8I20 THREE PHASE MOTOR DRIVE MANUAL

V1.9

CAUTION!

THE 8I20 USES VOLTAGE AND POWER LEVELS THAT REPRESENT A HAZARD TO LIFE AND LIMB.

THE 8I20 IS INTENDED FOR USE BY OEMS THAT WILL INTEGRATE IT INTO A SYSTEM WITH INTERLOCKS AND OTHER SAFETY FEATURES TO PREVENT USERS FROM CONTACTING HAZARDOUS POTENTIALS OR BEING INJURED BY MECHANISMS POWERED BY THE 8I20.

WHEN CHANGING JUMPERS OR OTHER OPERATIONS THAT REQUIRE PHYSICAL CONTACT TO THE 8I20 CARD:

1. DISCONNECT MOTOR POWER AT THE 8I20

2. DISCONNECT MOTOR U,V,W AT AT THE 8I20. EVEN WITH NO POWER APPLIED TO THE 8I20, A SPINNING SERVO MOTOR CAN GENERATE LETHAL VOLTAGES.

3. WAIT 5 MINUTES FOR ON CARD MOTOR POWER CAPACITORS TO DISCHARGE.

WHEN TESTING THE 8I20 ON THE BENCH IT IS SUGGESTED TO AT THE **MINIMUM:**

1. CONNECT THE 8I20'S CHASSIS GROUND CONNECTION TO A SECURE BUILDING GROUND.

2. USE A ISOLATED MOTOR POWER SUPPLY

3. TEST FIRST WITH A LOW VOLTAGE MOTOR POWER SUPPLY

4. TAKE EXTREME CARE WITH SERVO SYSTEMS, **EXPECT** THEM TO RUN_AWAY WHEN FIRST TESTED.

Table of Contents

Table of Contents

GENERAL

DESCRIPTION

The 8I20 is a low cost 2200W 400V three phase torque mode/voltage mode amplifier for synchronous permanent magnet servo motors (Brushless AC servo) up to approximately 3HP. The 8I20 will supply peak currents of 30A.

High side logic and gate power are derived from the low side allowing a wide operational bus voltage range (24 to 400VDC) and host communication when bus voltage is off. Low side power can be 5V +-5% or 8-40 V unregulated. 3750V RMS isolation is provided between high and low side electronics. Hall effect current sensing is used along with high speed bus voltage monitoring for an accurate, stable, and fast current/torque control loop.

Host communications are provided by a low overhead 2.5 Mbps serial protocol over isolated RS-422 link. Link speed allows up to 5 KHz update rates from host.

High side over current sensing protects the IGBT module from line-line and line-ground faults. A brake output capable of 15A drive is provided. The brake output can be driven at a presettable overvoltage setting.

Even though it uses a 40 MIPs DSP, the 8I20 is a 'dumb' amplifier suitable for integration in host based motion control systems. The 8I20 requires reference angle and requested current/torque or voltage values sent from a host controller. The 8I20 uses the requested torque and reference angle to control the current loop. The 8I20 can echo status information to the host controller including bus voltage, phase currents, card temperature, and other parameters..

Both smart (SoftDMC) and host based (HostMot2) multi axiscontrollers are available for the 8I20. Up to 32 axis of motion can be controlled by a single low cost FPGA based controller.

HARDWARE CONFIGURATION

GENERAL

Hardware setup jumper positions assume that the 8I20 card is oriented in an upright position, that is, with the I/O and power connectora towards the top of the card, away from the person doing the configuration;.

LOGIC POWER SOURCE

The 8I20's logic power can be supplied via the serial cable or via pluggable terminal block TB1. Jumper W2 determines the power source,

W2	MODE
UP	Logic side power from serial cable (5VDC)
DOWN	Logic side power from TB1 (7 to 48VDC)

SETUP/OPERATE MODE

The 8I20 can run in setup mode or operate mode. In setup mode, the serial interface baud rate is fixed at 115.2 KBaud and the motor drive circuits are disabled. In the operate mode, the baud rate is set to 2.5 Mbaud (default) and the motor drive circuitry is enabled. Setup mode is enabled to allow normal PC serial ports or USB serial adaptors to communicate with the 8I20 for setup purposes. W6 controls the setup/normal mode selection.

W6	MODE	BAUD RATE
UP	Operate mode	2.5 Mbps (default, can be changed)
DOWN	Setup Mode	115.2 Kbps (fixed)

