LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note **LIGO-T060212-00 - D** 9/6/06

CONVERTER UPLINK

Paul Schwinberg, Josh Myers, Daniel Sigg

Distribution of this draft:

all

This is an internal working note
of the LIGO Project.

LIGO Hanford Observatory
P.O. Box 159
Richland, WA 99352
Phone (509) 372-8106
FAX (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (504) 686-3100
FAX (504) 686-7189
E-mail: info@ligo.caltech.edu

California Institute of Technology
LIGO Project - MS 51-33
Pasadena CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS NW17-161
Cambridge, MA 01239
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

Table of Contents

1 Introduction and Overview

This note describes the uplink interface of the converter units. The interface consists of 2 physical gigabit ethernet ports. SFP fiber transceivers are used and can therefore support either single mode or multi mode fibers. Both physical interfaces can be used independently. For example, one could be dedicated for a servo control system whereas the other one could be used by a data acquisition system.

The protocol is based on a basic ethernet frame without any IP extensions. This should allow for simple and low latency interface on the computer end. Each interface has to be programmed through its port to select the converter channels and to select the acquisition rate.

The converter unit operates at a fixed sampling rate of 2^{19} Hz (524.288 kHz). An optional decimation filter is available for each channel. The user programmed acquisition rate must be an integer fraction of the sampling rate and not lower than 1 Hz. It is also possible to define a processing rate which is slower than the acquisition rate. This will then lead to multiple data samples per converter transferred by each ethernet frame.

Since all channels of one acquisition cycle have to be copied to the ethernet frame within one sampling period, a maximum number of about 100 channels can be transferred at once. It is however possible to increase this number by shifting the start of the acquisition cycle for some channels by an integral of the sampling period. The uplink supports jumbo ethernet frames up to 9000 bytes long. In reality however, the number of channel data words which can be transferred by each processing cycle is limited by the on chip FIFO buffers. For both input and output channels the limit is about 1000 channels per FIFO.

To allow for greater flexibility each gigabit ethernet interface supports up to 4 independent FIFO buffers or data queues in both directions. Each of this data queues can be programmed independently to support different data rates and to support interfacing multiple hosts. The only limitation is that the 4 queues access the converters one after the other. Meaning there is a transfer limit of 100 channels total per sampling period. However, each queue can easily be aligned to a different sampling period to avoid this problem.

The interface also supports binary input and output. A binary input will be treated just like a 32 bit analog-to-digital value, whereas a binary output will be treated like a 32 bit digital-to-analog value. For this application the decimation filter can be bypassed.

Section 2 gives an overview of the timing convention. Section 3 goes into more details about the functioning of the uplink interface. And section 4 gives a detailed description of the used ethernet protocol.

2 Timing and Clock Signals

The following timing and clock signals are used by the converter board:

i) Master clock at 2^{26} Hz (67.108864 MHz). This clock is locked to the 2^{23} Hz (8.388608 MHz) clock from the timing distribution.

ii) 1 PPS (one pulse per second). The 1 PPS is synchronized with the GPS clock and also derived from the timing distribution system.

iii) The sampling clock. This is the conversion rate of the ADCs and DACs. For a low noise ADC such as the AD7679 this clock runs at 2^{19} Hz (524.288 kHz) and is derived from the master clock and synchronized by the 1 PPS.

iv) The acquisition clock. This is the rate the data is acquired and is typically a decimated version of the sampling clock. Most often its rate is 2^{14} Hz (16.384 kHz). This clock is aligned with the sampling clock but its rising edge can be shifted relative to the 1 PPS. With the typical rates above there are 32 ($2^{19}/2^{14}$) possibilities to align the acquisition clock to the sampling clock. Basically, this allows the acquisition clock to be adjusted in small increments within a processing cycle.

v) The processing clock. This is the rate the data processing is clocked back on the computer side. It is also a power of two in frequency. It is also the rate the data is shipped through the uplink interface. In most circumstances its rate is the same as the acquisition rate. However, it is always aligned to the 1 PPS. If the acquisition rate is higher than the processing rate multiple data consecutive converter words are shipped per uplink request.

vi) The uplink clock. This is the clock of the uplink interface. For gigabit ethernet and 16 bit words this rate is 62.5 MHz, or about 7 % lower than the master clock. It is not aligned with the 1 PPS.

In Figure 1 an example of a timing diagram is shown. At time   the sample clock and the processing clock are aligned to the 1 PPS. The period of the sampling clock is about 2 μs and the period of the processing clock is about 61 μs (to   ). There are 32 sampling clock periods (   ) for every processing clock. In this example the ADC conversion clock is applied at time   in the middle of a sampling period. The ADC then requires somewhat less than 2 μs to complete the conversion (until about   ). The ADC is then read through the serial interface and the data is available in the FPGA at time   . The time between   and   is used by the decimation filter. This sampling process is repeated at the sampling rate using a pipeline. The acquisition clock is at the same rate as the processing clock but in this example delayed by 3 clock cycles   . Data will be ready for sending through the uplink interface at time   . Converter data which has passed through the decimation filter and which is ready at each sampling period is ignored in between the acquisition clock—or in other words in this example only every 32nd converter word is read out.

The sampling clock is typically fixed and depends on the exact converter chips which is used. However, the user can specify both the acquisition rate and the processing rate as well as the offset of the acquisition clock relative to the processing clock (in increments of the sampling period). Obviously, the ratio between the acquisition rate and the sampling rate determines the corner frequency of the decimation filter.

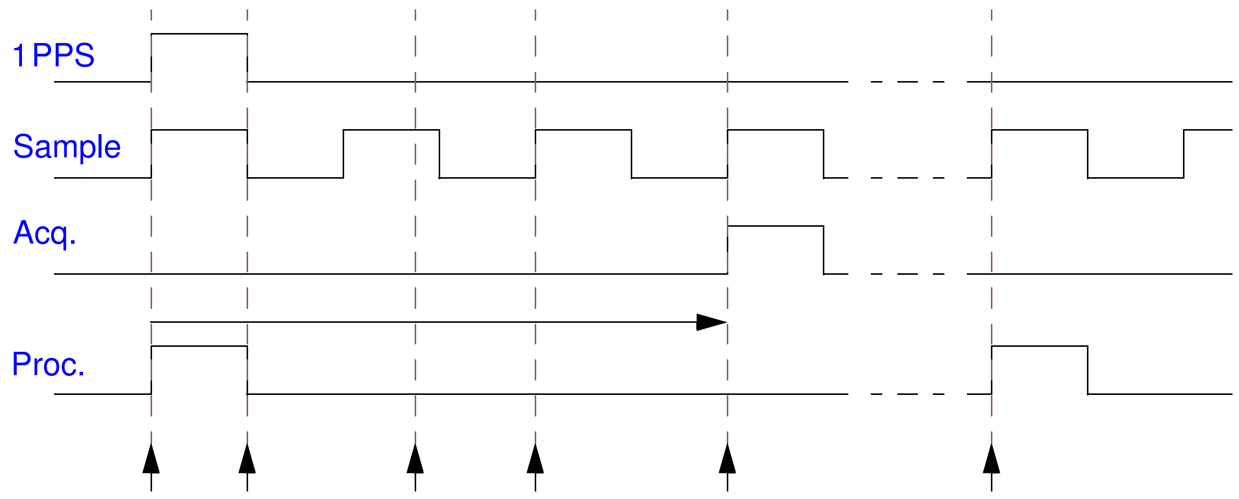

Figure 1: Timing diagram example.

