AN77835

PSoC® 3 to PSoC 5LP Migration Guide

Author: Mark Ainsworth

Associated Project: No

Associated Part Family: All PSoC 3 and PSoC 5LP parts

Software Version: PSoC® Creator™ 2.1 SP1 and higher

For a complete list of the application notes, click here.

To get the latest version of this application note, or the associated project file, please visit http://www.cypress.com/go/AN77835.

AN77835 provides an overview of topics to consider when designing a project for easy migration between PSoC® 3 and PSoC 5LP devices. Device considerations, PSoC Creator topics, and firmware porting are discussed.

Contents

Introduction 1
Why Migrate? 2
Device Differences 2
Hardware Design Considerations 3
  Power System 3
  Pin Assignment and PCB Design 3
  Analog Performance 3
PSoC Creator Considerations 4
  PSoC 5LP Device Selection 4
  Analog Routing Differences 5
Porting Firmware 5
  Conditional Compilation 5
  Editable Code Sections 6
  DMA Addresses 6
  Endian Format 7
  Timing 7
  Compiler Keywords 8
  Assembly Language Code 8
Summary 9
Related Application Notes 9
Appendix A – Memory Maps 10
Worldwide Sales and Design Support 13

Introduction

PSoC 3 and PSoC 5LP devices are designed for easy migration of designs from PSoC 3 to PSoC 5LP. Although there are some differences such as the CPU cores, 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 is as simple as specifying a new part then rebuilding the project.

However, there are some considerations that you should keep in mind when planning for migration. For example, additional features in PSoC 5LP may make it desirable to reassign pins on the PCB. A faster and more efficient CPU may suggest 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 5LP. Both device level and PSoC Creator project level considerations are covered. If you are new to PSoC 3 or PSoC 5LP, see one of the PSoC 3 or PSoC 5LP datasheets, AN54181, Getting Started with PSoC 3, or AN77759, Getting Started with PSoC 5LP. For hardware design topics, see AN61290, PSoC 3 and PSoC 5LP Hardware Starting Guide.

Why Migrate?

The major difference between PSoC 3 and PSoC 5LP is the CPU core: PSoC 3 has an 8-bit 8051 and PSoC 5LP has a 32-bit ARM Cortex-M3. Also, PSoC 5LP has more flash and SRAM memory than PSoC 3, has a higher maximum operating frequency (80 MHz), 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 5LP features do come at an additional cost. Therefore, before migrating it makes sense to determine if it really is necessary. There are elements in the PSoC 3 and PSoC 5LP architecture that may reduce the need to migrate. They include the following:

1. All PSoC 3 and PSoC 5LP 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 5LP devices 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. For more information, see AN82250, PSoC PLDs and AN82156, PSoC Datapaths.
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 5LP 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 5LP. 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 the PSoC 3 and PSoC 5LP 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 5LP

Consideration	PSoC 3	PSoC 5LP
CPU	8-bit 8051, dual DPTR, most instructions execute in 1 or 2 cycles, supported in PSoC Creator by Keil compiler. CPU has a separate clock divider sourced from bus clock.	32-bit ARM Cortex-M3, supported in PSoC Creator by gcc and MDK compilers. CPU is always clocked from bus clock.
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 5LP		One or two SAR ADCs: 12-bit, up to 1 Msps
Operating frequency	67 MHz max	80 MHz max
Packages	100-TQFP, 68-QFN, 48-QFN, 48-SSOP, 72-CSP	100-TQFP, 68-QFN, and 99-CSP. No 48-pin packages.
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 74 MHz, 1% accuracy at 3 MHz With either IMO or crystal as a source, the PLL can be used to generate higher frequencies, with the same accuracy as the source.
Debug	JTAG, SWD, SWV JTAG support available from a variety of third party sources, see General PSoC Programming	JTAG, SWD, SWV, TRACEPORT JTAG support available from a variety of third party sources, see General PSoC Programming

Consideration	PSoC 3	PSoC 5LP
Electrical specifications	Full-chip I_{DD} at 6 MHz: 2.3 mA typ	Full-chip I_{DD} at 6 MHz: 3.1 mA typ
	GPIO: input capacitance 7 pF max	GPIO: input capacitance 9 pF max
	Voltage reference: ±0.1% (CY8C38 and CY8C36 parts)	Voltage reference: ±0.1%
	Flash: total device programming time 1.5 s typ.	Flash: total device programming time 5 s typ.
	Interrupt response time: 25 CPU cycles max.	Interrupt response time: 12 CPU cycles max.

