MPC566EVB User's Manual

MPC566EVBUM
Rev. 1.2, 3/2003

Digital DNA
from Motorola

MOTOROLA

Revision History

Version Number	Revision Date	Description of Changes
1.1	11/2002	Initial Version
1.2	3/2003	Fixed typos. Added appendix describing dBUG ethernet configuration. Added appendix for emulating the MPC53X parts.

DigitalDNA and Mfax are trademarks of Motorola, Inc.
IBM PC and IBM AT are registered trademark of IBM Corp.
All other trademark names mentioned in this manual are the registered trade mark of respective owners

No part of this manual and the dBUG software provided in Flash ROM's/EPROM's may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. Use of the program or any part thereof, for any purpose other than single end user by the purchaser is prohibited.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1–303–675–2140 or 1–800–441–2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3–20–1, Minami–Azabu. Minato–ku, Tokyo 106–8573 Japan. 81–3–3440–3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852–26668334

Technical Information Center: 1–800–521–6274

HOME PAGE: http://www.motorola.com/semiconductors

Document Comments: FAX (512) 895-2638, Attn: RISC Applications Engineering

World Wide Web Addresses: http://www.motorola.com/PowerPC
http://www.motorola.com/NetComm
http://www.motorola.com/ColdFire

Axiom Manufacturing: http://www.axman.com

© Motorola Inc., 2002. All rights reserved.

Cautionary Notes

Axiom Manufacturing (http://www.axman.com) reserves the right to make changes without further notice to any products to improve reliability, function or design. Axiom does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under patent rights or the rights of others. Axiom products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Axiom product could create a situation where personal injury or death may occur. Should Buyer purchase or use Axiom manufacturing products for any such unintended or unauthorized application, Buyer shall indemnify and hold Axiom Manufacturing and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Axiom Manufacturing was negligent regarding the design or manufacture of the part or system.

EMC Information on MPC566EVB

1. This product as shipped from the factory with associated power supplies and cables, has been tested and meets with requirements of EN5022 and EN 50082-1: 1998 as a **CLASS A** product.
2. This product is designed and intended for use as a development platform for hardware or software in an educational or professional laboratory.
3. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
4. Anti-static precautions must be adhered to when using this product.
5. Attaching additional cables or wiring to this product or modifying the products operation from the factory default as shipped may effect its performance and also cause interference with other apparatus in the immediate vicinity. If such interference is detected, suitable mitigating measures should be taken.

WARNING

This board generates, uses, and can radiate radio frequency energy and, if not installed properly, may cause interference to radio communications. As temporarily permitted by regulation, it has not been tested for compliance with the limits for class a computing devices pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference. Operation of this product in a residential area is likely to cause interference, in which case the user, at his/her own expense, will be required to correct the interference.

Contents

Paragraph Number **Title** **Page Number**

Chapter 1
MPC566 EVB Board

1.1 Processor 1-3
1.2 System Memory 1-4
1.2.1 External Flash 1-4
1.2.2 SRAM 1-5
1.2.3 Internal SRAM 1-5
1.2.4 Internal Flash 1-5
1.2.5 MPC566EVB Memory Map 1-5
1.2.5.1 Memory Device / Bank Selection and Configuration 1-6
1.2.5.2 Memory Bank Chip Select Configuration 1-7
1.2.5.3 Reset Vector Mapping 1-7
1.3 Support Logic 1-8
1.3.1 Reset Logic 1-8
1.3.2 Clock Circuitry 1-8
1.3.3 Watchdog Timer 1-8
1.3.4 Exception Sources 1-8
1.3.5 TA Generation 1-9
1.3.6 User’s Program 1-9
1.3.7 Power Oak K/I/S Hardware Options 1-10
1.4 Communication Ports 1-11
1.4.1 COM1 - COM4 1-11
1.4.2 CAN PORTs and Options 1-13
1.4.3 10/100T Ethernet Port 1-14
1.4.4 BDM and NEXUS Development Ports 1-15
1.4.4.1 BDM Port Options 1-15
1.4.4.2 Nexus Connector 1-16
1.5 Connectors and User Components 1-19
1.5.1 Keypad 1-19
1.5.2 LCD Port 1-19
1.5.3 User Components 1-21
1.5.4 MPC566EVB Hardware Options 1-22
1.5.5 Signals Available on Board 1-22
1.5.5.1 IRQ PORT 1-22
1.5.5.2 BUS_PORT 1-22
1.5.5.3 TPU_PORTs 1-24
1.5.5.4 CONTROL_PORT 1-24
1.5.5.5 MIOS_PORT 1-25
1.5.5.6 QADC_PORTs 1-26
1.5.5.7 QSM_PORTs 1-27
1.5.5.8 MICTOR 1 – 3 PORTs 1-28

Contents

Paragraph Number | Title | Page Number

1.6 Reference Documents 1-28
1.7 Software Development 1-28

Chapter 2 Initialization and Setup

2.1 System Configuration 2-1
2.2 Installation And Setup 2-3
2.2.1 Unpacking 2-3
2.2.2 Preparing the Board for Use 2-3
2.2.3 Providing Power to the Board 2-3
2.2.4 Selecting Terminal Baud Rate 2-5
2.2.5 The Terminal Character Format 2-5
2.2.6 Connecting the Terminal 2-5
2.2.7 Using a Personal Computer as a Terminal 2-6
2.3 MPC566EVB Jumper and Switch Setup 2-6
2.3.1 Reset Configuration Word and Configuration Switch (CONFIG_SW) 2-8
2.3.2 Memory Configuration (MAP_SW) 2-10
2.4 System Power-up and Initial Operation 2-10

Chapter 3
Using the Monitor/Debug Firmware

3.1 What Is dBUG? 3-1
3.2 Operational Procedure 3-2
3.2.1 System Power-up 3-2
3.2.2 System Initialization 3-4
3.2.2.1 Hard RESET Button. 3-5
3.2.2.2 Non-Maskable Interrupt Button. 3-5
3.2.2.3 Software Reset Command. 3-5
3.3 Command Line Usage 3-6
3.4 Commands 3-6
3.5 System Call Functions 3-39
3.5.1 OUT_CHAR 3-39
3.5.2 IN_CHAR 3-39
3.5.3 IN_STAT 3-40
3.5.4 ISR_REGISTER 3-40
3.5.5 ISR_REMOVE 3-40
3.5.6 EXIT_TO_dBUG 3-41

Appendix A
MPC535/536 Emulation

Contents

Paragraph Number | **Title** | **Page Number**

Appendix B
Configuring dBUG for Network Downloads

B.1 Required Network Parameters B-1
B.2 Configuring dBUG Network Parameters B-1
B.3 Troubleshooting Network Problems B-2

Contents

Paragraph Number	Title	Page Number

Chapter 1

MPC566 EVB Board

The MPC566EVB is an MPC566-based evaluation board that can be used for the development and test of microcontroller systems[1] (see Figure 1-1). The MPC566 is a member of the Motorola MPC500 RISC microcontroller family. It is a 32-bit processor with a 32-bit internal address bus and 32 lines of data.

The evaluation board is a development and test platform for software and hardware for the MPC566[1]. The system provides for development of target applications for the similar MPC561, MPC562, or MPC563 microcontrollers also. It can be used by software and hardware developers to test programs, tools, or circuits without having to develop a complete microcontroller system themselves. All special features of the MPC566[1] are supported.

The heart of the evaluation board is the MPC566. The MPC566EVB has 1Mbyte (256K x 32) external SRAM for development or application memory, 2Mbyte (512K x 32) external Flash memory, and numerous hardware expansion possibilities. The MPC566EVB board also provides an Ethernet interface (10/100BaseT), TouCAN, and RS232 interface in addition to the built-in I/O functions of the MPC566[1] device for programming and evaluating the attributes of the microprocessor. To support development and test, the evaluation board can be connected to debuggers and emulators produced by different manufacturers.

The MPC566EVB provides for low cost software testing with the use of a ROM resident debug monitor, dBUG, programmed into the external Flash device. Operation allows the user to load code in the on-board RAM, execute applications, set breakpoints, and display or modify registers or memory. After software is operational, the user may program the MPC566 Internal Flash EEPROM or the on-board FLASH memory for dedicated operation of new software application. No additional hardware or software is required for basic operation. For high level debug, extensive third-party tools are available for the MPC500 series. The Nexus and BDM debug ports are available to connect the tool sets.

Specifications

Clock: 56 MHz Maximum, 4Mhz reference
Operating temperature: 0°C to +70°C
Power requirement: 6 – 26V DC @ 300 ma Typical
Power output: 5.8V @ 1.5A output with 5V, 3.3V, and 2.6V regulated supplies
Board Size: 7.00 x 7.60 inches, 8 layers

---

[1] The MPC566EVB can be used to emulate the MPC535 and MPC536. See Appendix A and MPC566 Electrical Spec for limitations.

•**Memory Devices**:
1M Byte (256K x 32) Sync. SRAM
2M Byte (512K x 32) Sync. FLASH
1M Byte FLASH internal to MPC566 device
36K Byte SRAM internal to MPC566 device
•**POWER OAK** (PC33394 P2.6) regulated power supply for 5V, 3.3V and 2.6V supplies.
•**MAP Switch** – provides easy assignment of chip selects and memory mapping.
•**CONFIG Switch** – Basic necessary Reset Word Configuration options.
•**COM1** - SCIA1 with RS232 type DB9-S Connection
•**COM2** - SCIA2 with RS232 type DB9-S Connection, TX / RX polarity option.
•**COM3 & 4** - SCIB1 and SCIB2 with RS232 type DB9-S Connection, 10 pin IDC ribbon header connectors.
•**CAN Ports**[1] – 3 CAN transceiver interfaced ports, 1 x 4 headers.
•**10/100T Ethernet Port** – LAN91C111 based MAC+PHY, memory mapped.
•**DEVELOPMENT Ports** – Nexus 50 pin and dual voltage BDM Port.
•**LCD Port** - LCD Module Interface Connector w/ Contrast Adjust, Buffered and Memory Mapped
•**KEYPAD Port**[1] - 16 Key passive interface, applies QADC_B channels for operation.
•**BUS Port** – 32 data and 24 address lines on 60 pin header.
•**CONTROL Port** - Bus Controls with 40 pin header.
•**QSM Ports** – 2 Serial I/O ports with 16 pin socket headers.
•**MIOS Port** - MDA, PWM, and MGPIO ports with 34 pin socket header.
•**TPU Ports**[1] - 3 Timing Processor I/O ports with 20 pin socket headers.
•**QADC Ports**[1] - 2 Analog I/O ports, one 20 pin and one 24 pin socket header.
•**IRQ Port** – Interrupt or MPC566 port I/O with 10 pin socket header.
•**POWER Port** – Primary and standby power supply access port, no header.
•**I/O Connectors** in .1 grid, pin headers for bus and control provide easy ribbon cable connection for external connections. Socket headers provide easy wire connection to breadboard prototype area with 22ga solid wire.
•**Large Prototyping Area** with +5V and ground connection grids.
•Mictor Logic Probe connectors for the Address and Data bus (Not installed default)
•**Breadboard Prototyping area** (2.5 x 1.5 inch) for easy installation of test connections.
•**System Indicators** – Reset Indicator, Supply voltage indications for 5V, 3.3V, and 2.6V supplies
•**Reset Switches** – POReset, Hard, Soft reset buttons.
•**User Components** – 4 user LEDs (one with debounce), 4 user Switches, 1 user Potentiometer with socket header for I/O connection.

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A

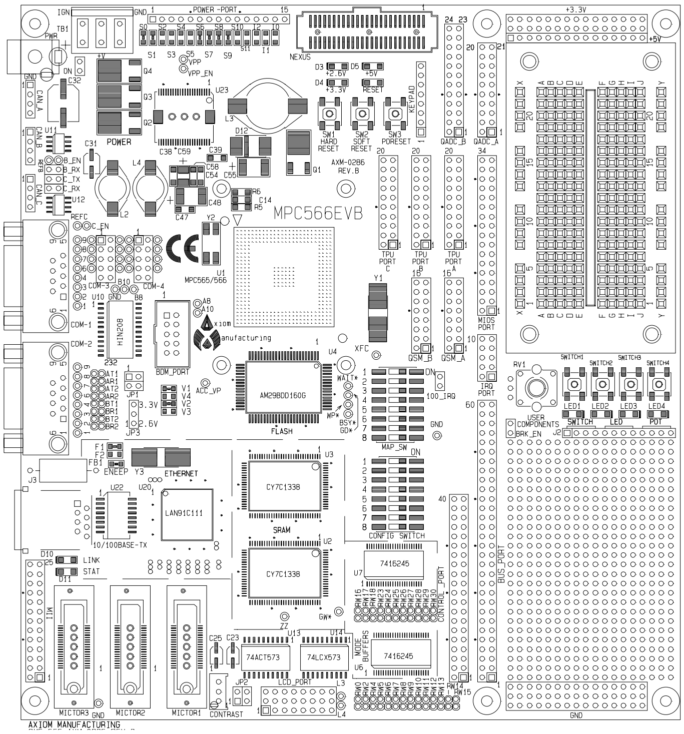

POWER -PORT
NEXUS
POWER
MPC566EVB
AXM-0286 REV.B
SW1 HARD RESET
SW2 SOFT RESET
SW3 PORESET
TPU PORT C
TPU PORT B
TPU PORT A
QADC_B
QADC_A
MIOS PORT
QSM_B
QSM_A
IRQ PORT
COM-1
COM-2
COM-3
COM-4
BDM_PORT
HIN208
MPC565/566
AM29BDD160G
FLASH
MAP_SW
CONFIG SWITCH
CY7C1338
SRAM
LAN91C111
ETHERNET
10/100BASE-TX
LINK
STAT
7416245
74ACT573
74LCX573
LCD_PORT
CONTRAST
MICTOR3
MICTOR2
MICTOR1
BUS_PORT
USER COMPONENTS
SWITCH1
SWITCH2
SWITCH3
SWITCH4
LED1
LED2
LED3
LED4
+3.3V
+5V
GND
AXIOM MANUFACTURING

Figure 1-1. MPC566 EVB top view

1.1 Processor

The microprocessor used on the MPC566EVB is the highly integrated Motorola PowerPC MPC566 32-bit microcontroller. The MPC566 implements a PPC ISA core with 1MByte UC3F flash, four UART channels, three Timing Processor Units (TPUs)[1], 36 KBytes of SRAM plus 4KBytes DECRAM, a QSPI (Queued Serial Peripheral Interface) module, three TouCAN modules[1], 12 PWMs and 6 counter submodules in the MIOS, enhanced QADC64E, PPM module[1], a Nexus debug interface port, and 6KByte DPTRAM[1]. This processor communicates with external devices over a 32-bit wide data bus, D[0:31]. The MPC566 can address a 32 bit address range. Only 24 bits are available on the bus however. There are internally generated chip selects to allow the full 32 bit address range to be selected. There are regions that can be decoded to allow superviser, user, instruction, and data each to have the 32-bit address range. All the processor's signals are available through the expansion connector (BUS_PORT). Refer to the schematic for their pin assignments.

The MPC566 processor has the capability to support both an IEEE-ISTO 5001-1999 NEXUS port and a BDM debug port. These ports are multiplexed and can be used with third party tools to allow the user to download code to the board. The board is configured to boot up in the normal/BDM mode of operation. The BDM signals are available at the port labeled BDM_PORT. The NEXUS connector is near the reset switches on the board. It is the 2002 50 pin standard I/O connections and connector. The BDM and NEXUS ports can not be used at the same time.

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A.

Figure 1-2 shows the MPC566 block diagram.

