TEXAS INSTRUMENTS

PCI2040

Implementation Guide

1999

Mixed Signal Products

TEXAS INSTRUMENTS

Printed in U.S.A.
10/99

SCPU004

PCI2040
Implementation Guide

Literature Number: SCPU004
October 1999

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Preface

Read This First

About This Manual

This manual addresses a number of varied and important, but easily overlooked items which should be of interest to the developer implementing an application that uses the PCI2040 PCI–DSP bridge controller.

How to Use This Manual

This document contains the following chapters:

Chapter 1, *Introduction*, provides basic information on features and functions of the PCI2040.

Chapter 2, *PCI2040 TMS320C54x System Implementation*, illustrates a platform using the PCI2040 and four TMS320C54x DSPs.

Chapter 3, *PCI2040 TMS320C6x System Implementation*, illustrates a platform using the PCI2040 and four TMS320C54x DSPs.

Chapter 4, *PCI2040 JTAG TBC System Implementation*, illustrates a platform using the PCI2040 and a JTAG TBC.

Chapter 5, *PCI2040 Pinout*, lists the signal on each terminal of the PCI2040.

Chapter 6, Implementation Considerations, contains card-level hardware considerations for designing with the PCI2040.

Chapter 7, Software Considerations, contains several software-related items of particular interest to those working with the PCI2040.

Chapter 8, Other Considerations, covers some miscellaneous topics likely to be of interest in PCI2040 applications.

Notational Conventions

This document uses the following conventions.

- Program listings, program examples, and interactive displays are shown in a `special typeface` similar to a typewriter's. Examples use a **`bold`**

`version` of the special typeface for emphasis; interactive displays use a **`bold version`** of the special typeface to distinguish commands that you enter from items that the system displays (such as prompts, command output, error messages, etc.).

Here is a sample program listing:

```
0011  0005  0001          .field    1, 2
0012  0005  0003          .field    3, 4
0013  0005  0006          .field    6, 3
0014  0006                .even
```

Here is an example of a system prompt and a command that you might enter:

```
C:  csr –a /user/ti/simuboard/utilities
```

- In syntax descriptions, the instruction, command, or directive is in a **bold typeface** font and parameters are in an *italic typeface*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a directive syntax:

  .asect "*section name***",** *address*

  .asect is the directive. This directive has two parameters, indicated by *section name* and *address*. When you use .asect, the first parameter must be an actual section name, enclosed in double quotes; the second parameter must be an address.

- Square brackets (**[** and **]**) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you don't enter the brackets themselves. Here's an example of an instruction that has an optional parameter:

  LALK *16–bit constant [, shift]*

  The LALK instruction has two parameters. The first parameter, *16-bit constant*, is required. The second parameter, *shift*, is optional. As this syntax shows, if you use the optional second parameter, you must precede it with a comma.

  Square brackets are also used as part of the pathname specification for VMS pathnames; in this case, the brackets are actually part of the pathname (they are not optional).

- Braces ({ and }) indicate a list. The symbol **|** (read as *or*) separates items within the list. Here's an example of a list:

  ```
  { * | *+ | *– }
  ```

  This provides three choices: *, *+, or *–.

  Unless the list is enclosed in square brackets, you must choose one item from the list.

- Some directives can have a varying number of parameters. For example, the .byte directive can have up to 100 parameters. The syntax for this directive is:

.byte $value_1$ [, ... , $value_n$]

This syntax shows that .byte must have at least one value parameter, but you have the option of supplying additional value parameters, separated by commas.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

Related Documentation From Texas Instruments

PCI2040 DSP–PCI Bridge Controller Data Manual

PCI2040 EVM Hardware Guide

PCI2040 EVM Software Guide

Contents

Figures

Tables

Chapter 1

Introduction

This document is provided to assist platform developers using the PCI2040. Schematics of a sample design using the PCI2040 on an add-in card illustrate many design considerations necessary for proper implementation of the device. This design is available as an evaluation module.

Topic **Page**

1.1 PCI2040 Features and Function

The Texas Instruments PCI2040 PCI-to-DSP bridge can connect up to four TMS320C54x or TMS320C6x DSP's to a PCI bus. The PCI2040 provides a glueless interface that is compliant with PCI Local Bus Specification Revision 2.2, PCI Power Management Interface Specification Revision 1.1 and CPCI Hot Swap PICMG 2.1 Revision 1. The PCI2040 is a slave-only interface that provides access to each DSP through its host port interface (HPI). The PCI2040 connects directly to both 5-V and 3.3-V PCI buses with a maximum 32-bit PCI bus frequency of 33 MHz.

The PCI2040 supports only PCI bus target transactions. PCI bus slave transactions enable any PCI device to transfer data to and from the DSP's memory space.

The PCI2040 interfaces directly with the PCI bus and the HPI interface on the DSP. PCI transactions are converted internally to HPI transactions. A serial EEPROM can be used to load configuration information when the system boots. The PCI2040 general-purpose (GP) bus interface can be used to interface to one 16-bit device. The GP bus was designed to be a glueless interface to a JTAG test bus controller (TBC).

An advanced CMOS process achieves low system power consumption while operating at PCI clock rates up to 33 MHz. Several low-power modes allow the host power management system to further reduce power consumption.

