CENTIPEDE-PCI CNC Interface Board
GPIO Firmware

KSI Labs, LLC

2011

COPYRIGHT

Disclaimer

Information in this manual is protected by copyright laws and is the property of **KSI Labs, LLC**. Changes to specification and features in this manual may be made by **KSI Labs, LLC** at any time without prior notice. For product-related information and latest versions check our web site:

http://www.ksilabs.com

Contents

Chapter 1

HARDWARE

CENTIPEDE-PCI is a PCI-2.2 compatible interface card for CNC control applications. It may be used for other purposes as well but it was designed to control a CNC machine when used with optional breakout board, CENTIPEDE-BRK. It is universal PCI Add-In card i.e. it will work in both 3.3V and 5V PCI/PCI-X slot.

CENTIPEDE-PCI (simply PCI from now on) board is designed in the way that allows for maximum flexibility and extensibility. Entire functionality is implemented in infinitely reprogrammable Altera MaxII © CPLD so all board's hardware is actually an HDL code. That code can be changed and programmed into the CPLD thus allowing for easy bug fixes, new functionality addition, and ultimately for making it into something totally differerent. All CPLD source code (in VHDL) is available for free from KSI Labs, LLC web site. New releases, extensions, user-contibuted add-ons etc. will be also available from our web site as well as precompiled CPLD images (*.pof files) that can be directly written into the CPLD.

All external connections are made through a single 100-pin connector. There are 32 galvanically isolated dedicated inputs, 32 Open Drain dedicated outputs, 2 LVDS inputs and 10 LVDS outputs (all galvanically isolated.) LVDS I/O is supposed to implement SPI-like interfaces to external devices with up to 1MHz clock rate. Some of them used for communication with DAC/ADC peripherals on the breakout board (BRK from now on,) some are free for future use with add-on extension boards.

ALL external I/O is designed as galvanically isolated; there is no provision for a direct galvanic connection to the PCI board. Open Drain outputs are supposed to drive optocouplers so no ground connection is provided. There is +5V power from the external connector and external devices' optocouplers are connected between that +5V power and OD outputs. In the similar fashion all input optocouplers' LED anodes are connected together and routed to the external connector for connection to *external* +5V power and their cathodes are supposed to be connected to that *external* 5V ground to pass a signal to the PC.

LVDS I/O is also galvanically isolated with ISO72xx digital isolators. Those have 2 separate power supplies; the PCI board side is powered from the PC and connected to CPLD pins while the external side is powered from external 5V power. There is no galvanic connection between those two sides.

8 of 32 digital inputs use high speed FOD053L optocouplers with 1 μS propagation time for time-critical signals. Remaining 24 use regular MOCD207M devices with 3 μS propagation time.

1.1 Connectors

1.1.1 JTAG

Connector ***J2*** in the center of the board is a standard Altera JTAG connector for programming the CPLD. It is fully compatible with standard Altera programming tools (ByteBlasterII, USB-Blaster etc.) This connector is keyed to prevent from inserting the programming tool plug a wrong way. Other than that there is nothing more to say about it. CPLD programming is fully documented in QuartusII Web Edition software available for free from Altera web site. There is also a step-by-step programming procedure description in "*Setting up CENTIPEDE board set for Mach3*" document that is also available from our web site. Here is the schematics fragment with JTAG connector:

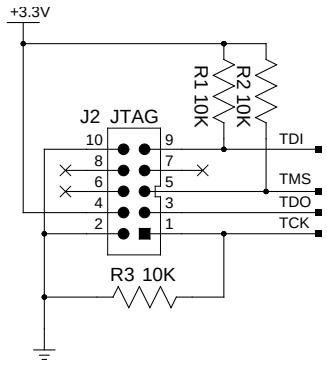

Figure 1.1: JTAG Connector

1.1.2 External Connector

External connector is high quality 100-pin N102A0-52E2PC connector from 3M. All I/O signals come from this connector. Special cable is used for connecting PCI to BRK board. It is 1:1 cable and it is made to order when ordering PCI/BRK boards from **KSI Labs, LLC**. Cable length can be up to 20 ft. according to the customer's specification.