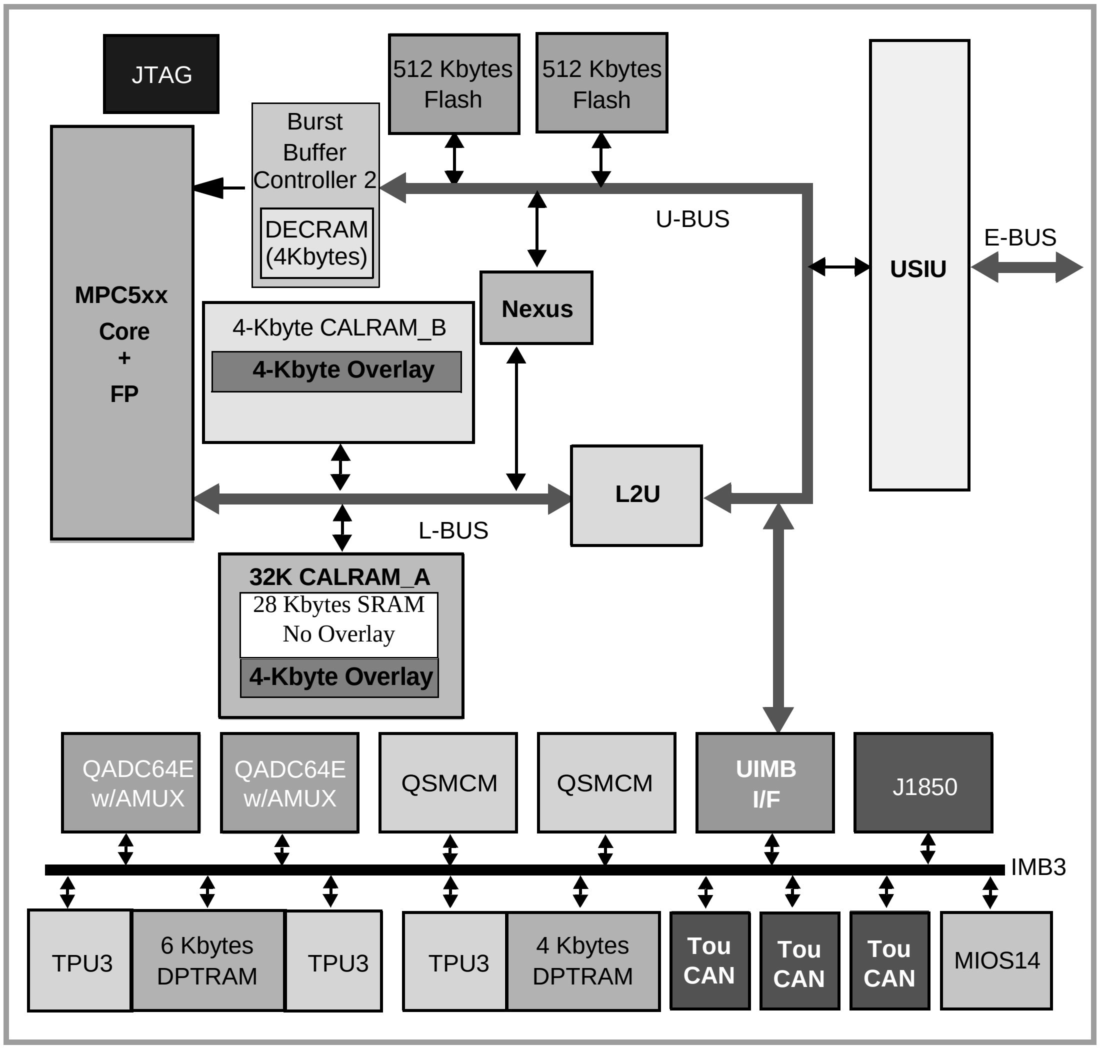

JTAG
512 Kbytes Flash
512 Kbytes Flash
Burst Buffer Controller 2
DECRAM (4Kbytes)
U-BUS
E-BUS
USIU
Nexus
MPC5xx Core + FP
4-Kbyte CALRAM_B
4-Kbyte Overlay
L2U
L-BUS
32K CALRAM_A
28 Kbytes SRAM No Overlay
4-Kbyte Overlay
QADC64E w/AMUX
QADC64E w/AMUX
QSMCM
QSMCM
UIMB I/F
J1850
IMB3
TPU3
6 Kbytes DPTRAM
TPU3
TPU3
4 Kbytes DPTRAM
Tou CAN
Tou CAN
Tou CAN
MIOS14

Figure 1-2. MPC566[1] Block Diagram

1.2 System Memory

1.2.1 External Flash

One on-board Flash ROM (U4) is used in the system. The Am29BDD160G device contains 16Mbits of non-volatile storage (1 M x 16-bit/512 K x 32-bit) giving a total of 2MBytes of Flash memory. This device requires a 32 bit wide port and must be written in 32 bit word size data. It may be read in bytes, half words, or words. Wait states are required to access in asynchronous mode and the same wait state delay is required during the first cycle of a burst type access. Refer to the specific device data sheet for configuring the flash memory. User should note that the debug monitor firmware is installed in this flash device. Development tools or user application programs may erase or corrupt the debug monitor. If the debug monitor becomes corrupted and it's operation is desired, the firmware must be programmed into the flash by applying a development port tool such as BDM or Nexus. User should use caution to avoid this situation. The upper 1 MByte is used to store the MPC566EVB dBUG debugger/monitor firmware (0x0090_0000 to 0x009F_FFFF).

---

[1] See Appendix A for block diagram of MPC535/6

1.2.2 SRAM

The MPC566EVB has two 512 KByte device on the board (U2). It's starting address is 0xFFF0_0000.

The synchronous SRAM Memory Bank is composed of two128K x 32 memory devices. These memory devices are connected in linear order from U2 to U3 if more than one is available, so that the low order address of the memory bank will access U2 and the high order addresses of the memory bank will access U3. This memory bank must be configured as a 32 bit wide port but is byte, half word, and word accessible for read or write operations.

Also see Section 1.2.5, "MPC566EVB Memory Map".

1.2.3 Internal SRAM

The MPC566 processor has 36-KBtyes of internal memory which may be used as data or instruction memory. This memory is mapped to 0x003F_8000 and configured as data space but is not used by the dBUG monitor except during system initialization. After system initialization is complete, the internal memory is available to the user. The memory is relocatable to any 32-KByte boundary.

1.2.4 Internal Flash

The MPC566 has a U-bus CDR3 flash EEPROM module (UC3F). The primary function of the UC3F flash EEPROM module is to serve as electrically programmable and erasable non-volitle memory (NVM) to store program instructions and/or data. The MPC566 flash EEPROM array has 1 Mbytes of NVM that is divided into sixteen 64-Kbyte array blocks. If the flash array is disabled in the IMMR register (FLEN=0), then neither the UC3F array or the UC3F control registers are accessible. This feature allows the MPC566 to emulate the MPC561/562.

Please refer to the MPC566 User's Manual for more details.

1.2.5 MPC566EVB Memory Map

Interface signals to support interface to external memory and peripheral devices are generated by the memory controller. It supports four regions on four chip-select pins. The general purpose chip-selects are available on lines $\overline{\text{CS[0]}}$ through $\overline{\text{CS[3]}}$. $\overline{\text{CS[0]}}$ also functions as the global (boot) chip-select for booting out of external flash.

Since the MPC566 chip selects are fully programmable, the memory banks can be located at any location in the MPC5xx memory space.

Following is the default memory map for this board as configured by the Debug Monitor located in the external flash bank. The internal memory space of the MPC566 is detailed further in the MPC565-6 Users Manual. Chip Selects 0-3 can be changed by user software to map the external memory in different locations but the chip select configuration such as wait states and transfer acknowledge for each memory type should be maintained.

Possible Chip Select usage:

Synchronous SRAM Memory Bank	CS0 or CS1	default CS1, MAP SW. 1,2
Synchronous FLASH Memory Bank	CS0 or CS2	default CS0, MAP SW. 3,4
10/100T / LCD Port	CS3	default CS3, MAP SW. 6,7
MPC566 Internal Flash	N/A	MAP_SW. 8, may effect memory map of chip selects

The MPC566EVB Default Memory Map shows the MPC566EVB memory map.

Table 1-1. The MPC566EVB Default Memory Map

Address Range	Signal and Device
0x0000_0000 - 0x0007_FFFF	1MB UC3F Flash
0x0010_0000 - 0x002F_7FFF	Reserved for Flash
0x002F_8000 - 0x002F_8FFF	BBC DECRAM 4 KBytes
0x002F_9000 - 0x002F_9FFF	Reserved for BBC
0x002F_A000 - 0x002F_BFFF	BBC Control
0x002F_C000 - 0x002F_FFFF	USIU & Flash Control
0x0030_0000 - 0x0030_7FFF	UIMB I/F & IMB Modules 32KBytes
0x0030_8000 - 0x0037_FFFF	Reserved for IMB
0x0038_0000 - 0x0038_007F	CALRAM/READI Control
0x0038_0080 - 0x0038_3FFF	Reserved (L-bus Control)
0x0038_4000 - 0x003F_6FFF	Reserved (L-bus Memory)
0x003F_7000 - 0x003F_FFFF	CALRAM (internal SRAM)
0x0080_0000 - 0x00A0_0000	External Flash 2MByte (0x0090_0000 - 0x00A0_0000 reserved for dBUG)
0x0100_0000 - 0x0108_0000	Ethernet
0xFFF0_0000 - 0xFFFF_FFFF	External SRAM 1MB

1.2.5.1 Memory Device / Bank Selection and Configuration.

The MPC566EVB board has one internal memory bank, two external memory banks and a Peripheral memory bank that provide:

- 36KByte Internal SRAM
- MPC566 512K byte Internal FLASH Memory (U1)
- 256K x 32bit (1MByte) Synchronous Static RAM (U2/3).

- 512K x 32bit (2MByte) Synchronous Flash EEPROM (U4)
- Peripherals 10/100T Ethernet and LCD Port

Each external RAM or Flash memory bank can be configured individually to operate from the MPC566 chip selects. Caution should be used not to place more than one memory bank on the CS0 chip select and to properly configure the chip select to control the memory devices provided in the memory bank correctly. **Failure to observe precautions may render the external memory bus inoperable.**

The MAP Switch (MAP_SW) connects MPC566 chip selects to the different external memory banks. If memory access problems occur, the settings of these options and the associated chip select configurations should be reviewed with some detail. Information to configure the chip selects and memory is detailed in the following section.

1.2.5.2 Memory Bank Chip Select Configuration

Application software that executes on Reset must configure each memory bank chip select properly for correct operation. Chip Select Memory Options shows the default memory settings programmed by the dBUG ROM monitor and may be applied for most user applications:

Table 1-2. Chip Select Memory Options

Memory Bank	Reg.	Default Value	Notes
CS1 = SRAM	BR1	0xFFF0_0003	Base Address = 0xFFF0_0000, Port width = 32 bit ***Default**
CS1 = SRAM, asynchronous access mode	OR1	0xFFF0_0000	Memory Range = 0xFFF0_0000 > 0xFFFF_FFFF, wait state = 0.
CS0 = FLASH	BR0	0x0080_0003	Base address 0x0080_0000, Port width = 32 bit ***Default**
CS0 = FLASH, asynchronous access mode	OR0	0xFFE0_0030	Memory range = 0x0080_0000 > 0x009F_FFFF, wait state = 3, asynchronous operation 40Mhz clock, 95ns device. Note U4 = 2M bytes and will mirror 2x with this setting. Usable range = 0x0080_0000 > 0x008F_FFFF (dBUG monitor is in upper half starting at 0x0090_0000)
CS3 = Peripheral	BR3	0x0100_0807	Base address = 0x0100_0000, Port width = 16 bit ***Default** External TA* generation provided.
CS3 = Peripheral, asynchronous	OR3	0xFFFF_80F0	Memory Range 0x0100_0000 > 0100_7FFF, wait state = External Terminate (TA*) ***Default** Note Peripheral memory map.

1.2.5.3 Reset Vector Mapping

After reset, the processor attempts to execute at physical address 0x0000_0100 if the hard reset configuration word IP bit is cleared to 0 or physical address 0xFFF0_0100 if the hard reset configuration word IP bit is set to 1. This requires the board to have a non-volatile memory device in this range with the correct information stored in it. The MPC566 processor chip-select zero ($\overline{\text{CS0}}$) responds to any accesses after reset until the OR0 is written. Since $\overline{\text{CS0}}$ (the global chip select) is connected to the Flash ROM (U6), the Flash ROM initially appears at address 0xFFF0_0000. The initialization routine then programs the chip-select logic, locates the Flash

ROM to start at 0x0080_0000 and configures the rest of the internal and external peripherals. Please refer to the MPC565 user's manual (Global (Boot) Chip-Select Operation) for more information.

1.3 Support Logic

1.3.1 Reset Logic

The reset logic provides system initialization. Reset occurs during power-on or via assertion of the signal $\overline{\text{RESET}}$ which causes the MPC566 to reset. $\overline{\text{HRESET}}$ is triggered by the reset switch (SW1) which resets the entire processor/system.

dBUG configures the MPC566 microprocessor internal resources during initialization. The contents of the exception table are copied to address 0xFFF0_0000 in the SDRAM. The Software Watchdog Timer is disabled, the Bus Monitor is enabled, and the internal timers are placed in a stop condition. A memory map for the entire board can be seen in Table 1-1., "The MPC566EVB Default Memory Map".

RW0 – 30: External Reset Configuration Word (RCW) Options

RW0, RW2, RW4 – 18, RW23 – 30 provide the user access to external Reset Configuration Word (RCW) bits not normally required for default MPC566EVB operation. The RW0 – 30 designations reflect the data bus D0 – D30 bit effected when the RCW word is enabled externally. All RW0 – 30 option bits are defaulted to the logic low value during external RCW word operation. The user may apply a wire jumper between the 2 pad positions of each RW0 – 30 option to provide a logic high level on the respective bit position during external RCW operation. Refer to the MPC566 user manual Reset chapter for the respective RCW bit definitions.

1.3.2 Clock Circuitry

The MPC566EVB board uses a 4MHz crystal (Y1 on the schematics) to provide the clock to the on-chip oscillator of the MPC566. In addition to the 4MHz crystal, there is also a 25MHz oscillator (Y3) which feeds the Ethernet chip (U20).

1.3.3 Watchdog Timer

The duration of the Watchdog is selected by the SWT[1:0] bits in the System Protection and Control Register (SYPCR), SWT[1:0] = 0b11 gives a maximum timeout period of 2^{28}/System frequency. The dBUG monitor initializes these bits with the value 0b11, which provides the maximum time-out period, but dBUG does **NOT** enable the watchdog timer via the SYPCR register SWE bit.

1.3.4 Exception Sources

The MPC500 family of processors can receive exceptions as a result of external signals, errors, interrupts, or unusual conditions arising in the execution of instructions. When the processor receives an exception, information about the state of the processor is saved and, after switching to supervisor mode, the processor begins handling the exception based on instructions in the

Exception Vector Table in memory. Exceptions are handled in program order based on PowerPC architecture requirements. When an exception occurs that was caused by an instruction, any unexecuted instructions that appear earlier in the instruction stream are required to complete before the exception is taken. Exceptions no associated with a specific instruction (asynchronous exceptions) are recognized when they occur. Exception handlers should save the information in SRR0 and SRR1 soon after the exception is taken to prevent this information from being lost due to another exception being taken.

The processor goes to an exception routine via the exception table. This table is stored in the Flash EEPROM. The address of the table location is set by the IP bit (switch 5 of MAP_SWITCH). The dBUG ROM monitor writes a copy of the exception table into the RAM starting at 0xFFF0_0000. To set an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at 0xFFF0_0000.

The MPC566's interrupt controller supports up to 8 external interrups (0 - 7), eight levels for all internal USIU interrupt sources and 32 levels for internal peripheral modules. It has an enhanced mode of operation, which simplifies the MPC566 interrupt structure and speeds up interrupt processing.

NOTE:

No interrupt sources should have the same level and priority as another. Programming two interrupt sources with the same level and priority can result in undefined operation.

The MPC566EVB hardware uses $\overline{\text{IRQ}}$[0]/SGPIOC[0] to support the ABORT (Non Maskable Interrupt) function using the ABORT switch (SWITCH1 when BRK_EN jumper is inserted). This switch is used to force a non-maskable interrupt if the user's program execution should be aborted without issuing a RESET.

Refer to MPC566 User's Manual for more information about the interrupt controller.

1.3.5 TA Generation

The $\overline{\text{TA}}$ signal is driven by the slave device from which the current transaction was addressed. It indicates that the slave has received the data on the write cycle or returned data on the read cycle. If the transaction is a burst, $\overline{\text{TA}}$ should be asserted for each one of the transaction beats. The MPC566 drives $\overline{\text{TA}}$ when the slave device is controlled by the on-chip memory controller or when an external master initiated a transaction to an internal slave module. $\overline{\text{TA}}$ is used to indicate the completion of the bus cycle. It also allows devices with different access times to communicate with the processor properly (i.e. asynchronously) like the Ethernet controller. The internal TA generator is used for all external memories. External TA is only used for Ethernet/LCD.

1.3.6 User's Program

Switch 5 on the MAP_SW bank of switches allows users to test code from boot/PORESET without having to overwrite the ROM Monitor. The user's code will boot from internal flash (0x0000_0000) needs to contain the start of the Exception Vector Table).

When the switch is ON (IP is set), the behavior of the system is normal, dBUG boots and then runs from 0x0090_0000.

Procedure:

1. Compile and link as though the code was to be placed at the base of the internal flash, but setup so that it will download to the SRAM starting at address 0xFFF0_8000. The user should refer to their compiler documentation for this, since it will depend upon the compiler used.
2. Set IP bit (Switch 5 ON).
3. Download to SRAM (If using serial or ethernet, start the ROM Monitor first. If using BDM via a wiggler cable, download first, then start ROM Monitor by pointing the program counter (PC) to 0x0090_0100 and run.)
4. In the ROM Monitor, execute the 'upuser' command.
5. Turn off IP bit (Switch 5 OFF). User code should now be running from reset/POR.

1.3.7 Power Oak K/I/S Hardware Options

Several hardware options surround the Power Oak supply to allow the user access to many of the features. The options are sorted by leading character to indicate functionality. ‘K’ designated options refer to VKAM and MPC566 back-up supply options. ‘I’ designated options refer to Interrupt operation options. ‘S’ designated options refer to MPC566 Reset or I/O signal connection options. Following is the summary table (also refer to MPC566EVB schematic):

Table 1-3. K/I/S Option Table

Option Designator	Power Oak Signal	MPC566 signal	Associated Option[1]	Default Connection
K0	VKAM	KAPWR	K1	Closed, Note: REV B board has 0 ohm resistor installed
K1	2.6V	KAPWR	K0	Open
K2	VKAM	VDDSRAM1	K3	Closed
K3	2.6V	VDDSRAM1	K2	Open
K4	VKAM	VDDSRAM2	K5	Closed
K5	+2.6V	VDDSRAM2	K4	Open
K6	VKAM	VDDSRAM3	K7	Closed
K7	2.6V	VDDSRAM3	K6	Open
K8	VKAM	VDDSYN	K9	Open
K9	2.6V	VDDSYN	K8	Closed
K10	VKAM	VDDRTC	K11	Closed
K11	2.6V	VDDRTC	K10	Open
I0	WAKEUP	IRQ4		Open
I1	WAKEUP	IRQ0	I2	Open
I2	PRERESET	IRQ0	I1	Open

Option Designator	Power Oak Signal	MPC566 signal	Associated Option[1]	Default Connection
S0	PORESETB	PORESET		Closed
S1	HRESETB	HRESET		Closed
S2	SLEEP	RSTCONF_TEXP		Open
S3	REGON	MGPIO15		Open
S4	CANTXD	B_CANTX0	A_CANTX0	Open
S5	CANRXD	B_CANRX0	A_CANRX0	Open
S6	$\overline{\text{CS}}$	A_QSPI_PCS1		Closed
S7	DO	A_QSPI_MISO		Closed
S8	DI	A_QSPI_MOSI		Closed
S9	VREF3	BOEPEE	S11	Open
S10	VREF3	EPEE	S10	Open

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A

1.4 Communication Ports

The MPC566EVB provides external interfaces for 4 SCI serial ports, 3 CAN ports and a 10/100T ethernet port.

