M5282EVB User's Manual

Devices Supported:

MCF5282
MCF5281
MCF5280
MCF5216
MCF5214

Document Number: M5282EVBUM
Rev. 2
1/2009

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd. Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

M5282EVBUM
Rev. 2
1/2009

EMC Information on M5282EVB

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.

Chapter 1
M5282EVB Introduction

Chapter 2
Initialization and Setup

Chapter 3
Using the Monitor/Debug Firmware

Appendix A
Configuring dBUG for Network Downloads

Appendix B
Schematics

Appendix C
Evaluation Board BOM

Appendix D
Jumper Settings

Appendix E
Using the M5282EVB to Evaluate Subset Devices

Appendix F
Revision History

Chapter 1
M5282EVB Introduction

The M5282EVB is a MCF5282-based evaluation board that can be used for the development and test of microcontroller systems (see Figure 1-1). The MCF5282 is a member of the Freescale ColdFire 32-bit processor family.

The evaluation board is a development and test platform for software and hardware for the MCF5282. It can be used by software and hardware developers to test programs, tools, or circuits without having to develop a complete microprocessor system themselves. All special features of the MCF5282 are supported.

The M5282EVB now supports the evaluation of the MCF5280, MCF5281, MCF5214, and MCF5216 microcontrollers in addition to the MCF5282. The evaluation board comes fitted with 80MHz MCF5282 microcontroller (512 Kbyte internal flash). The MCF5282 microcontroller has a superset of the same functional modules and interfaces as those on the other devices supported.

The heart of the evaluation board is the MCF5282. The M5282EVB has 8 Mbyte external SDRAM for development or application memory, 2Mbyte external Flash memory, and numerous hardware expansion possibilities. The M5282EVB board also provides an Ethernet interface (10/100BaseT), FlexCAN, QSPI, QADC, I2C, and RS232 interface in addition to the built-in I/O functions of the MCF5282 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 using the BDM or JTAG interface.

The M5282EVB 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 MCF5282 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.

Specifications

- Freescale MCF5282 Microprocessor (80MHz max core/bus frequency, Firmware running @ 64MHz)
- External Clock source: 8MHz
- Operating temperature: 0°C to +70°C
- Power requirement: 6 – 14V DC @ 300 ma Typical
- Power output: 5V and 3.3V regulated supplies
- Board Size: 7.00 x 7.60 inches, 8 layers

Memory Devices

- 16-Mbyte SDRAM
- 2-MByte (512K x 32) Sync. FLASH
- 512-Kbyte SRAM (optional)
- 512-Kbyte FLASH internal to MCF5282 device
- 64-Kbyte SRAM internal to MCF5282 device

Peripherals

- Ethernet port 10/100Mb/s (Dual-Speed Fast Ethernet Transceiver, with MII)
- UART0 (RS-232 serial port for dBUG firmware)
- UART1 (auxiliary RS-232 serial port)
- I2C interface
- QSPI interface
- QADC interface
- FlexCan interface
- BDM/JTAG interface

User Interface

- Reset logic switch (debounced)
- Boot logic selectable (dip switch)
- Abort/IRQ7 logic switch (debounced)
- Clocking options - Oscillator, Crystal or SMA for external clocking signals
- LEDs for power-up indication, general purpose I/O, and timer output signals
- Expansion connectors for daughter card
- Expansion connectors for MCU Target Board (Axiom)

Software

- Resident firmware package that provides a self-contained programming and operating environment (dBUG)

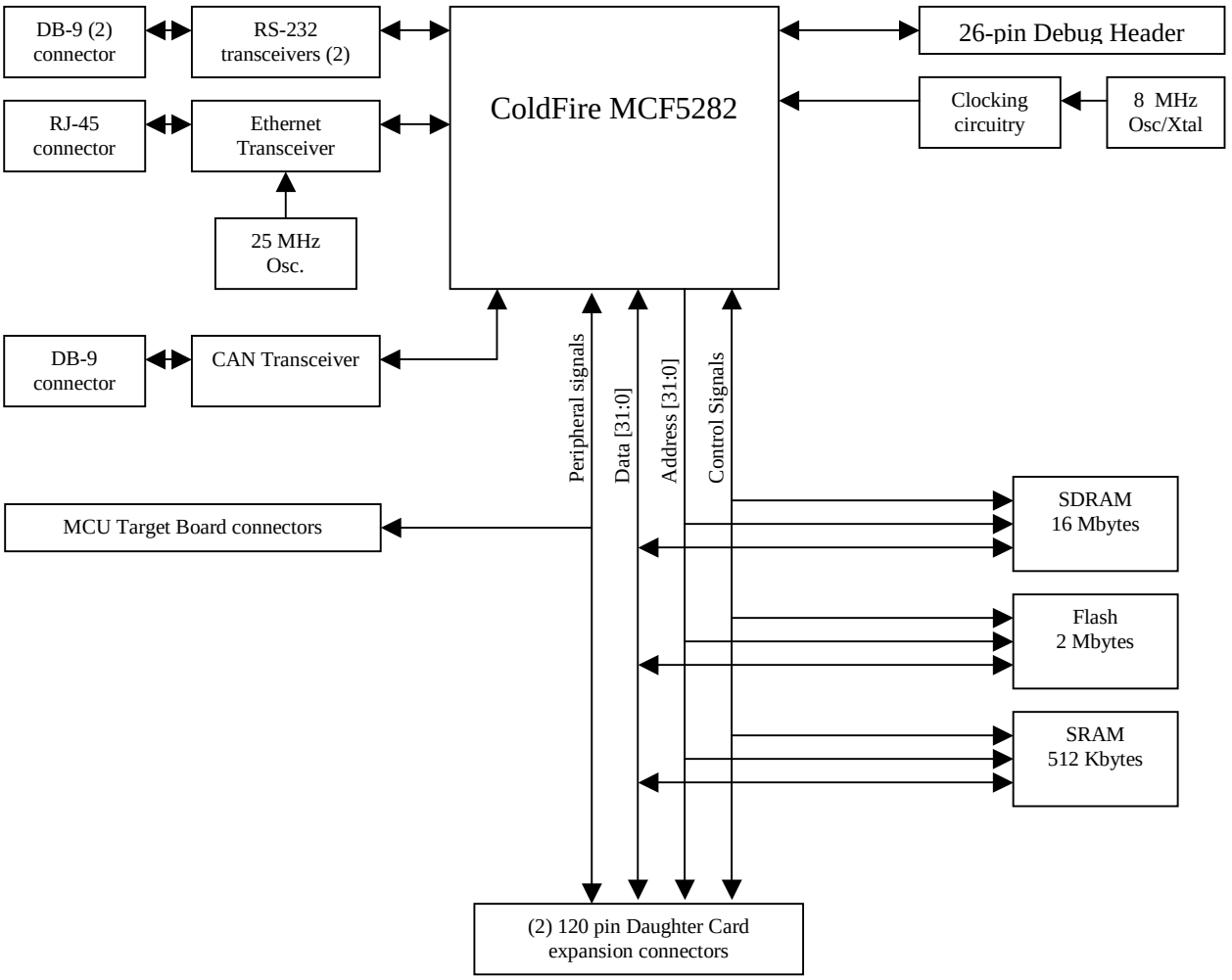


Figure 1-1. M5282EVB Block Diagram

1.1 MCF5282 Microprocessor

The microprocessor used on the M5282EVB is the highly integrated Freescale MCF5282 32-bit ColdFire variable-length RISC processor. The MCF5282 implements a ColdFire Version 2 core with a maximum core frequency and external bus speed of 80MHz. Features of the MCF5282 include:

- 512 Kbyte Flash memory
- 64-Kbytes of dual ported SRAM
- System debug support
- Fast Ethernet Controller (FEC)
- FlexCan 2.0B
- I^2C
- QSPI
- QADC
- Four 32-bit DAM timers

- Two 4-channel general purpose timers
- Four periodic interrupt timers(PITs)
- Software watchdog timer
- Phase Locked Loop (PLL)
- Two interrupt controllers
- DMA controller (4 channels)
- External bus interface
- General purpose I/O interface
- JTAG

The MCF5282 communicates with external devices over a 32-bit wide data bus, D[0:31]. The MCF5282 can address a 32 bit address range. However, only 24 bits are available on the external bus. 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 supervisor, user, instruction, and data each to have the 32-bit address range.

All the processor's signals are available through the daughter card expansion connectors. Refer to the schematic for their pin assignments.

The MCF5282 processor has the capability to support both BDM and JTAG. 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.

Figure 1-2 shows the MCF5282 processor block diagram.

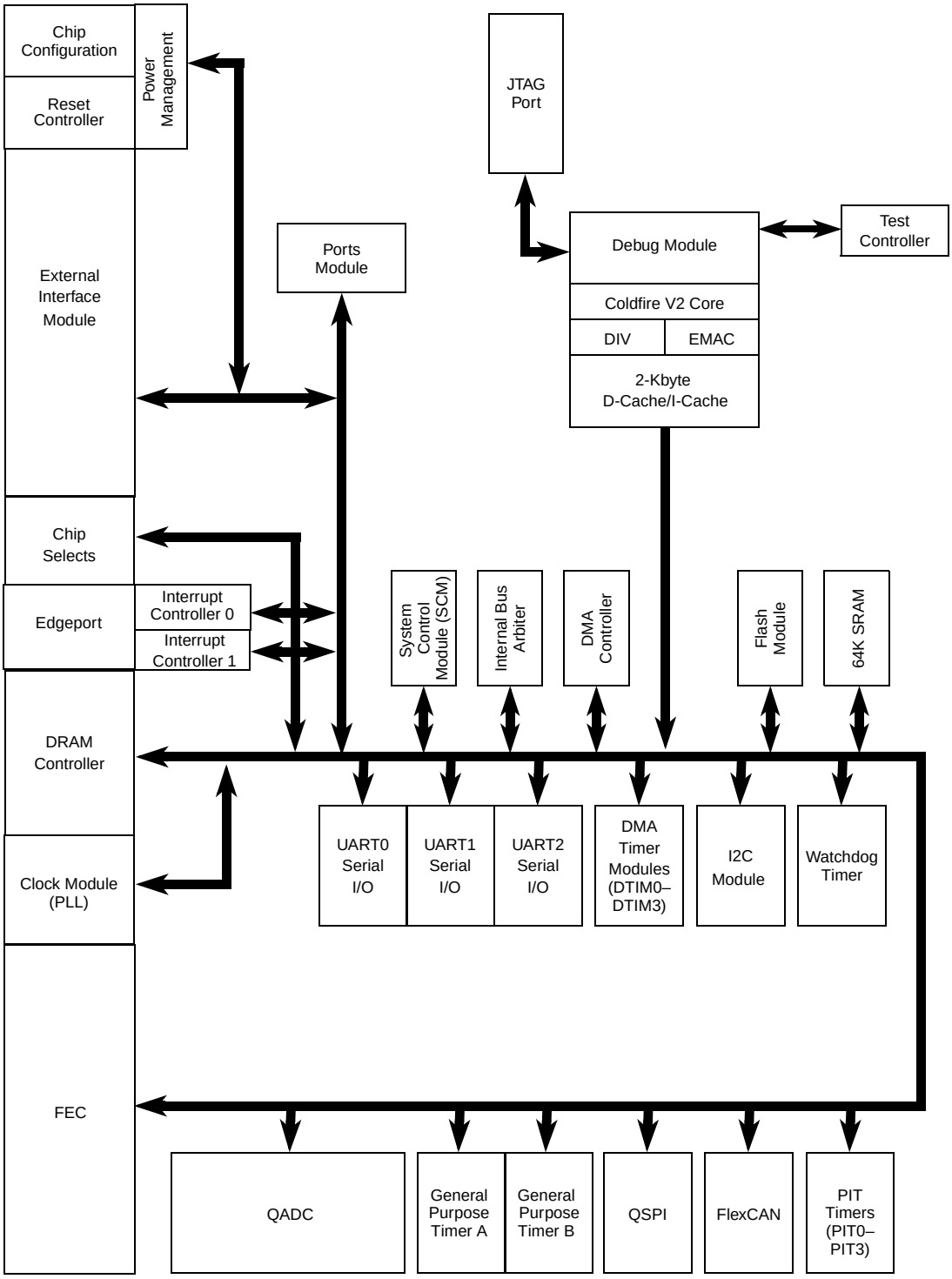


Figure 1-2. MCF5282 Block Diagram

1.2 System Memory

1.2.1 External Flash

One on-board Flash ROM (U5) is used in the system. The Am29lv160DB device contains 16Mbits of non-volatile storage (2 M x 8-bit/1 M x 16-bit) giving a total of 2MBytes of Flash memory. Refer to the specific device data sheet and sample software provided for configuring the flash memory.

Users 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. Users should use caution to avoid this situation. The M5282EVB dBUG debugger/monitor firmware is programmed into the lower sectors of Flash (0xFFE0_0000 to 0xFFE2_FFFF).

1.2.2 SDRAM

The M5282EVB has 16 Mbytes of SDRAM populated on the EVB. This is done with two devices (Micron MT48LC4M16A2TG) 16 bit data bus each. Each device is organized as 1 Meg x 16 x 4 banks with a 16 bit data bus. One device stores the upper 16-bit word and the other the lower 16 bit word of the MCF5282 32 bit data bus.

1.2.3 SRAM

The M5282EVB has a footprint for one 4 Mbit (128 x 36) SRAM device (Micron- MT58L128L36F1) on the EVB (U2). This memory device may be populated by the user for benchmarking purposes.

Also see Section 1.2.6, "M5282EVB Memory Map".

1.2.4 Internal SRAM

The MCF5282 processor has 64-KBtyes of internal SRAM memory which may be used as data or instruction memory. This memory is mapped to 0x2000_0000 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 within the processor's four gigabyte address space.

1.2.5 Internal Flash

The ColdFire Flash Module (CFM) is constructed with eight banks of 32K x 16-bit Flash to generate a 512-Kbyte, 32-bit wide electrically erasable and programmable read-only memory array. The CFM is ideal for program and data storage for single-chip applications and allows for field reprogramming without external high-voltage sources. The voltage required to program and erase the Flash is generated internally by on-chip charge pumps. Program and erase operations are performed under CPU control through a command driven interface to an internal state machine. All Flash physical blocks can be programmed or erased at the same time; however, it is not possible to read from a Flash physical block while the same

block is being programmed or erased. The array used in the MCF5282 makes it possible to program or erase one pair of Flash physical blocks under the control of software routines executing out of another pair.

Please refer to the *MCF5282 User's Manual* for more details.

1.2.6 M5282EVB Memory Map

Interface signals to support interface to external memory and peripheral devices are generated by the memory controller. The MCF5282 supports 7 external chip selects, however three of them are multiplexed with external address lines. $\overline{CS}$[1:0] are used with external memories and $\overline{CS}$[3:2] are easily accessible to users through the daughter card expansion connectors. $\overline{CS}$[0] also functions as the global (boot) chip-select for booting out of external flash.

Since the MCF5282 chip selects are fully programmable, the memory banks may be located at any any 64-KByte boundary within the processor's four gigabyte address 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 MCF5282 is detailed further in the *MCF5282 User's 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:

```
External FLASH Memory     CS0 or CS1        default CS0 (JP6 =1&2,3&4)
External SRAM Memory      CS0 or CS1        default CS1 (JP6 =1&2,3&4)
```

Table 1-1 shows the M5282EVB memory map.

Table 1-1. The M5282EVB Default Memory Map

Address Range	Signal and Device
0x0000_0000 - 0x00FF_FFFF	16 Mbyte SDRAM
0x2000_0000 - 0x2000_0000	64 Kbytes Internal SRAM
0x3000_0000 - 0x3000_0000	External SRAM (not fitted)
0xF000_0000 - 0xF007_FFFF	512 Kbytes Internal Flash
0xFFE0_0000 - 0xFFFF_FFFF	2 Mbytes External Flash

1.2.6.1 CS0 selection

When booting from an external device, the MCF5282 accesses this device using CS0. CS0 can be configured to connect to the external flash or external SRAM.

Table 1-2. JP 6 - CS0 Settings

JP6	CS0	CS1
Across 1&2, 3&4	External Flash	External SRAM
Across 1&3, 2&4	External SRAM	External Flash

1.2.6.2 Reset Vector Mapping

Asserting the reset input signal to the processor causes a reset exception. The reset exception has the highest priority of any exception; it provides for system initialization and recovery from catastrophic failure. Reset also aborts any processing in progress when the reset input is recognized. Processing cannot be recovered.

The reset exception places the processor in the supervisor mode by setting the S-bit and disables tracing by clearing the T bit in the SR. This exception also clears the M-bit and sets the processor's interrupt priority mask in the SR to the highest level (level 7). Next, the VBR is initialized to zero (0x0000_0000). The control registers specifying the operation of any memories (e.g., cache and/or RAM modules) connected directly to the processor are disabled.

Once the processor is granted the bus, it then performs two longword read bus cycles. The first longword at address 0 is loaded into the stack pointer and the second longword at address 4 is loaded into the program counter. After the initial instruction is fetched from memory, program execution begins at the address in the PC. If an access error or address error occurs before the first instruction is executed, the processor enters the fault-on-fault halted state.

The Memory that the MCF5282 accesses at address 0 is determined at reset by sampling D[19:18].

Table 1-3. D[19:18] External Boot Chip Select Configuration

D[19:18]	Boot Device/ Data Port Size
00	Internal (32-bit)
01	External (16-bit)
10	External (8-bit)
11	External (32-bit)

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 MCF5282 to reset. $\overline{\text{RSTI}}$ is triggered by the reset switch (SW2) which resets the entire processor/system.

dBUG configures the MCF5282 microprocessor internal resources during initialization. The contents of the exception table are copied to address 0x0000_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.

If the external $\overline{\text{RCON}}$ pin is asserted (SW1-1 ON) during reset, then various chip functions, including the reset configuration pin functions after reset, are configured according to the levels driven onto the external data pins. See the tables below on settings for reset configurations.

