AN211122

Getting Started with FM4 Development

Author: James Trudeau

Associated Part Family: All FM4 parts

Related Application Note & Code Examples: See the FM4 Family Resources section

AN211122 introduces you to the FM4 family of 32-bit general-purpose microcontrollers, The FM4 family is based on the ARM® Cortex®-M4 processor core, which features DSP and Floating Point Unit (FPU) functions. This note provides an overview of hardware features and capabilities, firmware development, and the multitude of technical resources available to you. This application note uses the FM4 S6E2GM Pioneer Kit as a foundation.

Contents

1 FM4 Family Overview

Cypress' FM4 is a family of 32-bit, general-purpose and high performance microcontrollers based on the ARM Cortex-M4 processor with FPU and DSP functionality. FM4 microcontrollers operate at frequencies up to 200 MHz and support a diverse set of on-chip peripherals for motor control, factory automation, and home appliance applications. The portfolio delivers low-latency, reliable, machine-to-machine communication required for network-computing technologies that advance design and manufacturing.

There are five series within the FM4 Family. Table 1-1 list some of the defining characteristics of each family.

Table 1-1. FM4 Family Series

Series	Max CPU Speed	Flash up to…	SRAM up to…	GPIOs up to…
MB9B	160 MHz	1 MB	128 KB	100
S6E2C	200 MHz	2 MB	256 KB	190
S6E2D	160 MHz	384 KB	36 KB 512 KB VRAM	154
S6E2G	180 MHz	1 MB	192 KB	153
S6E2H	160 MHz	512 KB	64 KB	100

The Cypress FM4 family of 32-bit, general-purpose MCUs is designed for applications that require advanced, high-speed computing performance such as:

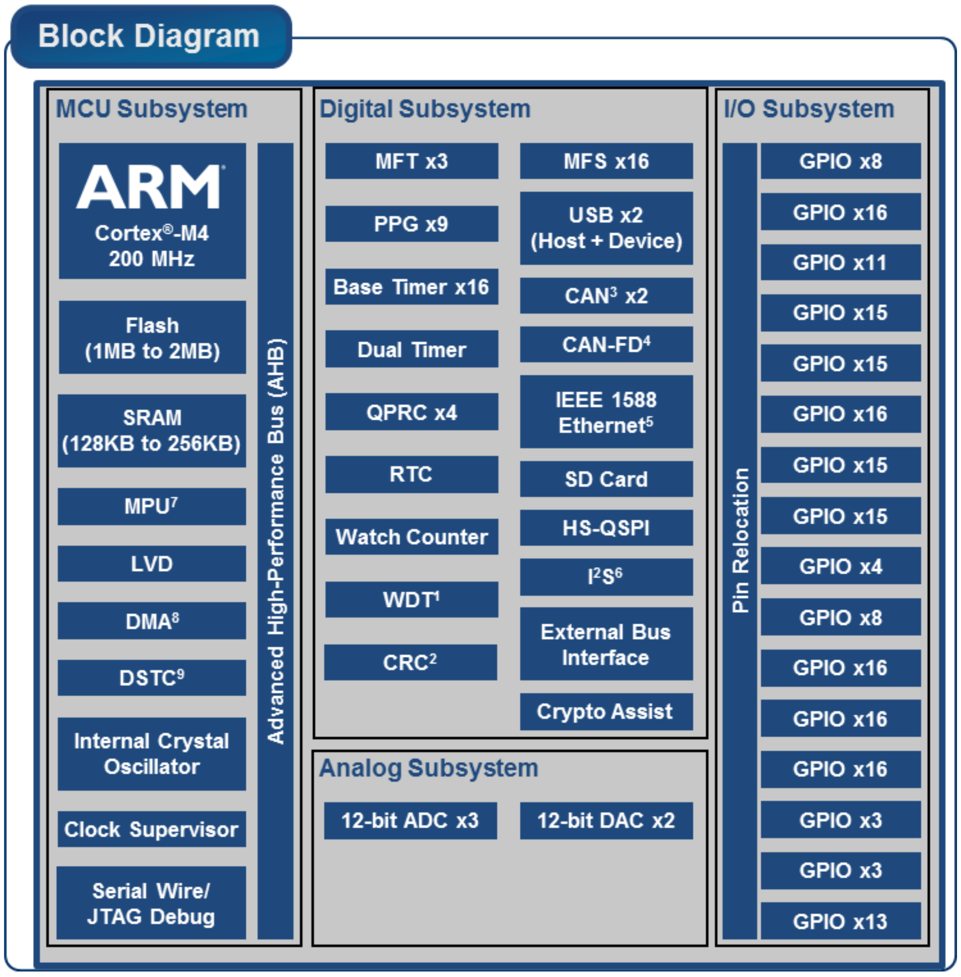
- General-purpose inverters for accurate control of motor speed
- Servomotors
- Programmable logic controllers and other industrial equipment
- Medical products
- Inverter-based home appliances such as washing machines and air conditioners
- Meters
- Printers

Key features and capabilities include:

- Frequency up to 200 MHz
- Operating voltage: 2.7-5.5 V
- Low power consumption: 365 µA/MHz, 1.5 µA in real-time clock (RTC) mode
- Flash: 384 KB – 2 MB
- RAM: 36 KB – 256 KB
- 48- through 216-pin packages

Figure 1-1 shows the block diagram for the FM4 S6E2C series as an example. The FM4 family provides a range of peripherals, such as Ethernet, CAN, USB2.0, DMA, and A/D Converters. Peripheral support varies among the series in this family. For details on variations within each series, such as pin packages, voltage range, or peripheral support, review the Product Selector Guide.

Figure 1-1. Block Diagram for the FM4 S6E2CC-Series



1. Watchdog Timer
2. Cyclical Redundancy Check
3. Controller Area Network
4. Controller Area Network with Flexible Data-Rate
5. Ethernet Communications with IEEE 1588 Precision Time Protocol (PTP) Standard
6. Inter-IC Sound
7. Memory Protection Unit
8. Direct Memory Access
9. Descriptor System Transfer Controller

2 Firmware Development

Cypress provides the Peripheral Driver Library (PDL) to simplify FM4 software development. The PDL eases software development for the extensive set of peripherals in the FM4 family by reducing the need to understand peripheral registers and bit structures. You configure the library for the particular device, and then use API calls to initialize and use a peripheral. In addition to the FM4 family, the PDL supports Cypress FM0+ processors and peripherals. Using the PDL makes it easier to port code from one family to the other.

For developers who wish to work at the register level, Cypress also provides template projects configured for each particular processor series. You can use these template projects as a starting point for your software development. The template projects provide support for several additional IDEs. You can also use the PDL source code as a learning tool. Combined with study of the appropriate data sheet and peripheral manual, you can learn the information you need to use the peripheral. Refer to the FM4 Family Resources section of this document for links to the extensive technical documentation available.

2.1 Software Tools Overview

There is a large ecosystem of tool providers that support the FM4 family. There is an extensive list of choices at the Support Tools for FM4 Family web page. Click the link for a particular series to see what’s available.

This document walks you through the process of building an example project with either the IAR Embedded Workbench or the Keil µVision IDEs.

2.2 Peripheral Driver Library Overview

The PDL is a superset of all the code required to build any driver for any supported device. This superset design means:

- All APIs needed to initialize, configure and use a peripheral are available.
- The PDL includes error checking, because a peripheral may not be present on the selected device.

The superset design means the PDL is useful across all devices, whatever peripherals are available. This enables code to maintain compatibility across platforms where peripherals remain present.

If you configure the PDL to include a peripheral that is unavailable on the specified hardware, your project will fail at compile time, rather than at runtime. The PDL configuration logic knows the target processor and removes the peripheral register headers for unsupported peripherals from the build.

Before writing code to use a peripheral, consult the datasheet for the particular series or device to confirm support for the peripheral.

2.2.1 Getting and Installing the PDL

Some FM4 Cypress starter kits, such as the FM4 S6E2GM Pioneer Kit, install the PDL as part of the kit. If you install such a kit, the default location for the PDL is *My Documents/Cypress/FM_PDL_<version number>*.

If your kit does not install the PDL, you can download the PDL Installer from the FM4 product page. On the **Tools & Software** tab, look for the link to download the PDL.

2.2.2 PDL Structure

The PDL is organized into several folders. Table 2-1 shows the PDL folder structure.

Table 2-1. PDL Folder Structure

Path\Folder	Description
common	Common header files
doc	PDL documentation
driver	Driver source code and headers
example	Example code
template\ARM	Keil project and configuration files for use with example code
template\IAR	IAR project and configuration files for use with example code
template\source	Source files, replace with files from example code, or write your own code
template\source\backup	Copy of original source files, use to restore template
utility	Various utility files

When you use the PDL, typically you do not modify the common files, documentation, or the driver code.

To develop your own code (or use example code) you work with three files in the *template* folder:

- *pdl_device.h*
- *pdl_user.h*
- *main.c*

There is a single IAR Embedded Workbench IDE project for all the example code, located in *template\IAR*. Similarly, there is a single Keil µVision IDE project for all the example code, located in *template\ARM*.

The Build and Run a PDL Project section explains how to use the source files and project files.

2.2.3 Using PDL Example Code

The PDL installation includes more than 100 code examples. Most peripherals have multiple examples. Each example demonstrates the basic initialization and configuration for the peripheral. It includes a *ReadMe* file that provides a description of what the example demonstrates, and how to use the example.

