Setting up CENTIPEDE board set for Mach3

Firmware V3.0.1

KSI Labs, LLC

2013

COPYRIGHT

Disclaimer

Contents

Chapter 1

HARDWARE

1.1 General

The full hardware set consists of CENTIPEDE-PCI (PCI from now on) controller board with Mach3 firmware, CENTIPEDE-BRK (BRK from now on) breakout board, and special connecting cable to connect them to each other. CENTIPEDE-PCI should be installed in any free PCI slot of your computer running Mach3. BRK is going into some closed grounded electric box on the machine, probably together with Stepper/Servo drives and other electronics.

BRK use is optional but highly recommended. You can make your own breakout board if you want or need to–there is enough information on our web site, `http://www.ksilabs.com` to design and build one–but it is way more complex that usual boards for Parallel Ports and it is unlikely such a board would be cheaper than ready-made CENTIPEDE-BRK if made in single quantities less for all the time spent on designing and manufacturing one. But anyways you can try to make your own, its OK. To make such a task easier full schematics are included in CENTIPEDE-PCI and CENTIPEDE-BRK manuals.

Our Mach3 driver also grabs all the parallel ports on the PC if they are not used for anything else in Windows and allows for using their I/O pins together with those provided by BRK. Up to 2 parallel ports supported and each of them can be made into "mostly input" or "mostly output" configuration. That essentially allows to have a separate dedicated I/O pin for each and every Input and Output signal supported by Mach3; you do not have to do any sacrifices (like paralleling switches etc.) any more. If parallel ports are used you will need a parallel port breakout board for their signals. There are plenty of such boards available for cheap.

Please note – ***ALL*** those signals are "fast" ones, not emulated or slow ModBus signals. Any of this signals can be used for actual machine control in real time.

There are 2 types of relays on the BRK board – *POWER* and *SIGNAL*. POWER relays have SPDT contacts and are capable to switch up to 12A @

125VAC and up to 7A @ 250VAC. They are suitable for direct switching of cooling pumps, machine lights etc.

SIGNAL relays have DPDT contacts so they give more flexibility but they are NOT designed to switch high currents. Their contacts are good for 0.5A tops and their purpose is to switch low level logical or control signals like various drive enables, VFD signals etc.

Relays can be connected to BRK output signals or disconnected from them so those outputs can be used to drive some optocouplers directly. This is configurable with 8-position DIP switch SW1 on the BRK board. Switch mapping to particular relays/outputs is described in BRK manual.

1.2 Optocouplers and galvanic isolation, CONNector Board

Those knowledgeable in electronics probably can skip this section because they should already knew what the galvanic isolation and optocouplers are. This material is intended for those who are not electronic engineers by trade to give them a brief introduction on the subject. Hopefully this will help to understand the importance of galvanic isolation and let them avoid costly mistakes of creating improper circuits when building their machines.

There is a section on this in "Mach3 CNC Controller Software Installation and Configuration" manual from ArtSoft, namely 4.2 and 4.2.1 that also explains this subject and it is highly recommended for reading too. We would like to reiterate some points and maybe provide some additional information that would help to understand how to build a robust reliable machine control system and why it should be done this way. This is a very brief and grossly oversimplified introduction and we didn't want to be very scientific so futher reading is suggested if one wanted to get more information. What we want to explain is the very basics that should be sufficient for building your CNC system.

Let's start with logic signals. Almost all control signals used by Mach3 and CENTIPEDE are logic ones. They can only take 2 different states, '0' and '1.' It is "either motor is on or off," "switch is open or closed" etc. Mach3 works with ones and zeroes, it doesn't know anything about wires, volts, amperes etc. All commands from Mach3 to the machine are usually passed as electric signals and machine communicates its state back to Mach3 in the same manner. It is the task for the interface circuits–it is CENTIPEDE set in our case–to convert electric signals to ones and zeroes and back.

On physical level those electric signals are represented by different voltages. Voltage less then some threshhold is interpreted as one logic level, over that threshhold is the opposite logical level. We can treat low voltage as '0' and high voltage as '1' or vice versa; that does not matter for the subject of this section. What matters is voltage levels.

For signal voltages to be meaningful those signals must have some reference point to measure them from. That reference point must be common for signal

source (output) and its destination (input) so both have the same idea what the signal voltage really is. Usual TTL logic signal is considered "high" if it is over 2.4V and "low" when it is less than 0.8V. This will only work if the output and input has the same 0V reference point. If that point at the input is e.g. 2.5V higher then that at output such circuit will not work because even "high" level of 3V at the output will translate to 3V−2.5V=0.5V at the input and will be taken as "low."

To make this work those reference points should be all connected together so they have the same potential. That interconnected net of signal reference points is usually called "signal ground." It is important to understand that although it is called "ground" it does ***NOT*** have to be connected to the ground connector of the mains outlet of even to the appliance metallic case that is often grounded by connecting it to the mains ground wire. "Ground" noun in "signal ground" is just a figure of speach and it can be at any potential wrt the real ground. It is just the common reference voltage net for a group of signals. It is also important to understand that there might be ***SEVERAL*** different "signal grounds" for different groups of signals and those "grounds" can be either connected to each other or not.

In a regular PC computer all *LOGICAL* signals usually share the same signal ground. That makes things simpler and works fine because all computer parts are powered from the same power supply and live inside the same relatively small metall enclosure. That enclosure is usually grounded by connecting it to the mains ground terminal that effectively eliminates most of external electromagnetic noice influence. Some of external devices are also connected to a PC in such a way that their signal ground is connected to PC signals ground (parallel printers, modems, USB devices etc.) but that only works for devices that are physically close to the PC and located in sterile office environment. It is also done this way not because it is the right way but because it is cheaper and "works most of the time." Things change for devices that made the right way. E.g. probably everybody knows that ***EVERY*** network card has an isolation transformer on it. It is 1:1 transformer so its output voltage is the same as input and its only purpose is to ***ISOLATE*** the network from the PC electronics so no network conductors are *ELECTRICALLY* connected to the PC.

The reason behind this is simple – PC and network switch can be quite far from each other and their signal grounds can be at significantly different potentials. If we connect them with a wire it will result in faulty operation in the best case because of signal level shift described above. In the worst case it might result in mighty short-circuit fireworks and both PC and the switch fried and destroyed. That is why ***EVERY*** network port is connected to the rest of electronics via isolation transformer. Yes, there is 16 such transformers in 16-port network switch, one on each and every port and ***ALL*** ports are electrically (or *GALVANICALLY* as another name for a direct electrical connection) isolated from the switch electronics and from each other.

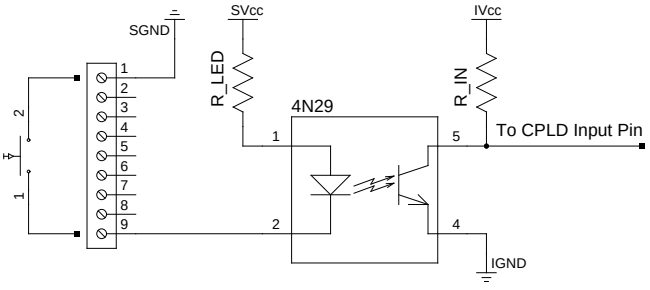
There is yet another reason for such isolation beyond those safety concerns. Even if both the switch and PC are actually at the same potential (e.g. they are connected to the same mains ground) there is a long wire connecting their

signal grounds. Any electromagnetic pulse or alternating magnetic field from nearby power cord etc. will generate some voltage in that wire; it will work as a transformer winding. That will turn our wire into voltage source so instead of connecting 2 signal grounds it will act as a voltage source connected between them thus causing a level shift that can mess with the signals and even damage both units in extreme cases. The higher the currents in nearby wiring the higher the voltage they generate. An it is not just currents – moving magnets or magnetic mass interacting with natural Earth magnetic field can also generate such voltage.

That is why it is ***BAD*** idea to connect anything beyond PC to its signal ground. It is absolutely no-no in any industrial environment such as big machine with lot of motors and other stuff generating tens of amperes of current, intense electromagnetic fields with plasma torches etc. Those machines ***MUST*** be electrically isolated from the PC for entire system to work.

Special device called ***optocoupler*** is usually used to provide galvanic isolation between different circuits while allowing to pass logical signals. It is a combination of a LED and phototransistor (there are other types but they are usually special purpose ones) in a same package. They are ***optically*** coupled and isolated electrically from each other. This way there is no path for electrical current from the LED to the phototransistor. Electrical insulation between the two is quite substantial, usually able to withstand 1,000V or more.

While electrically (or ***galvanically***) isolated from each other the LED-Phototransistor combination allows to pass a logical signal across this barrier. When LED is powered it emits light that forces the normally closed phototransistor to open i.e. go into a kinda short-circuit mode thus passing logical signal to other side. It is similar to a relay that switches its contacts when its coil energized but unlike relay there is no moving parts in optocoupler so it is way more reliable and much faster than electromechanical relay. Regular optocoupler is capable of 100-200KHz signal frequency, special high-speed ones are even faster.

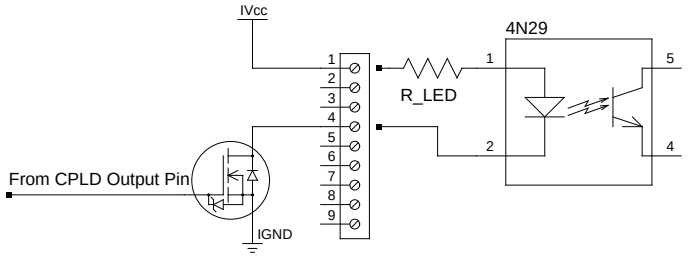


Here is how ***Inputs*** are connected. You can see input switch and one of 4 9-wire BRK terminals on the left. The topmost terminal is ***Machine-Side*** signal ground. *SVcc* is ***Machine-Side*** +5V power. They are connected to the BRK +5V power supply on the BRK. It is easy to see that current will flow from SVcc through R_LED and optocoupler LED to SGND if switch is closed. That

current will lit the LED. Light emitted by the LED will make the transistor to open that will short the R_IN resistor to IGND that is ***PC-Side*** signal ground. That will make it to go into "low" state. If the switch is open there will be no current through the LED and transistor will close that will disconnect R_IN from PC-Side ground and IVcc that is ***PC-Side*** logic power (that happens to be 3.3V) will make the input to go "high". As we can see there is no electrical contact between the machine an PC so they can be at any potential wrt each other and the system will still work without problems.

Most of the parts on this schematic are located on the PCI board. Optocouplers are usually located as close to signal ***Destination*** as it is possible. In our case CPLD on the PCI board is the destination. Terminals and ***Machine-Side*** power supply (SVcc/SGND) are located on the BRK, everything else is on the PCI board.

Now let's take a look on how ***Outputs*** are done.



CPLD output pin drives a FET on the left. That is connected to ***PC-Side*** ground. When CPLD outputs "high" level FET turns on i.e. shorts the output terminal it is connected to to ***PC-Side*** signal ground. That completes the circuit so current flows from a common IVcc that is ***PC-Side*** +5V power through R_LED and optocoupler LED to IGND that is ***PC-Side*** signal ground. 32 FETs (one per output signal) are located on the PCI board and IVcc comes right off of PCI bus. Terminals (one of 4 groups of 9 output terminals shown with only one FET out of 8) are located on the BRK (right vertical row.) R_LED and optocoupler is usually placed close to the signal ***Destination*** e.g. inside Stepper Drive.

So there are few single rules to remember. First, optocouplers are ***almost always*** placed close to signals ***Destination*** i.e. near corresponding ***Inputs.*** Second, optocoupler LED is ***always*** powered off of ***Output*** power supply and thus uses ***Output*** signal ground. Third, optocoupler transistor is ***always*** powered off of ***Input*** power supply and thus uses ***Input*** signal ground.

For CENTIPEDE kit that means ***ALL Inputs*** i.e. everything that goes ***from Machine to PC*** must be powered from ***Machine-Side*** power supply and thus use ***Machine-Side*** signal ground. BRK *Digital* input terminals (i.e. left vertical column of terminals on the BRK) are already setup this way so you only have to use ground coming from those terminals. +5V power supply on the BRK board is that ***Machine-Side*** power and its ground is connected to common ***Machine-Side*** signal ground. That power supply itself is available

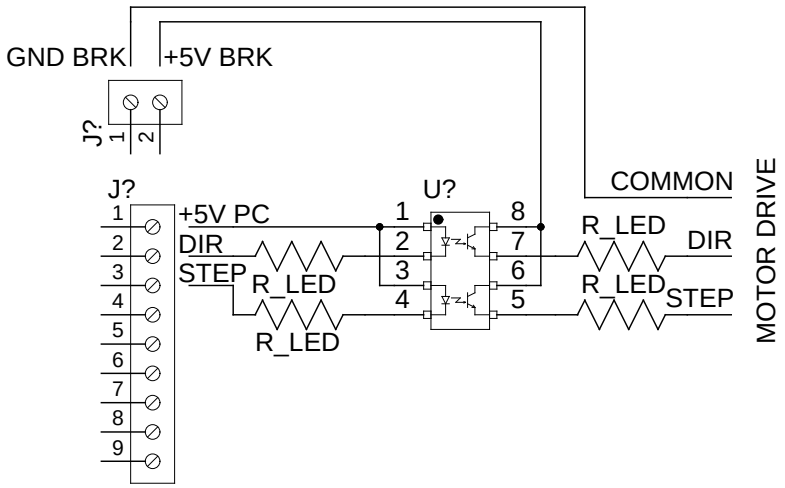
on a separate terminal (the rightmost 2-wire one in the top horizontal row) and can be used for powering other ***Machine-Side*** circuitry. That includes e.g. rotary and linear encoders or anything connected to an optional Parallel Port breakout board ***Inputs*** (signals ***from Machine to PC*** Parallel Port.) Under ***NO*** circumstances ***ANY*** of +5V, ground, or signal wire from that side should be connected to ***ANY BRK Digital Output*** circuit.

