Application Note

AN2157/D
Rev. 3, 05/2002

Programming the MSC8101ADS Flash Memory

MOTOROLA
intelligence everywhere™

digital dna™

Donald Simon and Duberly Mazuelos

CONTENTS

This application note describes how to set up and program the MSC8101ADS board to run an application out of its Flash memory so that the application can execute immediately after the board is powered up or reset. The application is first built using CodeWarrior® to generate an S-record file. Then, an application provided with CodeWarrior software runs on the MSC8101ADS. This application interacts with the HyperTerminal application on a PC (using an RS-232 interface) that sends the S-record file to the MSC8101ADS. Finally, the application running on the MSC8101ADS parses the S-records and loads them into Flash memory.

After describing the general programming procedure, this application note walks you through a simple example that causes two LEDs on the MSC8101ADS board to flash at power-up or reset. This application note assumes you are familiar with the Metrowerks® CodeWarrior development tools for the MSC8101 and the MSC8101ADS.

1 References

- *MSC8101 Reference Manual*
- *MSC8101ADS User's Manual*
- CodeWarrior Flash program and documentation. The Flash program is located in `C:\Program Files\Metrowerks\CodeWarrior\StarCore_Support\flash_programmer_support`.

2 Background

On power-up, the MSC8101 goes through a power-on reset ($\overline{\text{PORESET}}$) sequence followed by either jumping to an address vector (HPE signal = 0) or downloading code through the host interface (HPE signal = 1). This application note assumes the former, and that the address vector is located in Flash memory. To run an application from Flash memory automatically after $\overline{\text{PORESET}}$, three sections must be programmed in Flash memory:

- Hard Reset Configuration Word (HRCW)
- Address table vector
- The application

The MSC8101ADS offers an option to allow the MSC8101 to load the HRCW from the on-board Altera FPGA. When this option is selected, the HRCW need not be programmed in Flash memory. For completeness however, this application note assumes that the HRCW is programmed in Flash memory.

The following hardware is used in the procedure described in this application note:

- *Personal computer (PC)*. Runs CodeWarrior and the HyperTerminal application.
- *Parallel cable and parallel command converter*. Connects the PC to the MSC8101ADS for use by the CodeWarrior debugger.
- *(straight through) Serial cable*. Connects the PC to the MSC8101ADS for use by the HyperTerminal application to send the S-record file to the MSC8101ADS.

The following software tools are used in the procedure outlined in this application note:

- *CodeWarrior, production release 1.5*. Previous versions of CodeWarrior software do not work with this procedure. The default directory in which CodeWarrior is installed is `C:\Program Files\Metrowerks\CodeWarrior`. Other programs and material are stored in subdirectories under this directory. All of these files are placed on the PC hard drive when CodeWarrior code is installed.
- *HyperTerminal*. Application on the PC to allow serial communication with the SMC UART interface on the MSC8101ADS.

The following files are used to generate the S-record file to execute from the MSC8101ADS Flash memory using CodeWarrior software:

- *flash.lcf*. The linker command file for using the MSC8101ADS Flash memory. This file places variable initialization values and code in Flash memory range. The build settings in CodeWarrior software are configured to point to this file. Refer to **Appendix A**.

3 Flash Memory

The MSC8101ADS provides 8 MB of 90 ns Flash memory on an 80-pin SIMM. It is buffered from the 60x-compatible system bus to reduce loading on the bus. The Flash memory is made by Smart Modular Technology (SM73228XG1JHBGO) and is composed of four LH28F016SCT-L95 integrated circuits by Sharp, arranged as 2 M × 32 in a single bank. The Flash memory can be expanded to 16 MB and 32 MB by replacing the SIMM. Each sector in the Flash memory is 256 KB. Sector 0 is the first sector (0xFF800000–0xFF83FFFF), sector 1 is the second sector (0xFF840000–0xFF87FFFF), and so on.

The Flash memory contains the HRCW, the application starting address, and the application. The HRCW and the application address must be programmed at specific locations in the Flash memory. The application can be placed where you deem necessary.

The addresses for the HRCW and address table as programmed on the MSC8101ADS are:

- *0xFF800000*. The starting address of Flash memory used on the MSC8101ADS board. The HRCW contains four bytes (least significant byte first) residing in the following four addresses: 0xFF800000, 0xFF800008, 0xFF800010, and 0xFF800018.
- *0xFE000110*. The starting address of the (vector) address table. This address is mapped to 0xFF800110 in the `8101_Initialization.cfg` file used by default when a CodeWarrior project is loaded on the MSC8101ADS. 0xFF800110 is programmed with the starting address of the application.

The starting address of the application can be placed anywhere, but it is convenient to place it outside sector 0 of the Flash memory so that you can erase the sector containing the HRCW and address table without having to reprogram the application. Note that the starting address of the application is also set in the linker command file (`flash.lcf`).

4 Generating an S-record File

The following steps generate an S-record file from an application that is later loaded into Flash memory.

1. Create a project for the application.

   Using the **Project->Add Files** menu option, create a project within the CodeWarrior directory and include all the relevant files for the application.

