MCF5275EVB User's Manual

Devices Supported:
MCF5275
MCF5275L
MCF5274
MCF5274L

MCF5275EVBUM
Rev. 1.1
11/2008

How to Reach Us:

USA/Europe/Locations Not Listed:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
852-26668334

Home Page:
www.freescale.com

Learn More: For more information about Freescale products, please visit www.freescale.com.

MCF5275EVBUM
Rev. 1.0
10/2004

EMC Information on M5275EVB

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

WARNING

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

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Paragraph Number	Title	Page Number

Chapter 1
M5275EVB Introduction

This document details the setup and configuration of the ColdFire M5275EVB evaluation board (hereafter referred to as the EVB). The EVB is intended to provide a mechanism for easy customer evaluation of the MCF5274, MCF574L, MCF5275, and MCF5275L ColdFire microprocessors and to facilitate hardware and software development. The EVB 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 MCF5274(L) and MCF5275(L) are supported.

The heart of the evaluation board is the MCF5275. The MCF5275L and MCF5274(L) have a subset of the MCF5275 specification and can therefore be fully emulated using the MCF5275 device. Table 1-1., “MCF5274/75 Product Family”. below details the two devices.

Table 1-1. MCF5274/75 Product Family

Part Number	Package	FEC	CRYPTO	Max Core/Bus speed
MCF5274LVM133	196 MAPBGA	x1	No	133 / 66 MHz
MCF5274LVM166	196 MAPBGA	x1	No	166 / 83 MHz
MCF5274VM133	256 MAPBGA	x2	No	133 / 66 MHz
MCF5274VM166	256 MAPBGA	x2	No	166 / 83 MHz
MCF5275LCVM133	196 MAPBGA	x1	Yes	133 / 66 MHz
MCF5275LCVM166	196 MAPBGA	x1	Yes	166 / 83 MHz
MCF5275CVM133	256 MAPBGA	x2	Yes	133 / 66 MHz
MCF5275CVM166	256 MAPBGA	x2	Yes	166 / 83 MHz

All of the devices in the same package are pin compatible.

The EVB 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. No additional hardware or software is required for basic operation.

Specifications

- Motorola MCF5275 Microprocessor (166 MHz max core frequency)
- External Clock source: 25MHz
- Operating temperature: 0°C to +70°C
- Power requirement: 6 – 14V DC @ 1A Typical
- Power output: 5V, 3.3V, 2.5V and 1.5V regulated supplies
- Board Size: 10.00 x 5.40 inches, 8 layers

Memory Devices:

- 16-Mbyte DDR SDRAM
- 2-Mbyte (512K x 16) Page Mode Flash or 4-Mbyte (512K x 32) Page mode Flash
- 1-Mbyte ASRAM (footprint only)
- 64-Kbyte SRAM internal to MCF5275 device

Peripherals:

- FEC0 Ethernet port 10/100Mb/s (Dual-Speed Fast Ethernet Transceiver, with MII)
- FEC1 Ethernet port 10/100Mb/s (Dual-Speed Fast Ethernet Transceiver, with MII)
- USB 2.0 Full Speed (Device)
- UART0 (RS-232 serial port for dBUG firmware)
- UART1 (auxiliary RS-232 serial port)
- UART2 (auxiliary RS-232 serial port)
- I^{2}C interface
- QSPI interface
- BDM/JTAG interface

User Interface:

- Reset logic switch (debounced)
- Boot logic selectable (dip switch)
- Abort/IRQ7 logic switch (debounced)
- PLL 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

Software:

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

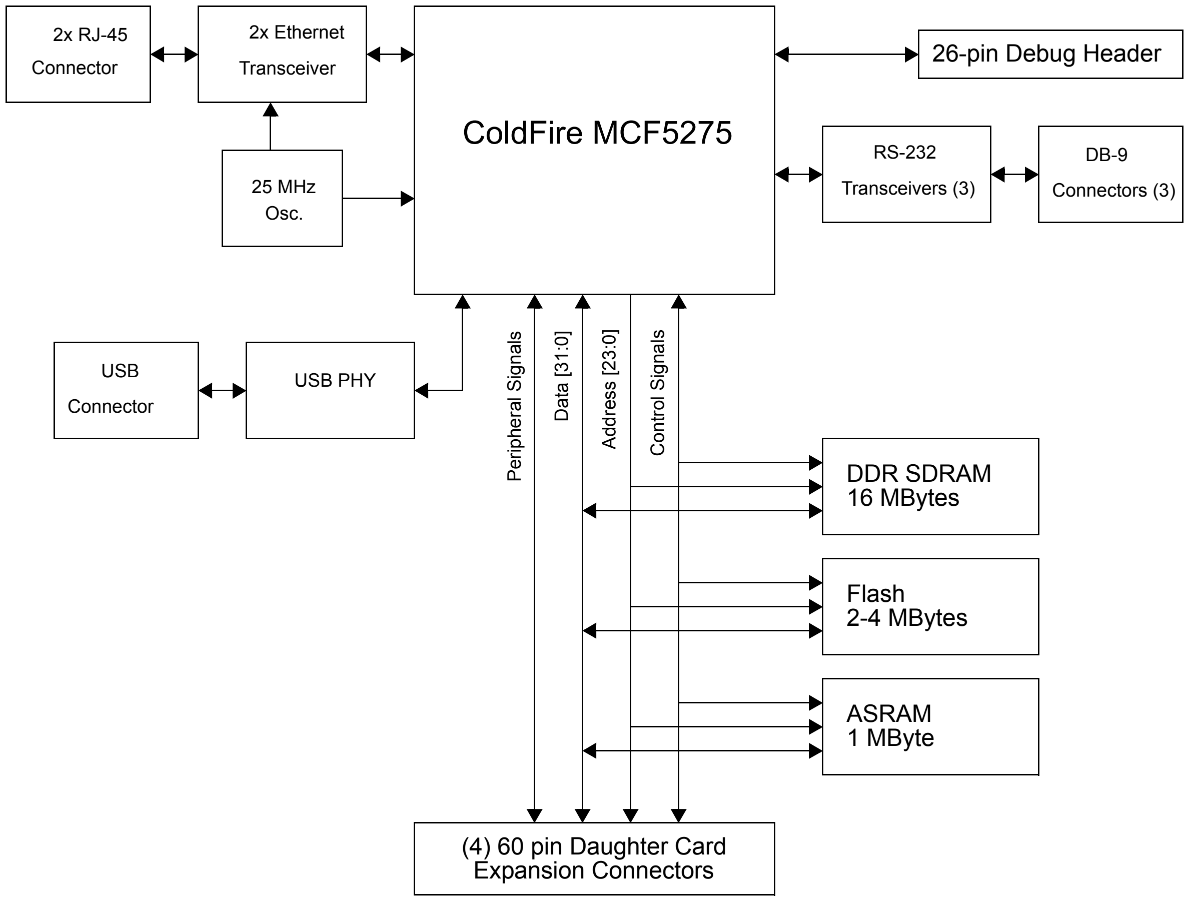


Figure 1-1. M5275EVB block diagram

1.1 MCF5275 Microprocessor

The microprocessor used on the EVB is the highly integrated Motorola MCF5275 32-bit ColdFire variable-length RISC processor. The MCF5275 implements a ColdFire Version 2 core with a maximum core frequency of 166MHz and external bus speed of 83MHz. Features of the MCF5275 include:

- V2 ColdFire core with enhanced multiply-accumulate unit (EMAC) providing 159 Dhrystone 2.1MIPS @ 166MHz
- 64 KBytes of internal SRAM
- External bus speed of one half the CPU operating frequency (83MHz bus @ 166MHz core)
- Two 10/100 BaseT Fast Ethernet Controllers (FECs)
- 16 Kbytes of configurable instruction/data cache

- Three universal asynchronous receiver/transmitters (UARTs) with DMA support
- Inter-integrated circuit (I^2C) bus controller
- Queued serial peripheral interface (QSPI) module
- Hardware cryptography accelerator (optional)
  - Random number generator
  - DES/3DES/AES block cipher engine
  - MD5/SHA-1/HMAC accelerator
- Four channel 32-bit direct memory access (DMA) controller
- Four channel 32-bit input capture/output compare timers with optional DMA support
- Four channel 16-bit periodic interrupt timers (PITs)
- Programmable software watchdog timer
- Interrupt controller capable of handling up to 126 interrupt sources
- Clock module with Phase Locked Loop (PLL)
- External bus interface module including a DDR SDRAM controller
- 32-bit non-multiplexed bus with up to 8 chip select signals that support page-mode Flash memories

The MCF5275 communicates with external devices over a 16-bit wide data bus, D[31:16]. The MCF5275 can address a 32 bit address range. However, only 24 bits are available on the external bus A[23:0]. 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 via daughter card expansion connectors. Refer to the schematic (Appendix B) for their pin assignments.

The MCF5275 processor has the capability to support both BDM and JTAG. These ports are multiplexed and together. In BDM mode it can be used with third party tools to allow the user to download code to the board. In JTAG mode it can be used for boundary scan operations. 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 MCF5275 processor block diagram.

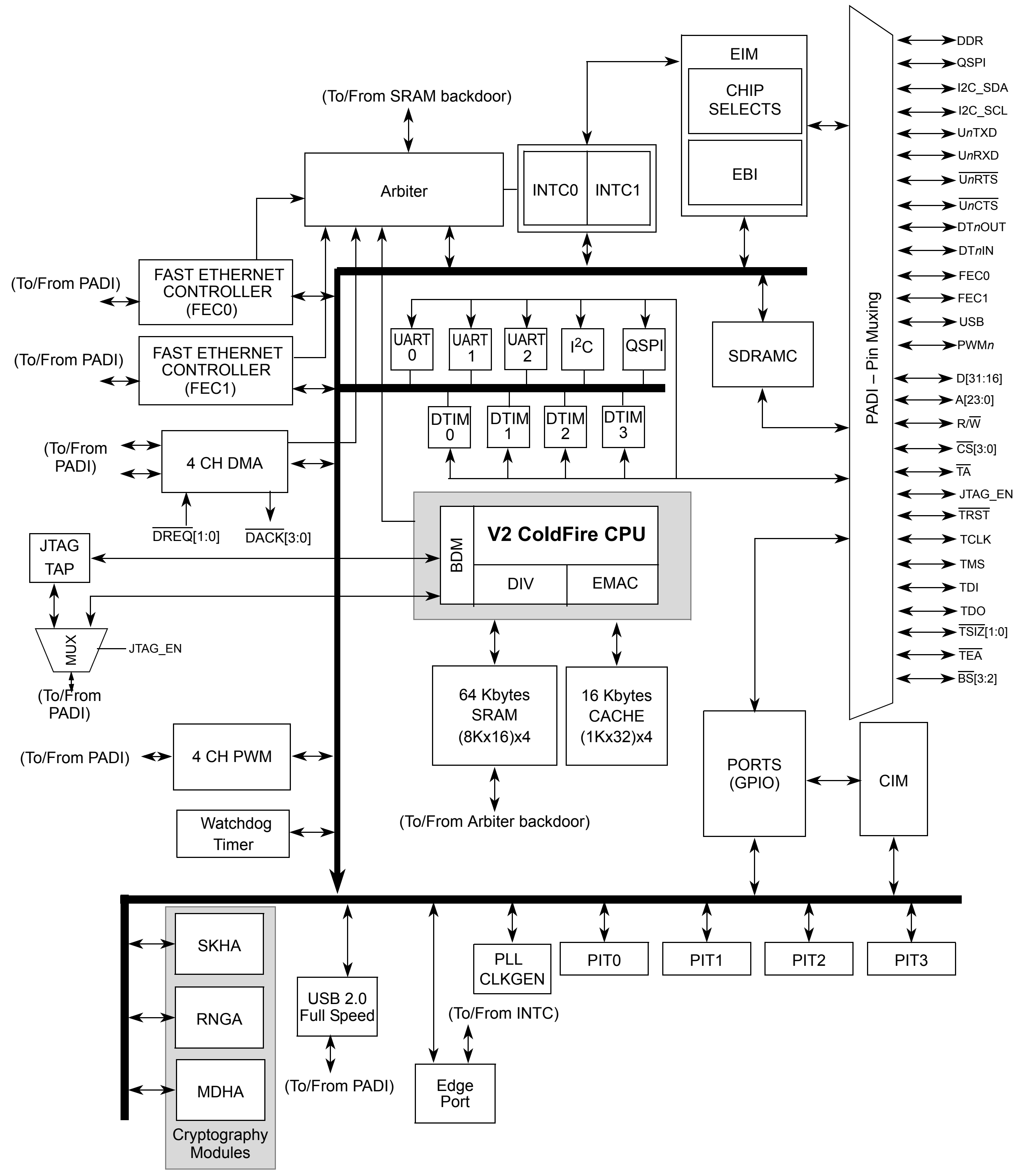


Figure 1-2. MCF5275 Block Diagram

1.2 System Memory

The following diagram shows the external memory implementation on the M5275EVB.

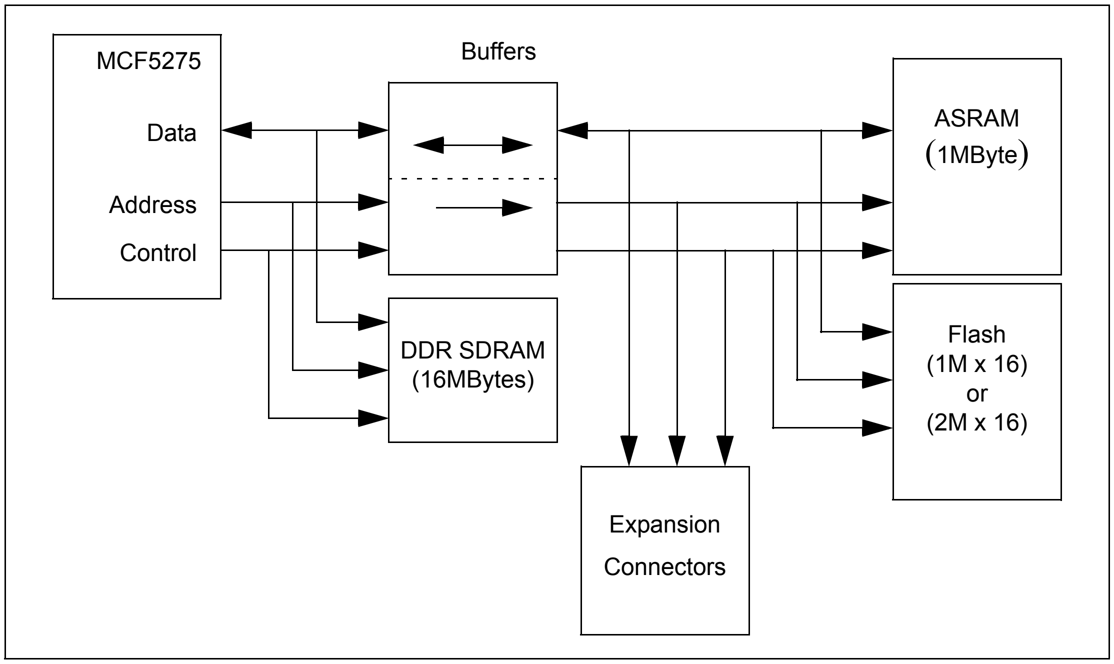


Figure 1-3. External Memory Scheme

NOTE

The external bus interface signals to the external ASRAM and Flash are buffered. This is in order to separate the 2.5V bus to the DDR from the 3.3V bus to the Flash and ASRAM.

The external bus signals to the expansion connectors are buffered to 3.3V to provide a 3.3V interface to the user.

