Keywords: computation engine (CE) code, microprocessing unit (MPU) code, compile and link meter/demo code

APPLICATION NOTE 5751

Compiling Meter Code with New Computation Engine (CE) Code Images

Nov 19, 2013

Abstract: This application note describes how to modify demo code or meter application code so it can be compiled and linked with new computation engine (CE) code images for the 71M6531/71M6532 and 71M6533/71M6534 and 71M6541/71M6542/71M6543 family of electricity metering ICs.

Introduction

Background

The 71M6531/71M6532 and 71M6533/71M6534 and 71M6541/71M6542/71M6543 families of electricity metering ICs are designed to be used with computation engine (CE) codes that can be adapted to the application at hand. In some cases, special CE codes are required and supplied by Maxim to support different sensor types or features such as filtering or harmonic analysis.

Since CE codes occupy different spaces in flash memory and XRAM, depending on the application, several steps are needed to accommodate a specific CE code image. This is done by modifying both the meter code (demo code or meter application code) sources and the Keil® compiler settings (Keil µVision® 4).

This application note discusses the necessary steps to successfully integrate a new CE code image into meter code. A general knowledge of the tools, i.e. Keil µVision and Signum Systems™ WEMU51 is helpful when performing the tasks described in this application note. Reading the software user guide (SUG) for the 71M653x and/or 71M654x ICs is recommended.

General Interface Between Meter Code and CE Code

How CE Code and CE Data Files Are Delivered

Source code for the CE is not available to the customer. CE code is delivered as a code image that is linked into the meter code flash image and then interpreted as CE op codes and executed by the CE at run time. When executing, the CE code needs certain registers in XRAM (CE RAM) to be initialized. For example, calibration coefficients for an uncalibrated (default) meter are set to 0x4000, and typically a standard value for WRATE is also part of the CE data image. The files for CE code that are delivered to the customer are the following:

- CE code image (for example CE34B07G_ce.c): This code is for the 71M6534 or 71M6533, and its version is B07G.
- CE data image (for example CE34B07G_dat.c): This set of data is for the code version B07G for the 71M6534 or 71M6533.

CE files are named with the leading characters “CE”, followed by two digits specifying the meter system-on-chip (SoC) family, e.g. “34” for the 71M6533 and 71M6534, or “41” for the 71M6541 and 71M6542. The next letter generally specifies the silicon revision (A, B, C, …) that the code is compatible with. The two digits after the revision letter are the code type. When a new code is developed with significantly different functionality, a new number is selected for these two digits. No features or functionality can be directly derived from the two digits. Typically, an application note provided with a CE code describes the features and usage of a particular code type. The letter after the two digits is the revision code of the particular code type. For example, revision “D” contains fixes for issues that were in revision “C”. When several revisions of the same code type are available, the latest revision should be used.

A portion of a typical CE code image is shown below:

```
// File: ce34a02d_ce.c
//
// CE Program Image File (8051 C format)
const short code NumCeCode=1280;        // The number of words in the
'CeCode' array.
const unsigned char code CeCode[]={
 0x20,0xe3, 0x74,0x45, 0xec,0xff, 0x8c,0x45, 0x20,0xe4, 0x74,0x46,
0xed,0xff, 0x8c,0x46,
 0x20,0xe3, 0x74,0x45, 0xec,0xff, 0x8c,0x45, 0x20,0xe4, 0x74,0x46,
0xed,0xff, 0x8c,0x46,
 0x20,0xe3, 0x74,0x45, 0xec,0xff, 0x8c,0x45, 0x20,0xe4, 0x74,0x46,
0xed,0xff, 0x8c,0x46,
 0x20,0xe3, 0x74,0x45, 0xec,0xff, 0x8c,0x45, 0x20,0xe4, 0x74,0x46,
0xed,0xff, 0x8c,0x46,
 ...
 0x8f,0x05, 0x21,0x03, 0x8f,0x03, 0x21,0x04, 0x8f,0x04, 0x21,0x05,
0x8f,0x05, 0x20,0x03,
 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff,
0xff,0xff, 0xff,0xff,
 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff,
0xff,0xff, 0xff,0xff,
 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff,
0xff,0xff, 0xff,0xff,
 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff, 0xff,0xff,
0xff,0xff, 0xff,0xff
};
```

Two bytes (16 bits) combined form one op code for the CE. 0xFFFF is the op code for the STOP instruction.

A portion of a typical CE data image is shown below:

```
// File: ce34a02d_dat.c
//
// CE Data Image File (8051 C format)
const short code NumCeData=268; // The number of words in the 'CeData'
array.
const unsigned char code CeData[]={
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00,
0x00,0x00,0x40,0x00,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00,
0x00,0x00,0x40,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x50,0x00, 0x00,0x00,0x00,0xab, 0x00,0x00,0x19,0x2c,
0x00,0x00,0x09,0xd8,
 0x01,0x6d,0x24,0x90, 0xff,0xff,0xff,0xff, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,
 0xff,0xff,0xff,0xff, 0x63,0x65,0x33,0x34, 0x61,0x30,0x32,0x64,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x0c, 0x00,0x00,0x59,0xba, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x18,0x00, 0x00,0x00,0x09,0xd8, 0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x20,0x00,0x00,0x00, 0x20,0x00,0x00,0x00,
0x60,0x00,0x00,0x00,
 0x60,0x00,0x00,0x00, 0x00,0x00,0x80,0x00, 0x40,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 …
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00
};
```

CE data is organized in 4-byte (32-bit) words. These are fixed-point data and can be any type of data processed by the CE, such as calibration coefficients, constants, intermediate results, or output data.

Note the 0x40/0x00 byte combinations (shown in yellow) starting in line 5 of the data field. These are the default settings for the calibration coefficients located at CE data registers 0x0010 and up. The values are actually to be interpreted as 0x00004000. At startup, the essential part of the CE data image needs to be copied to XRAM, starting at address 0x0000, for the CE code to have valid data to work with. This is done by the meter code. Data towards the end of the CE data image is 0x00, and the meter code may take a shortcut by not copying the zero data to CE RAM.

Note that some CE codes have non-zero data towards the end of the data image and that a shortcut cannot be used for those codes. An example for this type of code is shown below (note the non-zero bytes marked in yellow embedded in zero bytes):

```
const short code NumCeData=414; // The number of words in the 'CeData'
array.
```

```
const unsigned char code CeData[]={
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00,
0x00,0x00,0x40,0x00,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00, 0x00,0x00,0x40,0x00,
0x00,0x00,0x40,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x50,0x00, 0x00,0x00,0x00,0xe3, 0x00,0x00,0x19,0x2c,
0x00,0x00,0x08,0x88,
 0x01,0x6d,0x24,0x90, 0xff,0xff,0xff,0xff, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0xff,0xff,0xff,0xff, 0x63,0x65,0x33,0x34, 0x61,0x32,0x30,0x62,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x0c, 0x00,0x00,0x58,0xc1, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x18,0x00, 0x00,0x00,0x09,0xd8, 0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,
 0x00,0x00,0x40,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x20,0x00,0x00,0x00, 0x20,0x00,0x00,0x00,
0x60,0x00,0x00,0x00,
 0x60,0x00,0x00,0x00, 0xff,0xff,0xff,0xff, 0x00,0x00,0x80,0x00,
0x40,0x00,0x00,0x00,
 0x00,0x00,0x00,0x50, 0x00,0x00,0x00,0x03, 0x03,0x35,0x92,0x44,
0x02,0xfe,0xcc,0xc8,
 0x02,0xfe,0xcc,0xc8, 0x02,0xfe,0xcc,0xc8, 0x02,0xfe,0xcc,0xc8,
0x05,0x90,0x0e,0x98,
 0x05,0x59,0x49,0x1c, 0x05,0x90,0x0e,0x98, 0x05,0x90,0x0e,0x98,
0x05,0x90,0x0e,0x98,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x0d, 0x12,0xe0, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x01,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0xff,0xff,0xff,0xff, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,
 0x00,0x00,0x00,0x00, 0x00,0x00,0x00,0x00,
};
```

Flash Memory of the 71M653x and 71M654x ICs Reserved for CE Code

The flash memory spans from 0x0000 to 0xFFFF for a product with 64KB flash, to 0x1FFFF for a 128KB product, and to 0x3FFFF for a 256KB product. The CE code image is placed somewhere in this range on a 1KB boundary (0x0400) by the linker and then found by the CE at run time using the *`CE_LCTN`* pointer in I/O RAM. *`CE_LCTN`* multiplied by 1024 (0x0400) yields the flash location of the CE code. It is advantageous to locate the CE code at the top end of the flash memory where it can grow in size without affecting other code. For large size CE codes, *`CE_LCTN`* must move to an address 1KB or 2KB lower, because the end of that particular CE code may exceed the end address of the code bank.

