Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.



THIS SPEC IS OBSOLETE

Spec Number: 001-62083

Spec Title: PSOC (R) 3 TO PSOC 5 MIGRATION GUIDE - AN6208

Sunset Owner: MKEA

Replaced By: 001-77835, 001-84741

PSoC® 3 to PSoC 5 Migration Guide

Author: Mark Ainsworth
Associated Project: No
Associated Part Family: CY8C3xxxx/CY8C5xxxx
Software Version: PSoC® Creator™ 2.0
Related Application Notes: None

If you have a question, or need help with this application note, contact the author at mkea@cypress.com

OBSOLETE

Abstract

AN62083 provides an overview of topics to consider when designing a project for easy migration from PSoC® 3 to PSoC 5 devices. Device considerations, PSoC Creator topics, and firmware porting are discussed.

Introduction 1
- Why Migrate? 2

Device Differences 2

Hardware Design Considerations 4
- Power System 4
- Pin Assignment and PCB Design 4
- Analog Performance 4

PSoC Creator Considerations 4
- PSoC 5 Device Selection 5
- Initial Build Errors 5
- Other PSoC Creator Error Messages 9
- Static Timing Analysis Warnings 9

Porting Firmware 10
- Conditional Compilation 10
- Editable Code Sections 10
- DMA Addresses 10
- Endian Format 10
- Timing 11
- Compiler Keywords 11
- Assembly Language Code 12

Summary 12

About the Author 12

Appendix A – Memory Maps 13

Introduction

PSoC 3 and PSoC 5 devices are designed for easy migration between the two product families. Although the CPU cores are different, the programmable analog, programmable digital, programmable routing, pin functions, and other features are quite similar. Furthermore, the PSoC Creator IDE handles a lot of the migration issues for you, automatically. Often, migrating a PSoC Creator design from one device to another is as simple as selecting a new part then rebuilding the project.

However, there are still some considerations that you should keep in mind when planning for migration. For example, additional features in PSoC 5 may make it desirable to reassign pins on the PCB. A faster and more efficient CPU may cause a review of system clock speed settings.

This application note discusses all of the topics that you should consider when migrating a project or product from PSoC 3 to PSoC 5. Both device level and PSoC Creator project level considerations are covered. If you are new to PSoC 3 or PSoC 5, see one of the PSoC 3 or PSoC 5 datasheets or AN54181, Getting Started with a PSoC 3 Design Project. For hardware design topics, see AN61290, PSoC 3 and PSoC 5 Hardware Starting Guide.

Why Migrate?

The major difference between PSoC 3 and PSoC 5 is the CPU core: PSoC 3 has an 8-bit 8051 and PSoC 5 has a 32-bit ARM Cortex-M3. Also, PSoC 5 has more flash and SRAM memory than PSoC 3 has, and adds one or two SAR ADCs. The usual reason to migrate is to take advantage of one or more of these features.

However, the PSoC 5 features do come at an additional cost, so before migrating it makes sense to determine if it really is necessary. There are elements in the PSoC 3 and PSoC 5 architecture that may reduce the need to migrate. They include:

1. All PSoC 3 and PSoC 5 devices have a powerful and highly flexible DMA controller, and many devices have a digital filter block (DFB). You may be able to offload significant functionality from the CPU to these blocks, and if the CPU has a lower processing burden then you may not need to migrate.
2. All PSoC 3 and PSoC 5 devices also have a set of highly configurable universal digital blocks (UDBs), which contain both PLDs and 8-bit ALU datapaths. It is possible to offload functionality from the CPU to standard or even custom intelligent peripheral components that are built using UDBs.
3. You may be able to optimize your PSoC 3 8051 code, and reduce its size in flash. For more information, see AN60630, PSoC 3 8051 Code Optimization.
4. All PSoC 3 and PSoC 5 devices have abundant analog routing resources. Using PSoC Creator components, you can build complex analog multiplexing systems, some of which are controlled by UDB peripherals instead of the CPU. This may allow more efficient use of the single delta-sigma ADC in PSoC 3 and reduce the need for the multiple ADCs in PSoC 5. For more information, see one of the multiplexer component datasheets, for example Hardware Multiplexer.

When planning a PSoC-based product, you should review your project design relative to the above subjects, for opportunities to remove the need to migrate and avoid possible cost increases.

Device Differences

Although PSoC 3 and PSoC 5 families have many features in common, there are some important differences.

For details see Table 1, and for more information see the specific device datasheets.

Table 1. Major Differences between PSoC 3 and PSoC 5

Consideration	PSoC 3	PSoC 5
CPU	8-bit 8051, dual DPTR, most instructions execute in 1 or 2 cycles, supported in PSoC Creator by Keil compiler.	32-bit ARM Cortex-M3, supported in PSoC Creator by gcc, MDK and RVDS compilers.
Memory	Up to 72 K flash, 8 K SRAM, 2 K EEPROM	Up to 288 K flash, 64 K SRAM, 2 K EEPROM
Additional functional blocks available in PSoC 5		Up to two SAR ADCs: 12-bit, 700 ksps
Functional blocks available only in PSoC 3	Boost	Not supported
	External Memory Interface (EMIF)	Not supported
	CAN	Not supported
	Die temperature sensor	Not supported
Non-volatile latches (NVLs) NOT available in PSoC 5		Fixed settings are: I/O ports always in high impedance analog configuration until they are configured. always using SWD - no JTAG no configurable XRES - Pin P1[2] is always a GPIO no ECC, that flash can only be used for configuration or general purpose storage boot speed is per 12 MHz FIMO
Operating voltage	1.8 V – 5.5 V	2.7 V – 5.5 V
Operating frequency	67 MHz max	67 MHz max (ES1 versions were formerly specified at 80 MHz)
Packages	100-TQFP, 68-QFN, 48-QFN, 48-SSOP	100-TQFP and 68-QFN. No 48-pin packages.