In the same fashion ***ALL Outputs*** i.e. everything that goes ***to Machine from PC*** must be powered from ***PC-Side*** power supply and thus use ***PC-Side*** signal ground. BRK *Digital* output terminals (i.e. right vertical column on the BRK board) are already setup this way so you only have to use +5V coming from those terminals for Stepper Drive etc. inputs. Under ***NO*** circumstances ***ANY*** of +5V or signal wire coming from those ***Output*** terminals should be connected to ***ANY BRK Digital Input*** circuit or that +5 V power supply on the BRK.

There is one special case–common cathode optocouplers on some drives. It is not clear what their designers were thinking when designing them that way but anyways they do exist. There are too many reasons why it should be ***NEVER EVER*** done this way but we will not list them all here trying to keep this document under thousand pages...

Those drives have common GROUND terminal for their DIR/STEP/whatever inputs instead of usual +5V or so and their inputs must be ***DRIVEN*** with +5V (or some other voltage) instead of shorted to ground. One notable example of such drive is Gecko G203V but it is probably not the only one out there...

Here is the proper way to connect such drives to CENTIPEDE BRK.



Top connector here is BRK topmost right 2-pin terminal providing ***Machine-Side*** +5V power. Dual optocoupler shown on the schematic but any single, dual, or quad optocouplers can be used here. It is better to use ones with 100% CTR (Current Transfer Ratio) but anything will work if motor drive's input optocouplers are sensitive enough (2.5mA or so.) Gecko's G203V are. Left pair

of R_LED resistors is mandatory. Right pair, going from optocouplers to motor drive is only required if the drive does not have its own resistors (almost all of them do have those, including G203V, so no resistors needed, just connect drive's inputs directly to the optocouplers.) Any resistor value beween 330 and 680 Ohm should work fine here.

For those who might find this too complex to do it theirselves there is a small cheap board available from our site, `http://www.ksilabs.com` named CENTIPEDE-CONNector. This board has 8 channels of optocoupled "inverters" that is enough for connecting motor drives for 4 axes. If you need more just use 2 boards.

The CENTIPEDE-CONNector also provides all hardware for connecting 2 Slotted Optical Switches to CENTIPEDE-BRK. It has Schmitt Triggers on both inputs that eliminate noise and uncertainty from Optical Switches signals thus providing a clean signal to the CENTIPEDE system. Those switches can be used as Spindle Index/Timing sensors or as Home switches with very good repeatability and robustness.

For Spindle Index (and/or Timing) you will have to provide a slotted disk or a narrow flag on your Spindle and matching Slotted Optical Switch. Those switches are essentially an IR LED (called "***Emitter***") and Phototransistor (called "***Sensor***") installed in some kind of plastic enclosure. There is a slot in that enclosure for some kind of light interruptor. Emitter and Sensor are installed at the opposite sides of that slot facing each other so the IR light emitted by the Emitter lights up the Sensor making it to open i.e. to conduct current. When some opaque object is inserted into that slot it will block (or interrupt) the light path that will make the Sensor to close i.e. to *STOP* conducting current. That makes 2 distinctive states that are translated to electrical signals CENTIPEDE understands by that CONNector board. Sensor signals are routed through Schmitt triggers that remove noise from them and amplify those signals so they can be used to drive the optocouplers going to the CENTIPEDE-BRK.

Unlike original Mach3 Parallel Port driver CENTIPEDE processes Spindle Index/Timing signal in hardware that samples it approximately 33 million times per second so there is no need to make those signals very long. The shorter the signal is the better it works for CENTIPEDE. That means there is no need for ***Slotted*** disc with a sector slot for those sensors. The optimal configuration is a solid opaque disc with a single e.g. 1/8" (3mm) hole close to its outer edge. Sensor is installed in such a way that disc goes all the way into its slot thus blocking the light path inside the sensor. The hole is located at such distance from the outer edge that it covers the sensor aperture when the disc is at appropriate angle thus momentarily ***OPENING*** the light path that will generate a short Index pulse to the CENTIPEDE. The resulting pulse will have ***NORMAL*** polarity i.e. Index signal should be configured as "*Active High*" ("Active Low" unchecked) in Mach3.

An alternative approach can also be used. In this case a narrow (slightly wider than Slotted Optical Switch width) opaque flag or protruding pin can be used that will momentarily ***BLOCK*** the light path. That will require setting Index signal to "***Active Low***" ("Active Low" ***CHECKED***) in Mach3 configu-

ration. No other changes necessary and this will work exactly like a slotted disc configuration.

Spindle Timing, if used, is implemented the same way with an additional slotted disc. Timing disc should have not one but several holes located at even angle increments (e.g. 10 holes located every 36 degrees) to generate several pulses per revolution. Other than that it is identical to the Index setup. Alternative configuration with something like e.g. "Daisy Wheel" will also work.

Home switches are implemented in a similar fashion. They can use both blocking and opening of the light path at their trip point; either one would work. Just don't forget to setup Mach3 signal polarity appropriately (signal goes ***HIGH*** when light path is open.)

No sensors are supplied with CENTIPEDE-CONNector boards. It simply doesn't make sense for us to purchase those and sell them to our clients – we don't know which sensor would be appropriate for each parcticular machine setup and if they are required at all (e.g. when those boards are only used for interfacing to "inverted" motor drives.) Those *Slotted Optical Switches* are readily available from all major electronic components distributors for something like $4 per switch so everybody would be able to purchase ones that suit them best. We ***can*** supply those if requested but that is not done by default. A couple of recommended sensors is noted on the CENTIPEDE-CONNector schematic (they are Mouser `http://www.mouser.com` parts) but you are free to use whatever you want. There are sensors with wire leads and with PCB pins, with different mechanical design, etc. We recommend using sensors with opaque housing and small aperture like those recommended – they give the sharpest and most repeatable signals of them all. Datasheets for those are available from Mouser that contain all the mechanical and electrical data, aperture location, etc.

All ***Analog*** signals are special. They are ***ALL*** powered from ***Machine-Side*** power and thus share ***Machine-Side*** signals ground. So although DAC outputs are formally ***Outputs*** because they receive data ***from PC*** and convert them to a corresponding 0-10V ***Output*** voltage they all use ***Analog*** signal ground. DAC and ADC use a different galvanic isolation mechanism, not optocouplers and even their logic signal levels are different on Machine and PC sides. So please use signal ground coming from DAC/ADC terminals if you use them e.g. for controlling your spindle rotational speed via a VFD. It is yet ***another*** signal ground so it should be connected to your VFD signal ground with a separate wire. That means there will be ***TWO*** signal ground wires between the BRK and VFD if you are also using some VFD logical ***Outputs*** to signal some VFD state or whatever to Mach3. Those two signal grounds (***Analog*** and ***Machine-Side*** logical/digital) should be tied together at VFD terminals. However they ***MUST*** be connected to the VFD with ***Separate*** wires. Don't fall to a temptation of connecting them together at both sides and using a single wire for interconnection. It will result in chaotic operation and can even damage BRK board or your VFD. And please be aware that your VFD ***MUST*** have optocouplers on its inputs if you want to control it with BRK *Output* signals. If it is not the case you should either make your own galvanic isolation board with

optocouplers between BRK and VFD or use signal relays on BRK to control the VFD with ***Machine-Side*** power.

BTW, there is yet another, often overlooked advantage of using optocouplers. Turning an optocoupler on requires substantial power (something like 50 mW, i.e. 10mA @ 5V) through the LED. It is much less likely to have such powerful noise currents in interconnecting cables than just 5V voltage shift with almost no power required if it was just logical inputs with high impedance.

1.3 Connectors

This section describes all connectors found on PCI and BRK boards, their purpose, description, pin mappings for BRK terminals (i.e. what signal goes to which pin) etc. All BRK references (like "top horizontal row," "left vertical column," etc.) are made according to Figure 1.2 in BRK Manual (page 6.) Please read that manual before making any connections and make it handy when connecting your machine to the system.

Please note that that Figure 1.2 in BRK Manual uses PROGRAMMER numbering that starts with zero while all Mach3 numbers start from 1. That means you'll have to add 1 to the numbers on that figure to get Mach3 numbers. *E.g. you should put 22 as a pin number in Mach3 configuration dialog for a signal connected to input terminal IN21 on BRK as it shown on that Figure 1.2.*

1.3.1 JTAG (PCI Board)

Connector ***J2*** in the center of the PCI 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.

This connector is not used for normal system operation, only for flashing a new firmware in the CPLD when it is available. Simple step-by-step procedure of firmware update is described in Firmware section (2.3) further down this manual.

1.3.2 External Connector (PCI Board)

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.

This connector is located on the back panel side of the PCI board and used for connecting the BRK board to the computer running Mach3. There is nothing more to say about it, it is obvious.

1.3.3 Extension Connector (PCI Board)

Extension connector ***J1*** at the PCI board edge is used for connecting add-on boards. There is no particular add-on interface implemented for this connector as of right now (firmware v3.0.1) so it is not used for anything at the moment.

That will definitely change in the future–there are several extension boards on our drawing board.

1.3.4 External Connector (BRK Board)

Long 100-pin connector in the center of the board is used for connecting BRK board to the PCI board with that special cable. Nothing special, should be obvious.

1.3.5 Power, Relay Outputs, +5V supply (BRK Board)

This is top horizontal row of terminals on the BRK. There are 6 connectors there. They are, left to right:

K1(OUT28) SPDT Relay contacts for a *POWER* relay connected to OUT28 output (output Pin 29 Port 11 in Mach3.) Switch **SW1.5** to connect or disconnect.

K2(OUT29) SPDT Relay contacts for a *POWER* relay connected to OUT29 output (output Pin 30 Port 11 in Mach3.) Switch **SW1.6** to connect or disconnect.

Mains This is where mains power is connected. It can be either 110 or 220 VAC. No switches or configuration required, it will work from any mains voltage between 92 and 267 VAC. The middle terminal is where the ground conductor should be connected. This should be also connected to the machine base. Two outer terminals are for Live/Neutral or 2 Live wires depending on what mains voltage is used. There is no polarity here, you can connect 110VAC Live/Neutral wires either way.

K3(OUT30) SPDT Relay contacts for a *POWER* relay connected to OUT30 output (output Pin 31 Port 11 in Mach3.) Switch **SW1.7** to connect or disconnect.

K4(OUT31) SPDT Relay contacts for a *POWER* relay connected to OUT31 output (output Pin 32 Port 11 in Mach3.) Switch **SW1.8** to connect or disconnect.

5VOut 5VDC ***Machine-Side*** power supply for external circuits (e.g. Linear Glass Scales or Encoders.) The rightmost terminal (closest to the mounting hole) is +5V, the left one is Ground. You can draw up to 2A from that Power Supply.

1.3.6 Input signals (BRK Board)

Those are all routed to the terminals in the left vertical column of the BRK board. They are made in 4 groups of 9 terminals each. The upper terminal in each group is ***Machine-Side*** ground that is used as a return terminal for all 8

signals connected to that group. It is easily identified by missing LED next to it. For a generalized schematics of this part please see a section on Optocouplers (1.2)

All 4 grounds are connected to the same ground plane so there is no requirement that signal terminals use a ground from their own group. Furthermore, although not recommended but it is acceptable to use only one wire from each switch to a signal terminal. The other switch pin should be connected to the machine base. Please note that the middle pin of Mains terminal (see above) ***MUST*** be connected to the machine base and the Mains ground wire also ***MUST*** be connected to it for this to work.

Each input has a corresponding LED indicator by its terminal. LED is lit when signal is active i.e. switch is *CLOSED* or shorted to ground. That is different from Mach3 "Active" meaning that can be exactly opposite if signal is configured a "Negated" or "Active low" (more on this later.)

Most of input signals have a hardwired specific mapping to Mach3 signals so those signals can not be remapped to different pins. However those pins can be used as general purpose input pins that are mappable to other Mach3 signals if the corresponding functionality is disabled. That might seem confusing so here is the more detailed explanation.

First of all each Mach3 Axis has a Home Switch and two Limit Switches (one for Plus side, e.g. X++ in Mach3 terms and one for Minus side, e.g. X--.) Those are Mach3 Signals that should be mapped (bound, assigned etc.) to a particular Port and Pin to be operational. They can also be disabled that means they are not mapped to anything and always inactive. This is a usual case with Mach3 stock Parallel Port driver because there is not enough input pins on those parallel ports to accommodate all input signals. That is why a machine built on that driver is one big hack with either all limit switches connected together or one Home switch for all axes etc.

