User Guide for PCI SSD4 /
Addendum for SSD4 Firmware

SSD4 Interface

PCI SSD4 / PCI CD board

Synchronous serial DMA interface (1-, 2-, or 4- channel) for LVDS or RS422

Doc. 008-01134-04
Rev. 2015 September 16

EDT | Engineering Design Team, Inc.
3423 NE John Olsen Ave
Hillsboro, OR 97124
U.S.A.
Tel: +1-503-690-1234 | Toll free (in U.S.A.): 800-435-4320
Fax: +1-503-690-1243
www.edt.com

Terms of Use Agreement

Definitions. This agreement, between Engineering Design Team, Inc. (“Seller”) and the user or distributor (“Buyer”), covers the use and distribution of the following items provided by Seller: a) the binary and all provided source code for any and all device drivers, software libraries, utilities, and example applications (collectively, “Software”); b) the binary and all provided source code for any and all configurable or programmable devices (collectively, “Firmware”); and c) the computer boards and all other physical components (collectively, “Hardware”). Software, Firmware, and Hardware are collectively referred to as “Products.” This agreement also covers Seller’s published Limited Warranty (“Warranty”) and all other published manuals and product information in physical, electronic, or any other form (“Documentation”).

License. Seller grants Buyer the right to use or distribute Seller’s Software and Firmware Products solely to enable Seller’s Hardware Products. Seller’s Software and Firmware must be used on the same computer as Seller’s Hardware. Seller’s Products and Documentation are furnished under, and may be used only in accordance with, the terms of this agreement. By using or distributing Seller’s Products and Documentation, Buyer agrees to the terms of this agreement, as well as any additional agreements (such as a nondisclosure agreement) between Buyer and Seller.

Export Restrictions. Buyer will not permit Seller’s Software, Firmware, or Hardware to be sent to, or used in, any other country except in compliance with applicable U.S. laws and regulations.

For clarification or advice on such laws and regulations, Buyer should contact...

- **The U.S. Department of Commerce, Export Division, Washington, D.C., U.S.A.**

...or, if the beginning or title page of this documentation shows an ITAR status statement, Buyer should contact...

- **The U.S. Department of Commerce, Export Division, Washington, D.C., U.S.A.**

Limitation of Rights. Seller grants Buyer a royalty-free right to modify, reproduce, and distribute executable files using the Seller’s Software and Firmware, provided that: a) the source code and executable files will be used only with Seller’s Hardware; b) Buyer agrees to indemnify, hold harmless, and defend Seller from and against any claims or lawsuits, including attorneys’ fees, that arise or result from the use or distribution of Buyer’s products containing Seller’s Products. Seller’s Hardware may not be copied or recreated in any form or by any means without Seller’s express written consent.

No Liability for Consequential Damages. In no event will Seller, its directors, officers, employees, or agents be liable to Buyer for any consequential, incidental, or indirect damages (including damages for business interruptions, loss of business profits or information, and the like) arising out of the use or inability to use the Products, even if Seller has been advised of the possibility of such damages. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitations may not apply to Buyer. Seller’s liability to Buyer for actual damages for any cause whatsoever, and regardless of the form of the action (whether in contract, product liability, tort including negligence, or otherwise) will be limited to fifty U.S. dollars ($50.00).

Limited Hardware Warranty. Seller warrants that the Hardware it manufactures and sells shall be free of defects in materials and workmanship for a period of 12 months from date of shipment to initial Buyer. This warranty does not apply to any product that is misused, abused, repaired, or otherwise modified by Buyer or others. Seller’s sole obligation for breach of this warranty shall be to repair or replace (F.O.B. Seller’s plant, Beaverton, Oregon, USA) any goods that are found to be non-conforming or defective as specified by Buyer within 30 days of discovery of any defect. Buyer shall bear all installation and transportation expenses, and all other incidental expenses and damages.

Limitation of Liability. *In no event shall Seller be liable for any type of special consequential, incidental, or penal damages, whether such damages arise from, or are a result of, breach of contract, warranty, tort (including negligence), strict liability, or otherwise.* All references to damages herein shall include, but not be limited to: loss of profit or revenue; loss of use of the goods or associated equipment; costs of substitute goods, equipment, or facilities; downtime costs; or claims for damages. Seller shall not be liable for any loss, claim, expense, or damage caused by, contributed to, or arising out of the acts or omissions of Buyer, whether negligent or otherwise.

No Other Warranties. Seller makes no other warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose, regarding Seller’s Products or Documentation. Seller does not warrant, guarantee, or make any representations regarding the use or the results of the use of the Products or Documentation or their correctness, accuracy, reliability, currentness, or otherwise. All risk related to the results and performance of the Products and Documentation is assumed by Buyer. The exclusion of implied warranties is not permitted by some jurisdictions. The above exclusion may not apply to Buyer.

Disclaimer. Seller’s Products and Documentation, including this document, are subject to change without notice. Documentation does not represent a commitment from Seller.

Contents

User Guide for PCI SSD4 / Addendum for SSD4 Firmware

Overview

About the SSD4 Interface

The SSD4 (synchronous serial DMA interface for 1, 2, or 4 channels) enables an EDT board to support 1, 2, or 4 independent single-bit serial channels, each accepting a differential data signal (twisted pair) and a differential clock (twisted pair) at standard RS422 or LVDS signal levels.

NOTE The SSD4 interface uses the same registers as the PCI CD/a board – except that a few of the PCI CD/a registers are modified for the SSD4 interface. This document contains only those modified registers; to see the complete set of PCI CD/a registers, consult the PCI CD/a user guide (see Related Resources below).

About This Document

This document – designed to be used with several EDT products – functions as:

- a user guide for the PCI SSD4; or
- an addendum to use the SSD4 firmware with other EDT boards, including at least the following...
  - — PCI CD/a;
  - — PMC CD-20 and -60;
  - — SSD4 (for legacy Sbus);
  - — SCD-20 and -60 (for legacy Sbus).

NOTE Use the SSD4 firmware only with a specified product – and consult the user guide for that product and for the PCI CD/a (see Related Resources below), both of which contain additional information not found in this document.

Related Resources

The resources below may be helpful or necessary for your applications.

EDT resource	*Weblink*
• User guides and documentation for your specific product:	edt.com
— PCI SSD4 (PCI form factor)	
— PCI CD/a	
— PMC CD-20 or -60 (PMC form factor)	
— SSD4 (Sbus form factor)	
— SCD-20 or -60 (Sbus form factor)	
• Application Programming Interface (API)	edt.com/download-hub/
• Installation packages: Windows, Linux	edt.com/download-hub/

About the Software and Firmware

Install the SSD4 hardware and software according to the instructions in the PCI CD / CDa user guide, but using one of the firmware files listed below...