1.2.1 External Flash

The EVB is fitted with a single 1M x 16 page-mode Flash memory (U11) giving a total memory space of 2 MBytes. Alternatively a footprint is available for the EVB user to upgrade this device to a 2M x 16 page-mode Flash memory (U12), doubling the memory size to 4 MBytes. Either U11 OR U12 should be fitted on the board - both devices cannot be populated at the same time. Refer to the specific device data sheet and sample software provided for information on configuring the Flash memory.

Users should note that the debug monitor firmware is installed on 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 reprogrammed into the flash by using a development tool through the BDM port. Users should use caution to avoid this situation. The M5275EVB dBUG debugger/monitor firmware is programmed into the lower sectors of Flash (0xFFE0_0000 to 0xFFE3_FFFF for 2 MBytes of Flash or 0xFFC0_0000 to 0xFFC3_FFFF for 4 MBytes of Flash).

When U11 is fitted on the EVB, jumper 3 (JP3) provides an alternative hardware mechanism for write protection. This feature is not available when U10 is populated.

1.2.2 SDRAM

The EVB is populated with 16 MBytes of SDRAM. This is done with a single device (Micron MT46V16M16TG) with a 16 bit data bus. The device (U7) is organized as 2Meg x 16 x 4 banks with a 16 bit data bus.

1.2.3 ASRAM

The EVB has a footprint for one 256K x 16 Asynchronous SRAM devices (Cypress Semiconductor - CY7C1041CV3310ZC). These memory devices (U1) may be populated by the user for benchmarking purposes.

1.2.4 Internal SRAM

The MCF5275 processor has 64 KBytes of internal SRAM memory which may be used as data or instruction memory. This memory is mapped to 0x2000_0000 by the dBUG monitor, 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 64 KByte boundary within the processor’s four gigabyte address space.

1.2.5 M5275EVB Memory Map

Signals to support the interface to external memory and peripheral devices are generated by the memory controller. The MCF5275 supports 8 external chip selects, $\overline{\text{CS}}$[1:0] and $\overline{\text{SD\_CS0}}$ are used with external memories on the EVB. $\overline{\text{CS0}}$ also functions as the global (boot) chip-select for booting out of external flash. All chip selects ($\overline{\text{CS}}$[7:0] and $\overline{\text{SD\_CS}}$[1:0]) are easily accessible to users via the daughter card expansion connectors.

Since the MCF5275 chip selects are fully programmable, the memory banks may be located at any 64-KByte boundary within the processor’s four gigabyte address space.

Table 1-2 shows the default memory map for this board as configured by the dBUG monitor located in the external Flash bank. The internal memory space of the MCF5275 is detailed further in the MCF5275 Reference Manual. Chip Selects 0 and 1 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.

Table 1-2 shows the M5275EVB memory map.

Table 1-2. The M5275EVB Default Memory Map

Address Range	Chip Select	Signal and Device
0x0000_0000 - 0x00FF_FFFF	$\overline{\text{SD\_CS0}}$	16 MByte SDRAM (U7)
0x2000_0000 - 0x2000_FFFF	—	64 KBytes Internal SRAM
0x3000_0000 - 0x300F_FFFF	$\overline{\text{CS1}}$	512 KByte External ASRAM (not fitted) (U1)
0xFFE0_0000 - 0xFFFF_FFFF or 0xFFC0_0000 - 0xFFFF_FFFF	$\overline{\text{CS0}}$	2 MBytes External Flash (U11) or 4 MBytes External Flash (U12)

1.2.5.1 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 port size of the memory that the MCF5275 accesses at address 0x0000_0000 is determined at reset by sampling D[20:19].

Table 1-3. D[20:19] External Boot Chip Select Configuration

D[20:19]	Boot Device/Data Port Size
00	Reserved
01	External (16-bit)
10	External (8-bit)
11	Reserved

1.3 Support Logic

1.3.1 Reset Logic

Reset occurs during power-on or via assertion of the signal $\overline{\text{RESET}}$ which causes the MCF5275 to reset. $\overline{\text{RESET}}$ is triggered by the reset switch (SW5) which resets the entire processor/system.

The dBUG Firmware configures the MCF5275 microprocessor internal resources during initialization. The contents of the exception vector 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 1-2.

If the external $\overline{\text{RCON}}$ pin is asserted (SW6-1 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 tables below on settings for reset configurations.

If the $\overline{\text{RCON}}$ pin is negated (SW6-1 OFF) during reset, the chip configuration and the reset configuration pin functions after reset are determined by the RCON register or fixed defaults, regardless of the states of the external data pins.

Table 1-4. SW6-1 $\overline{\text{RCON}}$

SW6-1	Reset Configuration
OFF	$\overline{\text{RCON}}$ not asserted, Default Chip configuration or RCON register settings
ON	$\overline{\text{RCON}}$ is asserted, Chip functions, including the reset configuration after reset, are configured according to the levels driven onto the external data pins.

Table 1-5. SW6-2 JTAG_EN

SW6-2	JTAG Enable
OFF	JTAG interface enabled
ON	BDM interface enabled

Table 1-6. SW6-[4:3] Encoded Clock Mode

SW6-3	SW6-4	Clock Mode
ON	ON	Normal PLL mode with external clock reference
ON	OFF	Normal PLL mode with external clock reference
OFF	ON	External clock mode- (PLL disabled)
OFF	OFF	1:1 PLL mode

Table 1-7. SW6-[7:5] Chip Mode

SW6-5	SW6-6	SW6-7	$\overline{\text{RCON}}$ (SW6-1)	Mode
ON	ON	ON	ON	Master mode
ON	ON	OFF	ON	Reserved
ON	OFF	ON	ON	Reserved
ON	OFF	OFF	ON	Reserved
OFF	X	X	ON	Reserved
X	X	X	OFF	Master mode

Table 1-8. SW6-[9:8] Boot Device

SW6-8	SW6-9	$\overline{\text{RCON}}$ (SW6-1)	Boot Device
ON	ON	ON	External (16-bit)
ON	OFF	ON	Reserved
OFF	ON	ON	Reserved
OFF	OFF	ON	External (8-bit)
X	X	OFF	External (16-bit)

Table 1-9. SW6-10 Bus Drive Strength

SW6-10	RCON (SW6-1)	Drive Strength
ON	ON	Full Bus Drive
OFF	ON	Partial Bus Drive
X	OFF	Full Bus Drive

Table 1-10. SW6-[12:11] Address/Chip Select Mode

SW6-11	SW6-12	RCON (SW6-1)	Mode
ON	ON	ON	PADDR[7:5] = A[23:21]
ON	OFF	ON	PADDR7 = $\overline{\text{CS6}}$; PADDR[6:5] = A[22:21]
OFF	ON	ON	PADDR[7:6] = $\overline{\text{CS}}$[6:5]; PADDR5 = A21
OFF	OFF	ON	PADDR[7:5] = $\overline{\text{CS}}$[6:4]
X	X	OFF	PADDR[7:5] = A[23:21]

1.3.2 Clock Circuitry

The are three options to provide the clock to the CPU. Table 1-11 shows how these options can be configured by setting JP9 and JP10.

Table 1-11. M5275EVB Clock Source Selection

JP9	JP10	Clock Selection
1-2	1-2	25MHz Oscillator (default setting)
2-3	1-2	25MHz External Clock
X	2-3	25MHz Crystal

The 25MHz oscillator (U19) also feeds the Ethernet transceiver chips (U8 and U9).

1.3.3 Watchdog Timer

The dBUG Firmware does **NOT** enable the watchdog timer on the MCF5275.

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. The MCF5275's interrupt controller will respond to the interrupt acknowledge cycle with the vector number for the interrupt (refer to the MCF5275 Reference Manual for more information on the interrupt controller function).

The vector number is used as an index into the exception vector table that contains the interrupt service routine locations. This table is stored in the board's Flash memory. 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 MCF5275 microprocessor has seven external interrupt request lines $\overline{\text{IRQ}}$[7:1]. Each external interrupt can be configured individually as a level-sensitive interrupt pin or an edge-detecting interrupt pin (rising edge, falling edge, or both).

Two on-chip interrupt controllers are capable of providing unique vectors for all of the on-chip and external interrupt sources. Interrupt controller 0 (INTC0) handles the following interrupt sources:

- External interrupt signals $\overline{\text{IRQ}}$[7:1] (EPORT)
- Software watchdog timer module
- Four DMA channels
- UART modules (UART0–UART2)

- I^{2}C module
- QSPI module
- Timer modules
- Fast Ethernet Controller (FEC0)
- Periodic Interrupt Timers (PIT0–PIT3)
- Random Number Generator (RNG)
- Symmetric Key Hardware Accelerator (SKHA)
- Message Digest Hardware Accelerator (MDHA)
- USB module

Interrupt controller 1 (INTC1) handles these interrupt sources:

- Fast Ethernet Controller (FEC1)

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

The M5275EVB hardware uses $\overline{\text{IRQ7}}$ to support the ABORT function using the ABORT switch (SW4). This switch is used to force an interrupt (level 7, mid-point priority) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT).

Refer to MCF5275 Reference Manual for more information about the interrupt controller.

1.3.5 TA Generation

The processor starts a bus cycle by asserting $\overline{\text{CS}n}$ with the other control signals. The processor then waits for a transfer acknowledgment ($\overline{\text{TA}}$) either internally (using the chip select's auto acknowledge - AA mode) or externally before it can complete the bus cycle. $\overline{\text{TA}}$ is used to indicate the completion of the bus cycle. It also allows devices with different access times to communicate with the processor properly asynchronously. The MCF5275 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 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 M5275EVB, the $\overline{\text{TA}}$ input of the processor is also connected to the expansion connectors (U9). 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 $\overline{\text{TA}}$ signals from expansion boards should be connected to this line.

1.3.6 User's Program

JP4 on the 2 MByte Flash (U11) or JP5 if using the 4 MByte Flash (U12) allows users to test code from boot/POR without having to overwrite the dBUG 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 (0xFFC0_0000). When the jumper is set between pins 2 and 3, the board boots from the top half of the Flash 0xFFF0_0000 (0xFFE0_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 JP4 (JP5 for U12) 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 (0xFFC0_0400) and run.
4. In the ROM Monitor, execute the 'FL write <dest> <src> <bytes>' command.
5. Move jumper JP4 (JP5 for U12) to pin 2 connected to pin 3 and push the reset button (SW5). User code should now be running from reset/POR.

1.4 Communication Ports

The EVB provides external communication interfaces for three UART serial ports, QSPI, I^{2}C port, two 10/100T ethernet ports, and BDM/JTAG port.

1.4.1 UART0, UART1, UART2 Ports

The MCF5275 device has three built in UARTs, each with its own software programmable baud rate generator. These UART interfaces are brought out to RS-232 transceivers. One channel is the ROM Monitor to Terminal output and the other two are available to the user.

Refer to the MCF5275 Reference Manual for programming the UARTs and their register maps.

1.4.2 10/100T Ethernet Ports

The MCF5275 device performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The MCF5275 Ethernet Controller requires an external interface adaptor and transceiver function to complete the interface to the ethernet media. The MCF5275 Ethernet module also features an integrated fast (100baseT) Ethernet media access controller (MAC).

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

- Support for three different Ethernet physical interfaces:
  - 100-Mbps IEEE 802.3 MII

  - 10-Mbps IEEE 802.3 MII
  - 10-Mbps 7-wire interface (industry standard)
- IEEE 802.3 full duplex flow control
- Programmable max frame length supports IEEE 802.1 VLAN tags and priority
- Support for full-duplex operation (200Mbps throughput) with a minimum system clock rate of 50MHz
- Support for half-duplex operation (100Mbps throughput) with a minimum system clock rate of 25 MHz
- Retransmission from transmit FIFO following a collision (no processor bus utilization)
- Automatic internal flushing of the receive FIFO for runts (collision fragments) and address recognition rejects (no processor bus utilization)
- Address recognition
  - Frames with broadcast address may be always accepted or always rejected
  - Exact match for single 48-bit individual (unicast) address
  - Hash (64-bit hash) check of individual (unicast) addresses
  - Hash (64-bit hash) check of group (multicast) addresses
  - Promiscuous mode

For more details see the MCF5275 Reference manual. 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.

1.4.3 BDM/JTAG Port

The MCF5275 processor has a Background Debug Mode (BDM) port, which supports Real-Time Trace and Real-Time Debug. The signals which are necessary for debug are available at connector (J1). Figure 1-4 shows the (J1) Connector pin assignment.

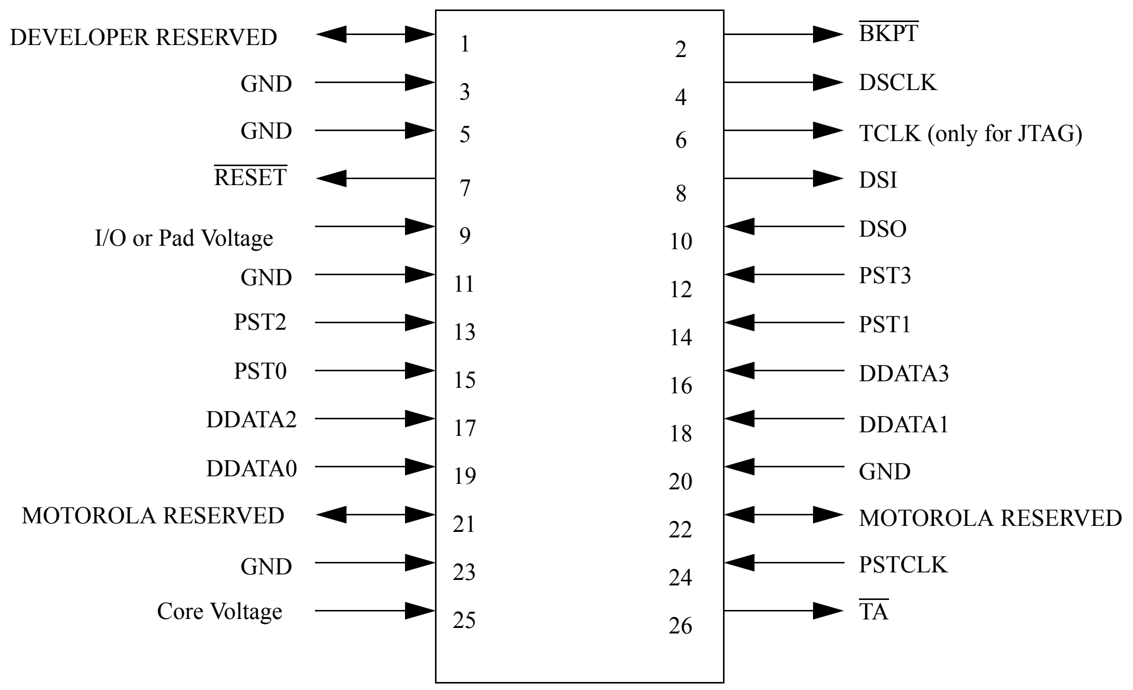


Figure 1-4. J1- BDM Connector pin assignment

The BDM connector can also be used to interface to JTAG signals. On reset, the JTAG_EN signal selects between multiplexed debug module and JTAG signals. See Table 1-5.

1.4.4 I^{2}C

The MCF5275's I^{2}C module includes the following features:

- Compatibility with the I^{2}C bus standard version 2.1
- Multi master operation
- Software programmable for one of 50 different clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte by byte data transfer
- Arbitration-lost interrupt with automatic 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 MCF5275 Reference Manual for more detail. The I^{2}C signals from the MCF5275 device are brought out to a connector (J10).

1.4.5 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 147.1-Kbps to 18.75-Mbps at 75MHz
- Programmable delays before and after transfers
- Programmable QSPI clock phase and polarity
- Supports wrap-around mode for continuous transfers

Please see the MCF5275 Reference Manual for more detail. The QSPI signals from the MCF5275 device are brought out to a header (J9).

1.5 Connectors and User Components

1.5.1 Daughter Card Expansion Connectors

Four, 60-way SMT connectors (J3, J4, J5 and J6) provide access to all MCF5275 signals. These connectors are ideal for interfacing to a custom daughter card or for simple probing of processor signals. Below is a pinout description of these connectors.