3 Uplink Block Diagram

Figure 2 shows block diagrams of the receiver and transmitter sections of the uplink interface. Data is received through gigabit ethernet and forwarded to a ethernet MAC (Media Access Controller). This is part of the available resources of the selected FPGA. The gigabit ethernet interface is fiber based. After the MAC a priority arbiter decides whether the ethernet frame goes in the high priority FIFO (low latency) or the low priority FIFO. The high priority FIFO handles converter data (in this case DAC data) only, whereas the low priority FIFO handles all the other requests such as configuration commands and status information. The decision whether an ethernet frame is low or high priority is based on the protocol subtype.

The transmitter block diagram is essentially the same as the receiver one—just with the data flow arrows pointing the opposite way. The arbiter will always give priority to the high priority FIFO. Furthermore, it will only accept ethernet frames from the low priority FIFO in the middle of the acquisition cycle to prevent conflicts and make sure converter data is sent to the uplink interface without delay.

As a matter of fact both transmit and receive sections support up to 4 high priority queues. The idea is to be able to support multiple destinations for the sampled data. For example, there can be one data queue dedicated for the servo control system and another one for the data acquisition system. Furthermore, it allows to support multiple data rates simultaneously. A diagnostics system may request data at full rate while a servo control system is running. Conversely, on the receiving side multiple converters can be fed from different processing systems. A converter unit

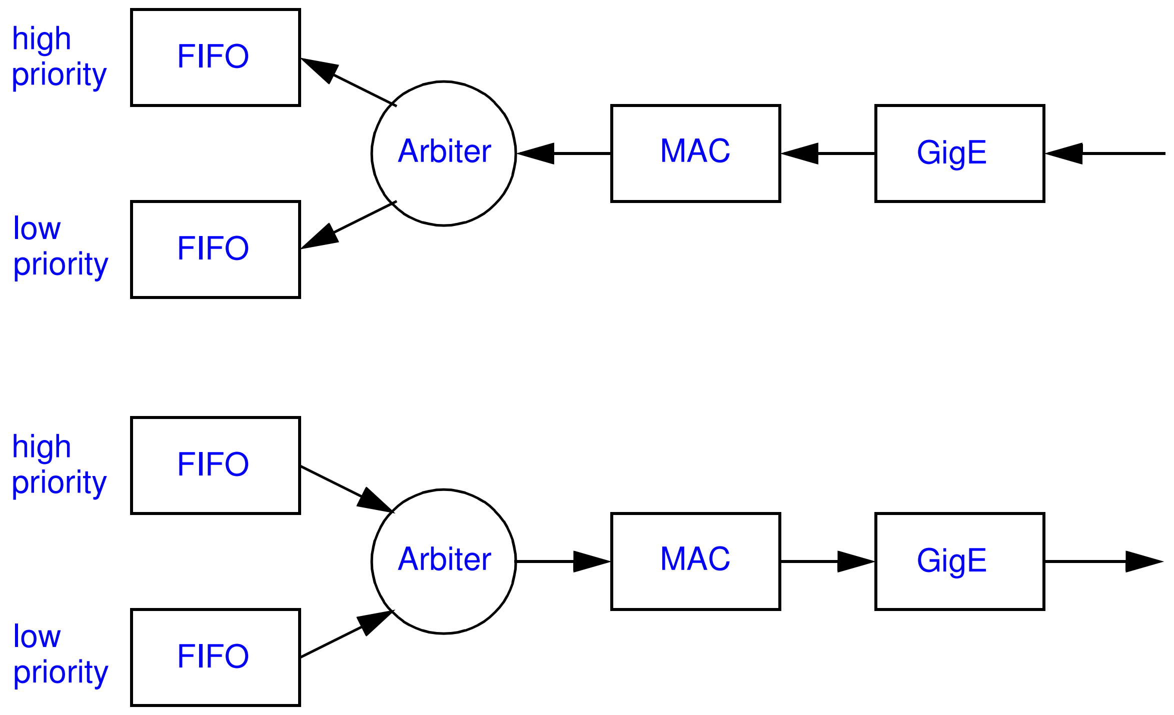

Figure 2: Block diagram of receiver and transmitter link interface.

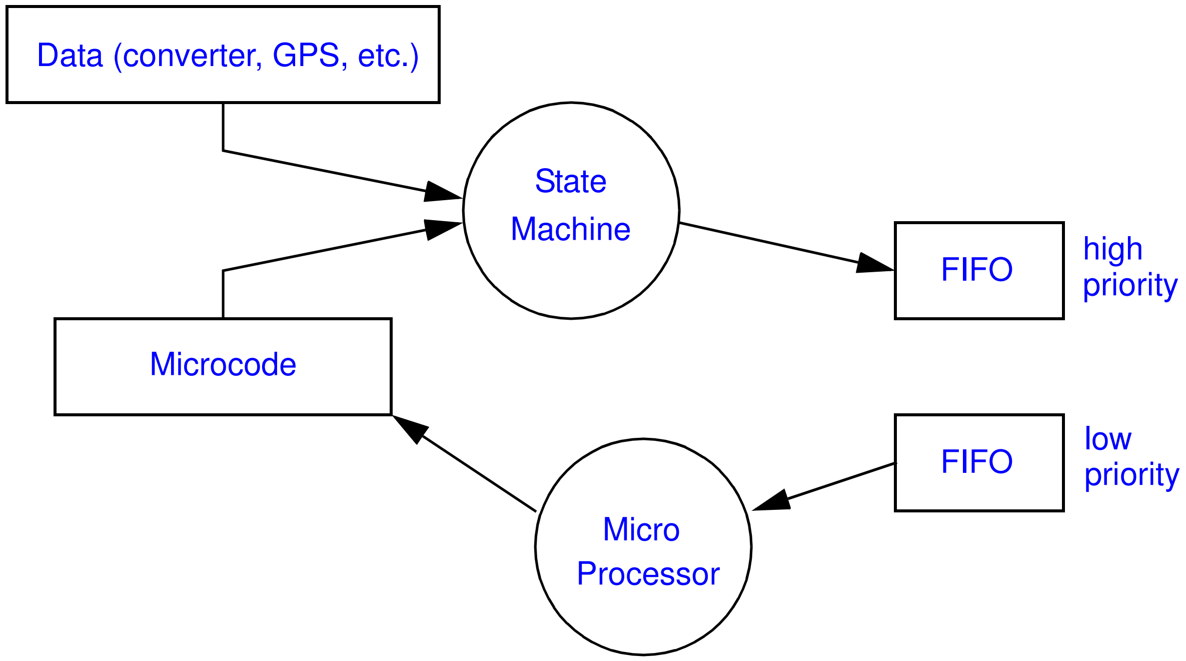

Figure 3: Block diagram of the transmitter state machine.

can also support more than a single physical uplink interface. This allows to keep data path from different systems—such as data acquisition and servo control system—completely separated.