The Build and Run a PDL Project section of this document guides you step-by-step through the process of configuring the PDL, building a project using an IDE, and debugging the code on the hardware, using a kit example.

To use the example code, you:

- Copy two files, *main.c* and *pdl_user.h* from the *example* folder into the *template\source* folder, replacing the existing files of the same name.
- Build and download the code to the target.

If you wish to restore the template to its initial state, copy the *main.c* and *pdl_user.h* files from the *template\source\backup* folder into the *template\source* folder.

2.2.4 Writing Your Own Code Using the PDL

You typically begin with an existing project, like an example in a starter kit, which already has the PDL files added, project options set correctly, and peripherals selected for the example.

When developing firmware with PDL, you perform the following tasks:

- If necessary, modify project options in your IDE to target the correct device.
- Specify the target device and package in *pdl_device.h*.
- Configure the PDL in *pdl_user.h.*
- Add your firmware in *main.c,* or additional source files as required.
- Build and download the code to the target.

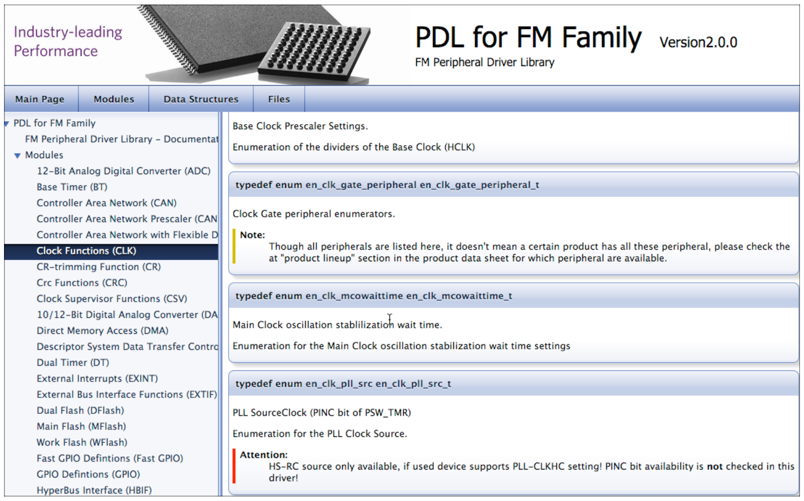
As noted, *pdl_device.h*, *pdl_user.h* and *main.c* are provided for you in the example code. The project files for both IAR and Keil tools use the PDL source code and these three files.

2.2.5 Other PDL Resources

The PDL *User Quick Start Guide* is installed with the PDL, in the *doc* folder.

API documentation is available in the *doc\doxy* path. Figure 2-1 shows the home page, *index.html*.

Figure 2-1. PDL Documentation


Industry-leading
Performance
PDL for FM Family
Version2.0.0
FM Peripheral Driver Library
Main Page
Modules
Data Structures
Files
PDL for FM Family
FM Peripheral Driver Library – Documentat
Modules
12-Bit Analog Digital Converter (ADC)
Base Timer (BT)
Controller Area Network (CAN)
Controller Area Network Prescaler (CAN
Controller Area Network with Flexible D
Clock Functions (CLK)
CR-trimming Function (CR)
Crc Functions (CRC)
Clock Supervisor Functions (CSV)
10/12-Bit Digital Analog Converter (DA
Direct Memory Access (DMA)
Descriptor System Data Transfer Contro
Dual Timer (DT)
External Interrupts (EXINT)
External Bus Interface Functions (EXTIF)
Dual Flash (DFlash)
Main Flash (MFlash)
Work Flash (WFlash)
Fast GPIO Defintions (Fast GPIO)
GPIO Defintions (GPIO)
HyperBus Interface (HBIF)

Base Clock Prescaler Settings.

Enumeration of the dividers of the Base Clock (HCLK)

typedef enum en_clk_gate_peripheral en_clk_gate_peripheral_t

Clock Gate peripheral enumerators.

Note:
Though all peripherals are listed here, it doesn't mean a certain product has all these peripheral, please check the at "product lineup" section in the product data sheet for which peripheral are available.

typedef enum en_clk_mcowaittime en_clk_mcowaittime_t

Main Clock oscillation stablilization wait time.

Enumeration for the Main Clock oscillation stabilization wait time settings

typedef enum en_clk_pll_src en_clk_pll_src_t

PLL SourceClock (PINC bit of PSW_TMR)

Enumeration for the PLL Clock Source.

Attention:
HS-RC source only available, if used device supports PLL-CLKHC setting! PINC bit availability is **not** checked in this driver!

Refer to the FM4 Family Resources section of this document for additional links to helpful information.

2.3 MCU Template Projects

An MCU template project does not include or use the PDL, so you must write code to access registers directly. A template project is a simple software framework that serves as a starting point to develop any software (user applications, third-party algorithms, peripheral drivers, or others) running on Cypress FM family microcontrollers. A template project contains startup files, the peripheral header file, the GPIO header file, the system initialization file, the CMSIS System View Description (SVD) file, and all IDE options set correctly such as those for the compiler, assembler, linker, and debugger.

There is a template project file for every FM4 series. Unlike the PDL, which has a single IDE project file that you configure for your target processor, a template project is already configured. However, the *main()* function contains an empty loop. You write your own code to initialize, configure, and use any peripheral.

FM4 template projects are available for download here.

Several FM4 starter kits have associated MCU template projects. The FM4 Family Resources section has links to the kits. The contents on the kit pages vary but typically include a template project, a large collection of code examples built for the kit, and other resources.

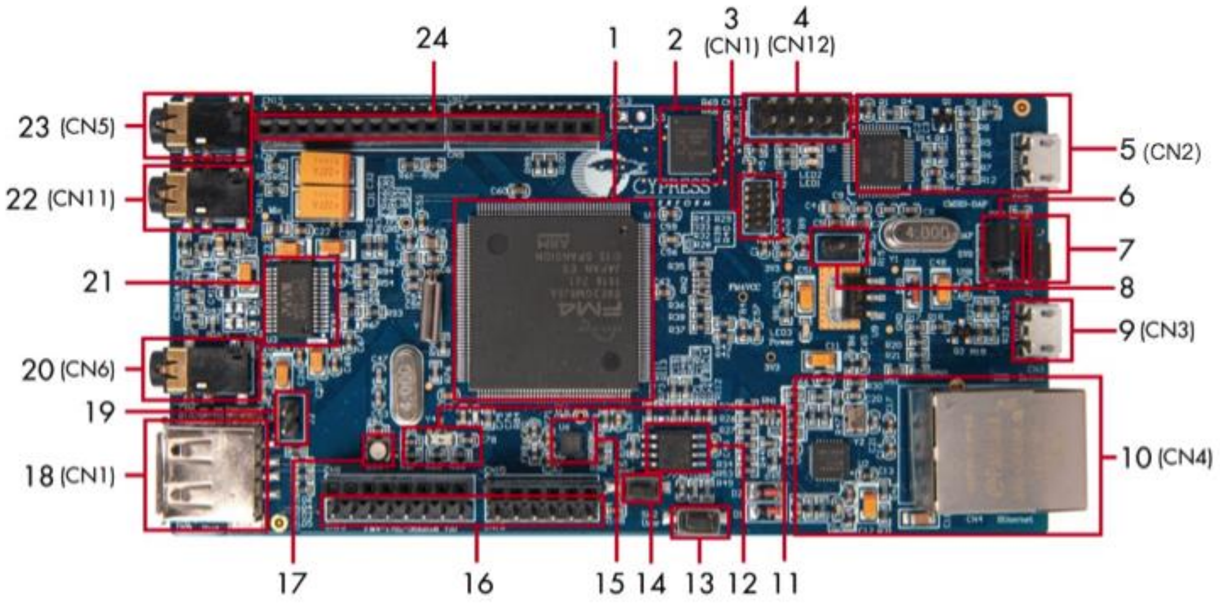
3 Build and Run a PDL Project

In this section you build and run an example PDL project, using either of these IDEs:

- IAR Embedded Workbench version 7.3 or later
- Keil µVision, version 5 or later

These instructions use the FM4 S6E2GM Pioneer Kit, along with an example project from that kit. Figure 3-1 shows the kit board.

Figure 3-1. The FM4 S6E2GM Pioneer Kit



1. Cypress FM4 MCU S6E2GM8J0A
2. Cypress 4-Mb SRAM
3. 10-pin JTAG connector (CN1)
4. Multicon connector (CN12)
5. Programmer and Debugger (CMSIS-DAP) (CN2)
6. Power supply resource select (J4)
7. Serial programming mode select (J3)
8. Programming mode jumper of MB9F312K (J1)
9. USB device connector (CN3)
10. Ethernet PHY and RJ45 connector (CN4)
11. Phototransistor
12. Cypress 32-Mb Quad-SPI NOR flash
13. User button
14. Reset button
15. Accelerometer
16. Additional GPIO headers (CN15-CN18)
17. RGB LED
18. USB host connector (CN1)
19. Programming mode jumper of S6E2GM (J2)
20. Line-in jack (CN6)
21. Stereo codec
22. Microphone jack (CN11)
23. Headphone jack (CN5)
24. Arduino™ interface (CN7-CN10)

For either IDE you will perform three major tasks:

- Examine the IDE's project options for your hardware.
- Configure the PDL for your hardware.
- Build and run the code example.

The example projects that come with a Starter Kit have project options set correctly, because the kit has a specific processor. If you use an example project from a kit, you do not need to configure the IDE.