Sometimes the default settings of the meter code do not work, because the CE code is too large. The meter code's defaults already have 25% extra space, especially to permit some growth. However, very large CE codes may need more than 25% extra space.

The size of the code is specified in the `NumCeCode` constant, which is part of the CE code image. `NumCeCode` is the actual number of CE instructions in the code. It is inserted in the _ce.c code file when the file is generated. For example, `NumCeCode` is 1950 for many standard 71M653x CE codes. Since CE code is structured in 16-bit words, this makes the overall length of the code 2 × 1950 = 3900 bytes. Complex CE codes typically are larger, such as CE34A20, which has 2235 words, or 4470 bytes.

RAM of the 71M653x and 71M654x ICs

RAM for the CE codes is located in general XRAM, which is shared between the CE and the microprocessing unit (MPU) in all 71M653x and 71M654x ICs.

The XRAM for the 71M653x spans from 0x0000 to 0x0FFF (4KB). XRAM for 71M654x ranges between 3KB (0x0000 to 0x0BFF) and 5KB (0x0000 to 0x13FF), depending on the part type.

In all 71M653x and 71M654x parts, the lowest XRAM locations, starting at 0x0000 are reserved for the ADC outputs and calibration coefficients, to be followed by the CE registers and transfer variables (for exact CE register locations, see the data sheet). Above that, CE code uses XRAM space for internal variables. It is important that there is no overlap between XRAM usage by the MPU and by the CE. The MPU should occupy the XRAM towards the upper addresses.

CE codes state their XRAM usage in the `NumCeData` constant, which is measured in 32-bit words. For example, 320 32-bit words will result in 1280 bytes (0x500).

At run time, i.e. when the meter code starts, the MPU copies the constants in the CE data image to XRAM as part of the initialization. `CeData` is the source address specified for this copy operation. Note that `CeData` is declared as `const unsigned char code`, which means that the data image is treated as code and consequently placed in flash memory by the linker. The MPU code can find the CE data image at the location that the linker puts it at.

Basically, the CE data image can be located anywhere in flash. However, 1KB boundaries (0x0400) are re-commended because the MPU code may want to update the image in flash, e.g. when the user wants to store the calibration and other parameters not in EEPROM but in flash. Erasing a whole flash page is straightforward, which is what must happen before the CE parameters are written to non-zero flash. Having the CE data image not cross page boundaries is advantageous: when the CE data image is updated to the flash memory, the read/write operation is simpler when the image is within a flash page boundary (see the data sheet for the meter SoC for details).

General Memory Map

Figure 1 shows the general memory map: flash memory contains the code for the MPU plus the code and data images for the CE. For the CE data image, there is a non-zero portion and a zero portion. At run time, both the MPU and CE operate out of flash memory. When the MPU code starts, it copies the CE data image into XRAM, which is shared between the CE and MPU.

The MPU code may choose to copy only the non-zero portion of the CE data image or to copy the whole CE data image.

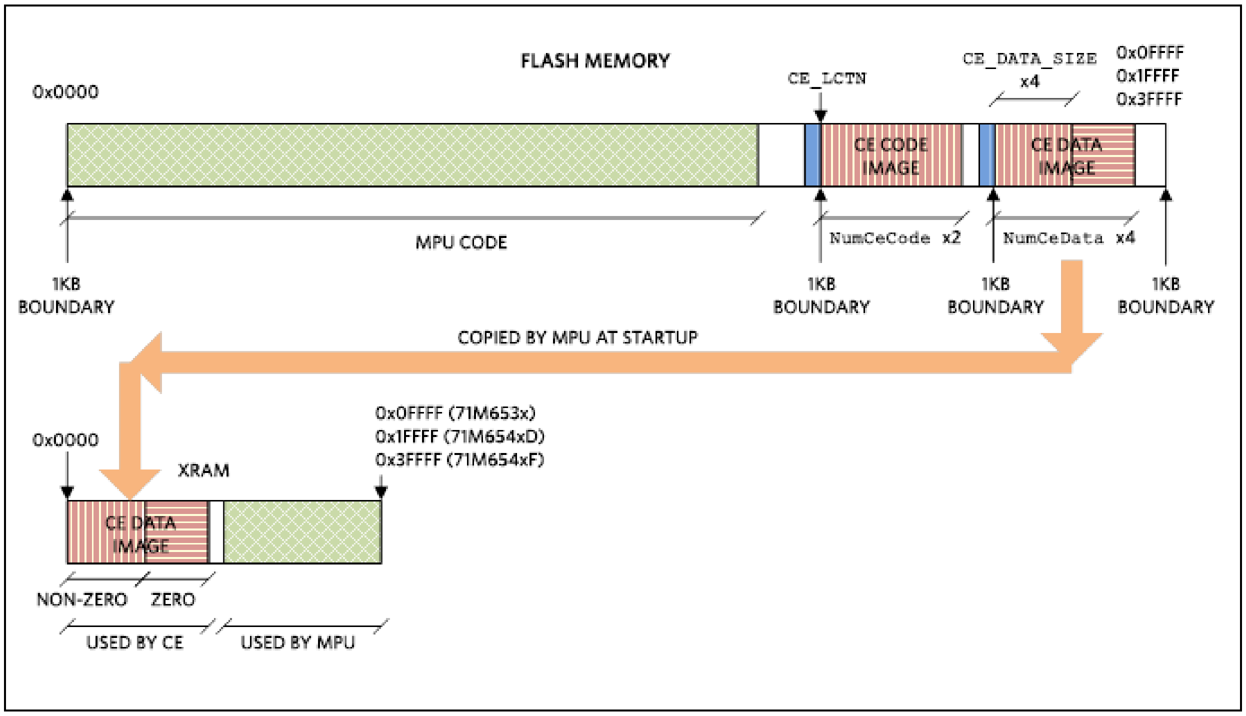
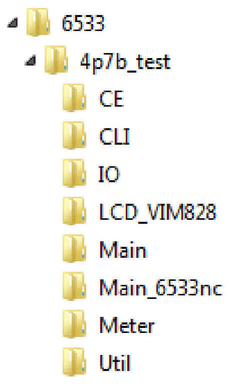