`ssd.bit`	UI FPGA configuration file for single-channel synchronous serial input, multiplexed single-channel DMA.
`ssd2.bit`	UI FPGA configuration file for dual-channel synchronous serial input, multiplexed single-channel DMA.
`ssd4.bit`	UI FPGA configuration file for four-channel synchronous serial input, multiplexed single-channel DMA.
`ssdtest.bit`	UI FPGA configuration file for generating test data used for testing the SSD4.

Compatible single-channel bitfiles must be loaded in the PCI FPGA on the PCI CD. These are:

`pcd20.bit`	The PCI FPGA configuration file for PCI CD-20 boards.
`pcd60.bit`	The PCI FPGA configuration file for PCI CD-60 boards.

The following software initialization files are available for use with the `initpcd` automatic configuration utility...

`ssd.cfg`	Software initialization file for single-channel synchronous serial input, multiplexed single-channel DMA.
`ssd2.cfg`	Software initialization file for dual-channel synchronous serial input, multiplexed single-channel DMA.
`ssd4.cfg`	Software initialization file for four-channel synchronous serial input, multiplexed single-channel DMA.

The following utilities are also available...

`readssd`	Executable test program to load test firmware into two boards, initialize them, send out test data from one board, and acquire the test data on the other board.
`readssd.c`	C source for the test program, useful as an example of acquiring data.

In some cases, the firmware file names you see in the top-level `PCD` directory may not match the file names listed in this guide, because PCI bus slots come in two varieties: those supplying 3 V power, and those supplying 5 V power. Different firmware is required for the two kinds of slots, but the correct firmware file is chosen automatically when you run `pciload` or any other firmware-loading utility from EDT.

For example, you may see files named `cda16_3v.bit` and `cda16_5v.bit`, but the correct argument – in this example, `cda16_classic.bit` – is chosen and loaded automatically.

In some cases, you may also see additional firmware files incorporating changes required for various board revisions, or files with the same name in different subdirectories. However, you need not be concerned with any of these variations of name or path. In all cases, the names given above are the correct arguments to supply to the firmware-loading utilities.

The PCD Device Driver

The `PCD` device driver is EDT software which runs on the host computer so that the host operating system can communicate with the SSD4. The driver is loaded into the kernel upon installation, and thereafter runs as a kernel module. The driver name and subdirectory are specific to each supported operating system; the EDT installation script automatically installs the correct device driver, in the correct manner, as required by the specified interface and operating system.

FPGA Configuration (.bit) Files

FPGA configuration (typically `.bit`) files, located in the top-level `PCD` directory, are the firmware required for PCI/e and UI FPGA functionalities. Typically the files for PCI/e FPGA functionality are in the `flash` subdirectory, while the files for UI FPGA functionality are in the `bitfiles` subdirectory.

Each FPGA must be loaded with the correct firmware in order to supply the functionality chosen by the user, and also in order to be compatible with the firmware loaded in any other FPGA. Typically the correct file for PCI FPGA functionality is preloaded by EDT, but the correct file for UI FPGA functionality is user-loaded.

The firmware files specific to your SSD4 are listed at the beginning of this section. Instructions for loading them are provided in Configuring the SSD4.

Software Initialization (.cfg) Files

Software initialization (typically `.cfg`) files, located in the `pcd_config` subdirectory of the top-level `PCD` directory, are editable text files that run like scripts to prepare EDT boards to perform DMA transfers. The commands in a software initialization file are defined in a C application named `initpcd.` When you invoke `initpcd`, you use the `-f` flag to specify which software initialization file to use.

A typical software initialization file loads an FPGA configuration file for UI FPGA functionality, and configures the board for DMA. Some software initialization files also load FPGA configuration files for other FPGAs and/or FPGA functionality.

Typically there is at least one software initialization file for each EDT product and/or FPGA configuration file. The software initialization files specific to your SSD4 are listed at the beginning of this section. Instructions for their use are provided in Configuring the SSD4.

Commands defined in `initpcd` and typically found in software initialization files allow for specific FPGA configuration files to be loaded (for example, `bitfile:`), write specified hexadecimal values to specified registers (for example, `command_reg:`), enable or disable byte-swapping or short-swapping to accommodate different operating systems' requirements for bit ordering (for example, `byteswap:`), or invoke arbitrary commands (for example, `run_command:`). For example:

```
bitfile: ssd16io.bit
    command_reg: 0x08
    byteswap: 1
    run_command: set_ss_vco -F 1000000 2
```

For complete usage details, enter `initpcd --help`.

C source for `initpcd` is included so that you can add your own commands, if you wish. You can then edit your own software initialization file to use your new commands and specify that `initpcd` use your new file when configuring your board. If you would like us to include your new software initialization commands in subsequent releases of `initpcd`, email us at tech@edt.com.

Sample Applications and Utilities

Along with the driver, the FPGA configuration files, and the software initialization files, the EDT installation CD includes a number of applications and utilities that you can use to initialize and configure the board, access registers, or test the board. For many of these applications and utiilities, C source is also provided, so that you can use them as starting points to write your own applications. The most commonly useful are described below; see the README file for the complete list.

NOTE To build a new application, we recommend downloading the latest EDT installation package (see Related Resources). To rebuild an existing application, avoid version issues by using the same package used to build it, or relink / recompile the application using our latest download package.

Sample Applications

`rd16`	Performs simple ring buffer input for multiple DMA channels.
`wr16`	Performs simple ring buffer output for multiple DMA channels.
`simple_read`	Performs DMA input without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
`simple_write`	Performs DMA output without using ring buffers. Data is therefore subject to interruptions, depending on system performance.
`simple_getdata`	An example of a variety of DMA-related operations, including reading the data from the connector interface and writing it to a file, as well as measuring input rate.
`simple_putdata`	An example of a variety of DMA-related operations, including reading data from a file and writing it out to the connector interface.
`test_timeout`	Under normal operation, timeouts cancel DMA transfers. This application exemplifies giving notification when a timeout occurs, without canceling DMA
`set_ss_vco`	A utility for programming the output clock or clocks on the SSD4 to specific frequencies used by the UI FPGA for input and output.

Utility Files

`initpcd`	A utility for initializing and configuring the SSD4.
`pdb`	A utility that enables interactive reading and writing of the UI FPGA registers.

Basic Self-Test Files

EDT provides files to perform basic tests, such as verifying proper installation, on your EDT board (for details, see Basic Self-Testing). These files include at least:

`sslooptest`	Tests most EDT boards. Determines the board model and runs the correct loopback test.

Building or Rebuilding an Application

To build or rebuild an application, run `make` in the top-level `PCD` directory, where the executables and `PCD` source files are located. Use a compiler that works with your operating system, as indicated below.

