Keywords: energy meter, Teridian, SoIC, flash

APPLICATION NOTE 5539

Programming Meter ICs Guide

Jan 08, 2013

Abstract: This document describes the tools, procedures, and precautions to be used when flash programming Maxim metering ICs in prototyping and production.

This application note describes interfaces, tools, precautions, and processes that are to be used for programming the flash memories of the Maxim Integrated's 71M6511/71M6511H, 71M6513/71M6513H, 71M6531D/F, 71M6532D/F, 71M6533/G/H, 71M6534/H, 71M6541D/F/G, 71M6542F/G, 71M6543F/H, and 71M6545/H energy meter ICs.

In addition, recommendations are given for the design of meter boards that ensure that program code can be erased and reloaded under all circumstances.

Programming Tools

The simplest approach to programming Maxim meter ICs is to use one of the existing tools. Programming tools currently available are:

- TFP2 Flash Programmer
- ADM51 in-circuit emulator (ICE)
- Third-party programmers

The TFP2 Flash Programmer

The TFP2 is available from distributors such as Digi-Key or Mouser. It is capable of programming the target IC in the target circuit. The operation is manual (one presses a button to initiate the programming process), but can also be controlled by automatic test equipment (ATE). You can combine several TFP2 devices in parallel to achieve higher throughput.

The target code can be loaded into the TFP2 using a serial cable and HyperTerminal® (or any other terminal) program. The operational code for the TFP2 can be field-upgraded. The latest firmware for the TFP2 at the time of editing this document is revision 1.53.

The target-interface cable of the TFP2 can be a simple twisted cable terminated with a 0.1" header connector. In addition, the TFP2 provides the TYCO/AMP 10x2 high-density connector compatible with the connector used by the ADM51 ICE. This connector includes access to the ICE_E signal (used for the 71M653x ICs) on pin 2.

Simple programmers can be implemented using the TFP2 and an external customer-supplied printed circuit board (PCB) with a socket that accommodates the target IC. See the Required Hardware Adjustments section for a description of the necessary pin connections.

Note: The TFP2 requires all records to be in sequential (ascending) order. Maxim provides a utility (CHKSUM.EXE) that preprocesses Intel® hex files generated by the Keil® compiler/linker for use with the TFP2. This utility is provided with the TFP2.

ICE (Signum Systems Model ADM51)

The ADM51 is available from Signum Systems. It is primarily used for ICE, i.e., a development tool, but it can also be used to flash-program small quantities of metering ICs.

Third-Party Programmers

Single IC, multiple, and high-volume programmers with handlers and feeders, are currently being sold and supported by BPM Microsystems.

Discontinued Programming Tools

Please note that the following tools are discontinued and no longer supported by Maxim:

- TFP1—the predecessor of the TFP2, allowing in-system programming. The TFP2 covers the full functionality of the TFP1.
- FDBM—a simple board that could be used for in-system programming. The FDBM required a graphical user interface (GUI) running on a Windows® PC.
- TGP1—this gang programmer is being replaced by third-party products from well-known manufacturers of programming devices (see note above).

Hardware Interface for Programming

The 71M65xx are programmed via their ICE interface. The signals shown in **Table 1** are necessary to communicate between the flash programmer and the DUT. See the individual data sheets for the location of the ICE interface pins.

Table 1. Flash Interface Signals

Signal	Direction	Function
E_TCLK	Output from DUT	Data clock (5MHz)
E_RXTX	Bidirectional	Data input/output
E_RST	Bidirectional	Flash programmer reset (active-low)
ICE_E	Input to DUT	Enables the ICE interface (71M653x only)

*The E_RST signal should only be driven by the flash programmer when enabling the interface signals.

**The flash programmer must release E_RST at all other times.

***The programming protocol is proprietary to Signum Systems.

Required Hardware Adjustments

Besides establishing IC power supply and ground, the following pins have to be taken care of to successfully flash program the meter ICs:

V1:	V1 must be stable and above the V_{BIAS} threshold (see the data sheet for the value of V_{BIAS}). For the 71M651x series, V1 must be tied to 3.3 V_{DC} to deactivate the hardware watchdog timer. For the 71M653x series, V1 should be above V_{BIAS}. For these devices, activating ICE_E takes care of the watchdog timer.
ICE_E:	This pin is only available on the 71M653x devices. This pin has to be at 3.3 V_{DC} during the programming operation.
XIN/XOUT:	The meter ICs generate the E_TCLK signal from the 32kHz signal generated by the internal clock oscillator in conjunction with the external crystal and two load capacitors. See the data sheet for appropriate crystal and capacitor values.
V_{BIAS}:	This pin should have a 0.1µF capacitor to ground.
V2P5:	This pin should have a 0.1µF capacitor to ground.
RESET(Z):	The low-active RESETZ pin (71M651x) should be tied to 3.3 V_{DC}. The high-active RESET pin (71M653x) should be tied to ground.

All other pins may float.

Special Cases

Virgin devices (all flash locations are 0xFF) or devices that have been flash erased require no special precautions. All solutions may contain a short test pattern that is applied during ATE testing at the factory. These can be treated as unprogrammed parts.

A few special cases apply under the circumstances listed below:

- The SECURE bit in the SFR memory is set.
- Target code sets the SECURE bit in the SFR memory.
- The ECK_DIS bit in I/O RAM memory range is set, disabling the E_TCLK output.
- The part is programmed with valid or partial operational firmware, and the on-chip compute engine (CE) is active, preventing flash access (71M653x only).

These special cases will be discussed in the following sections.

The SECURE Bit

The SECURE bit prevents access to the code image. If the SECURE bit is set during the preboot phase of the IC, there is no hardware measure that defeats it. The only way to reset the SECURE bit is to mass erase all flash memory, followed by a reset.

It is important to note that the SECURE bit is set by the MPU executing the target code (see **Figure 1** for a code example), not by the programmer. The SECURE bit protects the customer's IP from unwanted access, but requires a few additional steps from the programmer when the IC needs to be programmed/verified or reprogrammed.

```
STARTUP1:
                CLR     0xA8^7          ; Disable interrupts
                MOV     0B2h,#40h       ; Set SECURE bit.
                MOV     0E8h,#FFh       ; Refresh WDT
```

Figure 1. Assembler code activating the SECURE bit.

This is how flash security works: When the IC powers up, the MPU's PC is reset to 0x0000 (reset vector) and starts executing the preboot code. This is a block of code consisting of the first 60 MPU cycles located at address 0x0000, during which the ICE interface is disabled. Startup code can set the SECURE bit in the SFR space during the preboot cycle to enable flash security. Since SECURE can only be set but not reset (this bit can only be reset by a hardware reset), and ICE access is impossible during the preboot cycle, external hardware has no way to access flash memory. The SECURE bit is part of a register in the SFR space of the on-chip MPU. The individual IC data sheets contain information about the location of the SECURE bit.