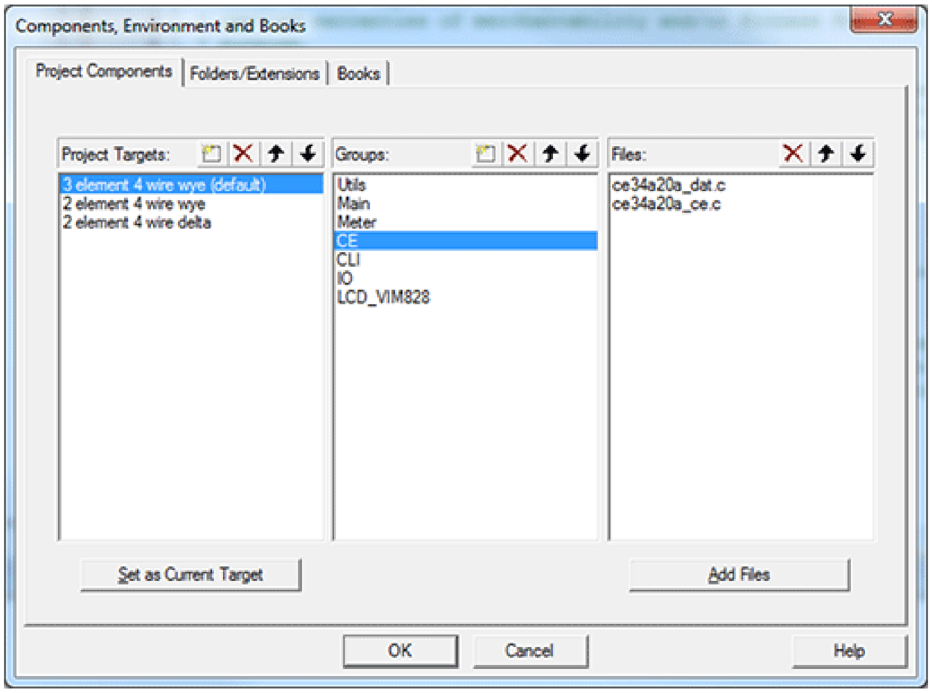
Figure 1. General memory map.

Keil Compiler Settings
Using new CE code requires the code and data files, for example CE34B07G_ce.c and CE34B07G_dat.c, to be copied into the CE folder of the code folder environment. Demo code sources are supplied by Maxim in a fixed folder structure. The meter application code used by customers may have a similar structure. An example for a 71M6533 Demo Code folder structure is shown below.



```
6533
  4p7b_test
    CE
    CLI
    IO
    LCD_VIM828
    Main
    Main_6533nc
    Meter
    Util
```

When we open the CE folder, we see the collection of CE codes and CE data images that were supplied as part of the demo code. Note that only <u>one</u> CE code image and <u>one</u> data image are typically used for a code build. We have to let the Keil µVision environment know which CE code images to use. This is done by selecting the CE group in the left panel (Project Panel) of the Keil µVision 4 graphical user interface (GUI), right clicking, and then selecting 'Manage Components'. When the 'CE' entry under 'Groups' is selected, as shown in **Figure 2**, the available CE files show up in the right panel under 'Files'.



Project Targets:
3 element 4 wire wye (default)
2 element 4 wire wye
2 element 4 wire delta

Groups:
Utils
Main
Meter
CE
CLI
IO
LCD_VIM828

Files:
ce34a20a_dat.c
ce34a20a_ce.c

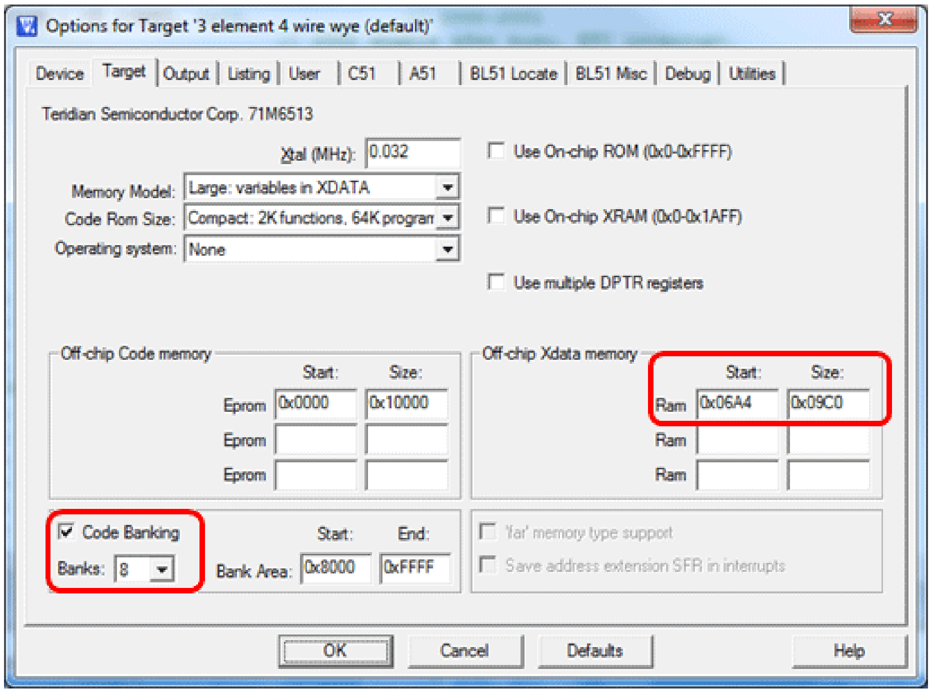
Figure 2. Project Components tab.

Pressing the button 'Add Files' opens a dialog box that allows for browsing to the physical location of the CE code and data images for selection and inclusion into the code build. After this, the 'Close' button of the browse dialog box must be clicked. The unneeded CE files should be removed by selecting their file names and by pressing the delete button in the right panel under 'Files'.

Note: This operation only removes the file from the Keil project. The removed files are still physically present on the hard drive.

Another tab in 'Options for Target' in the Keil PK51 environment must be adapted to accommodate the CE code to be merged: The 'Target' tab has an area where the RAM space needed for the CE code needs defining. This entry allows the Keil PK51 compiler/linker to define where in XRAM variables and stack for the 80515 MPU can be located. The **Figure 3** below shows entries 0x06A4 in the 'Start' and 0x096C in the 'Size' window. The entry 0x06A4 reserves 0x06A4 or 1700 decimal bytes in XRAM space, i.e. the range from 0x0000 to 0x06A3, for CE RAM usage and leaves the area from 0x06A4 to 0x0FFF (for a 4KB RAM size) for the MPU. The entry 0x06A4 corresponds to `NumCeData` multiplied by 4, since each CE register/word is equivalent to four bytes. The entry under 'Size' is the difference between the maximum available XRAM size (4KB, or 0x1000 in this example) and the address given under 'Start'.

Note that for all 71M653x ICs and for the 71M654xG ICs 'Code Banking' is checked and the number of banks is properly selected, i.e. 8 for 256KB flash memory and 4 for 128KB.



Teridian Semiconductor Corp. 71M6513

Xtal (MHz): 0.032

Memory Model: Large: variables in XDATA

Code Rom Size: Compact: 2K functions, 64K program

Operating system: None

☐ Use On-chip ROM (0x0-0xFFFF)

☐ Use On-chip XRAM (0x0-0x1AFF)

☐ Use multiple DPTR registers

Off-chip Code memory

	Start:	Size:
Eprom	0x0000	0x10000
Eprom		
Eprom		

Off-chip Xdata memory

	Start:	Size:
Ram	0x06A4	0x09C0
Ram		
Ram		

☑ Code Banking

Banks: 8

Bank Area: Start: 0x8000 End: 0xFFFF

☐ 'far' memory type support

☐ Save address extension SFR in interrupts

Figure 3. Target tab.

One nice thing about the settings in the Keil compiler is that they are reflected in the Keil project file (*.uv2, *.uvproj), once the project is saved. This means that copying the Keil project file guarantees that the settings are properly applied and do not require manually reentering.

General Settings for MPU Code

CE codes are generated for a specific setting of the analog front-end (AFE), i.e. the combination of sampling frequency, finite impulse response (FIR) filter length, number and sequence of sampled channels, and other parameters. In some cases, the MPU clock has to be adapted, also. Operating the CE code with an AFE setting other than the specified results in malfunction.

