBV46 shared memory is described in [Section 5.6](#).

- **On-Board Shared PEROM**

A 128K x 8 shared PEROM is supplied on DBV46, accessible via the shared bus. This PEROM can be used to store common bootcode or embedded application code storage. DBV46 shared memory is described in [Section 5.6](#).

- **Control and Status Registers**

A number of memory-mapped control and status registers are provided on the shared bus. These comprise:

- DBV46 Control Registers, see [Section 5.2](#).
- DBV46 Interrupt Registers, see [Section 5.3](#).
- SCV64 Register Set, see [Section 5.4](#).
- TBC Register Set, see [Section 5.5](#).

The shared bus memory map and the master/slave capabilities of each of the shared bus resources are summarised in [Figure 5.1](#) and [Table 5.1](#) respectively. More detailed/specific shared bus memory maps are provided for each shared bus master and slave in the appropriate sections and chapters of this TRM.

Table 5.1: Shared Bus Master/Slave Access

Master → Slave ↓	TIM-40 A	C40 B	C40 C	TIM-40 D	PIM-3	VMEbus
TIM-40 A Global Memory	X	✓	✓	✓	✓	✓
C40 B Global Memory	✓	X	✓	✓	✓	✓
C40 C Global Memory	✓	✓	X	✓	✓	✓
TIM-40 D Global Memory	✓	✓	✓	X	✓	✓
PIM-3 Interface	✓	✓	✓	✓	X	X
VMEbus Interface	✓	✓	✓	✓	✓	X
Shared DRAM	✓	✓	✓	✓	✓	✓
Shared PEROM	✓	✓	✓	✓	✓	✓
DBV46 Register Set	✓	✓	✓	✓	✓	✓
SCV64 Register Set	✓	✓	✓	✓	✓	✓
TBC Registers (JTAG)	X	X	X	X	X	✓
VME IACK Space	✓	✓	✓	✓	✓	X
PIM Interrupt Space	✓	✓	✓	✓	✓	✓

✓ = Access

X = No Access

Figure 5.1: Shared Bus Memory Map

Longword Address BASE+		Byte Address SBASE+
0800 0000h	TBC Register Set	0000 0000h
0804 0000h	DBV46 Control Registers	0010 0000h
0808 0000h	SCV64 Register Set	0020 0000h
080C 0000h	Reserved	0030 0000h
0820 0000h	VME IACK Space	0080 0000h
0830 0000h	Reserved	00C0 0000h
0840 0000h	DBV46 Interrupt Registers	0100 0000h
0850 0000h	Shared PEROM (128K x 8)	0140 0000h
0860 0000h	Reserved	0180 0000h
0870 0000h	PIM Interrupt Space	01C0 0000h
0880 0000h	Reserved	0200 0000h
0888 0000h	TIM-40 A Global Memory	0220 0000h
0890 0000h	C40 B Global Memory	0240 0000h
0898 0000h	Nodes A and B Global Memory	0260 0000h
08A0 0000h	C40 C Global Memory	0280 0000h
08A8 0000h	Nodes A and C Global Memory	02A0 0000h
08B0 0000h	Nodes B and C Global Memory	02C0 0000h
08B8 0000h	Nodes A, B and C Global Memory	02E0 0000h
08C0 0000h	TIM-40 D Global Memory	0300 0000h
08C8 0000h	Nodes A and D Global Memory	0320 0000h
08D0 0000h	Nodes B and D Global Memory	0340 0000h
08D8 0000h	Nodes A, B and D Global Memory	0360 0000h
08E0 0000h	Nodes C and D Global Memory	0380 0000h
08E8 0000h	Nodes A, C and D Global Memory	03A0 0000h
08F0 0000h	Nodes B, C and D Global Memory	03C0 0000h
08F8 0000h	Nodes A, B, C and D Global Memory	03E0 0000h
0900 0000h	Shared DRAM	0400 0000h
0BFF FFFFh		07FF FFFFh
0C00 0000h	Reserved	
1000 0000h	PIM-3 Interface	No Access from the
2000 0000h	Reserved	VMEbus.
4000 0000h	VMEbus Address Space	
7FFF FFFFh		

BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map.

Processing Nodes BASE = 8000 0000h

PIM-3 Interface Module Specific

VMEbus Interface SBASE = 1000 0000h (A32 default)



Note that not all address spaces are accessible by all shared bus masters.

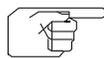
5.2 DBV46 Control Registers

All shared bus masters have access to the set of DBV46 Control Registers. These registers provide access to the general control and status functionality of DBV46 such as reset, LED control and system status. The shared bus register map is shown in [Table 5.2](#) below and each is described in the following subsections. Note that the addresses given in the table below, refer to the address of each register as mapped to the respective shared bus master's memory map. BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map (Processing Nodes - BASE = 8000 0000h, PIM-3 Interface - Module Specific, VMEbus Interface - SBASE = 1000 0000h (A32 default)).

Table 5.2: DBV46 Control Registers

DSP/PIM Address (longwords)	VMEbus Address (bytes)	Register
¹ BASE+0804 0000h	SBASE+0010 0000h	System Reset Register
BASE+0804 0001h	SBASE+0010 0004h	System Control Register
BASE+0804 0002h	SBASE+0010 0008h	LED Control Register
BASE+0804 0003h	SBASE+0010 000Ch	Communication Port Control Register
BASE+0804 0004h	SBASE+0010 0010h	System Status Register
BASE+0804 0005h	SBASE+0010 0014h	Node Identification Register
BASE+0804 0006h	SBASE+0010 0018h	Shared Bus Lock Register
BASE+0804 0007h	SBASE+0010 001Ch	Timeout Error Register
BASE+0804 0008h to BASE+0807 FFFFh	SBASE+0010 0020h to SBASE+001F FFFFh	Reflections of the above registers.

¹ Note that no access is provided to the System Reset Register from the processing nodes or PIM-3 interface.



It is recommended that any modifications made to any of the DBV46 Control Registers are performed using a 'read-modify-write' operation. Note that the VME read-modify-write cycle is not supported.

5.2.1 System Reset Register

This 8 bit read/write register is used to reset the processing nodes within your system and is cleared to zero on power-up/reset. Note that this register can ONLY be accessed by the VMEbus. The individual bit functions of this register are shown in [Table 5.3](#) below.

Table 5.3: System Reset Register

D31 - D8	D7	D6	D5	D4	D3	D2	D1	D0
Not Used	PIM_RESET	DRESET	CRESET	BRESET	ARESET	NODE_RESET	GRESET	BRD_RESET

Bit	Name	Function
0	BRD_RESET	<p>Board Reset</p> <p>Write a '1' to this bit to reset DBV46. This bit is automatically cleared to zero once the reset operation has been performed. A board reset will also:</p> <ul style="list-style-type: none"> Assert the /GRESET line. Therefore, in a multi-board system where the JTAG headers must be connected across boards (see Chapter 11), setting this bit on one board resets all the processing nodes in the system. Reset certain SCV64 internal registers by asserting /LRST (see Section 5.4 of this User Guide and Section 2.12 of the SCV64 User Manual for their reset values).
1	GRESET	<p>Global Reset</p> <p>This active high control bit asserts the global reset line, which is active low (when set to '1', the line is pulled low). The global reset line is part of the TIM-40 specification. To reset the processing nodes on DBV46, this bit should be set to '1'. To release the nodes from reset, this bit should be cleared to '0'. Note that setting this bit will reset the processing nodes only (that is, no other DBV46 resources will be reset).</p> <p>In a multi-board system where the JTAG headers must be connected across boards (see Chapter 11), setting this bit on one board resets all the processing nodes in the system.</p> <p>Note that reading back this register on a board only tells you if that board has pulled the global reset line low. This differs from the /GRESET bit in the System Status Register which reads '0' when any board in the system has pulled the line low.</p>
2	NODE_RESET	<p>All Node Reset</p> <p>1 - All nodes (A, B, C and D) held in reset 0 - Release all nodes (A, B, C and D) from reset</p>
3	ARESET	Individual Node Reset
4	BRESET	1 - Node held in reset
5	CRESET	0 - Release node from reset
6	DRESET	Note that these bits should only be used if you wish to have the option of releasing nodes from reset individually, see WARNING note below.
7	PIM_RESET	<p>1 - PIM held in reset 0 - Release PIM from reset</p>



To prevent port contention between processing nodes, all nodes connected via communication port **MUST** be reset at the same time. Therefore, it is recommended that you use the **NODE_RESET**, **BRD_RESET** or **GRESET** bits as required to simultaneously reset and release from reset all processing nodes. However, if you wish to have the option of releasing nodes from reset individually, the Individual Node Reset bits can be used. In this case all appropriate bits **MUST** set to '1' at the same time, that is, in the same write cycle. Failure to reset nodes simultaneously could cause damage to the C4x processors due to port contention.

5.2.2 System Control Register

This 16 bit read/write register can be used to lock the shared bus and is also used to define the VMEbus master cycle when performing master accesses from the processing nodes to the VMEbus. An overview of the individual bit functions of this register are shown in [Table 5.4](#) below. Shared bus timeout and locking are described in [Sections 5.8.2 and 5.8.3](#) respectively. The VMEbus master cycle functionality of this register is described in detail in [Chapter 7](#). This register is cleared to zero on power-up/reset.

Table 5.4: System Control Register

D31 - D16	D15	D14 - D8	D7	D6 - D5	D4	D3 - D2	D1	D0
Not Used	GPB	VME Master Cycle	Timeout Duration	Time Lock	Timeout Disable	Reserved	Timeout Ready	Reserved

Bit	Name	Function
0	-	Reserved, must be set to '0'.
1	TOUT_RDY	Timeout ready signal disable. 0 - Enable slave ready signal in timeout situation. 1 - Disable slave ready signal in timeout situation. The capability of disabling the slave ready signal in a timeout situation is provided for debug purposes only. The shared bus timeout facility is described further in Section 5.8.2 .
3 - 2		Reserved, must be set to '0'.
4	TOUT_DIS	Shared bus timeout disable. 0 - Shared bus timeout enabled. When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μ s (set via TOUT_DUR bit, see below). 1 - Shared bus timeout disabled. The shared bus timeout facility is described further in Section 5.8.2 .
5	TLCK_DIS	Shared bus time lock disable. 0 - Shared bus time lock enabled. Once shared bus ownership is granted, the bus will be locked for 8 or 16 complete accesses (set via TLCK_DUR bit, see below). 1 - Shared bus time lock disabled. Once shared bus ownership is granted, the bus will be locked for one complete access only. The shared bus time lock facility is described further in Section 5.8.3 .
6	TLCK_DUR	Shared bus time lock duration. The function of this bit is enabled via bit 5 (TLCK), see above. 0 - Once shared bus ownership is granted, the bus will be locked for 8 complete accesses. 1 - Once shared bus ownership is granted, the bus will be locked for 16 complete accesses. The shared bus time lock facility is described further in Section 5.8.3 .
7	TOUT_DUR	Shared bus timeout duration. 0 - When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 μ s. 1 - When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 800 μ s. The shared bus timeout facility is described further in Section 5.8.2 .
8	KADDR0	Used to set the value of the SCV64 local byte address lines (KADDR1 and KADDR0) at the start of a VME master cycle (VME Byte 0, 1, 2, or 3), see Chapter 7 .
9	KADDR1	
10	KSIZE0	Used to determine the data cycle type for a VME master cycle (D8, D16, D24 or D32 transfer), see Chapter 7 .
11	KSIZE1	
12	KFC0	Used to determine the type of VME master cycle in terms of user/supervisor program/data for A16, A24 and A32 transfers, see Chapter 7 .
13	KFC1	
14	KFC2	
15	GPB	General purpose bit. This bit is provided for general purpose use. For example, you may wish to use it to provide a software semaphore.

5.2.3 LED Control Register

This 8 bit read/write register controls the operation of the four front panel LEDs and is cleared to zero on power-up/reset. The individual bit functions of this register are shown in [Table 5.5](#).

Table 5.5: LED Control Register

D31 - D8	D7 - D6	D5 - D4	D3 - D0
Not Used	Reserved	LED Source	LED User Command

Bit	Name	Function
0 1 2 3	LEDA_CMD LEDB_CMD LEDC_CMD LEDD_CMD	These bits switch the appropriate front panel LEDs on and off. Can be used to provide user defined LED functions. These bits are only active when bits 5 and 4 are set to '11', see below. 0 - LED on 1 - LED off
5 - 4	LED_SRC	These bits are used to determine the LED functions. The individual bit settings are shown below and the LED functions are listed in Table 5.6 . Table 5.7 provides a brief description of each LED function. Bit 5 Bit 4 Function 0 0 System Status (Default) 0 1 Shared Bus Status 1 0 Node Grant 1 1 User Defined
7 - 6	-	Reserved, set to '0'.

Table 5.6: LED Source Selection

Function	LED A	LED B	LED C	LED D
System Status	+5 V Power	SYSFAIL	RESET	CONFIG
Shared Bus Status	VME Grant	Deadlock	VME_BER	TOUT
Node Grant	Node A Grant	Node B Grant	Node C Grant	Node D Grant
User Defined	LEDA_CMD	LEDB_CMD	LEDC_CMD	LEDD_CMD

Table 5.7: LED Function Description

LED Function	Description (LED On)
+5 V Power	DBV46 receiving VMEbus +5 V supply.
SYSFAIL	DBV46 has asserted /SYSFAIL.
RESET	C40 B on DBV46 is in reset. LED is cleared, when C40 B is no longer in reset.
CONFIG	At least one module/processor in your system is pulling the global configuration line low. LED is cleared when all modules/processors in your system have released this line. Reflects the status of bit 3 of the System Status Register, see Section 5.2.5 .
VME Grant	The VMEbus has ownership of the shared bus.
Deadlock	The VMEbus is being accessed by a processing node and the shared bus is being accessed by the VMEbus at the same time.
VME_BER	A VME bus error has occurred during a master access to the VMEbus.
TOUT	A shared bus timeout has occurred.
Node Grant	Indicates that the respective processing node has ownership of the shared bus.
LED_CMD	Reflects the status of the appropriate LED_CMD bit in the LED Control Register.

5.2.4 Communication Port Control Register

This 8 bit read/write register is used to select the communication port routing of ports A5, B2, C2 and D5. The register is set to AAh on power-up/reset (all bit pairs set to '10'). The individual bit functions are shown in [Table 5.8](#) below. The use and configuration of this register is described further in [Chapter 9](#)

Table 5.8: Communication Port Control Register

D31 - D8		D7 - D6		D5 - D4		D3 - D2		D1 - D0	
Not Used		D5_ROUTE	C2_ROUTE	B2_ROUTE	A5_ROUTE				
Bit	Name	Function							
1 - 0	A5_ROUTE	These pairs of bits select the routing of communication ports A5, B2, C2 and D5. The individual bit settings are shown below.							
3 - 2	B2_ROUTE								
5 - 4	C2_ROUTE	MSB	LSB	Function					
		0	0	Not used - must not be set to '00'.					
		0	1	Port routed to PIM site.					
7 - 6	D5_ROUTE	1	0	Port routed to front panel (default).					
		1	1	Port disabled.					

5.2.5 System Status Register

This 8 bit read only register can be used to monitor the system status. The individual bit functions of this register are shown in [Table 5.9](#) below.

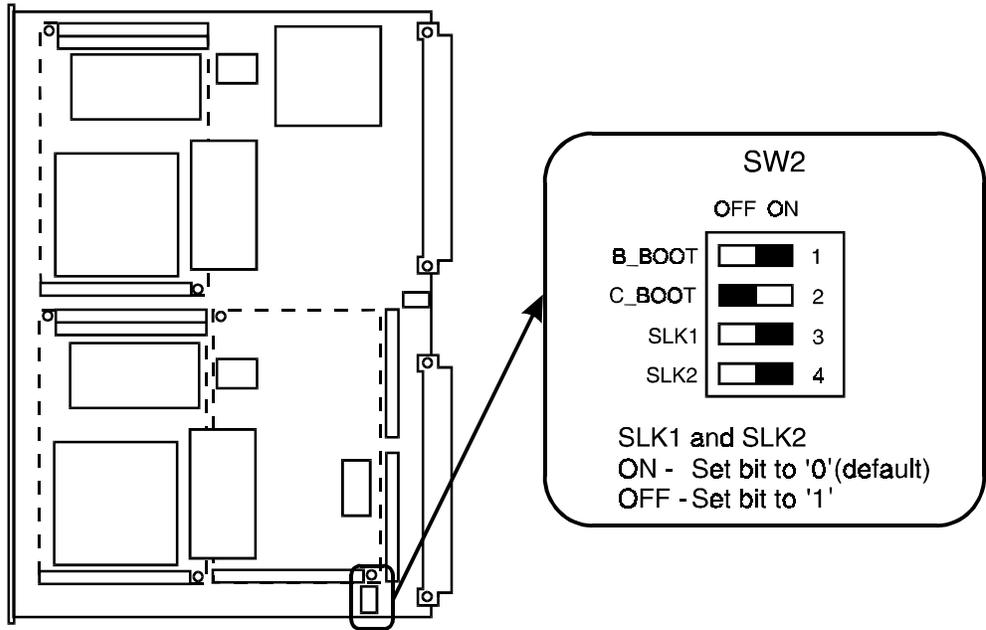
Table 5.9: System Status Register

D31 - D8	D7	D6	D5	D4	D3	D2 - D1	D0
Not Used	Reserved	/SENSEPIM	/SENSED	/SENSEA	/CONFIG	Reserved	/GRESET

Bit	Name	Function
0	/GRESET	Reflects the status of the global reset line. 0 - All boards in your system are in reset. 1 - Bit is reset to '1' when all boards are no longer in reset.
1	/SLK1	Reflects the status of the SLK1 switch of SW2. 0 - SLK1 is ON 1 - SLK1 is OFF The location and default setting of SLK1 is shown in Figure 5.2 below.
2	/SLK2	Reflects the status of the SLK2 switch of SW2. 0 - SLK2 is ON 1 - SLK2 is OFF The location and default setting of SLK2 is shown in Figure 5.2 below.
* 3	/CONFIG	Reflects the status of the global configuration line, which is part of the TIM-40 specification. When '0', at least one module/processor in your system is pulling the global configuration line low. This bit reads '1' when all modules/processors in your system have released this line. The status of this line is also reflected in the front panel CONFIG LED (ON bit 3 = 0, OFF bit 3 = 1), see Table 5.7 .
4	/SENSEA	0 - A TIM-40 module is located in Site A.
5	/SENSED	0 - A TIM-40 module is located in Site D.
6	/SENSEPIM	0 - A master PIM-3 module is present on DBV46.
7	-	Reserved, read as '0'.

* The global configuration line is pulled low during C4x bootstrap and may be released by bootcode. The Blue Wave C40 B PEROM bootcode will release the line for C40 B once processor initialisation is complete. In general, the use of the /CONFIG line is application-specific.

Figure 5.2: SLK1 and SLK2



5.2.6 Node Identification Register

This 8 bit read only register can be read by a node to determine which site it occupies. The individual bit functions of this register are shown in [Table 5.10](#) below.

Table 5.10: Node Identification Register

D31 - D8	D7 - D4	D3	D2	D1	D0
Not Used	Reserved	NODE_D	NODE_C	NODE_B	NODE_A

Bit	Name	Function
0	NODE_A	0 - Node is TIM-40 A
1	NODE_B	0 - Node is C40 B
2	NODE_C	0 - Node is C40 C
3	NODE_D	0 - Node is TIM-40 D
7 - 4	-	Reserved, read all bits as '0'.

5.2.7 Shared Bus Lock Register

This 8 bit read/write register can be used to lock the shared bus for exclusive use by the current bus master, that is, the device writing to this register. The register is cleared to zero on power-up/reset. The individual bit functions are shown in [Table 5.11](#) below. The shared bus lock facility is described further in [Section 5.8.3](#).

Table 5.11: Shared Bus Lock Register

D31 - D8	D7	D6	D5	D4	D3	D2	D1	D0
Not Used	D_LOCK	C_LOCK	B_LOCK	A_LOCK	PIM_LOCK	VME_LOCK	Reserved	SBUS_LOCK

Bit	Name	Function
0	SBUS_LOCK	Shared Bus Lock Write '0' - Unlock shared bus Write '1' - Lock shared bus for exclusive use by the current bus master, that is, the device writing to this register. Read '0' - Shared bus unlocked by the bus master indicated by bits 2 to 7. Read '1' - Shared bus locked for exclusive use by the master indicated by bits 2 to 7.
1	-	Reserved, read as '0' (read only).
2	VME_LOCK	These read only bits show which shared bus master last accessed the SBUS_LOCK bit (bit 0) and are provided for debug purposes. They can be used to show which shared bus master has locked or unlocked the shared bus. For example, if a bus master locks the shared bus via bit 0 (SBUS_LOCK) and a timeout occurs during any subsequent accesses to the shared bus, the current bus master is removed from the bus. This allows any bus master to access this register and determine which bus master, if any, has locked the shared bus. A bit set to '1' indicates that the bus has been locked by the corresponding master. Note that to remove the lock, only the bus master indicated by bits 2 to 7 can remove the lock via bit 0. This master will 'reinherit' the bus lock.
3	PIM_LOCK	
4	A_LOCK	
5	B_LOCK	
6	C_LOCK	
7	D_LOCK	

5.2.8 Timeout Error Register

This 8 bit read/write register can be used to determine which shared bus master caused a timeout error. The register is cleared to zero on power-up/reset or if written to. The individual bit functions are shown in [Table 5.12](#) below. Timeout is described further in [Section 5.8.2](#).