Here is the external connector pinout and signals description:

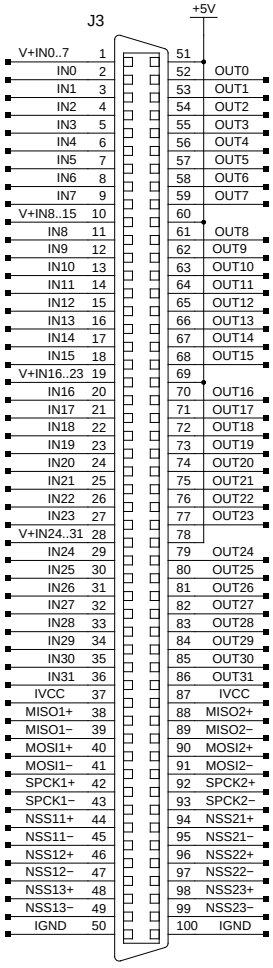

Figure 1.2: External Connector

IN0..31 Input optocouplers cathodes. Connect them to external power negative (or ground) rail to set input to '1' level. IN0 is bit 0 of DATA_IN register.

V+IN0..7-V+IN24..32 Input optocouplers anodes in groups of 8. They should be connected to the positive rail of an external power supply. There are series 330 Ω resistors installed for each optocoupler so there is no need for additional resistors if 5V external power is used. Those are usually powered from BRK board 5V power supply. 4 separate pins are used to spread the load between 4 wires.

OUT0..31 Open Drain outputs for driving external optocouplers. There is ***NO*** common ground connection on the external connector from PCI board

version 1.1 and up. These outputs ***MUST*** drive external optocouplers only to provide galvanical isolation. They should be connected to external optocouplers LED cathodes. Anodes of those LEDs should be connected to +5V outputs (pins 51,60,69,78.) Series resistors are required for external optocouplers. They should be designed for 5V operation. OUT0 is bit 0 of DATA_OUT register.

+5V Power for external optocouplers on OUT0..31 lines. This is taken directly from PCI bus and should ***NOT*** be used for anything else (there is no ground connection anyway.) There is 4 pins for this power to spread the load between 4 wires. OUT0..31 OD outputs make ground connection for this power rail when corresponding DATA_OUT register bits set to '1.'

MISO1+/- SPI1 data input differential pair. Not accessible directly from application software.

MOSI1+/- SPI1 data output differential pair. Not accessible directly from application software.

SPCK1+/- SPI1 clock differential pair. Not accessible directly from application software.

NSS1[1..3]+/- SPI1 chip select differential pairs. Not accessible directly from application software.

MISO2+/- SPI2 data input differential pair. Not accessible directly from application software.

MOSI2+/- SPI2 data output differential pair. Not accessible directly from application software.

SPCK2+/- SPI2 clock differential pair. Not accessible directly from application software.

NSS2[1..3]+/- SPI2 chip select differential pairs. Not accessible directly from application software.

IVCC External +5V power for machine side SPI interface. Usually connected to BRK board 5V power supply.

IGND External 5V power supply ground for machine side SPI interface. Usually connected to BRK board ground.

1.1.3 Extension Connector

Extension connector is used for connecting add-on boards. It is ***NOT*** isolated from PC so add-on boards implementing external interface to a machine must provide their own galvanic isolation.

There is no particular add-on interface implemented for this connector as of right now (firmware v.1.0) so it is made in user accessible GPIO for a time being. That will change when add-on boards are out.