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 5LP has some differences from PSoC 3.

Power System

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

- Depending on usage, PSoC 3 and PSoC 5LP consume different amounts of current. Generally, PSoC 3 consumes less current than PSoC 5LP.

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. Here are some considerations:

- External references: PSoC 5LP 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.

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

Table 2. ADC Reference Pins

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

- Package: PSoC 3 has one or 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 5LP there are no 48-pin packages. All PSoC 5LP packages have a dedicated external reset (XRES) pin, and P1[2] cannot be configured as a XRES pin.

  PSoC 3 is also available in a 72-ball CSP package, and PSoC 5LP in a 99-ball CSP package. To migrate between the devices in CSP packages, a PCB redesign is required.

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

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 5LP.

PSoC Creator Considerations

When you change the device in your product, you must update your PSoC Creator project. This application note covers how to migrate PSoC Creator projects created with version 2.1 SP1 or higher. If you currently have a PSoC Creator 1.0 project, you should first migrate it to PSoC Creator 2.1 SP1. 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 and the PSoC Creator 2.1 SP1 Migration Guide.

Once you have a stable and working project with PSoC Creator 2.1 SP1 or higher, save an archive or backup copy – this is always good practice. You can then change the device in that project. There are several steps involved:

- Selecting the new device
- Clearing initial build errors
- Porting firmware

The remainder of this application note shows how each of these steps is done.