Table 5.12: Timeout Error Register

D31 - D8	D7	D6	D5	D4	D3	D2 - D0
Not Used	D_TOUT	C_TOUT	B_TOUT	A_TOUT	PIM_TOUT	Reserved

Bit	Name	Function
2 - 0	-	Reserved, read as '0'.
3	PIM_TOUT	These bits show which shared bus master has caused a timeout. A bit set to '1' indicates that a shared bus timeout has occurred, caused by the corresponding master. To clear a bit to '0', simply write any value to this register.
4	A_TOUT	
5	B_TOUT	
6	C_TOUT	
7	D_TOUT	

5.3 DBV46 Interrupt Registers

Three of the four IIOF lines (IIOF0, 1 and 2) provided by each processing node can be configured to receive or generate interrupts on DBV46. IIOF3 is used to control the /CONFIG line.

The source and direction of interrupts for all the processing nodes are configured via the DBV46 Interrupt Registers. Each 'Group' Register listed in the table above refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2). The IIOF lines are controlled via the C4x processor *IIOF Flag Register*, a summary of this register is given in Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide. [Chapter 10](#) of this TRM details the configuration of interrupts to/from the processing nodes and provides example step by step procedures.

All shared bus masters have access to the set of DBV46 Interrupt Registers. The shared bus register map is shown in [Table 5.13](#) below and each is described in the following subsections. Note that the addresses given in the table below, refer to the address of each register as mapped to the respective shared bus master's memory map. BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map (Processing Nodes - BASE = 8000 0000h, PIM-3 Interface - Module Specific, VMEbus Interface - SBASE = 1000 0000h (A32 default)).

Table 5.13: DBV46 Interrupt Registers

DSP/PIM Address (longwords)	VMEbus Address (bytes)	Register
<i>BASE+0840 0000h</i>	SBASE+0100 0000h	Group 0 Interrupt Enable Register
<i>BASE+0840 0001h</i>	SBASE+0100 0004h	Group 1 Interrupt Enable Register
<i>BASE+0840 0002h</i>	SBASE+0100 0008h	Group 2 Interrupt Enable Register
<i>BASE+0840 0003h</i>	SBASE+0100 000Ch	Reserved
<i>BASE+0840 0004h</i>	SBASE+0100 0010h	Reserved
<i>BASE+0840 0005h</i>	SBASE+0100 0014h	Reserved
<i>BASE+0840 0006h</i>	SBASE+0100 0018h	Group 0 Interrupt Line Status Register
<i>BASE+0840 0007h</i>	SBASE+0100 001Ch	Group 1 Interrupt Line Status Register
<i>BASE+0840 0008h</i>	SBASE+0100 0020h	Group 2 Interrupt Line Status Register
<i>BASE+0840 0009h</i>	SBASE+0100 0024h	Group 0 Interrupt Condition Status Register
<i>BASE+0840 000Ah</i>	SBASE+0100 0028h	Group 1 Interrupt Condition Status Register
<i>BASE+0840 000Bh</i>	SBASE+0100 002Ch	Group 2 Interrupt Condition Status Register
<i>BASE+0840 000Ch</i>	SBASE+0100 0030h	Group 0 Interrupt Pending Register
<i>BASE+0840 000Dh</i>	SBASE+0100 0034h	Group 1 Interrupt Pending Register
<i>BASE+0840 000Eh</i>	SBASE+0100 0038h	Group 2 Interrupt Pending Register
<i>BASE+0840 000Fh</i>	SBASE+0100 003Ch	Reserved
<i>BASE+0840 0010h to BASE+084F FFFFh</i>	SBASE+0100 0040h to SBASE+013F FFFFh	Reflections of the above registers.



It is recommended that any modifications made to any of the DBV46 Interrupt Registers are performed using a read-modify-write operation. Note that the VME read-modify-write cycle is not supported.

5.3.1 Interrupt Enable Registers

These 16 bit read/write registers are used to select and enable the interrupt source(s) of the IIOF lines for each processing node. Each 'Group' Register refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2). A bit set to '1' enables an interrupt to be generated on the appropriate IIOF line from the corresponding source. This will also enable the appropriate bits of the Interrupt Pending Register, see [Section 5.3.2](#).

These registers are set to F000h on power-up/reset (bit 15 to bit 12 set to '1') and the individual bit functions are shown in [Table 5.14](#) below. Refer to the appropriate reference sections for more detailed information on the individual interrupts.

Table 5.14: Interrupt Enable Registers

D31 - D16	D15 - D11	D10	D9 - D8	D7	D6 - D2	D1 - D0
Not Used	Interrupt Source Select/Enable	Reserved	Interrupt Source Select/Enable	Reserved	Interrupt Source Select/Enable	Reserved

Bit	Name	Function
1 - 0	-	Reserved - must be set to '0', undefined for reads.
2	VME_INT	Enables an interrupt from the VMEbus (Sections 6.3 and 10.2).
3	VBER_INT	VME bus error interrupt (Sections 7.4 and 10.2).
4	DLK_INT	Deadlock interrupt. A deadlock will occur if the VMEbus is being accessed by a processing node and the shared bus is being accessed by the VMEbus at the same time (Section 5.8.4).
5	SCV_INT	Enables interrupts from the SCV64 device (Sections 7.4 and 10.2).
6	LM_INT	SCV64 location monitor interrupt (Sections 6.2 and 10.2).
7	-	Reserved - must be set to '0'.
8	PIM0_INT	Enables interrupt via the /EXT_INT line from the PIM site (Sections 9.2 and 10.2).
9	PIM1_INT	Enables interrupt via the /EXT_INT1 line from the PIM site (Sections 9.2 and 10.2).
10	-	Reserved - must be set to '0'.
11	TOUT_INT	Shared bus timeout interrupt. When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μ s (Section 5.8.2). The timeout duration is set via bit 6 (TOUT_DUR) of the System Control Register.
12	A_IIOF_INT	Enables TIM-40 Site A to configure its IIOF line as an output (Chapter 10). This enables the module to generate an interrupt to one or more of the other processing nodes (assuming their IIOF_INT bits are set to '0').
13	B_IIOF_INT	Enables C40 B to configure its IIOF line as an output (Chapter 10). This enables the module to generate an interrupt to one or more of the other processing nodes (assuming their IIOF_INT bits are set to '0').
14	C_IIOF_INT	Enables C40 C to configure its IIOF line as an output (Chapter 10). This enables the module to generate an interrupt to one or more of the other processing nodes (assuming their IIOF_INT bits are set to '0').
15	D_IIOF_INT	Enables TIM-40 Site D to configure its IIOF line as an output (Chapter 10). This enables the module to generate an interrupt to one or more of the other processing nodes (assuming their IIOF_INT bits are set to '0').



The special Blue Wave bootcode contained in the C40 B PEROM will configure the Interrupt Enable Registers to enable LM_INT on IIOF0 and SCV_INT on IIOF2. Therefore, in general, if you wish to use an Blue Wave software support package you must not alter this configuration. If you wish to make use of these IIOF lines from your DSP or host code, refer to your software support package User Guide which will describe how to achieve this.

5.3.2 Interrupt Pending Registers

These 16 bit read/write registers reflect the pending interrupts for each processing node. Each 'Group' Register refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2). A bit set to '1' indicates an interrupt on the appropriate IIOF line from the corresponding source. Note that the individual bits of this register are only active if they have been enabled via the Interrupt Enable Register, see [Section 5.3.1](#).

The individual bit functions are shown in [Table 5.15](#) below. Refer to the appropriate reference sections for more detailed information on the individual interrupts.

Table 5.15: Interrupt Pending Registers

D31 - D16	D15 - D11	D10	D9 - D8	D7	D6 - D0
Not Used	Interrupt Pending	Reserved	Interrupt Pending	Reserved	Interrupt Pending

Bit	Name	Function																																				
2 - 0	VME_INT	<p>Interrupt from the VMEbus (Sections 6.3 and 10.2). This three bit value indicates the VMEbus interrupt level.</p> <table border="1"> <thead> <tr> <th>Bit 2</th> <th>Bit 1</th> <th>Bit 0</th> <th>VME Interrupt Level</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>No interrupt received.</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>IRQ1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>IRQ2</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>IRQ3</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>IRQ4</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>IRQ5</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>IRQ6</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>IRQ7</td> </tr> </tbody> </table> <p>To clear a VMEbus interrupt you must perform a VME IACK cycle. This is described in Section 6.3. Note that interrupts TO the VMEbus are described in detail in Section 7.4.</p>	Bit 2	Bit 1	Bit 0	VME Interrupt Level	0	0	0	No interrupt received.	0	0	1	IRQ1	0	1	0	IRQ2	0	1	1	IRQ3	1	0	0	IRQ4	1	0	1	IRQ5	1	1	0	IRQ6	1	1	1	IRQ7
Bit 2	Bit 1	Bit 0	VME Interrupt Level																																			
0	0	0	No interrupt received.																																			
0	0	1	IRQ1																																			
0	1	0	IRQ2																																			
0	1	1	IRQ3																																			
1	0	0	IRQ4																																			
1	0	1	IRQ5																																			
1	1	0	IRQ6																																			
1	1	1	IRQ7																																			
3	VBER_INT	<p>VME bus error interrupt (Sections 7.4 and 10.22). This interrupt is cleared by writing a '1' to this bit and a '0' to the VBERR bit of the SCV64 DCSR Register, see Section 7.4.</p>																																				
4	DLK_INT	<p>Deadlock interrupt. A deadlock will occur if the VMEbus is being accessed by a processing node and the shared bus is being accessed by the VMEbus at the same time (Section 5.8.4). This interrupt is cleared by writing a '1' to this bit.</p>																																				
5	SCV_INT	<p>Interrupt from the SCV64 device (Sections 7.4 and 10.2). This interrupt is cleared by writing a '0' to the appropriate bit of the SCV64 DCSR Register, see Section 5.4.</p>																																				
6	LM_INT	<p>SCV64 location monitor interrupt (Sections 6.2 and 10.2). This interrupt is cleared by reading the SCV64 LMFIFO Register, see Section 5.4.</p>																																				
7	-	Reserved - read as '0'.																																				
8	PIM0_INT	Interrupt via the /EXT_INT line from the PIM site (Sections 9.2 and 10.2).																																				
9	PIM1_INT	Interrupt via the /EXT_INT1 line from the PIM site (Sections 9.2 and 10.2).																																				
10	-	Reserved - read as '0'.																																				
11	TOUT_INT	<p>Shared bus timeout interrupt (Section 5.8). When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μs (Section 5.8.2). The timeout duration is set via bit 6 (TOUT_DUR) of the System Control Register. This interrupt is cleared by writing a '1' to this bit.</p>																																				
12	A_IIOF_INT	<p>Interrupt from TIM-40 Site A (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
13	B_IIOF_INT	<p>Interrupt from C40 B (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
14	C_IIOF_INT	<p>Interrupt from C40 C (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
15	D_IIOF_INT	<p>Interrupt from TIM-40 Site D (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				

5.3.3 Interrupt Line Status Registers

These 16 bit read/write registers reflect the status of the 'external' interrupt line used to assert the IIOF line of each processing node. These registers can also be polled to act as a 'trigger' mechanism if interrupts are not desirable. Note that since the bits reflect the status of the actual interrupt lines, only the bits referring to level-triggered interrupts are of use. The Interrupt Condition Status Registers should be used to reflect the IIOF interrupt condition status for each processing node, see [Section 5.3.4](#) below.

Each 'Group' Register refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2). Unlike the Interrupt Pending Registers, the bits in these registers are always active. The individual bit functions are shown in [Table 5.16](#) below. Refer to the appropriate reference sections for more detailed information on the individual interrupts.

Table 5.16: Interrupt Line Status Registers

D31 - D16	D15 - D12	D11 - D10	D9 - D8	D7	D6 - D5	D4 - D3	D2 - D0
Not Used	Interrupt Line Status	Not Used	Interrupt Line Status	Reserved	Interrupt Line Status	Not Used	Interrupt Line Status

Bit	Name	Function																																				
2 - 0	VME_INT	<p>Reflects the status of the /IRQ interrupt lines from the VMEbus (Sections 6.3 and 10.2). This three bit value indicates the VMEbus interrupt level.</p> <table border="1"> <thead> <tr> <th>Bit 2</th> <th>Bit 1</th> <th>Bit 0</th> <th>VME Interrupt Level</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>No interrupt received.</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>IRQ1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>IRQ2</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>IRQ3</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>IRQ4</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>IRQ5</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>IRQ6</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>IRQ7</td> </tr> </tbody> </table> <p>To clear a VMEbus interrupt you must perform a VME IACK cycle. This is described in Section 6.3. Note that interrupts TO the VMEbus are described in detail in Section 7.4.</p>	Bit 2	Bit 1	Bit 0	VME Interrupt Level	0	0	0	No interrupt received.	0	0	1	IRQ1	0	1	0	IRQ2	0	1	1	IRQ3	1	0	0	IRQ4	1	0	1	IRQ5	1	1	0	IRQ6	1	1	1	IRQ7
Bit 2	Bit 1	Bit 0	VME Interrupt Level																																			
0	0	0	No interrupt received.																																			
0	0	1	IRQ1																																			
0	1	0	IRQ2																																			
0	1	1	IRQ3																																			
1	0	0	IRQ4																																			
1	0	1	IRQ5																																			
1	1	0	IRQ6																																			
1	1	1	IRQ7																																			
3	-	Not used - read as '0'.																																				
4	-	Not used - read as '0'.																																				
5	SCV_INT	Reflects the status of the /SCV_INT interrupt line from the SCV64 device (Sections 7.4 and 10.2). This interrupt is cleared by writing a '0' to the appropriate bit of the SCV64 DCSR Register, see Section 5.4 .																																				
6	LM_INT	Reflects the status of the SCV64 location monitor interrupt line, /LM_INT (Sections 6.2 and 10.2). This interrupt is cleared by reading the SCV64 LMFIFO Register, see Section 5.4 .																																				
7	-	Reserved - read as '0'.																																				

Table 5.16: Continued

Bit	Name	Function
8	PIM0_INT	Reflects the status of the /EXT_INT interrupt line from the PIM site (Sections 9.2 and 10.2). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. The type of interrupt generated is described in your PIM module User Manual.
9	PIM1_INT	Reflects the status of the /EXT_INT1 interrupt line from the PIM site (Sections 9.2 and 10.2). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. The type of interrupt generated is described in your PIM module User Manual.
10	-	Not used - read as '0'.
11	-	Not used - read as '0'.
12	A_IIOF_INT	Reflects the status of the TIM-40 Site A IIOF line (when used as an output to interrupt another processing node). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. This is configured via the internal C4x IIOF Flag Register, see Section 10.2 of this TRM and Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide.
13	B_IIOF_INT	Reflects the status of the C40 B IIOF line (when used as an output to interrupt another processing node). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. This is configured via the internal C4x IIOF Flag Register, see Section 10.2 of this TRM and Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide.
14	C_IIOF_INT	Reflects the status of the C40 C IIOF line (when used as an output to interrupt another processing node). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. This is configured via the internal C4x IIOF Flag Register, see Section 10.2 of this TRM and Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide.
15	D_IIOF_INT	Reflects the status of the TIM-40 Site D IIOF line (when used as an output to interrupt another processing node). This interrupt is cleared by writing a '1' to this bit in the Interrupt Pending Register. Note that to make use of this bit, the interrupt must be level-triggered. This is configured via the internal C4x IIOF Flag Register, see Section 10.2 of this TRM and Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide.

5.3.4 Interrupt Condition Status Registers

These 16 bit read/write registers reflect the IIOF interrupt condition status for each processing node. Note that unlike the Interrupt Pending Register the bits in these registers are always active and act as interrupt condition flags. Each 'Group' Register refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2). A bit set to '1' indicates an interrupt condition is true.

The individual bit functions are shown in [Table 5.17](#) below. Refer to the appropriate reference sections for more detailed information on the individual interrupts.

Table 5.17: Interrupt Condition Status Registers

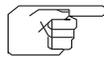
D31 - D16	D15 - D11	D10	D9 - D8	D7	D6 - D0
Not Used	Interrupt Status	Reserved	Interrupt Status	Reserved	Interrupt Status

Bit	Name	Function																																				
2 - 0	VME_INT	<p>Interrupt from the VMEbus (Sections 6.3 and 10.2). This three bit value indicates the VMEbus interrupt level.</p> <table border="0"> <tr> <td>Bit 2</td> <td>Bit 1</td> <td>Bit 0</td> <td>VME Interrupt Level</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>No interrupt received.</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>IRQ1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>IRQ2</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>IRQ3</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>IRQ4</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>IRQ5</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>IRQ6</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>IRQ7</td> </tr> </table> <p>To clear a VMEbus interrupt you must perform a VME IACK cycle. This is described in Section 6.3. Note that interrupts TO the VMEbus are described in detail in Section 7.4.</p>	Bit 2	Bit 1	Bit 0	VME Interrupt Level	0	0	0	No interrupt received.	0	0	1	IRQ1	0	1	0	IRQ2	0	1	1	IRQ3	1	0	0	IRQ4	1	0	1	IRQ5	1	1	0	IRQ6	1	1	1	IRQ7
Bit 2	Bit 1	Bit 0	VME Interrupt Level																																			
0	0	0	No interrupt received.																																			
0	0	1	IRQ1																																			
0	1	0	IRQ2																																			
0	1	1	IRQ3																																			
1	0	0	IRQ4																																			
1	0	1	IRQ5																																			
1	1	0	IRQ6																																			
1	1	1	IRQ7																																			
3	VBER_INT	<p>VME bus error interrupt (Sections 7.4 and 10.2). This interrupt is cleared by writing a '1' to this bit and a '0' to the VBERR bit of the SCV64 DCSR Register, see Section 7.4.</p>																																				
4	DLK_INT	<p>Deadlock interrupt. A deadlock will occur if the VMEbus is being accessed by a processing node and the shared bus is being accessed by the VMEbus at the same time (Section 5.8.4). This interrupt is cleared by writing a '1' to this bit.</p>																																				
5	SCV_INT	<p>Interrupt from the SCV64 device (Sections 7.4 and 10.2). This interrupt is cleared by writing a '0' to the appropriate bit of the SCV64 DCSR Register, see Section 5.4.</p>																																				
6	LM_INT	<p>SCV64 location monitor interrupt (Sections 6.2 and 10.2). This interrupt is cleared by reading the SCV64 LMFIFO Register, see Section 5.4.</p>																																				
7	-	Reserved - read as '0'.																																				
8	PIM0_INT	Interrupt via the /EXT_INT line from the PIM site (Sections 9.2 and 10.2).																																				
9	PIM1_INT	Interrupt via the /EXT_INT1 line from the PIM site (Sections 9.2 and 10.2).																																				
10	-	Reserved - read as '0'.																																				
11	TOUT_INT	<p>Shared bus timeout interrupt (Section 5.8). When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μs. The timeout duration is set via bit 6 (TOUT_DUR) of the System Control Register. This interrupt is cleared by writing a '1' to this bit.</p>																																				
12	A_IIOF_INT	<p>Interrupt from TIM-40 Site A (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
13	B_IIOF_INT	<p>Interrupt from C40 B (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
14	C_IIOF_INT	<p>Interrupt from C40 C (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				
15	D_IIOF_INT	<p>Interrupt from TIM-40 Site D (Chapter 10). This interrupt is cleared by writing a '1' to this bit.</p>																																				

5.4 SCV64 Register Set

All shared bus masters have access to the set of SCV64 control and status registers. These registers allow you to program the operating mode of the SCV64 device, that is, DBV46's VME master/slave interface. The shared bus register map is shown in [Table 5.18](#) below. Note that the addresses given in the table below, refer to the address of each register as mapped to the respective shared bus master's memory map. BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map (Processing Nodes - BASE = 8000 0000h, PIM-3 Interface - Module Specific, VMEbus Interface - SBASE = 1000 0000h (A32 default)).

[Table 5.19](#) lists the SCV64 registers which are configured at boot-up by the PEROM bootcode on C40 B supplied with DBV46. Only the registers affected by this bootcode are listed here. All other SCV64 registers take their default reset values as described in the SCV64 User Manual. The SCV64 registers are discussed further in [Chapters 6](#) and [7](#). Detailed information on each register is given in the SCV64 User Manual.



It is recommended that any modifications made to any of the SCV64 registers are performed using a read-modify-write operation. Note that the VME read-modify-write cycle is not supported.

