AN210985

Getting Started with FM0+ Firmware Development

Author: James Trudeau

Associated Parts: All FM0+ parts

Related Application Notes & Code Examples: See the FM0+ Portfolio Resources section

AN210985 introduces you to the FM0+ portfolio of 32-bit general-purpose microcontrollers based on the Arm® Cortex®-M0+ processor core, ideal for ultra-low-power designs. This note provides an overview of hardware features and capabilities, firmware development, and technical resources available to you.

Contents

1 FM0+ MCU Overview

Cypress' FM0+ is a portfolio of 32-bit, general-purpose and energy-efficient microcontrollers based on the CM0+ processor. FM0+ microcontrollers operate at 40 MHz and support a diverse set of on-chip peripherals ideal for white goods, sensors, meters, HMI systems, power tools, and network-aware (Internet of Things) battery-powered or wearable devices.

There are two series within the FM0+ Portfolio. Each series represents multiple device packages with different capabilities. Table 1 lists the maximum value for some of the defining characteristics of each series.

Table 1. FM0+ Overview

Series	S16E1C	S6E1A
Flash/SRAM(KB)	128/16	88/6
GPIO	54	37
Base Timer	8	4
Multi-function Timer	-	1
Quadrature Counter	-	1
Multi-Function Serial	6	3
USB	1	-
HDMI-CEC	2	-
DMA/DSTC	-/64	2/-
ADC Inputs	8	8
CRC	Y	-
I2S	2	-
Smart Card Interface	1	-

Key features of the FM0+ MCU include:

- Performance and Energy Savings

  The MCUs are based on the CM0+ core, the most energy-efficient Arm processor available today. The optimized processing and flash architecture of the FM0+ MCU make it the industry's most energy-efficient CM0+ MCU, achieving an industry-leading 35 µA/CoreMark® score.

- Ultra-Low Power

  The ultra-low-power devices have an operating voltage range of 1.65 - 3.6 V; a maximum CPU clock frequency of 40 MHz; active mode current of 40 µA/MHz; and an RTC standby mode current of 0.6 µA.

- High-Performance Flash Memory

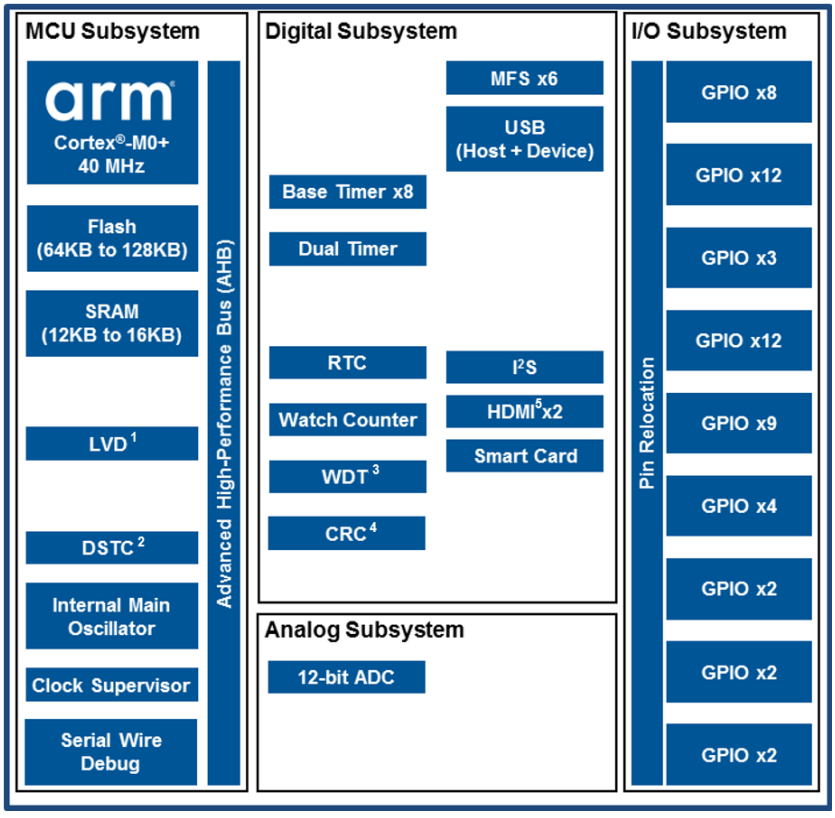
  Memory densities range from 56 KB to 512 KB flash and up to 64 KB RAM – densities typically found only in MCUs with larger Cortex-M3/M4 cores. The flash memory features a true zero-wait-state operation at full CPU speed and data retention of up to 20 years.

