1 Preface

EMC information:

• 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.
• This product is designed and intended for use as a development platform for hardware or software in an educational or professional laboratory.
• In a domestic environment this product may cause radio interference in which case you may be required to take adequate measures.
• Anti-static precautions must be adhered to when using this product.
• 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 you, at your own expense, are required to correct the interference.

2 M523xEVB

This document details the setup and configuration of the ColdFire M523xEVB evaluation board (hereafter referred to as the EVB). The EVB is intended to provide a mechanism for easy customer evaluation of the MCF523x family of 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 MCF523x family are supported.

The heart of the evaluation board is the MCF5235. All the other M523x family members have a subset of the MCF5235 specification and therefore can be fully emulated using the MCF5235 device. Table 1 below details the full product family.

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Package</th>
<th>eTPU</th>
<th>FEC</th>
<th>Crypto</th>
<th>CAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCF5232CAB80</td>
<td>160 QFP</td>
<td>16-channel</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>MCF5232CVM100</td>
<td>196 MAPBGA</td>
<td>16-channel</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>MCF5232CVM150</td>
<td>196 MAPBGA</td>
<td>16-channel</td>
<td>No</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>MCF5233CVM100</td>
<td>256 MAPBGA</td>
<td>32-channel</td>
<td>No</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>MCF5233CVM150</td>
<td>256 MAPBGA</td>
<td>32-channel</td>
<td>No</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>MCF5234CVM100</td>
<td>256 MAPBGA</td>
<td>16-channel</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>MCF5234CVM150</td>
<td>256 MAPBGA</td>
<td>16-channel</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
</tr>
<tr>
<td>MCF5235CVM100</td>
<td>256 MAPBGA</td>
<td>16-channel</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>MCF5235CVM150</td>
<td>256 MAPBGA</td>
<td>16-channel</td>
<td>Yes</td>
<td>Yes</td>
<td>2</td>
</tr>
</tbody>
</table>

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

The EVB provides low cost software testing using a ROM-resident debug monitor, dBUG, programmed into the external flash device. Operation allows you to load code in the on-board RAM, execute applications, set breakpoints, and display/modify registers or memory. No additional hardware or software is required for basic operation.
Specifications:
- Freescale MCF5235 Microprocessor (150 MHz max core frequency)
- External Clock source: 25 MHz
- Operating temperature: 0°C to +70°C
- Power requirement: 7–14V DC @ 300 mA Typical
- Power output: 5V, 3.3V and 1.5V regulated supplies
- Board Size: 10.00 × 5.40 inches, 8 layers

Memory Devices:
- 16-Mbyte SDRAM
- 2-Mbyte (512K × 16) page mode flash or 4-Mbyte (512K × 32) page mode flash
- 1-Mbyte MRAM
- 64-Kbyte SRAM internal to MCF523x device

Peripherals:
- Ethernet port 10/100Mbps (dual-speed Fast Ethernet transceiver with MII)
- UART0 (RS-232 serial port for dBUG firmware)
- UART1 (auxiliary RS-232 serial port)
- UART2 (auxiliary1 RS-232 serial port jumper selectable with FlexCAN1)
- Enhanced Time Processor Unit (eTPU)
- I²C interface
- QSPI interface to ADC
- FlexCAN0 interface
- USB Host and Device Interface
- BDM/JTAG interface

User Interface:
- Reset logic switch (debounced)
- Boot logic selectable (dip switch)
- Abort/IRQ7 logic switch (debounced)
- PLL Clocking options—Oscillator, Crystal or SMA for external clocking signals
- LEDs for power-up indication, general purpose I/O, and timer output signals
- Expansion connectors for daughter card
- UNI-3 connector for motor control cards

Software:
- Resident firmware package that provides a self-contained programming and operating environment (dBUG)
A jumper chooses between 16 eTPU channels and Ethernet or 32 eTPU channels and no Ethernet.

Figure 1. M523xEVB Block Diagram

2.1 MCF5235 Microprocessor

The microprocessor used on the EVB is the highly integrated Freescale MCF5235 32-bit ColdFire variable-length RISC processor. The MCF5235 implements a Version 2 ColdFire core with a maximum core frequency of 150 MHz and external bus speed of 75 MHz. Features of the MCF5235 include:

- V2 ColdFire core with enhanced multiply-accumulate unit (EMAC) providing 144 (Dhrystone 2.1) MIPS @ 150 MHz
- eTPU with 16 or 32 channels, 6 Kbytes of code memory and 1.5 Kbytes of data memory with debug support
- 64 Kbytes of internal SRAM
- External bus speed of one half the CPU operating frequency (75 MHz bus @ 150 MHz core)
- 10/100 Mbps bus-mastering Ethernet controller
- 8 Kbytes of configurable instruction/data cache
- Three universal asynchronous receiver/transmitters (UARTs) with DMA support
- Controller area network 2.0B (FlexCAN module)
  - Optional second FlexCAN module multiplexed with the third UART
- Inter-integrated circuit (I²C) 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 MCF5235 communicates with external devices over a 32-bit wide data bus, D[31:0]. The MCF5235 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 Section 6, “Schematics” for their pin assignments.

The MCF5235 processor supports BDM and JTAG. These ports are multiplexed and can be used with third party tools to allow you to download code to the board. The board is configured to boot up in the normal/BDM mode of operation. The BDM signals are available at the port labeled BDM.
Figure 2 shows the MCF5235 processor block diagram.
2.2 System Memory

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

![Figure 3. External Memory Scheme](image)

**NOTE**
The external bus interface signals to the external MRAM 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.

### 2.2.1 External Flash

The EVB is fitted with a single 512K × 16 page-mode flash memory (U19) giving a total memory space of 2Mbytes. Alternatively, a footprint is available for you to upgrade this device to a 512K × 32 page-mode flash memory (U35), doubling the memory size to 4 Mbytes. U19 or U35 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.

**NOTE**
The debug monitor firmware is installed in this flash device. Development tools or your application programs may erase or corrupt the debug monitor. If the debug monitor becomes corrupted and it’s operation is desired, the firmware must be programmed into the flash by applying a development port tool such as BDM. Use caution to avoid this situation. The M523xEVB dBUG debugger/monitor firmware is programmed into the lower sectors of flash (0xFFE0_0000–0xFFE2_FFFF for 2 Mbytes of flash or 0xFFC0_0000–0xFFC2_FFFF for 4 Mbytes of flash).
By default with U19 fitted on the EVB, jumper 64 (JP64) provides an alternative hardware mechanism for write protection.

If you replaced U19 with the 32-bit flash device (U35), jumper 31 (JP31) has the same functionality as JP64. U35 also has it’s own hardware write protect pin (C5) that protects the bottom boot sector when pulled to ground.

2.2.2 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 \( \times 16 \times 4 \) 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 MCF523x 32 bit data bus.

2.2.3 MRAM

The EVB is populated with two 512K \( \times 16 \) MRAM devices (Freescale MR2A16A). Also see Section 2.2.5, “M523xEVB Memory Map”.

2.2.4 Internal SRAM

The MCF5235 processor has 64-Kbytes of internal SRAM memory that 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 you. The memory is relocatable to any 32-Kbyte boundary within the processor’s four gigabyte address space.

2.2.5 M523xEVB Memory Map

Interface signals to support the interface to external memory and peripheral devices are generated by the memory controller. The MCF5235 supports eight external chip selects: CS\([1:0]\) are used with external memories, CS2 is used for the USB controller, and CS\([7:3]\) are easily accessible to you via the daughter card expansion connectors. CS0 also functions as the global (boot) chip-select for booting out of external flash.

Because the MCF5235 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, is found in Table 2. The internal memory space of the MCF5235 is detailed further in the MCF5235 Reference Manual. Chip selects 0 and 1 can be changed by your 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:**

<table>
<thead>
<tr>
<th>External flash Memory</th>
<th>CS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>External MRAM Memory</td>
<td>CS1</td>
</tr>
</tbody>
</table>
Table 2 shows the M523xEVB memory map.

### Table 2. Default M523xEVB Memory Map

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Signal and Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000_0000–0x00FF_FFFF</td>
<td>16 Mbyte SDRAM</td>
</tr>
<tr>
<td>0x2000_0000–0x2000_FFFF</td>
<td>64 Kbytes Internal SRAM</td>
</tr>
<tr>
<td>0x3000_0000–0x300F_FFFF</td>
<td>External MRAM</td>
</tr>
<tr>
<td>0xFFE0_0000–0xFFFF_FFFF</td>
<td>2 Mbytes External Flash</td>
</tr>
<tr>
<td>or</td>
<td>or</td>
</tr>
<tr>
<td>0xFFFF_FFFF</td>
<td>4 Mbytes External Flash</td>
</tr>
</tbody>
</table>

#### 2.2.5.1 Reset Vector Mapping

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

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

After the processor is granted the bus, it then performs two longword read bus cycles. The first longword at address 0x0000_0000 is loaded into the stack pointer and the second longword at address 0x0000_0004 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 size that the MCF5235 accesses at address 0x0000_0000 is determined at reset by sampling D[20:19].

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

<table>
<thead>
<tr>
<th>D[20:19]</th>
<th>Boot Device/Data Port Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>External (32-bit)</td>
</tr>
<tr>
<td>01</td>
<td>External (16-bit)</td>
</tr>
<tr>
<td>10</td>
<td>External (8-bit)</td>
</tr>
<tr>
<td>11</td>
<td>External (32-bit)</td>
</tr>
</tbody>
</table>

#### 2.3 Support Logic

##### 2.3.1 Reset Logic

The reset logic provides system initialization. Reset occurs at power-on or by asserting \texttt{RESET} via SW6.
dBUG configures the MCF5235 microprocessor internal resources during initialization.

- Instruction cache is invalidated and disabled
- Vector base register (VBR) contains an address that initially points to the flash memory
- Contents of the exception table are written to address 0x0000_0000 in the SDRAM
- Software watchdog timer is disabled
- Bus monitor is enabled
- Internal timers are placed in a stop condition
- Interrupt controller registers are initialized with unique interrupt level/priority pairs

If the external RCON pin is asserted during reset (SW7-1 ON), 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 RCON is not asserted during reset (SW7-1 OFF), 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 4. SW7-1 RCON

<table>
<thead>
<tr>
<th>SW7-1</th>
<th>Reset Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>RCON not asserted, Default Chip configuration or RCON register settings</td>
</tr>
<tr>
<td>ON</td>
<td>RCON is asserted, Chip functions, including the reset configuration after reset, are configured according to the levels driven onto the external data pins.</td>
</tr>
</tbody>
</table>

### Table 5. SW7-2 JTAG_EN

<table>
<thead>
<tr>
<th>SW1-2</th>
<th>JTAG Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>JTAG interface enabled</td>
</tr>
<tr>
<td>ON</td>
<td>BDM interface enabled</td>
</tr>
</tbody>
</table>

### Table 6. SW7-[4:3] Encoded Clock Mode

<table>
<thead>
<tr>
<th>SW7-3</th>
<th>SW7-4</th>
<th>Clock Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>External clock mode- (PLL disabled)</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>1:1 PLL</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>Normal PLL mode with external clock reference</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
<td>Normal PLL mode w/crystal oscillator reference</td>
</tr>
</tbody>
</table>
2.3.2 Clock Circuitry

There are three options to supply the clock to the CPU. These options are configured by setting JP[35:37].

Table 11. M523xEVB Clock Source Selection

<table>
<thead>
<tr>
<th>JP35</th>
<th>JP36</th>
<th>JP37</th>
<th>Clock Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>1-2</td>
<td>ON</td>
<td>25 MHz Oscillator (default setting)</td>
</tr>
<tr>
<td>2-3</td>
<td>1-2</td>
<td>ON</td>
<td>25 MHz External Clock</td>
</tr>
<tr>
<td>X</td>
<td>2-3</td>
<td>OFF</td>
<td>25 MHz Crystal (not populated)</td>
</tr>
</tbody>
</table>
The 25-MHz oscillator (U23) also feeds the Ethernet chip (U11).
There is also a 12-MHz crystal feeding the USB controller (U33).