Note that the C40 B bootcode and Blue Wave DBV46 software support packages make use of the following SCV64 registers:

- Mailbox Register 1 (MBOX1). If you wish to use a Blue Wave software support package you should not access this register. Three further Mailbox Registers (MBOX0, MBOX2 and MBOX3) are available for use.
- Certain SCV64 registers are used to initiate DMA controlled transfers. Therefore, in general, if you wish to use a Blue Wave software support package you should ensure that your host and DSP code avoids accessing these registers. If you wish to perform SCV64 DMA controlled transfers via your host/DSP code, refer to your software support package User Guide which will describe how to achieve this.

Table 5.18: SCV64 Register Set

DSP/PIM Address (longwords)	VMEbus Address (bytes)	Register	Name
BASE+0808 0000h	SBASE+0020 0000h	DMA Local Address	DMALAR
BASE+0808 0001h	SBASE+0020 0004h	DMA VMEbus Address	DMAVAR
BASE+0808 0002h	SBASE+0020 0008h	DMA Transfer Count	DMATC
BASE+0808 0003h	SBASE+0020 000Ch	Control and Status	DCSR
BASE+0808 0004h	SBASE+0020 0010h	VMEbus Slave Base Address	VMEBAR
BASE+0808 0005h	SBASE+0020 0014h	Rx FIFO Data	RXDATA
BASE+0808 0006h	SBASE+0020 0018h	Rx FIFO Address Register	RXADDR
BASE+0808 0007h	SBASE+0020 001Ch	Rx FIFO Control Register	RXCTL
BASE+0808 0008h	SBASE+0020 0020h	VMEbus/VSB Bus Select	BUSSEL
BASE+0808 0009h	SBASE+0020 0024h	VMEbus Interrupter Vector	IVECT
BASE+0808 000Ah	SBASE+0020 0028h	Access Protect Boundary	APBR
BASE+0808 000Bh	SBASE+0020 002Ch	Tx FIFO Data Output Latch	TXDATA
BASE+0808 000Ch	SBASE+0020 0030h	Tx FIFO Address Output Latch	TXADDR
BASE+0808 000Dh	SBASE+0020 0034h	Tx FIFO AM Code And Control Bit Latch	TXCTL
BASE+0808 000Eh	SBASE+0020 0038h	Location Monitor FIFO Read Port	LMFIFO
BASE+0808 000Fh	SBASE+0020 003Ch	SCV64 Mode Control	MODE
BASE+0808 0010h	SBASE+0020 0040h	Slave A64 Base Address	SA64BAR
BASE+0808 0011h	SBASE+0020 0044h	Master A64 Base Address	MA64BAR
BASE+0808 0012h	SBASE+0020 0048h	Local Address Generator	LAG
BASE+0808 0013h	SBASE+0020 004Ch	DMA VMEbus Transfer Count	DMAVTC
BASE+0808 0014h to BASE+0808 001Fh	SBASE+0020 0050h to SBASE+0020 007Fh	Reserved	-
BASE+0808 0020h	SBASE+0020 0080h	Status Register 0	STAT0
BASE+0808 0021h	SBASE+0020 0084h	Status Register 1	STAT1
BASE+0808 0022h	SBASE+0020 0088h	General Control Register	GENCTL
BASE+0808 0023h	SBASE+0020 008Ch	VMEbus Interrupter Requester	VINT
BASE+0808 0024h	SBASE+0020 0090h	VMEbus Requester Register	VREQ
BASE+0808 0025h	SBASE+0020 0094h	VMEbus Arbiter Register	VARB
BASE+0808 0026h	SBASE+0020 0098h	ID Register	ID
BASE+0808 0027h	SBASE+0020 009Ch	Control and Status Register	CTL2
BASE+0808 0028h	SBASE+0020 00A0h	Level 7 Interrupt Status Register	7IS
BASE+0808 0029h	SBASE+0020 00A4h	Local Interrupt Status Register	LIS
BASE+0808 002Ah	SBASE+0020 00A8h	Level 7 Interrupt Enable Register	7IE
BASE+0808 002Bh	SBASE+0020 00ACh	Local Interrupt Enable Register	LIE
BASE+0808 002Ch	SBASE+0020 00B0h	VMEbus Interrupt Enable Register	VIE
BASE+0808 002Dh	SBASE+0020 00B4h	Local Interrupts 1 & 0 Control Register	IC10
BASE+0808 002Eh	SBASE+0020 00B8h	Local Interrupts 3 & 2 Control Register	IC32
BASE+0808 002Fh	SBASE+0020 00BCh	Local Interrupts 5 & 4 Control Register	IC54
BASE+0808 0030h	SBASE+0020 00C0h	Miscellaneous Control Register	MISC
BASE+0808 0031h	SBASE+0020 00C4h	Delay Line Control Register	DLCT
BASE+0808 0032h	SBASE+0020 00C8h	Delay Line Status Register 1	DLST1
BASE+0808 0033h	SBASE+0020 00CCh	Delay Line Status Register 2	DLST2
BASE+0808 0034h	SBASE+0020 00D0h	Delay Line Status Register 3	DLST3
BASE+0808 0035h	SBASE+0020 00D4h	Mailbox Register 0	MBOX0
BASE+0808 0036h	SBASE+0020 00D8h	Mailbox Register 1	MBOX1
BASE+0808 0037h	SBASE+0020 00DCh	Mailbox Register 2	MBOX2
BASE+0808 0038h	SBASE+0020 00E0h	Mailbox Register 3	MBOX3
BASE+0808 0039h to BASE+0808 007Fh	SBASE+0020 00E4h to SBASE+0020 01FFh	Reserved	-
BASE+0808 0080h to BASE+080B FFFFh	-	Reflections of the above registers.	-

Table 5.19: SCV64 Default Register Values

SCV64 Register	PEROM Setting	Description
VMEBAR	0070 01C2h (DBV46D0) or 0070 01E2h (DBV46D1 and DBV46D2)	Sets an A32 VME base address of 1000 0000h and image size of 64M bytes (DBV46D0) or 128M bytes (DBV46 D1 and DBV46D2). Also sets an A24 base address of 800000h and image size of 4M bytes.
BUSSEL	0000 0000h	This ensures that the VMEbus is selected for the entire SCV64 VME address space mapping. This setting should not be changed since the VSB bus has not been implemented on DBV46.
GENCTL	0000 001Ch	This setting configures the VME IRQ1 line as an interrupt line only and sets the Tick Timer interval to 100 ms. This setting is used by Blue Wave's DBV46 software support packages and should not be altered.
VREQ	0000 00BFh	This setting configures the SCV64 VMEbus requester to use Level 3 (highest priority) when requesting the VMEbus.
VARB	0000 0034h	This setting configures the SCV64 VMEbus arbiter to use the standard round robin arbitration scheme with arbitration timeout enabled. This is only valid when the SCV64 is the system controller.
CTL2	0000 0002h	This setting ensures that the Tick Timer interval is set to normal rates.
MODE	3C00 E001h	This setting: <ul style="list-style-type: none"> • Enables all programmed DBV46 slave images ensuring that the A24 slave image is disabled. • Sets the DBV46 shared bus maximum burst length to 32 longwords and enables burst mode. Burst mode is used when performing slave block transfers to DBV46, see Section 6.1.1, and SCV64 DMA controlled transfers to DBV46. • Allows for unaligned VMEbus transfers for master accesses from DBV46. • Enables DMA burst mode.
MISC	0000 0000h	This releases the VMEbus /SYSFAIL line.

5.5 TBC Register Set

The Test Bus Controller (TBC) on DBV46 provides essentially the same functionality as Texas Instruments' XDS510 emulation system and Blue Wave's XDSC40 system. This emulation system can be used to monitor the board via a JTAG scan path which is routed through all processing nodes. The on-board TBC, together with the shared memory resources, is primarily used by Blue Wave to implement the DB40 debugger and the interface libraries which are provided with Blue Wave's development support packages. **You should not need to access the TBC directly if you have purchased a Blue Wave software support package.** The JTAG emulation system is discussed in detail in [Chapter 11](#). The shared bus register map is given in [Table 5.20](#) below. Note that these registers can only be accessed from the VMEbus.

Table 5.20: TBC Register Set

VME Address (bytes)	Register	Read/Write
SBASE+02h	JTAG Control 0	Read/Write
SBASE+06h	JTAG Control 1	Read/Write
SBASE+0Ah	JTAG Control 2	Read/Write
SBASE+0Eh	JTAG Control 3	Read/Write
SBASE+12h	JTAG Control 4	Read/Write
SBASE+16h	JTAG Control 5	Read/Write
SBASE+1Ah	JTAG Control 6	Read/Write
SBASE+1Eh	JTAG Control 7	Read/Write
SBASE+22h	JTAG Control 8	Read/Write
SBASE+26h	JTAG Control 9	Read/Write
SBASE+2Ah	JTAG Minor Command	Read/Write
SBASE+2Eh	JTAG Major Command	Read/Write
SBASE+32h	JTAG Counter 1 Update 0	Read/Write
SBASE+36h	JTAG Counter 1 Update 1	Read/Write
SBASE+3Ah	JTAG Counter 2 Update 0	Read/Write
SBASE+3Eh	JTAG Counter 2 Update 1	Read/Write
SBASE+42h	JTAG Status 0	Read Only
SBASE+46h	JTAG Status 1	Read Only
SBASE+4Ah	JTAG Status 2	Read Only
SBASE+4Eh	JTAG Status 3	Read Only
SBASE+52h	JTAG Capture 0	Read Only
SBASE+56h	JTAG Capture 1	Read Only
SBASE+5Ah	JTAG Read Buffer	Read Only
SBASE+5Eh	JTAG Write Buffer	Write Only
SBASE+60h to SBASE+7Eh	Reserved	-

SBASE refers to the VMEbus base address of DBV46. SBASE+00h, SBASE+04h etc. are not used.

The 16 bit TBC registers are mapped into a 32 longword area. They reside on data lines D15 to D0 so that D16 accesses from the host may be used if required. Data lines D31 to D16 do not contain any valid information. Note that block transfers to the TBC are not supported.



Note that access to the TBC register set requires access to the DBV46 shared bus. Therefore, when using a debugger if your code locks the shared bus for exclusive use this can cause the debugger to hang or timeout if the code is halted before releasing the shared bus.

5.6 DBV46 Shared Memory Resources

5.6.1 Shared DRAM

DRAM can be supplied on DBV46 which is accessible via the shared bus. This DRAM can be used to implement a shared memory architecture between the shared bus masters. There are three factory-fitted size options:

- DBV46D0 - No DRAM
- DBV46D1 - 4M x 32 (16M bytes) shared DRAM
- DBV46D2 - 16M x 32 (64M bytes) shared DRAM

The mapping of the shared DRAM in the respective memory maps of each of the shared bus masters, is shown in the appropriate chapters of this TRM.

Fast page mode DRAM is implemented on DBV46 with a page size of 1K x 32. Once the shared bus has been granted, accessing the DRAM incurs one wait state within a page and 4 ws across a page. A DRAM refresh is performed automatically every 14 μ s and takes 240 ns.

5.6.2 Shared PEROM

A 128K byte PEROM is supplied as standard on DBV46 which is accessible via the shared bus. This PEROM can be used to store common bootcode or configuration information or to store embedded application code. Note that the shared PEROM supplied with DBV46 does not contain any code. The mapping of the shared PEROM in the respective memory maps of each of the shared bus masters, is shown in the appropriate chapters of this TRM.

The PEROM device is connected to data lines D0 to D7. Note that data lines D8 to D31 are not used by the PEROM and are undefined for reads. Once the shared bus has been granted, accessing the PEROM incurs five wait states.



Note that due to access time requirements, data corruption may occur if a read from the shared PEROM is immediately followed by a read or write to another shared bus resource. This situation is easily avoided by simply ensuring that at least two assembly instructions that do not access the PEROM (such as no operation, NOP, instructions) are inserted in your code between these accesses.



At boot-up, the PEROM bootcode supplied for C40 B automatically checks the shared PEROM. If the shared PEROM is empty, the booting procedure described in [Section 3.6.2](#) will continue. If bootcode is stored in the shared PEROM, C40 B will boot from this code.

The shared PEROM can also be programmed via the shared bus by any of the shared bus masters, allowing it to be updated in situ. The procedure for programming a PEROM is explained below.

The support software supplied with Blue Wave carrier board software development packages contains a PEROM programming tool for use with DBV46. This tool enables you to program the shared PEROM with user defined code/data. Details of this software are given in the C4x VME Support Manual which accompanies the support software. If you have not purchased a software support package, please contact your distributor for more information. If you do not have this software or wish to perform a PEROM programming function not covered by the tools, then the method for doing this is detailed below.

To protect the PEROM from accidental programming, a software data protection algorithm is present on the PEROM. The PEROM is supplied with this algorithm enabled, therefore you must first disable the algorithm if you wish to write to the PEROM.

The software data protection algorithm consists of a series of three program commands to specific addresses with specific data (see [Figure 5.3\(i\)](#)). After the software data protection is enabled, the same three program commands must begin each program cycle in order for programming to occur. Once set, the software data protection feature remains active unless its disable command is issued. Power transitions will not reset the software data protection feature, however the software feature will guard against inadvertent program cycles during power transitions. In order to disable this algorithm so that you can write to the PEROM the programming steps detailed in [Figure 5.3\(ii\)](#) must be performed.

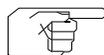


Note that due to access time requirements, data corruption may occur if a read from the shared PEROM is immediately followed by a read write. This situation is easily avoided by simply ensuring that at least two assembly instructions that do not write to the PEROM (such as no operation, NOP, instructions) are inserted in your code between a read and write.

The device is programmed on a page basis, each page being 128 bytes in length. If a single byte is to be changed, the whole of that page has to be written. Any location that is not loaded during the programming of the page is erased to read FFh. Writes in each page must be performed within 150 μ s of each other.

Once the page has been written to the device there is a period where those values are programmed into the storage matrix. Data polling allows the device performing the PEROM programming to attempt to read the last value written to the data page and will return the complement of the data loaded on bit 7 of the byte. This byte may be polled and when the programming cycle has completed bit 7 will reflect the true value of the data written to it. This should take no longer than 10 ms.

[Figure 5.4](#) gives a general flow chart which shows the steps necessary to allow you to write to the PEROM.



Due to the properties of the PEROM device, data integrity can only be assured up to 1000 programming cycles.

Figure 5.3: Software Data Protection Algorithm

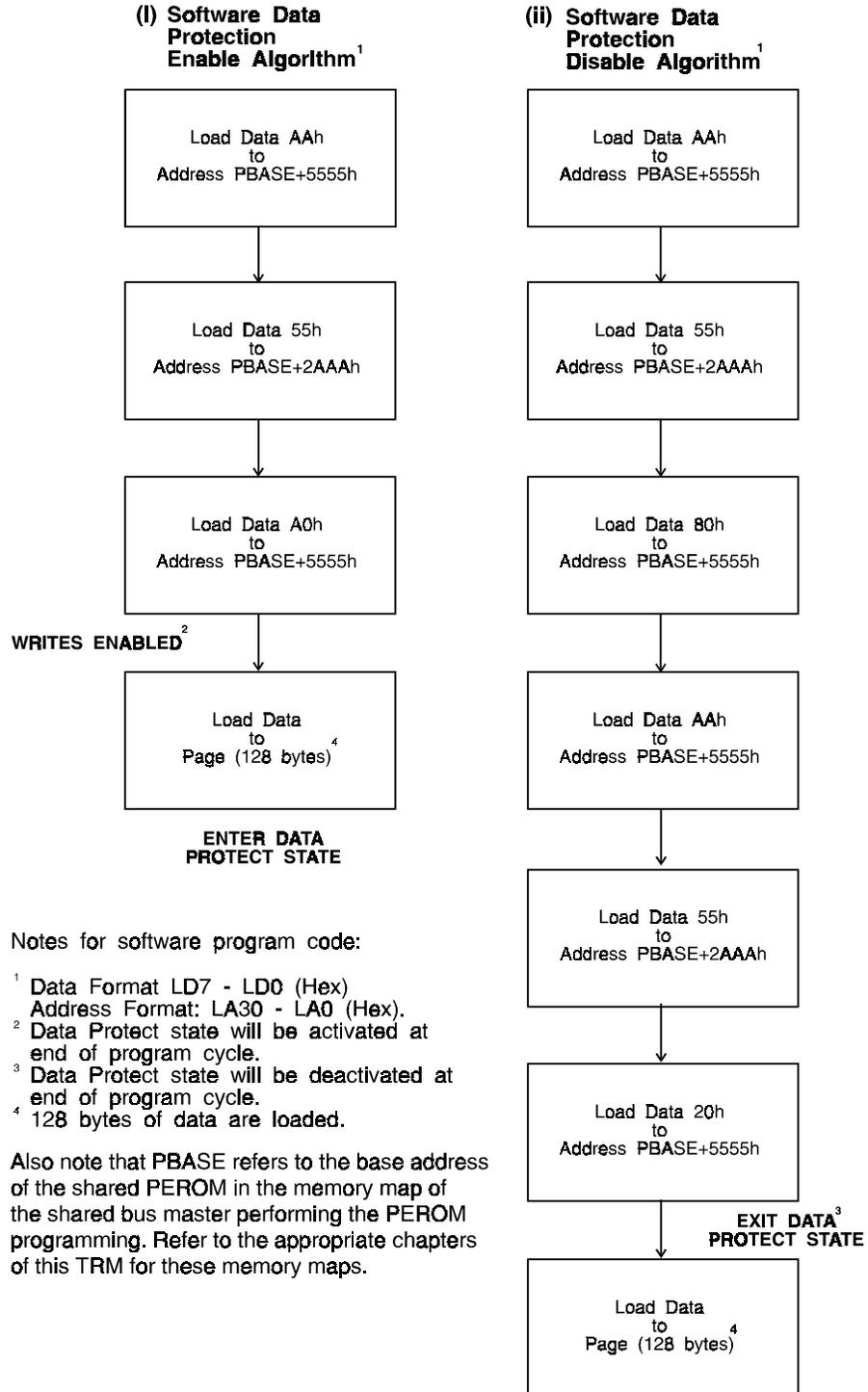
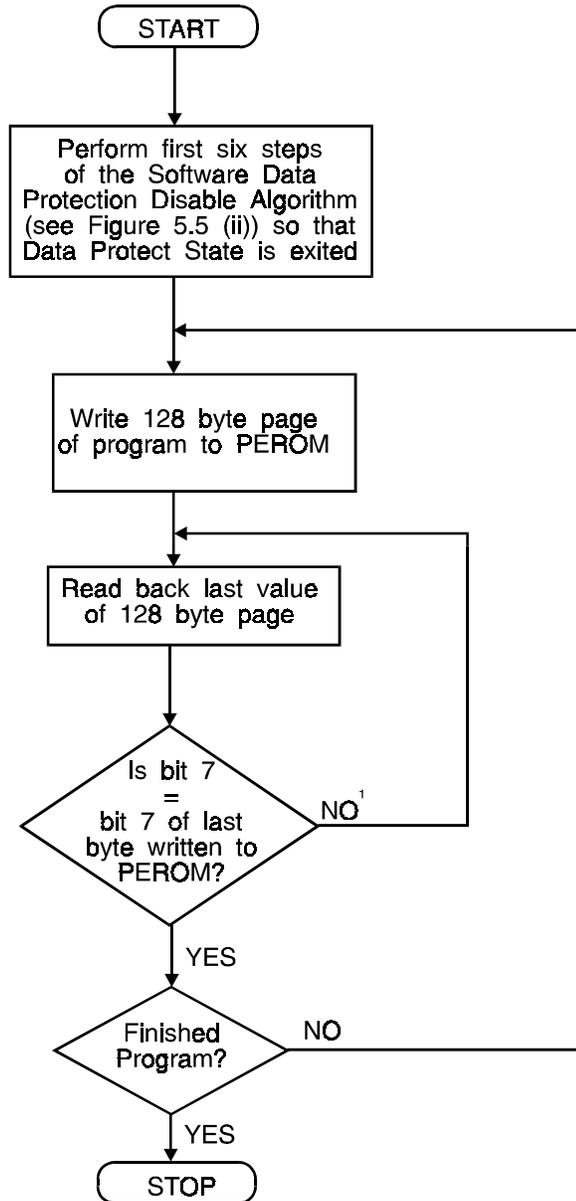


Figure 5.4: Flow Chart For Programming the Shared PEROM



¹This loop should take no longer than 10 ms.

5.7 Private Global Memory Resources

All shared bus masters have access to the private global memory resources of each processing node via the shared bus. Each node is mapped to a separate region in the shared bus memory map. The first address of each space relates directly to address *8000 0000h* in the processing node's global memory map. Where more than one node is indicated, these spaces are write only and provide a broadcast write facility. For example, a write to address *SBASE+03C0 0000h* from the VMEbus will write to address *8000 0000h* of all the processing nodes (A, B, C and D).

