A Maxim Integrated Products Brand

71M651X
Energy Meter IC

APPLICATION NOTE

AN_651X_031

AUGUST 2009

Boot Loader

Some applications benefit from the ability to download code into the IC without requiring external tools such as emulators or download boards. 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 ICs with operational code. When programmed into ICs of the Energy Meter IC family (71M651X, 71M652X), it allows easy field upgrade of customer applications (operational code) via the UART. The boot loader only takes up to the lower 2K bytes of flash memory.

The boot loader described in this Application Note can be programmed into the ICs even with a low-cost tool, such as the TFP-2 Flash Download Module or a Signum ICE. Once the boot loader is programmed into Flash, application 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.

Note: In the past, Teridian has made 1K and 1.5K boot loader codes available; please refer to previous versions of this Application Note for the specifics.

The Boot Loader Code

The source code for the boot loader is available from TERIDIAN Semiconductor. It was developed and built with Keil compiler version 7.0 and later.

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. The object code was built and tested at 9600 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 M12XX   0
#define M651X   1
#define M6511   1
#define M6513   0
#define M6515   0
#define M652X   0
#define M6521   0
#define M6521B  0
#define M6521D  0
#define M653X   0
#define M6532   0
#define M6533   0
```

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

```
// Serial protocols

#define SERIAL0            1            // UART 0 is a valid serial port.
#define SERIAL1            0            // UART 1 is a valid serial port.
#define CLI                1            // command line interface exists...
#define SERIAL0_CLI        1            // command line on serial port 0.
#define SERIAL1_CLI        0            // command lines on serial port 1.
#define TMR0               1            // Use hardware timer0.
#define TMR1               0            // Use hardware timer1.
```

The current Boot Loader code uses DIO_7 pin to invoke the boot loader. This pin is configurable in ‘reg651x.h’ as follow:

#define PROGRAM (!DIO_7)

Note: DIO_7 is an active low pin.

Functionality of the Boot Loader

The boot loader itself is a small program that can be compiled to an Intel hex file using the source files provided by Teridian. After compilation, it can be downloaded to the target board.

After invocation, the boot loader is ready to receive and program the application (operation code). The application has to be built following specific conditions since it must be compatible with the boot loader. Details are explained under “Preparing the Application”.

Location of the boot loader

The boot loader code resides in internal flash memory in the lower 2K bytes (i.e. the first two 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 operation 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 DIO_7 (pin 63) for the 71M6513, SEG36/DIO16 (pin 22), and DIO0 (PB, pin 62) for the 71M6521. Is this still the case – default seems to be DIO7?

On the TERIDIAN D6513 Demo Boards the boot loader can easily be invoked using both the push button on the Debug Board and the RESET button on the Demo Board. The following sequence will be used:

1) Hold down both the push button on the Debug Board and the RESET button on the Demo Board.

   (For push button locations of other products, please see *‘Additional Instructions for 71M6511 and 71M6521’ section*)

2) Release the RESET button while keep holding the pushbutton for about two more seconds.

3) Release the push button.

Equally, a jump from the operation code to the BootLoad function call will also invoke the boot loader. This can be done with a single line of ‘C’ code, such as:

```
((void (code *) (void)) BootLoad) ();
```

The boot loader will then wait indefinitely for 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, to the boot loader will jump to the new application (operation code).

If errors did occur, the boot loader will loop back to 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 (typically a bad checksum or non-Intel hex record) is discovered, it will keep attempting to download until a valid load is performed.

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

Preparing the Application (Operation Code)

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

Development Tools

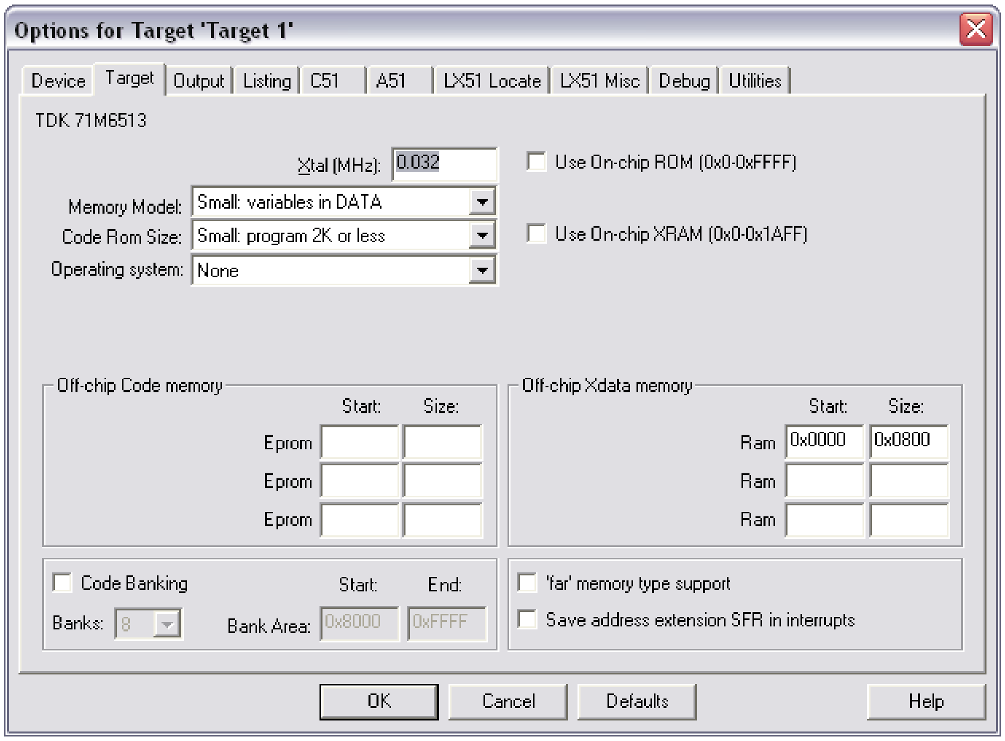
Generally, applications are built using a compiler (C or assembly), linker, and locator. Since the Demo Code for the Demo Boards of the TERIDIAN meter ICs is generated using the Keil compiler (www.keil.com), the graphical user interface of the Keil compiler will be referenced in this chapter. After invoking 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 invoke the Keil compiler with proper settings is to load the UV2 project file provided with the ZIP file that accompanies this Application Note.

Linker Settings

The application build environment needs to be setup such that its program space, interrupt vectors and STARTUP.A51 assembly code are all starting at address 0x0800.

Figure 1 shows the settings using the Keil compiler for this build environment. 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 maximum size of the application will be 2048 bytes less than the total FLASH size. In this example, the intended IC part has a 32K byte flash (0x7FFF).



TDK 71M6513

Xtal (MHz): 0.032

Memory Model: Small: variables in DATA

