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THIS SPEC IS OBSOLETE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 No: 001-98546

Spec Title: AN98546 - DESIGNING WITH THE CYPRESS SPI PDD

Replaced by: None

AN98546

Designing with the Cypress SPI PDD

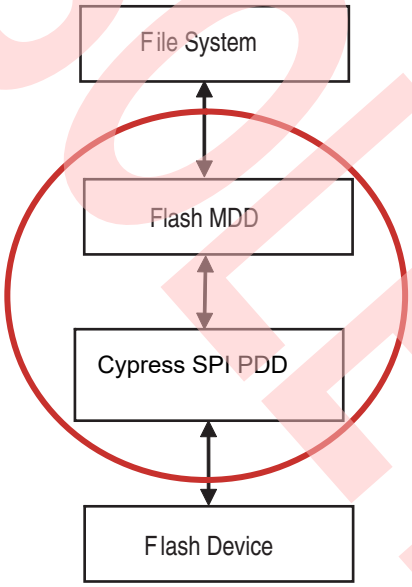
Author: Garret Swalling

AN98546 describes the features and parameters of the Cypress SPI physical device driver (PDD) to unlock the maximum potential for Cypress SPI devices in Windows Embedded platforms.

1 Introduction

The Microsoft® Flash MDD (Model Device Driver) presents a block device interface to file systems in Microsoft Windows® CE. The Cypress SPI (Serial Peripheral Interface) Flash PDD (Physical Device Driver) is the low-level driver that enables the Flash MDD to operate on Cypress SPI devices. The Cypress SPI PDD can also be integrated into the OAL (OEM Adaptation Layer) in order to program raw data to the flash memory with no MDD metadata. This application note describes the features and parameters of the Cypress SPI PDD to unlock the maximum potential for Cypress SPI devices in Windows Embedded platforms.

Figure 1. Block Driver



2 Features

- Detects device characteristics from CFI information or device identification data
- Supports single, dual, or quad bit access
- Optionally reserves code space at the start and/or end of the flash array
- Supports various file system sector sizes
- Modular code allows for simple platform customizing

3 Customizing

The Cypress SPI PDD can be customized to meet the needs of different systems. Separate libraries are linked together to create the Cypress SPI PDD. These libraries include the core PDD logic, the SPI controller interface, and flash parameters. Any one of these modules can be updated without having to rebuild the other modules.

Initially, each library must be built before linking the DLL in the project build. Using Platform Builder, the component libraries can be compiled from the Solution Explorer window by browsing to the applicable component under PUBLIC\SPN_SPI.

```
+---Catalog
+---CESYSGEN
+---oak
|   +---FILES
|   +---lib
|   +---target
+---SPICTL
|   +---SpnPci3
|   +---SpnPci4
|   +---SpnPci4Board226
|   +---stub
+---SPIDRVR
+---SPIPARMS
|   +---oal
|   +---registry
+---SPIPCI
```

The SPICTL tree contains multiple options for code that is specific to the SPI controller. These modules contain the low-level routines to initialize and access the SPI controller for a given platform. The stub code (TemplateCtl.cpp) is a starting point for writing a new SPI controller interface. The SpnPci3 and SpnPci4 examples show how different SPI controllers are implemented with two versions of the Cypress PISMO PCI card. In order to support devices larger than 16MB, select a 4-byte addressing method for the n4CycleAddrMethod member of the SPI controller storage descriptor in the SpiControllerInit routine. Valid options are defined by the FOUR_ADDR_METHOD enumeration in spn_spi.h.

The SPI controller options also appear as radio button selections in the catalog. Note that changing this catalog selection does not force a rebuild of the flash driver DLL (spidrvr.dll). Rebuild any component library before building the Windows CE project. This will cause the correct libraries to be linked.

The SPIPARMS tree contains two options for providing flash and system parameters to the core Cypress SPI PDD code. The registry example (registryparms.cpp) queries the system registry for most of the parameter values. This code is built by default. The OAL example (oalparms.cpp) shows how to specify these parameters statically. If the Cypress SPI PDD is integrated into the OAL, the registry is not available. Parameters for a given platform must be specified with either approach.