Table 1-12. J3 Pinout

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+2.5V	4	+3.3V
5	GND	6	GND
7	FEC1_RXD0	8	FEC1_RXD2
9	FEC1_RXD1	10	FEC1_RXDV
11	FEC0_RXER	12	FEC0_TXEN
13	FEC1_RXCLK	14	GND
15	FEC1_CRS	16	FEC0_RXD3

Table 1-12. J3 Pinout (Continued)

Pin	Signal	Pin	Signal
17	FEC0_RXDV	18	FEC0_RXD2
19	FEC1_COL	20	FEC0_RXD1
21	FEC0_RXCLK	22	FEC0_RXD0
23	FEC0_CRS	24	FEC0_COL
25	FEC0_MDC	26	GND
27	FEC0_MDIO	28	$\overline{\text{RTS0}}$
29	GND	30	$\overline{\text{CTS0}}$
31	$\overline{\text{RTS1}}$	32	TXD0
33	$\overline{\text{CTS1}}$	34	RXD0
35	TXD1	36	GND
37	RXD1	38	$\overline{\text{CS7}}$
39	GND	40	$\overline{\text{CS6}}$
41	SCL	42	$\overline{\text{CS5}}$
43	SDA	44	$\overline{\text{CS4}}$
45	GND	46	GND
47	B_A12	48	B_A16
49	B_A13	50	B_A19
51	B_A14	52	B_A20
53	B_A15	54	B_A21
55	B_A16	56	B_A22
57	B_A17	58	B_A23
59	GND	60	GND

Table 1-13. J4 Pinout

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+2.5V	4	+3.3V
5	FEC1_RXD3	6	FEC1_TXCLK
7	FEC1_RXER	8	FEC0_TXCLK
9	FEC1_TXER	10	FEC1_TXEN
11	FEC0_TXER	12	GND

Table 1-13. J4 Pinout (Continued)

Pin	Signal	Pin	Signal
13	GND	14	FEC1_TXD3
15	FEC1_TXD2	16	FEC1_TXD0
17	FEC1_TXD1	18	FEC0_TXD3
19	FEC0_TXD2	20	FEC0_TXD1
21	FEC0_TXD0	22	FEC1_MDIO
23	FEC1_MDC	24	GND
25	GND	26	TOUT0
27	TOUT1	28	TOUT2
29	TOUT3	30	GND
31	GND	32	TIN0
33	TIN1	34	TIN2
35	TIN3	36	$\overline{\text{OE}}$
37	$\overline{\text{SDWE}}$	38	$\overline{\text{SD\_CAS}}$
39	$\overline{\text{SD\_RAS}}$	40	SD_CKE
41	$\overline{\text{TS}}$	42	SD_DQS1
43	$\overline{\text{BS3}}$	44	B_D31
45	$\overline{\text{BS2}}$	46	B_D30
47	SD_CS1	48	B_D29
49	$\overline{\text{SD\_CS0}}$	50	GND
51	B_D27	52	B_D28
53	B_D25	54	B_D26
55	B_D23	56	B_D24
57	SD_VREF	58	SD_VREF
59	GND	60	GND

Table 1-14. J5 Pinout

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+2.5V	4	+3.3V
5	GND	6	GND
7	B_A11	8	B_A5
9	B_A10	10	B_A4

Table 1-14. J5 Pinout (Continued)

Pin	Signal	Pin	Signal
11	B_A9	12	B_A3
13	B_A8	14	B_A2
15	B_A7	16	B_A1
17	B_A6	18	B_A0
19	GND	20	GND
21	$\overline{\text{IRQ7}}$	22	$\overline{\text{CS3}}$
23	$\overline{\text{IRQ6}}$	24	$\overline{\text{CS2}}$
25	$\overline{\text{IRQ5}}$	26	TSIZ1
27	$\overline{\text{IRQ4}}$	28	TSIZ0
29	$\overline{\text{IRQ3}}$	30	GND
31	$\overline{\text{IRQ2}}$	32	USB_SPEED
33	$\overline{\text{IRQ1}}$	34	USB_CLK
35	GND	36	GND
37	USB_TN	38	USB_RN
39	USB_TP	40	USB_RP
41	GND	42	GND
43	$\overline{\text{TA}}$	44	USB_TXEN
45	GND	46	USB_RXD
47	EXTAL	48	USB_SUSP
49	XTAL	50	GND
51	$\overline{\text{RSTOUT}}$	52	$\overline{\text{EXT\_RST}}$
53	TRST/DSCLK	54	$\overline{\text{RESET}}$
55	TCLK/PSTCLK	56	TDO/DSO
57	TDI/DSI	58	TMS/$\overline{\text{BKPT}}$
59	GND	60	GND

Table 1-15. J6 Pinout

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+2.5V	4	+3.3V
5	B_D22	6	B_D18
7	B_D21	8	B_D17
9	B_D20	10	B_D16
11	B_D19	12	SD_DQS0
13	GND	14	GND

Table 1-15. J6 Pinout (Continued)

Pin	Signal	Pin	Signal
15	$\overline{\text{CS0}}$	16	$\overline{\text{SD\_CLKOUT}}$
17	SD_A10	18	$\overline{\text{TIP}}$
19	SD_CLKOUT	20	R/$\overline{\text{W}}$
21	$\overline{\text{CS1}}$	22	$\overline{\text{TEA}}$
23	$\overline{\text{RTS2}}$	24	$\overline{\text{RCON}}$
25	RXD2	26	TXD2
27	$\overline{\text{CTS2}}$	28	GND
29	GND	30	PST3
31	PST2	32	PST1
33	PST0	34	GND
35	GND	36	DDATA3
37	DDATA2	38	DDATA1
39	DDATA0	40	GND
41	GND	42	QSDO
43	CLKOUT	44	GND
45	GND	46	PCS0
47	PSC2	48	PSC3
49	PSC1	50	GND
51	GND	52	QSDI
53	CLKMOD0	54	CLKMOD1
55	JTAG_EN	56	SCK
57	GND	58	GND
59	GND	60	GND

1.5.2 Reset Switch (SW5)

The reset logic provides system initilization. Reset occurs during power-on or via assertion of the signal $\overline{\text{RESET}}$ which causes the MCF5275 to perform a hardware reset. Reset is also triggered by the reset switch (SW5) which resets the entire processor/system.

A hard reset and voltage sense controller (U20) is used to produce an active low power-on $\overline{\text{RESET}}$ signal. The reset switch SW5 is fed into U18 which generates the signal which is fed to the MCF5275 reset, $\overline{\text{RESET}}$. There are three sources of reset on the board:

1. Power sense and reset switch circuit (U20 and SW5)
2. BDM reset from J1
3. External reset for the expansion connector (J5)

An OR gate (U21) is used to OR all three of the boards reset sources. The output of the OR gate is connected directly to the MCF5275's $\overline{\text{RESET}}$ pin.

dBUG configures the MCF5275 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.

1.5.3 User LEDs

There are eight LEDs available to the user. Each of these LEDs are pulled to +3.3V through a 10 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 details which MCF5275 signal is associated with which LED.

Table 1-16. User LEDs

LED	MCF5275 Signal	Jumper to disconnect
D21	TOUT0	JP12
D22	TIN0	JP13
D23	TOUT1	JP14
D24	TIN1	JP15
D25	TOUT2	JP16
D26	TIN2	JP17
D27	TOUT3	JP18
D28	TIN3	JP19

1.5.4 Other LEDs

There are several other LED's on the M5275EVB to signal to the user various board/processor/component state. Below is a list of those LEDs and their functions:

Table 1-17. LED Functions

LED	Function
D1-D4	Ethernet PHY functionality
D8	+3.3V Power Good
D14	+5V Power Good
D18	+2.5V Power Good
D19	Abort ($\overline{\text{IRQ7}}$) asserted

Table 1-17. LED Functions (Continued)

LED	Function
D20	Reset ($\overline{\text{RESET}}$) asserted
D21-D28	User LEDs (See Table 1-16)

Chapter 2 Initialization and Setup

2.1 System Configuration

The M5275EVB board requires the following items for minimum system configuration:

- The M5275EVB board (provided).
- Power supply, +6V to 14V DC with minimum of 1 A (9V, 2.7A supply provided).
- RS232C compatible terminal or a PC with terminal emulation software.
- RS232 Communication cable (provided).

Figure 2-1 displays the minimum system configuration.

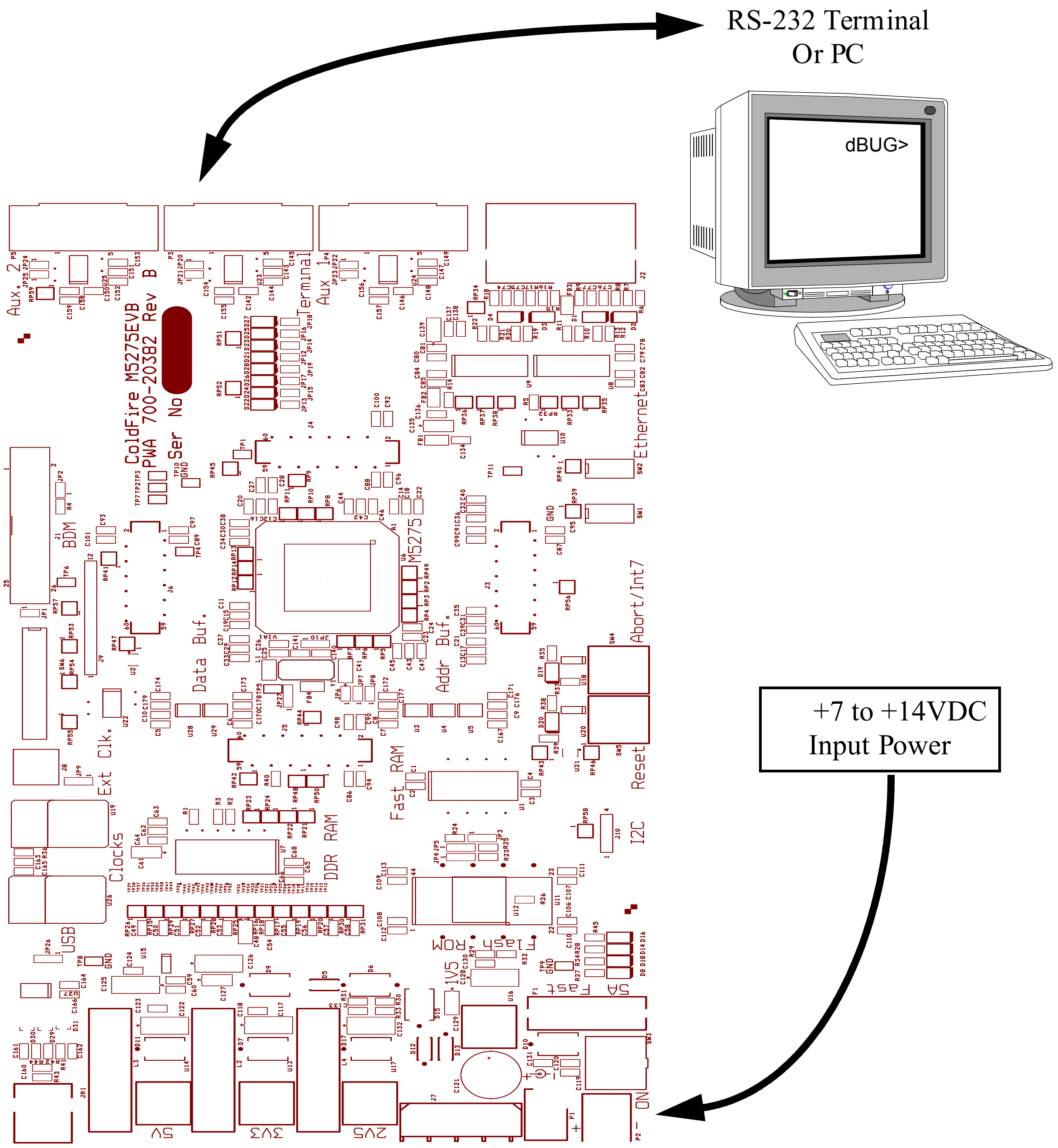


Figure 2-1. Minimum System Configuration

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, “Configuring dBUG for Network Downloads.”

2.2.1 Unpacking

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

- M5275EVB Single Board Computer
- M5275EVB User's Manual (this document)
- One RS-232 communication cable
- One DB25 parallel port cable
- One BDM (Background Debug Mode) “wiggler” cable
- One ethernet crossover cable
- EVB power supply kit
- MCF5275RM ColdFire Integrated Microprocessor Reference 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 refer to the enclosed warranty card for instructions.

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.

2.2.3 Providing Power to the Board

The EVB requires an external supply voltage of 6-14V DC, minimum 1 A. This is regulated on board using three switching voltage regulators to provide the necessary EVB voltages of 5V, 3.3V, 2.5V, and 1.5V. There are three different power supply input connectors on the EVB. Connector P1 is a 2.1mm power jack (Figure 2-2). P2 is a lever actuated connector (Figure 2-3). J7 is a PC disk drive power connector.

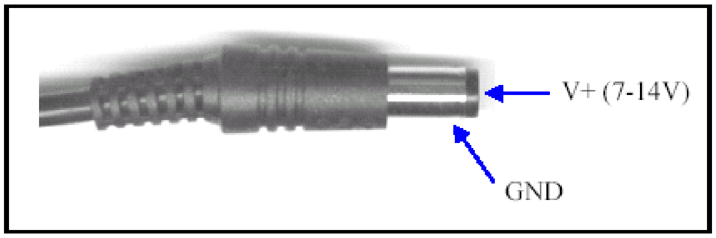


Figure 2-2. 2.1mm Power Connector

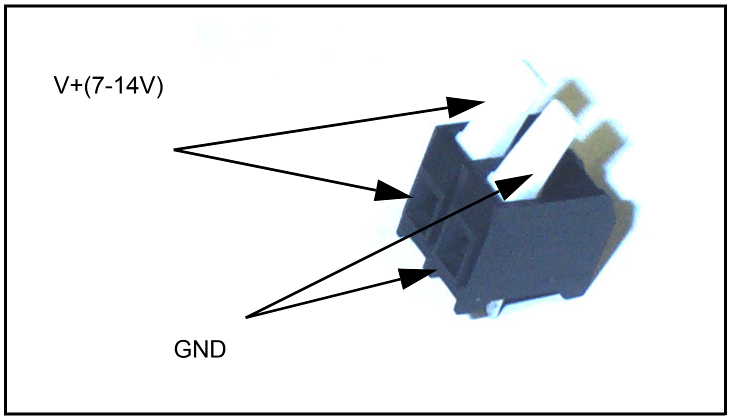


Figure 2-3. 2-Lever Power Connector

2.2.4 Power Switch (SW3)

Slide switch SW3 can be used to isolate the power supply input from the EVB voltage regulators if required.

Moving the slide switch to the left (towards connector P2) will turn the EVB ON.

Moving the slide switch to the right (away from connector P2) will turn the EVB OFF.

2.2.5 Power Status LEDs and Fuse

When power is applied to the EVB, green power LEDs adjacent to the voltage regulators show the presence of the supply voltage as follows:

Table 2-1. Power LEDs

LED	Function
D14	Indicates that the +5V regulator is working correctly
D8	Indicates that the +3.3V regulator is working correctly
D18	Indicates that the +2.5V regulator is working correctly

If no LEDs are illuminated when the power is applied to the EVB, it is possible that either power switch SW3 is in the “OFF” position or that the fuse F1 has blown. This can occur if power is applied to the EVB in reverse-bias where a protection diode ensures that the fuse blows rather than causing damage to the EVB. Replace F1 with a 20mm 5A fast blow fuse.