3.1 Before You Begin

Ensure that you have hardware on which to run the code, the PDL, and a development environment.

Hardware

This application note uses the FM4 S6E2GM Pioneer Kit. If you prefer to use different hardware, you must adapt the instructions as required.

PDL

Ensure that you have the PDL installed.

The S6E2GM Kit installs the PDL. If you do not have the PDL, see Getting and Installing the PDL. These instructions assume the PDL is installed in the default location, *Documents\Cypress\FM_PDL_<version>*.

Development Environment

Ensure that you have installed a development environment. These instructions cover:

- IAR Embeded Workbench version 7.3 or later
- Keil µVision, version 5 or later

Because the kit has a particular processor, all project options are set correctly.

When targeting a different processor or board, you must update project options accordingly. The precise options, and how to set them, vary based on your IDE. The PDL Quick Start Guide (installed with the PDL) addresses this issue in detail. Among the project options you may need to update would be the target device, the linker configuration file, and the flash configuration file.

3.2 Building with IAR Embedded Workbench

This section guides you through the steps required to build, download, and run a PDL project with IAR tools. These instructions use the DMA example from the FM4 S6E2GM Pioneer Kit. This code creates a buffer in memory, initializes the values in that buffer, copies the data to a different location using DMA, and then tests that the two buffers are identical.

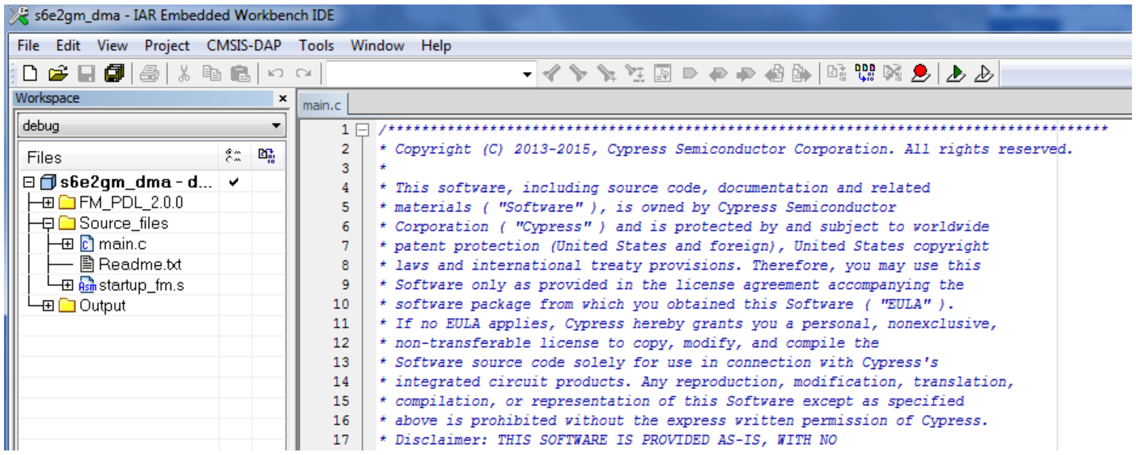
1. **Open the IAR workspace for the DMA example.**

   The file name is *s6e2gm_dma.eww.* In a default kit installation, the path to the file is:

   \Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\s6e2gm_dma\IAR

   Double-click the file, and the IAR Embedded Workbench IDE opens, as shown in Figure 3-2.

Figure 3-2. Open the IAR Workspace for the DMA Example



```
/*********************************************************************************
* Copyright (C) 2013-2015, Cypress Semiconductor Corporation. All rights reserved.
*
* This software, including source code, documentation and related
* materials ( "Software" ), is owned by Cypress Semiconductor
* Corporation ( "Cypress" ) and is protected by and subject to worldwide
* patent protection (United States and foreign), United States copyright
* laws and international treaty provisions. Therefore, you may use this
* Software only as provided in the license agreement accompanying the
* software package from which you obtained this Software ( "EULA" ).
* If no EULA applies, Cypress hereby grants you a personal, nonexclusive,
* non-transferable license to copy, modify, and compile the
* Software source code solely for use in connection with Cypress's
* integrated circuit products. Any reproduction, modification, translation,
* compilation, or representation of this Software except as specified
* above is prohibited without the express written permission of Cypress.
* Disclaimer: THIS SOFTWARE IS PROVIDED AS-IS, WITH NO
```

2. **Configure the PDL for the target processor.**

   For this example you don't need to do anything; the code already configures the PDL correctly. This step ensures you are aware of how this works and what you need to do when developing your own project.

   You specify the target processor in *pdl_device.h.*

   A. Open pdl_device.h.

      Choose **File** > **Open** > **File…** and navigate to the *Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\s6e2gm_dma* folder. Locate the file and open it. It appears in the source window.

   B. Examine the code.

      Look for the code that defines the values of *PDL_MCU_SERIES* and *PDL_MCU_PACKAGE*. At the time of this writing, the code at line 57 defines the series targeted in this example.

```
#define PDL_MCU_SERIES        PDL_DEVICE_SERIES_S6E2GMX
```

      The code at line 68 defines the package targeted in this example.

```
#define PDL_MCU_PACKAGE       PDL_DEVICE_PACKAGE_S6_J
```

      Note that the comments in the file also point you to the location where you can find the available definitions.

   C. Close the file.

      You don't need to make any changes.

3. **Configure PDL functionality.**

   For this example you don't need to do anything; the code already configures the PDL correctly. This step ensures you are aware of how this works and what you need to do when developing your own project.

   You configure PDL functionality in *pdl_user.h*.

   A. Open pdl_user.h.

      Use **File** > **Open** > **File…** and navigate to the *Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\s6e2gm_dma* folder. Locate the file and open it. It appears in the source window.

   B. Examine the contents of the file.

      This file contains a long list of *#define* statements, one for each configurable feature of the PDL. To enable a feature, you define it as *PDL_ON*. Otherwise it is *PDL_OFF*. Pertinent to this example, find the line of code that sets *PDL_PERIPHERAL_ENABLE_DSTC* to the value *PDL_ON*. At the time of this writing, it is at line 112 of the file.

```
// DMA
#define PDL_PERIPHERAL_ENABLE_DMA0              PDL_ON
#define PDL_PERIPHERAL_ENABLE_DMA1              PDL_OFF
```

      To learn more about the PDL and its capabilities, refer to the PDL documentation. In a default installation, the documentation is here: *C:\Program Files (x86)\Cypress\FM PDL\<version number>\doc*

   C. Close the file. You don't need to make any changes.

4. **Connect the board to your PC.**

   If you have not already done so, follow the directions in the kit. Use the provided cable and connect to the CN2 (CMSIS-DAP) port on the board. There are two similar ports side by side. See Figure 3-1 if you're not sure which is which.

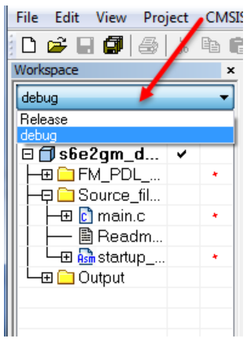
   When properly connected, **LED3 Power** on the board will be green.

5. **Select the debug build for the project.**

   The debug build has options set to generate debugger symbols. You will use the debugger in subsequent steps.

   Click the Build drop-down menu and select **debug**. See Figure 3-3.

Figure 3-3. Select the Debug Build for the Project

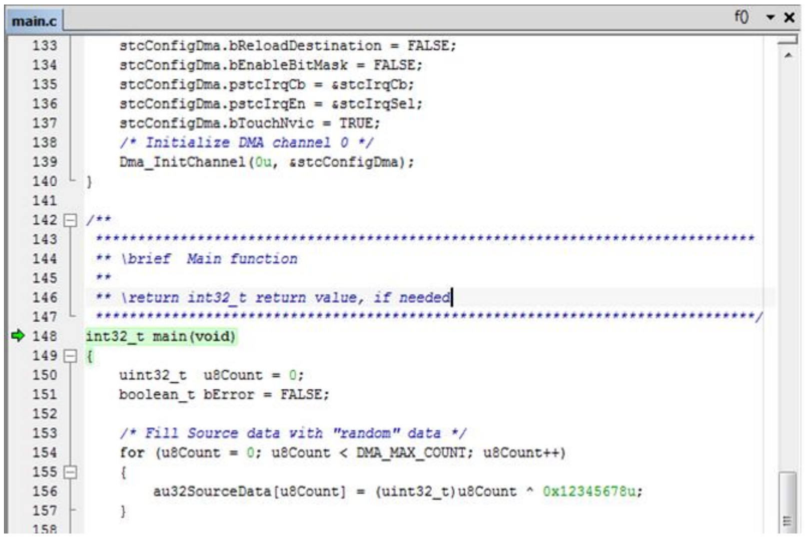


6. **Download the code and launch the debugger.**

   Choose **Project** > **Download & Debug**.

   The code will compile, link, download to the board, and launch the debugger. You should see no warnings or errors. The program counter will be halted at the first line of *main()* as shown in Figure 3-4.