CE codes typically come either with an application note of their own or with a brief description of the required settings for the AFE. The settings specified by the application note or description MUST be implemented exactly by corresponding changes in the MPU source code.

Note that LSB values must be adapted in source code if the LSBs for Wh, VARh, VnSQSUM_X, InSQSUM_X, or other metrology parameters have changed with the new CE code.

Flash Memory Operations and CE Code

By design, operations by the MPU code on the flash memory require the CE to be disabled. This is achieved by clearing the *CE_E* bit (bit 3 at I/O RAM address 0x2000). After clearing this bit, a delay of a complete CE code run (e.g. 396µs for codes sampling at 2520Hz, 458µs for codes sampling at 2184Hz) is required to allow the CE to encounter the STOP command before commencing any flash operation.

71M653x Family of Electricity Metering ICs

CE Code Location

For 71M653x demo code (as of revision 4p7b), an instruction in main.c loads *CE_LCTN* with decimal 31, resulting in

0x400 * 31 = 0x7C00. Later, the pointer is reloaded in ce_30.c with *CE_LCTN_ADR*, which is defined in options.h as 0x78. This makes the final CE location in flash 0x78 * 0x400 = 0x1E000. As mentioned before, it is advantageous to locate the CE code at the top end of the flash memory where it can grow in size without affecting other code. For large size CE codes (i.e. if NumCeCode > 512), *CE_LCTN* has to point to an address 1KB or 2KB lower, i.e. *CE_LCTN* must be decremented by 1 or 2.

CE_LCTN points to the address where the CE code starts, i.e. the first instruction. Note that NumCeCode occupies two bytes that are not part of the code itself. This means that when the CE code image is placed using the linker settings of Keil PK51, the image location is *CE_LCTN* - 2.

CE Clock Rate and Code Length

The 71M653x electricity metering ICs can operate the CE at the standard 4.915MHz rate or at the double clock rate of 9.3804MHz (when the *CE_10MHZ* bit is set). The values for sample frequency (frame rate), executable CE code cycles, and CE clock rate cannot be arbitrary due to the fact that the CE must complete one code run during the time slot allocated for the multiplexer frame.

A typical frame duration is 396.729µs (for a sampling frequency of 2520.62Hz), and at 4.9152MHz CE clock rate (203ns per instruction), 1950 CE instructions fit into one frame. All CE codes that exceed 1950 instructions (or more than 3.8KB code size) must be operated at a CE clock rate of 9.8304MHz (101.5ns per instruction), which allows for up to 3900 instructions (equivalent to 7.6KB CE code size) into one frame.

Typical polyphase CE codes that include the sampling of the neutral current operate at 2184.53Hz or 457.76µs. These codes allow for the execution of 2255 instructions at 4.9152MHz CE clock rate.

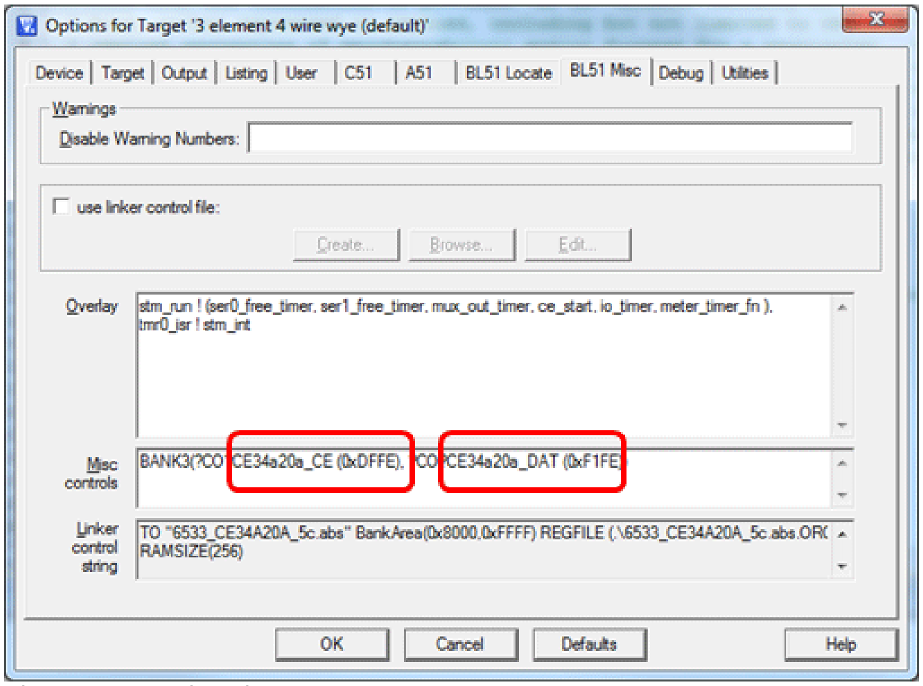
Single-phase codes can have faster frame rates, e.g. 3276.8Hz, with 305.176µs per frame. These codes fit only 1500 instructions at 4.9152MHz or 3000 instructions at 9.8304MHz.

Knowledge of the relation between sample frequency (frame rate), CE code cycles, and CE clock rate helps the user understand how the *CE_10MHZ* bit should be set, if this is not already mentioned in the application note for the given CE code.

If the *CE_10MHZ* bit is set, the *M40MHZ* bit must also be set. This means that the MPU clock rate must be adjusted, and, therefore, the baud rates and timer constants should also all be adjusted.

Keil Settings

The entry shown in **Figure 4** in the ‘BL51 Misc’ tab (accessible with ‘Options for Target’ in the Keil PK51 environment) allocates the space above 0xDFFE for the CE code image (CE34A20A_CE.C in this case). Note that BANK3 is used, which places the code at flash location 0x1DFFE.


Options for Target '3 element 4 wire wye (default)'

Device | Target | Output | Listing | User | C51 | A51 | BL51 Locate | BL51 Misc | Debug | Utilities

Warnings
Disable Warning Numbers:

☐ use linker control file:

Overlay:
```
stm_run ! (ser0_free_timer, ser1_free_timer, mux_out_timer, ce_start, io_timer, meter_timer_fn ),
tmr0_isr ! stm_int
```

Misc controls:
```
BANK3(?CO?CE34a20a_CE (0xDFFE), ?CO?CE34a20a_DAT (0xF1FE)
```

Linker control string:
```
TO "6533_CE34A20A_5c.abs" BankArea(0x8000,0xFFFF) REGFILE (.\6533_CE34A20A_5c.abs.ORC
RAMSIZE(256)
```

Figure 4. BL51 Misc tab.

The CE data image is typically placed 1024 bytes above that, at 0xEFFE, resulting in 0x1EFFE using BANK3. In the figure above, the CE data image has been selected to start at 0x1F200 (0x1F1FE + 2).