1.2 PCI2040 Feature Set

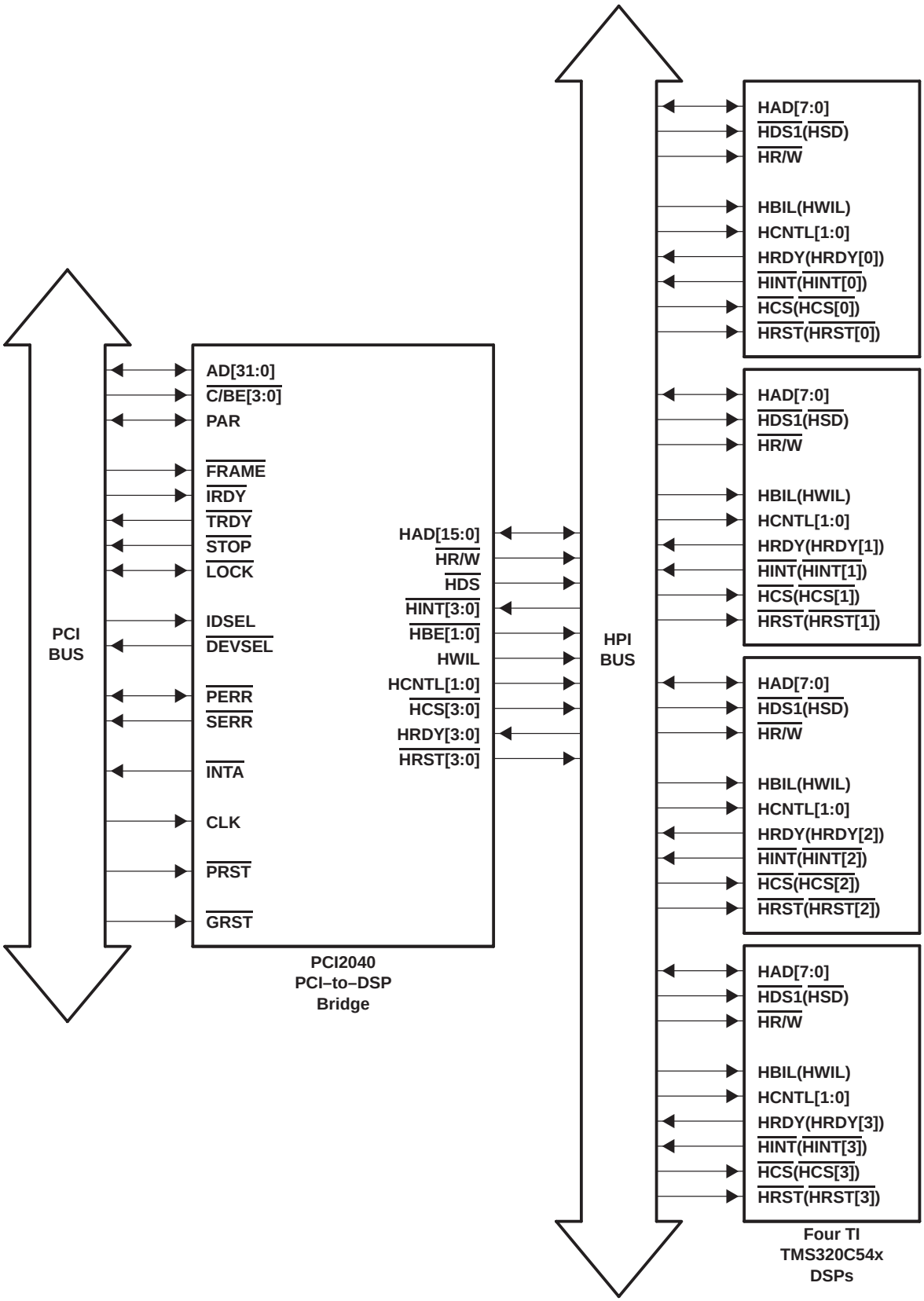
- 3.3-V Core Logic With Universal PCI Interface
- PCI Local Bus Specification Revision 2.2 Compliant
- PCI Power Management Specification Rev 1.1 Compliant
- Advanced Configuration and Power Interface Specification Revision 1.0 Compliant
- CPCI Hot Swap PICMG 2.1 R 1.0 Compliant
- Glueless Interface to HPI Port of TMS320C54x and TMS320C6x
- Provides a Serial EEPROM Interface for Loading Configuration Registers

Chapter 2

PCI2040 TMS320C54x System Implementation

Figure 2–1 illustrates a platform using the PCI2040 and four TMS320C54x DSPs.