2.2.6 Selecting Terminal Baud Rate

The serial channel UART0 of the MCF5275 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.

2.2.7 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.

2.2.8 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 M5275EVB 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 M5275EVB 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 2-4.

2.2.9 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 2.2.8, “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.

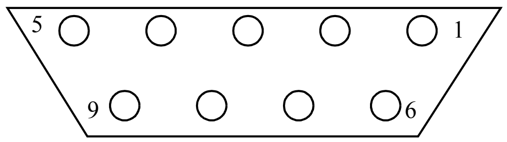


Figure 2-4. Pin Assignment for Female (Terminal) Connector

Pin assignments are as follows:

Table 2-2. 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 2-5 on the next page shows the jumper locations for the board.

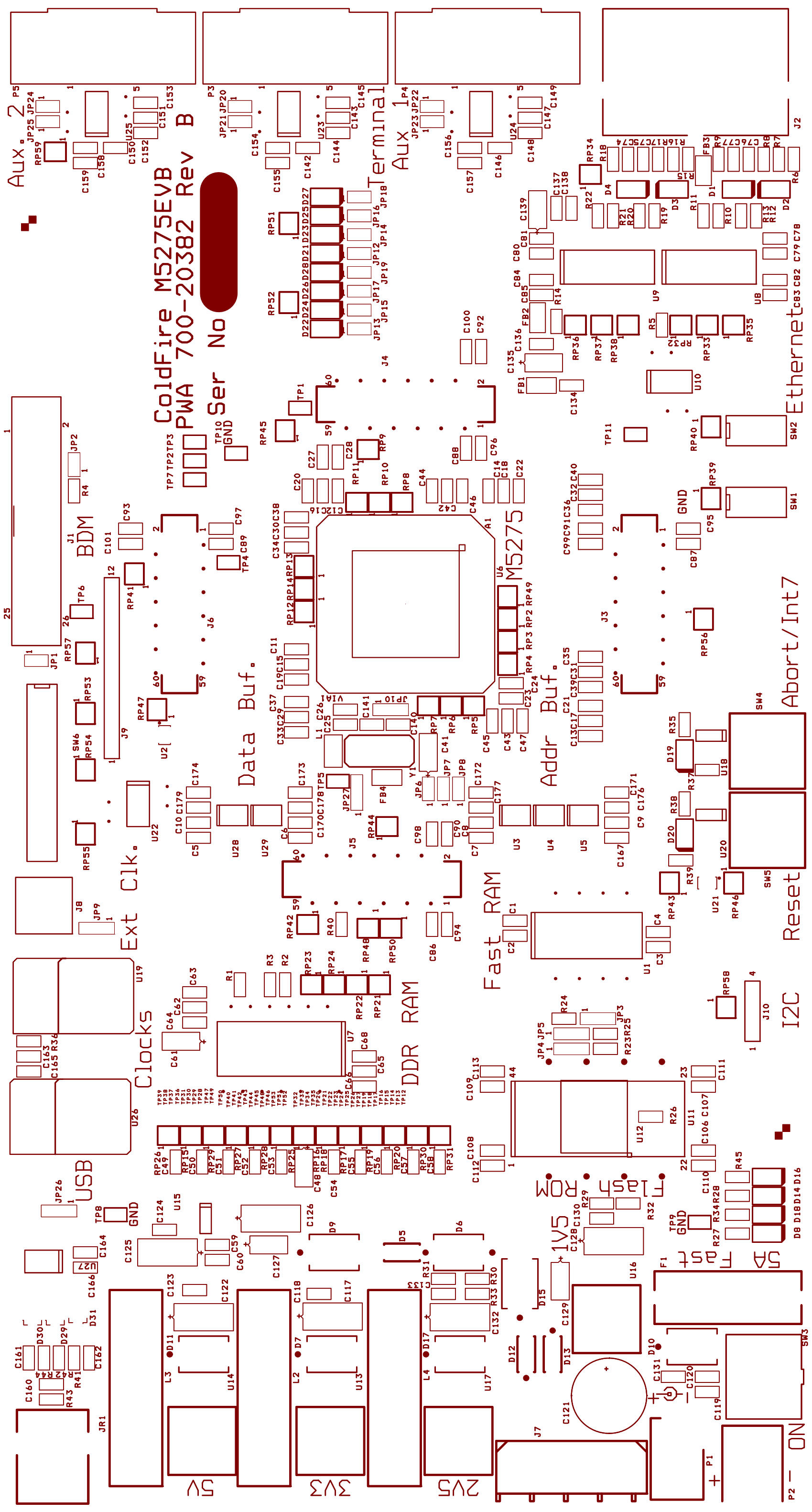


Figure 2-5. Jumper Locations

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 1995-2004 Motorola, Inc. All Rights Reserved.
ColdFire MCF5275 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 3, “Using the Monitor/Debug Firmware.” 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. Please refer to the enclosed warranty card for return instructions.

2.4 Using The BDM Port

The MCF5275 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 (J1) 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 the CD-ROM included in this kit.

Chapter 3
Using the Monitor/Debug Firmware

The M5275EVB 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 2, “Initialization and Setup.” 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 (P3) connector.
- Turn power on to the board.

Figure 3-1 shows the dBUG operational mode.

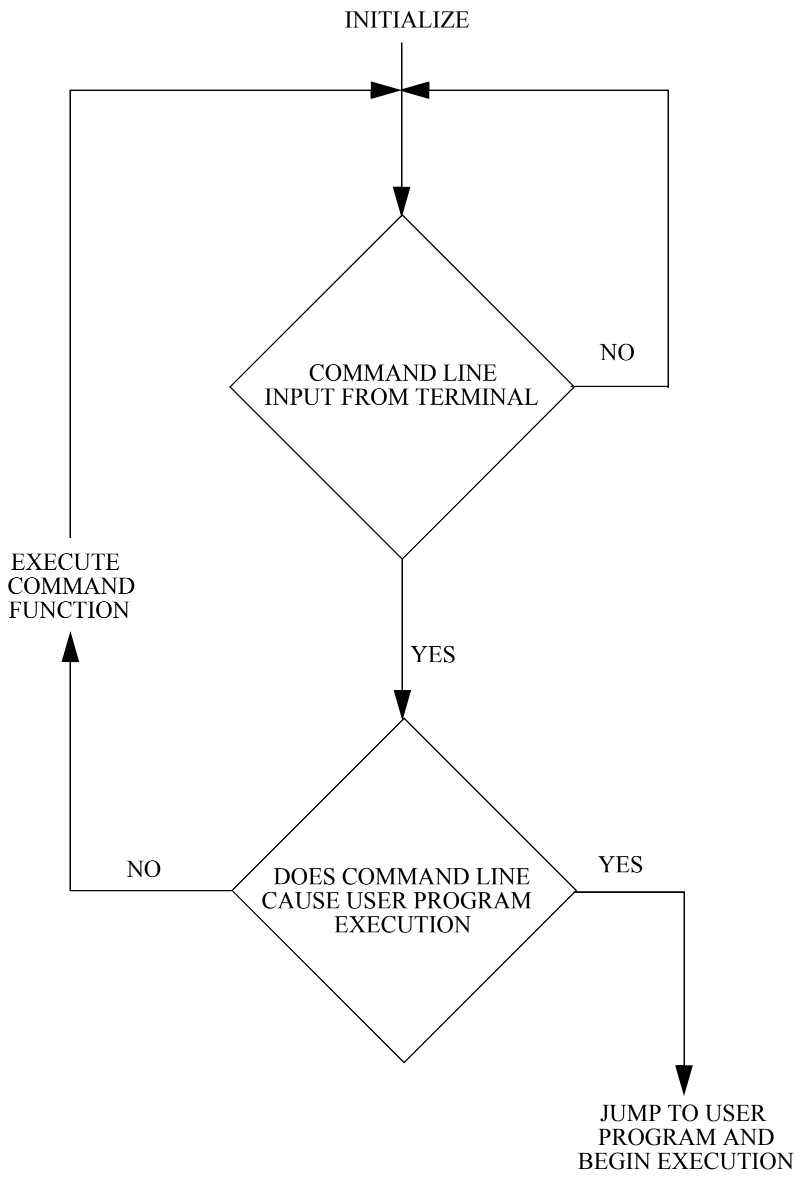

INITIALIZE
COMMAND LINE INPUT FROM TERMINAL
NO
EXECUTE COMMAND FUNCTION
YES
NO
DOES COMMAND LINE CAUSE USER PROGRAM EXECUTION
YES
JUMP TO USER PROGRAM AND BEGIN EXECUTION

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:

```
Hard Reset
DRAM Size: 16M

Copyright 1995-2004 Motorola, Inc. All Rights Reserved.
ColdFire MCF5275 EVS Firmware v2e.1a.xx (Build XXX on XXX XX 20XX
xx:xx:xx)

Enter 'help' for help.


dBUG>
```

Other means can be used to re-initialize the M5275EVB firmware. These means are discussed in the following paragraphs.

3.2.2.1 External RESET Button

External RESET (SW5) is the red button. Depressing this button causes all processes to terminate, resets the MCF5275 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 (SW4) is the button located next to the RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5275) 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 MCF5275 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) with no flow control. 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
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 then 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 assemble 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 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.

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 MCF5275. 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 MCF5275 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 MCF5275. 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 M5275EVB 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:

Do we want to update

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

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 to contain the value 0x1234, the command is:

```
rm      D0 1234
```

RESET and dBUG Reset the Board

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, 19200, 38400, 57600, and 115200 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

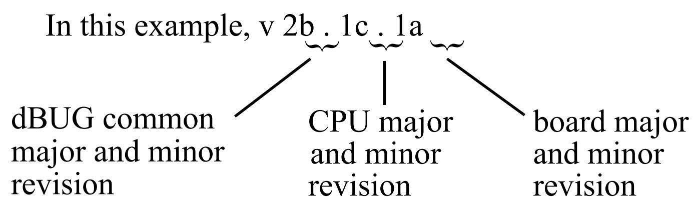
The UP command uploads the data from a memory region (specified by begin and end) to a file (specified by filename) over the network. The file created contains the raw binary data from the specified memory region. The UP command uses the Trivial File Transfer Protocol (TFTP) to transfer files to a network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


In this example, v 2b . 1c . 1a
dBUG common major and minor revision
CPU major and minor revision
board major and minor revision

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

Examples:

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

```
        version
```

3.5 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 the lower 8 bits of D1, to the 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:52:75: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.

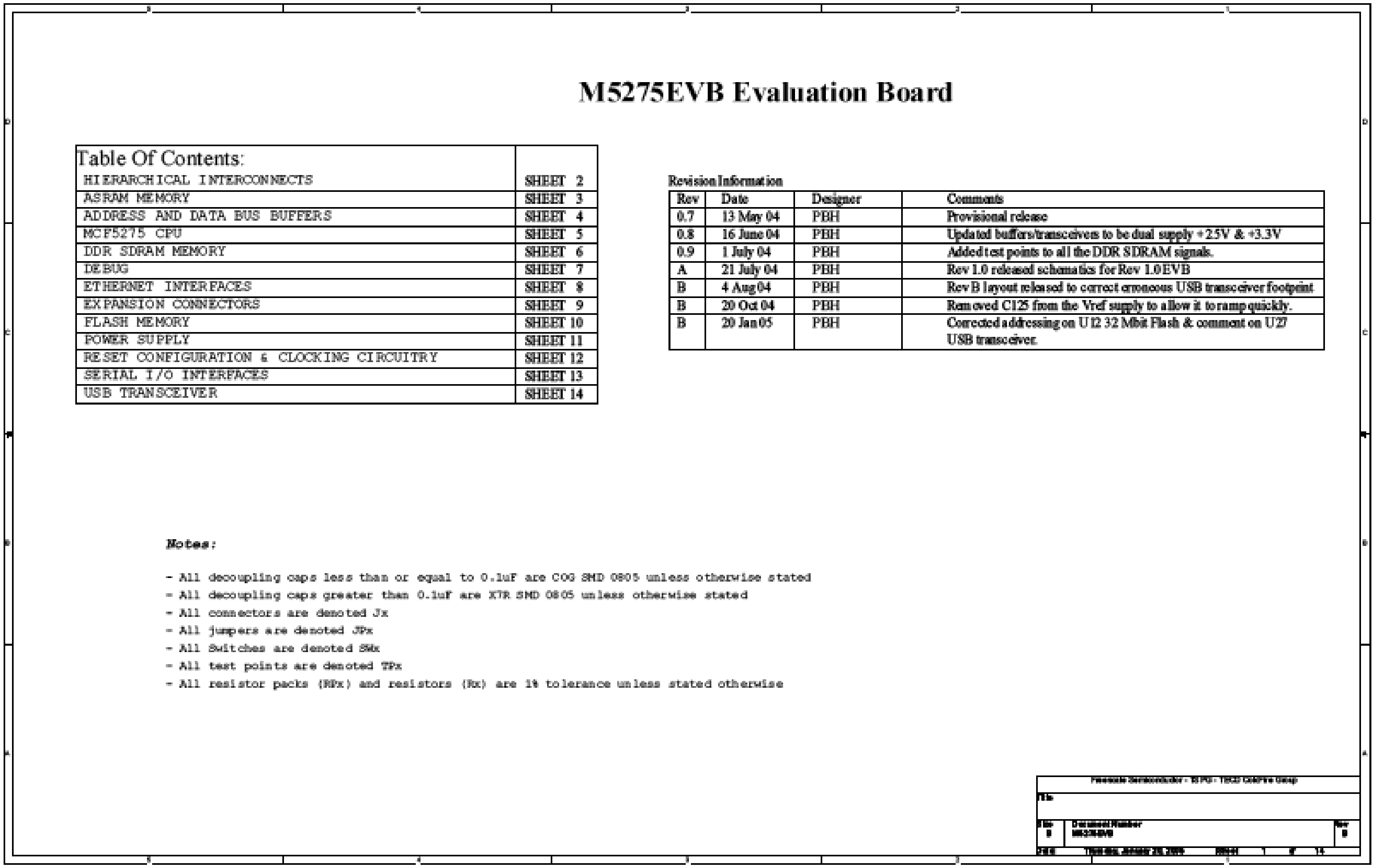
Check for proper insertion or connection of the network cable. Is the status LED lit indicating that network traffic is present?

