M5271EVB

User's Manual

ColdFire
Evaluation Board

M5271EVBUM
Rev. 1.1
05/2006

freescale.com

freescale™
semiconductor

EMC Information on M5271EVB

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.

Table of Contents

Chapter 1
M5271EVB Introduction

1.1 MCF5271 Microprocessor 1-3
1.2 System Memory 1-6
1.2.1 External Flash 1-6
1.2.2 SDRAM. 1-7
1.2.3 ASRAM. 1-7
1.2.4 Internal SRAM 1-7
1.2.5 M5271EVB Memory Map 1-7
1.2.6 Reset Vector Mapping 1-8
1.3 Support Logic 1-9
1.3.1 Reset Logic 1-9
1.3.2 Clock Circuitry 1-11
1.3.3 Watchdog Timer 1-11
1.3.4 Exception Sources 1-11
1.3.5 TA Generation 1-12
1.3.6 User's Program 1-12
1.4 Communication Ports 1-13
1.4.1 UART0, UART1, UART2 Ports 1-13
1.4.2 10/100T Ethernet Port 1-13
1.4.3 BDM/JTAG Port 1-14
1.4.4 I2C. 1-15
1.4.5 QSPI 1-15
1.5 Connectors and User Components 1-16
1.5.1 Daughter Card Expansion Connectors 1-16
1.5.2 Reset Switch (SW3) 1-20
1.5.3 User LEDs 1-20
1.5.4 Other LEDs 1-21

Chapter 2
Initialization and Setup

2.1 System Configuration 2-1
2.2 Installation and Setup 2-3
2.2.1 Unpacking 2-3
2.2.2 Preparing the Board for Use 2-3
2.2.3 Providing Power to the Board 2-4
2.2.4 Power Switch (SW1). 2-4

2.2.5 Power Status LEDs and Fuse 2-5
2.2.6 Selecting Terminal Baud Rate 2-5
2.2.7 The Terminal Character Format 2-5
2.2.8 Connecting the Terminal 2-5
2.2.9 Using a Personal Computer as a Terminal 2-6
2.3 System Power-up and Initial Operation 2-8
2.4 Using The BDM Port 2-8

Chapter 3
Using the Monitor/Debug Firmware

3.1 What Is dBUG? 3-1
3.2 Operational Procedure 3-2
3.2.1 System Power-up 3-2
3.2.2 System Initialization 3-4
3.2.2.1 External RESET Button 3-4
3.2.2.2 ABORT Button 3-4
3.2.2.3 Software Reset Command 3-4
3.3 Command Line Usage 3-4
3.4 Commands 3-5
ASM Assembler 3-7
BC Block Compare 3-8
BF Block Fill 3-9
BM Block Move 3-10
BR Breakpoints 3-11
BS Block Search 3-12
DC Data Conversion 3-13
DI Disassemble 3-14
DL Download Console 3-15
DLDBUG Download dBUG 3-16
DN Download Network 3-17
FL Flash Utilities 3-18
GO Execute 3-19
GT Execute To 3-20
IRD Internal Register Display 3-21
IRM Internal Register Modify 3-22
HELP Help 3-23
LR Loop Read 3-24
LW Loop Write 3-25
MD Memory Display 3-26
MM Memory Modify 3-27
MMAP Memory Map Display 3-28
RD Register Display 3-29
RM Register Modify 3-30
RESET Reset the Board and dBUG 3-31

SD Stack Dump 3-32
SET Set Configurations 3-33
SHOW Show Configurations 3-34
STEP Step Over 3-35
SYMBOL Symbol Name Management 3-36
TRACE Trace Into 3-37
UP Upload Data 3-38
VERSION Display dBUG Version 3-39
3.5 TRAP #15 Functions 3-40
3.5.1 OUT_CHAR 3-40
3.5.2 IN_CHAR 3-41
3.5.3 CHAR_PRESENT 3-41
3.5.4 EXIT_TO_dBUG 3-42

Appendix A
Configuring dBUG for Network Downloads

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

Appendix B
Schematics

B.1 MCF5271EVM Schematics B-1

Appendix C
M5271EVB BOM

C.1 M5271EVB BOM C-1

List of Figures

Figure 1-1 M5271EVB Block Diagram 1-3
Figure 1-2 MCF5271 Block Diagram. 1-5
Figure 1-3 External Memory Scheme. 1-6
Figure 1-4 J1- BDM Connector Pin Assignment 1-14

Figure 2-1 Minimum System Configuration 2-2
Figure 2-2 2.1mm Power Connector. 2-4
Figure 2-3 2-Lever Power Connector 2-4
Figure 2-4 Pin Assignment for Female (Terminal) Connector. 2-6
Figure 2-5 Jumper Locations 2-7

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

MCF5271EVM Schematics B-1

List of Tables

Table 1-1 M5270/71 Product Family 1-1
Table 1-2 The M5271EVB Default Memory Map 1-8
Table 1-3 D[20:19] External Boot Chip Select Configuration 1-8
Table 1-4 SW4-1 RCON. 1-9
Table 1-5 SW4-2 JTAG_EN. 1-9
Table 1-6 SW4-[4:3] Encoded Clock Mode 1-9
Table 1-7 SW4-5 Chip Configuration Mode 1-10
Table 1-8 SW4-[7:6] Boot Device 1-10
Table 1-9 SW4-8 Bus Drive Strength. 1-10
Table 1-10 SW4-[10:9] Address/Chip Select Mode. 1-10
Table 1-11 M5271EVB Clock Source Selection 1-11
Table 1-12 J3 1-16
Table 1-13 J4 1-17
Table 1-14 J5 1-18
Table 1-15 J6 1-19
Table 1-16 User LEDs 1-20
Table 1-17 LED Functions 1-21

Table 2-1 Power LEDs 2-5
Table 2-2 Pin Assignment for Female (Terminal) Connector 2-6

Table 3-1 dBUG Command Summary 3-5

Table C-1 M5271EVB BOM C-1

Chapter 1
M5271EVB Introduction

This document details the setup and configuration of the ColdFire M5271EVB evaluation board (hereafter referred to as the EVB). The EVB is intended to provide a mechanism for easy customer evaluation of the MCF5270 and MCF5271 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 MCF5270 and MCF5271 are supported.

The heart of the evaluation board is the MCF5271, the MCF5270 has a subset of the MCF5271 specification and can therefore be fully emulated using the MCF5271 device. Table 1-1 details the two devices.

Table 1-1. M5270/71 Product Family

Part Number	Package	FEC	CRYPTO
MCF5270AB100	160 QFP	Yes	No
MCF5270VM100	196 MAPBGA	Yes	No
MCF5271CAB100	160 QFP	Yes	Yes
MCF5271CVM100	196 MAPBGA	Yes	Yes

NOTE

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:

- Freescale MCF5271 Microprocessor (100MHz max core frequency)
- External Clock source: 25MHz
- Operating temperature: 0°C to +70°C
- Power requirement: 6 – 14V DC @ 300 ma Typical
- Power output: 5V, 3.3V and 1.5V regulated supplies
- Board Size: 8.00 x 5.40 inches, 8 layers

Memory Devices:

- 16-Mbyte 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 MCF5271 device

Peripherals:

- Ethernet port 10/100Mb/s (Dual-Speed Fast Ethernet Transceiver, with MII)
- UART0 (RS-232 serial port for dBUG firmware)
- UART1 (auxiliary RS-232 serial port)
- UART2 (auxiliary RS-232 serial port)
- I^2C interface
- QSPI interface to ADC
- 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)

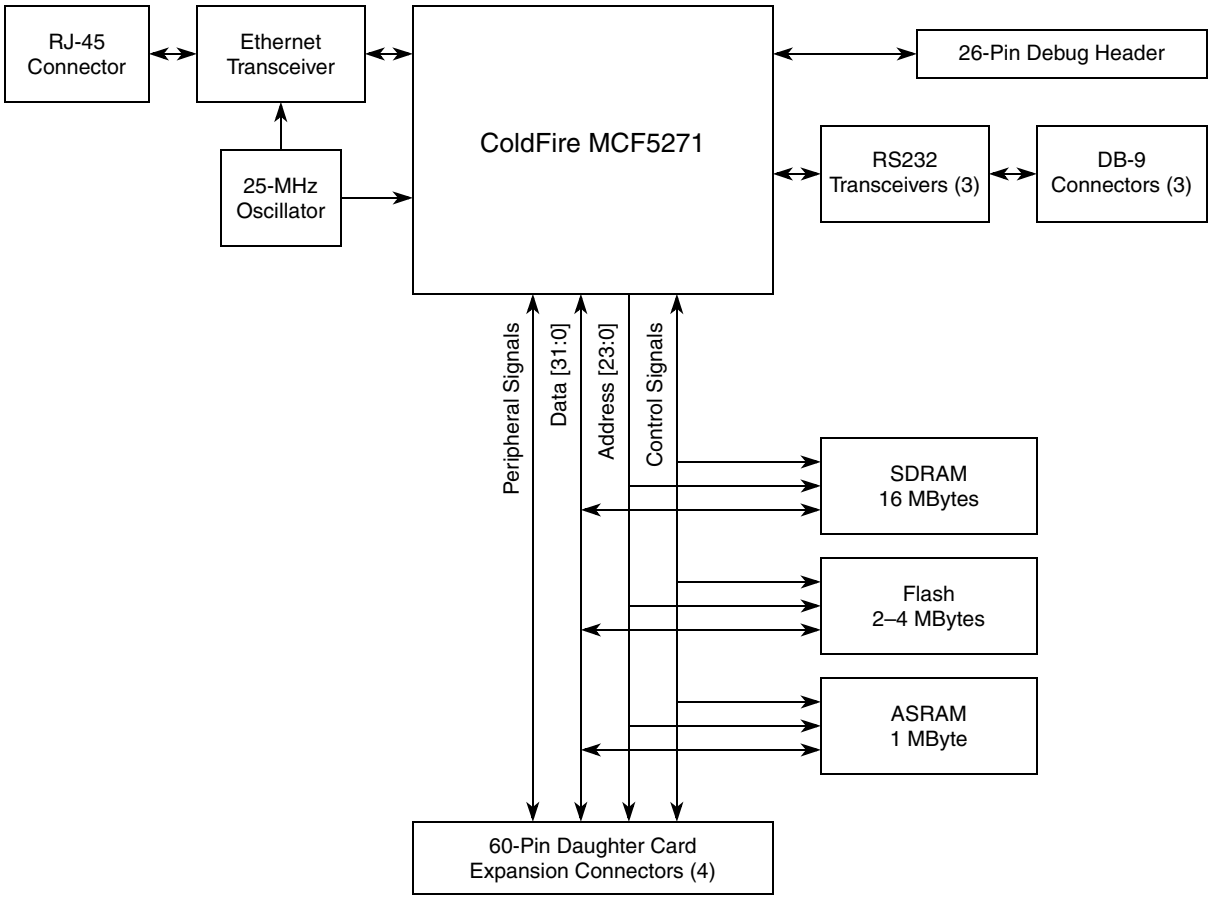


Figure 1-1. M5271EVB Block Diagram

1.1 MCF5271 Microprocessor

The microprocessor used on the EVB is the highly integrated Freescale MCF5271 32-bit ColdFire variable-length RISC processor. The MCF5271 implements a ColdFire Version 2 core with a maximum core frequency of 100MHz and external bus speed of 50MHz. Features of the MCF5271 include:

- V2 ColdFire core with enhanced multiply-accumulate unit (EMAC) providing 96 Dhrystone 2.1MIPS @ 100MHz
- 64 Kbytes of internal SRAM
- External bus speed of one half the CPU operating frequency (50MHz bus @ 100Mhz core)
- 10/100 Mbps bus-mastering Ethernet controller
- 8 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 2-bank synchronous DRAM controller
- 32-bit non-multiplexed bus with up to 8 chip select signals that support page-mode FLASH memories

The MCF5271 communicates with external devices over a 32-bit wide data bus, D[31:0]. The MCF5271 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 MCF5271 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 MCF5271 processor block diagram.