It is not a case with CENTIPEDE. We *DO* have enough inputs to accommodate almost all Mach3 input signals on the CENTIPEDE along. There are 53 input signals in Mach3 total. That is *ALL* input signals it is aware of. CENTIPEDE has 32 inputs. If there are Parallel Ports available we can have up to 13 inputs from each Parallel Port if we use it in "mostly input" mode. That means we have more that enough inputs to accommodate *ALL* 53 Mach3 input signals on separate physical inputs if we have 2 Parallel Ports available.

However it is very unlikely that we would need that many signals for any machine. It is extremely improbable that some machine would have 6 moving axes, 4 encoders and 3 MPGs etc. That means we can accommodate all required signals for almost any configuration imaginable with just CENTIPEDE board alone or with CENTIPEDE and one Parallel Port in some extreme cases.

30 out of 32 CENTIPEDE inputs are permanently mapped to particular Mach3 signals in hardware. That means that corresponding Mach3 signals can ***NOT*** be assigned to different pins; the corresponding hardware (switches, encoders etc.) ***MUST*** be connected to preassigned BRK terminals. The same is true for some output signals.

Each of 6 Axes Mach3 supports (XYZABC) has 3 input and 2 output signals

permanently mapped to BRK terminals. Inputs are 2 Limit Switches (one for Plus and one for Minus sides) and one Home Switch. Outputs are motor Dir and Step signals. They can ***NOT*** be remapped to other terminals or used for anything else if the corresponding Axis is enabled. Let's say you have axis X enabled (that you almost certainly have.) That means that CENTIPEDE driver and Mach3 expects X++ switch to be connected to input terminal IN0, X-- to IN1, XHome to IN2 and it will output the corresponding motor Dir signal on output terminal OUT0 and Step signal will appear on OUT1. All those inputs and outputs are mapped to X Axis in hardware and this mapping can ***NOT*** be changed. It is this way because all those signals have some hardware connected to them inside CENTIPEDE CPLD (FIFO for Dir/Step outputs, Hit Position latch for Home Switch etc.) There is also no way to take any of this signals from the Axis (e.g. you decided you don't need a Home Switch on this axis so you would like to use the corresponding input for something else.) If you don't have a Home Switch on an Axis you enabled you ***MUST*** either short the corresponding Home Switch input to ground or configure it as Negated in Mach3 config to make it appear as always Inactive to Mach3.

Although it is a ***VERY BAD*** idea it ***IS*** possible to use a same switch for Axis Limit ***and*** Home starting with driver version 2.0.0.1. Please see section 4.5 for instructions setting CENTIPEDE for such configuration.

On the other hand any of 3 inputs and 2 outputs associated with a particular axis will become a General Purpose I/O if that axis is disabled in Mach3 config and you are free to map any Mach3 signal to them as you see fit. But again, please remember they are coming in groups of 3 inputs and 2 outputs per axis and they can be either assigned to that axis as the entire group if that axis is enabled or released to the pool of available for assignment inputs/outputs as the entire group. You can not take one of them from the axis signals group if the axis is enabled or reassign any of those axis signals to other terminals.

There are also 2 permanently mapped ***outputs*** for Dir/Step Spindle motor if it is used. They are exactly the same as axis Dir/Step outputs and follow them in mapping making something like 7th axis however there are no ***inputs*** mapped to the Spindle. There are no Limit/Home switches on the Spindle...

Other permanently wired input signals are Probe Switch Input, Emergency Switch (**Big Red Button**) Input, Spindle Index Input, Spindle Timing Input, and 4 groups of Encoder Inputs, 2 inputs per Encoder. There are no other pre-assigned outputs except those 2 (Dir and Step) per axis and Dir/Step Spindle. ***Correction:*** with newer firmware starting with v3.0.1 there is one more pre-assigned output that can not be remapped anywhere else. It is **ChargePump** output. It is permanently mapped to *OUT14* output if enabled. If ChargePump is not enabled in Mach3 configuration this output can be used for any other purpose.

Emergency and Probe Switches are permanent. They can ***NOT*** be disabled or used for anything else. Spindle Index, Spindle Timing, and encoder inputs are released to the available pool if the corresponding signal is disabled so they can be used for any other Mach3 signals.

Emergency Switch input is special. It can ***NOT*** be configured as Negated

so Emergency switch ***MUST*** be a Normally Closed one i.e. it must ***BREAK*** connection when activated. This is a usual rule for this kind of switches that ensures that machine will ***ALWAYS*** stop if the switch is operated even if its cable is disconnected or cut (which will make machine not to start at all.) If a Normally Open switch was used it would fail to stop the machine if its cable was broken.

Here is the full mapping table of CENTIPEDE-BRK ***input*** terminals to Mach3 signals. ***CENTIPEDE is Port 11 for Mach3***.

BRK Terminal	***Mach3 Signal***
IN0	X++ / Pin 1
IN1	X-- / Pin 2
IN2	XHome / Pin 3
IN3	Y++ / Pin 4
IN4	Y-- / Pin 5
IN5	YHome / Pin 6
IN6	Z++ / Pin 7
IN7	Z-- / Pin 8
IN8	ZHome / Pin 9
IN9	A++ / Pin 10
IN10	A-- / Pin 11
IN11	AHome / Pin 12
IN12	B++ / Pin 13
IN13	B-- / Pin 14
IN14	BHome / Pin 15
IN15	C++ / Pin 16
IN16	C-- / Pin 17
IN17	CHome / Pin 18
IN18	Probe Switch
IN19	EStop Switch
IN20	Pin 21
IN21	Pin 22
IN22	Spindle Timing/Pin 23
IN23	Spindle Index/Pin 24
IN24	Encoder1 A / Pin 25
IN25	Encoder1 B / Pin 26
IN26	Encoder2 A / Pin 27
IN27	Encoder2 B / Pin 28
IN28	Encoder3 A / Pin 29
IN29	Encoder3 B / Pin 30
IN30	Encoder4 A / Pin 31
IN31	Encoder4 B / Pin 32

Table 1.1: Centipede-BRK Input Terminals to Mach3 Signals mapping

1.3.7 Output signals (BRK Board)

Those are all routed to the terminals in the right vertical column of the BRK board. They are made in 4 groups of 9 terminals each. The upper terminal in each group is ***PC-Side*** +5V power for Step Drives etc. optocoupler LEDs. It is used as a power terminal for all 8 signals connected to that group. It is easily identified by missing LED next to it. For a generalized schematics of this part please see a section on Optocouplers (1.2)

All 4 +5V power terminals are connected to the same PCI +5V Power plane so there is no requirement that signal terminals should use +5V from their own group. ***Please be aware that it is +5V coming from the PC running Mach3 and it should be NEVER connected to +5V power coming from the BRK power supply!*** It should be only used for connecting to optocoupler LED anodes in Step/Servo Drives or whatever else is controlled by these outputs. Once again please see a section on Optocouplers (1.2) for explanation. Please contact KSI Labs, LLC if you still have questions on it. For those fluent in Electronics *FULL* schematics are included in CENTIPEDE-PCI and CENTIPEDE-BRK Hardware Manuals.

Each output has a corresponding LED indicator by its terminal. LED is lit when signal is active i.e. FET on the PCI board is turned ON i.e. output terminal is shorted to ***PC-Side*** ground that will power up the optocoupler LED connected between one of 4 those +5V ***PC-Side*** terminals and the output terminal. That is different from Mach3 "Active" meaning that can be exactly opposite if signal is configured a "Negated" or "Active low" (more on it later.)

Some of output signals have a hardwired specific mapping to Mach3 signals so those signals can not be remapped to different pins. However those pins can be used as general purpose output pins that are mappable to other Mach3 signals if the corresponding functionality is disabled. There is no way to remap those Mach3 signals to different pins or use those pins for anything else if the corresponding axis (or Dir/Step Spindle) is enabled in Mach3 config.

Here is the full mapping table of CENTIPEDE-BRK ***output*** terminals to Mach3 signals. ***CENTIPEDE is Port 11 for Mach3 and output pins are also numbered 1 to 32***.

BRK Terminal	*Mach3 Signal*
OUT0	X Dir / Pin 1
OUT1	X Step / Pin 2
OUT2	Y Dir / Pin 3
OUT3	Y Step / Pin 4
OUT4	Z Dir / Pin 5
OUT5	Z Step / Pin 6
OUT6	A Dir / Pin 7
OUT7	A Step / Pin 8
OUT8	B Dir / Pin 9
OUT9	B Step / Pin 10
OUT10	C Dir / Pin 11
OUT11	C Step / Pin 12
OUT12	Spindle Dir/Pin 13
OUT13	Spindle Step/Pin 14
OUT14	ChargePump/Pin 15
OUT15	Pin 16
OUT16	Pin 17
OUT17	Pin 18
OUT18	Pin 19
OUT19	Pin 20
OUT20	Pin 21
OUT21	Pin 22
OUT22	Pin 23
OUT23	Pin 24
OUT24	Pin 25
OUT25	Pin 26
OUT26	Pin 27
OUT27	Pin 28
OUT28	Pin 29
OUT29	Pin 30
OUT30	Pin 31
OUT31	Pin 32

Table 1.2: Centipede-BRK Output Terminals to Mach3 Signals mapping

1.3.8 Signal relays, Analog I/O (BRK Board)

This is bottom horizontal row. There is 7 terminal groups there. They are, left to right:

K5(OUT24) DPDT Relay contacts for a *SIGNAL* relay connected to OUT24 output (output Pin 25 Port 11 in Mach3.) Switch **SW1.1** to connect or disconnect.

K6(OUT25) DPDT Relay contacts for a *SIGNAL* relay connected to OUT25

output (output Pin 26 Port 11 in Mach3.) Switch **SW1.2** to connect or disconnect.

K7(OUT26) DPDT Relay contacts for a *SIGNAL* relay connected to OUT26 output (output Pin 27 Port 11 in Mach3.) Switch **SW1.3** to connect or disconnect.

K8(OUT27) DPDT Relay contacts for a *SIGNAL* relay connected to OUT27 output (output Pin 28 Port 11 in Mach3.) Switch **SW1.4** to connect or disconnect.

DACA/DACB Digital-to-Analog Converter (DAC) outputs, channels A and B. ***Analog*** 0-10V outputs for controlling spindle speed etc. DACA (left one) outputs control voltage representing set spindle speed when PWM spindle control is enabled in Mach3 configuration. DACB is not used for anything as of right now. Middle terminal is ***Analog*** signal ground. Please see section on Optocouplers (1.2) for connecting it to the rest of the system. ***Correction:*** both DACs can be controlled via PutPortByte() macros as described in Changelog.

ADC0/ADC1 Analog-to-Digital Converter (ADC) inputs, channels 0 and 1. ***Analog*** 0-10V inputs. Not used for anything right now. That will definitely change in the nearest future – they can be used for Feed Overrides, Jog Speed Overrides etc. Middle terminal is ***Analog*** signal ground. Please see section on Optocouplers (1.2) for connecting it to the rest of the system.

ADC2/ADC3 Analog-to-Digital Converter (ADC) inputs, channels 2 and 3. ***Analog*** 0-10V inputs. Not used for anything right now. That will definitely change in the nearest future – they can be used for Feed Overrides, Jog Speed Overrides etc. Middle terminal is ***Analog*** signal ground. Please see section on Optocouplers (1.2) for connecting it to the rest of the system. ***Correction:*** all 4 ADC channels can be read at any time using GetPortByte() macros as described in Changelog.

Please note: ***There is no protection on ADC inputs so please make sure you are not exceeding their 0-10V range and their input terminals are connected to POSITIVE input voltage.***

Chapter 2

FIRMWARE

2.1 General

CENTIPEDE-PCI board is actually 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 functionality can be radically changed by programming CPLD with different firmware. Such design allows for easy hardware bug fixes, 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 microcontroller 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 work 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 flip-flop 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 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 in 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 lack of a better one.

This chapter describes what's in the Mach3 Firmware for CENTIPEDE kit and how to flash a new (or different) firmware in the CPLD. New versions will be put on our web site, `http://www.ksilabs.com` when available.

The included information describes Mach3-specific firmware version 2.0.0 that is programmed in CENTIPEDE-PCI boards as they shipped.

2.2 What's in the Mach3 firmware

This firmware is designed specifically for Mach3 CNC software. It implements a very specific targeted set of hardware functions and components for Mach3, not a generic set of I/O ports. Those functions and components are used by our Mach3 software driver that links Mach3 application to CENTIPEDE firmware and ultimately to the actual machine.

Mach3 software itself does ***NOT*** control the machine. It is a user-level Windows application and as such it is way too slow to perform all necessary low level actions like generating motor Step/Dir signals etc. It is wisely designed as two-part system where each part performs its own set of functions.

The bigger part, Mach3 application does most of the job. It provides GUI, calculates all toolpaths, translates g-code into tool movement etc. It is too slow to control the actual machine so it works on a huge memory structure with long elementary step buffers, variables for input/output signals, encoder readings etc. It can not e.g. stop the machine at the precise moment so it generates instructions kinda "stop at X coordinate 12.3456."

The second part, the Engine is a very fast program that executes those low level commands Mach3 prepared and stuffed in that memory structure. It is not very smart but it is very fast. It is impossible to make such Engine as Windows application so it is made into kernel driver that is not bound by user space limitations and can work much faster.

Those two parts are relatively independent. Mach3 just fills that memory buffer and considers its work done. It doesn't care who is going to use that data, how it is used and even if it is used at all. Engine, on the other hand doesn't know about Mach3, DRO, GUI, g-code etc. It only knows that somebody gives it instructions using that memory buffer and its only job is to execute them and report results back to that buffer.