2. Set the CodeWarrior build settings.

   Set up the project so that the generated S-records can execute out of Flash memory. Note that the settings are not necessarily used for applications built to execute from internal SRAM memory:

   a. Set up the proper environment and code generation options:

   — **Target** → **StarCore Environment**. Enable Big Memory mode
   — **StarCore Compiler** → **Enterprise Compiler**. Enable Init Variables from ROM

   b. Use the proper linker command file and C start-up file:

   — **Linker** → **Enterprise Linker** → **Start-Up File**. Point to the `flash_crt140b.eln` file

   c. Generate an S-record file:

   — **Target** → **Target Settings** → **Post-Linker**. SC100 ELF to S-Record
   — **Post-Linker** → **SC100 ELF to S-Record**. Enable Long Word Addressability
   — **Post-Linker** → **SC100 ELF to S-Record**. Enter the desired Output File Name for the S-record file

3. Build the application.

   Building the application generates an S-record file. Refer to the `C:\Program Files\Metrowerks\CodeWarrior\CodeWarrior Manuals\PDF\Targeting_Starcore.pdf` file for information on this S-record generation utility.

5 Loading the S-record File

The following steps outline the procedure for programming the Flash memory in the MSC8101ADS with an S-record file. The S-records are sent to the MSC8101ADS for the PC over an RS-232 connection using the HyperTerminal application on the PC. There is also an MSC8101 application running on the MSC8101ADS to service this link with the PC. This application is provided with the CodeWarrior program production release 1.5 (see step 3).

1. Set up the HyperTerminal application.

   The HyperTerminal application is set up with the following parameters:

   — Baud rate: 115200
   — Data bits: 8
   — Stop bits: 1
   — Parity: none
   — Handshaking: XON/XOFF
   — <ASCII Setup…>, Line delay: 5 milliseconds.

   It is important to verify that the PC can support the 115200 baud rate. If another data rate is used, change it here and in the Flash loader application (*calc.h*).

2. Set up the serial connection.

Connect a serial cable between the COM port on the PC and the top RS-232 connector on the MSC8101ADS. This is the physical connection that allows the HyperTerminal application on the PC to communicate with the Flash loader application on the MSC8101ADS.

3. Start up the flash programmer.

   Open the Flash memory programmer project provided with CodeWarrior. The project file is `cflash.mcp` and can be found at `C:\Program Files \Metrowerks\CodeWarrior\StarCore Support\flash_programmer_support`.

   This application calculates the appropriate baud-rate generator (BRG) clock for the MSC8101 CPM UART interface based on the crystal oscillator frequency on the MSC8101ADS, the desired baud rate, and the multiplication factors in the System Mode Clock Register (SCMR). Thus, the mode number (MODCK_H, MODCK) on the MSC8101ADS does not need to be modified. However, the Flash memory loader application needs the crystal frequency used on the MSC8101ADS and the desired baud rate. This baud rate is the same as that used to set up the HyperTerminal application.

   The crystal frequency and baud rate are indicated in the `calc.h` header file included with the Flash memory loader application. This header file contains several define statements, two of which must be set according to the existing crystal frequency and supported baud rate by the serial communication interface. The define statements are as follows, and if any of these defines are changed the project must be rebuilt:

   — `#define XTAL 20000000 /*` External system clock. Currently the MSC8101ADS boards are being shipped with 20 MHz clocks. Some of the older boards have 16.384 (16384000) or 25 MHz (25000000) clocks. */

   — `#define console_baudrate 115200 /*` PC com port baud rate. This may need to be changed depending on what your PC supports. One possibility is 57.6 Kbaud (57600) */

Note: Ensure that the debug switch is turned on in the MSC8101ADS board *(SW10-1: ON)* so that the MSC8101 can go into Debug mode after power-on reset.

   Once the project is built, start and run the application on the MSC8101ADS. The HyperTerminal window on the PC shows the following command interface if everything is set up correctly.

```
MSC8101ADS Flash Loader
print     dump      load_flash sec_erase program_word help sec_addr
-->
```

4. Program the Flash memory.

   Program the HRCW, the address table, and the S-record file into the MSC8101ADS Flash memory, as follows:

   a. Erase the sectors where the HRCW, address table and application are to reside. The application described in this application note uses sector 0 for the HRCW and address table and sector 1 for the application. Each sector contains 256 KB.

Note: The application was placed in a different sector than the HRCW and address table. If the application needs to be modified, sector 0 may not need to be reprogrammed.