ADM51 ICE

In the user interface of the emulator, a target IC with the SECURE bit set will be indicated as shown in **Figure 2**. The user has the option of erasing the target IC flash and then resetting the IC, which is best achieved by removing the board power. For systems containing a battery, the battery must be briefly disconnected, or if the reset signal is accessible, it can be activated to force the target IC into reset.

After this, the target IC comes up as a regular erased part in the emulator user interface.

TFP2

The TFP2 will issue a simple status message on its terminal interface indicating that the SECURE bit in the target IC is set.

The TFP2 then continues with a flash erase operation, followed by a flash programming operation, with no action required by the user.

If the target is a 71M653x and the ICE_E signal of the target IC is accessible on the programming interface, the sequence is as follows:

- The TFP2 erases the target flash memory.
- The TFP2 releases the ICE_E signal. This will cause a target reset if the watchdog timer of the target IC is enabled.

- The TFP2 now programs the target IC.

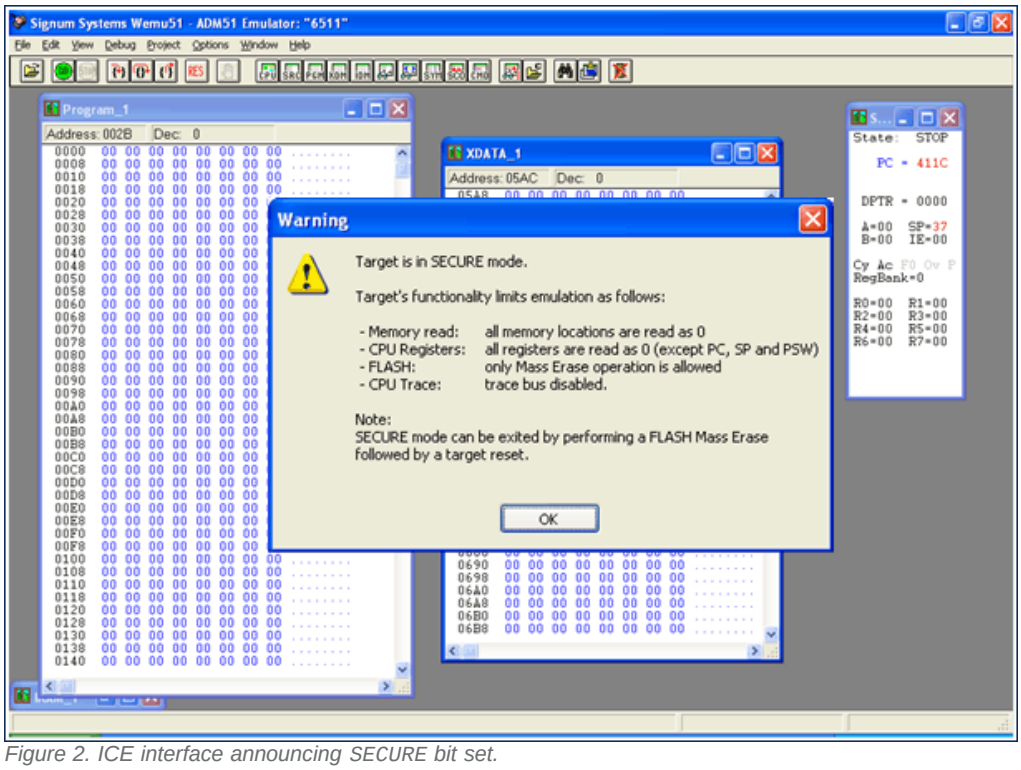


Figure 2. ICE interface announcing SECURE bit set.

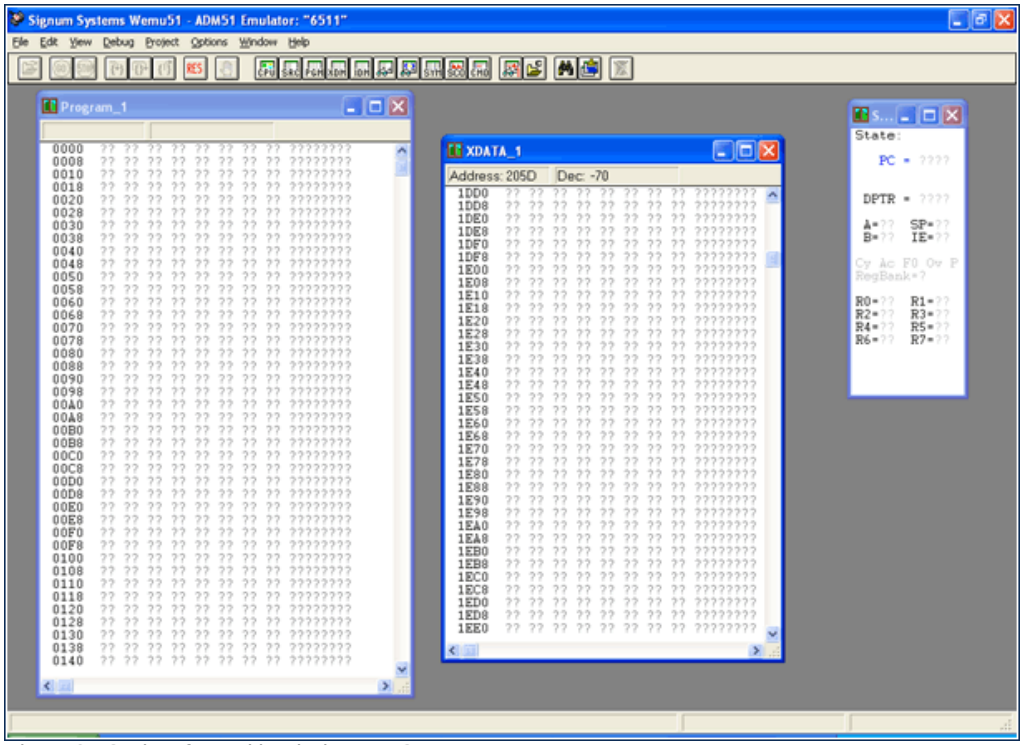


Figure 3. ICE interface with missing E_TCLK.

Programming ICs with Target Code that Sets the SECURE Bit

There is no difference between programming targets with regular code and programming targets with code that sets the SECURE bit. However, verifying code containing instructions that set the SECURE bit requires special care. This is because once the code is allowed to execute, no access to the flash memory is possible.