See **Figure 5** and refer to Table 38 in the data sheet for the physical location of BANK 3:

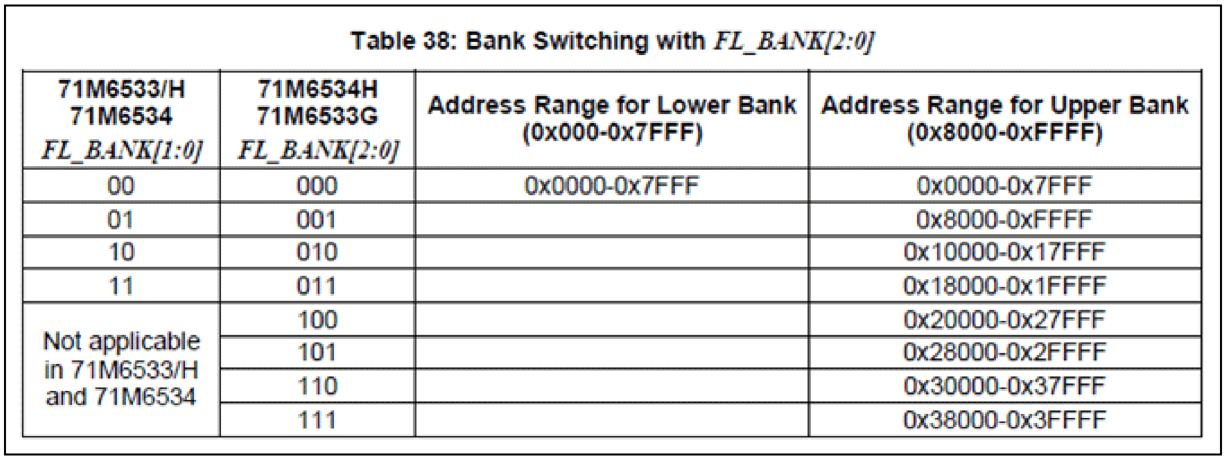


Table 38: Bank Switching with *FL_BANK[2:0]*

71M6533/H 71M6534 *FL_BANK[1:0]*	71M6534H 71M6533G *FL_BANK[2:0]*	Address Range for Lower Bank (0x000-0x7FFF)	Address Range for Upper Bank (0x8000-0xFFFF)
00	000	0x0000-0x7FFF	0x0000-0x7FFF
01	001		0x8000-0xFFFF
10	010		0x10000-0x17FFF
11	011		0x18000-0x1FFFF
Not applicable in 71M6533/H and 71M6534	100		0x20000-0x27FFF
	101		0x28000-0x2FFFF
	110		0x30000-0x37FFF
	111		0x38000-0x3FFFF

Figure 5. 71M653X bank switching table.

CE Data

The same space considerations applied to `NumCeCode` apply to `NumCeData:` If `NumCeData` × 4 is larger than 1024, the code image must move down in flash by 1KB. In that case, *`CE_LCTN`* must be decremented also to avoid overlap of CE code and CE data in flash memory.

At run time, the constants in the CE data image are copied to XRAM as part of the initialization. The following code in ce_30.c performs the copy operation:

```
FL_BANK = BANK_CE;
        memcpy_cer (
        (int32x_t *) CE_DATA_BASE, (int32r_t *) CeData, min(CE_DATA_SIZE,
NumCeData)
           );
       FL_BANK = saved_bank;
       }
```

`CE_DATA_BASE` is defined in ce653x.h as 0x0000. In this file, `CE_DATA_SIZE` is defined as 0x0140 32-bit words (= 0x0500). Note that the copy routine uses the minimum of `CE_DATA_SIZE` and `NumCeData`. This is because only the lower part of the CE code image needs to be copied using `CE_DATA_SIZE`, which is the count used to copy the essential data from the CE data image in flash to the CE data area in XRAM. The rest of the data area (i.e. up to the full `NumCeData` size is not copied and should work alright when it is just cleared to zero. The purpose for this partial copy operation is to reduce the flash size consumed by the CE data tables. The CE programmers typically move all the special starting values (non-zero values) within `CE_DATA_SIZE`.

Note: Some CE codes have non-zero CE data at the end of their CE data image. These must be copied by the MPU. In this case, using the minimum of the `CE_DATA_SIZE` and `NumCeData` is not the recommended procedure, and the copy routine should use `NumCeData` × 4.

If a larger CE XRAM usage is required, `CE_DATA_SIZE` needs to change.

CE codes with larger allocation for XRAM need the following changes:

- ‘Target’ tab accessible with ‘Options for Target’ in the Keil PK51 environment must specify the XRAM space needed for the CE code.
- `CE_DATA_SIZE` needs to be updated (ce653x.h).

Hardware Initialization in MPU Code

Initialization of the I/O RAM is done using the structure `uint8r_t_ri_defaults[ ]` in defaults.c. This structure consists of several segments that treat a number of I/O addresses starting at a given offset from 0x2000 (the base address for the I/O RAM). It helps to have the I/O RAM map from the Firmware Interface section of the data sheet to decipher the entries. Most entries are hard-coded, for example 0x34 for address 0x2001, which initializes *SUM_CYCLES* with 0x34, or 52 decimal. In conjunction with *PRE_SAMPS* (two bits, set to zero, resulting in a numerical value of 42), the resulting number of samples per accumulation interval is 42 × 52 = 2184.

Some of the entries are derived from definitions, such as `EQUATION | CE_10MHZ` at address 0x2000, which is the logical OR of `EQUATION`, as defined in options.h and `CE_10MHZ`, as defined in IO653X.h, or `MPU_FREQ` at address 0x2004, which is defined in OPTIONS_GBL.H.

The later segments, e.g. for the general-purpose input output (GPIO) configuration starting at I/O RAM address 0x2030, are not as critical for CE function. An example for `uint8r_t_ri_defaults[ ]` in defaults.c is given below:

```
uint8r_t ri_defaults[] =
{ //   0     1     2     3     4     5     6     7
   0x00, 0x10,                            // address and length
   EQUATION | CE_10MHZ, 0x34,             // 2000-2001
   0x03,                                  // 2002 enable xfer_busy, RTC
interrupt.
   0x00, MPU_FREQ, CONFIG1_VAL, 0x00,     // 2003..6
   0x24,                                  // 2007 enable PLL_OK 0x24
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 2008-F. DIO
Resources, etc.
   0x11, 0x01,                        // address and length, index 18
   0x40,                              // Default value for the analog clock
adjustment
   0x1C, 0x03,                        // address and length, index 21
   0x04, 0x00, 0x00,                  // Default value for PREG and QREG
   0x20, 0x0A,                        // address and length, index 26
   0x00, 0x23,                        // 2020-1.  all states, all on
   0x00, 0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,  // 2022-9
   0x30, 0x2B,                            // address and length; LCD buffer/
DIO outputs
   // Clear segments visible on LCD or not DIOs; Leave DIOs as outputs
   0x00, 0x00,                            // 2030..1=seg42,43
   0x80, 0x80, 0x80, 0x80, 0x00, 0x00, // 2030..7=seg44..49=dio24..29
   0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, // 2038..F=seg50-
57=dio30..37
   0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, //
2040..7=58..65=dio38..45
   0x00, 0x80, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, //
2048..F=66..73=dio46..53
   0x80, 0x80, 0x90, 0x90, 0x00, 0x00, 0x00, 0x00, //
2050..7=74..81=dio54..61
   0x00, 0x00, 0x00,                               //
2058..A=dio40,41,blink19&18
   0x60, 0x08,                                     // address and length
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // 2060-7. RTM.
   0x70, 0x01,                                     // address and length
   0x00,                                           // 2070
   0x80, 0x02,                                // address and length
   0x4B, 0x51,                                // 2080-1. 10ms Pulse_Width,
Pulse_Interval.
   0x90, 0x0B,                           // address and length
   0x10, 0x32, 0x54, 0x76, 0x98, 0x00,  // 2090 normal sequence
   0x1A, 0x3B, 0x54, 0x76, 0x98,        // Alt sequence reads temp, vbat
and aux
   0x9D, 0x01,                // address and length
   0x07,                      // 209D: 6 states
   0xA7, 0x09,                // address and length
   0x00,                      // 20A7: boot size
   0x31, 0x00, 0x00,                   // 20A8-A. CE5, WAKE, TMUX.
   0x00, 0x22, 0x22, 0x00, 0x00,       // 20AB..F; differential, diff. chop
enabled
   0xFD, 0x03,                         // address and length
   0x00, 0x00, 0x00,                   // 20FD-F. TRIMSEL, TRIMX, TRIM.
   0, 0 // ending record
};
```

Adapting Meter Code and Keil Settings for Larger or Smaller CE Codes and Data

CE codes with sizes larger than the size used for the existing build (as stated in `NumCeCode`) need the following adjustments:

- Adjust *`CE_LCTN_ADR`* downwards

- 'Target' tab accessible with 'Options for Target' in the Keil PK51 environment must specify the flash space needed for the CE code. This entry must correspond to the new choice of *`CE_LCTN_ADR`*.