Figure 2–1. Typical TMS320C54x DSP Implementation

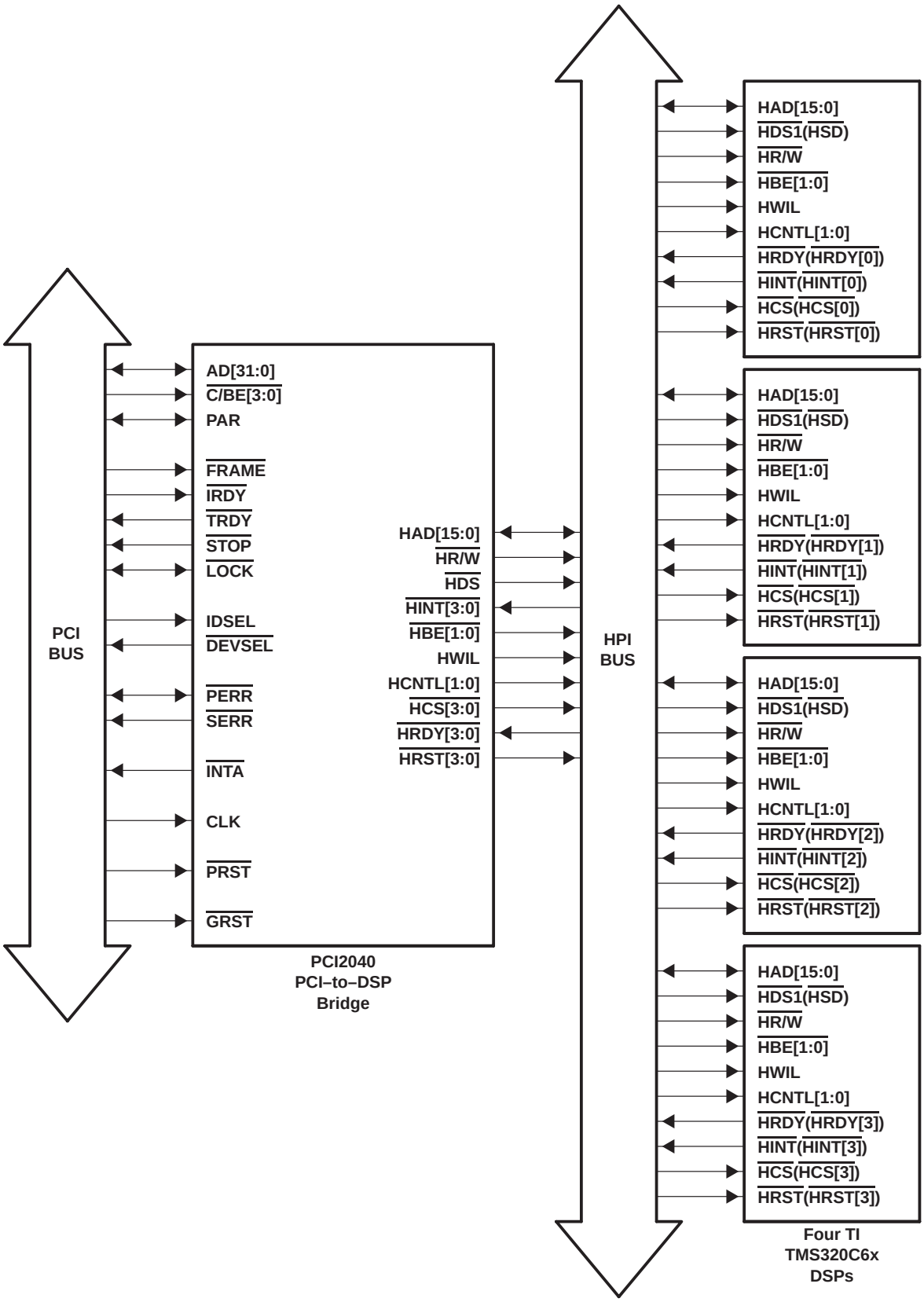


Chapter 3

PCI2040 TMS320C6x System Implementation

Figure 3–1 illustrates a platform using the PCI2040 and four TMS320C6x DSPs.