ADM51 ICE Emulator and TFP2

The following method is used to verify the contents of the target flash memory:

- Once the MPU starts executing the just programmed code, no access to read or verify the flash will be possible.
- The verify operation is performed by halting the target IC, preventing the IC from executing the code.
- Once the target flash is verified and code starts executing, no further verification will be possible.

This process is transparent to the user.

Programming ICs when the ECK_DIS Bit Is Set

This case can present a challenge since the E_TCLK signal is essential for the function of the programming interface.

ADM51 ICE

With the target emulator clock missing, the user interface of the ADM51 (WEMU51) will generate the display shown in **Figure 3**. Depending on how early in the code the ECK_DIS bit is set, repeated resets of the target may give the ADM51 a chance to halt the target before it has a chance to set the ECK_DIS bit. Once this has been achieved, the user interface will appear normal and the user has the opportunity to erase the target flash memory.

TFP2

The TFP2 can react very fast to activity on the target programmer interface. A disabled E_TCLK signal is usually not a problem for the TFP2.

Programming ICs Containing Partial or Full Operational Code

In the 71M653x ICs, the on-chip CE reads its instructions from the flash memory that is shared with the MPU. An active CE can block access to the flash memory for external equipment. The control bit that enables and disables the CE is located at I/O RAM address 0x2000, bit 4.

ADM51 ICE

A typical screen shot of the ADM51 user interface (WEMU51) is shown in **Figure 4**. The "XDATA_1" window shows the contents of the I/O RAM addresses 0x1FF0 to 0x2137. The area highlighted in yellow displays addresses that contain actual I/O RAM hardware registers. The register at 0x2000 contains the value 0xB0.

Before erasing or programming the target flash memory, the value 0x00 must be entered at 0x2000. This will stop the CE and prepare the target IC for programming.

TFP2

The TFP2 takes care of the CE automatically. No user inputs are required.

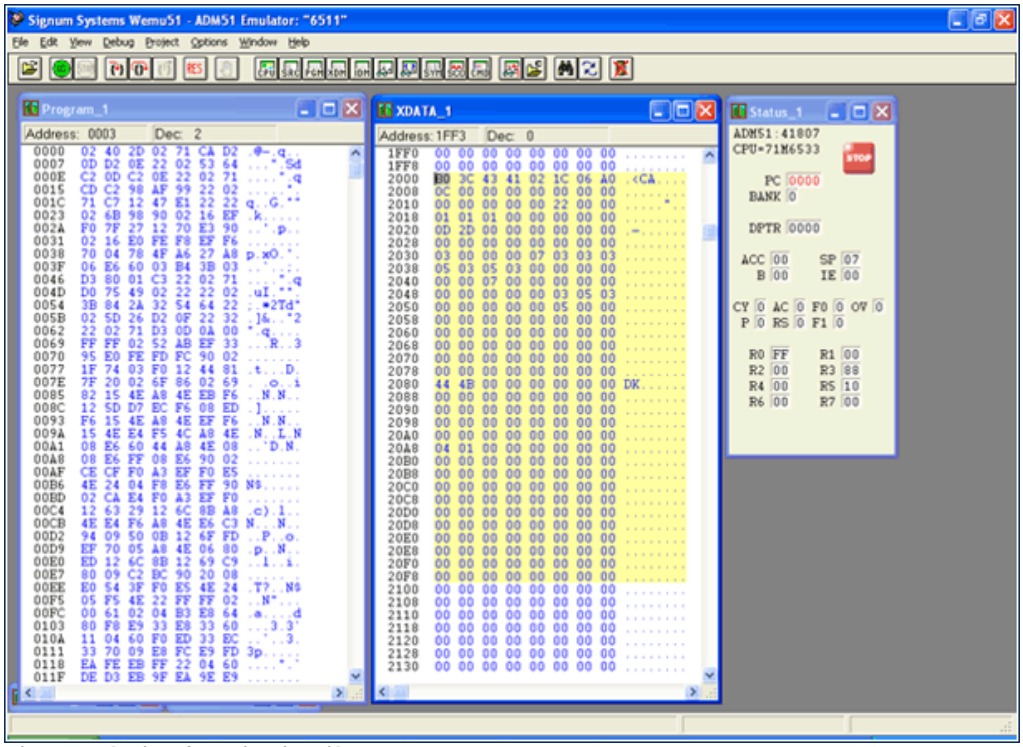


Figure 4. ICE interface showing I/O RAM.

Preparing Target Image Files (Hex Files)

Both the ADM51 ICE and the TFP2 process target image data in the form of Intel hex files. The same format is used by the BPM Microsystems programmers.

The Intel hex file is an ASCII text file with lines of text that conform to the Intel hex file format. Each line in an Intel hex file contains one hex record. These records are made up of hexadecimal numbers that represent machine language code and/or constant data.

Note: Proper device programming requires that all records are in sequential (ascending) order. Maxim provides a utility called CHKSUM.EXE that preprocesses Intel hex files for use with programmers. This utility is provided with the TFP2.

Record Format

An Intel hex file is composed of any number of hex records. Each record is made up of five fields that are arranged in the following format:

:llaaaatt[dd...]cc

Each group of letters corresponds to a different field, and each letter represents a single hexadecimal digit. Each field is composed of at least two hexadecimal digits—which make up a byte—as described below:

- **:** is the colon that starts every Intel hex record.
- ***ll*** is the record-length field that represents the number of data bytes (**dd**) in the record.
- ***aaaa*** is the address field that represents the starting address for subsequent data in the record.
- ***tt*** is the field that represents the hex record type, which may be one of the following:
  00 - data record
  01 - end-of-file record
  02 - extended segment address record
  04 - extended linear address record (ELAR)

- ***dd*** is a data field that represents one byte of data. A record may have multiple data bytes. The number of data bytes in the record must match the number specified by the **ll** field.
- ***cc*** is the checksum field that represents the checksum of the record. The checksum is calculated by summing the values of all hexadecimal digit pairs in the record modulo 256 and taking the two's complement.

The Intel hex file is made up of any number of data records that are terminated with a carriage return and a linefeed. Data records appear as shown in the following example:

```
:10246200464C5549442050524F46494C4500464C33
```

This record is decoded as follows:

```
:102462 00 464C5549442050524F46494C4500464C 33
|||||||||||||||||||||||||||||||||||||||||||CC->Checksum (0x33)
|||||||||DD->Data (0x46, 0X4C, 0X55, ... 0X4C)
|||||||TT->Record Type (0x00)
|||AAAA->Address (0x2462)
|LL->Record Length (0x10)
:->Colon
```

where:

- 10 is the number of data bytes in the record.
- 2462 is the address where the data are to be located in memory.
- 00 is the record type 00 (a data record).
- 464C...464C is the data.
- 33 is the checksum of the record.