If the $\overline{\text{RCON}}$ pin is not asserted (SW1-1 OFF) during reset, the chip configuration and the reset configuration pin functions after reset are determined by fixed defaults, regardless of the states of the external data pins.

Table 1-4. SW1-1 $\overline{\text{RCON}}$

SW1-1	Reset Configuration
OFF	$\overline{\text{RCON}}$ not asserted. Default chip configuration
ON	$\overline{\text{RCON}}$ is asserted. Chip functions, including the are configured according to the levels driven onto the external data pins

Table 1-5. SW1-2 JTAG_EN

SW1-2	JTAG Enable
OFF	JTAG interface enabled
ON	BDM interface enabled

Table 1-6. SW1-[4:3] Encoded Clock Mode

SW1-3	SW1-4	Clock Mode
OFF	OFF	External clock mode- (PLL disabled)
OFF	ON	1:1 PLL
ON	OFF	Normal PLL mode with external clock reference
ON	ON	Normal PLL mode w/crystal oscillator reference

Table 1-7. SW1-[7:5] Chip Configuration Mode

SW1-5	SW1-6	SW1-7	RCON (SW1-1)	Mode
OFF	X	X	ON	Reserved
ON	OFF	ON	ON	Reserved
ON	OFF	OFF	ON	Factory Test
ON	ON	OFF	ON	Single Chip
ON	ON	ON	ON	Master
X	X	X	OFF	Single Chip

Table 1-8. SW1-[9:8] Boot Device

SW1-8	SW1-9	$\overline{\text{RCON}}$ (SW1-1)	Boot Device
OFF	OFF	ON	Internal (32-bit)
OFF	ON	ON	External (16-bit)
ON	OFF	ON	External (8-bit)
ON	ON	ON	External (32-bit)
X	X	OFF	Internal (32-bit)

Table 1-9. SW1-10 Bus Drive Strength

SW1-10	RCON (SW1-1)	Drive Strength
OFF	ON	Partial Bus Drive
ON	ON	Full Bus Drive
X	OFF	Full Bus Drive

Table 1-10. SW1-[12:11] Address/Chip Select Mode

SW1-11	SW1-12	RCON (SW1-1)	Mode
OFF	OFF	ON	PF[7:5] = CS[6:4]
OFF	ON	ON	PF[7] = CS6, PF[6:5] = A[22:21]
ON	OFF	ON	PF[7:6] = CS[6:5], PF[5] = A21
ON	ON	ON	PF[7:5] = A[23:21]
X	X	OFF	PF[7:5] = A[23:21]

1.3.2 Clock Circuitry

The M5282EVB three options to provide the clock to the MCF5282. These options can be configured by setting JP[25:27]. See Table 1-11.

Table 1-11. M5282EVB Clock Source Selection

JP25	JP26	JP27	Clock Selection
ON	1-2	ON	16 MHz External Clock
OFF	1-2	ON	16 MHz Oscillator
ON	2-3	OFF	Crystal (Default setting)

There is also a 25MHz oscillator (U3) which feeds the Ethernet chip (U4).

1.3.3 Watchdog Timer

The dBUG Firmware does **NOT** enable the watchdog timer on the MCF5282.

1.3.4 Exception Sources

The ColdFire® family of processors can receive seven levels of interrupt priorities. When the processor receives an interrupt which has a higher priority than the current interrupt mask (in the status register), it will perform an interrupt acknowledge cycle at the end of the current instruction cycle. This interrupt acknowledge cycle indicates to the source of the interrupt that the request is being acknowledged and the device should provide the proper vector number to indicate where the service routine for this interrupt level is located. If the source of interrupt is not capable of providing a vector, its interrupt should be set up as an autovector interrupt which directs the processor to a predefined entry in the exception table (refer to the MCF5282 User's Manual).

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 stored in the VBR. The dBUG ROM monitor writes a copy of the exception table into the RAM starting at 0x0000_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 0x0000_0000 and then points the VBR to 0x0000_0000.

The MCF5282 microprocessor has eight external interrupt request lines $\overline{\text{IRQ}}$[7:0], all of which are multiplexed with other functions. The interrupt controller is capable of providing up to 32 interrupt sources. These sources are:

- External interrupt signals $\overline{\text{IRQ}}$[7:1] (EPORT)
- Software watchdog timer module
- Timer modules
- UART module
- I^2C module
- DMA module
- QSPI module
- FEC module
- QADC

All external interrupt inputs are edge sensitive. The active level is programmable. An interrupt request must be held valid until an IACK cycle starts to guarantee correct processing. Each interrupt input can have it's priority programmed by setting the xIPL[2:0] bits in the Interrupt Control Registers.

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 M5282EVB hardware uses $\overline{\text{IRQ7}}$ to support the ABORT function using the ABORT switch (S1). This switch is used to force an interrupt (level 7, priority 3) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT). Since the ABORT switch is not capable of generating a vector in response to a level seven interrupt acknowledge from the processor, the dBUG programs this interrupt request for autovector mode.

Refer to *MCF5282 User's Manual* for more information about the interrupt controller.

1.3.5 TA Generation

The processor starts a bus cycle by asserting $\overline{\text{CSx}}$ with the other control signals. The processor then waits for a transfer acknowledgment ($\overline{\text{TA}}$) either from within (Auto acknowledge - AA mode) or from the externally addressed device before it can complete the bus cycle. -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 asynchronously. The MCF5282 processor, as part of the chip-select logic, has a built-in mechanism to generate $\overline{\text{TA}}$ for all external devices which do not have the capability to generate this signal. For example the Flash ROM cannot generate a $\overline{\text{TA}}$ signal. The chip-select logic is programmed by the dBUG ROM Monitor to generate $\overline{\text{TA}}$ internally after a pre-programmed number of wait states. In order to support future expansion of the M5282EVB, the $\overline{\text{TA}}$ input of the processor is also connected to the Processor Expansion Bus (J2, pin 68). This allows any expansion boards to assert this line to provide a $\overline{\text{TA}}$

signal to the processor. On the expansion boards this signal should be generated through an open collector buffer with no pull-up resistor; a pull-up resistor is included on this board. All $\overline{TA}$ signals from expansion boards should be connected to this line.

1.3.6 User's Program

Jumper JP16 allows users to test code from boot/POR without having to overwrite the ROM Monitor.

When the jumper is set between pins 1 and 2, the behavior of the system is normal, dBUG boots and then runs from 0xFFE0_0000. When the jumper is set between pins 2 and 3, the board boots from the second half of the Flash (0xFFF0_0000).

Procedure:

1. Compile and link as though the code was to be placed at the base of the flash.
2. Set up the jumper (JP16) for Normal operation, pin1 connected to pin 2.
3. Download to SDRAM (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 0xFFE0_0400 and run.)
4. In the ROM Monitor, execute the 'FL write <dest> <src> <bytes>' command.
5. Move jumper (JP16) to pin 2 connected to pin 3 and push the reset button (S2). User code should now be running from reset/POR.

1.4 Communication Ports

The M5282EVB provides external communication interfaces for 2 UART serial ports, QSPI, FlexCan 2.0B port, I^2C port, 10/100T ethernet port, and BDM/JTAG port.

1.4.1 UART0 and UART1

The MCF5282 device has three built in UARTs, each with its own software programmable baud rate generators. Two of these UART interfaces are brought out to RS232 transceivers. One channel is the ROM Monitor to Terminal output and the other is available to the user. The ROM Monitor programs the interrupt level for UART0 to Level 3, priority 2 and autovector mode of operation. The interrupt level for UART1 is programmed to Level 3, priority 1 and autovector mode of operation. The signals from these channels are available on expansion connector (J3). The signals of UART0 and UART1 are passed through the RS-232 transceivers (U15) & (U16) and are available on DB-9 connectors (P4) and (P5). UART signal pins that are multiplexed with other functions on the MCF5282 can be disconnected from the UART transceivers by removing specific jumper JP[28:35].

Table 1-12. UART0 and UART1 Jumpers

Jumper	UART Signal
JP28	GPTB[0]0 / PTB[0]
JP29	GPTB[1] / PTB[1]
JP30	DTIN3 / URTS0

Table 1-12. UART0 and UART1 Jumpers (continued)

Jumper	UART Signal
JP31	DTIN2 / UCTS0
JP32	GPTB[2] / PTB[2]
JP33	GPTB[3] / PTB[3]
JP34	DTOUT3 / URTS1
JP35	DTOUT2 / UCTS1

Refer to the *MCF5282 User's Manual* for programming the UART's and their register maps.

1.4.2 FlexCAN 2.0 B Port

The M5282EVB board provides 1 CAN transceivers. The CAN TX and RX signals are brought out to a 3.3V CAN transceiver (Texas Instruments - SN65HVD230D). Jumper JP1 and JP2 control the CAN hardware configuration.

Table 1-13. CAN Jumper Configuration

Jumper	Function	ON	OFF
JP1	Transceiver mode	Standby	High Speed (No Slope Control)
JP2	CAN Termination	Terminating resistor between CANL and CANH	No terminating resistor

The CANL and CANH signals are brought our from the CAN transceiver to a female DB-9 connector (P9) in the configuration below.

Table 1-14. CAN Bus Connector Pinout

DB-9 pin	Signal
1,4-6,7-9	Not Connected
2	CANL
3	Ground
7	CANH

1.4.3 10/100T Ethernet Port

The MCF5272 device performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The MCF5282 Ethernet Controller requires an external interface adaptor and transceiver function to complete the interface to the ethernet media. The MCF5282 Ethernet module also features an integrated fast (100baseT) Ethernet media access controller (MAC).

The Fast Ethernet controller (FEC) incorporates the following features:

- Full compliance with the IEEE 802.3 standard
- Support for three different physical interfaces:

  - 100 Mbps 802.3 media independent interface (MII)
  - 10 Mbps 802.3 MII
  - 10 Mbps seven-wire interface
- Half-duplex 100 Mbps operation at system clock frequency 50 MHz
- 448 bytes total on-chip transmit and receive FIFO memory to support a range of bus latencies. Note: the total FIFO size is 448 bytes. It is not intended to hold entire frames but only to compensate for external bus latency. The FIFO can be partitioned on any 32-bit boundary between receive and transmit, for example, 32 x 56 receive and 32 x 56 transmit.
- Retransmission from transmit FIFO following a collision, no processor bus used
- Automatic internal flushing of the receive FIFO for runts and collisions with no processor bus use

For more details see the *MCF5282 User's Manual*, this module's signals are also brought to expansion connector J3. The on board ROM MONITOR is programmed to allow a user to download files from a network to memory in different formats. The current compiler formats supported are S-Record, COFF, ELF or Image.

Table 1-15. Ethernet Hardware Configuration

Jumper	Function (See AMD-AM79C874VC User's Manual for Function Description)
JP8	Auto Negotiate
JP9	Tech 0
JP10	Tech 1
JP11	Tech 2

1.4.4 BDM/JTAG Port

The MCF5282 processor has a Background Debug Mode (BDM) port, which supports Real-Time Trace Support and Real-Time Debug. The signals which are necessary for debug are available at connector (J4). Figure 1-3 shows the (J4) Connector pin assignment.

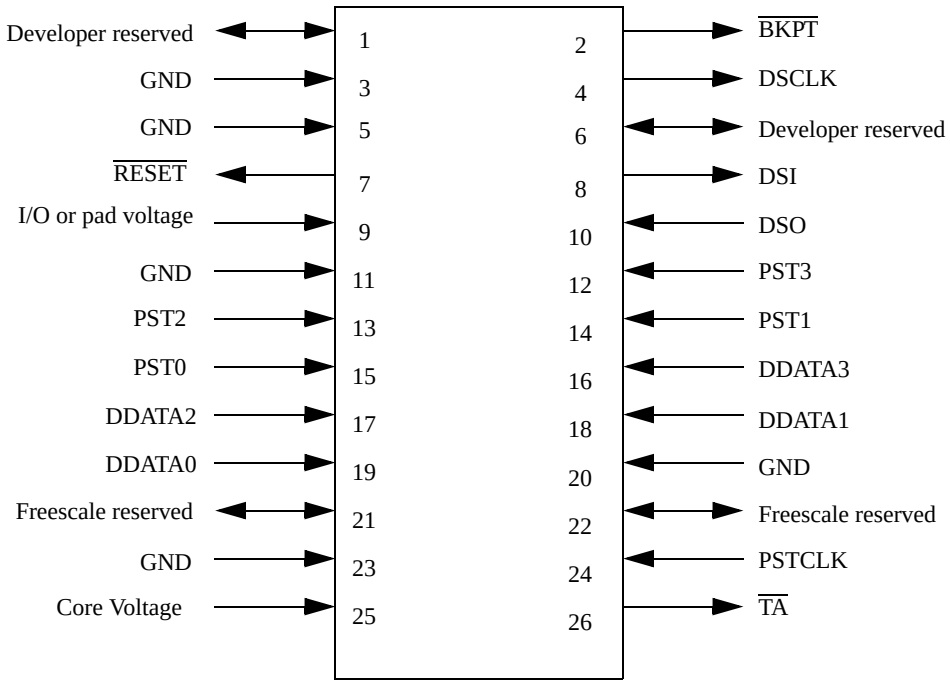


Figure 1-3. J4- BDM Connector Pin Assignment

Pin 7 of the BDM connector on the M5282EVB may be configured to connect to the RESET or TCLK signal of the MCF5282. For BDM communication the default is to configure this pin for RESET.

Table 1-16. BDM Header Pin 7 Selection

JP17	Source
1-2	Pin 7 is RESET
2-3	Pin 7 is TCLK

The BDM connector can also be used to interface to JTAG signals. If you may configure this Pin 7 of the BDM header for TCLK for custom JTAG hardware.On reset, the JTAG_EN signal selects between multiplexed debug module and JTAG signals. See Table 1-5.

1.4.5 I²C

The MCF5282's I²C module includes the following features:

- Compatibility with the I²C bus standard
- Multi master operation
- Software programmable for one of 64 different clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte by byte data transfer
- Arbitration-lost interrupt with auto mode switching from master to slave
- Calling address identification interrupt

- Start and stop signal generation and detection
- Repeated start signal generation
- Acknowledge bit generation and detection
- Bus busy detection

Please see the *MCF5282 User's Manual* for more detail. The I^2C signals from the MCF5282 device are brought out to expansion connector (J10). Jumpers JP36 and JP37 can be used to connect/disconnect the I^2C signals, SDA and SCL, from the daughter card expansion connector, J3.

1.4.6 QSPI

The QSPI (Queued Serial Peripheral Interface) module provides a serial peripheral interface with queued transfer capability. It will support up to 16 stacked transfers at one time, minimizing CPU intervention between transfers. Transfer RAMs in the QSPI are indirectly accessible using address and data registers.

Functionality is very similar, but not identical, to the QSPI portion of the QSM (Queued Serial Module) implemented in the MC68332 processor.

- Programmable queue to support up to 16 transfers without user intervention
- Supports transfer sizes of 8 to 16 bits in 1-bit increments
- Four peripheral chip-select lines for control of up to 15 devices
- Baud rates from 274.5-Kbps to 17.5-Mbps at 140MHz.
- Programmable delays before and after transfers
- Programmable clock phase and polarity
- Supports wrap-around mode for continuous transfers

Please see the *MCF5282 User's Manual* for more detail. The QSPI signals from the MCF5282 device are brought out to expansion connector (J9). Some of these signals are multiplexed with other functions.

1.5 Connectors and User Components

1.5.1 Expansion Connectors

Three 2x10 dual row 100mil Berg Headers provide access to a number of MCF5282 peripheral signals. Below is a pinout description of these connectors.

Table 1-17. J5

Pin	MCF5282 Signal	Pin	MCF5282 Signal
1	VSSA	2	VRL
3	AN52/PQA0	4	GPTA0
5	AN0/PQB0	6	GPTA1
7	RESET	8	GPTA3
9	RSTOUT	10	GPTA2
11	AN56/PQA4	12	GPTB3

Table 1-17. J5 (continued)

Pin	MCF5282 Signal	Pin	MCF5282 Signal
13	AN55/PQA3	14	AN53/PQA1
15	GPTB2	16	AN1/PQB1
17	AN2/PQB2	18	VDDA
19	AN3/PQB3	20	VRH

Table 1-18. J6

Pin	MCF5282 Signal	Pin	MCF5282 Signal
1	VSS	2	VSS
3	QSPIDI	4	GPTB1
5	QSPIDO	6	GPTB0
7	QSPICLK	8	QSPICS3
9	QSPICS2	10	DTOUT3/PTC2
11	QSPICS1	12	CLKOUT
13	QSPICS0	14	URXD2/PAS1
15	IRQ1	16	UTXD2/PAS0
17	DTIN3/PTC3	18	DTIN2/PTC1
19	DTIN2/PTD3	20	DTIN0/PTD1

Table 1-19. J7

Pin	MCF5282 Signal	Pin	MCF5282 Signal
1	VSS	2	VSS
3	IRQ2	4	N/C
5	IRQ3	6	N/C
7	IRQ4	8	N/C
9	IRQ5	10	N/C
11	DTOUT0/PTD0	12	N/C
13	DTOUT1/PTD2	14	N/C
15	DTOUT2/PTC0	16	N/C
17	N/C	18	N/C
19	IRQ6	20	IRQ7

1.5.2 Daughter Card Expansion Connectors

Two, 120-way SMT connectors (J2 and J3) provide access to all MCF5282 signals. These connectors are idea for interfacing to a custom daughter card or for simple probing of processor signals. Below is a pinout description of these connectors.