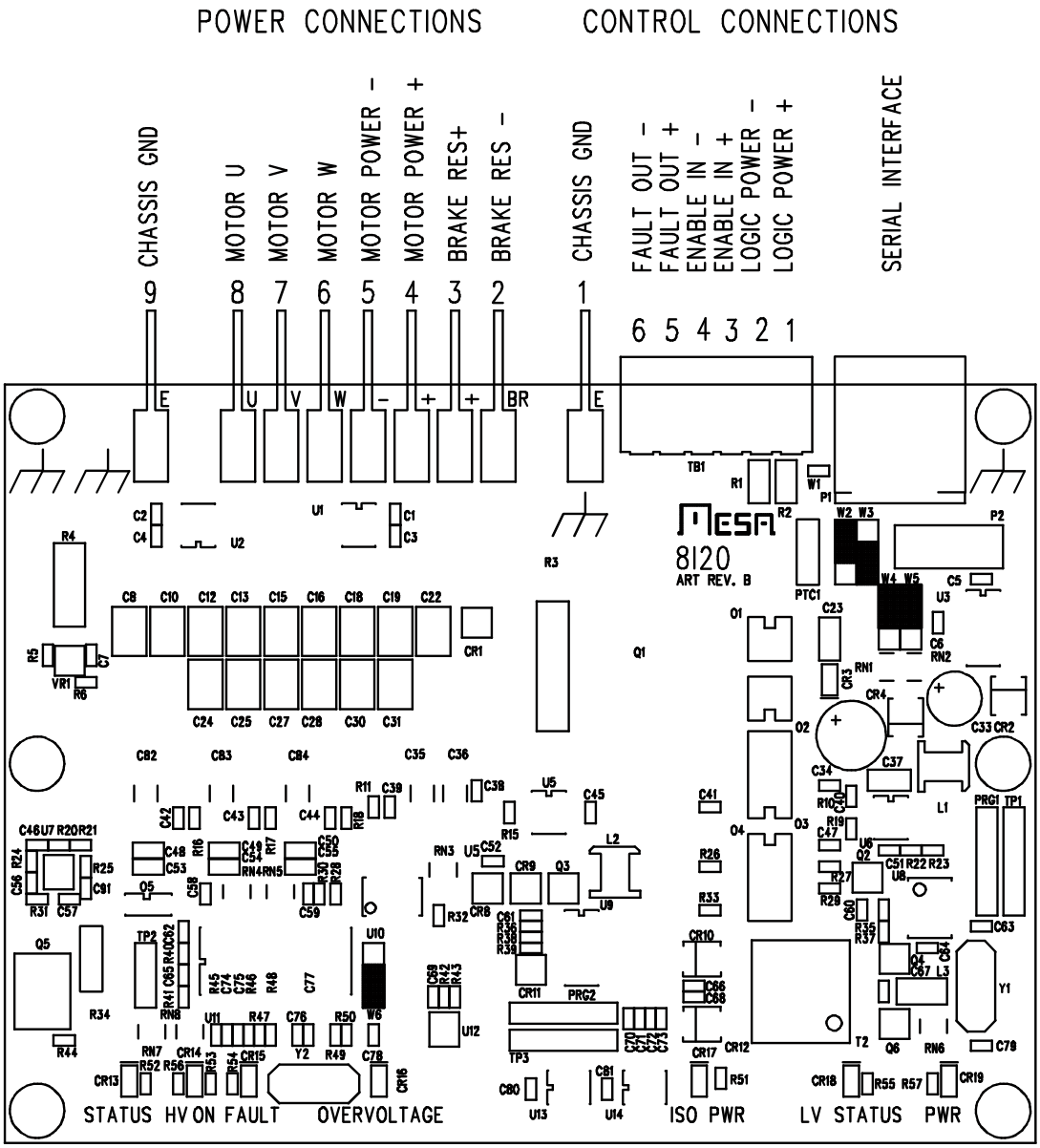
SERIAL PORT TERMINATION

The RS-422 serial port on the 8I20 can be terminated or un-terminated. Normally the 8I20 is the serial cable endpoint so the port must be terminated. If the 8I20 is used with other devices on a shared RS-422 interface, the 8I20 should only be terminated if it is the last physical device on the RS-422 cable. W4 and W5 enable and disable the termination

W4,W5	MODE
UP,UP	Terminated (default)
DOWN.DOWN	Unterminated

CONNECTORS

8I20 CONNECTOR LOCATIONS AND DEFAULT JUMPER POSITIONS



CONNECTORS

MOTOR - POWER - BRAKE CONNECTORS

The motor, motor power and brake connections are brought out on .250" spade terminals on the top edge of the 8I20 card. From left t right the terminal functions are:

1 Chassis Gnd

2 Brake resistor -

3 Brake resistor +

4 Motor power +

5 Motor power -

6 Motor W

7 Motor V

8 Motor U

9 Chassis Gnd

LOGIC POWER/FAULT CONNECTOR

TB1 is the logic power/ fault output/enable input connector. TB1 is a six terminal 3.5 mm pluggable screw terminal block. TB1 pinout is as follows:

1 Unregulated logic power +

2 Unregulated Logic power -

3 Enable in +

4 Enable in -

5 Fault out +

6 Fault out -

CONNECTORS

SERIAL PORT

J1 is the 8I20s serial interface. J1 is a RJ-45 jack. The serial interface pinout is compatible with standard 8 wire CAT5 Ethernet cables. J1 pinout is as follows:

1	RXA
2	RXB
3	TXA
4	GND
5	GND
6	TXB
7	+5V
8	+5V

J1s pinout is designed to match breakout cards like the 7I44. A standard CAT5 or CAT5E cable can be used to connect the 8I20 to a 7I44. CAT5E cable is suggested if the serial cable is used for powering the 8I20, as the larger wire size result in lower voltage drop.

OPERATION

LOGIC POWER

All logic on the 8I20 runs from 5V logic power. This can be supplied over the serial cable, or via a on card 5V regulator. Jumper W2 determines the logic power source. When jumper W2 is in the 'UP' position, 5V logic power is suppled by the serial cable. Due to voltage drop in the serial cable. The 8I20 should only be powered via the serial cable when serial cable length is 8 feet or less. Typical 5V power consumption is 350 mA.

Logic power can also be supplied via the on card regulator and TB1. When W2 is in the DOWN position, logic power comes from the on card regulator which is fed by TB1. Unregulated power to TB1 pins 1 and 2 is regulated to 5V logic power. Unregulated input to TB1 pins 1 and 2 can range from 8 to 40 VDC

MOTOR POWER

Motor power is supplied to the 8I20 via spade lugs #4 and #5. Motor power can range from 35 to 400 VDC. All communication to the 8I20 will work down to 0V motor power but the under-voltage fault setting will prevent motor operation at lower then ~35 VDC unless the default setting is changed. Default over-voltage fault is set for 385 VDC. Voltages high that 385VDC will result in a overvoltage fault.

The 8I20 has local high frequency bypassing but relies on the power supplies output capacitors to supply dc power with less than ~15% ripple. The 8I20 also relies on power supply output capacitors to store motor inductive energy. For this reason, power should not be disconnected from the 8I20 power input terminals when in operation. The 8I20 motor power leads should use 18 GA wire and must be shorter than 3 feet. That is, motor power should not be routed more than 3 feet from the main power supply filter capacitors. Power supply circuit breakers or fuses should be located on the power supply primary, not between the 8I20 and power supply capacitors.

MOTOR BRAKE

To prevent regenerated motor power from creating a over-voltage condition, the 8I20 can dump regenerated motor power into a user supplied brake resistor. The brake circuitry can provide a maximum of 15 amps of braking current. At the maximum BrakeOn setting of 385 VDC, this equals a ~25 Ohm resistor for maximum braking current.

Actual brake on/off voltages are programmable. Default brake on voltage is 360 VDC and default brake off voltage is 340 VDC. BrakeOn voltage must always be set higher than BrakeOff voltage.

In addition to the programmable brake voltage settings, there is a fixed 385V BrakeOn setting that cannot be changed. This is for 8I20 module protection.

OPERATION

MOTOR CONNECTIONS

The U/V/W motor connections are the direct PWM outputs of the 8I20s bridge. Larger motors may have enough winding to frame capacitance to generate considerable ground noise. This noise source can be reduced by:

1. Make sure that the motor frame ground signal is returned to the frame ground on the 8I20 card.

2. Use a ferrite bead as a common mode choke around all phase wires, that is the U/V/W motor wires pass through a single bead. Do not route the frame ground wire through the bead. A suggested ferrite bead is: Laird-Signal Integrity Products 28B1122-100.

ENABLE INPUT

The 8I20 has a isolated 4-24V enable input. This input must be driven to enable motor drive. If enable input is not driven, the 8I20 will go into a fault state, and turn off motor PWM.

FAULT OUTPUT

The fault output is an isolated transistor (OPTO coupler) output. The fault output is off in non-masked fault conditions and on in normal operation. The fault output can supply a maximum of 5 mA of output current and will switch voltages up to 24 VDC. Normally the fault outputs of all 8I20s would be wired in series.