Figure 3 shows the transmitter state machine which writes to the transmitter FIFOs, respectively. The FIFOs also mark the clock boundary between the uplink clock and the converter clocks. On the MAC side 16 bit words are processed at a rate of 62.5 MHz, whereas on the state machine side they get processed at a rate of 67.108864 MHz. The state machine is programmable by microcode which is stored in a dual ported RAM. It is triggered every time an acquisition cycle starts.

The receiver state machine is similar to the transmitter one but with the data flow going the opposite direction. The state machine is triggered every time a frame is received.

3.1 Transmit Engine

The transmit engine consists of 5 transmission queues: A through D and LP (low priority). The FIFOs associated with queues A through D are fed by the microcode state machine, whereas the LP FIFO is fed by the microprocessor and handles lower priority configuration and status requests.

3.1.1 Transmit Arbiter

The transmitter FIFOs are 17 bits wide. 16 bits are copied to the ethernet MAC as data whereas the 17th bit serves to mark the end of frame. Each FIFO also has a ‘go’ bit which is used to start the ethernet frame transmission. Up to 16 ‘go’ commands can be stacked up and therefore up to 16 ethernet frames can be backed up in a FIFO—assuming there is enough storage space. The transmit arbiter also provides a status word which indicates an overflow or underflow in each of the FIFOs as well as an error in one of the ‘go’ command bits. If multiple frames are ready at the same time the arbiter will first empty the FIFO of queue A, then move to B, C, D and finally LP. Furthermore, the LP queue is only served in between data transmissions. This window can be set by the management processor, see Table 1.

Table 1: Arbiter configuration in the transmit IO space.

Configuration Word	Offset	Access	Description
LPStart	0x0030	R/W	This word indicates the beginning of the low priority queue window.
LPStop	0x0034	R/W	This word indicates the end of the low priority queue window.
LPMask	0x0038	R/W	This word indicates the mask which is used to determine the window alignment.
TransmitStatus	0x003C	R	The lower 16 bits represent the status of the transmit arbiter. (Writing to this address will send data to the EMAC!) Bit 0 (LSB): FIFO A overflow Bit 1: FIFO A underflow Bit 2: FIFO A too many ‘go’ commands Bit 3: FIFO B overflow Bit 4: FIFO B underflow Bit 5: FIFO B too many ‘go’ commands Bit 6: FIFO C overflow Bit 7: FIFO C underflow Bit 8: FIFO C too many ‘go’ commands Bit 9: FIFO D overflow Bit 10: FIFO D underflow Bit 11: FIFO D too many ‘go’ commands Bit 12: FIFO LP overflow Bit 13: FIFO LP underflow Bit 14: FIFO LP too many ‘go’ commands Bit 15: arbiter transmit error

The valid transmit window of the LP queue is computed as follows:

$$LP_{valid} = (cnt \wedge LPmask > LPStart \wedge LPmask) \wedge (cnt \wedge LPmask \le LPStop \wedge LPmask) \qquad (1)$$

where cnt is a counter value derived from the 67.108864 MHz clock and synchronized with the 1 PPS from the timing system. It counts from 0 to 67108863 every second.

Example: If the frame transmit rate is 16384 Hz, the LPMask has to be 0x00000FFF. Meaning only 12 bits of the counter are used and thus it repeats every 61 μs. The LPStart and the LPStop values could then be set to 0x00000800 and 0x00000A00, respectively. This would allow for the transmission of low priority ethernet frames from 30.5 μs to 38.2 μs into the cycle. A valid mask must always be filled with ones from the right.

3.1.2 Transmit Microcode State Machine

3.1.2.1 Programming the Acquisition Rate

The transmit microcode state machine consists of 4 microcode tables which are associated with transmission queues A through D. The start of each microcode state machine is programmed through the IO configuration space of the management processor. Table 2 lists the configuration words.

Table 2: Acquisition rate configuration in the transmit IO space.

Configuration Word	Offset	Access	Description
ATRIG	0x0000	R/W	This word indicates the start condition for the microcode state machine A. Bit 0 is used as an enable for the state machine.
BTRIG	0x0004	R/W	This word indicates the start condition for the microcode state machine B. Bit 0 is used as an enable for the state machine.
CTRIG	0x0008	R/W	This word indicates the start condition for the microcode state machine C. Bit 0 is used as an enable for the state machine.
DTRIG	0x000C	R/W	This word indicates the start condition for the microcode state machine D. Bit 0 is used as an enable for the state machine.
AMASK	0x0010	R/W	This word indicates the mask which is used to determine the length of the acquisition cycle of the microcode state machine A.
BMASK	0x0014	R/W	This word indicates the mask which is used to determine the length of the acquisition cycle of the microcode state machine B.
CMASK	0x0018	R/W	This word indicates the mask which is used to determine the length of the acquisition cycle of the microcode state machine C.
DMASK	0x001C	R/W	This word indicates the mask which is used to determine the length of the acquisition cycle of the microcode state machine D.
AENDM	0x0020	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last acquisition in the processing cycle in microcode state machine A.
BENDM	0x0024	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last acquisition in the processing cycle in microcode state machine B.

Table 2: Acquisition rate configuration in the transmit IO space.

CENDM	0x0028	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last acquisition in the processing cycle in microcode state machine C.
DENDM	0x002C	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last acquisition in the processing cycle in microcode state machine D.
TransmitStatus	0x003C	R	The upper 16 bits represent the status of the transmit microcode state machine. (Writing to this address will send data to the EMAC!) Bit 16: decoder error Bit 17 to 31: not used

Only the most significant 26 bits of the trigger and mask values are used. The most significant bit indicates the half second count, whereas the least significant bit (bit 6 in the 32 bit word) marks the clock count at 2^{26} Hz. The last 6 bits should be set to one except bit 0 of the trigger value which is used as an enable.

The acquisition rate is determined by TRIG and MASK (interpreted as 26 bit values) using the following formula:

$$\text{go} = (\text{cnt} \wedge \text{MASK} = (\text{TRIG} + 1) \wedge \text{MASK}) \tag{2}$$

The cnt value is derived from a 26 bit counter which is synchronized with the 1 PPS signal and is clocked at a rate of 2^{26} Hz. A valid mask must always be filled with ones from the right. Also $(\text{TRIG} + 1)$ has to be multiple of the sampling cycle. i.e., a multiple of 0x80. If the processing cycle is longer than the acquisition cycle, i.e., there are multiple samples acquired in each processing cycle, the ENDM mask has to be set accordingly

$$\text{last} = (\text{cnt} \wedge \text{ENDM} = (\text{TRIG} + 1) \wedge \text{ENDM}) \tag{3}$$

otherwise it should be set equal to MASK. The lowest supported acquisition and processing rate is 1 Hz.

Example 1: The processing cycle has a rate of 16.384 kHz (or a period of 61 µs) and we want to acquire 4 samples in each processing cycle. This means the acquisition cycle has a rate of 65.536 kHz. This yields a MASK value of 0x0003FF and the ENDM value becomes 0x000FFF. The TRIG value can now be set to adjust the last sample in the processing cycle. If we choose to start the acquisition with the sample aligned to the 1 PPS, the TRIG value should be set to 0x000BFF. The acquisition cycle that starts at 45.8 µs is then marked as the last cycle, whereas the one which starts at 0 µs is marked as the first. The ones which start at 15.3 µs and 30.5 µs will have no marking. Don't forget to insert 6 ones at the end of each value when setting the register values.