   The following commands erases sectors 0 and 1.

```
-->0 sec_erase
-->1 sec_erase
```

   b. Program the HRCW. This step is not required if the system is booting from the Altera gate array on the MSC8101ADS (SW9:7 OFF). Since the 32-bit HRCW is programmed into four separate locations, the following four commands are required to program an HRCW of 0x2C00020A into locations 0x0, 0x8, 0x10 and 0x18 offset from the beginning of the Flash memory:

```
-->h2c000000 hff800000 program_word
```

```
-->h00000000 hff800008 program_word
-->h02000000 hff800010 program_word
-->h0a000000 hff800018 program_word
```

c. Program the starting address of the application in the address table at location 0xFF800110. The address table starts at 0xFE000110. The MSC8101ADS maps this address to 0xFF800110. The following command loads a vector of 0xFF840000:

```
-->hff840000 hff800110 program_word
```

In this example, 0xFF840000 is the location where the address table starts in Flash memory. It must match the starting address of the application defined in the `*.lcf` file (`CodeStart and _ROMStart)`. See **Appendix A**.

d. Program the application into Flash memory with the following command.

```
-->0 load_flash
```

The `0` (zero) option uses the addresses in the S-record to program the Flash memory. Alternatively, if you specify the starting address in Flash memory in place of the `0`, the data is loaded in consecutive locations starting at the specified address (ignoring addresses in the S-record). If there are gaps in the S-record address, these gaps are ignored.

e. Send the S-record application `file (*.s)` to be programmed in the Flash memory to the MSC8101ADS using the **Transfer -> Send Text File** option in the HyperTerminal application.

These steps can be verified by viewing the Flash memory addresses programmed using the dump command on the HyperTerminal window. For details on the MSC8101ADS Flash loader utility, refer to the `C:\Program Files\Metrowerks\ CodeWarrior\CodeWarrior Manuals\PDF\Targeting_Starcore.pdf` file.

6 Switch Settings to Run the Application

Table 1 shows MSC8101ADS switch settings if an application is running from MSC8101 internal SRAM.

Table 1. MSC8101ADS Switch Settings, Application Running from MSC8101 Internal SRAM

Switch	Name	Setting	Description
SW10-1	DBG	ON	The MSC8101 (SC140) is placed in Debug mode immediately after reset.
SW9-7	FCFG	OFF ON	Read the HRCW to come from the Altera gate array. Read the HRCW to come from Flash memory.

Table 2 shows specific MSC8101ADS switch settings if an application is running from Flash memory.

Table 2. MSC8101ADS Switch Settings, Application Running from Flash Memory

Switch	Name	Setting	Description
SW10-1	DBG	OFF	Allows the application to execute the boot code after reset, which in turn jumps to the application loaded in Flash memory.
SW2-5, 6	EE4, EE5	ON, ON	Boot from external Flash memory (as opposed to the HI16 host port).
SW2-1	EE0	ON	Normal processing mode after reset (instead of Debug mode).
SW9-7	FCFG	OFF ON	Read the HRCW to come from the Altera gate array. Read the HRCW to come from Flash memory.

Table 3 shows all the MSC8101ADS switch settings to run the application discussed in **Appendix B** from Flash memory

Table 3. MSC8101ADS Switch Settings

Switch	1	2	3	4	5	6	7	8
SW1	on	off	on	on				
SW2	on	on	on	on	on	on	on	on
SW9	on	on	on	off	on	off	off	off
SW10	off	on	on	on				
SW11	off	off	off	on				

To run an application from the MSC8101ADS Flash memory, perform the following steps:

1. (Optional) Remove the command converter connector from the MSC8101ADS.

   This connection is for the debugger and is not used when an application runs from Flash memory.

2. Set up the MSC8101ADS. Run the application in Flash memory.

   Either press PRESET (SW8) or re-apply power to the board.

7 Application Example

In the example application discussed here, the green and red LEDs (LD10 and LD9, respectively) flash continuously. This application note was tested with the following two files:

- `LEDBlinker.c` (see **Appendix B)**
- `find_IMM.asm` (see **Appendix C)**

`main()` within `LEDBlinker.c` calls `Sinit()`, which is located in `find_IMM.asm`. The purpose of `Sinit()` is to initialize the MSC8101 internal memory map to 0x14710000. Determine what the ISB (Internal Space Base) bits are after reset, and then set the memory map to 0x14710000 in the Internal Memory Map Register (IMMR).

Execute this application out of Flash memory as follows:

1. Generate an S-record file for the application.

   Create a C project for the MSC8101within the CodeWarrior folder and include the two files `LEDblinker.c` and `find_IMM.asm`. This is done using the `Project->Add Files` menu option.

2. Referring to **Section 4,** *Generating an S-record File*, step 2, enter the CodeWarrior settings to select the proper environment and code generation settings, the correct linker command and C start-up file, and to generate an S-record file.

   Set up the **Post-Linker -> SC100 EFL to S-Record: Output File Name** to `LEDblinker.s.` Building the application generates an S-record file called `LEDblinker.s.`

3. Program the MSC8101ADS Flash memory.

   Set up the HyperTerminal application (refer to **Section 5**, step 1) on the PC with the desired parameters (baud rate, data bits, stop bits, parity, handshaking, and ASCII line delay). Also set up a serial cable connection between the RS-232 serial COM port on the PC and the upper RS-232 connector on the MSC8101ADS.

4. Open the Flash programmer project provided with the CodeWarrior software.

Edit the `calc.h` file for the corresponding crystal oscillator frequency on the MSC8101ADS (U18) and the baud rate used with the HyperTerminal application. Build the application if the calc.h file was modified.