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

B.1 M5275EVB Schematics



M5275EVB Evaluation Board

Table Of Contents:

Revision Information

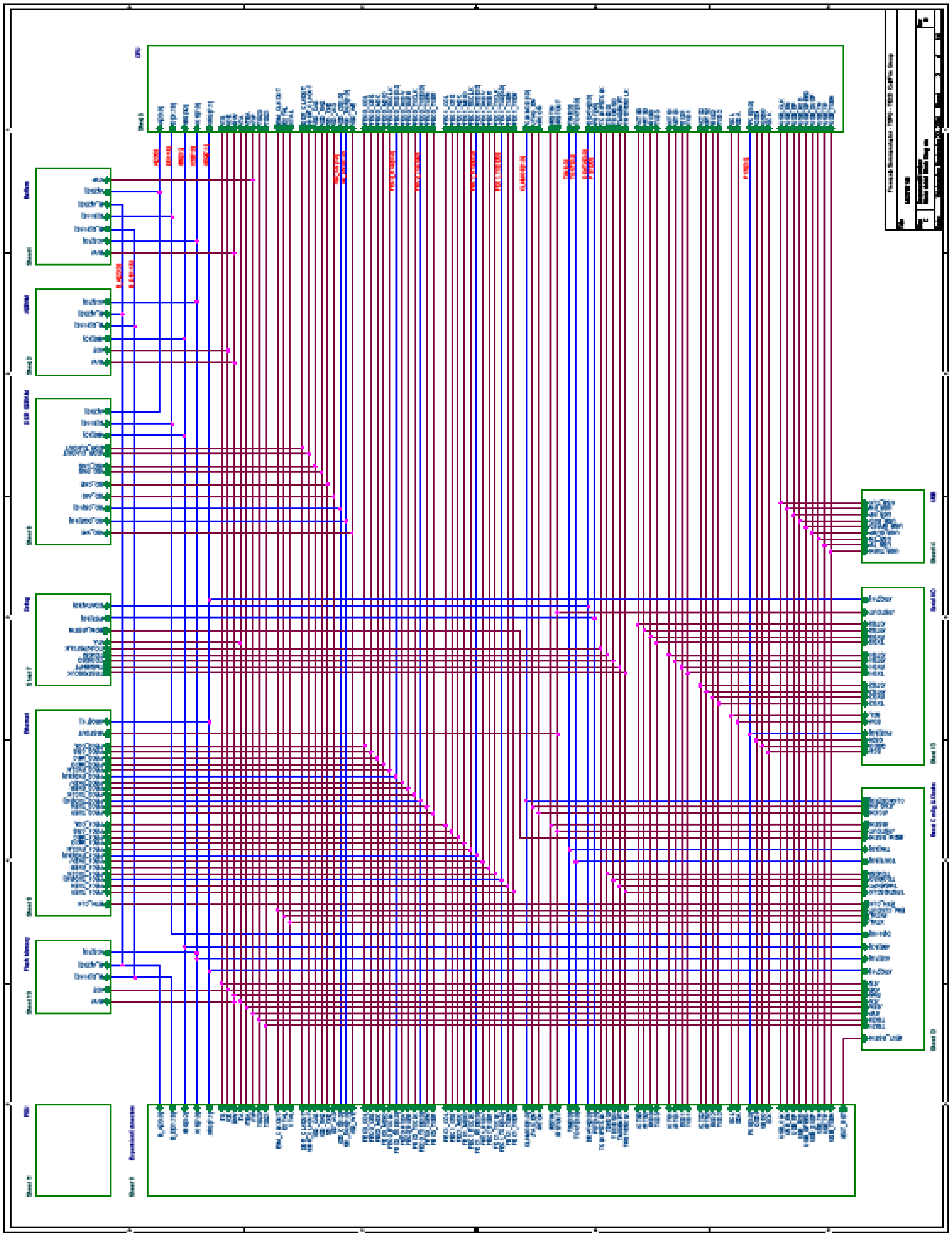
Rev	Date	Designer	Comments
0.7	13 May 04	PBH	Provisional release
0.8	16 June 04	PBH	Updated buffers/transceivers to be dual supply +2.5V & +3.3V
0.9	1 July 04	PBH	Added test points to all the DDR SDRAM signals.
A	21 July 04	PBH	Rev 1.0 released schematics for Rev 1.0 EVB
B	4 Aug 04	PBH	Rev B layout released to correct erroneous USB transceiver footprint
B	20 Oct 04	PBH	Removed C125 from the Vref supply to allow it to ramp quickly.
B	20 Jan 05	PBH	Corrected addressing on U12 32 Mbit Flash & comment on U27 USB transceiver.

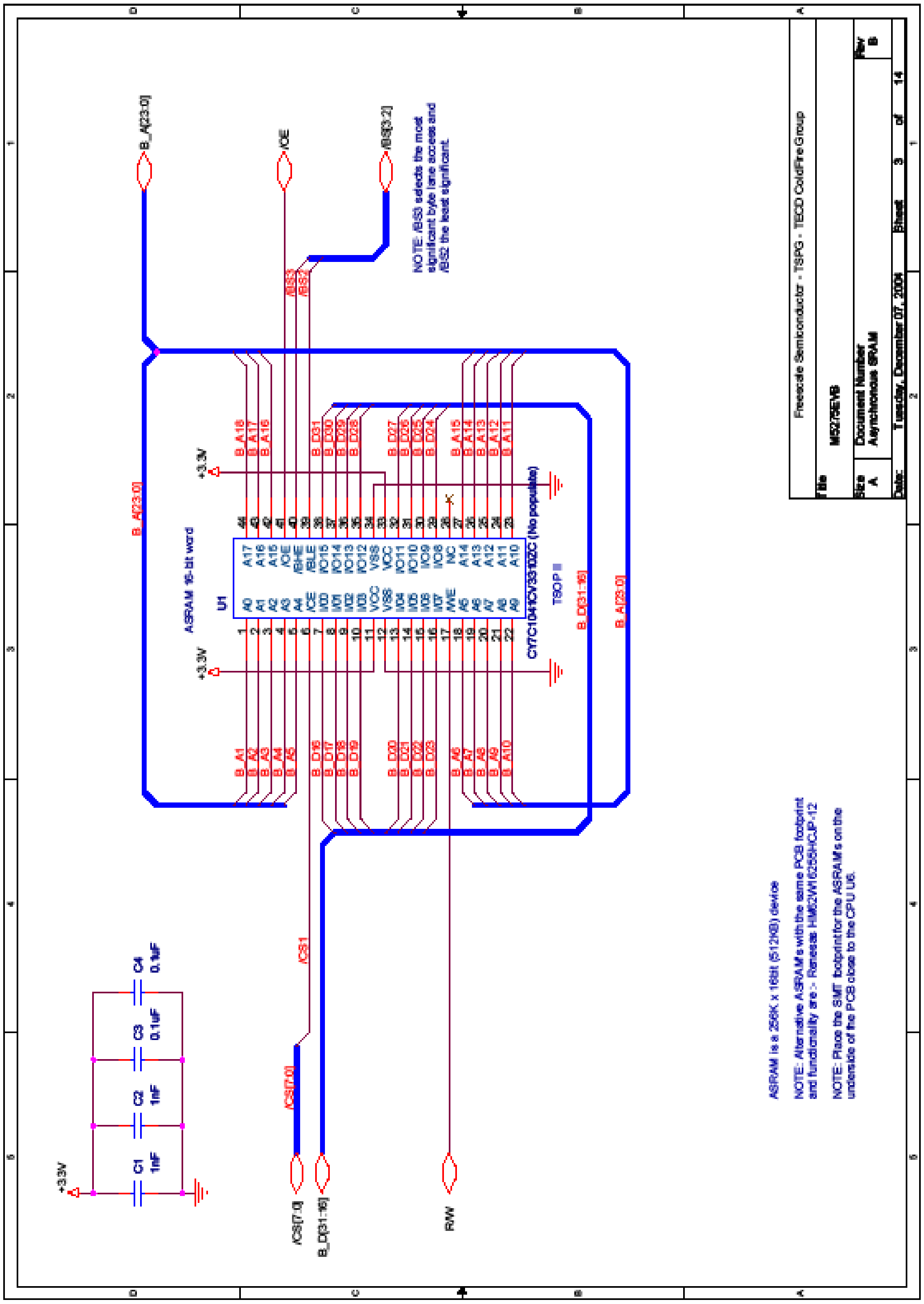
Notes:

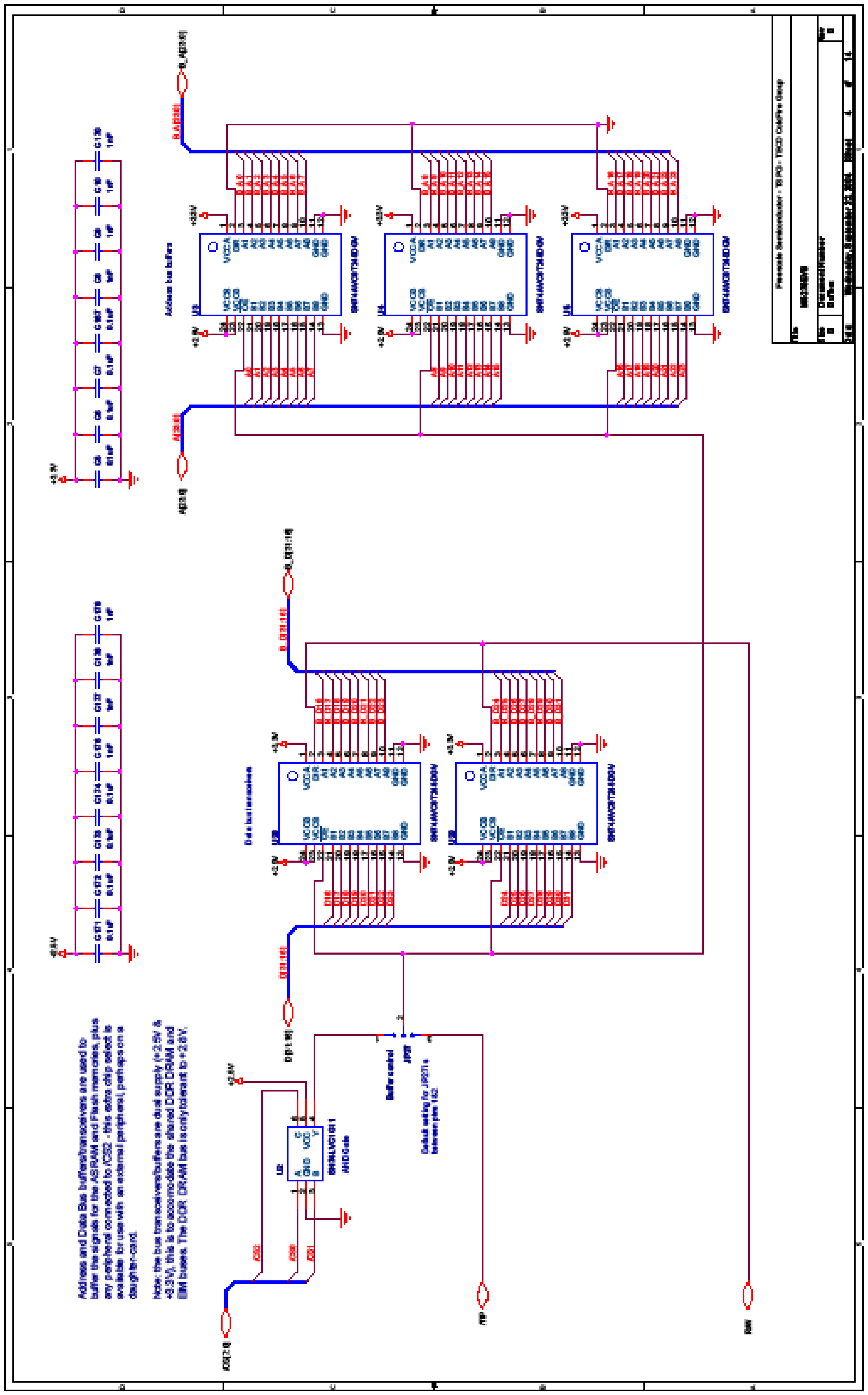
- All decoupling caps less than or equal to 0.1uF are COG SMD 0805 unless otherwise stated
- All decoupling caps greater than 0.1uF are X7R SMD 0805 unless otherwise stated
- All connectors are denoted Jx
- All jumpers are denoted JPx
- All Switches are denoted SWx
- All test points are denoted TPx
- All resistor packs (RPx) and resistors (Rx) are 1% tolerance unless stated otherwise

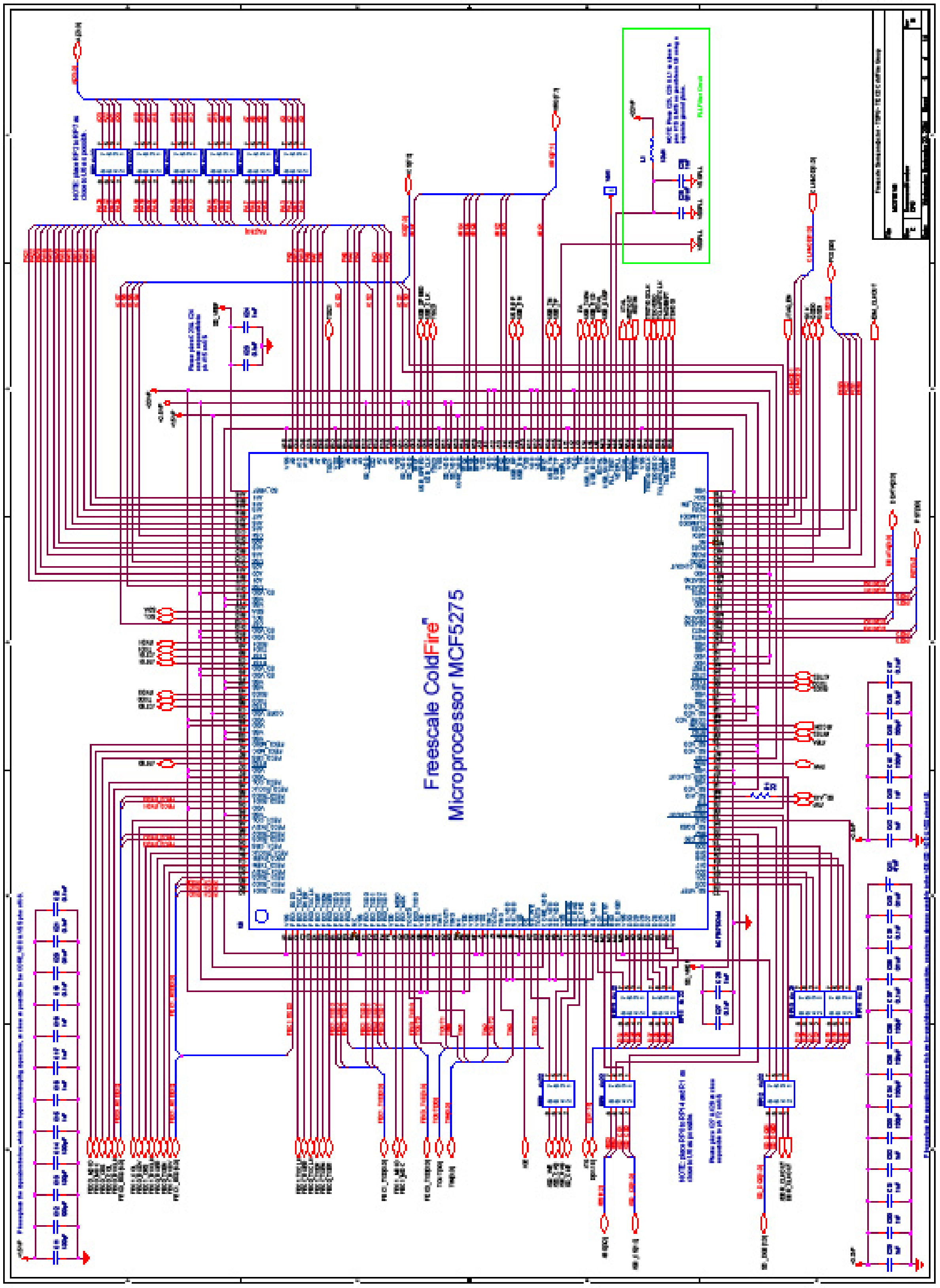
Freescale Semiconductor - TSPG - TECD ColdFire Group