1.4.1 COM1 - COM4

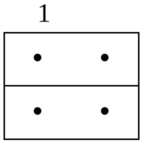
The MPC566 processor has two queued serial multi-channel modules (QSMCM_A and QSMCM_B) which provides four serial communications interfaces (SCI/UART). These submodules communicate with the CPU via a common slave bus interface unit (SBIU). The signals of COM1 and COM2 pass through external Driver/Receivers to make the channels RS-232 compatible. COM3 and COM4 serial ports with 10 pin IDC to DB9 connector headers are available for RS232 operation of MPC566 SCI B channels 1 and 2. An RS-232 serial cable with DB9 connectors is included with the board. The signals of both channels are available on the QSM_PORT connector. SCI0 (COM-1) is the “TERMINAL” channel used by dBUG for communication with an external terminal/PC. The “TERMINAL” baud rate defaults to 19200.

Notes:

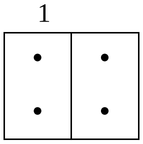
1. COM ports provide connection pads 1 – 9 behind the DB9 cable connectors so the user may modify operation of the serial connection. Each connection pad is numbered for the associated serial connector pin. Each connection pad can be isolated from the others if grouped above, by cutting the associated trace to the pad on the bottom side of the board. See the MPC566EVB schematic.
2. COM-2 has the JP1 DCE/DTE option, see below.

3. RS232 translators available to COM3 and COM4 that are not required by user application may be applied to other COM ports by isolating the MPC566 SCI signals to the RS232 transceiver and applying the associated RS232 level input or output to another COM port. User should refer to the schematic diagrams of the board to make sure correct signals and connections are isolated and reconnected for the new application.

JP1 – COM2 DCE/DTE Option:



COM-2 is optioned as a DCE type RS232 connection by default (same as COM-1). This allows direct connection to a standard 9 pin PC COM serial port.



COM-2 DTE option. This requires a NULL modem adapter to connect to a standard 9 pin PC COM serial port.

AT1/2, AR1/2, BT1/2, BR1/2 - Serial Port Configuration

The Axx and Bxx cut-away options provide a means of isolating the individual SCI RXD and TXD signals from the RS232 interface translator device (U10) and COM port operation. This allows the SCI channels to be used for other purposes, possibly on the QSM port connector. Following is a table of the SCI signals and AT/R_ positions used for enabling RS232 and COM port operation. Note: 2mm header maybe installed after cutaways are cut to allow jumper option.

Table 1-4. Serial Port Configuration

AT_ Position	SCI Channel Signal	RS232 COM Port Connection	COM_PORT Signal Direction to RS232 interface translator
AT1	SCI_A_ TXD1	COM-1	Output
AR1	SCI_A_ RXD1	COM-1	Input
AT2	SCI_A_ TXD2	COM-2	Output
AR2	SCI_A_ RXD2	COM-2	Input
BT1	SCI_B_ TXD1	COM-3	Output
BR1	SCI_B_ RXD1	COM-3	Input
BT2	SCI_B_ TXD2	COM-4	Output
BR2	SCI_B_ RXD2	COM-4	Input

1.4.2 CAN PORTs and Options[1]

The MPC566EVB board provides 3 CAN transceivers with I/O ports: CAN_A, CAN_B, and CAN_C. CAN_A is supported by the PC33394 Power Oak CAN transceiver. The CAN_B and CAN_C ports are supported by Philips PCA82C250 1M Baud CAN transceivers. The MPC566 CAN_A port is directly interfaced to the Power Oak transceiver and can not be isolated easily. The MPC566 CAN_B and C ports are interfaced to the MPC566 TOUCAN channels B and C by option jumpers B_RX, C_TX, and C_RX.

CAN_A

The CAN_A channel transceiver is provided by the Power Oak (PC33394). This transceiver has software selectable options via the QSPI 0 channel which may communicate with the Power Oak device. See the PC33394 data sheet for details. A 4.7K ohm pull-up is provided on the CAN_A TX signal. Options S4 and S5 are provided near the Power Oak device to provide both MPC566 CAN_A and CAN_B channels for messaging on the Power Oak transceiver. If S4 and S5 are connected, the B_RX option from the CAN_B port must be open.

B_RX Option Jumper

This option jumper enables the receive connection from the CAN_B port transceiver to the MPC566 CAN B RX channel. The option allows the isolation of the CAN_B port transceiver RX signal so that the user may use a different connection or transceiver for the MPC566 CAN B port.

C_RX and C_TX Option Jumpers

These options enable the CAN_C port transceiver RX and TX signals to be placed on the MPC566 MGPIO port CAN C signals. The CAN C operation on the MPC566 MGPIO port must be enabled in software, see example source code. The MPC566 MGPIO Port bits 13 and 14 are effected along with the MPC566EVB MIOS Port pins 32 and 33 respectfully.

B_EN and C_EN Option Pads, CN1 and CN2 Option Cut-Aways

These options provide access to the output enable and slew rate control of the respective CAN transceiver. By default the transceivers are set to provide minimum slew rate (fast edge) and to be constantly enabled for output. The configuration of the transceivers maybe modified for slew rate or output control or both. Signaling CAN bus slew rate can be modified by increasing the value of R66 and R67 for CAN_B and CAN_C respectfully. Opening the CN1 and CN2 away options for CAN_B and CAN_C respectfully allows a MPC566 I/O port to be applied to the B_EN and C_EN option pads to provide output control. A high level on the B_EN or C_EN would disable the respective CAN transceiver output. See the PCA82C250 data sheet on the support CD for additional information.

CAN_A, CAN_B, and CAN_C Port Connectors

These ports provide the CAN transceiver input and output connection to the CAN bus. The bias or termination for the CAN bus is provided on the EVB board but not pupualted. If required the user must install these components in the RAx, RBx, or RCx locations near the respective CAN port.. Following are the pin connections for the ports:

Pin 1 = CAN-Hi level signal

Pin 2 = CAN-Lo level signal

Pin 3 = Ground or common (this is required for proper return path on CAN bus)

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A

Pin 4 = +5V supply for remote use or bias of CAN bus.

CAN_A, CAN_B, and CAN_C Port Termination Options

The RA1-3, RB1-3, and RC1-3 option locations provide the respective CAN A, B, or C port with the ability to add bias and/or termination resistance. RA1, RB1, and RC1 locations provide low bias (to ground) on the respective CAN Port CAN Hi signal. RA3, RB3, and RC3 locations provide high bias (to +5V) on the respective CAN Port CAN Low signal. RA2, RB2, and RC2 locations provide termination between the respective CAN Port CAN Hi and CAN Low signals.

1.4.3 10/100T Ethernet Port

The MPC566EVB has an Ethernet controller (SMSC LAN91C111 U20) operating at 10M bits/sec or 100Mbits/sec (see the device data sheet on the support CD forr operation details). The dBUG ROM monitor is programmed to allow a user to download files over a network to memory in different formats. The compiler-formats currently supported are S-Record, COFF, ELF, or Image (raw binary). Refer to Appendix B, “Configuring dBUG for Network Downloads”, for details on how to configure the board for network download.

The Ethernet registers are located at chip select CS3 base address in the address range 0x0000 - 0x000F. The access is 16 bits wide or half word transfers only. The LAN91C111 device applies a register bank selection technique to provide a minimum memory space size. Users should review the device data sheet in detail for operation notes. The debug monitor applies the Ethernet for file downloads only, no high level stacks are applied in the sample source code.

RJ45 jack J3 of the Ethernet port provides a direct to HUB type connection. The Ethernet cable provided with the MPC566EVB kit is a crossover type for direct connection of the EVB to a PC host network card. If connection to a HUB is desired, a standard Ethernet cable should be applied.

Table 1-5. Ethernet Jack J3

PIN	SIGNAL
1	TX+
2	TX-
3	RX+
4	Term 1 75 ohm
5	Term 1 75 ohm
6	RX-
7	Term 2 75 ohm
8	Term 2 75 ohm

100_IRQ Option Jumper

The 100_IRQ Option jumper provides Ethernet Interrupt capability to the MPC566 processor. With the option installed and the LAN91C111 device properly configured, the MPC566 IRQ1 interrupt can be applied to service the port.

LINK and STAT Indicators

The LAN91C111 Ethernet controller provides two indication drivers under software control. The LINK indicator is driven by the LAN91C111 LEDA output and the STAT indicator is driven by the LEDB output.

MII Connector

The MII connector location is for testing and the connection of an external Ethernet PHY device. This connector is not installed or supported by the EVB application.

1.4.4 BDM and NEXUS Development Ports

Both NEXUS (MPC566 Readi Module) and standard BDM (background debug module) development ports are provided on the MPC566EVB for application of integrated software debug tool suites. In order to use the BDM, simply connect the 10-pin debug connector on the board, BDM_PORT, to the P&E BDM wiggler cable provided in the kit. No special setting is needed. Refer to the MPC566 User’s Manual BDM Section for additiona instructions. The NEXUS interface provides the IEEE-ISTO 5001 50 pin standard I/O connections and connector and the BDM port provides the standard 10 pin interface (refer to MPC566EVB schematic sheet 3 for details). User should observe that both ports can not be applied at the same time. Note that the NEXUS interface applies some of the MPC566 standard I/O signals from the MIOS module as alternate development port I/O signals. Following are the I/O effected:

MGPIO 7, 8, 9, 10.

NOTE:

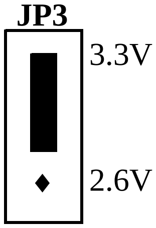
> BDM functionality and use is supported via third party developer software tools. Details may be found on CD-ROM included in this kit

1.4.4.1 BDM Port Options

The BDM Port provides several options for flexibility of operation.

JP3 - BDM Port Interface Level

JP3 provides the option of 2.6V or 3.3V interface levels on the BDM port. This allows the use of legacy MPC555 BDM tools on the MPC566. The option is set for 3.3V interface from the factory. The following JP3 reference is with the MPC566EVB setting with the COM ports facing left.



The signals which are neccessary for debug are available at connector (BDM_PORT). Figure 1-3 shows the (BDM_PORT) Connector pin assignment.

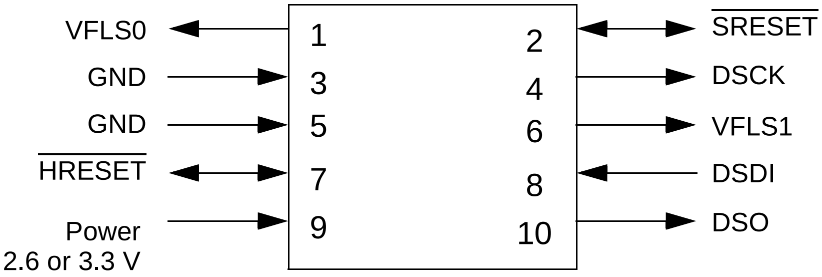


Figure 1-3. The BDM_PORT Connector pin assignment

V1, V2, V3, and V4

The V1 – V4 options provide a way to use the alternate VFLS0 and VFLS1 BDM signals from the MPC566. By default, the primary VFLS0 and VFLS1 signals are applied by V1 and V2 default connections. To modify the BDM port to apply the alternate VFLS0 and VFLS1 signals, options V1 and V2 should be cut and options V3 and V4 connected.

1.4.4.2 Nexus Connector

The 2001-Nexus standard defines several different standards for different speed accesses to a microcontroller in a target system. These standards have been revised since that release. The new connectors come in both a robust and a non-robust configuration. In addition, each connector has 2 definitions depending upon whether the connection is an Auxiliary only (Auxiliary In and Auxiliary Out) connection or a JTAG IEEE 1149.1 port with an Auxiliary Output port.

NOTE

The MPC56x parts do not support the JTAG IEEE 1149.1 configuration.

Table 1-6. MPC56x Nexus 50-Pin Definition (Full-Port Mode)

MPC56x Signal	Nexus Auxiliary Signal	I/O	Pin Number	Pin Number	I/O	Nexus Auxiliary Signal	MPC56x Signal
—	UBATT	OUT	1	2	OUT	UBATT	—
VSTBY2.6	VSTBY	OUT	3	4	IN or OUT	TOOL_IO0	—
—	TOOL_IO1	IN or OUT	5	6	IN or OUT	TOOL_IO2	—
$\overline{\text{HRESET}}$	/RESET[1]	IN[2]	7	8	OUT	VREF	VDD2.6
$\overline{\text{EVTI}}$	/EVTI	IN[2]	9	10	—	GND	GND
$\overline{\text{RSTI}}$	/RSTI	IN[2]	11	12		GND	GND
$\overline{\text{MSEI}}$	/MSEI	IN[2]	13	14	—	GND	GND
MDI[0]	MDI0	IN[2]	15	16	—	GND	GND
MCKI	MCKI	IN[2]	17	18	—	GND	GND
MDO[0]	MDO0	OUT	19	20	—	GND	GND
MCKO	MCKO	OUT	21	22	—	GND	GND
LWP[1]	/EVTO	OUT	23	24	—	GND	GND
$\overline{\text{MSEO}}$	/MSEO0	OUT	25	26	IN or OUT	VENDOR_IO0	LWP[0]
MDO[1]	MDO1	OUT	27	28	—	GND	GND
MDO[2]	MDO2	OUT	29	30	—	GND	GND
MDO[3]	MDO3	OUT	31	32	—	GND	GND
MDI[1]	MDI1	IN[2]	33	34	—	GND	GND
—	/MSEO1	OUT	35	36	—	GND	GND
MDO[4]	MDO4	OUT	37	38	—	GND	GND
MDO[5]	MDO5	OUT	39	40	—	GND	GND
MDO[6]	MDO6	OUT	41	42	—	GND	GND
MDO[7]	MDO7	OUT	43	44	—	GND	GND
—	MDI2	IN[2]	45	46	—	GND	GND
—	MDI3	IN[2]	47	48	—	GND	GND
EPEE & B0EPEE[3]	VENDOR_IO1	IN or OUT	49	50	—	GND	GND

[1] The Nexus specification labels active low signals with a forward slash (/) before the signal name.

[2] The Nexus standard recommends that inputs should have 10K Ω pull-up resistors to VREF (2.6 volts). Exception: The RSTI input should have a 10K Ω pull-down resistor. This is in line with the proposed new standard.

[3] This signal is needed only if control of EPEE or B0EPEE is required by the Nexus tool.

On the MPC56x devices, some of the Nexus signals are shared with other pin functions. Table 1-7 shows the Nexus signal versus the MPC565/566 pins.

Table 1-7. MPC56x Signal Sharing

Nexus Signal		MPC565/566	
		Signal	Ball
	/RESET	$\overline{\text{HRESET}}$	AB23
	/RSTI	$\overline{\text{RSTI}}$	M3
Aux In Port	MCKI	MCKI	L4
	/MSEI	$\overline{\text{MSEI}}$	M4
	MDI0	MDI[0]	L1
	MDI1	MDI[1]	L3
	MDI2	—[1]	—
	MDI3	—[1]	—
	/EVTI	$\overline{\text{EVTI}}$	M2
Aux Out Port	MCKO	MCKO	P3
	/MSEO	$\overline{\text{MSEO}}$	T2
	MDO0	MDO[0]	P4
	MDO1	MDO[1]	R1
	MDO2	MDO[2]	R3
	MDO3	MDO[3]	T1
	MDO4	MDO[4]/ MPIO32B[10]	N2
	MDO5	MDO[5]/ MPIO32B[9]	N4
	MDO6	MDO[6]/ MPIO32B[8]	N3
	MDO7	MDO[7]/ MPIO32B[7]	P1
	/EVTO	$\overline{\text{BG}}$/VF0/LWP1	AF14
Port	PORT0	—[1]	—
Vendor Defined	VENDOR_IO0	SGPIOC[7]/ $\overline{\text{IRQOUT}}$/LWP[0]	AC14
	VENDOR_IO1	EPEE & B0EPEE[2]	AF21 & AD20
Tool Defined	TOOL_IO0	—[3]	—
	TOOL_IO1	—	—
	TOOL_IO2	—	—
	VREF	VDD2.6	VDD
	VALTREF	VSTBY2.6	—[4]

[1] This optional signal is not available on the MPC56x devices.

[2] This signal is needed only if control of EPEE or B0EPEE is required by the Nexus tool.

[3] This optional signal is defined for use by tool vendors and has no defined connection to the MPC56x device.

[4] This signal depends on the following system requirements: KAPWR (Y26), VDDSRAM1 (E3), VDDSRAM2 (D2), or VDDSRAM3 (G4). This pin should be connected to the standby supply used in a particular system.

1.5 Connectors and User Components

1.5.1 Keypad[1]

The Keypad port provides MPC566 I/O connections for a passive 16 key 4x4 matrix keypad (Axiom HC-KP). The port is applied as 4 column drivers and 4 row inputs. The MPC566 QADC_B Ports PQA[4:7] are column outputs and PQB[4:7] are row inputs. The row inputs have 10K ohm pull-down resistance applied to idle the row inputs low. A simple keypad scan will enable each column output high one at a time and test the input rows for a high input. Column and row position can then determine the value of the key pressed. Sample software for driving the Keypad Port is provided on the MPC5xx support CD.