5. Start the HyperTerminal application on the PC and then run the Flash programmer application on the MSC8101ADS.

   The following display appears on the HyperTerminal window:

```
MSC8101ADS Flash Loader Utility
print dump load_flash sec_erase program_word help sec_addr
-->
```

6. Erase the first two sectors of the Flash memory.

   Use the following commands:

```
-->0 sec_erase
-->1 sec_erase
```

Note: The HyperTerminal commands can also be run from a script, as shown in **Appendix E**.

7. Verify that the sectors in Flash memory are erased.

   Sector 0 starts at 0xFF800000, which is where the HRCW starts. Sector 1 starts at 0xFF840000, which is the starting address of the LEDblinker application. Use the dump command on the HyperTerminal window, as follows:

```
-->hff800000 dump
```

   This displays the following:

```
Hit return for next sixteen, q to quit
FF800000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
-->
```

   Additional <return> key strokes display more memory locations.

```
FF800010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
FF800020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
FF800030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
-->
```

   Pressing ‘q’ ends the memory display.

8. Verify that the application memory is erased.

```
-->hff840000 dump

Hit return for next sixteen, q to quit
FF840000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
FF840010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
FF840020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   ................
-->
```

9. Program the HRCW with the following commands:

```
-->h2c000000 hff800000 program_word
-->h00000000 hff800008 program_word
-->h02000000 hff800010 program_word
-->h0a000000 hff800018 program_word
```

10. Verify that the HRCW is programmed correctly.

Note that the HRCW is four bytes and is every eighth byte starting at 0xFF800000.

```
-->hff800000 dump

Hit return for next sixteen, q to quit
FF800000 2C  0  0  0 FF FF FF FF  0  0  0  0 FF FF FF FF   ................
FF800010  2  0  0  0 FF FF FF FF  A  0  0  0 FF FF FF FF   ................
```

11. Program the address table entry to 0xFF840000, which is the starting address of the LEDblinker application code.

    ```
    -->hff840000 hff800110 program_word
    ```

    Recall that the start of the application depends on how the linker command file is set up and also on the ISB bits in the HRCW. Thus, it must match the `_CodeStart, _ROMStart` labels in the linker command file. In this application, `flash.lcf` is used and the application is set to 0xff840000. Refer to **Appendix A** listing of the `flash.lcf`.

12. Verify that the address table entry is correctly programmed.

    ```
    -->hff800110 dump

    Press return for next sixteen, q to quit
    FF800110  FF 84 00  0 FF FF FF FF FF FF FF FF FF FF FF FF   ...............
    -->
    ```

13. Program the application.

    ```
    -->0 load_flash
    ```

    Then use the HyperTerminal menu **Transfer -> Send Text File** to point to the `LEDblinker.s` S-record file.

    ```
    A "." is printed for every 20 lines of S-records read. When the load
    finishes, a status message displayed as follows:

    -->0 load_flash...
    Flash programed without error
    -->
    ```

14. Verify that the application is programmed.

    ```
    -->hff840000 dump

    Hit return for next sixteen, q to quit
    FF840000  31 1C 22 44 BF 84 90 C0 90 C0 90 C0 90 C0 90 C0   1."D............
    -->
    ```

15. Run the LEDblinker application from Flash memory.

    Change SW10-1 (DBG) to OFF. Press the PRESET switch (SW8) or reapply power to the MSC8101ADS. The MSC8101ADS LEDs (LD10, LD9) should flash.

8 Running LEDblinker from Internal SRAM

The LEDblinker application listed in **Appendix B** flashes two LEDs on the MSC8101ADS. The amount of time the LEDs stay on and off depends on the CLKIN frequency, mode number, and where the application resides. An application runs much faster from internal SRAM than from external memory.

If the application in **Appendix B** runs out of internal SRAM, the following changes must be made:

- LED on and off times modified in the FlashLed function in both *case* statements. Currently there are example wait times for using Flash memory and internal SRAM in the function.
- In the main function, comment out *Sinit*(). This function initializes the IMMR and disables the watchdog. CodeWarrior version 1.5 has a problem with this function.

Appendix A flash.lcf

```
; for linking to Flash memory

.provide _CodeStart,  0xff840000  ; Sets the code start address
.provide _ROMStart,   0xff840000  ; Sets the ROM start address
.provide _StackStart, 0x4f000     ; Sets the stack start address
                            ; The stack grows upwards.

.provide _TopOfStack, 0x7fe00     ; The highest address to be used
; by the C/C++ run-time.
; By default, this serves as the heap
; start address.
; The heap grows downwards.

.provide _SR_Setting, 0xe4000c    ; The value to set the SR after reset:
; exception mode
; interrupt level 7
; saturation on
; rounding mode: nearest even

.memory 0x20000000, 0x207fffff, "rwx" ; sync dram
.provide _sdram_start, 0x20000000

.memory 0, 0xfffff, "rwx"    ; Start execution at interrupt
.memory 0xff800000, 0xffffffff, "rwx" ; 8MB Flash memory declaration
.reserve _StackStart, _TopOfStack ; Reserve for stack space
.provide FlashBase, 0xff800000

.entry _CodeStart ; this is the value programmed in the Flash boot vector

.org _DataStart
; The following line is used if running code out of internal SRAM
; .segment .data, ".data",".ramsp_0",".default",".bss"
; Notes:
; 1.  .bss is used for uninitialized data
; 2.  S-record generator ignores .bss

.org _ROMStart
.segment .intvec, ".intvec"
.segment .text,".text"
.segment .roinit, ".rom_init"
.segment .rotable, ".init_table"
. Take out the following line if using the debugger and running code
. out of internal SRAM
.segment .data,".data"
```