FAULT CONDITIONS

The 8I20 monitors many parameters and reports fault conditions in the FAULT register. When a non-masked fault occurs, the 8I20 turns off motor drive and the current control loop. The following is a list of the fault conditions:

NAME	BIT	FUNCTION
WATCHDOGFAULT	0	Communication timeout fault
NOENABLEFAULT	1	No external enable fault
OVERTEMPFAULT	2	8I20 PCB temperature > 85C
CURRENTFAULT	4	8I20 high side overcurrent Fault

OPERATION

FAULT CONDITIONS

NAME	BIT	FUNCTION
ULOOPCURRENTFAULT	5	U current >125% of MAXCURRENT
VLOOPCURRENTFAULT	6	V current >125% of MAXCURRENT
WLOOPCURRENTFAULT	7	W current >125% of MAXCURRENT
BUSLOVFAULT	8	Low motor voltage fault
BUSHIVFAULT	9	High motor voltage fault (settable)
MAXBUSVFAULT	10	High motor voltage fault (fixed at 400V)
MODULEFAULT	11	Module over temperature or low gate voltage
OERRFAULT	14	8I20 serial port overrun error
FERRFAULT	15	8I20 serial port framing error

CLEARING FAULTS

Faults can not be cleared by writing the fault register but are cleared by setting the CLEARFAULT flag. The host must then poll the state of CLEARFAULT until it is zero.

FAULT MASK

Certain faults can be masked so that they do not affect the external fault output. Currently only the BUSLOVFAULT is maskable. A zero bit in the FAULTMASK register will mask the corresponding fault bit.

OPERATION

STATUS REGISTER

The 8I20 maintains a status register that reflects various internal 8I20 conditions.

NAME	BIT	FUNCTION
ILIMITEDSTATUS	0	Indicates that MAXCURRENT is not available due to 8I20 temperature/current limiting.
BRAKESTATUS	1	Current brake on status
BRAKEWASONSTATUS	2	Brake has been applied status (sticky)
WDTO	4	DSP startup due to Hardware watchdog timeout (sticky)
SWR	5	DSP startup due to software reset (sticky)
EXTR	6	DSP startup due to external reset (sticky)

Sticky status bits remember events until low side power is removed or the status register is cleared.

OPERATION

STATUS LEDS

The 8I20 has 7 on card status LEDs located on the bottom edge of the 8I20 card. The status LEDs in left to right order:

LED	COLOR	NAME	FUNCTION
CR13	Green	DSP Status	Blinks with host communication
CR14	Yellow	HV On	On when motor power is present
CR15	Red	Fault	On when drive fault is present
CR16	Red	OverVoltage/Brake	On when brake is applied
CR17	Green	Isolated power	On when isolated power is present
CR18	Yellow	LV status	On when isolated inverter is OK
CR19	Green	LV power	On when logic power is present

HEAT SINKING

The 8I20 requires additional heatsinking unless used for low continuous power servo applications. When used to deliver its rated continuous 2.2KW power, the 8I20 can dissipate up to 65W. Heatsink thermal resistance should be chosen to keep the module temperature below 100C. For example, in a 50C ambient environment and continuous 2.2KW load, keeping the module temperature below 100C with 65W of power dissipation requires a total thermal resistance of 100-50/65 = 0.77 C/W. The 8I20s mounting plate adds to the thermal resistance.

Two styles of 8I20 mounting plates are available, right angle and parallel. The right angle plate has the advantage of tighter packing density but has higher thermal resistance. The right angle mounting plate adds approximately 0.5C per watt to the 8I20s thermal resistance. The parallel mounting plate adds only about 0.1C per watt thermal resistance. So for a parallel mount 8I20, the external heatsink thermal resistance needed to sustain the full 2.2 KW continuous power is 0.77-0.1 = 0.67 C/W.

Achieving 0.67 C/W in a reasonable size normally will require fan cooling.

OPERATION

PC HOST ADAPTER

In order to run any of the command line utilities a RS-422 adapter is needed. Mesa can provide a suitable adapter. Two such adapters are 3I21 or 3I22. These adapters connects the RJ-45 RS-422 interface on the 8I20 to a DB9 serial port (3I21) or USB port (3I22) and provide 5V link power.

MINIMAL HOST PC ADAPTER

A simple home made host adapter can be made by directly connecting RS-232 signals from a 9 pin PC serial port or USB RS-232 adapter to the 8I20s RS-422 signals via a one ended CAT5 cable. A single resistor between RS-232 TXD and RS-422 RXB is needed to prevent overloading the RS-232 TXD output

CAT5 PIN	DE-9F PIN	CAT5 SIGNAL	DE-9F SIGNAL	CAT5 COLOR
1	5	RXA	GND	ORANGE WHITE
2	3	RXB (1)	TXD (1)	ORANGE
3	XX	TXA	XX	GREEN WHITE
4	5	GND	GND	BLUE
5	5	GND	GND	BLUE WHITE
6	2	TXB	RXD	GREEN
7	XX	+5V (2)	XX	BROWN WHITE
8	XX	+5V (2)	XX	BROWN

Notes:

1. Connect via 470 Ohm 1/4 watt resistor. All other signals directly connected

2. If not run from field power, +5V power must be supplied via the 8I20s serial cable

OPERATION

DRIVE PARAMETER SETUP

The 8I20 has both working and EEPROM drive parameters. The working parameters are volatile (lost at power down) 8I20 uses the working parameters for all normal operations. At power up or reset, a subset of the working parameters get initialized from non-volatile EEPROM memory. Because of this, when adjusting parameters the working parameters can be changed 'live' but changing the EEPROM parameters will not have any effect until the 8I20 is reset. All EEPROM parameters have a NV prefix.

SETUP COMMUNICATION WITH 8I20

To enable communication between a PC and the 8I20 three things are required:

1. 8I20 must have logic power (normally supplied via TB1)

2. Setup jumper must be in "SETUP" position = DOWN

3. A RS-422 adapter must connect from the PCs serial port/ USB serial adapter to the 8I20. This adapter is available from Mesa (3I21, 3I22)or home made adapter shown above

4. The parameter read and write programs are simple command line utilities that require some environment variables to be set before use:

SET COMPORT = COM1 (set to match the serial port)

SET BAUDRATE = 115200

SET PROTOCOL = LBP

WPD

The WPD utility writes 8I20 parameters including non-volatile setup parameters. It uses symbolic names for the parameters so numeric constants do not need to be memorized, for example

WPD BRAKEONV 30000

Would set the working brake-on voltage parameter to 300V

(units of voltage are .1V)

WPD NVBRAKEONV 32000

Would set the EEPROM brake-on voltage parameter to 320V

OPERATION

DRIVE PARAMETER SETUP

WPD

WPD FAULT 0

Would clear the FAULT parameter

RPD

The RPD utility reads a parameter from the 8I20. It uses symbolic names for the parameters so numeric constants do not need to be memorized, for example:

RPD FAULT H

Would read the fault parameter as a Hexadecimal word

RPD BUSV

Would read the motor bus voltage (in units of 10 mv), that is a bus voltage of 320 would read as 32000

MAIN DRIVE SETUP PARAMETERS

For normal setup only a couple of parameters need to be changed to match a specific motor. The most important are NVMAXCURRENT, NVBRAKEONV and NVBRAKEOFFV.

