1.0 Introduction

This document details the commands used by the Ethernet Power Supply Controller (340-260). The command set is based on the previous power supply controller that used the Bitbus for communication. The new controller uses the same commands carried in UDP packets over 10/100 Mb/sec UTP Ethernet instead of Bitbus.

2.0 Network

The controller communicates over Ethernet using TCP/IP. The commands are carried in UDP packets. The controller is a slave device that only sends UDP packets in response to commands.

10 or 100 Mb/sec, half or full duplex UTP Ethernet (auto or manual configuration)
Static IP address and mask
IP/UDP headers use standard format (big endian)
UDP data fields are Intel Format (little endian)
Packets are only sent in response to commands (slave device)
Maximum Ethernet packet length 1500 bytes

The controller will use dynamic host configuration (DHCP) if the IP address is set to 0.0.0.0 in the daughterboard EEPROM.

3.0 Packet Structure

All commands and responses are carried in the data field of UDP packets. The format of the packet is shown below for reference only. The standard Ethernet/IP/UDP header is 42 bytes, but optional fields can increase the length. All packets end with a 4 byte CRC.

14 Byte Ethernet Header

6 byte	48 bit destination address
6 byte	48 bit source address
2 byte	Protocol descriptor (length field in raw Ethernet)

20 Byte IP Header

1 byte	4 bit version, 4 bit header length
1 byte	Type of service
2 byte	Total length
2 byte	Identification
2 byte	3 bits flags, 13 bits fragment offset
1 byte	Time to live
1 byte	Protocol
2 byte	Header Checksum
4 byte	32 bit Source IP Address
4 byte	32 bit Destination IP Address

8 Byte UDP Header
- 2 byte Source Port Number
- 2 byte Destination Port Number
- 2 byte UDP length
- 2 byte UDP Checksum

UDP data field (total IP/UDP header and data must be 46 to 1500 bytes)

Data field pad bytes (if required to provide 64 byte minimum length for Ethernet)

4 Byte Ethernet CRC

This document only defines the UDP data field. All other fields are defined by other standards.

Setting Power Supply Currents

The controller processes all commands in real units. The controller interfaces to the power supply and current transducers through ADCs and DACs that are calibrated in volts. The daughter board contains an EEPROM that stores four conversion factors;

Regulated Transductor amps per ADC volt
Auxiliary Transductor amps per ADC volt
Ground current amps per ADC volt
Power Supply output volts per ADC volt

The controller uses these scale factors to convert between its ADC readings and the actual values. These values are set for each power supply and current transducer. The values are programmed into the daughter board EEPROM since the daughter board has all components that are unique to a given power supply / magnet combination.

The power supply turn ON commands (0xC6 and 0xC7) always set the power supply current to zero before turning on the power supply. After the power supply has been successfully turned on, the power supply current can be set using the 0xC1 and 0xC2 commands.

The controller provides for smoothly ramping the power supply current from one operating point to another. The starting current is always the present setpoint. The setpoint commands specify the final current value and the ramp time. The controller will smoothly change the power supply current from the starting setpoint to the final setpoint in the time requested. The master is responsible for calculating the ramp time for any current change and insuring that the maximum ramp rate of the magnet is not exceeded.

The controller can be programmed to provide a linear ramp or a cosine ramp. The ramp function can be described mathematically as follows;

Is	Setpoint of power supply at start of ramp
If	Final current specified in command
I	Instantaneous value of current
Tr	Ramp time specified in command
T	Integrated time of ramp (time does not change in hold state)

Linear Ramp

$I = Is + (If - Is) * (T / Tr)$

Cosine Ramp

$I = Is + (If - Is) * (1 - \cos(pi * (T / Tr)) / 2$

The ramp commands allow for up to 5 ramps to be specified in one command. Each ramp has a final current value and a ramp time. The controller executes the ramps sequentially starting each ramp as soon as the previous ramp completes. The time remaining is the time to the end of the final ramp.

The ramp may be put into a hold state using a hardware signal. While in the hold state, the ramp time is not incremented and the current setpoint remains unchanged. The basic difference between the 0xC1 and 0xC2 commands is the ramps starts when the command is received and the 0xC2 command starts with hold state waiting for a hardware signal to start.

The controller does not allow new setpoint commands while the power supply is ramping. Any setpoint command received during a ramp will not be processed and the response will indicate an error in the command. If the ramp is in the hold state, any valid setpoint request will abort the current ramp and start a new ramp at the present value of the power supply current.

Reverse Polarity (0xC7) Operation

The controller can be operated with a reversing switch. The daughter board EEPROM has a reversing switch flag that is set to allow running the controller in reverse polarity mode. In reverse polarity operation, only the current in the magnet and the auxiliary transductor are actually reversed. The regulated transductor and DAC remain positive. When in reverse polarity, the controller negates the following values in commands and responses;

All ramp setpoints (0xC1 and 0xC2 commands)
All ramp setpoint Readbacks (0xC3, 0xCA, 0xCF)
DAC Readback (0xC8, 0xCF)
Regulated Transductor Readback (0xC0, 0xC8, 0xCD, 0xCF)

Command Latency

All commands (for UDP port 2000) are processed sequentially in the order received. ARP, PING, and UDP packets for other ports are handled by other tasks. There are 64 buffers in the memory pool used to store commands and responses.