Example 2: Both the processing and the acquisition cycle are at a rate of 2048 Hz. The MASK and ENDM values are then both set to 0x007FFF. The TRIG value is 0x007FFF and all acquisition cycles are marked as 'last'.

The most significant bit of the TRIG value is used to enable the microcode state machine. Therefore, it should be written after the MASK and ENDM values have been set. Also, in order to stop the acquisition process a zero should be written into the TRIG value first.

3.1.2.2 Programming the State Machine

The 4 transmit states machines associated with queues A through D can be programmed through dual ported RAM. The transmit state machine uses an 32 bit wide decoder and a 32 bit wide data pool. Both of them are mapped into the memory space of the management processor. Table 3 describes the mapping.

Table 3: Mapping of the transmit state machine RAM into the processor memory space.

Memory block	Offset	Access	Description
A microcode	0x000000	R/W	Microcode of state machine A.
A data pool	0x002000	R/W	Data Pool of state machine A.
B microcode	0x004000	R/W	Microcode of state machine B.
B data pool	0x006000	R/W	Data Pool of state machine B.
C microcode	0x008000	R/W	Microcode of state machine C.
C data pool	0x00A000	R/W	Data Pool of state machine C.
D microcode	0x00C000	R/W	Microcode of state machine D.
D data pool	0x00E000	R/W	Data Pool of state machine D.

Each microcode pool contains 128 entries and each data pool contains 128 entries. Since the clock rate of the state machine is 67.108864 MHz and the sampling rate is 524.288 kHz, only the first ~125 entries are useful for storing code. Since state machines B, C and D run after A, the last command in state machine D should not be later than 125 clock cycles.

The microcode command structure is listed in Table 4.

Table 4: Transmitter microcode.

Bit	pipeline delay	Name	Description
11 down to 0	2	Address	12 bits to address the data source. If the MSB is 0, the remaining 11 bits are used to address the ADC. If the MSB is set, the address is used to decode the memory pool or the GPS time stamp. Memory map: 0x0000 to 0x07FF: ADC data 0x0800 to 0x087F: Memory pool 0x0A00: GPS time stamp (seconds) 0x0B00: GPS time stamp (fractional part) 0x0F00: all zeros.
12	—	—	Not used at the moment.

Table 4: Transmitter microcode.

15 down to 13	0	Ignore on condition	Ignores the command depending on the condition of the first and last flag of the acquisition process. Bit 15: the acquisition cycles with the last flag set are ignored Bit 14: the acquisition cycles in between set are ignored Bit 13: the acquisition cycles with the first flag set are ignored
16	0	FIFO write	Write one 32 bit word into the FIFO
17	0	Start transmission	Instructs the arbiter to start transmitting an ethernet frame.
30 down to 18	—	—	Not used at the moment.
31	0	End of program	This indicates the last command in the microcode.

It is important that the first instruction has the FIFO write and the start transmission flags set to zero. Otherwise, the FIFO will be filled and ethernet frames are scheduled for transmission with every clock cycle while state machine waits for the next ‘go’. However, it is appropriate to load the first address. Because of the pipeline delay, it will only be available on the 3rd cycle anyway. Since the state machine can fill the FIFO faster than the arbiter can read it, it is also appropriate to give the start transmission command as one of the first instructions.

The state machine writes 32 bit words at a time to the EMAC FIFO. The first words must be the 32 most significant bits of the destination MAC address. The next word must be the remaining bits of the source address (mapped to the 16 most significant bits) as well as the 16 most significant bits of the source MAC address (mapped to the 16 least significant bits). The third word must be the remaining bits of the source MAC address. The forth word contains the frame type (mapped to the high bits) and the protocol subtype (mapped to the low bits). The fifth word contains the protocol version (mapped to the high bits) and the data length (mapped to the low bits).

Example: The destination MAC address is 00-13-20-04-4E-D1 and the source MAC address is DE-AD-FA-CE-00-01. The protocol subtype for the highest priority queue is then 80-00. Assuming we want to send 4 ADC data words the following words should be written to the A data pool (staring at zero offset): 0x00132004, 0x4ED1DEAD, 0xFACE0001, 0x88B58000, 0x00000018. To generate an ethernet frame these words should be sent to the EMAC FIFO in order followed by the GPS seconds, the GPS fractional seconds and the 4 ADC data words. For a more detailed description of the ethernet frame see section 4. The corresponding microcode is then 0x00000800, 0x00000801, 0x00010802, 0x00030803, 0x00010804, 0x00010A00, 0x00010B00, 0x00010000, 0x00010001, 0x00010002, 0x00010003, 0x00010F00, 0x80010F00. Keep in mind that there is a 2 cycle pipeline delay before the addressed data sample will be available for writing to the EMAC FIFO. Notice that the EMAC gets the start transmission command when the second data word is written to the FIFO.

3.1.2.3 Transmit Microcode Address Space

The address space of the state machine is 12 bits. The memory map is outlined below:

3.1.2.4 Transmit Statistics Counter

Seven 32 bit statistics counters keep track of the number of errors in the transmit engine. These counters are clocked at a rate of 2^{26} Hz. The counters are enabled by a low to high transition of

Table 5: Layout of the transmit microcode address space.

Address	Description
0x000	ADC 1 filtered input
0x001	ADC 2 filtered input
0x002	ADC 3 filtered input
0x003	ADC 4 filtered input
0x004	ADC 1 input bypassing the filter
0x005	ADC 2 input bypassing the filter
0x006	ADC 3 input bypassing the filter
0x007	ADC 4 input bypassing the filter
0x008 and 0x009	ADC 2 minus ADC 1 filtered input
0x00A and 0x00B	ADC 4 minus ADC 3 filtered input
0x00C to 0x00F	ADC 4 minus ADC 3 plus ADC 2 minus ADC 1 filtered input
0x010 to 0x01F	same as 0x000 to 0x00F but for ADCs 5 through 8
0x020 to 0x02F	same as 0x000 to 0x00F but for ADCs 9 through 12
0x030 to 0x03F	same as 0x000 to 0x00F but for ADCs 13 through 16
0x800 to 0x87F	Memory pool
0xA00	GPS time stamp (seconds)
0xB00	GPS time stamp (fractional part)
0xF00	all zeros

the corresponding bits in the transmit status word. The statistics counters are mapped into the transmit IO space and are listed in Table 6.

Table 6: Statistics counters in the transmit IO space.

Configuration Word	Offset	Access	Description
AFIFO_ERR	0x0040	R	Counts the number of errors in the FIFO of the transmit queue A. Any of the corresponding bits in the transmit status going high will increase this counter.
BFIFO_ERR	0x0044	R	Counts the number of errors in the FIFO of the transmit queue B. Any of the corresponding bits in the transmit status going high will increase this counter.
CFIFO_ERR	0x0048	R	Counts the number of errors in the FIFO of the transmit queue C. Any of the corresponding bits in the transmit status going high will increase this counter.
DFIFO_ERR	0x004C	R	Counts the number of errors in the FIFO of the transmit queue D. Any of the corresponding bits in the transmit status going high will increase this counter.

Table 6: Statistics counters in the transmit IO space.