There are several restriction for access to the private global memory resources of each processing node:

- A processing node cannot access its own global memory via the shared bus.
- An attempt to access a global memory resource which is not available will result in a shared bus error.
- In order to access the on-module global memory of a TIM-40 module via the shared bus, this module must be a Blue Wave MDC40S TIM-40 module. Note that you must also set the STRB0 SWW bits of the C4x Global Memory Interface Control Register to '00' for external wait states, see [Section 4.2.3](#).
- Where more than one node is indicated, these spaces are write only.

The shared bus memory map for these regions is shown in [Figure 5.5](#) below. Note that the addresses given in the table below, refer to the address of each register as mapped to the respective shared bus master's memory map. For the VMEbus, SBASE refers to the VMEbus base address of DBV46, see [Chapter 6](#).

Figure 5.5: Private Global Memory Resources Memory Map

Longword Address		Byte Address
<i>BASE+0888 0000h</i>	TIM-40 A Global Memory	<i>SBASE+0220 0000h</i>
<i>BASE+0890 0000h</i>	C40 B Global Memory	<i>SBASE+0240 0000h</i>
<i>BASE+0898 0000h</i>	Nodes A and B Global Memory	<i>SBASE+0260 0000h</i>
<i>BASE+08A0 0000h</i>	C40 C Global Memory	<i>SBASE+0280 0000h</i>
<i>BASE+08A8 0000h</i>	Nodes A and C Global Memory	<i>SBASE+02A0 0000h</i>
<i>BASE+08B0 0000h</i>	Nodes B and C Global Memory	<i>SBASE+02C0 0000h</i>
<i>BASE+08B8 0000h</i>	Nodes A, B and C Global Memory	<i>SBASE+02E0 0000h</i>
<i>BASE+08C0 0000h</i>	TIM-40 D Global Memory	<i>SBASE+0300 0000h</i>
<i>BASE+08C8 0000h</i>	Nodes A and D Global Memory	<i>SBASE+0320 0000h</i>
<i>BASE+08D0 0000h</i>	Nodes B and D Global Memory	<i>SBASE+0340 0000h</i>
<i>BASE+08D8 0000h</i>	Nodes A, B and D Global Memory	<i>SBASE+0360 0000h</i>
<i>BASE+08E0 0000h</i>	Nodes C and D Global Memory	<i>SBASE+0380 0000h</i>
<i>BASE+08E8 0000h</i>	Nodes A, C and D Global Memory	<i>SBASE+03A0 0000h</i>
<i>BASE+08F0 0000h</i>	Nodes B, C and D Global Memory	<i>SBASE+03C0 0000h</i>
<i>BASE+08F8 0000h</i>	Nodes A, B, C and D Global Memory	<i>SBASE+03E0 0000h</i>

BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map.
 Processing Nodes BASE = *8000 0000h*
 PIM-3 Interface Module Specific
 VMEbus Interface SBASE = *1000 0000h* (A32 default)



Note that not all address spaces are accessible by all shared bus masters.

5.8 Shared Bus Arbitration

The DBV46 shared bus can only be accessed by one master at any one time. Therefore, to prevent contention between the shared bus masters, an arbitration system is employed. Although the actual operation of this arbitration system is transparent to you, the following subsections detail the arbitration and bus locking mechanism in order for your application software to make the best use of the shared bus.

5.8.1 Arbitration

The arbiter recognises when the shared bus is being accessed and accepts requests to read from or write to a shared bus resource from any master. The arbiter will grant shared bus access to a master when a request is made, once the current access is completed. However, there is an order of preference which dictates which master is granted access permission, if access to the shared bus is requested by more than one master.

The shared bus is shared on round robin arbitration between the VMEbus, PIM-3 interface, TIM-40 Site A, C40 B, C40 C and TIM-40 Site D, in that order. At reset, access defaults to the VMEbus. Therefore, for a master to gain bus ownership, it must be the next master in the chain to have asserted a bus request signal. Once bus ownership has been granted, the master will maintain ownership until the current access is complete AND another request is made by a different master. For example:

- C40 C has use of the bus.
- C40 B makes a request for the bus.
- TIM-40 Site D makes a request for the bus.
- When C40 C completes its access it releases the bus.
- The bus is then passed to Site D even though C40 B made a request before Site D.
- The bus will pass to Site B when Site D has completed its access UNLESS the VMEbus, PIM-3 interface or TIM-40 Site A request the bus in the meantime.



The signals requesting and granting the use of the bus to specific masters are built into the design of the arbitration system. You will not need to consider these signals when writing your software.

5.8.2 Timeout

DBV46 provides a timeout facility on the shared bus for all the processing nodes and the PIM-3 interface. When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μ s. The timeout duration is set via bit 7 (TOUT_DUR) of the DBV46 System Control Register ('0' - 50 μ s (default), '1' - 800 μ s). Note that a DBV46 shared bus timeout is not provided for VMEbus slave accesses to DBV46. In this case the standard VME timeout facility should be used.

A timeout interrupt to each of the processing nodes is also provided on DBV46. This interrupt is configured via the TOUT_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). The use and configuration of interrupts is described further in [Chapter 10](#). If a timeout error occurs the DBV46 Timeout Error Register can be read to determine which shared bus master caused a timeout error.

Bit 1 (TOUT_RDY) of the DBV46 System Control Register provides an additional timeout facility that can be used during debug. If this bit is set to '1', whenever a timeout occurs a slave ready signal will not be asserted to the shared bus master. This means that when debugging C4x code, the processor will hang at the instruction which caused the timeout. In general, this bit should be set to '0' (default) at all other times.

Note that DBV46 also allows you to disable the shared bus timeout facility altogether via bit 4 (TOUT_DIS) of the DBV46 System Control Register ('0' - Enabled (default), '1' - Disabled). In normal operation it is recommended that timeout is always enabled.

5.8.3 Bus Locking

The shared bus can be locked for exclusive use by a single bus master in a number of ways. These are listed and described below:

- **Time Lock**

The shared bus time lock facility provides three levels of bus locking and is configured via bits 5 and 6 (TLCK_DIS and TLCK_DUR) of the DBV46 System Control Register. These bits allow the bus to be locked for up to 1, 8 or 16 complete accesses once shared bus ownership is granted. Once configured, time lock is valid for all bus masters. The bit settings are shown below and the System Control Register is described in detail in [Section 5.2.2](#).

Bit 6 (TLCK_DUR)	Bit 5 (TLCK_DIS)	Description
0	0	Once shared bus ownership is granted, the bus will be locked for 8 complete accesses.
1	0	Once shared bus ownership is granted, the bus will be locked for 16 complete accesses.
0	1	Shared bus time lock disabled. Once shared bus ownership is granted, the bus will be locked for one complete access only.
1	1	

- **Bus Lock**

The bus lock facility allows the current shared bus master to lock ownership of the bus for exclusive use. This is configured via bit 0 (SBUS_LOCK) of the DBV46 Shared Bus Lock Register ('0' - Not Locked, '1' - Locked). Once this bit has been set, the shared bus will be locked for exclusive use by the current bus master (the device writing to the Shared Bus Lock Register) until SBUS_LOCK is cleared to '0'. Note that the bus lock facility can be abused to the detriment of the other bus masters and should be used with care. The Shared Bus Lock Register is described in detail in [Section 5.2.7](#).

Bits 2 to 7 of the Shared Bus Lock Register provide an additional facility that can be used during debug. These read only bits show which shared bus master has locked or unlocked the shared bus. For example, if a bus master locks the shared bus via SBUS_LOCK and a timeout occurs during any subsequent accesses to the shared bus, the current bus master is removed from the bus. This allows any bus master to access the Shared Bus Lock Register and determine which bus master, if any, has locked the shared bus. A bit set to '1' indicates that the bus has been locked by the corresponding master. Note that to remove the lock, only the bus master indicated by bits 2 to 7 can remove the lock. This master will 'reinherit' the bus lock.

- **Interlock Instructions**

A C4x processor can lock the shared bus for exclusive use by asserting a /LOCK signal on the arbiter. This is imposed in your software by using an interlock instruction. Table 6-3 in Texas Instruments' TMS320C4x User's Guide provides a summary of the assembly language instructions that can be used to assert and clear interlocks.

When a /LOCK is imposed, any requests for access to the shared bus will not be acted upon until the /LOCK is released. As with the bus lock facility, the use of interlock instructions can be abused to the detriment of the other bus masters and should be used with care.

Note that, if the shared bus is not locked, it will still remain granted to its user until it is requested by another source.

5.8.4 Deadlocks

Deadlocks are a potential problem with any multi-master processor system. A deadlock will occur when a processing node has acquired the shared bus and tries to access the VMEbus, and at the same time an external VME master board has acquired the VMEbus and attempts to access a DBV46 shared bus address.

Whenever a deadlock occurs, DBV46 automatically returns the shared bus to the VMEbus. The VMEbus completes its cycle and then DBV46 returns the bus to the normal round robin arbitration scheme, where the DSP cycle can be retried.

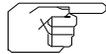
An interrupt to the DSP can be generated when a deadlock occurs, configured via the DLK_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). The use and configuration of interrupts is described further in [Chapter 10](#).

6 VMEbus Slave Interface

DBV46 provides a slave interface to the VMEbus that complies with the VME64 Specification. This interface provides access to the following features via the DBV46 shared bus:

- A control/status interface, which allows you to access and monitor various features of the board, such as global reset.
- An on-board Test Bus Controller (TBC), which can be used to control the TMS320C4x JTAG-based scan path circuitry.
- Shared DRAM which provides a data transfer system to/from the processing nodes.
- Shared PEROM which can be used to store common bootcode or embedded application code.
- The private global memory resources of each processing node, which also provides a broadcast write facility.

Blue Wave's development support packages for DBV46 operate using the JTAG emulation system and the shared memory data transfer mechanism.



If you have ordered a DBV46 development support package, you do not need to access the VMEbus interface directly because Blue Wave's interface library software can be used instead. The library software allows simple and easy access to the board's features from the VMEbus.

6.1 VME Slave Access Configuration

The VME interface is controlled by the Newbridge Microsystems SCV64 device. The SCV64 registers and register bits used to configure the DBV46 VME slave interface are detailed in this chapter.

Note that the PEROM for C40 B contains code that will automatically load default values into the SCV64 registers at boot-up. These values will correctly configure the VME memory map for your board. If you wish to set up a multi-DBV46 system, you must reprogram the PEROM on all but one board. The default VME memory map needs to be changed on these boards to prevent contention. The PEROM programming tools and bootcode generator, provided with all DBV46 software development packages, can be used to change the default setting of the SCV64 register set to configure the DBV46 VME slave interface as required. [Section 5.4](#) of this User Guide provides the default settings of the SCV64 registers affected by the C40 B bootcode and Appendix A of the SCV64 User Manual describes the register set in detail.

DBV46 can be mapped into the A64, A32 and A24 VME address spaces and provides access to the DBV46 shared bus as shown in [Figure 6.1](#).

- **A24 Space**

The base address and size of DBV46 in the A24 VME space is set via the SCV64 VME Base Address Register (VMEBAR). At boot up the base address is set to 800000h by default, but the A24 space is disabled (the A24SLVEN bit of the SCV64 MODE Register is set to '0'). This is because the size of DBV46 is limited to 4M bytes in A24 space which will only allow access to the TBC Register Set, DBV46 Control Registers and the SCV64 Register Set. Therefore, unless you only wish to perform control/status functions to DBV46, it is recommended that A32 space is used in preference to A24.

- **A32 Space**

The base address and size of DBV46 in the A32 VME space is set via the SCV64 VME Base Address Register (VMEBAR). At boot up, the base address is set to 1000 0000h and the size is set to 64M bytes for boards with no shared DRAM (DBV46D0) or 128M bytes for boards with shared DRAM (DBV46D1 and DBV46D2) as default.

- **A64 Space**

The base address of DBV46 in the A64 VME space is set via the SCV64 Slave A64 Base Address Register (SA64BAR) and is enabled by writing to the Master A64 Base Address Register (MA64BAR). However, at boot up the A64 space is disabled by default (neither register is configured).

Figure 6.1: DBV46 VMEbus Slave Memory Map

	VME Byte Address SBASE+	
TBC Register Set	0000 0000h	
DBV46 Control Registers	0010 0000h	
SCV64 Register Set	0020 0000h	
Reserved	0030 0000h	
A32 Space - Reserved	003F FFFCh	
¹ A24 Space - Location Monitor	003F FFFFh	
Reserved	0040 0000h	
No Access	0080 0000h	
Reserved	00C0 0000h	
DBV46 Interrupt Registers	0100 0000h	
Shared PEROM (128K x 8)	0140 0000h	
Reserved	0180 0000h	
PIM Interrupt Space	01C0 0000h	
Reserved	0200 0000h	
TIM-40 A Global Memory	0220 0000h	
C40 B Global Memory	0240 0000h	
Nodes A and B Global Memory	0260 0000h	
C40 C Global Memory	0280 0000h	
Nodes A and C Global Memory	02A0 0000h	
Nodes B and C Global Memory	02C0 0000h	
Nodes A, B and C Global Memory	02E0 0000h	
TIM-40 D Global Memory	0300 0000h	
Nodes A and D Global Memory	0320 0000h	
Nodes B and D Global Memory	0340 0000h	
Nodes A, B and D Global Memory	0360 0000h	
Nodes C and D Global Memory	0380 0000h	
Nodes A, C and D Global Memory	03A0 0000h	
Nodes B, C and D Global Memory	03C0 0000h	
² Nodes A, B, C and D Global Memory	03E0 0000h	
³ Shared DRAM	0400 0000h 07FF FFFFh	

SBASE refers to the VMEbus base address of DBV46.

A32 - 1000 0000h (default)

A24 - 800000h (default)

¹ This is the default location for the location monitor in A24 space.

² Note that for DBV46D0, the default location for the location monitor in A32 space occupies SBASE+03FF FFFCh to SBASE+03FF FFFFh.

³ Note that for DBV46D1 and DBV46D2, the default location for the location monitor in A32 space occupies SBASE+07FF FFFCh to SBASE+07FF FFFFh.

An attempt to access the global memory of a processing node that does not exist will result in a shared bus error and VME bus error.

6.1.1 Supported Transfers

DBV46 supports A64:D64 transfers and slave block transfers. These can be performed in coupled, decoupled and for block transfers, burst mode. The RXATOM and FIFOBEN bits of the SCV64 MODE Register are used to set the mode required to access a DBV46 shared bus resource. These modes and register bits are listed and described below.

- **Coupled Mode**

Coupled mode is the conventional VMEbus cycle. A coupled transfer will lock ownership of all three buses (master local bus, VMEbus and DBV46 shared bus) for the complete access. Reads from DBV46 are always performed in coupled mode. Writes to DBV46 can be performed in coupled or decoupled mode.

- **Decoupled Mode**

Writes to DBV46 can be performed in decoupled mode. This mode allows bus cycles to occur on the shared bus and VMEbus independently which will improve the data transfer rate.

- **Burst Mode**

When performing slave block transfers to DBV46, burst mode can be used to further improve the data transfer rate on the DBV46 shared bus. In burst mode, multiple blocks of data are transferred with only one address cycle.

Table 6.1: SCV64 MODE Register - RXATOM and FIFOBEN

Bit	Description
RXATOM (Bit 12)	Used to configure for coupled or decoupled writes. 0 - Decoupled Mode (default) 1 - Coupled Mode
FIFOBEN (Bit 15)	Used to configure for burst mode block transfers. 0 - Burst Mode Disabled 1 - Burst Mode Enabled (default)

Note that if you require large blocks of data to be read from DBV46, it is recommended that you use the SCV64 DMA controller. The VME board requesting the data can program the DMA controller to act as the VME master, reading data from the shared bus resource and writing that data to the target board. This will improve the data transfer rate from DBV46. Blue Wave's software support packages make use of the SCV64 DMA controller in this way. The use of the DMA controller is described in detail in the SCV64 User Manual.

The supported transfers for each of the shared bus resources are listed and described below.

- **DBV46 Control/Status Registers**

D32 transfers to the DBV46 register set are supported. Slave block transfers are also supported. Sections 5.2 and 5.3 detail each of these registers. Note that writes to these registers should be performed in coupled mode.

- **SCV64 Register Set**

D32 transfers to the SCV64 register set are supported. Slave block transfers are also supported. Section 5.4 details each of these registers. Certain SCV64 registers are also described in this chapter. Note that writes to these registers should be performed in coupled mode.

- **TBC Register Set**

D32 and D16 transfers to the TBC register set are supported. Slave block transfers are also supported. Section 5.5 and Chapter 11 detail the use of these registers. Note that writes to these registers should be performed in coupled mode.

- **Shared DRAM**

D64, D32, D24, D16 and D8 transfers to the shared DRAM are supported. Slave block transfers are also supported. Shared DRAM is described in Section 5.6.

- **Shared PEROM**

D32 transfers to the shared PEROM are supported. Slave block transfers are also supported. Shared PEROM is described in Section 5.6.

- **Private Global Memory Resources**

D64 and D32 transfers to the private global memory resources of each processing node are supported. Slave block transfers are also supported. Access to the private global memory resources via the shared bus is described in Section 5.7.

6.2 The SCV64 Location Monitor

A single 32 bit location in the DBV46 VME slave memory map is used to provide the SCV64 location monitor. The location monitor allows for inter-process communication between the VMEbus and the processing nodes and is fully described in the SCV64 User Manual. This section provides information specific to the use of the location monitor on DBV46.

The location monitor occupies the last four byte locations in the DBV46 VME slave memory map. For DBV46D0, the default location for the location monitor in A32 space occupies SBASE+03FF FFFCh to SBASE+03FF FFFFh. For DBV46D1 and DBV46D2, the default location for the location monitor in A32 space occupies SBASE+07FF FFFCh to SBASE+07FF FFFFh. The default location for the location monitor in A24 space occupies SBASE+003F FFFCh to SBASE+003F FFFFh for all DBV46 variants.

A D32 write performed to the location monitor will load the SCV64 location monitor FIFO (LMFIFO) with the 32 bit value. The FIFO can take up to 31 values. The oldest entry in the LMFIFO is reflected in the SCV64 LMFIFO Register. This register can then be read to retrieve the contents of the LMFIFO.

A location monitor interrupt to each of the processing nodes is also provided on DBV46. This interrupt is generated when the LMFIFO contains at least one entry. The interrupt condition is also flagged in the DBV46 Interrupt Status Registers and the SCV64 Control and Status Register (DCSR). The location monitor interrupt is configured via the LM_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). The use and configuration of interrupts is described further in [Chapter 10](#).

Any values written to/read from the location monitor are completely application dependent. In general, the location monitor can be used as a software signalling system between the VMEbus host and DBV46.

Note that Blue Wave's DBV46 interface libraries use the location monitor after C40 bootup to signal C40 B to perform certain initialisation routines. Therefore, in general, if you wish to use a Blue Wave software support package do not use the SCV64 location monitor. If you wish to make use of the SCV64 location monitor from your host or DSP code, your software support package User Guide will describe how to do this.

6.3 VME Interrupts to DBV46

DBV46 provides for interrupts from the VMEbus to each of the processing nodes. These are configured via the VME_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). On receipt of a VMEbus interrupt, you will need to determine the interrupt level, retrieve the VME Status/ID vector and initiate a VME IACK cycle. This process is detailed in the step by step procedure below which also include the configuration of the DBV46 and SCV64 registers for VMEbus interrupts to the processing nodes. The configuration of interrupts is also described in [Chapter 10](#).

Step 1 *Configure for VMEbus interrupts*

- Set bit 2 (VME_INT) of a DBV46 Interrupt Enable Register to '1'. This will enable interrupts from the VMEbus to the appropriate IIOF line.
- Set the appropriate bits of the SCV64 VME Interrupt Enable Register (VIE) to '1' to enable receipt of one or more VME interrupt levels from the VMEbus (bit 0 - Level 1, bit 6 - Level 7).

Step 2 *Configure interrupts on the C4x processor*

Configure the C4x processor to receive level-triggered interrupts on the appropriate IIOF line and enable interrupts on this line. This process is detailed in Section 6.5 of TMS320C4x User's Guide and is also covered in [Chapter 10](#) of this TRM.

Step 3 *On receipt of a VMEbus interrupt*

If a single IIOF line has been configured via its DBV46 Interrupt Enable Register to receive interrupts from a number of sources, the interrupt source can be determined by simply reading the DBV46 Interrupt Pending Register for the appropriate IIOF line, when an interrupt is received on that line. If the source is identified as an interrupt from the VMEbus (VME_INT bits > 0), your DSP interrupt service routine should perform the following:

- Set bits 12 to 14 (KFC0 - KFC2) of the DBV46 System Control Register to '1'.
- Set bit 8 (KADDR0) of the DBV46 System Control Register to '1'. Set bit 9 (KADDR1) of the DBV46 System Control Register to the least significant bit of the VME interrupt level received (bit 0 of the DBV46 Interrupt Pending Register).
- Read the appropriate location in the VME IACK space of the shared bus for the interrupt level received. The shared bus memory map is shown in [Figure 6.2](#) below.