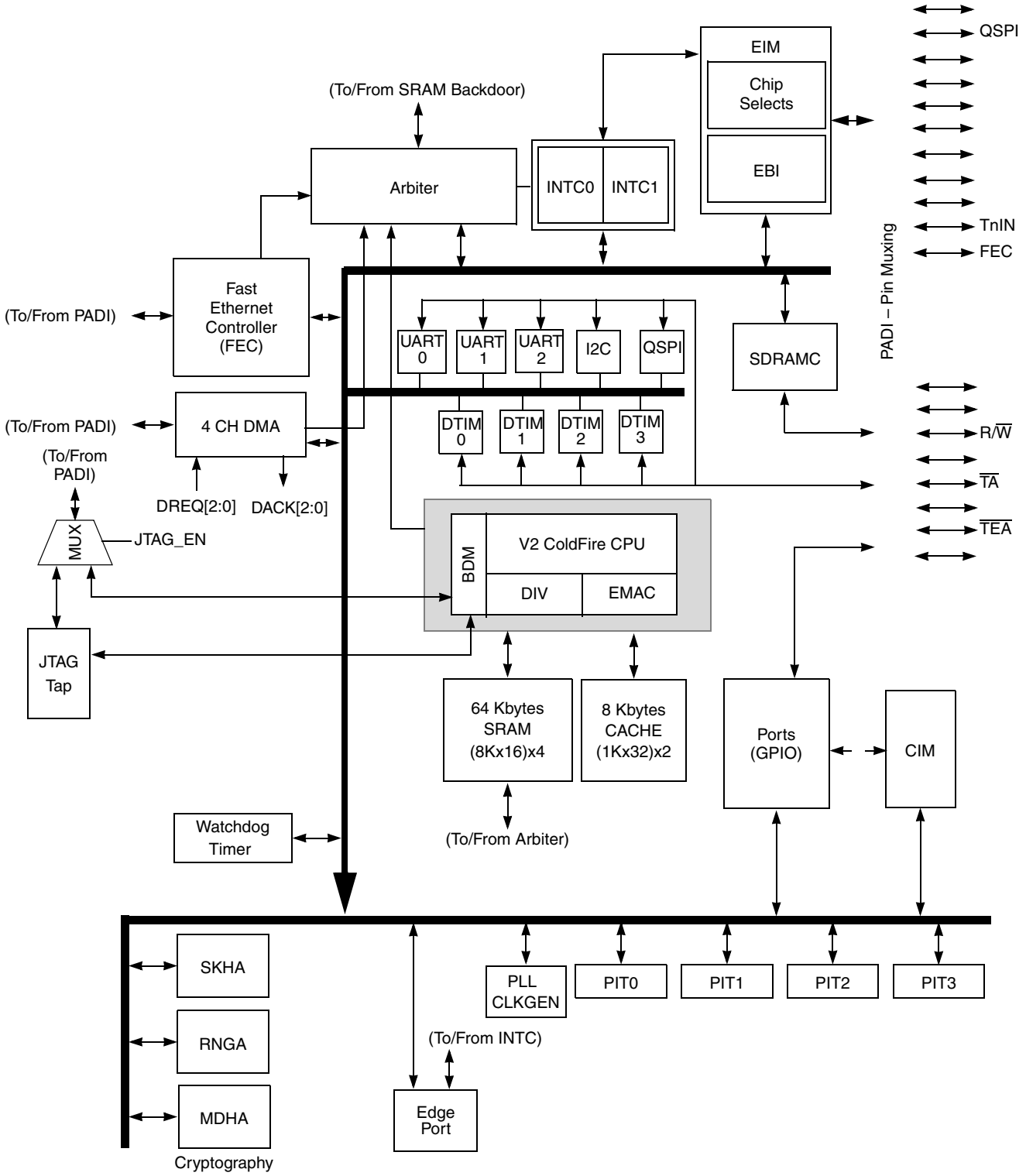


Figure 1-2. MCF5271 Block Diagram

1.2 System Memory

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

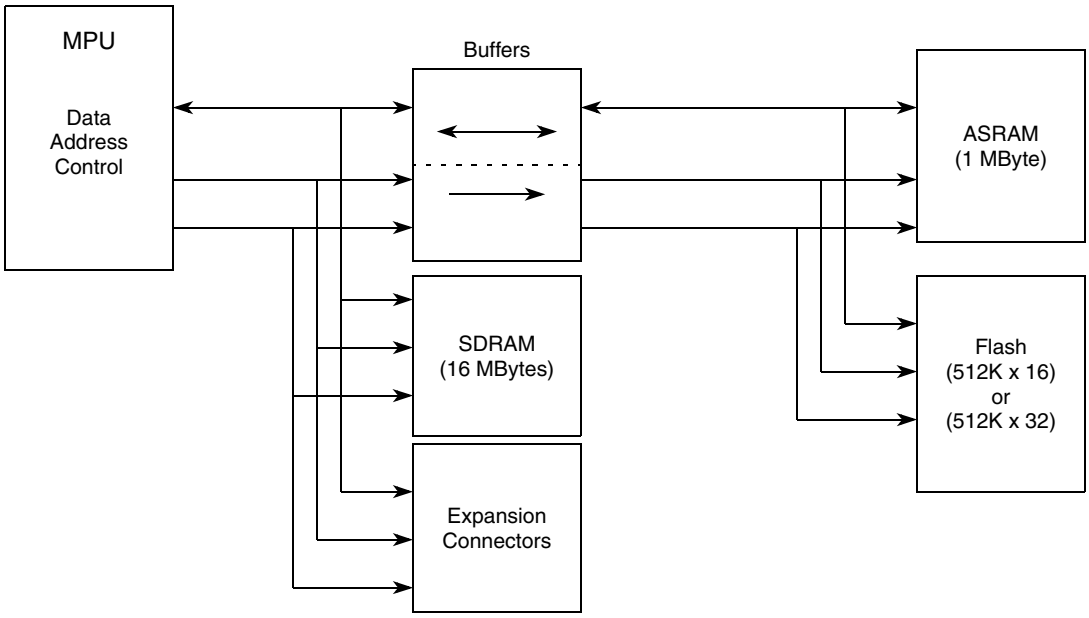


Figure 1-3. External Memory Scheme

NOTE

The external bus interface signals to the external ASRAM and FLASH (and USB) are buffered. This is in order not to exceed the maximum output load capacitance of the microprocessor on the EVB.

The signals to the expansion connectors remain unbuffered to provide a “true” interface to the user.

1.2.1 External Flash

The EVB is fitted with a single 512K x 16 page-mode FLASH memory (U10) giving a total memory space of 2Mbytes. Alternatively a footprint is available for the EVB user to upgrade this device to a 512K x 32 page-mode FLASH memory (U11), doubling the memory size to 4Mbytes. Either U10 OR U11 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 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 M5271EVB dBUG debugger/monitor firmware is programmed into the lower sectors of Flash (0xFFE0_0000 to 0xFFE3_FFFF for 2Mbytes of FLASH or 0xFFC0_0000 to 0xFFC3_FFFF for 4 Mbytes of FLASH).

When U11 is fitted on the EVB, jumper 5 (JP5) 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 two devices (Micron MT48LC4M16A2TG) each with a 16 bit data bus. Each device is organized as 1 Meg x 16 x 4 banks with a 16 bit data bus. One device stores the upper 16-bit word and the other the lower 16 bit word of the MCF5271 32 bit data bus.

1.2.3 ASRAM

The EVB has a footprint for two 512K x 16 Asynchronous SRAM devices (Cypress Semiconductor - CY7C1041CV3310ZC). These memory devices (U1 and U2) may be populated by the user for benchmarking purposes.

Also see Section 1.2.5, “M5271EVB Memory Map”.

1.2.4 Internal SRAM

The MCF5271 processor has 64-KBytes of internal SRAM memory which may be used as data or instruction memory. This memory is mapped to 0x2000_0000 and configured as data space but is not used by the dBUG monitor except during system initialization. After system initialization is complete, the internal memory is available to the user. The memory is relocatable to any 32-KByte boundary within the processor’s four gigabyte address space.

1.2.5 M5271EVB Memory Map

Interface signals to support the interface to external memory and peripheral devices are generated by the memory controller. The MCF5271 supports 8 external chip selects, $\overline{CS[1:0]}$ are used with external memories, and $\overline{CS[7:2]}$ are easily accessible to users via the daughter card expansion connectors. $\overline{CS[0]}$ also functions as the global (boot) chip-select for booting out of external flash.

Since the MCF5271 chip selects are fully programmable, the memory banks may be located at any 64-KByte boundary within the processor’s four gigabyte address space.

The default memory map for this board as configured by the Debug Monitor located in the external FLASH bank can be found in Table 1-2. The internal memory space of the MCF5271 is detailed further in the MCF5271 Users 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.

Chip Select usage:

External FLASH Memory	CS0
External ASRAM Memory	CS1

Table 1-2 shows the M5271EVB memory map

Table 1-2. The M5271EVB Default Memory Map

Address Range	Signal and Device
0x0000_0000–0x00FF_FFFF	16 Mbyte SDRAM
0x2000_0000–0x2000_FFFF	64 Kbytes Internal SRAM
0x3000_0000–0x300F_FFFF	External ASRAM (not fitted)
0xFFE0_0000–0xFFFF_FFFF or 0xFFC0_0000–0xFFFF_FFFF	2 Mbytes External Flash (U10) or 4 Mbytes External Flash (U11)

1.2.6 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 (0x00000000). The control registers specifying the operation of any memories (e.g., cache and/or RAM modules) connected directly to the processor are disabled.

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

The Memory that the MCF5271 accesses at address 0 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	External (32-bit)
01	External (16-bit)
10	External (8-bit)
11	External (32-bit)

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 MCF5271 to reset. $\overline{\text{RESET}}$ is triggered by the reset switch (SW3) which resets the entire processor/system.

The dBUG Firmware configures the MCF5271 microprocessor internal resources during initialization. The contents of the exception table are copied to address 0x0000_0000 in the SDRAM. The Software Watchdog Timer is disabled, the Bus Monitor is enabled, and the internal timers are placed in a stop condition. A memory map for the entire board can be seen in Table 1-2.

If the external $\overline{\text{RCON}}$ pin is asserted (SW4-1 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 not asserted (SW4-1 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. SW4-1 $\overline{\text{RCON}}$

SW4–1	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. SW4-2 JTAG_EN

SW4–2	JTAG Enable
OFF	JTAG interface enabled
ON	BDM interface enabled

Table 1-6. SW4-[4:3] Encoded Clock Mode

SW4–3	SW4–4	Clock Mode
OFF	OFF	External clock mode (PLL disabled)
OFF	ON	1:1 PLL
ON	OFF	Normal PLL mode with external clock reference
ON	ON	Normal PLL mode w/crystal oscillator reference

Table 1-7. SW4-5 Chip Configuration Mode

SW4–5	RCON (SW4–1)	Mode
OFF	ON	Reserved
ON	ON	Master
X	OFF	Master

Table 1-8. SW4-[7:6] Boot Device

SW4–6	SW4–7	RCON (SW4–1)	Boot Device
OFF	OFF	ON	External (32-bit)
OFF	ON	ON	External (8-bit)
ON	OFF	ON	External (16-bit)
ON	ON	ON	External (32-bit)
X	X	OFF	External (32-bit)

Table 1-9. SW4-8 Bus Drive Strength

SW4–8	RCON (SW4–1)	Drive Strength
OFF	ON	Partial Bus Drive
ON	ON	Full Bus Drive
X	OFF	Partial Bus Drive

Table 1-10. SW4-[10:9] Address/Chip Select Mode

SW4–9	SW4–10	RCON (SW4–1)	Mode
OFF	OFF	ON	PF[7:5] = /CS[6:4]
OFF	ON	ON	PF[7] = /CS6, PF[6:5] = A[22:21]
ON	OFF	ON	PF[7:6] = /CS[6:5], PF[5] = A21
ON	ON	ON	PF[7:5] = A[23:21]
X	X	OFF	PF[7:5] = A[23:21]

1.3.2 Clock Circuitry

The are three options to provide the clock to the CPU. These options can be configured by setting JP[10:12]. See Table 1-11.

Table 1-11. M5271EVB Clock Source Selection

JP10	JP11	JP12	Clock Selection
1–2	1–2	ON	25MHz Oscillator (default setting)
2–3	1–2	ON	25MHz External Clock
X	2–3	OFF	25MHz Crystal (not populated)

The 25MHz oscillator (U15) also feeds the Ethernet chip (U9).

1.3.3 Watchdog Timer

The dBUG Firmware does **NOT** enable the watchdog timer on the MCF5271.

1.3.4 Exception Sources

The ColdFire® family of processors can receive seven levels of interrupt priorities. When the processor receives an interrupt which has a higher priority than the current interrupt mask (in the status register), it will perform an interrupt acknowledge cycle at the end of the current instruction cycle. This interrupt acknowledge cycle indicates to the source of the interrupt that the request is being acknowledged and the device should provide the proper vector number to indicate where the service routine for this interrupt level is located. If the source of interrupt is not capable of providing a vector, its interrupt should be set up as an autovector interrupt which directs the processor to a predefined entry in the exception table (refer to the MCF5271 User's Manual).

The processor goes to an exception routine via the exception table. This table is stored in the Flash EEPROM. The address of the table location is stored in the VBR. The dBUG ROM monitor writes a copy of the exception table into the RAM starting at $00000000. To set an exception vector, the user places the address of the exception handler in the appropriate vector in the vector table located at $00000000 and then points the VBR to $00000000.

The MCF5271 microprocessor has seven external interrupt request lines $\overline{\text{IRQ}}$[7:1]. The interrupt controller is capable of providing up to 63 interrupt sources. These sources are:

- External interrupt signals $\overline{\text{IRQ}}$[7:1] (EPORT)
- Software watchdog timer module
- Timer modules
- UART modules 0, 1 and 2
- I^2C module
- DMA module
- QSPI module
- FEC module
- PIT
- Security module

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

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

The M5271EVB hardware uses $\overline{IRQ7}$ to support the ABORT function using the ABORT switch (SW2). This switch is used to force an interrupt (level 7, priority 3) if the user's program execution should be aborted without issuing a RESET (refer to Chapter 2 for more information on ABORT). Since the ABORT switch is not capable of generating a vector in response to a level seven interrupt acknowledge from the processor, the dBUG programs this interrupt request for autovector mode.

Refer to MCF5271 User's Manual for more information about the interrupt controller.

1.3.5 TA Generation

The processor starts a bus cycle by asserting $\overline{CSx}$ with the other control signals. The processor then waits for a transfer acknowledgment ($\overline{TA}$) either from within (Auto acknowledge - AA mode) or from the externally addressed device before it can complete the bus cycle. -TA is used to indicate the completion of the bus cycle. It also allows devices with different access times to communicate with the processor properly asynchronously. The MCF5271 processor, as part of the chip-select logic, has a built-in mechanism to generate $\overline{TA}$ for all external devices which do not have the capability to generate this signal. For example the Flash ROM cannot generate a $\overline{TA}$ signal. The chip-select logic is programmed by the dBUG ROM Monitor to generate $\overline{TA}$ internally after a pre-programmed number of wait states. In order to support future expansion of the M5271EVB, the $\overline{TA}$ input of the processor is also connected to the Processor Expansion Bus. This allows any expansion boards to assert this line to provide a $\overline{TA}$ signal to the processor. On the expansion boards this signal should be generated through an open collector buffer with no pull-up resistor; a pull-up resistor is included on this board. All $\overline{TA}$ signals from expansion boards should be connected to this line.

1.3.6 User's Program

JP6 on the 16Mbit FLASH (U10) or JP7 if using 32Mbit FLASH (U11) allows users to test code from boot/POR without having to overwrite the ROM Monitor.

When the jumper is set between pins 1 and 2, the behavior of the system is normal, dBUG boots and then runs from 0xFFE00000 (0xFFC00000). When the jumper is set between pins 2 and 3, the board boots from the top half of the FLASH (0xFFF00000).

Procedure:

1. Compile and link as though the code was to be placed at the base of the flash.
2. Set up the jumper JP6 (JP7 for U11) 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 0xFFE00400(0xFFC00400) and run.)
4. In the ROM Monitor, execute the 'FL write <dest> <src> <bytes>' command.

5. Move jumper JP6 (JP7 for U11) to pin 2 connected to pin 3 and push the reset button (SW3). 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^2C port, 10/100T ethernet port, and BDM/JTAG port.

1.4.1 UART0, UART1, UART2 Ports

The MCF5271 device has three built in UARTs, each with its own software programmable baud rate generator. These UART interfaces are brought out to RS232 transceivers. One channel is the ROM Monitor to Terminal output and the other two are available to the user. The ROM Monitor programs the interrupt level for UART0 to Level 3, priority 2 and autovector mode of operation.

Refer to the MCF5271 User's Manual for programming the UART's and their register maps.

1.4.2 10/100T Ethernet Port

The MCF5271 device performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The MCF5271 Ethernet Controller requires an external interface adaptor and transceiver function to complete the interface to the ethernet media. The MCF5271 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 MCF5271 Users 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 MCF5271 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.

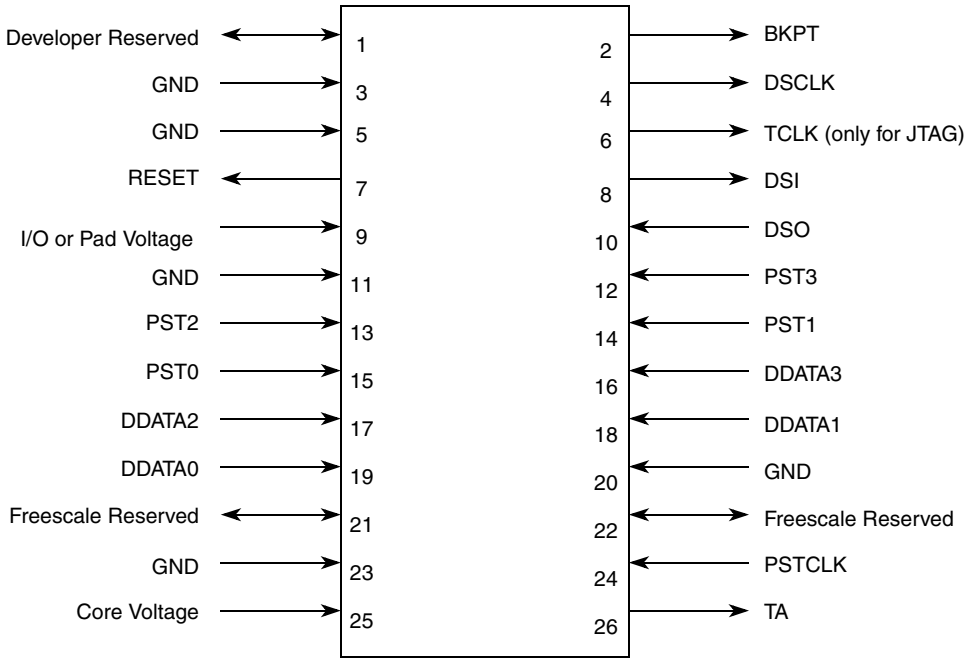


Figure 1-4. J1- BDM Connector Pin Assignment

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 MCF5271'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 MCF5271 User's Manual for more detail. The I^{2}C signals from the MCF5271 device are brought out to expansion connector (J9).

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 MCF5271 User's Manual for more detail. The QSPI signals from the MCF5271 device are brought out to expansion connector (J8).