LPFIFO_ERR	0x0050	R	Counts the number of errors in the FIFO of the transmit low priority queue. Any of the corresponding bits in the transmit status going high will increase this counter.
ARBITER_ERR	0x0054	R	Counts the number of errors in the transmit arbiter. The corresponding bit in the transmit status going high will increase this counter.
DECODE_ERR	0x0058	R	Counts the number of errors in the decoder of the microcode state machine. The corresponding bit in the transmit status going high will increase this counter.

3.1.3 Low Priority Transmit Queue

The low priority transmit queue is programmed by the management processor. Configuration and status frames are written to address 0x003C in the transmit IO space. The ethernet frame has to be written 16 bits at a time using the least significant bits of a 32 bit word. The 17th bit is then used to indicate the last data in the frame. It will also trigger the transmission of the frame the next time the low priority queue reaches the transmit valid window and no higher priority queues are ready.

3.2 Receive Engine

The receive engine consists of 5 receiver queues: A through D and LP (low priority). The FIFOs associated with queues A through D are used by the microcode state machine, whereas the LP FIFO is used by the microprocessor and handles lower priority configuration and status requests.

3.2.1 Receive Arbiter

Compared to the transmit arbiter the receive arbiter has to decode part of the ethernet frame to decide if a valid frame was received. It has to decode the target queue and make sure the frame is long enough. It also has to check the GPS time stamp to make sure data is received on time. To facilitate the decision process the arbiter has a 14 word buffer which is used to delay the ethernet frame. This delay buffer is used for the LP queue only since only data words are sent to the high priority queues. Since it is important to make sure that data is received for each processing cycle, the arbiter maintains diagnostics counters which count the number of times frames arrive too late, to early or not at all. *The sender has to guarantee that there are never two ethernet frames sent to the same queue (A through D) with identical time stamp in the same processing cycle.*

Data has to arrive at the receive arbiter two sampling cycles (~4 μ s) prior to the first acquisition (or output) cycle. Otherwise, the receive microcode state machine will not have enough time to copy the data to the converter outputs. This also means that the maximum acquisition rate is a fourth of the sampling clock, or 131072 Hz. To avoid frames arriving out of synchronization the allowed window is exactly one processing cycle long. The GPS time stamp of an arriving frame has to be the time of the first output cycle of the following processing cycle—when the data is actually written to the converter outputs. All frames which arrive outside their time window will be rejected.

The receive arbiter and the receive microcode state machine share the following configuration words.

Table 7: Configuration in the receive IO space.

Configuration Word	Offset	Access	Description
ATRIG	0x0000	R/W	This word indicates the start condition for the receive queue A. Bit 0 is used as an enable for the state machine and the arbiter.
BTRIG	0x0004	R/W	This word indicates the start condition for the receive queue B. Bit 0 is used as an enable for the state machine and the arbiter.
CTRIG	0x0008	R/W	This word indicates the start condition for the receive queue C. Bit 0 is used as an enable for the state machine and the arbiter.
DTRIG	0x000C	R/W	This word indicates the start condition for the receive queue D. Bit 0 is used as an enable for the state machine and the arbiter.
AMASK	0x0010	R/W	This word indicates the mask which is used to determine the length of the output cycle of the receive queue A.
BMASK	0x0014	R/W	This word indicates the mask which is used to determine the length of the output cycle of the receive queue B.
CMASK	0x0018	R/W	This word indicates the mask which is used to determine the length of the output cycle of the receive queue C.
DMASK	0x001C	R/W	This word indicates the mask which is used to determine the length of the output cycle of the receive queue D.
AENDM	0x0020	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last output in the processing cycle in receive queue A.
BENDM	0x0024	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last output in the processing cycle in receive queue B.
CENDM	0x0028	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last output in the processing cycle in receive queue C.
DENDM	0x002C	R/W	This word indicates the mask which is used to determine the length of the processing cycle as well as mark the last output in the processing cycle in receive queue D.
ALEN	0x0030	R/W	This word indicates the length of the data frame for receive queue A.
BLEN	0x0034	R/W	This word indicates the length of the data frame for receive queue B.

Table 7: Configuration in the receive IO space.

CLEN	0x0038	R/W	This word indicates the length of the data frame for receive queue C.
DLEN	0x003C	R/W	This word indicates the length of the data frame for receive queue D.
ReceiveStatus	0x00F8	R	The lower 16 bits represent the status of the receive arbiter, whereas the upper 16 bits represent the status of the receive microcode state machine. Bit 0: Data ready in the LP FIFO queue Bit 1: Out-of-sync frame received on queue A Bit 2: Out-of-sync frame received on queue B Bit 3: Out-of-sync frame received on queue C Bit 4: Out-of-sync frame received on queue D Bit 5: Frame type error Bit 6: Frame length error Bit 15: Decoder error Bit 16: Receiver queue A empty at start of output cycle Bit 17: Receiver queue B empty at start of output cycle Bit 18: Receiver queue C empty at start of output cycle Bit 19: Receiver queue D empty at start of output cycle
frame	0x00FC	R	Read a low priority frame

3.2.2 Receive Microcode State Machine

The receive microcode state machine is somewhat simpler than its transmit counter part, since some of the processing has already be done by the receive arbiter. All that is left to do is to copy the data from the queue into the converter output.

3.2.2.1 Programming the Output Rate

The output rate is determined by TRIG and MASK using the following formula:

$$\text{go} = (\text{cnt} \wedge \text{MASK} = (\text{TRIG} + 1) \wedge \text{MASK}) \tag{4}$$

The convention is the same as for setting the acquisition rate in the transmit microcode machine. A valid mask must always be filled with ones from the right and 6 ones have to be filled in from the right to form a 32 bit word. If the processing cycle is longer than the output cycle, i.e., there are multiple samples acquired in each processing cycle, the ENDM mask has to be set accordingly

$$\text{last} = (\text{cnt} \wedge \text{ENDM} = (\text{TRIG} + 1) \wedge \text{ENDM}) \tag{5}$$

otherwise it should be set equal to MASK. The start of the microcode state machine and the start of the arbiter receive window are then 1.91 μs and 4.29 μs ahead of the first output cycle in each processing cycle, respectively.

Example: The output rate is 16384 Hz and aligned in the center of the 61 μs processing cycle. The MASK and ENDM values would then both be set to 0x0003FFFF, and the TRIG value would be set to 0x0003FFFF.

3.2.2.2 Programming the State Machine

The 4 receive states machines associated with queues A through D can be programmed through dual ported RAM. The transmit state machine uses a 32 bit wide decoder which is mapped into the memory space of the management processor. Table 8 describes the mapping.

Table 8: Mapping of the receive microcode RAM into the processor memory space.

Memory block	Offset	Access	Description
A microcode	0x000000	R/W	Microcode of state machine A.
B microcode	0x004000	R/W	Microcode of state machine B.
C microcode	0x008000	R/W	Microcode of state machine C.
D microcode	0x00C000	R/W	Microcode of state machine D.

Each microcode pool contains 128 entries. Since the clock rate of the state machine is 67.108864 MHz and the intrinsic output rate of the converters is 524.288 kHz, all entries are useful for storing code. Since state machines B, C and D run after A, the last command in state machine D should not be later than 128 clock cycles.

The microcode command structure is listed in Table 9.

Table 9: Receiver microcode.