2.3.3 Watchdog Timer
The dBUG firmware does not enable the watchdog timer on the MCF5235.

2.3.4 Exception Sources
The ColdFire family of processors supports seven levels of interrupt priorities. When the processor receives an interrupt with a higher priority than the current interrupt mask (in the status register), it performs an interrupt acknowledge cycle at the end of the current instruction cycle. This interrupt acknowledge cycle tells the interrupt source that the request is acknowledged and the source should provide the vector number to indicate where the service routine for this interrupt level is located. If the interrupt source is not capable of providing a vector, its interrupt should be set up as an auto-vector interrupt that directs the processor to a predefined entry in the exception table (refer to the MCF5235 Reference Manual).

The processor goes to an exception routine via the exception table. This table is stored in the flash EEPROM and its address location is stored in the VBR. The dBUG ROM monitor writes a copy of the exception table into the RAM starting at 0x0000_0000. To set an exception vector, place the address of the exception handler in the appropriate vector in the vector table and then point the VBR to 0x0000_0000.

The MCF5235 microprocessor has seven external interrupt request lines IRQ[7:1]. The interrupt controller is capable of providing up to 63 interrupt sources. These sources are:

- External interrupt signals IRQ[7:1] (EPORT)
- Software watchdog timer module
- Timer modules
- UART modules 0, 1 and 2
- I²C module
- DMA module
- QSPI module
- FEC module
- PIT
- Security module
- FlexCAN0 and FlexCAN1
- eTPU

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 its priority programmed by setting the xIPL[2:0] bits in the interrupt control registers except interrupts 1–7 because they have a fixed priority.
No interrupt sources should have the same level and priority as another. Programming two interrupt sources with the same level and priority results in undefined operation.

The M523xEVB hardware uses IRQ7 to support the ABORT function using the ABORT switch (SW5). This switch is used to force an interrupt (level 7, priority 3) if your program execution should be aborted without issuing a reset (refer to Section 4, “Using the Monitor/Debug Firmware”, for more information on ABORT). Because 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 the MCF5235 Reference Manual for more information about the interrupt controller.

### 2.3.5 \( \overline{TA} \) Generation

The processor starts a bus cycle by asserting \( \overline{CSx} \) with the other control signals. The processor then waits for a transfer acknowledgment (\( \overline{TA} \)) from within (auto-acknowledge, AA mode) or from the externally addressed device before it can complete the bus cycle. \( \overline{TA} \) is indicates the completion of the bus cycle. It also allows devices with different access times to communicate with the processor properly asynchronously.

The MCF5235 processor, as part of the chip-select logic, has a built-in mechanism to generate \( \overline{TA} \) for all external devices that cannot 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. To support future expansion of the M523xEVB, the \( \overline{TA} \) input of the processor is also connected to the processor expansion bus (J9, pin 44). 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.

### 2.3.6 User’s Program

JP64 on the 16-Mbit flash (U19) or JP31 if using 32-Mbit flash (U35) allows you to test code from boot/POR without having to overwrite the ROM monitor.

When the jumper is set between pins 1 and 2, the behavior of the system is normal: dBUG boots and then runs from 0xFFE0_0000 (0xFFC0_0000). When the jumper is set between pins 2 and 3, the board boots from the top half of the flash (0xFFF0_0000).

Procedure:

1. Compile and link as though the code was to be placed at the base of the flash.
2. Set up the jumper JP64 (JP31) for normal operation, pin 1 connected to pin 2.
3. Download to SDRAM. (If using serial or Ethernet, start the ROM monitor first. If using BDM via a wiggler cable, download first, then start ROM monitor by pointing the program counter (PC) to 0xFFE0_0400 (0xFFC0_0400) and run.)
4. In the ROM monitor, execute the \$FL write <dest> <src> <bytes>’ command.
5. Move jumper JP64 (JP31) to pin 2 connected to pin 3 and push the reset button (SW6). Your code should now be running from reset/POR.
2.4 Communication Ports

The EVB provides external communication interfaces for two UART serial ports, a UART/FlexCAN1 port, FlexCAN0 port, QSPI, I²C port, 10/100T Ethernet port, eTPU port (including UNI3 and HS/ENCO connectors for auxiliary motor control cards), USB host port, USB device port, and BDM/JTAG port.

2.4.1 UART0 and UART1 Ports

The MCF5235 device has three built-in UARTs, each with its own software-programmable baud rate generator. Two of these UART interfaces are brought out to RS232 transceivers. One channel is the ROM monitor to terminal output and the other is available to you. The ROM monitor programs the interrupt level for UART0 to level 3, priority 2 and autovector mode of operation. The interrupt level for UART1 is programmed to level 3, priority 1 and autovector mode of operation. The signals from these channels are available on expansion connectors J7 and J8. The UART0 and UART1 signals are passed through the RS-232 transceivers (U30) & (U31) and are available on DB-9 connectors (P4) and (P5).

Refer to the MCF5235 Reference Manual for programming the UARTs and their register maps.

2.4.2 UART2/FlexCAN1 Port

The third UART on the MCF5235 is multiplexed with the second FlexCAN module (FlexCAN1). The functionality of these ports is jumper-selectable on the EVB. Table 12 shows the jumper configuration to activate UART2 or FlexCAN1.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>UART2 Setting</th>
<th>FlexCAN1 Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP7</td>
<td>1-2</td>
<td>2-3</td>
</tr>
<tr>
<td>JP12</td>
<td>1-2</td>
<td>2-3</td>
</tr>
<tr>
<td>JP25</td>
<td>2-3</td>
<td>X</td>
</tr>
<tr>
<td>JP26</td>
<td>2-3</td>
<td>X</td>
</tr>
<tr>
<td>JP50</td>
<td>2-3</td>
<td>1-2</td>
</tr>
<tr>
<td>JP51</td>
<td>2-3</td>
<td>1-2</td>
</tr>
<tr>
<td>JP52</td>
<td>2-3</td>
<td>1-2</td>
</tr>
</tbody>
</table>

The signals of UART2 are passed through RS-232 transceiver U32 and are jumper-selectable (see Table 12) on DB-9 connector P6.

The CAN1TX and CAN1RX signals from FlexCAN1 are brought out to a 3.3-V CAN transceiver (Texas Instruments SN65HVD230D) and are jumper-selectable (see Table 12) on DB-9 connector P6. Jumpers JP3 and JP4 control the CAN hardware configuration.
2.4.3 **FlexCAN0 Port**

The EVB provides one dedicated CAN transceiver. The CAN0TX and CAN0RX signals are brought out to a 3.3V CAN transceiver (Texas Instruments SN65HVD230D). Jumper JP1 and JP2 control the CAN hardware configuration.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Function</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP3</td>
<td>Transceiver mode</td>
<td>Standby</td>
<td>High Speed (No Slope Control)</td>
</tr>
<tr>
<td>JP4</td>
<td>CAN Termination</td>
<td>Terminating resistor between CANL and CANH</td>
<td>No terminating resistor</td>
</tr>
</tbody>
</table>

The CANL and CANH signals are brought out from the CAN transceiver to a female DB-9 connector (P1) in the configuration below.

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Function</th>
<th>ON</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP1</td>
<td>Transceiver mode</td>
<td>Standby</td>
<td>High Speed (No Slope Control)</td>
</tr>
<tr>
<td>JP2</td>
<td>CAN Termination</td>
<td>Terminating resistor between CANL and CANH</td>
<td>No terminating resistor</td>
</tr>
</tbody>
</table>

2.4.4 **10/100T Ethernet Port**

The MCF5235 microprocessor populated on the EVB is a superset device of the MCF523x family. The upper 16 eTPU channels are multiplexed with the ethernet port giving you the choice of using the full 32-channels of eTPU or 16-channels of eTPU with the Fast Ethernet controller (FEC) activated. Pin M4 on the MCF5235 configures the internal functionality of these 16 pins. If you are using the FEC, pin M4 must be pulled low by setting SW7-11 to the ON position.

These 16 pins are also jumper-selectable between the eTPU and the FEC to isolate the external circuitry required to implement the functionality of these modules. Table 16 lists the appropriate jumper settings to enable eTPU or FEC functionality on these pins.

The MCF5235 device performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions. The MCF5235 Ethernet controller requires an external interface adaptor and
transceiver function to complete the interface to the ethernet media. The MCF5235 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 50 MHz
- 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 MCF5235 Reference Manual. The on-board ROM monitor is programmed to allow you to download files from a network to memory in different formats. The current compiler formats supported are S-Record, COFF, ELF or Image.

### Table 16. Ethernet/eTPU Jumper Configuration

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Pin</th>
<th>Ethernet Setting</th>
<th>Ethernet Signal</th>
<th>eTPU Setting</th>
<th>eTPU Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP5</td>
<td>D5</td>
<td>2-3</td>
<td>ERXER</td>
<td>1-2</td>
<td>23</td>
</tr>
<tr>
<td>JP9</td>
<td>C5</td>
<td>2-3</td>
<td>ETXCLK</td>
<td>1-2</td>
<td>22</td>
</tr>
<tr>
<td>JP10</td>
<td>B5</td>
<td>2-3</td>
<td>ETXD2</td>
<td>1-2</td>
<td>18</td>
</tr>
<tr>
<td>JP11</td>
<td>A5</td>
<td>2-3</td>
<td>ETXD1</td>
<td>1-2</td>
<td>17</td>
</tr>
<tr>
<td>JP13</td>
<td>D6</td>
<td>2-3</td>
<td>ETXEN</td>
<td>1-2</td>
<td>21</td>
</tr>
<tr>
<td>JP14</td>
<td>C6</td>
<td>2-3</td>
<td>ETXER</td>
<td>1-2</td>
<td>20</td>
</tr>
<tr>
<td>JP15</td>
<td>B6</td>
<td>2-3</td>
<td>ETXD3</td>
<td>1-2</td>
<td>19</td>
</tr>
<tr>
<td>JP16</td>
<td>C4</td>
<td>2-3</td>
<td>ERXD0</td>
<td>1-2</td>
<td>24</td>
</tr>
<tr>
<td>JP17</td>
<td>B7</td>
<td>2-3</td>
<td>ETXD0</td>
<td>1-2</td>
<td>16</td>
</tr>
<tr>
<td>JP18</td>
<td>C3</td>
<td>2-3</td>
<td>ERXD1</td>
<td>1-2</td>
<td>25</td>
</tr>
</tbody>
</table>
2.4.5 eTPU

The eTPU is an intelligent programmable I/O controller with its own core and memory system, allowing it to perform complex timing and I/O management independently of the CPU. The eTPU is essentially a co-processor designed for timing control, I/O handling, serial communications, motor control, and engine control applications and accesses data without the host CPU’s intervention. Consequently, the host CPU setup and service times for each timer event are minimized or eliminated.

The eTPU is an enhanced version of the TPU module implemented on the MC68332 and MPC500 products. Enhancements of the eTPU include a more powerful processor that handles high-level C code efficiently and has more functionality and increased performance. Although there is no compatibility at the microcode level, the eTPU maintains several features of older TPU versions and is conceptually almost identical. The eTPU library is a superset of the standard TPU library functions modified to take advantage of enhancements in the eTPU. These, along with a C compiler, make it relatively easy to port older applications. By providing source code for the Freescale library, it is possible for the eTPU to support your own function development.

The eTPU has up to 32 timer channels in addition to having 6 Kbytes of code memory and 1.5 Kbytes of data memory that stores software modules downloaded at boot time and that can be mixed and matched as required for any specific application.

