AN205550

F²MC-16FX Family, ROM Mirror

This application note describes the functionality of the ROM Mirror function.

Contents

1 Introduction

This application note describes the functionality of the ROM Mirror function and gives an example.

The ROM Mirror makes the contents of a selectable area of the Flash Memory/ROM available is bank `0x00`. This can be used to make the constants stored in non-volatile memory accessible by 16-bit addressing when using “Small” or “Medium” memory model.

1.1 Key Features

- ROM Mirror 0K, 8K, 16K, 24K or 32K Bytes of any bank from `0xFF` to `0xF0`.
- RAM/ROM Mirror 32K, 24K, 16K, 8K or 0K Bytes of bank `0x01` (External Memory)

2 ROM Mirror

The Basic Functionality of the Rom Mirror Function

2.1 Functionality

When using the “medium” or “small” compiler memory model, data access is done by 16-Bit addressing which saves code size and execution time. For this reason the lowest program bank `0x00` is used, which represents 64 Kbytes of addressing space and holds the basic I/O register space and the main RAM.

Constants are located in the ROM (Flash) addressing area. Therefore a 24-Bit addressing would be needed for access. The ROM Mirror function allows to “mirror” a ROM area to the lower `0x00` program bank, so that a 16-Bit access also to constants is possible.

There may be some devices which has on-chip ROM of more than 28 KByte. In such cases, the RAM is located in the Bank `0x01` from address `0x018000` to `0x01FFFF`. This internal RAM from bank `0x01` can be mirrored to the lower `0x00` program bank. The size of this area can be set by the user.

The address space between `0x008000` and `0x00FFFF` is used for ROM/RAM mirroring. After Reset the address space between `0xFF8000` and `0xFFFFFF` is mirrored to program bank `0x00` (default).

It should be noted that the wait state limitation while reading Flash Memory (corresponding to the clock speed) is equally applicable when accessing the constants using ROM Mirror.

2.2 ROM Mirroring Register (ROMM)

The ROMM consists of the following bits:

Table 1. ROMM

Bit No.	Bit Name	Initial Value	Value	Description
7 … 4	BS[3:0]	1, 1, 1, 1	0, 0, 0, 0	0xF0 bank is mirrored
			0, 0, 0, 1	0xF1 bank is mirrored
			0, 0, 1, 0	0xF2 bank is mirrored
			. . .	...
			1, 1, 1, 1	0xFF bank is mirrored
3	-	X	X	*Undefined. Please always write "0" to this Bit.*
2 … 0	SZ[1:0], MI	1, 1, 1	X, X, 0	ROM Mirror disabled. 0x018000 – 0x01FFFF is mirrored to 0x008000 – 0x00FFFF
			0, 0, 1	0xFXE000 – 0xFXFFFF and 0x018000 – 0x01DFFF is mirrored
			0, 1, 1	0xFXC000 – 0xFXFFFF and 0x018000 – 0x01BFFF is mirrored
			1, 0, 1	0xFXA000 – 0xFXFFFF and 0x018000 – 0x019FFF is mirrored
			1, 1, 1	0xFX8000 – 0xFXFFFF is mirrored

0xFX.... stands for the program bank selected by the BS[3:0] Bits.

2.3 Linker Settings

In order to use, the ROM mirror functionality the following steps needs to be carried out in the project workspace:

- Set the ROM Mirror Configuration (4.11) in start.asm file.
- Place the constant (CONST) section in the corresponding Flash Memory area.
- Define the memory model as "Small" or "Medium".

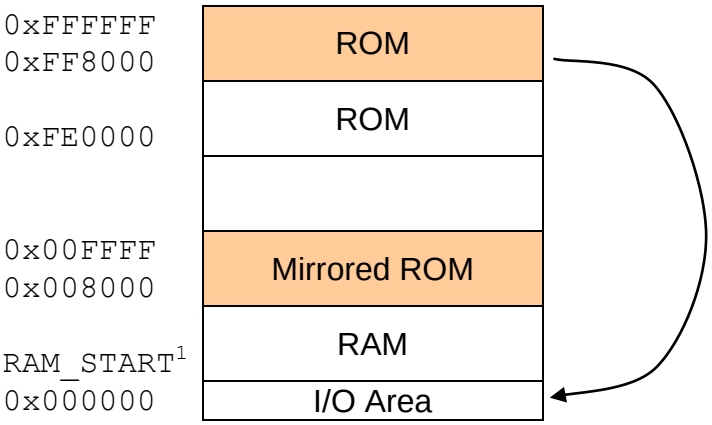
After following the above steps, if one of the constant resides at a memory address 0xFE8008, then it is accessed at address 0x008008. Section 3.1 describes this in details with example.