Appendix B LEDblinker.c

```
/*
file name: LedblinkerOutOfFlash.c
v0.1  drs  15May01  original
v0.2  dm   07June01 clean up
v0.3  drs  30Jan02  clean up
v0.4  drs  01Apr02  Add comments for Sinit()

Description: Flash green and red LED on MSC8101ADS board
(green on, green off, red on, red off, then repeat all)

Notes:
1.  Change loop time in FlashLed if running out of internal
SRAM/Flash.
2.  Take out Sinit() if running code from internal SRAM (ie. using
debugger)
*/

#include "msc8101.h" // memory map registers and locations
#define OFF 0
#define RED 1
#define GREEN 2
#define GP_LED0_PIL 0x02000000
#define GP_LED1_PIL 0x01000000

// function prototypes
void Led( UWord16 );
void FlashLed( UWord16 );
void init( void );

// global variables
UWord32 *BCSR0; // Board Control and Status Register Address Values.
Register bit
UWord32 *BCSR1;
UWord32 *BCSR2;
t_8101IMM *IMM; /* IMM base pointer */

void main(void)
{
       Sinit();        // TAKE OUT IF RUNNING FROM INTERNAL SRAM
                       // Initialize IMMR and disable watchdog
       init();      // initialize variables, pointers, LEDs, ...
       while ( 1 ) {
                 FlashLed( GREEN );
                 FlashLed( RED );
       }
}


/*------------------------------------------------------------------------
--
*
* FUNCTION NAME: init
* DESCRIPTION:  8101 and 8101ADS initialization
*
*/
void init( void ) {
       IMM = (t_8101IMM *)0x14700000; // MSC8101 internal register map
       IMM->memc_regs[1].br = 0x14501801; // Base register 1. Allows access
to BCSR
       IMM->memc_regs[1].or = 0xffff8010; // Option register 1
       BCSR0 = (UWord32 *)0x14500000;// Init Board Control/Status Registers
(BSCR)
       BCSR1 = (UWord32 *)0x14500004;
       BCSR2 = (UWord32 *)0x14500008;
       Led(OFF);                     // Turn On Green & Red LEDs
}


/*-------------------------------------------------------------------------
--
*
* FUNCTION NAME: Led
*
* DESCRIPTION:
*
* Turn On/Off either the Green or Red LED on 8260ADS board.
*
* EXTERNAL EFFECTS:
*
* PARAMETERS:
*
* 0: turns off red and green LEDs
* GREEN: turns green LED on
* RED: turns red LED on
*
* RETURNS: NONE
*
*--------------------------------------------------------------------------
-*/
void Led(UWord16 setting)
{
       switch(setting)
       {
       // Turn red and green LEDs off
       case OFF:
                 *BCSR0 |= (GP_LED0_PIL | GP_LED1_PIL);
                 break;
                 // Turn green LED on
       case GREEN:
                 *BCSR0 &= ~GP_LED0_PIL;
                 break;
                 // Turn red LED on
       case RED:
                 *BCSR0 &= ~GP_LED1_PIL;
                 break;
                 // Turn red LED on to indicate an error
       default:
                 *BCSR0 &= ~GP_LED1_PIL;
                 break;
       }
} /* end Led */

/*-------------------------------------------------------------------------
--
*
```

```
* FUNCTION NAME: FlashLed
*
* DESCRIPTION: This function flashes the Red LED on the 8260 Board.
*
* EXTERNAL EFFECTS: None
*
* PARAMETERS: GREEN or RED
*
* RETURNS: None
*
*--------------------------------------------------------------------------
-*/
void FlashLed(UWord16 setting)
{
UWord32 jj;
switch(setting)
{
       // Flash red LED
       case RED:
       {
                 Led(RED);
                 //for (jj=0; jj < 1000000; jj++); // Wait-use if run from
SRAM
                 for (jj=0; jj < 10000; jj++);  // Wait-use if run from
Flash
                 Led(OFF);
                 //for (jj=0; jj < 1000000; jj++); // Wait-use if run from
SRAM
                 for (jj=0; jj < 10000; jj++);  // Wait-use if run from
Flash
                 break;
       }

       // Flash green LED
       case GREEN:
       {
                 Led(GREEN);
                 //for (jj=0; jj < 1000000; jj++); // Wait-use if run from
SRAM
                 for (jj=0; jj < 10000; jj++);  // Wait-use if run from
Flash
                 Led(OFF);
                 //for (jj=0; jj < 1000000; jj++); // Wait-use if run from
SRAM
                 for (jj=0; jj < 10000; jj++);  // Wait-use if run from
Flash
                 break;
       }

       default:
                 break;
       }
} // end FlashLed
```