Figure 3–1. Typical TMS320C6x DSP Implementation

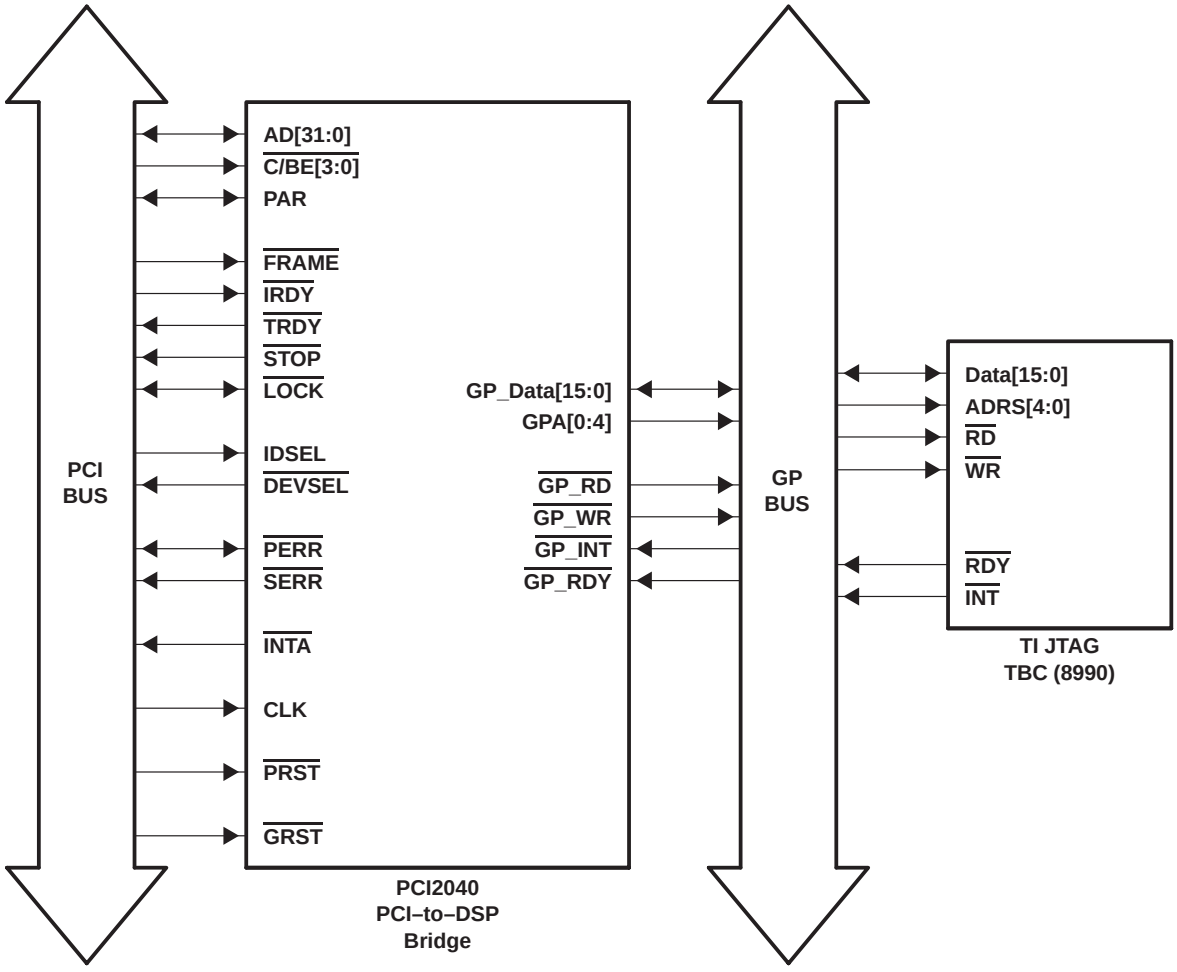


Chapter 4

PCI2040 JTAG TBC System Implementation

Figure 4–1 illustrates a platform using the PCI2040 and a JTAG TBC.

Figure 4–1. Typical GP Bus Implementation



Note: The TI JTAG TBC (SN74ACT8990) is a 5-V device; therefore, it drives 5-V CMOS levels (V_{OH} is minimum of 4.4 V). When interfacing this device to a DSP, voltage translation logic is required. Texas Instruments recommends using the SN74CBT3384 to perform this translation.

Chapter 5

PCI2040 Pinout

Table 5–1 lists the PCI2040 pinout. It contains both LQFP and Micro-star BGA pinouts of the PCI2040.

Table 5–1. Terminal Assignments

PGE	GGU	Terminal Name	PGE	GGU	Terminal Name
1	A01	PCI_AD31	37	N01	$\overline{\text{PCI\_PERR}}$
2	B01	PCI_AD30	38	N02	$\overline{\text{PCI\_SERR}}$
3	C02	PCI_AD29	39	M03	PCI_PAR
4	C01	PCI_AD28	40	N03	VCC
5	D04	VCC	41	K04	$\overline{\text{PCI\_LOCK}}$
6	D03	PCI_AD27	42	L04	$\overline{\text{PCI\_C/BE1}}$
7	D02	PCI_AD26	43	M04	GND
8	D01	PCI_AD25	44	N04	PCI_AD15
9	E04	PCI_AD24	45	K05	PCI_AD14
10	E03	GND	46	L05	PCI_AD13
11	E02	$\overline{\text{PCI\_C/BE3}}$	47	M05	PCI_AD12
12	E01	PCI_IDSEL	48	N05	PCI_AD11
13	F04	VCC	49	K06	VCC
14	F03	PCI_AD23	50	L06	$\overline{\text{PCI\_INTA}}$
15	F02	PCI_AD22	51	M06	GND
16	F01	PCI_AD21	52	N06	PCI_AD10
17	G02	PCI_AD20	53	M07	PCI_AD9
18	G01	VCCP	54	N07	PCI_AD8
19	G03	$\overline{\text{PCI\_RST}}$	55	L07	VCCP
20	G04	$\overline{\text{G\_RST}}$	56	K07	$\overline{\text{PCI\_C/BE0}}$
21	H01	PCI_PCLK	57	N08	VCC
22	H02	GND	58	M08	PCI_AD7
23	H03	PCI_AD19	59	L08	PCI_AD6
24	H04	PCI_AD18	60	K08	PCI_AD5
25	J01	PCI_AD17	61	N09	PCI_AD4
26	J02	PCI_AD16	62	M09	PCI_AD3
27	J03	VCC	63	L09	GND
28	J04	$\overline{\text{PCI\_C/BE2}}$	64	K09	PCI_AD2
29	K01	$\overline{\text{PCI\_FRAME}}$	65	N10	PCI_AD1
30	K02	$\overline{\text{PCI\_IRDY}}$	66	M10	PCI_AD0
31	K03	GND	67	L10	VCC
32	L01	$\overline{\text{PCI\_TRDY}}$	68	N11	$\overline{\text{PME}}$
33	L02	$\overline{\text{PCI\_DEVSEL}}$	69	M11	RSVD
34	L03	$\overline{\text{PCI\_STOP}}$	70	L11	$\overline{\text{GP\_RST}}$
35	M01	RSVD	71	N12	$\overline{\text{HSENUM}}$
36	M02	RSVD	72	M12	HSLED