MAXCURRENT

The MAXCURRENT parameter sets the full scale RMS motor current of the 8I20. This is the single most important setup parameter needed to match the 8I20 to a specific motor. This should be set to the maximum RMS motor current value (or to 30A if the motor has a higher than 30 A RMS current rating). Current (torque) commands are signed 16 bit numbers sent to the 8I20 and scaled such that:

RMS motor current = QISETPOINT * NVMAXCURRENT/3276700

MAXCURRENT is a unsigned 16 bit number with units of 10 mA, that is maximum per phase current in Amperes is MAXCURRENT/100. Like most setup parameters, MAXCURRENT is the working current limit, and NVMAXCURRENT is the non-volatile EEPROM current limit.

For example:

WPD NVMAXCURRENT 2500

Would set the EEPROM maximum current parameter to 25A. This will not take effect until the 8I20 is reset.

OPERATION

DRIVE PARAMETER SETUP

BRAKEONV and BRAKEOFFV

BRAKONV and BRAKEOFFV set the brake on and brake off voltage thresholds. These are used to clamp the rising motor bus voltage created when decelerating. When decelerating, the 8I20 will charge the motor power supply filter capacitors with regenerated energy .

The default values (on at 360V off at 340V) are chosen to match a 240V off line power supply, and to protect the 8I20. If lower power supply voltages are used (with lower filter capacitor voltage ratings) the BRAKEONV and BRAKEOFFV voltages should be set lower to protect the power supply. It is suggested that BRAKEOFFV be set 20V lower than BRAKEON. BRAKEONV and BRAKEOFFV are specified in units of 10mV. That is a BRAKEONV value of 30000 would specify 300.0 V.

The startup values of BRAKEONV and BRAKEOFFV are set with the parameters NVBRAKEONV and NVBRAKEOFFV.

For example:

WPD NVBRAKEONV 19000

WPD NVBRAKEOFFV 17000

Would be suitable values for a 120V line operated supply with a maximum rating of 200V. Note that like all EEPROM parameters, these settings will not take effect until the 8I20 has been reset.

OPERATE MODE BAUD RATE

The operate mode baud rate default is 2.5 MBaud. This should not be changed unless needed for non-standard applications. Baud rates are selected by writing an index value to the NVBAUDRATE parameter. The index numbers for available baud rates are as follows:

INDEX	BAUD	INDEX	BAUD	INDEX	BAUD
0	9600B	1	19200B	2	38400B
3	57600B	4	115200B	5	230400B
6	460800B	7	921600B	8	1.25MB
9	2.5MB*	10	5MB	11	10MB

OPERATION

CURRENT LOOP TUNING

The 8I20 has settable tuning parameters for its PI current control loop. The parameters are KQP, KDP, KQI, and KDI. KQP and KDP are the Proportional terms and KQI and KDI are the Integral terms. Normally KQP should equal KDP and KQI should equal KDI. The Startup values of KQP, KDP, KQI, and KDI are set by the cooresponding EEPROM parameters NVKQP, NVKDP, NVKQI, and NVKDI. Default values forNV KQP and NVKDP are 50 and default values for NVKQI and NVKDI are 150000. These defaults are suitable for a large range of motors, but may need to be changed to optimize current loop operation for specific motors.

CONTROLLERS

MULTI-AXIS CONTROLLERS

Two basic types of multi-axis controllers are available for use with the 8I20, host based and "Smart". Host based controllers use the host computer to read encoders and output angle and run a PID loop that outputs torque commands to the 8I20. The HostMot2 FPGA firmware suite is a host based motion controller that is compatible with the 8I20. Smart controllers like SoftDMC take higher level motion commands from the host and control all low level processes like reading encoders, running the PID loop and flagging error conditions.

HOSTMOT2 8I20 INTERFACE

The Hostmot2 interface to the 8I20 is a smart serial interface for Mesa's Anything I/O series of FPGA cards that encapsulates the serial protocol details and presents a simple parallel register set to the host computer. Registers for writing torque and angle and registers for reading bus voltage, card status, communication status, and card temperature are provided for all connected 8I20 cards.

The 8I20 Hostmot2 interface is a sserial module with specific firmware (SSLBP) for 8I20 card or other LBP interfaced cards. Each sserial module can support up to eight 8I20 cards. The sserial module supports the standard 2.5 M Baud communication rate and torque update rates to 5KHz. With the default configuration, the Hostmot2 interface sends reference angle (ANGLE parameter) and current (QSETPOINT parameter) commands and receives bus voltage (BUSV parameter), 8I20 card temperature (TEMPERATURE parameter), and 8I20 status and fault information (STATUS and FAULT parametera).

A complete host based controller will also need position sensing. This may consist of quadrature encoders, SSI absolute encoders (including magnetic absolute types), or resolvers. All three types of position sensors are available in the Hostmot2 firmware suite.

SOFTDMC 8I20 INTERFACE

SoftDMC is a smart multi-axis motion controller with built in motion profile generator, programmable exception handling and a high performance FIFO based host interface. SoftDMC is a firmware option for Mesa's Anything I/O series of FPGA cards. For more information on SoftDMC, please consult the SoftDMC manual.

ANYTHING I/O INTERFACE DAUGHTER CARDS

8I20 compatible daughter cards are available to simplify connecting the 8I20 to Mesa's Anything I/O FPGA cards. One interface card is the 7I44. The 7I44 provides 8 channels of RS-422/RS-485 serial communication interface. The 7I44 uses RJ-45 connectors for the serial interface. These connectors are compatible with the 8I20 so a common CAT5 or CAT5E cable may be used to connect from the 7I44 to the 8I20. The 7I44 can also provide 5V power to the 8I20 subject to the CAT5/5E cable length restrictions.