The following table shows the Cypress SPI PDD parameters and the associated registry entries. It is assumed that direct map mode is never used when storing file system data, so no registry entry is needed.

Table 1. SPI PDD Parameters

Parameter	Registry Entry
SPN_PARM_BASE_ADDR	“MemBase”
SPN_PARM_SECTORSIZE	“FileSystemSectorSize”
SPN_PARM_RESERVED_AT_START	“ReservedSizeAtStart”
SPN_PARM_RESERVED_AT_END	“ReservedSizeAtEnd“
SPN_PARM_BUS_WIDTH_BITS	“BusWidthBits”
SPN_PARM_FAST_READ	“FastRead”
SPN_PARM_HIGH_PERF	“HighPerformance”
SPN_PARM_DDR	“DoubleDataRate”
SPN_PARM_LC	“LatencyCode”
SPN_PARM_DIRECT_MAP_MODE	–

4 Registry Entries

Several registry entries are used to configure the Cypress SPI PDD.

Table 2. SPI PDD Registry Entries

Registry Entry	Optional?	Description
MemBase	No	Physical start of the flash array
MemLen	No	This value is not used directly by the PDD
ReservedSizeAtStart	Yes	Reserved bytes from start of flash
ReservesSizeAtEnd	Yes	Reserved bytes from end of flash
FileSystemSectorSize	Yes	Minimum sector size reported to Flash MDD
BusWidthBits	Yes	1 for single, 2 for dual, 4 for quad I/O operations
FastRead	Yes	1 (TRUE) to enable the Fast Read command
HighPerformance	Yes	1 (TRUE) for High Performance I/O reads
DoubleDataRate	Yes	1 (TRUE) for DDR reads
LatencyCode	Yes	Latency Code for CR1 (0, 1, 2, or 3)

The “ReservedSizeAtStart” and “ReservedSizeAtEnd” registry entry values are rounded up to the next erase block size.

The “FileSystemSectorSize” must match the sector size for the file system used. For the best performance, it should be set to the flash page size. This value must not be less than the page size, but may be an even multiple of the page size.

Setting the “BusWidthBits” entry to “1” indicates single bit I/O should be used for read and program. The value “2” indicates dual I/O for reads and single bit I/O for program. The value “4” indicates quad I/O for both read and program.

The “FastRead”, “HighPerformance”, and “DoubleDataRate” entries, along with “BusWidthBits”, are used to determine the read command for flash access.

Table 3. Read Command Valid Registry Entry Combinations (Sheet 1 of 2)

DoubleDataRate	HighPerformance	FastRead	4B Addr	BusWidthBits	Instruction
0	0	0	0	1	0x03
0	0	0	0	2	0x3B
0	0	0	0	3	—
0	0	0	0	4	0x6B
0	0	0	1	1	0x13
0	0	0	1	2	0x3C
0	0	0	1	3	—
0	0	0	1	4	0x6C
0	0	1	0	1	0x0B
0	0	1	0	2	—
0	0	1	0	3	—
0	0	1	0	4	—
0	0	1	1	1	0x0C
0	0	1	1	2	—
0	0	1	1	3	—
0	0	1	1	4	—
0	1	0	0	1	—
0	1	0	0	2	0xBB
0	1	0	0	3	—
0	1	0	0	4	0xEB

Table 3. Read Command Valid Registry Entry Combinations (Sheet 2 of 2)

DoubleDataRate	HighPerformance	FastRead	4B Addr	BusWidthBits	Instruction
0	1	0	1	1	—
0	1	0	1	2	0xBC
0	1	0	1	3	—
0	1	0	1	4	0xEC
0	1	1	0	1	—
0	1	1	0	2	—
0	1	1	0	3	—
0	1	1	0	4	—
0	1	1	1	1	—
0	1	1	1	2	—
0	1	1	1	3	—
0	1	1	1	4	—
1	0	0	0	1	0x0D
1	0	0	0	2	0xBD
1	0	0	0	3	—
1	0	0	0	4	0xED
1	0	0	1	1	0x0E
1	0	0	1	2	0xBE
1	0	0	1	3	—
1	0	0	1	4	0xEE
1	0	1	0	1	0x0D
1	0	1	0	2	—
1	0	1	0	3	—
1	0	1	0	4	—
1	0	1	1	1	0x0E
1	0	1	1	2	—
1	0	1	1	3	—
1	0	1	1	4	—
1	1	0	0	1	—
1	1	0	0	2	0xBD
1	1	0	0	3	—
1	1	0	0	4	0xED
1	1	0	1	1	—
1	1	0	1	2	0xBE
1	1	0	1	3	—
1	1	0	1	4	0xEE
1	1	1	0	1	—
1	1	1	0	2	—
1	1	1	0	3	—
1	1	1	0	4	—
1	1	1	1	1	—
1	1	1	1	2	—
1	1	1	1	3	—
1	1	1	1	4	—