Table 5–1. Terminal Assignments (Continued)

PGE	GGU	Terminal Name	PGE	GGU	Terminal Name
73	N13	HSSWITCH	109	A13	HWII/GPA2
74	M13	$\overline{\text{GP\_INT}}$	110	A12	$\overline{\text{HBE1}}$/GPA1
75	L12	$\overline{\text{GP\_RDY}}$	111	B11	$\overline{\text{HBE0}}$/GPA0
76	L13	$\overline{\text{HINT0}}$	112	A11	HRD/$\overline{\text{GP\_CS}}$
77	K10	GND	113	D10	VCC
78	K11	$\overline{\text{HINT1}}$	114	C10	$\overline{\text{HRST0}}$
79	K12	$\overline{\text{HINT2}}$	115	B10	$\overline{\text{HRST1}}$
80	K13	$\overline{\text{HINT3}}$	116	A10	$\overline{\text{HRST2}}$
81	J10	VCC	117	D09	$\overline{\text{HRST3}}$
82	J11	HAD0/GPD0	118	C09	GND
83	J12	HAD1/GPD1	119	B09	HRDY5x0/$\overline{\text{HRDY6x0}}$
84	J13	HAD2/GPD2	120	A09	HRDY5x1/$\overline{\text{HRDY6x1}}$
85	H10	HAD3/GPD3	121	D08	HRDY5x2/$\overline{\text{HRDY6x2}}$
86	H11	HAD4/GPD4	122	C08	HRDY5x3/$\overline{\text{HRDY6x3}}$
87	H12	GND	123	B08	VCCH
88	H13	HAD5/GPD5	124	A08	GPIO0
89	G12	HAD6/GPD6	125	B07	GPIO1
90	G13	HAD7/GPD7	126	A07	GPIO2
91	G11	VCCH	127	C07	GPIO3
92	G10	HAD8/GPD8	128	D07	VCC
93	F13	HAD9/GPD9	129	A06	GPIO4
94	F12	HAD10/GPD10	130	B06	GPIO5
95	F11	VCC	131	C06	RSVD
96	F10	HAD11/GPD11	132	D06	RSVD
97	E13	HAD12/GPD12	133	A05	RSVD
98	E12	HAD13/GPD13	134	B05	RSVD
99	E11	HAD14/GPD14	135	C05	RSVD
100	E10	HAD15/GPD15	136	D05	RSVD
101	D13	GND	137	A04	RSVD
102	D12	$\overline{\text{HCS0}}$	138	B04	RSVD
103	D11	$\overline{\text{HCS1}}$	139	C04	RSVD
104	C13	$\overline{\text{HCS2}}$	140	A03	RSVD
105	C12	$\overline{\text{HCS3}}$	141	B03	RSVD
106	C11	HR/$\overline{\text{W}}$/GPA5	142	C03	RSVD
107	B13	HCNTL1/GPA4	143	A02	RSVD
108	B12	HCNTL0/GPA3	144	B02	RSVD

Chapter 6

Implementation Considerations

This chapter contains card-level hardware considerations for designing with the PCI2040.

Topic **Page**

6.1 Power and Signaling Levels

The PCI2040 supports both 3.3-V and 5-V signal environments. This is accomplished by the V_{CCH} and V_{CCP} clamping rails. These two rails are not power rails. They only clamp the signals at the rail voltage (3.3 V or 5 V). All I/O buffers are powered by the V_{CC} rail. Because V_{CC} powers both the core and the I/O buffers, it must always be at 3.3 V. Table 6–1 describes the different voltage combinations supported by the PCI2040.

Table 6–1. Supported Signaling Levels

PCI Bus Signaling Level	HPI/GP Bus Signaling Level	V_{CCP}	V_{CCH}	V_{CC}
5	5	5	5	3.3
5	3.3	5	3.3	3.3
3.3	5	3.3	5	3.3
3.3	3.3	3.3	3.3	3.3

Table 6–2 contains the power measurements for the PCI2040. The measurements were taken under the following conditions: 33-MHz PCI bus clock,
V_{CC} = 3.3 V, V_{CCP} = 5 V, and V_{CCH} = 3.3 V.

Table 6–2. PCI2040 Power Measurements

Device State	I_{CC} (mA)	I_{CCP} (μA)	I_{CCH} (μA)
D0 (fully on)	15	10	< 1
D1	10	10	< 1
D2	10	10	< 1
D3 hot	10	10	< 1

6.2 Required Pullup Resistors

All designs using the PCI2040 require pullup resistors for various PCI and HPI signals. In addition, some implementation-specific pullups may be necessary depending upon the specific configuration of the PCI2040. The following paragraphs provide a detailed discussion of the required pullup resistors.

