AN_651X_031

APPLICATION NOTE

MARCH 2006

Boot Loader

Some applications benefit from the ability to download code into the IC with no external tools such as emulators or download boards required. A software application that allows such field software updates is typically called a Boot Loader.

Teridian Semiconductor Corporation has developed a Boot Loader to support field upgrades of the Energy Meter and SmardCard Reader ICs with operational code. When programmed into ICs of the Energy Meter IC family (71M651X, 71M652X, 71M653X or 71M64XX) or into ICs of the SmartCard Reader family (73S12XX), it allows easy field upgrade of customer applications (operational code) via the UART. The Boot Loader only takes up the first 1K or 1.5K bytes of flash memory, depending on which version is used.

The Boot Loader described in this Application Note can be programmed into the ICs even with a low-cost tool, such as the TFP-1 Flash Download Module. Once the Boot Loader is established, further code downloads require only a terminal program, such as HyperTerminal.

This Application Note describes how a Boot Loader can be implemented, how it is deployed and how compatible firmware applications (operational code to be downloaded) can be generated.

The Boot Loader Code

The source code for the Boot Loader is available from TERIDIAN Semiconductor. It is highly recommended to compile the source code using the Keil family of C compilers.

At compile time, the Boot Loader can be configured for:

- IC selection (meter ICs, SmartCard ICs)
- Serial UART selection – determines which UART is used for the code download
- Serial UART bit rate selection of up to 38400 bps

Various targets can be configured by settings bits in the OPTIONS.H header file. In the sample below, the project is configured for a 71M6513:

```
#define M1200   0
#define M6510   1
#define M6511   0
#define M6513   1
#define M6515   0
#define M6520   0
#define M6521F  0
#define M6521B  0
#define M6521D  0
#define M6530   0
#define M6531   0
#define M6532   0
#define M6533   0
#define M6534   0
#define M6403   0
```

Similarly, the UART to be used is defined in OPTIONS.H, as shown in the code segment below:

```
// Serial protocols
#define SERIAL0            1            // Use UART 0 to download.
#define SERIAL1            0            // Use UART 1 to download.
#define TMR0               1            // Use hardware timer0.
#define TMR1               0            // Use hardware timer1.
```

The DIO pins to be used to invoke the Boot Loader are defined as follows:

```
#if M6511
#define PROGRAM (!DIO_16)
#elif M6513
#define PROGRAM (!DIO_2)
#endif
#elif M6520
#define CLK_FREQ 4915200L   // MPU Clock.
#define PROGRAM (DIO_0)
```

DIO_16 and DIO_2 are low-active, while DIO_0 is high-active.

Two versions are available for the Boot Loader: The 1kbyte version without signaling capability and the 1.5kbyte version capable of signaling error conditions via the UART. The 1.5kbyte version may be useful for trouble-shooting configurations and fixtures, since it provides some extend of feedback.

Functionality of the Boot Loader

Location of the Boot Loader

A fixed image of the Boot Loader resides in internal flash in the lower 1K bytes (i.e. The first one (1) or two (2) flash pages) of the device. This means that the code of the Boot Loader is executed after every reset or power-up. If no code download is required, which is generally the case, the Boot Loader simply jumps to the first instruction of the operational code.

The upper 64 bytes of the Boot Loader space are reserved for a one-time write of a serial number or other appropriate identifiers. The serial number is written via the download of an Intel hex data record.

Invocation of the Boot Loader

The Boot Loader is invoked by pulling a DIO pin low while the reset button is released. The default settings are DIO02 (pin 20) for the 71M6513, SEG36/DIO16 (pin 22), and DIO0 (PB, pin 62) for the 71M6521.

On the TERIDIAN 6513 Demo Boards invoking the Boot Loader can easily be done using both the push button on the Debug Board and the RESET button on the Demo Board. The following sequence should be used:

1) Hold down both the push button on the Debug Board and the RESET button on the Demo Board.
2) Release the RESET button, keep holding the pushbutton for about two more seconds.
3) Release the push button

Equally, a jump from the operational code to the code address 0x100 will also invoke the Boot Loader. This can be done with a single line of 'C' code, such as:

```
((void (code *) (void)) 0x0100) ();
```

The Boot Loader will then wait indefinitely to download an Intel hex image of the application via a serial port at 9600 bps, when compiled for the 71M651X ICs, or at 38400 bps 8N1 when compiled for all other ICs. The interface will use the XON/XOFF flow control.