Setting the “LatencyCode” entry to 2 will cause 10b to be written to Configuration Register 1 (bits 7 and 6) for devices that support this feature. The Latency Code bits determine the dummy and mode cycles for certain read commands. Refer to the flash device data sheet for more details.

The “MemLen” registry key must exist for the PDD to successfully call MmMapIoSpace, which maps the physical flash address to a virtual address space the PDD can access. The “MemLen” value is ignored.

The remaining registry entries are documented at www.msdn.com. Search for “block driver registry” and “MemBase MemLen”.

Below is an example of registry settings for a Cypress SPI device, without reserving space:

```
[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\SPIDRVR]
    "Dll"="flashmdd.dll"
    "FlashPddDll"="spidrvr.dll"
    "Order"=dword:2
    "Prefix"="DSK"
    "Ioctl"=dword:4
    "Profile"="SpansionSPI"
    "IClass"="{A4E7EDDA-E575-4252-9D6B-4195D48BB865}"
    "MemBase"=dword:00000000
    "MemLen"=dword:00000000
    "FileSystemSectorSize"=dword:200
    "ReservedSizeAtStart"=dword:00000000
    "ReservedSizeAtEnd"=dword:00000000
    "BusWidthBits"=dword:1
    "FastRead"=dword:1
    "HighPerformance"=dword:0
    "DoubleDataRate"=dword:0
    "LatencyCode"=dword:0

; Override names in default profile
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\SpansionSPI]
    "Name"="MSFLASH for Spansion SPI"
    "Folder"="SPI Flash"
    "PartitionDriver"="flashpart.dll"
    "DefaultFileSystem"="FATFS"

[HKEY_LOCAL_MACHINE\System\StorageManager\AutoLoad\SpansionSPI]
    "DriverPath"="Drivers\\BuiltIn\\SPIDRVR"
    ; LoadFlags 0x01 == load synchronously
    "LoadFlags"=dword:1
    "Order"=dword:0
```

The following optional registry entries can be used to control how the Storage Manager handles the flash block drive at boot time.

Table 4. Optional Registry Entries

Registry Entry	Default	Description
AutoMount	1	Automatically attempt to mount, if set to 1
AutoPart	0	Automatically create largest possible partition, if set to 1
AutoFormat	0	Automatically format the store when unformatted, if set to 1

In order to override the Storage Manager default value for a given entry, add the entry to the profile key in spn_spi.reg, for example:

```
[HKEY_LOCAL_MACHINE\System\StorageManager\Profiles\SpansionSPI]
"Name"="MSFLASH for Spansion SPI"
"Folder"="SPI Flash"
"PartitionDriver"="flashpart.dll"
"DefaultFileSystem"="FATFS"
"AutoPart"=dword:1
"AutoFormat"=dword:1
```

One way to manually format the drive and create a partition is using the Storage Manager Control Panel applet. This approach is useful during integration and development with the flash block driver.

5 Flash Regions

The PDD reports to the Flash MDD the number of flash regions, where each region has a uniform erase block size. The Flash MDD supports one or more partitions per flash region, but not a single partition spanning multiple flash regions. The Cypress SPI PDD presents a single flash region with uniform size erase blocks. Since a single flash region is reported, a partition can span the entire Cypress SPI device.

Sub-sectors (parameter blocks) are ignored and uniform sectors are used. However, if the parameter blocks are not intended for file system storage, they can be excluded from management by the Flash MDD using the ReservedSizeAtStart and/or ReservedSizeAtEnd registry entries.