- Linux users: You can use the `gcc` compiler that is typically included with the Linux installation.
- Windows users: Install a C compiler (such as Microsoft Visual C), or contact tech@edt.com to use `gcc.`
- Solaris users: Contact EDT.

After your build is finished, use the `--help` command line option for a list of usage options and descriptions.

Configuring the SSD4

For proper operation, all EDT products must be loaded with the correct FPGA configuration files (firmware) for their FPGAs. At powerup, the PCI / PCIe FPGA is loaded automatically with the correct file from flash memory; however, you must load the user interface (UI) FPGA yourself. Before doing so, you may wish to check the firmware in the PCI / PCIe FPGA to ensure that it is correct and up to date.

Checking or Updating the PCI / PCIe FPGA Firmware

When upgrading to a new device driver, or switching to an FPGA configuration (firmware) file with special functionality, you may need to reprogram the PCI / PCIe interface flash memory using `pciload.`

NOTE The presence of a newer version of the firmware with a new driver does not necessarily mean that the firmware must be updated; the README file will tell you if a package contains a mandatory upgrade.

The following procedure applies to standard firmware only. If you are running a custom firmware file and need to update it, first get a custom firmware configuation file from EDT.

- On Windows systems, double-click the Pcd Utilities icon to bring up a command shell in the installation directory `\EDT\Pcd`.
- On Linux systems, `pciload` is an application in the installation directory `/opt/EDTpcd.`
- On Mac systems, `pciload` is an application in the installation directory `/Applications/EDT/pcd.`

To see currently installed and recognized EDT products and drivers, enter...

```
pciload
```

The program will output the date and revision number of the firmware in the flash memory.

To compare the PCI / PCIe FPGA firmware in the package with the firmware loaded on the product, enter...

```
pciload verify
```

If they match, there's no need to upgrade the firmware. If they differ, you'll see error messages. This does not necessarily indicate a problem; if your application is operating correctly, you may not need to upgrade the firmware. However, if you do wish to update the standard firmware, enter...

```
pciload update
```

1. To upgrade or switch to a custom firmware file, enter...

   `pciload` *`firmware_filename`*

   ...replacing *`firmware_filename`* with the name of the PCI / PCIe FPGA configuration file, with or without the `.bit` file extension.

NOTE If the host computer holds more than one board, you can specify the correct board to load with the optional *`unit_number`* argument (by default, 0 for the first or only board in a host), as shown...

`pciload -u` *`unit_number filename`*

2. At the prompt, press Enter to confirm the loading operation. (If the file date is older than the date of the file already in flash memory, you may need to press Enter twice.)

   The board reloads the firmware from the flash memory only during power-up, so after running `pciload,` the old firmware remains in the PCI / PCIe FPGA until the system has power-cycled.

NOTE Updating the firmware requires cycling power, not simply rebooting.

For a list of all `pciload` options, enter...

```
pciload --help
```

Loading the UI FPGA Firmware and Configuring the Interface

The utility `initpcd` loads the UI FPGA configuration files, programs the registers, sets the clocks (if necessary), and gets the SSD4 ready to perform DMA. This utility takes, as an argument, a software initialization file, and then automatically runs the pertinent commands.

If you use `initpcd` to configure the SSD4, your application can focus on performing DMA and other application-specific operations, so it will omit SSD4-specific operations and be portable to other EDT boards that peform DMA.

To configure the SSD4, enter...

`initpcd -u` *`unit_number`* `-f pcd_config/`*`filename`*`.cfg`

...replacing *`unit_number`* with the number of the board (by default, 0), and replacing *`filename`* with one of the initialization files listed in About the Software and Firmware; for example...

```
initpcd -f ssd4.cfg
```

NOTE Software initialization files are editable text files. If the files provided do not meet your needs, copy and modify the one that's closest to your required configuration; then run `initpcd` with your new file.

Using Custom FPGA Configuration Files

You can substitute your own FPGA configuration file, if necessary. If you wish to develop your own VHDL design, contact EDT. When you're done, be sure to create a new software initialization file for your new firmware file and update the `pcd_config` directory to include it.

Basic Self-Test

The SSD4 modification is provided with a basic self-test for verification. Running this test requires a second SSD4, and a straight-through cable (EDT part number CAB-DD).

You can also use a differential cable and four 25-pin D connector female-to-female adaptors, which are commonly available for extending and adapting computer serial and printer cables.

NOTE Some of these adaptors are wired to reverse the transmit and receive signals of an RS-232 interface. Such adaptors do not work for testing the SSD4. For testing, the SSD4 adaptors must be wired straight through: pin 1 wired to pin 1, pin 2 wired to pin 2, and so on throughout.

To set up for testing the SSD4:

1. Cable the two SSD4 boards together.If you're using the differential cable with the four DB-25 connectors, install an adaptor in each channel.
   - — The board that you are testing can use either a TTL or a differential cable.
   - — The board generating test data must use a differential cable. The TTL cable contains input circuitry inside the connector backshells that will block the test data.
   - — The SSD4 generating the test data can be in the same host computeras the board under test, or in a different one.
2. Start data output. On the host computer containing the board you wish to test, at the command prompt, enter:

```
readssd -u unit_number
```

The default unit number is 0, indicating the first board in a system. If you have only one board in your system, you can omit the `-u` option. For a complete list of optional arguments, enter:

```
readssd -h
```

By default, `readssd` uses a clock rate of 10 MHz for a PCI CD-20 or 30 MHz for a PCI CD-60. If the default clock rate is not acceptable, you can select the clock rate using the
`-s` option, followed by 0 (default), 1 (divide by 2), 2 (divide by 4), or 3 (divide by 8):

The program `readssd` loads the test firmware and outputs a known stream of serial data. It detects the configuration of the device under test and acquires the specified block of data. The data is compared to the expected values and errors, if any, are printed.

If an error occurs, the output includes the string `ERROR`. If you do not see that string, then the board is operating correctly.

Data Formats

The SSD4 can store data in memory from up to four channels. Each channel can have 1, 2 or 4 data bits, and the data can be stored either least significant bit first or most significant bit first.

The number of channels supported is determined by the specific FPGA configuration file, as described in About the Software and Firmware on page 5. The number of bits per channel and the bit storage order is determined by three bits, as described in the register 0x02 Funct (modified) on page 20.

The following six figures illustrate the way data is stored using all valid combinations of the three bits, for the case of the single-channel configuration. The multiple-channel configurations store data in the same six ways, with the addition of a channel identifier tag stored in the next memory word as illustrated in Figure 7 on page 17.