Here is the connector pinout and signals description:

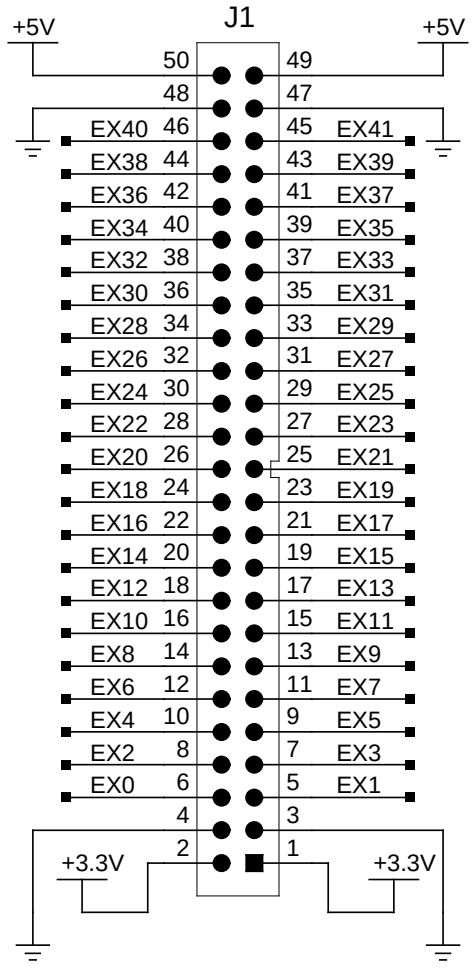

Figure 1.3: Extension Connector

EX0..41 GPIO signals. EX0..31 lines are controlled by EX0-31 register (EX0 is bit 0) and EX32..41 lines are under control of EX32-41 register (EX32 is bit 0.) This is a default assignment for GPIO firmware v.1.0 and may change in future versions when additional add-on boards are out.

Power Power rails are self-explanatory. They are supposed to power add-on boards. +5V, +3.3V, and ground all come directly from PCI bus.

Chapter 2

FIRMWARE

CENTIPEDE-PCI board is a set of different buffers/optocouplers/isolators and one big CPLD. The entire functionality is implemented in the CPLD; all other components are just simple interface components not implementing any logical functions. That means that functionality may by radically changed by programming CPLD with different firmware. Such design allows for easy hardware bugfixes, almost unlimited flexibility, new functionality addition by simply reprogramming the CPLD etc.

It is necessary to understand that CPLD code implements ***HARDWARE*** unlike some code for an embedded microcontroller that implements *FIRMWARE*. The principal difference is that CPLD code is actually a table of interconnects between different basic hardware blocks that CPLD is made of. In other words it is a bunch of wires and instructions where each wire is connected. Microcontroller firmware, on the other hand, is a *PROGRAM* i.e a set of instructions that microcomtroller fetches from memory and executes one-by-one. That means that firmware is always slower because every action is usually a sequence of instructions. Another firmware disadvantage is that MCU (MicroController Unit) can only execute a single instruction at a time (actually there are MCUs that are able of executing several instructions at a single step but that is not a regular case and there are other limitations) so it can not act fast enough on several different tasks, it can get into an infinite loop in one execution branch and all other tasks will get suspended indefinitely and there are other issues with such approach. In CPLD, on the other hand we can implement several different units that are working in parallel totally independent of each other.

There is another fundamental difference between CPLD and MCU–there is no program running in CPLD. The interconnection table is loaded from internal FLASH-like memory only once (usually on powerup) and then it is pure hardware operation. It is not bootup like it is in MCU where the initial program is loaded and then that program executes for the entire time the MCU is operating; it is a one-time ***CONFIGURATION*** that only executes once.

That does not mean that CPLD can not execute some action sequences but those sequences are purely hardware ones. Machine gun also performs some

sequence of actions when trigger is pulled but there is no software program in it. MCU counts pulses by incrementing some variable while CPLD implements it as a string of triggers changing their states.