Code Rom Size: Small: program 2K or less

Operating system: None

☐ Use On-chip ROM (0x0-0xFFFF)

☐ Use On-chip XRAM (0x0-0x1AFF)

Off-chip Code memory

	Start:	Size:
Eprom		
Eprom		
Eprom		

Off-chip Xdata memory

	Start:	Size:
Ram	0x0000	0x0800
Ram		
Ram		

☐ Code Banking Start: End:

Banks: 8 Bank Area: 0x8000 0xFFFF

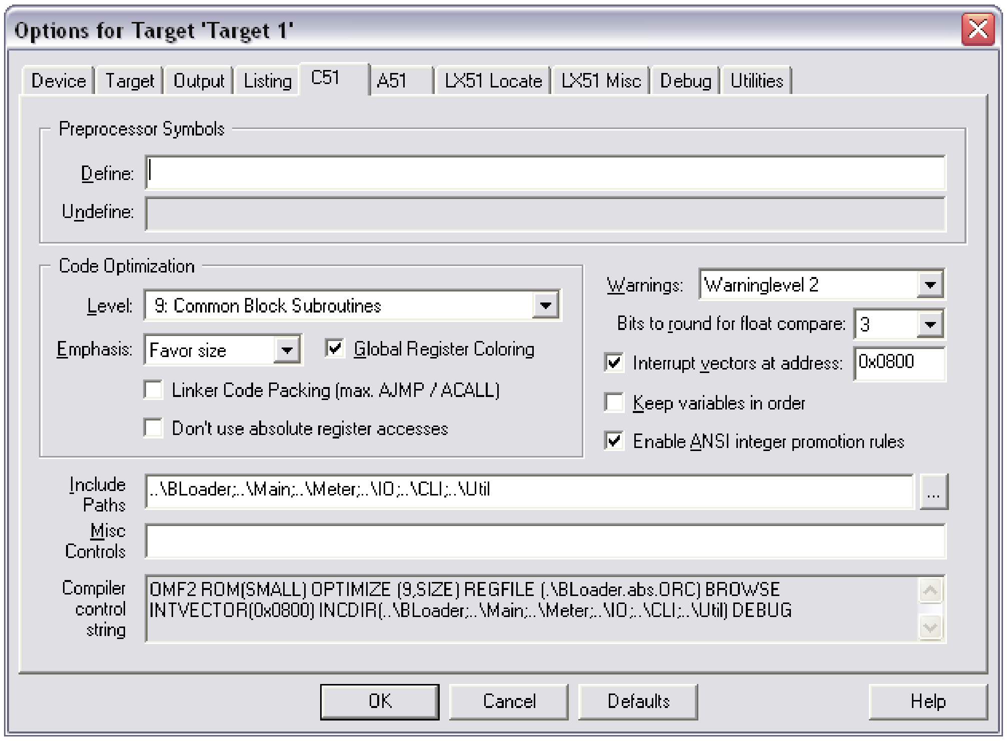
☐ 'far' memory type support

☐ Save address extension SFR in interrupts

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 check box **"Interrupt vectors at address"** is selected and the address 0x0800 is entered in the field right next to it.


Options for Target 'Target 1'
Device | Target | Output | Listing | C51 | A51 | LX51 Locate | LX51 Misc | Debug | Utilities
OK | Cancel | Defaults | Help

Preprocessor Symbols
Define:
Undefine:

Code Optimization
Level: 9: Common Block Subroutines
Emphasis: Favor size [✓] Global Register Coloring
[ ] Linker Code Packing (max. AJMP / ACALL)
[ ] Don't use absolute register accesses

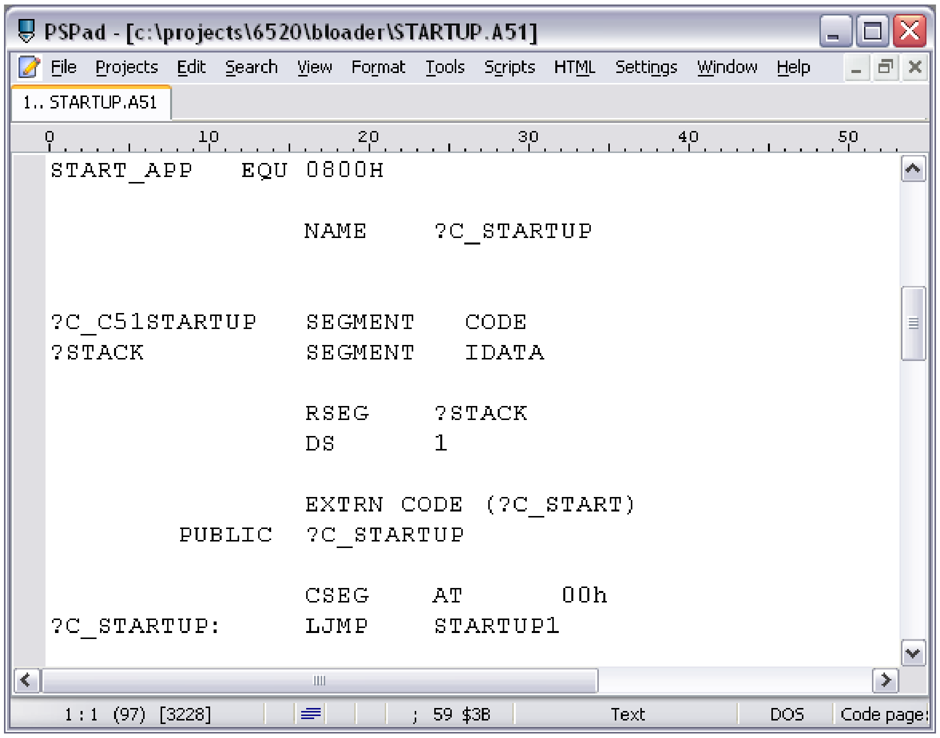
Warnings: Warninglevel 2
Bits to round for float compare: 3
[✓] Interrupt vectors at address: 0x0800
[ ] Keep variables in order
[✓] Enable ANSI integer promotion rules

Include Paths: ..\BLoader;..\Main;..\Meter;..\IO;..\CLI;..\Util
Misc Controls:
Compiler control string: OMF2 ROM(SMALL) OPTIMIZE (9,SIZE) REGFILE (.\BLoader.abs.ORC) BROWSE INTVECTOR(0x0800) INCDIR(..\BLoader;..\Main;..\Meter;..\IO;..\CLI;..\Util) DEBUG

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 0x0800, as shown in Figure 3.


PSPad - [c:\projects\6520\bloader\STARTUP.A51]
File Projects Edit Search View Format Tools Scripts HTML Settings Window Help
1.. STARTUP.A51
1 : 1 (97) [3228] ; 59 $3B Text DOS Code page:

```
START_APP    EQU  0800H

                 NAME     ?C_STARTUP


?C_C51STARTUP    SEGMENT   CODE
?STACK           SEGMENT   IDATA

                 RSEG     ?STACK
                 DS       1

                 EXTRN CODE (?C_START)
        PUBLIC   ?C_STARTUP

                 CSEG     AT       00h
?C_STARTUP:      LJMP     STARTUP1
```

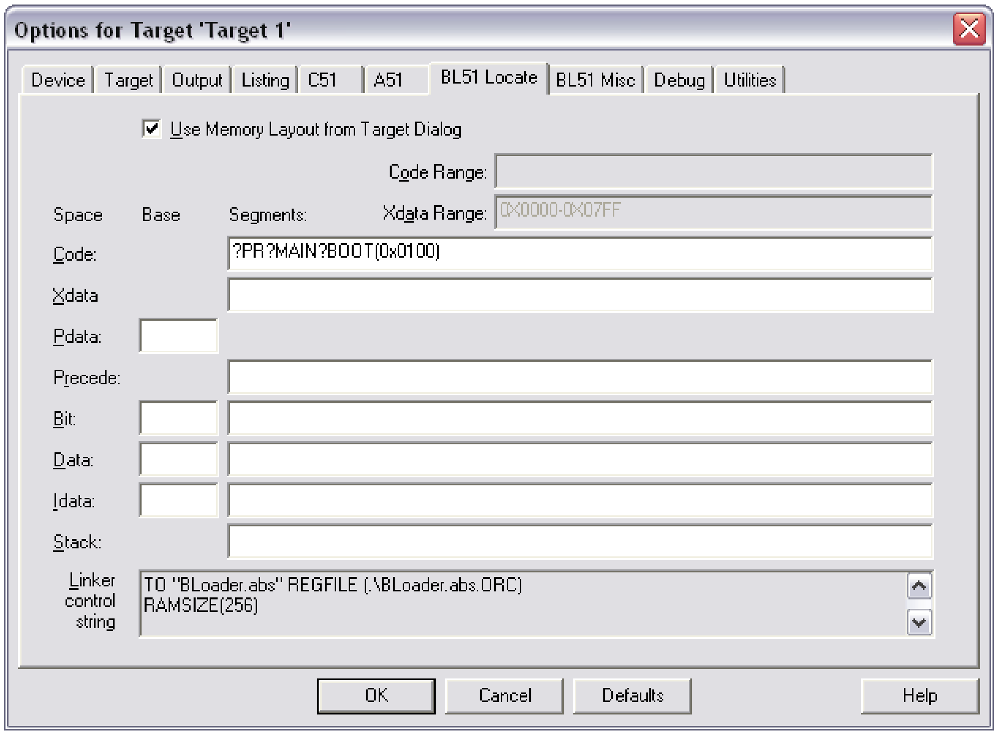
Figure 3: Modification of the STARTUP.A51 File

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 (or more, in some cases) containing the pattern 0x00 that will be placed at the beginning of the I/O RAM 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.

Other flash locations that have to be excluded for the 71M6511 and 71M6513 (not currently implemented in code “DEAD”):

- If UART1 (optical port) is used for the download, it is important to maintain I/O RAM address 0x2008, which contains the bit register *OPT_TXDIS*. Since UART1 is disabled when *OPT_TXDIS* is set high, address 0x2008 should not be written to.
- If the pins DIO4 or DIO5 of the I2C interface are directly tied to external voltages, it is important to maintain I/O RAM address 0x2008, which contains the bit registers *DIO_PW* and *DIO_PV*. Since hardware contention could result if *DIO_PW* or *DIO_PV* is set high, address 0x2008 should not be written to.
- I/O RAM address 0x2070 is also critical, since it contains the *SSI_EN* bit, which, when overwritten by flash code, may enable the SSI interface, which disables the LCD segment pins SEG3 through SEG6.



[✓] Use Memory Layout from Target Dialog

Code Range:

Xdata Range: 0X0000-0X07FF

Code: ?PR?MAIN?BOOT(0x0100)

Xdata:

Pdata:

Precede:

Bit:

Data:

Idata:

Stack:

Linker control string: TO "BLoader.abs" REGFILE (.\BLoader.abs.ORC) RAMSIZE(256)

Figure 4: Locator Options

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 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.

Additional Instructions for 71M6511 and 71M6521

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.

Revision History

Revision	Date	Description
Rev. 1.0	3/22/2006	First publication.
Rev. 1.1	9/16/2006	Added reference to “Additional Instructions” chapter in section “Invocation of boot loader”. Restricted applicability to 71M651X and 71M652X IC families.
Rev. 1.2	9/16/2006	Removed 6515H from IC selection in OPTIONS.H.
Rev. 1.3	11/03/2008	Added cautionary notes for usage of I/O RAM address 0x2008. Reduced size of some figures. Added Revision History table.
Rev. 1.4	03/11/2009	Added I/O RAM address 0x2070 to list of addresses to be excluded for 71M6511 and 71M6513. Added explanation under “Functionality of the boot loader”. Re-moved references to Teridian SmartCard products.
Rev. 2.0	08/20/2009	Adopted version 2K boot loader (fixed 64k flash erase for 71M6513)