OBSOLETE

Consideration	PSoC 3	PSoC 5
MHzECO pins	Can be used as GPIOs	MHzECO pins cannot be used as GPIOs
Low-power modes	Sleep: wakeup from a variety of sources including central time wheel (CTW) at 4096 ms max interval, RTC, I^2C, comparator, and pin interrupts (PICU).	Sleep: wakeup only from central time wheel (CTW) at 128 msec max intervals. No PICU (including USBIOs), I^2C, LCD, or comparator wakeup. 10 µF capacitors required on each Vccx for sleep. Watchdog timer (WDT) can only be used in active mode
	Hibernate: wakeup from PICU.	Hibernate: wakeup only from device reset. No PICU (including USBIOs) wakeup.
Internal main oscillator (IMO)	3 MHz to 62 MHz, 1% accuracy at 3 MHz (CY8C38 and CY8C36 parts) With either IMO or crystal as a source, the PLL can be used to generate higher frequencies, with the same accuracy as the source.	3 MHz to 48 MHz, 5% accuracy at 3 MHz. For communication functions such as UART it is recommended to use an external crystal instead of the IMO. With either IMO or crystal as a source, the PLL can be used to generate higher frequencies, with the same accuracy as the source.
USB	IMO has a mode that adjusts to the USB signal as a reference, requiring no external crystal for USB.	For USB a 24 MHz external crystal is required, and bus clock must be > 33 MHz.
Timers	Four fixed-function timers that can operate in counter, timer or PWM mode. UDB-based timer, counter and PWM components can also be created.	Fixed-function timer interrupt outputs not available, can still route the TC output to an interrupt through the DSI. UDB-based timer, counter and PWM components can be created.
UDBs		Sync and pulse modes not available in control registers
Debug	JTAG, SWD, SWV. JTAG support available from a variety of third party sources, see General PSoC Programming	JTAG not available, only SWD. SWV is always on when SWD is being used; impacts usage of P1[3]. Pull-down required on SWDCK if doing SWD over USBIOs.
Electrical specifications	Full-chip Idd at 6 MHz: 1.2 mA typ	Full-chip Idd at 6 MHz: 6 mA typ
	Each Vddio can source up to 100 mA through its pins, and sink up to 100 mA.	Each Vddio can source up to 20 mA through its pins, and sink up to 100 mA.
	Opamp: Vos 2.5 mV max, max drive current 25 mA. Four power modes available.	Opamp: Vos 3 mV max, max drive current 10 mA. Only one power mode (high).
	Del-sig ADC: Ge 0.2%, Vos (buffered) ±0.1 mV, Idd from 1.2 mA to 4 mA	Del-sig ADC: Ge 0.6%, Vos (buffered) ±0.65 mV, Idd 4 mA. No rail-to-rail mode in the buffer, EOC signal cannot be routed through the DSI.
	Voltage reference: ±0.1%	Voltage reference: -0.7%/+0.9%
	Comparator: Vos 9 mV	Comparator: Vos 15 mV
	DAC: update rate 8 Msps, Ge ±2.5%, ZSe ±1 LSB, INL ±1 LSB, DNL ±1 LSB,	DAC: update rate 5.5 Msps, Ge ±5%, ZSe ±2.5 LSB, INL ±3 LSB, DNL ±1.6 LSB, glitches in response
	SC/CT (PGA, TIA, mixer): TIA, PGA Vos 10 mV, TIA input BW 1.5 MHz, PGA Ge ±0.15%	SC/CT (PGA, TIA, mixer): TIA, PGA Vos 20 mV, TIA input BW 1 MHz, PGA Ge ±2%
	LCD: Idd 38 µA, segment driver current 260 µA	LCD: Idd 63 µA, segment driver current 148 µA
	USB: Iusb in configured mode 10 mA	USB: Iusb in configured mode 55 mA
	Flash: total device programming time 5 s typ.	Flash: total device programming time 9 s typ.

OBSOLETE

Hardware Design Considerations

To plan for project migration, the first step is to review your overall system, focusing on the power system, PCB design, and the PSoC Creator project. As Table 1 shows, PSoC 5 has some differences from PSoC 3.

Power System

Table 1 shows several considerations that may affect your product's power system design:

- The minimum operating voltage for PSoC 5 is 2.7 V, whereas PSoC 3 and PSoC 5LP can run down to 1.8 V. So if for example your overall system runs on 1.8 V then you may need to include a 2.7 V supply for PSoC 5. Note that PSoC Vddio voltages and pins can be set up for level shifting, to more easily interface PSoC with other ICs on the PCB.
- PSoC 5 current consumption is higher than that for PSoC 3.
- PSoC 5 low-power modes and wakeup conditions are more restricted than for PSoC 3, so you should review your product's sleep / wakeup requirements.
- The boost feature is not available in PSoC 5.

Pin Assignment and PCB Design

Table 1 also shows considerations that may affect your PSoC pin assignments, as well as your overall PCB layout and design. Specifically:

- External references: PSoC 5 adds one or two SAR ADCs; with the delta-sigma ADC as many as three ADCs are available. The pins listed in Table 2 are used to provide external references for the ADCs. To take advantage of this feature you should reserve these pins from being used for other functions.

  Table 2. ADC Reference Pins

ADC	Pin
Delta-sigma	P0[3] or P3[2]
SAR0	P0[4]
SAR1	P0[2]

  For more information on external reference pins see AN61290, PSoC 3 and PSoC 5 Hardware Starting Guide.

- Package: PSoC 3 has up to two external reset sources - either a dedicated XRES pin, or P1[2] can be configured as a reset pin. For 48-pin packages only the P1[2] configurable reset is available.

  For PSoC 5 there are no 48-pin packages. All PSoC 5 packages have a dedicated external reset (XRES) pin, and P1[2] cannot be configured as a XRES pin.

  So when planning for migration you should use only 68-QFN or 100-TQFP parts, and use only the dedicated XRES pin for device reset.

- The PSoC 5 MHzECO pins cannot be used as GPIOs. Also with PSoC 5, many functions such as UART require a crystal (MHzECO) or other high precision clock source. So to ease migration a crystal should be used as the primary clock source, instead of the IMO.
- JTAG is not available in PSoC 5, only SWD / SWV. This impacts usage of certain P1 pins, including P1[3].
- The EMIF, CAN, and temperature sensor features are not available in PSoC 5.

Analog Performance

Finally, Table 1 shows differences in the electrical specifications that may affect the analog performance of your PSoC Creator project. You should review your product's analog system requirements and make sure that they can be met by both PSoC 3 and PSoC 5.