Table 1-8. Keypad

PIN	MPC566 **I/O SIGNAL**
1	QADC_B PQA4 out
2	QADC_B PQA5 out
3	QADC_B PQA6 out
4	QADC_B PQA7 out
5	QADC_B PQB4 in
6	QADC_B PQB5 in
7	QADC_B PQB6 in
8	QADC_B PQB7 in

1.5.2 LCD Port

The LCD Port provides a versatile connector to attach 80 or 160 character display modules and some graphics display modules with embedded controllers. Most LCD modules operate very slowly compared to the MPC566 bus operation speeds so the LCD port provides a 16 bit wide register port for access. The registered port allows writing the LCD module control signals and full Read and Write capability for LCD module command and data bytes without the CPU waiting for LCD access time. LCD Port access are performed as a 3 or 4 bus cycle transaction as follows:

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A

Access cycle 1: Write LCD control bits RS, R/W, LCD data byte. LCD1 and 2 select bits = 0.

Access cycle 2: Write LCD control bits LCD1 select, LCD2 select active as required. RS, R/W, LCD data byte values do not change but must be written again.

Access cycle 3 = READ: Read LCD Port if a Read access type, determined by R/W = 1 in first access cycles.

Access cycle 3 = WRITE: Write LCD control bits LCD1 select, LCD2 select idle. RS, R/W, LCD data byte do not change but must be written again. This terminates a Write access sequence.

Access cycle 4 = READ end: Write LCD control bits LCD1 select, LCD2 select idle, RS, R/W bits do not change but must be written again. This terminates a Read access sequence.

Table 1-9. LCD PORT REGISTER Definition (MPC500 core register aligned)

BIT #	FUNCTION
D0	LCD1 select, 80 character or first 80 characters of 160 character module select. 1 = Active
D1	LCD2 select, second 80 characters of 160 character module select. 1 = Active
D2 – D5	N/A, not applied
D6	LCD RS or Register Select. 0 = Command, 1 = Display Data access
D7	LCD Read / Write select. 0 = Write, 1 = Read access
D8 > D15	LCD Data Byte D7 > D0, Write output to LCD if D7 = 0, Read input if D7 = 1.

See LCD Module data sheet for command codes

Example source code for the LCD and Keypad drivers are provided on the Axiom MPC5xx support CD. The LCD Port is assigned on chip select CS3 if enabled with MAP Switch position 6. Memory map offset for the LCD Port = CS3 base + 0x0010.

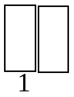
LCD Display CONTRAST

The CONTRAST adjustment allows a contrast Vee voltage to be presented to the LCD_PORT of –5V to +5V DC.

JP2 - LCD_PORT Power Polarity Select

JP2 determines the display power pin polarity on the LCD_PORT. Depending on the type and location of the IDC connector on your display module, the power connections may need to be reversed. Care should be used to verify proper connection and signal matching at the IDC Cable Connector and LCD_PORT.

See the schematic to match this jumper setting to your LCD device connector. Contact support@axman.com for assistance applying a LCD module.



Typical JP2 positions for 80 character or smaller LCD. Rotate 90 degrees for 160 character type modules.

USE CAUTION when connecting your LCD to the LCD-PORT - make sure the power polarity (JP2) and correct placement of the LCD cable so that signals are correctly matched.

1.5.3 User Components

The EVB provides a set of user components that maybe applied in user applications or for testing purposes. The components are interfaced via the J2 Socket Header and are not dedicated to any particular MPC566 I/O signal. Provided are 4 push button switches (SW1 – 4), 4 LED Indicators (LED1 – 4), and the user POT (RV1). Following are J2 connections and interface notes:

Table 1-10. J2 User Component I/O

J2 PIN	COMPONENT	NOTES
1	SW1	Active Low, also drives BRK_EN option
2	SW2	Active high, +5V
3	SW3	Active high, +5V
4	SW4	Active high, +5V
5	LED1	Active High
6	LED2	Active High
7	LED3	Active High
8	LED4	Active High
9	RV1 – Center Tap	0 – 5V
10	RV1 +Input or reference	+5V, open O4 option to change
11	RV1 –Input or reference	Ground, open O5 option to change
12	GROUND	Meter or probe ground / common

BRK_EN Option

The BRK_EN or Break Enable option is provided to allow an IRQ0 NMI interrupt to occur on SW1 being pressed. This feature is provided to support the Monitor ABORT operation to stop user code execution and return to the Monitor command prompt. When the option jumper is installed, depressing SW1 will cause a low active level to be applied to the MPC566 IRQ0.

1.5.4 MPC566EVB Hardware Options

XFC Filter Capacitor C5

Capacitor C5 provides the XFC filtering for the PLL circuits. The capacitor may be changed by the user if PLL locking problems are experienced at the frequency of operation selected or if the reference crystal is replaced. Current value is 3.3nF.

VRH and VRL QADC Reference Supplies

EVB zero ohm resistors R4 and R5 provide connection to MPC566 VDDA and VSSA for VRH and VRL reference signals respectfully. One or both of these resistors can be removed to apply an external reference voltage to the QADC_A Port.

EPEE and BOEPEE CUTAWAY E0

The MPC566EVB board has the EPEE and BOEPEE signals connected by CUT_AWAY pad E0. This connection is for NEXUS port programming of the MPC566 internal flash. This connection will cause the CONFIG_SW position 7 or 8 to enable both signals. If this operation is not desired by the user, cut the CUT-AWAY E0 pad to isolate the signals from each other.

1.5.5 Signals Available on Board

1.5.5.1 IRQ PORT

The IRQ Port provides access to the MPC566 Port C I/O or IRQ inputs on a 10 pin socket header.

SIGNAL	PIN	PIN	SIGNAL
IRQOUT / SGPIO7	10	9	FRZ / SGPIO6
IRQ7	8	7	IRQ6
IRQ5 / SGPIO5	6	5	IRQ4 / SGPIO4
IRQ3 / SGPIO3	4	3	IRQ2 / SGPIO2
IRQ1 / SGPIO1 (IRQ_100)	2	1	IRQ0 / SGPIO0 (BRK_EN)

1.5.5.2 BUS_PORT

The BUS Port provides the data and address line access to the MPC566 memory bus on a 60 pin header.

BUS PORT

SIGNAL	PIN	PIN	SIGNAL
GND	60	59	+2.6V
A31	58	57	A30
A29	56	55	A28

BUS PORT (Continued)

SIGNAL	PIN	PIN	SIGNAL
A27	54	53	A26
A25	52	51	A24
A23	50	49	A22
A21	48	47	A20
A19	46	45	A18
A17	44	43	A16
A15	42	41	A14
A13	40	39	A12
A11	38	37	A10
A9	36	35	A8
D31	34	33	D30
D29	32	31	D28
D27	30	29	D26
D25	28	27	D24
D23	26	25	D22
D21	24	23	D20
D19	22	21	D18
D17	20	19	D16
D15	18	17	D14
D13	16	15	D12
D11	14	13	D10
D9	12	11	D8
D7	10	9	D6
D5	8	7	D4
D3	6	5	D2
D1	4	3	D0
GND	2	1	2.6V

1.5.5.3 TPU_PORTs[1]

The TPU (Timing Processor Unit) Ports provide access to the MPC566 TPU A, B and C channels on 3 identical socket headers, TPU PORT A, TPU PORT B and TPU PORT C.

TPU PORT

SIGNAL	PIN	PIN	SIGNAL
GND	20	19	GND
+5V	18	17	T2CLK
TPU CH15	16	15	TPU CH14
TPU CH13	14	13	TPU CH12
TPU CH11	12	11	TPU CH10
TPU CH9	10	9	TPU CH8
TPU CH7	8	7	TPU CH6
TPU CH5	6	5	TPU CH4
TPU CH3	4	3	TPU CH2
TPU CH1	2	1	TPU CH0

1.5.5.4 CONTROL_PORT

The CONTROL Port provides access to the MPC566 chip selects, bus controls, resets, clocks, and other signals on a 40 pin header.

CONTROL PORT

SIGNAL	PIN	PIN	SIGNAL
ALTREF	40	39	$\overline{TRST}$
BOEPEE	38	37	EPEE
$\overline{BR}$	36	35	TMS
TSIZ1	34	33	$\overline{BB}$
TSIZ0	32	31	$\overline{BG}$
$\overline{TA}$	30	29	$\overline{TEA}$
$\overline{BI}$	28	27	$\overline{TS}$
$\overline{BURST}$	26	25	$\overline{BDIP}$
	24	23	RSTCONF*

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A

CONTROL PORT (Continued)

SIGNAL	PIN	PIN	SIGNAL
$\overline{\text{SRESET}}$	22	21	
EXTCLK	20	19	$\overline{\text{HRESET}}$
GND	18	17	$\overline{\text{PORESET}}$
ENGCLK	16	15	CLKOUT
GND	14	13	GND
$\overline{\text{OE}}$	12	11	$\overline{\text{RD\_WR}}$
$\overline{\text{WE3}}$	10	9	$\overline{\text{CS3}}$
$\overline{\text{WE2}}$	8	7	$\overline{\text{CS2}}$
$\overline{\text{WE1}}$	6	5	$\overline{\text{CS1}}$
$\overline{\text{WE0}}$	4	3	$\overline{\text{CS0}}$
3.3V	2	1	3.3V

1.5.5.5 MIOS_PORT

The MIOS Port provides access to the MPC566 MIOS14 module Timer and I/O signals on a 34 pin socket header. The port has many multiplexed pins so the auxiliary signal connections are also provided for reference.

MIOS PORT

AUX SIGNAL	SIGNAL	PIN	PIN	SIGNAL	AUX SIGNAL
	MGPIO15	34	33	MGPIO14	CAN_C RX
CAN_C TX	MGPIO13	32	31	MGPIO12	
	MGPIO11	30	29	MGPIO10	NEXUS MDO_4
NEXUS MDO_5	MGPIO9	28	27	MGPIO8	NEXUS MDO_6
NEXUS MDO_7	MGPIO7	26	25	MGPIO6	
	MGPIO5	24	23	MGPIO4	BDM VFLS0 (V4 option)
BDM VFLS1 (V3 option)	MGPIO3	22	21	MGPIO2	
	MGPIO1	20	19	MGPIO0	
	MPWM19	18	17	MPWM18	
	MPWM17	16	15	MPWM16	
	MPWM3	14	13	MPWM2	

MIOS PORT (Continued)

AUX SIGNAL	SIGNAL	PIN	PIN	SIGNAL	AUX SIGNAL
	MPWM1	12	11	MPWM0	
	MDA31	10	9	MDA30	
	MDA29	8	7	MDA28	
	MDA27	6	5	MDA15	
	MDA14	4	3	MDA13	
	MDA12	2	1	MDA11	

1.5.5.6 QADC_PORTs[1]

The QADC (Qued Analog to Digital Converter) Ports provide access the MPC566 QADC A and B channels on to socket headers, QADC_A and QADC_B.

QADC_A

SIGNAL	PIN	PIN	SIGNAL
VRL	20	19	VRH
ETRIG2	18	17	ETRIG1
A_PQB7/AN51	16	15	A_PQA7/AN59
A_PQB6/AN50	14	13	A_PQA6/AN58
A_PQB5/AN49	12	11	A_PQA5/AN57
A_PQB4/AN48	10	9	A_PQA4/AN56
A_PQB3/AN3	8	7	A_PQA3/AN55
A_PQB2/AN2	6	5	A_PQA2/AN54
A_PQB1/AN1	4	3	A_PQA1/AN53
A_PQB0/AN0	2	1	A_PQA0/AN52

QADC_B

SIGNAL	PIN	PIN	SIGNAL
AN87	24	23	AN86
AN85	22	21	AN84

---

[1] The MPC535/6 has limited or no functionality for this module. See Appendix A.

QADC_B (Continued)

SIGNAL	PIN	PIN	SIGNAL
AN83	20	19	AN82
AN81	18	17	AN80
B_PQB7/AN51 (Keypad)	16	15	B_PQA7/AN59 (Keypad)
B_PQB6/AN50 (Keypad)	14	13	B_PQA6/AN58 (Keypad)
B_PQB5/AN49 (keypad)	12	11	B_PQA5/AN57 (Keypad)
B_PQB4/AN48 (Keypad)	10	9	B_PQA4/AN56 (Keypad)
B_PQB3/AN3	8	7	B_PQA3/AN55
B_PQB2/AN2	6	5	B_PQA2/AN54
B_PQB1/AN1	4	3	B_PQA1/AN53
B_PQB0/AN0	2	1	B_PQA0/AN52

1.5.5.7 QSM_PORTs

The QSM (Queued Serial Module) Ports provide access to the MPC566 QSM (A and B) I/O ports, SCI ports, SPI ports, and CAN ports on a 16 pin socket header. The signals are provided directly from the MPC566. User should note other connections and options for the communication interfaces mentioned earlier in the manual before applying this port.

QSM_A

EVB USE	SIGNAL	PIN	PIN	SIGNAL	EVB USE
	GND	16	15	+5V	
CAN_A	A_CANRX	14	13	A_CANTX	CAN_A
COM2	A_RXD2	12	11	A_RXD1	COM1
COM2	A_TXD2	10	9	A_TXD1	COM1
	A_ECK	8	7	A_SCK	POWER OAK
POWER OAK	A_MOSI	6	5	A_MISO	POWER OAK
	A_PCS3	4	3	A_PCS2	
POWER OAK	A_PCS1	2	1	A_PCS0 / $\overline{\text{SS}}$	

QSM_B

EVB USE	SIGNAL	PIN	PIN	SIGNAL	EVB USE
	GND	16	15	+5V	

QSM_B (Continued)

EVB USE	SIGNAL	PIN	PIN	SIGNAL	EVB USE
CAN_B	B_CANRX	14	13	B_CANTX	CAN_B
COM4	B_RXD2	12	11	B_RXD1	COM3
COM4	B_TXD2	10	9	B_TXD1	COM3
	B_ECK	8	7	B_SCK	
	B_MOSI	6	5	B_MISO	
	B_PCS3	4	3	B_PCS2	
	B_PCS1	2	1	B_PCS0 / $\overline{\text{SS}}$	

1.5.5.8 MICTOR 1 – 3 PORTs

The Mictor Ports are not installed at the factory but are available to apply the HP logic analysis system. The Mictor 1 – 3 positions provide address and data bus connections for the HP system. See schematic sheet 3 for details of the connections.

1.6 Reference Documents

The Provided MPC5xx support CD contains many documents that the user will find valuable during development. Following is a partial list:

MPC566EVB_B_SCH.PDF - MPC566EVB schematic diagrams

PC33394_P24.pdf - Power Oak data sheet

LAN91C111.pdf - Ethernet controller data sheet

AM28BDD160.pdf - Flash Memory Data sheet and Errata List from AMD

CY1338.pdf - SRAM data sheet

MPC566 User Manual - MPC566 device user guide

1.7 Software Development

The provided dBUG monitor/utility software initializes the clock to run this board at 56 MHz on power-up. The user can set this by changing the PLL Registers of the MPC566 in software.

Software development on the MPC566EVB is best performed using a development tool connected to the BDM-PORT or NEXUS connector. This provides real-time access to all hardware, peripherals and memory on the board. Development tool software provides high-level (C/C++) source code debugging. The Monitor installed will provide Assembly or Object source level debugging. User should review the demonstration and sample software tools provided separately in the MPC566EVB kit for high level debug capability.

The development environment and procedure for best success is to place software to be tested into RAM memory. Execute software to be tested under Monitor or development tool control, then program into FLASH memory to execute new application when power is applied.

Chapter 2 Initialization and Setup

2.1 System Configuration

The MPC566 board requires the following items for minimum system configuration:

- The MPC566EVB board (provided).
- Power supply (provided).
- RS232 compatible terminal or a PC with terminal emulation software.
- RS232 Communication cable (provided).

Figure 2-1 displays the minimum system configuration.

