TEXAS INSTRUMENTS

# DSP/BIOS PAL SYS PCI Device Driver

# User's Manual

# IMPORTANT NOTICE

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| Products | | Applications | |
|---|---|---|---|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address:
Texas Instruments
Post Office Box 655303, Dallas, Texas 75265

# Preface

# Read This First

### *About This Manual*

The API reference guide serves as a software programmer's handbook for working with the PCI device driver modules. This reference guide provides necessary information regarding how to use these modules in user systems and applications.

## Abbreviations

*Table of Abbreviations*

| Abbreviation | Description |
|---|---|
| API | Application Programming Interface |
| ISR | Interrupt Service Routine |
| OS | Operating System |
| SOC | System On Chip |
| PCI | Peripheral component Interconnect |

## Revision History

| Date | Author | Comments | Version |
|---|---|---|---|
| January 20, 2007 | Rinkal Shah | Initial draft | 1.0 |
| March 23, 2007 | Rinkal Shah | Document modified according to the code changes due to new TRM of PCI controller | 1.1 |
| June 4, 2007 | Rinkal shah | SOC C6452 added | 1.2 |
| May 21, 2008 | Chandan Nath | Updated for adding compiler switches in build options | 1.3 |

# TABLE OF CONTENTS

**DSP/BIOS PAL SYS PCI Device Driver** .......... 1
**Preface** .......... 3
**Abbreviations** .......... 3
**Revision History** .......... 4
**TABLE OF CONTENTS** .......... 5
**CHAPTER 1** .......... 6
**INTRODUCTION** .......... 6
    1.1. Introduction .......... 7
    1.2. H/W S/W Support .......... 8
    1.3. Driver Components .......... 9
    1.4. Driver Capabilities .......... 10
    1.5. System Requirements .......... 10
**CHAPTER 2** .......... 11
**INSTALLATION GUIDE** .......... 11
    2.1. Component Folder .......... 12
    2.2. Build .......... 13
    2.3. Build Options .......... 13
    2.4. Linux application .......... 14
**CHAPTER 3** .......... 15
**PAL/SYS PCI** .......... 15
    3.1. Functions .......... 16
    3.1.1 PAL_sysPCICreate ( ) - Create the instance of PCI driver .......... 16
    3.1.2 PAL_sysPCIDelete ( ) - Delete the instance of PCI driver. .......... 16
    3.1.3 PAL_sysPCIOpen ( ) - Open the PCI driver instance .......... 16
    3.1.4 PAL_sysPCIClose ( ) - Close the PCI driver instance. .......... 17
    3.1.5 PAL_sysPCIEnableInterrupt ( ) - Enables PCI interrupt/s. .......... 17
    3.1.6 PAL_sysPCIDisableInterrupt ( ) - Disables PCI interrupt/s. .......... 18
    3.1.6 PAL_sysPCISetMemMapReg ( ) - Set memory mapped register. .......... 18
    3.1.7 PAL_sysPCIGetMemMapReg ( ) - Get memory mapped register. .......... 19
    3.1.8 PAL_sysPCISetHookReg( ) - Set Hook register. .......... 19
    3.1.9 PAL_sysPCIGetHookReg( ) - Get Hook register. .......... 20
    3.1.12 PAL_sysPCIEnableBasePrefetch ( ) - API to Enable BasePrefetch. .......... 20
    3.1.13 PAL_sysPCIDisableBasePrefetch ( ) - API to Disable Base Prefetch. .......... 21
    3.1.14 PAL_sysPCIProgramCacheLineSize ( ) - API to set Cache line size. .......... 21
    3.1.15 PAL_sysPCIProgramLatencyTimer ( ) - API to program Latency timer .......... 21
    3.2. Data Structures .......... 22
    3.3. Enumerations .......... 22
    3.4. Macros .......... 27
    3.5 Dependency of Sample application: .......... 28

# CHAPTER 1

# INTRODUCTION

Topic

1.1. Introduction
1.2. H/W S/W Support
1.3. Driver Components
1.4. Default driver Configuration
1.5. Driver Capabilities
1.6. System Requirements

## 1.1. Introduction

This document is an API reference guide on PCI Device Driver.