Data Processing by the Boot Loader

Any data received preceding a valid Intel hex record will be ignored. One can abort the download by a simple hard reset.

Once a valid Intel hex record has been received, all flash memory, except the Boot Loader and any protected flash, will be erased.

Erasing all flash memory is a security measure. If partial downloads were allowed, it would enable the loading of a dump facility, exposing the contents of flash memory.

On receipt of an Intel 'end' record, and if no errors were detected, control will pass to the new application (operational code).

If errors did occur, the Boot Loader will again wait for a complete image of the application.

The Boot Loader checks the integrity of each download record and flash write operation to ensure a valid load before running a newly loaded application. If an error is discovered, it will keep attempting to download until a valid load is performed.

Any valid extended Intel hex formatted file is accepted, and the bytes are very efficiently written to flash memory.

Preparing the Application (Operational Code)

Only applications for meter ICs (71M651X, 71M652X, 71M64XX, and 71M653X families) will be presented in this chapter.

Development Tools

Generally, applications will be built using a C compiler, linker, and locator. Since the Demo Code for the Demo Boards of the TERIDIAN meter ICs is generated using the C compiler by Keil (www.keil.com), the graphical user interface of the Keil compiler will be referenced in this chapter. After invoking the Keil compiler executable (uVision REV 2 or REV 3), all settings can be selected with a graphical user interface, as demonstrated by the screen shots shown in this chapter. An easy way to bring up the Keil compiler with the proper settings is to load the UV2 project file provided with the ZIP file that accompanies this Application Note.

Linker Settings

The application will need to be linked with its program space, interrupt vectors and STARTUP.A51 assembly code starting at address 0x400.

Figure 1 shows the settings of the Keil compiler that are to be used for the linker portion of the Keil compiler. In the Options for Target menu the **Target** tab is selected. In this dialog box, the Start/Size of "Off-chip Code Memory" is set as shown in Figure 1.

The total size for the application is 0x400 less than the total FLASH size. In the above case the part had a 32K byte flash (0x8000).