Table 1-20. J2

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	D31	2	A0	61	D9	62	A23
3	D30	4	GND	63	D8	64	$\overline{OE}$
5	D29	6	A1	65	D7	66	GND
7	D28	8	A2	67	GND	68	$\overline{TA}$
9	GND	10	A3	69	D6	70	$\overline{TEA}$
11	D27	12	A4	71	D5	72	R/$\overline{W}$
13	D26	14	A5	73	D4	74	+3.3V
15	+3.3V	16	A6	75	+3.3V	76	TSIZ0
17	D25	18	A7	77	D3	78	TSIZ1
19	D24	20	GND	79	D2	80	$\overline{TS}$
21	GND	22	A8	81	D1	82	$\overline{TIP}$
23	D23	24	A9	83	GND	84	$\overline{CS0}$
25	D22	26	+3.3V	85	D0	86	$\overline{CS1}$
27	+3.3V	28	A10	87	$\overline{SRAS}$	88	$\overline{CS2}$
29	D21	30	A11	89	$\overline{SCAS}$	90	+3.3V
31	D20	32	A12	91	+3.3V	92	$\overline{CS3}$
33	D19	34	GND	93	$\overline{SDWE}$	94	$\overline{BS0}$
35	GND	36	A13	95	+5V	96	$\overline{BS1}$
37	D18	38	A14	97	GND	98	GND
39	D17	40	A15	99	$\overline{SD\_CS0}$	100	$\overline{BS2}$
41	D16	42	+3.3V	101	$\overline{SD\_CS1}$	102	$\overline{BS3}$
43	+3.3V	44	A16	103	SCKE	104	$\overline{RSTI}$
45	D15	46	A17	105	+3.3V	106	$\overline{RSTO}$
47	D14	48	A18	107	CLKOUT	108	CLKMOD0
49	D13	50	GND	109	GND	110	$\overline{RCON}$
51	GND	52	A19	111	CLKMOD1	112	JTAG_EN
53	D12	54	A20	113	+5V	114	+5V
55	D11	56	A21	115	+5V	116	+5V
57	D10	58	+3.3V	117	GND	118	GND
59	+3.3V	60	A22	119	GND	120	GND

Table 1-21. J3

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
1	+3.3V	2	+3.3V	61	QSPIDO	62	GND
3	GND	4	VDDA	63	QSPIDI	64	SDA

Table 1-21. J3 (continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
5	$\overline{\text{IRQ1}}$	6	DDATA0	65	QSPICLK	66	SCL
7	$\overline{\text{IRQ2}}$	8	DDATA1	67	GND	68	GND
9	$\overline{\text{IRQ3}}$	10	+3.3V	69	QSPICS0	70	EXTAL
11	$\overline{\text{IRQ4}}$	12	DDATA2	71	QSPICS1	72	XTAL
13	$\overline{\text{IRQ5}}$	14	DDATA3	73	QSPICS2	74	GND
15	$\overline{\text{IRQ6}}$	16	PST0	75	+3.3V	76	GND
17	$\overline{\text{IRQ7}}$	18	VDDA	77	QSPICS3	78	+3.3V
19	+3.3V	20	PST1	79	URXD1	80	+3.3V
21	EMDIO	22	PST2	81	UTXD1	82	GND
23	EMDC	24	PST3	83	GND	84	GND
25	ETXCLK	26	GND	85	URXD0	86	+3.3V
27	ETXEN	28	DSO	87	UTXD0	88	AN52
29	ETXD0	30	DSI	89	GPTA0	90	AN53
31	ETXD1	32	GND	91	+3.3V	92	AN55
33	ETXD2	34	DSCLK	93	GPTA1	94	AN56
35	GND	36	TCLK	95	GPTA2	96	GND
37	ETXD3	38	+3.3V	97	GPTA3	98	AN0
39	ERXD0	40	$\overline{\text{BKPT}}$	99	GND	100	AN1
41	ERXD1	42	DTOUT0	101	GTPB0	102	VSSA
43	ERXD2	44	DTIN0	103	GTPB1	104	VSSA
45	GND	46	DTOUT1	105	GND	106	AN2
47	ERXD3	48	DTIN1	107	GTPB2	108	AN3
49	ECOL	50	DTOUT2	109	+3.3V	110	+3.3V
51	ERXCLK	52	DTIN2	111	GTPB3	112	VSSA
53	ERXDV	54	DTOUT3	113	+5V	114	+5V
55	ECRS	56	DTIN3	115	+5V	116	+5V
57	ETXER	58	CANRX	117	GND	118	GND
59	ERXER	60	CANTX	119	GND	120	GND

1.5.3 Reset Switch (S2)

The reset logic provides system initilization. Reset occurs during power-on or via assertion of the signal -RESET which causes the MCF5249 to reset. Reset is also triggered by the reset switch (S1) which resets the entire processor/system.

A hard reset and voltage sense controller (U12) is used to produce an active low power-on RESET signal. The reset switch S2 is fed into U12 which generates the signal which is fed to the MCF5282 reset, $\overline{\text{RSTI}}$.

The $\overline{\text{RSTI}}$ signal is an open collector signal and so can be wire OR'ed with other reset signals from additional peripherals.

dBUG configures the MCF5282 microprocessor internal resources during initialization. The instruction cache is invalidated and disabled. The Vector Base Register, VBR, contains an address which initially points to the Flash memory. The contents of the exception table are written to address 0x0000_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. The interrupt controller registers are initialized with unique interrupt level/priority pairs.

1.5.4 User LEDs

There are four LEDs available to the user. Each of these LEDs are pulled to +3.3V through a 470 ohm resistor and can be illuminated by driving a logic "0" on the appropriate signal to "sink" the current. Each of these signals can be disconnected from it's associated LED with a jumper. The table below which MCF5282 signal is associated with LED.

Table 1-22. User LEDs

LED	MCF5282 Signal	Jumper to disconnect
D6	DTOUT0	JP12
D7	DTOUT1	JP13
D8	DTOUT2	JP14
D9	DTOUT3	JP15

1.5.5 Other LEDs

There are several other LED on the M5282EVB to signal to the user various board/processor/component state. Below is a list of those LEDs and their functions:

Table 1-23. LED Functions

LED	Function
D1-D5	Ethernet Phy functionary (See AMD - AM79C874VC documentation)
D6-D9	User LEDs (See Table 1-22)
D11	+3.3V Power Good
D14	+5V Power Good
D15	Abort ($\overline{\text{IRQ7}}$) asserted
D16	Reset ($\overline{\text{RSTI}}$) asserted

Chapter 2
Initialization and Setup

2.1 System Configuration

The M5282EVB board requires the following items for minimum system configuration:

- The M5282EVB board (provided)
- Power supply, +6V to 14V DC with minimum of 300 mA
- RS232C compatible terminal or a PC with terminal emulation software
- RS232 communication cable (provided)

Figure 2-1 displays the minimum system configuration.

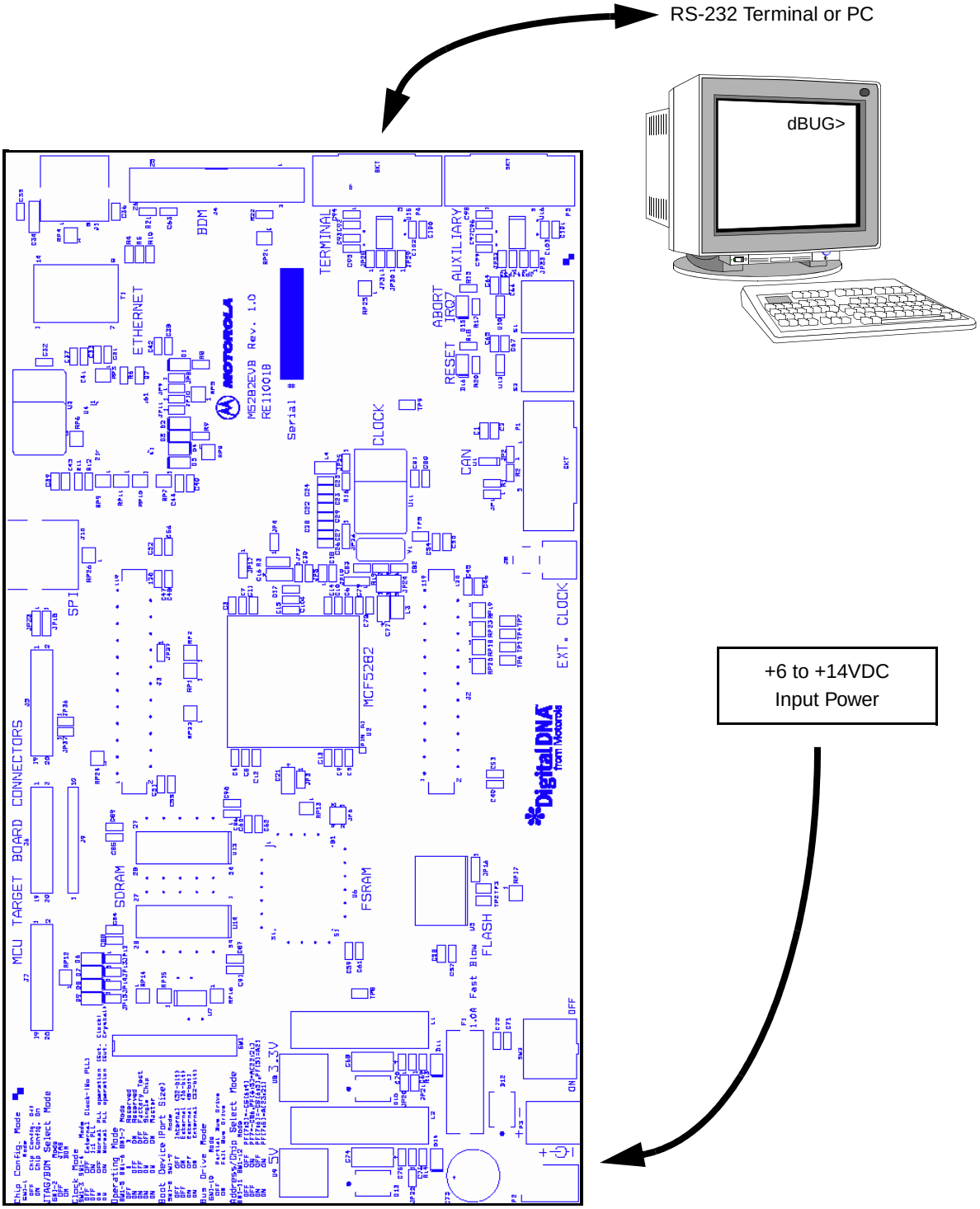


Figure 2-1. Minimum System Configuration

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 are in the default locations. Default jumper markings are documented on the master jumper table and printed on the underside of the board. After the board is functional in its default mode, the Ethernet interface may be used by following the instructions provided in Appendix A.

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:

- M5282EVB Single Board Computer
- M5282EVB User's Manual (this document)
- One RS232 communication cable
- One BDM (Background Debug Mode) “wiggler” cable
- MCF5282UM ColdFire Integrated Microprocessor User Manual
- ColdFire® Programmers Reference Manual
- A selection of 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 Matrix Design 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 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 P2 or P3. Connector P2 is a 2.1mm power jack, P3 a lever actuated connector. The board accepts +6V to +12V DC at 1.0 Amp via either of the connectors.

Table 2-1. Power Supply Connections on P2

Contact number	Voltage
1	Ground

Table 2-1. Power Supply Connections on P2 (continued)

Contact number	Voltage
2	N/C
3 (Center)	+6V to +14V DC

Table 2-2. Power Supply Connections on P3

Contact Number	Voltage
1	+6V to +14V DC
2	Ground

2.2.4 Selecting Terminal Baud Rate

The serial channel UART0 of the MCF5282 is used for serial communication and has a built in timer. This timer is used by the dBUG ROM monitor to generate the baud rate used to communicate with a serial terminal. 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.

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 necessary 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 RS-232 serial cable to connect the PC/terminal to the M5282EVB PCB. 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 P3 on the M5282EVB 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. The connector on the PC/terminal may be either male 25-pin or 9-pin. It may be necessary to obtain a 25pin-to-9pin adapter to make this connection. If an adapter is required, refer to Figure 2-2.

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 PROCOMM, KERMIT, QMODEM, Windows 95/98/2000/XP Hyper Terminal 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 necessary to select the baud rate and character format for the channel. Most terminal emulation software packages provide a command known as "Alt-p" (press the p

key while pressing the Alt key) to choose the baud rate and character format. The character format should be 8 bits, no parity, one stop bit. (see section 1.9.5 The Terminal Character Format.) The baud rate should be set to 19200. Power can now be applied to the board.

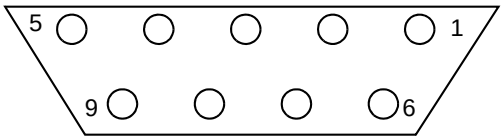


Figure 2-2. Pin Assignment for Female (Terminal) Connector

Pin assignments are as follows:

Table 2-3. Pin Assignment for Female (Terminal) Connector

DB9 Pin	Function
1	Data Carrier Detect, Output (shorted to pins 4 and 6)
2	Receive Data, Output from board (receive refers to terminal side)
3	Transmit Data, Input to board (transmit refers to terminal side)
4	Data Terminal Ready, Input (shorted to pin 1 and 6)
5	Signal Ground
6	Data Set Ready, Output (shorted to pins 1 and 4)
7	Request to Send, Input
8	Clear to send, Output
9	Not connected

Figure 2-3 shows the jumper locations for the board.