Appendix C find_IMM.asm

```
;  write IMM address to IMMR
                    section .text
                    global _Sinit
_Sinit             type func
                   move.l emr,d1
                   extractu #3,#19,d1,d3
                   eor    #4,d3.l
                   cmpeq.w #0,d3
                   bf      st1
                   move.l  #$f0000000,d1
                   bra     stcmp

st1              cmpeq.w #1,d3
                  bf      st2
                          move.l  #$f0f00000,d1
                          bra     stcmp
st2              cmpeq.w #2,d3
                          bf      st3
                           move.l  #$ff000000,d1
                   bra     stcmp
st3               cmpeq.w #3,d3
                   bf      st4
                   move.l  #$fff00000,d1
                   bra     stcmp
st4               cmpeq.w #5,d3
                   bf      st5
                   move.l  #$00f00000,d1
                   bra     stcmp
st5               cmpeq.w #6,d3
                   bf      st6
                   move.l  #$0f000000,d1
                   bra     stcmp
st6              cmpeq.w #7,d3
                   bf      st7
                   move.l  #$0ff00000,d1
                   bra     stcmp
st7              cmpeq.w #4,d3
                   bf      stcmp
                   debug
stcmp            move.l  #$10000,d0
                  add     d0,d1,d1
                  move.l  d1,r0
                  move.l  #$ffffffc3,d0
                  move.l  d0,(r0+4) ; disable watch dog timer
                  move.w  #$1470,d2
                  move.w  d2,(r0+$1a8)
                  move.l  #$14710000,r0
                  move.l  #$14501801,d0 ;init bcr reg address
                  move.l  d0,(r0+$108)
                  move.l  #$ffff8010,d0 ; br mask and machine
                  move.l  d0,(r0+$10c)
                  rts
                  endsec
```