The read from the IACK space will initiate a VME IACK cycle on the VMEbus and return the VME Status/ID vector. Your interrupt service routine can now service the interrupt.

Figure 6.2: VME IACK Space

**C4x Longword
Address**

8820 0000h	Not Used
8823 BFFFh	Level 1 IACK Space
8823 C000h	Level 2 and Level 3 IACK Space
8823 C001h	Level 4 and Level 5 IACK Space
8823 C002h	Level 6 and Level 7 IACK Space
8823 C003h	Reflections of the above IACK Spaces
8823 C004h	
8823 FFFFh	
8824 0000h	Not Used
8827 BFFFh	Level 1 IACK Space
8827 C000h	Level 2 and Level 3 IACK Space
8827 C001h	Level 4 and Level 5 IACK Space
8827 C002h	Level 6 and Level 7 IACK Space
8827 C003h	Reflections of the above IACK Spaces
8827 C004h	
8827 FFFFh	
8828 0000h	Not Used
882B BFFFh	Level 1 IACK Space
882B C000h	Level 2 and Level 3 IACK Space
882B C001h	Level 4 and Level 5 IACK Space
882B C002h	Level 6 and Level 7 IACK Space
882B C003h	Reflections of the above IACK Spaces
882B C004h	
882B FFFFh	
882C 0000h	Not Used
882F BFFFh	Level 1 IACK Space
882F C000h	Level 2 and Level 3 IACK Space
882F C001h	Level 4 and Level 5 IACK Space
882F C002h	Level 6 and Level 7 IACK Space
882F C003h	Reflections of the above IACK Spaces
882F C004h	
882F FFFFh	
8830 0000h	Reserved
8840 0000h	DBV46 Interrupt Registers
8850 0000h	Shared PEROM (128K x 8)
8860 0000h	Reserved
8870 0000h	VME IACK Space (Same as 8820 0000h to 882F FFFFh)
887F FFFFh	

Note that no access is provided to the VME IACK space from the VMEbus. An attempt to access this space from the VMEbus will result in a VME bus error.

The VME IACK space is read only. A VME bus error will occur if a processor attempts a write to the VME IACK space.

A VME bus error will also occur if a processor attempts a read from the VME IACK space when no VMEbus interrupt is pending.

Note that access to the VME IACK space is also possible from a PIM module, see [Section 9.1.2](#).

6.4 Access Protection

The SCV64 device can be programmed to prevent access to the DBV46 slave image from the VMEbus. Access protection is enabled via bit 1 of the SCV64 MODE Register. The address space to protect is set via the SCV64 Access Protect Boundary Register (APBR). To set access protection simply set the PROT bit to the type of protection required and set APBR to the size of address space you wish to protect. If access protection is not required (default), APBR should be set to zero.

The PROT bit and APBR registers are shown in [Table 6.2](#) below. These registers and access protection are described further in Section 2.7 of the SCV64 User Manual.

Table 6.2: SCV64 MODE and APBR Registers

Register	Bit	Description
MODE	1 (PROT)	Used to set the access protection type. 0 - Write protection only. 1 - Read and Write protection.
APBR	3 to 0 (APB03 to 00)	These bits are used to set the access protection boundary or size. The settings are shown below. Setting Boundary 0h 64K bytes (below SBASE+0001 0000h) 1h 28K bytes (below SBASE+0002 0000h) 2h 256K bytes (below SBASE+0004 0000h) 3h 512K bytes (below SBASE+0008 0000h) 4h 1M byte (below SBASE+0010 0000h) 5h 2M byte (below SBASE+0020 0000h) 6h 4M byte (below SBASE+0040 0000h) 7h 8M byte (below SBASE+0080 0000h) 8h 16M byte (below SBASE+0100 0000h) 9h 32M byte (below SBASE+0200 0000h) Ah 64M byte (below SBASE+0300 0000h) Bh 128M byte (below SBASE+0800 0000h) Ch 128M byte (below SBASE+0800 0000h) Dh 128M byte (below SBASE+0800 0000h) Eh 128M byte (below SBASE+0800 0000h) Fh 128M byte (below SBASE+0800 0000h)

7 VMEbus Master Interface

DBV46 provides a master interface to the VMEbus that complies with the VME64 Specification. The VMEbus master memory map is mapped to the DBV46 shared bus providing all processing nodes with shared master access to the VMEbus (assuming that the TIM-40 modules have access to the shared bus).

The ability to act as VMEbus master allows DBV46 to control and communicate with any compliant VME board, such as Blue Wave's DBV44 board which is a slave VME board with four TIM-40 sites. DBV46 can also act as the Slot 1 controller for your VME system, this is discussed separately in [Section 7.5](#).

Note that master access to the VMEbus is also possible from a PIM module, see [Section 9.1.2](#).

7.1 Overview

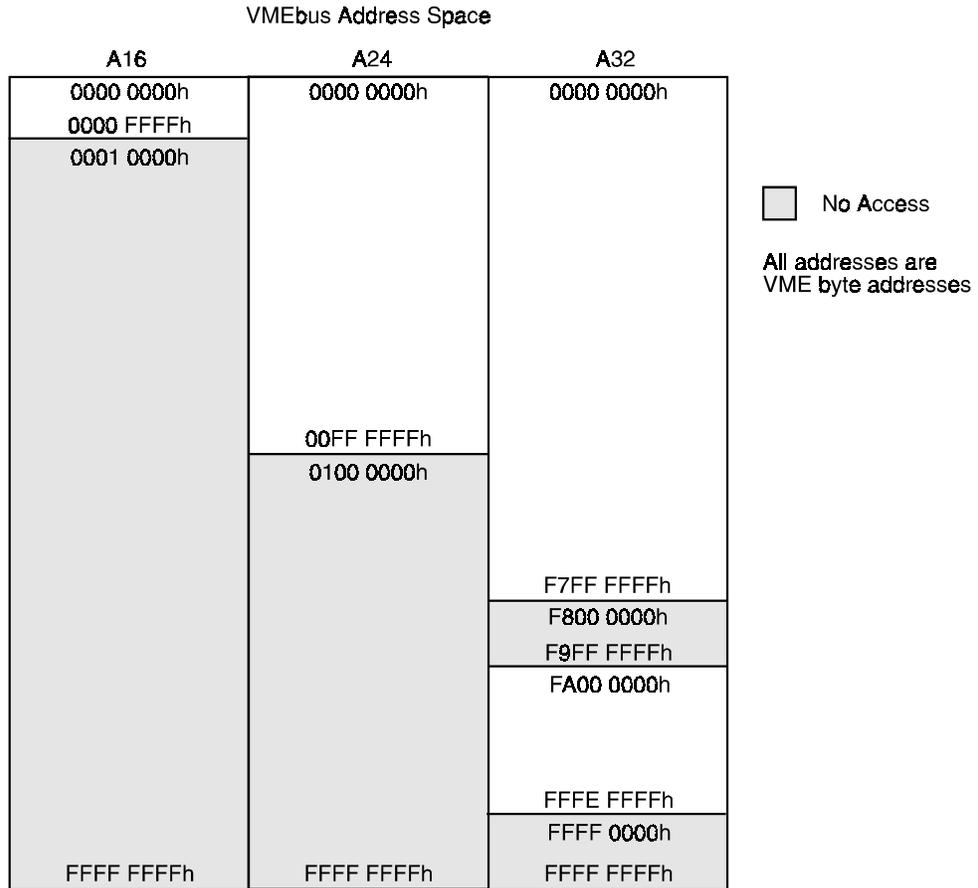
The SCV64 device is used by DBV46 to implement the VME master interface. The SCV64 maps the VMEbus master memory map to the shared bus. A processing node can access the VMEbus by reading and writing to the region `C000 0000h` to `FFFF FFFFh` in its global memory map, see [Sections 3.2](#) and [4.2](#).

Each VME address space (A16, A24, A32) is accessed via a different DSP address range. The default DBV46 VMEbus master memory map is shown in [Figure 7.1](#) below. Note that each of these address spaces can be reconfigured via the SCV64 MODE Register. This register is described in detail in the SCV64 User Manual.

DBV46 is also capable of performing A64:D64 master accesses and master block transfers. These are all performed using the SCV64 DMA controller. The use of the DMA controller to perform these types of accesses is described in detail in the SCV64 User Manual. Note that Blue Wave's software support packages make use of the SCV64 DMA controller. To avoid any conflicts in its use, please refer to the User Guide accompanying your DBV46 software support package.

The following sections describe how to perform single VME A16, A24 and A32 master accesses.

Figure 7.1: Default DBV46 VMEbus Master Memory Map



DSP Address Range (longwords)	VME Address Space	VMEbus Address Range (bytes)	SCV64 Address Offset (bytes)	SCV64 Address Range (bytes)
C000 0000h - FDFE FFFFh	VME A32 Space (D32)	0000 0000h - F7FF FFFFh	0000 0000h	0000 0000h - F7FF FFFFh
FE00 0000h - FE3F FFFFh	VME A24 Space (D16)	0000 0000h - 00FF FFFFh	F800 0000h	F800 0000h - F8FF FFFFh
FE40 0000h - FE7F FFFFh	VME A24 Space (D32)	0000 0000h - 00FF FFFFh	F900 0000h	F900 0000h - F9FF FFFFh
FE80 0000h - FFFF BFFFh	VME A32 Space (D32)	FA00 0000h - FFFE FFFFh	FA00 0000h	FA00 0000h - FFFE FFFFh
FFFF C000 - FFFF FFFFh	VME A16 Space (D16)	0000 0000h - 0000 FFFFh	FFFF 0000h	FFFF 0000h - FFFF FFFFh

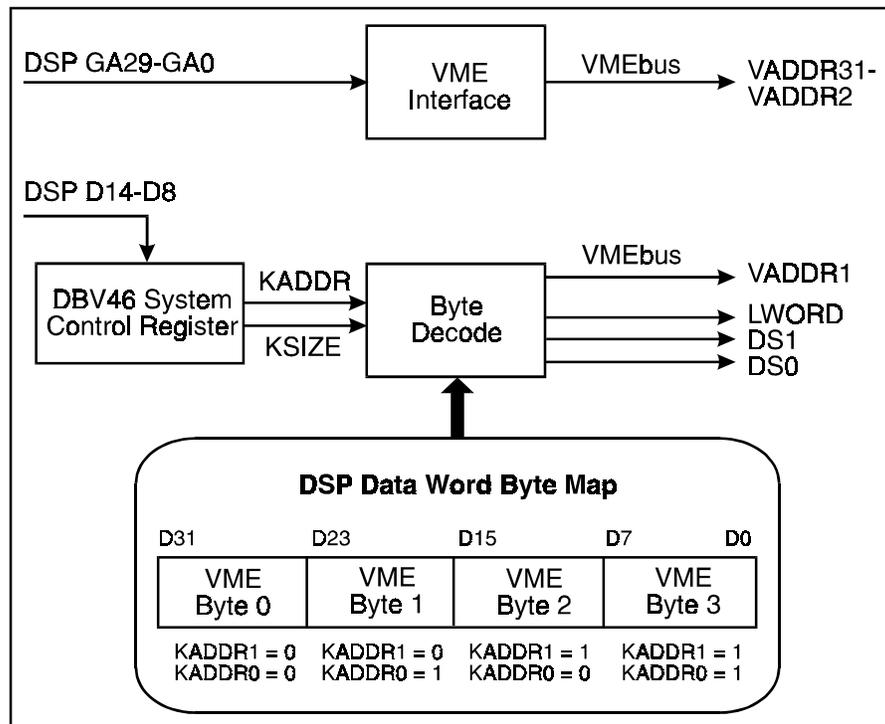
The DSP longword address required to access a VME byte location can be calculated from the following formula:

$$\text{DSP address} = ((\text{VME address} + \text{SCV64 Offset}) / 4) + \text{C000 0000h}$$

Accesses to the A16, A24 and A32 regions will map the access onto the VMEbus, with the VME address being placed on the VMEbus by SCV64. The TMS320C4x global address lines GA0 to GA29 are directly connected to SCV64 local address lines KADDR2 to KADDR31, which are then buffered onto the VME address lines VADDR2 to VADDR31. This is illustrated in [Figure 7.2](#). Before accessing the VMEbus, one of the processing nodes must set up the cycle type via the DBV46 System Control Register, this is explained in [Section 7.2](#).

The DBV46 System Control Register is used to set the SCV64 lower byte address lines (KADDR1 and KADDR0) at the start of the VME master cycle which will define the starting byte of the transfer. The number of bytes to be transferred is also set in this register via the KSIZE bits. The KADDR and KSIZE bits are decoded to set the appropriate VME address and strobe lines. The settings for each data cycle are detailed in [Section 7.2](#).

Figure 7.2: DSP and VME Master Interconnection



7.2 DBV46 System Control Register

The upper byte of this register must be written to define the VME bus master cycle. Once written the register contents will remain valid for all subsequent VME cycles. This register can be accessed on the shared bus at the DSP global address *8804 0001h*, see [Section 5.2](#).

D15				D8			
Reserved	KFC2	KFC1	KFC0	KSIZE1	KSIZE0	KADDR1	KADDR0

Bits 14 to 12 determine the type of VME cycle in terms of user/supervisor program/data for A16, A24 and A32 transfers. They are directly connected to the local SCV64 signals, KFC2, KFC1 and KFC0. The setting of these bits also determines the address modifier output on the VMEbus during the cycle, see [Section 7.3](#).

Bit 14	Bit 13	Bit 12	Function
KFC2	KFC1	KFC0	
0	0	0	User Program access
0	0	1	User Data access
0	1	0	User Program access
0	1	1	User Data access
1	0	0	Supervisor Program access
1	0	1	Supervisor Data access
1	1	0	Supervisor Program access
1	1	1	Supervisor Data access

Bits 11 and 10 indicate the number of bytes to be transferred in the cycle:

Bit 11	Bit 10	Function
KSIZE1	KSIZE0	
0	0	longword cycle
0	1	byte cycle
1	0	word cycle
1	1	three byte cycle

Bits 9 and 8 set the value of KADDR1 and KADDR0 respectively at the start of the cycle (see [Figure 7.2](#)):

Bit 9	Bit 8	Function
KADDR1	KADDR0	
0	0	VME Byte 0 (D31-D24)
0	1	VME Byte 1 (D23-D16)
1	0	VME Byte 2 (D15-D8)
1	1	VME Byte 3 (D7-D0)

7.3 Setting Address Modifiers on VME Master Cycles

Transfer of data is initiated by a VMEbus master. A DBV46 master controls the type of transfer and provides the address modifiers for the transfer via the DBV46 System Control Register and the VME address space being accessed. The address modifiers allow a slave board to monitor this information so that they can determine which address lines to decode. Note that not all slave boards respond to all address modifier codes.

Table 7.1: VME Master Address Modifiers

VME Address Space	KFC2	KFC1	KFC0	Cycle Type	SCV64 Address Modifiers VAM5-0
A32 VME addressing	0	0	0	User Program access	0Ah
	0	0	1	User Data access	09h
	0	1	0	User Program access	0Ah
	0	1	1	User Data access	09h
	1	0	0	Supervisor Program access	0Eh
	1	0	1	Supervisor Data access	0Dh
	1	1	0	Supervisor Program access	0Eh
	1	1	1	Supervisor Data access	0Dh
A24 VME addressing	0	0	0	User Program access	3Ah
	0	0	1	User Data access	39h
	0	1	0	User Program access	3Ah
	0	1	1	User Data access	39h
	1	0	0	Supervisor Program access	3Eh
	1	0	1	Supervisor Data access	3Dh
	1	1	0	Supervisor Program access	3Eh
	1	1	1	Supervisor Data access	3Dh
A16 VME addressing	0	x	x	User access	29h
	1	x	x	Supervisor access	2Dh

x = '0' or '1'

Note also that the default configuration of SCV64 on DBV46 does not allow the production of user-defined address modifier codes.

Listing 7.1 provides some general example code for performing VME master read and write cycles.

Listing 7.1: VME Master Read and Write Cycles

```
/* DBV46 VME Bus Master Access Example Code */
/* NOTE : This code assumes the SCV64 registers are in their default state */

/* Define types required for VME bus access */

#define ULONG unsigned long
#define UBYTE unsigned char
enum Data_Type {BYTE,WORD,TRI_BYTE,QUAD_BYTE};
enum Addr_Modifier_Type {NO_PRIV_PROG,NO_PRIV_DATA,SUPER_PROG,SUPER_DATA};
enum VME_Access_Type {A32D32_LO,A32D32_HI,A24D16,A24D32,A16D16};

/* Define masks for the VME bus accesses */

/* Non VME control byte in System Control Register */
#define NON_VME_MASK(ULONG)0x000000FF
/* Lowest two bits of desired VME bus address give KAddr settings */
#define LOW_ADDRESS_MASK(UBYTE)0x03
/* Define SCV64 VME Access offsets */
#define A32D32_LO_OFFSET(ULONG)0x00000000
#define A32D32_HI_OFFSET(ULONG)0xFA000000
#define A24D16_OFFSET(ULONG)0xF8000000
#define A24D32_OFFSET(ULONG)0xF9000000
#define A16D16_OFFSET(ULONG)0xFFFF0000
/* Define DBV46 Shared Bus Addresses ( C40 perspective ) */

#define VME_BASE_OFFSET(ULONG)0xC0000000
#define SYS_CON_REG(ULONG)0x88040001

/* Define read & write macros for VME bus accesses */

#define Read(Addr)*(volatile ULONG *) (Addr)
#define Write(Addr,Data)*(volatile ULONG *) (Addr) = (Data)

/* *****/
/* Calc_VME_Access_Value */
/*
/* Constructs a control byte which can be used to program the System */
/* Control Register on the DBV46 to perform a VME access of the desired */
/* data transfer size from a given address offset using a given Address */
/* Modifier code on the VME bus. */
/* The constructed control byte is returned by the function. */
/* *****/

UBYTE Calc_VME_Access_Value(enum Data_Type Data_Size,
    UBYTE Addr_Offset,
    enum Addr_Modifier_Type Addr_Modifier)

{ /* Calc_VME_Access_Value */

    UBYTE VME_Access_Value; /* Used to construct control byte */
    UBYTE KAddr; /* Used to construct address offset */
    UBYTE KSize; /* Used to construct data transfer size */
    UBYTE AM_Code; /* Used to construct address modifier code */

    KAddr = Addr_Offset; /* Determine byte offset of address */
```

Listing 7.1: Continued

```
switch(Data_Size)/* Determine data transfer size required */
{ /* KSize switch */
  case BYTE :
    KSize = 1;
    break;
  case WORD :
    KSize = 2;
    break;
  case TRI_BYTE :
    KSize = 3;
    break;
  case QUAD_BYTE :
    KSize = 0;
    break;
  default :
    KSize = 0;
    break;
} /* KSize switch */
switch(Addr_Modifier)/* Determine Address Modifier code required */
{ /* AM_Code switch */
  case NO_PRIV_PROG :
    AM_Code = 0;
    break;
  case NO_PRIV_DATA :
    AM_Code = 1;
    break;
  case SUPER_PROG :
    AM_Code = 4;
    break;
  case SUPER_DATA :
    AM_Code = 5;
    break;
  default :
    AM_Code = 1;
    break;
} /* AM_Code switch */

/* Construct control byte */
VME_Access_Value = KAddr + (KSize << 2) + (AM_Code << 4);
return(VME_Access_Value);

} /* Calc_VME_Access_Value */
```

Listing 7.1: Continued

```
/* *****/
/* Calc_VME_Access_Space */
/*
/* Returns the SCV64 offset ( pre adjusted ) to provide access to the */
/* desired VME bus address space */
/* *****/

ULONG Calc_VME_Access_Space(enum VME_Access_Type VME_Space)

{ /* Calc_VME_Access_Space */

    ULONG VME_Access_Space; /* Used to construct control byte */

    switch(VME_Space) /* Determine data transfer size required */
    { /* VME_Space switch */
        case A32D32_LO :
            VME_Access_Space = A32D32_LO_OFFSET;
            break;
        case A32D32_HI :
            VME_Access_Space = A32D32_HI_OFFSET;
            break;
        case A24D16 :
            VME_Access_Space = A24D16_OFFSET;
            break;
        case A24D32 :
            VME_Access_Space = A24D32_OFFSET;
            break;
        case A16D16 :
            VME_Access_Space = A16D16_OFFSET;
            break;
        default :
            VME_Access_Space = A32D32_LO_OFFSET;
            break;
    } /* VME_Space switch */

    /* Adjust the offset to account for Shared Bus / VME bus addressing */
    VME_Access_Space = VME_Access_Space >> 2;
    return(VME_Access_Space);

} /* Calc_VME_Access_Space */
```