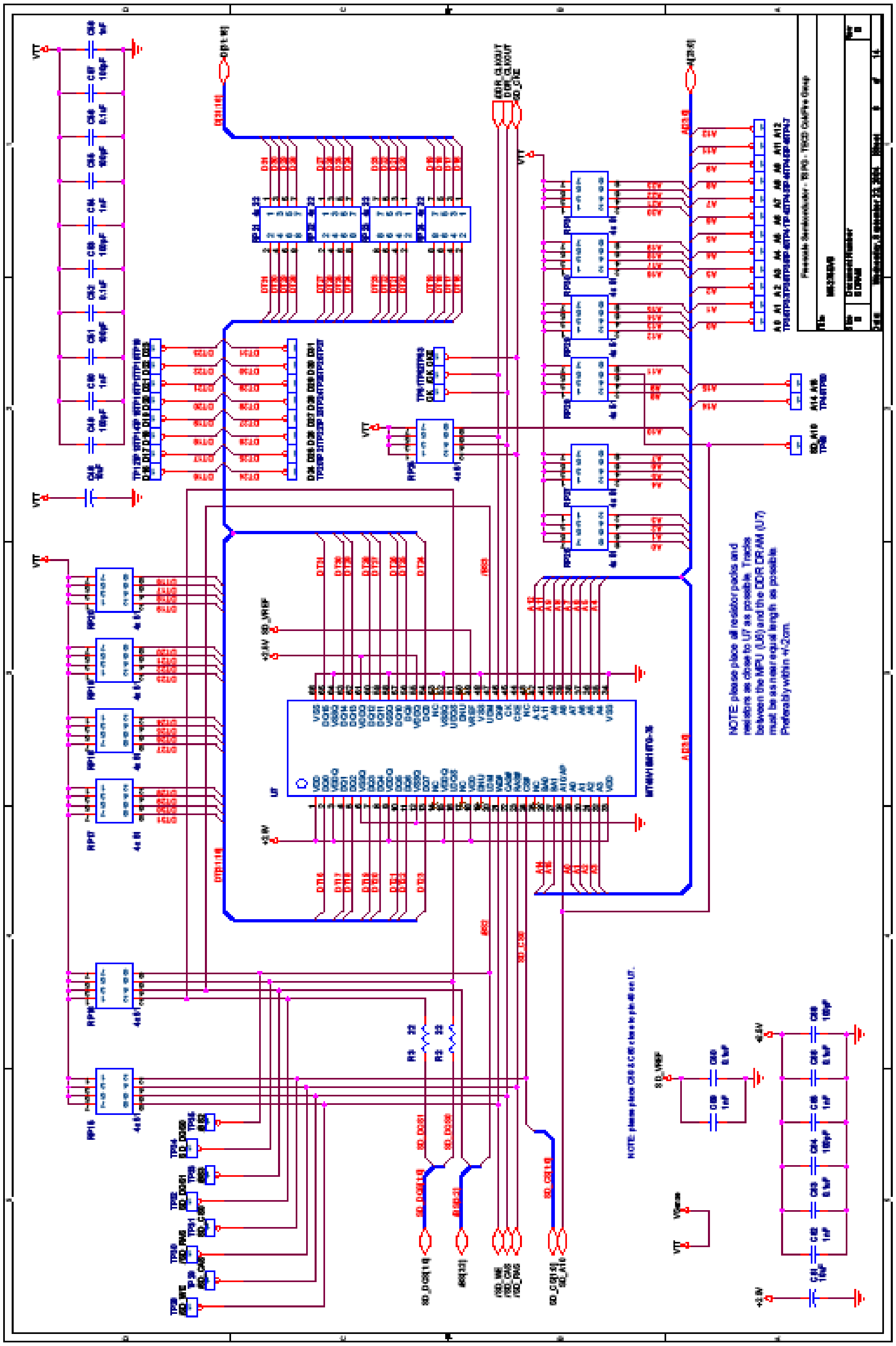
Title		
Size B	Document Number M5275EVB	Rev B
Date: Thursday, January 20, 2005	Sheet 1	of 14

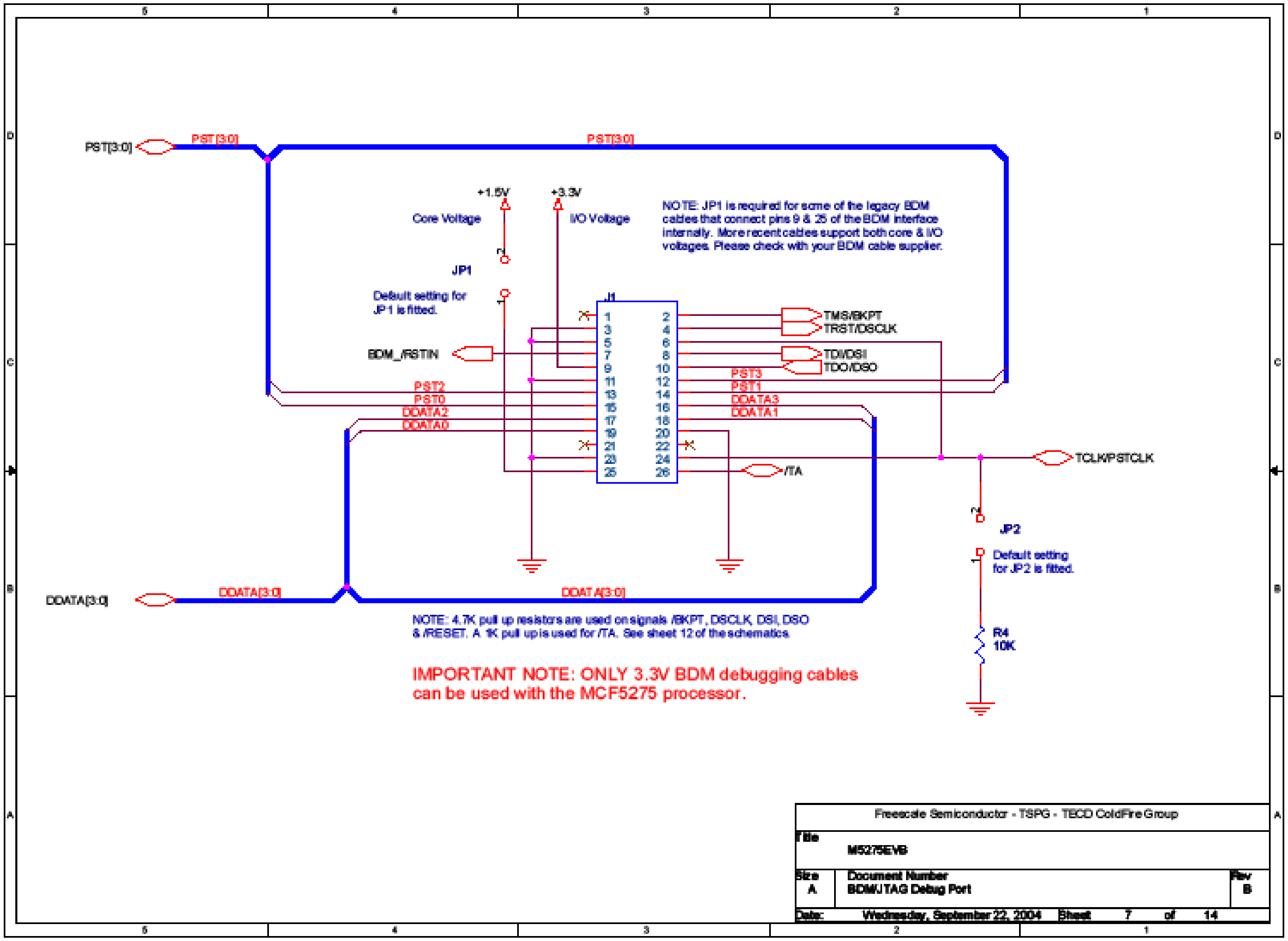



B_A[23:0]
/OE
/BS[3:2]
NOTE: /BS3 selects the most significant byte lane access and /BS2 the least significant.
/BS3
/BS2
+3.3V
ASRAM 16-bit word
U1
CY7C1041CV33-10ZC (No populate)
TSOP II
B_D[31:16]
B_A[23:0]
A0 A1 A2 A3 A4 /CE IO0 IO1 IO2 IO3 VCC VSS IO4 IO5 IO6 IO7 /WE A5 A6 A7 A8 A9
A17 A16 A15 /OE /BHE /BLE IO15 IO14 IO13 IO12 VSS VCC IO11 IO10 IO9 IO8 NC A14 A13 A12 A11 A10
B_A18 B_A17 B_A16 B_D31 B_D30 B_D29 B_D28 B_D27 B_D26 B_D25 B_D24 B_A15 B_A14 B_A13 B_A12 B_A11
B_A1 B_A2 B_A3 B_A4 B_A5 B_D16 B_D17 B_D18 B_D19 B_D20 B_D21 B_D22 B_D23 B_A6 B_A7 B_A8 B_A9 B_A10
/CS1
/CS[7:0]
B_D[31:16]
R/W
+3.3V
C1 1nF
C2 1nF
C3 0.1uF
C4 0.1uF
ASRAM is a 256K x 16bit (512KB) device
NOTE: Alternative ASRAMs with the same PCB footprint and functionality are :- Renesas HM62W16255HCJP-12
NOTE: Place the SMT footprint for the ASRAMs on the underside of the PCB close to the CPU U6.
Freescale Semiconductor - TSPG - TECD ColdFire Group
Title
M5275EVB
Size A
Document Number
Asynchronous SRAM
Rev B
Date: Tuesday, December 07, 2004
Sheet 3 of 14


Address and Data Bus buffers/transceivers are used to buffer the signals for the ASRAM and Flash memories, plus any peripheral connected to /CS2 - this extra chip select is available for use with an external peripheral, perhaps on a daughter-card.
Note: the bus transceivers/buffers are dual supply (+2.5V & +3.3V), this is to accomodate the shared DDR DRAM and BM buses. The DDR DRAM bus is only tolerant to +2.8V.
Address bus buffers
Data bus transceivers
SN74AVC16T245DGV
SN74LVC1G11
AND Gate
Buffer control
JP27
Default setting for JP27 is between pins 1&2
A[23:0]
B_A[23:0]
D[31:16]
B_D[31:16]
R/W
/TIP
/CS[7:0]
+3.3V
+2.5V
Freescale Semiconductor - 16 PQ - TSCD ColdFire Group
M5275EVB
Buffers


Freescale ColdFire
Microprocessor MCF5275





NOTE: JP1 is required for some of the legacy BDM cables that connect pins 9 & 25 of the BDM interface internally. More recent cables support both core & I/O voltages. Please check with your BDM cable supplier.

NOTE: 4.7K pull up resistors are used on signals /BKPT, DSCLK, DSI, DSO & /RESET. A 1K pull up is used for /TA. See sheet 12 of the schematics.

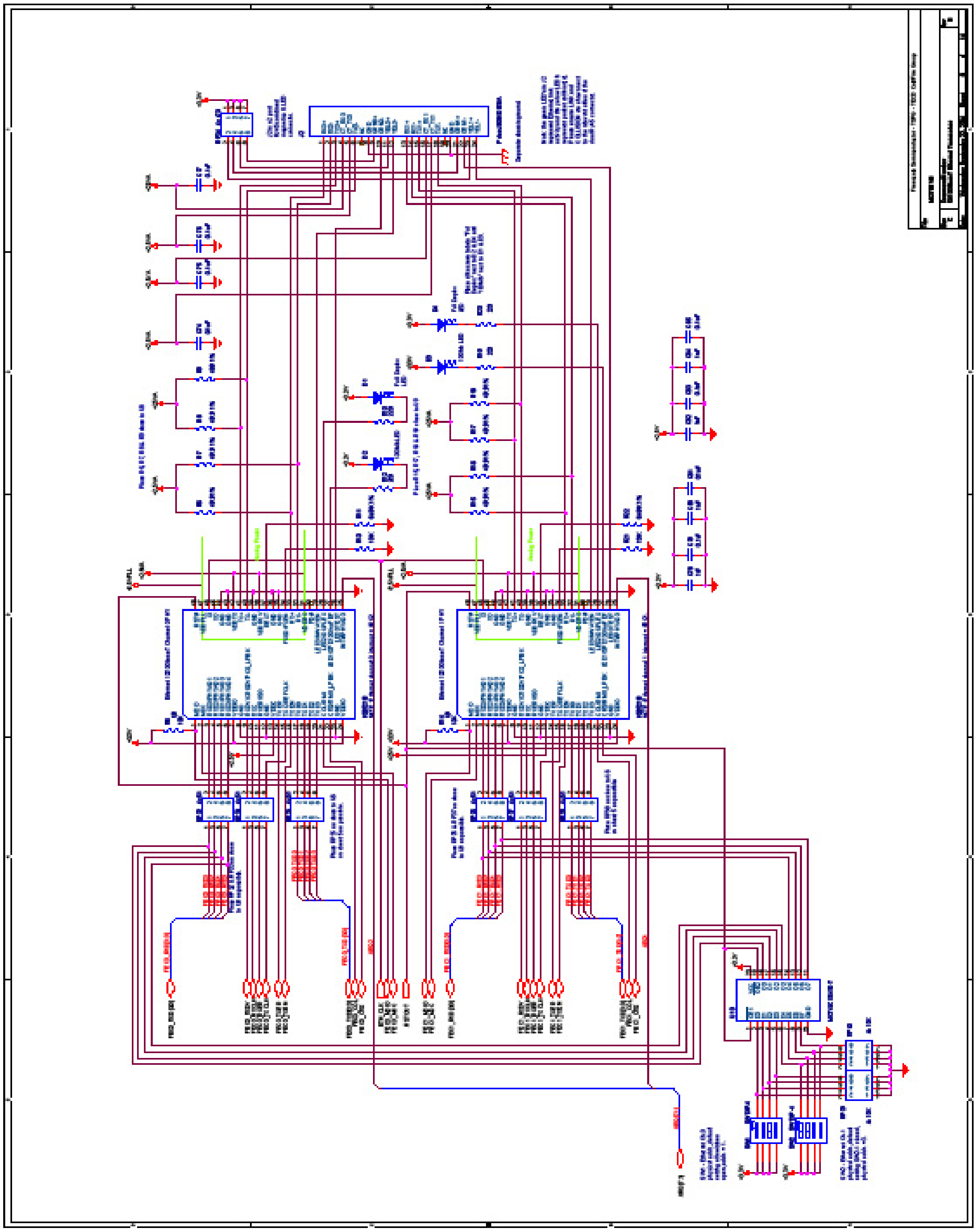
IMPORTANT NOTE: ONLY 3.3V BDM debugging cables can be used with the MCF5275 processor.