2.4 Memory Allocation Examples

2.4.1 Default

```
MI = 1; SZ[1:0] = 1, 1; BS[3:0] = 1, 1, 1, 1
```

Figure 1. Memory Allocation - Default

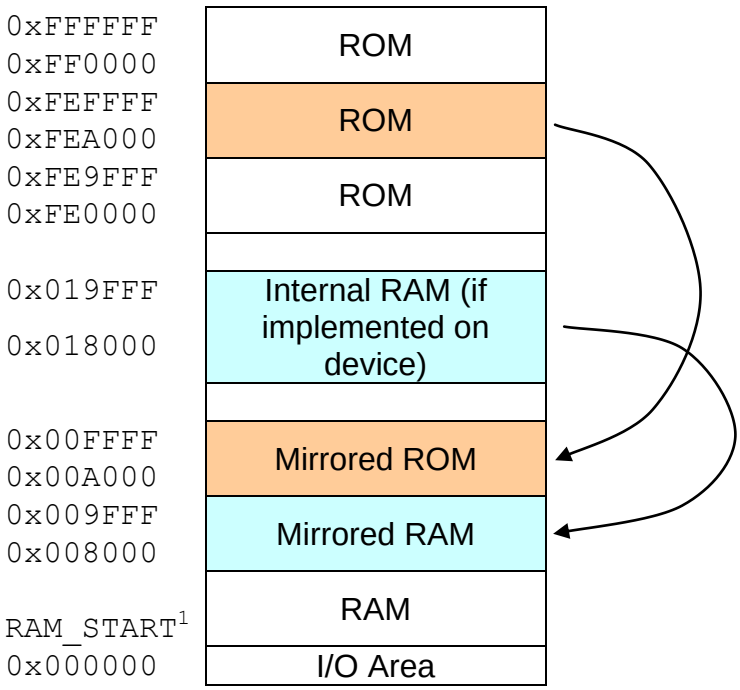


[1] Check datasheet for actual address

2.4.2 Split RAM/Flash Memory Mirror

`MI = 1; SZ[1:0] = 1, 0; BS[3:0] = 1, 1, 1, 0`

Figure 2. Memory Allocation – Split RAM/Flash Memory Mirror



3 ROM Mirror Example

Example for the Rom Mirror Usage

3.1 Enabling ROM Mirror using start.asm

The below sample code demonstrates how to enable and use the ROM Mirror functionality.

In this example, it is considered that the constant `array [5]` needs to reside in the bank `0xFE` starting from address `0xFE8000` and the entire area from address `0xFE8000` to address `0xFEFFFF` needs to be mirrored to bank `0x00`.

In order to achieve the same, the following configuration needs to be done in *start.asm* file:

- The define `ROMMIRROR` needs to be set to `ON` to enable the ROM Mirroring functionality.
- The define `MIRROR_BANK` needs to be set to `0xE` to mirror bank `0xFE`.
- Also the define `MIRROR_SIZE` needs to be set to `MIRROR_32KB` to mirror entire 32KB area starting from `0xFE8000`.

start.asm

```
;=====================================================================
; 4.11 ROM Mirror configuration
;=====================================================================

#set      MIRROR_8KB    0
#set      MIRROR_16KB   1
#set      MIRROR_24KB   2
#set      MIRROR_32KB   3

#set     ROMMIRROR     ON      ; <<< ROM mirror function ON/OFF
#set     MIRROR_BANK   0xE     ; <<< ROM Mirror bank, allowed entries:
                               ; <<< 0x0..0xF for the banks 0xF0..0xFF
#set     MIRROR_SIZE   MIRROR_32KB ; <<< ROM Mirror size
```

The constant `array[5]` needs to be defined in *main.c*.