Figure 3-4. Debugger Halted at the *main()* Function



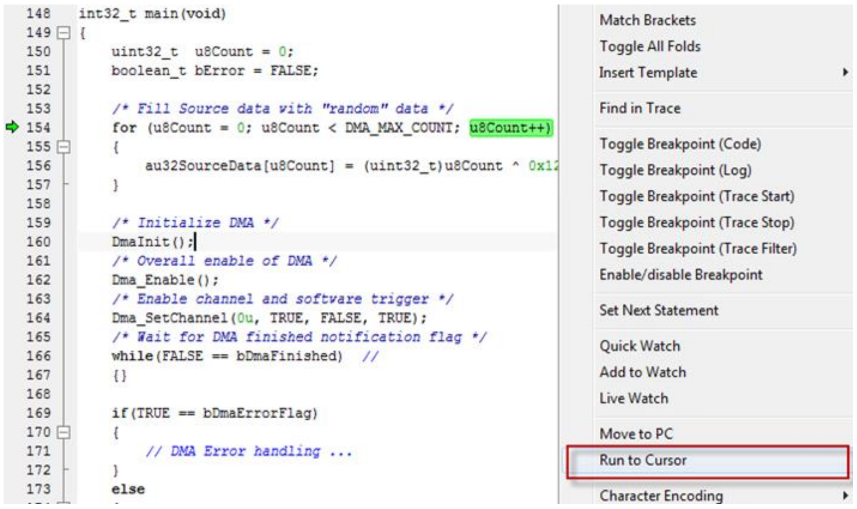
```
    stcConfigDma.bReloadDestination = FALSE;
    stcConfigDma.bEnableBitMask = FALSE;
    stcConfigDma.pstcIrqCb = &stcIrqCb;
    stcConfigDma.pstcIrqEn = &stcIrqSel;
    stcConfigDma.bTouchNvic = TRUE;
    /* Initialize DMA channel 0 */
    Dma_InitChannel(0u, &stcConfigDma);
}

/**
 ******************************************************************************
 ** \brief  Main function
 **
 ** \return int32_t return value, if needed
 ******************************************************************************/
int32_t main(void)
{
    uint32_t  u8Count = 0;
    boolean_t bError = FALSE;

    /* Fill Source data with "random" data */
    for (u8Count = 0; u8Count < DMA_MAX_COUNT; u8Count++)
    {
        au32SourceData[u8Count] = (uint32_t)u8Count ^ 0x12345678u;
    }
```

7. **Initialize the source data buffer.**

   A. Run the code to the call to `DmaInit()`.

   You can right-click that line and choose **Run to Cursor**. In this example that code is at line 160. See Figure 3-5.

Figure 3-5. Using the **Run to Cursor** Command

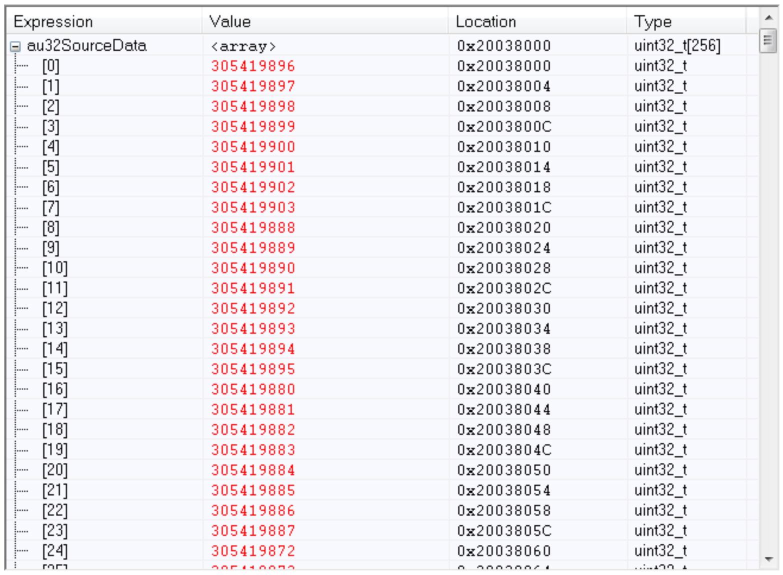


```
148  int32_t main(void)
149  {
150      uint32_t  u8Count = 0;
151      boolean_t bError = FALSE;
152
153      /* Fill Source data with "random" data */
154      for (u8Count = 0; u8Count < DMA_MAX_COUNT; u8Count++)
155      {
156          au32SourceData[u8Count] = (uint32_t)u8Count ^ 0x12
157      }
158
159      /* Initialize DMA */
160      DmaInit();
161      /* Overall enable of DMA */
162      Dma_Enable();
163      /* Enable channel and software trigger */
164      Dma_SetChannel(0u, TRUE, FALSE, TRUE);
165      /* Wait for DMA finished notification flag */
166      while(FALSE == bDmaFinished)  //
167      {}
168
169      if(TRUE == bDmaErrorFlag)
170      {
171          // DMA Error handling ...
172      }
173      else
```

   B. Observe the source data values in the Watch window.

   In the Watch window, click the + control next to *au32SourceData* to see the contents of the buffer. The buffer should be full of data. See Figure 3-6.

Figure 3-6. Initial Contents of the Source Data Buffer



Expression	Value	Location	Type
au32SourceData	<array>	0x20038000	uint32_t[256]
[0]	305419896	0x20038000	uint32_t
[1]	305419897	0x20038004	uint32_t
[2]	305419898	0x20038008	uint32_t
[3]	305419899	0x2003800C	uint32_t
[4]	305419900	0x20038010	uint32_t
[5]	305419901	0x20038014	uint32_t
[6]	305419902	0x20038018	uint32_t
[7]	305419903	0x2003801C	uint32_t
[8]	305419888	0x20038020	uint32_t
[9]	305419889	0x20038024	uint32_t
[10]	305419890	0x20038028	uint32_t
[11]	305419891	0x2003802C	uint32_t
[12]	305419892	0x20038030	uint32_t
[13]	305419893	0x20038034	uint32_t
[14]	305419894	0x20038038	uint32_t
[15]	305419895	0x2003803C	uint32_t
[16]	305419880	0x20038040	uint32_t
[17]	305419881	0x20038044	uint32_t
[18]	305419882	0x20038048	uint32_t
[19]	305419883	0x2003804C	uint32_t
[20]	305419884	0x20038050	uint32_t
[21]	305419885	0x20038054	uint32_t
[22]	305419886	0x20038058	uint32_t
[23]	305419887	0x2003805C	uint32_t
[24]	305419872	0x20038060	uint32_t

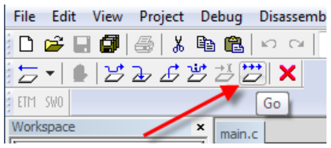
C. Optional: Observe the destination data values.

The destination buffer is filled with zeros. If you want to confirm this, examine the contents of *au32DestinationData* in the Watch window.

8. Use DMA to copy the data to the destination data buffer.

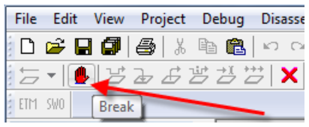
A. Choose **Debug** > **Go** or click the **Go** button as shown in Figure 3-7. The code will take a moment to execute completely.

Figure 3-7. Click the **Go** Button to Run the Code



B. Choose **Debug** > **Break** or click the **Break** button as shown in Figure 3-8.

Figure 3-8. Click the **Break** Button to Stop the Code



C. Observe the values of *au32DestinationData* in the Watch window. They should match the contents of the SourceData buffer.

The application used the DMA peripheral to copy the data from the source buffer to the destination buffer. You have successfully built a simple application using the PDL.

3.3 Building with Keil µVision

This section guides you through the steps required to build, download, and run a project that uses the PDL. These instructions use the DMA example from the FM4 S6E2GM Pioneer Kit. This code creates a buffer in memory, initializes the values in that buffer, copies the data to a different location using DMA, and then tests that the two buffers are identical.

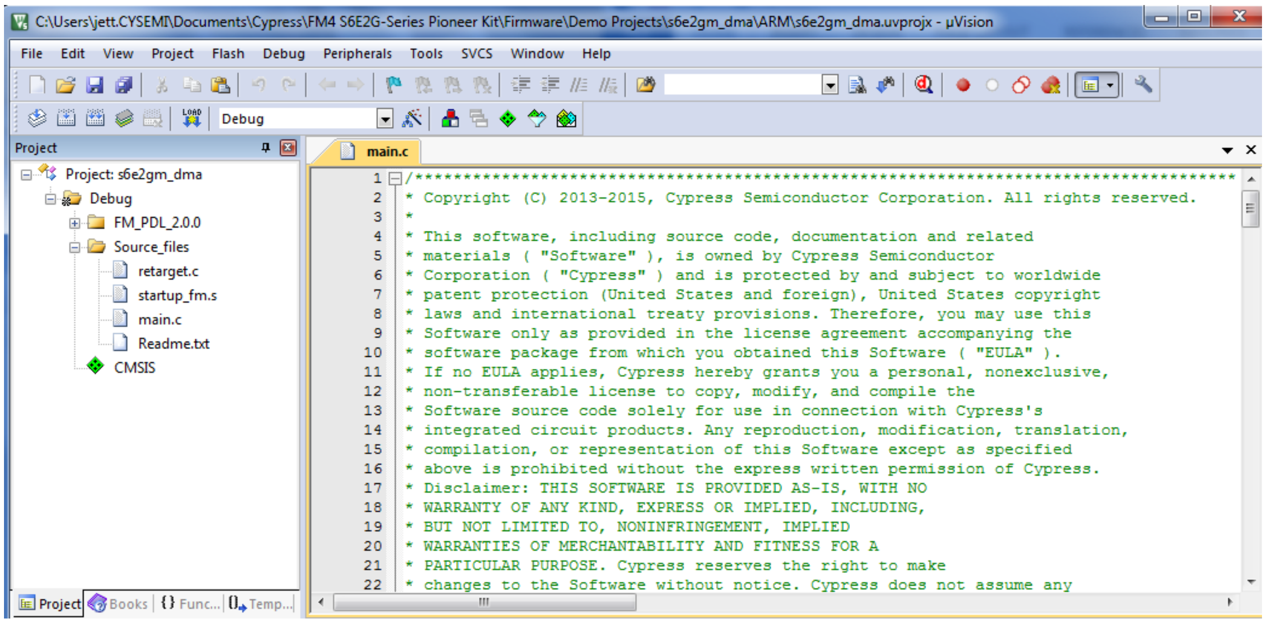
1. **Open the µVision project for the DMA example.**

   The file name is *s6e2gm_dma.uvprojx.* In a default kit installation, the path to the file is:

   \Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\s6e2gm_dma\ARM

   Double-click the file, and the µVision IDE opens, as shown in Figure 3-9.

