ADDENDUM TO SECTION 4.1.4 PACKET "D"

This addendum supercedes and augments the same section in the manual.

NOTE: Different DACs are used on the bc635VME (11603 assemblies) and the TTM635VME that this manual addresses. A linear DAC is used on the bc635/637VME 11603 assemblies, and a non linear DAC is used on the TTM635/637VME assemblies. Due to this hardware difference, there will be a variation in programming these two parts (i.e., the non-linear DAC used on the TTM635VME DAC uses 2's compliment).

4.1.4 PACKET "D" LOAD D/A CONVERTER

The TFP reference crystal oscillator is voltage controlled using the buffered output of a 16-bit D/A converter as the controlling voltage. Packet "D" allows the user to directly load a 16-bit value to the D/A converter. This feature would allow a user to fine tune the TFP time base in the external oscillator mode. Upon receipt by the timing engine, the input value is converted to two's complement format by subtracting 32768 from the value programmed. Users who are disciplining an external source using a read-modify-write paradigm based on this packet in conjunction with the read packet "O1" MUST take this into account. The value read back by the query command can be converted back to an unsigned value by adding 32768 modulo 65536. We are not aware of any other use for this packet in normal operation. Since this voltage is routed out of the TFP via pin 9 on the J1 connector to allow external oscillators to be disciplined, it would provide a means to devise a frequency control algorithm independent of the TFP. The format is shown below. (See also bit 3 of the path byte loaded by the "P" packet.)

byte	1	SOH	
byte	2	"D"	
byte	3	"0" - "F"	bits 12-15
byte	4	"0" - "F"	bits 08-11
byte	5	"0" - "F"	bits 04-07
byte	6	"0" - "F"	bits 01-03
byte	7	ETB	

NOTE: All data fields must be in ASCII format.

While the board is disciplining the oscillator and the TFP is locked to the input time source, the current value of the D/A converter is written to NVM 4 times a day. When the board is powered up or reset, the last D/A converter value is used as the startup value until disciplining begins.

Examples (see Packet 'O', response format '1', section 4.1.13)

Setting the DAC to its midpoint:

0x8000 is half of 0xFFFF. The input byte stream is "D","8","0","0","0".

Reading back the value:
The output is "0","0","0","0". Add 0x8000 to get 0x8000.
If value is greater than 0x10000, subtract 0x10000 => 0x8000 < 0x10000 => 0x8000.

Setting the DAC to 25% output:

0x4000 is 25% of 0xFFFF. The input byte stream is "D","4","0","0","0".

Reading back the value:
The output is "C","0","0","0". Add 0x8000 to get 0x14000,
If value is greater than 0x10000, subtract 0x10000 => 0x14000 > 0x10000 => 0x4000.

Setting the DAC to 75% output:

0xC000 is 75% of 0xFFFF. The input byte stream is "D","C","0","0","0".

Reading back the value:
The output is "4","0","0","0". Add 0x8000 to get 0xC000.
If value is greater than 0x10000, subtract 0x10000 => 0xC000 < 0x10000 => 0xC000.

Setting the DAC to current value + 1% of tuning range:

Read back the current value:
The output is "F","F","E","0". Add 0x8000 to get 0x17FE0.
If value is greater than 0x10000, subtract 0x10000 => 0x17FE0 > 0x10000 => 0x7FE0.
Add 1% of 0xFFFF which is 0x28F. 0x28F + 0x7FE0 = 0x826F.
The input byte stream is "D","8","2","6","F".

Reading back the value:
The output is "0","2","6","F". Add 0x8000 to get 0x826F.
If value is greater than 0x10000, subtract 0x10000 => 0x826F < 0x10000 => 0x826F.