Note:

All pullup/pulldown resistor value recommendations are provided as guidelines only. The best value for an individual design may vary depending upon board characteristics, standard design rules and practices, etc.

6.2.1 PCI Pullup Resistors

The following paragraph summarizes paragraph 4.3.3 of the PCI Local Bus Specification Revision 2.2, and is provided for reference only. Refer to the PCI Local Bus Specification for a full discussion on pullup resistors required for PCI local bus implementations.

All PCI control signals require pullup resistors on the motherboard to ensure they are at a stable state when no agent is actively driving the signal. Pullups should be implemented on the motherboard only; expansion boards or add-in cards should not provide pullup resistors for the PCI control signals. The PCI signals which require pullups are $\overline{\text{FRAME}}$, $\overline{\text{TRDY}}$, $\overline{\text{IRDY}}$, $\overline{\text{DEVSEL}}$, $\overline{\text{STOP}}$, $\overline{\text{SERR}}$, $\overline{\text{PERR}}$, $\overline{\text{LOCK}}$, $\overline{\text{INTA}}$, $\overline{\text{INTB}}$, $\overline{\text{INTC}}$, $\overline{\text{INTD}}$, and when used $\overline{\text{REQ64}}$ and $\overline{\text{ACK64}}$. Pullups are not required on point-to-point or shared 32-bit signals, as bus parking ensures their stability. Table 6–4 provides a list of the 32-bit PCI signals implemented on the PCI2040 which require pullup resistors on the system board.

Minimum and maximum values for required PCI pullup resistors can be calculated using the following formulas:

$$R_{min} = \left[V_{CC(max)} - V_{ol}\right] \Big/ \left[I_{ol} + \left(16 \times I_{il}\right)\right],$$

Where 16 = maximum number of loads, and

$$R_{max} = \left[V_{CC(min)} - V_x\right] \Big/ \left[num\_loads \times I_{ih}\right],$$

Where V_X = 2.7 V for 5 V signaling and V_X = 0.7 V_{CC} for 3.3 V signaling.

Minimum and typical values for 5 V and 3.3 V signaling environments are shown in Table 7–1.

Table 6–3. Minimum and Typical PCI Pullup Resistor Values

Signaling Rail	R_{min}	R_{typ}	R_{max}
5 V	963 Ω	2.7 kΩ ±10%	Dependent upon number of loads, see formula.
3.3 V	2.42 kΩ	8.2 kΩ ±10%	Dependent upon number of loads, see formula.

Table 6–4. 32-Bit PCI Signal System Board Pullup Requirements

PCI Signal	Pullup Voltage
$\overline{\text{FRAME}}$	V_{CCP}
$\overline{\text{TRDY}}$	V_{CCP}
$\overline{\text{IRDY}}$	V_{CCP}
$\overline{\text{DEVSEL}}$	V_{CCP}
$\overline{\text{STOP}}$	V_{CCP}
$\overline{\text{SERR}}$	V_{CCP}
$\overline{\text{PERR}}$	V_{CCP}
$\overline{\text{LOCK}}$	V_{CCP}
$\overline{\text{INTA}}$	V_{CCP}

6.2.2 HPI/GP Bus Pullups

Pullup resistors are required on all unused HPI and GPbus inputs in order to prevent oscillation and increased power consumption. Pullup resistors are not required on HAD[15:0] due to the fact that the PCI2040 drives these signals to stable values during idle times.

Table 6–5. HPI/GP Inputs That May Require Pullups

HPI/GPbus Signal	Pullup Voltage
$\overline{\text{GPINT}}$	V_{CCH}
$\overline{\text{HINT[3:0]}}$	V_{CCH}
HRDY5x[3:0]/$\overline{\text{HRDY6c[3:0]}}$	V_{CCH}

The $\overline{\text{GPRDY}}$ signal is used to notify the PCI2040 whether the device on the GPbus is ready for a transaction. Because this signal is active low, pulling this signal up will always put the GPbus in a not ready state. For this reason, $\overline{\text{GPRDY}}$ needs to be pulled down to GND.

6.2.3 DSP Pullup/Pulldown Requirements

The PCI2040 uses only one data strobe ($\overline{\text{HDS}}$) and does not implement $\overline{\text{HAS}}$. Because of this, the $\overline{\text{HAS}}$ on the DSP needs to be pulled up and one of the two $\overline{\text{HDS}}$ lines on the DSP needs to be pulled up. The other $\overline{\text{HDS}}$ line needs to be connected to the $\overline{\text{HDS}}$ signal on the PCI2040.

For information about DSP pullups refer to the implementation guide for the DSP being used.

6.2.4 Miscellaneous Pullup/Pulldown Requirements

6.2.4.1 $\overline{\text{PME}}$

A 43 kΩ pullup resistor to 3.3 V is required on the open-drain $\overline{\text{PME}}$ (pin 68/N11). A pullup resistor may already exist on the motherboard.

6.2.4.2 $\overline{\text{HSENUM}}$

A 43 kΩ pullup resistor to 3.3 V is required on the open-drain $\overline{\text{HSENUM}}$ (pin 71/N12). A pullup resistor may already exist on the motherboard.