Figure 3-9. Open the µVision project for the DMA Example



```
/*******************************************************************************
* Copyright (C) 2013-2015, Cypress Semiconductor Corporation. All rights reserved.
*
* This software, including source code, documentation and related
* materials ( "Software" ), is owned by Cypress Semiconductor
* Corporation ( "Cypress" ) and is protected by and subject to worldwide
* patent protection (United States and foreign), United States copyright
* laws and international treaty provisions. Therefore, you may use this
* Software only as provided in the license agreement accompanying the
* software package from which you obtained this Software ( "EULA" ).
* If no EULA applies, Cypress hereby grants you a personal, nonexclusive,
* non-transferable license to copy, modify, and compile the
* Software source code solely for use in connection with Cypress's
* integrated circuit products. Any reproduction, modification, translation,
* compilation, or representation of this Software except as specified
* above is prohibited without the express written permission of Cypress.
* Disclaimer: THIS SOFTWARE IS PROVIDED AS-IS, WITH NO
* WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING,
* BUT NOT LIMITED TO, NONINFRINGEMENT, IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
* PARTICULAR PURPOSE. Cypress reserves the right to make
* changes to the Software without notice. Cypress does not assume any
```

2. **Configure the PDL for the target processor.**

   For this example you don't need to do anything; the code already configures the PDL correctly. This step ensures you are aware of how this works and what you need to do when developing your own project.

   You specify the target processor in *pdl_device.h.*

   A. Open pdl_device.h.

   Use **File** > **Open** and navigate to the *Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\ Firmware\Demo Projects\s6e2gm_dma* folder. Locate the file and open it. It appears in the source window.

   B. Examine the code.

   Look for the code that defines the values of *PDL_MCU_SERIES* and *PDL_MCU_PACKAGE*. At the time of this writing, the code at line 57 defines the series targeted in this example.

```
#define PDL_MCU_SERIES          PDL_DEVICE_SERIES_S6E2GMX
```

   The code at line 68 defines the package targeted in this example.

```
#define PDL_MCU_PACKAGE         PDL_DEVICE_PACKAGE_S6_J
```

   Note that the comments in the file also point you to the location where you can find the available definitions.

   C. Close the file.

   You don't need to make any changes.

3. **Configure PDL functionality.**

   For this example you don't need to do anything; the code already configures the PDL correctly. This step ensures you are aware of how this works and what you need to do when developing your own project.

   You configure PDL functionality in *pdl_user.h*.

   A. Open pdl_user.h.

      Use **File** > **Open** and navigate to the *Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\ Firmware\Demo Projects\s6e2gm_dma* folder. Locate the file and open it. It appears in the source window.

   B. Examine the contents of the file.

      This file contains a long list of *#define* statements, one for each configurable feature of the PDL. To enable a feature, you define it as *PDL_ON*. Otherwise it is *PDL_OFF*. Pertinent to this example, find the line of code that sets *PDL_PERIPHERAL_ENABLE_*DSTC to the value *PDL_ON*. At the time of this writing, it is at line 112 of the file.

```
// DMA
#define PDL_PERIPHERAL_ENABLE_DMA0                PDL_ON
#define PDL_PERIPHERAL_ENABLE_DMA1                PDL_OFF
```

      To learn more about the PDL and its capabilities, refer to the PDL documentation. In a default installation, the documentation is here: *C:\Program Files (x86)\Cypress\FM PDL\<version number>\doc*

   C. Close the file.

      You don't need to make any changes.

4. **Connect the board to your PC.**

   If you have not already done so, follow the directions in the starter kit. Use the provided cable and connect to the CN2 (CMSIS-DAP) port on the board. There are two similar ports side by side. See Figure 3-1 if you're not sure which is which.

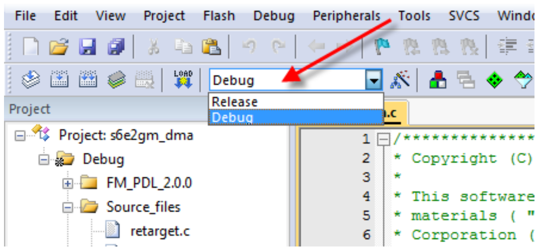
   When properly connected, **LED3 Power** on the board will be green.

5. **Select the debug build for the project.**

   The debug build has options set to generate debugger symbols. You will use the debugger in subsequent steps.

   Click the Build drop-down menu and select **Debug**. See Figure 3-10.

Figure 3-10. Select the Debug Build for the Project



```
/**************
* Copyright (C)
*
* This software
* materials ( "
* Corporation (
```

6. **Build the code.**

   Choose **Project** > **Build Target**.

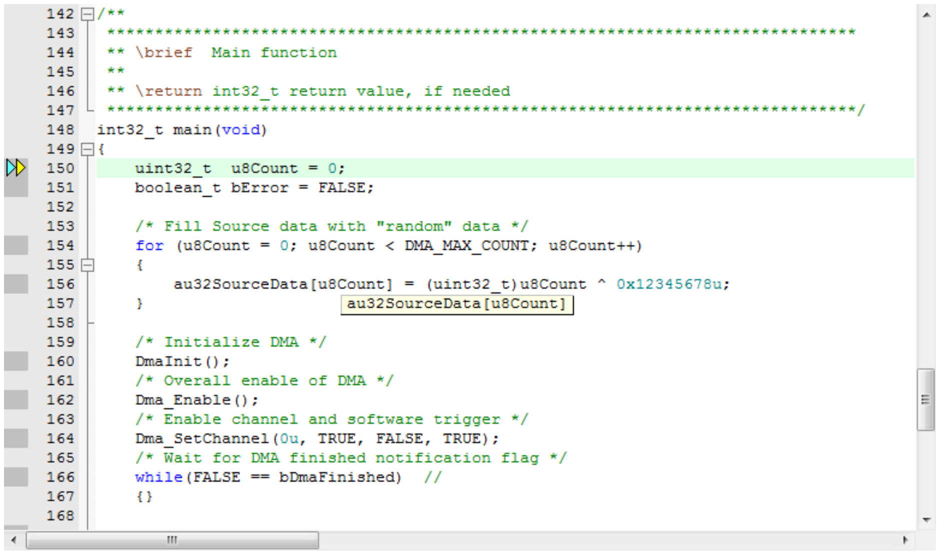
   The IDE compiles and links the code. You should see no warnings or errors.

7. **Download the code and launch the debugger.**

   Choose **Debug** > **Start/Stop Debug Session**.

   The code downloads to the board and the debugger launches. The program counter will be stopped at the first line of *main()* as shown in Figure 3-11.

Figure 3-11. Debugger Stopped at the *main()* Function



```
/**
 ********************************************************************************
 ** \brief  Main function
 **
 ** \return int32_t return value, if needed
 ********************************************************************************/
int32_t main(void)
{
    uint32_t  u8Count = 0;
    boolean_t bError = FALSE;

    /* Fill Source data with "random" data */
    for (u8Count = 0; u8Count < DMA_MAX_COUNT; u8Count++)
    {
        au32SourceData[u8Count] = (uint32_t)u8Count ^ 0x12345678u;
    }                    au32SourceData[u8Count]

    /* Initialize DMA */
    DmaInit();
    /* Overall enable of DMA */
    Dma_Enable();
    /* Enable channel and software trigger */
    Dma_SetChannel(0u, TRUE, FALSE, TRUE);
    /* Wait for DMA finished notification flag */
    while(FALSE == bDmaFinished)  //
    {}
```

8. **Configure a Watch window.**

   In this step you open a Watch window in the debugger and add two arrays so you can watch what happens as the code executes. Figure 3-12 shows the result.

   A. Add *au32SourceData* to the Watch window.

      Double-click *au32SourceData* in the source code to select that variable name.

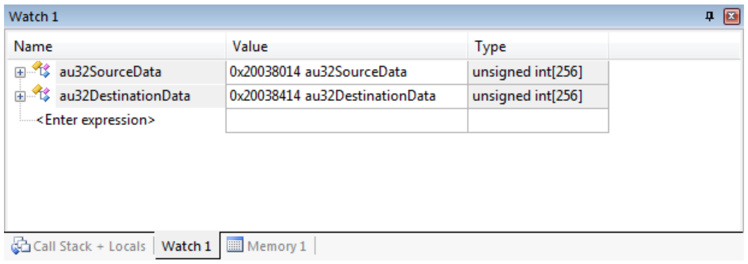
      Right-click the selected variable name and choose **Add 'au32SourceData' to** > **Watch 1**.