Mach 3 original Engine is its Parallel Port Driver. It is pure software working on a limited Parallel Port I/O. It does everything in software – generates motor pulse sequences, constantly analyzes e.g. encoder signals etc. This is very hard on the system because there is quite a lot of work and it must be done very fast and very often. That is why it is better to delegate as much work as possible to hardware that does it without stealing precious CPU time from the system. This is where CENTIPEDE firmware comes to play.

First of all there is hardware FIFO in there. It keeps up to 20 elementary steps for all 6 axes and outputs them one by one all by itself at the right moments. That allows the Engine to wake up and steal CPU time less frequently. Instead of doing it for each and every step it can wake up once per 10 steps or so. The process of temporary stopping the running application, saving its state, transferring control to the Engine, restoring the state of the application and transferring control back to it is orders of magnitude more expensive than moving 10 steps data from one memory region (Mach3 buffer) to another one (CENTIPEDE FIFO that looks like regular memory to the system thanks to PCI mapping.) That means we save a lot of CPU time by NOT waking the Engine 9 times out of 10. Actually there are other tasks it must do so we invoke it every 5 steps but it is also a huge improvement. That 5 figure comes from the fact Mach3 outputs step data in groups of 5 so it makes sense to check it every 5 steps to pick up a whole group at once if available. That means that unlike original Mach3 Parallel Port Driver that is invoked 25,000 times per second when Mach3 kernel frequency is 25KHz our Driver is only invoked 5,000 times per second.

Another advantage FIFO gives us is that individual pulses are output at precise intervals by the ***hardware*** based on a very stable crystal controlled system clock. There are no delays, no other software can mess up with it. Original Mach3 Engine outputs one step per software interrupt. It is not very reliable because of interrupt latency i.e. it can be delayed by random time

depending on what's going on in the system at the moment. We can even let us miss a couple of interrupts altogether and nobody will notice it because CENTIPEDE hardware will keep outputting steps from its FIFO at a proper pace and we will catch up with filling it once the interrupt finally came. It is all automatic, no software intervention required – we put steps in the FIFO on the software end as fast as we can (that takes orders of magnitude less than single output step interval) until it is full and hardware (*firmware* :)) pops them from the FIFO every step interval one by one and outputs them to the motors all by itself.

Dir/Step Spindle Motor is also controlled by hardware. As there is no planned motion on Spindle in Mach3 its controller is simpler than axis drive. It only does one of the two kinds of motion axis drives do, Jogging. It is a Variable Frequency 50% Duty Cycle square wave generator that does NOT require any software support for its operation at steady speed; it is all hardware. Single register write is only required for starting/stopping the spindle and changing its speed. Ramp up / slow down is implemented (in the Driver software) so real stepper/servo motors should work fine as Spindle drives.

Step/Dir Spindle can go as high as 1MHz pulse rate if required with no effect on other parts or software. It can go even higher than that up to approximately 8MHz (33MHz PCI Clock frequency divided by 4) but it would require special interconnection cable and something other than optocouplers in Spindle motor drive. So 1MHz is a conservative upper limit.

Spindle Index (single pulse per revolution) and Timing (multiple pulses per revolution) are both implemented. Index is required for lathe threading because it is used for starting thread at exactly same spot on each consecutive pass. It is also used for RPM reporting to Mach and for Z-Axis velocity compensation wrt Spindle RPM when threading or rigid tapping if no Timing sensor installed. There is no practical limit on Index/Timing pulse duration as long as it is more than ≈60 NANOsecons.

Both Index and Timing inputs have independent hardware debouncing. That is implemented as a configurable period of ignoring state changes after some initial signal transition (either high to low or low to high.) That is not actually necessary for hardware if those signals are clean (i.e. they are not from noisy mechanical switches) but they are here anyways.

Lathe threading / rigid tapping is implemented with both single pulse per revolution Index sensor and with ***additional*** multi pulse per revolution Timing sensor. If both are installed Timing sensor is used to compensate for Spindle RPM changes while threading. This is done by measuring RPM at each Timing pulse (i.e. as many times per rev as how many pulses Timing sensor generates,) comparing that actual measured RPM to the fixed RPM used by Mach for planning and changing kernel frequency to keep constant per-rev feed rate. There is no pulse skipping; it is constantly changing kernel speed that closely resembles manual lathe thread gear feed.

Single pulse per revolution Index sensor is used for starting each and every threading pass at exactly the same spot. It is kinda that half-nut engagement lever on a manual lathe. It is also used for RPM measurement and once-per-rev

feed compensation if there is no Timing sensor installed.

All RPM measurement and clock generation is done in hardware so no software intervention is required. When threading Index pulse is used to start fetching step data from hardware FIFO that is stopped when Mach notifies the driver it is about to start a threading pass. CENTIPEDE driver keeps stuffing step data in the hardware FIFO but it just sit there. At some moment FIFO is full and driver just waits for some room to free to continue moving step data from Mach queue to the FIFO. When Index pulse comes it pulls the hardware trigger and CENTIPEDE starts fetching data from FIFO and outputting steps. There is no software-related variations in this process, it is all pure hardware so the threading pass is started at exactly same spot each and every time no matter what software is doing or how busy the PC is. Maximum variation possible is 1 PCI clock i.e. 30 NANOseconds.

Each and every Timing (or Index if there is no Timing Sensor) pulse resets an internal ***hardware*** counter that starts counting PCI clock pulses (≈33MHz or 30nS). At the same time previous value is transferred to a software accessible register and an interrupt is generated. On that interrupt driver reads that value and makes a simple computation by multiplying kernel clock divisor value (kernel frequency is PCI clock, usually 33.333MHz, divided by that divisor) by this timing value and dividing it by reference value used for thread planning. The result is written to base divisor register thus adjusting feed rate to match actual Spindle RPM. This is repeated on each and every Timing (or Index) pulse as long as Mach is in threading mode.

Actual RPM is compared to a reference value, not to a previous sample that makes this process as close to actual physical threading gears on manual lathe as it is possible. There is no pulse-to-pulse error accumulation :)

CENTIPEDE can compensate for ≈70% spindle speedup at 100KHz kernel frequency. That is limited by Mach ability to fill its step data queue fast enough and ≈170KHz seems to be a limit for 2.93GHz CPU frequency. The range is wider at lower kernel frequencies.

There is practically no limit for Spindle slowdown compensation. CENTIPEDE will easily compensate for e.g. 5x spindle slowdown. There is probably something wrong with your machine if it slows down ***that*** much but it will be compensated anyways.

One other thing that got hardware acceleration is 4 encoders Mach3 supports. Actually it supports 7 of them but the remaining 3 are not encoders, they are MPG. Encoder can be very fast, especially high resolution one that is moved or rotated at significant speed. It is very difficult to process its signals by sampling them even at 25KHz. And usually they undergo what's called 4x multiplication that even further complicates that processing. The 4x multiplication is based on the fact there are 4 transitions per stated encoder step and it is possible to catch them all thus effectively quadrupling the resolution. That makes e.g. 2,500 PPR encoder to give 10,000 distinct readings per rotation. If it makes e.g. 600 revolutions per minute its state changes every 10 µS i.e. @ 100KHz frequency. It is impossible to process it purely in the software. CENTIPEDE firmware has ***hardware*** support for this thus allowing to use those

high resolution linear scales or rotary encoders with Mach3.

There is also different kinds of Home/Limit/Probe switches ***hardware*** support that make it automatically save hit positions and change operating state. EStop (**Big Red Button**) switch processing is also done by hardware that immediately stops any motion and notifies the software of that event. Hardware makes shure that no movement can be done until the condition is removed or (for Limit Switches) explicitely overridden. There is ***NO*** override for **Big Red Button** so no movement can occur when it is activated regardless of software state.

2 specialized SPI controllers also implemented in hardware. One of them is used to talk to TLV5617A dual channel 0-10V output DAC, another one is for TLV1544 quad channel 0-10V input ADC. Both DAC and ADC are installed on BRK board. One of DAC channels is used for Voltage Controlled Spindle (that is PWM Control in Mach3,) other DAC/ADC channels are not used as of now. ADC and DAC SPI controllers are independent and work simultaneously. All unused DAC/ADC channels can be accessed with GetPortByte/PutPortByte macros so they can be used for any purposes required by the user.

Starting with v3.0.1 firmware there are additional hardware features implemented in CENTIPEDE-PCI CPLD hardware.

First of all it is *hardware* ***ChargePump*** (it is implemented in software in original Mach3 Parallel Port Driver.) Not actually something having much sense, more like a workaround for simple cheap hardware but anyways, here it is. ChargePump signal is generated by the CPLD firmware and only output when CENTIPEDE is running (that means both CENTIPEDE driver is working and Mach3 is running and in good health.) It will stop if Mach3 is terminated (either by closing it or by software crash.) If you didn't check "ChargePump On in EStop" box in Mach3 General Config it will also stop when CENTIPEDE enters EStop state for any reason (Mach3 died, LimitSwitch hit, Watchdog kicked in, EStop button pressed, etc.) It will *ALWAYS* stop in '0' logical state that can actually be either High or Low depending on ChargePump signal "Active Low" setting in Mach3 Ports&Pins->OutputSignals configuration dialog. You can choose either state if it matters for your hardware that requires ChargePump signal. It only makes sense for ChargePump *stopped* state because inverting the actual 50% duty cycle clock still gives 50% duty cycle clock. ChargePump will output either "standard" 12.5kHz signal or 5kHz if "Set Charge Pump to 5kHz - Laser Stnby" box is checked in Mach3 General Config.

Then there is *hardware* ***Watchdog*** implemented in CENTIPEDE CPLD. It is implemented as 4-bit binary counter that increments every time CENTIPEDE-PCI generates timer interrupt request (Mach3 "Kernel Speed" divided by 5 i.e. 5kHz frequency or every 200 microseconds at the slowest speed.) When it overflows (on 16-th increment) EStop condition is generated and CENTIPEDE locks up in this state until reset by the driver with a specific command sequence. It is totally hardware so it does not require driver, Mach3, or even Windows to be running for it to operate properly. That means it will reliably stop your machine in couple of milliseconds after you PC showed you that infamous blue

screen. On normal conditions, when everything is healthy almost every interrupt request will be acknowledged by the driver Interrupt Handler routine. Such acknowledgement restarts the Watchdog so it starts counting from zero again. That means it will never overflow if everything is running properly - 16 interrupts in a row must be missed for it to overflow that should never happen on a properly running system. Watchdog is enabled by checking "Use WatchDogs" box in Mach3 General Config.

Finally, there are hardware ***Step Sequencers*** that can be inserted between internal Step/Dir signals and corresponding output pins. Those were added to allow quick and easy retrofit for older Bridgeport Series II Boss 6 CNC mills keeping their existing stepper motors and their drives. Usually all Step/Dir controllers output a *Dir* signal that tells the connected motor drive which direction it should move and *Step* ***Pulse*** for every step to be taken. These 2 signals are usually connected to a motor drive that has its own sequencer and actual motor coils drives. Stepper motors do *NOT* work off of Step pulses. They have several (usually 2 for regular steppers, may be up to 5 and more for special ones) phase coils that make them move. All stepper motors can be in several different states that are defined by which coils are energized and what is the current polarity in those. Those states are maintained all the time between the steps, they are *NOT PULSES* but *STEADY STATES*. Each type of motor has a limited number of allowed states (it is e.g. 4 states for 2-phase stepper motor operating in full-step mode) and those states are arranged in some numerical order. If motor is in some state number N at the moment this state should be changed to N+1 for the motor to make a step in one direction or to N-1 for opposite direction. If there is no more steps this state is *KEPT* until the next step. When motor is at the very last (for '+' direction) or very first (for '-' direction) state it will wrap to the beginning or the end of the state table on the next step i.e. the state table is cyclic.

All stepper motor drives have such sequencer followed by power amplifiers that actually generate current in motor coils based on sequencer state. Modern drives all have that sequencer built-in. Its index (state number) is incremented or decremented on every Step pulse depending on the Dir signal state at the time Step pulse comes in. It wraps around as it's been described when end or beginning of the state table is reached.

Older machines (e.g. above mentioned Bridgeport Series II Boss 6 CNC mills) do ***NOT*** have such sequencers. They only have amplifiers and rely on step sequences to be generated elsewhere and fed to them. They were usually fed from a punched tape that was prepared by some kind of standalone CNC programmer. They only had storage registers that held the last value (state) read from the tape until it's replaced by a new value. That means they can ***NOT*** be operated by a regular Dir signal and Step pulse pair. To make them work with Dir/Step signals their Axis Drives (amplifiers) should be replaced with new Dir/Step drives that means additional cost and labor. Furthermore it is highly probable that new drives would not be able to drive existing old motors so axis motors should be also replaced with new ones. That adds even more to the cost of retrofit and almost certainly means significant additional

work to fit new motors as it is very unlikely they will be exactly compatible with old motors mechanically.

However there is usually nothing wrong with those old amplifiers and motors (except they are bigger and heavier but it is hardly matters for several ton milling machine) and they could've been used if the controller had been able to output signals they understand. That is why step sequencers have been added to CENTIPEDE CPLD firmware. It allows for very quick and cheap retrofit by just disconnecting Axis Drives (amplifiers) inputs from existing ZDI (or whatever it is called in other such machines) board and connecting them to CENTIPEDE-BRK outputs while leaving everything else in place.