- Other Major Features

  The devices also feature analog peripherals. A simplified bus matrix reduces power consumption. The devices also include local clock gating for each peripheral, with a separated clock divider for the CPU and peripherals to fine-tune power consumption. The FM0+ MCUs feature Full-Speed USB2.0 Host and Device capabilities, and offer multiple serial communication interfaces and AES encryption.

  Figure 1 shows the block diagram of the S6E1C-series as an example. The FM0+ MCU provides a range of peripherals such as A/D converters, USB, a configurable multi-function serial interface, and real-time clock. Peripheral support varies among the devices. For details on each series, such as pin counts, package options, voltage operating range, available peripherals, and Flash/SRAM options, review the Product Selector Guide and data sheets. Read AN203277 to learn more about hardware design considerations.

Figure 1. Block Diagram for the FM0+ S6E1C Series



[1] Low-voltage detect
[2] Descriptor system transfer controller
[3] Watchdog timer
[4] Cyclic redundancy check
[5] HDMI consumer electronics control signal

2 Firmware Development

This section discusses the Cypress FM Peripheral Driver Library (PDL) v2.x. The PDL is central to firmware development for all FM portfolios. The PDL simplifies software development for the extensive set of peripherals available. It reduces the need to understand registers and bit structures. You configure the library for the desired functionality, and then use API function calls to initialize and use a peripheral. In addition to the FM0+ processors, the PDL supports Cypress FM4 and FM3 processors and peripherals. Using the PDL makes it easier to port code from one portfolio to the other.

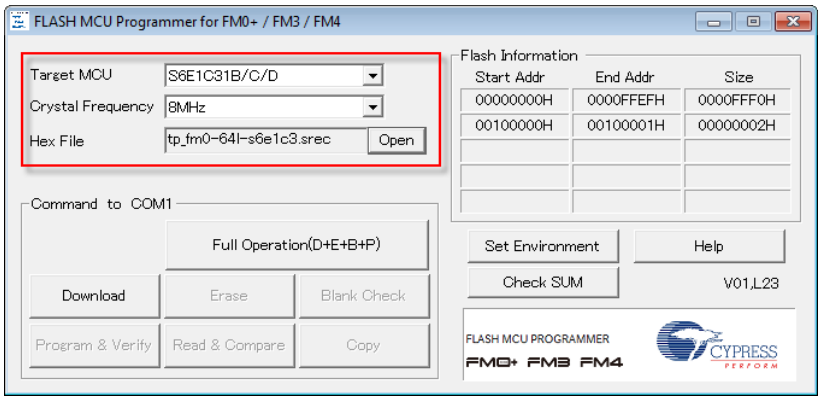
Developers who wish to work at the register level should also install the PDL. The PDL is where you get device-specific header files, startup code, configuration files, and IDE project files for every FM0+ device. You can use these files with or without the PDL.

The PDL is provided as source code. Reviewing the PDL source code is a useful way to approach the detailed knowledge required to program a microcontroller at a low level. Combined with the review of the appropriate data sheet and peripheral manual, you can learn the information you need to use a peripheral. See the Using the FLASH MCU Programmer

As noted in Section 3.1 Before You Begin, the S6E1C3 kit does not support this programmer. Therefore, this section does not explain how to use the FLASH MCU Programmer with the kit. However, the programmer supports FM0+ devices, so it is a useful alternative for custom designs. All FM0+ devices can be programmed by serial connection.

You configure the programmer like you do for the FLASH USB DIRECT programmer. You set the target MCU, Crystal Frequency, and specify a hex file, as shown in Figure 7.

Figure 7. Configure the Programmer



Click **Set Environment** to specify the correct COM port. Read the FLASH MCU Programmer documentation for details.

FM0+ Portfolio Resources section of this document for links to the extensive technical documentation available.

Because the PDL is central to all FM devices, you will find in-depth information on firmware development in the PDL Quick Start Guide. This includes simple step-by-step instructions on how to build and run a PDL code example. The PDL Quick Start Guide is installed along with the PDL. It is also available separately at the Cypress PDL Software Archive.