6.2.4.3 Two-Wire Serial Interface Signals, SDA and SCL

When implementing the serial EEPROM via generic two-wire serial bus interface on general-purpose terminals GPIO0 and GPIO1, pullup resistors are required on the open-drain SCL and SDA signals. For typical PCI2040 serial EEPROM applications, a pullup of between 2 kΩ and 10 kΩ is recommended.

6.3 Bypass Capacitors

Standard design rules for the supply bypass should be followed. Low inductance ceramic chip capacitors are best for bypass capacitors. A value of 0.1 μF is recommended for each of the power supply pins V_{CC}, V_{CCP}, and V_{CCH}.

Chapter 7

Software Considerations

This chapter contains several software-related items of particular interest to those working with the PCI2040.

Topic **Page**

7.1 PCI AD[12:11] Mapping

During the PCI address phase, AD[12:11] determine which HPI register in the DSP is being accessed. Because of the differences in the definitions of HCTL[1:0] on TMS320C54x and TMS320C6x DSPs, it is important to make sure that AD[12:11] are set properly. Table 7–1 shows the mapping for the two DSP families.

Table 7–1. AD[12:11] Mapping

AD[12:11]	TMS320C6x HPI Register	TMS320C54x HPI Register
00	HPI control register	HPI control register
01	HPI address register	HPI data register autoincrement
10	HPI data register autoincrement	HPI address register
11	HPI data register	HPI data register

7.2 PCI2040 Chip Select Decoding

During the address phase AD[14:13] determine which of the four possible DSPs connected to the PCI2040 is being accessed. Table 7–2 shows the mapping for AD[14:13].

Table 7–2. AD[14:13] Mapping

AD[14:13]	Chip Select Asserted
00	HCS0
01	HCS1
10	HCS2
11	HCS3

7.3 PCI Byte Enables

PCI byte enables are important when reading and writing from a TMS320C54x since only 16-bit words can be used. The only valid byte enables are 0000b, 0011b, and 1100b all other byte enables will cause the IntEvent.HPIError bit to be set, and an interrupt may be signaled. If a double word access is requested, the PCI2040 will convert the request into two successive word accesses. Byte enables are also important on the 16-bit GP bus. The only valid byte enable for the GP bus is 1100b. All other byte enables result in an error.

7.4 HPI Interface Initialization

After a power-on reset (G_RST), the PCI2040 will preload several registers from the serial ROM, if implemented. Among these registers are the HPI data

width and implementation registers. These two registers must be programmed either by the serial ROM or software so the PCI2040 knows which DSPs are available and what type of HPI interface to use (8-bit or 16-bit).

There are several steps, usually taken care of by the BIOS, that software must take to initialize the PCI2040. FIrst software must program the HPI CSR memory or I/O base address register to provide access to the HPI control/status registers in the PCI2040. Next software must program the control space base address register to map the HPI bus into host memory space. The command register then needs to be programmed to allow the PCI2040 to claim memory and I/O cycles. If a serial ROM is not used, the HPI data width and implementation registers can now be programmed. Software can then program the HPI reset register to de-assert reset on the HPI bus.

7.5 GP Bus Initialization

The GP bus should be enabled via the serial EEPROM so that BIOS can program the GPbus base address register. The GPbus must be enabled before the GPbus base address register can be programmed.

7.6 PCI Little Endian

The DSP can support both little endian and big endian. Because the PCI bus uses the little-endian notation, it is very important to remember that the PCI2040 will always transfer data on the HPI bus in little endian. For example, if the data on the PCI bus is DDCCBBAAh and this data is targeted for a C54X, then this data will be broken up into four byte transfers in order of AAh, BBh, CCh, and then DDh. Because of this, it is very important for software to set the BOB bit for the C54X or the HWOB bit for the C6X in the host port control register (HPIC) to the appropriate value to match the endian notation the DSP is expecting.

Chapter 8

Other Considerations

This chapter covers some miscellaneous topics likely to be of interest in PCI2040 applications.

Topic **Page**

8.1 Serial EEPROM Implementation

The PCI2040 provides a two-wire serial ROM interface that may be used to preload registers following a power-on reset ($\overline{G\_RST}$). The PCI2040 expects the serial EEPROM to be at address 0xA0. The serial EEPROM interface includes a serial clock (SCL) output and a serial data (SDA) input/output. The SCL signal maps to the GPIO0 terminal and the SDA signal maps to the GPIO1 terminal. The two-wire serial EEPROM interface is enabled by pulling up both GPIO0 and GPIO1 terminals to V_{CC} with resistors. The PCI2040 will only sense GPIO0 on $\overline{G\_RST}$ to identify the serial EEPROM; thus, only GPIO0 must be tied low to disable the serial EEPROM interface. The PCI2040 will not read from an EEPROM that has 0xFF as the first byte. This ensures the PCI2040 does not load its registers from an initialized EEPROM.

The registers that may be preloaded are given in the list that follows, and only write-accessible bits in these registers may be preloaded. Figure 8–1 illustrates the PCI2040 serial EEPROM data format.