Yet another difference is that unlike MCU CPLD does not have its pin functions preassigned to particular pin (except a few such as power/ground pins or JTAG pins for initial programming.) Almost any signal can be connected to almost any CPLD pin upon initial configuration that makes PCB design much easier because one can reassign signals to different pins if it makes PCB layout easier. There are some limitations of course but they are not all that strict.

There are different ways to make that CPLD (Complex Programmable Logic Device) configuration table. One can use a pure schematic approach by drawing schematics with special CAD software and then it is translated to the particular CPLD device configuration image by special "compiler." This is the most precise way but it is cumbersome and not actually suitable for bigger and more complex designs. Another way is using some kind of HDL (Hardware Description Language) that describes how the hardware is supposed to operate. Than such a description is processed by a set of CAD tools that synthesize a schematic implementation of the described behavior. This way is easier to work with, better suited for big complex designs, more maintainable and more portable between different devices. Here at KSI Labs, LLC we use one of HDL languages, VHDL for CPLD design. The entire VHDL source for CENTIPEDE-PCI board CPLD is available for free from our web site so everybody can customize our board as he sees fit and/or change/extend its functionality.

So strictly speaking "FIRMWARE" is not a right word to call the CPLD configuration but we will be using it for the lack of better one.

This chapter does NOT describe how to configure the CENTIPEDE set of boards for use with particular software (e.g. Mach 3;) it is the description what the board looks like from a programmer's standpoint so he can write his own software for it. Please note that "firmware" may change at every moment so please visit our web site, `http://www.ksilabs.com` for the latest information.

The included information describes CENTIPEDE GPIO firmware version 1.0 that is available from our web site as a time of writing this document. Step-by-step instruction on how to program it in the CPLD is available in "*Setting up CENTIPEDE board set for Mach3*" document that is also available from our web site.

2.1 PCI Device Registers

CENTIPEDE-PCI is a regular PCI board fully conforming to PCI 2.2 standard. It has one 4Kbyte PCI Memory BAR that is initialized for proper system memory mapping by PC BIOS. VendorID is 0xFEFF ***(That might be changed to KSI Labs, LLC VendorID when it is obtained.)*** DeviceID is 0x0008, ClassCode 0x078000. All register offsets are from BAR0 base. Registers are 32-bit and ***MUST*** be accessed with 32-bit instructions. Write operations other that 32-bit will have unpredictable results and probably will lead to faulty op-

eration and computer crash.

Here is the register map:

Register	Offset	Description
DATA_OUT	0x00	Output Data Register, R/W
DOUT_SET	0x04	Output Data Bits Set Register, W/O
DOUT_CLR	0x08	Output Data Bits Clear Register, W/O
DATA_IN	0x0c	Input Data Register, R/O
DEBOUNCE	0x10	Input Data Debounce Register, R/W
IRQ_MASK	0x14	Input Data IRQ Mask Register, R/W
IRQ_SENSE	0x18	Input Data IRQ Sense Register, R/W
IRQ_EDGE	0x1c	Input Data IRQ Edge Register, R/W
IRQ_BOTH	0x20	Input Data IRQ BothEdges Register, R/W
IRQ_RAW	0x24	Input Data Raw IRQ Status Register, R/C
IRQ_STAT	0x28	Input Data Masked IRQ Status Register, R/O
DAC_A	0x2c	DAC A Data Register, R/W
DAC_B	0x30	DAC B Data Register, R/W
ADC_0	0x34	ADC 0 Data Register, R/O
ADC_1	0x38	ADC 1 Data Register, R/O
ADC_2	0x3c	ADC 2 Data Register, R/O
ADC_3	0x40	ADC 3 Data Register, R/O
IRQ_RAW2	0x44	Other Events Raw IRQ Status Register, R/C
IRQ_MASK2	0x48	Other Events IRQ Mask Register, R/W
IRQ_STAT2	0x4c	Other Events Masked IRQ Status Register, R/O
EX_DATA_LOW	0x50	Extension Bus Data, Bits 0..31, R/W
EX_DATA_HI	0x54	Extension Bus Data, Bits 32..41, R/W
EX_DIR_LOW	0x58	Extension Bus Direction, Bits 0..31, R/W
EX_DIR_HI	0x5c	Extension Bus Direction, Bits 32..41, R/W