As mentioned in Section 2.4.4, “10/100T Ethernet Port,” the upper 16-channels of the eTPU are multiplexed with the Fast Ethernet controller. Refer to Table 16 to set the appropriate jumpers to enable 16 or 32-channels. To configure the device to operate with the top 16-channels of the eTPU activated, pin M4 must be pulled high by setting SW7-11 to the OFF position.

All 32 eTPU channels are available on a 0.1 2x20 Molex connector providing easy access to the eTPU.

### Table 16. Ethernet/eTPU Jumper Configuration (continued)

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Pin</th>
<th>Ethernet Setting</th>
<th>Ethernet Signal</th>
<th>eTPU Setting</th>
<th>eTPU Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP19</td>
<td>D4</td>
<td>2-3</td>
<td>ERXD2</td>
<td>1-2</td>
<td>26</td>
</tr>
<tr>
<td>JP20</td>
<td>D3</td>
<td>2-3</td>
<td>ERXD3</td>
<td>1-2</td>
<td>27</td>
</tr>
<tr>
<td>JP21</td>
<td>E3</td>
<td>2-3</td>
<td>ERXCLK</td>
<td>1-2</td>
<td>29</td>
</tr>
<tr>
<td>JP22</td>
<td>E4</td>
<td>2-3</td>
<td>ERXDV</td>
<td>1-2</td>
<td>28</td>
</tr>
<tr>
<td>JP23</td>
<td>F3</td>
<td>2-3</td>
<td>ECOL</td>
<td>1-2</td>
<td>31</td>
</tr>
<tr>
<td>JP24</td>
<td>F4</td>
<td>2-3</td>
<td>ECRS</td>
<td>1-2</td>
<td>30</td>
</tr>
</tbody>
</table>

### Table 17. eTPU Header Pin Assignment

<table>
<thead>
<tr>
<th>Pin</th>
<th>eTPU Signal</th>
<th>Pin</th>
<th>eTPU Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+3.3V</td>
<td>2</td>
<td>+5V</td>
</tr>
<tr>
<td>3</td>
<td>TPUCH16</td>
<td>4</td>
<td>UTPUODIS</td>
</tr>
<tr>
<td>5</td>
<td>TPUCH17</td>
<td>6</td>
<td>LTPUODIS</td>
</tr>
<tr>
<td>7</td>
<td>TPUCH18</td>
<td>8</td>
<td>TPUCH0</td>
</tr>
</tbody>
</table>
There is a UNI3 connector and HS/ENCO connector on the EVB for connection to an auxiliary card. The auxiliary card is intended for evaluation of the eTPU functionality.

### 2.4.6 BDM/JTAG Port

The MCF5235 processor has a background debug mode (BDM) port, which supports real-time trace and real-time debug. The signals necessary for debug are available at connector J1 as shown in Figure 4.

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 5.

<table>
<thead>
<tr>
<th>Pin</th>
<th>eTPU Signal</th>
<th>Pin</th>
<th>eTPU Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>TPUCH19</td>
<td>10</td>
<td>TPUCH1</td>
</tr>
<tr>
<td>11</td>
<td>TPUCH20</td>
<td>12</td>
<td>TPUCH2</td>
</tr>
<tr>
<td>13</td>
<td>TPUCH21</td>
<td>14</td>
<td>TPUCH3</td>
</tr>
<tr>
<td>15</td>
<td>TPUCH22</td>
<td>16</td>
<td>TPUCH4</td>
</tr>
<tr>
<td>17</td>
<td>TPUCH23</td>
<td>18</td>
<td>TPUCH5</td>
</tr>
<tr>
<td>19</td>
<td>TPUCH24</td>
<td>20</td>
<td>TPUCH6</td>
</tr>
<tr>
<td>21</td>
<td>TPUCH25</td>
<td>22</td>
<td>TPUCH7</td>
</tr>
<tr>
<td>23</td>
<td>TPUCH26</td>
<td>24</td>
<td>TPUCH8</td>
</tr>
<tr>
<td>25</td>
<td>TPUCH27</td>
<td>26</td>
<td>TPUCH9</td>
</tr>
<tr>
<td>27</td>
<td>TPUCH28</td>
<td>28</td>
<td>TPUCH10</td>
</tr>
<tr>
<td>29</td>
<td>TPUCH29</td>
<td>30</td>
<td>TPUCH11</td>
</tr>
<tr>
<td>31</td>
<td>TPUCH30</td>
<td>32</td>
<td>TPUCH12</td>
</tr>
<tr>
<td>33</td>
<td>TPUCH31</td>
<td>34</td>
<td>TPUCH13</td>
</tr>
<tr>
<td>35</td>
<td>GND</td>
<td>36</td>
<td>TPUCH14</td>
</tr>
<tr>
<td>37</td>
<td>TCRCLK</td>
<td>38</td>
<td>TPUCH15</td>
</tr>
<tr>
<td>39</td>
<td>GND</td>
<td>40</td>
<td>GND</td>
</tr>
</tbody>
</table>
2.4.7 I\(^2\)C

The MCF5235'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 *MCF5235 Reference Manual* for more details. The I\(^2\)C signals from the MCF5235 device are brought out to expansion connector J13.

The I\(^2\)C functionality of the MCF5235 is multiplexed on the same pins as the QSPI. Jumpers JP6 and JP8 are used to connect/disconnect the I\(^2\)C signals, SDA and SCL. To enable I\(^2\)C set JP6 and JP8 between pins 2 and 3.
2.4.8 Queued Serial Peripheral Interface (QSPI)

The QSPI module provides a serial peripheral interface with queued transfer capability. It supports 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 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 eight to 16 bits in one 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 75 MHz.
- Programmable delays before and after transfers
- Programmable QSPI clock phase and polarity
- Supports wrap-around mode for continuous transfers

Please see the *MCF5235 Reference Manual* for more details. The QSPI signals from the MCF5235 device are brought out to expansion connector J12.

Some of the QSPI signals are multiplexed with the I2C module. Set JP6 and JP8 between pins 1 and 2 to enable the QSPI module.

The EVB features an analog-to-digital converter (ADC) interfaced to the CPU via the QSPI. The ADC uses QSPI chip select 0. This chip select has a jumper that can be removed if you are not using the ADC and wish to connect QSPI_CS0 to an alternate device.

2.4.9 USB Host and Device

The EVB features a USB controller interfaced externally to the MCF5235 via the DMA and external bus modules. The USB controller can be configured to run in Host or Device mode.

There is a series A connector (host) and a series B connector (device) populated on the EVB. Either can be used depending on whether the USB controller is configured to run in host or device mode. Set JP56 between pins 2 and 3 to configure the controller for host mode or between pin 1 and 2 to configure the controller for device mode.

The USB controller also has On-The-Go (OTG) functionality. There is a footprint on the EVB for an OTG Mini-AB connector if you want to use USB OTG. If using OTG, JP55 must be fitted.

For more details see the Philips Semiconductor datasheet for the ISP1362 USB OTG controller.

There are a series of jumpers connected to the USB controller allowing you to disconnect the DMA and interrupt signals between the CPU and the USB controller if the USB controller is not in use. This gives you access to the DMA timer module channels 1 and 2 and an extra interrupt signal. Table 18 details these jumper settings.
2.5 Connectors and User Components

2.5.1 Daughter Card Expansion Connectors

Four, 60-way SMT connectors (J7, J8, J9 and J10) provide access to all MCF5235 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>
<thead>
<tr>
<th>Jumper</th>
<th>Functionality when Jumper is Fitted</th>
<th>Functionality when Jumper is NOT Fitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP57</td>
<td>USB DMA request signal</td>
<td>DMA Timer 1 input enabled</td>
</tr>
<tr>
<td>JP58</td>
<td>USB DMA request signal</td>
<td>DMA Timer 2 input enabled</td>
</tr>
<tr>
<td>JP59</td>
<td>USB DMA acknowledge signal</td>
<td>DMA Timer 2 output enabled</td>
</tr>
<tr>
<td>JP60</td>
<td>USB DMA acknowledge signal</td>
<td>DMA Timer 1 output enabled</td>
</tr>
<tr>
<td>JP61</td>
<td>DACK1 not in use - pulled high</td>
<td>DMA acknowledge 1 enabled</td>
</tr>
<tr>
<td>JP62</td>
<td>Interrupt 4 enabled for USB</td>
<td>Interrupt 4 disabled from USB</td>
</tr>
<tr>
<td>JP63</td>
<td>DACK2 not in use - pulled high</td>
<td>DMA acknowledge 2 enabled</td>
</tr>
</tbody>
</table>

Table 18. USB DMA Enable and Disable Settings

<table>
<thead>
<tr>
<th>Jumper</th>
<th>Functionality when Jumper is Fitted</th>
<th>Functionality when Jumper is NOT Fitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>JP57</td>
<td>USB DMA request signal</td>
<td>DMA Timer 1 input enabled</td>
</tr>
<tr>
<td>JP58</td>
<td>USB DMA request signal</td>
<td>DMA Timer 2 input enabled</td>
</tr>
<tr>
<td>JP59</td>
<td>USB DMA acknowledge signal</td>
<td>DMA Timer 2 output enabled</td>
</tr>
<tr>
<td>JP60</td>
<td>USB DMA acknowledge signal</td>
<td>DMA Timer 1 output enabled</td>
</tr>
<tr>
<td>JP61</td>
<td>DACK1 not in use - pulled high</td>
<td>DMA acknowledge 1 enabled</td>
</tr>
<tr>
<td>JP62</td>
<td>Interrupt 4 enabled for USB</td>
<td>Interrupt 4 disabled from USB</td>
</tr>
<tr>
<td>JP63</td>
<td>DACK2 not in use - pulled high</td>
<td>DMA acknowledge 2 enabled</td>
</tr>
</tbody>
</table>

Table 19. J7

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>2</td>
<td>+5V</td>
</tr>
<tr>
<td>3</td>
<td>+3.3V</td>
<td>4</td>
<td>+3.3V</td>
</tr>
<tr>
<td>5</td>
<td>+3.3V</td>
<td>6</td>
<td>+3.3V</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>TPUCH24</td>
<td>10</td>
<td>TPUCH6</td>
</tr>
<tr>
<td>11</td>
<td>TPUCH17</td>
<td>12</td>
<td>TPUCH4</td>
</tr>
<tr>
<td>13</td>
<td>TPUCH18</td>
<td>14</td>
<td>TPUCH5</td>
</tr>
<tr>
<td>15</td>
<td>TPUCH22</td>
<td>16</td>
<td>TPUCH2</td>
</tr>
<tr>
<td>17</td>
<td>TPUCH23</td>
<td>18</td>
<td>TPUCH3</td>
</tr>
<tr>
<td>19</td>
<td>TPUCH19</td>
<td>20</td>
<td>TPUCH1</td>
</tr>
<tr>
<td>21</td>
<td>TPUCH20</td>
<td>22</td>
<td>TPUCH0</td>
</tr>
<tr>
<td>23</td>
<td>TPUCH21</td>
<td>24</td>
<td>GND</td>
</tr>
<tr>
<td>25</td>
<td>TPUCH16</td>
<td>26</td>
<td>EMDIO</td>
</tr>
<tr>
<td>27</td>
<td>U2CTS</td>
<td>28</td>
<td>EMDC</td>
</tr>
<tr>
<td>29</td>
<td>I2C_SCL</td>
<td>30</td>
<td>I2C_SDA</td>
</tr>
<tr>
<td>31</td>
<td>QSPI_SCK</td>
<td>32</td>
<td>QSPI_DIN</td>
</tr>
<tr>
<td>33</td>
<td>BS3</td>
<td>34</td>
<td>QSPI_DOUT</td>
</tr>
</tbody>
</table>
### Table 19. J7 (continued)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>BS2</td>
<td>36</td>
<td>QSPI_PCS0</td>
</tr>
<tr>
<td>37</td>
<td>BS1</td>
<td>38</td>
<td>SD_SCKE</td>
</tr>
<tr>
<td>39</td>
<td>BS0</td>
<td>40</td>
<td>CAN1RX</td>
</tr>
<tr>
<td>41</td>
<td>U2RTS</td>
<td>42</td>
<td>U2RXD</td>
</tr>
<tr>
<td>43</td>
<td>QSPI_PCS1</td>
<td>44</td>
<td>U1CTS</td>
</tr>
<tr>
<td>45</td>
<td>U1RTS</td>
<td>46</td>
<td>CAN1TX</td>
</tr>
<tr>
<td>47</td>
<td>U1RXD</td>
<td>48</td>
<td>U2TXD</td>
</tr>
<tr>
<td>49</td>
<td>U1TXD</td>
<td>50</td>
<td>CS2</td>
</tr>
<tr>
<td>51</td>
<td>CS3</td>
<td>52</td>
<td>CS7</td>
</tr>
<tr>
<td>53</td>
<td>CS6</td>
<td>54</td>
<td>CS5</td>
</tr>
<tr>
<td>55</td>
<td>CS1</td>
<td>56</td>
<td>CS0</td>
</tr>
<tr>
<td>57</td>
<td>CS4</td>
<td>58</td>
<td>A23</td>
</tr>
<tr>
<td>59</td>
<td>GND</td>
<td>60</td>
<td>GND</td>
</tr>
</tbody>
</table>