Intel HEX386 File Format

For banked code, as used for the 71M653x devices (71M6531, 71M6532, 71M6533, 71M6534H), the Intel HEX386 file format (extended linear address records) is used.

Extended linear address records are also known as 32-bit address records and HEX386 records. These records contain the upper 16 bits (bits 16-31) of the data address. The extended linear address record always has two data bytes and appears as follows:

```
:020000040001F9
```

where:

- 02 is the number of data bytes in the record.
- 0000 is the address field. For the extended linear address record, this field is always 0000.
- 04 is the record type 04 (an extended linear address record).
- 0001 represents the upper 16 bits of the address.
- F9 is the checksum of the record and is calculated as 0x01 + NOT(0x02 + 0x00 + 0x00 + 0x04 + 0x00 + 0x01).

When an extended linear address record is read, the extended linear address stored in the data field is saved and is applied to subsequent records read from the Intel hex file. The linear address remains effective until changed by another extended address record.

The absolute-memory address of a data record is obtained by adding the address field in the record to the shifted address data from the extended linear address record. The following example illustrates this process:

Address from the data record's address field	0x2462
Extended linear address record data field	0x0001
Absolute-memory address	0x00012462

During the programming process, the programmer must parse the Intel hex file for the ELAR records and perform address translations depending on the bank that is to be programmed. A summary of the operations involved in this process is shown in **Figure 5**.

Intel Hex Record					71M653x				
Byte count	Address (HRA)	Record type	Data	Meaning	Target bank	Address in bank	Address Calculation	*FL_BANK[ ]*	
:20	0000	00	Data	First 32 bytes at 0x00000	Base bank	0x0000	=HRA	01	128 KB
:20	0020	00	Data	32 bytes at 0x00020	Base bank	0x0020	=HRA	01	
	...								
:20	7FE0	00	Data	Last 32 bytes at 0x07FE0	Base bank	0x7FE0	=HRA	01	
:20	8000	00	Data	First 32 bytes at 0x08000	Bank 1	0x0000	=HRA=0x8000	01	
:20	8020	00	Data	32 bytes at 0x08020	Bank 1	0x0020	=HRA=0x8000	01	
	...								
:20	FFE0	00	Data	Last 32 bytes at 0x0FFE0	Bank 1	0x7FE0	=HRA-0x8000	01	
:02	0000	04	0001	Address offset for records to follow = 0x0001	--	--	--	--	
:20	0000	00	Data	First 32 bytes at 0x10000	Bank 2	0x0000	=HRA	02	
:20	0020	00	Data	32 bytes at 0x10020	Bank 2	0x0020	=HRA	02	
	...								
:20	7FE0	00	Data	Last 32 bytes at 0x17FE0	Bank 2	0x7FE0	=HRA	02	
:20	8000	00	Data	First 32 bytes at 0x18000	Bank 3	0x0000	=HRA=0x8000	03	
:20	8020	00	Data	32 bytes at 0x18020	Bank 3	0x0020	=HRA=0x8000	03	
	...								
:20	FFE0	00	Data	Last 32 bytes at 0x1FFE0	Bank 3	0x7FE0	=HRA-0x8000	03	
:02	0000	04	0002	Address offset for records to follow = 0x0002	--	--	--	--	256 KB
:20	0000	00	Data	First 32 bytes at 0x20000	Bank 4	0x0000	=HRA	04	
:20	0020	00	Data	32 bytes at 0x20020	Bank 4	0x0020	=HRA	04	
	...								
:20	7FE0	00	Data	Last 32 bytes at 0x27FE0	Bank 4	0x7FE0	=HRA	04	
:20	8000	00	Data	First 32 bytes at 0x28000	Bank 5	0x0000	=HRA=0x8000	05	
:20	8020	00	Data	32 bytes at 0x28020	Bank 5	0x0020	=HRA=0x8000	05	
	...								
:20	FFE0	00	Data	Last 32 bytes at 0x2FFE0	Bank 5	0x7FE0	=HRA-0x8000	05	
:02	0000	04	0003	Address offset for records to follow = 0x0003	--	--	--	--	
:20	0000	00	Data	First 32 bytes at 0x30000	Bank 6	0x0000	=HRA	06	
:20	0020	00	Data	32 bytes at 0x30020	Bank 6	0x0020	=HRA	06	
	...								
:20	7FE0	00	Data	Last 32 bytes at 0x37FE0	Bank 6	0x7FE0	=HRA	06	
:20	8000	00	Data	First 32 bytes at 0x38000	Bank 7	0x0000	=HRA=0x8000	07	
:20	8020	00	Data	32 bytes at 0x38020	Bank 7	0x0020	=HRA=0x8000	07	
	...								
:20	FFE0	00	Data	Last 32 bytes at 0x3FFE0	Bank 7	0x7FE0	=HRA-0x8000	07	

Figure 5. Processing of Intel hex records for the programming of bank-switched code.

Generating Target Image Files for the 71M651x

Generating target image files for the 71M651x is straightforward when using the Keil PK51 Professional Developers Kit. **Figure 6** shows the dialog box that defines the characteristics of the output file in µVision3, the Keil development environment. "Create HEX File" should be checked and "HEX-80" should be selected under HEX Format. These settings guarantee that a target image file compatible with the ADM51 is created. After generating the target image file, the CHKSUM.exe utility should be used to guarantee compatibility with the TFP2 and other programmers.