1.5 Connectors and User Components

1.5.1 Daughter Card Expansion Connectors

Four, 60-way SMT connectors (J7, J8, J9 and J10) provide access to all MCF5271 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

Pin	Signal	Pin	Signal
1	+5V	2	+5V
3	+3.3V	4	+3.3V
5	+3.3V	6	+3.3V
7	GND	8	GND
9	ERXD0	10	NC
11	ETXD1	12	NC
13	ETXD2	14	NC
15	ETXCLK	16	NC
17	ERXER	18	NC
19	ETXEN	20	NC
21	ETXER	22	NC
23	ETXEN	24	GND
25	ETXD0	26	EMDIO
27	U2CTS	28	EMDC
29	I2C_SCL	30	I2C_SDA
31	QSPI_SCK	32	QSPI_DIN
33	BS3	34	QSPI_DOUT
35	BS2	36	QSPI_CS0
37	BS1	38	SD_SCKE
39	BS0	40	NC
41	U2RTS	42	U2RXD
43	QSPI_PCS1	44	U1CTS
45	U1RTS	46	NC
47	U1RXD	48	U2TXD
49	U1TXD	50	CS2
51	CS3	52	CS7
53	CS6	54	CS5
55	CS1	56	CS0
57	CS4	58	A23
59	GND	60	GND

Table 1-13. J4

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+3.3V	4	+3.3V
5	$\overline{NC}$	6	$\overline{NC}$
7	$\overline{NC}$	8	$\overline{NC}$
9	$\overline{ERXD1}$	10	$\overline{NC}$
11	$\overline{ERXD3}$	12	$\overline{NC}$
13	$\overline{ERXD2}$	14	$\overline{NC}$
15	$\overline{ERXCLK}$	16	$\overline{NC}$
17	$\overline{ERXDV}$	18	$\overline{NC}$
19	ECOL	20	$\overline{NC}$
21	ECRS	22	GND
23	GND	24	U0CTS
25	U0RXD	26	DTOUT0
27	DTIN0	28	U0TXD
29	U0RTS	30	GND
31	CLKMOD0	32	+3.3V
33	CLKMOD1	34	GND
35	GND	36	D28
37	D30	38	D29
39	D31	40	$\overline{D24}$
41	D26	42	D25
43	D27	44	D21
45	D23	46	D22
47	EXT_RSTIN	48	D19
49	GND	50	GND
51	D13	52	D20
53	D9	54	D17
55	D12	56	D18
57	D15	58	D16
59	GND	60	GND

Table 1-14. J5

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+3.3V	4	+3.3V
5	$\overline{+3.3V}$	6	+3.3V
7	$\overline{GND}$	8	GND
9	$\overline{A21}$	10	A22
11	$\overline{A19}$	12	A20
13	$\overline{A17}$	14	A18
15	$\overline{A16}$	16	A14
17	$\overline{A15}$	18	A11
19	A13	20	GND
21	GND	22	A10
23	A12	24	A8
25	A9	26	A7
27	A6	28	A4
29	A5	30	GND
31	A2	32	A0
33	A3	34	A1
35	GND	36	GND
37	DTIN3	38	NC
39	DTOUT3	40	$\overline{NC}$
41	TIP	42	TEA
43	TS	44	TA
45	NC	46	SD_WE
47	R/$\overline{W}$	48	NC
49	SD_CAS	50	SD_CS0
51	CLKOUT	52	SD_RAS
53	SD_CS1	54	DDATA3
55	XTAL	56	EXTAL
57	GND	58	GND
59	GND	60	GND

Table 1-15. J6

Pin	Signal	Pin	Signal
1	+5V	2	+1.5V
3	+3.3V	4	+3.3V
5	$\overline{D14}$	6	D10
7	$\overline{D11}$	8	D6
9	$\overline{D7}$	10	D8
11	$\overline{D5}$	12	D4
13	$\overline{GND}$	14	GND
15	$\overline{D1}$	16	D2
17	$\overline{D3}$	18	OE
19	D0	20	DTOUT1
21	DTIN1	22	+3.3V
23	+3.3V	24	IRQ6
25	IRQ7	26	TSIZ0
27	TSIZ1	28	IRQ2
29	IRQ3	30	IRQ4
31	IRQ5	32	TCLK/PSTCLK
33	DTOUT2	34	DTIN2
35	IRQ1	36	TDI/DSI
37	TDO/DSO	38	TMS/BKPT
39	TRST/DSCLK	40	$\overline{GND}$
41	GND	42	PST3
43	PST1	44	PST2
45	PST0	46	DDATA0
47	DDATA2	48	DDATA1
49	GND	50	GND
51	JTAG_EN	52	RCON
53	GND	54	RSTOUT
55	GND	56	RESET
57	GND	58	GND
59	GND	60	GND

1.5.2 Reset Switch (SW3)

The reset logic provides system initilization. Reset occurs during power-on or via assertion of the signal $\overline{\text{RESET}}$ which causes the MCF5271 to perform a hardare reset. Reset is also triggered by the reset switch (SW3) which resets the entire processor/system.

A hard reset and voltage sense controller (U17) is used to produce an active low power-on RESET signal. The reset switch SW3 is fed into U17 which generates the signal which is fed to the MCF5271 reset, $\overline{\text{RESET}}$. The $\overline{\text{RESET}}$ signal is an open collector signal and so can be wire OR'ed with other reset signals from additional peripherals. On the EVB, $\overline{\text{RESET}}$ is wire OR'd with the BDM reset signal and there is a reset signal brought out to the expansion connectors for use with user hardware.

dBUG configures the MCF5271 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 $00000000 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.Table 1-16 details which MCF5271 signal is associated with which LED.

Table 1-16. User LEDs

LED	MCF5271 Signal	Jumper to Disconnect
D17	DTOUT0	JP13
D18	DTIN0	JP14
D19	DTOUT1	JP15
D20	DTIN1	JP16
D21	DTOUT2	JP17
D22	DTIN2	JP18
D23	DTOUT3	JP19
D24	DTIN3	JP20

1.5.4 Other LEDs

There are several other LED's on the M5271EVB 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
D6	+3.3V Power Good
D9	+5V Power Good
D13	+1.5V Power Good (NOTE:1.5V is not enough to turn this LED on, ignore this LED)
D15	Abort ($\overline{IRQ7}$) asserted
D16	Reset ($\overline{RSTI}$) asserted
D17–D24	User LEDs (see Table 1-16)

Chapter 2
Initialization and Setup

2.1 System Configuration

The M5271EVB board requires the following items for minimum system configuration:

- The M5271EVB board (provided).
- Power supply, +6V to 14V DC with minimum of 300 mA.
- RS232C compatible terminal or a PC with terminal emulation software.
- RS232 Communication cable (provided).

Figure 2-1 displays the minimum system configuration.

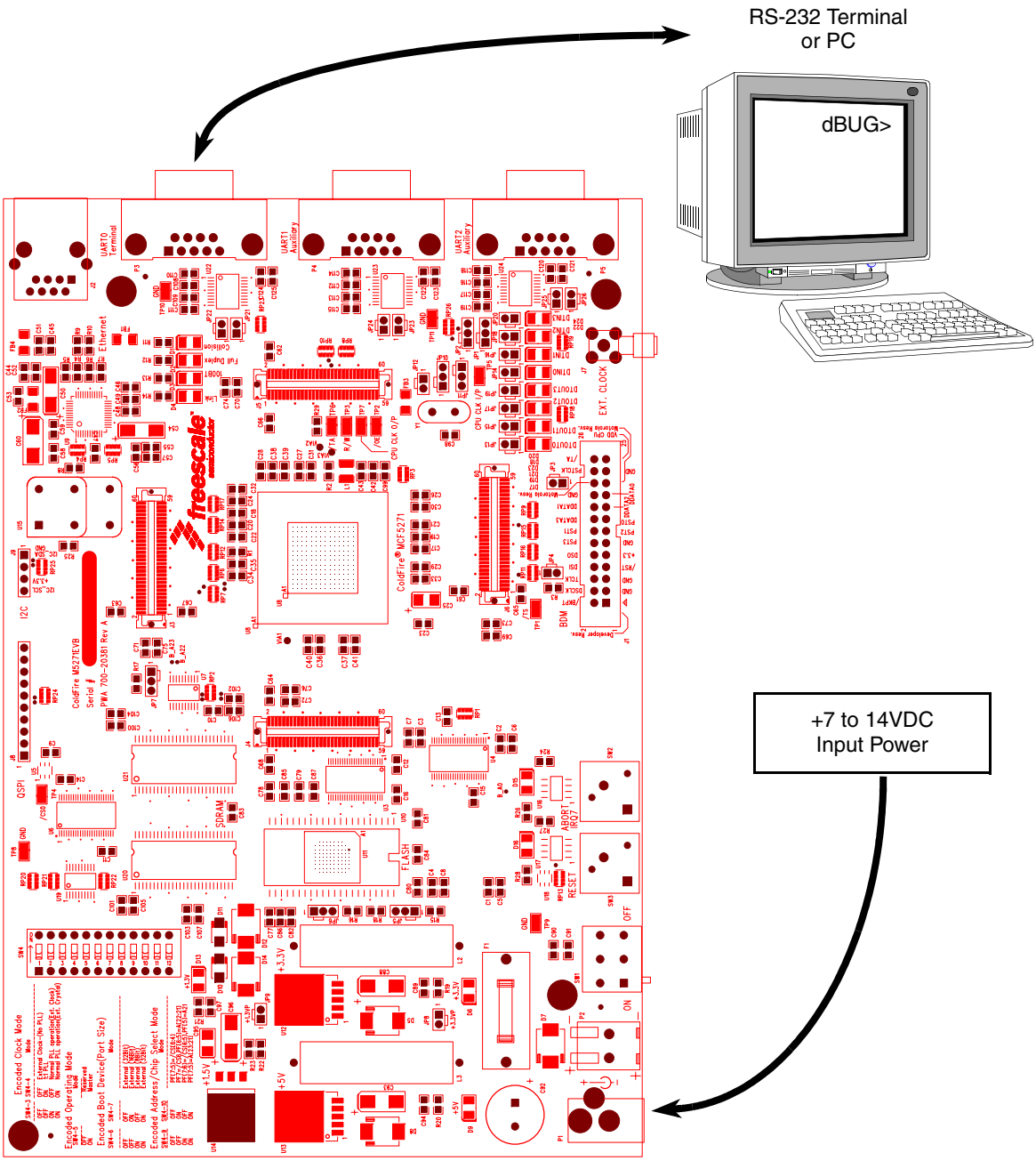


Figure 2-1. Minimum System Configuration

2.2 Installation and Setup

The following sections describe all the steps needed to prepare the board for operation. Please read the following sections carefully before using the board. When you are preparing the board for the first time, be sure to check that all jumpers are in the default locations. Default jumper markings are documented on the master jumper table and printed on the underside of the board. After the board is functional in its default mode, the Ethernet interface may be used by following the instructions provided in Appendix A.

2.2.1 Unpacking

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

- M5271EVB Single Board Computer
- M5271EVB User's Manual (this document)
- One RS232 communication cable
- One BDM (Background Debug Mode) “wiggler” cable
- MCF5271UM ColdFire Integrated Microprocessor User Manual
- ColdFire® Programmers Reference Manual
- A selection of Third Party Developer Tools and Literature

NOTE

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

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

2.2.2 Preparing the Board for Use

The board, as shipped, is ready to be connected to a terminal and power supply without any need for modification.Figure 2-5 shows the position of the jumpers and connectors.

2.2.3 Providing Power to the Board

The EVB requires an external supply voltage of 6-14V DC, minimum 300 mA. This is regulated on board using three switching voltage regulators to provide the necessary EVB voltages of 5V, 3.3V and 1.5V. There are two different power supply input connectors on the EVB. Connector P1 is a 2.1mm power jack (Figure 2-2), P3 a lever actuated connector (Figure 2-3).

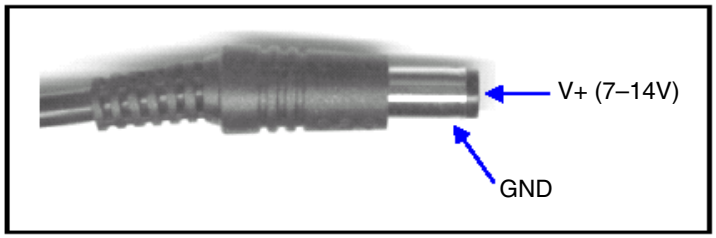


Figure 2-2. 2.1mm Power Connector

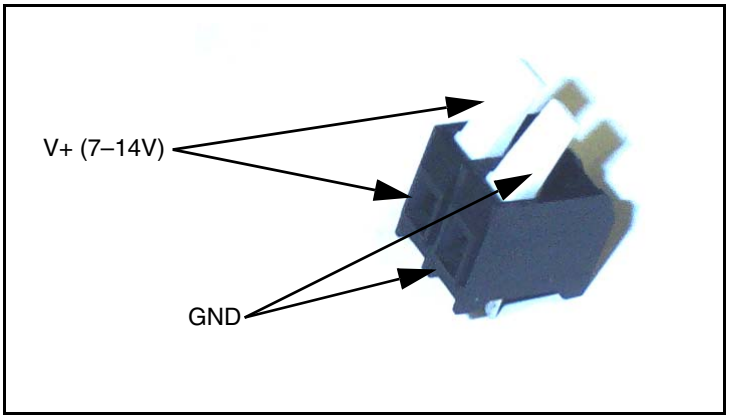


Figure 2-3. 2-Lever Power Connector

2.2.4 Power Switch (SW1)

Slide switch SW1 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
D9	Indicates that the +5V regulator is working correctly
D6	Indicates that the +3.3V regulator is working correctly
D13	Indicates that the +1.5V regulator is working correctly (this LED will not light up with only 1.5V, normal operation is to have this LED off)

If no LEDs are illuminated when the power is applied to the EVB, it is possible that either power switch SW4 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 1A fast blow fuse.

2.2.6 Selecting Terminal Baud Rate

The serial channel UART0 of the MCF5271 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 M5271EVB 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 M5271EVB 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.

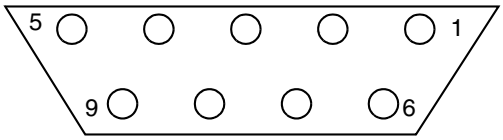


Figure 2-4. Pin Assignment for Female (Terminal) Connector

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.