## 1.2. H/W S/W Support

This PCI device driver has been developed for the following DSP/BIOS operating system. For more details on the version numbers refer to the release notes in the root of the installation.

## 1.3. Driver Components

The driver is constituted of following sub components:

**PCI PAL API’s** – OS Independent part of PCI driver Core

**System components:**

**PALOS -** BIOS Abstraction

## 1.4. Driver Capabilities

Driver has to be configured by the application. There is no default driver configuration.

## 1.5. System Requirements

Refer system level release notes for tools and BIOS versions.

# CHAPTER 2

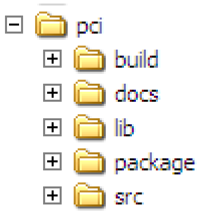
# INSTALLATION GUIDE

**Topic**

2.1. Component Folder
2.2. Build
2.3. Build Options

## 2.1. Component Folder

Upon installing the PCI driver the following directory structure is found in the driver's directory.



```
⊟ pci
    ⊞ build
    ⊞ docs
    ⊞ lib
    ⊞ package
    ⊞ src
```

*Figure 1. PCI Driver Directory Structure*

This top level pci folder contains pci driver psp header file and XDC package files (package.bld, package.xdc and package.xs)

- **build:** This folder contains pci driver library project file. The generated driver library shall be included in the application where PCI driver have to be used.
- **docs:** This folder contains architecture document, datasheet, release notes and user guide.

  Architecture document contains the driver details which can be helpful for the developers as well as consumers to understand the driver design.

  Datasheet gives the idea about the memory consumption by the driver and description of the top level APIs.

  Release Note gives the details about system requirements, steps to Install/Uninstall the package.This document list the known issues of the driver.

  User Guide provides information about how to use the driver. It contains description of sample applications which guide the end user to make their applications using this driver.
- **Lib:** This folder contains libraries generated in all the configuration modes(debug, idebug, irelease and release)
- **Package:** This folder contains files generated by XDC tool.
- **src:** This folder contains pci driver source files. It also contains header files that are used by the driver.

## 2.2. Build

This section describes for each supported target environment, the applicable build options, supported configurations and how selected, the featured capabilities and how enabled, the allowed user customizations for the software to be installed and how the same can be realized.

The component might be delivered to user in different formats:

- ☐ Source-less ie., binary executables and object libraries only
- ☐ Source-inclusive ie., The entire source code used to implement the driver is included in the delivered product
- ☐ Source-selective ie., Only a part of the overall source is included. This delivery mechanism might be required either because; certain parts of the driver require source-level extensions and/or customization at the user's end or because, specific parts of the driver is exposed to user at the source-level to insure user's software development.

When source is included as part of the product delivery, the CCS project file is provided as part of the package. When object format is distributed, the driver header files are part of the "inc" folder and the driver library is provided in /drivers/lib folder.

## 2.3. Build Options

This driver does not have any specific build option at the time of writing of this manual.

The build folder contains a CCS project file that builds the driver into a library for debug and release mode.

Following compiler switches are used to compile for different options.

- ☐ **_DEBUG**
  This is used as a flag to compiler whether to include the debug statements inserted in the code into the final image. This flag helps to build DEBUG image of the program. For RELEASE images this is not passed to the compiler.
- ☐ **CHIP_XXXX**
  The CSL layer is written in a common file for all the variants of a SOC. This flag differentiates the variant we are compiling for, for e.g. - CHIP_DM648, and the CSL definitions for that variant appropriately gets defined for register base addresses, num of ports of a peripheral etc.
- ☐ **PCI_ECM_HOOK_ENABLE**
  This option is used when Ecm Hook is enabled.
- ☐ **PCI_INSTRUMENTATION_ENABLED**
  This option is passed to the compiler to include the instrumentation code parts into the final image/lib of the program. This helps build the iRelease/iDebug versions of the image/lib with a common code base

## 2.4. Linux application

The Linux application located in linuxapp folder contains Linux driver for PCI device. When the driver is inserted, driver searches for the devices attached to the PCI bus, read the BAR address, enables the PCI device and allocates memory in linux kernel for providing read/write space for. Linux driver also makes EVM the master of the bus. Driver enables PCI interrupts a does the interrupt hooking with the operating system.