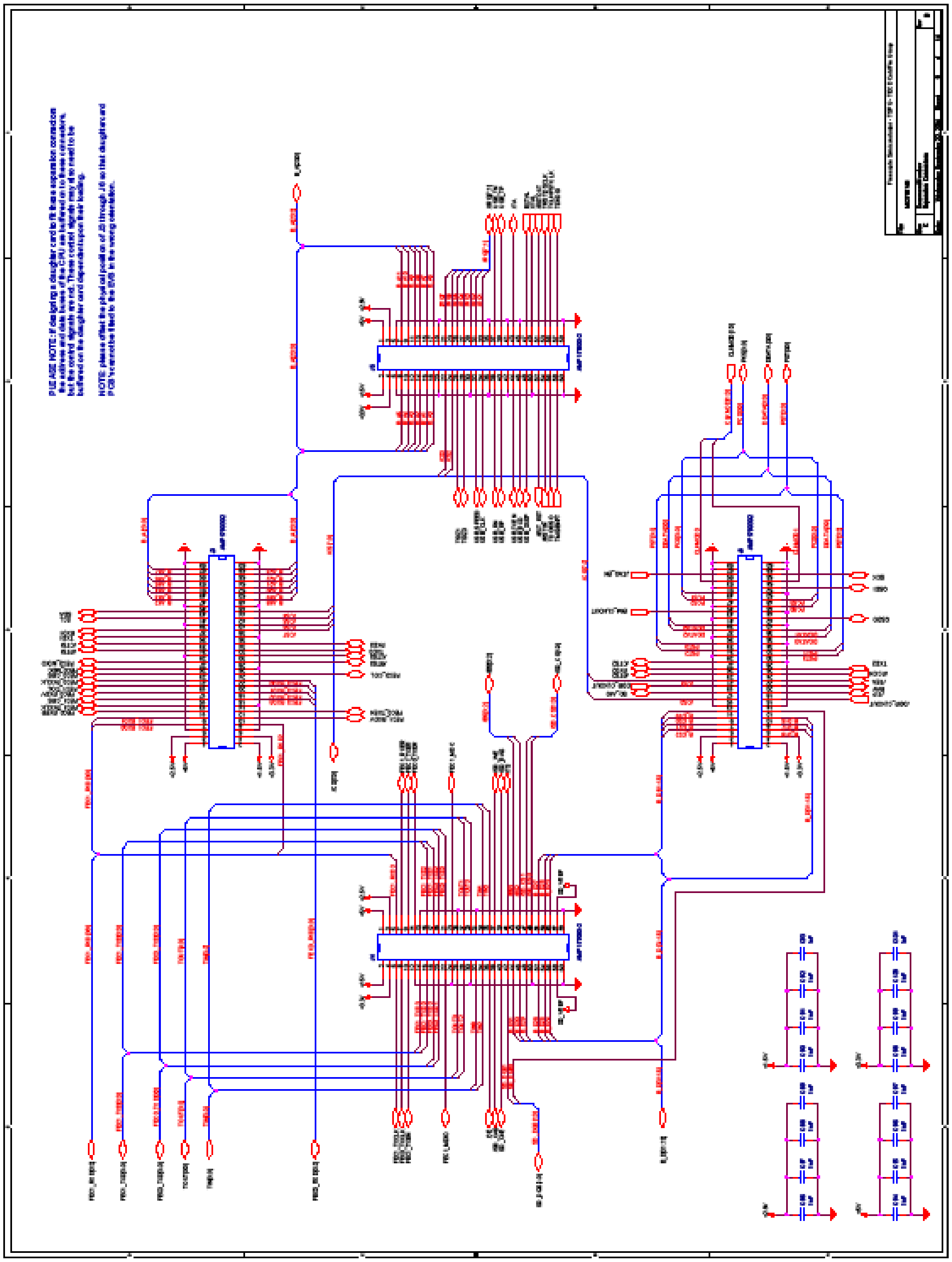
Freescale Semiconductor - TSPG - TECD ColdFire Group

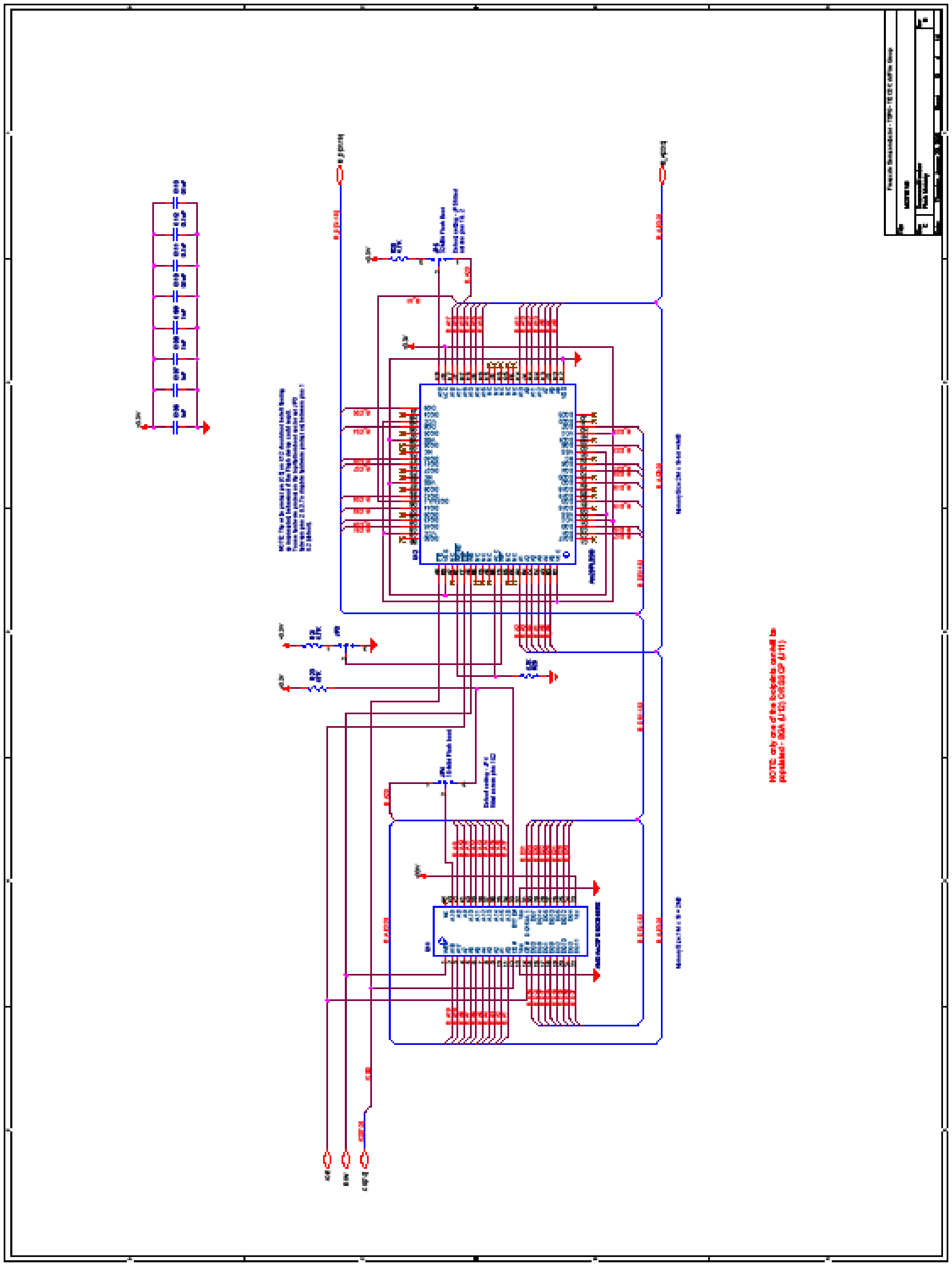
Title: M5275EVB

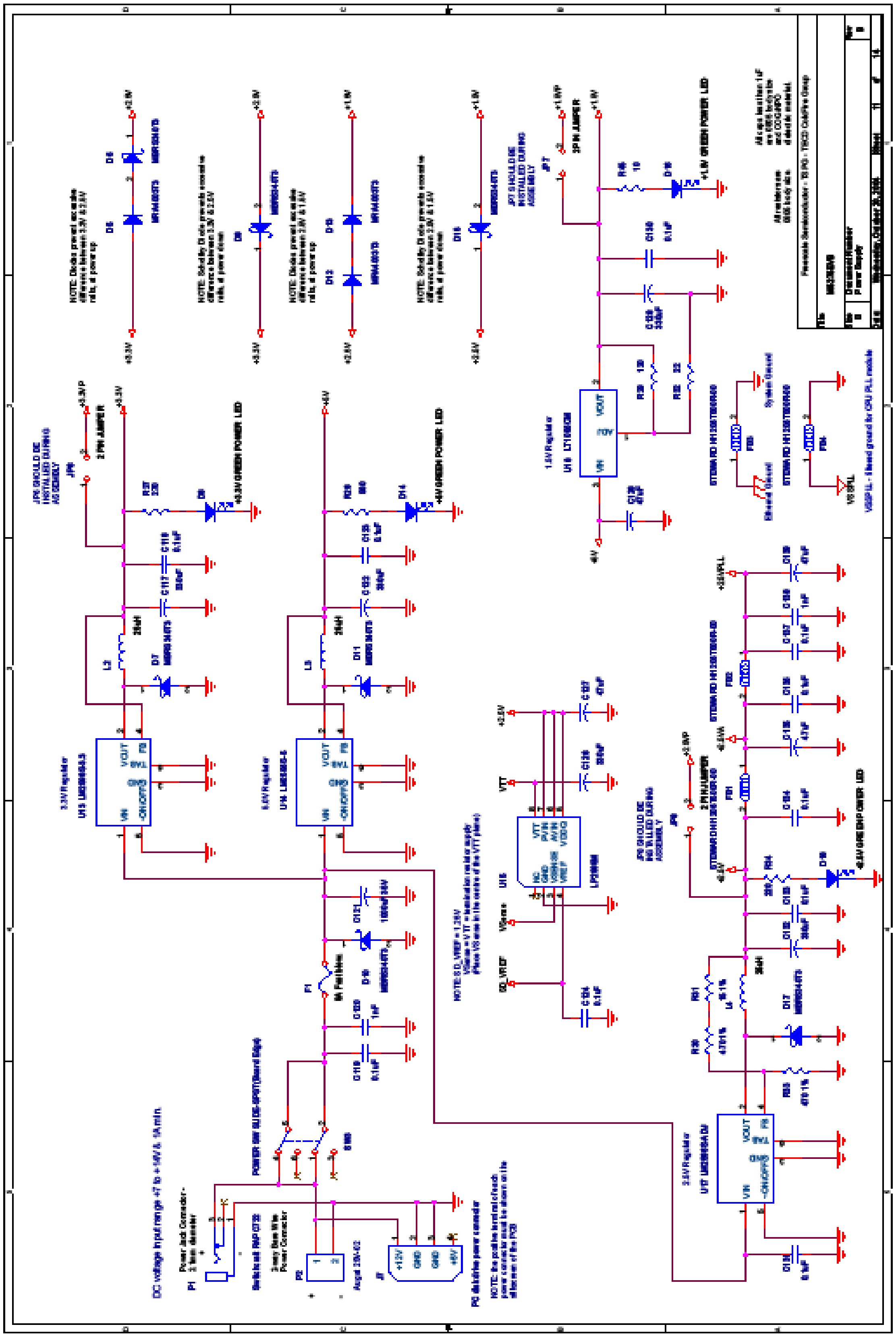
Size: A | Document Number: BDM/JTAG Debug Port | Rev: B

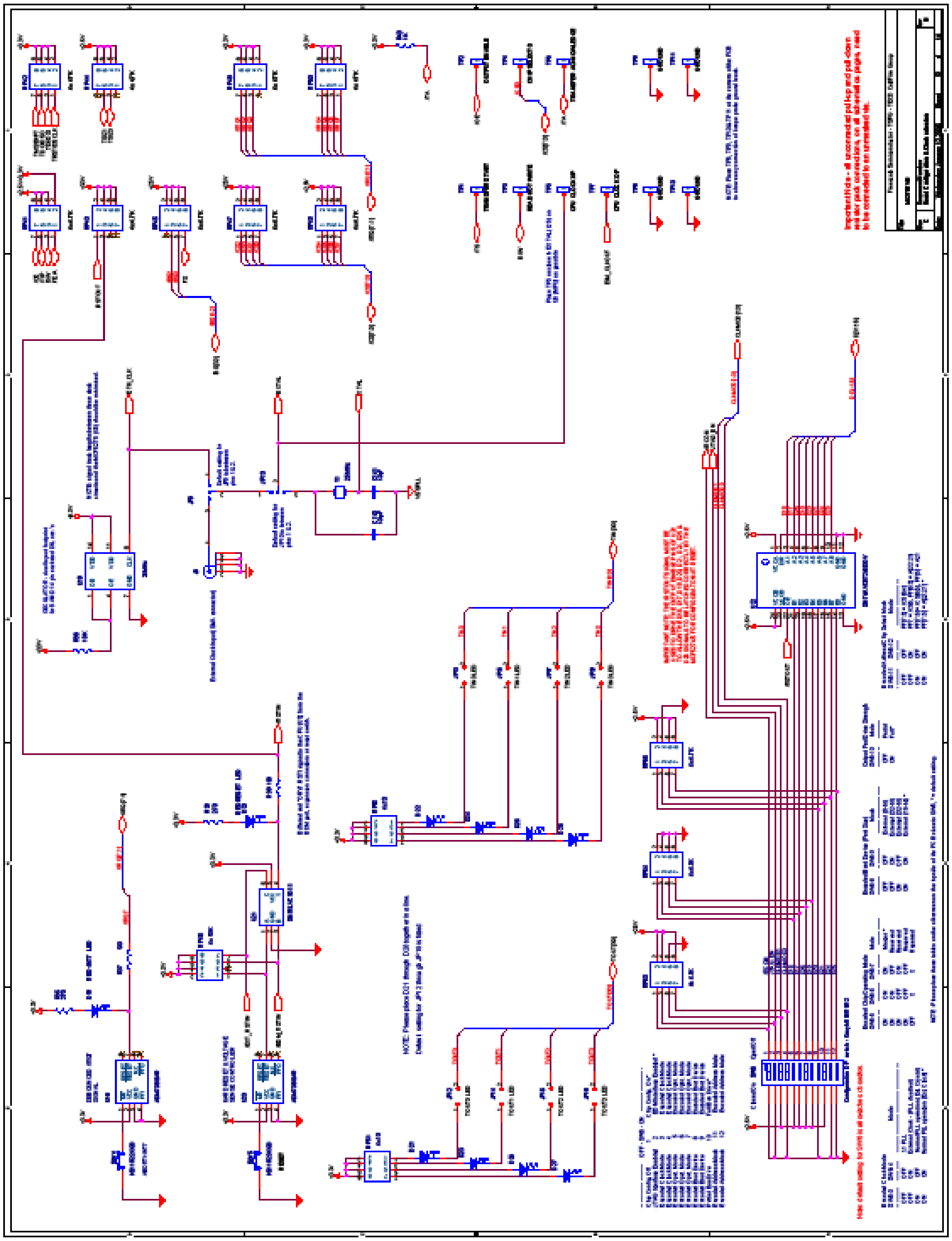
Date: Wednesday, September 22, 2004 | Sheet 7 of 14

