A CANbus Replacement for the BIMA Antenna Telemetry

A. D. Bolatto

1. Description of Current Telemetry System

As of February 2003, the telemetry data flows from the array control computer to the telemetry card in the antenna VME rack and back over coaxial or fiber optic links. The telemetry stream carries commands from the control computer to the antenna, such as pointing and walsh sequence, and information back from the antenna onto the control computer, such as encoder and temperature readouts. The current link operates at 1 MHz, using a balanced serial line and obsolete "Manchester code" electronics. This document describes the way the current VME telemetry control card (the telemetry card) uses the VME backplane to communicate with the other VME cards (the slave cards), and discusses an implementation of a replacement card based on the XAC3 microcontroller.

A communication cycle between the telemetry card and the other cards on the VME bus has four phases: 1) the time phase, where the CK bus is updated in the VME backplane. 2) The write phase, where a burst from the telemetry card is put in the VME backplane, containing 256 bytes of consecutive addresses and 256 data bytes. 3) The read phase, where the telemetry card addresses 512 consecutive registers, reading in their contents from the VME data bus. And, 4) a rest period. This cycle is repeated 100 times per second, on a basic clock tick of 10 ms. During the read and write phases, individual register addresses and their corresponding data are present in their respective buses (lines A[0..8] and D[0..7]) for 10 μs, clocked–in with the STROBE* line at a frequency of 100 KHz (data is valid for STROBE* low). The WRITE* line is held up during the write phase, then held down to indicate the read phase. Thus, phases 2 and 3 of the telemetry cycle take $(256 + 512) \times 10 = 7680$ μs to complete, using up 768 ms out of each second.

A key feature of the current VME–based hardware is that all addresses have to be presented in the bus in sequence. Some cards (e.g., the A/D card) do not listen to their own address, but *to a previous address* a few ticks before theirs. This address triggers the A/D conversion, which is then completed and ready to go on the bus by the time the register itself is addressed. Thus, random access for individual registers is not possible.

The following backplane lines are currently connected to the telemetry card:

- The D[0..7] data bus
- The A[0..8] address bus

- The CK[0..7] clock bus
- WRITE* (control line)
- STROBE* (control line)
- BIT CK (free running 1 MHz clock)
- PHR* (obsolete?)
- TMTYOK (debugging purposes, obsolete)
- 81S* (obsolete)
- 320P (walsh cycle start pulse)
- STRT* (begining of 10 ms cycle pulse)
- 10MHz (external master clock)

Inspection of the schematics indicates which of these lines are used by the other cards attached to the VME backplane. They have to be provided by the microprocessor telemetry replacement:

- Receiver/motor control card (D[],A[],WRITE*, STROBE*)
- Analog card (D[],A[],WRITE*, STROBE*)
- Encoder interface card (D[],A[],WRITE*, STROBE*)
- Resolver interface card (D[],A[],WRITE*, STROBE*)
- Offset generator card (D[],A[],WRITE*, STROBE*,320P,STRT,10MHz)
- Digital velocity filter card (D[],A[],WRITE*, STROBE*, BIT CK)
- Total power integrator card (D[],A[],WRITE*, 320P, 10MHz)
- Subreflector calibration control card (D[],A[],WRITE*, STROBE*,320P)

CARMA will send the walsh phase already encoded in the reference LO, thus making the 320P signal obsolete. During the transition to CARMA, however, it will be necessary to support the current offset generator card which requires the 320P and STRT signals.

2. Proposed Replacement

Figure 1 sketches the proposed telemetry card replacement solution, based on the Phytec Phillips XAC3 microprocessor module and a CANbus interface to the antenna computer, which is in communication with the lab via an ethernet link. The philosophy of this solution is to aim to reproduce the precise waveforms that the slave cards expect to see on the VME backplane. Thus, the slave cards will continue to perform their functions, blissfully unaware of the fact that the old telemetry card has been replaced.

The new telemetry works as follows: the STROBE* signal is generated using one of the XAC3 internal timers, tied down to the 32 MHz internal clock. To generate an address, this signal could be used to clock a standard binary counter that cycles through the 9-bit addresses on the VME address bus, taking advantage of the fact that they have to be consecutive. For added flexibility, however, we have chosen to generate addresses on the VME bus by writing to transceiver TR3. The 32 MHz clock itself, divided by 2^5, provides the 1 MHz BIT CK backplane signal for the digital velocity filter card.

The data bus has to be "written" or "read" on a 10 μs beat, and the timing of these actions is critical. To guarantee the timing, we propose to implement a double–buffering scheme using bi–directional registered transceivers. During the write phase the CPU presents to the A side of transceiver TR1 the byte value to be written *to the next* (not the current) register address during the following STROBE* cycle. Transceiver TR1 captures and holds this bit pattern on its B side, making it available to the A side of transceiver TR2. At the active flank of the STROBE* signal, this bit pattern is clocked from the A to the B side of transceiver TR2 and presented on the VME data bus. This frees the CPU from stringent timing requirements: data can be written to TR1 at any time during a 10 μs window before it has to be on the VME bus.