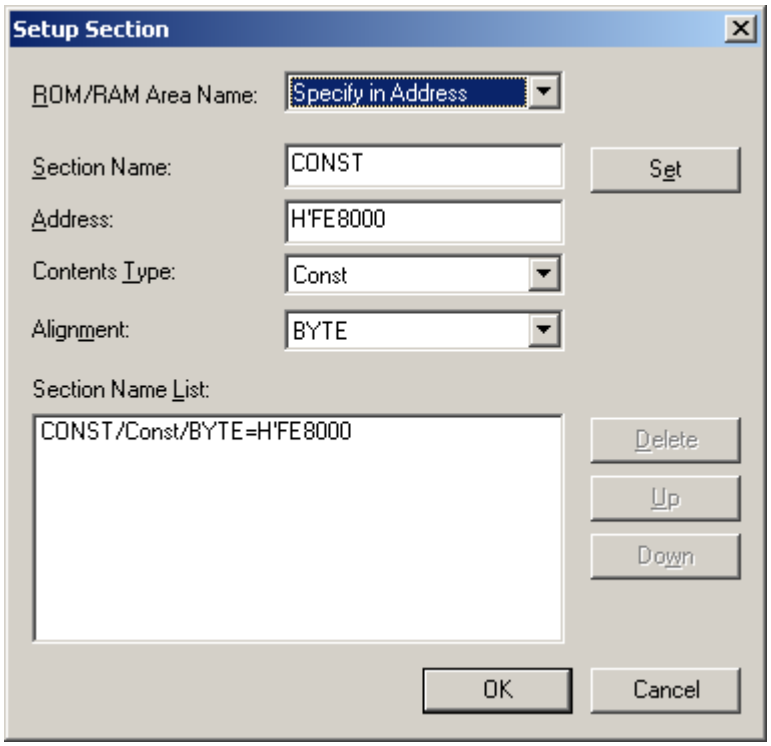
main.c

```
const int array[5] = {0x1000, 0x2000, 0x3000, 0x4000, 0x5000};
```

Then the CONST section needs to be set for the project. This can be done in the “Setup Section” dialog box as below:

In the “ROM/RAM Area Name” dropdown menu one has to select “Specify in Address”. Then in the “Section Name”, CONST needs to be specified and then the desired starting address (`0xFE8000` in this case). The “Contents Type” should be “Const” and the alignment as per the requirement. After filling all the information, “Set” button needs to be set and then press “Ok”.