Listing 7.1: Continued

```
/* Read_VME */
/*
/* Reads Data_Size bytes of data from the specified VME address using an
/* access with the given Address Modifier
/* The data read is returned by the function
/* VME_Address = Byte address of VME bus location to read ( ULONG )
/* Data_Size = The number of bytes to read ( enumerated type )
/* Addr_Modifier = Address Modifier to use ( enumerated type )
/*
/* (NOTE: D16 and D8 reads from the VME address will determine which
/* part of the returned value contains data. For example, a read from
/* address 0x800000 of D16 data would result in the data being returned
/* in the top 8 bits of the ULONG value and the data being returned from
/* address 0x800001 in bits 23 to 16. A read from address 0x800002
/* would result in bits 15-8 of the returned value containing the data
/* at this address and bits 7-0 the data from address 0x800003.)
/*
ULONG Read_VME(ULONG VME_Address,
               enum Data_Type Data_Size,
               enum VME_Access_Type VME_Space,
               enum Addr_Modifier_Type Addr_Modifier)

{ /* Read_VME */
  ULONGOriginal_Sys_Con_Reg;
  ULONGNew_Sys_Con_Reg;
  ULONGVME_Read_Data;
  ULONGModified_Address;
  UBYTELow_Address;
  ULONGSCV64_Offset;

  /* Extract the lowest two bits to provide the byte offset for the access */
  Low_Address = (UBYTE)VME_Address & LOW_ADDRESS_MASK;

  /* Modify the VME address to account for Shared Bus / VME bus correction */
  Modified_Address = VME_Address >> 2;

  /* Determine SCV64 Offset for VME address space required */
  SCV64_Offset = Calc_VME_Access_Space(VME_Space);

  /* Read the original System Control Register data */
  Original_Sys_Con_Reg = Read(SYS_CON_REG);

  /* Construct new System Control Register data */
  New_Sys_Con_Reg = Original_Sys_Con_Reg & NON_VME_MASK;
  New_Sys_Con_Reg |=
    (Calc_VME_Access_Value(Data_Size,Low_Address,Addr_Modifier) << 8);

  /* Write to the System Control Register to set the VME bus access type */
  Write(SYS_CON_REG, New_Sys_Con_Reg);

  /* Perform the VME read */
  VME_Read_Data = Read(VME_BASE_OFFSET + SCV64_Offset + Modified_Address);

  /* Restore original System Control Register data */
  Write(SYS_CON_REG, Original_Sys_Con_Reg);

  return(VME_Read_Data);
} /* Read_VME */
```

Listing 7.1: Continued

```
/* Write_VME */
/*
/* Writes Data_Size bytes of data to the specified VME address using an
/* access with the given Address Modifier
/* VME_Address = Byte address of VME bus location to read ( ULONG )
/* Write_Data = Data to be written to VME bus ( ULONG )
/*
/* (NOTE: For D16 and D8 accesses the VME address will determine which
/* part of the ULONG word is written. For example, a write to address
/* 0x800000 of D16 data would result in the top 8 bits of Write_Data
/* being written to 0x800000 and bits 23 to 16 being written to
/* 0x800001. A write to 0x800002 would result in bits 15-8 of
/* Write_Data being written to 0x800002 and bits 7-0 to 0x800003.)
/*
/* Data_Size = The number of bytes to read ( enumerated type )
/* Addr_Modifier = Address Modifier to use ( enumerated type )
*/

void Write_VME(ULONG VME_Address,
               ULONG Write_Data,
               enum Data_Type Data_Size,
               enum VME_Access_Type VME_Space,
               enum Addr_Modifier_Type Addr_Modifier)

{ /* Write_VME */

    ULONGOriginal_Sys_Con_Reg;
    ULONGNew_Sys_Con_Reg;
    ULONGModified_Address;
    UBYTELow_Address;
    ULONGSCV64_Offset;

    /* Extract the lowest two bits to provide the byte offset for the access */
    Low_Address = (UBYTE)VME_Address & LOW_ADDRESS_MASK;

    /* Modify the VME address to account for Shared Bus / VME bus correction */
    Modified_Address = VME_Address >> 2 ;

    /* Determine SCV64 Offset for VME address space required */
    SCV64_Offset = Calc_VME_Access_Space(VME_Space);

    /* Read the original system Control Register data */
    Original_Sys_Con_Reg = Read(SYS_CON_REG);

    /* Construct new System Control Register data */
    New_Sys_Con_Reg = Original_Sys_Con_Reg & NON_VME_MASK;
    New_Sys_Con_Reg |=
        (Calc_VME_Access_Value(Data_Size,Low_Address,Addr_Modifier) << 8);

    /* Write to the System Control Register to set the VME bus access type */
    Write(SYS_CON_REG, New_Sys_Con_Reg);

    /* Perform the VME write */
    Write(VME_BASE_OFFSET + SCV64_Offset + Modified_Address,Write_Data);

    /* Restore original System Control Register data */
    Write(SYS_CON_REG, Original_Sys_Con_Reg);
} /* Write_VME */
```

7.4 Interrupts

7.4.1 Interrupts to the VMEbus

The SCV64 device can be programmed to generate interrupts to the VMEbus on any one of the seven VME interrupt levels. The interrupt level and interrupt initiation are both controlled by the VMEbus Interrupter Requester Register (VINT). The Status/ID vector is set via the SCV64 VMEbus Interrupter Vector Register (IVECT). These registers are shown in [Table 7.2](#) below.

Table 7.2: SCV64 VINT and IVECT Registers

Register	Bit	Description
VINT	2 to 0 (IL2 to IL0)	Used to set the VMEbus interrupt level.
		IL2 IL1 IL0 Interrupt Level
		0 0 0 No Interrupt
		0 0 1 Level 1
		0 1 0 Level 2
		0 1 1 Level 3
		1 0 0 Level 4
		1 0 1 Level 5
		1 1 0 Level 6
1 1 1 Level 7		
	3 (INT)	Used to initiate the VME interrupt and reflect the interrupt status. Write 0 - No effect. Write 1 - Initiate VME interrupt. Read 0 - No interrupt pending. Read 1 - VME interrupt asserted.
IVECT	0 to 7	These bits are used to provide the Status/ID vector.

To generate an interrupt to the VMEbus, simply load the Status/ID vector into the IVECT Register and then set bits 2 to 0 (IL2 to IL0) of the VINT Register to the VME interrupt level required. To initiate the VME interrupt, set bit 3 (INT) to '1'. When the SCV64 detects a VME IACK cycle, the 8 bit Status/ID vector is placed on the VMEbus data lines VDATA7 to VDATA0. Bit 3 (INT) of the VINT register is cleared to zero on completion of the VME interrupt cycle.

The VINT and IVECT Registers and interrupts to the VMEbus are described further in Section 2.3 of the SCV64 User Manual.

7.4.2 SCV64 Interrupts

An SCV64 interrupt to each of the processing nodes is provided on DBV46. The SCV64 interrupt is configured via the SCV_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). Note that the specific SCV64 interrupt condition is flagged in bits 1, 2, 3, 5 and 16 of the SCV64 Control and Status Register (DCSR), see Appendix A of the SCV64 User Manual. The use and configuration of interrupts is described further in [Chapter 10](#).

7.4.3 VME Bus Error

DBV46 provides a VME bus error interrupt to all processing nodes. This interrupt is configured via the VBER_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#), and is generated if the VME /BERR line is asserted during a coupled master access to the VMEbus. Note that SCV_INT will also be asserted and the SCV64 VBERR interrupt must be cleared by setting the VBERR bit of the SCV64 Control and Status Register (DCSR) to '0'. The VME master interface is disabled until this operation has been performed. The use and configuration of interrupts is described further in [Chapter 10](#).

Note that if you are performing decoupled master accesses to the VMEbus, a VME bus error will not affect the DBV46 VBER_INT bits. It will however still assert SCV_INT and again you must clear the SCV64 VBERR interrupt by setting the VBERR bit of the SCV64 Control and Status Register (DCSR) to '0'. The VME master interface is disabled until this operation has been performed.

7.5 System Controller Facilities

The DBV46 can act as the system controller for your VME system. This is achieved by simply installing DBV46 in Slot 1 of your VME system. The SCV64 automatically detects Slot 1 installation and enables its system controller functions. Detailed information on the SCV64's system controller functions is provided in Section 2.5 of the SCV64 User Manual.



Your system must provide one (and only one) system controller.

8 Parallel Communication Ports

DBV46's parallel processor system can support optimum performance by distributing tasks between two or more processors. High performance multi-processing requires rapid transfer of data between processors. The DBV46 board achieves this by utilising the high speed parallel communication ports of the C4x processors.

This system has the advantage over shared memory that arbitration is performed internal to the processors and therefore does not incur extra cycles which result in a reduced processor communication bandwidth.

The key features of each C4x communication port include:

- Peak 20M bytes/s (50 MHz C4x) bidirectional data transfer operations (24M bytes/s for a 60 MHz C4x),
- Direct processor-to-processor communication,
- FIFO buffering of all data transfers, both input and output,
- Automatic arbitration and handshaking to ensure communication synchronisation,
- Synchronisation between the CPU or Direct Memory Access (DMA) coprocessor and the six ports via internal interrupts and internal ready signals,
- Support of a wide variety of multi-processor architectures, including rings, trees, hypercubes, bidirectional pipelines, two-dimensional Euclidean and three-dimensional grids.

Each processing node on DBV46 provides six communication ports, which are numbered 0 to 5. All six ports available from each processing node can be used to provide rapid processor-to-processor communication.

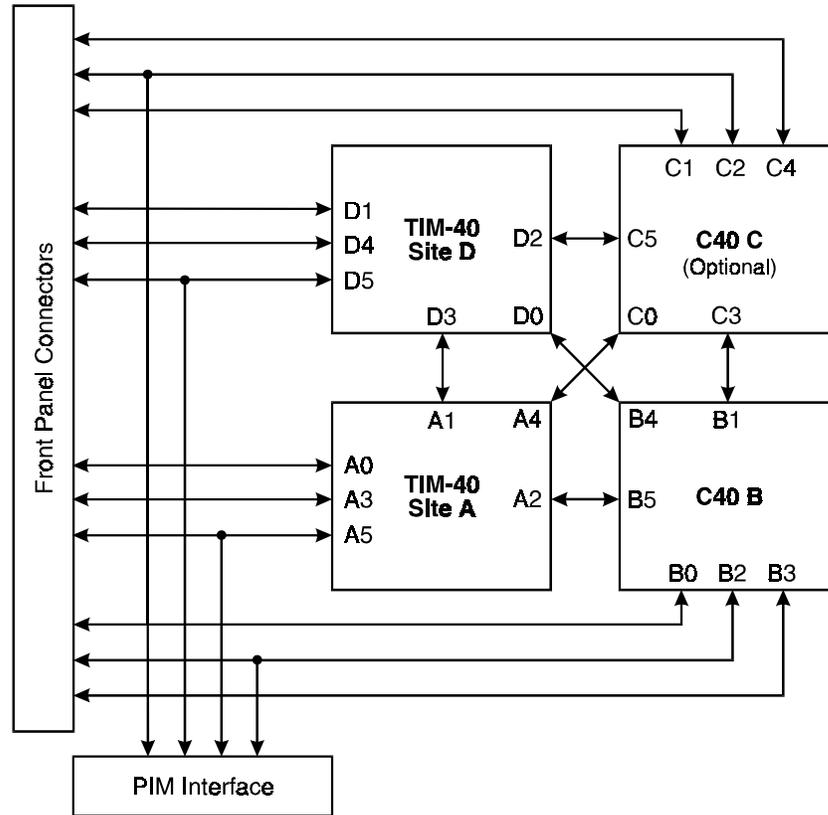
Two ports from each node are brought out to front panel connectors so that the user can interconnect ports on and off-board. Another port from each node can be routed to either the front panel or the PIM-3 interface under software control, see [Section 8.1](#). The three remaining ports from each node are interconnected to provide a dedicated communication link between nodes.

The interprocessor and off-board routing of the parallel communication ports is illustrated in [Figure 8.1](#) below.



If you have purchased the single processor variant of the board, the C40 C processor is NOT provided. All references to the parallel communication ports to/from this processor can simply be ignored.

Figure 8.1: On and Off-Board Routing of Communication Ports



Ports A5, B2, C2 and D5 are brought out to front panel connectors OR routed to the PIM-3 interface. The routing depends upon the configuration of the DBV46 Communication Port Control Register.



Note that none of the communication ports are buffered on DBV46. This means that the bidirectional data transfer rate is not reduced in any way.

8.1 Port Configuration

DBV46 provides a very flexible system of port interconnection, so that you can create a wide range of multi-processing configurations in multi-board systems:

- Three ports from each processing node are interconnected to provide a dedicated communication port link between nodes.
- Two ports from each processing node are brought out to connectors on the front panel of the DBV46.
- Another port from each node can be routed to either the front panel or the PIM-3 interface under software control as described below.

You can configure the routing of Ports A5, B2, C2 and D5 via the DBV46 Communication Port Control Register. This DBV46 Control Register can be accessed via the shared bus at address *BASE+0804 0003h* (DSP address) or *SBASE+0010 000Ch* (VME address). The individual bit functions are described in [Table 8.1](#) below. At power-up/reset all ports default to the front panel.

Table 8.1: Communication Port Control Register

		D31 - D8	D7 - D6	D5 - D4	D3 - D2	D1 - D0
		Not Used	D5_ROUTE	C2_ROUTE	B2_ROUTE	A5_ROUTE

Bit	Name	Function		
1 - 0	A5_ROUTE	These pairs of bits select the routing of communication ports A5, B2, C2 and D5. The individual bit settings are shown below.		
3 - 2	B2_ROUTE			
5 - 4	C2_ROUTE	MSB	LSB	Function
		0	0	Not used - must not be set to '00'.
7 - 6	D5_ROUTE	0	1	Port routed to PIM site.
		1	0	Port routed to front panel (default).
		1	1	Port disabled.



To avoid the danger of port contention and/or data corruption, the pairs of bits in the Communication Port Control Register must not be set to '00'.



More than one communication port can be routed to the PIM-3 interface.

The use of a communication port by the PIM-3 interface is described in [Chapter 9](#).

8.1.1 Front Panel Communication Port Connectors

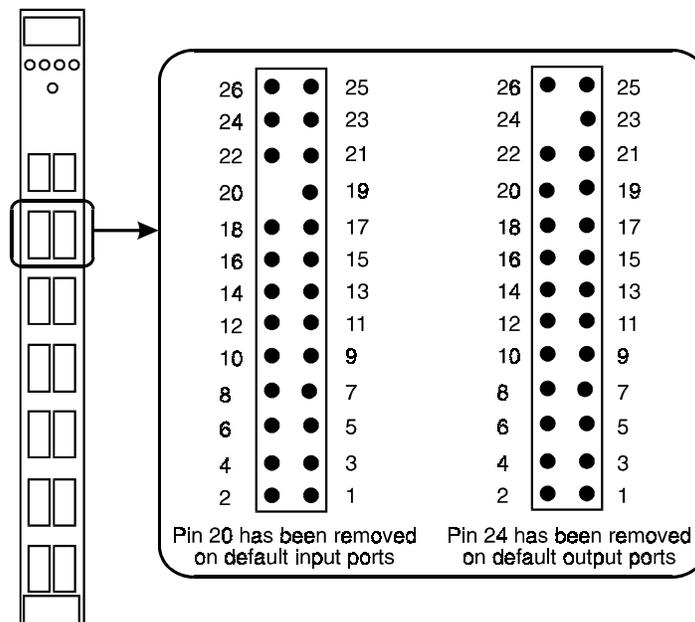
Each communication port is implemented using twelve signals. These signals are brought out to a 26-way high density unshrouded connector on DBV46's front panel, see [Figure 2.1](#) of [Chapter 2](#). The pinout of the communication port connectors is given in [Table 8.2](#) and their orientation is illustrated in [Figure 8.2](#).

Table 8.2: Front Panel Port Connector Pinout

Pin	Name	Pin	Name
1	CD0_x	2	GND
3	CD1_x	4	GND
5	CD2_x	6	GND
7	CD3_x	8	GND
9	CD4_x	10	GND
11	CD5_x	12	GND
13	CD6_x	14	GND
15	CD7_x	16	GND
17	/CREQ_x	18	GND
19	/CACK_x	20	GND or POL_IN
21	/CSTRB_x	22	GND
23	/CRDY_x	24	GND or POL_OUT
25	Not Connected	26	Not Connected

x = A0 for Port A0 front panel connector etc. POL_IN and POL_OUT are polarising pins for input and output ports respectively.

Figure 8.2: Front Panel Communication Port Connector Orientation



After reset of a C4x processor, Ports 0, 1 and 2 default to outputs and 3, 4 and 5 to inputs. **It is important that only default input ports are connected to default output ports to prevent damage to the C4x processors.** From [Figure 2.1](#), you can see that the ports on the left hand side default to inputs and those on the right hand side default to outputs. To prevent possible damage due to briefly misaligning the connectors, interconnection of C4x ports must be performed with the power switched off.



You must not connect an output port to another output (or input to input) as this may damage the C4x processor.



For multi-processor TIM-40 modules and/or modules that contain TMS320C44 processors, you should refer to your TIM-40 module User Manual to check which processor ports provide the individual module ports.

In order to reduce the chance of incorrect connection of communication port cables, polarising pins have been cut on the input and output connectors and blanking plugs similarly inserted into the supplied cables. Pin 24 has been removed on ports that default to output after reset, and pin 20 on ports that default to input. The supplied communication port cables will only fit in one direction due to the polarisation pins, and the headers only fit in one orientation. This helps to prevent illegal interconnection between communications ports. Cables should be labelled 'IN' and 'OUT' at either end so that easy identification of the orientation of the cable is possible.

The DBV46 is supplied with six high density 50mil cables. Ribbon cable lengths of up to 45 cm are possible. Any surplus of cable should be loosely folded.



If you are making your own cable you should include the polarisation pins to prevent illegal interconnection of the communication ports.

The provided cables have latching connectors (from AMP part number 111196-6). These secure the cable connections to the front panel. The latching tool supplied with DBV46 should be used when removing the connectors from the front panel to prevent damage to the pins.

It must be stressed that the direction of the port is only a default after reset. Each port is bidirectional and subsequent data transfer over the port may occur in either direction. This means it is important that all boards connected via C4x ports are reset and released from reset simultaneously, this is discussed in [Section 8.1.2](#) below.



To avoid the danger of port contention and data corruption, do not connect any front panel C4x communication ports whilst your VME system is switched on. Note that DBV46 is shipped with protective covers on all front panel connectors. These should be left in place on unused connectors.

8.1.2 Setting Up a Multi-Board System

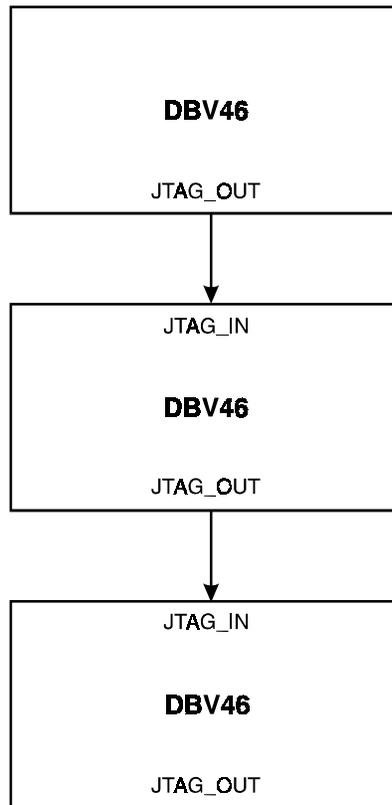
The communication ports can be connected across boards to create a wide variety of multi-processor architectures. The boards should be placed in adjacent VME slots to keep interconnection lengths to a minimum.

As described above, the direction of each port is only a default after reset and so subsequent data transfer over the port may occur in either direction. Therefore, if two processors on separate boards are connected together via a communication port, if only one of these processors is reset it is possible to have two input or two output ports connected together. To avoid contention between two ports, you must ensure that all modules in the system are reset and released from reset simultaneously. **To do this in a multi-board system, all boards must also be connected with high density 20-way IDC ribbon cable via the JTAG headers.** The boards should be connected in a single chain with the JTAG_OUT header of the first board connected to the JTAG_IN header of the next and so on, see [Figure 8.3](#).

This routes the C4x global reset (/GRESET) and configuration (/CONFIG) signals to all boards in the system. To simultaneously reset all boards you simply write '1' to bit 0, BRD_RESET, of the DBV46 System Reset Register on any DBV46 board in the system. This will assert the /GRESET line and release the line once the reset operation is complete. This register is only accessible from the VMEbus, and is described in [Section 5.2](#) and [Chapter 12](#).

More information on the JTAG headers is given in [Chapter 11](#).

Figure 8.3: Multi-Board System



Note that the boards will also be connected via the communication ports but the topology of this will depend upon your particular application. The JTAG connection must always be made in a single chain, irrespective of the port topology.



Due to the limitations of JTAG itself, the recommended maximum length of a single JTAG scan chain is three carrier boards or ten processors, whichever is the greater. This will ensure the propagation of all signals through the JTAG scan chain.

8.2 Port to Port Communication

The following subsections illustrate how two C4x processors can communicate across the parallel port interconnections, using program or DMA controlled transfers.