### Table 20. J8

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>2</td>
<td>+1.5V</td>
</tr>
<tr>
<td>3</td>
<td>+3.3V</td>
<td>4</td>
<td>+3.3V</td>
</tr>
<tr>
<td>5</td>
<td>TPUCH8</td>
<td>6</td>
<td>TPUCH7</td>
</tr>
<tr>
<td>7</td>
<td>TPUCH10</td>
<td>8</td>
<td>TPUCH9</td>
</tr>
<tr>
<td>9</td>
<td>TPUCH25</td>
<td>10</td>
<td>TPUCH12</td>
</tr>
<tr>
<td>11</td>
<td>TPUCH27</td>
<td>12</td>
<td>TPUCH11</td>
</tr>
<tr>
<td>13</td>
<td>TPUCH26</td>
<td>14</td>
<td>TPUCH14</td>
</tr>
<tr>
<td>15</td>
<td>TPUCH29</td>
<td>16</td>
<td>TPUCH13</td>
</tr>
<tr>
<td>17</td>
<td>TPUCH28</td>
<td>18</td>
<td>TCRCLK</td>
</tr>
<tr>
<td>19</td>
<td>TPUCH31</td>
<td>20</td>
<td>TPUCH15</td>
</tr>
<tr>
<td>21</td>
<td>TPUCH30</td>
<td>22</td>
<td>GND</td>
</tr>
<tr>
<td>23</td>
<td>GND</td>
<td>24</td>
<td>U0CTS</td>
</tr>
<tr>
<td>25</td>
<td>U0RXD</td>
<td>26</td>
<td>DTOUT0</td>
</tr>
<tr>
<td>27</td>
<td>DTIN0</td>
<td>28</td>
<td>U0TXD</td>
</tr>
<tr>
<td>29</td>
<td>U0RTS</td>
<td>30</td>
<td>GND</td>
</tr>
<tr>
<td>31</td>
<td>CLKMOD0</td>
<td>32</td>
<td>+3.3V</td>
</tr>
<tr>
<td>33</td>
<td>CLKMOD1</td>
<td>34</td>
<td>GND</td>
</tr>
</tbody>
</table>
Table 20. J8 (continued)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>GND</td>
<td>36</td>
<td>D28</td>
</tr>
<tr>
<td>37</td>
<td>D30</td>
<td>38</td>
<td>D29</td>
</tr>
<tr>
<td>39</td>
<td>D31</td>
<td>40</td>
<td>D24</td>
</tr>
<tr>
<td>41</td>
<td>D26</td>
<td>42</td>
<td>D25</td>
</tr>
<tr>
<td>43</td>
<td>D27</td>
<td>44</td>
<td>D21</td>
</tr>
<tr>
<td>45</td>
<td>D23</td>
<td>46</td>
<td>D22</td>
</tr>
<tr>
<td>47</td>
<td>EXT_RSTIN</td>
<td>48</td>
<td>D19</td>
</tr>
<tr>
<td>49</td>
<td>GND</td>
<td>50</td>
<td>GND</td>
</tr>
<tr>
<td>51</td>
<td>D13</td>
<td>52</td>
<td>D20</td>
</tr>
<tr>
<td>53</td>
<td>D9</td>
<td>54</td>
<td>D17</td>
</tr>
<tr>
<td>55</td>
<td>D12</td>
<td>56</td>
<td>D18</td>
</tr>
<tr>
<td>57</td>
<td>D15</td>
<td>58</td>
<td>D16</td>
</tr>
<tr>
<td>59</td>
<td>GND</td>
<td>60</td>
<td>GND</td>
</tr>
</tbody>
</table>

Table 21. J9

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>2</td>
<td>+1.5V</td>
</tr>
<tr>
<td>3</td>
<td>+3.3V</td>
<td>4</td>
<td>+3.3V</td>
</tr>
<tr>
<td>5</td>
<td>+3.3V</td>
<td>6</td>
<td>+3.3V</td>
</tr>
<tr>
<td>7</td>
<td>GND</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>9</td>
<td>A21</td>
<td>10</td>
<td>A22</td>
</tr>
<tr>
<td>11</td>
<td>A19</td>
<td>12</td>
<td>A20</td>
</tr>
<tr>
<td>13</td>
<td>A17</td>
<td>14</td>
<td>A18</td>
</tr>
<tr>
<td>15</td>
<td>A16</td>
<td>16</td>
<td>A14</td>
</tr>
<tr>
<td>17</td>
<td>A15</td>
<td>18</td>
<td>A11</td>
</tr>
<tr>
<td>19</td>
<td>A13</td>
<td>20</td>
<td>GND</td>
</tr>
<tr>
<td>21</td>
<td>GND</td>
<td>22</td>
<td>A10</td>
</tr>
<tr>
<td>23</td>
<td>A12</td>
<td>24</td>
<td>A8</td>
</tr>
<tr>
<td>25</td>
<td>A9</td>
<td>26</td>
<td>A7</td>
</tr>
<tr>
<td>27</td>
<td>A6</td>
<td>28</td>
<td>A4</td>
</tr>
<tr>
<td>29</td>
<td>A5</td>
<td>30</td>
<td>GND</td>
</tr>
<tr>
<td>31</td>
<td>A2</td>
<td>32</td>
<td>A0</td>
</tr>
<tr>
<td>33</td>
<td>A3</td>
<td>34</td>
<td>A1</td>
</tr>
</tbody>
</table>
### Table 21. J9 (continued)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>GND</td>
<td>36</td>
<td>GND</td>
</tr>
<tr>
<td>37</td>
<td>DTIN3</td>
<td>38</td>
<td>UTPUODIS</td>
</tr>
<tr>
<td>39</td>
<td>DTOUT3</td>
<td>40</td>
<td>LTPUODIS</td>
</tr>
<tr>
<td>41</td>
<td>TIP</td>
<td>42</td>
<td>TEA</td>
</tr>
<tr>
<td>43</td>
<td>TS</td>
<td>44</td>
<td>TA</td>
</tr>
<tr>
<td>45</td>
<td>CAN0RX</td>
<td>46</td>
<td>SD_WE</td>
</tr>
<tr>
<td>47</td>
<td>R/W</td>
<td>48</td>
<td>CAN0TX</td>
</tr>
<tr>
<td>49</td>
<td>SD_CAS</td>
<td>50</td>
<td>SD_CS0</td>
</tr>
<tr>
<td>51</td>
<td>CLKOUT</td>
<td>52</td>
<td>SD_RAS</td>
</tr>
<tr>
<td>53</td>
<td>SD_CS1</td>
<td>54</td>
<td>DDATA3</td>
</tr>
<tr>
<td>55</td>
<td>XTAL</td>
<td>56</td>
<td>EXTAL</td>
</tr>
<tr>
<td>57</td>
<td>GND</td>
<td>58</td>
<td>GND</td>
</tr>
<tr>
<td>59</td>
<td>GND</td>
<td>60</td>
<td>GND</td>
</tr>
</tbody>
</table>

### Table 22. J10

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Pin</th>
<th>Signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+5V</td>
<td>2</td>
<td>+1.5V</td>
</tr>
<tr>
<td>3</td>
<td>+3.3V</td>
<td>4</td>
<td>+3.3V</td>
</tr>
<tr>
<td>5</td>
<td>D14</td>
<td>6</td>
<td>D10</td>
</tr>
<tr>
<td>7</td>
<td>D11</td>
<td>8</td>
<td>D6</td>
</tr>
<tr>
<td>9</td>
<td>D7</td>
<td>10</td>
<td>D8</td>
</tr>
<tr>
<td>11</td>
<td>D5</td>
<td>12</td>
<td>D4</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>14</td>
<td>GND</td>
</tr>
<tr>
<td>15</td>
<td>D1</td>
<td>16</td>
<td>D2</td>
</tr>
<tr>
<td>17</td>
<td>D3</td>
<td>18</td>
<td>OE</td>
</tr>
<tr>
<td>19</td>
<td>D0</td>
<td>20</td>
<td>DTOUT1</td>
</tr>
<tr>
<td>21</td>
<td>DTIN1</td>
<td>22</td>
<td>+3.3V</td>
</tr>
<tr>
<td>23</td>
<td>+3.3V</td>
<td>24</td>
<td>IRQ6</td>
</tr>
<tr>
<td>25</td>
<td>IRQ7</td>
<td>26</td>
<td>TSIZ0</td>
</tr>
<tr>
<td>27</td>
<td>TSIZ1</td>
<td>28</td>
<td>IRQ2</td>
</tr>
<tr>
<td>29</td>
<td>IRQ3</td>
<td>30</td>
<td>IRQ4</td>
</tr>
<tr>
<td>31</td>
<td>IRQ5</td>
<td>32</td>
<td>TCLK/PSTCLK</td>
</tr>
<tr>
<td>33</td>
<td>DTOUT2</td>
<td>34</td>
<td>DTIN2</td>
</tr>
</tbody>
</table>
2.5.2 Reset Switch (SW6)

The reset logic provides system initialization. Reset occurs at power-on or by asserting \texttt{RESET} via SW6.

A hard reset and voltage sense controller (U25) is used to produce an active low power-on reset signal. The reset switch (SW6) is fed into U25 that generates the signal tied to the MCF5235’s \texttt{RESET}. \texttt{RESET} is an open collector signal. Therefore, it can be wire-OR’ed with other reset signals from additional peripherals. On the EVB, \texttt{RESET} is wire OR’d with the BDM reset signal and a reset signal is available on the expansion connectors for use with your hardware.

See Section 2.3.1, “Reset Logic”, for more details on reset.

2.5.3 User LEDs

There are eight LEDs available to you. Each of these LEDs are pulled to +3.3V through a 10 Ω 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 its associated LED with a jumper. The table below details which MCF5235 signal is associated with which LED.