2.1 Peripheral Driver Library v2.x 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 to ensure that the targeted peripheral is present on the selected device.

The superset design means the PDL is useful across all devices irrespective of the available peripherals. This enables the 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 would 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 series or device to confirm support for the peripheral.

2.1.1 Getting and Installing PDL v2.x

Download the PDL Installer from the Cypress PDL Software Archive. Launch the installer, and follow the prompts.

2.1.2 PDL Structure

PDL v2.x is organized into several folders as shown in Table 2.

Table 2. PDL Folder Structure

Path\Folder	Description
cmsis	cmsis header files
devices	For each device package: common header files configuration, startup, and project files for each IDE
doc	PDL documentation
driver	Driver source code and header files
examples	Code examples for each peripheral on each supported starter kit
utilities	Various utility files

When you use the PDL, typically the only files you modify are *pdl_user.h* and *main.c*.

2.2 Software Development Overview

Table 3. Supported Toolchains

Vendor	Tool	Version
IAR Systems	Embedded Workbench	7.50.1 or higher
ARM Keil	µVision	5.17 or higher
Open Source/ARM	GCC ARM Embedded	4.9-d015-Q1-Update or higher
iSYSTEM	winIDEA	9.12 or higher

2.2.1 Using PDL Code Examples

PDL v2.1 includes code examples configured for particular starter kits. You find the code examples in the examples folder, organized by starter kit. Each example demonstrates the basic initialization and configuration for a peripheral. Some peripherals have multiple examples.

Note: For step-by-step instructions on how to build and run a PDL project, see the PDL Quick Start Guide installed along with the PDL. It is also available separately at the Cypress PDL Software Archive.

Some Cypress FM0+ starter kits install a version of the PDL as part of the kit. The kits may install an older version of the PDL. Starter kit code examples work only with the version of the PDL used by the kit. They will not work with a different version of the PDL.

2.2.2 Writing Your Own Code Using the PDL

For detailed information on topics such as creating a custom project, configuring the PDL, configuring a peripheral, and using a peripheral, see the PDL Quick Start Guide, available in the doc folder in the PDL directory. It is also available separately at the Cypress PDL Software Archive.

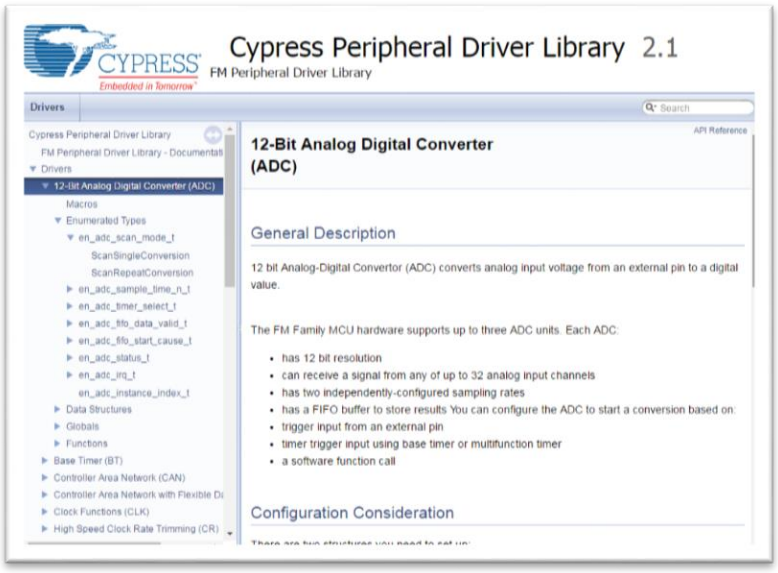
2.2.3 PDL API Documentation

PDL API documentation is HTML-based and generated from the source code. The PDL installer puts the documentation here: *<PDL directory>\doc\pdl_api_reference_manual.html.*

The first time you open the documentation, make a bookmark in your browser for easy access.

In the documentation, use the left navigation menu to find the information you need. The **Drivers** section lists all the information for a particular peripheral. Expand any driver to see the macros, types, structures, global variables, and API functions. Figure 2 shows the documentation home page.