Figure 3. CONST Section Setting

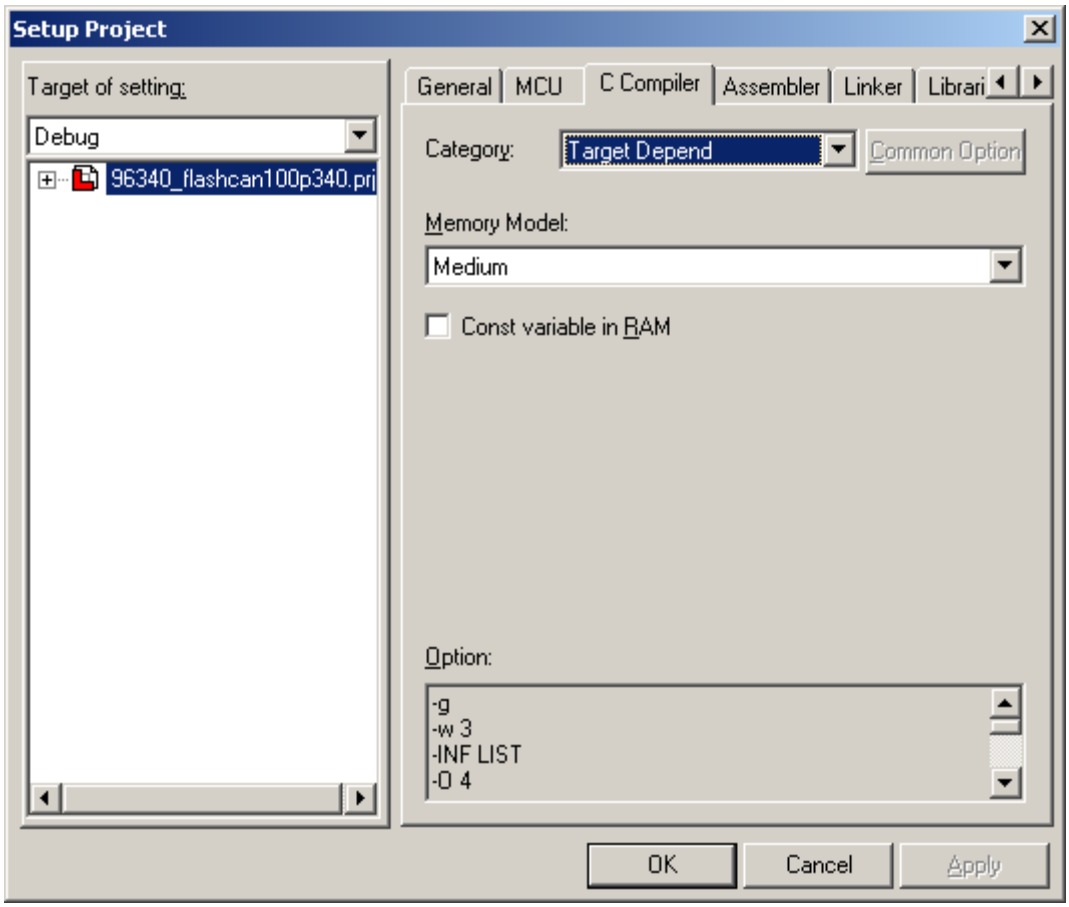


The above dialog box can be reached as mentioned below…

Project -> Setup Project -> Linker -> Disposition Connection -> Set Section

After setting up the section the “Memory Model” needs to be defined as “Medium” or “Small”.

Figure 4. Memory Model Setting



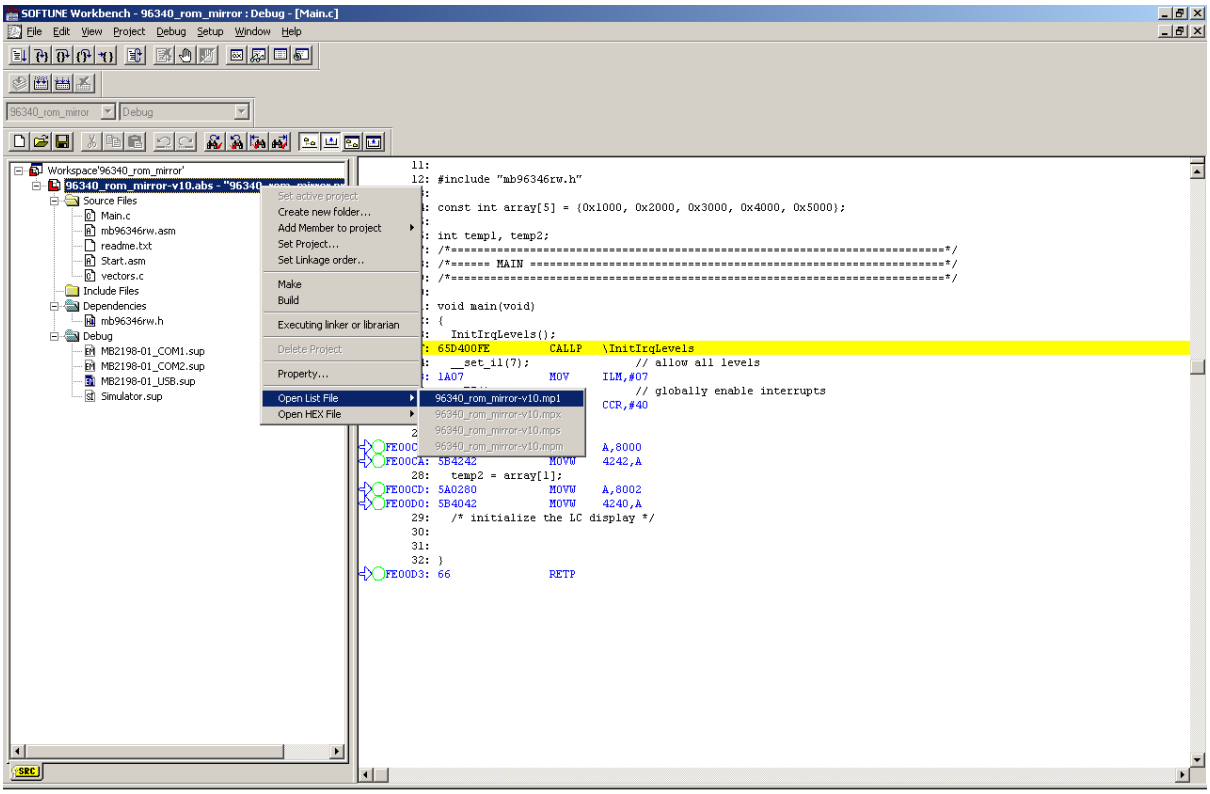
The above dialog box can be reached as mentioned below…