<table>
<thead>
<tr>
<th>LED</th>
<th>MCF5235 Signal</th>
<th>Jumper to disconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>D25</td>
<td>DTOUT0</td>
<td>JP38</td>
</tr>
<tr>
<td>D26</td>
<td>DTIN0</td>
<td>JP39</td>
</tr>
<tr>
<td>D27</td>
<td>DTOUT1</td>
<td>JP40</td>
</tr>
<tr>
<td>D28</td>
<td>DTIN1</td>
<td>JP41</td>
</tr>
</tbody>
</table>

Table 22. J10 (continued)

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Pin & Signal & Pin & Signal \\
\hline
35 & IRQ1 & 36 & TDI/DSI \\
37 & TDO/DSO & 38 & TMS/BKPT \\
39 & TRST/DSCLK & 40 & 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 \\
\hline
\end{tabular}
\end{table}

Table 23. User LEDs
2.5.4 Other LEDs

There are several other LEDs on the M523xEVB to signal various board/processor/component states. Below is a list of those LEDs and their functions:

Table 23. User LEDs (continued)

<table>
<thead>
<tr>
<th>LED</th>
<th>MCF5235 Signal</th>
<th>Jumper to disconnect</th>
</tr>
</thead>
<tbody>
<tr>
<td>D29</td>
<td>DOUT2</td>
<td>JP42</td>
</tr>
<tr>
<td>D30</td>
<td>DIN2</td>
<td>JP43</td>
</tr>
<tr>
<td>D31</td>
<td>DOUT3</td>
<td>JP44</td>
</tr>
<tr>
<td>D32</td>
<td>DIN3</td>
<td>JP45</td>
</tr>
</tbody>
</table>

Table 24. LED Functions

<table>
<thead>
<tr>
<th>LED</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1–D4</td>
<td>Ethernet Phy functionality</td>
</tr>
<tr>
<td>D5–D12</td>
<td>eTPU functionality</td>
</tr>
<tr>
<td>D14</td>
<td>+3.3V Power Good</td>
</tr>
<tr>
<td>D17</td>
<td>+5V Power Good</td>
</tr>
<tr>
<td>D23</td>
<td>Abort (IRQ7) asserted</td>
</tr>
<tr>
<td>D24</td>
<td>Reset (RSTI) asserted</td>
</tr>
<tr>
<td>D25–D32</td>
<td>User LEDs (See Table 23)</td>
</tr>
</tbody>
</table>

3 Initialization and Setup

3.1 System Configuration

The M523xEVB board requires the following items for minimum system configuration:

- The M523xEVB board (provided).
- Power supply, +7V to 14V DC with minimum of 300 mA.
- RS232C compatible terminal or a PC with terminal emulation software.
- RS232 communication cable (provided).
Figure 5 displays the minimum system configuration.

3.2 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,
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 Section 5, “Configuring dBUG for Network Downloads”.

3.2.1 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:

- M523xEVB Single Board Computer
- M5235EVB User’s Manual (this document)
- One RS232 communication cable
- One BDM (background debug mode) wiggler cable
- MCF5235 Reference Manual
- A selection of third party developer tools and literature

**NOTE**

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

After 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 end of this manual.

3.2.2 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 9 shows the position of the jumpers and connectors.

3.2.3 Providing Power to the Board

The EVB requires an external supply voltage of 7–14 V DC, minimum 1 Amp. 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 P2 is a 2.1mm power jack (Figure 6), P3 is a lever actuated connector (Figure 7).
3.2.4 Power Switch (SW4)

Slide switch SW4 isolates the power supply input from the EVB voltage regulators if required:

- Moving the slide switch to the left (towards connector P3) turns the EVB on.
- Moving the slide switch to the right (away from connector P3) turns the EVB off.

3.2.5 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>
<thead>
<tr>
<th>LED</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>D17</td>
<td>Indicates that the +5V regulator is working correctly</td>
</tr>
<tr>
<td>D14</td>
<td>Indicates that the +3.3V regulator is working correctly</td>
</tr>
</tbody>
</table>

If no LEDs are illuminated when the power is applied to the EVB, it is possible that 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.
3.2.6 Selecting Terminal Baud Rate

The serial channel UART0 of the MCF5235 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. When the dBUG ROM monitor is running, a SET command may be issued to select any baud rate supported by the ROM monitor.

3.2.7 The Terminal Character Format

The character format of the communication channel is fixed at power-up or reset. The default character format is eight 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.

3.2.8 Connecting the Terminal

The board is now ready to be connected to a PC/terminal.

1. Use the RS-232 serial cable to connect the PC/terminal to the M523xEVB 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.
2. Connect the 9-pin male connector to connector P4 on the M523xEVB board.
3. 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 male 25-pin or 9-pin. It may be necessary to obtain a 25 pin-to-9 pin adapter to make this connection. If an adapter is required, refer to Figure 8.

3.2.9 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 3.2.8, “Connecting the Terminal”.

After 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 as described in Section 3.2.7, “The Terminal Character Format”. The baud rate should be set to 19200. Power can now be applied to the board.

![Figure 8. Pin Assignment for Female (Terminal) Connector](image-url)
Pin assignments are as follows:

### Table 26. Pin Assignment for Female (Terminal) Connector

<table>
<thead>
<tr>
<th>DB9 Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Carrier Detect, Output (shorted to pins 4 and 6)</td>
</tr>
<tr>
<td>2</td>
<td>Receive Data, Output from board (receive refers to terminal side)</td>
</tr>
<tr>
<td>3</td>
<td>Transmit Data, Input to board (transmit refers to terminal side)</td>
</tr>
<tr>
<td>4</td>
<td>Data Terminal Ready, Input (shorted to pin 1 and 6)</td>
</tr>
<tr>
<td>5</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready, Output (shorted to pins 1 and 4)</td>
</tr>
<tr>
<td>7</td>
<td>Request to Send, Input</td>
</tr>
<tr>
<td>8</td>
<td>Clear to send, Output</td>
</tr>
<tr>
<td>9</td>
<td>Not connected</td>
</tr>
</tbody>
</table>
Figure 9 shows the jumper locations for the board.

3.3 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

ColdFire MCF5235 on the M523xEVB Firmware v3b.1a.xx (Built on xxx xx xxxx xx:xx:xx)
The board is now ready for operation under the control of the debugger as described in Section 4, “Using the Monitor/Debug Firmware”. If you do not get the above response, perform the following checks:

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

If a proper response does not occur after these steps, your board may have been damaged. Contact Freescale Semiconductor for further instructions. Please see the end of this manual for contact details.

3.4 Using The BDM Port

The MCF5235 microprocessor has a built in debug module referred to as BDM (background debug module). 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 MCF5235 Reference Manual for additional instructions.

NOTE

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

4 Using the Monitor/Debug Firmware

The M523xEVB single board computer has a resident firmware package that provides a self-contained programming and operating environment. The firmware, named dBUG, provides you with a monitor/debug interface, inline assembler and disassembly, program download, register and memory manipulation, and I/O control functions. This section is a how-to-use description of the dBUG package, including the user interface and command structure.

4.1 What is dBUG?

dBUG is a traditional ROM monitor/debugger that offers a comfortable and intuitive command line interface 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 you through pre-defined commands that are entered via the terminal. These commands are defined in Section 4.4, “Commands”.

An additional function called the system call allows your program to use various routines within dBUG. The system call is discussed at the end of this section.
The operational mode of dBUG is demonstrated in Figure 10. 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 your program, the dBUG firmware may or may not be re-entered, at the discretion of your program. For the alternate case, the command is 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 ColdFire core 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. Your 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.

# 4.2 Operational Procedure

System power-up and initial operation are described in detail in Section 3, “Initialization and Setup”. This information is repeated here for convenience and to prevent possible damage.

## 4.2.1 System Power-up

- Be sure the power supply is connected properly prior to power-up.
- Make sure the terminal is connected to the TERMINAL (P4) connector.
- Turn power on to the board.
Figure 10 shows the dBUG operational mode.

![Flow Diagram of dBUG Operational Mode](image)

**4.2.2 System Initialization**

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

Hard Reset
DRAM Size: 16M

ColdFire MCF5235 on the M523xEVB Firmware v3b.1a.xx (Built on xxx xx xxxx xx:xx:xx)
Copyright 1995-2004 Freescale Semiconductor, Inc.

Enter 'help' for help.

dBUG>

Other means can be used to re-initialize the M523xEVB firmware, as discussed in the following paragraphs.

**4.2.2.1 External RESET Button**

External RESET (SW6) is the red button. Depressing this button causes all processes to terminate, resets the MCF5235 processor and board logic, and restarts the dBUG firmware. Pressing the RESET button would be the appropriate action if all else fails.
4.2.2.2  ABORT Button

ABORT (SW5) is the button located next to the RESET button. The abort function causes an interrupt of the present processing (a level 7 interrupt on MCF5235) 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 MCF5235 core internal registers are displayed.

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

4.2.2.3  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.

4.3  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 in uppercase or lowercase, depending upon your 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.

4.4 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 27. dBUG Command Summary

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>asm &lt;&lt;addr&gt; stmt&gt;</td>
<td>Assemble</td>
</tr>
<tr>
<td>BC</td>
<td>bc addr1 addr2 length</td>
<td>Block Compare</td>
</tr>
<tr>
<td>BF</td>
<td>bf &lt;width&gt; begin end data &lt;inc&gt;</td>
<td>Block Fill</td>
</tr>
<tr>
<td>BM</td>
<td>bm begin end dest</td>
<td>Block Move</td>
</tr>
<tr>
<td>BR</td>
<td>br addr &lt;-r&gt; &lt;-c count&gt; &lt;-t trigger&gt;</td>
<td>Breakpoint</td>
</tr>
<tr>
<td>BS</td>
<td>bs &lt;width&gt; begin end data</td>
<td>Block Search</td>
</tr>
<tr>
<td>DC</td>
<td>dc value</td>
<td>Data Convert</td>
</tr>
<tr>
<td>DI</td>
<td>di&lt;addr&gt;</td>
<td>Disassemble</td>
</tr>
<tr>
<td>DL</td>
<td>dl &lt;offset&gt;</td>
<td>Download Serial</td>
</tr>
<tr>
<td>DLDBUG</td>
<td>dldbug</td>
<td>Download dBUG</td>
</tr>
<tr>
<td>DN</td>
<td>dn &lt;-c&gt; &lt;-e&gt; &lt;-i&gt; &lt;-s &lt;-o offset&gt;&gt; &lt;filename&gt;</td>
<td>Download Network</td>
</tr>
<tr>
<td>FL</td>
<td>fl erase addr bytes</td>
<td>Flash Utilities</td>
</tr>
<tr>
<td>GO</td>
<td>go &lt;addr&gt;</td>
<td>Execute</td>
</tr>
<tr>
<td>GT</td>
<td>gt addr</td>
<td>Execute To</td>
</tr>
<tr>
<td>HELP</td>
<td>help &lt;command&gt;</td>
<td>Help</td>
</tr>
<tr>
<td>IRD</td>
<td>ird &lt;module.register&gt;</td>
<td>Internal Register Display</td>
</tr>
<tr>
<td>IRM</td>
<td>irm module.register data</td>
<td>Internal Register Modify</td>
</tr>
<tr>
<td>LR</td>
<td>lr&lt;width&gt; addr</td>
<td>Loop Read</td>
</tr>
<tr>
<td>LW</td>
<td>lw&lt;width&gt; addr data</td>
<td>Loop Write</td>
</tr>
<tr>
<td>MD</td>
<td>md&lt;width&gt; &lt;begin&gt; &lt;end&gt;</td>
<td>Memory Display</td>
</tr>
<tr>
<td>MM</td>
<td>mm&lt;width&gt; addr &lt;data&gt;</td>
<td>Memory Modify</td>
</tr>
<tr>
<td>MMAP</td>
<td>mmap</td>
<td>Memory Map Display</td>
</tr>
<tr>
<td>RD</td>
<td>rd &lt;reg&gt;</td>
<td>Register Display</td>
</tr>
<tr>
<td>RM</td>
<td>rm reg data</td>
<td>Register Modify</td>
</tr>
<tr>
<td>RESET</td>
<td>reset</td>
<td></td>
</tr>
<tr>
<td>SD</td>
<td>sd</td>
<td>Stack Dump</td>
</tr>
</tbody>
</table>
Table 27. dBUG Command Summary (continued)

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM</td>
<td>asm &lt;&lt;addr&gt; stmt&gt;</td>
<td>Assemble</td>
</tr>
<tr>
<td>BC</td>
<td>bc addr1 addr2 length</td>
<td>Block Compare</td>
</tr>
<tr>
<td>BF</td>
<td>bf &lt;width&gt; begin end data &lt;inc&gt;</td>
<td>Block Fill</td>
</tr>
<tr>
<td>SET</td>
<td>set &lt;option value&gt;</td>
<td>Set Configurations</td>
</tr>
<tr>
<td>SHOW</td>
<td>show &lt;option&gt;</td>
<td>Show Configurations</td>
</tr>
<tr>
<td>STEP</td>
<td>step</td>
<td>Step (Over)</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>symbol &lt;symb&gt; &lt;-a symb value&gt; &lt;-r symb&gt; -Clls&gt;</td>
<td>Symbol Management</td>
</tr>
<tr>
<td>TRACE</td>
<td>trace &lt;num&gt;</td>
<td>Trace (Into)</td>
</tr>
<tr>
<td>UP</td>
<td>up begin end filename</td>
<td>Upload Memory to File</td>
</tr>
<tr>
<td>VERSION</td>
<td>version</td>
<td>Show Version</td>
</tr>
</tbody>
</table>
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 interactive mode, enter 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 you are 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.