There are step sequencers for those particular machines in v3.0.1 firmware only but it is not difficult to add different ones if needed. Just get in touch with us and give us documentation for your machine and we will make a new firmware with those added in matter of days with no charge for it.

Step Sequencer is a configurable option. Based on Mach3 configuration (check "Max NC-10 Wave Drive" box in Mach3 Ports&Pins configuration dialog to enable) they are either inserted between internal Step/Dir signals and output pins so it is Sequencer *STATE* ouputs show up at CENTIPEDE-BRK Axis Drive outputs or left unconnected and Step/Dir signals routed to those pins.

As of now it is one configuration setting for ***ALL*** axes i.e. they ***ALL*** can be either regular Step/Dir or Sequencer outputs but that can be changed in the future to allow per-Axis configuration.

This is a very brief incomplete description of CENTIPEDE/Mach3 internals. Those who want to know each and every detail are invited to read the detailed firmware manual and source code for CENTIPEDE firmware and software Mach3 Engine that is all available from KSI Labs, LLC web site, `http://www.ksilabs.com` for free.

2.3 Firmware update procedure

From time to time updates to CENTIPEDE firmware may be released or there might be some customized version with different or changed functionality that you would like to use or try. The CPLD on the PCI board is reprogrammable in flash-like fashion so you can easily program it with different firmware.

To reprogram the CPLD you will need 2 tools. First of them is Quartus II sotware, Web edition available for free from Altera (`https://www.altera.com/download/software/quartus-ii-we`) It is available for Windows and Linux, both of them work. We use Linux version here so there might be some slight differences between what is described here and your software but they are minor if any and nothing's required to figure them out but common sense. Those who do design with that software already know how to use it and how to program a CPLD file into an Altera device. For those who don't it is necessary to install it somewhere. You do not need optional parts if any are available, just the basic package. Just proceed with installation in the same fashion you install

any Windows application and follow the prompts. There is no need for any customization, all defaults will do.

The second tool you will need is programming adapter. There are plenty of those available from Ebay and hundreds of different places on the Net. You will have to look for "Altera ByteBlaster II" or "USB Blaster". The former uses PC parallel port, the latter, not surprisingly :) is connected via USB. This is inexpensive tool, you can get ByteBlaster II for something like $19.99 or even less including shipping. It is not very complicated device and all of them work so there is no reason to pick an expensive or "genuine" one. All such devices from China on Ebay work just fine so don't overpay for a boutique device. Please make sure you buy ByteBlaster ***II*** device; older Bit Blaster or Byte Blaster ***without II*** won't work with Max II series CPLD that is used on CENTIPEDE PCI board. "USB Blasters" usually don't have "II" in their name but please check if the description says it works with "Max II" series of Altera CPLD. Almost all of them do.

Now go grab the firmware file from our web site or wherever it is. CPLD image files have ".pof" extension. Put it somewhere on your hard drive on the machine your Blaster is connected to and Quartus II software is installed on. It can be the same machine you run Mach3 on and where CENTIPEDE board is installed or a different one; it will work either way. Connect your Blaster to that machine. It is probably easier to use USB Blaster because there might be no Parallel Port on your machine or you can have CENTIPEDE Mach3 driver already installed that will grab all parallel ports and won't let Parallel Port Blaster to work without disabling the driver first. You can also have some machine wiring on the parallel port. This all is not a problem, just requires some additional steps. USB Blaster does not require any additional setup, no driver required for Linux. For Windows you will need USB driver. It comes with Quartus II software, installation procedure is described in Quartus II documentation or on Altera web site: `http://www.altera.com/download/drivers/usb-blaster/dri-usb-blaster-xp.html`. Please follow instructions; it is no different from installing any Windows driver for a USB device.

In case you have a Parallel Port ByteBlaster II you have to connect it to your parallel port. If it is the same machine you run Mach3 on you might have to disconnect the machine from the parallel port first if it is connected. Then, if you have original Mach3 driver or CENTIPEDE driver installed it must be disabled otherwise Quartus won't be able to talk to the Blaster. How to disable original Mach3 driver is described in Mach3 Installation Manual. To disable CENTIPEDE Mach3 driver go to Control Panel, choose "*Performance and Maintenance*" → "*Administrative Tools*" → "*Computer Management*" → "*Device Manager.*" Look at the device tree, you should see a node called "*CENTIPEDE.*" Click on plus sign next to it. It will expand and you will see our driver named "*Mach3 Driver for KSI Labs CENTIPEDE Board.*" Right click on it and choose "*Uninstall.*" That's it, you are ready to go. Windows will show you its usual "Found New Hardware" dialog for CENTIPEDE on next bootup. Just let it reinstall the existing driver to get it back or install the updated one if available.

Now connect the Blaster 10-pin programming plug to the connector at the center of PCI board. As a matter of fact you can do it at any time; there is no special requirements. You can do it before you start with update with your computer either on or off, doesn't matter. Both connector on the PCI board and Blaster Device Plug are keyed so it is difficult to insert it the wrong way. Not impossible, but quite difficult :)

Start Quartus II. After some activity you should see a big complicated Quartus main window. It will go check for latest updates and can ask if you want to upgrade to the latest version or whatever. Dismiss all those, you don't need to upgrade or whatever it suggests. Now you should setup Quartus Device Programmer for using the particular Blaster you have.

From "*Tools*" menu choose "*Programmer*." On the top left side you will see a button "*Hardware Setup*." Press that button. "*Hardware Setup*" dialog will pop up. In "*Hardware Settings*" tab you will see all available programming adapters. There will be only one adapter there because you probably have only one connected. If you have more that one that is unlikely they will be all listed there. It should also be chosen on "*Currently Selected Hardware*" listbox above hardware list. If it is not there ("*No Hardware*" is shown) click that box and select your adapter from the list. Click "*Close*" to return to the main Programmer screen. Check "*Enable real-time ISP...*" checkbox. Click on "*Mode*" listbox to the right of "*Hardware Settings*" button and select "*JTAG*." That's it, your programmer is all set and ready to go. Quartus will remember your settings on exit so you will not have to configure it again next time.

Now click "*Add File*" button on the programmer's left panel. That will bring a "*Select Programming File*" dialog where you could select the *.pof file you want to program into CENTIPEDE PCI board. Once you have a suitable file selected you will see a drawing of a CPLD chip and its part number that should read "*EPM2210F256*" in the bottom programmer's window. The top window will be filled with the image file information – file name, device name, checksum etc. Check the top checkbox in "*Program/Configure*" column. That should make the entire vertical column of 3 checkboxes checked and the top button named "*Start*" on the left panel should become active (it was greyed out before.) You are ready to program now.

Please make sure CENTIPEDE PCI is powered up. It sure is if you run Quartus on the same computer but it might be not the case if you use another computer for that. Now click start and you should see the process running – CPLD is erased first, then it is programmed and finally verified. Message box with progress messages should apper once you've clicked "*Start*" but if it didn't you can click "*Processing*" tab in Quartus "*Messages*" notebook (almost at the bottom of the Quartus main screen) to bring it up front. It should state that operation is completed successfully. You are done with CPLD programming; now you can exit Quartus, unplug the Blaster from the PCI board, disconnect it from your parallel port or USB connector and powercycle your computer.

Please remember that CPLD reads its configuration only on ***POWERUP***. That means it is ***NOT*** enough to reboot your computer for the new CPLD firmware to take effect. The old one will survive through anything but power-

cycle. So you ***MUST*** turn your computer off and then back on. It must be real powercycle, not putting the computer to sleep or whatever. CPLD must have its power fully removed and then, probably after 30 seconds or so it can be powered up. From now on the new configuration will take over.

There is no any special requirements on when the programming plug can be connected to the CPLD. You can plug it when the board is powered of or when it is on. Programming also does not affect current operation – board will be running as if nothing happened with the old firmware until powercycled. That means it is almost impossible to do something wrong, the process is extremely robust and reliable. You can even leave the Blaster permanently connected to the PCI in case you will want to reprogram the CPLD some later time; it doesn't have any effect on device operation unless put in programming mode. However it is a good practice to have it connected only for programming and disconnect it after it's done.

More (or less :)) information on CPLD programming is available in Quartus II documentation and from Altera web site.

Chapter 3

SOFTWARE

3.1 General

As it's been said earlier Mach3 is a 2-part system. One part, Mach3 application is the part one sees on his computer monitor and communicates with. It provides GUI with all information and controls, translates g-code program into series of instructions to machine axis motors, keeps track of all configuration and current activity etc. It is the only visible part of Mach3 system. It would probably be right to call it *Frontend*.

The second part, invisible to the user and not as smart as Mach3 application but very fast and essential one is Engine that takes instructions from Mach3 and converts them into real-time motor pulses and other control signals that actually control the machine. It also takes all switches signals, encoders readings etc. from the machine and communicates them back to Mach3 in a form and speed it can understand. That is kinda *Backend* to Mach3.

There are 2 different kinds of such Engines in Mach3. One is the kernel level driver-like Engine included with Mach3. This kind works directly on the huge memory buffer that Mach3 provides (struct TrajBuffer for those who is familiar with Mach3 SDK) and uses ***DIRECTLY*** connected PC interface circuitry to communicate with the machine. The only one existent as of time of writing is the Mach3 driver.

Another kind is so called "***Plugin***." All of 3rd-party interface boards for Mach3 use this type of Engine. Plugins are not as tightly coupled to Mach3 proper as kernel-level Engine. They use special software interface functions exported by Mach3 and usually do their own buffering etc. The very first thing they do upon startup is telling Mach3 they are in control so it should abandon its own Engine. That has its set of advantages and disadvantages. First of all it can not use PC resources such as Parallel Ports so they get wasted. Then they are separate applications with their own configuration etc.. That makes them more complex. Also they somewhat limited in using all Mach3 features because that Plugin interface is limited to motion control and some basic functionality.

On the other hand this kind of backend does not have to be connected to the PC internal bus. It might be connected via USB/Ethernet/etc. That poses some challenges because those interfaces are not predictable and have big latency but it works most of the time.

CENTIPEDE backend is implemented as kernel level Windows driver unlike *ALL* other third-party backends that are Plugins. It is probably one-of-the-kind Mach3 backend :)

The Engine is a single Windows driver named exactly the same as original Mach3 Parallel Port Engine, `mach3.sys`. It is installed in Windows in usual Windows way using supplied `.inf` file. Unlike other third-party engines it can use not only CENTIPEDE I/O resources but also grabs all available parallel ports so they can be used together with our board. This allows for mapping ***ALL*** Mach3 signals to individual I/O pins thus using Mach3 to its full potential. CENTIPEDE Mach3 driver ***replaces*** original Parallel Port Engine (or driver.)

Unlike ***ALL*** other vendors we only sell ***HARDWARE*** i.e. our boards. ***Software***, although it takes a lot of effort and precious time we give you for free. Keeping it secret rarely helps preventing copycats from stealing it but leaves customers at the mercy of vendor when it comes to fixing bugs, implementing new features, and providing customer support. There is another reason behind this decision – it would not leave our customers holding cat in a bag if something happened to us. They would still be able to fix bugs and have the products they bought supported. Sure it requires programming skills but those skills can be contracted if they don't have them in house. And there is no need for reverse engineering (not that only time consuming and expensive but may be illegal) because you have all the original source code. We give the *ENTIRE* source code including VHDL firmware for free. There is no gimmicks in that code – that is exactly the same source code we used to build our firmware and software. That also allows our customer to add their own features or change functionality as they see fit. No permits required for that and they are under no obligation to share their changes although it is good thing to do and highly appreciated. We do also hope there will be people that would find other uses for our hardware/software and share their work with community. We would be glad to accept any useful additions/changes to what we did and share them with the community via our web site.

However please note that although it is open source it is ***NOT*** public domain or GPL. We are a small company trying to make our living from what we do. Software development is expensive and time consuming job and we can ***NOT*** do it for nothing and let anybody else reap the benefits. So it is free as long as it is used with ***OUR*** hardware or anything we sell or distribute. If somebody wants to use our code for their own product he will need our permit to do so. Please read the license at the beginning of each and every source file for more information and please don't hesitate to get in touch with us if needed. One more time – there is no limitation on what you do with our Software/Firmware source code as long as it is used with our products. Using it for products not coming from us is explicitly prohibited without our written permit. The same is true for schematics added to PCI and BRK manuals – they are ***ONLY*** for our

customers to be able to implement their own additions, custom board, etc or make repairs. Those should not be used for building any commercial products without our explicit written permission.

3.2 Installation

First of all our Engine ***REPLACES*** that Parallel Port Driver from Mach3 distribution. That means that driver ***MUST*** be either removed from the system if it is already installed or ***NOT*** installed if it is a new Mach3 installation. Neither they can live together nor that original driver is required for proper operation. So before installing our Engine/Driver please make sure the original one is not installed. There are removal instruction in Mach3 install and configuration manual, please follow them for manual uninstall.