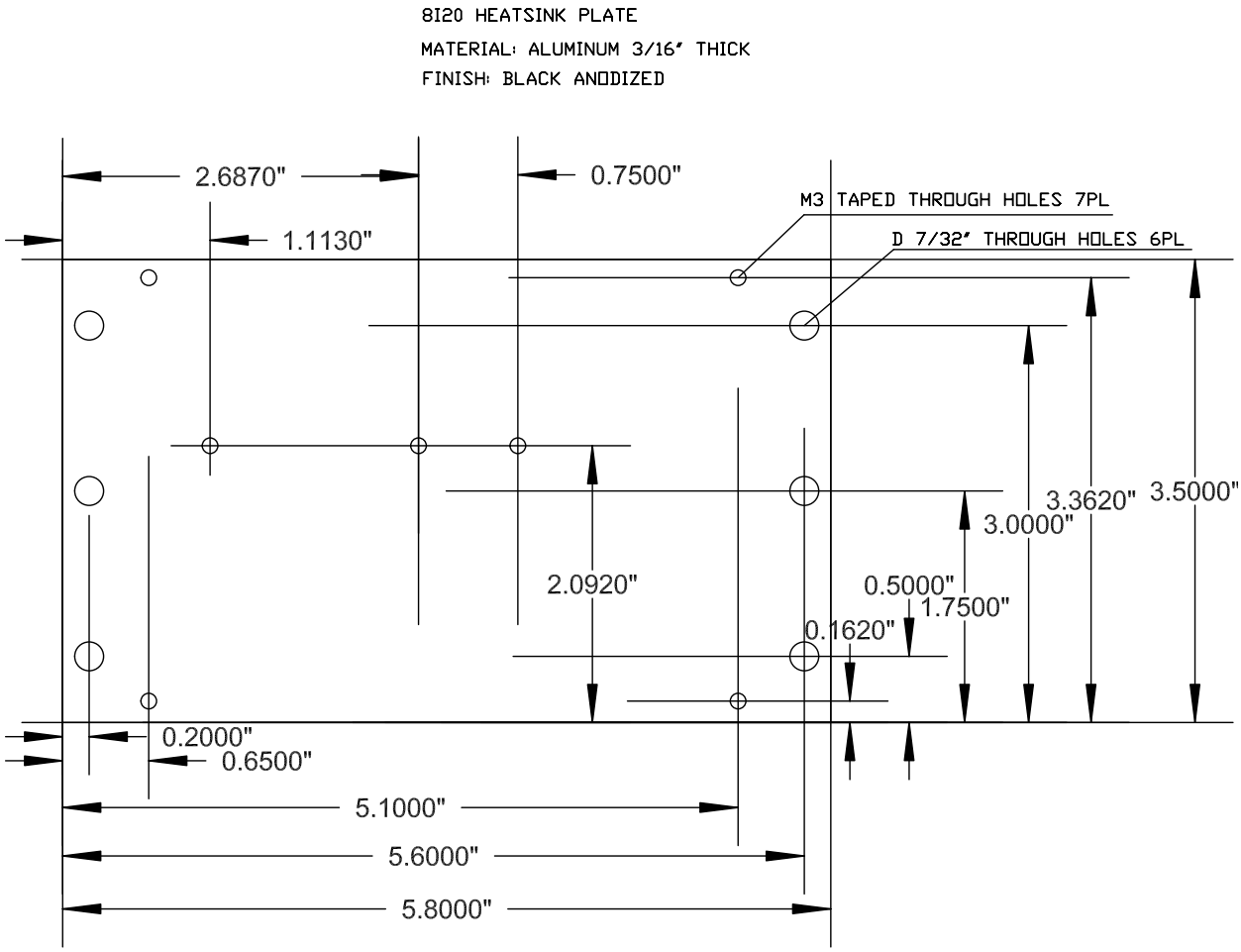
REFERENCE INFORMATION

SPECIFICATIONS

	MIN	MAX	NOTES
LOGIC SUPPLY VOLTAGE 5V	4.5V	5.5V	
5V CURRENT	----	400 mA	.
UNREGULATED SUPPLY VOLTAGE	8V	40V	
MOTOR SUPPLY VOLTAGE	35V	400V	
MAX MOTOR CURRENT 25C CASE	----	30A	RMS
MAX MOTOR CURRENT 70C CASE	----	15A	RMS
SWITCHING FREQUENCY	----	20KHz	Default is 12 KHz
V ISOLATION VBUS/LOGIC	3750V	----	RMS
V ISOLATION VBUS/CASE/EARTH	2500V	----	RMS
POWER DISSIPATION 4 KW LOAD	----	125W	
(340V 10A/PHASE 12 KHZ PWM	60 HZ MODULATION)		
THERMAL RESISTANCE	----	0.5C/W	RIGHT ANGLE
THERMAL RESISTANCE	----	0.1C/W	PARALLEL
TEMPERATURE -C VERSION	0^oC	70^oC	
TEMPERATURE -I VERSION	-40^oC	85^oC	

REFERENCE INFORMATION

HEATSINK PLATE DRAWING



REFERENCE INFORMATION

LBP

LBP is a simple binary master slave protocol where the host sends read, write, or RPC commands to the 8I20, and the 8I20 responds. All controller communication to the 8I20 is done via LBP. LBP commands always start with a command header byte. This header specifies whether the command is a read or write or RPC, the number of address bytes(0, or 2), and the number of data bytes(1 through 8).The 0 address size option indicates that the current address pointer should be used. This address pointer will be post incremented by the data size if the auto increment bit is set. RPC commands allow any of up to 64 stored commands to be executed in response to the single byte command.

LBP DATA READ/WRITE COMMAND

0	1	WR	RID	AI	AS	DS1	DS0

Bit 7.. 6 **CommandType:** Must be 01b to specify data read/write command

Bit 5 **Write:** 1 to specify write, 0 to specify read

Bit 4 **RPCIncludesData:** 0 specifies that data is from stream, 1, that data is from RPC (RPC only, ignored for non RPC commands)

Bit 3 **AutoInc:** 0 leaves address unchanged, 1 specifies that address is post incremented by data size in bytes.

BIT 2 **AddressSize:** 0 to specify current address, 1 to specify 2 byte address.

Bit 1..0 **DataSize:** Specifies data size, 00b = 1 bytes, 01b = 2 bytes, 10 b= 4 bytes, 011b = 8 bytes.

When multiple bytes are specified in a read or write command, the bytes are always written to or read from successive addresses. That is, a 4 byte read at location 0x21 will read locations 0x21, 0x22, 0x23, 0x24. The address pointer is not modified after the command unless the AutoInc bit is set.

REFERENCE INFORMATION

LBP

EXAMPLE LBP COMMANDS

Write 4 bytes (0xAA, 0xBB,0xCC,0xDD) to addresses 0x010,0x011,0x012,0x013 with AutoInc so that the address pointer will be left at 0x014 when the command is completed:

COMMAND BITS	CT1	CT0	WR	RID	AI	AS	DS1	DS0
LBPWrite: 2 add 4 data	0	1	1	0	1	1	1	0
Write Address LSB	0	0	0	1	0	0	0	0
Write Address MSB	0	0	0	0	0	0	0	0
Write data 0	1	0	1	0	1	0	1	0
Write Data 1	1	0	1	1	1	0	1	1
Write Data 2	1	1	0	0	1	1	0	0
Write Data 3	1	1	0	1	1	1	0	1

Write 2 more bytes (0xEE,0xFF) at 0x014 and 0x015:

COMMAND BITS	CT1	CT0	WR	RID	AI	AS	DS1	DS0
LBPWrite: 0 add 2 data	0	1	1	0	0	0	0	1
Write data 0	1	1	1	0	1	1	1	0
Write data 1	1	1	1	1	1	1	1	1

Read 8 bytes at 0x010,0x011,0x012,0x013,0x014,0x015,0x016,0x017:

COMMAND BITS	CT1	CT0	WR	RID	AI	AS	DS1	DS0
LBPRead: 2 add 8 data	0	1	0	0	0	1	1	1
Read Address LSB	0	0	0	1	0	0	0	0
Read Address MSB	0	0	0	0	0	0	0	0

REFERENCE INFORMATION

LBP

LOCAL LBP COMMANDS

In addition to the basic data access commands, there are a set of commands that access LBP status and control the operation of LBP itself. These are organized as READ and WRITE commands

LOCAL LBP READ COMMANDS

(HEX), all of these commands return a single byte of data.