Table 2.1: CENTIPEDE-PCI Register Map

2.2 Registers Description

2.2.1 DATA_OUT, Offset 0x00

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.2: DATA_OUT Register

D0..31 Output Data, R/W. All data written to this register is immediately reflected on External Connector OUT0..31 pins. '1' will make the output FET to open i.e. the output will go ***LOW***. In other words writing '1' to a particular bit will turn the corresponding external optocoupler LED to ON. Read operation will give the current actual output register state. Please note that this state can be changed not only by writing to the DATA_OUT register but also by setting/resetting individual bits via DOUT_SET/DOUT_CLR registers. Defaults to all '0' on powerup.

2.2.2 DOUT_SET, Offset 0x04

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.3: DOUT_SET Register

D0..D31 Output Data Set, W/O. Writing '1' to a bit will set the corresponding bit in DATA_OUT register to '1.' Writing '0' has no effect i.e. the corresponding bit is not changed. As DATA_OUT register outputs are directly connected to OUT0..31 connector pins all changes take immediate effect.

2.2.3 DOUT_CLR, Offset 0x08

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.4: DOUT_CLR Register

D0..31 Output Data Clear, W/O. Writing '1' to a bit will set the corresponding bit in DATA_OUT register to '0'. Writing '0' has no effect i.e. the corresponding bit is not changed. As DATA_OUT register outputs are directly connected to OUT0..31 connector pins all changes take immediate effect.

2.2.4 DATA_IN, Offset 0x0c

D31	D30*	D29	D28*	D27	D26*	D25	D24*
D23*	D22	D21*	D20	D19*	D18	D17*	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.5: DATA_IN Register

D0..31 Input Data, R/O. IN0..31 External Connector data state, either directly from connector pin (through an optocoupler) or after debouncing circuitry depending on corresponding DEBOUNCE register bit. The same rule as for DATA_OUT register applies–'1' means there is current through the input optocoupler LED (switch connected to the corresponding BRK board terminal is ***CLOSED***,) '0' means optocoupler LED is OFF. Bits marked with '*' have fast optocouplers on inputs on rev.1.0 CENTIPEDE-PCI board. Starting from rev.1.1 fast optocouplers are on bits 24..31. Input changes are reflected immediately on raw (not debounced) bits. Debounced inputs have '0'→'1' transitions reflected immediately while '1'→'0' transitions are delayed until contact bounce stops.

2.2.5 DEBOUNCE, Offset 0x10

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.6: DEBOUNCE Register

D0..31 Route input data through hardware debouncer, R/W. If a bit is set to '0' External Connector pin is connected directly to corresponding DATA_IN register bit (through an optocoupler.) If a bit is set to '1' hardware debouncer is inserted between optocoupler and DATA_IN bit. Debouncer works by setting its output to '1' immediately once optocoupler LED is energized and start sampling input every 50 mS. When 3 consecutive samples are all '0' it sets its output to '0.' Defaults to all '0' on powerup.

2.2.6 IRQ_MASK, Offset 0x14

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.7: IRQ_MASK Register

D0..31 Input Data IRQ Mask, R/W. Setting a bit to '1' will make a corresponding bit in IRQ_RAW register to generate an IRQ to the PCI bus when there is an interrupt condition (IRQ_RAW bit is set to '1'.) Defaults to all '0' on powerup.