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

In the non-interactive mode, specify 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 0x0001_0000, the command is:

```
asm 10000 nop
```

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

```
asm 400000
```
**Block Compare**

**Usage:** BC addr1 addr2 length

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

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

**Example:**

To verify that the data starting at 0x2_0000 and ending at 0x3_0000 is identical to the data starting at 0x8_0000, the command is:

```
bc 20000 80000 10000
```
**BF Block Fill**

**Usage:** \( \text{BF}<\text{width}> \) begin end data \(<\text{inc}>\)

The BF command fills a contiguous block of memory starting at address begin, stopping at address end, with the value data. \(<\text{Width}>\) modifies the size of the data that is written. If no \(<\text{width}>\) 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 \(<\text{inc}>\) 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 0x0002_0000 and ending at 0x0004_0000 with the value 0x1234, the command is:

\[
\text{bf} \quad 20000 \quad 40000 \quad 1234
\]

To fill a block of memory starting at 0x0002_0000 and ending at 0x0004_0000 with a byte value of 0xAB, the command is:

\[
\text{bf.b} \quad 20000 \quad 40000 \quad \text{AB}
\]

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

\[
\text{bf} \quad \text{bss_start} \quad \text{bss_end} \quad 0
\]

To fill a block of memory starting at 0x0002_0000 and ending at 0x0004_0000 with data that increments by 2 for each \(<\text{width}>\), the command is:

\[
\text{bf} \quad 20000 \quad 40000 \quad 0 \quad 2
\]
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 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 0x0004_0000 and ending at 0x0008_0000 to the location 0x0020_0000, 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 0x0020_0000, the command is:

```
  bm data_start data_end 200000
```

NOTE

Refer to the upuser command for copying code/data into flash memory.
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 is returned to dBUG.

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

```
br _bench -t 3
```

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

To remove all breakpoints, the command is:

```
br -r
```
BS Block Search

Usage: BS<width> begin end data

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

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

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

Examples:

To search for the 16-bit value 0x1234 in the memory block starting at 0x0004_0000 and ending at 0x0008_0000:

```
bs  40000 80000 1234
```

This reads the 16-bit word located at 0x0004_0000 and compares it against the 16-bit value 0x1234. If no match is found, then the address is incremented to 0x0004_0002 and the next 16-bit value is read and compared.

To search for the 32-bit value 0xABCD in the memory block starting at 0x0004_0000 and ending at 0x0008_0000:

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

This reads the 32-bit word located at 0x0004_0000 and compares it against the 32-bit value 0x0000_ABCD. If no match is found, then the address is incremented to 0x0004_0004 and the next 32-bit value is read and compared.
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, 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
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 uses 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 0x0004_0000, the command is:

   di  40000

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

   di _main
### 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

Usage: DL <offset>

Download dBUG

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 prompts you for verification before any actions are taken. If the command is affirmed, the flash is erased and you are prompted to begin sending the new dBUG S-record file. The file should be sent as a text file with no special transfer protocol.

CAUTION

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

Usage: 

\[ \text{DN} \ (<-c> \ <e> \ <i> \ <s> \ <o \ offset> \ <filename>) \]

The DN command downloads code from the network. The DN command manages files that are 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.

- `-c` option indicates a COFF download
- `-e` option indicates an ELF download
- `-i` option indicates an image download
- `-s` indicates an S-record download
- `-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 is 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 that 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 your 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
```
The FL command provides a set of flash utilities that displays information about the flash devices on the EVB, erases a specified range of flash, or erases and programs a specified range of flash.

When issued with no parameters, the FL command displays 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 attempts 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 is adjusted automatically and you are prompted for verification before proceeding.

When the write command is given, the FL command programs the number of bytes specified from src to dest. An erase of this region is first attempted. As with the erase command, if the flash range to be programmed does not start and end on flash sector boundaries, the range is adjusted and you are 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**

**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 that causes control to be handed back to dBUG.

The GO command is repeatable.

Examples:

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

```
go
```

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

```
go _main
```

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

```
go 40000
```
GT

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 that causes control to be handed back to dBUG.

Examples:

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

    gt _bench
HELP

Usage: HELP <command>

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

Examples:

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

    help

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

    help br
IRD

Usage: IRD <module.register>

This command displays the internal registers of different modules inside the MCF5235. 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 *MCF5235 Reference Manual* for more information on these modules and the registers they contain.

Example:

    ird sim.rsr
IRM

Internal Register Modify

Usage: IRM module.register data

This command modifies the contents of the internal registers of different modules inside the MCF5235. 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`
**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 0x2_0000, the command is:

```plaintext
lr.l 20000
```
**LW**

**Usage:** \( \text{LW<width>} \text{ addr data} \)

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

**Examples:**

To continually write the longword data `0x1234_5678` to address `0x2_0000`, the command is:

```
 lw.l 20000 12345678
```

The following command writes `0x78` into memory:

```
 lw.b 20000 12345678
```
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 displays memory up to an address that is 128 beyond the starting address.

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

Examples:

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

```
md 400000
```

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

```
md data_start
```

To display a range of bytes from 0x0004_0000 to 0x0005_0000, the command is:

```
md.b 40000 50000
```

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

```
md.l 40000 50000
```
**MM**

Usage: \( MM <\text{width}> \text{ addr } <\text{data}> \)

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 0x0001_0000 to be 0xFF, the command is:

\[ \text{mm.b } 10000 \text{ FF} \]

To interactively modify memory beginning at 0x0001_0000, the command is:

\[ \text{mm } 10000 \]
**MMAP Memory Map Display**

**Usage:** `mmap`

This command displays the memory map information for the M523xEVB 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 (protected) for dBUG use.

Here is an example of the output from this command:

<table>
<thead>
<tr>
<th>Type</th>
<th>Start</th>
<th>End</th>
<th>Port Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDRAM</td>
<td>0x00000000</td>
<td>0xFFFFFFFF</td>
<td>32-bit</td>
</tr>
<tr>
<td>SRAM (Int)</td>
<td>0x20000000</td>
<td>0x2000FFFF</td>
<td>32-bit</td>
</tr>
<tr>
<td>MRAM (Ext)</td>
<td>0x30000000</td>
<td>0x3007FFFF</td>
<td>32-bit</td>
</tr>
<tr>
<td>IPSBAR</td>
<td>0x40000000</td>
<td>0x7FFFFFFF</td>
<td>32-bit</td>
</tr>
<tr>
<td>Flash (Ext)</td>
<td>0xFFE00000</td>
<td>0xFFFFFFFF</td>
<td>16-bit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protected</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>dBUG Code</td>
<td>0xFFE00000</td>
<td>0xFFE3FFFFF</td>
</tr>
<tr>
<td>dBUG Data</td>
<td>0x00000000</td>
<td>0x0000FFFFF</td>
</tr>
</tbody>
</table>

**Chip Selects**

- CS0 Ext Flash
- CS1 Ext MRAM
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
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 is not written to the register until target code is executed.

Examples:

To change register D0 to contain the value 0x1234, the command is:

```
rm D0 1234
```
RESET

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
Stack Dump

Usage:  SD

The SD command displays a back trace of stack frames. This command is useful after some user code has executed that creates stack frames (i.e., nested function calls). After control is returned to dBUG, the SD command decodes the stack frames and display a trace of the function calls.
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 you and dBUG occur using 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 that contains files accessible via TFTP. Your local network administrator has 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 has 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 has 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 filetype 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 (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

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**

Usage: STEP

The STEP command steps over a subroutine call, rather than tracing every instruction in the subroutine. The command sets a temporary breakpoint one instruction beyond the current program counter and then executes the target code.

The STEP command steps over BSR and JSR instructions. The command works for other instructions as well, but if STEP is used with an instruction that does not return (i.e. BRA) then the temporary breakpoint may never be encountered and dBUG may never regain control.

Examples:

To pass over a subroutine call, the command is:

```
  step
```
SYMBOL

Symbol Name Management

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

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

- -a option adds a symbol name and its value into the symbol table
- -r option removes a symbol name from the table
- -c option clears the entire symbol table
- -l option lists the contents of the symbol table
- -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, by the SYMBOL command or by the disassembler, only use the first 31 characters. Symbol names are case-sensitive.

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

Examples:

To define the symbol main to have the value 0x0004_0000, 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

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

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.

Example:
To upload a portion of SDRAM to a file sdram.bin, the command is:

up 40000 50000 sdram.bin
The VERSION command displays the version information for dBUG. The dBUG version, build number and build date are all given.

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

In this example, v 2b . 1c . 1a

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

Examples:

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

```version```

### 4.5 TRAP #15 Functions

An additional utility within the dBUG firmware is a function called the TRAP 15 handler. This function can be called by your program to use 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.

#### 4.5.1 OUT_CHAR

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

Assembly example:

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

C example:

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

4.5.2 IN_CHAR

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

Assembly example:

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

C example:

int board_in_char (void)
{
    asm (" move.l#0x0010,d0"); /* select the function */
    asm (" trap#15"); /* make the call */
    asm (" move.ld1,d0"); /* put the character in d0 */
}

4.5.3 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 */
}

4.5.4 EXIT_TO_dBUG

This function (function code 0x0000) transfers the control back to the dBUG, by terminating your 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 */
}
```

5 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.

5.1 Required Network Parameters

For performing network downloads, dBUG needs six parameters; four are network-related, and two 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 three 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)
- 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)
```
5.2 Configuring dBUG Network Parameters

After the network parameters have been obtained, the dBUG ROM monitor must be configured. The following commands are used to configure the network parameters.

```plaintext
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:

```plaintext
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 only allow access to files starting at a particular sub-directory. (This is a security feature that 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 are 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:

```plaintext
set filename <filename>
```

When using the Ethernet network for download, 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:

```plaintext
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:

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

Change the default filename and filetype with the commands:

```plaintext
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.
5.3 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 causes 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 that 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.
# M523X Evaluation Board

## Table Of Contents:

<table>
<thead>
<tr>
<th>Hierarchical Interconnects</th>
<th>SHEET 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM Memory</td>
<td>SHEET 3</td>
</tr>
<tr>
<td>Address and Data Bus Buffers</td>
<td>SHEET 4</td>
</tr>
<tr>
<td>Card Interface</td>
<td>SHEET 5</td>
</tr>
<tr>
<td>M523X CPU</td>
<td>SHEET 6</td>
</tr>
<tr>
<td>Debug</td>
<td>SHEET 7</td>
</tr>
<tr>
<td>Ethernet Interface</td>
<td>SHEET 8</td>
</tr>
<tr>
<td>ESD Interface</td>
<td>SHEET 9</td>
</tr>
<tr>
<td>Expansion Connectors</td>
<td>SHEET 10</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>SHEET 11</td>
</tr>
<tr>
<td>Power Supply Unit</td>
<td>SHEET 12</td>
</tr>
<tr>
<td>RESET Configuration and Clocking Circuitry</td>
<td>SHEET 13</td>
</tr>
<tr>
<td>SDRAM Memory</td>
<td>SHEET 14</td>
</tr>
<tr>
<td>Serial I/O Interface</td>
<td>SHEET 15</td>
</tr>
<tr>
<td>USB Interface</td>
<td>SHEET 16</td>
</tr>
</tbody>
</table>