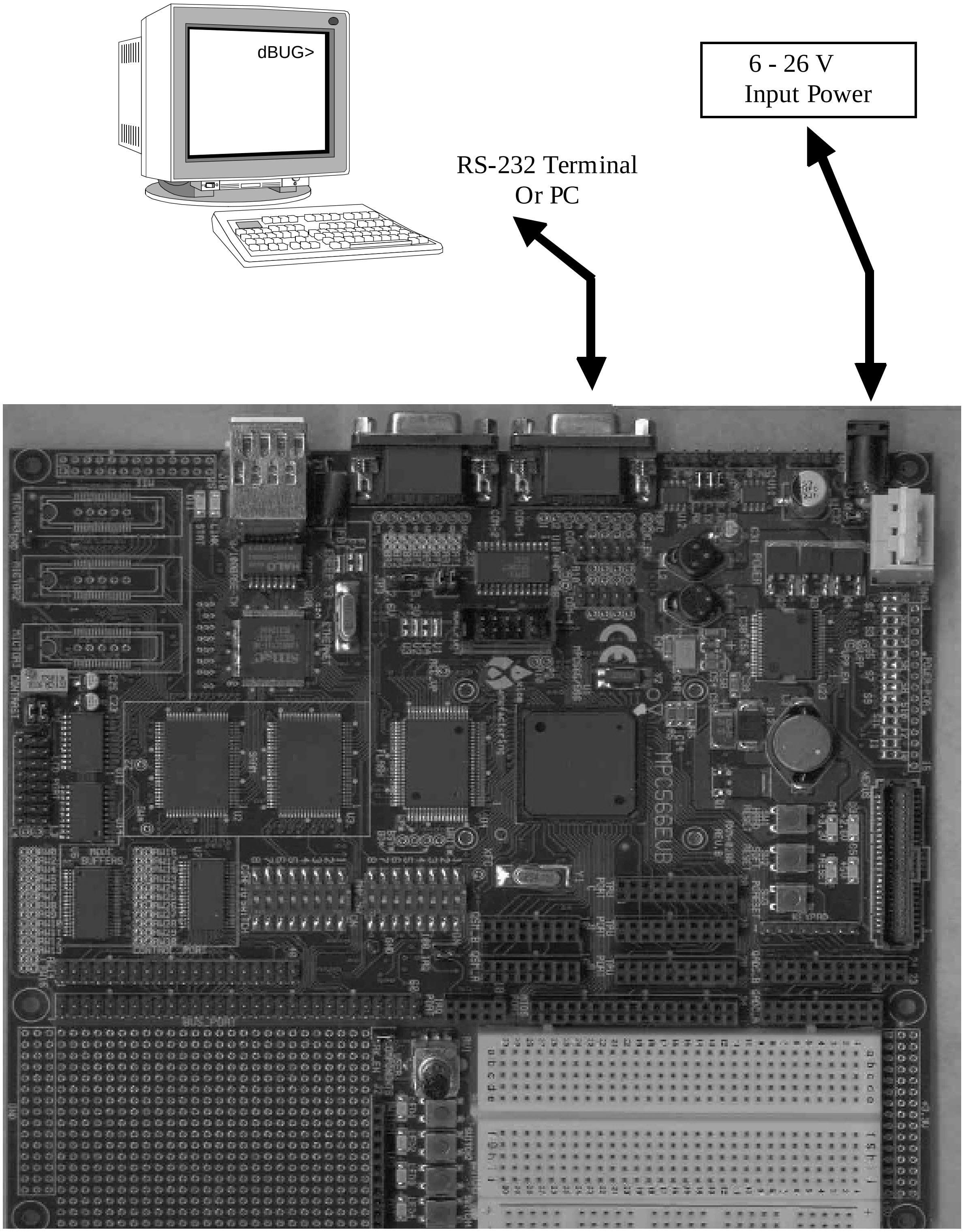

dBUG>
RS-232 Terminal
Or PC
6 - 26 V
Input Power
MPC566EVB

Figure 2-1. Minimum System Configuration

2.2 Installation And Setup

The following sections describe all the steps needed to prepare the board for operation. Please read the following sections carefully before using the board. When you are preparing the board for the first time, be sure to check that all jumpers and switches are in the default locations. Default jumper and switch settings are documented on the master jumper table (see Table 1-2). After the board is functional in its default mode, the Ethernet interface may be used by following the instructions provided in Appendix B, “Configuring dBUG for Network Downloads.

2.2.1 Unpacking

Unpack the computer board from its shipping box. Save the box for storing or reshipping. Refer to the following list and verify that all the items are present. You should have received:

- MPC566EVB Single Board Computer
- MPC566EVB User's Manual (this document)
- One RS232 9-pin Serial Cable
- One CAT5E Ethernet cable, crossover type
- One P&E Micro BDM (Background Debug Mode) “wiggler” cable
- MPC5xx support CD with all documents, drawings, source codes, examples, AxIDE software, and GNU compiler.
- 25-pin parallel cable (DB32 M/F) for BDM
- A selection of demo Third Party Developer Tools and Literature

NOTE:

> Avoid touching the MOS devices. Static discharge can and will damage these devices.

Once you have verified that all the items are present, remove the board from its protective jacket and anti-static bag. Check the board for any visible damage. Ensure that there are no broken, damaged, or missing parts. If you have not received all the items listed above or they are damaged, please contact Axiom manufacturing immediately - for contact details please see the front of this manual.

2.2.2 Preparing the Board for Use

The board, as shipped, is ready to be connected to a terminal and power supply without any need for modification. Figure 2-3, “Jumper Locations”, shows the position of the jumpers and connectors.

2.2.3 Providing Power to the Board

The board accepts two means of power supply connection, either PWR or TB1. Connector PWR is a 2.1mm power jack and TB1 is a power clamp connector with inputs of +12VDC, Ground, and an IGN (ignition) signal. If not using the provided power supply, the board accepts a supply voltage of 4 - 26VDC (current 300ma typical at 12VDC input).

The Motorola MPC500 Family companion power supply Power Oak (PC33394) is provided on the EVB board. This device provides many features designed for automotive applications but may also be useful for industrial or general purpose applications. See the PC33394 data sheet for full description of features, operation, and capability.

The Power Oak provides a regulated switch mode power source for the 2.6V, 3.3V, and 5V supplies on the EVB board. The basic application of the Power Oak provides the Reset signals, back-up supplies for the KAPWR and VDDSRAMto the MPC566, and the CAN A port transceiver. Many options are available on the board and some under software control via the MPC566 QSPI port to modify the Power Oak application.

The Power Port provides tap points for the power supplies and switched power outputs from the Power Oak. Following is the Power Port pin assignment:

Table 2-1. Power Port Connections

PIN	SIGNAL
1	+VIN DC supply
2	+5V
3	+3.3V
4	+2.6V
5	KAPWR
6	VDDSRAM1
7	VDDSRAM2
8	VDDSRAM3
9	VDDSYN
10	VDDRTC
11	VREF1
12	VREF2
13	VREF3
14	VPRE
15	VSEN

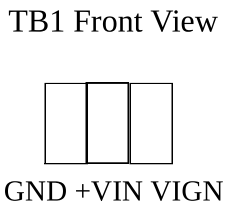
Refer to Power Oak document for application of the provided power sources.

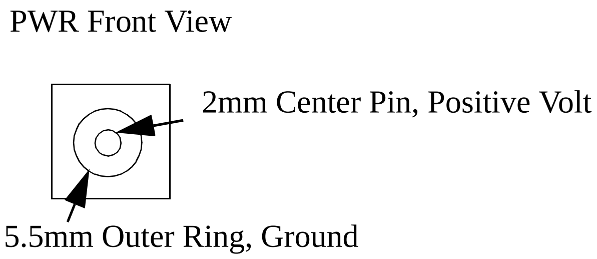
TB1, ON Option, and PWR

TB1 Provides main input power source access and the Power Oak VIGN input signal. The +VIN and ground sources may be applied or tapped from the TB1 pin 2 and 3 connections respectfully. TB1 pin 1 provides access the VIGN signal to the Power Oak, user should note ON jumper operation in the following text.

The ON jumper provides a constant enable to the VIGN signal so the Power Oak is enabled upon power application. The ON option jumper may be removed to apply the VIGN signal externally with a switch or other source. Basic operation of the VIGN signal is to apply +VIN to enable the Power Oak supplies.

PWR jack connection provides power input from the supplied wall plug power source. Typical input is +12VDC. The PWR jack provides a center positive terminal and ground outside sleeve. The jack accepts 2.1mm inside x 5.5mm outside power plugs.





2.2.4 Selecting Terminal Baud Rate

The serial channel SCI0 of the MPC566 is used for serial communication and has a built in baud rate generator. A number of baud rates can be programmed. On power-up or manual RESET, the dBUG ROM monitor firmware configures the channel for 19200 baud. Once the dBUG ROM monitor is running, a SET command may be issued to select any baud rate supported by the ROM monitor. Refer to Chapter 3 for the discussion of this command.

2.2.5 The Terminal Character Format

The character format of the communication channel is fixed at power-up or RESET. The default character format is 8 bits per character, no parity and one stop bit with no flow control. It is neccessary to ensure that the terminal or PC is set to this format.

2.2.6 Connecting the Terminal

The board is now ready to be connected to a PC/terminal. Use the RS232 serial cable to connect the PC/terminal to the MPC566EVB at COM-1. The cable has a 9-pin female D-sub terminal connector at one end and a 9-pin male D-sub connector at the other end. Connect the 9-pin male connector to connector COM-1 on the MPC566EVB board. Connect the 9-pin female connector to one of the available serial communication channels normally referred to as COM1 (COM2, etc.) on the PC running terminal emulation software (such as AxIDE, TeraTerm or Hyperterminal). The connector on the PC/terminal may be either male 25-pin or 9-pin. It may be neccessary to obtain a 25-pin-to-9-pin adapter to make this connection. If an adapter is required, refer to Figure 2-2 which shows the pin assignment for the 9-pin connector on the board.

2.2.7 Using a Personal Computer as a Terminal

A personal computer may be used as a terminal provided a terminal emulation software package is available. Examples of this software are AxIDE, TeraTerm, PROCOMM, KERMIT, QMODEM, Windows 95/98/2000 HyperTerminal or similar packages. The board should then be connected as described in Section 2.2.6, “Connecting the Terminal.”

Once the connection to the PC is made, power may be applied to the PC and the terminal emulation software can be run. In terminal mode, it is neccessary to select the baud rate and character format for the channel. The character format should be 8 bits, no parity, one stop bit. (see Section 2.2.5, “The Terminal Character Format”.) The baud rate should be set to 19200 and XON/XOFF flow control should be turn on. Power can now be applied to the board.

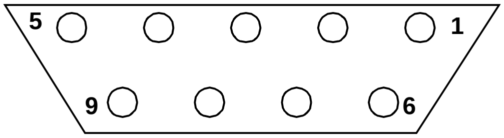


Figure 2-2. Pin assignment for female (Terminal) connector.

Pin assignments are as follows.

Pin 1, 4, and 6 = group connected for DTR/DSR flow control null back to host

Pin 2 = TXD output (RS232 level)

Pin 3 = RXD input (RS232 level)

Pin 5 = Ground/Vss/Common

Pin 7 and 8 = group connected for RTS/CTS flow control null back to host

Pin 9 = open

2.3 MPC566EVB Jumper and Switch Setup

Jumper settings are as follows:

Note ‘*’ is used to indicate that default setting.
‘**’ is used to indicate mandatory setting for proper operation.

Table 2-2. Jumper Settings

Jumper Setting		Function
JP1	* 1-2,3-4	COM2: TX=Pin3, RX=Pin2 (see silk screen on bottom of board)
	1-3,2-4	COM2: TX=Pin2, RX=Pin3 (see silk screen on bottom of board)
JP2		Determines the display power pin polarity on the LCD_PORT. See the schematic
JP3	*1-2 (closest to BDM_PORT)	BDM being used runs at 3.3 volts. So the signals need to go through a level shifter to get 2.6 volts to the part.

Table 2-2. Jumper Settings (Continued)

Jumper Setting		Function
	2-3	BDM being used runs at 2.6 volts.
BRK_EN	*inserted	SWITCH1 will act as a debounced Non-Maskable Interrupt (ABORT)
	removed	SWITCH1 is for user use
B_RX	*insterted	CAN: see Section 1.4.2, “CAN PORTs and Options
	removed	CAN: see Section 1.4.2, “CAN PORTs and Options
C_TX	inserted	CAN: see Section 1.4.2, “CAN PORTs and Options
	*removed	CAN: see Section 1.4.2, “CAN PORTs and Options
C_RX	inserted	CAN: see Section 1.4.2, “CAN PORTs and Options
	*removed	CAN: see Section 1.4.2, “CAN PORTs and Options
100_IRQ	*inserted	Ties the interrupt line from the Ethernet chip to the interrupt line of the processor
	removed	Disconnects the interrupt line from the ethernet chip to the interrupt line of the processor
“ON”	* insterted	Allows the board to have power.
	removed	Power to the board should be controlled by an ignition switch connected to the IGN input on the Power Clamp, TB1.

Figure 2-3 on the next page shows the jumper locations for the board.

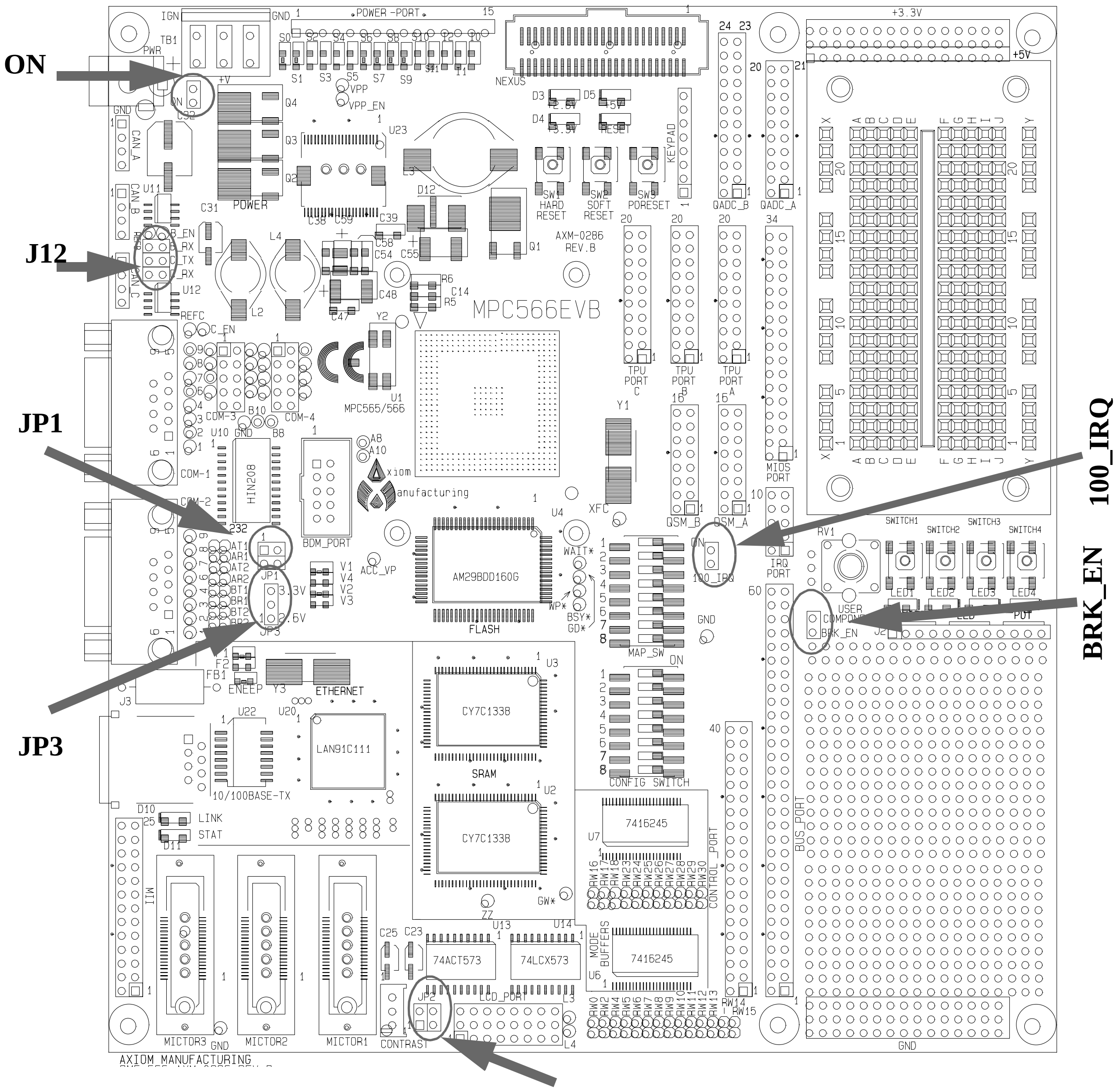

ON
J12
JP1
JP3
100_IRQ
BRK_EN
MPC566EVB
POWER
CONFIG_SWITCH
MAP_SW
FLASH
SRAM
ETHERNET
BDM_PORT
NEXUS
SW1 HARD RESET
SW2 SOFT RESET
SW3 PORESET
QADC_B
QADC_A
TPU PORT C
TPU PORT B
TPU PORT A
QSM_B
QSM_A
MIOS PORT
IRQ PORT
BUS_PORT
CONTROL_PORT
MODE BUFFERS
LCD_PORT
MICTOR1
MICTOR2
MICTOR3
CONTRAST
JP2
AXIOM MANUFACTURING

Figure 2-3. Jumper Locations on the Board

2.3.1 Reset Configuration Word and Configuration Switch (CONFIG_SW)

Configuration Switch provides several key external Reset Configuration Word (RCW)options and the programming enable options for programming the MPC566 internal flash memory. These switches provide a logic 0 or low level when off and a logic 1 or high level when on. The RCW configuration options are only presented to the data bus during Hard Reset if enabled by CONFIG switch position 1. Note that MAP switch also has positions (5 and 8) that are part of the Reset Configuration Word. All other RCW options can be located on the RW test pads, see RW options for description of how to apply. The Reset Configuration Word options are all defaulted to logic 0 when the CONFIG Switches are in the OFF position. For more information on the MPC566 Reset Configuration Word refer to the Reset Chapter in the MPC566 users manual. Five bits are available on CONFIG_SW (see Table 2-3).I

Table 2-3. CONFIG_SW

Switch	Name	MPC565/566 Function	Function	
1	CFG_EN	RST_CONF	Enable External Config: This bit enables the external reset config	0 = Internal Reset Config Word selected 1 = External Reset Config Word selected (default)
2	CFG1	BDIS	Data Bus 3/Boot Dis: This bit enables booting from external flash memory	0 = Boot from internal memory (default) 1 = Boot from external flash
3	CFG2	ETRE	This switch controls the Exception Table Relocation feature	0 = Exception table relocation is off (default) 1 = Exception table relocation is on
4	CFG3	COMP1	This switch enables Compression Mode	0 = “Decompression ON” mode is disabled (default) 1 = “Decompression ON” mode is enabled
5	CFG4	COMP2	This switch controls whether Exception vector code is compressed	0 = MPC566 assumes that the exception routines are non-compressed (default) 1 = MPC566 assumes that all exception routines are compressed
6	CFG5	DME	This switch enables Dual Mapping	0 = Dual Mapping disabled (default) 1 = Dual Mapping enabled
7	EPEE	EPEE	This pin enables erasing and programming of the internal MPC566 flash	0 = Internal Flash P/E disabled 1 = Internal Flash P/E enabled (default)
8	B0EPEE	B0EPEE	This switch enables programming or erasing of BLock 0 the internal flash	0 =P/E of Block 0 disabled 1 = P/E of Block 0 enabled (default)

Notes:
1) Position 1 should be ON at all times until the MPC566 internal flash RCW word is programmed. This switch will cause the external value RCW to override the internal flash value of the RCW word.
2) Position 7 and 8 must be enabled for MPC566 internal flash erase or programming operations.
3) All other external RCW bits are provided with the RW hardware options.