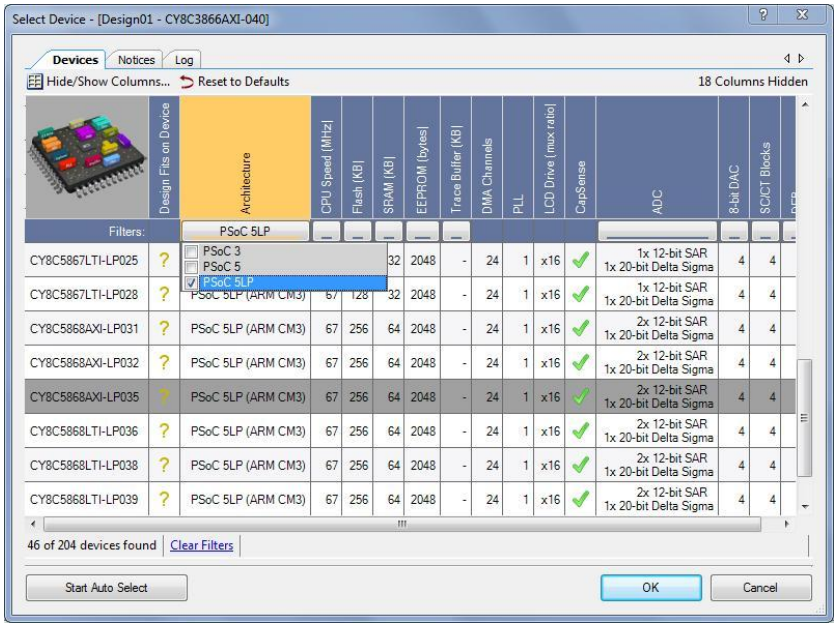
PSoC 5LP Device Selection

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

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

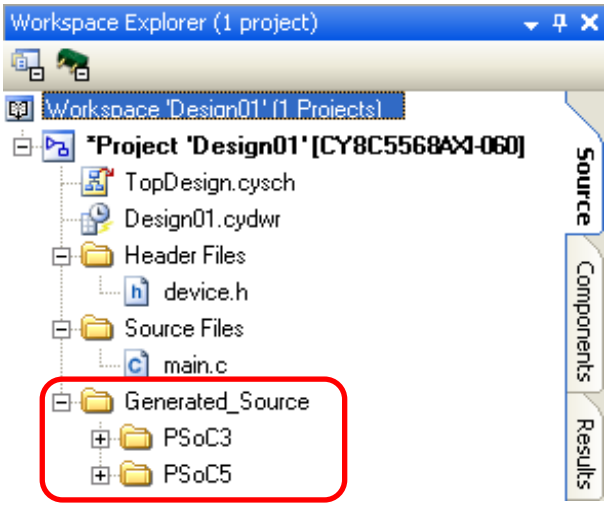
- Go to menu item Project > Device Selector
- In the dialog box (see Figure 1) click the button under the "Architecture" column, and make sure that “PSoC 5LP” is checked.

Figure 1. PSoC Creator Device Selector Dialog



- Click the row for the device you want, then click OK. The dialog box closes and the project in the Workspace Explorer window shows the new device.
- Rebuild the changed project. Note that the source files generated by PSoC Creator are saved in a folder “PSoC5”, separate from the “PSoC3” folder, as Figure 2 shows. Your PSoC 3 project files are not changed.

Figure 2. Generated Source Folders



Analog Routing Differences

The analog routing in PSoC 5LP is slightly different from that in PSoC 3; for details see Figure 8-2 in the respective datasheets. PSoC Creator is aware of the routing differences, so in many cases you can just rebuild the project for the new device, and the project’s functions and performance will remain unchanged.

However, note that you can constrain or manually optimize the analog routing, by using the Manual Routing components or the Analog Editor in PSoC Creator. If your design has these constraints, it may not directly port to the new device. In this case you must either change your usage of the Manual Routing components or re-route manually using the Analog Editor.

Porting Firmware

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

However, there are some special considerations to keep in mind when porting code between the compilers:

Conditional Compilation

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

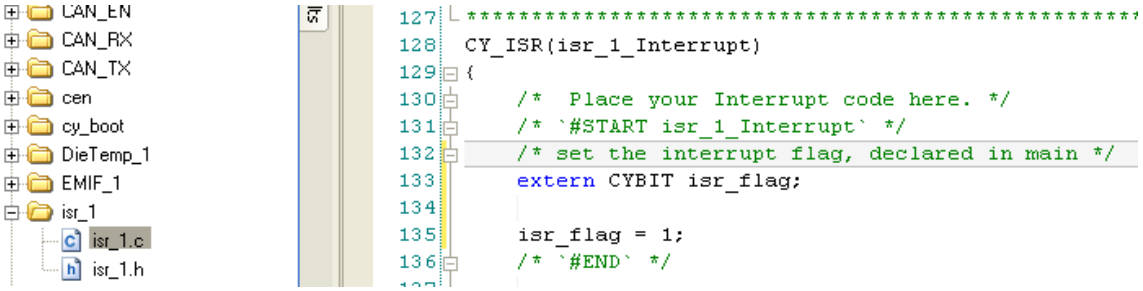
```
#if (defined(__C51__))
    /* PSoC 3 unique code */
#else /* PSoC 5 */
    /* PSoC 5 unique code */
#endif
```

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 2 shows.

Some component API functions, especially interrupt (ISR) functions, have user editable sections. These sections are delineated by #START and #END comments, as Figure 3 shows.

Figure 3. User Editable Sections of Component API Code



Your 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

Both device families have the same DMA controller (DMAC). The DMAC uses 32-bit addresses (source and destination), each stored in two 16-bit registers. The upper halves of the addresses are specified in a DMA channel:

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

The lower halves of the addresses are specified in transaction descriptors (TDs) associated with a DMA channel:

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

For more information, see AN52705, Getting Started with DMA.