Linux driver searches for EVM on the PCI bus by reading the vendor ID and device ID of all the devices attached to the bus.

Once the device (EVM) is found on the bus, linux driver reads the BAR addresses of EVM and maps the memory spaces to linux memory space by calling Linux specific functions for mapping the memory.

After configuration of BAR address and memory mapping, linux driver configures as the bus master. Now DM648/C6452 can perform read/write operation on the bus. Before initiating read and write transfers, memory is allocated by the driver and virtual to physical memory mapping is done to make the allocated memory available to EVM through PCI bus. Now, linux driver access the EDMA configuration space of EVM to perform a read and write operation on mapped memory of host PC. These read and write operations are done using EDMA on. EDMA is configured by the linux driver.

The memory that is mapped in the above step is used by respective EVM DSP/BIOS driver to perform read/write operation.

## CHAPTER 3

# PAL/SYS PCI

This chapter describes the functions, data structures, enumerations and macros for the List module.

**Topic**

3.1. Functions

3.2. Data Structures

3.3. Enumeration

3.4. Macros

## 3.1. Functions

This section lists the functions available in the PSP module.

### *3.1.1 PAL_sysPCICreate ( ) - Create the instance of PCI driver.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCICreate** | **(** | | |
| | | | **Uint32** | **instId** |
| | | | **Ptr** | **param** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| instId | Instance Id of PCI controller |
| param | Pointer parameters required for creating PCI driver handle |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
PAL_sysPCICreate creates an instance of PCI driver. This function checks for the maximum instance number. If instance number is proper, it changes the state of driver. This function is implemented just to make PCI driver consistent with other PAL SYS drivers.

### *3.1.2 PAL_sysPCIDelete ( ) - Delete the instance of PCI driver.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIDelete** | **(** | | |
| | | | **Uint32** | **instId** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| instId | Instance Id of PCI controller |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
PAL_sysPCIDelete deletes an instance of PCI driver. This function checks for the maximum instance number. If instance number is proper, it changes the state of driver. This function is implemented just to make PCI driver consistent with other PAL SYS drivers.

### *3.1.3 PAL_sysPCIOpen ( ) - Open the PCI driver instance.*

| | | | | |
|---|---|---|---|---|
| **PSP_Handle** | **PAL_sysPCIOpen** | **(** | | |
| | | | **Uint32** | **instId** |
| | | | **PAL_sysPciInitConfig** | ***param** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| instId | Instance Id of PCI controller |
| param | Pointer parameters required for opening PCI driver handle. Application have to fill |

this param structure with proper values for initialization of driver

**Returns:**
Returns PSP_Handle on success or NULL on failure

**Description:**
This function opens the driver instance. If the handle is proper, it initializes the base address of PCI controller in the driver handle. This function also registers the interrupt callback for PCI interrupt. This callback will receive the interrupt status as one argument and application data pointer as other argument on occurrence of interrupt. Interrupt status is provided to application in application callback.
“param” should be a pointer to PAL_sysPCIinitConfig structure which should contain non-NULL value of application callback if application wants a callback on occurrence of interrupt. Application can also pass parameter to ISR. “param” contains one field as “appData”. “appData” should be pointing to the data that application wants to pass to callback function.

### *3.1.4 PAL_sysPCIClose ( ) - Close the PCI driver instance.*

```
PAL_Result          PAL_sysPCIClose     (
                                           PSP_Handle          hPci
                                           Ptr                 param
                                        )
```

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| param | Pointer parameters required for closing PCI driver handle if any. |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function closes the PCI driver instance. It checks the instance number and state of driver. If state and instance number are proper, it removes the hooking of ISR with interrupt. The function also make pointer to base addresses NULL. After doing all this operations, this function changes the state of driver instance to CLOSED state.