2.2.7 IRQ_SENSE, Offset 0x18

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.8: IRQ_SENSE Register

D0..31 Level/Edge IRQ Sense, R/W. Setting a bit to '1' will make corresponding IRQ_RAW register bit to go '1' on corresponding DATA_IN register LEVEL. Setting it to '0' will make that bit EDGE sensitive. What level or edge is used depends on IRQ_EDGE and IRQ_BOTH registers settings. Defaults to all '0' on powerup.

2.2.8 IRQ_EDGE, Offset 0x1c

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.9: IRQ_EDGE Register

D0..31 Polarity select for DATA_IN register change to generate interrupt, R/W. When a bit is set to '1' interrupt is generated on HIGH level or RISING edge of the corresponding DATA_IN register bit depending on

IRQ_SENSE bit value otherwise opposite edge/level is used. Has no effect if IRQ_SENSE bit is set to '0' (EDGE) and corresponding bit in IRQ_BOTH register is set to '1'. Defaults to all '0' on powerup.

2.2.9 IRQ_BOTH, Offset 0x20

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.10: IRQ_BOTH Register

D0..31 Generate interrupt on BOTH DATA_IN register bit edges, R/W. If a bit is set to '1' and corresponding IRQ_SENSE bit is set to EDGE interrupt will be generated on every change of the corresponding DATA_IN bit. Has no effect if IRQ_SENSE bit is set to LEVEL. Defaults to all '0' on powerup.

2.2.10 IRQ_RAW, Offset 0x24

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.11: IRQ_RAW Register

D0..31 Raw DATA_IN-generated interrupt status, R/Clear. Bits are set to '1' on DATA_IN bits changes based on IRQ_SENSE, IRQ_EDGE, and IRQ_BOTH bits. '1' is sticky so it won't go away by itself once set. Writing '1' to a bit clears it and it stays at '0' until the next qualifying event. Writing '0' has no effect. For a bit to generate actual interrupt to the PCI bus the corresponding bit in IRQ_MASK register must be set to '1.' Otherwise this reqister can be used to monitor DATA_IN state/activity by polling it. Reset on powerup so it starts with all '0.'

2.2.11 IRQ_STAT, Offset 0x28

D31	D30	D29	D28	D27	D26	D25	D24
D23	D22	D21	D20	D19	D18	D17	D16
D15	D14	D13	D12	D11	D10	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.12: IRQ_STAT Register

D0..31 Masked IRQ status, R/O. Shows which bit generated actual IRQ to the PCI bus. Read-Only, any bit set to '1' causes an IRQ. Bits are generated as logical AND between IRQ_MASK and IRQ_RAW registers. Can only be reset by writing '1' to the corresponding IRQ_RAW bit or setting the corresponding bit in IRQ_MASK to '0.' No defaults but comes up as all '0' because IRQ_MASK is set to all '0' on powerup.

2.2.12 DAC_A/DAC_B, Offsets 0x2c and 0x30

X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.13: DAC_A/DAC_B Registers

D0..9 Data for DAC A / DAC B on BRK Board, R/W. 10 lower bits written to these registers are transferred to a 0-10V output DAC (A or B) on BRK board using SPI-like interface. The entire State Machine for this interface is implemented in hardware so there is no need to wait for something; these registers can be read or written at any time. However because it is NOT a parallel interface actual DAC output is NOT set to the corresponding voltage immediately. It might take up to 100 μS for such write to get propagated to the DAC output i.e. maximum update frequency is 10 KHz. It is usually fast enough for all practical purposes and DAC is setup much less frequently so there is no reason to monitor it. However if maximum update frequency is required software should poll IRQ_RAW2 register or set an interrupt on its bit by setting the corresponding IRQ_MASK2 bit. Each write to one of DAC registers clears appropriate bit in IRQ_RAW2 register and that bit goes back to '1' once that written data is transferred to the DAC.

X Don't care on write, read as '0.'