RW0 – 30: External Reset Configuration Word (RCW) Options

RW0, RW2, RW4 – 18, RW23 – 30 provide the user access to external Reset Configuration Word (RCW) bits not normally required for default MPC566EVB operation. The RW0 – 30 designations reflect the data bus D0 – D30 bit affected when the RCW word is enabled externally. All RW0 – 30 option bits are defaulted to the logic low value during external RCW word operation. The user may apply a wire jumper between the 2 pad positions of each RW0 – 30 option to provide a logic high level on the respective bit position during external RCW operation. Note that the user could add physical headers (2mm type) to these signals on the board. Refer to the MPC566 user manual Reset chapter for the respective RCW bit definitions.

2.3.2 Memory Configuration (MAP_SW)

The Memory Configuration switch (MAP_SW) controls the mapping of the MPC566 chip selects to the different external memories and whether the internal flash is enabled.I

Table 2-4. MAP_SW

Switch	Name	MPC565/566 Function	Switch Closed	Default
1	CS0[1]	RAM_CS	CS0 connected to the External SRAM	off
2	CS1[1]	RAM_CS	CS1 connected to the External SRAM	on
3	CS0[1]	FL_CS	CS0 connected to the External Flash	on
4	CS2[1]	FL_CS	CS1 connected to the External Flash	off
5	IP	RCW-IP	IP bit is set	on
6	CS3[1]	100_CS/LCD_CS	CS3 connected to the Ethernet and LCD Interface	on
7	CS3-TA	TA	TA for LCD or Connect the EtherNet Controller TA to the Processor TA	on
8	2.6v	FLEN	Enable Internal MPC565/566 Flash	off

[1] Each Chip Select should only be connected to one device.

2.4 System Power-up and Initial Operation

When all of the cables are connected to the board, power may be applied. The dBUG ROM Monitor initializes the board and then displays a power-up message on the terminal.

```
Part Number: 0xXX
MaskNum: 0xXX

Copyright 1995-2002 Motorola, Inc.  All Rights Reserved.
MPC566 MPC566EVB Firmware v3b.1a.xx (Build XXX on XXX  XX 20XX
xx:xx:xx)

Enter 'help' for help.
```

```
dBUG>
```

The board is now ready for operation under the control of the debugger as described in Chapter 3. If you do not get the above response, perform the following checks:

1. Make sure that the power supply is properly configured for polarity, voltage level and current capability (~300mA) and is connected to the board.
2. Check that the terminal and board are set for the same character format and baud.
3. Check that your switches are set to their default settings.
4. Press the RESET button to insure that the board has been initialized properly.

If you still are not receiving the proper response, your board may have been damaged. Contact Axiom Manufacturing for further instructions, please see the beginning of this manual for contact details.

Chapter 3
Using the Monitor/Debug Firmware

The MPC566EVB single board computer has a resident firmware package that provides a self-contained programming and operating environment. The firmware, named dBUG, provides the user with monitor/debug interface, inline assembler and disassembly, program download, register and memory manipulation, and I/O control functions. This chapter is a how-to-use description of the dBUG package, including the user interface and command structure.

3.1 What Is dBUG?

dBUG is a traditional ROM monitor/debugger that offers a comfortable and intuitive command line interface that can be used to download and execute code. It contains all the primary features needed in a debugger to create a useful debugging environment.

The firmware (stored in the upper 1MByte of the Flash ROM device) provides a self-contained programming and operating environment. dBUG interacts with the user through pre-defined commands that are entered via the terminal. These commands are defined in Section 3.4, “Commands”.

The user interface to dBUG is the command line. A number of features have been implemented to achieve an easy and intuitive command line interface.

dBUG assumes that an 80x24 character dumb-terminal is utilized to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit, 8N1 with no flow control. The default baud rate is 19200 but can be changed after the power-up.

The command line prompt is “dBUG> “. Any dBUG command may be entered from this prompt. dBUG does not allow command lines to exceed 80 characters. Wherever possible, dBUG displays data in 80 columns or less. dBUG echoes each character as it is typed, eliminating the need for any “local echo” on the terminal side.

In general, dBUG is not case sensitive. Commands may be entered either in upper or lower case, depending upon the user’s equipment and preference. Only symbol names require that the exact case be used.

Most commands can be recognized by using an abbreviated name. For instance, entering “h” is the same as entering “help”. Thus, it is not necessary to type the entire command name.

The commands DI, GO, MD, STEP and TRACE are used repeatedly when debugging. dBUG recognizes this and allows for repeated execution of these commands with minimal typing. After a command is entered, simply press <RETURN> or <ENTER> to invoke the command again. The command is executed as if no command line parameters were provided.

An additional function called the "System Call" allows the user program to utilize various routines within dBUG. The System Call is discussed at the end of this chapter.

The operational mode of dBUG is demonstrated in Figure 3-1. After the system initialization, the board waits for a command-line input from the user terminal. When a proper command is entered, the operation continues in one of the two basic modes. If the command causes execution of the user program, the dBUG firmware may or may not be re-entered, at the discretion of the user’s program. For the alternate case, the command will be executed under control of the dBUG firmware, and after command completion, the system returns to command entry mode.

During command execution, additional user input may be required depending on the command function.

For commands that accept an optional <width> to modify the memory access size, the valid values are:

- B 8-bit (byte) access
- H 16-bit (half-word) access
- W 32-bit (word) access

When no <width> option is provided, the default width is .W, 32-bit.

The core MPC500 register set is maintained by dBUG. These are listed below:

- GPR0-GPR31
- IP (SRR0 is IP)
- MSR (SRR1 is MSR)
- CR, XER, LR, CTR, DSISR, DAR, DEC

All control registers on MPC500 core are not readable by the supervisor-programming model, and thus not accessible via dBUG. User code may change these registers, but caution must be exercised as changes may render dBUG inoperable.

A reference to “SP” (stack pointer) actually refers to general purpose address register one, “GPR1."

3.2 Operational Procedure

System power-up and initial operation are described in detail in Chapter 2. This information is repeated here for convenience and to prevent possible damage.

3.2.1 System Power-up

- Be sure the power supply is connected properly prior to power-up.
- Make sure the terminal is connected to RS232 COM-1 connector.

- Make sure the IP bit is set (switch 5 ON in MAP_SW). This will cause the board to boot out of external flash (where the dBUG code resides).
- Turn power on to the board.

Figure 3-1 shows the dUBG operational mode.

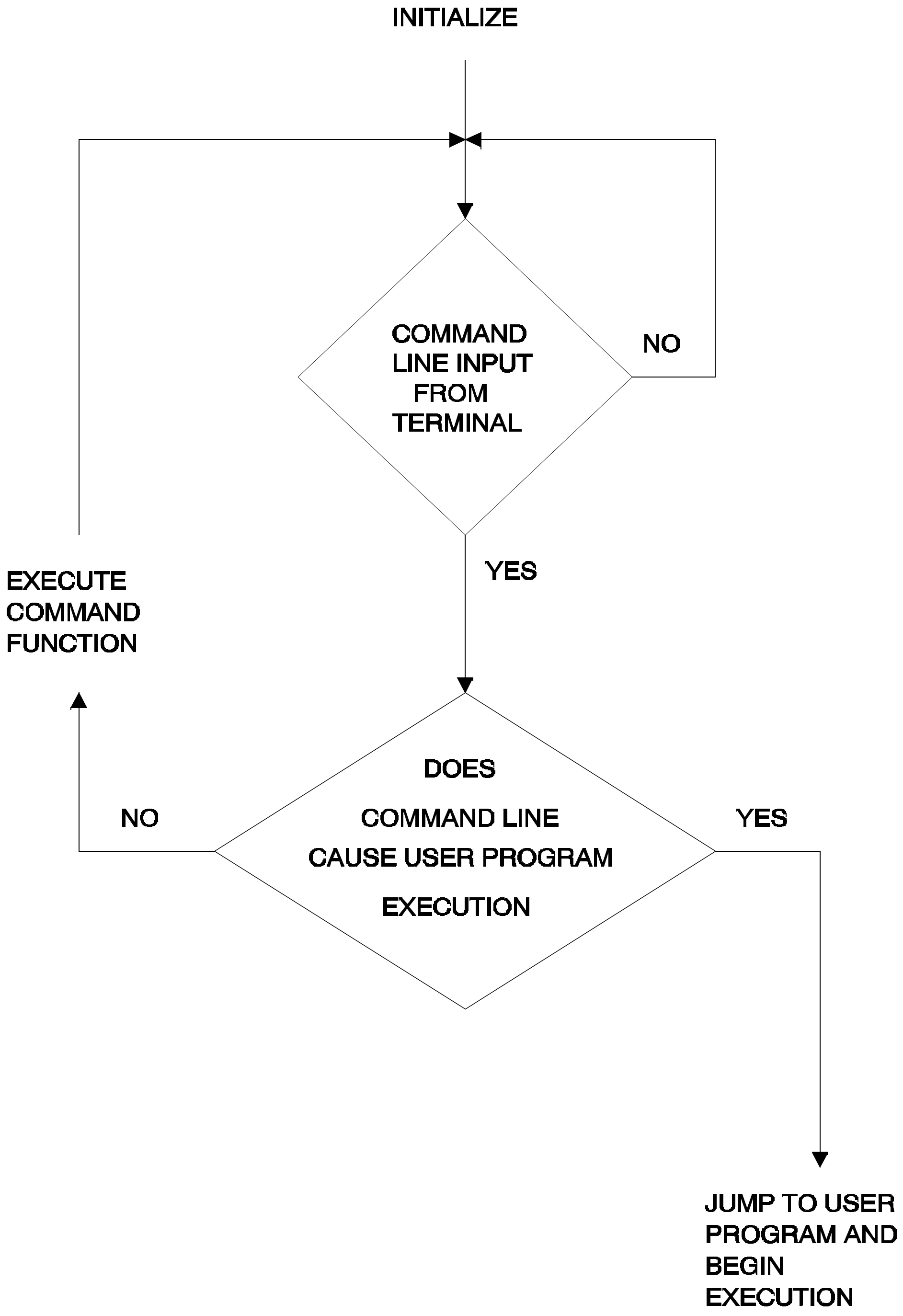

INITIALIZE
COMMAND LINE INPUT FROM TERMINAL
NO
YES
EXECUTE COMMAND FUNCTION
DOES COMMAND LINE CAUSE USER PROGRAM EXECUTION
NO
YES
JUMP TO USER PROGRAM AND BEGIN EXECUTION

Figure 3-1. Flow Diagram of dBUG Operational Mode.

3.2.2 System Initialization

The act of powering up the board will initialize the system. The processor is reset and dBUG is invoked.

dBUG performs the following configurations of internal resources during the initialization. The IP bit is set by default, placing the vector table at 0xFFF0_0000 (external SRAM). To take over an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at 0xFFF0_0000.

The Software Watchdog Timer is disabled and internal timers are placed in a stop condition. Interrupt controller registers are initialized with unique interrupt level/priority pairs. Please refer to the dBUG source files on the PowerPC website (www.motorola.com/powerpc) for the complete initialization code sequence.

After initialization, the terminal will display:

```
Part Number: 0x36
MaskNum: 0x10

Copyright 1995-2002 Motorola, Inc.  All Rights Reserved.
MPC566 MPC566EVB Firmware v2e.1a.xx (Build XXX on XXX  XX 20XX xx:xx:xx)


Enter 'help' for help.



dBUG>
```

If you did not get this response check the setup, refer to Section 2.4, “System Power-up and Initial Operation”.

Other means can be used to re-initialize the MPC566EVB Computer Board firmware. These means are discussed in the following paragraphs.

3.2.2.1 Hard RESET Button.

Pressing the Hard RESET button (SW1-HARD RESET) causes all processes to terminate, resets the MPC566 processor and board logic and restarts the dBUG firmware. Pressing the RESET button would be the appropriate action if all else fails.

3.2.2.2 Non-Maskable Interrupt Button.

SWITCH1 can be used as a non-maskable interrupt button. It is available for the user to use in their code as an input if the jumper BRK_EN is removed. The NMI function causes an interrupt of the present processing (a level 0 interrupt on MPC566) and gives control to the dBUG firmware. This action differs from RESET in that no processor register or memory contents are changed, the processor and peripherals are not reset, and dBUG is not restarted. Also, in response to depressing the NMI button, the contents of the MPC566 core internal registers are displayed.

The NMI function is most appropriate when software is being debugged. The user can interrupt the processor without destroying the present state of the system. This is accomplished by forcing a non-maskable interrupt that will call a dBUG routine that will save the current state of the registers to shadow registers in the monitor for display to the user. The user will be returned to the ROM monitor prompt after exception handling.

3.2.2.3 Software Reset Command.

dBUG does have a command that causes dBUG to restart as if a hardware reset was invoked. The command is "RESET".

3.3 Command Line Usage

The user interface to dBUG is the command line. A number of features have been implemented to achieve an easy and intuitive command line interface.

dBUG assumes that an 80x24 ASCII character dumb terminal is used to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit (8N1). The baud rate default is 19200 bps — a speed commonly available from workstations, personal computers and dedicated terminals.

The command line prompt is: dBUG>

Any dBUG command may be entered from this prompt. dBUG does not allow command lines to exceed 80 characters. Wherever possible, dBUG displays data in 80 columns or less. dBUG echoes each character as it is typed, eliminating the need for any local echo on the terminal side.

The <Backspace> and <Delete> keys are recognized as rub-out keys for correcting typographical mistakes.

Command lines may be recalled using the <Control> U, <Control> D and <Control> R key sequences. <Control> U and <Control> D cycle up and down through previous command lines. <Control> R recalls and executes the last command line.

In general, dBUG is not case-sensitive. Commands may be entered either in uppercase or lowercase, depending upon the user’s equipment and preference. Only symbol names require that the exact case be used.

Most commands can be recognized by using an abbreviated name. For instance, entering h is the same as entering help. Thus it is not necessary to type the entire command name.

The commands DI, GO, MD, STEP and TRACE are used repeatedly when debugging. dBUG recognizes this and allows for repeated execution of these commands with minimal typing. After a command is entered, press the <Return> or <Enter> key to invoke the command again. The command is executed as if no command line parameters were provided.

3.4 Commands

This section lists the commands that are available with all versions of dBUG. Some board or CPU combinations may use additional commands not listed below.

Table 3-1. dBUG Command Summary

MNEMONIC	SYNTAX	DESCRIPTION
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
BM	bm begin end dest	Block Move
BR	br addr <-r> <-c count> <-t trigger>	Breakpoint
BS	bs <width> begin end data	Block Search
DC	dc value	Data Convert
DI	di<addr>	Disassemble
DL	dl <offset>	Download Serial
DLDBUG	dldbug	Download dBUG
DN	dn <-c> <-e> <-i> <-s <-o offset>> <filename>	Download Network

Table 3-1. dBUG Command Summary

MNEMONIC	SYNTAX	DESCRIPTION
FL	fl <command> dest <src> size	Erase/Program External Flash
GO	go <addr>	Execute
GT	gt addr	Execute To
HBR	hbr addr <-r>	Hardware Breakpoint
HELP	help <command>	Help
IRD	ird <module.register>	Internal Register Display
IRM	irm module.register data	Internal Register Modify
LR	lr<width> addr	Loop Read
LW	lw<width> addr data	Loop Write
MD	md<width> <begin> <end>	Memory Display
MM	mm<width> addr <data>	Memory Modify
MMAP	mmap	Memory Map Display
RD	rd <reg>	Register Display
RM	rm reg data	Register Modify
RESET	reset	Reset
SD	sd	Stack Dump
SET	set <option value>	Set Configurations
SHOW	show <option>	Show Configurations
STEP	step	Step (Over)
SYMBOL	symbol <symb> <-a symb value> <-r symb> <-C|l|s>	Symbol Management
TRACE	trace <num>	Trace (Into)
VERSION	version	Show Version

ASM Assembler

Usage: ASM <<addr> stmt>

The ASM command is a primitive assembler. The <stmt> is assembled and the resulting code placed at <addr>. This command has an interactive and non-interactive mode of operation.

The value for address <addr> may be an absolute address specified as a hexadecimal value, or a symbol name. The value for stmt must be valid assembler mnemonics for the CPU.

For the interactive mode, the user enters the command and the optional <addr>. If the address is not specified, then the last address is used. The memory contents at the address are disassembled, and the user prompted for the new assembly. If valid, the new assembly is placed into memory, and the address incremented accordingly. If the assembly is not valid, then memory is not modified, and an error message produced. In either case, memory is disassembled and the process repeats.

The user may press the <Enter> or <Return> key to accept the current memory contents and skip to the next instruction, or a enter period to quit the interactive mode.

In the non-interactive mode, the user specifies the address and the assembly statement on the command line. The statement is the assembled, and if valid, placed into memory, otherwise an error message is produced.

Examples:

To place a NOP instruction at address 0x0001_0000, the command is:

```
asm     10000 nop
```

To interactively assembly memory at address 0x0040_0000, the command is:

```
asm     400000
```

BC Block Compare

Usage: BC addr1 addr2 length

The BC command compares two contiguous blocks of memory on a byte by byte basis. The first block starts at address addr1 and the second starts at address addr2, both of length bytes.

If the blocks are not identical, the address of the first mismatch is displayed. The value for addresses addr1 and addr2 may be an absolute address specified as a hexadecimal value or a symbol name. The value for length may be a symbol name or a number converted according to the user defined radix (hexadecimal by default).