### *3.1.5 PAL_sysPCIEnableInterrupt ( ) - Enables  PCI interrupt/s.*

```
PAL_Result          PAL_sysPCIEnableInterrupt   (
                                                   PSP_Handle              hPci
                                                   Uint32        intEnableCode
                                                )
```

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| intEnableCode | Interrupt enable mask. 5 LSBs of intEnableCode will correspond to 5 interrupts of PCI. If a particular bit is set, corresponding interrupt would be enabled. |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**

This function enables interrupt/s. Interrupts are enabled depending upon the mask that is passed as argument to this function. Each bit of “intEnableCode” signifies one interrupt. Particular bit have to be set by the application to enable corresponding interrupt. Multiple interrupts can be set by setting multiple bits in the “intEnableCode”.

### *3.1.6 PAL_sysPCIDisableInterrupt ( ) - Disables PCI interrupt/s.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIDisableInterrupt** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **Uint32** | **intDisableCode** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| intDisableCode | Interrupt disable mask. 5 LSBs of intDisableCode will correspond to 5 interrupts of PCI. If a particular bit is set, corresponding interrupt would be disabled. |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function disables interrupt/s. Interrupts are disabled depending upon the mask that is passed as argument to this function. Each bit of “intDisableCode” signifies one interrupt. Particular bit have to be set by the application to disable corresponding interrupt. Multiple interrupts can be disabled by setting multiple bits in the “intDisableCode”.

### *3.1.6 PAL_sysPCISetMemMapReg ( ) - Set memory mapped register.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCISetMemMapReg** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **PAL_sysPciMemMapField** | **memMapField** |
| | | | **PAL_sysPciConfigFieldSize** | **writeSize** |
| | | | **Uint32** | **newFieldVal** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| memMapField | Field of memory mapped register that has to be set |
| writeSize | Size of data to be set (byte/word/dword) |
| newFieldVal | New value of field that is to be set |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function sets values in memory mapped registers. Register whose value has to be set should be passed as “memMapField” argument to PAL_sysPCISetMemMapReg function. The values of “memMapField” are restricted to the values available in “PAL_sysPciMemMapField” enum. “writeSize” specifies the number of bytes to written on to the field. Number of bytes can be

1, 2 or 4. The value that has to be written to the field shall to be specified as “newFieldVal” argument.

### 3.1.7 PAL_sysPCIGetMemMapReg ( ) - Get memory mapped register.

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIGetMemMapReg** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **PAL_sysPciMemMapField** | **memMapField** |
| | | | **PAL_sysPciConfigFieldSize** | **readSize** |
| | | | **Uint32** | ***newFieldVal** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| memMapField | Field of memory mapped register that has to be read |
| readSize | Size of data to be read (byte/word/dword)) |
| *newFieldVal | Pointer to new value of field that is to be read from hardware |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function reads values in memory mapped registers. Register whose value has to be read should be passed as “memMapField” argument to PAL_sysPCISetMemMapReg function. The values of “memMapField” are restricted to the values available in “PAL_sysPciMemMapField” enum. “readSize” specifies the number of bytes to read from the specified field. Number of bytes can be 1, 2 or 4. The value that has to be read from the field will be store at 4 byte memory pointer by “newFieldVal” pointer.

### 3.1.8 PAL_sysPCISetHookReg( ) - Set Hook register.

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCISetHookReg** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **PAL_sysPciHookRegister** | **hookRegField** |
| | | | **PAL_sysPciConfigFieldSize** | **writeSize** |
| | | | **Uint32** | **newFieldVal** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| hookRegField | Field of hook register that has to be set |
| writeSize | Size of data to be set (byte/word/dword) |
| newFieldVal | New value of field that is to be set |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**

This function sets values in hook registers. Register whose value has to be set should be passed as “hookRegField” argument to PAL_sysPCISetHookReg function. The values of “hookRegField” are restricted to the values available in “PAL_sysPciHookRegister” enum. “writeSize” specifies the number of bytes to written on to the field. Number of bytes can be 1, 2 or 4. The value that has to be written to the field shall be specified as “newFieldVal” argument.