Project -> Setup Project -> C Compiler -> Category -> Target Depend -> Memory Model

Once all the above steps are carried out, then the project can be build.

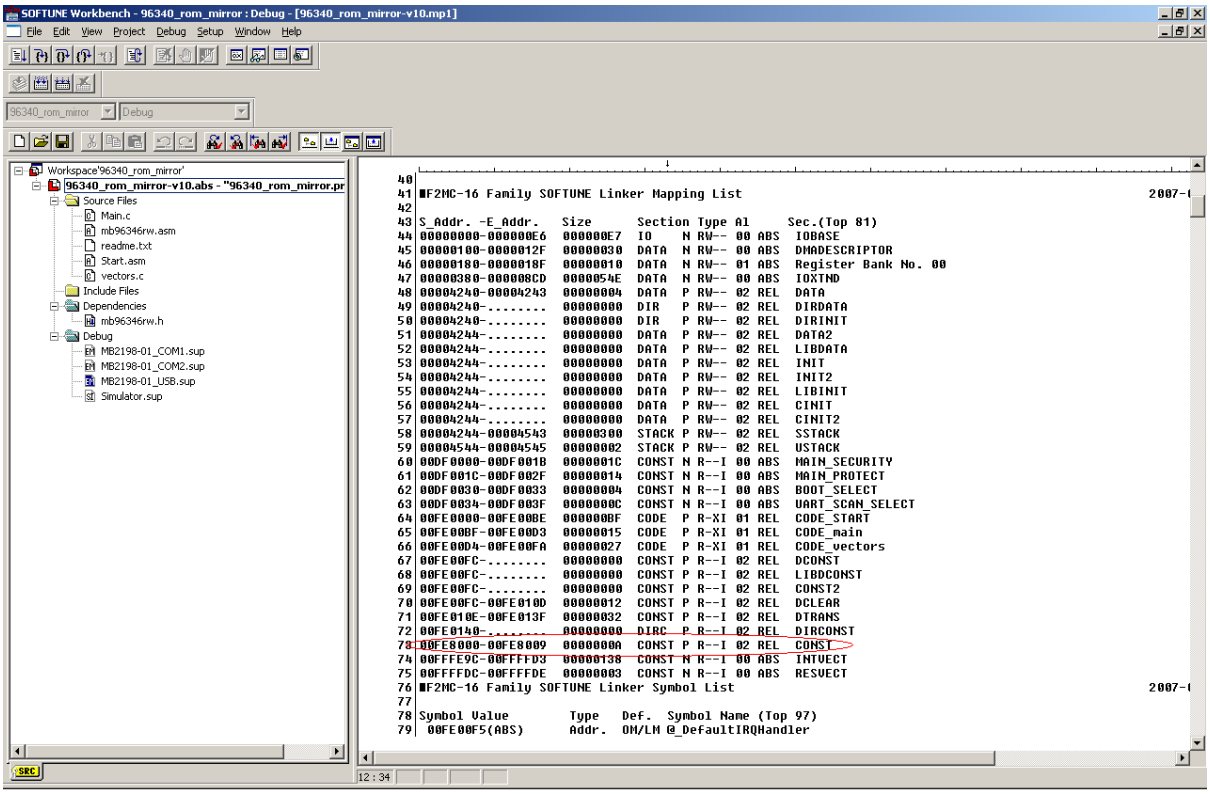
After building the project, in order to confirm that the `CONST` section is mapped correctly to the desired bank `0xFE`, one can right click on *96340_rom_mirror-v10.abs*. Then in the popup window go to *Open List File -> 96340_rom_mirror-v10.mp1*.