Figure 1. Single Channel, Single Bit, Least Significant Bit First

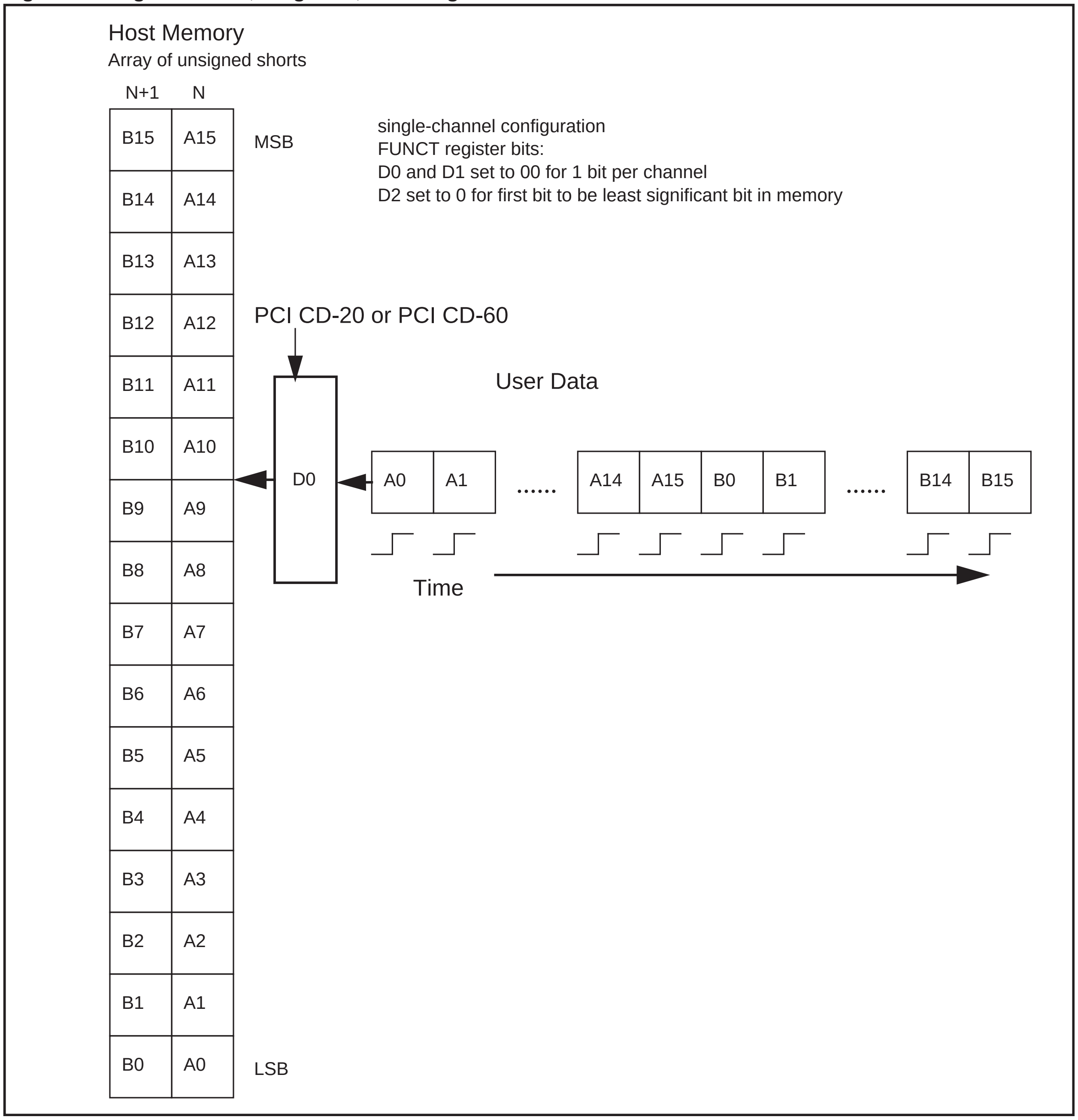


Figure 2. Single Channel, Single Bit, Most Significant Bit First

