Ethernet Access Library for QB

Preliminary Specification

E. Hazen – 3 October 2006

J. Raaf – last modified 28 March 2007

This function library is designed to provide access to the Ethernet daughterboard on the QB module. A preliminary version of TCP data transfer is supported. *Revision 6 firmware is required for TCP support.*

List of Functions

`int EthSetVerbosity( int level )`

Set level of information and error reporting.

`level` Integer between 0 and 6

0 = Do not print messages to screen

1 = Print only error messages

2 = Print errors and some infomational messages

3 = Print errors, some informational messages, and received data

4 = Print errors, more informational messages, and received data

5 = Print errors, all informational messages, and received data

6 = Print all messages

`int EthOpen( const char *ipAd, unsigned int udpPort)`

Open connection to Ethernet on QB. Returns a positive integer handle to be used for further communication, or a negative error code if open fails. EthOpen will try MAX_ATTEMPTS times to establish connection and send UDP test packet for verification, BCP field "ID" is incremented at each new attempt. MAX_ATTEMPTS currently defined as 256.

`IpAd` I/P address in standard numbers/dots format (i.e. "192.168.1.1")

`udpPort` Port number to use for connection (SiTCP defines as 0x1234)

Return codes:

1	Success
-1	UDP socket creation failed
-2	Error in initial UDP packet sendto()
-3	Too many connections

-4 Timeout in initial UDP packet recvfrom()

-5 Error in initial UDP packet recvfrom()

-6 No ACK flag in UDP test packet recvfrom()

-7 Mis-matched packet ID in UDP returned test packet

-11 TCP socket creation failed

-12 Error establishing TCP connection

-13 Error setting TCP socket options

`int EthUDPOpen( const char *ipAd, unsigned int udpPort)`

Open only UDP connection to Ethernet on QB. Returns a positive integer handle to be used for further communication, or a negative error code if open fails. EthUDPOpen will try up to MAX_ATTEMPTS times to establish connection and send UDP test packet for verification, BCP field "ID" is incremented at each new attempt. MAX_ATTEMPTS currently defined as 256.

`IpAd` I/P address in standard numbers/dots format (i.e. "192.168.1.1")

`udpPort` Port number to use for connection (defined in SiTCP as 0x1234)

Return codes:

1 Success

-1 UDP socket creation failed

-2 Error in initial UDP packet sendto()

-3 Too many connections

-4 Timeout in initial UDP packet recvfrom()

-5 Error in initial UDP packet recvfrom()

-6 No ACK flag in UDP test packet recvfrom()

-7 Mis-matched packet ID in UDP returned test packet

```
int EthTKOSingle(  int handle,
          unsigned int f,
          unsigned int sa,
          unsigned short int* data,
          int *st);
```

Perform a single TKO read/write operation. Return 1 on success, or a negative error code on

failure. EthTKOSingle will try up to MAX_ATTEMPTS times to re-send the packet before returning failure.

`handle`	handle returned by EthOpen() or EthUDPOpen()
`f`	TKO function code 0-15 (0-7 are read, 8-15 are write)
`sa`	TKO sub-address (11 bits)
`data`	TKO read/write data
`st`	Status of last operation bit 0 = 1 if module returned 'Q' response bit 1 = 1 if module returned YSSIR* response

Return codes:

1	Success
-1	Invalid handle
-2	Invalid TKO command
-3	Error sending packet
-4	Timeout while waiting for response packet
-5	Error while waiting for response packet
-6	No ACK flag in received packet
-7	Mis-matched BCP packet ID in returned packet
-8	Error sending packet to request status of TKO operation
-9	Timeout while requesting status of TKO operation
-10	Error requesting status of TKO operation

`int EthUDPRead( int handle, unsigned int addr, unsigned short *data);`

Perform a two-byte UDP read operation. Return 1 on success, or a negative error code on failure. EthUDPRead will try up to MAX_ATTEMPTS times to re-send/receive.

handle	handle returned by EthOpen() or EthUDPOpen()
addr	register address. Must be even number in range 0-0x7ffe
data	pointer to buffer for read data

Return codes:

1	Success
-3	Error sending packet

-4 Timeout while waiting for response packet

-5 Error while waiting for response packet

-6 No ACK flag in received packet

-7 Mis-matched packet ID number in returned packet

```
int EthUDPWrite( int handle,
          unsigned int addr,
          unsigned short *data);
```