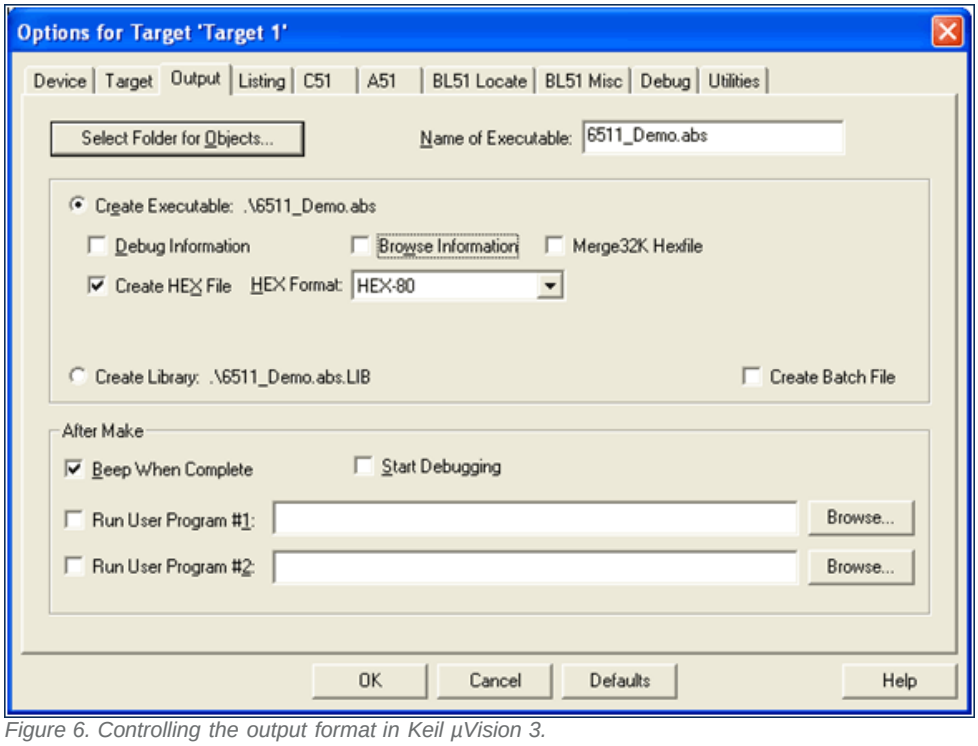


Figure 6. Controlling the output format in Keil µVision 3.

Target Image Files for the 71M653x

The Keil BL51 linker and its associated hex converter produce a separate Intel hex file for each flash memory bank. These files end with names such as .H01 for bank 1, .H02 for bank 2, etc. The ADM51 and the TFP2 expect a single Intel 386 hex file with a .hex file extension.

The CD ROM shipped with 71M653x demo kits contains a utility called bank_merge.exe. This utility runs in a DOS command window and merges the bank files generated by the Keil BL51 linker into one Intel-386 hex file.

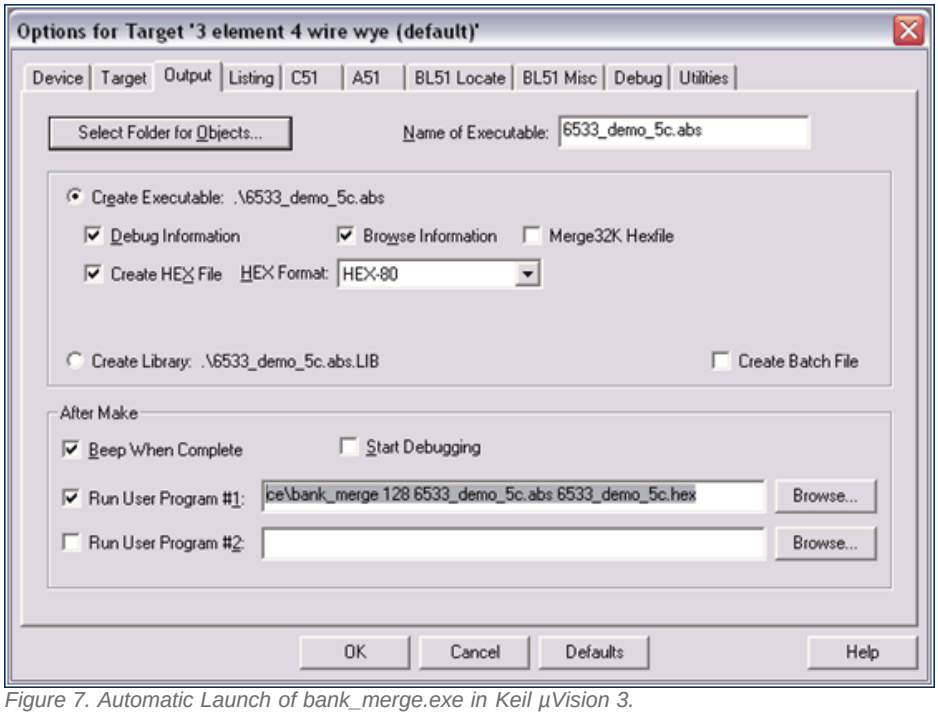


Figure 7. Automatic Launch of bank_merge.exe in Keil µVision 3.

When using Keil's Lx51 advanced linker, the output dialog contains a pulldown list of hex formats. Select the "i386" hex file option.

Note: A hex file processed with the bank_merge.exe utility should not be processed with the CHKSUM.exe utility.

The CHKSUM.exe Utility

Prior to downloading the target image hex file to the TFP2 or other programmers, the image file must be preprocessed using the CHKSUM.exe utility provided by Maxim. A hex file not processed with CHKSUM.exe will result in incomplete programming of the target's FLASH memory. The target's code must be of the Extended Intel ASCII HEX-80 format for processing by CHKSUM.exe.

From the DOS Command prompt, invoke CHKSUM.exe as follows:

```
chksum kb <infile.hex >outfile.hex
```

where:

kb = desired file size, Memory Size Switch setting to be used during TFP2 downloading and target programming

infile = target code hex file to be processed

outfile = processed target code hex file to be downloaded to TFP2

Figure 8 shows a typical invocation of CHKSUM.exe.

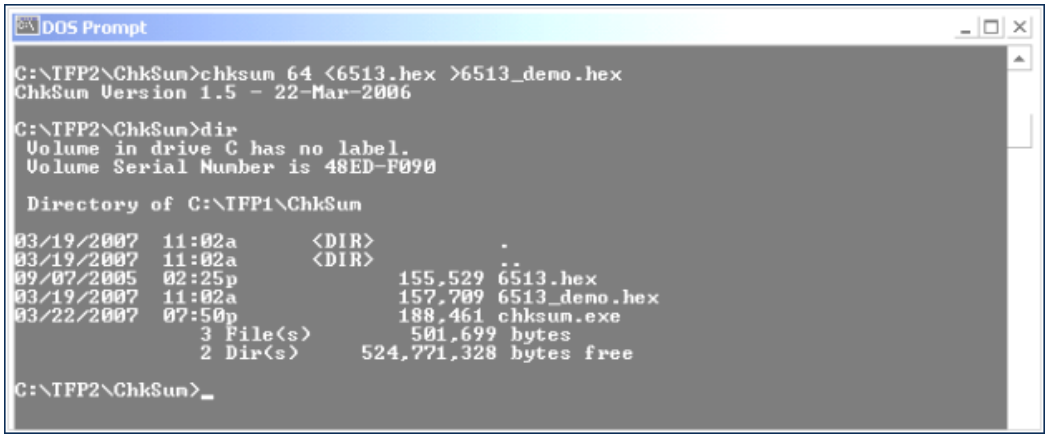


Figure 8. CHKSUM.exe hex file processing.