Figure 2. PDL Documentation

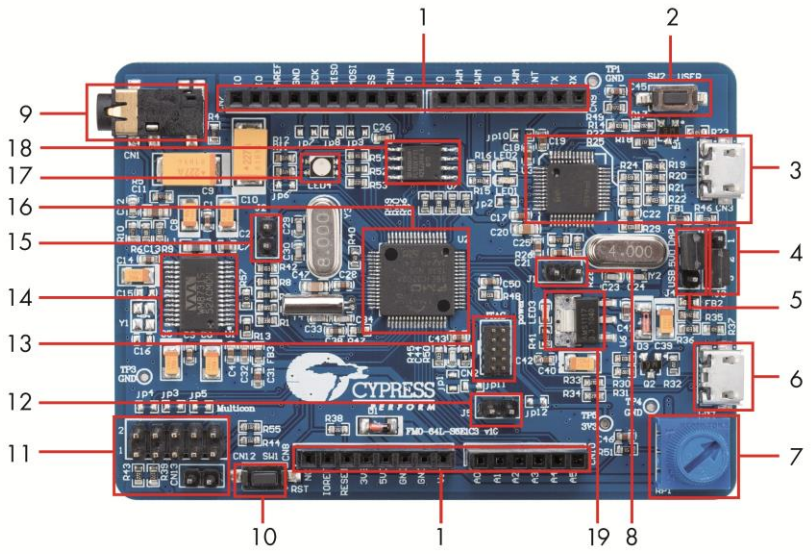


See the FM0+ Portfolio Resources section of this document for additional links to helpful information.

3 Programming Embedded Flash

Full information on firmware development for FM microcontrollers is in the *PDL Quick Start Guide*. 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 for the S6E1C device. These instructions target the board found in the FM0+ S6E1C3 MCU Starter Kit. Figure 3 has a key to the components on the hardware.

Figure 3. The S6E1C3 MCU Starter Kit Board



1. Arduino interface (CN7-CN10)
2. User button
3. Programmer and debugger (CMSIS-DAP)
4. Series programming mode select (J3)
5. Power supply resource select (J4)
6. USB device connector (CN4)
7. Potentiometer
8. Programming mode jumper of MB9AF312K (J1)
9. Headphone and microphone jack (CN1)
10. Reset button
11. Multicon connector (CN12, CN13)
12. Jumper for current measure (J5)
13. 10-pin JTAG connector
14. Stereo codec
15. Programming mode jumper of FM0+ (J2)
16. Cypress FM0+ MCU S6E1C32D0A
17. RGB LED
18. Cypress 4-Mb SRAM
19. 3.3V Voltage Regulator

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.

3.1 Before You Begin

Cypress provides two flash programmers for use with FM0+ devices:

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

The S6E1C3 kit supports the FLASH USB Direct Programmer.

This kit does not support the FLASH MCU Programmer. The kit's CMSIS-DAP interface uses the SIN/SOTO_0 pin required by the programmer. It is expected that most user designs will not keep the CMSIS-DAP interface (U3) on their custom boards. In that case, the FLASH MCU Programmer is the choice for designs that do not have a USB connection.

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.

You also need a file to download. The file format must be either Motorola S-Record or Intel-HEX. The instructions use 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.

3.2 Using the FLASH USB DIRECT Programmer with the S6E1C3 kit

These instructions assume you have downloaded and installed the starter kit files, 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 board are placed according to Table 4.

   Table 4: Jumper Settings for S6E1C3 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 2 to Pin 3	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 connector (CN4)

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

   Connect the USB cable to the CN4 connector. The Power LED (LED3) should be lit (green). See Figure 3 for the location of the correct connector.

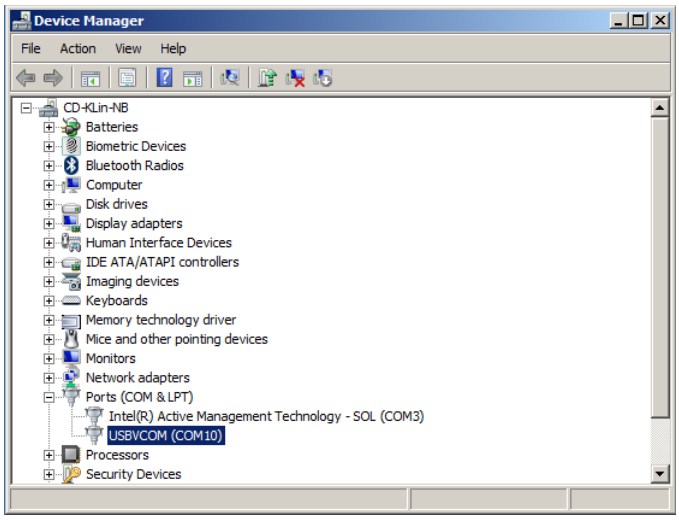
3. **Identify the COM Port in use.**

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

   If you use the Cypress Serial Port Viewer and Terminal tool (installed with the kit), you will see a popup notification that contains this information as you connect the board. If you 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.