- ClassCode Register : SubClass – SubClass
- ClassCode Register : BaseClass – BaseClass
- Subsystem Vendor ID Register – SubSys Byte 0 and SubSys Byte 1
- Subsystem ID Register – SubSys Byte 2 and SubSys Byte 3
- GPIO Select Register – GPIO Select Register
- Miscellaneous Control Register – MiscCtrl Byte 0 and MiscCtrl Byte 1
- Diagnostic Register – Diagnostic
- HPI DSP Implementation Register – HPI_Imp Byte 0
- HPI Data Width Register – HPI_DW Byte 0

Figure 8–1. PCI2040 Serial EEPROM Data Format

Slave Address 1010 0000b

SubClass	Word Address 0	RSVD	Word Address 16
BaseClass	Word Address 1	RSVD	Word Address 17
SubSys Byte 0	Word Address 2	RSVD	Word Address 18
SubSys Byte 1	Word Address 3	RSVD	Word Address 19
SubSys Byte 2	Word Address 4	RSVD	Word Address 20
SubSys Byte 3	Word Address 5	RSVD	Word Address 21
GPIO Select	Word Address 6	RSVD	Word Address 22
RSVD	Word Address 7	RSVD	Word Address 23
RSVD	Word Address 8	RSVD	Word Address 24
Misc Ctrl Byte 0	Word Address 9	RSVD	Word Address 25
Misc Ctrl Byte 1	Word Address 10	RSVD	Word Address 26
Diagnostic	Word Address 11	RSVD	Word Address 27
HPI_Imp Byte 0	Word Address 12	RSVD	Word Address 28
RSVD	Word Address 13	RSVD	Word Address 29
HPI_DW Byte 0	Word Address 14	RSVD	Word Address 30
RSVD	Word Address 15	RSVD_DIAG	Word Address 31
		... Avail ...	

8.2 Compact PCI Hot Swap Implementation

The PCI2040 is hot-swap-friendly silicon, which means it contains support for software connection control plus the following features:

- PCI Specification 2.1 compliant
- Tolerates V_{CC} from early power
- Asynchronous reset
- Tolerates precharge voltage
- I/O buffers meet modified V/I requirements
- Limited I/O pin leakage at precharge voltage
- Hot swap terminals: $\overline{\text{HSENUM}}$, HSSWITCH, HSLED

For information on implementing compact PCI hot swap see the Compact PCI Hot Swap Specification R1.0.

8.3 Benefit of the Extra Address Line on the GPbus

The extra address line on the GPbus can be used as a chip select if a JTAG TBC is being used. This can allow for the implementation of a second 16-bit peripheral on the GPbus. Some glue logic will be required for this implementation. See the PCI2040 EVM guide for a sample implementation.

8.4 GPIO Interface

The GPIO interface on the PCI2040 consists of six GPIO pins. GPIO5 and GPIO4 can be used either as GPIO pins or as $\overline{\text{GP\_RD}}$ and $\overline{\text{GPWR}}$ strobes. GPIO[5:4] are programmed as GPbus pins automatically when the GPbus is enabled. GPIO pins 3 and 2 can be used as either GPIO pins or general-purpose interrupt pins. GPIO[3:2] can be programmed as interrupts by setting bits 3 and 2 in the GPIO select register. GPIO[1:0] are always configured as GPIO pins.

8.5 PCI Power Management

The PCI2040 is PCI Power Management 1.1 compliant and also supports wake-up from a low power state through the use of $\overline{\text{PME}}$ signal (power management event). The $\overline{\text{PME}}$ signal is an open drain signal, therefore requiring a pullup resistor. If the PCI2040 is being implemented on an add-in board, then the pullup resistor will usually reside on the motherboard. The $\overline{\text{PME}}$ must be routed to A19 on the PCI connector. This pin used to be a reserved signal but now, according to the *PCI Specification 2.2*, is used for $\overline{\text{PME}}$.

The PCI2040 does support all four power states: D0, D1, D2, D3hot, and D3cold. In order for the PCI2040 to support D3cold, a method must exist to switch the PCI2040 V_{CC} to an auxiliary power source (V_{AUX}). If D3cold is going to be supported, then software must make sure bit 15 in the PCI power management capabilities register (PCI offset 52h) is a 1. This bit notifies the operating system that D3cold is supported.

For more information on PCI Power Management, consult the *PCI Bus Power Management Specification Version 1.1*.

8.6 Global Reset

There are two reset inputs, $\overline{\text{G\_RST}}$ and $\overline{\text{PCI\_RST}}$, to the PCI2040. Both of these resets will reset the PCI2040 to its default state. The difference between the two resets is that the $\overline{\text{G\_RST}}$ will reset every PCI2040 internal register when asserted. The $\overline{\text{PCI\_RST}}$ will reset every register if bit 8 in the power management control/status register (PCI offset 54h) is not set. Otherwise, if bit 8 is set, the assertion of $\overline{\text{PCI\_RST}}$ will clear all internal registers except those designated as sticky. A complete list of sticky bits can be found in the PCI2040 datamanual.

If only one reset is available for the PCI2040, tying together $\overline{\text{PCI\_RST}}$ and $\overline{\text{G\_RST}}$ is recommended. Otherwise, two different resets, a local reset ($\overline{\text{PCI\_RST}}$) and a system wide reset ($\overline{\text{G\_RST}}$), need to be connected to the PCI2040.