OBSOLETE

PSoC Creator Considerations

When you do change the device in your product from a PSoC 3 to a PSoC 5, you must update your PSoC Creator project. This application note covers how to migrate PSoC Creator 2.0 projects. If you currently have a PSoC Creator 1.0 project, you should first migrate it to PSoC Creator 2.0. When you do, you may need to handle some issues, including:

- Obsolete devices
- Migrating components from PSoC Creator 1.0

Information on these topics can be found in the PSoC Creator 2.0 Migration Guide.

Once you have a stable and working PSoC 3 project, with PSoC Creator 2.0, you should save an archive or backup copy – this is always good practice. You can then change the project to PSoC 5. There are several steps involved:

- Selecting a PSoC 5 device
- Clearing initial build errors
- Porting firmware

The remainder of this application note details how each of these steps should be done.

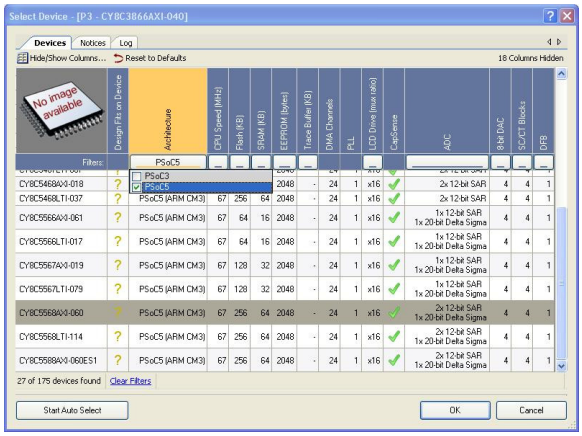
PSoC 5 Device Selection

The first step to migrate a project to PSoC 5 is to select the appropriate PSoC 5 device. The differences noted in Table 1 notwithstanding, PSoC 5 devices have similar packages, pinouts, peripheral blocks, and functions to those in the PSoC 3 family, so in many cases it is easy to find the right device. The best way to start is to review the PSoC 3 and PSoC 5 device datasheets. Also, PSoC Creator offers a handy dialog to assist with device selection, as Figure 1 shows.

Once you have selected your PSoC 5 device, you must change your PSoC Creator project from a PSoC 3 project to a PSoC 5 project—do the following steps:

- Go to menu item Project > Device Selector
- In the dialog box (see Figure 1) click the button under the "Architecture" column.
- Click on the row for the device you want, then click OK. The dialog box will close and the project in the Workspace Explorer window will show the new device.

  Figure 1. PSoC Creator Device Selector Dialog



- Rebuild the changed project.

Initial Build Errors

Even with proper planning for migration, as described previously in

OBSOLETE

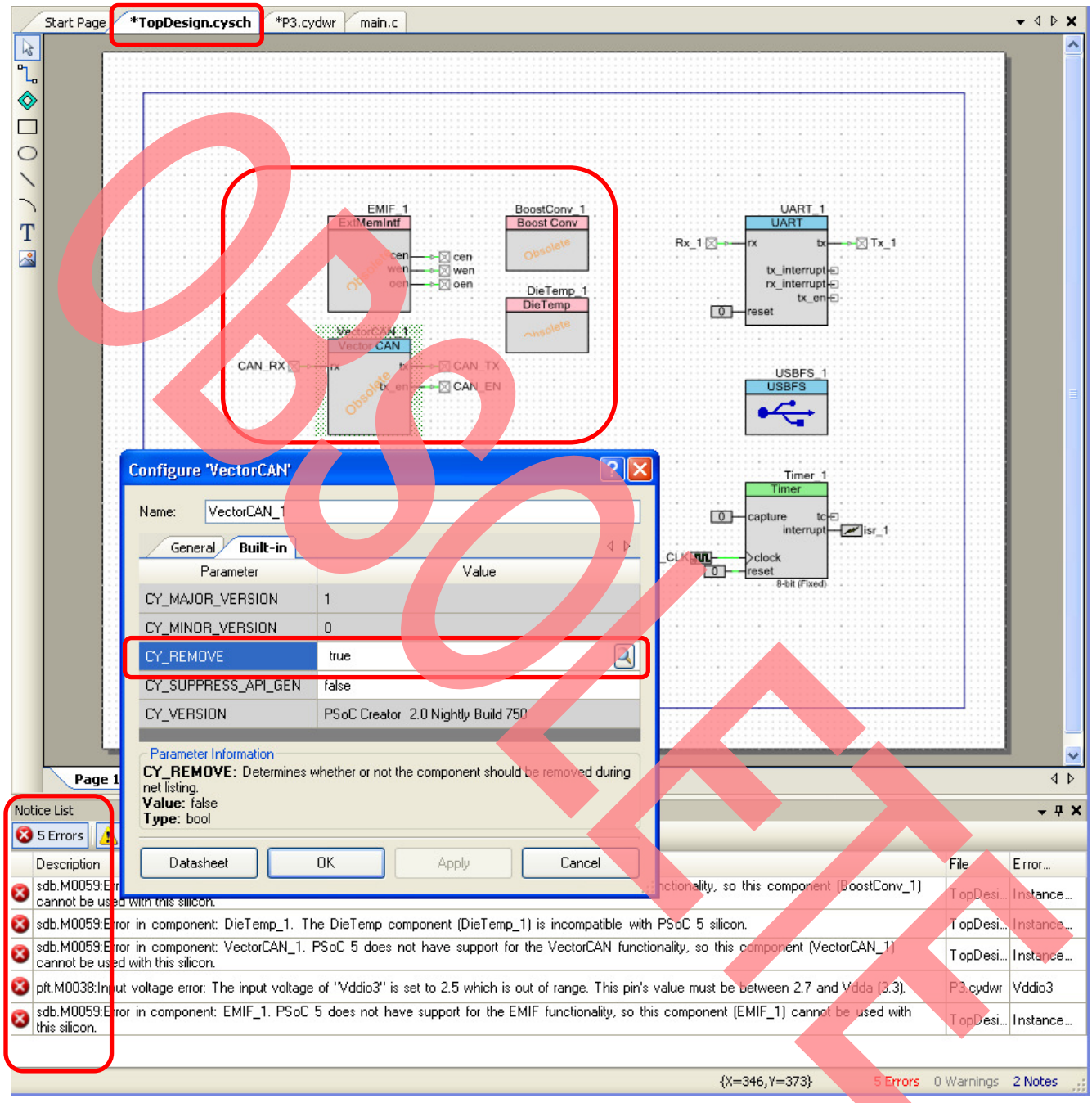
Hardware Design Considerations on page 4, you may still get errors on your first attempt to rebuild the PSoC Creator project. Depending on the components in your project schematic, and the configurations in your *.cydwr* tab, PSoC Creator may show the following:

- Components on the schematic marked as "Obsolete"
- Error messages in the Notice List window

The PSoC Creator components BoostConv, VectorCAN, EMIF, and DieTemp support hardware blocks that exist in PSoC 3 but are not available in PSoC 5. These components must be removed from your project schematic, or their CY_REMOVE configuration parameter set to true, as Figure 2 shows.

OBSOLETE

Figure 2. PSoC Creator Notice List and Initial Error Messages



OBSOLETE

Another source of error messages may be in the pin voltage settings (see Power System on page 4). For PSoC 5, each of these must be from 2.7 to 5.5 V. Remember too that Vdda must be greater than or equal to all other supply voltages—for details see AN61290, PSoC 3 and PSoC 5 Hardware Starting Guide.

You may also get clock errors, for different reasons. The first is that because the PSoC 5 IMO accuracy is 5% at 3 MHz, and the PSoC 3 IMO accuracy is as low as 1% at 3 MHz, you may get warnings that some clocks do not meet your accuracy specifications. This may apply to internal clocks such as for the UART, as Figure 4 shows. The best way to prevent these errors is to use a crystal instead of the IMO as discussed previously in Pin Assignment and PCB Design on page 4—see Figure 6.

Figure 3. PSoC Creator Pin Voltage Settings

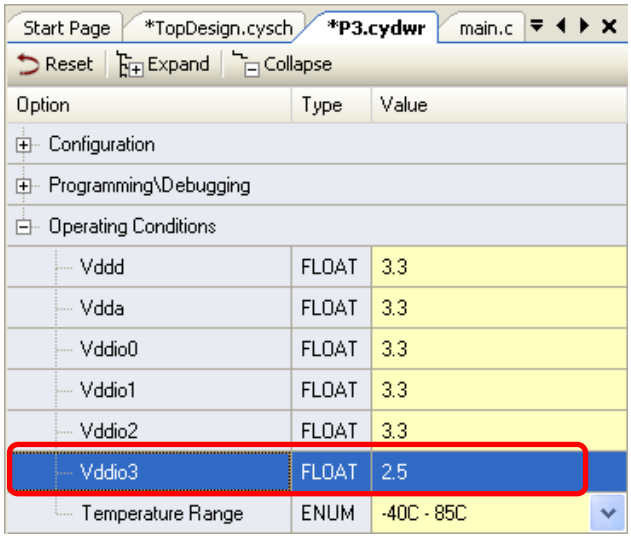
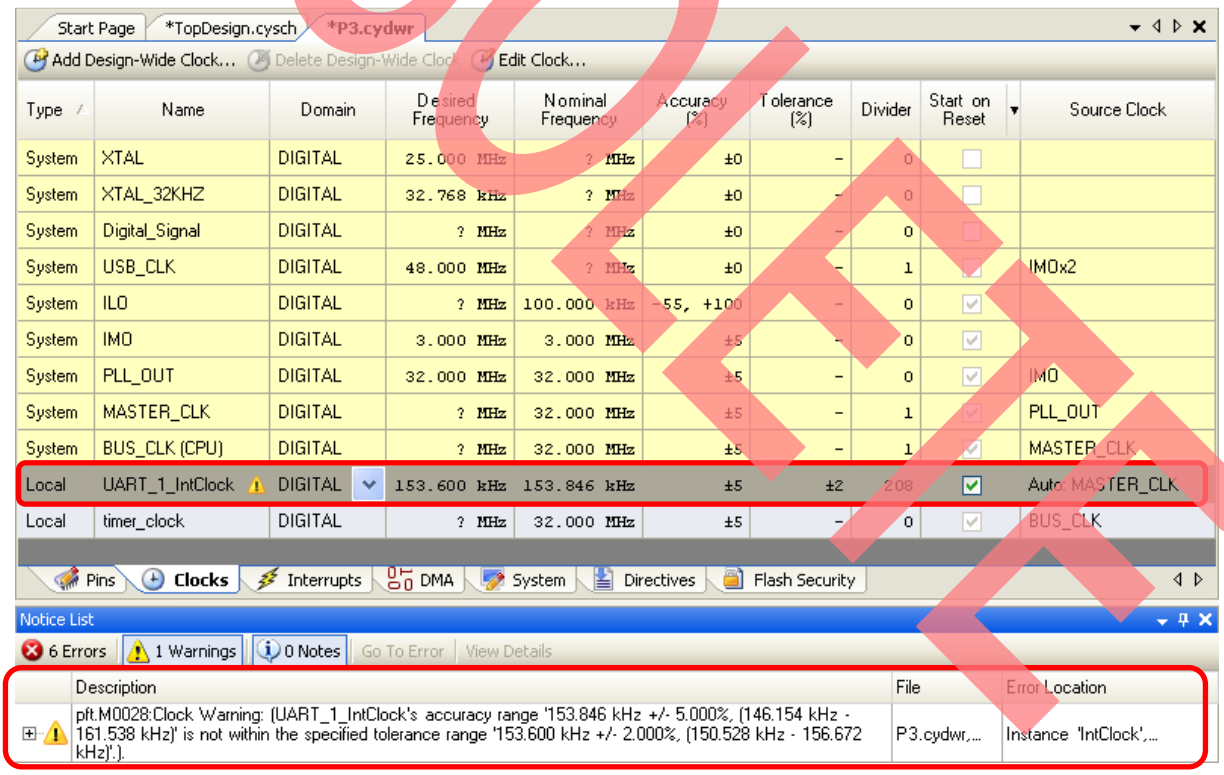


Figure 4. PSoC Creator UART Clock Accuracy Warning



OBSOLETE

The second possible source of clock errors is that for USB to work in PSoC 5, the bus clock must be ≥ 33 MHz, and a 24 MHz crystal is required for the necessary accuracy, as Figure 5 shows.