Why two transceivers and not one? The first transceiver is essentially acting as a memory for the second one. The CPU writes new data to the first one by clocking it, while the second transceiver is busy presenting the older data to the VME bus. A similar scheme could be implemented using a memory and a transceiver. However, since one transceiver is needed to boost the signal on the VME backplane anyway, we thought more elegant to use two identical chips.

A similar procedure to the one described above is followed during the read phase. The bit patterns present on the VME data bus are sampled and clocked from the B to the A side of TR2 at the active flank of STROBE*. However, the A side of TR1 is in a high impedance state until the CPU initiates a read from the tranceiver memory address. Then the bit pattern is clocked from the B to the A side of TR1 and its A outputs are enabled, therefore

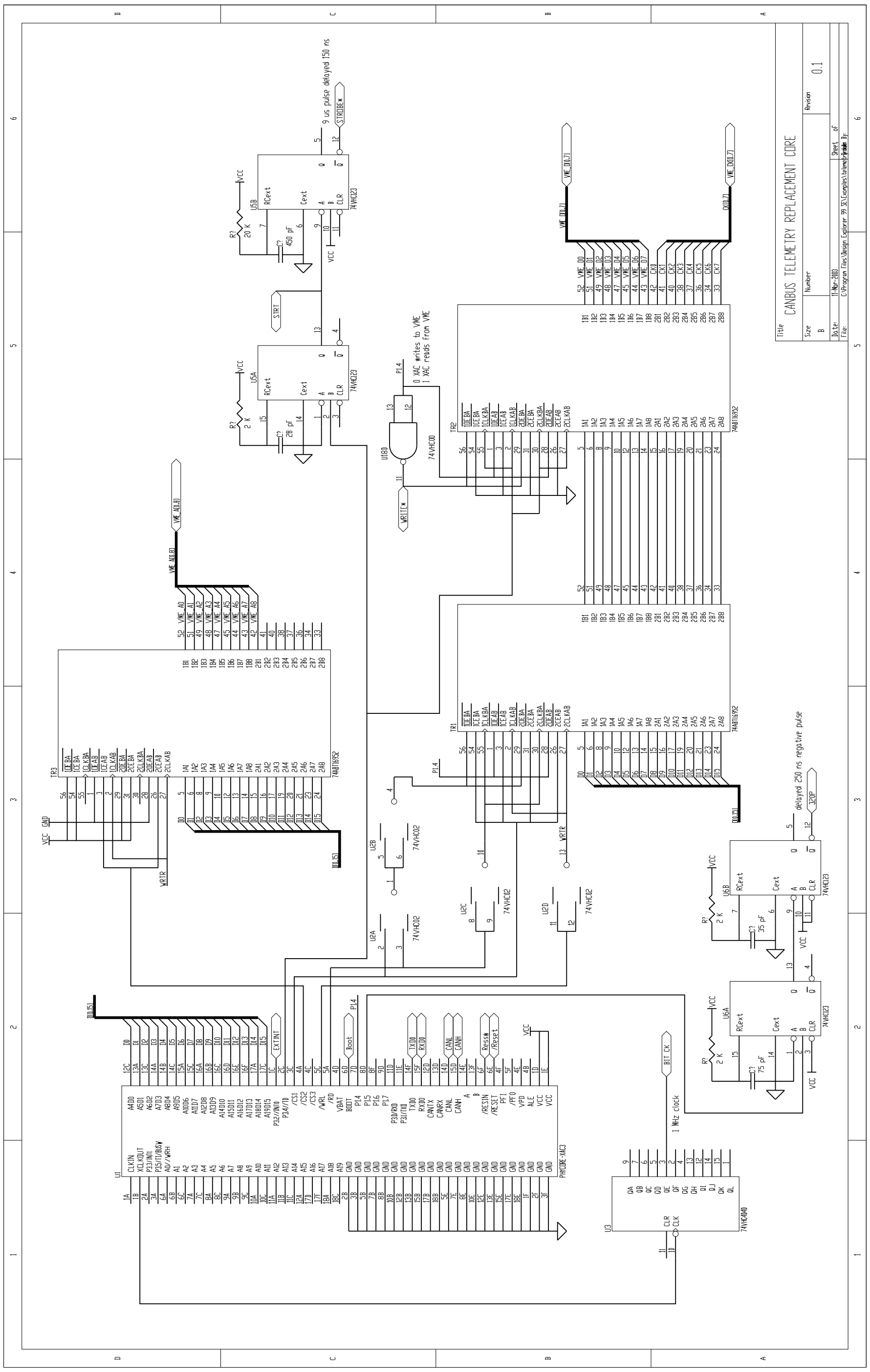

Fig. 1.— The proposed replacement for the current BIMA telemetry based on a 32 MHz Phytec XAC3 μP core and registered transceivers for buffering.

presenting the bit pattern in the lower 8 bits of the CPU data bus. This read operation can take place any time in the 10 μs frame during which data is present in the VME.