8.2.1 Program Controlled Transfers

Consider two C4x processors, X and Y, where, for example, Port 0 of X is connected to Port 3 of Y. If Processor X wishes to write to Processor Y, X writes the block of memory to be transferred to Port X0's output FIFO in the local port memory map. This is transferred to Port Y3's input FIFO in the memory map (see Section 8.4 of Texas Instruments' TMS320C4x User's Guide). The block is then read by Processor Y.

Interrupts should be set up to establish when the buffers on both modules are empty and full via the C4x's *Internal Interrupt Enable (IIE) Register* on each C4x (see Section 3.1.9 of Texas Instruments' TMS320C4x User's Guide). Alternatively you can poll the C4x's *Communication Port Control Register* (see Section 8.4.1 of Texas Instruments' TMS320C4x User's Guide).

This process can function in the opposite direction allowing transfers to and from both modules. One drawback with this method of block transfer is that the processor's operation is disrupted because the CPU core has to read and store the memory blocks.

8.2.2 DMA Controlled Transfers

This system of memory block transfer is similar to program control except that neither processor is disrupted because the reading and storing of memory is handled by a DMA controller rather than the CPU core of the processors. This method is more complex to set up because of the number of registers that need to be configured. See Chapter 9 of Texas Instruments' TMS320C4x User's Guide for a full description of the DMA coprocessors. Each TMS320C4x processor has six DMA channels, one is used by each parallel port for transferring data. Therefore, if three ports on the one processor are connected to those on another, three times the data transfer speed of one port can be achieved, providing extremely fast inter-processor communication.

8.2.3 Bidirectional Communication

If the communication ports are being used for bidirectional transfers, it is possible to enter a lockout state whereby no further data exchange can take place until the relevant processors are reset. The port that is transmitting data holds a token. If the other port wishes to transmit data, it must request this token. A lockout will occur in some instances where one port requests the token while the other side is still attempting to transfer information.

For example, say two processors, X and Y, are exchanging information. The block of data being transferred by Processor X is larger than the input FIFO of the parallel communications ports. Processor X starts transferring a block to Processor Y. However, if Processor Y is not removing the data as quickly as it is being transmitted, the input FIFO of Processor Y will become full. Processor X is still trying to transfer the rest of the data, but it cannot until Processor Y removes some data from its input FIFO.

A lockout state occurs if Processor Y needs to transfer some data to X over the same ports before it can remove any information from its input FIFO. Processor Y requests the token from Processor X, but X is not able to grant it until it has finished sending its current word of data to Processor Y. This is impossible because Y's input FIFO will always be full.

The problem does not arise if different ports are used to transmit information from Y to X, but this means that a processor can only be connected to half as many processors. Lockouts can be avoided if your application software knows the size of blocks to be transmitted and synchronises their transfer. In the case of program controlled transfers (see [Section 8.2.1](#)) lockouts can also be avoided if the processor that requires to transmit information always checks that it has a partially empty input FIFO before requesting the token. This is done by polling bits 9 to 12 of the C4x processor's *Communication Port Control Register*.

DBV46 can also accommodate a Peripheral Interface Module (PIM) adjacent to TIM-40 Site A. DBV46 incorporates Blue Wave's standard PIM-3 interface which provides a general purpose expansion capability to DBV46. PIM-3 is an extension of Blue Wave's PIM interface and is compatible with all Blue Wave PIM modules.

A PIM module has up to three connectors:

- **PIM Global Connector**

This connector allows direct access to a PIM module from the shared bus.

- **PIM P2 Connector**

This connector allows direct access to/from the user-defined I/O pins of the VMEbus P2 connector.

- **PIM-3 Connector**

This connector allows direct access to the shared bus from a PIM module. This connector also allows bidirectional access to the four C4x parallel communication ports which can be optionally routed to the PIM site.

By implementing one or more of the above connectors, a PIM module can provide:

- Global memory expansion for a processing node.
- A C4x communication port link via P2.
- Access to/from a P2 I/O subsystem bus, such as the VME Subsystem Bus (VSB).
- Access to/from any custom interface, such as a data acquisition interface, either memory-mapped, via a communication port or via the VMEbus P2 connector.

The TIM-40 connectors for Site A and the adjacent PIM connectors are arranged so that DBV46 can support a double width PIM module that also incorporates all the facilities of a TIM-40 module.

Blue Wave's current range of PIM modules includes PIM-VSB2 which provides a memory-mapped interface from the processing nodes to the VME Subsystem Bus (VSB). The module provides a full master/slave VSB interface with on-board DPRAM (Dual-Port RAM) and DMA controller and can also act as the Slot 1 controller for your VSB system.

Each Blue Wave PIM module is documented in a separate User Manual. For information on Blue Wave's current range of PIM modules and the PIM-3 specification, please contact your distributor.

Table 9.1: PIM-3 Connector Pinout

Pin	Shared Bus Connection	Pin	Shared Bus Connection
1	GND	2	CD0_A5
3	CD1_A5	4	CD2_A5
5	CD3_A5	6	CD4_A5
7	CD5_A5	8	CD6_A5
9	CD7_A5	10	/CREQ_A5
11	/CACK_A5	12	/CSTRB_A5
13	/CRDY_A5	14	GND
15	CD0_B2	16	CD1_B2
17	CD2_B2	18	CD3_B2
19	CD4_B2	20	CD5_B2
21	CD6_B2	22	CD7_B2
23	/CREQ_B2	24	/CACK_B2
25	/CSTRB_B2	26	/CRDY_B2
27	GND	28	CD0_C2
29	CD1_C2	30	CD2_C2
31	CD3_C2	32	CD3_C2
33	CD5_C2	34	CD6_C2
35	CD7_C2	36	/CREQ_C2
37	/CACK_C2	38	/CSTRB_C2
39	/CRDY_C2	40	GND
41	CD0_D5	42	CD1_D5
43	CD2_D5	44	CD3_D5
45	CD4_D5	46	CD5_D5
47	CD6_D5	48	CD7_D5
49	/CREQ_D5	50	/CACK_D5
51	/CSTRB_D5	52	/CRDY_D5
53	GND	54	GA20
55	GA21	56	GA22
57	GA23	58	GA24
59	GA25	60	GA26
61	GA27	62	GA28
63	GA29	64	GA30
65	S_PAGE	66	/S_MAST_RDY
67	/S_LOCK	68	/S_BUS_GRNT
69	/S_BUS_REQ	70	/S_SLAVE_RDY
71	/P_BUS_REQ	72	+12 V
73	/P_BUS_GRNT	74	PIM_H1 (Clock)
75	DBV46_CLK (50 or 60 MHz)	76	-12 V
77	/XPIM_INT	78	/EXT_INT1
79	TCLK0	80	GND

Table 9.2: PIM P2 Connector Pinout Table 9.3: PIM Global Connector Pinout

Pin	VMEbus P2 Connection	Pin	VMEbus P2 Connection	Pin	Shared Bus Connection	Pin	Shared Bus Connection
1	GND	2	GND	1	GND	2	GND
3	P2A1	4	P2C1	3	PIMCLK	4	/RST
5	P2A2	6	P2C2	5	/XRDY	6	/EXT_INT0
7	P2A3	8	P2C3	7	/STRB1	8	R/W
9	P2A4	10	P2C4	9	/PIMCS	10	GND
11	P2A5	12	P2C5	11	GND	12	GA19
13	P2A6	14	P2C6	13	GA18	14	GA17
15	P2A7	16	P2C7	15	GA16	16	GA15
17	P2A8	18	P2C8	17	GA14	18	GA13
19	P2A9	20	P2C9	19	GA12	20	GA11
21	P2A10	22	P2C10	21	GA10	22	GA9
23	P2A11	24	P2C11	23	GA8	24	GA7
25	P2A12	26	P2C12	25	GA6	26	GA5
27	P2A13	28	P2C13	27	GA4	28	GA3
29	P2A14	30	P2C14	29	GA2	30	GA1
31	P2A15	32	P2C15	31	GA0	32	GD31
33	P2A16	34	P2C16	33	GD30	34	GD29
35	P2A17	36	P2C17	35	GD28	36	GD27
37	P2A18	38	P2C18	37	GD26	38	GD25
39	P2A19	40	P2C19	39	GD24	40	GD23
41	P2A20	42	P2C20	41	GD22	42	GD21
43	P2A21	44	P2C21	43	GD20	44	GD19
45	P2A22	46	P2C22	45	GD18	46	GD17
47	P2A23	48	P2C23	47	GD16	48	GD15
49	P2A24	50	P2C24	49	GD14	50	GD13
51	P2A25	52	P2C25	51	GD12	52	GD11
53	P2A26	54	P2C26	53	GD10	54	GD9
55	P2A27	56	P2C27	55	GD8	56	GD7
57	P2A28	58	P2C28	57	GD6	58	GD5
59	P2A29	60	P2C29	59	GD4	60	GD3
61	P2A30	62	P2C30	61	GD2	62	GD1
63	P2A31	64	P2C31	63	GD0	64	TCLK1
65	P2A32	66	P2C32	65	GND	66	GND
67	Vcc	68	Vcc	67	VCC	68	VCC

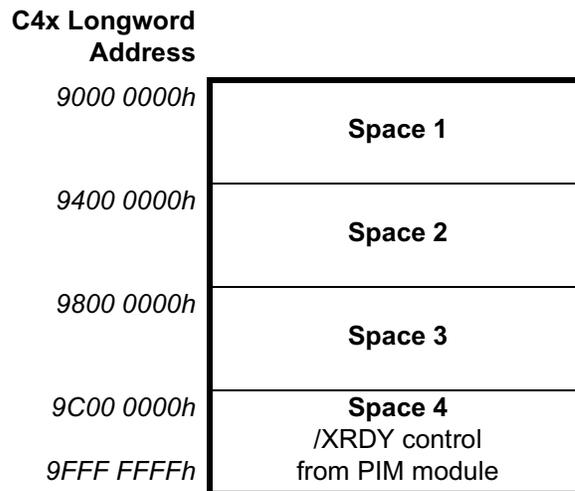
P2A1 = Pin 1 of Row A of VMEbus P2 connector

9.1 Shared Bus Access

9.1.1 Access to the PIM-3 Interface

The PIM-3 interface is mapped to the shared bus and is accessible by all processing nodes from *9000 0000h* to *9FFF FFFFh* in each node's global memory map, see Sections 3.3 and 4.2. An expanded map of the PIM-3 interface is given in Figure 9.1.

Figure 9.1: PIM-3 Interface Slave Memory Map



A PIM module can be accessed via Space 1, 2 or 3 in the PIM-3 interface slave memory map (Space 4 is discussed below). The space selected depends upon the speed of accesses required. Space 1 allows the fastest accesses, Space 3 the slowest. Accesses to Spaces 1 to 3 are controlled via a hardware wait state generator on DBV46.

The User Manual that accompanies your PIM module states the minimum access time that the master must provide. Table 9.4 summarises the timings for Space 1 to 3 for a 50 MHz and 60 MHz DBV46 board/C4x processor.

Table 9.4: PIM-3 Space Access Times

Processor/Board Speed	Space	Timings (ns)		Total Processor Cycles	
		Read	Write	Read	Write
50 MHz	Space 1	80	120	2 cycle transfer	3 cycle transfer
	Space 2	120	160	3 cycle transfer	4 cycle transfer
	Space 3	160	200	4 cycle transfer	5 cycle transfer
60 MHz	Space 1	66	100	2 cycle transfer	3 cycle transfer
	Space 2	100	133	3 cycle transfer	4 cycle transfer
	Space 3	133	166	4 cycle transfer	5 cycle transfer

1 cycle = 40 ns for a 50 MHz processor and 33.33 ns for a 60 MHz processor

In order to determine the fastest space, you must choose the space which provides timings greater than (or equal to) that required by the slave board. For example, a PIM module that states an access time requirement of 110 ns can be accessed via Space 2 of a 50 MHz DBV46 board for both read and write accesses and Space 1 for write accesses.

Alternatively, a PIM module can be accessed via Space 4. Here the access time is controlled via the /XRDY signal from the module. This allows you to run with slower access times than can be achieved via Spaces 1 to 3.

9.1.2 Access to Shared Bus Resources

Note that a PIM module can also implement its own processor, allowing the module to act as a shared bus master. The PIM-3 connector provides access to the DBV46 shared bus resources. Each of these resources is described in [Chapter 5](#) along with a full memory map. The PIM-3 master memory map is shown in [Figure 9.2](#) below.

Note that in order to access the shared bus from a PIM module, the processor/module must be using the DBV46 on-board clock as its clock source. Your module User Manual will detail how to configure your module to use the carrier board clock source.

As you can see from the PIM-3 master memory map, a PIM module can access the VMEbus as a VME master. However, note that VME interrupts (IRQ1 to IRQ7) to the PIM-3 interface are not directly supported and no access is provided to the PIM-3 interface from the VMEbus. If you wish to use a PIM module to perform VME master cycles or VME IACK cycles, the details provided in [Chapter 7](#) and [Section 6.3](#) are also applicable to a PIM module. When referring to this information, simply replace processing node addresses with PIM addresses.

Figure 9.2: PIM-3 Master Memory Map

**Longword Address
BASE+**

0800 0000h	No Access
0804 0000h	DBV46 Control Registers
0808 0000h	SCV64 Register Set
080C 0000h	Reserved
0820 0000h	VME IACK Space
0830 0000h	Reserved
0840 0000h	DBV46 Interrupt Registers
0850 0000h	Shared PEROM (128K x 8)
0860 0000h	Reserved
0870 0000h	No Access
0880 0000h	Reserved
0888 0000h	TIM-40 A Global Memory
0890 0000h	C40 B Global Memory
0898 0000h	Nodes A and B Global Memory
08A0 0000h	C40 C Global Memory
08A8 0000h	Nodes A and C Global Memory
08B0 0000h	Nodes B and C Global Memory
08B8 0000h	Nodes A, B and C Global Memory
08C0 0000h	TIM-40 D Global Memory
08C8 0000h	Nodes A and D Global Memory
08D0 0000h	Nodes B and D Global Memory
08D8 0000h	Nodes A, B and D Global Memory
08E0 0000h	Nodes C and D Global Memory
08E8 0000h	Nodes A, C and D Global Memory
08F0 0000h	Nodes B, C and D Global Memory
08F8 0000h	Nodes A, B, C and D Global Memory
0900 0000h 0BFF FFFFh	Shared DRAM
0C00 0000h	Reserved
1000 0000h	No Access
2000 0000h	Reserved
4000 0000h 7FFF FFFFh	VMEbus Address Space

BASE refers to the base address of the shared bus in the PIM module master memory map. This is module specific, refer to your PIM module User Manual.

An attempt to access the global memory of a processing node that does not exist will result in a shared bus error.

9.2 Interrupts

Interrupts from the PIM-3 interface to each of the processing nodes are also provided on DBV46. These interrupts are configured via the PIM_INT bits of the DBV46 Interrupt Registers, see [Section 5.3](#). The use and configuration of interrupts is described further in [Chapter 10](#). The PIM interrupts are provided via the /EXT_INT and /EXT_INT1 interrupt lines. Your PIM module User Manual will describe how these interrupt lines are utilised (if implemented). Note that interrupts from a PIM module must be level-triggered.

Interrupts TO the PIM-3 interface are also supported. A separate PIM Interrupt Space is mapped to the shared bus and is accessible by all shared bus masters. By performing a dummy write to any location within this space, an interrupt is generated to the PIM-3 interface via the /XPIM_INT line (40 ns interrupt pulse). The use of this interrupt line is purely application specific. Your PIM module User Manual will describe how this interrupt line is utilised (if implemented).

10 Interrupts

10.1 Interrupt Implementation

Three of the four IIOF lines (IIOF0, 1 and 2) provided by each processing node can be configured to receive or generate interrupts on DBV46. IIOF3 is used to control the /CONFIG line.

The source and direction of interrupts for all the processing nodes are configured via the DBV46 Interrupt Registers. Each 'Group' Register listed in the table below refers to the appropriate interrupt line (Group 0 - IIOF0, Group 1 - IIOF1, Group 2 - IIOF2) and each line is routed to all processing nodes.

The shared bus register map is shown in [Table 10.1](#) below and each is described in [Section 5.3](#). Note that the addresses given in the table below, refer to the address of each register as mapped to the respective shared bus master's memory map. BASE and SBASE refer to the base address of the shared bus in the relevant shared bus master memory map (Processing Nodes - BASE = 8000 0000h, PIM-3 Interface - Module Specific, VMEbus Interface - SBASE = 1000 0000h (A32 default)).

Table 10.1: DBV46 Interrupt Registers

DSP/PIM Address (longwords)	VMEbus Address (bytes)	Register
BASE+0840 0000h	SBASE+0100 0000h	Group 0 Interrupt Enable Register
BASE+0840 0001h	SBASE+0100 0004h	Group 1 Interrupt Enable Register
BASE+0840 0002h	SBASE+0100 0008h	Group 2 Interrupt Enable Register
BASE+0840 0003h	SBASE+0100 000Ch	Reserved
BASE+0840 0004h	SBASE+0100 0010h	Reserved
BASE+0840 0005h	SBASE+0100 0014h	Reserved
BASE+0840 0006h	SBASE+0100 0018h	Group 0 Interrupt Line Status Register
BASE+0840 0007h	SBASE+0100 001Ch	Group 1 Interrupt Line Status Register
BASE+0840 0008h	SBASE+0100 0020h	Group 2 Interrupt Line Status Register
BASE+0840 0009h	SBASE+0100 0024h	Group 0 Interrupt Condition Status Register
BASE+0840 000Ah	SBASE+0100 0028h	Group 1 Interrupt Condition Status Register
BASE+0840 000Bh	SBASE+0100 002Ch	Group 2 Interrupt Condition Status Register
BASE+0840 000Ch	SBASE+0100 0030h	Group 0 Interrupt Pending Register
BASE+0840 000Dh	SBASE+0100 0034h	Group 1 Interrupt Pending Register
BASE+0840 000Eh	SBASE+0100 0038h	Group 2 Interrupt Pending Register
BASE+0840 000Fh	SBASE+0100 003Ch	Reserved
BASE+0840 0010h to BASE+084F FFFFh	SBASE+0100 0040h to SBASE+013F FFFFh	Reflections of the above registers.

The DBV46 Interrupt Registers provide for nine types of interrupt to each of the processing nodes:

- **VME_INT**
Interrupts from the VMEbus (see [Section 6.3](#)).
- **VBER_INT**
VME bus error interrupt. This is generated if the VME /BERR line is asserted during a coupled master access to the VMEbus (see [Section 7.4.3](#)).
- **DLK_INT**
Deadlock interrupt. A deadlock will occur if the VMEbus is being accessed by a processing node and the shared bus is being accessed by the VMEbus at the same time (see [Section 5.8](#)).
- **SCV_INT**
Interrupts from the SCV64 device (see [Section 7.4.2](#)).
- **LM_INT**
SCV64 location monitor interrupt (see [Section 6.2](#)).
- **PIM0_INT and PIM1_INT**
Interrupts via the /EXT_INT and /EXT_INT1 interrupt lines from the PIM site (see [Section 9.2](#)).
- **TOUT_INT**
Shared bus timeout interrupt. When accessing a shared bus resource, a timeout will occur if a ready signal is not returned to the shared bus master within approximately 50 or 800 μ s (see [Section 5.8](#)).
- **IIOF_INT**
Allows a processing node to use its IIOF line as an output. This enables a node to generate an interrupt to one or more of the other processing nodes (see [Section 10.2](#)).



The special Blue Wave bootcode contained in the C40 B PEROM will configure the Interrupt Enable Registers to enable LM_INT on IIOF0 and SCV_INT on IIOF2. If you wish to use Blue Wave's software support package, you must not alter this configuration.

10.2 Interrupt Configuration

As described above, the source and direction of interrupts for all the processing nodes are configured via the DBV46 Interrupt Registers. The actual IIOF lines of a processing node are configured via the C4x processor *IIOF Flag Register (IIF)*, a summary of this register is given in Section 3.1.10 of Texas Instruments' TMS320C4x User's Guide. The following example step by step procedure describes how to configure these registers to provide for interrupts to/from the processing nodes.

Step 1 **Configure the DBV46 Interrupt Enable Registers**

Write a '1' to the appropriate bits (bits 2 to 11) of each Group Interrupt Enable Register to enable receipt of an interrupt from the corresponding source.