   B. Add au32DestinationData.

      Double-click the variable *au32DestinationData*. This selects just the variable name.

      Right-click the selected variable name and choose **Add 'au32DestinationData' to** > **Watch 1**.

Figure 3-12. Configure a Watch Window



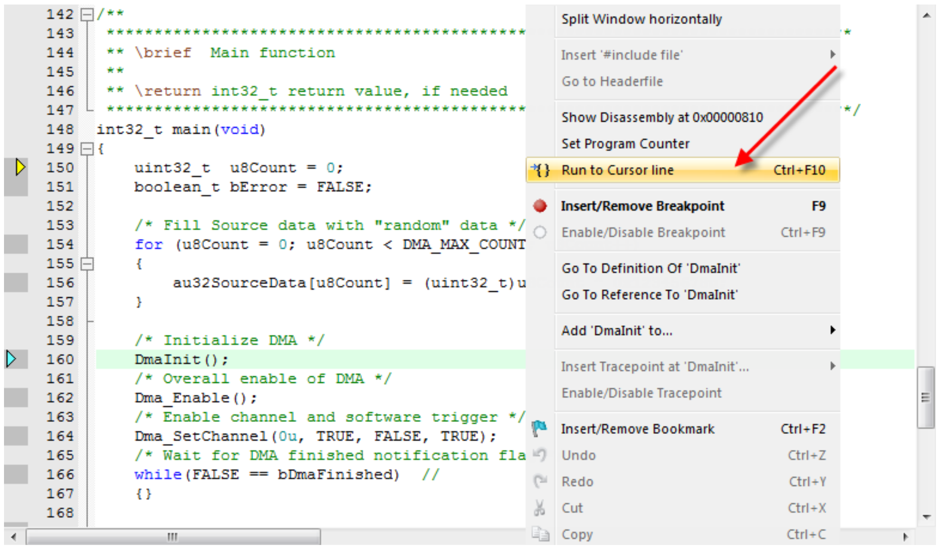
Name	Value	Type
au32SourceData	0x20038014 au32SourceData	unsigned int[256]
au32DestinationData	0x20038414 au32DestinationData	unsigned int[256]
<Enter expression>		

9. Initialize the source data buffer.

A. Run the code to the call to `DmaInit()`.

You can right-click that line and choose **Run to Cursor Line**. At the time of this writing, that code is at line 160. See Figure 3-13.

Figure 3-13. Using the **Run to Cursor line** Command



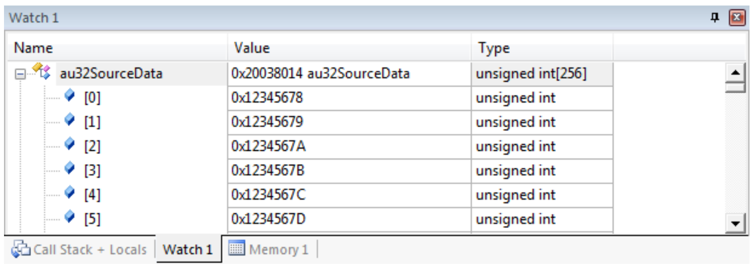
```
142 /**
143  *********************************************
144  ** \brief  Main function
145  **
146  ** \return int32_t return value, if needed
147  *********************************************
148 int32_t main(void)
149 {
150     uint32_t  u8Count = 0;
151     boolean_t bError = FALSE;
152
153     /* Fill Source data with "random" data */
154     for (u8Count = 0; u8Count < DMA_MAX_COUNT
155     {
156         au32SourceData[u8Count] = (uint32_t)u
157     }
158
159     /* Initialize DMA */
160     DmaInit();
161     /* Overall enable of DMA */
162     Dma_Enable();
163     /* Enable channel and software trigger */
164     Dma_SetChannel(0u, TRUE, FALSE, TRUE);
165     /* Wait for DMA finished notification fla
166     while(FALSE == bDmaFinished)   //
167     {}
168
```

Split Window horizontally
Insert '#include file'
Go to Headerfile
Show Disassembly at 0x00000810
Set Program Counter
Run to Cursor line Ctrl+F10
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Go To Definition Of 'DmaInit'
Go To Reference To 'DmaInit'
Add 'DmaInit' to...
Insert Tracepoint at 'DmaInit'...
Enable/Disable Tracepoint
Insert/Remove Bookmark Ctrl+F2
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C

B. Observe the source data values in the Watch window.

In the Watch window, click the + control next to *au32SourceData* to see the contents of the buffer. The buffer should be full of data. See Figure 3-14.

Figure 3-14. Initial Contents of the Source Data Buffer



Watch 1

Name	Value	Type
au32SourceData	0x20038014 au32SourceData	unsigned int[256]
[0]	0x12345678	unsigned int
[1]	0x12345679	unsigned int
[2]	0x1234567A	unsigned int
[3]	0x1234567B	unsigned int
[4]	0x1234567C	unsigned int
[5]	0x1234567D	unsigned int

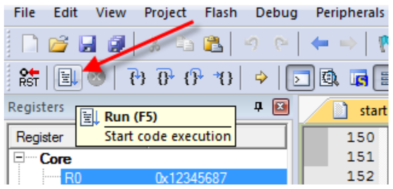
C. Optional: Observe the destination data values.

The destination buffer is filled with zeros. If you want to confirm this, examine the contents of *au32DestinationData* in the Watch window.

10. Use DMA to copy the data to the destination data buffer.

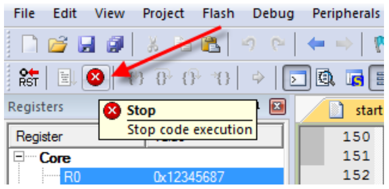
A. Choose **Debug** > **Run** or click the **Run** button as shown in Figure 3-15. The code will take a moment to execute completely.

Figure 3-15. Click the **Go** Button to Run the Code



B. Choose **Debug** > **Stop** or click the **Stop** button as shown in Figure 3-16.

Figure 3-16. Click the **Stop** Button to Stop the Code



C. Observe the values of *au32DestinationData* in the Watch window. They should match the contents of the SourceData buffer.

The application used the DMA peripheral to copy the data from the source buffer to the destination buffer. You have successfully built a simple application using the PDL.

4 Programming Embedded Flash

Most IDEs are capable of programming embedded flash. However, a flash programmer may be your preferred or only solution in some cases. This section shows you how to program embedded flash using either a serial or a USB connection. The USB connection requires USB support on the target.

These instructions target the S6E2GM processor found in the FM4 S6E2GM Pioneer Kit. See Figure 3-1 for a key to the components on the hardware.

If you are not using this kit, you must modify the instructions to fit your specific target hardware. Check the documentation provided with your board for jumper configuration and other details.

4.1 Before You Begin

Ensure you have a programmer. These instructions cover the tools listed here.

- FLASH MCU Programmer for FM0+/FM3/FM4
- FLASH USB Direct Programmer

There are two ways to use these flash programmers: single step or automatic programming (full operation). Note that only single step works for secured flash devices that need chip erase. In this example we use automatic programming.

You also need a file to download. The file format must be either Motorola S-Record or Intel-HEX. This example uses a Motorola S-Record file provided with the kit.

When you build code in an IDE, you may be able to generate an S-Record or Intel-HEX format file. Consult the documentation for your IDE. In IAR Embedded Workbench, use the **Project** > **Options** > **Output Converter** panel. In Keil µVision, use the **Project** > **Options for Target** > **Output** panel.

4.2 Using the FLASH MCU Programmer

These instructions assume you have downloaded and installed the starter kit so you have access to the required S-Record file. If not, locate an S-Record or Intel-HEX file, and use that file instead.

1. **Configure the jumpers.**

   Make sure the jumpers on the FM4 S6E2G-Series Pioneer board are placed according to Table 4-1.

   Table 4-1. Jumper Settings for S6E2GM programming by FLASH MCU Programmer.

Jumper	Default	Program by Serial	Purpose
J1	Open	Open	Sets MB9AF312K (CMSIS-DAP) to run mode.
J2	Open	Closed	Sets S6E2GM to programming mode.
J3	Pin 1 to Pin 2	Pin 1 to Pin 2	Sets for UART programming mode.
J4	Pin 1 to Pin 2	Pin 1 to Pin 2	Get power from the CMSIS-DAP port

2. **Provide power to the board.**

   Connect the USB cable to the CN2 port. The Power LED (LED3) should be lit (green). See Figure 3-1 for the location of the correct port.