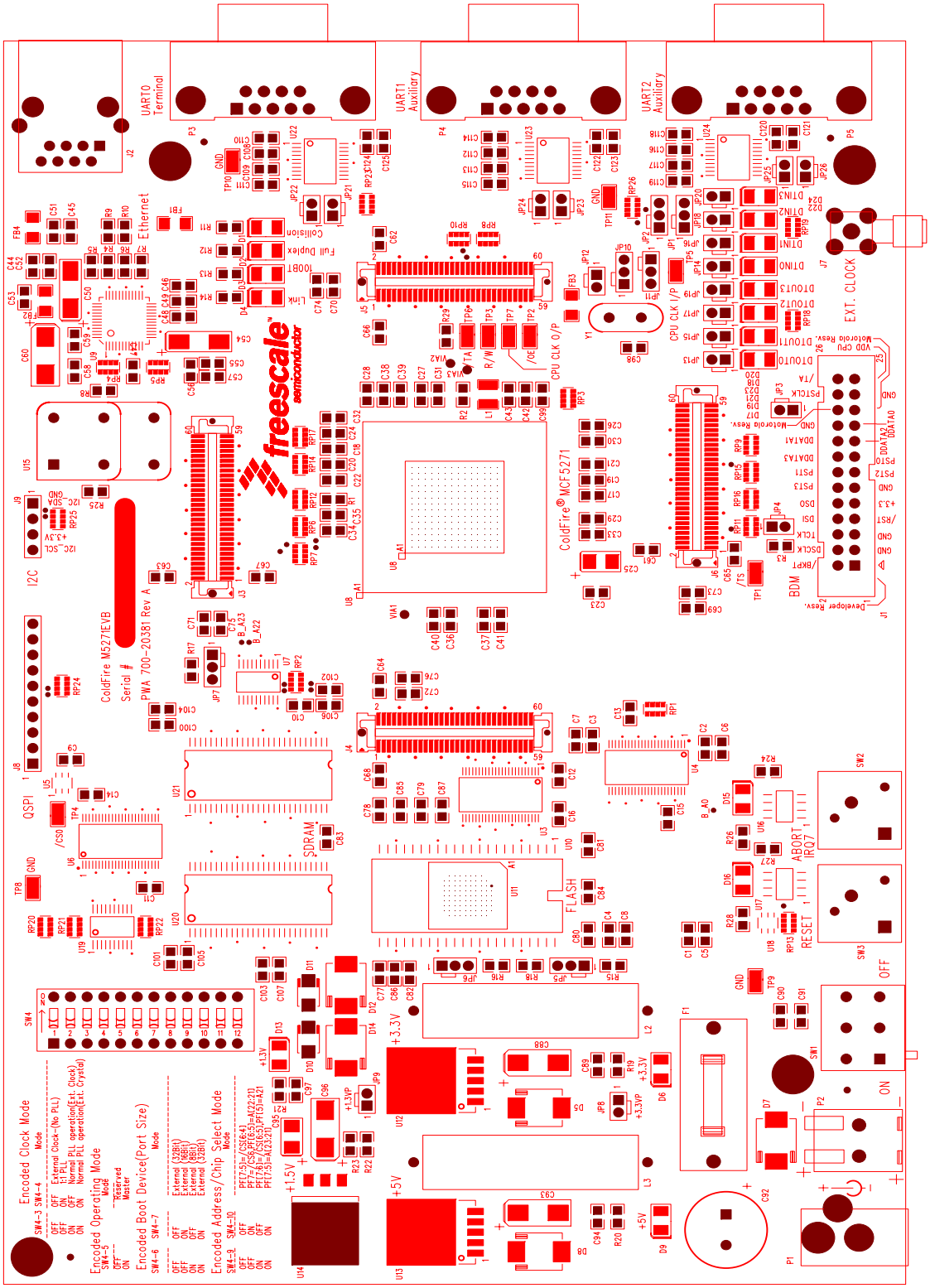


Figure 2-5. Jumper Locations

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 Freescale, Inc. All Rights Reserved.
ColdFire MCF5271 EVS Firmware v2e.1a.xx (Build XXX on XXX  XX 20XX
xx:xx:xx)
Enter 'help' for help.

dBUG>
```

The board is now ready for operation under the control of the debugger as described in Chapter 2. If you do not get the above response, perform the following checks:

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

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

2.4 Using The BDM Port

The MCF5271 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 M5271EVB 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.

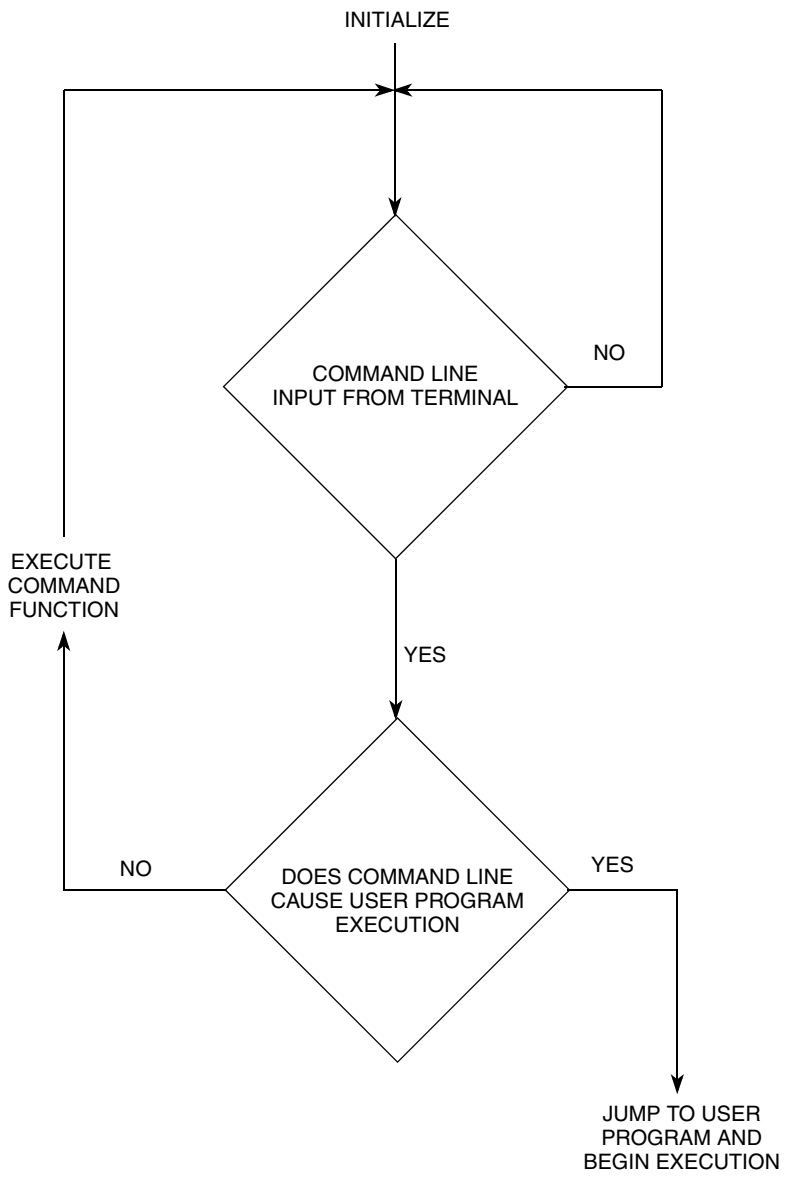


Figure 3-1. Flow Diagram of dBUG Operational Mode

3.2.2 System Initialization

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

```
Hard Reset
DRAM Size: 16M

ColdFire MCF5271 on the M5271EVB
Firmware vXX.XX.XX (Build X on XXXX)
Copyright 1995-2004 Freescale, Inc.  All Rights Reserved.

Enter 'help' for help.

dBUG>
```

Other means can be used to re-initialize the M5271EVB firmware. These means are discussed in the following paragraphs.

3.2.2.1 External RESET Button

External RESET (SW3) is the red button. Depressing this button causes all processes to terminate, resets the MCF5271 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 (SW2) is the button located next to the RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5271) 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 MCF5271 core internal registers are displayed.

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

3.2.2.3 Software Reset Command

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

3.3 Command Line Usage

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

dBUG assumes that an 80x24 ASCII character dumb terminal is used to connect to the debugger. For serial communications, dBUG requires eight data bits, no parity, and one stop bit (8-N-1). The baud rate default

is 19200 bps — a speed commonly available from workstations, personal computers and dedicated terminals.

The command line prompt is: dBUG>

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

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

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

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

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

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

3.4 Commands

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

Table 3-1. dBUG Command Summary

Mnemonic	Syntax	Description
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
BM	bm begin end dest	Block Move
BR	br addr <-r> <-c count> <-t trigger>	Breakpoint
BS	bs <width> begin end data	Block Search
DC	dc value	Data Convert
DI	di<addr>	Disassemble
DL	dl <offset>	Download Serial
DLDBUG	dldbug	Download dBUG

Table 3-1. dBUG Command Summary (continued)

Mnemonic	Syntax	Description
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
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> -Cllls>	Symbol Management
TRACE	trace <num>	Trace (Into)
UP	up begin end filename	Upload Memory to File
VERSION	version	Show Version

ASM Assembler ASM

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 BC

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 BF

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 BM

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 BR

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 BS

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 DC

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 DI

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 DL

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 DLDBUG

Usage: DL <offset>

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

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

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

DN Download Network DN

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 FL

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 GO

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 GT

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 IRD

Usage: IRD <module.register>

This command displays the internal registers of different modules inside the MCF5271. 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 MCF5271 user's manual for more information on these modules and the registers they contain.

Example:

```
ird                 sim.rsr
```

IRM Internal Register Modify IRM

Usage: IRM module.register data

This command modifies the contents of the internal registers of different modules inside the MCF5271. 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 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 LR

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 LW

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 MD

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 MM

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 MMAP

Usage: mmap

This command displays the memory map information for the M5271EVB evaluation board. The information displayed includes the type of memory, the start and end address of the memory, and the port size of the memory. The display also includes information on how the Chip-selects are used on the board and which regions of memory are reserved for dBUG use (protected).

Here is an example of the output from this command:

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

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 RM

Usage: RM reg data

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

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

Examples:

To change register D0 on MC68000 and ColdFire to contain the value 0x1234, the command is:

```
rm                D0 1234
```

RESET Reset the Board and dBUG RESET

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 SD

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 SET

Usage: SET <option value>

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

baud — This is the baud rate for the first serial port on the board. All communications between dBUG and the user occur using either 9600 or 19200 bps, eight data bits, no parity, and one stop bit, 8-N-1, with no flow control.

base — This is the default radix for use in converting a number from its ASCII text representation to the internal quantity used by dBUG. The default is hexadecimal (base 16), and other choices are binary (base 2), octal (base 8), and decimal (base 10).

client — This is the network Internet Protocol (IP) address of the board. For network communications, the client IP is required to be set to a unique value, usually assigned by your local network administrator.

server — This is the network IP address of the machine which contains files accessible via TFTP. Your local network administrator will have this information and can assist in properly configuring a TFTP server if one does not exist.

gateway — This is the network IP address of the gateway for your local subnetwork. If the client IP address and server IP address are not on the same subnetwork, then this option must be properly set. Your local network administrator will have this information.

netmask — This is the network address mask to determine if use of a gateway is required. This field must be properly set. Your local network administrator will have this information.

filename — This is the default filename to be used for network download if no name is provided to the DN command.

filetype — This is the default file type to be used for network download if no type is provided to the DN command. Valid values are: “srecord”, “coff”, and “elf”.

mac — This is the ethernet Media Access Control (MAC) address (a.k.a hardware address) for the evaluation board. This should be set to a unique value, and the most significant nibble should always be even.

Examples:

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

```
set                 baud 19200
```

NOTE

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

SHOW Show Configurations SHOW

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 STEP

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 SYMBOL

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 TRACE

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 UP

Usage: UP begin end filename

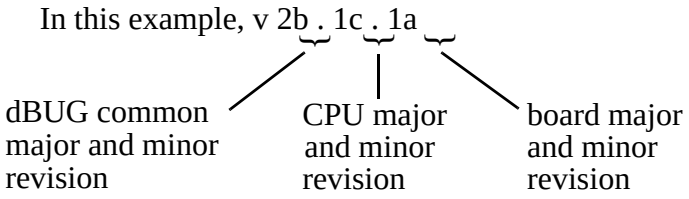
The UP command uploads the data from a memory region (specified by begin and end) to a file (specified by filename) over the network. The file created contains the raw binary data from the specified memory region. The UP command uses the Trivial File Transfer Protocol (TFTP) to transfer files to a network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

VERSION Display dBUG Version VERSION

Usage: VERSION

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

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



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

Examples:

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

```
version
```

3.5 TRAP #15 Functions

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

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

3.5.1 OUT_CHAR

This function (function code 0x0013) sends a character, which is in 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.l     8(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.l    4(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.l     d1,d0");             /* put the character in d0 */
}
```

3.5.3 CHAR_PRESENT

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

Assembly example:

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

C example:

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

3.5.4 EXIT_TO_dBUG

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

Assembly example:

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

C example:

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

Appendix A
Configuring dBUG for Network Downloads

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

A.1 Required Network Parameters

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

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

Internet Protocol, IP, address for the computer (client IP),

IP address of the Gateway for non-local traffic (gateway IP), and

Network netmask for flagging traffic as local or non-local (netmask).

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

IP address of the TFTP server (server IP),

Name of the file to download (filename),

Type of the file to download (filetype of S-record, COFF, ELF, or Image).

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

Client IP: ___.___.___.___(IP address of the board)

Server IP: ___.___.___.___(IP address of the TFTP server)

Gateway: ___.___.___.___(IP address of the gateway)

Netmask: ___.___.___.___(Network netmask)

A.2 Configuring dBUG Network Parameters

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

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

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

```
set client 123.45.68.15
set server 123.45.67.1
set gateway 123.45.68.250
set netmask 255.255.255.0
set mac 00:CF:52:82:EB:01
```

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

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

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

```
set filename <filename>
```

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

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

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

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

Change the default filename and filetype with the commands:

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

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

A.3 Troubleshooting Network Problems

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

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

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

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

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

Appendix B
Schematics

B.1 MCF5271EVM Schematics

M5271EVB Evaluation Board

Table Of Contents:

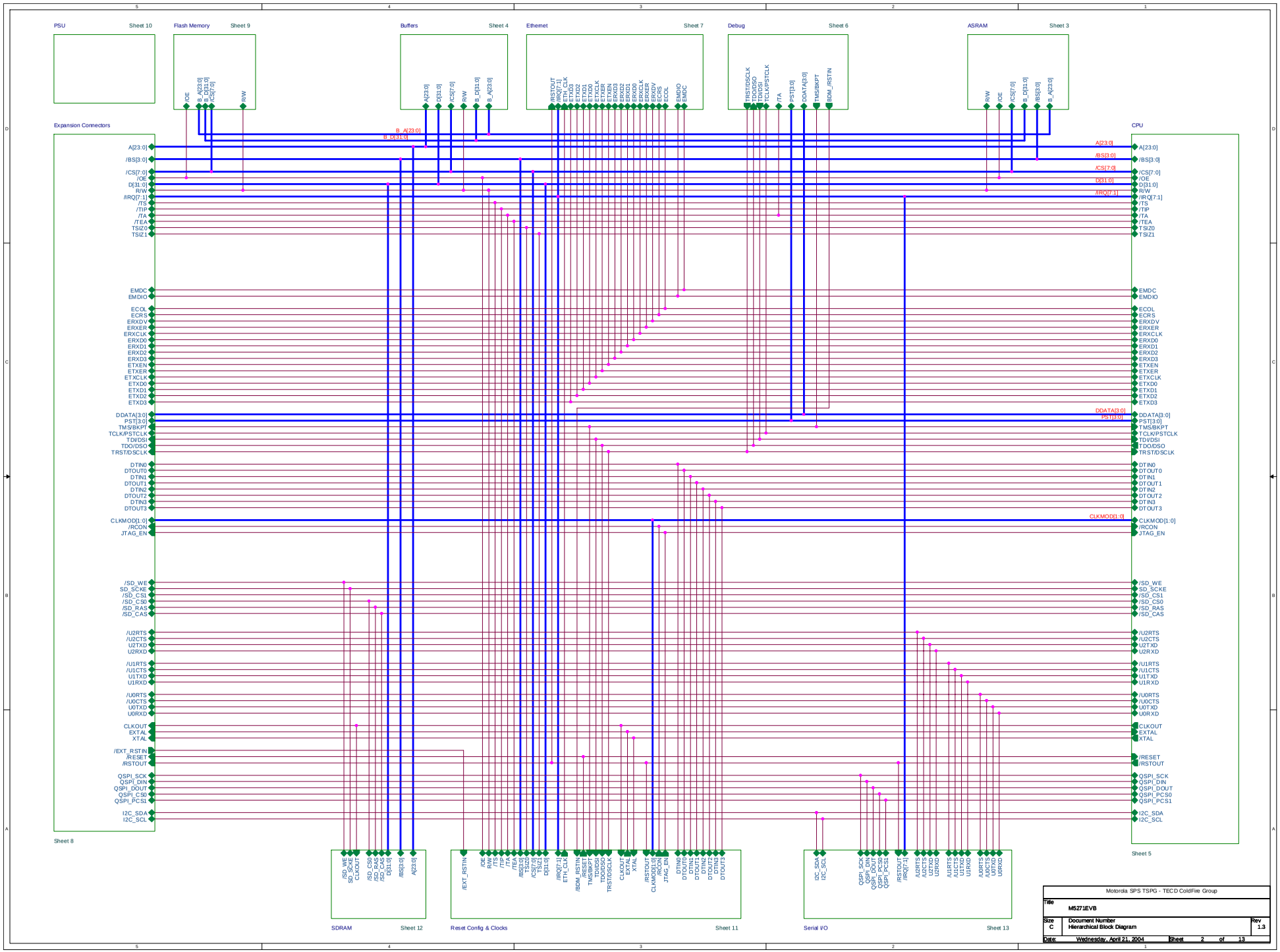
HIERARCHICAL INTERCONNECTS	SHEET 2
ASRAM MEMORY	SHEET 3
ADDRESS AND DATA BUS BUFFERS	SHEET 4
MCF5271 CPU	SHEET 5
DEBUG	SHEET 6
ETHERNET INTERFACE	SHEET 7
EXPANSION CONNECTORS	SHEET 8
FLASH MEMORY	SHEET 9
POWER SUPPLY UNIT	SHEET 10
RESET CONFIGURATION AND CLOCKING CIRCUITRY	SHEET 11
SDRAM MEMORY	SHEET 12
SERIAL I/O INTERFACES	SHEET 13