   Remember the number.

Figure 4. Identify the COM Port in Use



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

   In a default installation, the programmer is here:

   C:\Program Files (x86)\Cypress\FLASH USB Direct Programmer

5. **Configure the programmer.**

   In this step, you set up the programmer for the target device. See Figure 5.

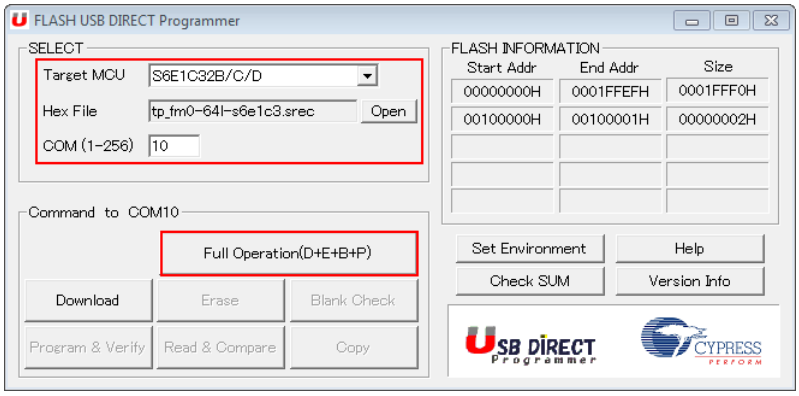
   A. Set **Target MCU** to S6E1C32B/C/D.

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

      This example uses *tp_fm0-64l-s6e1c3.srec*. This file restores the starter kit board to its initial state. The S-Record file is here: < *Kit Directory*>*\Firmware\Demo Projects\Test_Demo_Code*.

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

Figure 5. Configure the Programmer



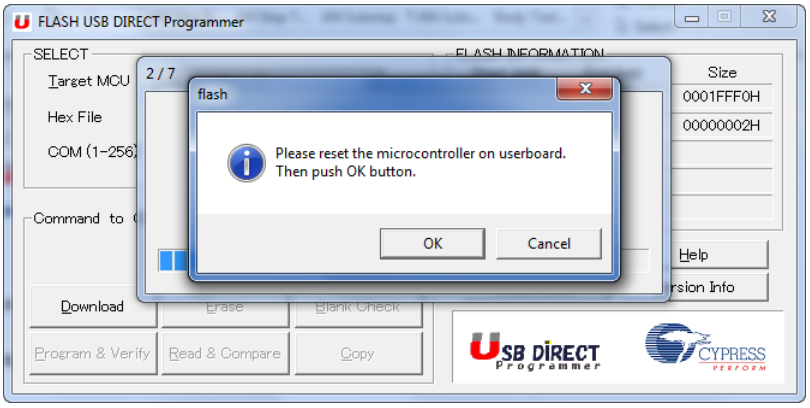
6. **Program the Flash.**

   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.

   B. Reset the board. Press the reset switch (SW1) on the board, and then click **OK**, as shown in Figure 6.

   You may need to do this more than once during the process.

Figure 6. Reset the Microcontroller



7. **Restore the board to normal operation.**

   When done, restore the jumpers to their original configuration, or to default values as shown in Table 4.