Perform a two-byte UDP write operation. Return 1 on success, or a negative error code on failure. EthUDPWrite will try up to MAX_ATTEMPTS times to re-send/receive packet.

handle handle returned by EthOpen() or EthUDPOpen()

addr register address. Must be even number in range 0-0x7ffe

data pointer to buffer for read data

Return codes: same as for EthUDPRead()

```
int EthSetMemoryTestMode( int handle, unsigned int onoff );
```

Toggle DB "memory test" mode.

handle handle returned by EthOpen() or EthUDPOpen()

onoff should be '1' to turn ON memory test mode, '0' to turn OFF.

Return codes: same as for EthUDPWrite()

```
int EthSetSDSDebugMode( int handle, unsigned int onoff );
```

Toggle DB "SDS debug" mode. In this mode, no DB header/trailer/warning cells are inserted into data stream.

handle handle returned by EthOpen() or EthUDPOpen()

onoff should be '1' to turn ON SDS debug mode, '0' to turn OFF.

Return codes: same as for EthUDPWrite()

```
int EthSetTCPByteOrder( int handle, unsigned short int byteorder );
```

Toggle DB byte order of TCP data stream. Generally, this function should not be necessary since endianness is checked automatically during EthOpen() where the appropriate value is set (big-

endian or little-endian) for the computer on which it is running.

handle — handle returned by EthOpen() or EthUDPOpen()

byteorder — '1' for little-endian TCP data

'0' for big-endian TCP data

Return codes:

1 Success

-1 Invalid byte order

`int EthClose( int handle );`

Close connection to a module. Return 1 on success, or a negative error code on failure.

Closes UDP and TCP sockets and connections.

Return codes:

1 Success

-1 Failed to close socket connections.

```
int EthTCPReadBytes( int handle,
          unsigned char *databuf,
          int databuf_maxbytes,
          int *numbytes);
```

Perform a non-blocking TCP read of single bytes. Always reads with TKO F=0, SA=0. Reads any data available from SiTCP, then returns immediately. Can return odd number of bytes. *Note that the user should only mix multiple calls to different TCPRead functions in the same program AT HIS/HER OWN RISK. Data stream may become unsynchronized if different TCPRead functions are called without careful thought!!*

Return 1 on success, or a negative error code on failure.

handle — handle returned by EthOpen()

databuf — pointer to buffer for read data (1-byte chars)

databuf_max — maximum size of buffer in bytes

numbytes — EthTCPReadBytes() sets to number of bytes read on return

Return codes:

1 Success

-1 Connection reset. Error closing TCP socket.

-2 Connection reset. Error recreating TCP socket.

-3 Error establishing TCP connection

-10 TCP recv() failed

```
int EthTCPRead16BitWords( int handle,
                  uint16_t *databuf,
                  int databuf_maxbytes,
                  int *numbytes);
```

Perform a non-blocking TCP read of 16-bit words to a buffer. Always reads with TKO F=0, SA=0. Reads any data available from SiTCP, then returns immediately. Will not return odd number of bytes – only returns full words. If partial word is received, it is saved until next call to EthTCPRead16BitWords and prepended to start of databuf during next call. *Note that the user should only mix multiple calls to different TCPRead functions in the same program AT HIS/HER OWN RISK. Data stream may become unsynchronized if different TCPRead functions are called without careful thought!!*

Return 1 on success, or a negative error code on failure.

handle handle returned by EthOpen()

databuf pointer to buffer for read data (uint16_t = unsigned 16-bit integer)

databuf_max maximum size of buffer in bytes

numbytes EthTCPRead16BitWords() sets to number of bytes read on return

Return codes:

1 Success

-1 Connection reset. Error closing TCP socket.

-2 Connection reset. Error recreating TCP socket.

-3 Error establishing TCP connection

-10 TCP recv() failed

```
int EthTCPRead6ByteCells( int handle,
                  uint16_t *databuf,
                  int databuf_maxbytes,
                  int *numbytes);
```

Perform a non-blocking TCP read of 6-byte cells (3 16-bit words) to a buffer. Always reads with TKO F=0, SA=0. Reads any data available from SiTCP, then returns immediately. Will not return odd number of bytes – only returns full cells. If partial cell is received, it is saved until next call to EthTCPRead6ByteCells and prepended to start of databuf during next call. *Note that the*

user should only mix multiple calls to different TCPRead functions in the same program AT HIS/HER OWN RISK. Data stream may become unsynchronized if different TCPRead functions are called without careful thought!!