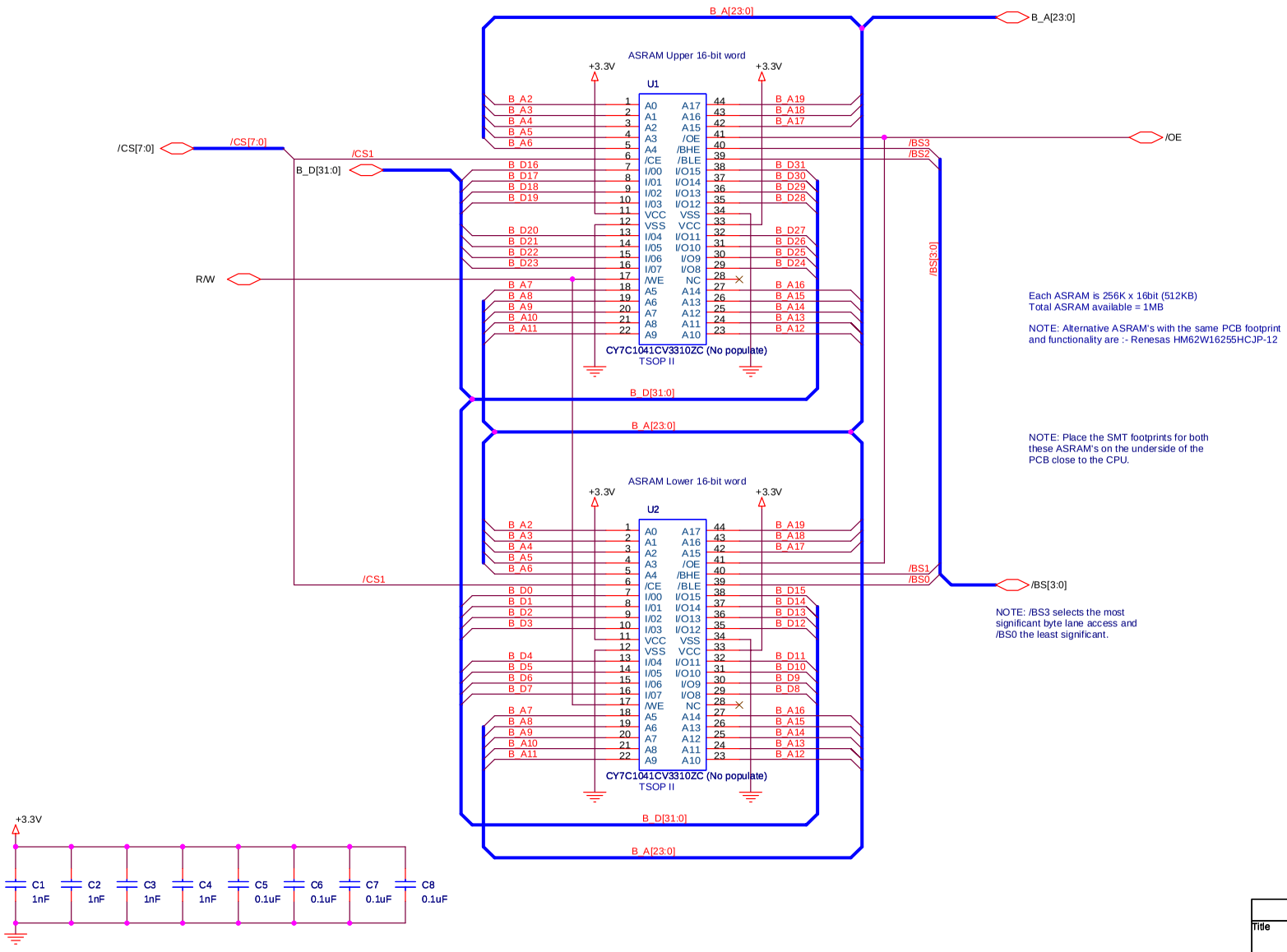
Revision Information

Rev	Date	Designer	Comments
1.0	4 Mar 04	PBH	Provisional release
1.1	12 Mar 04	PBH	Removed USB controller and placed ASRAM footprints on the underside of the PCB.
1.2	26 Mar 04	PBH	Modified Flash page to incorporate 16Mbit burst Flash & ensured expansion connections were compatible with the M523xEVB.
1.3	21 Apr 04	PBH	Corrected RJ45 pinout

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


PSU
Sheet 10
Flash Memory
Sheet 9
Buffers
Sheet 4
Ethernet
Sheet 7
Debug
Sheet 6
ASRAM
Sheet 3
Expansion Connectors
Sheet 8
CPU
Sheet 5
SDRAM
Sheet 12
Reset Config & Clocks
Sheet 11
Serial I/O
Sheet 13
Motorola SPS TSPG - TECD ColdFire Group
Title
M5271EVB
Size
C
Document Number
Hierarchical Block Diagram
Rev
1.3
Date: Wednesday, April 21, 2004
Sheet 2 of 13



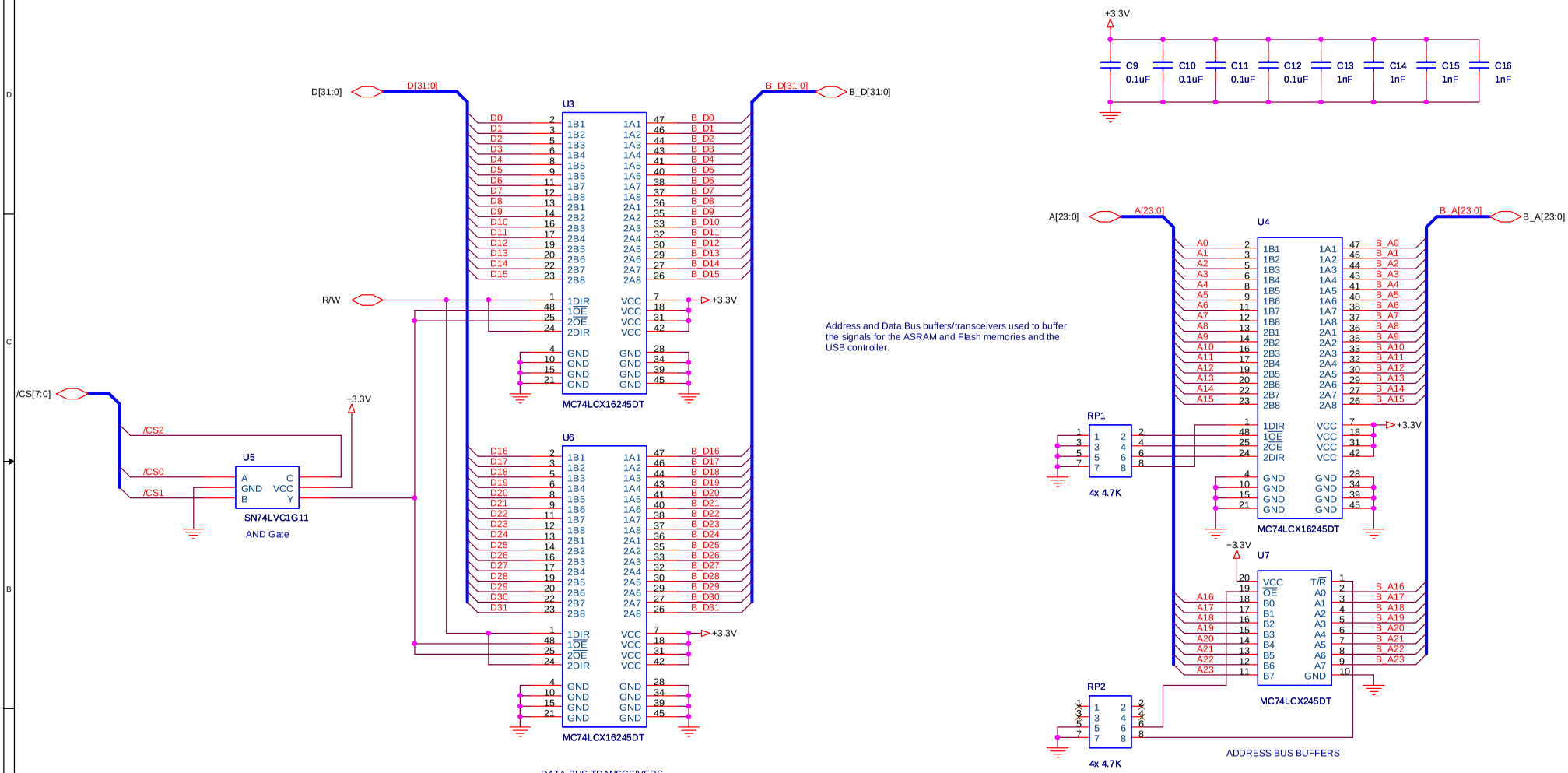
Each ASRAM is 256K x 16bit (512KB)
Total ASRAM available = 1MB

NOTE: Alternative ASRAM's with the same PCB footprint and functionality are :- Renesas HM62W16255HCJP-12

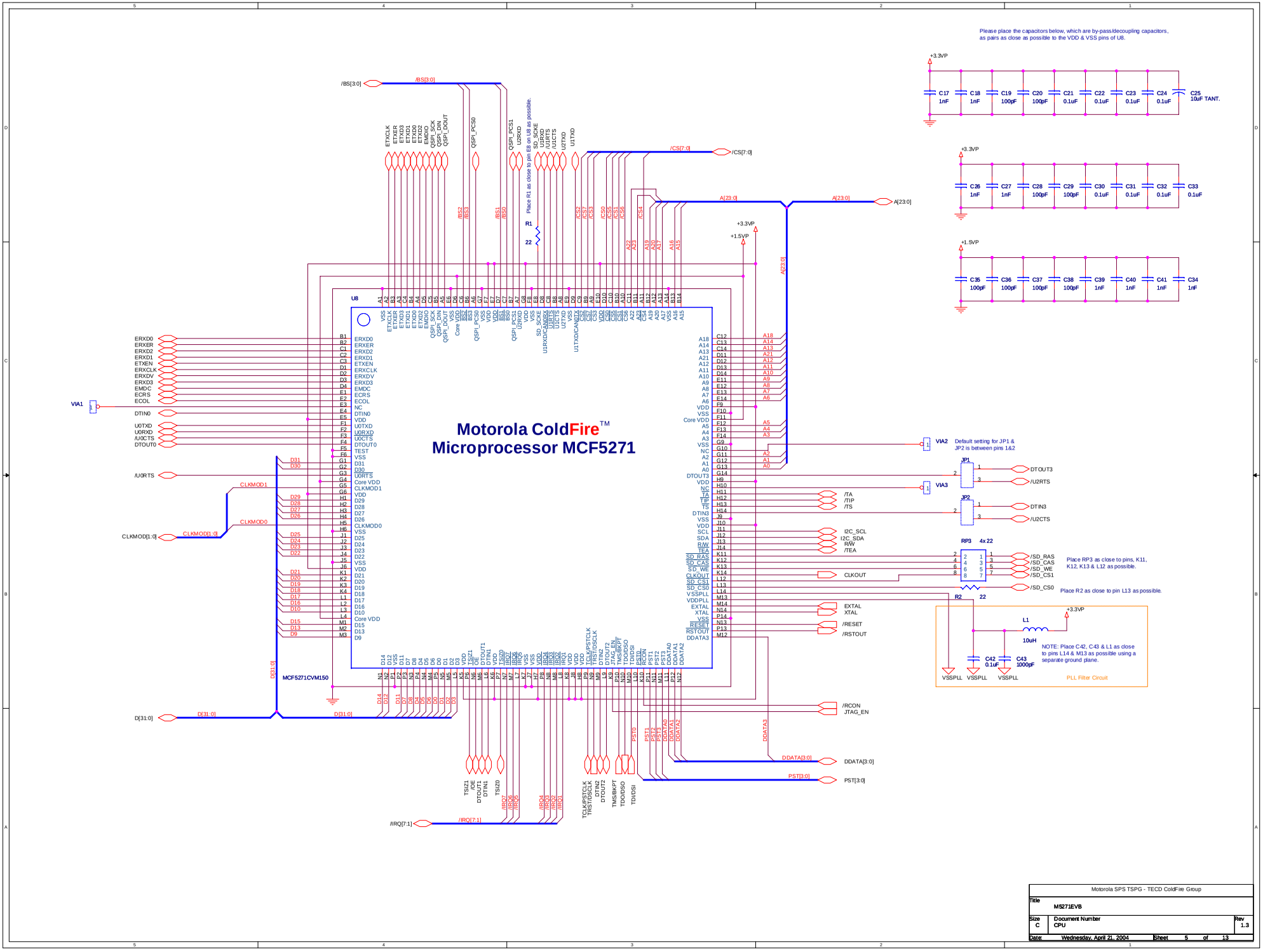
NOTE: Place the SMT footprints for both these ASRAM's on the underside of the PCB close to the CPU.

NOTE: /BS3 selects the most significant byte lane access and /BS0 the least significant.