Our Engine is built as regular Windows WDM Driver so it will install in exactly same fashion as any other Windows Plug-and-Play driver. Once you have the CENTIPEDE PCI board installed in your computer Windows will find it automatically and show you the standard "Found New Hardware Wizard" dialog on the next system startup. You should decline the generous offer to search the Net for a driver – it is obvious Microsoft did NOT make a driver for our board... So you choose "No, not this time" and click "Next." On the next screen you should choose "Install from a list or specific location (Advanced)" and click "Next" one more time. On the following screen choose "Don't search. I will choose a driver to install" and click "Next" again. On the next screen click "Have Disk..." and either type the full path to the directory where you put `mach3.inf` and `mach3.sys` files or browse to that directory and click "Ok." Now you should see "Mach3 Driver for KSI Labs CENTIPEDE Board" in the "model" window. You will also see a bold font warning "**This driver is not digitally signed!**" under the model window. You can safely ignore it. Our Driver did not and will not go through Microsoft "approval" process because it would mean we could not give our source code to our customers and unnecessary waste of time and our limited finances to get such approval. We might start signing binaries we built and distributed by our own certificate but it is not Microsoft certificate anyways so it wouldn't get rid of that warning. Ok, back to installation. All you have to do now is to click "Next" for the last time, see the driver installed and then click "Finish" on the last screen to complete the install.

Now you have the CENTIPEDE Engine/Driver installed and ready to go. Unlike original Mach3 Parallel Port Driver there is no need for any reboot or whatever. Driver is ready to run as soon as it is installed. Also you do not have to have Mach3 installed to install CENTIPEDE driver and it is not required that anything (read BRK Board) is connected to the CENTIPEDE PCI board. You only have to install PCI board in your computer, that's it. CENTIPEDE driver will not mess with your system so everything will work fine with the driver installed but not used; there is no need to disable or uninstall it if you are not using Mach3 for a while. However it will grab all of the PC Parallel Ports if available so other software or drivers will fail to aquire them if the

driver is running. It is unlikely those ports are used for anything else on the PC running Mach3 but anyways. You should uninstall the driver (or Disable it) as it is described in Firmware update section if you want to use PC parallel ports for anything else.

There is no DriverTest program for CENTIPEDE driver because there is nothing to test there outside Mach3. Also unlike original Mach3 driver there is ***NO*** need for any emergency actions (like running DriverTest app) in case Mach3 crashed. All Mach3 remnants will be cleaned up by the system and the driver automatically – there are Cancel routines installed on critical parts so it is not going to make your system to BSOD.

There is nothing to do in Mach3 to tell it about the CENTIPEDE Driver. If everything is done right it will find and use it automatically, no intervention required. There is also no need for any special configuration for the driver – it takes all its settings from Mach3. If something is not right you will get something like "Failed to start timer" message from Mach3 in the bottom message box. Please remember also that Mach3 will ***NOT*** go out of EStop if BRK board is not hooked up and powered up with EStop switch closed. Also please remember that it takes something like 2 seconds for Mach3 to start with CENTIPEDE Driver so don't panic if you see Mach logo for more than a second :)

Please make sure you picked the right driver version for your system. The `mach3.inf` file is the same for all versions, `mach3.sys` comes in Windows2000, XP, 2003, 2008, Vista, and Windows 7 versions. Please make sure you took a right one. Driver for older Windows version should work with a newer one but it doesn't make sense to take chances.

BIG FAT WARNING for Windows 7 – you ***MUST*** run Mach3 with administrative privileges ***AND USER ACCESS CONTROL DISABLED***. That means that account (user, login) it is run from ***MUST*** be *Administrator*, ***NOT*** *Standard User*. Go to *Control Panel -> User Accounts and Family Safety -> User Accounts -> Change your account type* and change it to Administrator. Then go to *Control Panel -> User Accounts and Family Safety -> User Accounts -> Change User Account Control settings* and slide the slider all the way down, then OK and ***reboot***. Mach3 will ***NOT*** work without it, it will simulate. The very first indication something is not right is "*Simulate*" in "*Pulse Rate*" field of Mach3 Diagnostics Page. Normally it is "0."

3.3 Driver update

To update the driver with a newer/different version you should put the new .inf and .sys files somewhere on you hard drive where the system can find them. Then go to Control Panel, choose "*Performance and Maintenance*" → "*Administrative Tools*" → "*Computer Management*" → "*Device Manager*." Look at the device tree, you should see a node called "*CENTIPEDE*." Click on plus sign next to it. It will expand and you will see our driver named "*Mach3 Driver for KSI Labs CENTIPEDE Board*." Right click on it and choose "*Update Driver...*" That will bring that familiar "Found New Hardware Wizard" dialog. From now

on proceed exactly like it is described in "Installation" section above. There is no need for a reboot or any specific action after New Hardware Wizard finished. Driver is ready to work right after it is installed. However you'll have to powercycle if you updated CENTIPEDE *Firmware* (see the corresponding section, 2.3 on firmware update procedure.)

Please note that newer/different driver version may require different firmware version. This is a case with current v.2.0.0 driver–it requires v.2.0.0 firmware. You will ***NOT*** be able to use this driver with old firmware. To make sure it is not used with incompatible firmware PCI ID for CENTIPEDE-PCI board has been changed so the old driver will not be started. PCI board will appear as a totally new hardware for Windows and it will bring "Found New Hardware Wizard" dialog. When installing new driver Windows might tell that `mach3.sys` file already exist and ask if you want to overwrite it. Just say "Yes"...

Chapter 4

Mach3 configuration

4.1 Big Fat Warning

One ***BIG FAT WARNING*** before you start. Remember all that has been said about ***Plugins***? Just to reiterate it–*Motion Plugins* ***DISABLE*** kernel driver first thing when they started. That means you can ***NOT*** use CENTIPEDE with ***ANY*** other plugin be it Smoothstepper, Galil, or whatever. Once a motion Plugin starts it will disable CENTIPEDE and Parallel Ports.

4.2 General

CENTIPEDE Engine takes all configuration from Mach3. That means there is nothing to setup in the driver; all setup is done in Mach3. Configuration process is generally similar to the usual Mach3 configuration described in details in Mach3 manuals. However there are some differences that are described in this chapter.

Original Mach3 is supposed to work with only 2 PC Parallel Ports. Those ports are known to the original Mach3 Driver as ports number 1 and 2. Mach3 frontend i.e. the main Mach3 application that you see on the monitor screen and that does most of the job except actual communication with the machine hardware does ***NOT*** care what the port and pin numbers are. It does ***NOT*** work with the actual PC interface circuits. It works with software ***SIGNALS*** in system memory. Each ***signal*** is a programmatic structure that contains a set of variables that describe the signal configuration and state. Configuration variables define if a particular signal is *Enabled* so Mach3 will be keeping an eye on it, what *Port* and what *Pin* on that port is used for that signal, is signal *Negated* or not, and some others. ***SIGNALS*** are ***logical*** ones i.e. they mean some event happened or not for input signals or some command like "*Turn Mist Cooling On*" for outputs. They are ***NOT*** electrical signals like "*Set Pin #2 on Port #1 to High.*" Mach3 does ***NOT*** care about electrical signals, how those signals are output, and even if somebody is doing something with those

signals at all. It sets *Port/Pin/Negated* variables in signal structures (there is one such structure per every signal Mach3 knows about; there is 53 *Input* and 30 *Output* signals Mach3 is aware of) only for the Driver use. Mach3 only works on ***Status*** variables in signal structures and there is exactly one such variable called ***Activated***.

Mach3 app checks all *Input* signal structures ***Activated*** variables and if it finds this variable set to *TRUE* it knows the corresponding *Signal* is on. It does ***NOT*** care who set this variable to TRUE or FALSE.

Outputs are treated in a similar way but in reverse direction. When Mach3 decided that e.g. Mist Cooling should be turned on it ***sets*** the corresponding signal ***Activated*** variable to *TRUE*. It does ***NOT*** care if somebody takes any action when this variable got set. It assumes its work done when it has this variable set, plain and simple.

Signals are assigned fixed numbers in Mach3 e.g ZHOME input signal that is "Z Axis Home Switch is Hit" is Input Signal #8. Output signal ENABLE3 that is "Enable Axis 3" is Output Signal #3. That is how those signals are known to Mach3 app. It doesn't know what Ports and Pins are used for what signals and even if they are associated with any electrical I/O at all.

Mach3 ***DOES*** set *Port/Pin/Negated* variables for each and every signal according to the stored configuration but that process is kinda *Write-Only* – it sets those variables once on startup (or after you made some changes in its configuration) and never checks them again.

Those variables are set for the ***Driver/Engine*** exclusive use so it is able to setup ***Port/Pin to Signal Mapping***. Having this information the Driver knows that e.g. Pin 10 on Port 1 is ***Mapped*** to ZHOME signal that is Input signal #8 so it has to set that signal *Activated* variable to *TRUE* if Pin 10 on Port 1 is at "high" level or to *FALSE* otherwise. It also checks *Negated* variable to find out what is considered TRUE, "high" level or "low" one so if *Negated* is *TRUE* for that signal #8 it will do exactly the opposite i.e. *Activated* will be set to *TRUE* if Pin 10 on Port 1 is at "***low***" level, not "high." For Output signals Driver checks if *Activated* is *TRUE* or *FALSE* and sets the corresponding Pin on the mapped Port to the "high" or "low" state depending on *Negated* setting for this particular signal.

That means all those *Port/Pin* numbers are only meaningful to the ***Driver*** and they can be anything that Driver understands. Stock Mach3 Parallel Port Driver only works with up to 2 PC parallel ports that are considered ports #1 and #2. That means it will ***NOT*** understand anything else. Parallel port has only 25 pins so the Driver will ***NOT*** understand any pin number that is not in the range from 1 to 25. Also some of the parallel port pins can only be used as Outputs while some are Input only so there are limitations which pin numbers can be used for Input signals and which ones can be used as Outputs.

Now please ***FORGET*** about all those hints about Pin numbers you can see in Mach3 configuration dialogs :) They ***ONLY*** have meaning for the stock Mach3 Parallel Port Driver. Don't also try to use "Automatic" I/O assignment that is offered in some Mach3 configuration dialogs. It is designed for the stock driver and will do no good for anything else.

CENTIPEDE Engine/Driver has its own numbering scheme. There are 2 distinct I/O interfaces it can use – CENTIPEDE BRK and PC Parallel Ports if available. You should ***ALWAYS*** try to fit all of your signals to CENTIPEDE. Parallel Ports are last resort and you should avoid them like plague. The problem with Parallel Ports is that I/O operation on a parallel port is ***EXTREMELY*** slow and using them will have severe impact on your system performance. CENTIPEDE is a PCI device that looks like memory for the system so it is accessed in the same fashion as system RAM and it is read/written 32 bits per single CPU instruction. Parallel Port is a different beast. First of all it can be read/written only 8-bit at a time. Then it lives in totally different realm called *I/O Space* so it requires different machine instruction to access it. Those instruction theirselves are orders of magnitude slower than blazingly fast memory access. They are also not cached and there is no PCI buffers for them that allow PCI I/O operations to complete in a single memory cycle without any wait states. And the actual parallel port hardware (I/O chip) is painfully slow so the CPU must wait for I/O access to complete before it can do anything else. This is ***Hardware*** wait where the CPU is halted until the operation completes, not a ***Software*** one where CPU can do other work while checking from time to time if that I/O is finally done.

If you have no other choice and absolutely must use parallel ports because there is no enough pins on CENTIPEDE BRK for all of your signals try to put faster signals on the BRK while mapping only the slowest to parallel ports and try to fit everything in one port if you have 2 of them in the system.

There is another reason to avoid parallel ports – you won't need additional breakout boards for them :)

Ok, now let's get to work. There are separate sections for CENTIPEDE and Parallel Ports configuration. I hope you won't have to read the latter one :)

And one more thing – Engine/Driver only maps Input/Output Ports/Pins ***ONCE*** on Mach3 startup. So you ***MUST*** restart Mach3 after making ***ANY*** changes to those mappings including ***"Active low***" or whatever else changed in port mappings. There is no need to restart Mach3 if you changed anything else in configuration but you ***MUST*** restart it if port mappings changed. New values will ***NOT*** take effect until Mach3 application is restarted. That is Mach3 frontend a.k.a. GUI part, ***NOT*** Driver. There is no need in driver restart on ***ANY*** change. It fully reconfigures itself every time Mach3 application starts.

4.3 CENTIPEDE configuration

CENTIPEDE is ***Port 11*** for Mach3. Both Input and Output pins are numbered from 1 to 32. That means there are ***TWO*** pins #11 etc. in ***Port 11***, one for *Input* and one for *Output*. That should not be *that* confusing and our Driver will ***NOT*** try to send any data to *Input* pin or to read something from *Output*.

Pin numbers to BRK terminals mapping is described in section 1.3.6 for *Input* signals and in section 1.3.7 for Outputs. It should be obvious, they are counted from top to bottom with common terminals (top ones in each group of

9, with no LED by them) skipped.

Most of CENTIPEDE input pins and some of outputs are permanently mapped to Mach3 signals. This mapping is effective ***ONLY*** if the corresponding functionality is enabled. Two inputs are mapped to function that can ***NOT*** be disabled so those inputs can ***NOT*** be used for anything else. These are *EStop Switch* on *Input 20* and *Probe Switch* on *Input 19*. *EStop Switch* can ***NOT*** be configured as *not Negated*; it ***MUST*** be shorted for system to work.