Figure 5. PSoC Creator USB Clock Errors

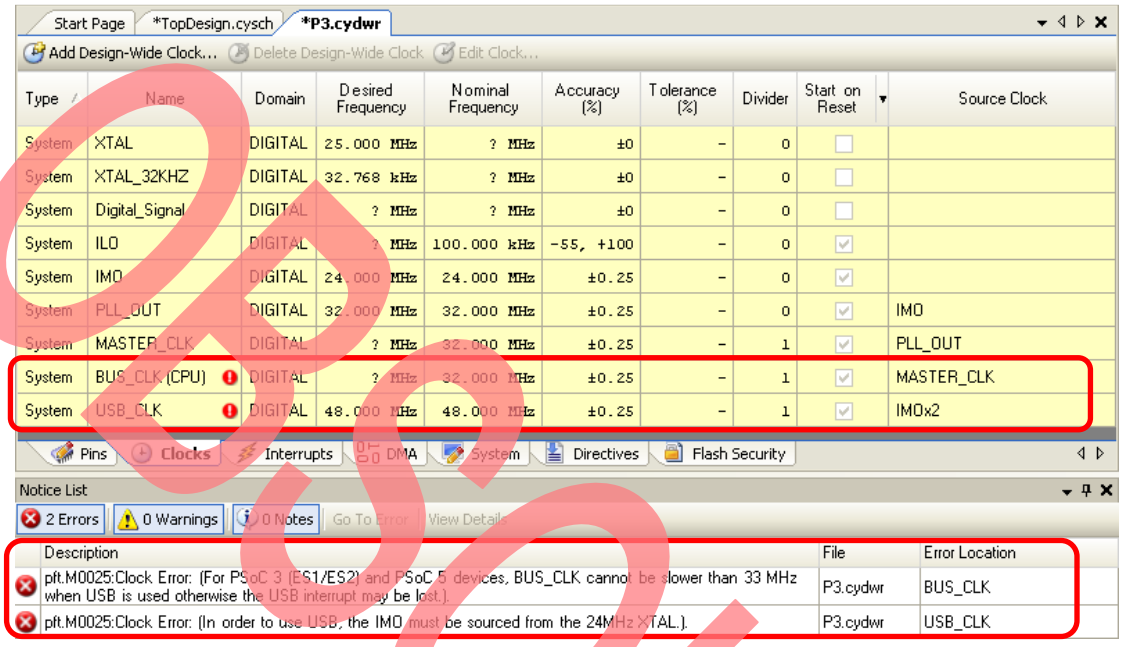
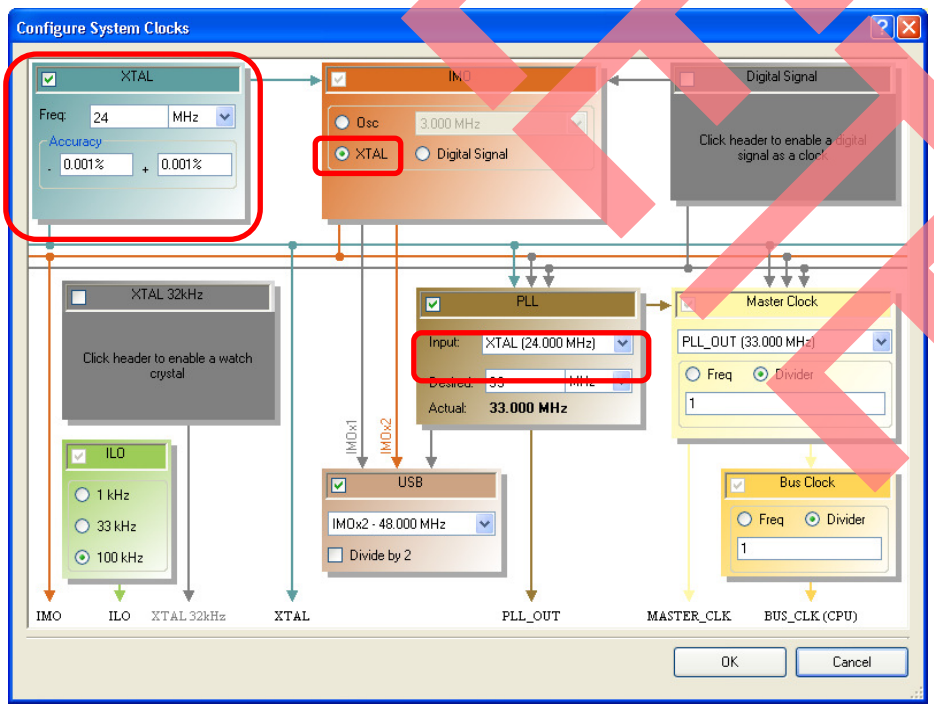


Figure 6 shows the proper clock setup for USB, or for general-purpose use of a 24 MHz crystal. Note that you enter the XTAL accuracy based on the crystal selected. Since most crystals specify accuracy in parts per million (ppm), use the conversion factor 1% = 10,000 ppm.

Figure 6. PSoC Creator Settings for a 24 MHz Crystal, and USB



OBSOLETE

Other PSoC Creator Error Messages

Depending on specific settings in a PSoC 3 design, when you change to a PSoC 5 device PSoC Creator may also display one or more of the following error messages:

1. **IMO Setting:** The IMO clock source was set to xx MHz, which exceeds the maximum allowed value of 48 MHz for this device.

   Adjust the IMO setting in the Clocks tab of the design-wide resource editor and check that your local and design-wide clocks still meet your requirements.
2. **MHzECO Pins Used as GPIOs:** The design uses pins p15[0] and p15[1], which are now dedicated to the crystal (MHzECO).

   Pin instances xxx and yyy shall be re-assigned to new physical pins when the application is next built.
3. **Flash ECC:** Error correcting code (ECC) is no longer supported and has been disabled (it is no longer presented in the System design-wide editor). You may still use this flash memory for configuration data.
4. **SWD Clock Speed:** The SWD debug clock was set to xx MHz, which exceeds the maximum allowed value of yy MHz (based on the design's Vddd setting).