The purpose of the CHKSUM utility is to organize the individual hex records into a contiguous structure with addresses sequentially increasing. Some compilers produce nonsequential hex files. The TFP2 assumes a sequential file structure. A nonsequential hex file downloaded to the TFP2 results in missing bytes (the missing bytes are the out-of-sequence hex records) in the target flash memory (when the target is then programmed with the TFP2).

Note: Do not use CHKSUM.exe on files processed with the bank_merge.exe utility!

The CHKSUM.exe utility may or may not overwrite the last four bytes of the downloaded target hex file, depending on whether these locations are used.

The following cases apply to using the CHKSUM.exe utility (see **Table 2** for examples):

1. If the last four bytes of the target hex file are unused (0xFF), the CHKSUM.exe utility will insert its own calculated two-byte CRC and two-byte checksum.
2. If any of the last four bytes of the target hex file are non-0xFF values, the CHKSUM.exe utility will NOT overwrite the four original values.

Table 2. Flash Images Generated by CHKSUM.exe (Example)

Last 4 Bytes of Original Intel Hex File				Target Image Created in TFP2 EEPROM by CHKSUM.exe				Comment
0xFF	0xFF	0xFF	0xFF	CRC MSB	CRC LSB	CS MSB	CS LSB	All four bytes are 0xFF. These bytes are replaced with two CRC and two checksum bytes.
0xFF	0xB5	0xFF	0xFF	0xFF	0xB5	0xFF	0xFF	At least one byte is not 0xFF. All four bytes are maintained.
0xA3	0xF1	0x72	0x8C	0xA3	0xF1	0x72	0x8C	All four bytes are not equal to 0xFF. The original content of the last four bytes is maintained.

The CHKSUM.exe utility displays the warning shown in **Figure 9** when it encounters non-0xFF values at the last four memory locations.

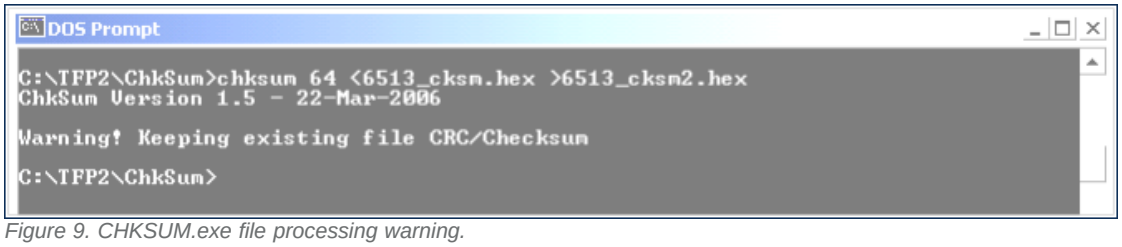


Figure 9. CHKSUM.exe file processing warning.

When programming the target flash memory, the last four bytes of the target hex file are transferred intact. Either the CHKSUM.exe calculated CRC and checksum bytes are copied or the original target's hex data are copied. If the last two bytes of the target hex file are 0xFF (CHKSUM not used), the TFP2 overwrites the last two 0xFF bytes with its calculated checksum during the HyperTerminal file download operation.

The TFP2 will display the message shown in **Figure 10** when powering up.

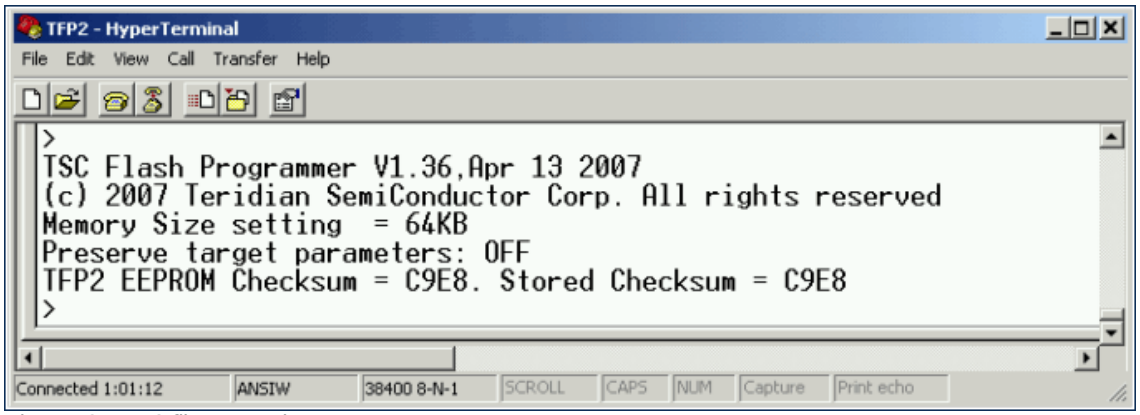


Figure 10. TFP2 file processing message.

If the target hex file has non-FF values in any of the last four bytes, the power-on message screen shown in Figure 10 may display a "TFP2 EEPROM verification error" message. In this case, the displayed "TFP2 EEPROM Checksum =" value and "Stored Checksum =" value will be different. This occurs when the stored checksum value (from the customer target image file) is derived from a different checksum calculation method than from what the TFP2 uses. Therefore, the TFP2 cannot confirm the EEPROM contents in this case. However, the checksum verification error will not prevent the TFP2 from programming the target IC. The "TFP2 EEPROM Checksum =" value is recalculated upon every power-on or system reset of the TFP2. Manual verification of the EEPROM's contents requires comparing the TFP2 EEPROM checksum value derived after the file download to subsequent power-on recalculated checksum values.

Hardware Precautions for Meter Boards

Meter Boards with 71M651x ICs

A capacitor from E_RST to ground and pullup resistors for E_RXTX, E_RST, and E_TCLK should be used on the programming interface for protection from electromagnetic interference (EMI), as shown in **Figure 11**.

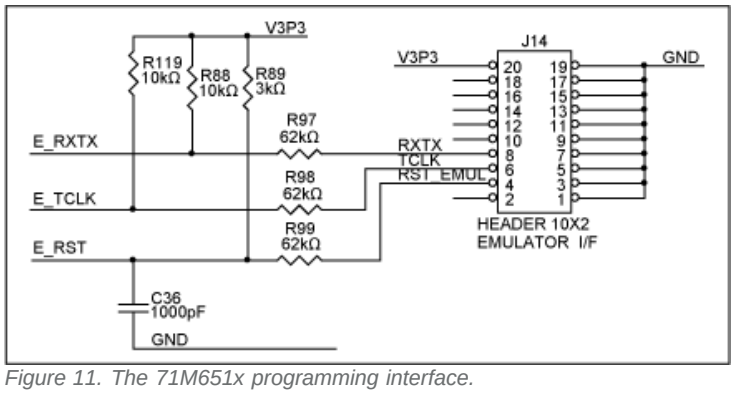


Figure 11. The 71M651x programming interface.