Bit	pipeline delay	Name	Description
11 down to 0	2	Address	12 bits to address the data source. If the MSB is 0, the remaining 11 bits are used to address the DAC. If the MSB is set, the address is ignored. Memory map: 0x0000 to 0x07FF: DAC output data 0x0800 to 0x0FFF: not used
12	—	—	Not used at the moment.
15 down to 13	0	Ignore on condition	Ignores the command depending on the condition of the first and last flag of the output process. Bit 15: the outputs cycles with the last flag set are ignored Bit 14: the outputs cycles in between set are ignored Bit 13: the outputs cycles with the first flag set are ignored
16	0	DAC write	Reads one 32 bit word from the FIFO and writes it to the DAC output.
30 down to 17	—	—	Not used at the moment.
31	0	End of program	This indicates the last command in the microcode.

There is pipeline delay for transferring data form the FIFO to a DAC output. The FIFO only contains data and the ethernet frame has been stripped.

3.2.2.3 Receive Microcode Address Space

The address space of the state machine is 12 bits. The memory map is outlined below:

Table 10: Layout of the receive microcode address space.

Address	Description
0x000	DAC 1 output with filter
0x001	DAC 2 output with filter
0x002	DAC 3 output with filter
0x003	DAC 4 output with filter
0x004	DAC 1 output bypassing the filter
0x005	DAC 2 output bypassing the filter
0x006	DAC 3 output bypassing the filter
0x007	DAC 4 output bypassing the filter
0x008 or 0x009	DAC 2 output with filter and negative DAC 1 output with filter
0x00A or 0x00B	DAC 4 output with filter and negative DAC 3 output with filter
0x00C to 0x00F	DAC 4 and DAC 2 output with filter and negative DAC 3 and DAC 1 output with filter
0x010 to 0x01F	same as 0x000 to 0x00F but for DACs 5 through 8
0x020 to 0x02F	same as 0x000 to 0x00F but for DACs 9 through 12
0x030 to 0x03F	same as 0x000 to 0x00F but for DACs 13 through 16

3.2.2.4 Receive Statistics Counter

A number of 32 bit statistics counters keep track of the number of errors in the receive engine. These counters are clocked at a rate of 2^{26} Hz. The counters are enabled by a low to high transition of the corresponding bits in the receive status word. The statistics counters are mapped into the receive IO space and are listed in Table 11.

Table 11: Statistics counters in the receive IO space.

Configuration Word	Offset	Access	Description
A_DISCARD	0x0040	R	Counts the number of discarded frames in receiver queue A which were received outside the time window. The corresponding bit in the receive status going high will increase this counter.
B_DISCARD	0x0044	R	Counts the number of discarded frames in receiver queue B which were received outside the time window. The corresponding bit in the receive status going high will increase this counter.
C_DISCARD	0x0048	R	Counts the number of discarded frames in receiver queue C which were received outside the time window. The corresponding bit in the receive status going high will increase this counter.
D_DISCARD	0x004C	R	Counts the number of discarded frames in receiver queue D which were received outside the time window. The corresponding bit in the receive status going high will increase this counter.

Table 11: Statistics counters in the receive IO space.

A_MISSING	0x0050	R	Counts the processing cycles on receiver queue A which have not received a data frame in time. The corresponding bit in the receive status going high will increase this counter.
B_MISSING	0x0054	R	Counts the processing cycles on receiver queue B which have not received a data frame in time. The corresponding bit in the receive status going high will increase this counter.
C_MISSING	0x0058	R	Counts the processing cycles on receiver queue C which have not received a data frame in time. The corresponding bit in the receive status going high will increase this counter.
D_MISSING	0x005C	R	Counts the processing cycles on receiver queue D which have not received a data frame in time. The corresponding bit in the receive status going high will increase this counter.
FTYPE_ERR	0x0060	R	Counts frames with incorrect frame type or data frames with incorrect protocol version. The corresponding bit in the receive status going high will increase this counter.
FLEN_ERR	0x0064	R	Counts data frames with incorrect length. The corresponding bit in the receive status going high will increase this counter.
DECODE_ERR	0x0068	R	Counts the number of errors in the decoder of the microcode state machine. The corresponding bit in the receive status going high will increase this counter.

3.2.3 Low Priority Receive Queue

The low priority receive queue is handled by the management processor. Configuration and status frames are read from address 0x00FC in the receive IO space. The full ethernet frame is available and has to be read 16 bits at a time using the least significant bits of the 32 bit word. The 17th bit is then used to indicate the last data in the frame. The receive status register at 0x00F8 indicates with bit 0, if a frame is available in the FIFO.

3.3 Management Processor

A management processor is responsible to configure the transmit and receive engines as well as the ethernet MAC. It also handles the low priority queues and processes non-data ethernet frames.

3.3.1 IO Space

Table 12: IO space layout in the management processor.

IO block	Offset	Description
IOCFG0	0x000000	Basic configuration
IOCFG1	0x080000	Extended configuration
IOCLK	0x100000	Interface to timing system
IOEMAC0	0x180000	Access to host interface of EMAC 0 and EMAC 1
IOEMAC2	0x1C0000	Access to host interface of EMAC 2 and EMAC 3
IOTX0	0x200000	Access to configuration of transmit engine 0
IORX0	0x280000	Access to configuration of receive engine 0
IOTX1	0x300000	Access to configuration of transmit engine 1
IORX1	0x380000	Access to configuration of receive engine 1
IOTX2	0x400000	Access to configuration of transmit engine 2
IORX2	0x480000	Access to configuration of receive engine 2
IOTX3	0x500000	Access to configuration of transmit engine 3
IORX3	0x580000	Access to configuration of receive engine 3
IOADC	0x800000	Access to configuration of input section
IODAC	0x900000	Access to configuration of output section

The layout of the IO space is shown in Table 12. The configuration for the transmit and receive configuration regions are described in the uplink section. The configuration IO block is described in Table 13.

Table 13: Layout of the configuration IO space.

Configuration Word	Offset	Access	Description
CHANNEL_DIR	0x0000	R/W	Determines the direction of IO channels S1 to S28. S1 is mapped to bit 0. Writing a one into the corresponding bit location will make this an input channel, whereas a zero indicates an output channel.
			Bit 31 is used as a soft reset.

Table 13: Layout of the configuration IO space.

AUX1	0x0004	R/W	The LEDs are mapped to bits 0 through 3.
			The chip selects for the SPI interface are using bit 8 through 11; these 4 bits are decoded into 16 select lines.
AUX2	0x0008	R	The dip switches are mapped into bits 0 through 7.
BRD	0x080000 0x080004	R/W	XML decoded parameter values for drawing number of board
REV	0x080008 0x08000C	R/W	XML decoded parameter value for revision number
SRV	0x080010 0x080014	R/W	XML decoded parameter value for sub revision number
BSN	0x080018 0x08001C	R/W	XML decoded parameter value for board serial number
EA0	0x080020 0x080024	R/W	XML decoded parameter value for ethernet address of EMAC 0
EA1	0x080028 0x08002C	R/W	XML decoded parameter value for ethernet address of EMAC 1
EA2	0x080030 0x080034	R/W	XML decoded parameter value for ethernet address of EMAC 2
EA3	0x080038 0x08003C	R/W	XML decoded parameter value for ethernet address of EMAC 3
TAG8-F	0x080040 to 0x08007C	R/W	XML decoded parameter value for user tags
TIMING_STATUS	0x100000	R	Bit 0: Signal detect of the timing signal fiber transceiver.
			Bits 1 through 4: 1PSS error code.
GPSsec	0x100008	R	GPS time (seconds).
GPSfrac	0x00000C	R	GPS time (fractional part).