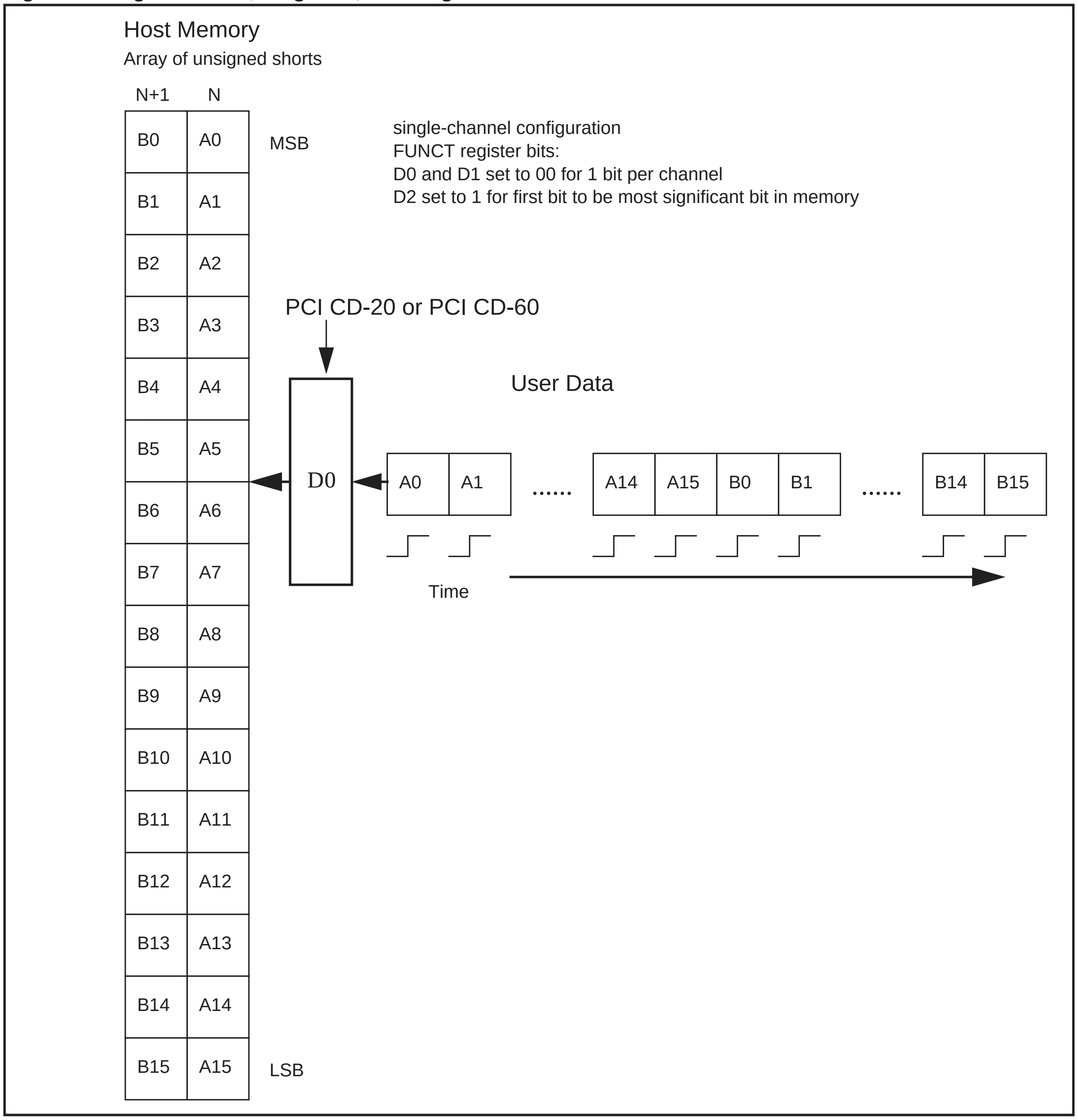


Figure 3. Single Channel, Two Bit, Least Significant Bit First

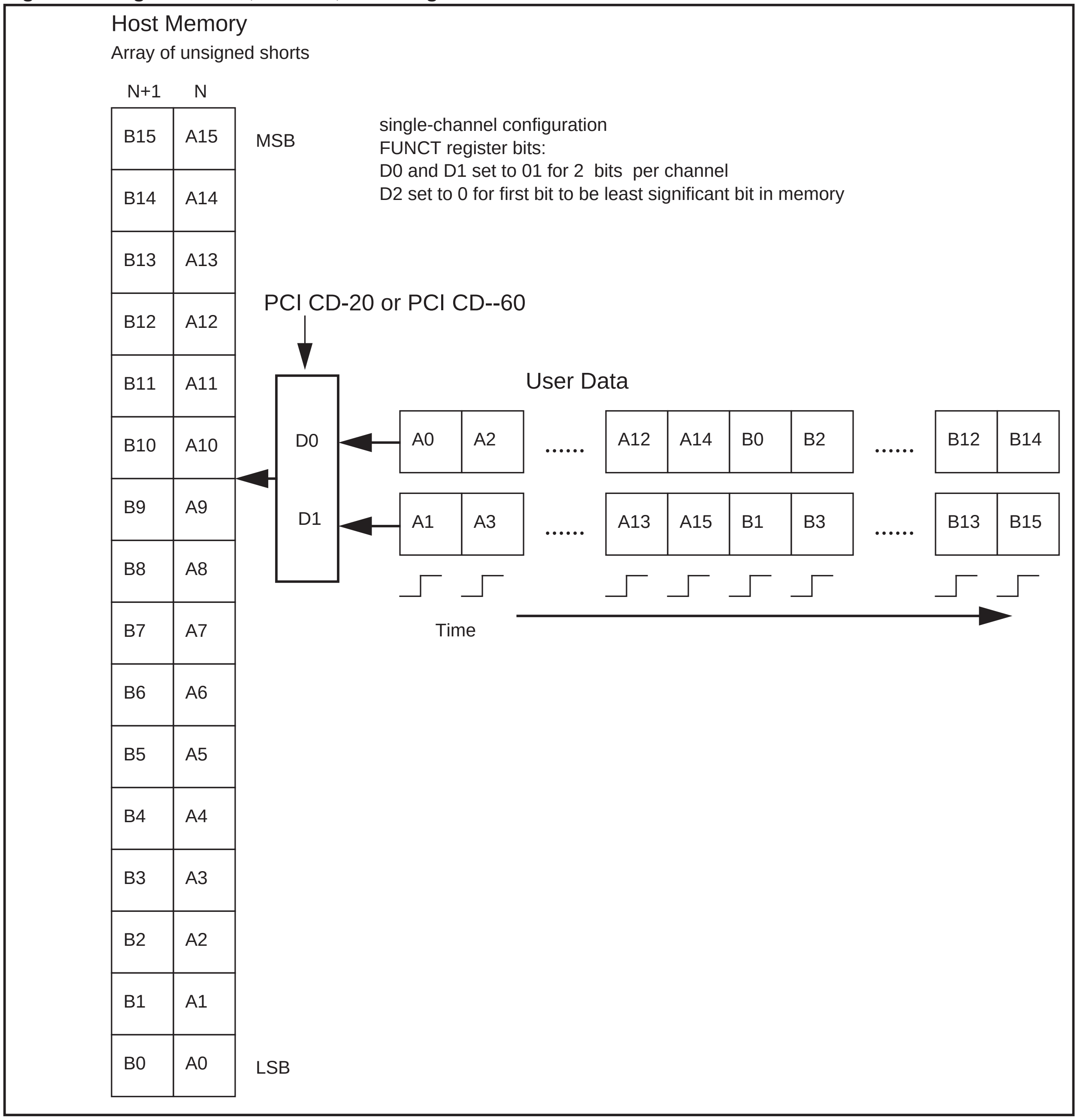


Figure 4. Single Channel, Two Bit, Most Significant Bit First