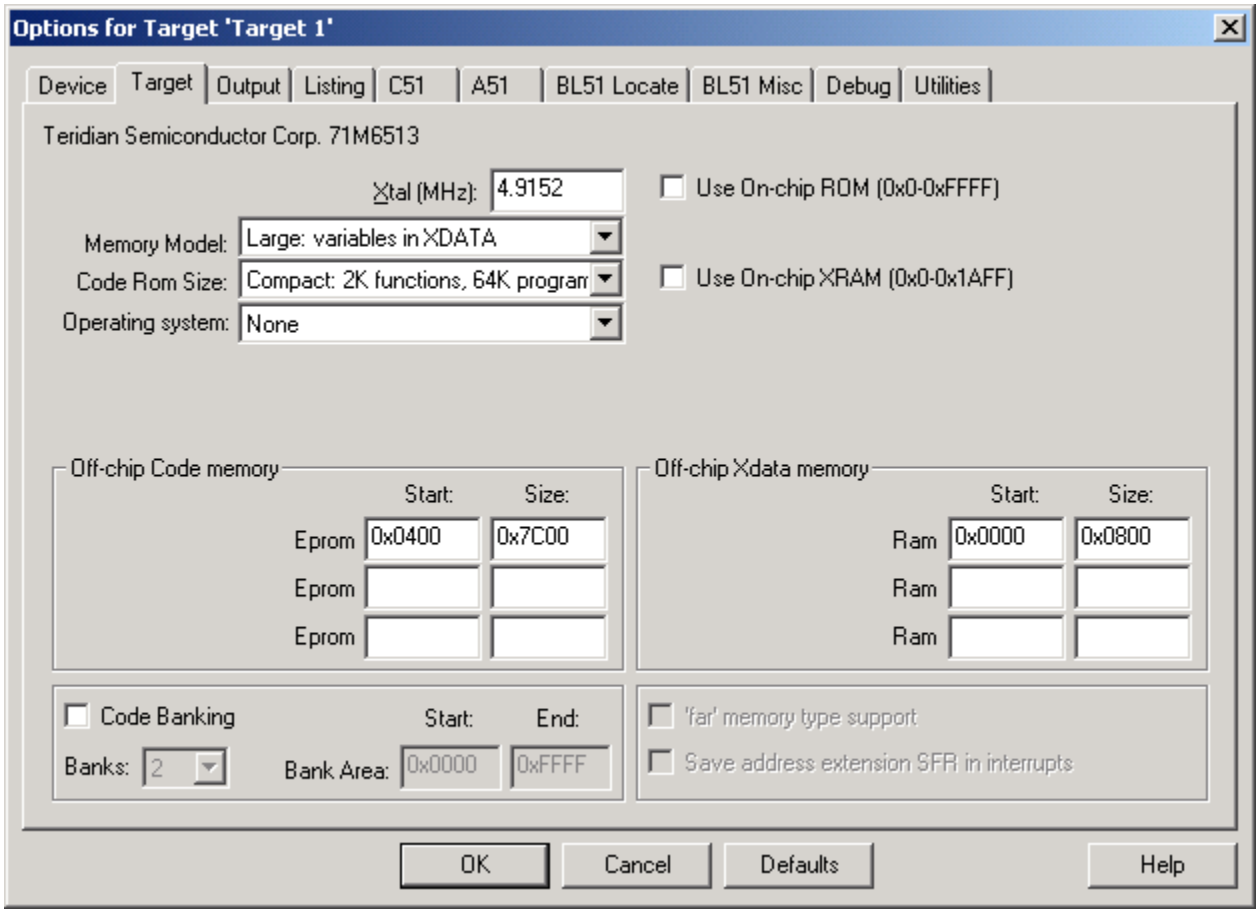

Figure 1: Linker Options for the Keil Compiler

Interrupt Vector Addresses

Figure 2 shows the settings necessary for the interrupt vector addresses. In the Options for Target menu the **C51** tab is selected. In this dialog box, the **Interrupt vectors at address box** is selected and the address 0x400 is entered in the field right next to it.