### *3.1.9 PAL_sysPCIGetHookReg( ) - Get Hook register.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIGetHookReg** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **PAL_sysPciHookRegister** | **hookRegField** |
| | | | **PAL_sysPciConfigFieldSize** | **readSize** |
| | | | **Uint32** | ***newFieldVal** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| hookRegField | Field of hook register that has to be set |
| readSize | Size of data to be read (byte/word/dword) |
| *newFieldVal | Pointer to new value of field that is to be read from hardware |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function reads values from hook registers. Register whose value has to be read should be passed as “hookRegField” argument to PAL_sysPCISetHookReg function. The values of “hookRegField” are restricted to the values available in “PAL_sysPciHookRegister” enum. “readSize” specifies the number of bytes to read from the field. Number of bytes can be 1, 2 or 4. The value that is read from the field will be stored at the 4 byte memory location pointed by “newFieldVal” pointer.

### *3.1.12 PAL_sysPCIEnableBasePrefetch ( ) - API to Enable BasePrefetch.*

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIEnableBasePrefetch** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **Uint32** | **baseId** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| baseId | ID of BASE for which prefetch have to be enabled |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**

This function will enable pre-fetchable functionality of BAR memory space. “baseId” will specify the ID of BAR whose pre-fetchable functionality have to be enabled. The value of “baseId” can vary from 0 to 5.

### 3.1.13 PAL_sysPCIDisableBasePrefetch ( ) - API to Disable Base Prefetch.

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIDisableBasePrefetch** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **Uint32** | **baseId** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| baseId | ID of BASE for which pre-fetch have to be disabled |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function will disable pre-fetchable functionality of BAR memory space. “baseId” will specify the ID of BAR whose pre-fetchable functionality have to be disabled. The value of “baseId” can vary from 0 to 5.

### 3.1.14 PAL_sysPCIProgramCacheLineSize ( ) - API to set Cache line size.

| | | | | |
|---|---|---|---|---|
| **PAL_Result** | **PAL_sysPCIProgramCacheLineSize** | **(** | | |
| | | | **PSP_Handle** | **hPci** |
| | | | **PAL_sysPciCacheLineSize** | **valCacheLineSize** |
| | | **)** | | |

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| valCacheLineSize | Size of cache line to be set |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function will program the length of cache line. “valCacheLineSize” will specify the new size of cache line. The values of “valCacheLineSize” can vary from 16 bytes, 32 bytes, 64 bytes, 128 bytes or cache can be disabled. The values of “valCacheLineSize” are restricted by “PAL_sysPciCacheLineSize” enum.

### 3.1.15 PAL_sysPCIProgramLatencyTimer ( ) - API to program Latency timer.

| | | |
|---|---|---|
| **PAL_Result** | **PAL_sysPCIProgramLatencyTimer** | **(** |

```
                                        PSP_Handle                   hPci
                                        Uint32              valLatencyTi
                                                                     mer
                                      )
```

**Parameters:**

| | |
|---|---|
| hPci | Handle of PCI controller driver |
| valLatencyTimer | Value of latency timer |

**Returns:**
Returns PAL_SOK on success or PAL Error Code on failure

**Description:**
This function will set the value of latency timer. The new value of latency timer has to be specified as “valLatencyTimer” argument.

## 3.2. Data Structures

This section lists the data structures available in the PCI module.

```
 typedef struct _PAL_sysPciInitConfig
{
          PAL_sysPciAppCallback appCb;
          Ptr appData;
} PAL_sysPciInitConfig;
```

**Data Fields**

| | |
|---|---|
| PAL_sysPciAppCallback | appCb |
| Ptr | appData |

**Detailed Description:**
PCI driver configuration.

**Field Documentation**

PAL_sysPciAppCallback appCb
Instance wide callback function to catch errors.

Ptr appData
Application data to be passed back to the app callback

## 3.3. Enumerations

This section lists the enumerations available in the PSP module.