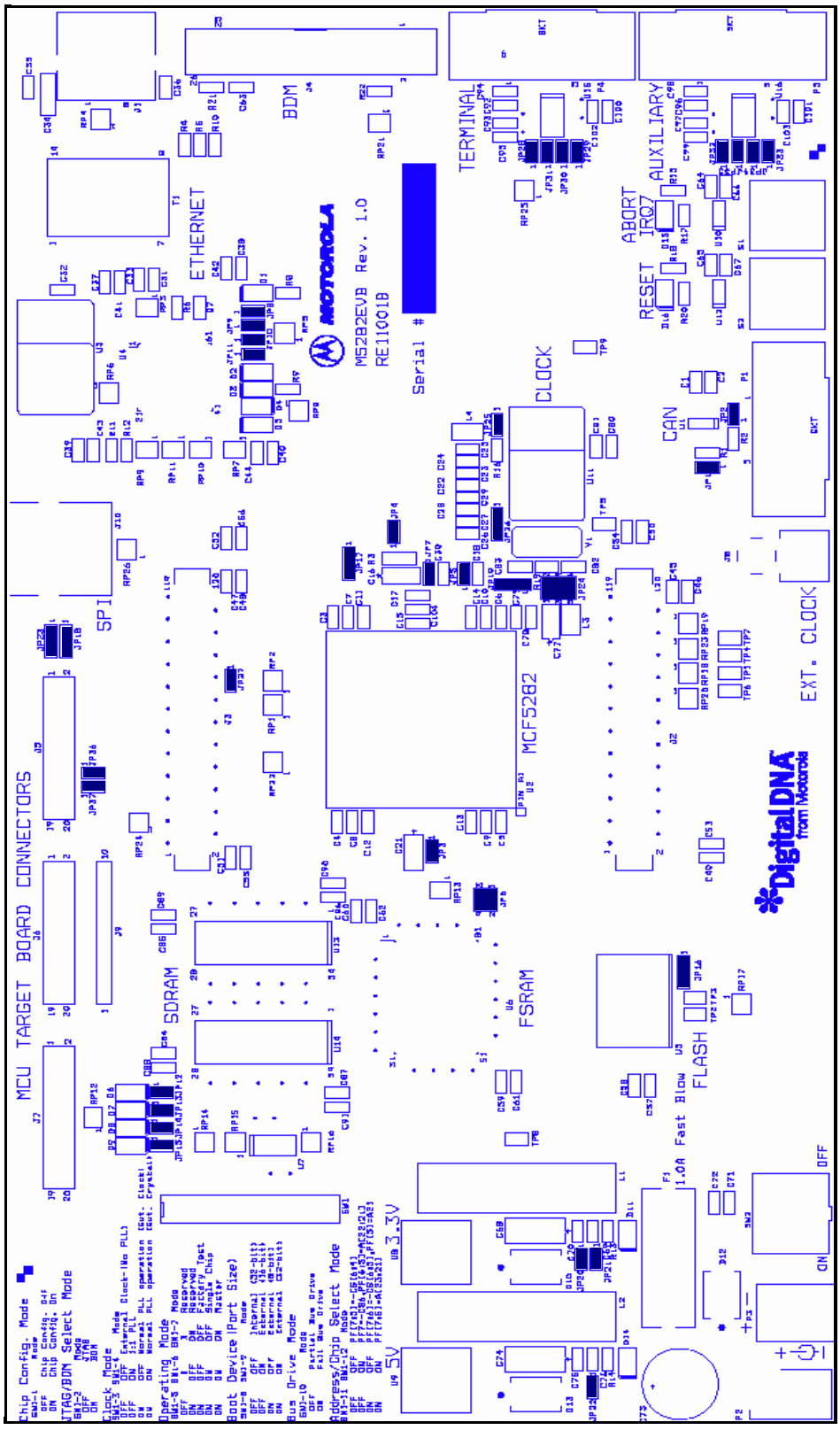


Figure 2-3. Jumper Locations

2.3 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, which includes the amount of memory present on the board.

```
Hard Reset
DRAM Size: 16M
Copyright 1995-2003 Motorola, Inc. All Rights Reserved.
ColdFire MCF5282 EVS Firmware v2e.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 2. 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 (~1A) and is connected to the board.
2. Check that the terminal and board are set for the same character format and baud.
3. 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 Matrix Design for further instructions, please see the beginning of this manual for contact details.

2.4 Using The BDM Port

The MCF5282 microprocessor has a built in debug module referred to as BDM (background debug module). In order to use BDM, simply connect the 26-pin debug connector on the board, J4, to the P&E BDM wiggler cable provided in the kit. No special setting is needed. Refer to the ColdFire® User's Manual BDM Section for additional instructions.

NOTE

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

Chapter 3
Using the Monitor/Debug Firmware

The M5282EVB 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 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 (8-N-1) with no flow control. The default baud rate is 19200 but can be changed after 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
- W 16-bit (word) access
- L 32-bit (long) access

When no <width> option is provided, the default width is.W, 16-bit.

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

- A0-A7
- D0-D7
- PC
- SR

All control registers on ColdFire® 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 seven, "A7."

3.2 Operational Procedure

System power-up and initial operation are described in detail in Chapter 1. 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 TERMINAL (P4) connector.
- Turn power on to the board.

Figure 3-1 shows the dBUG operational mode.

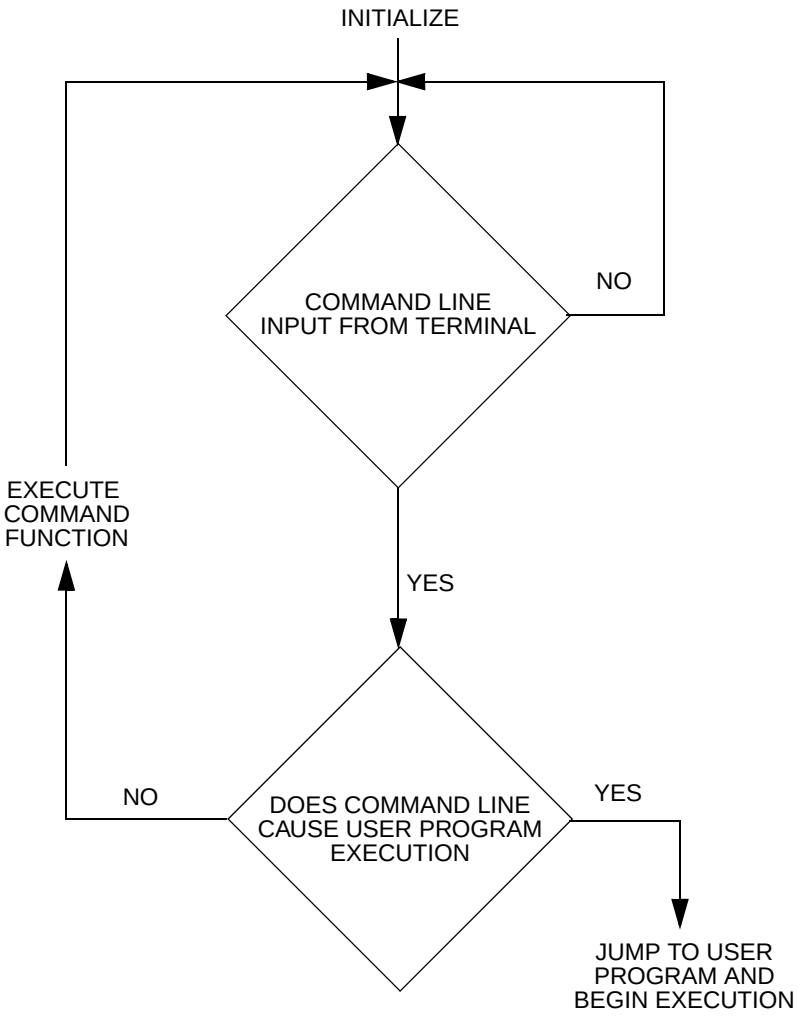


Figure 3-1. Flow Diagram of dBUG Operational Mode

3.2.2 System Initialization

After the EVB is powered-up and initialized, the terminal will display:

```
Low Voltage Detect Reset
Power-on Reset

ColdFire MCF5282 on the M5282EVB
Firmware vXX.XX.XX (Build X on XXXX)
Copyright 1995-2003 Motorola, Inc.  All Rights Reserved.

Enter 'help' for help.

dBUG>
```

Other means can be used to re-initialize the M5282EVB firmware. These means are discussed in the following paragraphs.

3.2.2.1 External RESET Button

External RESET (S2) is the red button. Depressing this button causes all processes to terminate, resets the MCF5282 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 ABORT Button

ABORT (S1) is the button located next to the RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5282) 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 ABORT button, the contents of the MCF5282 core internal registers are displayed.

The abort 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 the 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 (8-N-1). 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
FL	fl erase addr bytes fl write dest src bytes	Flash Utilities
GO	go <addr>	Execute
GT	gt addr	Execute To
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

Table 3-1. dBUG Command Summary (continued)

Mnemonic	Syntax	Description
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)
UP	up begin end filename	Upload Memory to File
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 0x00010000, the command is:

```
asm     10000 nop
```

To interactively assembly memory at address 0x00400000, 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 0x20000 and ending at 0x30000 is identical to the data starting at 0x80000, the command is:

```
bc      20000 80000 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 0x00020000 and ending at 0x00040000 with the value 0x1234, the command is:

```
bf      20000 40000 1234
```

To fill a block of memory starting at 0x00020000 and ending at 0x0004000 with a byte value of 0xAB, the command is:

```
bf.b    20000 40000 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 0x00020000 and ending at 0x00040000 with data that increments by 2 for each <width>, the command is:

```
bf      20000 40000 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 0x00040000 and ending at 0x00080000 to the location 0x00200000, the command is:

```
bm      40000 80000 200000
```

To copy the target code's data section (defined by the symbols data_start and data_end) to 0x00200000, the command is:

```
bm      data_start data_end 200000
```

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 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 16-bit value 0x1234 in the memory block starting at 0x00040000 and ending at 0x00080000:

```
bs       40000 80000 1234
```

This reads the 16-bit word located at 0x00040000 and compares it against the 16-bit value 0x1234. If no match is found, then the address is incremented to 0x00040002 and the next 16-bit value is read and compared.

To search for the 32-bit value 0xABCD in the memory block starting at 0x00040000 and ending at 0x00080000:

```
bs.l     40000 80000 ABCD
```

This reads the 32-bit word located at 0x00040000 and compares it against the 32-bit value 0x0000ABCD. If no match is found, then the address is incremented to 0x00040004 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 0x00040000, the command is:

```
di       40000
```

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 a serial port. The value for offset is converted according to the user-defined radix, normally hexadecimal. Please reference the ColdFire Microprocessor Family Programmer's Reference Manual for details on the S-Record format.

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.

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: DL <offset>

The DLDBUG command is used to update the dBUG image in Flash. It erases the Flash sectors containing the dBUG image, downloads a new dBUG image in S-record format obtained from the console, and programs the new dBUG image into Flash.

When the DLDBUG command is issued, dBUG will prompt the user for verification before any actions are taken. If the the command is affirmed, the Flash is erased and the user is prompted to begin sending the new dBUG S-record file. The file should be sent as a text file with no special transfer protocol.

CAUTION

Use this command with extreme caution, as any error can render dBUG useless!

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 Flash Utilities

Info Usage: FL

Erase Usage: FL erase addr bytes

Write Usage: FL write dest src bytes

The FL command provides a set of flash utilities that will display information about the Flash devices on the EVB, erase a specified range of Flash, or erase and program a specified range of Flash.

When issued with no parameters, the FL command will display usage information as well as device specific information for the Flash devices available. This information includes size, address range, protected range, access size, and sector boundaries.

When the erase command is given, the FL command will attempt to erase the number of bytes specified on the command line beginning at addr. If this range doesn't start and end on Flash sector boundaries, the range will be adjusted automatically and the user will be prompted for verification before proceeding.

When the write command is given, the FL command will program the number of bytes specified from src to dest. An erase of this region will first be attempted. As with the erase command, if the Flash range to be programmed doesn't start and end on Flash sector boundaries, the range will be adjusted and the user will be prompted for verification before the erase is performed. The specified range is also checked to insure that the entire destination range is valid within the same Flash device and that the src and dest are not within the same device.

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, trap #15 exception, 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 0x00040000, the command is:

```
go 40000
```

GT Execute To

Usage: GT addr

The GT command inserts a temporary 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
```

IRD Internal Register Display

Usage: IRD <module.register>

This command displays the internal registers of different modules inside the MCF5282. 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. Use the IRD command without any parameters to get a list of all the valid modules. Refer to the *MCF5282 User's Manual* for more information on these modules and the registers they contain.

Example:

```
ird     sim.rsr
```

IRM Internal Register Modify

Usage: IRM module.register data

This command modifies the contents of the internal registers of different modules inside the MCF5282. 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.

Example:

To modify the TMR register of the first Timer module to the value 0x0021, the command is:

```
irm      timer1.tmr 0021
```

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
```

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 longword data from address 0x20000, the command is:

```
lr.l    20000
```

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 longword data 0x12345678 to address 0x20000, the command is:

```
lw.l    20000 12345678
```

Note that the following command writes 0x78 into memory:

```
lw.b    20000 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 0x00400000, 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 0x00050000, the command is:

```
md.b     40000 50000
```

To display a range of 32-bit values starting at 0x00040000 and ending at 0x00050000:

```
md.l     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 0x00010000 to be 0xFF, the command is:

```
mm.b     10000 FF
```

To interactively modify memory beginning at 0x00010000, the command is:

```
mm       10000
```

MMAP Memory Map Display

Usage: mmap

This command displays the memory map information for the M5282EVB 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 and which regions of memory are reserved for dBUG use (protected).

Here is an example of the output from this command:

```
    Type           Start         End      Port Size
-------------------------------------------------
SDRAM            0x00000000   0x00FFFFFF   32-bit
SRAM (Int)       0x20000000   0x2000FFFF   32-bit
SRAM (Ext)       0x30000000   0x3007FFFF   32-bit
IPSBAR           0x40000000   0x7FFFFFFF   32-bit
Flash (Int)      0xF0000000   0xF007FFFF   32-bit
Flash (Ext)      0xFFE00000   0xFFFFFFFF   16-bit

 Protected         Start         End
---------------------------------------
dBUG Code        0xFFE00000   0xFFE3FFFF
dBUG Data        0x00000000   0x0000FFFF

  Chip Selects
----------------
CS0  Ext Flash
CS1  Ext SRAM
```

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: 00000000 SR: 2000 [t.Sm.000...xnzvc]
An: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 01000000
Dn: 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 register D0 on MC68000 and ColdFire to contain the value 0x1234, the command is:

```
rm      D0 1234
```

RESET Reset the Board and dBUG

Usage: RESET

The RESET command resets the board and dBUG to their 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 board adequately, cycle the power or press the reset button.

Examples:

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

```
reset
```

SD Stack Dump

Usage: SD

The SD command displays a back trace of stack frames. This command is useful after some user code has executed that creates stack frames (i.e. nested function calls). After control is returned to dBUG, the SD command will decode the stack frames and display a trace of the function calls.

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, 8-N-1, with no flow control.
- 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”.
- mac — This is the ethernet Media Access Control (MAC) 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
```

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: 0.0.0.0
      client: 0.0.0.0
     gateway: 0.0.0.0
     netmask: 255.255.255.0
    filename: test.s19
    filetype: S-Record
     ethaddr: 00:CF:52:82:CF:01
```

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 breakpoint one instruction beyond the current program counter and then executes the target code.

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 0x00040000, the command is:

```
symbol          -a main 40000
```

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
```

UP Upload Data

Usage: UP begin end filename

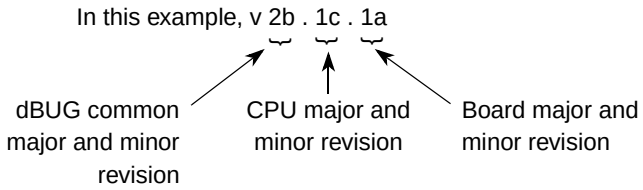
The UP command uploads the data from a memory region (specified by begin and end) to a file (specified by filename) over the network. The file created contains the raw binary data from the specified memory region. The UP command uses the Trivial File Transfer Protocol (TFTP) to transfer files to a network host.

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”.



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 TRAP #15 Functions

An additional utility within the dBUG firmware is a function called the TRAP 15 handler. This function can be called by the user program to utilize various routines within the dBUG, to perform a special task, and to return control to the dBUG. This section describes the TRAP 15 handler and how it is used.

There are four TRAP #15 functions. These are: OUT_CHAR, IN_CHAR, CHAR_PRESENT, and EXIT_TO_dBUG.

3.5.1 OUT_CHAR

This function (function code 0x0013) sends a character, which is in lower 8 bits of D1, to terminal.

Assembly example:

```
        /* assume d1 contains the character */
        move.l          #$0013,d0       Selects the function
        TRAP            #15             The character in d1 is sent to terminal
```

C example:

```
void board_out_char (int ch)
{
        /* If your C compiler produces a LINK/UNLK pair for this routine,
         * then use the following code which takes this into account
        */
#if  l
        /* LINK a6,#0 -- produced by C compiler */
        asm (" move.l8(a6),d1");            /* put 'ch'into d1 */
        asm (" move.l#0x0013,d0");  /* select the function */
        asm (" trap#15");                   /* make the call */
        /* UNLK a6  -- produced by C compiler */
#else
        /*  If C compiler does not produce a LINK/UNLK pair, the use
         *  the following code.
        */
         asm (" move.l4(sp),d1");           /* put 'ch'into d1 */
        asm (" move.l#0x0013,d0");  /* select the function */
        asm (" trap#15");                   /* make the call */
#endif
}
```

3.5.2 IN_CHAR

This function (function code 0x0010) returns an input character (from terminal) to the caller. The returned character is in D1.

Assembly example:

```
        move.l   #$0010,d0          Select the function
        trap     #15                Make the call, the input character is in d1.
```

C example:

```
int board_in_char (void)
{
        asm (" move.l#0x0010,d0");          /* select the function */
```

```
        asm (“ trap#15”);                   /* make the call */
        asm (“ move.ld1,d0”);               /* put the character in d0 */
}
```

3.5.3 CHAR_PRESENT

This function (function code 0x0014) checks if an input character is present to receive. A value of zero is returned in D0 when no character is present. A non-zero value in D0 means a character is present.

Assembly example:

```
        move.l   #$0014,d0         Select the function
        trap     #15       Make the call,  d0 contains the response (yes/no).
```

C example:

```
int board_char_present (void)
{
        asm (“ move.l#0x0014,d0”);          /* select the function */
        asm (“ trap#15”);                   /* make the call */
}
```

3.5.4 EXIT_TO_dBUG

This function (function code 0x0000) transfers the control back to the dBUG, by terminating the user code. The register context are preserved.

Assembly example:

```
        move.l   #$0000,d0         Select the function
        trap     #15               Make the call,  exit to dBUG.
```

C example:

```
void board_exit_to_dbug (void)
{
        asm (“ move.l#0x0000,d0”);          /* select the function */
        asm (“ trap#15”);                   /* exit and transfer to dBUG */
}
```

Appendix A
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.

A.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)

A.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 mac <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 MAC 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 mac 00:CF:52:82:EB: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.

A.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.

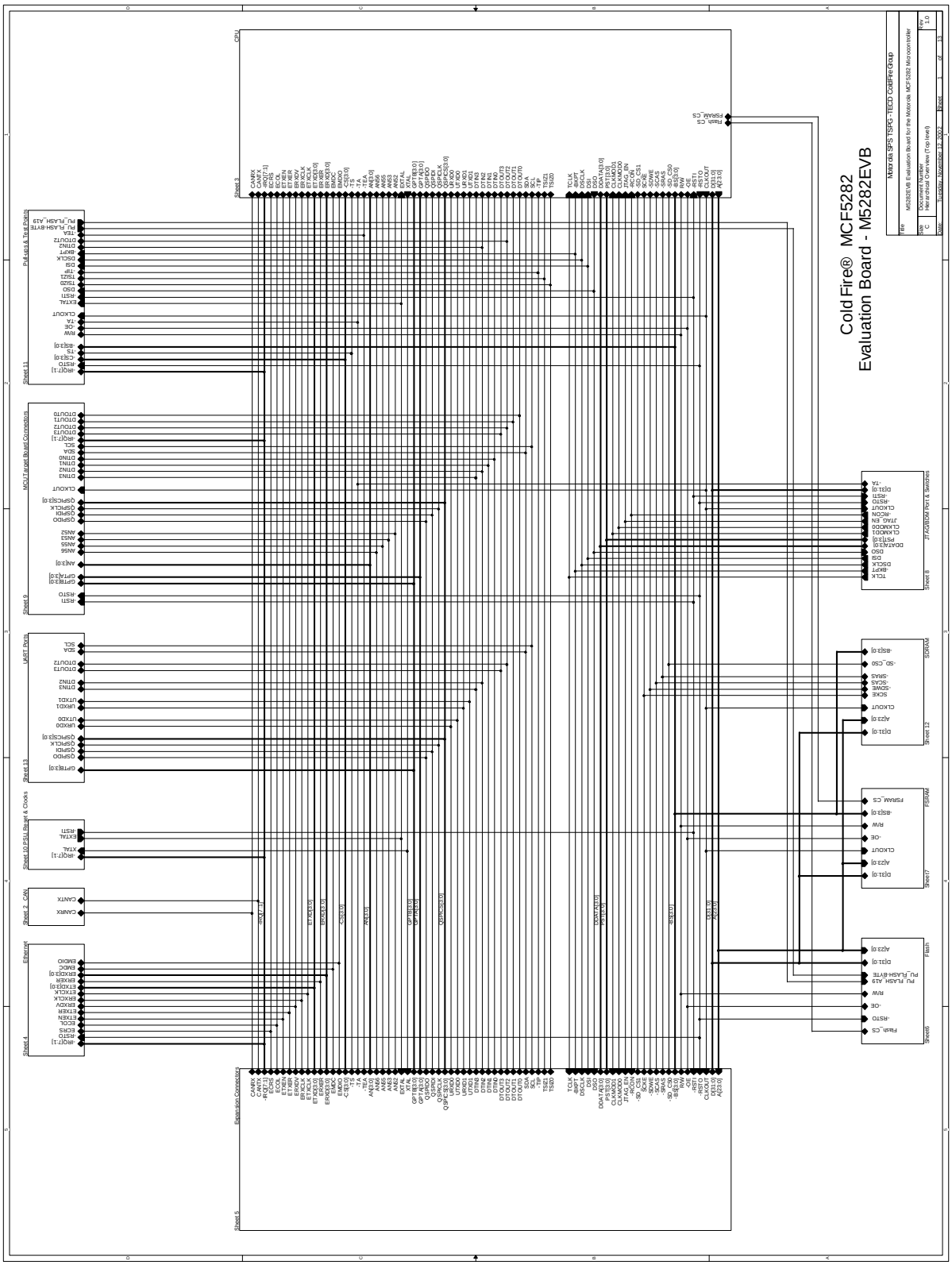
Using an IP address already assigned to another machine will cause dBUG network download to fail, and probably other severe network problems. Make certain the client IP address is unique for the board.

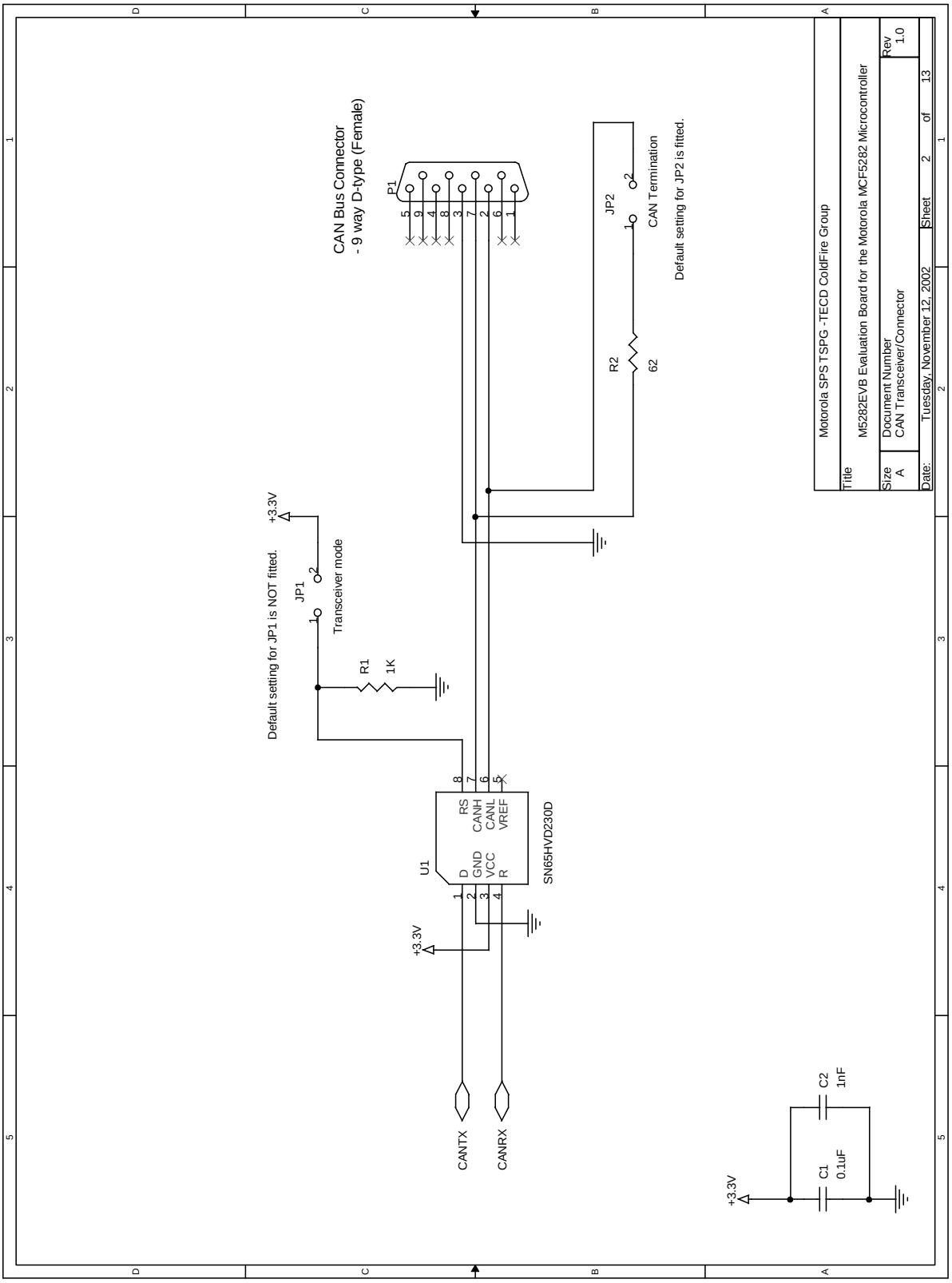
Check for proper insertion or connection of the network cable. Is the status LED lit indicating that network traffic is present?

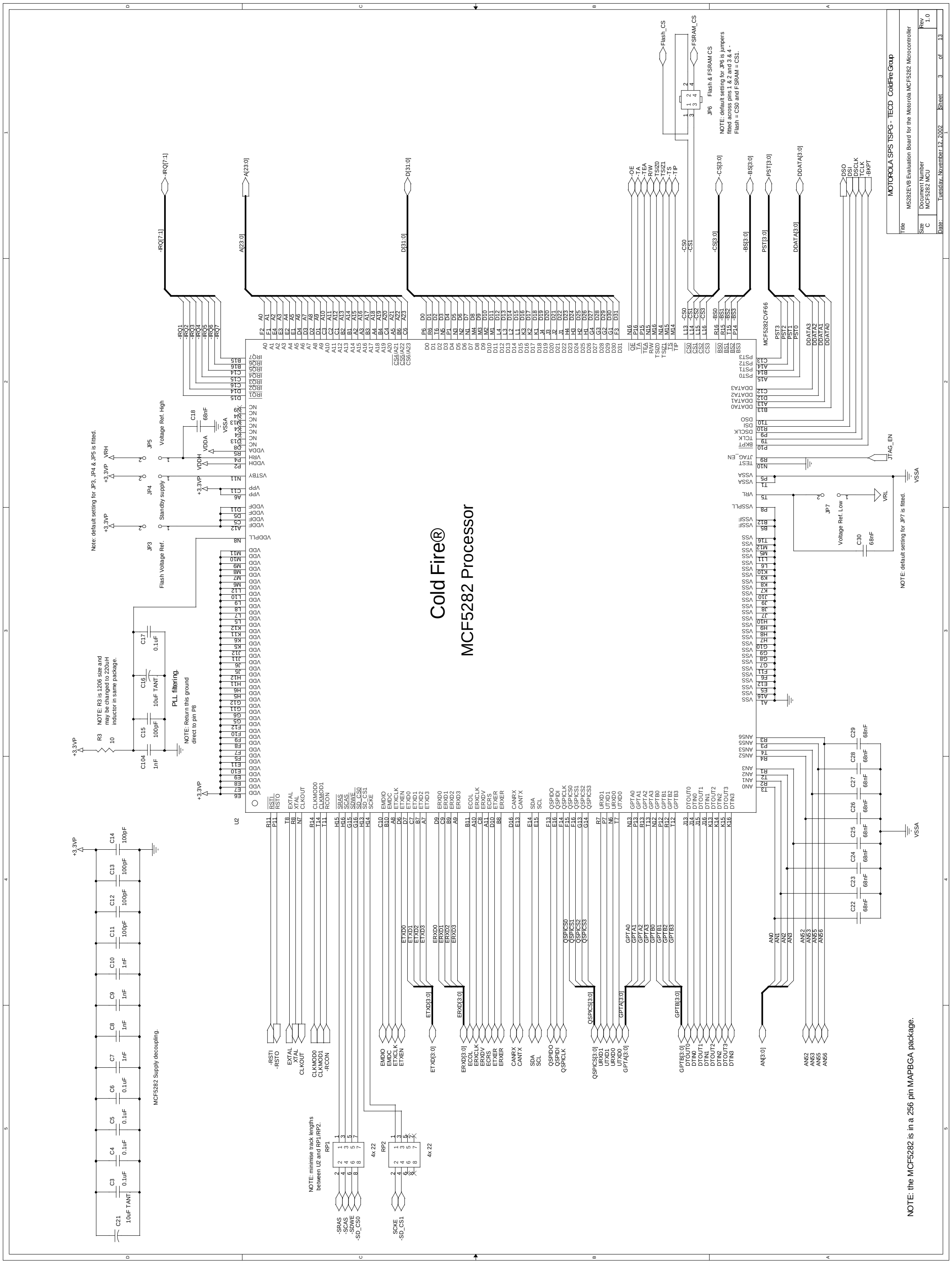
Check for proper configuration and operation of the TFTP server. Most Unix workstations can execute a command named ‘tftp’ which can be used to connect to the TFTP server as well. Is the default TFTP root directory present and readable?

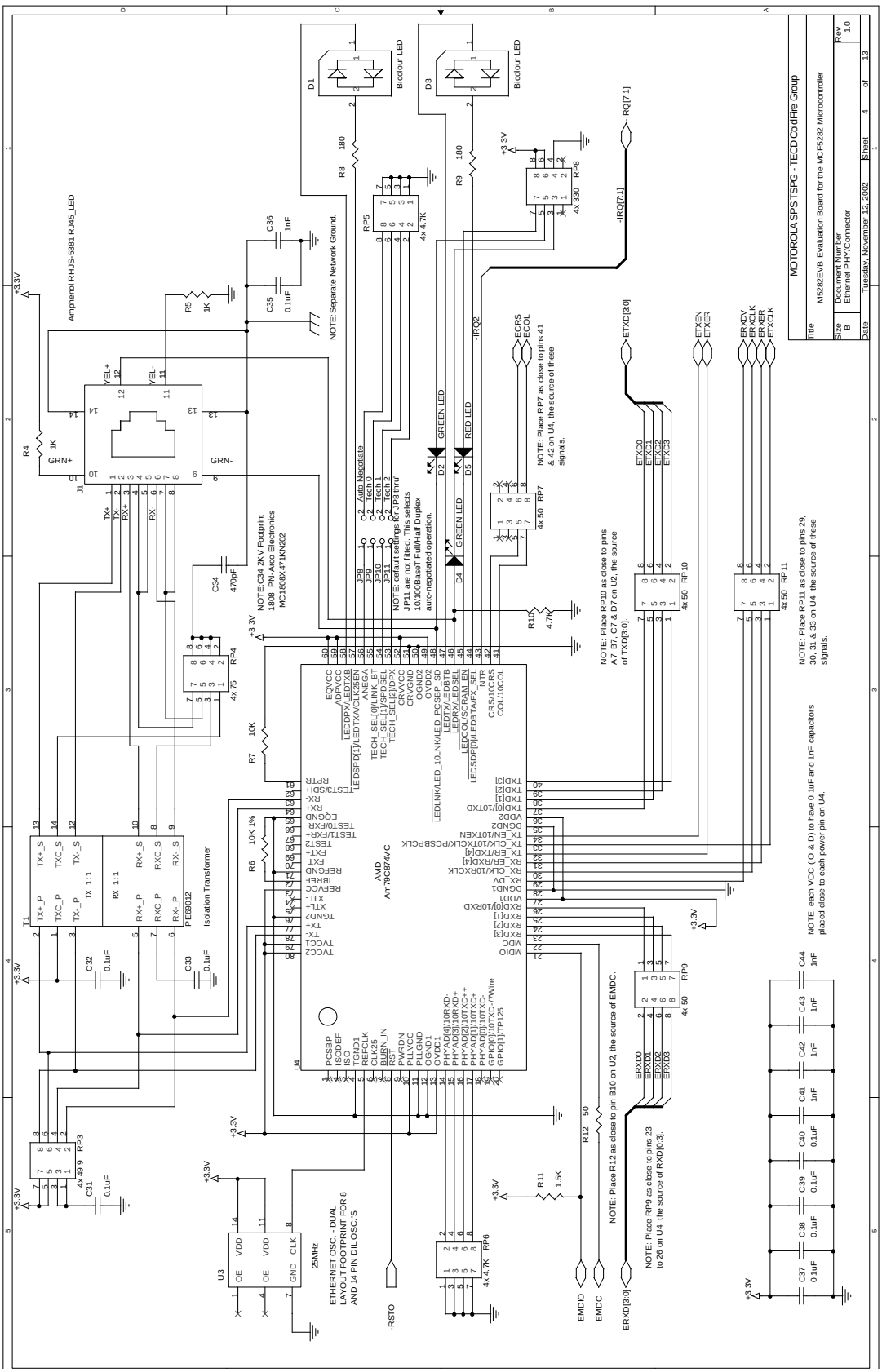
If ‘ICMP_DESTINATION_UNREACHABLE’ or similar ICMP message appears, then a serious error has occurred. Reset the board, and wait one minute for the TFTP server to time out and terminate any open connections. Verify that the IP addresses for the server and gateway are correct. Also verify that a TFTP server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

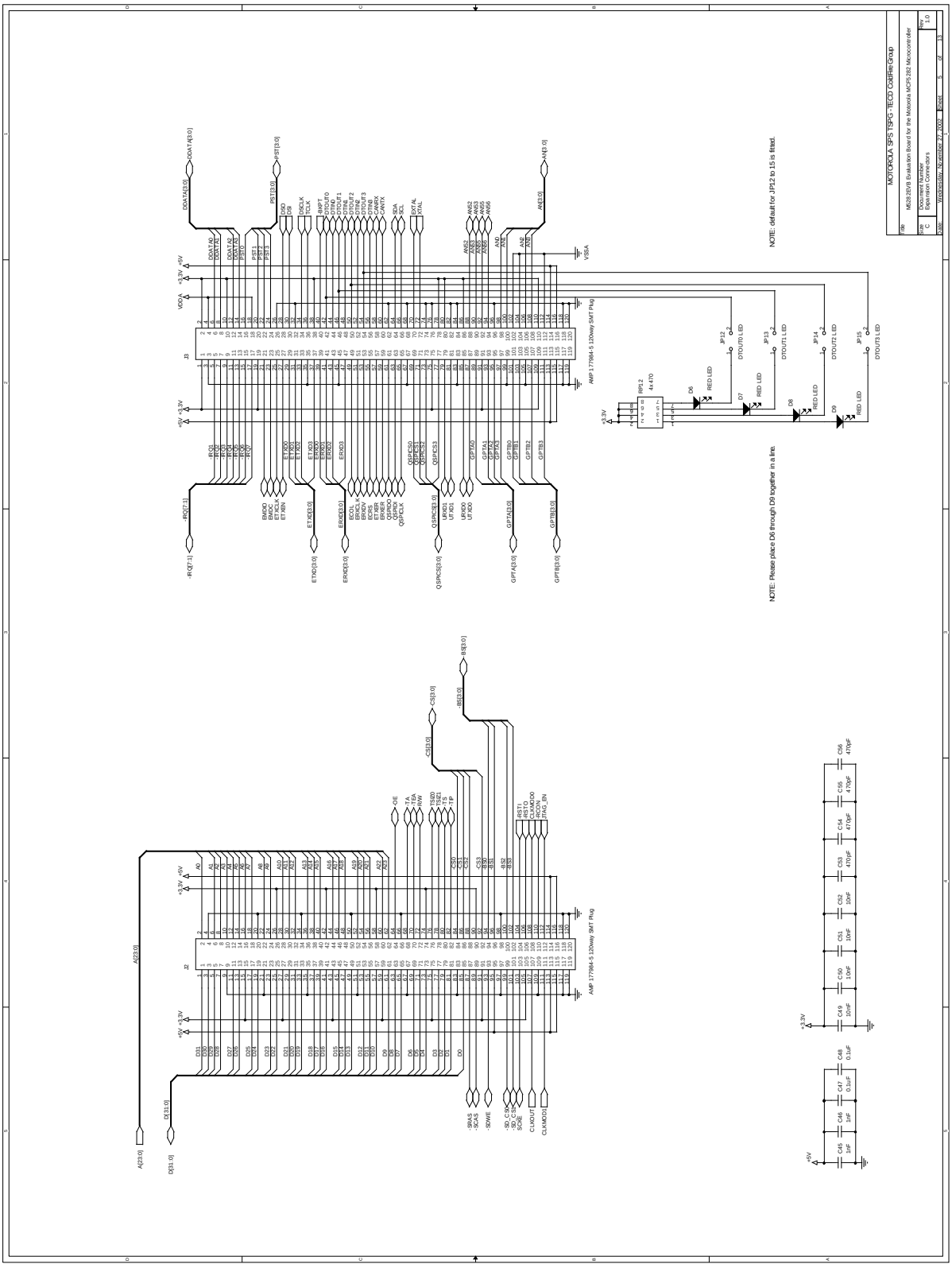
Appendix B
Schematics

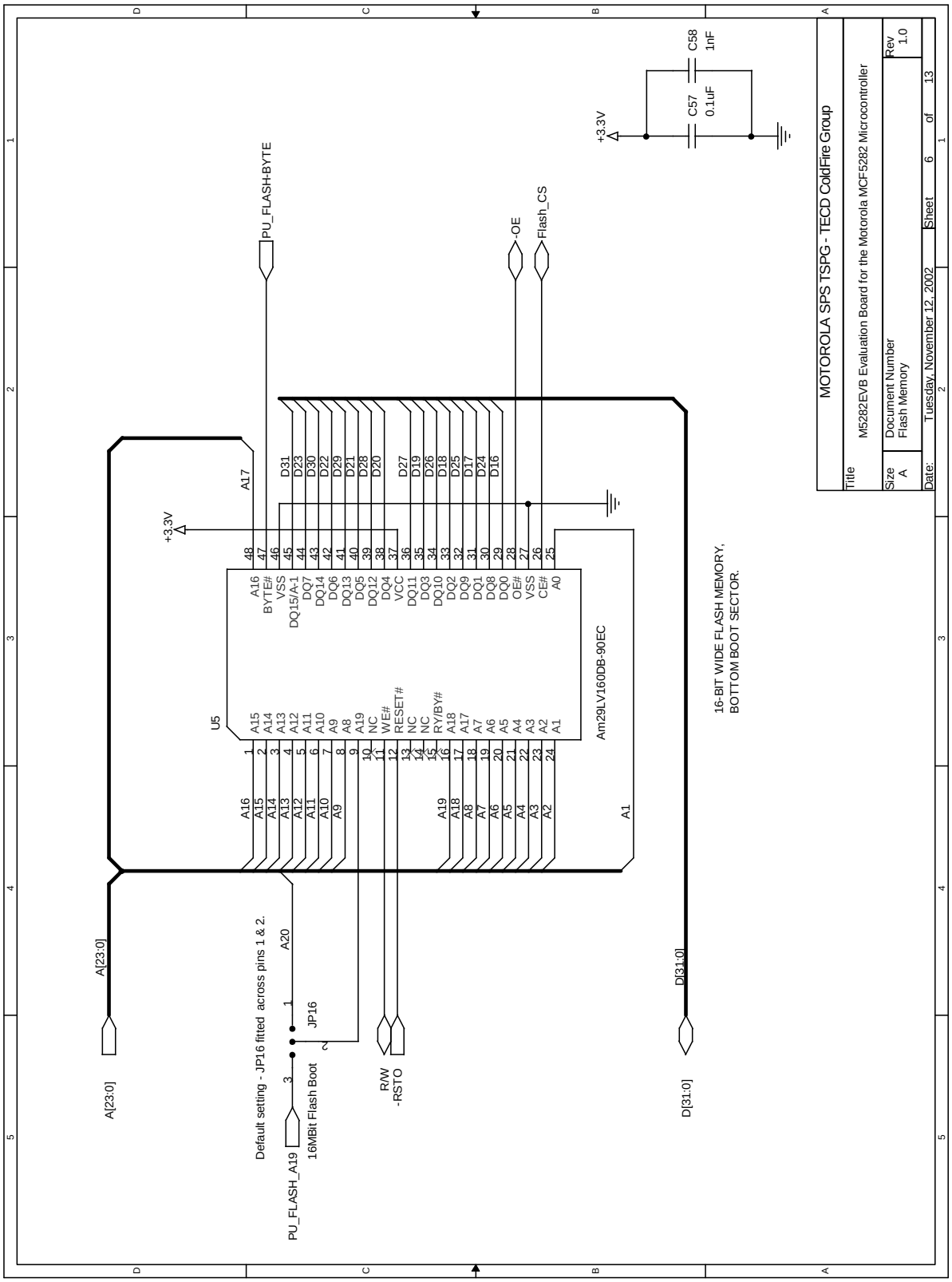

ColdFire® MCF5282
Evaluation Board - M5282EVB
Motorola SPS TSPG -TECD ColdFire Group
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Tuesday, November 12, 2002
CPU
Expansion Connector2
Ethernet
Sheet 2 - CAN
Sheet 10 FSIU Reset & Clocks
UART Ports
MCU Target Board Connectors
Pull-ups & Test Points
JTAG/BDM Port & Switches
Flash
FSRAM
SDRAM

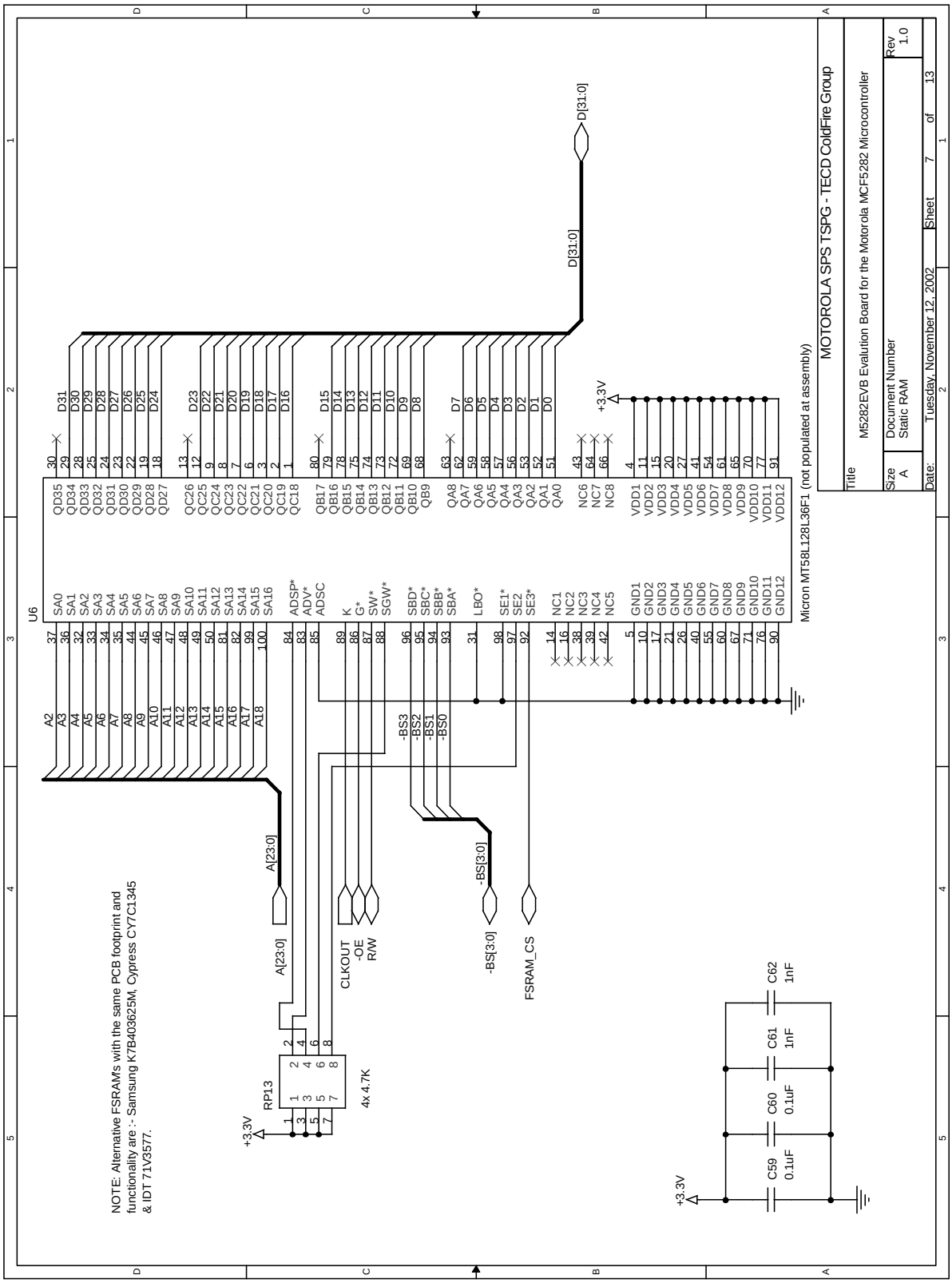

CAN Bus Connector
- 9 way D-type (Female)
P1
JP2
CAN Termination
Default setting for JP2 is fitted.
R2
62
+3.3V
JP1
Transceiver mode
Default setting for JP1 is NOT fitted.
R1
1K
U1
RS
CANH
CANL
VREF
D
GND
VCC
R
SN65HVD230D
+3.3V
CANTX
CANRX
C2
1nF
C1
0.1uF
+3.3V
Motorola SPS TSPG -TECD ColdFire Group
Title
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Size
A
Document Number
CAN Transceiver/Connector
Rev
1.0
Date:
Tuesday, November 12, 2002
Sheet
2
of
13

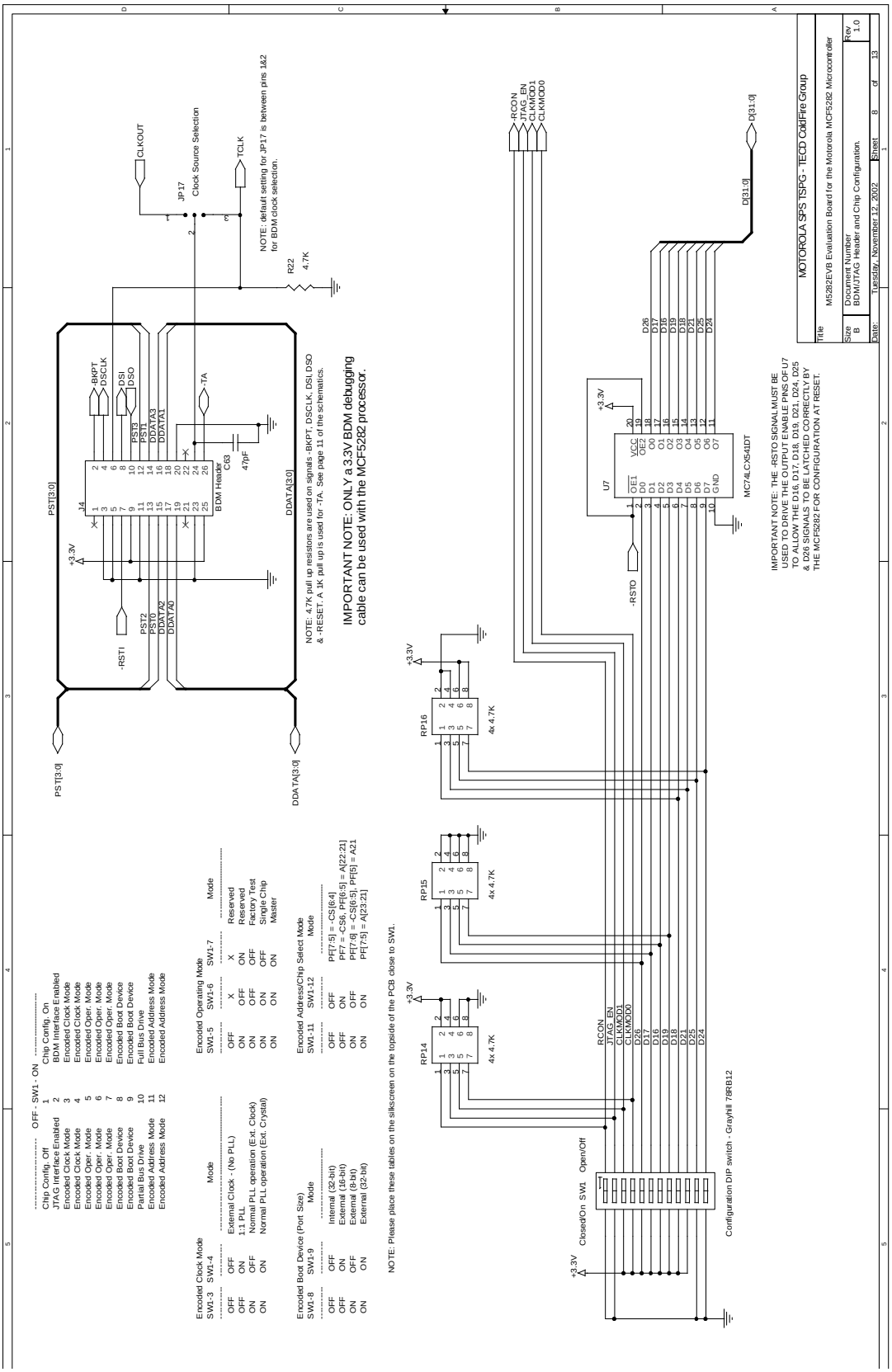

Cold Fire®
MCF5282 Processor
NOTE: the MCF5282 is in a 256 pin MAPBGA package.

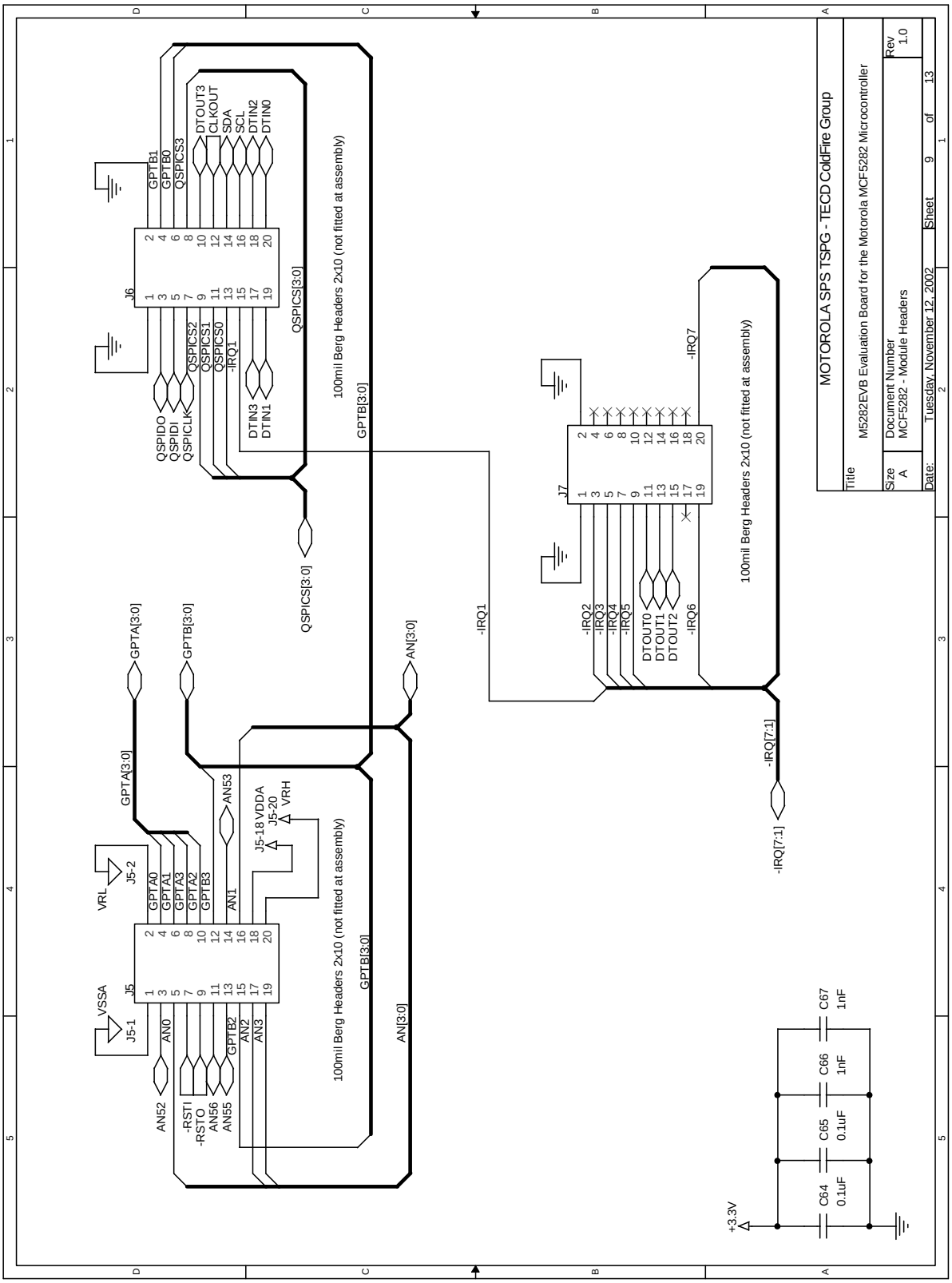

MOTOROLA SPS TSPG - TECD ColdFire Group
Title
M5282EVB Evaluation Board for the MCF5282 Microcontroller
Size
B
Document Number
Ethernet PHY/Connector
Rev
1.0
Date:
Tuesday, November 12, 2002
Sheet
4
of
13
Amphenol RHJS-5381 RJ45_LED
Bicolour LED
NOTE: Separate Network Ground.
NOTE:C34 2KV Footprint
1808 PN-Arco Electronics
MC1808X471KN202
GREEN LED
RED LED
NOTE: Place RP7 as close to pins 41 & 42 on U4, the source of these signals.
NOTE: Place RP10 as close to pins A7, B7, C7 & D7 on U2, the source of TX[3:0].
NOTE: Place RP11 as close to pins 29, 30, 31 & 33 on U4, the source of these signals.
Isolation Transformer
PE68012
AMD
Am79C874VC
NOTE: each VCC (IO & D) to have 0.1uF and 1nF capacitors placed close to each power pin on U4.
NOTE: Place R12 as close to pin B10 on U2, the source of EMDC.
NOTE: Place RP9 as close to pins 23 to 26 on U4, the source of RXD[0:3].
ETHERNET OSC. - DUAL LAYOUT FOOTPRINT FOR 8 AND 14 PIN DIL OSC.'S
25MHz
-RSTO
EMDIO
EMDC
ERXD[3:0]
-IRQ[7:1]
-IRQ2
ETXD[3:0]
ETXEN
ETXER
ERXDV
ERXCLK
ERXER
ETXCLK
ECRS
ECOL

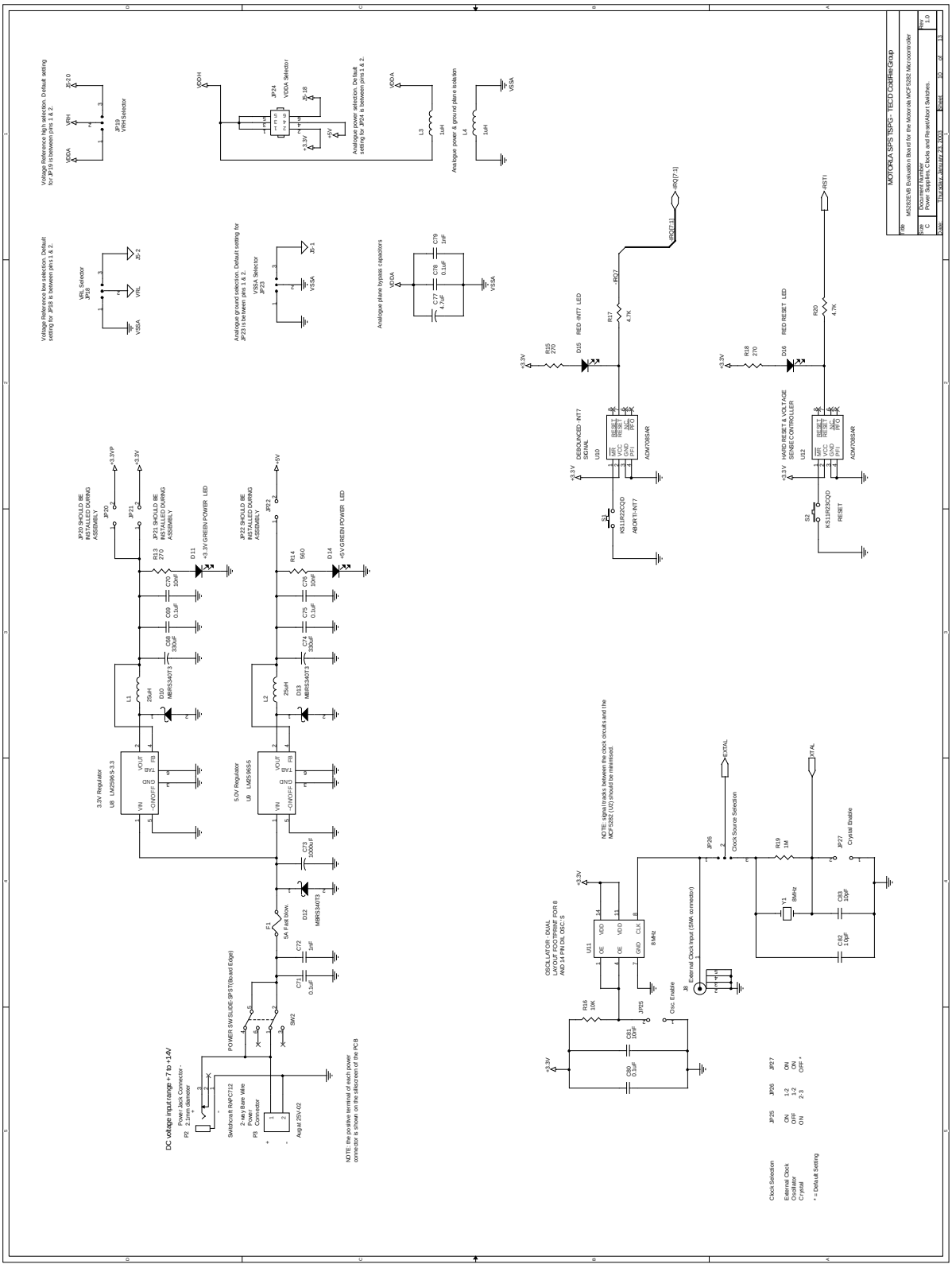

MOTOROLA SPS TSPG - TECD CodeWarGroup
MC9S12DB Evaluation Board for the Motorola MCPS 282 Microcontroller
Document Number
Electrical Connectors
Wednesday, September 27, 2002
NOTE: default for JP12 to 15 is fitted.
NOTE: Resise place D6 through D9 together in a line.
AMP 177984-5 120way SMT Plug
AMP 177984-5 120way SMT Plug

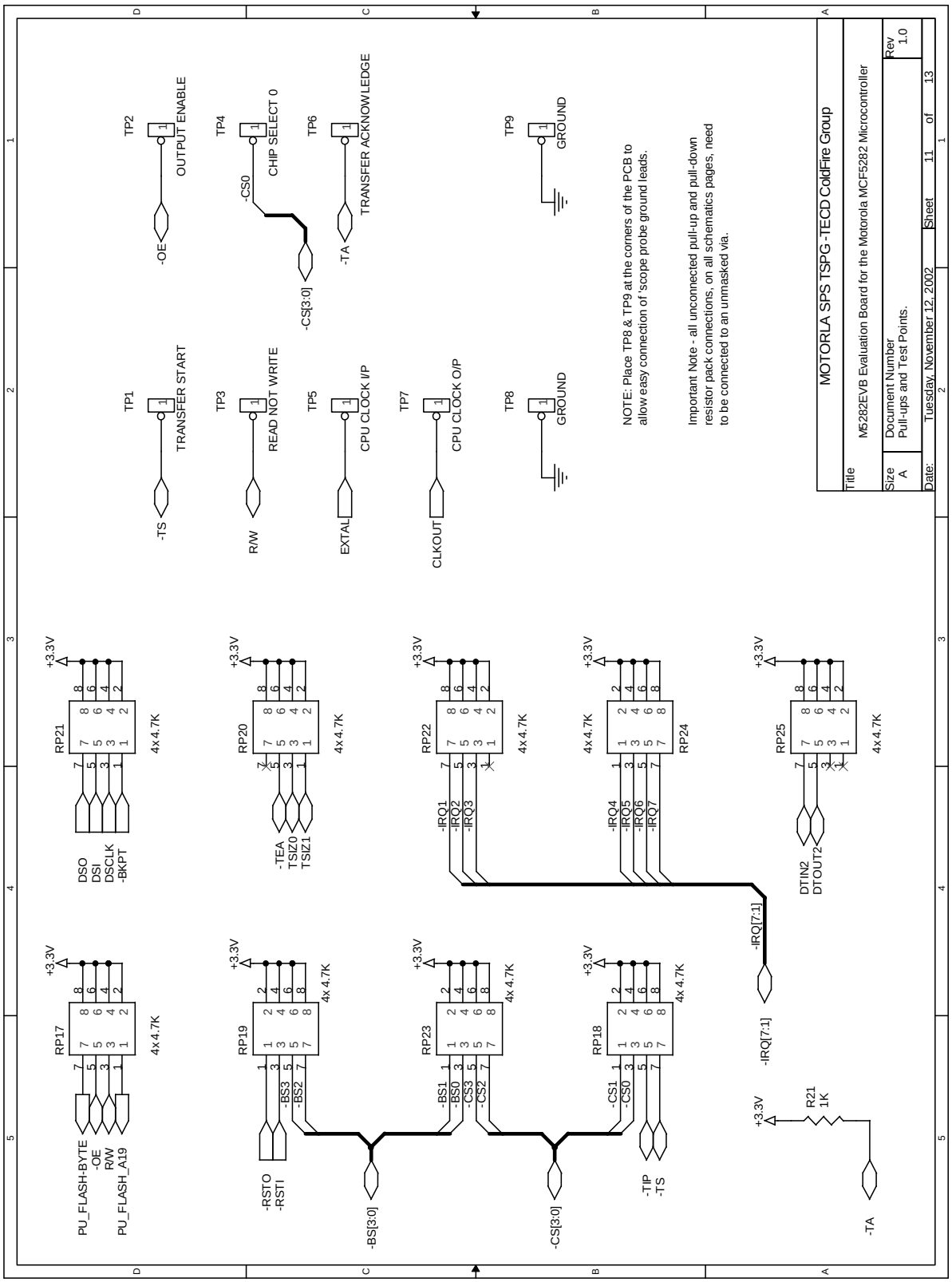

MOTOROLA SPS TSPG - TECD ColdFire Group
Title
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Size A
Document Number Flash Memory
Rev 1.0
Date: Tuesday, November 12, 2002
Sheet 6 of 13
C58 1nF
C57 0.1uF
+3.3V
PU_FLASH-BYTE
OE
Flash_CS
A17
U5
Am29LV160DB-90EC
16-BIT WIDE FLASH MEMORY,
BOTTOM BOOT SECTOR.
A[23:0]
D[31:0]
A20
JP16
Default setting - JP16 fitted across pins 1 & 2.
PU_FLASH_A19
16MBit Flash Boot
R/W
-RSTO

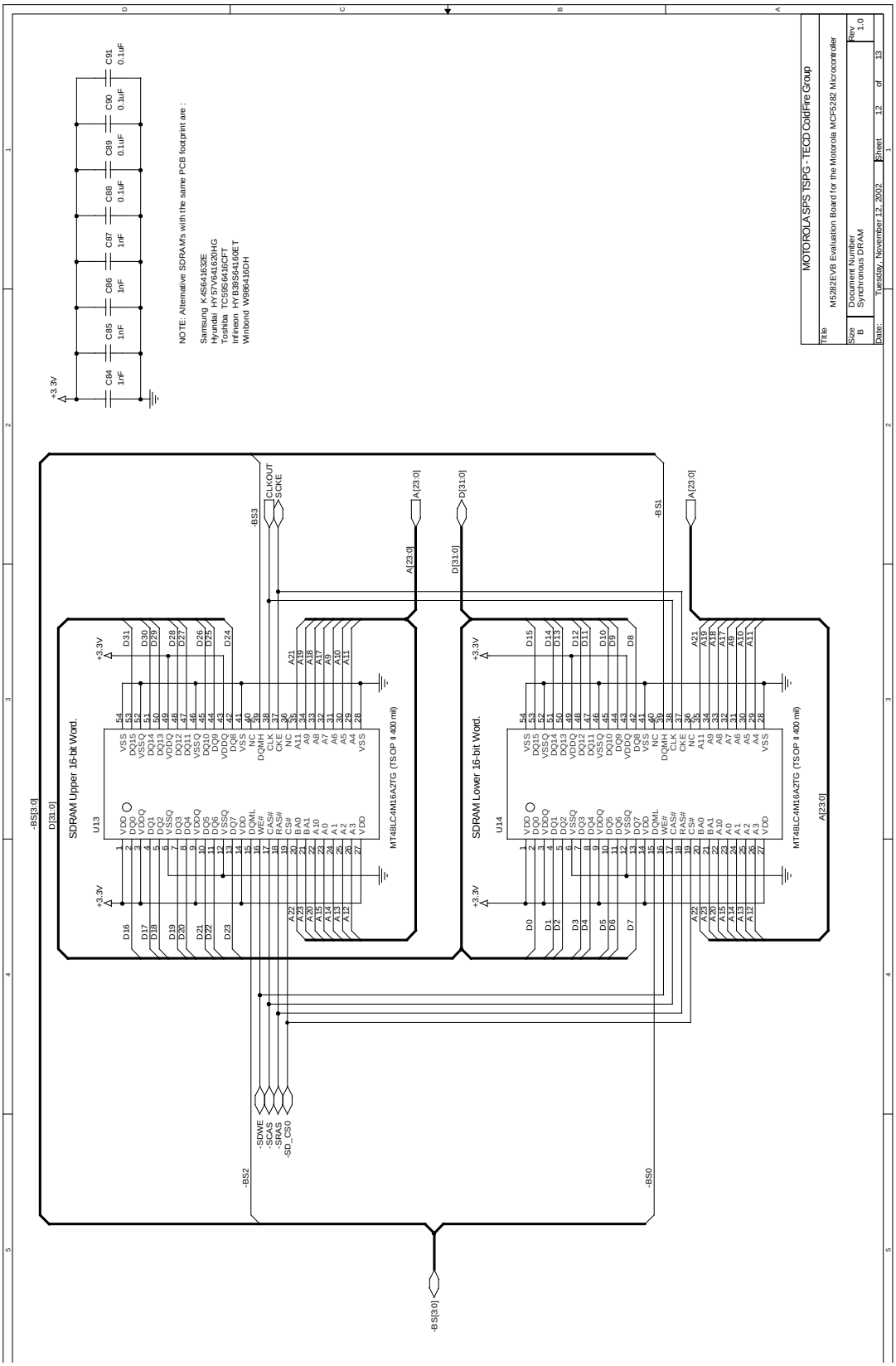

MOTOROLA SPS TSPG - TECD ColdFire Group
M5282EVB Evalution Board for the Motorola MCF5282 Microcontroller