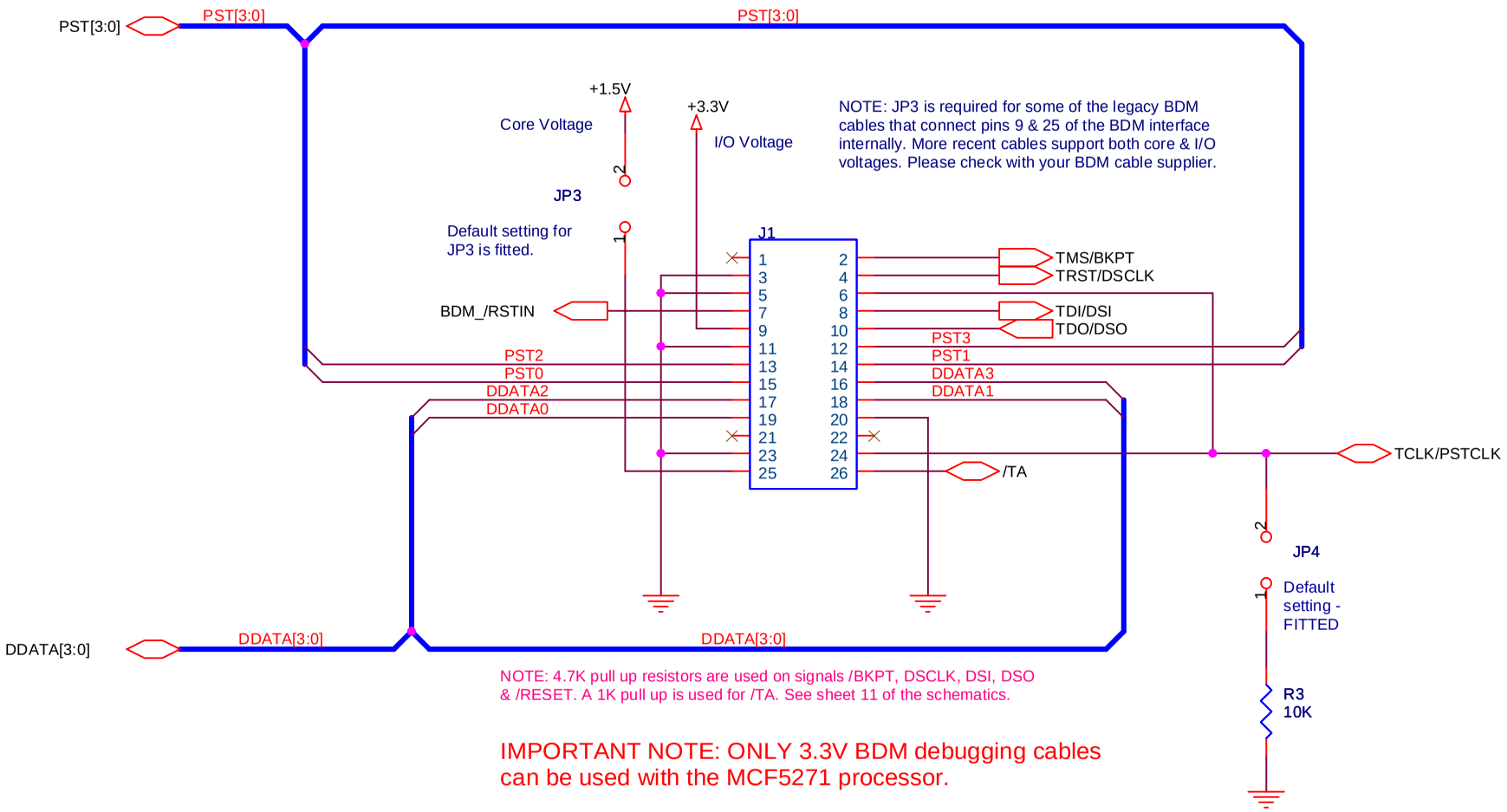
Motorola SPS TSPG - TECD ColdFire Group		
Title M5271EVB		
Size B	Document Number Asynchronous SRAM	Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 3 of 13	



Motorola SPS TSPG - TECD ColdFire Group		
Title M5271EVB		
Size B	Document Number Buffers	Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 4 of 13	



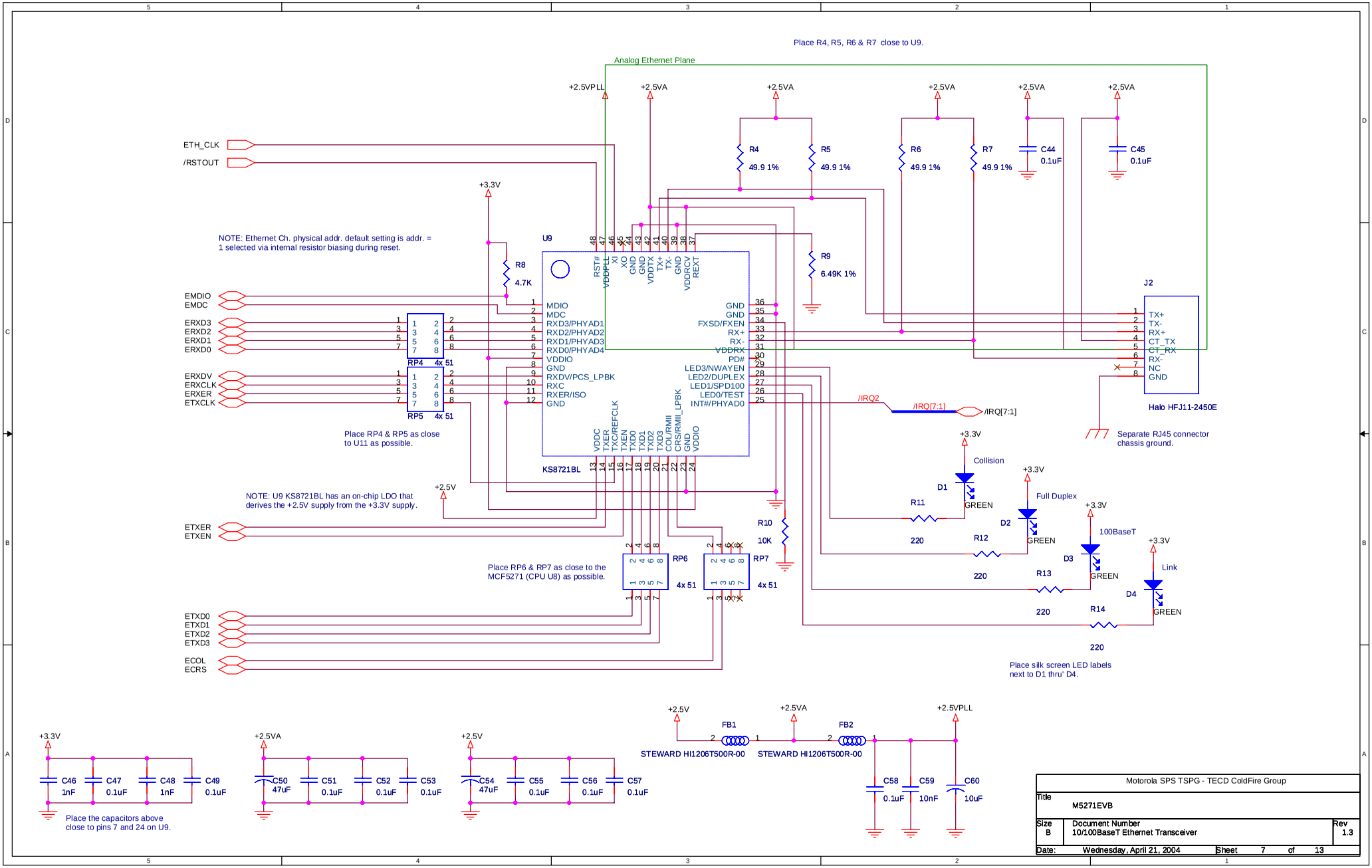
Motorola SPS TSPG - TECD ColdFire Group			
Title	M5271EVB		
Size C	Document Number CPU		Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 5 of 13		

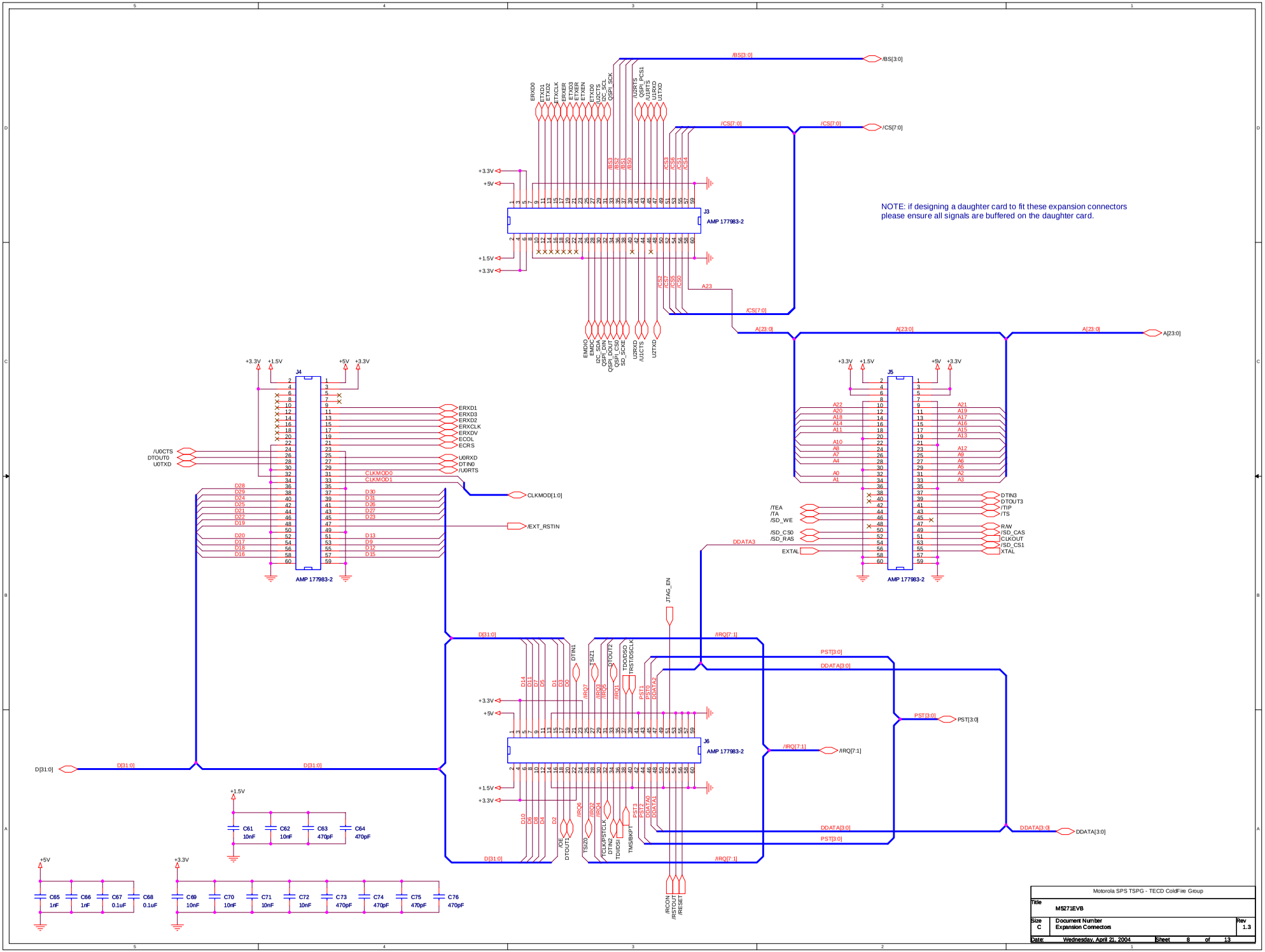


NOTE: 4.7K pull up resistors are used on signals /BKPT, DSCLK, DSI, DSO & /RESET. A 1K pull up is used for /TA. See sheet 11 of the schematics.

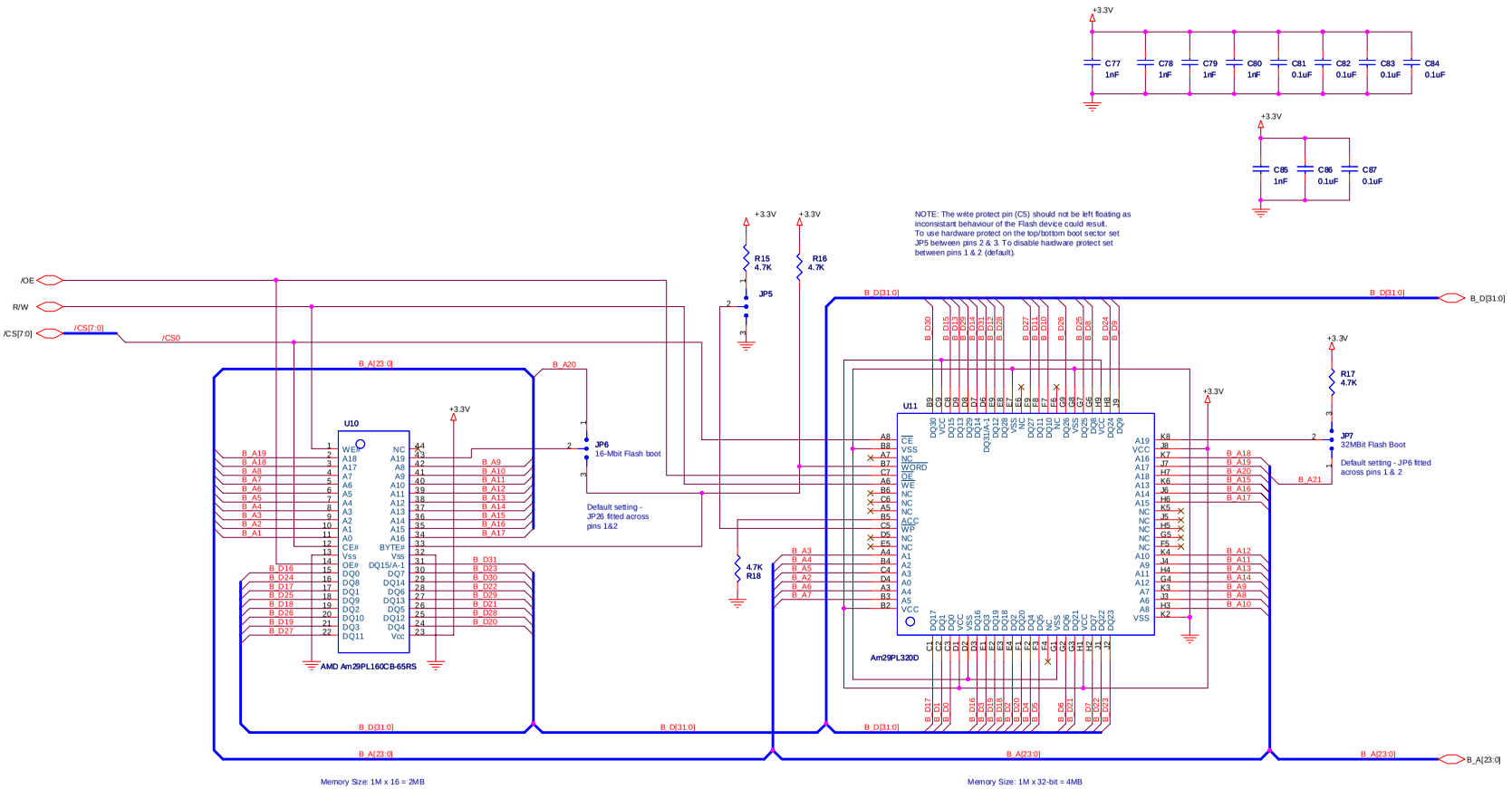
IMPORTANT NOTE: ONLY 3.3V BDM debugging cables can be used with the MCF5271 processor.