## Revision Information

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Designer</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>02 Mar 04</td>
<td>P. Highton</td>
<td>Provisional release</td>
</tr>
<tr>
<td>A</td>
<td>02 Mar 04</td>
<td>P. Highton</td>
<td>Added Metroworks schematic part number and revision letter</td>
</tr>
<tr>
<td>B</td>
<td>30 Apr 04</td>
<td>L. Anderson</td>
<td>Removed alternative footprint for 90-pin SSOP 32-bit Flash and replaced with 16-bit Flash, corrected R45 pinout and added 10K ohm pull-down resistors to U11 (ethernet transceiver)</td>
</tr>
<tr>
<td>C</td>
<td>27 Jun 04</td>
<td>L. Anderson &amp; P. Highton</td>
<td>Final update including a late change modification of the reset configuration tables. Correction of signals on the RJ-45 connector and addition of pull-down resistors on the ethernet signals</td>
</tr>
<tr>
<td>D</td>
<td>27 Jul 05</td>
<td>L. Anderson</td>
<td>Updated USB page. Jumpers 59 to 62 renumbered.</td>
</tr>
<tr>
<td>A</td>
<td>11 Nov 05</td>
<td>C. Chavez</td>
<td>Updated Revision Schematics Page to reflect RoHS conversion of EVB</td>
</tr>
<tr>
<td>B</td>
<td>03 Mar 07</td>
<td>C. Chavez</td>
<td>Added termination resistors R28, R29, R72 with 22 Ohm resistors in order to pass FCC. Replaced ASRAM with MRAM U1 and U2</td>
</tr>
<tr>
<td>C</td>
<td>06 Aug 07</td>
<td>C. Chavez</td>
<td>Replaced 1M Ohm with 1K Ohm resistors on R1, R3, R67, R68, R69</td>
</tr>
</tbody>
</table>

## Notes:

- All decoupling caps less than 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
Default setting for JP2 is fitted.

Default setting for JP4 is fitted.

Default setting for JP1 is NOT fitted.

CAN Bus Connector
- 9 way D-type (Female)

CAN1 and UART2 share the same DB9 connector on Sheet 15

Default setting for JP3 is NOT fitted.

CAN Channel 0

Transceiver Mode

CAN Termination

Motorola SPS TSPG - TECD ColdFire Group

JP1

1

2

R1

1K

R1

1K

R2

62

C16

0.1uF

C16

0.1uF

C17

0.1uF

C17

0.1uF

U8

SN65HVD230D

U8

SN65HVD230D

D1

GND

VCC

R4

VREF

5

CANL

6

CANH

7

RS

8

JP2

1

2

R1

1K

R1

1K

R2

62

C19

1nF

C19

1nF

U9

SN65HVD230D

U9

SN65HVD230D

D1

GND

VCC

R4

VREF

5

CANL

6

CANH

7

RS

8

JP3

1

2

R3

1K

R3

1K

P1

5

9

4

8

3

7

2

6

1

P1

5

9

4

8

3

7

2

6

1

C18

1nF

C18

1nF

R4

62
Default setting for JP27 is fitted.

NOTE: JP27 is required for some of the legacy BDM cables that connect pins 9 & 25 of the BDM interface internally. More recent cables support both core & I/O voltages. Please check with your BDM cable supplier.

IMPORTANT NOTE: ONLY 3.3V BDM debugging cables can be used with the MCF523x processors.

NOTE: 4.7K pull up resistors are used on signals /BKPT, DSCLK, DSI, DSO & /RESET. A 1K pull up is used for /TA. See page 13 of the schematics.
NOTE: Ethernet Ch. physical addr. default setting is addr. = 1 selected via internal resistor biasing during reset.

Place silk screen LED labels next to D1 thru' D4.

Separate RJ45 connector chassis ground.

NOTE: U11 KS8721BL has an on-chip LDO that derives the +2.5V supply from the +3.3V supply.
Place these zero ohm resistors as close to the junction as possible to the VinX pins on U12, as they represent an RC filter for the ADC inputs.
NOTE: the positive terminal of each power connector must be shown on the silkscreen of the PCB.

JP33 SHOULD BE INSTALLED DURING ASSEMBLY.

DC voltage input range +7 to +14V

NOTE: Diodes prevent excessive difference between 3.3V & 1.5V rails, at power up

NOTE: Schottky Diode prevents excessive difference between 3.3V & 1.5V rails, at power down

JP34 SHOULD BE INSTALLED DURING ASSEMBLY.

1.5V Regulator

VSSA - analog ground for eTPU channels

VSSPLL - filtered ground for CPU PLL module

Filtered ground for plane for ethernet RJ45 connector

1.5V Regulator

Motorola SPS TSPG - TECD ColdFire Group

D15 MBRS340T3

L3 25uH

U20 LM2596S-3.3

C123 0.1uF

VIN1 VOUT 2 ~ON/OFF5 GND3 FB 4 TAB6

D18 MRA4003T3

C121 330uF

U21 LT1086CM

C126 330uF

C124 1nF

C128 10uF TANT.

D14 MBRS340T3

U22 LT1086CM

R70 270

R73 120

R74 22

C127 0.1uF

D19 MRA4003T3

D17 +5V GREEN POWER LED

C125 1000uF

C129 330uF

D16 MBRS340T3

F1 5A Fast blow.

P2 Switchcraft RAPC712

P3

C122 0.1uF

D20 MBRS340T3

C120 330uF

D22 MBRS340T3

C129 330uF
SDRAM Upper 16-bit Word.

SDRAM Lower 16-bit Word.

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 K4S641632EH
- Hyundai HY57V641620HG
- Toshiba TC59S6416CFT
- Infineon HYB39S64160ET
- Winbond W986416DH

Motorola SPS TSPG - TECD ColdFire Group
### Evaluation Board BOM

Table 28. M523xEVB Bill of Materials

<table>
<thead>
<tr>
<th>Item</th>
<th>Qty</th>
<th>Reference</th>
<th>Description</th>
<th>Mfgr</th>
<th>Part Number</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>BARE PWB ; RE Rev X for M523XEVB</td>
<td>RapidPCB</td>
<td>170-20380</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>46</td>
<td>C1, C2, C3, C4, C12, C13, C14, C15, C18, C19, C24, C25, C26, C27, C37, C34, C39, C70, C71, C72, C73, C74, C75, C76, C77, C98, C99, C110, C111, C112, C113, C118, C124, C133, C134, C135, C136, C153, C155, C156, C168, C171, C173, C175, C182, C183</td>
<td>1nF 50V 0805, 5% COG Surface Mount Caps</td>
<td>KOA</td>
<td>AVX 08051C102KAT2A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>77</td>
<td>C5, C6, C7, C8, C9, C10, C11, C16, C17, C28, C29, C30, C31, C33, C35, C36, C38, C40, C42, C43, C44, C46, C47, C48, C49, C52, C53, C55, C57, C78, C79, C80, C81, C82, C83, C84, C85, C100, C101, C114, C115, C116, C117, C119, C120, C122, C123, C127, C130, C137, C138, C139, C140, C141, C142, C143, C144, C145, C146, C147, C148, C149, C150, C151, C152, C154, C156, C158, C166, C169, C172, C174, C176, C184, C185, C186, C187</td>
<td>0.1uF 25V 0805, 10% X7R Surface Mount Caps</td>
<td>KOA</td>
<td>AVX 08055C104KAT2A</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>C20, C21, C22, C23, C180, C181</td>
<td>100pF 50V 0805 NPO or COG Surface Mount Caps</td>
<td>Venkel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>C32, C128</td>
<td>10uF 16V 10% B Case Tant.</td>
<td>AVX</td>
<td>TA016TCM106KBR</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>C41, C45</td>
<td>47uF 16V C Case Tant.</td>
<td>AVX</td>
<td>TPSC476K016R0350</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>C50, C66, C94, C95, C102, C103, C104, C105, C170</td>
<td>10nF 25V X7R 0805 Surface Mount Caps</td>
<td>Venkel</td>
<td>C0805X7R250-103KNE</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>C51, C54, C56</td>
<td>10uF D Case 7343. Tant.</td>
<td>AVX</td>
<td>TPSD106K035R0300</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>C58, C59, C60, C61, C62, C63, C64, C179</td>
<td>2.2nF 25V NPO</td>
<td>Venkel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>C65, C67, C68, C69, C86, C88, C89</td>
<td>0.1uF 25V X7R 0805, 10%</td>
<td>AVX</td>
<td>AVX 08055C104KAT2A</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>C87</td>
<td>2.2uF 10V A Case Tant.</td>
<td>AVX</td>
<td>T494A225K010AS</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>C90, C91, C92, C93, C96, C97, C106, C107, C108, C109</td>
<td>470pF 50V NPO or COG 0805, 5%</td>
<td>Venkel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 28. M523xEVB Bill of Materials (continued)