Micron MT58L128L36F1 (not populated at assembly)
U6
D[31:0]
A[23:0]
-BS[3:0]
FSRAM_CS
CLKOUT
-OE
R/W
RP13
4x 4.7K
+3.3V
C59 0.1uF
C60 0.1uF
C61 1nF
C62 1nF
NOTE: Alternative FSRAMs with the same PCB footprint and functionality are :- Samsung K7B403625M, Cypress CY7C1345 & IDT 71V3577.

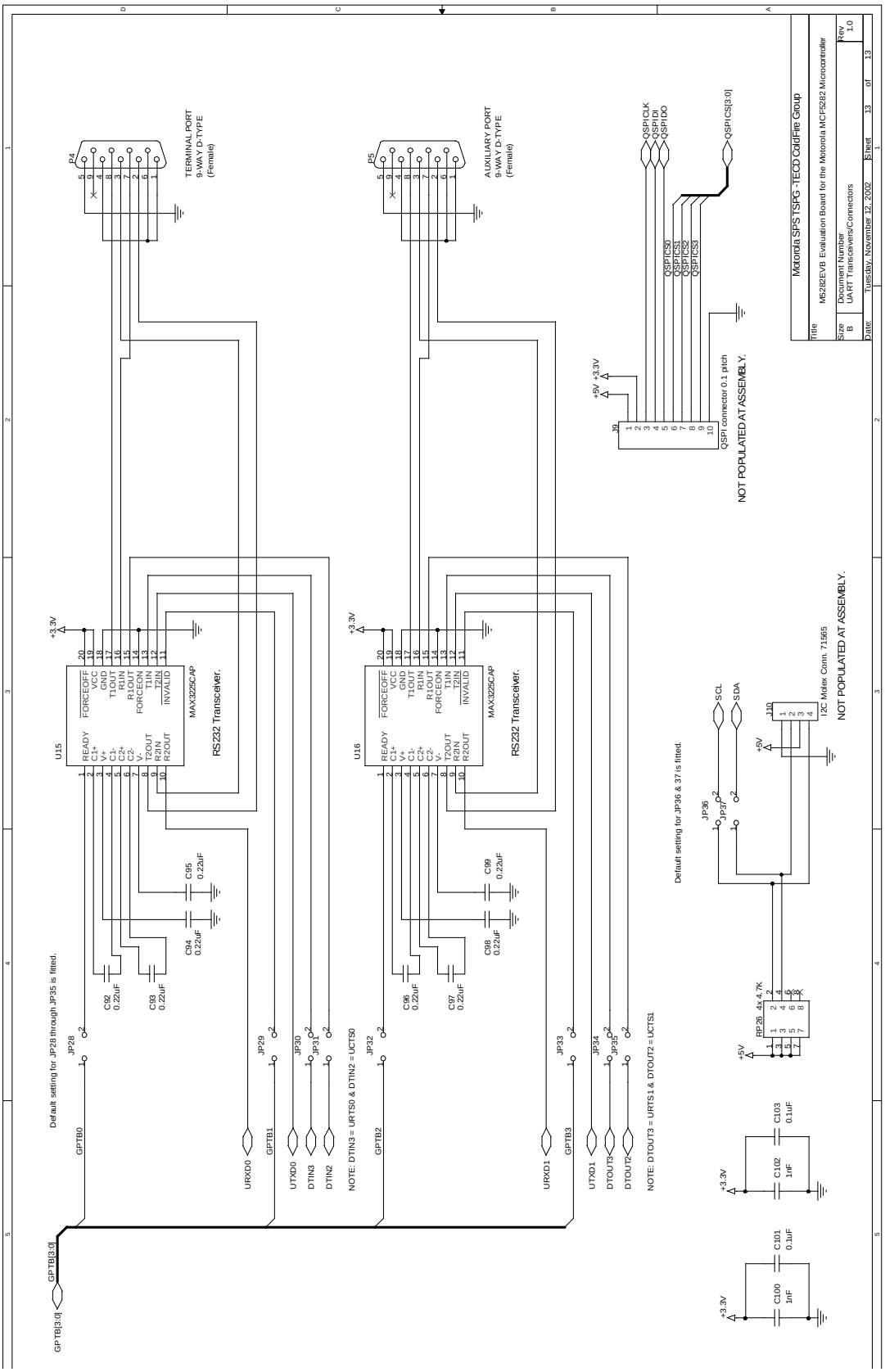

MOTOROLA SPS TSPG - TECD ColdFire Group
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Document Number: BDM/JTAG Header and Chip Configuration.
Rev 1.0
Tuesday, November 12, 2002
Sheet 8 of 13
JP17 Clock Source Selection
CLKOUT
TCLK
NOTE: default setting for JP17 is between pins 1&2 for BDM clock selection.
R22 4.7K
J4 BDM Header
C83 47pF
PST[3:0]
DDATA[3:0]
-BKPT
DSCLK
DSI
DSO
-TA
-RSTI
PST2
PST0
DDATA2
DDATA0
PST3
DDATA3
DDATA1
NOTE: 4.7K pull up resistors are used on signals -BKPT, DSCLK, DSI, DSO & -RESET. A 1K pull up is used for -TA. See page 11 of the schematics.
IMPORTANT NOTE: ONLY a 3.3V BDM debugging cable can be used with the MCF5282 processor.
Chip Config. Off 1
JTAG Interface Enabled 2
Encoded Clock Mode 3
Encoded Clock Mode 4
Encoded Oper. Mode 5
Encoded Oper. Mode 6
Encoded Oper. Mode 7
Encoded Boot Device 8
Encoded Boot Device 9
Partial Bus Drive 10
Encoded Address Mode 11
Encoded Address Mode 12
Encoded Clock Mode SW1-3 SW1-4
OFF OFF External Clock - (No PLL)
OFF ON 1:1 PLL
ON OFF Normal PLL operation (Ext. Clock)
ON ON Normal PLL operation (Ext. Crystal)
Encoded Operating Mode SW1-5 SW1-6 SW1-7
OFF X X Reserved
ON OFF ON Reserved
ON OFF OFF Factory Test
ON ON OFF Single Chip
ON ON ON Master
Encoded Boot Device (Port Size) SW1-8 SW1-9
OFF OFF Internal (32-bit)
OFF ON External (16-bit)
ON OFF External (8-bit)
ON ON External (32-bit)
Encoded Address/Chip Select Mode SW1-11 SW1-12
OFF OFF PF[7:5] = -CS[6:4]
OFF ON PF[7 = -CS6, PF[6:5] = A[22:21]
ON OFF PF[7:6] = -CS[6:5], PF[5] = A21
ON ON PF[7:5] = A[23:21]
NOTE: Please place these tables on the silkscreen on the topside of the PCB close to SW1.
Configuration DIP switch - Grayhill 78RB12
Closed/On SW1 Open/Off
RP14 4x 4.7K
RP15 4x 4.7K
RP16 4x 4.7K
RCON
JTAG_EN
CLKMOD1
CLKMOD0
D26
D17
D16
D19
D18
D21
D25
D24
-RSTO
U7 MC74LCX541DT
D[31:0]
IMPORTANT NOTE: THE -RSTO SIGNAL MUST BE USED TO DRIVE THE OUTPUT ENABLE PINS OF U7 TO ALLOW THE D16, D17, D18, D19, D21, D24, D25 & D26 SIGNALS TO BE LATCHED CORRECTLY BY THE MCF5282 FOR CONFIGURATION AT RESET.


MOTOROLA SPS TSPG - TECD ColdFire Group
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Document Number MCF5282 - Module Headers
Rev 1.0
Tuesday, November 12, 2002
Sheet 9 of 13
100mil Berg Headers 2x10 (not fitted at assembly)
J6
J7
J5
QSPICS[3:0]
GPTB[3:0]
GPTA[3:0]
AN[3:0]
-IRQ1
-IRQ[7:1]
QSPIDO
QSPIDI
QSPICLK
DTIN3
DTIN1
DTOUT3
CLKOUT
SDA
SCL
DTIN2
DTIN0
DTOUT0
DTOUT1
DTOUT2
VRL
VSSA
VRH
VDDA
AN52
AN0
AN56
AN55
AN53
AN1
AN2
AN3
+3.3V
C64 0.1uF
C65 0.1uF
C66 1nF
C67 1nF


MOTOROLA SPS TSPG - TECD ColdFire Group
MCF5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Document Number:
Power and Clocks and Re-set/Abort Switches
Rev 2.0
Sheet 10 of 13
Voltage Reference high selection. Default setting for JP19 is between pins 1 & 2.
Voltage Reference low selection. Default setting for JP18 is between pins 1 & 2.
Analogue ground selection. Default setting for JP23 is between pins 1 & 2.
Analogue power selection. Default setting for JP24 is between pins 1 & 2.
Analogue power & ground plane isolation
Analogue plane bypass capacitors
JP20 SHOULD BE INSTALLED DURING ASSEMBLY
JP21 SHOULD BE INSTALLED DURING ASSEMBLY
JP22 SHOULD BE INSTALLED DURING ASSEMBLY
+3.3V GREEN POWER LED
+5V GREEN POWER LED
3.3V Regulator
5.0V Regulator
DC voltage input range +7 to +14V
Power Jack Connector - 2.1mm diameter
Switchcraft RAPC712
2-way Bare Wire Power Connector
Augat 25V-02
POWER SW SLIDE SPST(Board Edge)
5A Fast blow.
NOTE: the positive terminal of each power connector is shown on the silkscreen of the PCB
OSCILLATOR - DUAL LAYOUT FOOTPRINT FOR 8 AND 14 PIN DIL OSC - S
NOTE: signal tracks between the clock circuits and the MCF5282 (U3) should be minimised.
Clock Source Selection
External Clock Input (SMA connector)
Osc Enable
Crystal Enable
EXTAL
XTAL
Clock Selection
External Clock
Oscillator
Crystal
* = Default Setting
DEBOUNCED INT7 SIGNAL
HARD RESET & VOLTAGE SENSE CONTROLLER
RED INT7 LED
RED RESET LED
ADM708SAR
KS11R22CQD
RESET
IRQ7 [7-1]
RST1


MOTORLA SPS TSPG -TECD ColdFire Group
Title
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Size A
Document Number
Pull-ups and Test Points.
Rev 1.0
Date: Tuesday, November 12, 2002
Sheet 11 of 13
TP1 TRANSFER START -TS
TP2 OUTPUT ENABLE -OE
TP3 READ NOT WRITE R/W
TP4 CHIP SELECT 0 -CS0
TP5 CPU CLOCK I/P EXTAL
TP6 TRANSFER ACKNOWLEDGE -TA
TP7 CPU CLOCK O/P CLKOUT
TP8 GROUND
TP9 GROUND
-CS[3:0]
NOTE: Place TP8 & TP9 at the corners of the PCB to allow easy connection of 'scope probe ground leads.