Appendix D Resulting S-Record File

Following is the S-record generated from the LEDblinker application.

```
S0030000FC
S325FF840000311C303CBF8490C090C090C090C090C090C090C090C090C090C090C090C090C04B
S325FF84002090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C037
S325FF84004090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C017
S325FF84006090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F7
S325FF840080311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0CD
S325FF8400A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B7
S325FF8400C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C08D
S325FF8400E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C077
S325FF840100311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C04C
S325FF84012090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C036
S325FF840140311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C00C
S325FF84016090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F6
S325FF840180311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0CC
S325FF8401A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B6
S325FF8401C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C08C
S325FF8401E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C076
S325FF840200311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C04B
S325FF84022090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C035
S325FF840240311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C00B
S325FF84026090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F5
S325FF840280311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0CB
S325FF8402A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B5
S325FF8402C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C08B
S325FF8402E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C075
S325FF840300311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C04A
S325FF84032090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C034
S325FF840340311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C00A
S325FF84036090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F4
S325FF840380311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0CA
S325FF8403A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B4
S325FF8403C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C08A
S325FF8403E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C074
S325FF840400311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C049
S325FF84042090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C033
S325FF840440311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C009
S325FF84046090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F3
S325FF840480311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C9
S325FF8404A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B3
S325FF8404C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C089
S325FF8404E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C073
S325FF840500311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C048
S325FF84052090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C032
S325FF840540311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C008
S325FF84056090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F2
S325FF840580311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C8
S325FF8405A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B2
S325FF8405C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C088
S325FF8405E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C072
S325FF840600311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C047
S325FF84062090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C031
S325FF840640311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C007
S325FF84066090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F1
S325FF840680311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C7
S325FF8406A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B1
S325FF8406C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C087
S325FF8406E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C071
S325FF840700311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C046
S325FF84072090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C030
S325FF840740311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C006
```

```
S325FF84076090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0F0
S325FF840780311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C6
S325FF8407A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0B0
S325FF8407C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C086
S325FF8407E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C070
S325FF840800311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C045
S325FF84082090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02F
S325FF840840311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C005
S325FF84086090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0EF
S325FF840880311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C5
S325FF8408A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AF
S325FF8408C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C085
S325FF8408E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06F
S325FF840900311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C044
S325FF84092090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02E
S325FF840940311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C004
S325FF84096090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0EE
S325FF840980311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C4
S325FF8409A090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AE
S325FF8409C0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C084
S325FF8409E090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06E
S325FF840A00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C043
S325FF840A2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02D
S325FF840A40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C003
S325FF840A6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0ED
S325FF840A80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C3
S325FF840AA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AD
S325FF840AC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C083
S325FF840AE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06D
S325FF840B00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C042
S325FF840B2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02C
S325FF840B40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C002
S325FF840B6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0EC
S325FF840B80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C2
S325FF840BA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AC
S325FF840BC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C082
S325FF840BE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06C
S325FF840C00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C041
S325FF840C2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02B
S325FF840C40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C001
S325FF840C6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0EB
S325FF840C80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C1
S325FF840CA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AB
S325FF840CC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C081
S325FF840CE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06B
S325FF840D00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C040
S325FF840D2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C02A
S325FF840D40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C000
S325FF840D6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0EA
S325FF840D80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0C0
S325FF840DA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0AA
S325FF840DC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C080
S325FF840DE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C06A
S325FF840E00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C03F
S325FF840E2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C029
S325FF840E40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0FF
S325FF840E6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0E9
S325FF840E80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0BF
S325FF840EA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0A9
S325FF840EC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C07F
S325FF840EE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C069
S325FF840F00311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C03E
S325FF840F2090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C028
S325FF840F40311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0FE
S325FF840F6090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0E8
```

```
S325FF840F80311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C0BE
S325FF840FA090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C0A8
S325FF840FC0311C303ABF8490C090C090C090C090C090C090C090C090C090C090C090C090C07E
S325FF840FE090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C090C068
S325FF841000311C30CCBF84081E34D8BF8490C0E7E8331C3438BF849F7C39002000800090C0E5
S325FF8410205099331C3416BF84331C343ABF84331C3440BF84311C30C8BF849F7194C0C08030
S325FF841040C1809CC07550765075D176D1C980CA803E00A00A7750745077D174D1CB80CC8028
S325FF8410603E00A00F7550765075D176D1EDE9EEE93A00A10C775077D1EFE9E8E93680A1003F
S325FF841080E9E9EAE93680A100EBE9ECE93680A100EDE9EEE93680A000EFE9E0E994C0E1E909
S325FF8410A0E2E9E3E99AC02C02800090C02D0280009AC02E02800090C02F0280003503200C9A
S325FF8410C080E4311C3000BF849F799F713818349CBF846C305A9890C02A0A8001EAF1EA611F
S325FF8410E0351C3006BF8459985B985C989F4B90C02103800A92D09DE19D829F4C90C021E39F
S325FF8411009FD792C890828FCF9F719F719F719F713480A0009F639F71C06932CB80D30A03DA
S325FF841120A004D1A02718800E31182000B0002118806CD1A12718800E31182000B0F0211885
S325FF841140805CD1A22718800E31182000BF002118804CD1A32718800E31182000BFF021182B
S325FF841160803CD1A52718800E3100200080F02118802CD1A62718800E310020008F002118E4
S325FF841180801CD1A72718800E310020008FF02118800CD1A4271880069E7030002000800137
S325FF8411A06CD1C84130F83FC3BFFFB0412200947002008 1A838002000947130003801945 08C
S325FF8411C000808108309820 10BFFF0080810C9F7194C096209720E748F003F004311C31E273
S325FF8411E0BF84C1827079351C325CBF847C42F0416461320021008000D442F041C84090C0AF
S325FF8412005990F94199610A06211C800090C05092F0410910A300F0414092311C3282BF84F5
S325FF8412200A06211C800090C05392F34134F93FFFBDFFDD84F3414392311C3282BF840A060A
S325FF841240211C800090C05592F54136F93FFFBEFFDE86F5414592311C3282BF840A06211CAF
S325FF841260800090C05792F7413820A00030F93FFFBEFF3400A800DF80F7414792311C32828B
S325FF841280BF84E76894C0962197219F7194C096209720E748300120009470000221148000 64
S325FF8412A0310021008001D441F042C84090C0E848F84232013801945042900006211480 00EC
S325FF8412C0D4C0F142C94190C0E948F942E944F94233992010BFFF4391340120009450040 2C3
S325FF8412E0211C80003501200494500502212080003601200894500602211880006C30331CB3
S325FF84130031D0BF84E76894C0962197219F71E750F005F006311C331ABF84C84090C0E8613E
S325FF841320F842C98190C0E889351C340CBF84E8FEF8423A00210C800090C0E81AF842509098
S325FF841340F042CB4090C09B61C081331C31D0BF846CB0F144222087107179371C3376BF8454
S325FF841360F8C490C0E841F8442920871090C0E988351C3360BF846C30331C31D0BF846CB060
S325FF841380F144222087107179371C33A4BF84F8C490C0E841F8442920871090C0E988351CFC
S325FF8413A0338EBF84311C3412BF84C082331C31D0BF846CB0F144222087107179371C33D883
S325FF8413C0BF84F8C490C0E841F8442920871090C0E988351C33C2BF846C30331C31D0BF8477
S325FF8413E06CB0F144222087107179371C3406BF84F8C490C0E841F8442920871090C0E9886E
S325FF841400351C33F0BF84311C3412BF84311C3412BF84E7709F71331C3118BF84331C328C91
S325FF841420BF84C082331C330EBF84C081331C330EBF84311C3422BF849F719F7190C090C011
S325FF841440001E3574BF8478612468001A3574BF84842B9AC07861C840391834F0BF84001AD2
S325FF8414603574BF84E8FEE8195190CA4190C09A63001E3574BF84646881DB331C3034BF84AF
S307FF8414809F71D1
S319FF841484FF841206FF84123EFF841220FF841348FF8413AA8A
S325FF841498FF8414CC0000010000000100000000000000240000010 0FF8414840000000CFA
S321FF8414B8000000000000010CFF8414900000000080000000FFFFFFFFFFFFFFFF5B
S325FF8414D80004F0000007FE00000001000007FE000000000000000000000000000000006C
S325FF8414F80000000000000000000000000000000000000000000000000000000000000000004B
S325FF8415180000000000000000000000000000000000000000000000000000000000000000002A
S325FF8415380000000000000000000000000000000000000000000000000000000000000000000A
S325FF841558000000000000000000000000000000000000000000000000000000000000000000EA
S705FF84000077
```

Appendix E Flash Program Commands

The Flash program supports the commands listed in **Table 4**.