<table>
<thead>
<tr>
<th>Item</th>
<th>Qty</th>
<th>Reference</th>
<th>Description</th>
<th>Mfr</th>
<th>Part Number</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>3</td>
<td>C121, C126, C129</td>
<td>330uF 10V D Case Tant</td>
<td>AVX</td>
<td>TPSE337K010R0100</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>C125</td>
<td>1000uF 35V</td>
<td>Panasonic</td>
<td>ECA-1VM102 or UVZ1H102MHH</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>3</td>
<td>C159, C165, C167</td>
<td>10uF 16V 10% B Case Tant.</td>
<td>Venkel</td>
<td>TA016TCM106KBR</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>C131, C132</td>
<td>10pF SMD 50V 0805 5%</td>
<td>KOA or Philips</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>C160, C161, C163, C164</td>
<td>47pF SMD 50V 0805 5%</td>
<td>KOA or Philips</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>C162</td>
<td>22nF SMD 50V 0805 5%</td>
<td>KOA or Philips</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>C178, C177</td>
<td>22pF SMD 50V 0805 5%</td>
<td>KOA or Philips</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>12</td>
<td>D1, D2, D3, D4, D6, D7, D9, D11, D14, D17, D33</td>
<td>GREEN LED</td>
<td>Kingbright</td>
<td>AA3528SGC</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>3</td>
<td>D5, D23, D24</td>
<td>RED LED</td>
<td>Kingbright</td>
<td>AA3528SRC</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>D8, D10, D12</td>
<td>YELLOW LED</td>
<td>Kingbright</td>
<td>AA3528YC</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>D13, D15, D16, D20, D22</td>
<td>Shottky Power Diodes</td>
<td>On-Semi</td>
<td>MBRS340T3</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>D18, D19</td>
<td>SMA</td>
<td>On-Semi</td>
<td>MRA4003T3</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>8</td>
<td>D25, D26, D27, D28, D29, D30, D31, D32</td>
<td>BLUE LED</td>
<td>Kingbright</td>
<td>AA3528MBC</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>FB1, FB2, FB3, FB4, FB5, FB6</td>
<td>Ferrite Beads</td>
<td>STEWARD</td>
<td>HI1206T500R-00</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>F1</td>
<td>5A Fast Blow FUSE</td>
<td>Keystone</td>
<td>4527K Fuse Holder by KEYSTONE +0216005.H: Fuse by Littlefuse, 5A, 250V, 5 x 20mm glass</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Qty</td>
<td>Reference</td>
<td>Description</td>
<td>Mfgr</td>
<td>Part Number</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>J1</td>
<td>2 x 13 BDM Connector - Shrouded Headers</td>
<td>Thomas &amp; Betts</td>
<td>609-2627</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>J2</td>
<td>RJ45 Connector</td>
<td>Halo</td>
<td>HFJ11-2450E</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>J3</td>
<td>1 x 12 ADC Header 0.1 Male</td>
<td>Molex</td>
<td>22-10-7128</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>1</td>
<td>J4</td>
<td>2 x 20 shrd. UNI3 Connector Male</td>
<td>Molex</td>
<td>39-26-7405</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>J5</td>
<td>2 x 20 eTPU Header 0.1 Male</td>
<td>Molex</td>
<td>10-89-6404</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>J6</td>
<td>1 x 6 HS/ENCO Header Male</td>
<td>Molex</td>
<td>22-27-2061</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>4</td>
<td>J7, J8, J9, J10</td>
<td>60 way Fine Pitch Surface Mount Connector</td>
<td>AMP</td>
<td>177983-2</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>J11</td>
<td>External Clock Input SMA Connector</td>
<td>AMP</td>
<td>1053378-1</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>J12</td>
<td>1 x 10 QSPI Header Male</td>
<td>Molex</td>
<td>22-10-2101</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>J13</td>
<td>1 x 4 I2C Header 0.1 Male</td>
<td>Molex</td>
<td>22-10-2041</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>J14</td>
<td>USB MiniAB Connector</td>
<td>AMP</td>
<td>440479-1</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>J15</td>
<td>USB &quot;series B&quot; Connector</td>
<td>AMP</td>
<td>787780-1</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>J16</td>
<td>USB &quot;series A&quot; Connector</td>
<td>AMP</td>
<td>787616-1</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Qty</td>
<td>Reference</td>
<td>Description</td>
<td>Mfr</td>
<td>Part Number</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>L1</td>
<td>10uH Inductor</td>
<td>Central Technologies or Delaven</td>
<td>CT1210LSC-100_ or 1210-103J</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>2</td>
<td>L2, L3</td>
<td>Power Inductor Thru Hole</td>
<td>Siemens</td>
<td>882111-B-C24</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>4</td>
<td>P1, P4, P5, P6</td>
<td>DB9 RS232 Port Thru Hole</td>
<td>AMP</td>
<td>747844-3</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>1</td>
<td>P2</td>
<td>2.1mm Barrel Power Connector</td>
<td>Switchcraft</td>
<td>RAPC722</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>2</td>
<td>P3</td>
<td>2-way Bare Wire Power Connector</td>
<td>Augat</td>
<td>2SV-02</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>18</td>
<td>RP1, RP2, RP8, RP9, RP10, RP11, RP12, RP14, RP15, RP16, RP17, RP20, RP21, RP22, RP23, RP24, RP25, RP26</td>
<td>ARV241, 4 x 4.7K ohm resistor pack</td>
<td>KOA or Philips</td>
<td>4 X 4.7K, 4 X 0603</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>RP3</td>
<td>ARV241, 4 x 22 ohm resistor pack</td>
<td>KOA or Philips</td>
<td>4 X 22</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>4</td>
<td>RP4, RP5, RP6, RP7</td>
<td>ARV241, 4 x 51 ohm resistor pack</td>
<td>KOA or Philips</td>
<td>4 X 51</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>1</td>
<td>RP13</td>
<td>ARV241, 4 x 10 K ohm resistor pack</td>
<td>KOA or Philips</td>
<td>4 X 10K</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>2</td>
<td>RP18, RP19</td>
<td>ARV241, 4 x 10 ohm resistor pack</td>
<td>KOA or Philips</td>
<td>4 X 10</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>8</td>
<td>R1, R3, R52, R56, R59, R63, R80, R89</td>
<td>1K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>1K</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>2</td>
<td>R2, R4</td>
<td>62 ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>8</td>
<td>R5, R6, R28, R29, R72, R74, R83, R84, R87, R88</td>
<td>22R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>22.1R, 1%</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>1</td>
<td>R41</td>
<td>SMT POT</td>
<td>Bourns</td>
<td>Bourns 3314J-1 or Spectral Vishay 4G-103</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>7</td>
<td>R7, R14, R76, R81, R82, R90, R91</td>
<td>10K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>10K</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Qty</td>
<td>Reference</td>
<td>Description</td>
<td>Mfgr</td>
<td>Part Number</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>59</td>
<td>4</td>
<td>R8, R9, R10, R11</td>
<td>49.9R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>49.9R, 1%</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>8</td>
<td>R12, R49, R55, R60, R66, R67, R68, R69</td>
<td>4.7K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>4.7K</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>1</td>
<td>R13</td>
<td>6.49K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>6.49K</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>4</td>
<td>R15, R16, R17, R18</td>
<td>220R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>220R</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>8</td>
<td>R19, R20, R21, R22, R23, R24, R25, R32</td>
<td>0 ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>330</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>9</td>
<td>R26, R27, R30, R31, R33, R34, R35, R36, R73</td>
<td>120 ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>2</td>
<td>R37, R93</td>
<td>1M ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>1M</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>1</td>
<td>R38</td>
<td>22.1K ohm Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>22.1K, 1%</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>2</td>
<td>R39</td>
<td>150 ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>11</td>
<td>R42, R43, R44, R45, R46, R47, R48, R50, R70, R75, R78</td>
<td>270R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>270R</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>5</td>
<td>R51, R54, R58, R62, R65</td>
<td>1.8K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>1.8K</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>4</td>
<td>R53, R57, R61, R64</td>
<td>24R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>24.9R, 1%</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>1</td>
<td>R71</td>
<td>560R ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>560R</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>2</td>
<td>R40, R77, R79</td>
<td>100 ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>3</td>
<td>R85, R86. R92</td>
<td>100K ohm 0805 Surface Mount Resistor</td>
<td>KOA or Philips</td>
<td>100K</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>1</td>
<td>SW1</td>
<td>Toggle Switch</td>
<td>Apem</td>
<td>108-2M1T1B1M2QE</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Qty</td>
<td>Reference</td>
<td>Description</td>
<td>Mfr</td>
<td>Part Number</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>75</td>
<td>3</td>
<td>SW2, SW3, SW5</td>
<td>Surface Mount Switch - Black</td>
<td>C &amp; K</td>
<td>KS11R22CQD</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>1</td>
<td>SW4</td>
<td>Slide Switch</td>
<td>Apem</td>
<td>25546NA6 (silver preferred) or 25546NLD (gold plate)</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>1</td>
<td>SW6</td>
<td>Surface Mount Switch - Red</td>
<td>C &amp; K</td>
<td>KS11R23CQD</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>1</td>
<td>SW7</td>
<td>Configuration DIP switch 10 position SPDT</td>
<td>Grayhill</td>
<td>78RB12</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>13</td>
<td>TP1, TP2, TP3, TP4, TP5, TP6, TP7, TP8, TP9, TP10, TP11, TP12, TP13</td>
<td>Surface Mount Test Points</td>
<td>Keystone</td>
<td>5015K</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>U1, U2</td>
<td>MRAM</td>
<td>Freescale</td>
<td>MR2A16ATS35C</td>
<td>A</td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>U3, U4, U6</td>
<td>Buffers</td>
<td>On-Semi</td>
<td>MC74LCX16245DT</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>2</td>
<td>U5, U26</td>
<td>AND Gate</td>
<td>TI</td>
<td>SN74LVC1G11DBVR</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>1</td>
<td>U7</td>
<td>Buffers</td>
<td>On-Semi</td>
<td>MC74LCX245DT</td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>2</td>
<td>U8, U9</td>
<td>CAN Transceivers</td>
<td>TI</td>
<td>SN65HVD230D</td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>1</td>
<td>U10</td>
<td>ColdFire MCF5235 microprocessor</td>
<td>Freescale</td>
<td>MCF5235CVM150</td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>1</td>
<td>U11</td>
<td>Ethernet Transceiver</td>
<td>Micrel</td>
<td>KS8721BL</td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>1</td>
<td>U12</td>
<td>A to D Converter</td>
<td>Analog Devices</td>
<td>AD9728BRU</td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>1</td>
<td>U13</td>
<td>Voltage Regulator</td>
<td>Analog Devices</td>
<td>AD780BR</td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>1</td>
<td>U14</td>
<td>AND Gate</td>
<td>On-Semi</td>
<td>NL17SZ08DFT2</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>1</td>
<td>U15</td>
<td>Comparator</td>
<td>LM393M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>1</td>
<td>U16</td>
<td>2 Gate Inverter</td>
<td>On-Semi</td>
<td>NL27WZ04DFT2</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>1</td>
<td>U17</td>
<td>6 Gate Inverter</td>
<td>SN74HC04D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>1</td>
<td>U18</td>
<td>EXOR Logic Gate</td>
<td>On-Semi</td>
<td>NL27WZ86US</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>1</td>
<td>U19</td>
<td>16-bit flash memory</td>
<td>AMD</td>
<td>Am29PL160CB-65RS</td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>1</td>
<td>U20</td>
<td>3.3V Regulator</td>
<td>National</td>
<td>LM2596S-3.3</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>U21</td>
<td>5V Regulator</td>
<td>National</td>
<td>LM2596S-5</td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>Qty</td>
<td>Reference</td>
<td>Description</td>
<td>Mfgr</td>
<td>Part Number</td>
<td>Notes</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----------</td>
<td>-------------</td>
<td>------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td>97</td>
<td>1</td>
<td>U22</td>
<td>1.5V Regulator</td>
<td>Linear Tech.</td>
<td>LT1086CM</td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>1</td>
<td>U23</td>
<td>OSC. 3.3V 25MHz 4-pin Thru Hole</td>
<td>FOX or Pletronics</td>
<td>H5C-2E3 or F5C-2E3 or P1100-HCV or P1145-HCV</td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>2</td>
<td>U24, U25</td>
<td>Voltage Monitoring uP Supervisory Circuit</td>
<td>Analog Devices</td>
<td>ADM708SAR</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>U27</td>
<td>Buffer</td>
<td>Freescale</td>
<td>MC74LCX541DT</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>2</td>
<td>U28, U29</td>
<td>SDRAM</td>
<td>Micron</td>
<td>MT48LC4M16A2TG75L</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>3</td>
<td>U30, U31, U32</td>
<td>RS232 Transceiver</td>
<td>Maxim</td>
<td>MAX3225CAP or ICL3225CA</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>1</td>
<td>U33</td>
<td>USB OTG controller</td>
<td>Philips</td>
<td>ISP1362BD</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>1</td>
<td>U34</td>
<td>Power Distribution Switch</td>
<td>Micrel</td>
<td>MIC2026-2BM</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>1</td>
<td>U35</td>
<td>32-bit flash memory</td>
<td>AMD or Fujitsu</td>
<td>Am29PL320DB60RWP or MBM29PL3200BE70PBT</td>
<td>A</td>
</tr>
<tr>
<td>106</td>
<td>2</td>
<td>VIA1, VIA2</td>
<td>SMTP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>1</td>
<td>Y1</td>
<td>25MHz Crystal</td>
<td>FOX</td>
<td>FOXS/250F-20</td>
<td>A</td>
</tr>
<tr>
<td>108</td>
<td>1</td>
<td>Y2</td>
<td>12MHz Crystal</td>
<td>FOX</td>
<td>FOXS/120-20</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** A = not populated at assembly.
How to Reach Us:

Home Page:  
www.freescale.com

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

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

Europe, Middle East, and Africa:  
Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 5220080 (French)  
www.freescale.com/support

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

Asia/Pacific:  
Freescale Semiconductor 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

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.

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™ 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. 2007. All rights reserved.