The ADC IO space is used to select the number of significant bits in the analog-to-digital converter. Its layout is shown in Table 14.

The DAC IO space is used to select the number of significant bits in the digital-to-analog converter. Its layout is shown in Table 15.

The ethernet MAC has its own configuration and statistics interface and is mapped into the IO space. The layout of the EMAC configuration follows Xilinx user guide UG074 and Xilinx user guide UG170. An abbreviated layout is shown in Table 16. The IO address space is divided into two 1024 byte blocks to access EMAC 0 and EMAC 1, respectively. Within each EMAC block the first 512 bytes are reserved for the statistics counters. The second 512 bytes are used for accessing the EMAC configuration. Statistics counters are 64 bytes wide and have to be read most significant word first. The least significant word can be accessed at an address with an additional offset of 0x100. It must be read immediately after the most significant word.

Table 14: Layout of the ADC IO space.

Configuration Word	Offset	Access	Description
ADC_CONF	0x0000	R/W	Bits 0 through 2 select which ADC channel is sent to the AES3 digital audio link. The selected channel is available on the left channel of the first link port. The next ADC channel is available on the right channel. The next two ADC channels will be available on the second link port.
			Bits 4 through 6 are used to select the decimation filter.
			Bits 8 through 11 are the mode bits which select the bit width the of ADC converter. The number 0 is used for a 32 bit device. For all others the ADC width is twice the selected number.
			Bit 12 is used to indicated that an additional register is used in the readback path of the ADC. Use 1 for no register.
			Bit 15 is used to activate the internal loopback path.

Table 15: Layout of the DAC IO space.

Configuration Word	Offset	Access	Description
DAC_CONF	0x0000	R/W	Bits 0 through 2 select which ADC channel is read from the AES3 digital audio link. The selected channel is taken from the left channel of the first link port. The next DAC channel comes from the right channel. The next two DAC channels will be taken from the second link port.
			Bit 3 has to be set to 1 to enable the AES3 digital link input.
			Bits 4 through 6 are used to select the decimation filter.
			Bits 8 through 11 are the mode bits which select the bit width the of ADC converter. The number 0 is used for a 32 bit device. For all others the ADC width is twice the selected number.

Table 16: Layout of the EMAC IO space.

Configuration Word	Offset	Access	Description
TRANSBYTES	0x0000	R	Amount of transmitted data in bytes (most significant word).
RECVBYTES	0x0004	R	Amount of received data in bytes (most significant word).
UNDERSIZE	0x0008	R	Number of undersized frames received.
FRAGMENTS	0x000C	R	Number of fragmented frames received.
RECVFRAMES1	0x0010	R	Number of 64 bytes frames received.
RECVFRAMES2	0x0014	R	Number of 65–127 bytes frames received.

Table 16: Layout of the EMAC IO space.

RECVFRAMES3	0x0018	R	Number of 128–255 bytes frames received.
RECVFRAMES4	0x001C	R	Number of 256–511 bytes frames received.
RECVFRAMES5	0x0020	R	Number of 512–1023 bytes frames received.
RECVFRAMES6	0x0024	R	Number of 1024–1518 bytes frames received.
RECVFRAMES7	0x0028	R	Number of jumbo frames received.
TRANSFRAMES1	0x002C	R	Number of 64 bytes frames transmitted.
TRANSFRAMES2	0x0030	R	Number of 65–127 bytes frames transmitted.
TRANSFRAMES3	0x0034	R	Number of 128–255 bytes frames transmitted.
TRANSFRAMES4	0x0038	R	Number of 256–511 bytes frames transmitted.
TRANSFRAMES5	0x003C	R	Number of 512–1023 bytes frames transmitted.
TRANSFRAMES6	0x0040	R	Number of 1024–1518 bytes frames transmitted.
TRANSFRAMES7	0x0044	R	Number of jumbo frames transmitted.
RECVOK	0x0048	R	Number of error-free frames received.
RECVCRCERR	0x004C	R	A count of received frames that failed the CRC check and were at least 64 bytes in length.
RECVBROADCAST	0x0050	R	A count of frames successfully received and directed to the broadcast group address.
RECVMULTICAST	0x0054	R	A count of frames successfully received and directed to a non broadcast group address.
	0x0058 to 0x0068	R	See Xilinx user guide.
TRANSOK	0x006C	R	Number of error-free frames transmitted.
TRANSBROADCAST	0x0070	R	A count of error-free frames that were transmitted to the broadcast address.
TRANSMULTICAST	0x0074	R	A count of error-free frames that were transmitted to a group destination address other than broadcast.
	0x0078 to 0x0084	R	See Xilinx user guide.
	0x0100 to 0x0184	R	Corresponding least significant word.
RECVCONF0	0x0200	R/W	Receiver Configuration (Word 0) Pause frame address

Table 16: Layout of the EMAC IO space.

RECVCONF1	0x0240	R/W	Receiver Configuration (Word 1) Bits 0 through 15: Pause frame address Bit 25: Length/type disable Bit 26: Half-duplex mode Bit 27: VLAN enable Bit 28: Receive enable Bit 29: In-band FCS enable Bit 30: Jumbo frame enable Bit 31: Reset
TRANSCONF	0x0280	R/W	Transmitter Configuration Bit 25: IFG adjustment enable Bit 26: Half-duplex mode Bit 27: VLAN enable Bit 28:Transmit enable Bit 29: In-band FCS enable Bit 30: Jumbo frame enable Bit 31: Reset
FLOWCNTR	0x02C0	R/W	Flow Control Configuration Bit 29: Receive flow control enable Bit 30: Transmit flow control enable
MODECONF	0x0300	R	Ethernet MAC Mode Configuration
	0x0320	R	RGMII/SGMII Configuration
MGTCONF	0x0340	R/W	Management Configuration
UNIADDR0	0x0380	R/W	Unicast Address (Word 0) least significant word (bytes 0 through 3)
UNIADDR1	0x0384	R/W	Unicast Address (Word 1) Bits 0 through 15: bytes 4 and 5
MULTIADDR0	0x0388	—	Multicast Address Table Access (Word 0) not supported at the moment
MULTIADDR1	0x038C	—	Multicast Address Table Access (Word 1) not supported at the moment
ADDRFILTER	0x0390	R/W	Address Filter Mode Bit 31: Promiscuous mode enable
EMAC1	0x0400 to 0x07FF		second EMAC; same as EMAC 0

3.3.2 Memory Space

A layout of its memory space is shown in Table 17.

Table 17: Memory layout in the management processor.

Memory block	Offset	Description
BOOT	0x00000000	Configuration EEPROM(s)
RAM	0x10000000	Internal fast block RAM

Table 17: Memory layout in the management processor.