Table 4. Flash Program Commands

Command	Syntax	Description	Examples

`print`	`<number><print>`	Prints the number of the result of the last operation.	`256 print :`
			Displays 00000100
			`10 sec_addr print :`
			Displays FC2800000

`dump`	`<address> <dump>`	Displays memory in increments of 16 bytes. Terminate by typing “q”. If any key other than “q” is entered, the next 16 consecutive locations are displayed.	`hfc280000 dump:` Displays 1 line of 16 memory addresses.

`load_flash`	`<number><load_flash>`	Programs Flash memory.	`0 load_flash:`
			Uses the address embedded in the S-record to program Flash memory. If invoked with any other number, it uses that number as the starting address in Flash memory.
			`hfc280000 load_flash:`
			Starts the Flash programming at location 0xFC280000 in Flash memory.

`sec_erase`	`<number> <sec_erase>`	Erases the sector specified by number.	`10 sec_erase:` Erases sector 10 in the Flash memory.
`program_word`	`<number1> <number2> <program_word>`	Programs the data specified by number1 at the address specified by number2 to Flash memory.	`DEADBEEF hfc280000:` Programs DEADBEEF to Flash memory at 0xFC280000.
`help`	`<help>`	Displays available commands.	
`sec_addr`	`<number> <sec_address>`	Returns the Flash sector address specified by number.	`10 sec_addr print :` Prints FC280000
`set_boot`	`<address><set_boot>`	Address vector set_boot programs address at boot table + vector. HRCW data is saved, sector 0 is erased, boot_table is updated and sector 0 is reprogrammed.	

Appendix F Example HyperTerminal Script File

The HRCW and address table example in this application note resides in sector 0 of Flash memory. The application resides in sector 1. Scripts can be used with HyperTerminal to program Flash memory. Erasing sectors are not included in the script because there is a delay when these commands are run, and HyperTerminal does not have a way to delay between sending lines out the next line. Therefore, the following commands can be manually typed into HyperTerminal.

```
0 sec_erase
1 sec_erase
```

Text files can be sent via HyperTerminal to automatically run commands on the MSC8101ADS.

The following script can be saved in a file (ex: hrcwAddrTable.txt) and sent through the HyperTerminal menu **Transfer -> Send Text File**. The first four commands program the HRCW (0x2C00020A).

```
h2c000000 hff800000 program_word
h00000000 hff800008 program_word
h02000000 hff800010 program_word
h0a000000 hff800018 program_word
```

The next command programs the first vector in the address table at address 0xFF800110 to 0xFF840000.

```
hff840000 hff800110 program_word
```

Appendix G Troubleshooting

Table 5. Common Problems

	Description	Solution or Corrective Action
1	Application does not run from reset or power-up.	Check the following: • SW10-1 (DBG) must be turned off. Verify that all switches and jumpers are set correctly. • Voltage levels (5 volts, 3.3 volts, MSC8101 core voltage). • Voltage on F1 (both sides of fuse) should be 5 volts. Check the MSC8101 data sheet and errata for MSC8101 voltage. The MSC8101 voltage is set via RP2 on the MSC8101ADS. • Check that there is a clock (P16: clkout). • Rerun this procedure. • If there is a problem in a new application, ensure that the starting address is set up in the linker command file, the Internal Memory Map Register (IMMR) is properly set up, and any required software structures are pointing to the same address as what is in the IMMR.
2	Application is not running correctly.	Verify that a simple application is working, as in this procedure. A common problem is: • IMMR and/or IMM pointer in software does not match. CodeWarrior release 1.5 initializes the IMMR[0–14] to 0x1470.
3	LEDs are not blinking using the application in this application note.	Run LED (LD17) should be on (green) when the application is running. The application discussed in this application note assumes: • 20 MHz crystal oscillator (U18) for a MSC8101 external system clock. • Application is running out of Flash memory. Running the application out of internal MSC8101 SRAM makes the LEDs appear on, since the application runs much faster out of internal SRAM than out of external Flash memory. The delay time for keeping the LEDs on / off may need to be changed in function FlashLED( ).
4	Multiple backspaces may not work.	The Flash program supplied by Metrowerks may not support multiple backspaces in HyperTerminal. Single backspaces to correct typing errors should be fine.
5	HyperTerminal not outputting.	Serial cable must be connected to the upper RS-232 interface connector. `calc.h` file was not set up with the proper MSC8101ADS crystal oscillator frequency or HyperTerminal baud rate.

NOTES:

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

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

ASIA/PACIFIC:

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

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

http://motorola.com/semiconductors

Information in this document is provided solely to enable system and software implementers to use Motorola 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.

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

MOTOROLA