Let's start with *Outputs*. There are 14 *Output* pins with permanent mapping. Strictly speaking they are ***NOT*** Mach3 signals but Motor Signals that Mach3 does not control (they are generated by the Driver according to Mach3 higher level motion commands) and they have their own configuration tab, "*Motor Outputs*" in Mach3 but we will be calling them "signals" for consistency. So there are 14 Outputs that are Motor ***Dir*** and ***Step*** signals, 2 signals per each of Mach3 axes and Dir/Step Spindle. ***Odd*** pins (1,3,5,7,9,11,13) are ***Dir*** Signals for X,Y,Z,A,B,C axes and Spindle, ***Even*** ones (2,4,6,8,10,12,14) are ***Step*** signals respectively (see section 1.3.7) They can ***NOT*** be remapped to any other pins if the corresponding axis or Dir/Step Spindle is enabled. That means if you have e.g. axis Z enabled, you'll have its *Dir* signal on Output 5 and *Step* on Output 6. It is fixed and can ***NOT*** be changed.

If the functionality is ***NOT*** enabled that permanent fixed mapping has ***NO*** effect. That means if you did ***NOT*** enable axis Z Outputs 5 and 6 are ***NOT*** mapped to anything and you are free to use them as you see fit (e.g. assign/map them to ENABLE1 and ENABLE2 *Output* signals.)

Only 14 of Output signals on CENTIPEDE have such fixed mapping. Signals *15..32* are not mapped to anything and you can use them for anything you want.

Input signals are more ridgid. 30 of 32 *Inputs* have fixed mappings. First of all it is *Plus* and *Minus Limit Switches* and *Home Switch*, i.e. 3 Inputs per axis, 18 total. They are made in 6 groups of 3 per axis (see section 1.3.6 for exact mapping.) The same rule stands – if the corresponding axis is enabled the ***entire group*** is bound to that axis; no Input can be disabled or used for anything else. If you even don't have e.g. Home switch on that axis you can ***NOT*** disable that *Input* or use it for anything else. In such a case it is recommended that you set that particular *Input* to *Negated* so it will be *always* inactive if you don't connect it to anything. If the axis is ***NOT*** enabled the entire group becomes free and you can use those 3 *Inputs* for anything you see fit.

Input 19 is *Probe Switch* input. Can ***NOT*** be disabled or remapped to anything. Set it to *Negated* if not used.

Input 20 is *EStop Switch* (**Big Red Button**.) That is the most special of all. Permanently mapped, permanently set to ***Negated***. Can ***NOT*** be disabled or configured to ***not Negated***. You ***MUST*** have this switch and it ***MUST*** be closed (shorted) when not activated.

Inputs 21..22 are not mapped to anything and you can use them for anything as you see fit.

Input 23 is *Spindle Timing Sensor* input. If you have a multiple-pulse per revolution sensor on your spindle it should be connected to this terminal.

Input 24 is *Spindle Index Sensor* input. If you have a single-pulse per revolution sensor on your spindle this is where you should connect it to. Please note that *Index* sensor is ***required*** for lathe threading and *Timing* sensor is ***optional***. That means you can cut threads with *Index* sensor only but you can ***NOT*** do it with just *Timing* sensor. Please see section 4.5 for more information on threading setup.

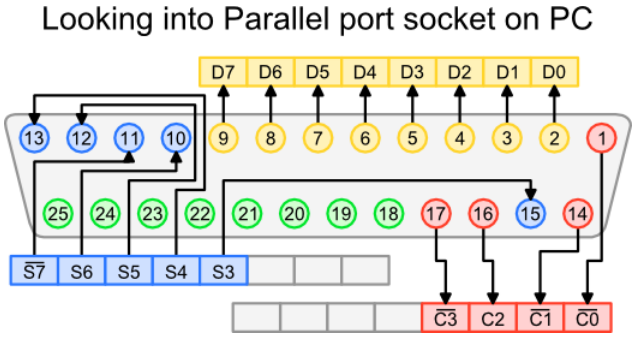
Inputs 25..32 are 4 groups of *Encoder* inputs, 2 per Encoder. Those are also not Mach3 signals to be precise and they have their own configuration tab in Mach3, "*Encoders/MPG's*." Those are Encoder1..4. For mapping please see section 1.3.6. Same rules apply – if the Encoder is not enabled corresponding Inputs are released from mapping and you are free to use them as you wish. To reiterate what was said about these encoders – they are *FAST* ones with hardware assistance. They are *NOT* for MPGs that are implemented in software and way slower. You connect high resolution Encoders/Scales that go in Mach3 DRO to these *Inputs*.

For fixed mappings CENTIPEDE Engine/Driver will ***OVERRIDE*** Mach3 configuration without asking. That means no matter what you put in e.g Z Axis Dir/Step outputs Driver will set them to Port 11 pins 5/6. You should probably put the proper values in Mach3 Motor Outputs anyways so you will know what they actually are but it will work fine even if they set to absolutely bogus values. The same is true for ***ALL*** fixed mappings that are currently in effect.

However, you can still configure ***OTHER*** parameters for signals with fixed mappings. You can configure e.g. "*Active low*"(*Negated*) field for Inputs/Outputs (***except EStop***) or Counts/Velocity for Encoders etc. and they will be saved and will take effect once Mach3 is started.

4.4 Parallel Ports configuration

So you decided to use parallell ports... OK, here is how they are used with CENTIPEDE Engine/Driver.



This is how PC parallel port looks to the outside world and to the software that works with it. As you can see there are 4 groups of pins in a single parallel

port. Green ones are ground wires. They are connected to the PC signal ground (remember that section 1.2 on Optocouplers and galvanic isolation?) and thus bear no signals so they are not controlled in any way.

Then you can see 3 other groups of signals. That, BTW one more time reiterates why parallel port is *SO* slow – you have to do 3 I/O operations (and those operations are extremely slow theirselves because they are going to I/O space, not memory and the parallel port chip is extremely slow to respond to those operations) to access those 17 signal pins because they leave in 3 separate groups each of those living at a different I/O address.

OK, let's get to those groups. First group is 5 *Blue* pins. Those are parallel port *Status* pins. In its original incarnation, used for connecting a printer to PC those pins were used to communicate printer status (Ready, Paper Out etc.) to PC. We are not talking to a printer and there is nothing special about those 5 pins so we can use them as generic ***Input*** pins. They are accessed by reading parallel port *Status* register. As you can see signal S7 has a dash over it that means this particular signal (pin 11) is *inverted* i.e it reads as '0' when pin 11 is at "high" signal level. So this *Blue* group gives us 5 *Input* pins that we can use for Mach3 *Input* signals.

Next one is a group of 4 *Red* pins. Those a parallel port *Control* pins that were used to control printer operations. They are ***Outputs*** and they are accessed via parallel port *Control* register. As you can see 3 out of those 4 signals are *inverted* i.e. they will output a "low" signal voltage when the corresponding register bit is set to '1'.

OK, so far we have 5 *Inputs* and 4 *Outputs* from a parallel port. There is one more group left, 8 *Yellow* pins. Those were *DATA* ***Outputs*** in original parallel port that were used for sending DATA to a printer. However all modern parallel ports allow to switch them into ***Input*** mode if needed thus using them for DATA *Input*. To enable such functionality parallel port ***MUST*** be setup as ***EPP*** mode port in PC BIOS. It might work OK in *SPP* mode but it is not guranteed. Do ***NOT*** use ***ECP*** mode because it is special and might not work as EPP mode parallel port.

So there are 2 scenarios here. We can either set ***ALL*** 8 *Yellow* pins to work as *Outputs* thus making that port "***mostly output***" or as *Inputs* turning it into "***mostly input***" one. There is no way to change directions for ***individual*** pins in that group; they can be either ***ALL*** *Inputs* or ***ALL*** *Outputs*. The choice is yours. Probably you would want to make your port into "*mostly input*" one because there is usually more than enough *Outputs* on the CENTIPEDE alone and that's *Inputs* that you might come short of. But again it is up to you how you want to setup your system.

Original Mach3 allows to set *only* Port 2 into "*mostly input*" mode by checking "*Pins 2-9 as inputs*" checkbox in "Ports & Pins" configuration dialog. We use a different configuration scheme for CENTIPEDE.

First of all we do NOT use those "*Port Enable*" and "*Port Address*" fields. Parallel Ports are detected automatically and they are ***ALWAYS*** grabbed by CENTIPEDE Driver i.e. they are *always enabled*. Then we do ***NOT*** use manually set addresses. We use what Windows tells they are. Windows Parallel

Port 1 becomes port 1/5 (please read on for explanation) and Parallel Port 2 becomes port 2/6.

To make a parallel port into "*mostly input*" one we add 4 to the port number. So Parallel Port 1 is referred as "*Port 1*" in Mach3 configuration if it is used in its regular "*mostly output*" form where pins 2..9 are *Outputs*. If you want to use that same Parallel Port 1 in "*mostly input*" mode where pins 2..9 are *Inputs* you should use "*Port **5***" in Mach3 configuration. Parallel Port 2 becomes "*Port 6*" if it should be "*mostly input*" one. That is how CENTIPEDE Engine knows what mode it should switch the port to.

Please remember, only ***ONE*** port number must be used for ***ALL*** its pins. If you decided your Parallel Port 1 is "*mostly input*" ***ALL*** its pins ***MUST*** be put as Pin xxx ***Port 5*** in Mach3 configuration fields. It is an ***ERROR*** to use ***BOTH*** port numbers i.e. "*Pin 9 Port 5*" and "*Pin 17 Port 1.*" They ***MUST*** be ***ALL*** either "*Port 1*" or "*Port 5.*"

Other than that there is nothing special in Parallel Ports setup for CENTIPEDE Engine/Driver in Mach3. And you can safely ***IGNORE*** all those warning labels like "*Pins 10-13 and 15 are inputs. Only these 5 pin numbers may be used on this screen.*" This do ***NOT*** apply to CENTIPEDE and Mach3 would accept anything you enter in those fields. You ***MUST*** be careful however to use proper pin/port numbers because e.g. *pin 17* is ***Output*** no matter what mode your port is in so you can ***NOT*** use it as an *Input*. In most cases that assignment will be silently ignored so that particular signal won't work but sometimes it can cause damage to your system so ***please be careful***.

4.5 Common cofiguration hints

There is no actual ***PWM*** control for spindle speed. It looks like original Mach3 Driver implemented PWM because of lack of better hardware. There is no way to do anything else with just parallel ports. It outputs PWM signal on one of outputs that you have to somehow convert to analog voltage to do actual speed control with. It is not all that complicated but not very precise and involves additional hardware.

In CENTIPEDE we use real 10-bit DAC for this. Speed control works and is configured as regular Mach3 PWM but instead of pulses you get actual 0-10V voltage at DACA output (see section 1.3.8 for connections.) Recommended base frequency is 5. It doesn't actually matter because there is *NO* frequency for DAC but it is used in internal calculations.

On MPGs – they are processed programmatically in software. There is no hardware assistance for those so they are quite slow. MPG signals also undergo 4x multiplication like Encoders do so they are actually 4x the PPR shown in their specs. It is ***NOT*** recommended to use anything beyond 100 PPR for MPGs; that is not needed for such a device and it will probably work erratically if used with CENTIPEDE and rotated too fast. Most of those pendants sold on the Net are 100 PPR so they should work fine.

Also please note that both hardware and software support for Encoders and

MPGs is rudimental. There is no support for index pulses, no absolute encoders support that would've excluded any need for Homing, and Encoder inputs are not suitable for direct Encoder hookup. Encoders usually use ***differential*** signals where there are "+" and "-" signals for each phase and index signals. You will need something like differential receiver, e.g. SN75ALS193 chip to convert those signals from differential to single-ended that are suitable for connecting to a regular optocoupler input. This is true for both CENTIPEDE and original Mach3. To make things even more complicated there are Encoders with ***Sinusoidal*** output signals (e.g. many of HEIDENHAIN encoders.) Those can ***NOT*** be connected to a differential receiver; you will need some analog circuits (comparators) to convert those to digital pulses suitable for single-ended Parallel Port or CENTIPEDE inputs. There is an add-on board in the works as of time of writing this document that will hookup to CENTIPEDE PCI board and accept any type of encoders, including sinusoidal input and DeviceNet/EnDat/ProfiBus absolute position Encoders directly without any additional interface converters but it is not out for production yet.

There is ***NO*** ChargePumps on Parallel Ports. They are not all that easy to implement to be really useful and parallel ports are only considered ***Auxillary*** I/O not actually controlling the machine motors etc. so there is little reason to do them.

There ***IS*** Step/Dir Spindle Motor support in this version. It can go as high as 1MHz pulse rate without any impact on Mach3 performance. If you want to use it just enable it in "Motor Outputs" dialog and check "Use Spindle Control Output" and "Step/Dir Spindle" in "Spindle Setup" dialog. You can also use it to control your VFD if it requires 50% Duty Cycle square wave. "Step Low Active" setting doesn't make any sense for Spindle motor so you can either check it or not; it should work just fine either way. Change "Dir Low Active" setting if it rotates in wrong direction.

For Spindle Index/Timing the best sensor is something like 100 PPR rotary encoder with separate Index pulse. It is ***NOT*** used as a quadrature encoder i.e. we do not use both A and B phases and do 4x multiplication. You use one of the phase signals (A or B, shouldn't matter) as Timing pulse and Index goes to Index input. But it is also OK to use separate sensors like slotted wheels etc.