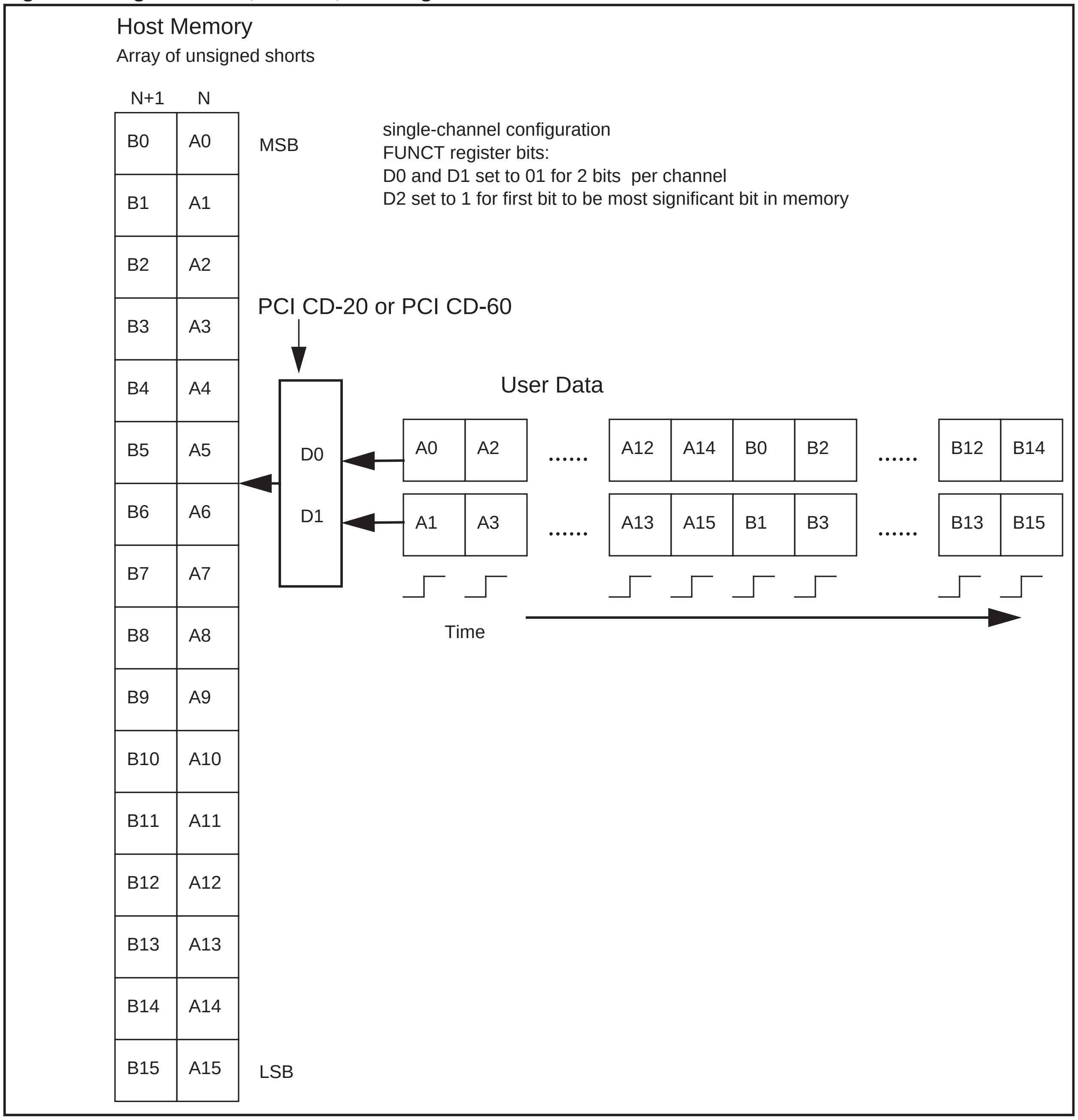


Figure 5. Single Channel, Four Bit, Least Significant Bit First

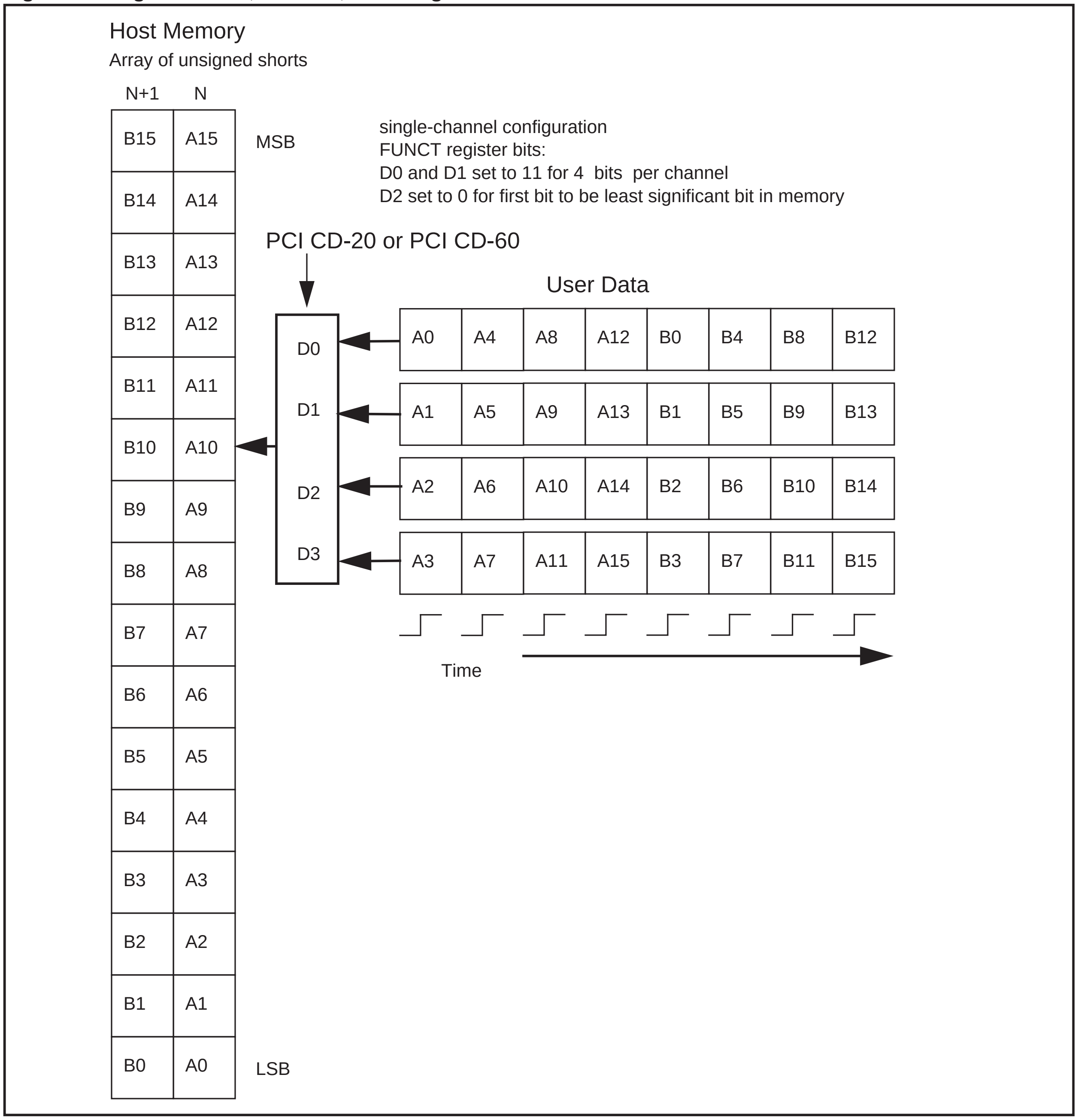
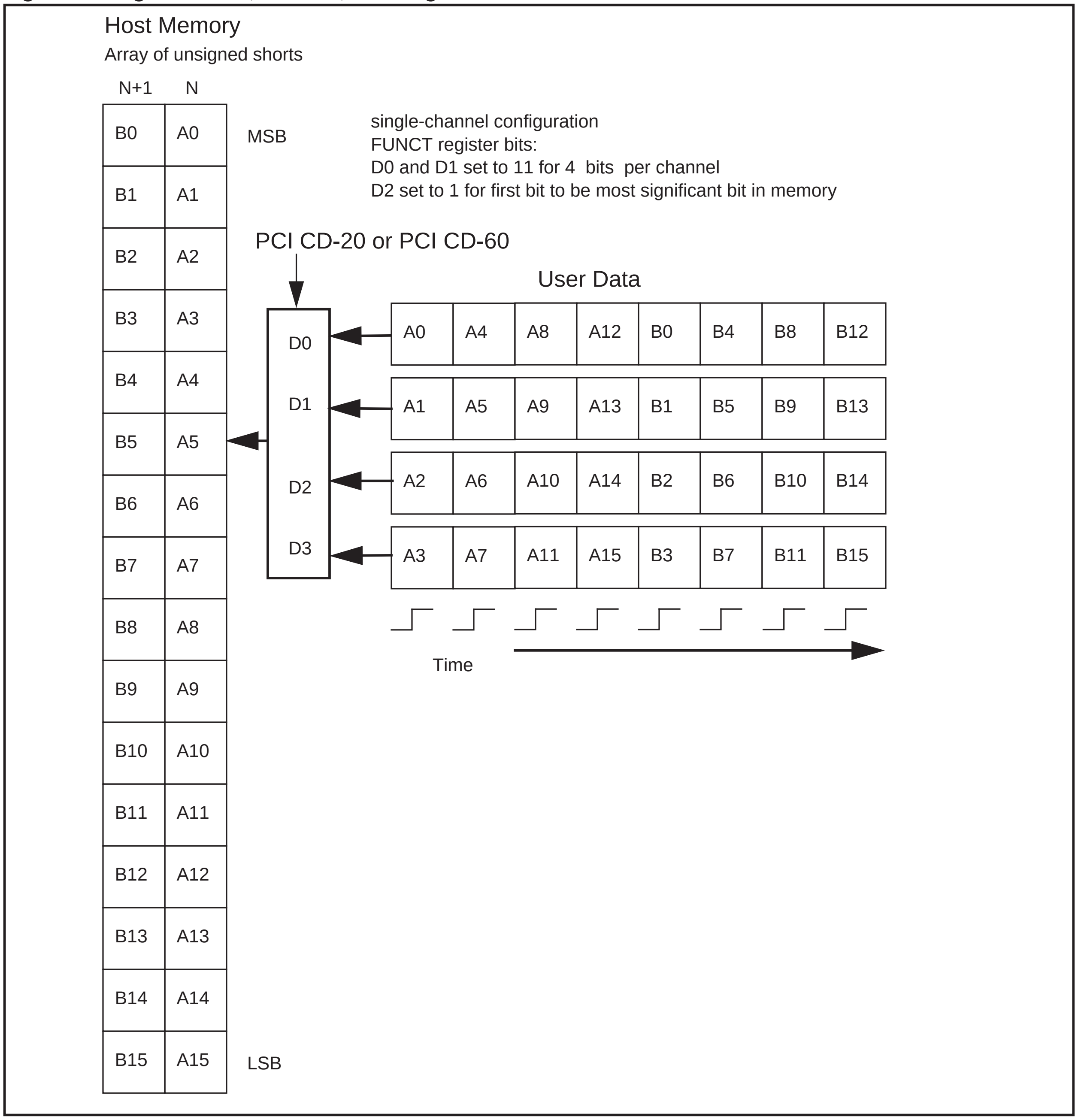