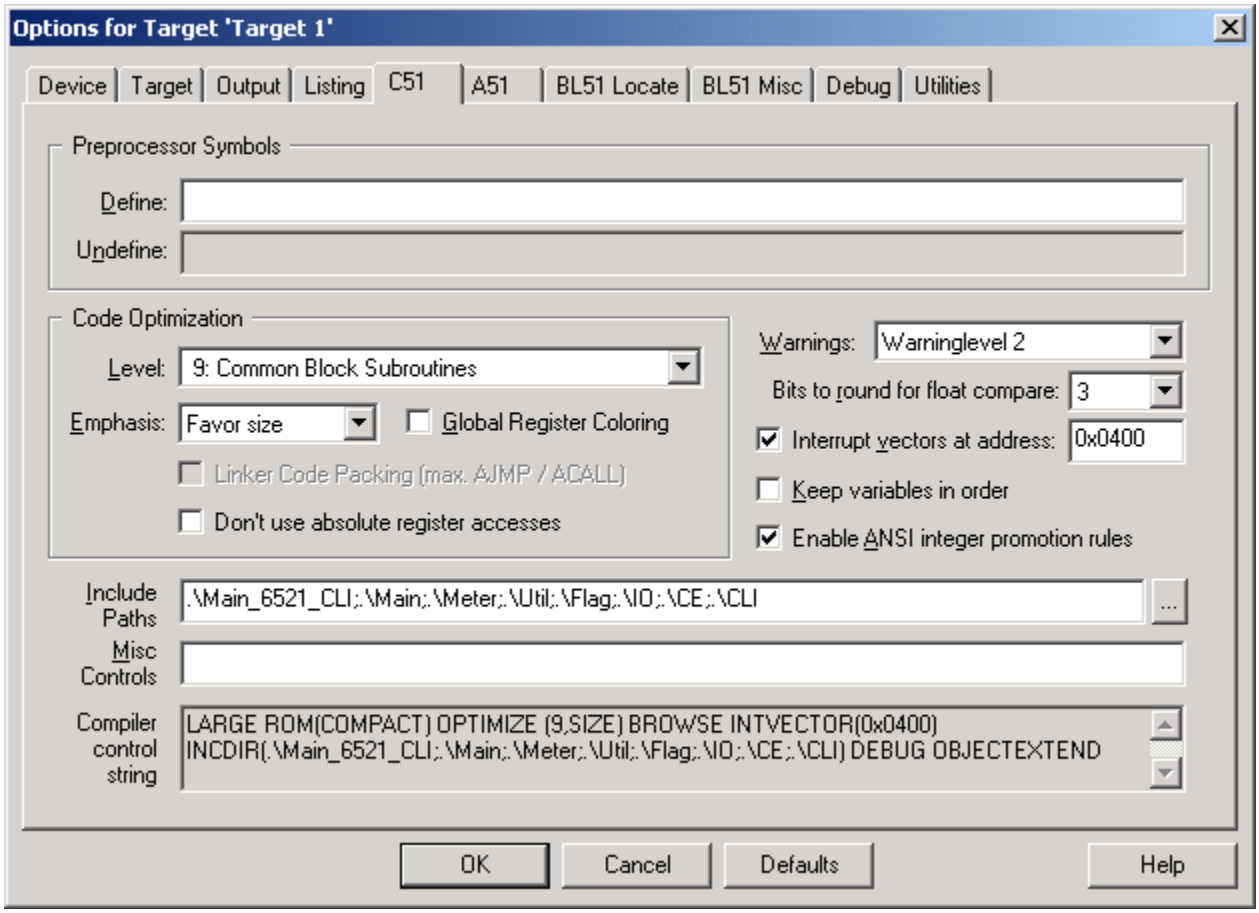

Figure 2: Interrupt Vector Options for the Keil Compiler

Startup File

The STARTUP.A51 assembler file has to be modified in one place. The code segment starting with the label **?C_STARTUP** has to be at 0x0400, as shown in Figure 3.

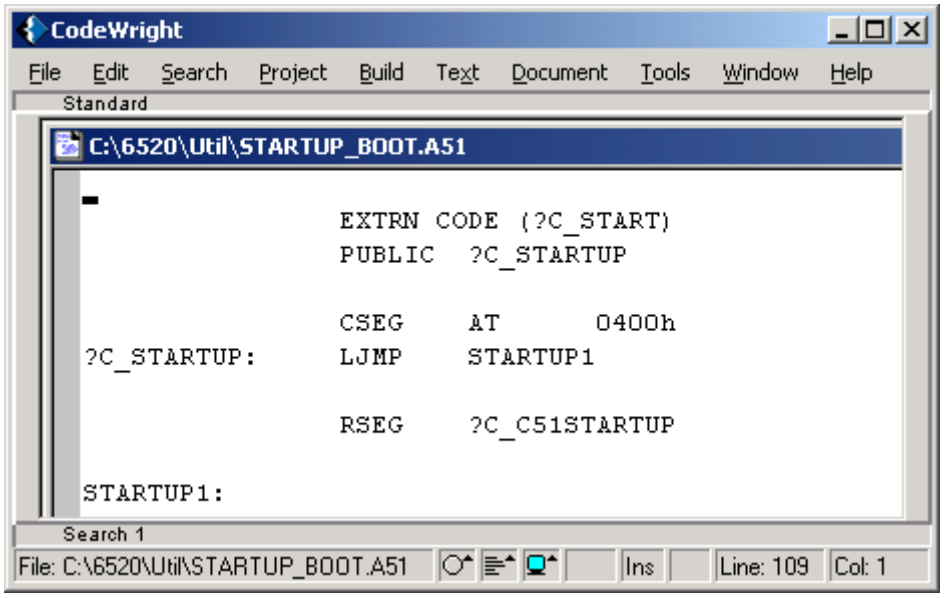

Figure 3: Modification of the STARTUP.A51 File

Locator Settings

When locating applications for the 71M6511 and for the 71M6513, the code object DEAD has to be specified, as shown in Figure 4. This file implements eight bytes containing the pattern 0x00 that will be placed at the beginning of the I/O RAM code space (address 0x2000). This is done so that the sensitive I/O RAM locations 0x2000 through 0x2007 are not overwritten with bit patterns that could disturb the operation of the MPU.