CE codes with larger or smaller allocation for XRAM need the following changes:

- 'Target' tab accessible with 'Options for Target' in the Keil PK51 environment must specify the XRAM space needed for the CE code.
- 'BL51 Misc' tab accessible with 'Options for Target' in the Keil PK51 environment must specify the location in flash memory.
- `CE_DATA_SIZE` needs to be updated in source code (ce653x.h).
- CE code should be checked for the existence of data beyond `CE_DATA_SIZE`.
- Move the location of the `totals` structure in meter.c in XRAM if CE data extends beyond 0x04FF.

Downloading Code to the 71M653x Flash Memory

CE codes that operate the CE at 10MHz require that the *`CE10MHZ`* bit (bit 3) at I/O RAM address 0x2000 is set while downloading code using the ADM51 ICE. The normal procedure before downloading code is to just clear the entire byte at 0x2000, which clears the *`CE_E`* bit (bit 4) in the process. For CE codes that operate the CE at 10MHz, it is helpful to write the value 0x08 into I/O RAM address 0x2000 before downloading code, which maintains the *`CE10MHZ`* bit (bit 3).

71M654x Family of Electricity Metering ICs

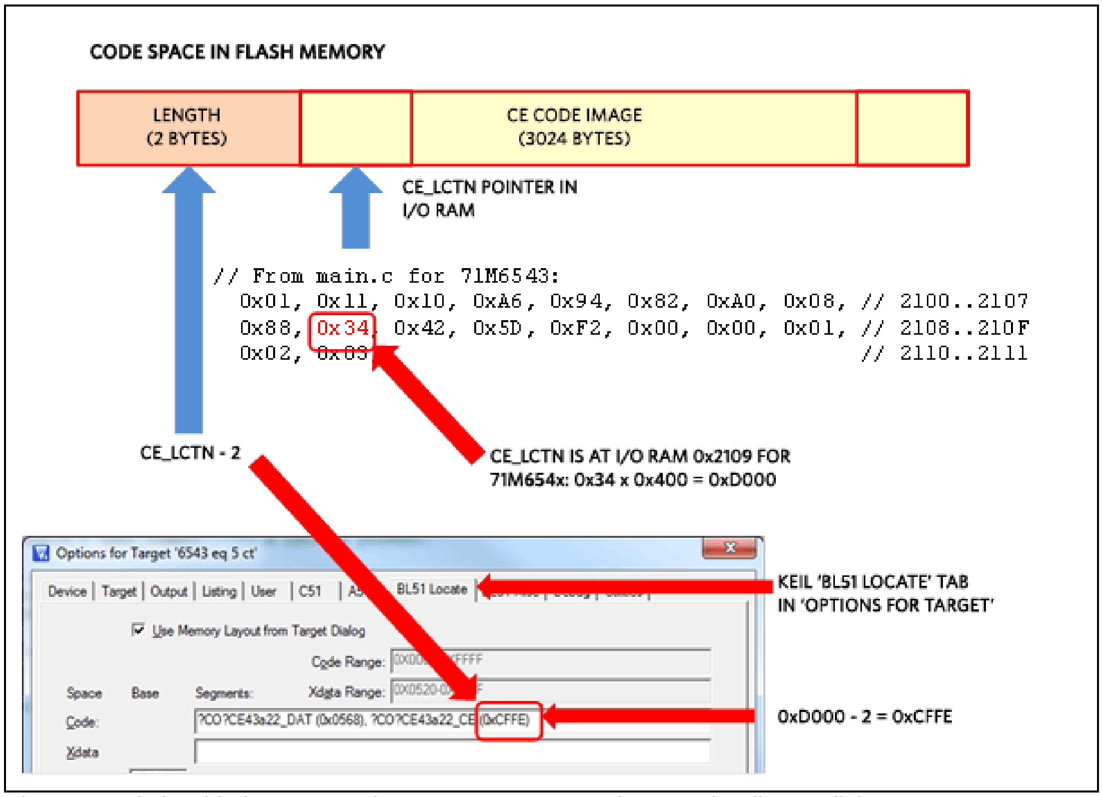
General Comments

The considerations for the 71M654x codes are very similar to the 71M653x codes. A typical CE code for the 71M6543 is shown below. Note that the same variable, `NumCeCode`, is used to specify the length of the code.

```
// File: ce43a22_ce.c
//
// CE Program Image File (8051 C format)
const short code NumCeCode=1512;        // The number of words in the
'CeCode' array.
const unsigned char code CeCode[]={
 0xec,0xff, 0x20,0x48, 0x74,0xd8, 0x8c,0x48, 0x74,0xd8, 0x8c,0x48,
0x74,0xd8, 0x8c,0x48,
 0x74,0xd8, 0x8c,0x48, 0x74,0xd8, 0x8c,0x48, 0x74,0xd8, 0x8c,0x48,
0xed,0xff, 0x20,0x4c,
 0x74,0xd9, 0x8c,0x4c, 0x74,0xd9, 0x8c,0x4c, 0x74,0xd9, 0x8c,0x4c,
0x74,0xd9, 0x8c,0x4c,
 0x74,0xd9, 0x8c,0x4c, 0x74,0xd9, 0x8c,0x4c, 0x20,0x5c, 0x8f,0x74,
0x20,0x5e, 0x8f,0x76,
 0x20,0x60, 0x8f,0x78, 0x20,0x5d, 0x8f,0x7a, 0x20,0x5f, 0x8f,0x7b,
0x20,0x61, 0x8f,0x7c,
 0x26,0x20, 0x8f,0x38, 0xf0,0x38, 0x20,0x5d, 0x68,0x5f, 0x69,0x61,
0x8f,0x38, 0x20,0xf9,
 0x05,0xfa, 0x15,0xf9, 0x8f,0xf9, 0x20,0xfa, 0x05,0x38, 0x11,0xf9,
0x8f,0xfa, 0x20,0xb0,
 0xd0,0x4f, 0xf0,0xf9, 0x2b,0x4f, 0x8f,0x38, 0x30,0x38, 0x81,0x38,
0x30,0x38, 0x86,0x38,
 0x20,0x38, 0x00,0xb2, 0x87,0xb3, 0x8f,0xb2, 0xb7,0xb2, 0x20,0xb3,
0x00,0xb4, 0x8f,0x38,
```

In this example, the specified length of 1512 words amounts to 3024 bytes (0x0BD0 bytes). In the 71M654x ICs, the location for the CE code in flash is coded into I/O RAM address 0x2109 (*`CE_LCTN`*).

Figure 6 shows the relationship between code space, *`CE_LCTN`* pointer, and the entry in the Keil BL51 linker tab.