Do not fall to a temptation of using high-PPR encoder on the Spindle. It is definitely better to have Timing sensor than to rely on Spindle inertia and use 1 PPR Index sensor only. But the higher the PPR number of the Timing sensor the smaller the pulse count between pulses. That will actually ***DECREASE*** the timing resolution if that number is too high. Please also remember that each and every Timing pulse generates IRQ to the driver and it must reprogram base clock divisor on every such IRQ. It takes time and ***DOES*** put a burden on the PC. 100 PPR is probably a good value to use. 200 PPR might be also fine. But 2048 PPR is definitely too high and will do more harm than good. That might change in future ***hardware*** versions when pure-hardware Spindle control will be implemented but it won't make much sence even then.

Please do also remember that Spindle ***Index*** is required for threading, no matter if you have *Timing* sensor or not.

There is another thing to Spindle Index and Timing sensors – debouncing. It is not clear what that setting was doing in original Parallel Port Driver but here is a brief explanation and short instruction on what it is and how to set it up for CENTIPEDE.

First of all a short explanations what it is and why is it needed. ***Mechanical*** contacts usually don't operate cleanly. When contact closes it starts to conduct current; that is what it is for. The problem is all mechanical contacts are springy and contact surfaces that come into contact with each other are not ideal. That is why after making initial contact those contact parts bounce back and briefly break contact. Then they make contact again and the entire cycle repeats until that decaying mechanical oscillation stopped and contact settles in the closed position. Contact surfaces oxidation and other impurities also add to this process because they must be broken (shifted, sheared, etc) for good contact to take place. That all results in contact "bouncing" i.e. contact going short/open several times before settling in closed (or opened–all that also applies to contact opening as well) state. The net result of this is that electronics sees not one open/close transition when switch closes or opens but several of those. The number and timing of those bouncing transitions is random and it can take up to 50 mS or even more for a mechanical switch to settle.

It is clear that such a switch confuses electronics it is connected to. If e.g. Timing sensor sends a stream of random pulses on each and every slot you will end up with totally bogus RPM readings. That is why some action should be taken to exclude those parasitic transitions. There are several different methods to do this. CENTIPEDE approach is based on the fact that very first transition is ***ALWAYS*** a right one – if switch was open it will ***ALWAYS*** close first before starting bouncing. The same is true for a closed switch that will ***ALWAYS*** open first before starting bouncing. So when this first transition is detected CENTIPEDE registers it as a valid one and stops checking that signal for some time. When this period ends it starts checking that signal again. By that time contacts already settled and their state is stable. That debounce time interval is configurable and should be set for slightly longer than contact settling time.

There are 2 independent debouncing circuits in CENTIPEDE; one for Spindle ***Index*** and one for Spindle ***Timing*** sensor. They are configured in Mach3 "General Logic Configuration" dialog (upper right corner of it.) *Debounce Interval* sets debounce time for Spindle Timing sensor, *Index Debounce* configures Spindle Index debouncing circuit. CENTIPEDE uses 5 μS as its unit, ***NOT*** 40 μS as it is stated in Mach3 dialog.

Debouncing is only required if you have really noisy bouncing signals from those sensors. There is no need for it if your sensors provide clean signals (e.g. you have an optical rotary encoder for them) so you can safely put 0 or some small value like 5 or so in those configuration fields.

For not so clean sensors you ***MUST*** use debouncing. Optimal 100% safe debouncing interval is slightly longer than Index/Timing pulse width at the slowest RPM but slightly less than *half of interval* between pulses at the highest RPM possible. That means you should make the pulse itself ***as short as possible***. CENTIPEDE runs at PCI clock speed i.e. at something like 33MHz.

That means it checks its input signals every 33 *NANO*seconds so there is no need for the pulse to be very long. 1 *MICRO*second is more than adequate so if you use e.g. a mirror patch on some kind of wheel do ***NOT*** make it too big or wide. The smaller it is the more reliable your Index/Timing sensor will be. It will also allow you to set debouncing to a value that will be long enough to cover the entire duration of that pulse at the lowest possible RPM while still be short enough to be less than half of interval between pulses at the highest possible RPM. That is the optimal value that will make your sensors 100% reliable. Remember, CENTIPEDE processes those pulses in ***HARDWARE*** unlike regular Mach3 Parallel Port driver that only checks it once per interrupt in ***SOFTWARE*** so it can easily miss that pulse if it is too short. That can easily happen if the pulse is shorter than 40 microseconds when running at 25KHz Kernel Frequency. That also adds to thread start uncertainty because it is not possible to predict where that interrupt happens within that 40 microseconds. CENTIPEDE, on the other hand, has only 33 ***NANO***seconds uncertainty that is at least 1000 times better. There is no need for the pulse to stay at active level once it is detected. CENTIPEDE only needs pulse ***EDGE*** to operate properly i.e. it only checks for Index/Timing level ***TRANSITION*** from inactive to active state.

The usual advice is to start with something like 5 and check if RPM readings are stable in Mach3. Do it with Spindle Timing disabled first so RPM will be measured with Index sensor then turn Timing on if you have such a sensor to set that up. If you DO have bouncy sensors you will get a wildly varying readings. Change the debouncing value to e.g. 100 and start adding 100 to that value until RPM display stabilized. Then add another 100 and leave it at that setting. *Do not forget to save configuration and restart Mach3 every time the configuration has changed for the new value to take effect.*

On Dir/Step Pulse settings in "Motor Tuning" dialog – those are ignored. CENTIPEDE outputs 50% Duty Cycle pulses on ***ALL*** Step Outputs regardless of those settings. As a matter of fact they were not such a good idea in the first place because they are ***NOT*** per-axis despite the fact they might look like they are. It is a single system-wide configuration value that makes it only good if ***ALL*** motors/drives are the same that is not always true. Those were there because it was not easy to generate 50% Duty Cycle pulses in original pure-software Parallel Port Driver.

On "Step Active Low" for axes motors – don't try to figure out what to set it to. Your motors will work either way. You set them when you get motors running – the right setting will give you smoother motion and nicer sound. Please do ***NOT*** forget to restart Mach3 after you have this configuration changed.

DACB and all 4 of ADC channels are not used for anything at the moment. Support for Feed Rate/Jog Speed/etc. will be probably added in the next driver version so it will be possible to connect potentiometers to ADC inputs and use them for those controls.

You should ***ALWAYS*** set up ***SEPARATE*** switches for Home and Limits. They serve totally different purposes–Limit switches are the last line of defence for your machine protecting it from a serious damage and you from being se-

riously hurt when the machine attempts to move beyond the limits for some reason. They do not have to be very precise but they must be very robust and reliable. The ***ONLY*** purpose of Limit switches is to put the machine to ***EMERGENCY STOP*** unconditionally and immediately no matter what it was doing and what problems it may cause (wasted part, broken cutter, lost position information, etc.) to prevent greater damage. One should not rely on those switches to activate at some precise position; the only requirement is they should trip soon enough that machine is able to stop without damaging itself and hurting its operator.

Home switches are totally different ones. They ***MUST*** be very precise because they are used to establish machine reference point. That means they must activate at the exactly same position each and every time. They should usually allow some overtravel because machine is not stopped abruptly when hitting a Home switch but rather decelerated to a full stop gently so no steps are lost in the process. They ***CAN*** be fragile–there is no safety issues if they fail to operate.

You should ***NEVER EVER*** use Limit switches for Homing or vice versa. It is not just unreliable, it is a clear and explicit ***SAFETY VIOLATION*** that puts your machine at risk and may cause great harm and even death to its operator, you.

Unfortunately it was a usual practice to do so for many hobbyists and Parallel Port Mach3 bears a great deal of responsibility for encouraging such an unsafe practice. There is simply not enough inputs available from 2 parallel ports to accomodate separate Limit and Home switches for several axes so people used to combine those. It is a ***BAD*** practice.

So one more time – you should ***NEVER EVER*** combine Home and Limit switches. However there might be situations when there is simply no other choice. CENTIPEDE allows to use one switch for both Home and Limit starting with v.2.0.0.1 Engine/Driver. But please be aware that ***YOU DO IT AT YOUR OWN RISK*** and ***IN NO CASE KSI LABS, LLC WILL BE RESPONSIBLE FOR ANY DAMAGE, PERSONAL INJURY OR DEATH OR WHATEVER UNEXPECTED RESULTS THAT MAY HAPPEN***. It is your decision so all the responsibility is yours and you have been forewarned.

If you decided to use the same switch for both Limit and Home despite all those warning here is how it is done.

You should not assign the same pins for Limit and Home. It won't work anyways because those assignments are hardcoded so Mach3 configuration for those is simply ignored. What you should do is to leave the assignments as they were for separate switches and *TIE* Limit and Home switch *terminals* together i.e. connect a wire from that switch to *BOTH* Limit and Home terminals.

That will make your switch to *ALWAYS* work as a Limit switch. To enable dual functionality you should go to Mach3 "Ports and Pins" configuration dialog and check "Pins 2-9 as inputs" box for Port2. It is not used for anything else, all other options are simply ignored. Then, after saving your configuration and restarting Mach3, you will be able to use those switches as Limit and Home. It

is done by ***DISABLING ALL LIMIT SWITCHES FUNCTIONALITY*** while Mach3 is in Homing state. That means you can damage your machine if you miss that Home switch or for whatever other reason. That also means that ***NO LIMIT SWITCHES FUNCTIONALITY*** will be restored until the Homing process is successfully completed. Once the Homing process is completed those switches will switch back to being Limit switches.

For those who need ***ChargePump*** signal to make their motors work or whatever other purpose – it is implemented starting with v3.0.1 ***firmware***. It has its mapping *FIXED* – if enabled it will *ALWAYS* be output on CENTIPEDE-BRK Out14 (Mach3 Pin15) pin. This mapping is set every time Mach3 starts and CENTIPEDE driver will ***OVERWRITE*** whatever you set in Ports&Pins config. So if you have some output signal mapped to that pin you will have to move it to some other output pin. To enable *ChargePump* check the Ports&Pins->OutputSignals->ChargePump->Enable box (it should become green check mark.) ChargePump signal will be only output if CENTIPEDE (and Mach3) is not in EStop mode unless you have *"ChargePump On in EStop"* box checked in Mach3 General Config. If it is checked ChargePump will be running all time when Mach3 is running but will stop if Mach3 is terminated. For those who need ***5kHz*** ChargePump signal it is also supported, just check *"Set Charge Pump to 5Khz - Laser Stnbdy"* box in Mach3 General Config. *Please note that ChargePump will* ***ALWAYS*** *start at 12.5kHz and only switch to 5kHz when Mach3 is out of EStop for the first time.* It is how Mach3 works so there is nothing we can do about it. If you need ChargePump signal to be in some particular state (High or Low) when stopped this state can be changed by setting/clearing "Active Low" setting in Ports&Pins->OutputSignals->ChargePump configuration. It does *NOT* affect ChargePump signal when it is running, just the state it stops in.

If you need additional safety you can enable ***Watchdog*** implemented in CENTIPEDE firmware starting with v3.0.1. It will put CENTIPEDE ***Hardware*** in EStop mode in less than 10 milliseconds if your Windows, Mach3, or CENTIPEDE driver crashed or if you Windows became irresponsive for more than 16 "Kernel Speed" intervals. That might trigger false EStops if you have your PC overloaded or use a cheap on-board graphic adapter but otherwise it is a good safety measure. So we encourage you to give it a try by checking *"Use WatchDogs"* box in Mach3 General Config. Uncheck it if you experience frequent EStops with "Driver Watchdog" message although it is an indicator your Mach3 system is not working properly and you should try to find out what is wrong with it.

If you are retrofitting an old CNC machine similar to Bridgeport Series II Boss 6 CNC Mills and want to keep existing motors and Axis Drives (from Textron or whatever) you should enable ***Step Sequencers*** by checking *"Mach NC-10 Wave Drive"* box in Ports&Pins config. For more detailed information on Step Sequencer please read the section 2.2on firmware description.

Torch Height Control (THC) for Plasma Cutters is fully implemented starting with driver version 2.2.1.6. All THC signals are fully remappable and work as expected. All regular Mach3 configuration options related to THC work

as expected. In case it is not described anywhere else there are 2 additional Mach3 signals associated with THC. Signal Output#5 is set by Mach3 when it is in THC mode if *"Set OUTPUT5 when in THC"* box is checked in Ports&Pins->MillOptions config (yes, that is where Mach3 guys decided to put it.) This is GENERAL signal meaning "THC is enabled;" it doesn't follow temporary THC stops due to Plunge control etc. You can map that signal to any of available CENTIPEDE outputs if you need it. The second signal is ***Output#6*** that not documented anywhere else as far as we know. This signal has *"Hold off THC temporarily"* meaning. It ***activates*** when otherwise active THC is put on hold temporarily to prevent it from plunging due to low speed etc. THC control in Mach3 is done by moving Z-axis according to "THC Up"/"THC Down" commands from THC unit and it will be paused at the right moments but there are THC Units that also do their own corrections so this signal can be used to tell them to hold off that correction for a while (i.e. it is *"THC Inhibit"* signal.) It can be also mapped to any available CENTIPEDE output pin as you see fit. Please do not forget that you can NOT use those pins or signals for anything else if they are used for THC.

And as a final note – you can use the Mach3 *Diagnostics* screen to check your hardware setup in exactly the same manner as it is suggested in Mach3 documentation.

Chapter 5

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. 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. CENTIPEDE-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.