Motorola SPS TSPG - TECD ColdFire Group		
Title: M5271EVB		
Size: A	Document Number: BDM/JTAG Debug Port	Rev: 1.3
Date: Wednesday, April 21, 2004	Sheet 6 of 13	


Place R4, R5, R6 & R7 close to U9.
Analog Ethernet Plane
+2.5VPLL
+2.5VA
+2.5VA
+2.5VA
+2.5VA
+2.5VA
ETH_CLK
/RSTOUT
R4
49.9 1%
R5
49.9 1%
R6
49.9 1%
R7
49.9 1%
C44
0.1uF
C45
0.1uF
+3.3V
NOTE: Ethernet Ch. physical addr. default setting is addr. = 1 selected via internal resistor biasing during reset.
U9
R8
4.7K
R9
6.49K 1%
RST#
VDDPLL
XI
XO
GND
GND
VDDTX
TX+
TX-
GND
VDDRCV
REXT
EMDIO
EMDC
ERXD3
ERXD2
ERXD1
ERXD0
RP4
4x 51
ERXDV
ERXCLK
ERXER
ETXCLK
RP5
4x 51
MDIO
MDC
RXD3/PHYAD1
RXD2/PHYAD2
RXD1/PHYAD3
RXD0/PHYAD4
VDDIO
GND
RXDV/PCS_LPBK
RXC
RXER/ISO
GND
GND
GND
FXSD/FXEN
RX+
RX-
VDDRX
PD#
LED3/NWAYEN
LED2/DUPLEX
LED1/SPD100
LED0/TEST
INT#/PHYAD0
VDDC
TXER
TXC/REFCLK
TXEN
TXD0
TXD1
TXD2
TXD3
COL/RMII
CRS/RMII_LPBK
GND
VDDIO
KS8721BL
J2
TX+
TX-
RX+
CT_TX
CT_RX
RX-
NC
GND
Halo HFJ11-2450E
/IRQ2
/IRQ[7:1]
/IRQ[7:1]
Separate RJ45 connector chassis ground.
Place RP4 & RP5 as close to U11 as possible.
+2.5V
NOTE: U9 KS8721BL has an on-chip LDO that derives the +2.5V supply from the +3.3V supply.
+3.3V
Collision
D1
GREEN
R11
220
+3.3V
Full Duplex
D2
GREEN
R12
220
+3.3V
100BaseT
D3
GREEN
R13
220
+3.3V
Link
D4
GREEN
R14
220
Place silk screen LED labels next to D1 thru' D4.
R10
10K
ETXER
ETXEN
RP6
4x 51
RP7
4x 51
Place RP6 & RP7 as close to the MCF5271 (CPU U8) as possible.
ETXD0
ETXD1
ETXD2
ETXD3
ECOL
ECRS
+3.3V
C46
1nF
C47
0.1uF
C48
1nF
C49
0.1uF
Place the capacitors above close to pins 7 and 24 on U9.
+2.5VA
C50
47uF
C51
0.1uF
C52
0.1uF
C53
0.1uF
+2.5V
C54
47uF
C55
0.1uF
C56
0.1uF
C57
0.1uF
+2.5V
FB1
STEWARD HI1206T500R-00
+2.5VA
FB2
STEWARD HI1206T500R-00
+2.5VPLL
C58
0.1uF
C59
10nF
C60
10uF
Motorola SPS TSPG - TECD ColdFire Group
Title
M5271EVB
Size
B
Document Number
10/100BaseT Ethernet Transceiver
Rev
1.3
Date:
Wednesday, April 21, 2004
Sheet
7
of
13



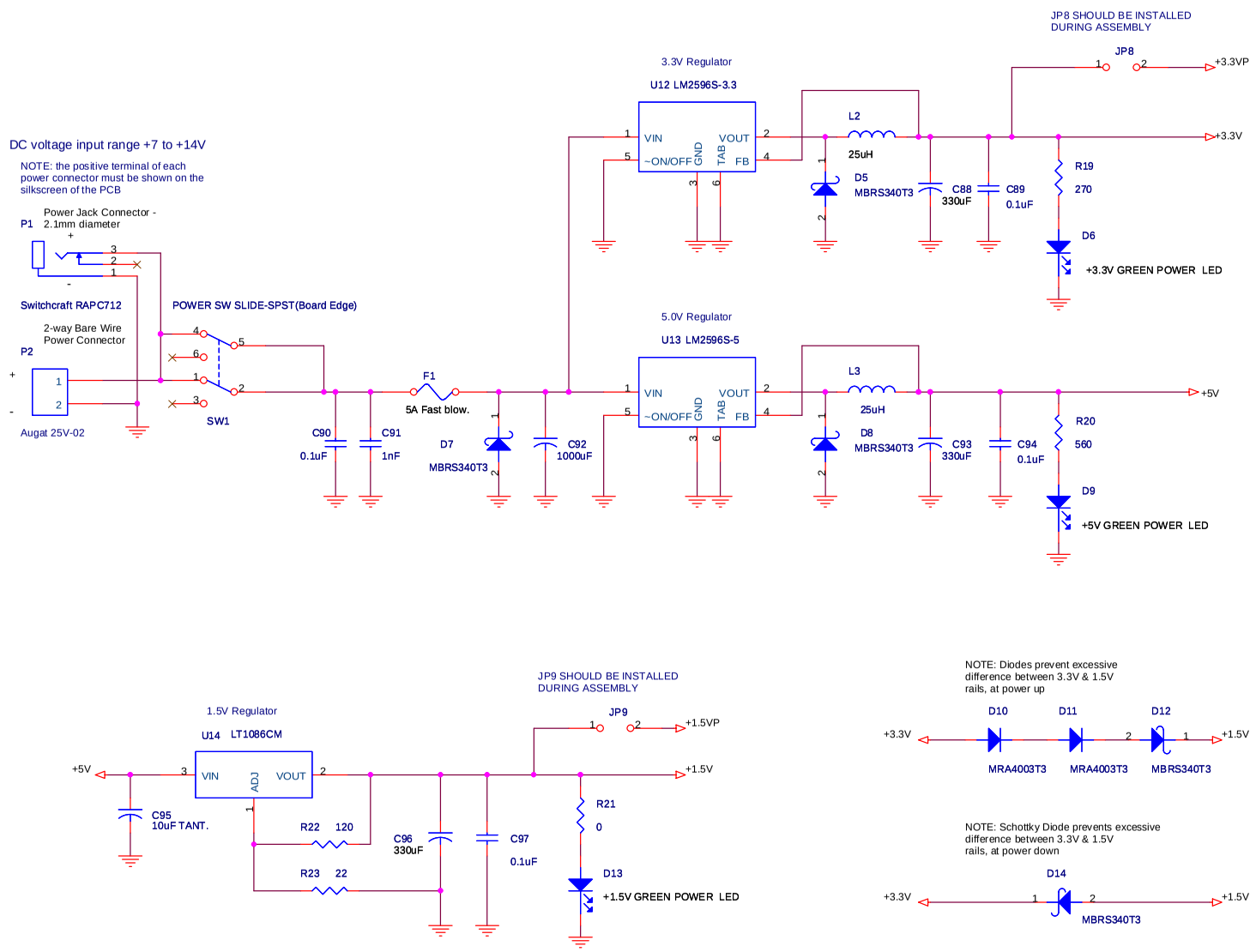
Motorola SPS TSPG - TECD ColdFire Group			
Title	MS271EVB		
Size C	Document Number Expansion Connectos		Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 8 of 13		



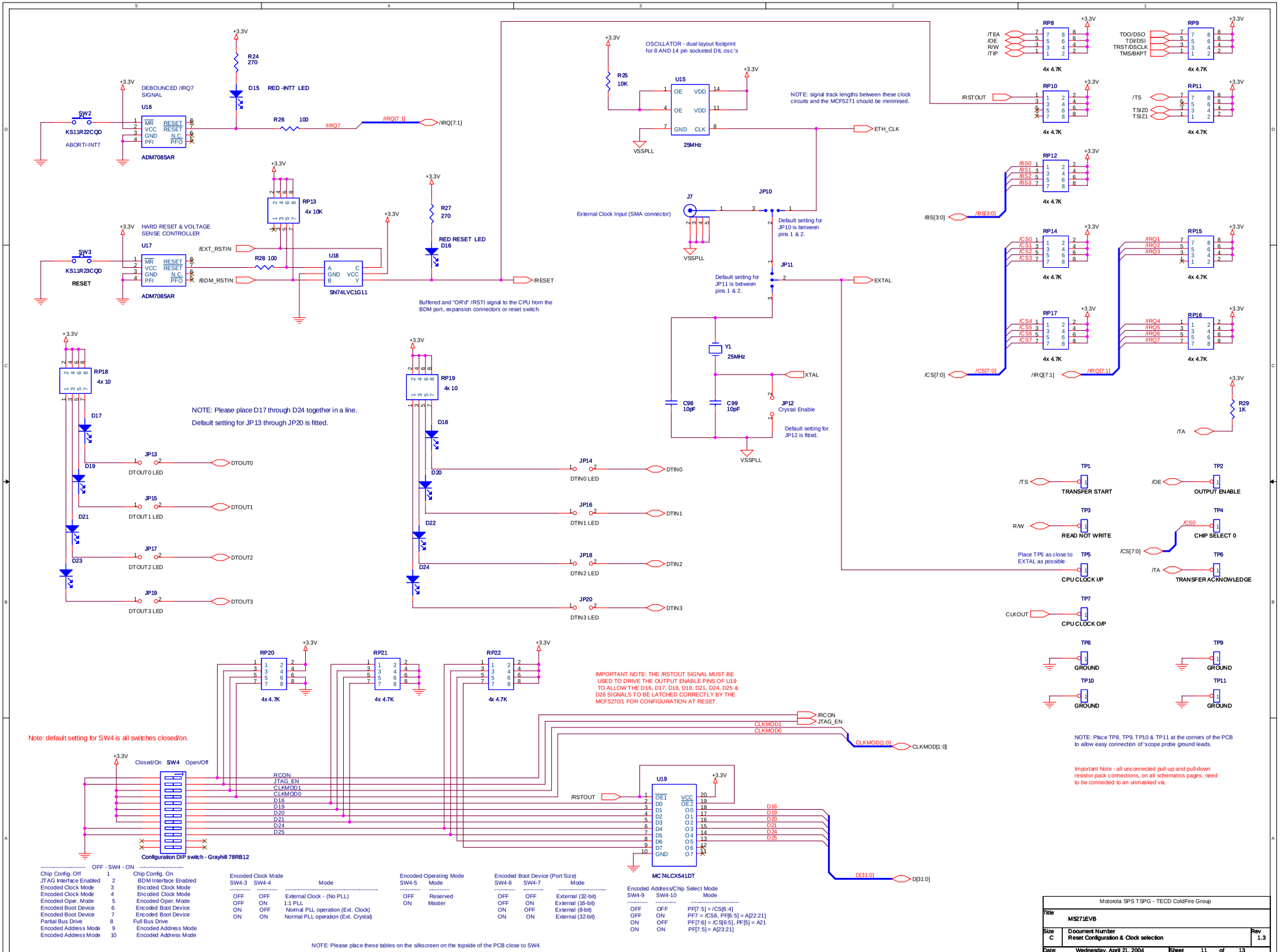
NOTE: only one of the footprints can/will be populated - BGA (U11) OR SSOP (U10)

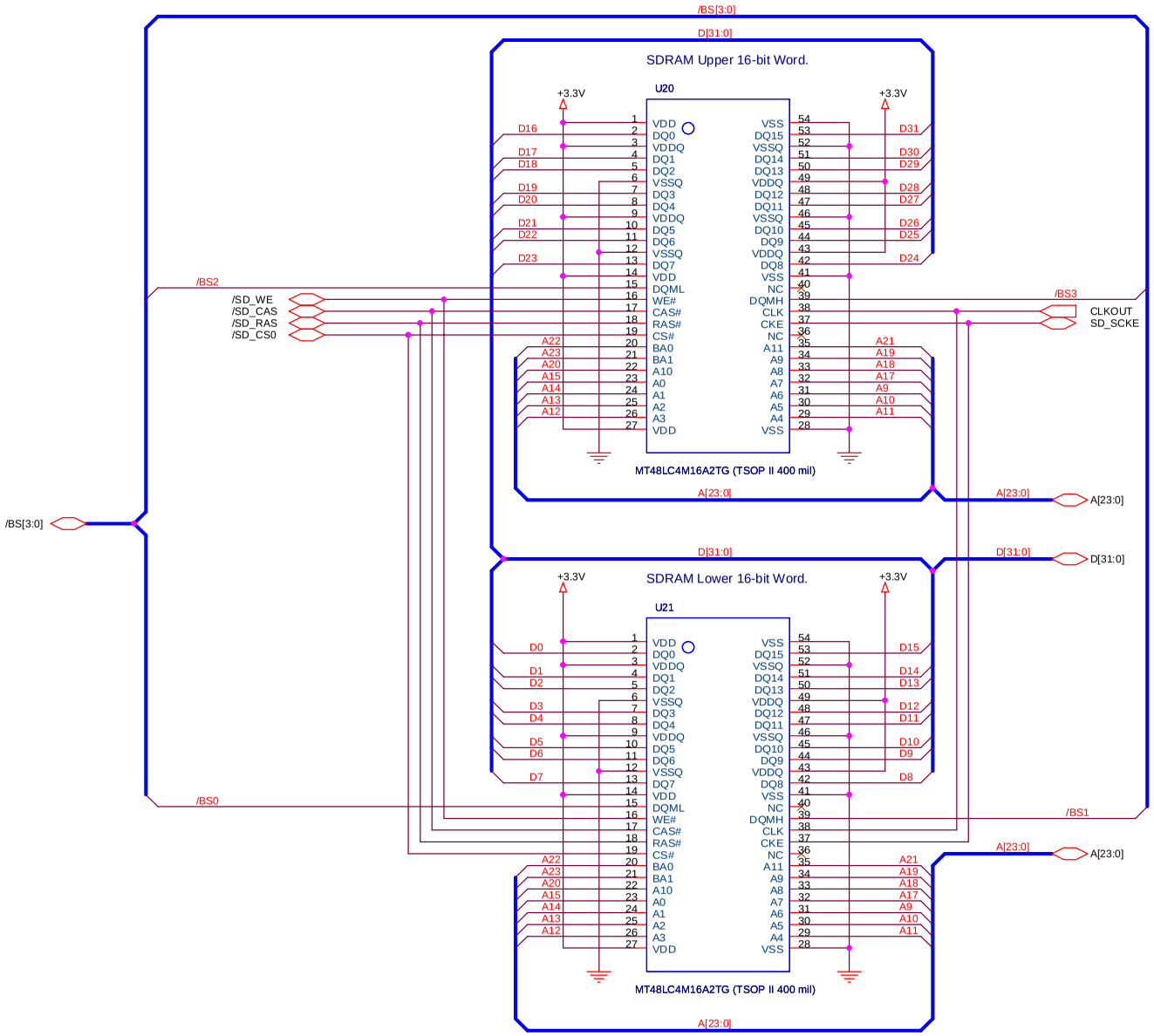
Motorola SPS TSPG - TECD ColdFire Group

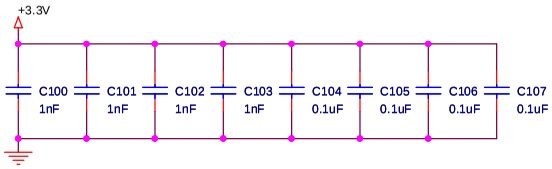
Title	M5271EVB	
Size C	Document Number Flash Memory	Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 9 of 13	



Motorola SPS TSPG - TECD ColdFire Group			
Title	M5271EVB		
Size B	Document Number Power Supply		Rev 1.3
Date: Wednesday, April 21, 2004	Sheet 10 of 13		