6 Reserving Space

In order to hide part of the flash array from the Flash MDD, registry entries can be used to reserve space at the start and/or end of the flash. Care must be taken to ensure there is no conflict between software accessing the different areas of flash. The Flash MDD, along with the partition manager, will control access to the area of flash that is not reserved. File system storage will require program and erase operations on the flash device. During these operations, the flash device will reject read data commands.

7 Code Execution

The reserved area can only be used for software that does not execute while the file system is running, like startup code. Likewise, the reserved area can store the Windows CE OS image (nk.bin) if the OS image is entirely shadowed to RAM. In order to demand page the OS image from flash, a BinFS partition can be created in the flash area managed by the Flash MDD.

8 Example Parameters

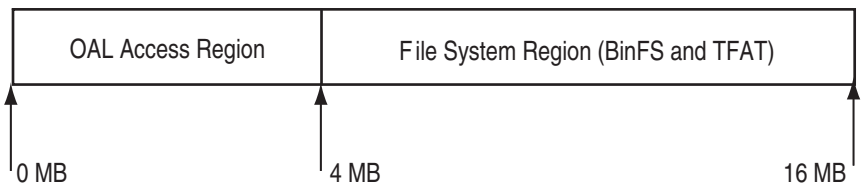
This example platform contains a S25FL129P device. The physical page size is 256B, so the logical sector size is 512B. The first 4 MB are reserved for a bootloader. No space is reserved at the end (top) of the flash.

The non-reserved area is partitioned by the flash partition manager into one partition for storing and demand paging the OS image with BinFS, and one partition for data file storage with TFAT. The S25FL129P supports quad read and program. The bus width is set to 4 and high performance reads are enabled for maximum throughput.

The following run-time registry entries (spn_spi.reg) configure the Cypress SPI PDD to support the Flash MDD for file system data storage.

```
"MemBase"=dword:80000000
"MemLen"=dword:00000000
"FileSystemSectorSize"=dword:200
"ReservedSizeAtStart"=dword:00400000
"ReservedSizeAtEnd"=dword:00000000
"BusWidthBits"=dword:4
"FastRead"=dword:0
"HighPerformance"=dword:1
"DoubleDataRate"=dword:0
```

Figure 2. Memory Map



The following example parameter values (oalparms.cpp) allow access to the first 4 MB of the flash array in order to program and update the bootloader code. The remaining 12 MB of the flash array are reserved from the perspective of the OAL. This example is for the same platform described above.

Table 5. Boot Time Parameters

Parameter	Value
SPN_PARM_BASE_ADDR	0x80000000
SPN_PARM_SECTORSIZE	0x200
SPN_PARM_RESERVED_AT_START	0
SPN_PARM_RESERVED_AT_END	0x00C00000
SPN_PARM_BUS_WIDTH_BITS	4
SPN_PARM_FAST_READ	FALSE
SPN_PARM_DIRECT_MAP_MODE	TRUE
SPN_PARM_HIGH_PERF	TRUE
SPN_PARM_DDR	FALSE

9 Conclusion

Please consult your flash device data sheet to ensure your platform adheres to the device specifications. If you have further questions about using Cypress SPI devices, please contact Cypress via the “Support” link on our web site www.cypress.com.

Document History Page

Document Title: AN98546 - Designing with the Cypress SPI PDD Document Number: 001-98546				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	–	GJSW	04/06/2010	Initial version
*A	–	GJSW	02/27/2012	**Customizing** Added directory tree Updated table **Registry Entries** Updated section/ Updated SPI PDD Registry Entries table Added Read Command Valid Registry Entry Combinations table **Example Parameters** Updated section Added Memory Map figure
*B	4928242	MSWI	09/21/2015	Updated to Cypress template.
*C	5025804	GJSW	11/24/2015	Updated Introduction on page 1: Updated Figure 1. Updated Customizing on page 1: Updated description.
*D	5844526	AESATMP8	08/04/2017	Updated Cypress Logo and Copyright.
*E	6599084	PRIT	06/19/2019	Obsolete document.

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 | Video | Blogs | Training | Components

Technical Support

cypress.com/support

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