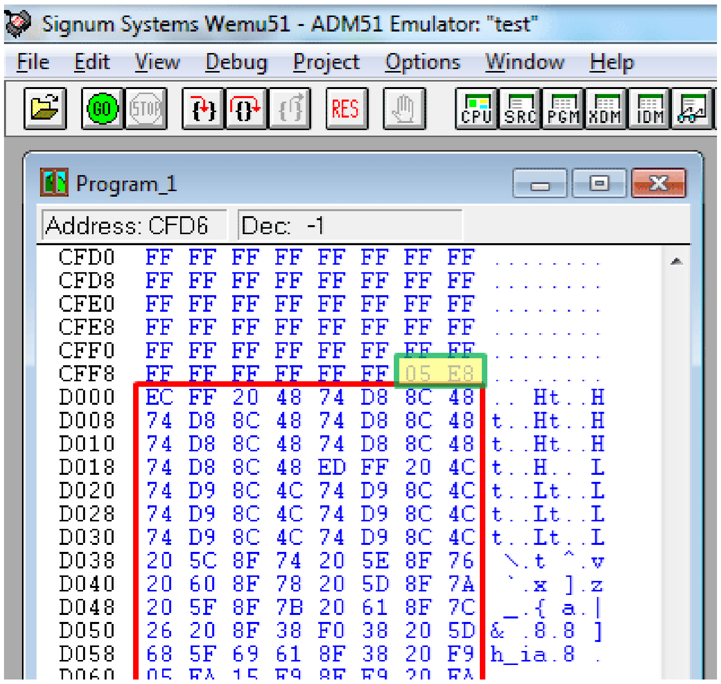

CODE SPACE IN FLASH MEMORY
LENGTH (2 BYTES)
CE CODE IMAGE (3024 BYTES)
CE_LCTN POINTER IN I/O RAM
CE_LCTN - 2
CE_LCTN IS AT I/O RAM 0x2109 FOR 71M654x: 0x34 x 0x400 = 0xD000
KEIL 'BL51 LOCATE' TAB IN 'OPTIONS FOR TARGET'
0xD000 - 2 = 0xCFFE
Options for Target '6543 eq 5 ct'
Device
Target
Output
Listing
User
C51
BL51 Locate
Use Memory Layout from Target Dialog
Space
Base
Segments:
Code:
Xdata

```
// From main.c for 71M6543:
  0x01, 0x11, 0x10, 0xA6, 0x94, 0x82, 0xA0, 0x08, // 2100..2107
  0x88, 0x34, 0x42, 0x5D, 0xF2, 0x00, 0x00, 0x01, // 2108..210F
  0x02, 0x03                                      // 2110..2111
```

Code: ?CO?CE43a22_DAT (0x0568), ?CO?CE43a22_CE (0xCFFE)

Figure 6. Relationship between code space, CE_LCTN pointer, and Keil BL51 linker.

The code window from the Signum emulator for flash area 0xCFD0 to 0xD05F is shown in **Figure 7**. The entry 0x05E8 (1512 decimal) at location 0xCFFE corresponds to `NumCeCode`. The CE code image itself starts at location 0xD000.



```
Address: CFD6   Dec: -1
CFD0  FF FF FF FF FF FF FF FF  ........
CFD8  FF FF FF FF FF FF FF FF  ........
CFE0  FF FF FF FF FF FF FF FF  ........
CFE8  FF FF FF FF FF FF FF FF  ........
CFF0  FF FF FF FF FF FF FF FF  ........
CFF8  FF FF FF FF FF FF 05 E8  ........
D000  EC FF 20 48 74 D8 8C 48  .. Ht..H
D008  74 D8 8C 48 74 D8 8C 48  t..Ht..H
D010  74 D8 8C 48 74 D8 8C 48  t..Ht..H
D018  74 D8 8C 48 ED FF 20 4C  t..H.. L
D020  74 D9 8C 4C 74 D9 8C 4C  t..Lt..L
D028  74 D9 8C 4C 74 D9 8C 4C  t..Lt..L
D030  74 D9 8C 4C 74 D9 8C 4C  t..Lt..L
D038  20 5C 8F 74 20 5E 8F 76   \.t ^.v
D040  20 60 8F 78 20 5D 8F 7A   `.x ].z
D048  20 5F 8F 7B 20 61 8F 7C   _.{ a.|
D050  26 20 8F 38 F0 38 20 5D  & .8.8 ]
D058  68 5F 69 61 8F 38 20 F9  h_ia.8 .
D060  05 FA 15 F9 8F F9 20 FA
```

Figure 7. Code window in the Signum emulator.

CE Data Image

In the 71M654x demo codes, the CE data image is copied into XRAM by a routine in Main.c():

```
    // Set CE RAM to default.
    memcpy((uint8x_t*)CE_DATA_BASE, CeData, (4 * NumCeData));
```

As opposed to the 71M653x demo code, the 71M654x demo code does not base the copy operation on the minimum of `CE_DATA_SIZE` and `NumCeData`. The 71M654x demo code copies the whole data image, as defined by `NumCeData` into XRAM.

Adapting I/O RAM Locations

To establish the environment for the CE code, the I/O RAM entries controlling multiplexer settings, FIR length, gain control, input pin configuration (single-ended/differential), clock selection, etc. must be examined and changed, if necessary.

In the 71M654x demo codes, the I/O RAM registers are controlled by a structure named `io_ram_table[]`. This structure can be found in main.c and does not contain named elements, which means that the bytes must be counted manually starting at I/O RAM address 0x2100, to determine their meaning.

The listing on the following page shows default entries for a typical code and their meaning. The structure contains four bytes at the beginning that determine the start address (0x2100) and the length of the entries (0x0013).

Note that *CE_LCTN* is set at 0x03, which places the CE code image at 0x0C00 in flash memory. This is typical for Maxim 71M654x demo codes. Allocating a location in the lower range of the flash memory makes the code compatible with both the 32KB and 64KB flash size variants of the ICs.

```
const uint8r_t io_ram_table[] =
{
```

```
// Configuration without M6000 (Remote sensor)
   // Wh = max(VA * IA); IA is shunt
   0x21, 0x00, 0x00, 0x13,                           // address (0x2100) and
length (0x13 = 19 bytes decimal)
   0x01, 0x11, 0x11, 0x11, 0x12, 0xA0, 0x00, 0x09,   // MUX_DIV=0,
MUX10_SEL=1, MUX7_SEL=1, MUX6_SEL=1,
                                                     // MUX5_SEL=1,
MUX4_SEL=1
                                                     // MUX3_SEL=1,
MUX2_SEL=2, MUX1_SEL=A, MUX0SEL=0
                                                     // EQU=0, no chop, no
RTM, CE_E=0, SUM_SAMPS[12:8]=9
   0xD8, 0x03, 0x42, 0x5D, 0x3A, 0x00, 0x00, 0x01,   // SUM_SAMPS[7:0]=0xD8
? SUM_SAMPS = 2520 decimal
                                                     // CE_LCTN=3,
PLS_MAXWDTH = 0x42, PLS_INTERVAL = 0x5D ? 93 decimal
                                                     // DIFFn_E = 1, FIR_LEN
= 01, RTM
   0x02, 0x03, 0x04,                                 // RTM, last byte not
used. There is no I/O RAM address 0x2112
   #if POWER_CONTROL
   0x22, 0x00, 0x00, 0x01,                           // address (0x2200) and
length (0x01)
   0x32,                                             // Clocks:
MCK=19.6608MHz, MPU=0.6644MHz, ADC=0.66MHz
   #endif
   0x21, 0x00, 0x00, 0x01,                           // address (0x2100) and
length (0x01)
   0x30,                                             // set mux_div after
ADC
```