DEBOUNCED /IRQ7 SIGNAL
U16
SW2
KS11R22CQD
ABORT/-INT7
ADM708SAR
MR RESET
VCC RESET
GND N.C.
PFI PFO
R24 270
D15 RED -INT7 LED
R26 100
/IRQ7
/IRQ[7:1]
HARD RESET & VOLTAGE SENSE CONTROLLER
U17
SW3
KS11R23CQD
RESET
/EXT_RSTIN
R28 100
/BDM_RSTIN
RP13 4x 10K
U18
SN74LVC1G11
A C
GND VCC
B Y
R27 270
RED RESET LED
D16
/RESET
Buffered and 'OR'd' /RSTI signal to the CPU from the BDM port, expansion connectors or reset switch.
OSCILLATOR - dual layout footprint for 8 AND 14 pin socketed DIL osc.'s
R25 10K
U15
OE VDD
OE VDD
GND CLK
25MHz
VSSPLL
NOTE: signal track lengths between these clock circuits and the MCF5271 should be minimised.
ETH_CLK
External Clock Input (SMA connector)
J7
JP10
Default setting for JP10 is between pins 1 & 2.
JP11
Default setting for JP11 is between pins 1 & 2.
EXTAL
Y1 25MHz
XTAL
C98 10pF
C99 10pF
JP12 Crystal Enable
Default setting for JP12 is fitted.
RP8 4x 4.7K
/TEA
/OE
R/W
/TIP
RP9 4x 4.7K
TDO/DSO
TDI/DSI
TRST/DSCLK
TMS/BKPT
RP10 4x 4.7K
/RSTOUT
RP11 4x 4.7K
/TS
TSIZ0
TSIZ1
RP12 4x 4.7K
/BS0
/BS1
/BS2
/BS3
/BS[3:0]
RP14 4x 4.7K
/CS0
/CS1
/CS2
/CS3
/CS[7:0]
RP15 4x 4.7K
/IRQ1
/IRQ2
/IRQ3
RP17 4x 4.7K
/CS4
/CS5
/CS6
/CS7
RP16 4x 4.7K
/IRQ4
/IRQ5
/IRQ6
/IRQ7
/IRQ[7:1]
R29 1K
/TA
RP18 4x 10
D17
D19
D21
D23
JP13
JP15
JP17
JP19
DTOUT0 LED
DTOUT1 LED
DTOUT2 LED
DTOUT3 LED
DTOUT0
DTOUT1
DTOUT2
DTOUT3
NOTE: Please place D17 through D24 together in a line.
Default setting for JP13 through JP20 is fitted.
RP19 4x 10
D18
D20
D22
D24
JP14
JP16
JP18
JP20
DTIN0 LED
DTIN1 LED
DTIN2 LED
DTIN3 LED
DTIN0
DTIN1
DTIN2
DTIN3
TP1 TRANSFER START
TP2 OUTPUT ENABLE
TP3 READ NOT WRITE
TP4 CHIP SELECT 0
TP5 CPU CLOCK I/P
TP6 TRANSFER ACKNOWLEDGE
TP7 CPU CLOCK O/P
Place TP5 as close to EXTAL as possible
TP8 GROUND
TP9 GROUND
TP10 GROUND
TP11 GROUND
NOTE: Place TP8, TP9, TP10 & TP11 at the corners of the PCB to allow easy connection of 'scope probe ground leads.
Important Note - all unconnected pull-up and pull-down resistor pack connections, on all schematics pages, need to be connected to an unmasked via.
RP20 4x 4.7K
RP21 4x 4.7K
RP22 4x 4.7K
IMPORTANT NOTE: THE /RSTOUT SIGNAL MUST BE USED TO DRIVE THE OUTPUT ENABLE PINS OF U19 TO ALLOW THE D16, D17, D18, D19, D21, D24, D25 & D26 SIGNALS TO BE LATCHED CORRECTLY BY THE MCF5270/1 FOR CONFIGURATION AT RESET.
/RCON
JTAG_EN
CLKMOD1
CLKMOD0
CLKMOD[1:0]
Note: default setting for SW4 is all switches closed/on.
Closed/On SW4 Open/Off
Configuration DIP switch - Grayhill 78RB12
U19
MC74LCX541DT
OE1 VCC
D0 OE2
D1 O0
D2 O1
D3 O2
D4 O3
D5 O4
D6 O5
D7 O6
GND O7
D16
D17
D18
D19
D20
D21
D24
D25
D[31:0]
OFF - SW4 - ON
Chip Config. Off 1 Chip Config. On
JTAG Interface Enabled 2 BDM Interface Enabled
Encoded Clock Mode 3 Encoded Clock Mode
Encoded Clock Mode 4 Encoded Clock Mode
Encoded Oper. Mode 5 Encoded Oper. Mode
Encoded Boot Device 6 Encoded Boot Device
Encoded Boot Device 7 Encoded Boot Device
Partial Bus Drive 8 Full Bus Drive
Encoded Address Mode 9 Encoded Address Mode
Encoded Address Mode 10 Encoded Address Mode
Encoded Clock Mode
SW4-3 SW4-4 Mode
OFF OFF External Clock - (No PLL)
OFF ON 1:1 PLL
ON OFF Normal PLL operation (Ext. Clock)
ON ON Normal PLL operation (Ext. Crystal)
Encoded Operating Mode
SW4-5 Mode
OFF Reserved
ON Master
Encoded Boot Device (Port Size)
SW4-6 SW4-7 Mode
OFF OFF External (32-bit)
OFF ON External (16-bit)
ON OFF External (8-bit)
ON ON External (32-bit)
Encoded Address/Chip Select Mode
SW4-9 SW4-10 Mode
OFF OFF PF[7:5] = /CS[6:4]
OFF ON PF7 = /CS6, PF[6:5] = A[22:21]
ON OFF PF[7:6] = /CS[6:5], PF[5] = A21
ON ON PF[7:5] = A[23:21]
NOTE: Please place these tables on the silkscreen on the topside of the PCB close to SW4.
Motorola SPS TSPG - TECD ColdFire Group
Title
M5271EVB
Size C
Document Number
Reset Configuration & Clock selection
Rev 1.3
Date: Wednesday, April 21, 2004
Sheet 11 of 13



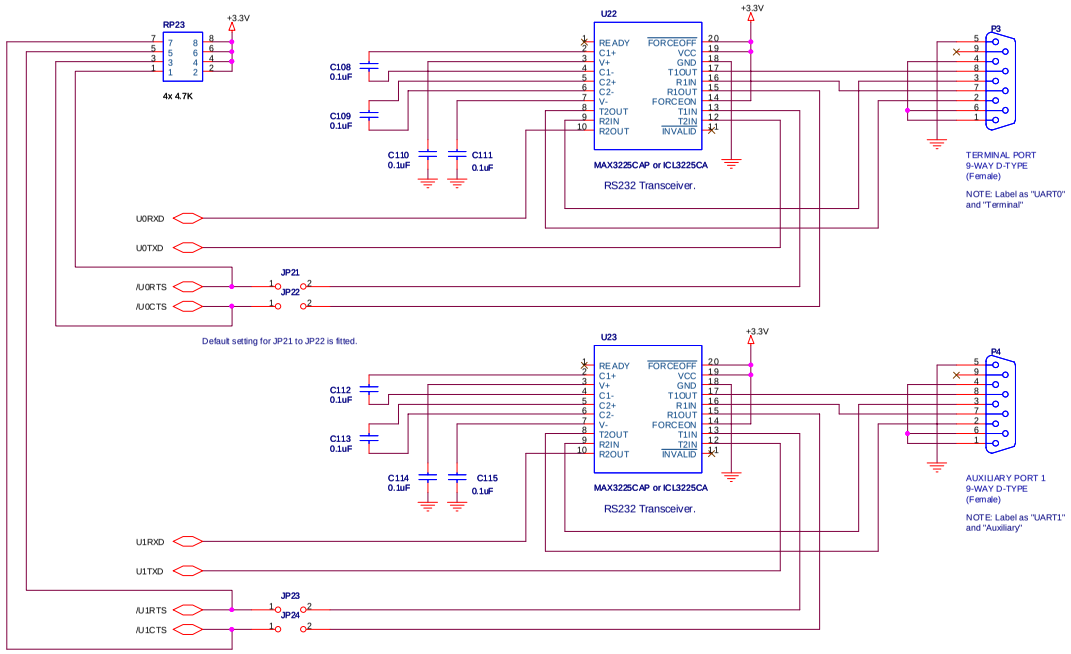


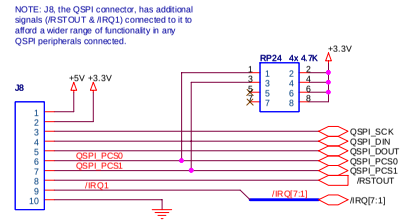
NOTE: Memory size: Each SDRAM memory is configured 4M x 16bit (8MB). Total available SDRAM is 16MB.

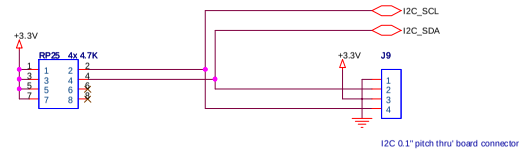
NOTE: Alternative SDRAM's with the same PCB footprint are:
Samsung K4S641632E
Hyundai HY57V641620HG
Toshiba TC59S6416CFT
Infineon HYB39S64160ET
Winbond W986416DH

NOTE: Please place the SDRAM (U20 & U21) as close to the MCF5271(U8) as possible to minimise track lengths. Particularly in relation to the SDRAM control signals & clocks where, if possible, these tracks should be a similar length i.e. within +/-5mm.

Motorola SPS TSPG - TECD ColdFire Group		
Title: M5271EVB		
Size: B	Document Number: SDRAM	Rev: 1.3
Date: Wednesday, April 21, 2004	Sheet 12 of 13	







Motorola SPS TSPG - TECD ColdFire Group		
Title: M5271EVB		
Size: C	Document Number: Serial I/O	Rev: 1.3
Date: Wednesday, April 21, 2004	Sheet 13 of 13	

Appendix C
M5271EVB BOM

C.1 M5271EVB BOM

Table C-1. M5271EVB BOM (Sheet 1 of 3)

Item	Qty	Reference	Part	Function
1	33	C1,C2,C3,C4,C13,C14,C15, C16,C17,C18,C26,C27,C34, C39,C40,C41,C43,C46,C48, C65,C66,C77,C78,C79,C80, C85,C91,C100,C101,C102,C103, C120,C122,C124	1nF	SMT Decoupling Capacitors
2	59	C12,C21,C22,C23,C24,C30,C31, C32,C33,C42,C44,C45,C47,C49, C51,C52,C53,C55,C56,C57,C58, C67,C68,C81,C82,C83,C84,C86, C87,C89,C90,C94,C97,C104,C105, C106,C107,C108,C109,C110, C111,C112,C113,C114,C115,C116, C117,C118,C119,C121,C123,C125	0.1uF	SMT Decoupling Capacitors
3	8	C19,C20,C28,C29,C35,C36,C37 ,C38	100pF	SMT Capacitors
4	2	C25,C95	10uF TANT	SMT Capacitors
5	2	C50,C54	47uF TANT	SMT Capacitors
6	7	C59,C61,C62,C69,C70,C71,,C72	10nF	SMT Capacitors
7	1	C60	10uF AVX	SMT Capacitors
8	6	C63,C64,C73,C74,C75,C76	470pF	SMT Capacitors
9	3	C88,C93,C96	33uF	SMT Capacitors
10	1	C92	1000uF 35V	SMT Capacitors
11	2	C98,C99	10pF	SMT Capacitors
12	8	C92,C93,C94,C95,C96,C97, C98,C99	0.22uF	SMT Capacitors
13	7	D1,D2,D3,D4,D6,D9,D13	AA3528SGC Kingbright Green LED	SMT LEDs
14	5	D5,D7,D8,D12,D14	MBRS340T3 On Semi	SMC Schottky Rectifier
15	2	D10,D11	MRA4003T3 On Semi	SMA Power Rectifier

Table C-1. M5271EVB BOM (Sheet 2 of 3)

Item	Qty	Reference	Part	Function
16	2	D15,D16	AA3528SRC Kingbright Red LED	SMT LEDs
17	8	D17,D18,D19,D20,D21,D22,D23, D24	AA3528MBC Kingbright Blue LED	SMT LEDs
18	4	FB1,FB2,FB3,FB4	HI1206T500R-00	Ferrite
19	1	F1	Fuse Holder by Keystone 0216005.H ; Fuse by Littlefuse, 5a, 250V, 5x 20mm, glass	5A Fast blow fuse and holder
20	7	JP1,JP2,JP5,JP6,JP7,JP10,JP11	Harwin M22-2010305	3-way jumper
21	19	JP3,JP4,JP8,JP9,JP12,JP13,JP14,JP15,JP16,JP17,JP18,JP19,JP20,JP21,JP22,JP23,JP24,JP25,JP26	Harwin M22-2010205	2-way jumper
22	1	J1	Thomas&Betts 609-2627	BDM 26-way header
23	1	J2	Halo HFJ11-2450E	RJ45 Connector w/mag
24	4	J3,J4,J5,J6	AMP 177983-2	60 SMT Recpetacle
25	1	J7	AMP 1053378-1	RF/SMB/V External Clock conn
26	1	J8	Molex 22-10-2101	Conn, 1x10 .1 male header
27	1	J9	Molex 22-10-2041	Conn, 1x4 .1 male header
28	1	L1	1210-103J API Delevan	External 10uH inductor
29	2	L2,L3	SIEMENS B82111-B-C24	25uH Inductors
30	1	P1	Switchcraft RAPC722	power jack 2.1mm
31	4	P2	Augat 25V-02	2-way bare wire power connector
32	3	P3,P4,P5	AMP 747844-3	DB9 Female
33	1	P2	Switchcraft RAPC722	PSU barrel connector
34	18	RP1,RP2,RP8,RP9,RP10,RP11, RP12,RP14,RP15,RP16,RP17, RP20,RP21,RP22,RP23,RP24, RP25,RP26	Philips	4x4.7 SMT 0603
35	1	RP3	Philips	4x22 SMT 0603
36	1	RP4,RP5,RP6,RP7	Philips	4x51 SMT 0603
37	1	RP13	Philips	4x10K SMT 0603
38	2	RP18,RP19	Philips	4x10 SMT 0603
39	3	R1,R2,R23	Philips	SM/R 0805 22 ohm resistor
40	3	R3,R10,R25	Philips	SM/R 0805 10K ohm resistor
41	4	R4,R5,R6,R7	Philips	SM/R 0805 49.9 1% ohm resistor
42	5	R8,R15,R16,R17,R18	Philips	SM/R 0805 4.7K ohm resistor

Table C-1. M5271EVB BOM (Sheet 3 of 3)

Item	Qty	Reference	Part	Function
43	1	R9	Philips	SM/R 0805 6.49K 1% ohm resistor
44	4	R11,R12,R13,R14	Philips	SM/R 0805 220 ohm resistor
45	3	R19,R24,R27	Philips	SM/R 0805 270 ohm resistor
46	1	R20	Philips	SM/R 0805 560 ohm resistor
47	1	R21	Philips	SM/R 0805 0 ohm resistor
48	1	R22	Philips	SM/R 0805 120ohm resistor
49	2	R26,R28	Philips	SM/R 0805 100 ohm resistor
50	1	R29	Philips	SM/R 0805 1K ohm resistor
51	1	SW1	EAO Switch	POWER SW SLIDE-SPST(Board Edge)
52	1	SW2	C&K KS11R22CQD	$\overline{\text{IRQ7}}$ black push-button switch
53	1	SW3	C&K KS11R23CQD	Hard reset push-button switch
54	1	SW4	Grayhill 78RB12	Configuration DIP switch
55	11	TP1,TP2,TP3,TP4,TP5,TP6,TP7,TP8,TP9,TP10,TP11	Keystone 5015	Test points
56	2	U1,U2	CY7C1041CV3310ZC	ASRAM (Not populated)
57	3	U3,U4,U6	MC74LCX16245DT	Bus transceiver
58	2	U5,U18	SN74LVC1G11	3-input positive AND gate
59	1	U7	MC74LCX245DT	Bus transceiver
60	1	U8	MCF5271CVM100	Freescale MCF5271 microprocessor
61	1	U9	KS8721BL	MCF5282 ColdFire
62	1	U10	Am29LV160CB	AMD 2MB Flash
63	1	U11	Am29PL320DB	AMD 4MB Flash (not populated)
64	1	U12	LM2596S-3.3	National Semi DCtoDC switcher
65	1	U13	LM2596S-5	National Semi DCtoDC switcher
66	1	U14	LT1086CM	Regulator
67	2	U16,U17	ADM708SAR	Voltage sensor
68	2	Y1	FOXS/250F-20	25MHz Crystal
69	1	U15	P1145-HCV	25MHz Oscillator
70	1	U19	MC74LCX541DT	Octal Buffer
71	2	U20,U21	MT48LC4M16A2TG (TSOP II 400 mil)	SDRAM
72	3	U22,U23,U24	MAX3225CAP	RS232 Transceivers

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

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

Document Number: M5271EVBUM
Rev. 1.1
05/2006

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2004, 2006. All rights reserved.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

freescale™
semiconductor