Example:

To verify that the data starting at 0xFFF2_0000 and ending at 0xFFF3_0000 is identical to the data starting at 0xFFF0_0000, the command is:

```
bc          FFF20000 FFF00000 10000
```

BF Block Fill

Usage: BF<width> begin end data <inc>

The BF command fills a contiguous block of memory starting at address begin, stopping at address end, with the value data. <Width> modifies the size of the data that is written. If no <width> is specified, the default of word sized data is used.

The value for addresses begin and end may be an absolute address specified as a hexadecimal value, or a symbol name. The value for data may be a symbol name, or a number converted according to the user-defined radix, normally hexadecimal.

The optional value <inc> can be used to increment (or decrement) the data value during the fill.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples:

To fill a memory block starting at 0xFFF2_0000 and ending at 0xFFF4_0000 with the value 0x1234, the command is:

```
bf          FFF20000 FFF40000 1234
```

To fill a block of memory starting at 0xFFF20000 and ending at 0xFFF4_0000 with a byte value of 0xAB, the command is:

```
bf.b        FFF20000 FFF40000 AB
```

To zero out the BSS section of the target code (defined by the symbols bss_start and bss_end), the command is:

```
bf          bss_start bss_end 0
```

To fill a block of memory starting at 0xFFF2_0000 and ending at 0xFFF4_0000 with data that increments by 2 for each <width>, the command is:

```
bf          FFF20000 FFF40000 0 2
```

BM Block Move

Usage: BM begin end dest

The BM command moves a contiguous block of memory starting at address begin and stopping at address end to the new address dest. The BM command copies memory as a series of bytes, and does not alter the original block.

The values for addresses begin, end, and dest may be absolute addresses specified as hexadecimal values, or symbol names. If the destination address overlaps the block defined by begin and end, an error message is produced and the command exits.

Examples:

To copy a block of memory starting at 0xFFF4_0000 and ending at 0xFFF7_0000 to the location 0xFFF0_0000, the command is:

```
bm          FFF40000 FFF70000 FFF00000
```

To copy the target code’s data section (defined by the symbols data_start and data_end) to 0xFFF0_0000, the command is:

```
bm          data_start data_end FFF00000
```

NOTE:

Refer to “upuser” command for copying code/data into Flash memory.

BR Breakpoints

Usage: BR addr <-r> <-c count> <-t trigger>

The BR command inserts or removes software breakpoints at address addr. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name. Count and trigger are numbers converted according to the user-defined radix, normally hexadecimal.

If no argument is provided to the BR command, a listing of all defined breakpoints is displayed.

The -r option to the BR command removes a breakpoint defined at address addr. If no address is specified in conjunction with the -r option, then all breakpoints are removed.

Each time a breakpoint is encountered during the execution of target code, its count value is incremented by one. By default, the initial count value for a breakpoint is zero, but the -c option allows setting the initial count for the breakpoint.

Each time a breakpoint is encountered during the execution of target code, the count value is compared against the trigger value. If the count value is equal to or greater than the trigger value, a breakpoint is encountered and control returned to dBUG. By default, the initial trigger value for a breakpoint is one, but the -t option allows setting the initial trigger for the breakpoint.

If no address is specified in conjunction with the -c or -t options, then all breakpoints are initialized to the values specified by the -c or -t option.

Examples:

To set a breakpoint at the C function main() (symbol _main; see “symbol” command), the command is:

```
br        _main
```

When the target code is executed and the processor reaches main(), control will be returned to dBUG.

To set a breakpoint at the C function bench() and set its trigger value to 3, the command is:

```
br        _bench -t 3
```

When the target code is executed, the processor must attempt to execute the function bench() a third time before returning control back to dBUG.

To remove all breakpoints, the command is:

```
br        -r
```

BS Block Search

Usage: BS<width> begin end data

The BS command searches a contiguous block of memory starting at address begin, stopping at address end, for the value data. <Width> modifies the size of the data that is compared during the search. If no <width> is specified, the default of word sized data is used.

The values for addresses begin and end may be absolute addresses specified as hexadecimal values, or symbol names. The value for data may be a symbol name or a number converted according to the user-defined radix, normally hexadecimal.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples:

To search for the 32-bit value 0x1234_5678 in the memory block starting at 0xFFF4_0000 and ending at 0xFFF7_0000:

```
bs      FFF40000 FFF70000 12345678
```

This reads the 32-bit word located at 0x0004_0000 and compares it against the 32-bit value 0x1234_5678. If no match is found, then the address is incremented to 0x0004_0002 and the next 32-bit value is read and compared.

To search for the 16-bit value 0x1234 in the memory block starting at 0xFFF4_0000 and ending at 0xFFF7_0000:

```
bs      40000 FFF70000 1234
```

This reads the 32-bit word located at 0xFFF4_0000 and compares it against the 16-bit value 0x0000_1234. If no match is found, then the address is incremented to 0xFFF4_0004 and the next 32-bit value is read and compared.

DC Data Conversion

Usage: DC data

The DC command displays the hexadecimal or decimal value data in hexadecimal, binary, and decimal notation.

The value for data may be a symbol name or an absolute value. If an absolute value passed into the DC command is prefixed by ‘0x’, then data is interpreted as a hexadecimal value. Otherwise data is interpreted as a decimal value.

All values are treated as 32-bit quantities.

Examples:

To display the decimal and binary equivalent of 0x1234, the command is:

```
dc          0x1234
```

To display the hexadecimal and binary equivalent of 1234, the command is:

```
dc          1234
```

DI Disassemble

Usage: DI <addr>

The DI command disassembles target code pointed to by addr. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name.

Wherever possible, the disassembler will use information from the symbol table to produce a more meaningful disassembly. This is especially useful for branch target addresses and subroutine calls.

The DI command attempts to track the address of the last disassembled opcode. If no address is provided to the DI command, then the DI command uses the address of the last opcode that was disassembled.

The DI command is repeatable.

Examples:

To disassemble code that starts at 0xFFF4_0000, the command is:

```
di          FFF40000
```

To disassemble code of the C function main(), the command is:

```
di          _main
```

DL Download Console

Usage: DL <offset>

The DL command performs an S-record download of data obtained from the console typically through a serial port. The value for offset is converted according to the user-defined radix, normally hexadecimal.

If offset is provided, then the destination address of each S-record is adjusted by offset.

The DL command checks the destination download address for validity. If the destination is an address outside the defined user space, then an error message is displayed and downloading aborted. DL can be used for downloading to internal flash, external flash, internal SRAM, and external SRAM.

If the S-record file contains the entry point address, then the program counter is set to reflect this address.

Examples:

To download an S-record file through the serial port, the command is:

```
dl
```

To download an S-record file through the serial port, and add an offset to the destination address of 0x40, the command is:

```
dl          0x40
```

DLDBUG Download dBUG

Usage: DLDBUG

The DLDBUG command will download the dBUG monitor to the MPC566EVB board. First it will erase all sectors of Flash that dBUG occupies, then it will download the code through the serial port. Upon asking if the user is sure they want to do this, the user should respond by typing “yes” if they want to continue. The DLDBUG command will work at baud rates up to and including 57600.

Xon/Xoff flow control needs to be turned on in the terminal window to download data.

To download the dBUG monitor to the board, the command is:

```
dldbug
```

DN Download Network

Usage: DN <-c> <-e> <-i> <-s> <-o offset> <filename>

The DN command downloads code from the network. The DN command handle files which are either S-record, COFF, ELF or Image formats. The DN command uses Trivial File Transfer Protocol (TFTP) to transfer files from a network host.

In general, the type of file to be downloaded and the name of the file must be specified to the DN command. The -c option indicates a COFF download, the -e option indicates an ELF download, the -i option indicates an Image download, and the -s indicates an S-record download. The -o option works only in conjunction with the -s option to indicate an optional offset for S-record download. The filename is passed directly to the TFTP server and therefore must be a valid filename on the server.

If neither of the -c, -e, -i, -s or filename options are specified, then a default filename and filetype will be used. Default filename and filetype parameters are manipulated using the SET and SHOW commands.

The DN command checks the destination download address for validity. If the destination is an address outside the defined user space, then an error message is displayed and downloading aborted.

For ELF and COFF files which contain symbolic debug information, the symbol tables are extracted from the file during download and used by dBUG. Only global symbols are kept in dBUG. The dBUG symbol table is not cleared prior to downloading, so it is the user’s responsibility to clear the symbol table as necessary prior to downloading.

If an entry point address is specified in the S-record, COFF or ELF file, the program counter is set accordingly.

Examples:

To download an S-record file with the name “srec.out”, the command is:

```
dn -s srec.out
```

To download a COFF file with the name “coff.out”, the command is:

```
dn -c coff.out
```

To download a file using the default filetype with the name “bench.out”, the command is:

```
dn bench.out
```

To download a file using the default filename and filetype, the command is:

```
dn
```

FL Erase/Program Flash

Usage: FL

FL (e)rase addr bytes

FL (w)rite dest src bytes

The FL command is used to erase the external flash, write to external flash, and display flash device information. Erase and Write operations must be done in sector blocks. dBUG assumes that the user has erased enough memory before writing to it. The destination address must be word (4byte) aligned and the byte count must be in word (4byte) multiples. To download a .s19 file straight to flash, please see the DL command.

Examples:

To view the flash device information, the command is:

```
fl
```

To erase 0x10000 bytes of flash starting at 0x00800000, the command is:

```
fl erase 800000 10000
```

To copy 0x40 bytes of data from internal SRAM (0x3FA000) to external flash at 0x00800000, the command is:

```
fl      write   800000 3fa000 0x40
```

GO Execute

Usage: GO <addr>

The GO command executes target code starting at address addr. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name.

If no argument is provided, the GO command begins executing instructions at the current program counter.

When the GO command is executed, all user-defined breakpoints are inserted into the target code, and the context is switched to the target program. Control is only regained when the target code encounters a breakpoint, illegal instruction, or other exception which causes control to be handed back to dBUG.

The GO command is repeatable.

Examples:

To execute code at the current program counter, the command is:

```
go
```

To execute code at the C function main(), the command is:

```
go _main
```

To execute code at the address 0x0004_0000, the command is:

```
go 40000
```

GT Execute To

Usage: GT addr

The GT command inserts a temporary software breakpoint at addr and then executes target code starting at the current program counter. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name.

When the GT command is executed, all breakpoints are inserted into the target code, and the context is switched to the target program. Control is only regained when the target code encounters a breakpoint, illegal instruction, or other exception which causes control to be handed back to dBUG.

Examples:

To execute code up to the C function bench(), the command is:

```
gt _bench
```

HELP Help

Usage: HELP <command>

The HELP command displays a brief syntax of the commands available within dBUG. In addition, the address of where user code may start is given. If command is provided, then a brief listing of the syntax of the specified command is displayed.

Examples:

To obtain a listing of all the commands available within dBUG, the command is:

```
help
```

To obtain help on the breakpoint command, the command is:

```
help br
```

IRD Internal Register Display

Usage: IRD <module.register>

This command displays the internal registers of different modules inside the MPC500. In the command line, module refers to the module name where the register is located and register refers to the specific register to display.

The registers are organized according to the module to which they belong. The available modules on the MPC500 are USIU, TPU_A, TPU_B, QADC_A, QADC_B, QSMCM_A, MIOS14, CAN_A. Refer to the MPC566 user’s manual for more information on these modules and the registers they contain.

Example:

```
ird usiu.plprcrk
```

IRM Internal Register Modify

Usage: IRM module.register data

This command modifies the contents of the internal registers of different modules inside the MPC500. In the command line, module refers to the module name where the register is located and register refers to the specific register to modify. The data parameter specifies the new value to be written into the register.

The registers are organized according to the module to which they belong. The available modules on the MPC500 are USIU, TPU_A, TPU_B, QADC_A, QADC_B, QSMCM_A, MIOS14, CAN_A. Refer to the MPC566 user’s manual for more information on these modules and the registers they contain.

Example:

To modify the PCPRCR in the USIU to the value 0x0091_4000, the command is:

```
irm        usiu.plprcr   914000
```

LR Loop Read

Usage: LR<width> addr

The LR command continually reads the data at addr until a key is pressed. The optional <width> specifies the size of the data to be read. If no <width> is specified, the command defaults to reading word sized data.

Example:

To continually read the word data from address 0xFFF2_0000, the command is:

```
lr          FFF20000
```

LW Loop Write

Usage: LW<width> addr data

The LW command continually writes data to addr. The optional width specifies the size of the access to memory. The default access size is a word.

Examples:

To continually write the data 0x1234_5678 to address 0xFFF2_0000, the command is:

```
lw        FFF20000 12345678
```

Note that the following command writes 0x78 into memory:

```
lw.b      FFF20000 12345678
```

MD Memory Display

Usage: MD<width> <begin> <end>

The MD command displays a contiguous block of memory starting at address begin and stopping at address end. The values for addresses begin and end may be absolute addresses specified as hexadecimal values, or symbol names. Width modifies the size of the data that is displayed. If no <width> is specified, the default of word sized data is used.

Memory display starts at the address begin. If no beginning address is provided, the MD command uses the last address that was displayed. If no ending address is provided, then MD will display memory up to an address that is 128 beyond the starting address.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples:

To display memory at address 0x0040_0000, the command is:

```
md 400000
```

To display memory in the data section (defined by the symbols data_start and data_end), the command is:

```
md data_start
```

To display a range of bytes from 0x00040000 to 0x0005_0000, the command is:

```
md.b     40000 50000
```

To display a range of 32-bit values starting at 0x0004_0000 and ending at 0x0005_0000:

```
md       40000 50000
```

MM Memory Modify

Usage: MM<width> addr <data>

The MM command modifies memory at the address addr. The value for addr may be an absolute address specified as a hexadecimal value, or a symbol name. Width specifies the size of the data that is modified. If no <width> is specified, the default of word sized data is used. The value for data may be a symbol name, or a number converted according to the user-defined radix, normally hexadecimal.

If a value for data is provided, then the MM command immediately sets the contents of addr to data. If no value for data is provided, then the MM command enters into a loop. The loop obtains a value for data, sets the contents of the current address to data, increments the address according to the data size, and repeats. The loop terminates when an invalid entry for the data value is entered, i.e., a period.

This command first aligns the starting address for the data access size, and then increments the address accordingly during the operation. Thus, for the duration of the operation, this command performs properly-aligned memory accesses.

Examples:

To set the byte at location 0xFFF1_0000 to be 0xFF, the command is:

```
mm.b        FFF10000 FF
```

To interactively modify memory beginning at 0xFFF1_0000, the command is:

```
mm          FFF10000
```

MMAP Memory Map Display

Usage: mmap

This command displays the memory map information for the MPC566 evaluation board. The information displayed includes the type of memory, the start and end address of the memory, and the port size of the memory. The display also includes information on how the Chip-selects are used on the board.

Here is an example of the output from this command:

```
Type                Start         End
----------------------------------------
ISB                  0x00000000   0x003FFFFF
Internal SRAM        0x003F8000   0x003FFFFF
Flash                0x00800000   0x009FFFFF
User Flash           0x00800000   0x008FFFFF
dBUG Flash           0x00900000   0x009FFFFF
Ethernet             0x01000000   0x01008000
External SRAM        0xFFF00000   0xFFF7FFFF
User Space           0xFFF08000   0xFFF7FFFF
```

RD Register Display

Usage: RD <reg>

The RD command displays the register set of the target. If no argument for reg is provided, then all registers are displayed. Otherwise, the value for reg is displayed.

dBUG preserves the registers by storing a copy of the register set in a buffer. The RD command displays register values from the register buffer.