SRAM	0x20000000	External SDRAM
TX0	0x80000000	Microcode for transmit engine 0
RX0	0x88000000	Microcode for receive engine 0
TX1	0x90000000	Microcode for transmit engine 1
RX1	0x98000000	Microcode for receive engine 1
ADC	0xA0000000	ADC coefficient memory
DAC	0xA8000000	DAC coefficient memory

4 Ethernet Protocol

The content of an ethernet frame is shown in Table 18.

Table 18: Ethernet frame.

Field	**Size**	**Offset**	**Description**
Preamble	—	—	Ethernet preamble, automatically generated.
Destination address	6	0	6 bytes, MAC address of computer board (presumably commercial).
Source address	6	6	6 bytes, MAC address of converter The next to LSB bit of the first octet is set to 1 to indicate a locally administrated address. LSB is 0 for non-broadcast. All other bits are user specific. For example, we could use DE-AD-FA-CE-##-## where ##-## indicates the serial number of the converter.
Frame type	2	12	Local experimental ethertype 1, 88-B5.
Protocol subtype	2	14	Defines the request or the transfer type.
Protocol version	2	16	00 for the initial version (also makes sure the data is aligned to a double word boundary).
Data length in bytes	2	18	Must a multiple of 4, can be between 0 to 1492 for normal frames or between 0 and 8992 for jumbo frames.
Data payload	N	20	Payload is depending on protocol subtype and transfer requirement. 32 bit double words are sent with the most significant byte first. Since the minimum ethernet payload length is 46 bytes (starting with the protocol subtype), up to 40 bytes of padding have to be added here for shorter frames with N < 40.
CRC	4	20+N	Frame check sequence, automatically generated.

4.1 Protocol Subtypes

The protocol subtype field is used to define the transmission request. The subtype consists of 16 bits. The highest 4 bits are being used to designate a converter data frame, whereas the lower 12 bits are used for configuration requests and status information. To be a valid converter data frame exactly one of the 4 high bits has to be non-zero. The remaining bits are then ignored. This allows for 4 different types of data frames. Currently, only one configuration type is supported by setting bit 7 to one. If bit 6 is set to 1 as well, a response frame will be generated. To be a valid configuration request all 4 high bits have to be zero.

4.1.1 Configuration Frame

A configuration frame contains a set of instructions. Each instruction consists of a command identifier, an address and an optional data word. All configuration registers are memory mapped. So, the command typically identifies either a write or a read operation. A write operation will have a data word associated with it, whereas a read operation does not. Both operations can be combined. The write operation will be done first and then the read back value will be sent back. Setting up a front-end system is done by writing configuration words to its registers. To enumerate all front-end systems connected to an ethernet segment, a configuration read can be sent to the broadcast address. To allow automatic identification of a front-end system its EEPROM which contains drawing number and serial number should be mapped to address 0.

Table 19: Configuration frame.

Field	Size	Offset	Value
Protocol subtype	2	14	128 or 192
Protocol version	2	16	0
Data length in bytes	2	18	N = 4 M, with M the number of instructions and data words
Data payload	8M	20	Instructions and/or data words

An instruction consists of a 32 bit word which contains a 24 bit address and an 8 bit command word. All data requests are 32 bit wide and have to be aligned. Hence, the lower 2 bits of the address are always zero. However, they are used to specify a burst length. If the command indicates a write operation, a number of 32 bit data words as specified in the burst length have to follow the instruction. Special care must be taken to make sure that instructions and associated data words do not cross the ethernet frame length boundary. The IO space uses a 24 bit wide address space and the address is taken as specified. The memory space uses a 32 bit wide address space. The 32 bit address is assembled from the 24 bit address provided in the instruction word by keeping the top 6 bits as the top 6 bits, by keeping the bottom 18 bits as the bottom 18 bits and by filling in zeros in between. If both a read and a write are specified, the write is executed first.

When the protocol subtype has bit 6 set, a response ethernet frame will be generated. Typically, it will contain the results of the read instructions. Again, special care should be taken to avoid the response frame from becoming too long.

Table 20: Instruction.

Field	Size	Offset	Value
Bit 0 - Bit 1	2	0	Burst length: 0 – 1 word 1 – 4 words 2 – 16 words 3 – 64 words
Bit 2 to Bit 23	22	2	24 bit address space aligned to 32 bits words
Bit 24	1	24	0 – Memory space 1 – IO space
Bit 25	1	25	1 – Write operation
Bit 26	1	26	1 – Read operation
Bit 27 – Bit 31	5	27	Must be set to zero (future expansion)

Instructions can also be read from memory during initialization. In this case an instruction of all ones indicates the end of the initialization routine.

4.1.2 Converter Data Frame

The converter data frame contains a time stamp and converter data.

Table 21: Data frame.

Field	Size	Offset	Value
Protocol subtype	2	14	32768, 16384, 8192 or 4096
Protocol version	2	16	0
Data length in bytes	2	18	N = 4M + 8, with M the number converter data words
Data payload	4	20	GPS second (most significant word first)
	4	24	GPS sub second in units of 2^{-32} s
	4M	28	Converter data words

5 Tools

5.1 Configuration Setup

In order to simplify the generation of configuration instructions a little assembler language was created which can be translated into binary values using the confasm program.

5.1.1 Mnemonic

The basic instruction is a simple store and/or load. The syntax is

‘Instruction’ ‘memory address’ {‘data field(s)’}

The instruction is one of the following:

ST: store data into the IO space
LD: load data from the IO space
STLD: store data into the IO space, then read it back
WR: write data into the memory space
RD: read data from the memory space
WRRD: write data to the memory space, then read it back

The memory address is a 6 digit hex number which has to be marked by a leading ‘0x’. Both IO and memory space are organized strictly by 32 bit data words. This means the last 2 bits of any address has to be zero. A store or write instruction must provide at least one 32 bit data word. If multiple data words are provided, they are written to successive addresses.

Data fields are separated by comma or space. Standard notation is decimal. Prepending ‘0x’ will make it a hex number. Using quotes denotes an ASCII string which will be loaded in little endian format.

5.1.2 confasm

This program will take the assembly instructions and generate a binary format. The usage is as follows:

```
confasm [-i ‘start address’] [-x] ‘file name’
```

The file extension of the input file has to be ‘.cfgasm’, whereas the output file will have the extension ‘.cfgdat’. The output file is organized in little endian format.

The optional argument ‘start address’ indicates that this is an initialization file which will be written to the specified memory address. The confasm program will generate the additional instructions which are needed to write the original assembly program into the destination address. This is useful to reprogram the initialization area in the EEPROM through the ethernet. If the start address is given as ‘eeprom’, the default address of 0x001000 is assumed. If the argument ‘-x’ is specified, the output file will be a Xilinx ‘.coe’ file which can be used to initialize a FPGA ROM block.

5.1.3 confprom

The confprom program is used to generate the data for an EEPROM. Its use is

```
confprom   [-x ‘xml file’]   [-b ‘binary file’]   [-i ‘initialization file’]   ‘output file’
```

At least one of the arguments has to be specified. The ‘xml file’ should be an XML file with less than 2 kB length. It will be loaded at address location 0x000000. The ‘binary file’ should point to a binary file which will be loaded at address location 0x000800. It should also be smaller than 2 kB. The initialization file will be loaded at address location 0x001000. It has to be smaller than 28 kB. The output format is an Intel HEX file.