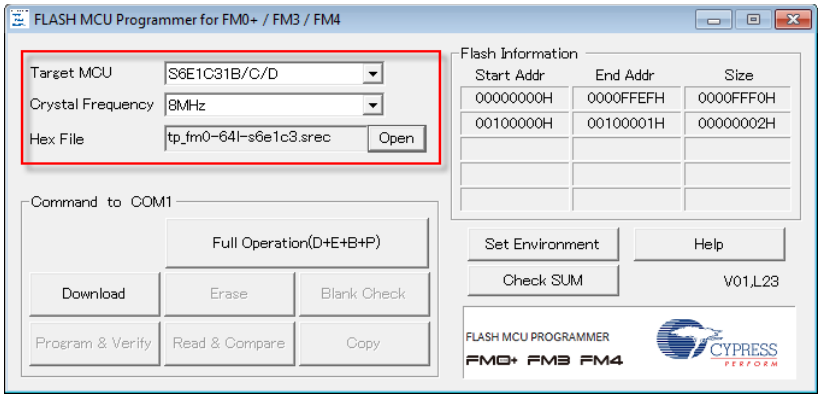
   To confirm success, use the Serial Port Viewer and Terminal tool (installed with the kit) to connect to the board and run the demo code. Full instructions are in the starter kit guide.

3.3 Using the FLASH MCU Programmer

As noted in Section 3.1 Before You Begin, the S6E1C3 kit does not support this programmer. Therefore, this section does not explain how to use the FLASH MCU Programmer with the kit. However, the programmer supports FM0+ devices, so it is a useful alternative for custom designs. All FM0+ devices can be programmed by serial connection.

You configure the programmer like you do for the FLASH USB DIRECT programmer. You set the target MCU, Crystal Frequency, and specify a hex file, as shown in Figure 7.

Figure 7. Configure the Programmer



Click **Set Environment** to specify the correct COM port. Read the FLASH MCU Programmer documentation for details.

4 FM0+ Portfolio Resources

Cypress provides many resources to help you learn about and become productive with the FM0+ portfolio. Use Table 5 to identify and choose the resource you want based on where you are in the design process.

Table 5. FM0+ Portfolio Resources Navigator

I Want To	Resources
Evaluate FM0+	Read this document. Explore the FM0+ product pages on the Cypress website. Purchase an FM0+ Starter Kit. FM0+ S6E1A1 MCU Evaluation Board FM0+ S6E1C3-Series Starter Kit Refer to FM0+ 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 Hardware Design	Read AN203277 – FM 32-bit Microcontroller Family Hardware Design Considerations
Learn About Available Software Development Tools	IAR Embedded Workbench Keil µVision IDE iSYSTEM winIDEA GCC ARM Embedded
Learn About the Peripheral Driver Library	Purchase the FM0+ S6E1C-Series Starter Kit. Download the PDL and read the *PDL Quick Start Guide*. Read the *Build and Run a PDL Project* section of the *PDL Quick Start Guide*. Explore the PDL code examples installed with the PDL
Learn about particular FM0+ peripherals	Search for an FM0+-related application note. Some examples include: AN202483 - FM0+ S6E1A1 Series MCU Low-Voltage 3-Phase BLDC and PMSM Control AN204389 - FM0+ Family 3-Phase ACIM Scalar Control AN99231 - Using Interrupts in FM0+ Portfolio S6E1C3 Series
Develop Low-level Software for FM0+	Read the *Creating a Custom Project* section of the *PDL Quick Start Guide*. Use project files and startup code from the PDL *devices* folder. Use PDL source code to see low-level programming techniques Refer to FM0+ Datasheets. Use the FM0+ Peripheral Manuals as a technical reference.
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 FM0+ series: S6E1Ax S6E1Cx Read AN204438 - How to Setup Flash Security for FM0+, FM3 and FM4 Families

About the Author

Name:	James Trudeau
Title:	Senior Principal Application Engineer
Background:	Jim Trudeau is a senior principal application engineer at Cypress.

Document History

Document Title: AN210985 - Getting Started with FM0+ Firmware Development

Document Number: 002-10985

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5132964	JETT	02/10/2016	New Application Note.
*A	5347639	JETT	07/25/2016	Update to new AN template. Update table 1.1, portfolio features (from Cypress website) Updated for information related to PDL v2.1 in sections 1 and 2 Deleted former section 3 Build and Run a PDL Project (information is in the PDL Quick Start Guide) Update links throughout document to current resources.
*B	5715380	AESATMP9	04/27/2017	Updated logo and copyright.
*C	6082127	JETT	02/27/2018	Update to use S6E1C as example kit and MCU Update links and references to PDL to make clear FM0+ requires v2.x. Update list of supported IDEs, and link to iSystem WinIDEA Update to new app note template and copyright
*D	6105718	JETT	03/21/2018	Updated Table 1 Added footnotes to Figure 1 Added information on MCU Flash Programmer

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
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709