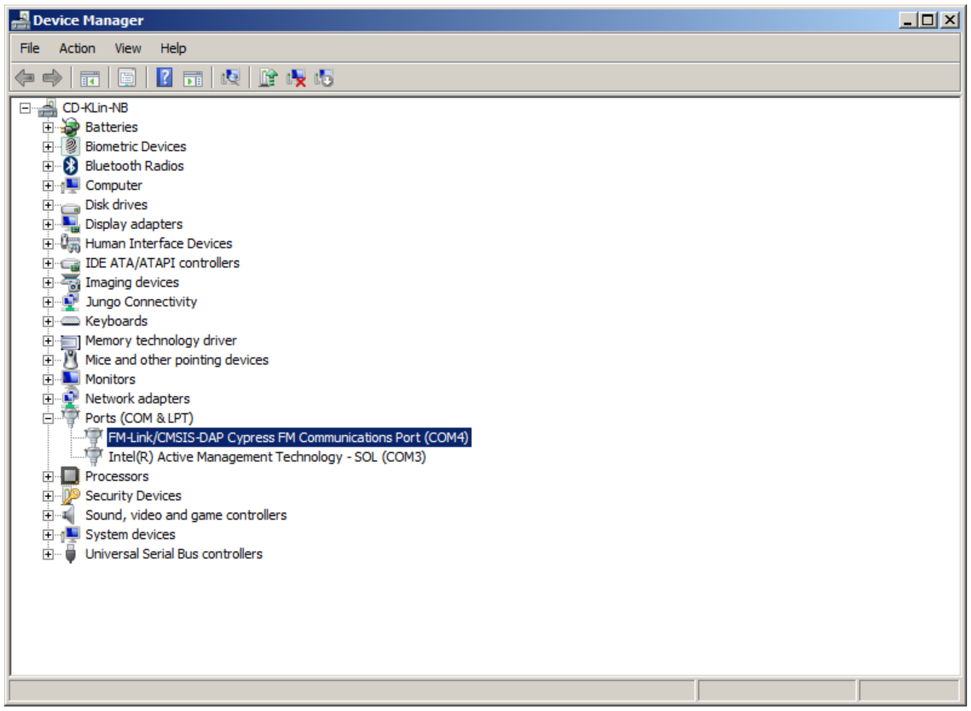
3. **Identify the COM port in use.**

   You need to know which COM port your board is connected to. You will use this number to configure the Flash programmer.

   You may see a popup notification that contains this information as you connect the board. If you don't see it, or don't know the COM port, open the Device Manager and look for **Ports (COM & LPT).** You should see an entry for FM-Link/CMSIS-DAP. The COM port is listed at the end of that entry, as shown in Figure 4-1.

   Remember the number.

Figure 4-1. Identify the Com Port In Use



```
CD-KLin-NB
    Batteries
    Biometric Devices
    Bluetooth Radios
    Computer
    Disk drives
    Display adapters
    Human Interface Devices
    IDE ATA/ATAPI controllers
    Imaging devices
    Jungo Connectivity
    Keyboards
    Memory technology driver
    Mice and other pointing devices
    Monitors
    Network adapters
    Ports (COM & LPT)
        FM-Link/CMSIS-DAP Cypress FM Communications Port (COM4)
        Intel(R) Active Management Technology - SOL (COM3)
    Processors
    Security Devices
    Sound, video and game controllers
    System devices
    Universal Serial Bus controllers
```

4. **Launch the FLASH MCU Programmer.**

   You can do this from the Start menu, using this path:

   Start Menu > All Programs > Cypress > FLASH MCU Programmer > FM0+ FM3 FM4

5. **Configure the programmer.**

   In this and the next step you configure the programmer for the target device. See Figure 4-2.

   A. Set **Target MCU** to S6E2GM8H/J.

   B. Set Crystal Frequency to 4MHz.

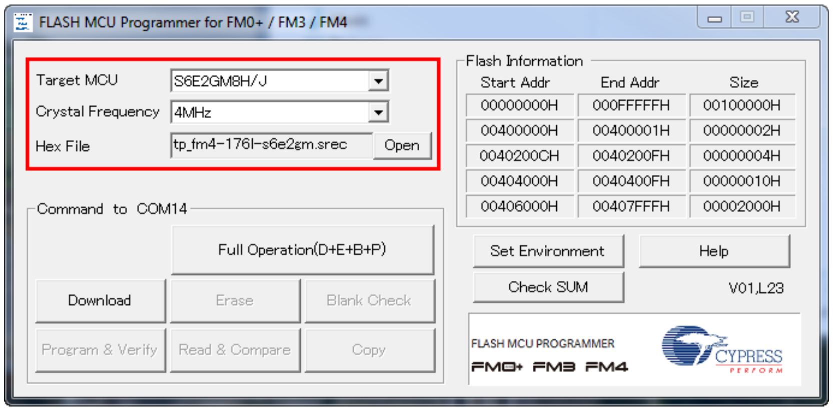
   C. Set **Hex File** to the file you wish to flash to the board.

   For purposes of this example we use *tp_fm4-176l-s6e2gm.srec*. This file restores the starter kit board to its initial state.

   In a default installation of the kit, the S-Record file is here:

   Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\Test_Demo_Code

Figure 4-2. Configure the Programmer



FLASH MCU Programmer for FM0+ / FM3 / FM4

Target MCU: S6E2GM8H/J
Crystal Frequency: 4MHz
Hex File: tp_fm4-176l-s6e2gm.srec

Flash Information

Start Addr	End Addr	Size
00000000H	000FFFFFH	00100000H
00400000H	00400001H	00000002H
0040200CH	0040200FH	00000004H
00404000H	0040400FH	00000010H
00406000H	00407FFFH	00002000H

Command to COM14

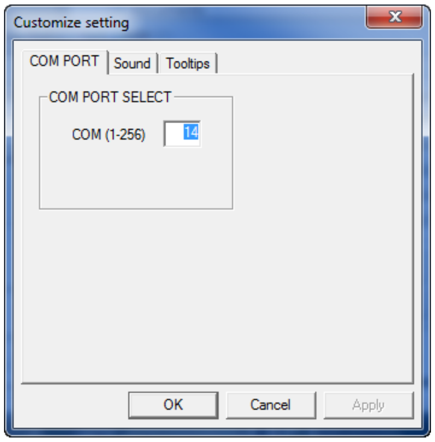
V01,L23

6. **Set the COM Port in the programmer.**

   A. Click Set Environment.

   B. Set **COM (1-256)** to the value you saw in the Device Manager. See Figure 4-3.

Figure 4-3. Set the COM Port in the Programmer



Customize setting

COM PORT SELECT

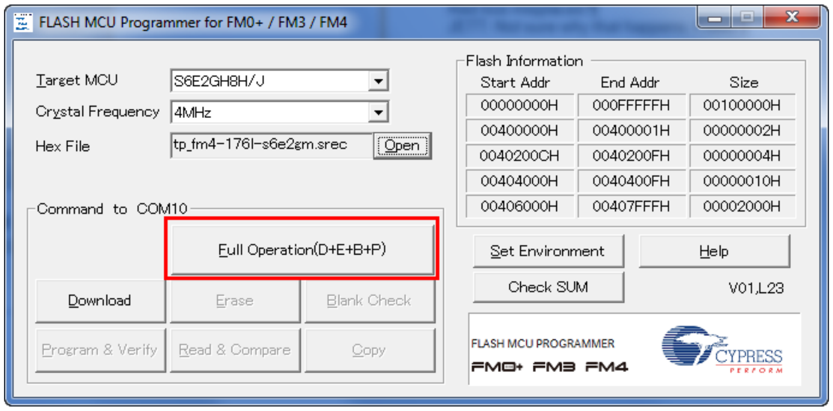
COM (1-256): 14

7. **Program the FLASH.**

You will need to reset the board as it is programmed. See Figure 4-4 and Figure 4-5.

A. Click **Full Operation (D+E+B+P)** button to start programming. (**Note**: Full Operation does not work on secured flash. You must use single steps.)

Figure 4-4. Click the Full Operation Button



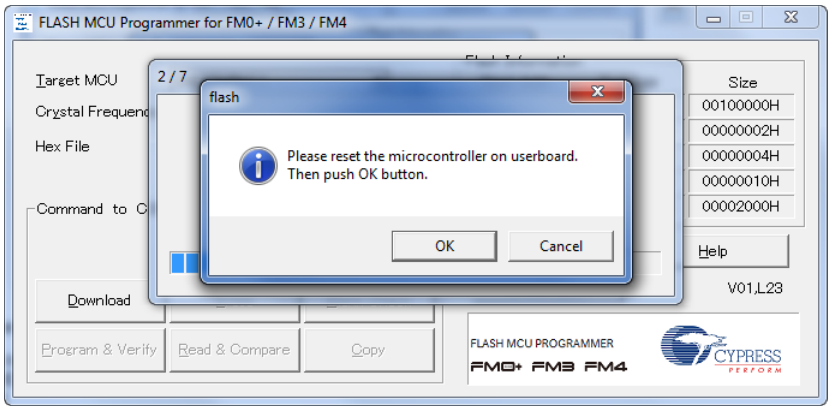
Start Addr	End Addr	Size
00000000H	000FFFFFH	00100000H
00400000H	00400001H	00000002H
0040200CH	0040200FH	00000004H
00404000H	0040400FH	00000010H
00406000H	00407FFFH	00002000H

The programming process begins, and a dialog window appears, as shown in Figure 4-5.

B. Reset the board.

Press the reset switch (SW1) on the board, and then click OK.

Figure 4-5. Reset the Microcontroller



Please reset the microcontroller on userboard.
Then push OK button.

The programmer downloads the selected file to the board.

To confirm success, use the Serial Port Viewer tool to connect to the board and run the demo code. Full instructions are in the FM4 S6E2G-Series Pioneer Kit Guide. Click **Help** for any issues or errors encountered during programming.

4.3 Using the FLASH USB Direct Programmer

These instructions assume you have downloaded and installed the starter kit so you have access to the required S-Record file. If not, locate an S-Record or Intel-HEX file, and use that file instead. The USB connection requires USB support on the target.

1. **Configure the jumpers.**

   Make sure the jumpers on the FM4 S6E2G-Series Pioneer board are placed according to Table 4-2.