To successfully emulate code or program the target flash using the ADM51, the meter board must have a provision to disable the hardware watchdog timer. Otherwise, the target is reset every 1.5 seconds, which makes programming the target flash impossible. Disabling the watchdog timer is usually done with a removable jumper, as shown in **Figure 12**.

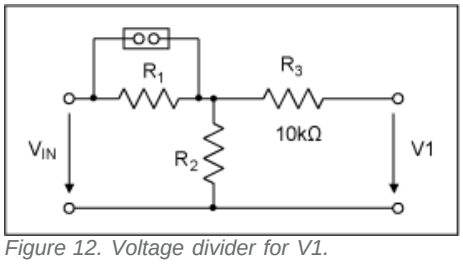


Figure 12. Voltage divider for V1.

If the TFP2 is used to program the target flash, disabling the target watchdog timer is not necessary. The TFP2 has a mechanism that allows it to trigger the watchdog timer frequently to prevent a target reset.

Meter Boards with 71M653x ICs

Capacitors to ground should be used on the programming interface for protection from EMI. Production boards should have the ICE_E pin connected to ground.

If the ICE pins are used to drive LCD segments, the pullup resistors should be omitted, as shown in **Figure 13**, and 22pF capacitors to GNDD should be used for protection from EMI.

It is important to bring out the ICE_E pin to the programming interface. This creates a way to reprogram meters that are difficult to reprogram, such as meters equipped with batteries and code that sets the SECURE bit. In those cases, powering down the meter will not generate the reset required for the SECURE bit to be reset. The rationale for this recommendation is as follows:

- Production meters containing batteries often have no reset button or other way of initiating a reset.
- The battery prevents the meter IC to be reset by simply removing the mains power.
- With no opportunity to cause a reset, the meter IC may be erased, but the reset required between the erase and programming operations is missing.

Providing access to ICE_E ensures that the part can be reset between erase and program cycles, which will enable programming devices to reprogram the part. The reset required is implemented with a watchdog timer reset (i.e., the hardware WDT must be enabled).

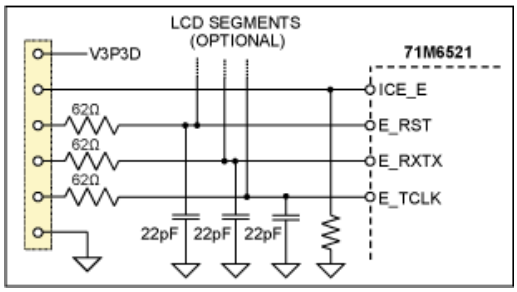


Figure 13. The 71M653x programming interface.

Appendix A: Schematics for Programmers Based on TFP2

It is fairly easy to design a simple device programmer using a TFP2, a PCB that contains a socket, and some added components, as shown in **Figure 14**.

This programmer can be manually operated using the pushbutton or controlled with ATE equipment. It is also possible to duplicate the arrangement using two TFP2 programmers and two socket boards. Alternatively, you can use multiple configurations to create the equivalent of a gang programmer, as long as each TFP2 has its own socket board.

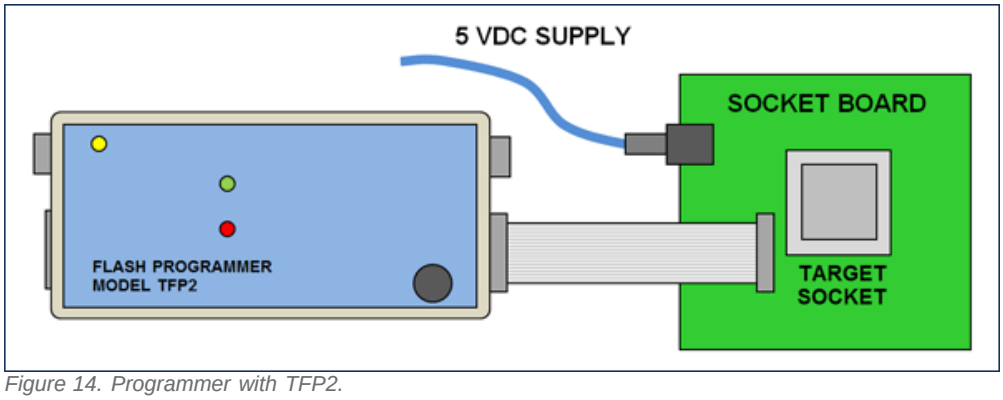


Figure 14. Programmer with TFP2.

Figures 15 to **18** show schematic designs of device programmers for various Maxim metering ICs. These programmers can be built based on regular PCBs or prototype boards such as "Perf-Board." Schematics for other metering ICs are similar to the examples given and can easily be derived from the schematics.

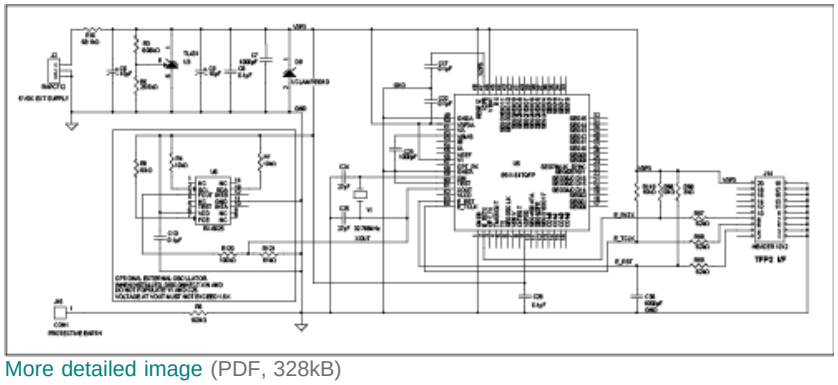
A few general rules apply when converting the schematics to actual circuits:

1. Crystal and crystal capacitors must be very close to the XIN and XOUT pins of the target IC.
2. Optionally, an external oscillator, such as an EPSON RX-8025 or any oscillator with digital 32kHz output can be used. This option is recommended because it guarantees a more stable operation, even in the presence of imperfect layout, contamination around the XIN/XOUT pins, and moisture buildup.
3. Bypass capacitors from V_{BIAS}, V3P3A, V2P5 and other pins to ground should be mounted very close to the target IC.
4. The connections for the E_RST, ERXTX, and E_TCLK signals should be very short. This means the 2X10 ICE connector used to plug in the ribbon cable from the TFP2 should be not more than 5cm (2 inches) from the target IC. The emulator interface signals E_RST, E_RXTX, and E_TCLK should be routed and connected carefully, i.e., away from the crystal oscillator inputs, with straight and short routes or wires (particularly E_TCLK, which carries a 10MHz signal).
5. The terminals labeled "Protective Earth" should be connected to a solid earth/ground to prevent the programming socket from accumulating electric charge when an isolated power supply is used.