The current telemetry is synchronized with the lab using the time information encoded in the first 4 bytes transmited (i.e., registers at addresses 0 to 3). That, plus the 320P signal seem to provide all the synchronization required. In the new telemetry scheme, a missing pulse encoded in the reference 10 MHz tone will provide a 1 PPS clock distributed throughout the antenna, synchronized to the lab with an accuracy $\sim 10^{-6}$. The question arises: which pulse will signal "$t = 0$"? It is up to the software to ensure synchronicity by providing identification of the "$t = 0$" pulse. A possible scheme could be to: 1) synchronize free–running clocks in the cPCI and XAC3, then 2) establish that the first pulse arriving after a given clock count will the "$t = 0$". Because the clock is a 1 PPS signal, latencies in the lab–cPCI ethernet link and the Linux kernel are unlikely to be a problem.

3. Timing & Programming

In the scheme described above, the XAC3 CPU has approximately 10 μs to update the VME bus and must do so regularly. For the 32 MHz CPU clock this translates into 320 clock cycles. With an average CPU instruction taking 4 clock cycles to execute (PXAC3x User Guide, Appendix C), this means we are allowed to execute only $\sim$ 80 (and to be safe more like $\sim$ 50) CPU machine instructions between VME bus updates. These strong time constraints suggest that no external interrupts should be allowed during a telemetry burst, and more importantly, that any CANbus exchange taking place at this time should be watched carefully. Perhaps CANbus communication should be allowed only during the rest period of the telemetry. This probably is not a crippling limitation, since even then more than 200 ms would be available each second for communication.

Fortunately, the tasks that need to be carried out by the CPU during 10 μs available in the telemetry cycle are very simple:

- Watch for timer overflow to initiate read or write.
- Increase a counter.
- Write a VME address to TR3.
- Write a byte from a table to TR1 during the write phase, or read a byte from TR1 into a table during the read phase.

- During the read phase only, a comparison could be carried out between the "last" readout from that memory location and the current readout. If the value has changed, the item should be flagged for transmision during the next CANbus communication period.

Because of their simplicity, these tasks could conceivably be packed in an interrupt routine that will attend interrupts caused by one of the XAC3 internal timers (e.g., T0), as long as it can be guaranteed that attention will be granted during the 10 μs window. This will require testing.

We should stress that the scheme outline above is just one of the simplest possible solutions. If attending a 10 μs interrupt stresses the system, or if more time is needed for CANbus communication for example, a possible solution is to replace transceiver TR1 with a 512 byte memory unit clocked by hardware. This buffer would have to be refreshed by the CPU only at a 100 Hz rate, considerably freeing up resources. It would be read from and then written to during the rest phase of the telemetry cycle.

4. Efficient Use of the CANbus Link

At the useable rates of 500 Kbps reported for the OVRO CANbus implementation, 200 ms available for communication imply that $\sim$ 12 KB of useful information can be exchanged every second (the link is 2–wire, half duplex, so this is the total available bandwidth for exchanges in both directions). The entire address space of the telemetry is 0.5 KB, and only a few of these registers change faster than once a second. The current CARMA antenna API calls for most channels to be updated on a 0.5 s (one frame) tick, with two high speed (100 Hz) monitoring channels reserved for engineering purposes. The data rate flowing from the XAC3 CPU to the antenna computer in such a system would be $\sim$ 1.2 KB/s. However, *identification of which registers need fast updating should be carried out* to accurately compute the data rates. Each register needing 100 Hz updating adds $\sim$ 0.1 KB/s to the bandwidth.

To further minimize the use of the CANbus link, we propose to transmit to (and from) the telemetry CPU only those registers whose value has changed (note that this scheme requires sending the 9 bit address of the register together with the 8 bit data word, thus it only reduces bandwidth if few registers need updating). Furthermore, many registers (e.g., ambient temperature monitor) should be marked to be transmited only once a frame (or slower), even if their values change faster. The telemetry API definition should implement this approach to ensure that the link (and the XAC3 CPU) are not overloaded.

Because of the limited bandwidth, we anticipate some crowding effects will occur during the first second after the telemetry is turned on, when the values of many registers need to be transmited back to the antenna computer. The code running in the XAC3 should be designed to cope with this problem, probably by implementing a queue. The reverse effect (the full write space of 256 bytes needs to be send to the telemetry CPU when the telemetry starts) can be avoided easily by transmiting the block of initial values to the telemetry CANbus node before effectively starting the telemetry. Some care should be taken to avoid desynchronization problems due to the associated delays.