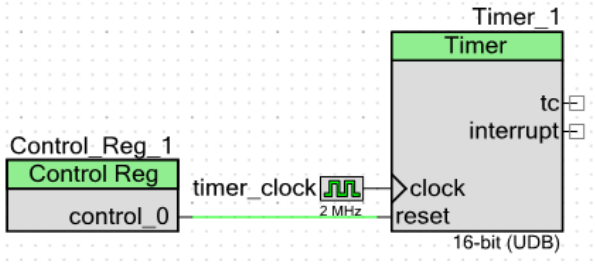
   Your speed has been adjusted to xx MHz in the Program/Debug section of the Tools->Options dialog.
5. **Fast IMO Setting:** The option for a Fast IMO during device startup is no longer available and has been disabled.

Static Timing Analysis Warnings

Another class of warnings that may come up when migrating a design is static timing analysis (STA). These occur when signals between digital blocks may not meet setup and hold time requirements. Let us take a look at one example.

In Figure 7 a control register is driving the reset input of a UDB-based timer. What is not apparent is that the control register is being clocked by the highest frequency clock in the design, i.e., bus clock. With a high-frequency bus clock you can get setup time violations between bus clock and the timer clock:

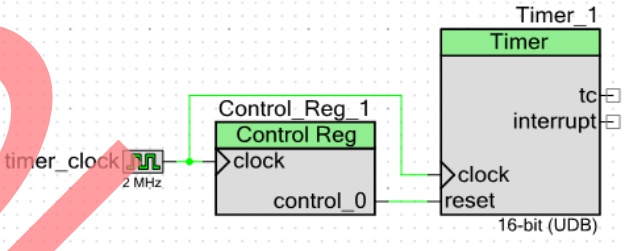
Figure 7. STA Errors From Multiple Clocks




```
Warning: sta.M0019: P3_timing.html:
Warning-1366: Setup time violation found in
a path from clock ( CyBUS_CLK ) to clock
( timer_clock ).
```

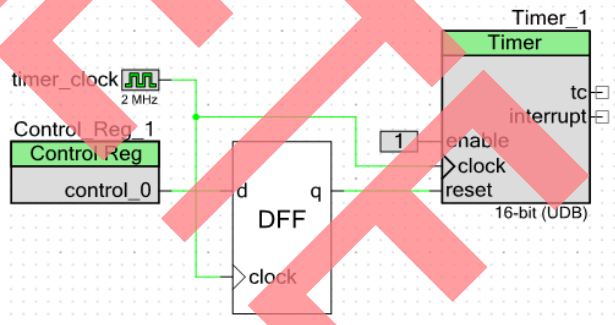
In PSoC 3 this easy to fix, by putting the control register in Sync mode, as Figure 8 shows:

Figure 8. PSoC 3 Control Register Sync Mode



However this solution cannot be used with PSoC 5 because the control registers are clocked only by bus clock. So an alternate solution, which works for both PSoC 3 and PSoC 5, is to use a DFF as Figure 9 shows:

Figure 9. PSoC 3 and PSoC 5 DFF Solution



Note also that this design actually causes another STA warning due to another control register associated with the UDB datapaths in the Timer component. The best way to handle this warning is to turn on the Timer hardware enable and then, assuming you want the Timer to be always enabled, just tie it high.

OBSOLETE

Porting Firmware

In most cases C code written for the Keil 8051 compiler for the PSoC 3 will port directly to the gcc and MDK compilers used with the PSoC 5 ARM Cortex-M3. This includes C code that is auto-generated by PSoC Creator.

However, there are some special considerations that must be kept in mind when porting code between the compilers:

Conditional Compilation

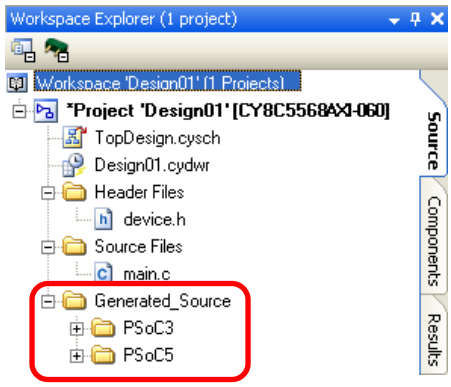
Although not usually needed, conditional compilation can be used for code that cannot be directly ported between 8051 and Cortex-M3:

```
#if (defined(__C51__))
    /* PSoC 3 unique code */
#else /* PSoC 5 */
    /* PSoC 5 unique code */
#endif
```

Editable Code Sections

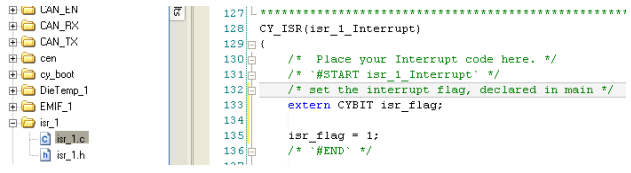
As part of the build process PSoC Creator generates APIs for the components. The APIs are placed in different files, in different folders, for PSoC 3 and PSoC 5, as Figure 10 shows.

Figure 10. Generated Source Folders



Some component APIs, especially interrupt (ISR) APIs, have user editable sections. These sections are delineated by #START and #END comments, as Figure 11 shows.

Figure 11. User Editable Sections of Code



User code in these sections is not automatically copied when rebuilding for a different device. It must be manually copied from the file generated for one device, and pasted to the file generated for the other device.

DMA Addresses

PSoC 3 and PSoC 5 have the same DMA controller (DMAC). The DMAC uses 32-bit addresses (source and destination) stored in two 16-bit registers.