Important Note - all unconnected pull-up and pull-down resistor pack connections, on all schematics pages, need to be connected to an unmasked via.
RP17 4x4.7K PU_FLASH-BYTE -OE R/W PU_FLASH_A19
RP21 4x4.7K DSO DSI DSCLK -BKPT
RP20 4x4.7K -TEA TSIZ0 TSIZ1
RP22 4x4.7K -IRQ1 -IRQ2 -IRQ3
RP24 4x4.7K -IRQ4 -IRQ5 -IRQ6 -IRQ7
RP25 4x4.7K DTIN2 DTOUT2
RP19 4x4.7K -RSTO -RSTI -BS3 -BS2 -BS[3:0]
RP23 4x4.7K -BS1 -BS0 -CS3 -CS2 -CS[3:0]
RP18 4x4.7K -CS1 -CS0 -TIP -TS
-IRQ[7:1]
R21 1K
+3.3V
-TA


MOTOROLA SPS TSPG - TECD ColdFire Group
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Document Number
Synchronous DRAM
Rev 1.0
Tuesday, November 12, 2002
Sheet 12 of 13
NOTE: Alternative SDRAMs with the same PCB footprint are :
Samsung K4S641632E
Hyundai HY57V641620HG
Toshiba TC59S6416CFT
Infineon HY63D54616ET
Winbond W986416DH
SDRAM Upper 16-bit Word.
SDRAM Lower 16-bit Word.
MT48LC4M16A2TG (TSOP II 400 mil)
-BS3
-BS2
-BS1
-BS0
-BS[3:0]
D[31:0]
A[23:0]
CLKOUT
SCKE
-SDWE
-SCAS
-SRAS
-SD_CS0
+3.3V


Motorola SPS TSPG -TECD ColdFire Group
M5282EVB Evaluation Board for the Motorola MCF5282 Microcontroller
Document Number
UART Transceivers/Connectors
Rev 1.0
Tuesday, November 12, 2002
Sheet 13 of 13
TERMINAL PORT 9-WAY D-TYPE (Female)
AUXILIARY PORT 9-WAY D-TYPE (Female)
MAX3225CAP RS232 Transceiver.
QSPI connector 0.1 pitch
NOT POPULATED AT ASSEMBLY.
I2C Molex Conn. 71565
Default setting for JP28 through JP35 is fitted.
NOTE: DTIN3 = URTS0 & DTIN2 = UCTS0
Default setting for JP36 & 37 is fitted.
NOTE: DTOUT3 = URTS1 & DTOUT2 = UCTS1

Appendix C
Evaluation Board BOM

Table C-1. MCF5282EVB BOM

Item	Qty	Reference	Part	Function
1	32	C1,C3,C4,C5,C6,C17,C31, C32,C33,C35,C37,C38,C39, C40,C47,C48,C57,C59,C60, C64,C65,C69,C71,C75,C78, C80,C88,C89,C90,C91,C92,C93, C94,C95,C96,C97,C98,C99,C101, C103	0.1uF	SMT Decoupling Capacitors
2	26	C2,C7,C8,C9,C10,C36,C41, C42,C43,C44,C45,C46,C58, C61,C62,C66,C67,C72,C79, C84,C85,C86,C87,C100, C102,C104	1nF	SMT Decoupling Capacitors
3	5	C11,C12,C13,C14,C15	100pF	SMT Capacitors
4	2	C16,C21	10uF TANT	SMT Capacitors
5	10	C18,C22,C23,C24,C25,C26, C27,C28,C29,C30	68nF	SMT Capacitors
6	5	C34,C53,C54,C55,C56	470pF	SMT Capacitors
7	7	C49,C50,C51,C52,C70,C76, C81	10nF	SMT Capacitors
8	3	C68,C74	AVX TPSE337K10CLR	SMT Capacitors
9	1	C73	Rubycon 1000uF 35V	SMT Capacitors
10	1	C77	4.7uF Tant	Capacitor
11	2	C83,C82	10pF	SMT Capacitors
12	2	D3,D1	LSGT670 Bicolour LED	Infineon SMT LEDs
13	2	D2,D4	LGT670 GREEN LED	Infineon SMT LEDs
14	3	D5,D15,D16	LST670 RED LED	Infineon SMT LEDs
15	4	D6,D7,D8,D9	LTL-94PURK-TA	LED
16	3	D10,D12,D13	MBRS340T3	LED
17	2	D14,D1	LTL-94PGK-TA	LED
18	1	F1	MULTICOMP MCHTE-15M	5A Fast blow fuse

Table C-1. MCF5282EVB BOM (continued)

Item	Qty	Reference	Part	Function
19	5	JP16, JP17, JP18, JP19, JP23	Harwin M22-2010305	3-way jumper
20	30	JP1, JP2, JP3, JP4, JP5, JP7, JP8, JP9, JP10, JP11, JP12, JP13, JP14, JP15, JP20, JP21, JP22, JP25, JP26, JP27, JP28, JP29, JP30, JP31, JP32, JP33, JP34, JP35, JP36, JP37	Harwin M22-2010205	2-way jumper
21	1	JP6	Samtec 2x2 male header, 2mm	2x2 jumper
22	1	JP24	Samtec 2x3 male header, 2mm	2x3 jumper
23	1	J1	Amphenol RHJS-5381 RJ45_LED	RJ45 connector
24	2	J3,J2	AMP 177984-5	120way SMT Receptacle
25	1	J4	Thomas&Betts 609-2627	BDM 26-way header
26	3	J5,J6,J7		100mil Berg Headers 2x10
27	1	J8	1053378-1	External Clock Input (SMA connector)
28	1	J9		QSPI connector 0.1 pitch
29	1	J10	I2C Molex Conn. 71565	I2C connector
30	4	L1,L2,L3,L4	SIEMENS B82111-B-C24	25uH Inductors
31	3	P1,P4,P5	Molex DB9 conn.	DB9 RS232 PORT
32	1	P2	Switchcraft RAPC712	PSU barrel connector
33	1	P3	Augat 25V-02	2-way bare wire power connector
34	2	RP1,RP2	Philips	SMT 4 x 22 resistor packs
35	1	RP3	Philips	SMT 4 x 49.9 resistor packs
36	1	RP4	Philips	SMT 4 x 75 resistor packs
37	16	RP5,RP6,RP13,RP14,RP15, RP16,RP17,RP18,RP19,RP20, RP21,RP22,RP23,RP24,RP25, RP26		SMT 4x 4.7K resistor packs
38	4	RP7,RP9,RP10,RP11	Panasonic	SMT 4x 50 resistor packs
39	1	RP8	Panasonic	SMT 4x 330 resistor packs
40	1	RP12	Panasonic	SMT 4x 470 resistor packs
41	4	R1,R4,R5,R21	Panasonic	SMT 1K resistor
42	1	R2	Panasonic	SMT 62 resistor

Table C-1. MCF5282EVB BOM (continued)

Item	Qty	Reference	Part	Function
43	1	R3	Panasonic	SMT 10 resistor
44	1	R6	Panasonic	SMT 10K 1% resistor
45	2	R16,R7	Panasonic	SMT 10K resistor
46	2	R9,R8	Panasonic	SMT 180 resistor
47	4	R10,R22	Panasonic	SMT 4.7K resistor
48	1	R11	Panasonic	SMT 5K resistor
49	1	R12	Panasonic	SMT 50 resistor
50	3	R13,R15,R18	Panasonic	SMT 270 resistor
51	1	R14	Panasonic	SMT 560 resistor
52	1	R17,R20	Panasonic	SMT 100 resistor
53	1	SW1	Grayhill 78RB12	Configuration DIP switch
54	1	SW2	EAO Switch	POWER SW SLIDE-SPST(Board Edge)
55	1	S1	C&K KS11R22CQD	$\overline{\text{IRQ7}}$ black push-button switch
56	1	S2	C&K KS11R23CQD	Hard reset push-button switch
57	9	TP1,TP2,TP3,TP4,TP5,TP6, TP7,TP8,TP9	Keystone 5015	Test points
58	1	T1	PE69012	Ethernet isolation transformer
59	1	U1	SN65HVD230D	CAN Transceiver