Host Memory
Array of unsigned shorts
N+1
N
B15 A15 MSB
B14 A14
B13 A13
B12 A12
B11 A11
B10 A10
B9 A9
B8 A8
B7 A7
B6 A6
B5 A5
B4 A4
B3 A3
B2 A2
B1 A1
B0 A0 LSB
single-channel configuration
FUNCT register bits:
D0 and D1 set to 11 for 4 bits per channel
D2 set to 0 for first bit to be least significant bit in memory
PCI CD-20 or PCI CD-60
D0
D1
D2
D3
User Data
A0 A4 A8 A12 B0 B4 B8 B12
A1 A5 A9 A13 B1 B5 B9 B13
A2 A6 A10 A14 B2 B6 B10 B14
A3 A7 A11 A15 B3 B7 B11 B15
Time

Figure 6. Single Channel, Four Bit, Most Significant Bit First


Host Memory
Array of unsigned shorts
N+1
N
B0 A0
B1 A1
B2 A2
B3 A3
B4 A4
B5 A5
B6 A6
B7 A7
B8 A8
B9 A9
B10 A10
B11 A11
B12 A12
B13 A13
B14 A14
B15 A15
MSB
LSB
single-channel configuration
FUNCT register bits:
D0 and D1 set to 11 for 4 bits per channel
D2 set to 1 for first bit to be most significant bit in memory
PCI CD-20 or PCI CD-60
User Data
D0
D1
D2
D3
A0 A4 A8 A12 B0 B4 B8 B12
A1 A5 A9 A13 B1 B5 B9 B13
A2 A6 A10 A14 B2 B6 B10 B14
A3 A7 A11 A15 B3 B7 B11 B15
Time

The memory pattern illustrated below (Figure 7) occurs using four channels with the same frequency clock. In this case, the even-numbered tag locations always indicate the channel from which the data came. If a channel is not used or clock frequencies are varied, the data channels can occur in any order, as you require.

Figure 7. Multiple Channels' Data Storage in Memory

n	n+1	n+2	n+3	n+4	n+5	n+6	n+7	n+8	n+9	n+10	n+11	n+12	n+13	n+14	n+15
chan-nel #	data	chan-nel #	data	chan-nel #	data	chan-nel #	data	chan-nel #	data	chan-nel #	data	chan-nel #	data	chan-nel #	data

Channel Interface and Connector Pin Assignments

Table 1 shows the signals going out the 80-pin connector.

Table 1. PCI CD (ssd.bit, ssd2.bit, ssd4.bit) Connector Pinout

Pin	Signal	Pin	Signal
1	ground	41	ground
2	ground	42	ground
3	CH2D0+	43	CH1D0+
4	CH2D0–	44	CH1D0–
5	CH2D1+	45	CH1D1+
6	CH2D1–	46	CH1D1–
7	CH2D2+	47	CH1D2+
8	CH2D2–	48	CH1D2–
9	CH2D3+	49	CH1D3+
10	CH2D3–	50	CH1D3–
11	CH4D0+	51	CH3D0+
12	CH4D0–	52	CH3D0–
13	CH4D1+	53	CH3D1+
14	CH4D1–	54	CH3D1–
15	CH4D2+	55	CH3D2+
16	CH4D2–	56	CH3D2–
17	CH4D3+	57	CH3D3+
18	CH4D3–	58	CH3D3–
19	not used	59	not used
20	+5V DC (fused)	60	+5V DC (fused)
21	not used	61	not used
22	not used	62	not used
23	ground	63	ground
24	reserved	64	CH1CLK+
25	reserved	65	CH1CLK–
26	reserved	66	reserved
27	reserved	67	reserved
28	reserved	68	reserved
29	reserved	69	reserved
30	reserved	70	CH3CLK+
31	reserved	71	CH3CLK–
32	CH2CLK+	72	reserved
33	CH2CLK–	73	reserved
34	reserved	74	reserved
35	reserved	75	reserved
36	reserved	76	reserved
37	reserved	77	reserved
38	CH4CLK+	78	reserved
39	CH4CLK–	79	reserved
40	ground	80	ground

You can attach either a TTL or an RS422 differential cable to the 80-pin connector. Each cable has one connector to attach to the SSD4, and four 25-pin D connectors to attach to the user signals. Each of the 25-pin connectors represents inputs associated with one channel.