Return 1 on success, or a negative error code on failure.

handle — handle returned by EthOpen()

databuf — pointer to buffer for read data (uint16_t = unsigned 16-bit integer)

databuf_max — maximum size of buffer in bytes

numbytes — EthTCPRead16BitWords() sets to number of bytes read on return

Return codes:

1	Success
-1	Connection reset. Error closing TCP socket.
-2	Connection reset. Error recreating TCP socket.
-3	Error establishing TCP connection
-10	TCP recv() failed

`int EthReboot( int handle, int sector);`

Reload FPGA firmware from specified flash sector. Reset all DB registers to default values.

handle — handle returned by EthOpen() or EthUDPOpen()

sector — 0 for default sector. 1 for backup sector

Return codes:

1	Success
-1	Not a valid sector

`int EthSDRAMTest( int handle, int clearFIFO );`

Perform test of SDRAM on daughter board. Pseudo-random sequential data are written to SDRAM instead of SDS data. Data generated by a 16-by LFSR with bit 0 input as an XNOR of bit 15, 14, 12, and 3 as input of bit 0. Data are transmitted via TCP connection and verified by generating the same data sequence on the computer side. Number of errors in the received TCP data are printed to stdout.

Return codes:

1	Success

Appendix – Daughterboard Protocol

The protocol used to communicate with the daughterboard is the BCP defined by Tomohisa Uchida (KEK). Each BCP transaction requires one UDP packet set to the board and one reply packet returned by the board. The packet contents are as follows:

Byte Number	Name	Description
0	Ver.[3:0] / Type [3:0]	Should be 0xFF (test version)
1	Command[3:0] / Flag [3:0]	Command 0xC = Read operation 0x8 = Write operation Flag, valid only ACK packet [3] = ACK packet [2:1] = always zero [0] = Bus error
2	ID	Number to identify, any number can be used.
3	Length	Length of read / write access
4	Address [31:24]	Read / Write Address
5	Address [23:16]	Read / Write Address
6	Address [15: 8]	Read / Write Address
7	Address [7: 0]	Read / Write Address
8	Write data [7:0]	1st write data
:		
7+N	Write data [7:0]	The last (N-th) write data

The Address[31:0] field is used as follows:

Address[31:16]	Ignored
Address[15]	'1' for TKO operations. '0' for local operations
Address[14:12]	TKO function code bits 2-0 (bit 3 is implied by read/write)
Address[11:1]	TKO sub-address bits 10-0

Addresses in the range 0x0000-0x7fff are reserved for on-board registers. All registers are two bytes wide and thus require the BCP 'Length' field set to 2. The following registers are defined ***for firmware version < 0x0010***:

0x500 (read-only)	last TKO readback data
0x502 (read-only)	last TKO write data
0x504 (read-only)	last TKO command bits 15-12 are TKO function bit 11 is always '0' bits 10-0 are TKO sub-address
0x506 (read-only)	last TKO operation status bit 0 is '1' if Q response returned bit 1 is '1' if YSSIR* response returned
0x507 (read/write)	G_TRIG count threshold Start SDS when this many G_TRIG seen
0x508 (read/write)	test register
0x50a (read-only)	my_MSS bits 11-0 (bits 15-12 always '0')
0x50c (read-only)	my_TCP_PORT bits 15-7
0x50e (read-only)	my_UDP_PORT bits 15-7
0x510 (read-only)	TCP_version bits 15-0
0x512 (read-only)	TCP_version bits 31-16
0x514 (read-only)	calculated CRC of SSN (serial number)
0x516 (read-only)	bit 0 '1' if SSN read completed bit 1 '1' if SSN CRC check succeeded bit 2 '1' if local acknowledge timeout occurred bits 3-6 always '0' bit 7 '1' if FPGA configured from backup sector bits 15-8 FPGA firmware revision
0x518 (read-only)	SSN bits 15-0
0x51a (read-only)	SSN bits 31-16
0x51c (read-only)	SSN bits 47-32
0x51e (read-only)	SSN bits 63-48