For PSoC 5LP, 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 that designates the memory space being used (see Appendix A). 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
```

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 5LP Cortex-M3 and all of its compilers use little endian format. PSoC 3 and PSoC 5LP multi-byte peripheral registers all 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 KB 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 KB 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 5LP 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 5LP 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, bdata, idata, 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 information 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 – see Conditional Compilation.

You can also define macros that contain the conditional compile statements. This technique is preferred because 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 AN60630. They are defined using the **sfr** keyword, and bits within those registers are defined using the **sbit** keyword. Because these alternative registers are not available in PSoC 5LP, there is no way to port the definitions of these registers.

reentrant

By default, the Keil compiler assumes that functions are not reentrant – a function's local variables are not stored on the stack; instead they are stored in fixed memory locations in RAM. If a function is called from different threads (such as main and interrupt 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
{
    . . .
}
```

Both PSoC 5LP compilers define functions as reentrant (which is actually standard in C), and neither of them support this keyword. To port functions with this keyword to PSoC 5LP, 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 from PSoC 3 to PSoC 5LP 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 all devices that you plan to use.

Related Application Notes

AN54181 – Getting Started With PSoC 3

AN77759 – Getting Started With PSoC 5LP

AN61290 – PSoC 3 and PSoC 5LP Hardware Starting Guide

AN52705 – Getting Started with PSoC DMA

AN60630 – PSoC 3 8051 Code and Memory Optimization

AN89610 – PSoC 4 and PSoC 5LP ARM Cortex Code Optimization

AN82156 – Designing PSoC Creator Components With UDB Datapaths

AN82250 – Implementing Programmable Logic Designs

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, as well as many years experience designing and building embedded systems.

Appendix A – Memory Maps

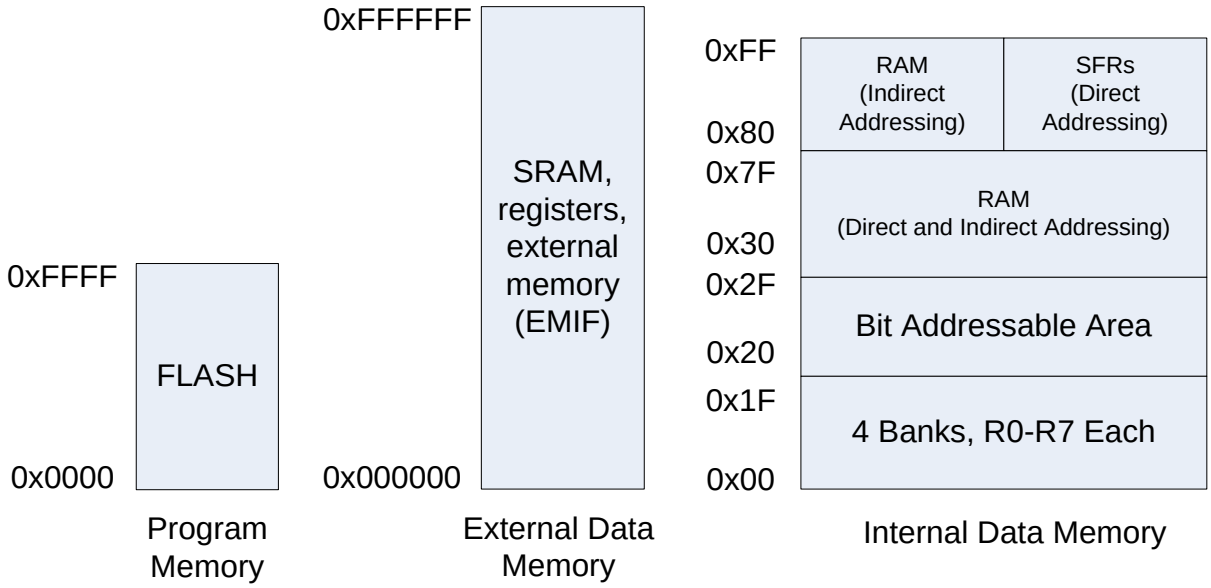
The PSoC 3 and PSoC 5LP memory maps are different, due to the architecture of their respective CPUs.

The PSoC 3 8051 memory map consists of three distinct memory spaces, as Figure 4 shows.