**enum PAL_sysPciConfigFieldSize _**
**Enumeration values:**

| | |
|---|---|
| *PAL_SYS_PCI_READ_WRITE_SIZE_BYTE* | Read/write one byte(8-bit) |
| *PAL_SYS_PCI_READ_WRITE_SIZE_HALFWORD* | Read/write half word(16-bit) |
| *PAL_SYS_PCI_READ_WRITE_SIZE_WORD* | Read/write word(32-bit) |

**enum PAL_sysPciMemMapField**

**Enumeration values:**

| | |
|---|---|
| PAL_SYSPCI_PCIIF_MINOR_REV | PCIIF minor revision |
| PAL_SYSPCI_PCIIF_MAJOR_REV | PCIIF major revision |
| PAL_SYSPCI_PCIIF_MODULE_ID | PCIIF module ID |
| PAL_SYSPCI_STATUS_SET | status set register |
| PAL_SYSPCI_STATUS_CLEAR | status clear register |
| PAL_SYSPCI_HOST_INT_ENABLE_SET | Host interrupt enable set register |
| PAL_SYSPCI_HOST_INT_ENABLE_CLEAR | Host interrupt enable clear register |
| PAL_SYSPCI_DSP_INT_ENABLE_SET | DSP interrupt enable set register |
| PAL_SYSPCI_DSP_INT_ENABLE_CLEAR | DSP interrupt enable clear register |
| PAL_SYSPCI_MIRROR_VENDOR_ID | Vendor ID mirror register |
| PAL_SYSPCI_MIRROR_DEVICE_ID | Device ID mirror register |
| PAL_SYSPCI_MIRROR_CMD_STATUS_REG | Command mirror register |
| PAL_SYSPCI_MIRROR_REVISION_ID | Revision ID mirror register |
| PAL_SYSPCI_MIRROR_CLASS_CODE | Class code mirror register |
| PAL_SYSPCI_MIRROR_CACHE_SIZE | Cache size mirror register |
| PAL_SYSPCI_MIRROR_LATENCY_TIMER | Latency timer mirror register |
| PAL_SYSPCI_MIRROR_HEADER_TYPE | Header type mirror register |
| PAL_SYSPCI_MIRROR_BIST_REG | BIST mirror register |
| PAL_SYSPCI_MASK_BAR_0 | BAR 0 mask register |
| PAL_SYSPCI_MASK_BAR_1 | BAR 1 mask register |
| PAL_SYSPCI_MASK_BAR_2 | BAR 2 mask register |
| PAL_SYSPCI_MASK_BAR_3 | BAR 3 mask register |
| PAL_SYSPCI_MASK_BAR_4 | BAR 4 mask register |
| PAL_SYSPCI_MASK_BAR_5 | BAR 5 mask register |
| PAL_SYSPCI_MIRROR_SUBSYS_VENDOR_ID | Subsystem vendor ID mirror register |
| PAL_SYSPCI_MIRROR_SUBSYS_ID | Subsystem ID mirror register |
| PAL_SYSPCI_MIRROR_CAP_PTR_ID | Capabilities pointer mirror register |
| PAL_SYSPCI_MIRROR_INT_LINE | Interrupt line mirror register |
| PAL_SYSPCI_MIRROR_INT_PIN | Interrupt Pin mirror register |
| PAL_SYSPCI_MIRROR_MIN_GRANT_BITS | Minimum grant bits mirror register |
| PAL_SYSPCI_MIRROR_MAX_LATENCY_BITS | Maximum latency bits mirror register |
| PAL_SYSPCI_SLAVE_CNTL_REG | Slave control register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_0 | Slave base address 0 translation register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_1 | Slave base address 1 translation register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_2 | Slave base address 2 translation register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_3 | Slave base address 3 translation register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_4 | Slave base address 4 translation register |
| PAL_SYSPCI_SLAVE_BASE_ADDR_TRANS_REG_5 | Slave base address 5 translation register |
| PAL_SYSPCI_MIRROR_BAR_0 | Base address 0 mirror register |
| PAL_SYSPCI_MIRROR_BAR_1 | Base address 1 mirror register |
| PAL_SYSPCI_MIRROR_BAR_2 | Base address 2 mirror register |
| PAL_SYSPCI_MIRROR_BAR_3 | Base address 3 mirror register |
| PAL_SYSPCI_MIRROR_BAR_4 | Base address 4 mirror register |
| PAL_SYSPCI_MIRROR_BAR_5 | Base address 5 mirror register |
| PAL_SYSPCI_MASTER_CONFIG_IO_DATA_REG | Master configuration/IO access data register |