The upper halves of the addresses are specified in a DMA channel:

```
DMA_DmaInitialize(..., upperSrcAddr,
                       upperDestAddr)
```

And the lower halves of the addresses are specified in transaction descriptors (TDs) within a DMA channel:

```
CyDmaTdSetAddress(..., lowerSrcAddr,
                       lowerDestAddr)
```

For PSoC 5, you can easily obtain the upper and lower halves of an address from a (4-byte) pointer variable, by using the HI16 or LO16 macros, defined in the *cytypes.h* file:

```
upperSrcAddr  = HI16(srcArray);
lowerSrcAddr  = LO16(srcArray);
```

For PSoC 3, the contents of a pointer variable cannot be used because the Keil 8051 compiler uses a 3-byte pointer. It contains a 16-bit absolute address and a third byte for the memory space being used (see Appendix A – Memory Maps). Instead, use one of the following code snippets to obtain the upper half of the address:

Source or destination in SRAM or peripheral register:

```
upperSrcAddr = 0;
```

Source in flash:

```
upperSrcAddr = (CYDEV_FLS_BASE) >> 16
```

And as noted previously, conditional compilation can be used:

```
#if (defined(__C51__))
    upperSrcAddr = 0;
    lowerSrcAddr = srcArray;
#else /* PSoC 5 */
    upperSrcAddr = HI16(srcArray);
    lowerSrcAddr = LO16(srcArray);
#endif
```

Endian Format

Endian format refers to how multi-byte variables are stored in a byte-wide memory. In big endian format, the most significant byte is stored in the first byte (lowest address). In little endian format the least significant byte is stored in the lowest address.

For PSoC 3, the Keil 8051 compiler uses big endian format. The PSoC 5 Cortex-M3 and all of its compilers use

OBSOLETE

little endian format. PSoC 3 and PSoC 5 multi-byte peripheral registers use little endian format. When porting code you should consider access of these variables and registers by both the CPU and the DMAC.

CPU Access

The following macros provided by PSoC Creator access registers with byte swapping when needed. These macros are for accessing registers mapped in the first 64 K bytes of the 8051 external data space:

```
CY_GET_REG8(addr)
CY_SET_REG8(addr, value)
CY_GET_REG16(addr)
CY_SET_REG16(addr, value)
CY_GET_REG24(addr)
CY_SET_REG24(addr, value)
CY_GET_REG32(addr)
CY_SET_REG32(addr, value)
```

Use these macros to access registers mapped above the first 64 K bytes of the 8051 external data space:

```
CY_GET_XTND_REG8(addr)
CY_SET_XTND_REG8(addr, value)
CY_GET_XTND_REG16(addr)
CY_SET_XTND_REG16(addr, value)
CY_GET_XTND_REG24(addr)
CY_SET_XTND_REG24(addr, value)
CY_GET_XTND_REG32(addr)
CY_SET_XTND_REG32(addr, value)
```

These macros can be directly ported to PSoC 5 compilers, and they handle byte swapping correctly. For more information on these macros, see the PSoC Creator System Reference Guide.

DMA Access

DMA transaction descriptors (TDs) can be programmed to have bytes swapped while transferring data. The swap size can be set to 2 bytes for 16-bit transfers or 4 bytes for 32-bit transfers. The following examples handle 2- and 4-byte swaps:

```
CyDmaTdSetConfiguration(myTd, 2, myTd,
    TD_TERMOUT0_EN | TD_SWAP_EN);

CyDmaTdSetConfiguration(myTd, 4, myTd,
    TD_TERMOUT0_EN | TD_SWAP_EN |
    TD_SWAP_SIZE4);
```

DMA byte swapping is required only in PSoC 3, when transferring multi-byte parameters between registers and variables in memory. It must be disabled for all other uses, including all PSoC 5 uses.

Timing

To generate more accurate time delay loops, use the CyDelay function. This function selects the number of loop iterations based on processor type and CPU speed. The CyDelay function is defined in *CyLib.c*, which is available in Generated Source files.

Compiler Keywords

The Keil 8051 compiler provides several keywords to place variables in different 8051 memory spaces, to increase code efficiency.

These keywords do not directly port to the gcc and MDK compilers used for the Cortex-M3. PSoC Creator defines several macros that can be used instead, and allow easy porting of code with these keywords—see Table 3.

Table 3. 8051 Memory Spaces, Keywords, and Macros

8051 Memory Space	Keyword	PSoC Creator Macro
Internal RAM	bit, data, idata, bdata, small	CYBIT, CYDATA, CYBDATA, CYIDATA, CYSMALL
Internal SFRs	sfr, sbit	-
External space	pdata, xdata, compact, large, far	CYPDATA, CYXDATA, CYCOMPACT, CYLARGE, CYFAR
Code space (flash)	Code	CYCODE

For more details on how to use the keywords in Table 3, see the application note AN60630, PSoC 3 8051 Code Optimization.

Non-Portable Keil Compiler Keywords

The Keil compiler keywords **sfr**, **sbit**, and **reentrant** are not portable. You should avoid using these keywords, or put them under conditional compile statements, as mentioned on page 10.

You can also define macros that contain the conditional compile statements. This technique is preferred as too many conditional compile statements may make your code difficult to read.

sfr and sbit

Special function registers (SFRs) exist in PSoC 3 as an alternate, faster way to access certain registers see Appendix A – Memory Maps. They are defined using the **sfr** keyword, and bits within those registers are defined using the **sbit** keyword.

These alternative registers are not available in PSoC 5, so there is no way to port definitions of these registers.

(The **sbit** keyword can also be used to access bits in variables of type bdata—for more information see AN60630, PSoC 3 8051 Code Optimization.)

reentrant

By default, the Keil compiler assumes that functions are not reentrant; a function's local variables are stored in fixed memory locations in RAM. If a function must be called from different threads (such as main and interrupt

OBSOLETE

handler), or recursively, then it must be specifically defined as a reentrant function:

```
/* reentrant function declaration */
void delay (uint32) reentrant;

/* reentrant function definition */
void delay (uint32 x) reentrant
{
    . . .
}
```

All of the PSoC 5 compilers define functions as reentrant (which is actually standard in C), and none of them support this keyword. To port functions with this keyword to PSoC 5 you can either define them within conditional compile statements or just redefine the keyword to be ignored:

```
#define reentrant /**/
```

Assembly Language Code

Code written in assembly language cannot be ported, as mnemonics for the 8051 and the Cortex-M3 are different.

Summary

This application note has provided a detailed discussion of considerations for migrating designs between PSoC 3 and PSoC 5 devices.