Examples:

To display all the registers and their values, the command is:

```
rd
```

To display only the program counter:

```
rd      pc
```

Here is an example of the output from this command:

```
pc: FFF08000     msr: 00009042 [EE,ME,IP,RI]
cr: 00000000     xer: 00000000      lr: 00000000     ctr: 00000000
r00-07: 00000000 003FFF00 00000000 00000000 00000000 00000000 00000000 00000000
r08-15: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r16-23: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
r24-31: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

RM Register Modify

Usage: RM reg data

The RM command modifies the contents of the register reg to data. The value for reg is the name of the register, and the value for data may be a symbol name, or it is converted according to the user-defined radix, normally hexadecimal.

dBUG preserves the registers by storing a copy of the register set in a buffer. The RM command updates the copy of the register in the buffer. The actual value will not be written to the register until target code is executed.

Examples:

To change program counter to contain the value 0x003f_8000, the command is:

```
rm          pc 3f8000
```

RESET Reset the Board and dBUG

Usage: RESET

The RESET command resets dBUG to it’s initial power-on states.

The RESET command executes the same sequence of code that occurs at power-on. If the RESET command fails to reset the system adequately, cycle the power or press the reset button.

Examples:

To reset the board and clear the dBUG data structures, the command is:

```
reset
```

SET Set Configurations

Usage: SET <option value>

The SET command allows the setting of user-configurable options within dBUG. With no arguments, SET displays the options and values available. The SHOW command displays the settings in the appropriate format. The standard set of options is listed below.

- baud - This is the baud rate for the first serial port on the board. All communications between dBUG and the user occur using either 9600 or 19200 bps, eight data bits, no parity, and one stop bit, 8N1, with no flow control.
- speed - This is the clock speed of the processor. Changing the speed parameter will automatically configure the part's wait states as well as the wait cycles on the external flash. The default is 56MHz. The only clock speeds supported in dBUG are 40 and 56 MHz .
- base - This is the default radix for use in converting a number from its ASCII text representation to the internal quantity used by dBUG. The default is hexadecimal (base 16), and other choices are binary (base 2), octal (base 8), and decimal (base 10).
- client - This is the network Internet Protocol (IP) address of the board. For network communications, the client IP is required to be set to a unique value, usually assigned by your local network administrator.
- server - This is the network IP address of the machine which contains files accessible via TFTP. Your local network administrator will have this information and can assist in properly configuring a TFTP server if one does not exist.
- gateway - This is the network IP address of the gateway for your local subnetwork. If the client IP address and server IP address are not on the same subnetwork, then this option must be properly set. Your local network administrator will have this information.
- netmask - This is the network address mask to determine if use of a gateway is required. This field must be properly set. Your local network administrator will have this information.
- filename - This is the default filename to be used for network download if no name is provided to the DN command.
- filetype - This is the default file type to be used for network download if no type is provided to the DN command. Valid values are: “srecord”, “coff”, and “elf”.
- ethaddr - This is the ethernet address (a.k.a hardware address) for the evaluation board. This should be set to a unique value, and the most significant nibble should always be even.

Examples:

To set the baud rate of the board to be 19200, the command is:

```
set         baud 19200
```

To set the clock speed of the processor to be 40MHz, the command is:

```
set         speed 40
```

NOTE:

See the SHOW command for a display containing the correct formatting of these options.

SHOW Show Configurations

Usage: SHOW <option>

The SHOW command displays the settings of the user-configurable options within dBUG. When no option is provided, SHOW displays all options and values.

Examples:

To display all options and settings, the command is:

```
show
```

To display the current baud rate of the board, the command is:

```
show    baud
```

Here is an example of the output from a show command:

```
dBUG> show
    base: 16
    baud: 19200
    server: 192.0.0.1
    client: 192.0.0.2
    gateway: 0.0.0.0
    netmask: 255.255.255.0
    filename: test.srec
    filetype: S-Record
    mac: 00:CF:52:49:C3:01
    speed: 56000000
```

STEP Step Over

Usage: STEP

The STEP command can be used to “step over” a subroutine call, rather than tracing every instruction in the subroutine. The ST command sets a temporary software breakpoint one instruction beyond the current program counter and then executes the target code. This command only works when executing code in SRAM.

The STEP command can be used to “step over” BSR and JSR instructions.

The STEP command will work for other instructions as well, but note that if the STEP command is used with an instruction that will not return, i.e. BRA, then the temporary breakpoint may never be encountered and dBUG may never regain control.

Examples:

To pass over a subroutine call, the command is:

```
step
```

SYMBOL Symbol Name Management

Usage: SYMBOL <symb> <-a symb value> <-r symb> <-c|l|s>

The SYMBOL command adds or removes symbol names from the symbol table. If only a symbol name is provided to the SYMBOL command, then the symbol table is searched for a match on the symbol name and its information displayed.

The -a option adds a symbol name and its value into the symbol table. The -r option removes a symbol name from the table.

The -c option clears the entire symbol table, the -l option lists the contents of the symbol table, and the -s option displays usage information for the symbol table.

Symbol names contained in the symbol table are truncated to 31 characters. Any symbol table lookups, either by the SYMBOL command or by the disassembler, will only use the first 31 characters. Symbol names are case-sensitive.

Symbols can also be added to the symbol table via in-line assembly labels and ethernet downloads of ELF formatted files.

Examples:

To define the symbol “main” to have the value 0xFFF4_0000, the command is:

```
symbol          -a main FFF40000
```

To remove the symbol “junk” from the table, the command is:

```
symbol          -r junk
```

To see how full the symbol table is, the command is:

```
symbol          -s
```

To display the symbol table, the command is:

```
symbol          -l
```

TRACE Trace Into

Usage: TRACE <num>

The TRACE command allows single-instruction execution. If num is provided, then num instructions are executed before control is handed back to dBUG. The value for num is a decimal number.

The TRACE command sets bits in the processors' supervisor registers to achieve single-instruction execution, and the target code executed. Control returns to dBUG after a single-instruction execution of the target code.

This command is repeatable.

Examples:

To trace one instruction at the program counter, the command is:

```
tr
```

To trace 20 instructions from the program counter, the command is:

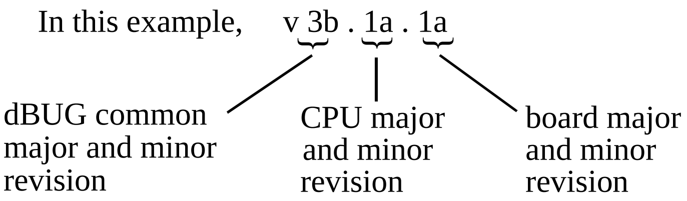
```
tr      20
```

VERSION Display dBUG Version

Usage: VERSION

The VERSION command displays the version information for dBUG. The dBUG version, build number and build date are all given.

The version number is separated by a decimal, for example, “v 2b.1c.1a”.


In this example,
v 3b . 1a . 1a
dBUG common major and minor revision
CPU major and minor revision
board major and minor revision

The version date is the day and time at which the entire dBUG monitor was compiled and built.

Examples:

To display the version of the dBUG monitor, the command is:

```
version
```

3.5 System Call Functions

An additional utility within the dBUG firmware is a function called the System Call handler. This function can be called by the user program to utilize various routines within dBUG, to perform a special task, and to return control to dBUG. This section describes the System Call handler and how it is used.

There are 6 System Call functions. These are: OUT_CHAR, IN_CHAR, IN_STAT, ISR_REGISTER, ISR_REMOVE and EXIT_TO_dBUG. The system call interface accepts an opcode in r10 to indicate which operation is to be performed. Various results are returned, usually in r3. When these routines are invoked, the following is true: sprg0 contains r31 and sprg1 contains LR.

3.5.1 OUT_CHAR

This function (function code 0x0020) sends a character, which is in r3, to terminal. The system call interface accepts an opcode in r10 to indicate which operation is to be performed.

Assembly example:

```
/* assume r3 contains the character */
addi r10, r0, 0x0020     Selects the function
sc                       The character in r3 is sent to terminal
```

C example:

```
/* assume r3 contains the character */
void board_out_char (int ch)
{
        asm("     addi r10, r0, 0x0020");  Selects the function
        asm("     sc");                    The character in r3 is sent to terminal
}
```

3.5.2 IN_CHAR

This function (function code 0x0000) returns an input character (from terminal) to the caller. The returned character is in r3.

Assembly example:

```
/* the character is returned to the user in r3*/
addi r10, r0, 0x0000     Selects the function
sc                       The character is returned in r3
```

C example:

```
int board_in_char (void)
```

```
{           /* assume r3 contains the character */
            asm("        addi r10, r0, 0x0000");  Selects the function
            asm("        sc");                    The character is returned in r3
}
```

3.5.3 IN_STAT

This function (function code 0x0001) checks if an input character is present to receive. A value of zero is returned in r3 when no character is present. A value of 1 in r3 means a character is present.

Assembly example:

```
addi r10, r0, 0x0001     Select the function
sc                        Make the call,  r3 contains the response (yes/no).
```

C example:

```
int board_char_present (void)
{
            asm(        "addi        r10,r0,0x0001");Select the function
            asm(        "sc");                            Make the call,  r3 contains the response (yes/no).
}
```

3.5.4 ISR_REGISTER

This function’s code is 0x0040. For ISR_REGISTER, the vector, handler, device ptr, and arg ptr are in r3, r4, r5, and r6 respectively.

C example:

```
int
board_isr_register (int vector, void *handler, void *device, void *arg)
{
            /*
             * Vector will normally be 0x0500 for IRQ.  Handler should be address
             * of your routine.  Device and Arg are both used as arguments to
             * Handler when it is invoked.  Ie. handler(device,arg);  It is
             * intended that Device point to the device. If the handler is
             * registered OK, 1 is returned, otherwise 0.
             */
            asm(        "addi        r10,r0,0x0040");
            asm(        "sc");
}
```

3.5.5 ISR_REMOVE

This function’s code is 0x0041. For ISR_REMOVE, the vector is in r3. Nothing is returned.

Assembly example:

```
addi r10, r0, 0x0041     Selects the function
sc                        The character is returned in r3
```

C example:

```
int
board_isr_remove (void *handler)
{
        asm(    "addi   r10,r0,0x0041");
        asm(    "sc");
}
```

3.5.6 EXIT_TO_dBUG

This function transfers the control back to the dBUG, by terminating the user code. The register context is preserved.

C example (see “scif.s”file : anything in R10 besides 0x0000, 0x0001, 0x0020, 0x0030, 0x0031, 0x0040, 0x0041):

```
asm(    "addi   r10,r0,0x0063");
asm(    "sc");
```

Appendix A

MPC535/536 Emulation

The MPC5646EVB can be used to emulate the MPC535 and MPC536 processors. This appendix will highlight the differences between the processors. Several modules present in the MPC566 are not available in the MPC535/6, so the user should not reference those modules.

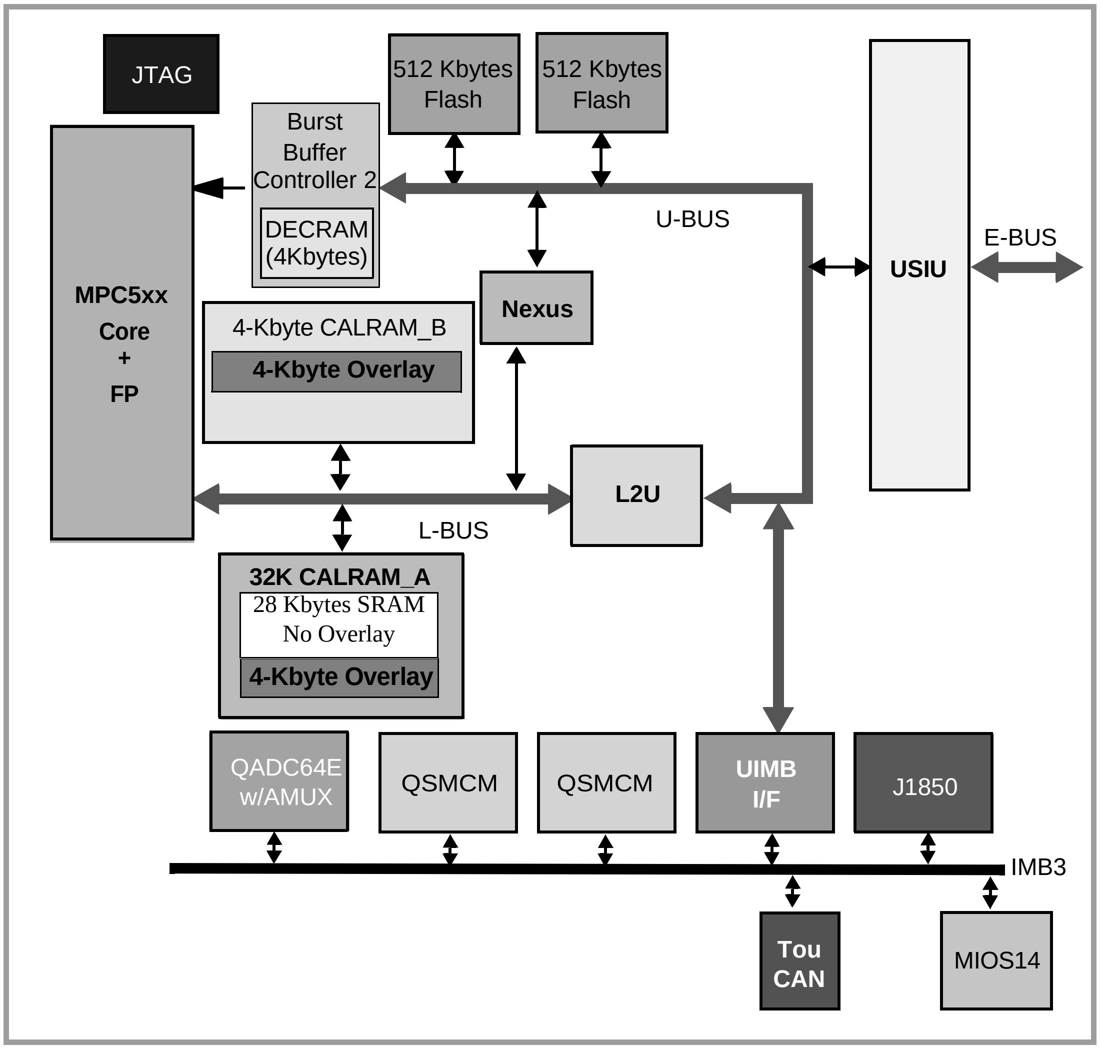
Table A-1 compares the MPC566, MPC535, and MPC536 module lists. Figure A-1shows the block diagram of the MPC535/6.

.

Table A-1. Module Differences of MPC566, MPC535, and MPC536

Module	MPC566	MPC535	MPC536
Core Speed	40 or 56MHz	**40MHz**	**40MHz**
Termperature Range	-40^{o}C to +85^{o}C and -40^{o}C to +125^{o}C	-40^{o}C to +85^{o}C	-40^{o}C to +85^{o}C
Code Compression	available	**N/A**	available
FLASH	1M	1M	1M
Static RAM	40K	40K	40K
TPU3	(2)	**N/A**	**N/A**
DPTRAM	8K	**N/A**	**N/A**
QADC Module	(2)	**(1) QADC_A**	**(1) QADC_A**
TouCAN (CAN Version 2.0B)	(3) CAN_A, CAN_B, CAN_C	**(1) CAN_B**	**(1) CAN_B**

Figure A-1. 5/6


JTAG
512 Kbytes Flash
512 Kbytes Flash
Burst Buffer Controller 2
DECRAM (4Kbytes)
U-BUS
E-BUS
USIU
MPC5xx Core + FP
4-Kbyte CALRAM_B
4-Kbyte Overlay
Nexus
L2U
L-BUS
32K CALRAM_A
28 Kbytes SRAM No Overlay
4-Kbyte Overlay
QADC64E w/AMUX
QSMCM
QSMCM
UIMB I/F
J1850
IMB3
Tou CAN
MIOS14

Appendix B
Configuring dBUG for Network Downloads

The dBUG module has the ability to perform downloads over an Ethernet network using the Trivial File Transfer Protocol, TFTP (NOTE: this requires a TFTP server to be running on the host attached to the board). Prior to using this feature, several parameters are required for network downloads to occur. The information that is required and the steps for configuring dBUG are described below.

B.1 Required Network Parameters

For performing network downloads, dBUG needs 6 parameters; 4 are network-related, and 2 are download-related. The parameters are listed below, with the dBUG designation following in parenthesis.

All computers connected to an Ethernet network running the IP protocol need 3 network-specific parameters. These parameters are:

Internet Protocol, IP, address for the computer (client IP),
IP address of the Gateway for non-local traffic (gateway IP), and
Network netmask for flagging traffic as local or non-local (netmask).

In addition, the dBUG network download command requires the following three parameters:

IP address of the TFTP server (server IP),
Name of the file to download (filename),
Type of the file to download (filetype of S-record, COFF, ELF, or Image).

Your local system administrator can assign a unique IP address for the board, and also provide you the IP addresses of the gateway, netmask, and TFTP server. Fill out the lines below with this information.

Client IP:___.___.___.___(IP address of the board)
Server IP:___.___.___.___(IP address of the TFTP server)
Gateway:___.___.___.___(IP address of the gateway)
Netmask:___.___.___.___(Network netmask)

B.2 Configuring dBUG Network Parameters

Once the network parameters have been obtained, the dBUG Rom Monitor must be configured. The following commands are used to configure the network parameters.

```
set client <client IP>
set server <server IP>
set gateway <gateway IP>
set netmask <netmask>
set ethaddr <addr>
```

For example, the TFTP server is named ‘santafe’ and has IP address 123.45.67.1. The board is assigned the IP address of 123.45.68.15. The gateway IP address is 123.45.68.250, and the netmask is 255.255.255.0. The ethaddr address is chosen arbitrarily and is unique. The commands to dBUG are:

```
set client 123.45.68.15
set server 123.45.67.1
set gateway 123.45.68.250
set netmask 255.255.255.0
set ethaddr 00:CF:52:49:C3:01
```

The last step is to inform dBUG of the name and type of the file to download. Prior to giving the name of the file, keep in mind the following.

Most, if not all, TFTP servers will only permit access to files starting at a particular sub-directory. (This is a security feature which prevents reading of arbitrary files by unknown persons.) For example, SunOS uses the directory /tftp_boot as the default TFTP directory. When specifying a filename to a SunOS TFTP server, all filenames are relative to /tftp_boot. As a result, you normally will be required to copy the file to download into the directory used by the TFTP server.

A default filename for network downloads is maintained by dBUG. To change the default filename, use the command:

```
set filename <filename>
```

When using the Ethernet network for download, either S-record, COFF, ELF, or Image files may be downloaded. A default filetype for network downloads is maintained by dBUG as well. To change the default filetype, use the command:

```
set filetype <srecord|coff|elf|image>
```

Continuing with the above example, the compiler produces an executable COFF file, ‘a.out’. This file is copied to the /tftp_boot directory on the server with the command:

```
rcp a.out santafe:/tftp_boot/a.out
```

Change the default filename and filetype with the commands:

```
set filename a.out
set filetype coff
```

Finally, perform the network download with the ‘dn’ command. The network download process uses the configured IP addresses and the default filename and filetype for initiating a TFTP download from the TFTP server.

B.3 Troubleshooting Network Problems

Most problems related to network downloads are a direct result of improper configuration. Verify that all IP addresses configured into dBUG are correct. This is accomplished via the ‘show ’command.

Using an IP address already assigned to another machine will cause dBUG network download to fail, and probably other severe network problems. Make certain the client IP address is unique for the board.

Check for proper insertion or connection of the network cable. Is the status LED lit indicating that network traffic is present?

Check for proper configuration and operation of the TFTP server. Most Unix workstations can execute a command named ‘tftp’ which can be used to connect to the TFTP server as well. Is the default TFTP root directory present and readable?

If ‘ICMP_DESTINATION_UNREACHABLE’ or similar ICMP message appears, then a serious error has occurred. Reset the board, and wait one minute for the TFTP server to time out and terminate any open connections. Verify that the IP addresses for the server and gateway are correct. Also verify that a TFTP server is running on the server.