Adapting Meter Code and Keil Settings for Larger CE Codes and Data

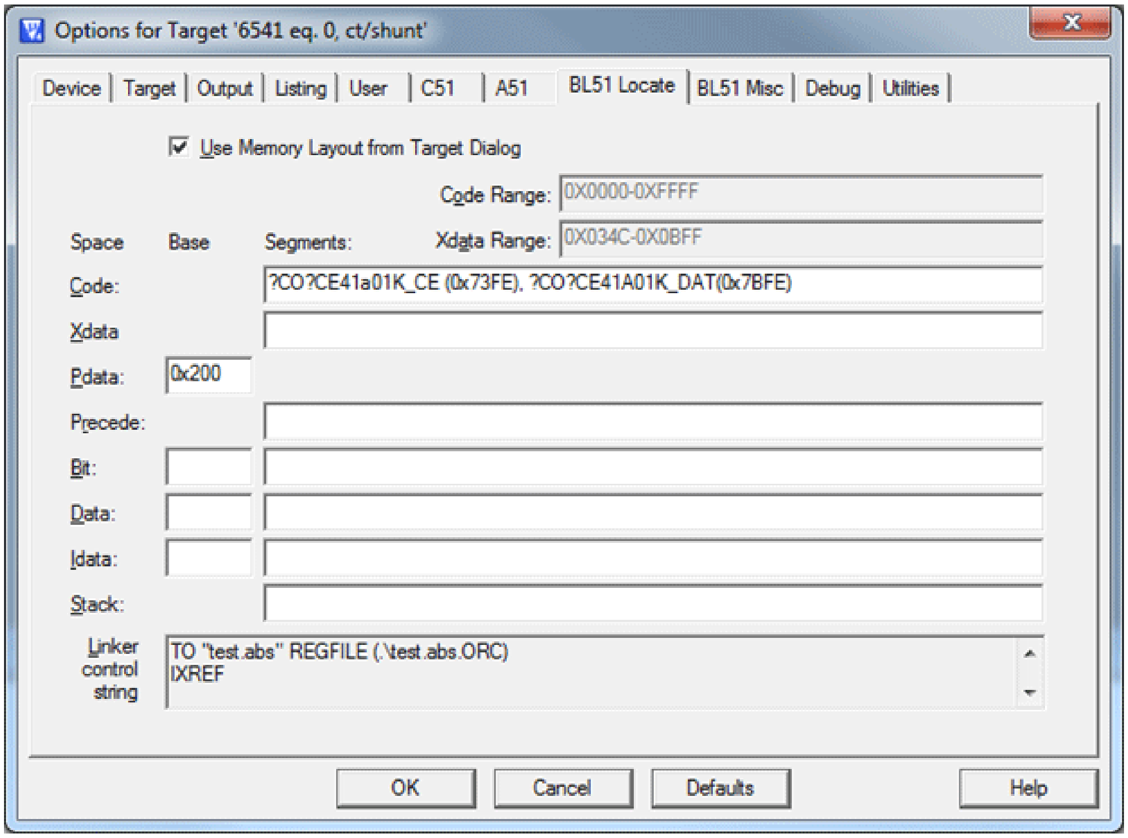
CE codes with sizes larger than the size used for the existing build (as stated in `NumCeCode`) need the following adjustments:

- Adjust the source code in main.c, byte *CE_LCTN* at 0x2109.
- ‘Target’ tab accessible with ‘Options for Target’ in the Keil PK51 environment must specify the flash space needed for the CE code. This entry must correspond to the new choice of *CE_LCTN*.
- The structure `totals` as used in the 71M653x codes has been renamed to `reg_data` for the 71M654x codes, which is part of the `nvm_s` structure. `reg_data`, is set to 0x2800, which is the start of the NV-RAM data field in I/O RAM.

CE codes with larger allocation for XRAM need the following changes:

- ‘Target’ tab accessible with ‘Options for Target’ in the Keil PK51 environment must specify the XRAM space needed for the CE code.
- `CE_DATA_SIZE` needs to be updated in source code (ce654x.h).
- CE code should be checked for the existence of data beyond `CE_DATA_SIZE`.

Note the window for allocating code space in flash memory looks different for the 71M654x family compared to the 71M653x family (see **Figure 8**):



☑ Use Memory Layout from Target Dialog

Code Range: 0X0000-0XFFFF

Xdata Range: 0X034C-0X0BFF

Space	Base	Segments:
Code:		?CO?CE41a01K_CE (0x73FE), ?CO?CE41A01K_DAT(0x7BFE)
Xdata		
Pdata:	0x200	
Precede:		
Bit:		
Data:		
Idata:		
Stack:		

Linker control string:

```
TO "test.abs" REGFILE (.\test.abs.ORC)
IXREF
```

Figure 8. BL51 Locate window in Keil µVision® 4.

Note that other fixed flash addresses are used in 71M654x demo code:

```
// Flash addresses.
#define FLASH_CE_CAL   0xfc00 // Address of CE calibration.
#define FLASH_MPU_CAL  0xf800 // Address of MPU calibration.
```

The flash address 0xFC00 is reserved for saving key CE parameters such as calibration coefficients, `WRATE`, `CECONFIG`, etc. with the]U command.

The flash address 0xF800 is reserved for saving key MPU parameters, i.e. the parameters that can be entered by the user with the ')' key word, e.g. when the user enters ")3=+2080" to set the IMAX value to 208A.

Care must be taken not to overlap CE code or data images with these fixed flash locations.

Troubleshooting

The code image generated with the techniques detailed above may not always work initially. Some troubleshooting tips are listed below.

A good approach to troubleshooting meter or demo code involves the following steps:

1. Connect the ADM51 emulator to the target and start the WEMU51 application.
2. Erase the target flash memory.
3. Clear the XRAM space using the SET command of the emulator. For example, the command SET /MDATA 0

3FF 0 fills the XRAM memory from 0 to 0x3FF with zeroes.

4. Load the generated code image into the flash using the File – Load command in WEMU51.
5. Check the location of CE code image in the Program Window (Did the image load into the expected location?, Is the image complete? Is there any overlap?)
6. Check the location of CE data image in the Program Window (Did the image load into the expected location? Is the image complete? Is there any overlap?)
7. Click RES and then GO in the WEMU51 user interface. Allow the code run for 1 to 2 seconds, then halt the code execution by clicking on the STOP button.
8. Examine the XRAM near 0x0000 in the Data Window. The code should now have copied the CE data image to XRAM. Addresses near 0x0000 are overwritten with the ADC samples, but addresses from 0x0010 and up should reflect the CE data image. Check if the complete data image has been copied, i.e. the range from 0x0000 to `NumCeData` or at least up to `CE_DATA_SIZE`).
9. Examine the I/O RAM, i.e. addresses starting at 0x2000 (71M653x) or 0x2000 and 0x2100 (71M654x). Check if the pointer for the CE code location has been initialized properly (*CE_LCTN*).
10. Examine the I/O RAM registers that control the AFE, such as *MUX_DIV*, *SUM_PRE*, multiplexer slot assignments, *FIR_LEN*, *PRE_E*, and control registers for differential/single-ended operation of analog pins. All settings must correspond to the settings specified in the application note for the CE code.

µVision is a registered trademark of ARM, Inc.
Keil is a registered trademark and registered service mark of ARM Limited.
Signum Systems is a registered trademark of Signum Systems Corporation.

Related Parts		
71M6531D	Energy Meter ICs	Free Samples
71M6531F	Energy Meter ICs	Free Samples
71M6532D	Energy Meter ICs	Free Samples
71M6533	Energy Meter ICs	Free Samples
71M6534	Energy Meter ICs	Free Samples
71M6541D	Energy Meter ICs	Free Samples
71M6541F	Energy Meter ICs	Free Samples
71M6541F-DB	Demo Board for the 71M6541F	
71M6541G	Energy Meter ICs	Free Samples
71M6542F	Energy Meter ICs	Free Samples
71M6542G	Energy Meter ICs	Free Samples
71M6543F	Energy Meter ICs	Free Samples
71M6543F-DB	Demo Board for the 71M6543F	
71M6543G	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 5751: http://www.maximintegrated.com/an5751
APPLICATION NOTE 5751, AN5751, AN 5751, APP5751, Appnote5751, Appnote 5751
© 2013 Maxim Integrated Products, Inc.
Additional Legal Notices: http://www.maximintegrated.com/legal