0xC0 Get unit address

0xC1 Get LBP status

LBP Status bit definitions:

BIT 7 Reserved

BIT 6 Command Timeout Error

BIT 5 Invalid write Error (attempted write to protected area)

BIT 4 Buffer overflow error

BIT 3 Watchdog timeout error

BIT 2 Reserved

BIT 1 Reserved

BIT 0 CRC error

0xC2 Get CRC enable status (note CRCs are always enabled on the 8I20)

0xC3 Get CRC error count

0xC4 .. 0xC9 Reserved

0xCA Get Enable_RPCMEM access flag

0xCB Get Command timeout (character times/10 for serial)

0xCC .. 0xCF Reserved

0xD0 .. 0xD3 4 character card name

REFERENCE INFORMATION

LBP

LOCAL LBP READ COMMANDS

0xD5 .. 0xD7 4 character configuration name (only on some configurations)

0xD8 Get low address

0xD9 Get high address

0xDA Get LBP version

0xDB Get LBP Unit ID (Serial only, not used with USB)

0xDC Get RPC Pitch

0xDD Get RPC SizeL (Low byte of RPCSize)

0xDE Get RPC SizeH (High byte of RPCSize)

0xDF Get LBP cookie (returns 0x5A)

REFERENCE INFORMATION

LBP

LOCAL LBP WRITE COMMANDS

(HEX), all of these commands except 0xFF expect a single byte of data.

0xE0 Reserved

0xE1 Set LBP status (0 to clear errors)

0xE2 Set CRC check enable (Flag non-zero to enable CRC checking)

0xE3 Set CRC error count

0xE4 .. 0xE9 Reserved

0xEA Set Enable_RPCMEM access flag (non zero to enable access to RPC memory)

0xEB Set Command timeout (in mS for USB and character times for serial)

0xEC .. 0xEF Reserved

0xF0 .. 0xF6 Reserved

0xF7 Write LEDs

0xF8 Set low address

0xF9 Set high address

0xFA Add byte to current address

0xFB .. 0xFC Reserved

0xFD Set unit ID (serial only)

0xFE Reset LBP processor if followed by 0x5A

0xFF Reset LBP parser (no data follows this command)

REFERENCE INFORMATION

LBP

RPC COMMANDS

RPC commands allow previously stored sequences of read/write commands to be executed with a single byte command. Up to 64 RPC's may be stored. RPC write commands may include data if desired, or the data may come from the serial data stream. RPCs allow significant command compression which improves communication bandwidth. The 8I20 has some pre-loaded RPCs for normal current (torque) mode operation.

LBP RPC COMMAND

1	0	RPC5	RPC4	RPC3	RPC2	RPC1	RPC0

Bit 7..6 **CommandType:** must be 10b to specify RPC

Bit 5..0 **RPCNumber:** Specifies RPC 0 through 63

In the 8I20 LBP implementation, RPCPitch is 0x8 bytes so each RPC command has native size of 0x08 bytes and start 0x8 byte boundaries in the RPC table area. RPCs can cross RPCPitch boundaries if larger than RPCPitch RPCs are needed. The stored RPC commands consist of LBP headers and addresses, and possibly data if the command header has the RID bit set. RPC command lists are terminated by a 0 byte.

The RPC table is accessed at addresses 0 through RPCSize-1 This means with a RPCPitch of 0x8 bytes, RPC0 starts at 0x0000, RPC1 starts at 0x008, RPC2 starts at 0x0010 and so on.

Before RPC commands can be written to the RPC table,the RPCMEM access flag must be set. The RPCMEM access flag must be clear for normal operation.

REFERENCE INFORMATION

LBP

EXAMPLE RPC COMMAND LIST

This is an example stored RPC command list. Note RPC command lists must start at a RPCPitch boundary in the RPC table but an individual RPC list can extend until the end of the table. This particular RPC example contains 2 LBP commands and uses 7 bytes starting at 0x0028 (RPC5 for 0x08 pitch RPC table)

Command1. Writes two data bytes to address 0x10, 0x11 with 2 data bytes supplied by host

Command2. Reads two data bytes from address 0x12,0x13

COMMAND BITS	CT1	CT0	WR	RID	I	AS	DS1	DS0
LBPWrite: 2 add 2 data	0	1	1	0	0	1	0	1
Write Address LSB	0	0	0	1	0	0	0	0
Write Address MSB	0	0	0	0	0	0	0	0
LBPRead: 2 add 2 data	0	1	0	0	0	1	0	1
Read Address LSB	0	0	0	1	0	0	1	0
Read Address MSB	0	0	0	0	0	0	0	0
Terminator	0	0	0	0	0	0	0	0

The data stream for this RPC would consist of these 3 bytes:

COMMAND BITS	CT1	CT0	R5	R4	R3	R2	R1	R0
RPC 5	1	0	0	0	0	1	0	1
Data 0 for Command 1	0	1	0	1	0	1	0	1
Data 1 for Command 1	1	1	0	0	1	1	0	0

REFERENCE INFORMATION

CRC

LBP on the 8I20 uses CRC checking of all commands and data to insure validity. The CRC used is a 8 bit CRC using the same polynomial as the Dallas/Maxim one wire devices (X^8+X^5++X^4+X^0). The CRC must be appended to all LBP commands and all returned data will have a CRC byte appended. Commands with no returned data (writes or RPCs with no reads) will still cause a CRC byte to be returned, this CRC byte will always be 00H.

FRAMING

Since LBP is a binary protocol with no special sync characters, the packet framing must be determined by other methods. The 8I20s LBP implementation uses two different framing models depending on whether it is in setup mode or operate mode.

In setup mode, framing is done by determining the end of the command by pre-parsing the data stream. This is done so that relaxed interface timing is acceptable. Non-realtime systems such as Windows and Linux cannot guarantee exact serial data timing so timing based framing cannot be used. Timing based framing is still used to maintain synchronization in case of aborted packets or noise but the timeout is set to the maximum time (25.5 character times) or ~2.2 mS at 115200 baud. Because normal communication will be framed by pre-parsing, this timeout delay need not be inserted between subsequent commands.