Figure 5. Opening Map File



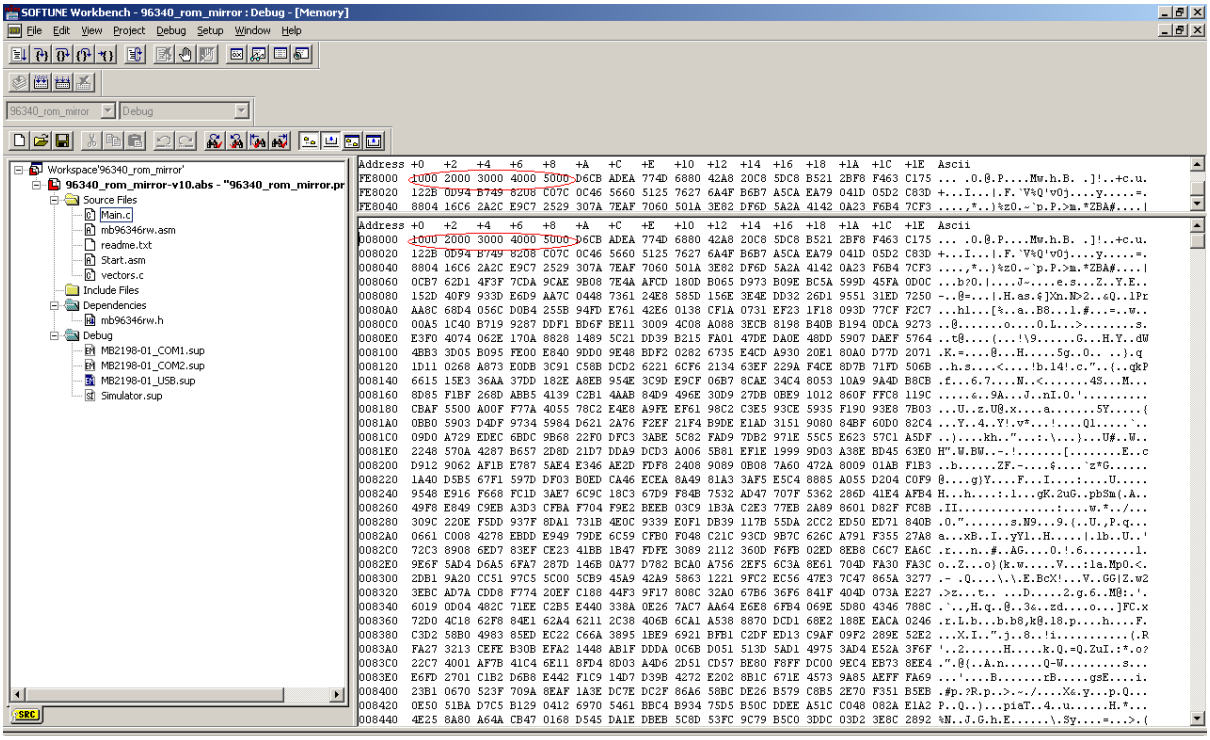
Then the map file *96340_rom_mirror-v10.mp1* would be opened as below. As highlighted in the figure, `CONST` section starts from `0xFE8000` and ends at `0xFE8009`. The section is 10 bytes long since the constant `array` is of integer type and has 5 elements.

Figure 6. Viewing Map File



In order to confirm that the ROM mirroring is got configured correctly, in the memory window bank `0xFE` and bank `0x00` can be viewed. The following figure shows that word data of the first five locations of bank `0xFE` (`0xFE8000 – 0xFE8008`) are exactly matching with those of `0x00` (`0x008000 – 0x008008`).

Figure 7. Bank 0xFE Mirror to Bank 0x00



The Memory window can be reached as mentioned below…

View -> Memory -> Desired Start Address

4 Additional Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

Document History

Document Title: AN205550 - F²MC-16FX Family, ROM Mirror

Document Number: 002-05550

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	06/26/2006	Initial Release
			07/23/2007	Error corrections; added linker settings
*A	5074679	NOFL	03/07/2016	Migrated Spansion Application Note MCU-AN-300212-E-V11 to Cypress format
*B	5870062	AESATP12	09/01/2017	Updated logo and copyright.

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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