- **Code space:** This space is 64 Kb, 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 size is at most 256 bytes.

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

Figure 4. PSoC 3 8051 Memory Map

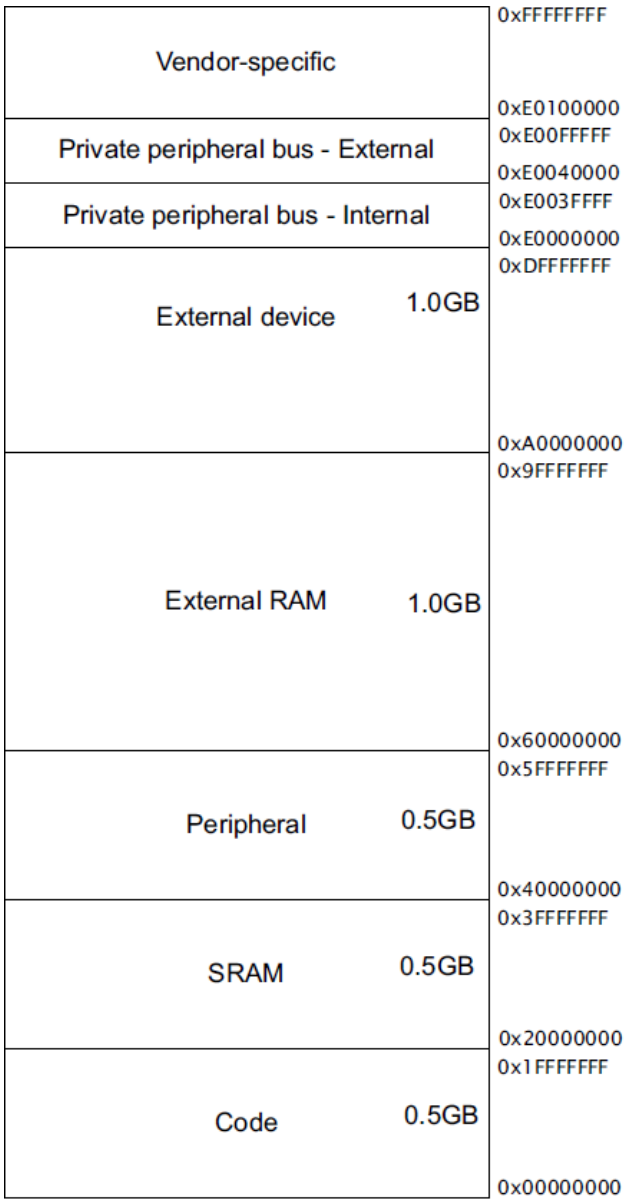


The PSoC 5LP ARM Cortex-M3 architecture is simpler—it uses a single 32-bit linear memory map, as Figure 5 shows. The SRAM in PSoC 5LP actually occupies the addresses 0x1FFF8000 to 0x20007FFF, centered on the boundary between the Cortex-M3 Code and SRAM spaces.

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.

For more information, see any PSoC 5LP datasheet or the application note AN89610, PSoC 4 and PSoC 5LP ARM Cortex Code Optimization.

Figure 5. PSoC 5LP Cortex-M3 Memory Map



Document History

Document Title: AN77835 - PSoC® 3 to PSoC 5LP Migration Guide

Document Number: 001-77835

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	3561494	MKEA	03/26/2012	New application note
*A	3708126	MKEA	11/21/2012	Added section on analog routing differences. Changed max operating frequency of PSoC 5LP from 80 to 67 MHz.
*B	4270683	MKEA	02/05/2014	Removed statement about PSoC 5LP datasheets being preliminary. Added mention of CSP packages. Restored maximum operating frequency to 80 MHz. Deleted reference to obsolete AN84741. Updated to *L template. Miscellaneous minor edits.
*C	4876816	MKEA	08/07/2015	Deleted EEPROM and boost from Table 1. Added references to AN89610. Miscellaneous minor text edits.
*D	5713847	AESATMP9	04/26/2017	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

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

©Cypress Semiconductor Corporation, 2012-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.