Table 4-2: Jumper Settings for S6E2GM programming by FLASH USB Direct Programmer

Jumper	Default	Program by USB	Purpose
J1	Open	Open	Sets MB9AF312K (CMSIS-DAP) to run mode.
J2	Open	Closed	Sets S6E2GM to programming mode.
J3	Pin 1 to Pin 2	Pin 2 to Pin 3	Sets for USB programming mode.
J4	Pin 1 to Pin 2	Pin 2 to Pin 3	Get power from the USB port

2. **Provide power to the board.**

   Connect the USB cable to the CN3 port. The Power LED (LED3) should be lit (green). See Figure 3-1 for the location of the correct port.

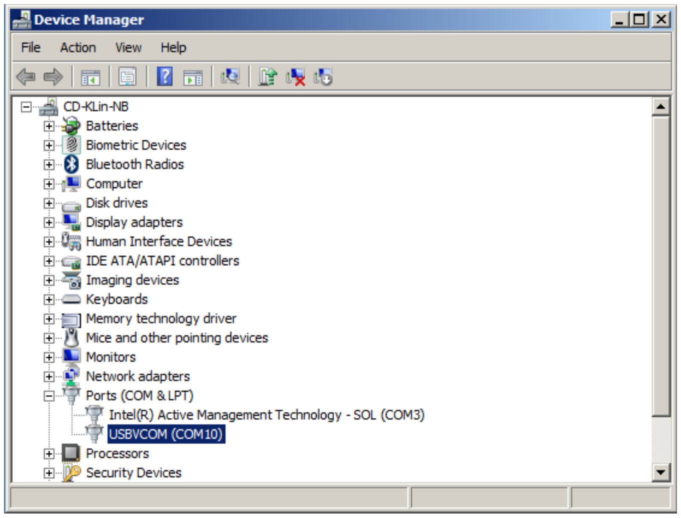
3. **Identify the COM Port in use.**

   You need to know which COM port your board is connected to. You will use this to configure the Flash programmer.

   You may see a popup notification that contains this information as you connect the board. If you don't see it, or don't know the COM port, open the Device Manager and look for **Ports (COM & LPT).).** You should see an entry for USBVCOM. The COM port is listed at the end of that entry, as shown in Figure 4-6.

   Remember the number.

Figure 4-6. Identify the COM Port In Use



4. **Launch the FLASH USB DIRECT Programmer.**

   You can do this from the Start menu, using this path:

   Start Menu > All Programs > Cypress > FLASH USB DIRECT Programmer > USBDirect

5. **Configure the programmer.**

   In this step you set up the programmer for the target device. See Figure 4-7.

   A. Set **Target MCU** to S6E2GM8H/J.

   B. Set **Hex File** to the file you wish to flash to the board.

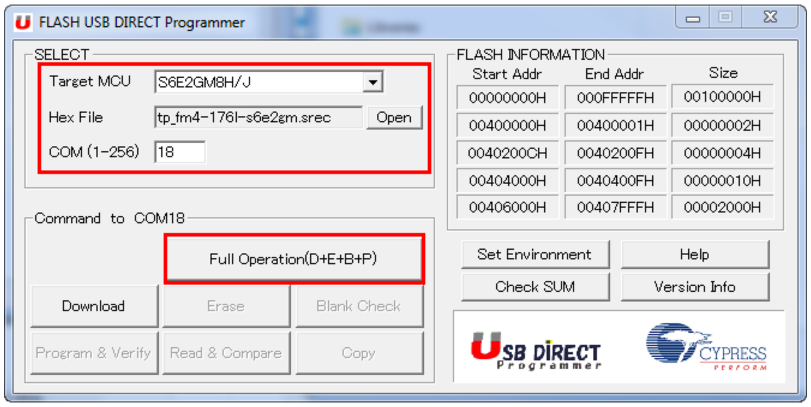
   For purposes of this step we use *tp_fm4-176l-s6e2gm.srec*. This file restores the kit to its initial state.

   In a default installation of the kit, the S-Record file is here:

   Documents\Cypress\FM4 S6E2G-Series Pioneer Kit\Firmware\Demo Projects\Test_Demo_Code

   C. Set **COM (1-256)** to the value you saw in the Device Manager.

Figure 4-7. Configure the Programmer



FLASH USB DIRECT Programmer

SELECT

Target MCU: S6E2GM8H/J
Hex File: tp_fm4-176l-s6e2gm.srec
COM (1-256): 18

Command to COM18

FLASH INFORMATION

Start Addr	End Addr	Size
00000000H	000FFFFFH	00100000H
00400000H	00400001H	00000002H
0040200CH	0040200FH	00000004H
00404000H	0040400FH	00000010H
00406000H	00407FFFH	00002000H

6. **Program the Flash.**

   You will need to reset the board as it is programmed.

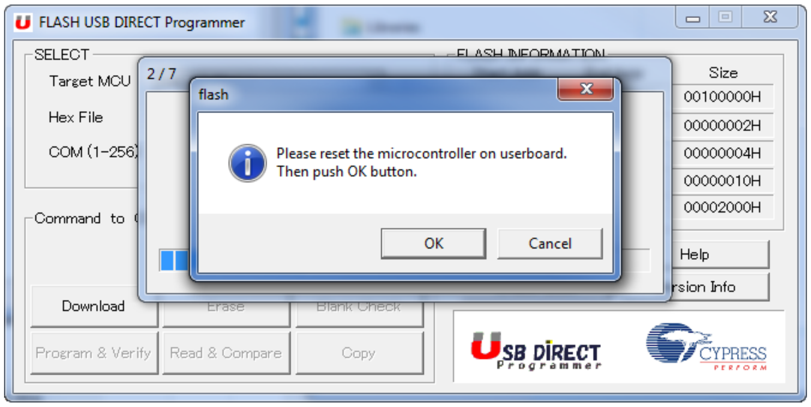
   A. Click **Full Operation (D+E+B+P)** button to start programming. (**Note**: Full Operation does not work on secured flash. You must use single steps.)

   The programming process begins, and a dialog window appears.

   B. Reset the board.

   Press the reset switch (SW1) on the board, and then click OK, as shown in Figure 4-8. You may need to do this more than once.

Figure 4-8. Reset the Microcontroller



flash

Please reset the microcontroller on userboard.
Then push OK button.

To confirm success, use the Serial Port Viewer tool to connect to the board and run the demo code. Full instructions are in the FM4 S6E2G-Series Pioneer Kit Guide.

5 FM4 Family Resources

Cypress provides many resources to help you learn about and become productive with the FM4 family. Use Table 5-1 to identify and choose the resource you want based on where you are in the design process.

Table 5-1. FM4 Family Resources Navigator

I Want To	Resources
Evaluate FM4	Read this document. Watch the FM4 introductory video. Explore the FM4 product pages on the Cypress website. Purchase an FM4 Starter Kit Refer to FM4 Datasheets. Read AN202487- Differences Among FM0+, FM3, and FM4 Families
Select an FM Part	Download and review the Product Selector Guide. Read AN202487 - Differences Among FM0+, FM3 and FM4 Families
Learn About Available Tools	Visit the FM4 Support Tools page. Visit the FM Ecosystem page. Follow the links to learn about partners who provide IDEs, compilers, debuggers, operating systems, middleware, boards, training, and more.
Learn About the Peripheral Driver Library & Example Code	Purchase the FM4 S6E2GM Pioneer Kit that installs and uses the PDL. Read the Peripheral Driver Library Overview in this document. Download the PDL. See Getting and Installing the PDL in this document. Review the PDL Quick Start Guide, installed as part of the PDL. Work with the 100+ examples included with the PDL.
Develop Software Without Using the PDL	Refer to FM4 Datasheets. Use the FM4 Peripheral Manuals as a technical reference. Get a template project for FM4 development. Visit FM4 Starter Kit web pages to get code examples. FM4-S6E2DH Graphics MCU Starter Kit with WQVGA TFT Display FM4 S6E2CC MCU Development Kit with Ethernet, CAN and USB Host FM4 S6E2CC-ETH MCU Starter Kit with Ethernet and USB Host FM4 9B560 MCU Starter Kit with USB and CMSIS-DAP Search for an FM4-related application note. Some examples include: AN213666 - PMSM Servo Motor Speed Control AN205411 - FM4 IEC60730 Class_B Self-Test Library AN203980 - S6E2Cx Series Over the Air Update AN99218 - Multi Function Serial Interface of FM MCU AN204468 – FM4 I2S USB MP3 Player Application
Learn About Flash Programming	Get a Flash programmer. FLASH MCU Programmer for FM0+/FM3/FM4 FLASH USB Direct Programmer Read the Programming Embedded Flash section of this document. Read the Flash Programming Manual for your FM4 series: MB9Bx S6E2Cx S6E2Dx S6E2Gx S6E2Hx Read AN204438 - How to Setup Flash Security for FM0+, FM3 and FM4 Families.

About the Author

Name:	James Trudeau
Title:	Senior Application Engineer
Background:	Jim Trudeau started at Cypress in 2015, continuing a long career in technical support, technical communication, and developer relations in the software tools and semiconductor industry.

Document History

Document Title: AN211122 - Getting Started with FM4 Development

Document Number: 002-11122

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5127784	JETT	2016-02-23	New application note

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Lighting & Power Control	cypress.com/powerpsoc
Memory	cypress.com/memory
PSoC	cypress.com/psoc
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless/RF	cypress.com/wireless

PSoC® Solutions

cypress.com/psoc

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/support

Cypress Semiconductor	Phone	: 408-943-2600
198 Champion Court	Fax	: 408-943-4730
San Jose, CA 95134-1709	Website	: www.cypress.com