2.2.13 ADC_0..3, Offsets 0x34, 0x38, 0x3c, 0x40

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	D9	D8
D7	D6	D5	D4	D3	D2	D1	D0

Table 2.14: ADC_0..3 Registers

D0..9 Data from ADC 0..3 on BRK board, R/O. 10 lower bits is data from 0-10V full scale ADC channels on BRK board. They are transferred through SPI-like interface. The entire State Machine is implemented in hardware so they can be read at any time. However because it is NOT a parallel interface and it is ONE ADC with 4 input multiplexer, NOT 4 separate ADC, data is not guaranteed to be updated since the last reading if it was not long enough time since then. In this case registers will still have the old value. It might take up to 1 mS for an ADC channel value to get updated because they are read sequentially in a loop i.e. update frequency is guaranteed to be not less that 1 KHz (that is ALL channels are updated at least 1,000 times per second.) If maximum update frequency is required software should poll IRQ_RAW2 register or setup an interrupt on its corresponding bit by setting appropriate IRQ_MASK2 register bit to '1.' Each read clears appropriate bit in IRQ_RAW2 and that bit goes back to '1' once ADC reading is updated. NOTE: Although remaining bits are always read as '0' these registers also must be read with 32 bit instructions.

2.2.14 IRQ_RAW2, Offset 0x44

BRK_ST	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	FAULT	ADC3	ADC2	ADC1	ADC0	DACB	DACA

Table 2.15: IRQ_RAW2 Register

DACA/DACB Data written to DAC_A/DAC_B register has been transferred to the corresponding DAC at least once. These bits are reset by writing to the corresponding DAC data register.

ADC0..3 Corresponding ADC data register has been updated with fresh data at least once. These bits are cleared by reading corresponding ADC data registers.

FAULT Communication with the BRK board has been broken at least for 1 mS. That bit is only set if a succesfull communication was established before i.e. it will not be set if such communication was never established. Write '1' to this bit to clear.

BRK_ST BRK board status. Not actually an interrupt per se but can be used as one. This bit is constantly updated unlike other IRQ bits i.e. it will go to '1' when communication is established, reset to '0' when it is lost, and go back to '1' when it is reestablished. ADC State Machine makes 5 cycles, not 4. There is 4 cycles, one for each ADC channel and one additional cycle reading self-calibration value in each and every ADC loop iteration. If the self-calibration value is right the status is considered OK and this bit is set to '1,' otherwise it is set to 0. When it happens BRK_LOST bit is set to '1' and all other bits are not updated until connection is reestablished. The reason for all that trouble is that BRK board is powered from a separate power supply and it might be turned off at any time (or turned ON at later time as a part of the CNC machine powerup when the PC is already on and CNC software is already running.)

2.2.15 IRQ_MASK2, Offset 0x48

BRK_ST	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	FAULT	ADC3	ADC2	ADC1	ADC0	DACB	DACA

Table 2.16: IRQ_MASK2 Register

X Don't care. Those bits can be written to and read back but they don't have any effect for now. They might be used in the future when additional functionality is implemented and add-on boards are out.

Rest Remaining bits are used to mask corresponding IRQ_RAW2 interrupt bits. Default to all '0' on powerup so no IRQs to PCI bus is generated.

2.2.16 IRQ_STAT2, Offset 0x4c

BRK_ST	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	FAULT	ADC3	ADC2	ADC1	ADC0	DACB	DACA

Table 2.17: IRQ_STAT2 Register

Not0 Those R/O bits show what particular event caused a PCI bus interrupt. Resettable by writing to DAC data registers, reading ADC data registers, writing '1' to FAULT bit of IRQ_RAW2 register, and clearing BRK_ST bit in IRQ_MASK2 register. Powers up with all '0.'