| | |
|---|---|
| PAL_SYSPCI_MASTER_CONFIG_IO_ADDR_REG | Master configuration/IO access address register |
| PAL_SYSPCI_MASTER_CONFIG_IO_CMD_REG | Master configuration/IO access command register |
| PAL_SYSPCI_MASTER_CONFIG_REG | Master configuration register |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG0 | PCI address substitution register 0 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG1 | PCI address substitution register 1 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG2 | PCI address substitution register 2 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG3 | PCI address substitution register 3 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG4 | PCI address substitution register 4 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG5 | PCI address substitution register 5 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG6 | PCI address substitution register 6 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG7 | PCI address substitution register 7 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG8 | PCI address substitution register 8 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG9 | PCI address substitution register 9 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG10 | PCI address substitution register 10 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG11 | PCI address substitution register 11 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG12 | PCI address substitution register 12 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG13 | PCI address substitution register 13 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG14 | PCI address substitution register 14 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG15 | PCI address substitution register 15 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG16 | PCI address substitution register 16 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG17 | PCI address substitution register 17 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG18 | PCI address substitution register 18 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG19 | PCI address substitution register 19 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG20 | PCI address substitution register 20 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG21 | PCI address substitution register 21 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG22 | PCI address substitution register 22 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG23 | PCI address substitution register 23 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG24 | PCI address substitution register 24 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG25 | PCI address substitution register 25 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG26 | PCI address substitution register 26 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG27 | PCI address substitution register 27 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG28 | PCI address substitution register 28 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG29 | PCI address substitution register 29 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG30 | PCI address substitution register 30 |
| PAL_SYSPCI_ADDR_SUBSTITUTION_REG31 | PCI address substitution register 31 |

**enum PAL_sysPciHookRegister**
**Enumeration values:**

| | |
|---|---|
| PAL_SYSPCI_HOOK_REG_PROGRAM_VENDOR_ID | Vendor ID program register |
| PAL_SYSPCI_HOOK_REG_PROGRAM_DEVICE_ID | Device ID program register |
| PAL_SYSPCI_HOOK_REG_PROGRAM_CLASS_CODE_REVISION_ID | Class code and revision ID program register |
| PAL_SYSPCI_HOOK_REG_PROGRAM_SUB_SYS_VENDOR_ID_SUB_SYS_ID | Register to program vendor ID |

| | |
|---|---|
| PAL_SYSPCI_HOOK_REG_PROGRAM_MAX_LATENCY_MIN_GRANT | Max latency and min grant program register |
| PAL_SYSPCI_HOOK_REG_CONFIG_DONE_REG | Configuration done registers |

**enum PAL_sysPciCacheLineSize**

**Enumeration values:**

| | |
|---|---|
| *PAL_SYSPCI_CACHELINE_SIZE_DISABLED* | Cache line size - Disabled |
| *PAL_SYSPCI_CACHELINE_SIZE_16BYTES* | Cache line size - 16 bytes |
| *PAL_SYSPCI_CACHELINE_SIZE_32BYTES* | Cache line size - 32 bytes |
| *PAL_SYSPCI_CACHELINE_SIZE_64BYTES* | Cache line size – 64 bytes |
| *PAL_SYSPCI_CACHELINE_SIZE_128BYTES* | Cache line size - 128 bytes |