When running in operate mode, the 8I20 uses timing based framing exclusively. Timing based framing is more robust in noisy environments where bad packets may be present, but it requires strict timing of the serial data stream. Packets sent to the 8I20 in real time mode must be sent in a single packet with less than the LBP command timeout between characters. The command timeout is set by SSLBP to be 4 character times (16 uSec at 2.5M baud). In operate mode, a delay of 16 uSec must always be inserted between packets.

REFERENCE INFORMATION

PARAMETERS

The 8I20 has many user settable parameters, but normally only a very few need be changed in normal operation The following is a short list of 8I20 parameters for reference only:

PARAMETER	TYPE	SCALING	FUNCTION
BRAKEOFFV	UINT	10 mV	Set working brake off voltage
NVBRAKEOFFV	UINT	10 mV	Set non-volatile brake off voltage
BRAKEONV	UINT	10 mV	Set working brake on voltage
NVBRAKEONV	UINT	10 mV	Set non-volatile brake on voltage
BUSV	UINT	10 mV	Read motor bus voltage
BUSOVERV	UINT	10 mV	Set working bus overvoltage threshold
NVBUSOVERV	UINT	10 mV	Set non-volatile bus overvoltage threshold
BUSUNDERV	UINT	10 mV	Set working bus undervoltage threshold
NVBUSUNDERV	UINT	10 mV	Set non-volatile bus undervoltage threshold
MAXCURRENT	UINT	10 mA	Set working full scale current
NVMAXCURRENT	UINT	10 mA	Set non-volatile full scale current

REFERENCE INFORMATION

PARAMETERS

DEADZONE	UINT	Set working pwm anti-deadzone
NVDEADZONE	UINT	Set non-volatile pwm anti-deadzone
FAULT	UINT	8I20 fault register
STATUS	UINT	8I20 status register

REFERENCE INFORMATION

PARAMETERS

KDP	UINT		Set working diloop p term
NVKDP	UINT		Set non-volatile diloop p term
KDI	ULONG		Set working diloop I term
NVKDI	ULONG		Set non-volatile diloop I term
KDIL	UINT		Set working diloop I limit
NVKDIL	UINT		Set non-volatile diloop I limit
KQP	UINT		Set working qiloop p term
NVKQP	UINT		Set non-volatile qiloop p term
KQI	ULONG		Set working qiloop I term
NVKQI	ULONG		Set non-volatile qiloop I term
KQIL	UINT		Set working qiloop I limit
NVKQIL	UINT		Set non-volatile qiloop I limit
DSETPOINT	INT		Set "Direct" current
QSETPOINT	INT		Set "Quadrature" current
ANGLE	UINT	2*PI/65536	Set reference angle, 65536 = 360 electrical degrees
TEMPERATURE	UINT	0.01C/	8I20 card temperature C in 1/100 degree per count.

REFERENCE INFORMATION

SSLBP

GENERAL

SSLBP is a firmware option to HostMot2s SSERIAL serial interface that allows simple communication to LBP based peripherals. SSERIAL is a part of the HostMot2 motion interface firmware for MESA's Anything-I/O FPGA cards.

REGISTER MAP

SSLBP has two global processor interface registers and three per channel remote device interface registers. For more details on mapping of these registers in HostMot2 memory space, see the REGMAP file that is included with the HostMot2 source distribution.

PROCESSOR INTERFACE REGISTERS

There are two processor interface registers, the COMMAND register and the DATA register. These registers allow low level communication to SSLBP's interface processor for issuing global commands, discovery, and debug operations.

COMMAND REGISTER

The commands register is a 16 bit register (right justified in the 32 bit interface) with the following format:

W	M	R	D	S	T	T	T	N	N	N	N	N	N	N	N

W = BIT 15 Write bit, set high for write commands

M = BIT 14 ROM enable/ reset bit, set high to reset processor / download ROM

R = BIT 13 Request bit, set high for read or write command

D = BIT 12 DoIt bit, set high for DoIt commands

S = BIT 11 Start/Stop bit, actual operation depends on N:

T = 0,0,0 Stop LBP interface

T = 0,0,1 Start LBP interface in normal mode

T = 1,1,1 Start LBP interface in setup mode

N bits determine which channels start or do data transfer with remote device. A set bit indicates that the corresponding channel will start or do a data transfer.

REFERENCE INFORMATION

SSLBP

DATA REGISTER

SSLBP has a global 8 bit data register for debug and custom setup purposes. This register allows access to internal SSLBP parameters. The data register is right justified in the 32 bit Hostmot2 register.

LOCAL READ OPERATIONS

The sequence used for reading a local SSLBP variable is as follows:

1. The parameter address ORed with the Request bit (bit 13) is written to the command register.

2. The host polls the command register until it reads as zero.

3. The host reads the parameter byte from the data register

LOCAL WRITE OPERATIONS

The sequence used for writing a local SSLBP variable is as follows:

1. The host polls the command register until it reads as zero.

2. The host writes the data byte to the data register

3. The host writes the command register with the the parameter address Ored with both the Request bit (bit 13) and the Write bit (bit 15)

NORMAL START

When the FPGA is first configured or after a STOP command, all local communication, error and status parameters are initialized and all LBP communication channels are idle. A normal START command begins to establish communications with all remote LBP devices. A normal start command is issued by writing a Start bit with type bits of 0,0,1 with a bit mask of the desired channels to start in the low byte (0x9NN) to the command register. Once a start command has been issued, all channels that are selected in the bit mask will be probed to determine if a LBP device exists. If a device exists on a channel, the SSLBP firmware will acquire the device type and device unit ID from the remote device. When the command completes (the command register is clear), the data register can be read to determine if all selected channels have started. A 1 bit in any position in the data register indicates that the corresponding channel has failed to start. If a channel has failed to start, more information about the failure can be determined by reading the CS register.

REFERENCE INFORMATION

SSLBP

8I20 DEVICE SPECIFIC SETUP

A normal start command does specific setup operations when it detects a 8I20 remote device. This setup includes clearing any faults, setting the qsetpoint current to 0 A, enabling the current control loop and setting the 8I20s watchdog timer to 40 mS. If no errors have occurred and all faults are clearable, the SSLBP firmware enters a "chatter" loop where it repeatedly asks the 8I20 for a cookie character. This keeps the 8I20's watchdog fed while waiting for the first DOIT command. Once a DOIT command has been executed, the firmware no longer "chatters" and the host interface must send DOIT commands at greater than 40 mS intervals or the 8I20s watchdog will bite, disabling its output.

STOP LBP INTERFACE

A STOP ALL command is issued to stop all channel communication. A STOP ALL followed by a START command can be used after a fault condition to re-establish communication with the remote LBP devices. Device discovery is only done once when START command is issued to a STOPed SSLBP. This means that if cabling or devices are changed, a STOP ALL command followed by a START command must be issued by the host to detect the changes.