The best way to avoid problems when migrating designs is to plan ahead. The overall product requirements and system-level design should be reviewed to make certain that both device families can be used. Then, design the PCB so that parts from both families can be used. Finally, design the PSoC Creator project schematic, *.cydwr* file, and code such that the project can be rebuilt for either device.

About the Author

Name:	Mark Ainsworth
Title:	Applications Engineer Principal
Background:	Mark Ainsworth has a BS in Computer Engineering from Syracuse University and a MSEE from University of Washington.
Contact:	mkea@cypress.com

OBSOLETE

Appendix A – Memory Maps

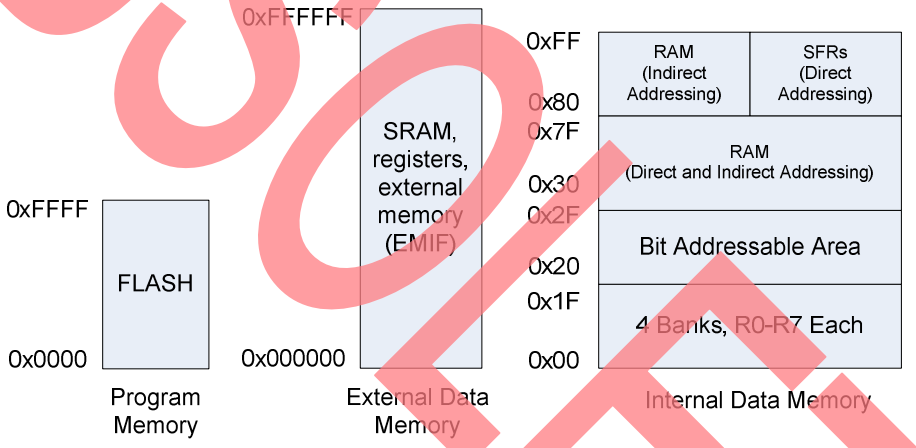
The PSoC 3 and PSoC 5 memory maps are different, due to the architecture of their respective CPUs.

The PSoC 3 8051 memory map consists of three distinct memory spaces, as Figure 12 shows.

- **Code space:** This space is 64 K bytes, and is occupied solely by flash memory. The 8051 executes instructions from this space.
- **External data space:** This space is "external" to the 8051 core but of course is "internal" to the PSoC 3 device. All SRAM, registers, and EMIF addresses are mapped into this space. Flash memory is also mapped into this space, mainly for DMA data access. This space is 16 Mb in size, and requires a 24-bit address to access it.
- **8051 internal data space:** This space is actually part of the 8051 core. It contains 256 bytes of RAM and several special function registers (SFRs). The 8051 uses fast register and bit instructions to access portions of this space. The 8051 hardware stack also occupies this space; the stack can be at most 256 bytes.

For more details see any PSoC 3 datasheet or the application note AN60630, PSoC 3 8051 Code Optimization.

Figure 12. PSoC 3 8051 Memory Map

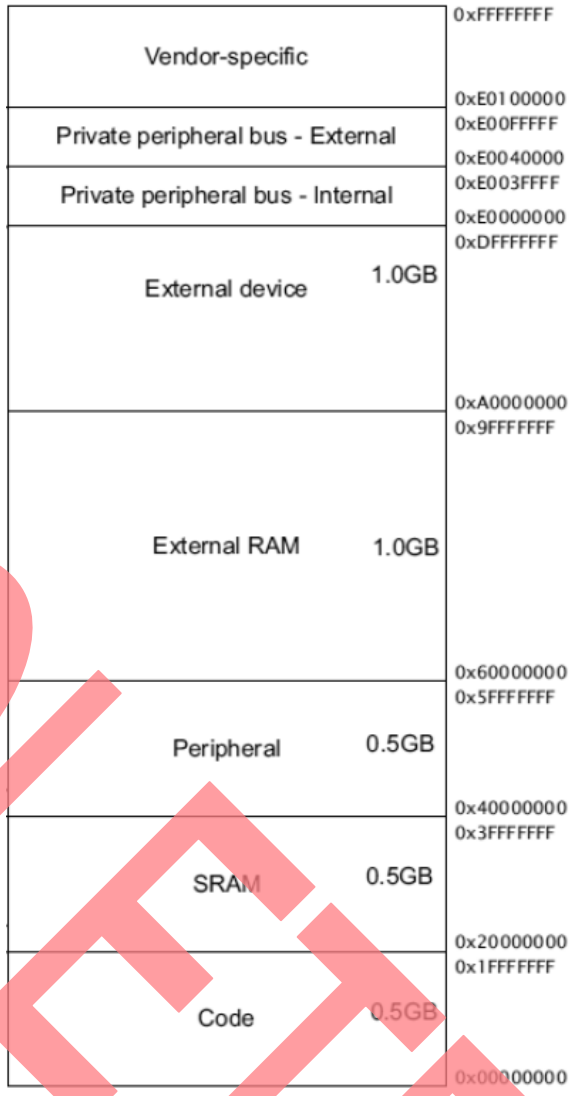


OBSOLETE

The PSoC 5 ARM Cortex-M3 architecture is simpler—it uses a single 32-bit linear memory map, as Figure 13 shows. The SRAM in PSoC 5 actually occupies the addresses 0x1FFF8000 to 0x20007FFF, centered on the boundary between the Cortex-M3 Code and SRAM spaces.

The remainder of the code space is occupied solely by flash, starting at address 0. The PSoC 5 registers occupy the Cortex-M3 Peripheral space.

Figure 13. PSoC 5 Cortex-M3 Memory Map



OBSOLETE

Document History

Document Title: PSoC® 3 to PSoC 5 Migration Guide – AN62083

Document Number: 001-62083

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2946234	SGUP	06/07/2010	New application note
*A	3250426	SGUP	05/05/2011	PSoC® Creator™ 1.0 production
*B	3561484	MKEA	03/26/2012	PSoC 5 updated specifications, PSoC Creator 2.0 Updated template
*C	3837240	MKEA	12/10/2012	Obsoleted

OBSOLETE

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
Optical Navigation Sensors	cypress.com/go/ons
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5

Cypress Developer Community

Community | Forums | Blogs | Video | Training

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2010-2012. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) 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. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

OBSOLETE