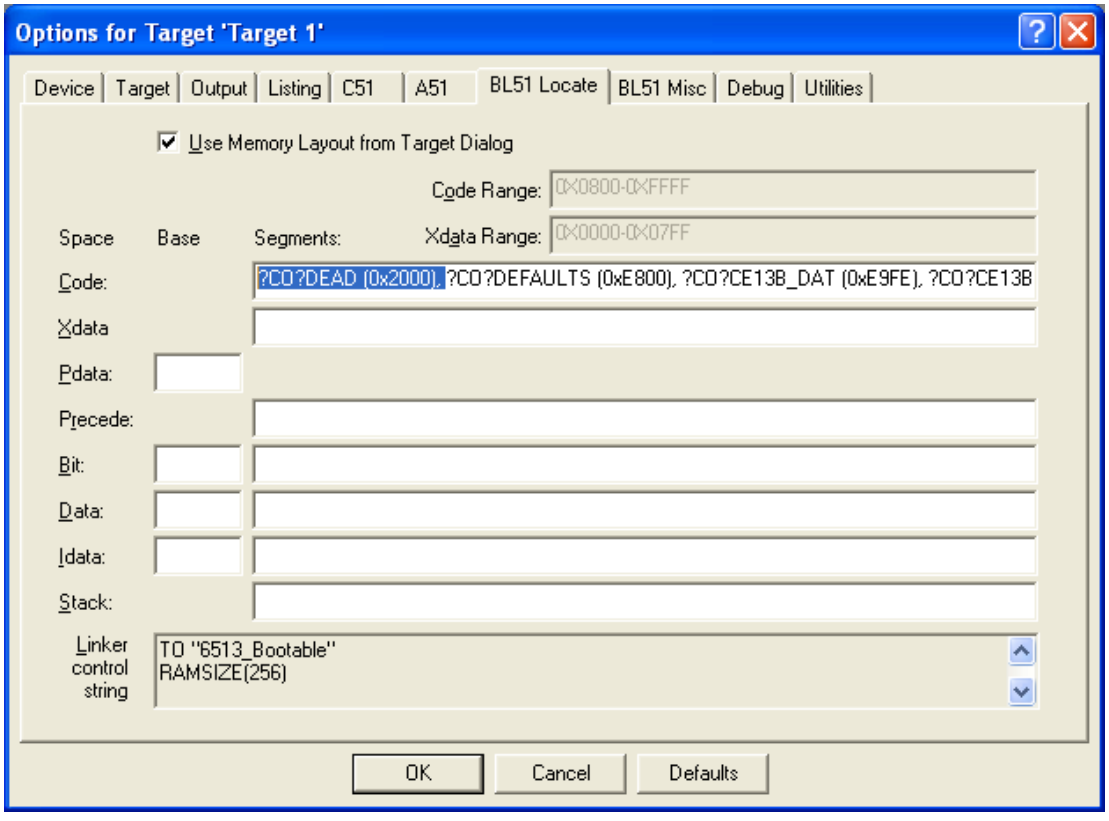

Figure 4: Locator Options

The listing below shows the contents of the file dead.c that implements the code object DEAD referenced above:

```
// File:  DEAD.C
//
#include "options.h"
#include "portable.h"

static U16r dead[] = { 0x0000, 0x0000, 0x0000, 0x0000 };
```

Testing the Boot Loader

Only a few steps are required to test the Boot Loader with an application. We will use the 6513 Demo Board as an example:

1) Use the ADM51 emulator to program the Boot Loader (compiled for 6513) into the flash memory of the Demo Board.
2) Connect the Debug Board to the Demo Board and connect the Debug Board with the serial cable (Digi-Key P/N AE1020-ND) to a PC. The Debug Board must be powered by a separate 5VDC supply.
3) Start HyperTerminal or a similar application capable of sending a file via the serial interface on the PC. The baud rate must be 9,600 and the data format must be 8-N-1. Flow control must be XON/XOFF.
4) Press the RESET button on the Demo Board while holding down the push button on the Debug Board.
5) Release the RESET button while still holding the push button for a few seconds.
6) Release the push button. The IC on the Demo Board should now be waiting for the application to be downloaded. There is no visual indication for this state.
7) Select **Transfer** and **Send Text File** in the HyperTerminal application.
8) In the dialog box of HyperTerminal, select the application file (hex file).
9) After confirming the selection with the **OPEN** button, a symbol (consisting of the \ and / characters) in the HyperTerminal display will “spin”, signaling that data is being transferred to the Demo Board. Data transfer may take several minutes, depending on file size.
10) After the completion of the data transfer, the symbol “1” will appear in the HyperTerminal display. This signals that the transfer has been completed successfully. If a “0” is displayed, the transfer was not successful.
11) Immediately after the data transfer, the downloaded application will start.

The procedure has to be altered for targets other than the 71M6513: For the 71M6511 and 71M6521 Demo Boards, connections to a push button have to be made as follows:

1) 71M6511 4-layer Demo Board: Add the push button at JP14, between pin 1 and pin 2. R102 has to be populated with 10kΩ.
2) 71M6511 2-layer Demo Board: Pin 22 (DIO16) is not connected on this board. A connection has to be added from pin 22 of the socket to a push button.
3) 71M6521 Demo Board: DIO0 is connected to the wake-up push button. No changes are necessary. The wake-up push button can be used in conjunction with the RESET button to invoke the Boot Loader.

TERIDIAN Semiconductor Corp., 6440 Oak Canyon Rd., Irvine, CA 92618

TEL (714) 508-8800, FAX (714) 508-8877, http://www.teridian.com

3/22/06 – REV 1.0