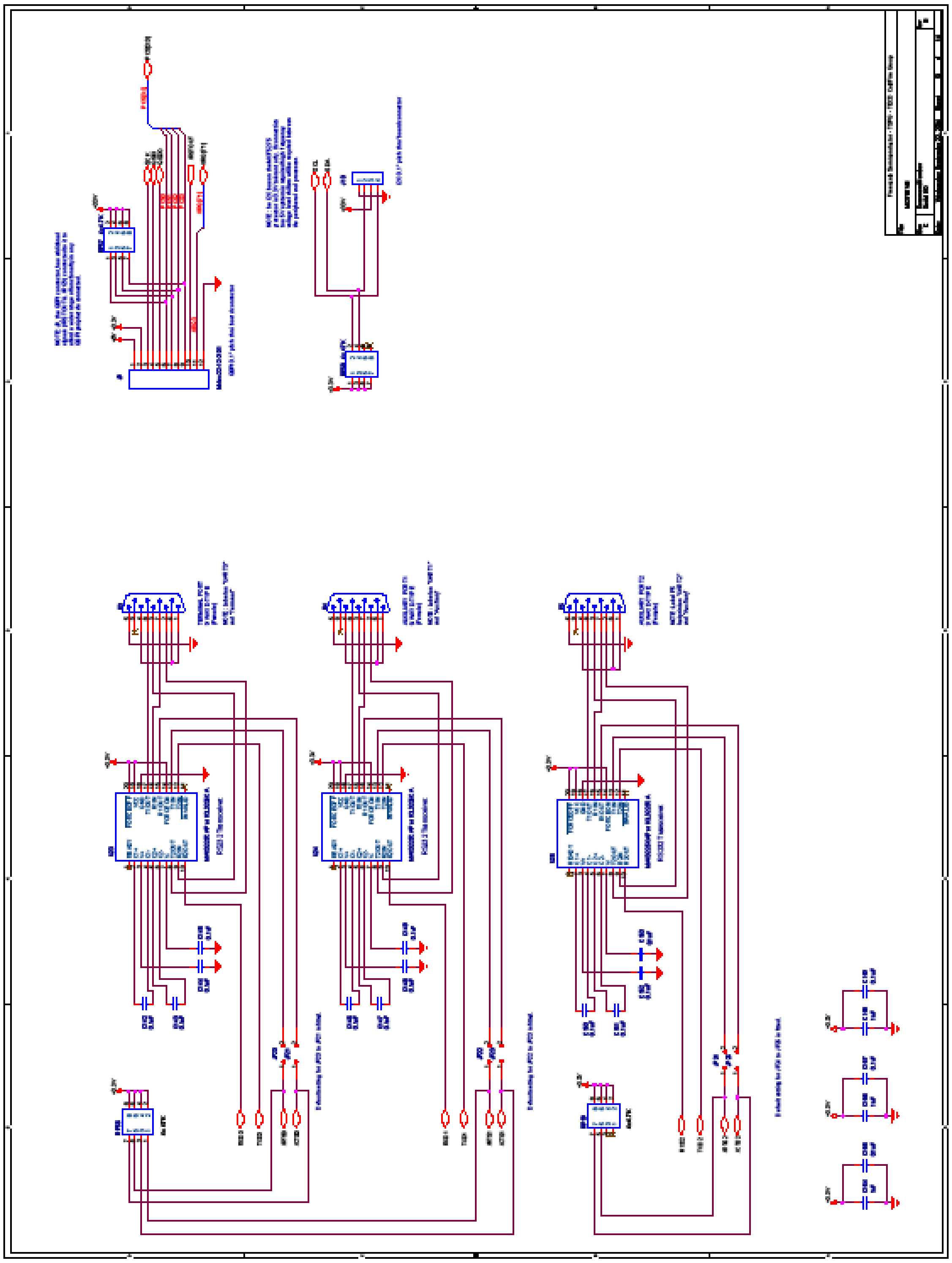


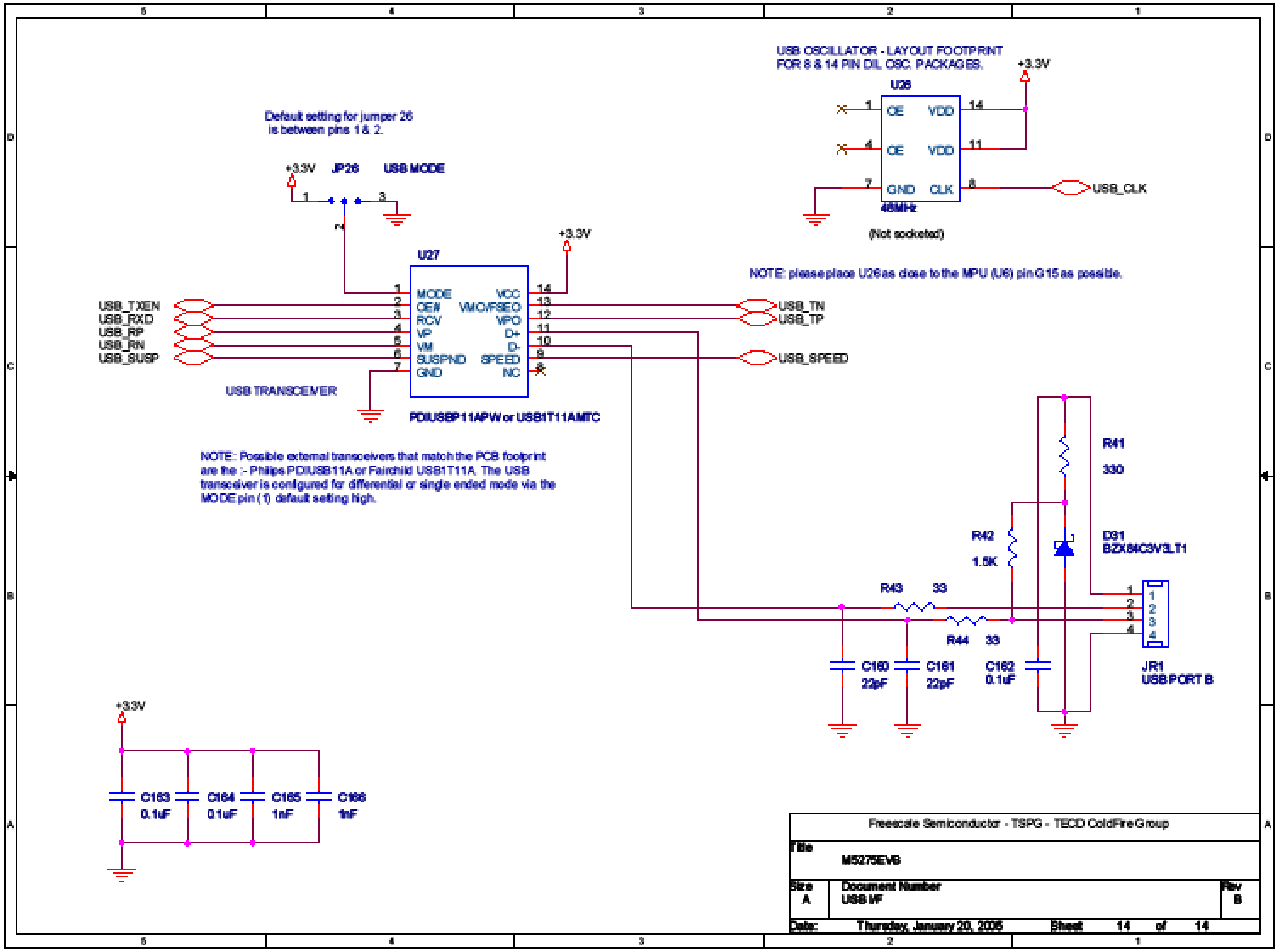











USB OSCILLATOR - LAYOUT FOOTPRINT
FOR 8 & 14 PIN DIL OSC. PACKAGES.
+3.3V
U26
OE
VDD
OE
VDD
GND
CLK
48MHz
(Not socketed)
USB_CLK
NOTE: please place U26 as close to the MPU (U6) pin G15 as possible.
Default setting for jumper 26
is between pins 1 & 2.
+3.3V
JP26
USB MODE
U27
MODE
OE#
RCV
VP
VM
SUSPND
GND
VCC
VMO/FSEO
VPO
D+
D-
SPEED
NC
USB_TXEN
USB_RXD
USB_RP
USB_RN
USB_SUSP
USB_TN
USB_TP
USB_SPEED
USB TRANSCEIVER
PDIUSBP11APW or USB1T11AMTC
NOTE: Possible external transceivers that match the PCB footprint
are the :- Philips PDIUSB11A or Fairchild USB1T11A. The USB
transceiver is configured for differential or single ended mode via the
MODE pin (1) default setting high.
R41
330
R42
1.5K
D31
BZX84C3V3LT1
R43
33
R44
33
C160
22pF
C161
22pF
C162
0.1uF
JR1
USB PORT B
C163
0.1uF
C164
0.1uF
C165
1nF
C166
1nF
Freescale Semiconductor - TSPG - TECD ColdFire Group
Title
M5275EVB
Size
A
Document Number
USB I/F
Rev
B
Date: Thursday, January 20, 2005
Sheet 14 of 14

Appendix C M5275EVB BOM

C.1 M5275EVB BOM

BOM		M5275EVB Evaluation Board Revised: Tuesday, June 22, 2004			*Rapid* PCB
QTY's		PART INFORMATION			521525
Item	Assy.	Part Number	Manufacturer	Description & Package Type	Ref. Des.
1	65	0805CG102J9BB0	Phycom	Cap., 1nF,NPO or COG, 25v, 0805,5%	C1,C2,C8,C9,C10,C15,C16,C17,C18,C24,C26,C28,C29,C30,C31,C32,C42,C43,C50,C54,C58,C59,C62,C65,C68,C71,C78,C80,C82,C84,C86,C87,C88,C89,C90,C91,C92,C93,C94,C95,C96,C97,C98,C99,C100,C101,C106,C107,C108,C109,C114,C120,C138,C154,C156,C158,C165,C166,C169,C170,C176,C177,C178,C179,C180
2	74	08055C104KAT2A	KOA	Cap, 0.1uf,25v,X7R, 0805,10%	C3,C4,C5,C6,C7,C19,C20,C21,C22,C23,C25,C27,C37,C38,C39,C40,C46,C47,C52,C56,C60,C63,C66,C67,C72,C74,C75,C76,C77,C79,C81,C83,C85,C110,C111,C112,C113,C115,C116,C118,C119,C123,C124,C130,C131,C133,C134,C136,C137,C142,C143,C144,C145,C146,C147,C148,C149,C150,C151,C152,C153,C155,C157,C159,C162,C163,C164,C167,C168,C171,C172,C173,C174,C175
3	19	100pF SMD 50V 0805 5%	Venkel	Cap., 100pf, 0805. 50V, NPO or COG	C11,C12,C13,C14,C33,C34,C35,C36,C44,C45,C49,C51,C53,C55,C57,C64,C69,C70,C73
4	5	TPSC476K016R0350	AVX	Cap, 47uf, C case	C41,C127,C129,C135,C139
5	2	TA020TCM106KBR	AVX	Cap., 10uF, 20v, 10% B case tant.	C48,C61
6	6	TPSE337K010R0100	AVX	Cap, 330uf, 10v, D case, SM/CT_7343_12	C117,C122,C125,C126,C128,C132
7	1	ECA-1VM102 or UVZ1H102MHH		Cap., 1000uF 35VCPCYL1/D.500/LS.200/.040	C121
8	2	10pF		Cap, 10pF SMD 50V 0805 5%	C140,C141
9	2	22pF		Cap, 22pF SMD 50V 0805 5%	C160,C161
10	8	AA3528SGC	Kingbright	LED, Green, SMT	D1,D2,D3,D4,D8,D14,D16,D18
11	3	MRA4003T3	Motorola	SMA	D5,D12,D13
12	7	MBRS340T3	Motorola	SMC	D6,D7,D9,D10,D11,D15,D17
14	2	AA3528SRC	Kingbright	LED, Red, SMT	D19,D20
15	8	AA3528MBC or TLMB3100-GS08	Kingbright	LED, Blue, SMT	D21,D22,D23,D24,D25,D26,D27,D28
16	2	MMBD301LT1	Motorola		D29,D30
17	1	BZX84C3V3LT1			D31
18	4	STEWARD HI1206T500R-00		Ind.1206	FB1,FB2,FB3,FB4
19	1	4527K Fuse Holder by KEYSTONE + 0216005.H : Fuse by Littlefuse, 5a, 250v, 5 x 20 mm, glass			F1
20	16	S2105-02 or M22-2510205		JP2M	JP1,JP2,JP12,JP13,JP14,JP15,JP16,JP17,JP18,JP19,JP20,JP21,JP22,JP23,JP24,JP25
21	5	S2105-03M22-2510305		JUMPER3_0	JP3,JP4,JP5,JP9,JP10
22	3	S2105-02 or M22-2510205		JP2M	JP6,JP7,JP8
23	1	S2105-03M22-2510305		JUMPER3_0	JP26
24	1	S2105-02 or M22-2510205		JP2M Buffer control	JP27
25	1	787780-1		USB PORT B	JR1
26	1	609-2627	Thomas & Betts		J1
27	1	J8064D628A	Pulse		J2
28	4	179030-2	AMP		J3,J4,J5,J6
29	1	350211-1	AMP	PC disk drive power connector	J7
30	1	1053378-1 or 901-143-6RFX			J8
31	1	PZC10SAAN or 22-12-2101		Conn, 1 x 10 male header	J9
32	1	PZC04SAAN or 22-10-2041		Conn, 1 x 4 male header	J10
33	1	1210-103J	API Delevan	Ind. 10uh, 1210 ferrite	L1
34	3	B82111-B-C24	SIEMENS		L2,L3,L4
35	1	RAPC722	Switchcraft		P1
36	1	25V-02	Augat		P2
37	2	747844-3	AMP	DB9 RS232 PORT THRU HOLE DB9	P3,P4

38	1			DB9 RS232 PORT THRU HOLE DB9 (No populate)	P5	
39	16	4x 4.7K	KOA or Philips		RP1,RP41,RP42,RP43,RP44,RP47,RP48,RP49,RP50,RP53,RP54,RP55,RP56,RP57,RP58,RP59	
40	17	4x 22	KOA or Philips		RP2,RP3,RP4,RP5,RP6,RP7,RP8,RP9,RP10,RP11,RP12,RP13,RP14,RP21,RP22,RP23,RP24	
41	19	4x 51	KOA or Philips		RP15,RP16,RP17,RP18,RP19,RP20,RP25,RP26,RP27,RP28,RP29,RP30,RP31,RP32,RP33,RP35,RP36,RP37,RP38	
42	1	4x 470	KOA or Philips		RP34	
43	3	4x 10K	KOA or Philips		RP39,RP40,RP46	
44	2	4x 10	KOA or Philips		RP51,RP52	
45	4	22	KOA or Philips		R1,R2,R3,R32	
46	6	10K	KOA or Philips		R4,R5,R10,R14,R21,R36	
47	8	49.9 1%	KOA or Philips		R6,R7,R8,R9,R15,R16,R17,R18	
48	2	6.49K 1%	KOA or Philips		R11,R22	
49	5	220	KOA or Philips		R12,R13,R19,R20,R34	
50	4	4.7K	KOA or Philips		R23,R24,R25,R26	
51	3	270	KOA or Philips		R27,R35,R38	
52	1	560	KOA or Philips		R28	
53	1	120	KOA or Philips		R29	
54	2	470 1%	KOA or Philips		R30,R33	
55	1	15 1%	KOA or Philips		R31	
56	2	100	KOA or Philips		R37,R39	
57	1	1K	KOA or Philips		R40	
58	1	330	KOA or Philips		R41	
59	1	1.5K	KOA or Philips		R42	
60	2	33	KOA or Philips		R43,R44	
61	1	10	KOA or Philips		R45	
62	2	78RB04S	Grayhill	SW DIP-4	SW1,SW2	
63	1	25546NA6(silver perfered)25546NLD (gold plate)	Apem	POWER SW SLIDE-SPST(Board Edge)	SW3	
64	1	KS11R22CQD	C & K		SW4	
65	1	KS11R23CQD	C & K		SW5	
66	1	78RB12	Grayhill	Configuration DIP switch	SW6	
67	11	5015K	Keystone	SMT, TEST Point	TP1,TP2,TP3,TP4,TP5,TP6,TP7,TP8,TP9,TP10,TP11	
68	1	CY7C1041CV3310ZC (No populate)			U1	
69	2	SN74LVC1G11	TI	IC, SOT-223, SOP-6	U2,U21	
70	5	SN74AVC8T245DGV	TI		U3,U4,U5,U28,U29	
71	1	MCF5475CVM			U6	
72	1	MT46V16M16TG-75	Micron		U7	
73	2	KS8721B	Micrel		U8,U9	
74	2	MC74LCX541DT	On Semi	IC, TSSOP 20	U10,U22	
75	1	AMD Am29PL160CB-65RS	AMD	IC, SO, 44 PIN	U11	
76	1	Am29PL320D	AMD	IC, BGA-84P-M01	U12	
77	1	LM2596S-3.3	National	IC, TO-263-5	U13	
78	1	LM2596S-5	National	IC, TO-263-5	U14	
79	1	LP2995M	National	IC REGULATOR DDR TERM 8-SOIC	U15	
80	1	LT1086CM	Linear Tech.	IC, TO-220-3-SM	U16	
81	1	LM2596S-ADJ	National	IC, TO-263-5	U17	
82	2	ADM708SAR	Analog Devices	IC, SOIC 8	U18,U20	
83	1	FOXS/250F-20	FOX	Xtal, HC49C	Y	
84	1	P1145-HCV series	Pletronics	OSC, 3.3V, 25 Mhz, 4 pin, T. H.	U19	
85	3	MAX3225CAP or	Maxim		U23,U24,U25	
86	1	P1145-HCV series		OSC, 3.3V, 48 Mhz, 4 pin, T. H.	U26	
87	1	PDIUSBP11APW	Philips	IC,14PIN,SSOP,INTERFACE,TRANSCEIVER	U27	
		NO POP	CUSTOMER SUPPLIED			