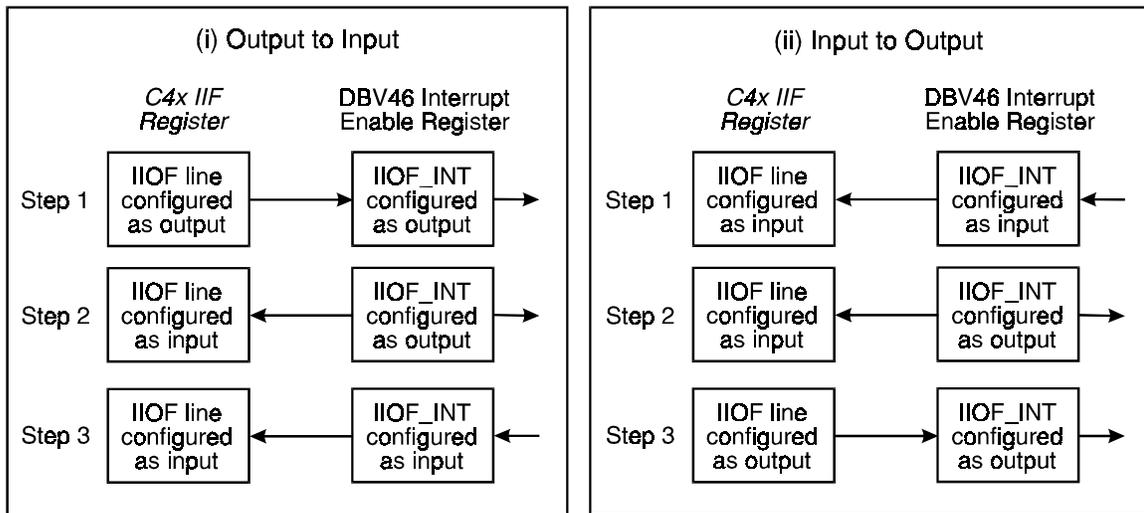
Configure bits 12 to 15 (A_IIOF_INT to D_IIOF_INT) to define the direction of the IIOF line for a particular node ('0' - Input to C4x, '1' - Output from C4x). As an input the processing node will receive interrupts from the sources set via bits 2 to 11. As an output, the processing node can generate an interrupt to the other nodes (assuming their IIOF_INT bits are set to '0').

! You must ensure that the setting of the DBV46 Interrupt Enable Register does not conflict with the setting of the internal C4x *IIOF Flag Register (IIF)*. If, for example, an interrupt line is configured via the DBV46 Interrupt Enable Register as an input to the processing node AND as an output via the C4x *IIF Register*, this may cause damage to certain components on DBV46.

To avoid the condition given in the warning above, when reconfiguring the interrupt lines, follow the points given below:

- If you have configured any of the interrupt lines as outputs from the C4x and then wish to use them as inputs, you must first reconfigure the interrupt lines via the node's internal C4x *IIF Register* before resetting the IIOF_INT bits of the DBV46 Interrupt Enable Register. This is illustrated in [Figure 10.1\(i\)](#) below.
- If you have configured any of the interrupt lines as inputs to the C4x and then wish to use them as outputs, you must first reset the IIOF_INT bits of the DBV46 Interrupt Enable Register before reconfiguring the interrupt lines via the node's internal C4x *IIF Register*. This is illustrated in [Figure 10.1\(ii\)](#) below.

Figure 10.1: IIOF line Configuration



Step 2 Configure the C4x IIOF Flag Register (IIF)

Write to the internal C4x *IIOF Flag Register (IIF)* of each processing node to configure the node's actual IIOF lines:

- If you wish to use an interrupt line as an input to the C4x to receive interrupts from the sources set via bits 2 to 11 of the DBV46 Interrupt Enable Register, you should configure the IIOF line as an interrupt line (FUNCx = '1') to receive level-triggered interrupts (TYPEx = '1').
- If you wish to use an interrupt line as an output from the C4x to generate interrupts to the other processing nodes, you should configure the IIOF line as a general purpose I/O line (FUNCx = '0') and as an output (TYPEx = '1').
- To enable the IIOF line, set the EIIOFx bit to '1'. If you do not wish a particular node to receive the interrupts configured on a particular line, you can simply disable the IIOF line by clearing EIIOFx to '0'.

Note that further C4x configuration is required to enable IIOF interrupts, see Section 6.5 of Texas Instruments' TMS320C4x User's Guide.

Step 3 On receipt of an interrupt

On receipt of an interrupt via an individual IIOF line, you can read the DBV46 Interrupt Pending Register to indicate the source of the interrupt. Clearing the interrupt depends on the source and is described in [Section 5.3.2](#). Note that all interrupts are also flagged in the DBV46 Interrupt Condition Status Register, see [Section 5.3.4](#).

11 JTAG Emulation System

The TBC on the DBV46 board provides essentially the same functionality as Blue Wave's XDSC40 or Texas Instruments' XDS510 TMS320C4x emulation system. It exercises the C4x's JTAG subsystem which is used to implement the DB40 debugger provided with Blue Wave's development support packages for DBV46. The TBC is controlled via a block of registers in the shared bus memory map accessible from the VMEbus, see Sections 5.6 and 6.1.



The DB40 debugger supplied with many DBV46 software support packages exercises the JTAG scan path. To ensure that the software reset bits, BRD_RESET and GRESET, operate correctly after using the debugger, you should set the processors running free before exiting DB40.

The following sections describe how to set up single and multiple board scan paths using either the on-board TBC or external TBC, such as XDSC40 or XDS510.

11.1 JTAG Emulation System Connectors

The DBV46 board has two JTAG connectors, JTAG_IN and JTAG_OUT, located on the board's front panel, see [Figure 2.1](#) in [Chapter 2](#). These can be used to route the JTAG scan path off-board to another DBV46 or to an external emulator. They have the same pinout as the XDS510 headers on the Texas Instruments' emulator, except that each has six extra pins for the C4x global reset and global configuration lines. The DBV46 JTAG_IN connector accepts signals from either the OUT connector of another DBV46 board or of another external emulator.

The pinouts of the JTAG connectors are given in [Figure 11.1](#). These are high density IDC 20-way connectors with pins 6 and 16 to 18 removed to provide a key for correct orientation.

Figure 11.1: JTAG_IN and JTAG_OUT Connector Pinouts

/GRESET	20	19	/CONFIG	/GRESET	20	19	/CONFIG
NC	18	17	NC	NC	18	17	NC
NC	16	15	XTBC_DETECT	NC	16	15	SENSE
EMU1	14	13	EMU0	EMU1	14	13	EMU0
GND	12	11	TCK	GND	12	11	TCK
GND	10	9	TCK_RET	GND	10	9	TCK_RET
GND	8	7	TDO	GND	8	7	TDI
NC	6	5	PD(+5V)	NC	6	5	SENSE
GND	4	3	TDI	GND	4	3	TDO
/TRST	2	1	TMS	/TRST	2	1	TMS
JTAG_IN				JTAG_OUT			

NC = Not Connected

11.2 Single Board Emulation System

In a single DBV46 board system you can either use the on-board TBC or an external emulator.

When using an external emulator, DBV46's JTAG_IN connector must be connected to the OUT connector on the emulator. DBV46 will automatically detect connection to the emulator and disable the TBC on DBV46. Similarly, the on-board TBC is automatically enabled if DBV46's JTAG_IN connector is not connected.

In the case of Blue Wave's XDSC40 emulator you can use the 20-way high density cable provided. However, the connectors on Texas Instruments' emulator are standard density, so you must use the specially adapted cable provided with DBV46 to connect the standard density OUT connector to the high density JTAG_IN connector.



You must ensure that DBV46 pin 1 is aligned to pin 1 of the Texas Instruments' connector, failure to do so will damage both the DBV46 board and the emulator.

11.3 Multi-Board Emulation System

In a multi-board system using one on-board TBC, the DBV46 JTAG connectors must be connected in a chain with the OUT connector of one board connected to the IN connector of the next board. The active on-board TBC will be at the top of the chain, see [Figure 11.2\(i\)](#).

Each board in the chain will detect that it is connected via its JTAG_IN connector and automatically disable its TBC, apart from the top board.

If you wish to use an external system, the OUT connector on the external emulator must be connected to the JTAG_IN connector of the first DBV46 board in the chain, see [Figure 11.2\(ii\)](#). Again, each board will detect that it is connected via its JTAG_IN connector and disable its TBC so that the emulator contains the only active TBC.



Due to the limitations of JTAG itself, the recommended maximum length of a single JTAG scan chain is three carrier boards or ten processors, whichever is the greater. This will ensure the propagation of all signals through the JTAG scan chain.

Figure 11.2(i):Multi-Board Emulation System Using On-Board TBC

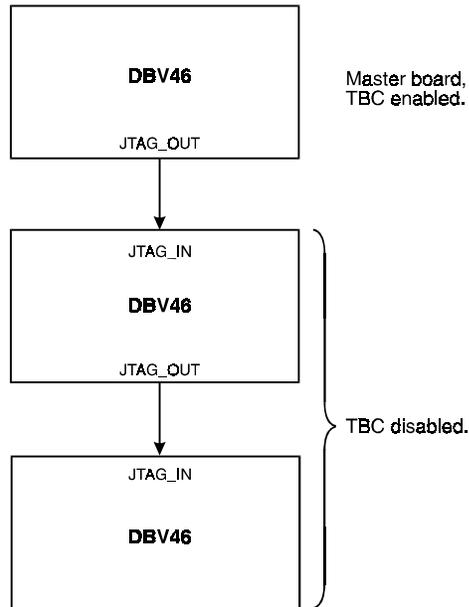
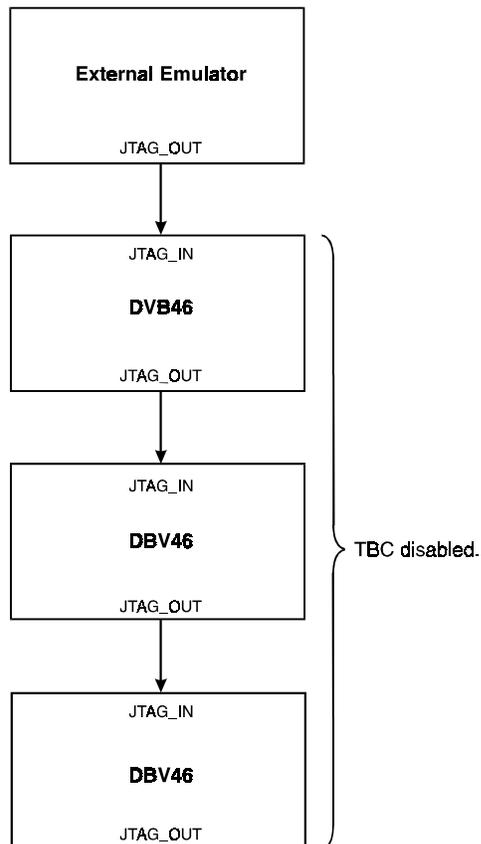


Figure 11.2(ii):Multi-Board Emulation System Using External Emulator



In a multi-board system, if you have connected any C4x communication ports between boards you must connect the JTAG connectors in a single chain, whether or not you intend to use the emulation system. This ensures that the /GRESET and /CONFIG lines are routed to all processors in the system avoiding the danger of port contention after reset see [Section 8.1.2](#).

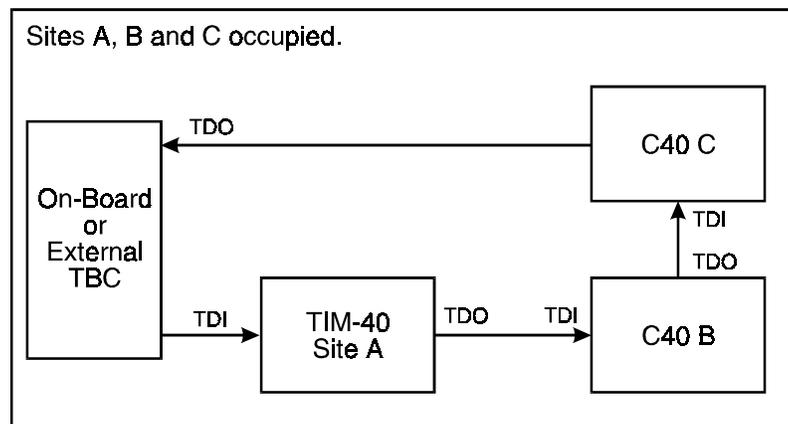
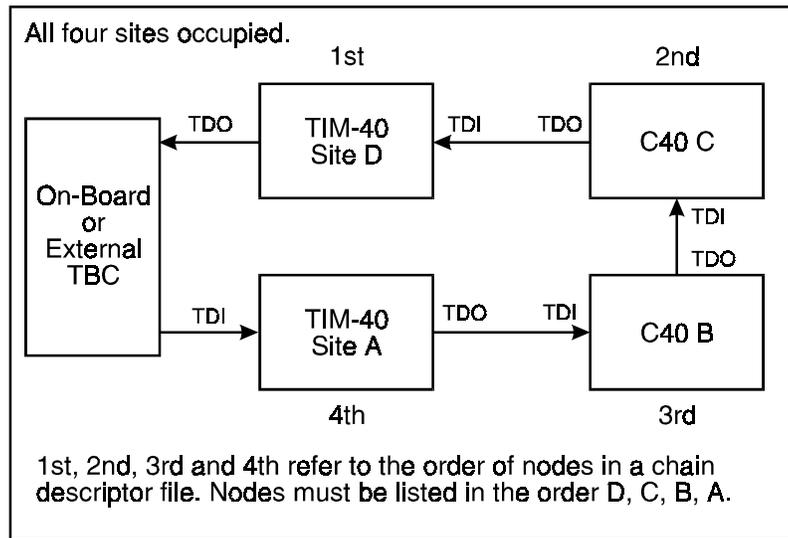
Note that by reading the DBV46 System Status Register on any of the DBV46 boards in the chain, you can establish some useful information about the system. Bit 0, /GRESET, detects if any board(s) in the chain is pulling the global reset line low and bit 3, /CONFIG, detects if any board(s) is pulling the configuration line low. See [Section 5.2.5](#) for more information on this register.

11.4 JTAG Scan Path Routing

Whether using the on-board TBC or an external TBC, DBV46 sets up a JTAG scan path which is routed through all processing nodes on the board. DBV46 automatically detects which TIM-40 module sites are occupied and routes the scan path accordingly so this is transparent to the user. However, the order of the scan path is important. For example, when generating a chain descriptor file, such as **board.cfg** for Texas Instruments' DB40 debugger, the processing nodes must be listed in a particular order.

The order of the scan path is illustrated in [Figure 11.3](#) below. This shows how the JTAG Test Data Input and Output signals (TDI and TDO) are daisy chained through all processing nodes. Although TDI is routed to TIM-40 Site A first, JTAG views Site A as the last node in the JTAG scan chain. Therefore, when listing each processing node in any chain descriptor file, such as **board.cfg**, the nodes must be listed in the order D, C, B, A. This is also illustrated in [Figure 11.3](#).

Figure 11.3: JTAG Scan Path Routing



12 Reset Sources

There are three possible reset sources for DBV46:

- VMEbus system reset pin, /SYSRESET.
- Front panel reset switch.
- RESET bits in the DBV46 System Reset Register.

Each of these reset sources is discussed in the sections below.

12.1 /SYSRESET

Whenever the host system is powered up, the VMEbus /SYSRESET pin is asserted. This resets the whole host system. This includes resetting all processing nodes, clearing all the DBV46 registers to their default state and resetting all the SCV64 internal registers (see [Section 5.4](#) of this User Guide and Appendix A of the SCV64 User Manual for their reset values) on all DBV46 boards in the system.

12.2 Reset Switch

The reset switch on the front panel is a useful debug feature. Toggling this switch will reset and release the DBV46 board, without having to reboot the whole host system. This resets all the processing nodes and clears the DBV46 registers to their default state. The reset switch also resets certain SCV64 internal registers by asserting /LRST (see [Section 5.4](#) of this User Guide and Section 2.12 of the SCV64 User Manual for their reset values).

In a multi-board system, where the boards are connected in a daisy-chain via the JTAG connectors on the front panel, toggling the reset switch on one board will generate a GRESET signal to all the other boards. This means that the DBV46 and SCV64 registers will only be reset on the board that the reset was issued from.

Note that if DBV46 is configured as the Slot 1 system controller (see [Section 7.5](#)), the reset switch will NOT assert the VME /SYSRESET line. In this situation, a rack mounted reset switch is required to assert /SYSRESET. This is described in detail in Appendix E of the SCV64 User Manual.

12.3 DBV46 System Reset Register

This 8 bit read/write register is used to reset the processing nodes within your system and is cleared to zero on power-up/reset. Note that this register can ONLY be accessed by the VMEbus. The individual bit functions of this register are shown in [Table 12.1](#) below.

Table 12.1: System Reset Register

D31 - D8	D7	D6	D5	D4	D3	D2	D1	D0
Not Used	PIM_RESET	DRESET	CRESET	BRESET	ARESET	NODE_RESET	GRESET	BRD_RESET

Bit	Name	Function
0	BRD_RESET	<p>Board Reset</p> <p>Write a '1' to this bit to reset DBV46. This bit is automatically cleared to zero once the reset operation has been performed. A board reset will also:</p> <ul style="list-style-type: none"> Assert the /GRESET line. Therefore, in a multi-board system where the JTAG headers must be connected across boards (see Chapter 11), setting this bit on one board resets all the processing nodes in the system. Reset certain SCV64 internal registers by asserting /LRST (see Section 5.4 of this User Guide and Section 2.12 of the SCV64 User Manual for their reset values).
1	GRESET	<p>Global Reset</p> <p>This active high control bit asserts the global reset line, which is active low (when set to '1', the line is pulled low). The global reset line is part of the TIM-40 specification. To reset the processing nodes on DBV46, this bit should be set to '1'. To release the nodes from reset, this bit should be cleared to '0'. Note that setting this bit will reset the processing nodes only (that is, no other DBV46 resources will be reset).</p> <p>In a multi-board system where the JTAG headers must be connected across boards (see Chapter 11), setting this bit on one board resets all the processing nodes in the system.</p> <p>Note that reading back this register on a board only tells you if that board has pulled the global reset line low. This differs from the /GRESET bit in the System Status Register which reads '0' when any board in the system has pulled the line low.</p>
2	NODE_RESET	<p>All Node Reset</p> <p>1 - All nodes (A, B, C and D) held in reset 0 - Release all nodes (A, B, C and D) from reset</p>
3	ARESET	Individual Node Reset
4	BRESET	1 - Node held in reset
5	CRESET	0 - Release node from reset
6	DRESET	Note that these bits should only be used if you wish to have the option of releasing nodes from reset individually, see WARNING note below.
7	PIM_RESET	<p>1 - PIM held in reset 0 - Release PIM from reset</p>



To prevent port contention between processing nodes, all nodes connected via communication port **MUST** be reset at the same time. Therefore, it is recommended that you use the **NODE_RESET**, **BRD_RESET** or **GRESET** bits as required to simultaneously reset and release from reset all processing nodes. However, if you wish to have the option of releasing nodes from reset individually, the Individual Node Reset bits can be used. In this case all appropriate bits **MUST** set to '1' at the same time, that is, in the same write cycle. Failure to reset nodes simultaneously could cause damage to the C4x processors due to port contention.

Index

	Page		Page
A		M	
Access Protection	89	Memory Maps	
Arbitration	22, 38, 77	DBV46 VMEbus Slave Memory Map	83
B		Default DBV46 VMEbus Master Memory Map	92
Board Layout	13, 15	On-Board C40 Global Memory Map	23
Booting	13, 18, 27, 44	On-Board C40 Local Memory Map	20
Bus Locking	50, 51, 56, 79	PIM-3 Interface Slave Memory Map	116
C		PIM-3 Master Memory Map	118
Communication Ports	53, 103, 113	Private Global Memory Resources	76
D		Shared Bus Memory Map	47
DBV46 Control Registers	48, 85	TIM-40 Module Global Memory Map	40
DBV46 Interrupt Registers	57, 121	VME IACK Space	88
DBV46 Shared Bus	10, 22, 39, 45, 116	P	
Deadlocks	52, 80	PIM-3 Interface	113
F		Processing Nodes	20, 37, 75, 81, 91
Front Panel LEDs	14, 52	R	
G		Register Maps	
Global Memory Interface Control Register	26, 41	DBV46 Control Registers	48
I		DBV46 Interrupt Registers	58
IACK		SCV64 Register Set	67
C4x IACK	36, 44	TBC Register Set	69
VME IACK	87	Reset	
Installation	16	DB40	127
Interrupts		JTAG	128
DBV46 to VME	101	LED	14, 52
Deadlock	80	Multi-Board System	108, 131
IIOF Lines	36, 44	Sources	133
Implementation and Configuration	121	Status	54
Location Monitor	86	System Reset Register	49
PIM-3	119	S	
Registers	57	SCV64 Register Set	66, 85
Timeout	78	Shared Memory	
VME to DBV46	87	DRAM	70, 81, 85
J		PEROM	71, 81, 85
JTAG	69, 108, 127	SW2	13, 27, 54
L		SW3	30
Local Memory Interface Control Register	25	System Controller Facilities	102
Location Monitor	67, 83, 86	T	
J		TBC	69, 81, 127
J		TBC Register Set	69, 85
J		Timeout	78, 80
J		Configuration	51
J		LED	53
J		Status	57
J		TMS320C4x	1, 4, 6, 7
J		V	
J		VMEbus Interface	
J		Master	91
J		Slave	81