2.2.17 EX_DATA_LOW, Offset 0x50

EX31	EX30	EX29	EX28	EX27	EX26	EX25	EX24
EX23	EX22	EX21	EX20	EX19	EX18	EX17	EX16
EX15	EX14	EX13	EX12	EX11	EX10	EX9	EX8
EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0

Table 2.18: EX_DATA_LOW Register

EX0..31 Extension Bus, Low part, R/W. This is Extension Connector I/O pins bits. If corresponding DIR bit is set to '1' in EX_DIR_LOW register pin is configured as output. Data written to that register bit immediately appear on the Extension connector pin (here '1' is HIGH level and '0' is LOW; no optocouplers, just straight connection from the register to the pin) and can be read back. If the corresponding DIR bit is set to '0' pin is input and register bit reads as corresponding Extension Connector pin logic state. Writes have no effects when bit is input.

2.2.18 EX_DATA_HI, Offset 0x54

X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	EX41	EX40
EX39	EX38	EX37	EX36	EX35	EX34	EX33	EX32

Table 2.19: EX_DATA_HI Register

EX32..41 Extension Bus, High Part, R/W. All written for the low part above apply.

X Don't care. No effect on write, read back as '0.'

2.2.19 EX_DIR_LOW, Offset 0x58

EX31	EX30	EX29	EX28	EX27	EX26	EX25	EX24
EX23	EX22	EX21	EX20	EX19	EX18	EX17	EX16
EX15	EX14	EX13	EX12	EX11	EX10	EX9	EX8
EX7	EX6	EX5	EX4	EX3	EX2	EX1	EX0

Table 2.20: EX_DIR_LOW Register

EX0..31 Extension Bus Direction, Low part, R/W. Extension Connector I/O pins direction. '1' is output, '0' input. All '0' on powerup.

2.2.20 EX_DIR_HI, Offset 0x5c

X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	EX41	EX40
EX39	EX38	EX37	EX36	EX35	EX34	EX33	EX32

Table 2.21: EX_DIR_HI Register

EX32..41 Extension Bus Direction, High part, R/W. Extension Connector I/O pins direction, high part. '1' is output, '0' input. All '0' on powerup.

X Don't care. No effect on write, read back as '0.'

Chapter 3

Appendix A - Hardware differences between rev.1.0 and rev.1.1

There are some differences between rev.1.0 and rev.1.1 CENTIPEDE-PCI boards. They are not all that significant but there some caveats.

- First of all, there were 4 unpopulated footprints for optional oscillator on rev.1.0 board (elements R4, C15, U2, and R5) close to the boards center. Those were removed completely from rev.1.1 PCB. There is absolutely no impact because those elements were never used.
- Rev.1.0 board has 4 out of 8 fast optocouplers on different input lines (see section 2.2.4.) It is not very likely somebody would use that feature but if he would one should take care to assign data lines properly with application software (e.g. configuration editor or whatever it's called) for his particular board revision.
- As a result of the above CPLD project file differs between those 2 revisions. VHDL code is absolutely identical between the two but there are some changes in pin assignments in centipede_pci.qsf file and resulting centipede_pci.pof file is different. Changes are minimal but please be careful anyway. All future CPLD firmware will be released in versions for both rev.1.0 and rev.1.1 PCI boards. There is no differences between those revisions from software point of view.
- ***IMPORTANT!*** Rev.1.0. PCI board had PCI bus ***GROUND*** connected to pins 69 and 78 of the External Connector, NOT +5V power. It had 2 +5V power pins (51 and 60) and 2 GROUND pins (69 and 78.) On rev.1.0 CENTIPEDE-BRK board pins 69 and 78 of the mating connectors are ***NOT CONNECTED*** to anything so it works OK with rev.1.0

CENTIPEDE-PCI board. Rev.1.1 PCI board will also work just fine with rev.1.1 BRK board though 2 +5V conductors in the cable will not be used. ***But rev.1.0 PCI board can NOT be used with rev.1.1 BRK board because the latter has pins 51, 60, 69, and 78 all connected together that will make a short circuit between PC +5V and ground.***

- Other than that those boards absolutely identical and fully compatible with each other.