Table 3 shows typical common components such as crystals, capacitors, connectors, and sockets that can be used for the socket boards.

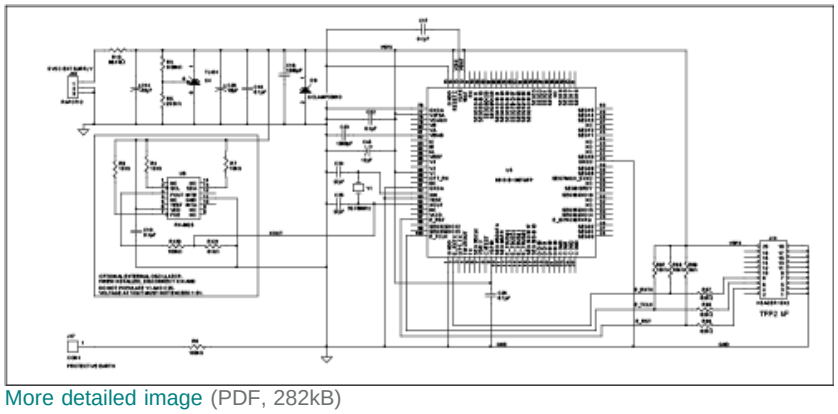
Table 3. Common Components for the Socket Board

Part	Manufacturer	Part Number	U.S. Distributor	Distributor Part Number
LQFP-64 socket	Yamaichi	IC169-064-*69-B5	Future Electronics and others	
LQFP-100 socket		IC169-100-*25-B5		
LQFP-120 socket		IC149-120-143-B5		
TFP2 connector (2x10)	TYCO/AMP	5-104068-1	Mouser	571-5-104068-1
Crystal, 32kHz	ECS	ECS.327-12.5-39-TR	Digi-Key	XC1658CT-ND
Oscillator, 32kHz	Epson	RX-8025SA	Digi-Key	SER3650-CT-ND
Voltage regulator	Texas Instruments	TL431AIDR	Digi-Key	296-1288-1-ND
DC connector	Switchcraft	RAPC712X	Digi-Key	SC237-ND



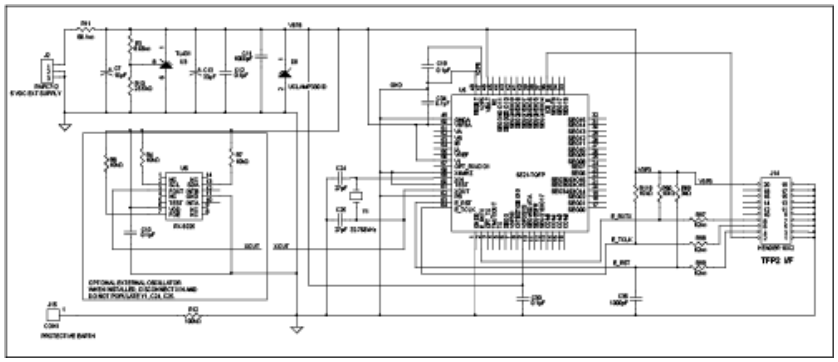
More detailed image (PDF, 328kB)

Figure 15. Programmer for the 71M6511/71M6511H.



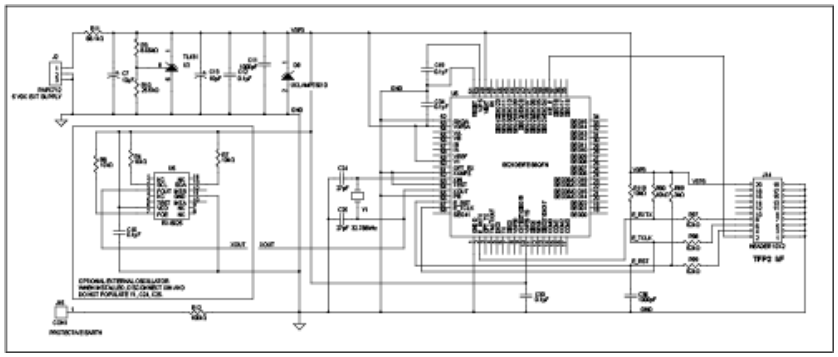
More detailed image (PDF, 282kB)

Figure 16. Programmer for the 71M6513/71M6513H.



More detailed image (PDF, 325kB)

Figure 17. Programmer for the 71M6521BE/DE/FE in a 64-pin package.



More detailed image (PDF, 326kB)

Figure 18. Programmer for the 71M6521DE/FE in a 68-pin package.

Related Parts		
71M6511	Single-Phase Energy Meter ICs	Free Samples
71M6511H	Single-Phase Energy Meter ICs	Free Samples
71M6513	3-Phase Energy Meter ICs	Free Samples
71M6513H	3-Phase Energy Meter ICs	Free Samples
71M6531D	Energy Meter ICs	Free Samples
71M6531F	Energy Meter ICs	Free Samples
71M6532D	Energy Meter ICs	Free Samples
71M6532F	Energy Meter ICs	Free Samples
71M6533	Energy Meter ICs	Free Samples
71M6533G	Energy Meter ICs	Free Samples
71M6533H	Energy Meter ICs	Free Samples
71M6534	Energy Meter ICs	Free Samples
71M6534H	Energy Meter ICs	Free Samples
71M6541D	Energy Meter ICs	Free Samples
71M6541F	Energy Meter ICs	Free Samples
71M6541G	Energy Meter ICs	Free Samples
71M6542F	Energy Meter ICs	Free Samples
71M6542G	Energy Meter ICs	Free Samples
71M6543F	Energy Meter ICs	Free Samples
71M6543GH	Energy Meter ICs	
71M6543H	Energy Meter ICs	

71M6545	Metrology Processors	Free Samples
71M6545H	Metrology Processors	

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5539: http://www.maximintegrated.com/an5539
APPLICATION NOTE 5539, AN5539, AN 5539, APP5539, Appnote5539, Appnote 5539, AN_65XX_057