## 3.4. Macros

This section lists the macros available in the PSP module.

```
/** Maximum PCI driver instance supported */
#define PAL_SYSPCI_MAX_PCI_INSTANCE     (1u)
/** Maximum number of slaves supported by PCI */
#define PAL_SYSPCI_MAX_NUM_SLAVES       (4u)
/** Maximum base address registers supported by PCI controller */
#define PAL_SYSPCI_MAX_BASE_ADDR        (6u)
/** Interrupt mask to enable/disable Target abort interrupt */
#define PAL_SYSPCI_TARGET_ABORT_INT_MASK              (0x00000002u)
/** Interrupt mask to enable/disable Master abort interrupt */
#define PAL_SYSPCI_MASTER_ABORT_INT_MASK              (0x00000004u)
/** Interrupt mask to enable/disable system error interrupt */
#define PAL_SYSPCI_SYS_ERR_INT_MASK                   (0x00000020u)
/** Interrupt mask to enable/disable parity error interrupt */
#define PAL_SYSPCI_PARITY_ERR_INT_MASK                (0x00000040u)
```

Error Codes returned by the PAL SYS PCI driver

```
/** PCI driver error base define */
#define PAL_SYS_PCI_ERR_BASE                         (-40)
/**
 * PCI Driver Object Not Deleted yet.
 * So the object cannot be created.
 */
#define PAL_SYS_PCI_OBJ_NOT_DELETED        (PAL_SYS_PCI_ERR_BASE)
/**
 * PCI Driver Object Not Created yet.
 * So the object cannot be deleted.
 */
#define PAL_SYS_PCI_OBJ_NOT_CREATED        (PAL_SYS_PCI_ERR_BASE-1)
/**
 * PCI Driver Object Not Closed yet.
 * So the object cannot be deleted.
 */
#define PAL_SYS_PCI_OBJ_NOT_CLOSED         (PAL_SYS_PCI_ERR_BASE-2)
/**
 * PCI Driver Not Opened yet
 * So the object cannot be closed.
 */
#define PAL_SYS_PCI_OBJ_NOT_OPENED         (PAL_SYS_PCI_ERR_BASE-3)
/**
 * Invalid Parameter passed to API
 */
#define PAL_SYS_PCI_INVALID_PARAM          (PAL_SYS_PCI_ERR_BASE-4)
/**
 * Error encountered in PCI driver while semaphore operation
 */
#define PAL_SYS_PCI_SEM_ERR                (PAL_SYS_PCI_ERR_BASE-5)
```

Codes to enable/disable PCI interrupts

```
/**
 * Code to enable/disable Target abort interrupt of Host
 */
#define PAL_SYS_PCI_HOST_INT_TGT_ABORT                                  (0x01u)
/**
 * Code to enable/disable Master abort interrupt of Host
 */
#define PAL_SYS_PCI_HOST_INT_MS_ABORT                                   (0x02u)
/**
 * Code to enable/disable system error detect interrupt of Host
 */
#define PAL_SYS_PCI_HOST_INT_SYS_ERR_DETECT                             (0x04u)
/**
 * Code to enable/disable parity error detect interrupt of Host
 */
#define PAL_SYS_PCI_HOST_INT_PARITY_ERR_DETECT                          (0x08u)


/**
 * Code to enable/disable target abort error interrupt of DSP
 */
#define PAL_SYS_PCI_DSP_INT_TGT_ABORT                                   (0x10u)
/**
 * Code to enable/disable master abort error interrupt of DSP
 */
#define PAL_SYS_PCI_DSP_INT_MS_ABORT                                    (0x20u)
/**
 * Code to enable/disable system error detect interrupt of DSP
 */
#define PAL_SYS_PCI_DSP_INT_SYS_ERR_DETECT                              (0x40u)
/**
 * Code to enable/disable parity error detect interrupt of DSP
 */
#define PAL_SYS_PCI_DSP_INT_PARITY_ERR_DETECT                           (0x80u)
```

## 3.5 Dependency of Sample application:

Following Components needs to be linked for successful build and functionality of the application.

- PCI
- PAL_OS
- SoC specific PAL_SYS
- EDMA3