Because the board itself has RS422 inputs terminated in 100 Ω, the differential cable is wired directly to the appropriate SSD4 inputs. The TTL cable contains active circuitry in each 25-pin connector backshell to convert TTL level signals to the RS422 signals that the SSD4 requires. These TTL inputs are high impedance and have circuitry to protect against constant input voltages to 30 V and moderate electrostatic discharge.

Each 25-pin D connector pinout is as shown below.

Table 2. D Connector Pinout

Pin	Differential	TTL	Pin	Differential	TTL
1	CHCLK +	CHCLK	14	not connected	Ground
2	CHCLK –	not connected	15	not connected	Ground
3	CHD0 +	CHD0	16	not connected	Ground
4	CHD0 –	not connected	17	not connected	Ground
5	CHD1 +	CHD1	18	not connected	Ground
6	CHD1 –	not connected	19	not connected	Ground
7	CHD2 +	CHD2	20	not connected	Ground
8	CHD2 –	not connected	21	not connected	Ground
9	CHD3 +	CHD3	22	not connected	Ground
10	CHD3 –	not connected	23	not connected	Ground
11	not connected	reserved	24	not connected	Ground
12	not connected	reserved	25	Ground	Ground
13	not connected	reserved			

Registers (Modified from PCI CD/a Registers)

The SSD4 registers shown below are modified from the PCI CD/a registers. For details, consult the PCI CD/a user guide (see Related Resources on page 5).

0x00 Command (modified)

Access / Notes: 8-bit read-write / PCD_CMD

Bit	Access	Name	Description
7		–	Reserved.
6–5		PCD_STAT_INT_EN	A value of one enables the corresponding STAT bit to cause an interrupt when it is asserted.
4			Reserved.
3		PCD_ENABLE	Set to one to enable the SSD4 interface. This bit is set after the direction is chosen and typically after the first DMA buffer is ready. To reset direction or flags, toggle this bit. To flush the DMA FIFOs, clear then set this bit.
2		–	Reserved.
1		PCD_FORCEBNR	When set, indicates that the board is not ready.
0		PCD_DIR	Reserved. Data direction is always into the host computer. If the software attempts a write on the SSD4 after the firmware is loaded, the program waits forever for completion or timeout.

0x01 Data Path Status (modified)

Access / Notes: 8-bit read-only / PCD_DATA_PATH_STAT

Bit	Access	Name	Description
7		PCD_IDV	Reflects IDV state.
6		PCD_INFFAFULL	If set, input FIFO is almost full.
5–4		PCD_INFFULL	If set, input FIFO is full.
3		PCD_OVERFLOW	Set if the input FIFO fills, indicating lost data. Reset this bit by toggling bit 3 (PCD_ENABLE) in 0x00 Command.
2		PCD_UNDERFLOW	Always clear.
1		PCD_IF_NOT_EMP	If this bit is set, the input FIFO is not empty.
0		PCD_OF_NOT_EMP	Reserved. The SSD4 cannot output data.

0x02 Funct (modified)

Access / Notes: 8-bit read-write / PCD_FUNCT
Used to set the data acquisition mode.

Bit	Access	Name	Description
7		PCD_PLLCLK	Set to enable the PLL circuit; see the PLL Divider Register for the use of this bit.
6		PCD_SELAV	Used to program the PLL and select the clock.
5		PCD_FREQ7	Used to program the PLL and select the clock.

Bit	Access	Name	Description
4		PCD_FREQ6	Used to program the PLL and select the clock.
3		–	Reserved.
2		PCD_FUNCT[2]	Controls the order in which the acquired data is placed in each 16-bit word stored in the host memory: • When set, the first bit acquired is placed in the most significant bit. • When clear, the first bit acquired is placed in the least significant bit. If multiple data bits are acquired per clock cycle, the most significant data bit is considered the first bit.
1–0		PCD_FUNCT[1,0]	Specifies the number of data bits acquired on each clock, for all channels: 00 Bit 0 of each channel is acquired with each clock. 01 Bits 0 and 1 of each channel are acquired with each clock. 10 undefined 11 Bits 0–3 of each channel are acquired on each clock.

0x03 Stat (modified)

Access / Notes: 8-bit read-only / PCD_STAT
The signals STAT(0–3) are disconnected.

Bit	Access	Name	Description
7–0		–	Not used; set to zero.

0x03 Stat Polarity (modified)

Access / Notes: 8-bit read-write / PCD_STAT_POLARITY
This register is implemented and can be read or written, but is not used.

Bit	Access	Name	Description
7–0		–	Not used.

0x05 Option (modified)

Access / Notes: 8-bit read-only / PCD_OPTION
Returns the following values, depending on which firmware is installed:

0x08	`ssd.bit`, single-channel 10 MHz maximum clock rate
0x09	`ssd2.bit`, two-channel 5 MHz maximum clock rate
0x0a	`ssd4.bit`, four-channel 1.25 MHz maximum clock rate

Revision Log

Below is a history of modifications to this guide.

Date	Rev	By	Pp	Detail
20150916	04	PH	1-5	• Per RH, revised title page and Overview to explain that this guide is the user guide for PCI SSD4 and also the addendum for SSD4 firmware with other EDT boards.
20130913	next	PH,MM	All	• Added Revision Log. • Changed "Xilinx" to "FPGA" throughout. • Repaginated with continuous arabic numerals from title page to end • Updated formatting and content (tables, part numbers, text, etc.). • Revised or deleted most references to test program `xtest` (for 1-channel operation) and FPGA configuration file `xtest.bit`. For general testing, users should use only `sslooptest` (the program for multi-channel operation), which checks for 1-channel operation and, if it is detected, automatically runs `xtest` and loads `xtest.bit` if needed. So users need not run `xtest`; they need only to recognize the related test output.
20070515	03	LW	All	• Made general updates.
2000-2007	00-03	SV/LW	All	• Created and updated this guide.