STOP INDIVIDUAL CHANNELS

In addition to stopping all channels, a individual stop command can be issued. A individual stop command include a bitmask of the channels to stop in the least significant 8 bits of the command.

DOIT

In normal operation SSLBP is designed to write data from local registers to the remote device and read remote device data for presentation to the host on a real time basis. Synchronization is accomplished with the DOIT command. When a DOIT command is written, all channel data from the host is sent to the remote devices and receive data is requested. Completion of the DOIT command is signaled by SSLBP clearing the COMMAND register. A DOIT command contains the DOIT bit and an 8 bit mask in the 8 LSBs that selects the channels that will transfer data. After DOIT command completion the data register will contain a bit mask of channel status data. If any bit is set in the data register, it indicates a problem with the transfer (all zeros indicates no faults or errors). The data returned after a DOIT command can be used to minimize host access cycles by avoiding the need to read the per channel status registers. If detailed fault information is desired, the CS register and Interface1 register can be read on any channel that shows a failed transfer.

REFERENCE INFORMATION

SSLBP

INTERFACE REGISTERS

Three per channel interface registers are used to pass information from the host to the remote LBP device and from the remote LBP device to the host. The registers are the CS Register, Interface0 register and Interface1 register. These registers are all 32 bits in width and read/write.

CS REGISTER

The CS register is used for local SSLBP, and remote LBP device status and control information. Read access returns status information in both normal and setup mode. In normal mode, writes to the CS register are not used. When read, the CS register has the following format:

Byte3 = Remote LBP device mode

= 0x80 for 8I20

= 0x74 for 7I64

Byte2 = Communication state code (debug only)

Byte1 = Communication status code (0x00 for OK)

Bit 7 = CommunicationNotReady

Bit 6 = NoRemoteID

Bit 5 = CommunicationError

Bit 0 = RemoteFault

Byte0 = Communication error code (sticky, cleared only by stop)

Bit 7 = TooManyerrors

Bit 4 = ExtraCharacterError

Bit 3 = TimeoutError

Bit 2 = OverrunError

Bit 1 = InvalidCookieError

Bit 0 = CRCError

REFERENCE INFORMATION

SSLBP

INTERFACE REGISTER 0

Interface register 0 is a general purpose 32 bit read/write register for transferring data to and from the remote LBP device, *After a start command and valid ready status, interface register 0 reports the 32 bit device unit number*. After DOIT has been asserted bit definitions in interface register 0 are device specific.

8I20 SPECIFIC INTERFACE REGISTER 0 DEFINITIONS

Reads:

After DOIT is asserted:

MSW = Bus voltage in 10s of mV (unsigned 16 bit number)

LSW = Card Temperature in ^{o}C (insigned 16 bit number)

Writes:

MSW = QSETPOINT current, signed 16 bit number, 32767 sets current to +MAXCURRENT and -32767 sets current to -MAXCURRENT

LSW = ANGLE, unsigned 16 bit number, 0 to 65535 = 0 to 359.9945 degrees.

INTERFACE REGISTER 1

Interface register 1 is a general purpose 32 bit read/write register for transferring data to and from the remote LBP device. Bit definitions in interface register 1 are device specific.

8I20 SPECIFIC INTERFACE REGISTER 1 DEFINITIONS:

Reads:

MSW = 8I20 FAULTS

LSW = 8I20 STATUS

Writes:

Writes to interface register 1 are not used in normal mode.

REFERENCE INFORMATION

SSLBP

NORMAL MODE OPERATION

In normal mode the sequence of operations is as follows:

1. Issue stop command, wait for COMMAND register clear to verify stop command completion.

2. Issue normal START command (0x9NN) with bitmask (NN) of channels to start.

3. Wait for COMMAND register clear to verify start command completion. (may be many mS)

4. Read data register to verify that all selected channels started (a 1 in any channel position bit means a fault in the channel that the bit represents)

5. Read device unit number (This can only be read before DOIT has been asserted)

6. Check command register, if not clear, cycle time is too short

7. Check data register, any 1 bits indicate previous DOIT command failed for in the corresponding channels

8. Write per channel output data (CURRENT AND ANGLE for 8I20) to interface 0 register

9. Read Interface register 0 for bus voltage and temperature, note that this is stale data from previous cycle (and invalid the first cycle)

10. Write DOIT command = 0x10NN where NN is the bit mask of channels to initiate transfers.

11. Wait for next cycle, at next cycle time, loop to state 6

This sequence can be modified if a read-modify-write sequence is required, but for the 8I20, the read data is not time critical so the read data timing is not important.

SETUP START

When the FPGA is first configured or after a stop all command, all LBP communication channels are idle. A SETUP START command first initializes and all local communication, error and status parameters and begins to establish communications with all remote LBP devices. Unlike the NORMAL START command, SETUP START does no device specific setup but instead creates a pass-through access mode that allows the host to read or write any remote LBP device parameter. This allows simple utilities to setup 8I20 volatile and non-volatile parameters.

REFERENCE INFORMATION

SSLBP

SETUP MODE OPERATION

In setup mode the SSLBP interface is used as a passthrough device to allow reading and writing parameters to the remote LBP device.

REMOTE READ EXAMPLE:

For a remote word read, the sequence of operations is as follows:

1. Issue a STOP command, wait for COMMAND register clear to verify stop command completion.

2. Issue a setup START command (0xFNN) with bitmask (NN) of channels to start

3. Wait for COMMAND register clear to verify start command completion. (may be many mS)

4. Read data register to verify that all selected channels started (a 1 bit means a fault in the channel that the bit represents)

5. Write LBP word read command (0x45) in the MSByte ORed with the parameter address to the selected channels CS register. (0x4500PPPP)

6. Issue a DOIT Command

7. Wait for the command register to be clear

8. Check that the data register is clear, any set bits indicate an error

9. Read the returned data from the selected channels Interface0 register

10. Repeat from step 5 for any additional remote data reads

REFERENCE INFORMATION

SSLBP

REMOTE WRITE EXAMPLE:

For a remote word write, the sequence of operations is as follows:

1. Issue a STOP command, wait for COMMAND register clear to verify stop command completion.

2. Issue a setup START command (0xFNN) with bitmask (NN) of channels to start

3. Wait for COMMAND register clear to verify start command completion. (may be many mS)

4. Read data register to verify that all selected channels started (a 1 bit means a fault in the channel that the bit represents)

5. Write the new parameter data to the selected channels Interface0 register (right justified)

6. Write LBP word write command (0x65) in the MSByte ORed with the parameter address to the selected channels CS register. (0x6500PPPP) 6. Issue a DOIT Command

7. Wait for the command register to be clear

8. Check that the data register is clear, any set bits indicate an error

9. Repeat from step 5 for any additional remote parameter writes