60	1	U2	MCF5282CVF80	MCF5282 ColdFire
61	1	U3	Pletronics Osc. 25MHz	25MHz oscillator
62	1	U4	Am79C874VC	Ethernet Phy
63	1	U5	Am29LV160DB-90EC	AMD 2MB Flash
64	1	U6	Micron MT58L128L36F1	Burst FSRAM
65	1	U7	MC74LCX541DT	Bus transceiver
66	1	U8	LM2596S-3.3	National Semi DCtoDC switcher
67	1	U9	LM2596S-5	National Semi DCtoDC switcher
68	2	U10,U12	ADM708SAR	Voltage sensor
69	1	U11	Epson SG-8002DC-8.0000M-PC	8MHz oscillator
70	1	Y1	FOXS080	8MHz crystal
71	2	U13,U14	MT48LC4M16A2TG (TSOP II 400 mil)	SDRAM
72	2	U15,U16	MAX3225CAP	RS232 Transceivers

Appendix D
Jumper Settings

Table D-1. M5282EVB Jumper Settings

Jumper Setting		Function
JP1	ON/OFF*	CAN Transceiver Mode
JP2	ON/OFF*	CAN Termination
JP3	*ON/OFF	Flash Voltage Reference
JP4	*ON/OFF	Standby Voltage Supply
JP5	*ON/OFF	Voltage Reference HIgh
JP6	*1-2 & 3-4	CS0 to Flash and CS1 to FSRAM
	1-3 & 2-4	CS1 to Flash and CS0 to FSRAM
JP7	*ON/OFF	Voltage Reference Low
JP8	ON/OFF*	Ethernet Auto Negotiate Enabled
JP9	ON/OFF*	Ethernet Tech 0 - 10 & 100 BaseT operation
JP10	ON/OFF*	Ethernet Tech 1 - Full and Half Duplex Operation
JP11	ON/OFF*	Ethernet Tech 2 - As above for JP9 and JP10
JP12	*ON/OFF	DTOUT0 LED Enable/Disable
JP13	*ON/OFF	DTOUT1 LED Enable/Disable
JP14	*ON/OFF	DTOUT2 LED Enable/Disable
JP15	*ON/OFF	DTOUT3 LED Enable/Disable
JP16	*1-2/2-3	Flash Boot dBug / System
JP17	*1-2/2-3	BDM / JTAG selection for Pin 7 of BDM Header
JP18	*1-2/2-3	VRL Selector VSSA / J5-2
JP19	*1-2/2-3	VRH Selector VDDA / J5-20
JP20	*ON/OFF	3.3VP Jumper
JP21	*ON/OFF	3.3V Jumper
JP22	*ON/OFF	5V jumper
JP23	*1-2/2-3	VSSA Selector GND / J5-1
JP24	1-2	VDDA +3.3V
	*3-4	VDDA +5V
	5-6	VDDA +J5-18

Table D-1. M5282EVB Jumper Settings (continued)

Jumper Setting		Function
JP25	*ON/OFF	Oscillator Disable / Enable
JP26	1-2/2-3*	Clock Source Selector - Osc. (external) / Crystal
JP27	ON/OFF*	Crystal Enable / Disable
JP28	*ON/OFF	Terminal Port GPTB0
JP29	*ON/OFF	Terminal Port GPTB1
JP30	ON/OFF*	Terminal Port DTIN3
JP31	ON/OFF*	Terminal Port DTIN2
JP32	*ON/OFF	Auxiliary Port GPTB2
JP33	*ON/OFF	Auxiliary Port GPTB3
JP34	ON/OFF*	Auxiliary Port DTOUT3
JP35	ON/OFF*	Auxiliary Port DTOUT2
JP36	*ON/OFF	I2C SCL
JP37	*ON/OFF	I2C SDA

Appendix E
Using the M5282EVB to Evaluate Subset Devices

The M5282EVB now supports the evaluation of the MCF5280, MCF5281, MCF5214, and MCF5216 microcontrollers in addition to the MCF5282. The evaluation board comes fitted with 80MHz MCF5282 microcontroller (512 Kbyte internal flash). The MCF5282 microcontroller has a superset of the same functional modules and interfaces as those on the other devices supported.

This appendix briefly points out the differences between the MCF5282 and the subset devices that the users should take into consideration when using the M5282EVB for evaluating these devices. Please see the specific microprocessor user's manual for more information regarding pinouts, package information, signal and functional descriptions.

E.1 Considerations for the MCF5281

The MCF5281 has only 256Kbytes of internal flash. The user should ignore the upper half of internal flash when using the M5282EVB to evaluate the MCF5281.

- Ignore the top half of the internal flash. This is from FLASHBAR + 0x40000 to FLASHBAR + 0x80000

E.2 Considerations for the MCF5280

The MCF5280 has no internal flash. The user should ignore all of the internal flash when using the M5282EVB to evaluate the MCF5280.

- Ignore all of internal Flash. This is from FLASHBAR + 0x00000 to FLASHBAR + 0x80000
- You cannot boot from Flash. The configuration setting for booting form internal flash on the MCF5280 is invalid

E.3 Considerations for the MCF5216

The MCF5216 does not have a Ethernet module. When using the M5282EVB to evaluate this device, the user should not use the Ethernet Module. Functional Ethernet pins should be used as GPIO with the exception of Port PEH[7:0] which is not available.

- Ignore the Ethernet signals
- Ignore GPIO port PEH[7:0]
- Pinout changes

E.4 Considerations for the MCF5214

The MCF5214 does not have a Ethernet module and only 256Kbytes of internal flash. When using the M5282EVB to evaluate this device, the user should not use the Ethernet Module. Functional Ethernet pins should be used as GPIO with the exception of Port PEH[7:0] which is not available. The user should also ignore the top have of the internal flash.

- Ignore the top half of the internal flash. This is from FLASHBAR + 0x40000 to FLASHBAR + 0x80000
- Ignore the Ethernet signals
- Ignore GPIO port PEH[7:0]
- Pinout changes

Appendix F
Revision History

This appendix lists major changes between versions of the M5282EVBUM document.

F.1 Changes Between Rev. 1.3 and Rev. 2

Table F-1. Rev. 1.3 to Rev. 2 Changes

Chapter	Description
Introduction	In “SW1-[4:3] Encoded Clock Mode” table, removed last row and the RCON column, since the CLKMOD pins are always sampled no matter the RCON value.