The processing time of most commands is less then 500 microseconds. The controller is capable of processing up to 2000 command/responses per second. There is four commands that take significantly longer to process and can delay other commands.

0xC0 Short status check, waits up to 100 milliseconds for ADC data
0xC5 Turn off command, waits up to 2 seconds for power supply status
0xC6 and 0xC7 Power supply turn ON, waits up to 5 seconds for power supply status and self test
0xE3 Controller Reset requires about 2.5 seconds for the processor to reboot.

Klixon Circuits

The two klixon interlock circuits (Klix 0 and Klix 1) employ current sensing to allow location of a faulted klaxon in a string of up to 40 klixons. Each klixon needs a 100 Kilohm resistor connecting its input to ground. When one klixon opens, the circuit measures the current to determine the number resistor still connected. The current is converted to conductance (in micro mhos). Each 100K resistor adds 10 micro mhos to the conductance.

The controller senses the current by measuring the voltage across a 500 ohm resistor in series with the klixon string. With a 14.5V source voltage, the conductance is calculated with the following formula;

G (micro mho) = 1e6 * I / V = 1e6 * (Vres / 500) / (14.5 - Vres)

5.0 Detailed Command Description

The commands are described in two sections. This section shows the arrangement of the data in the command and response packets. These use C language structures to define the data in the packets. The following section describes the meaning of the data and the values that it may have. There are several defined data types used;

BYTE	unsigned char	8 bit
WORD	unsigned int	16 bit
DWORD	unsigned long	32 bit

The controller responds to 18 commands. The command type is defined by the first byte.

0xC0 Short Status check (wait for current data from ADC)
0xC1 Set Power Supply Current with unsynchronized ramp
0xC2 Set Power Supply Current with synchronized ramp
0xC3 Read Ramp Setpoints and Times
0xC4 Reset Interlocks
0xC5 Turn Off Power Supply
0xC6 Turn On Power Supply
0xC7 Turn On Power Supply in Reverse Polarity
0xC8 Read Analog (reads 8 different ADC inputs)
0xC9 Read Error Message
0xCA Diagnostic Message 1
0xCB Diagnostic Message 2
0xCC Diagnostic Message 3
0xCD Short Status Check (same as 0xC0 except uses stored ADC data)
0xCE Configuration Summary (new message with all configuration data)
0xCF Dynamic Data Message (new message for all dynamic data)
0xE1 Communications Check Message (echo)
0xE3 Reset Controller
(0xF0-FF are reserved for chassis test commands)

The command type is defined by the first byte (first byte in the UDP data field). The second byte is always the response byte set by the controller. A non-zero value indicates that there was an error in processing the command. The third byte is always the task ID. It set by the master and returned unmodified by the controller.

Many of the commands have a byte for channel number for multi-channel controllers. The PEP II Bitbus MCOR controller has 16 channels. The Ethernet controller has only one channel (0) and a non-zero value will return an error.

Massages that cannot be processed return the message with the response code (second byte) set to one of the following;

0x00 Command OK, packet is normal response

0x11 Invalid Command
0x12 Invalid message length
0x13 Invalid channel number
0x14 Invalid number of entries (0xC1-3)

5.1 Short Status Check (0xC0 and 0xCD)

This command returns status bytes 0 and 1, and the power supply output current. To insure that the data is valid after a ramp, the controller gets fresh data from the ADC. However this does add a delay of up to 100 milliseconds in responding to the request. Since all commands are processed in the order received, other requests may be delayed.

The 0xCD command is the same command except that it returns value of the power supply current from the last ADC reading. This eliminates the delay of reading the ADC.

```
typedef struct          // 0xC0 and 0xCD command structure
{
  BYTE   Ctype;         // 00 command type 0xC0 or 0xCD
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_C0;

#define CMD_LEN_C0 4    //expected length of command message

typedef struct          // 0xC0 and 0xCD response structure
{
  BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
  float  PSCur;         // 06-09 PS output current read from reg transductor **
} __attribute__ ((packed))  Rsp_C0;

#define RSP_LEN_C0 10   //length of response message
```

** Negated when in reverse polarity

5.2 Set Power Supply Current Ramp (0xC1 and 0xC2)

The 0xC1 and 0xC2 commands set the power supply current and return status bytes 0 and 1. For the message to be processed successfully, the power supply must be turned on in remote mode and not presently ramping. If the ramp state is hold, the current ramp is aborted and the new request is processed.

The 0xC1 and 0xC2 commands are the same except that the 0xC1 command starts ramping immediately while the 0xC2 puts the ramp in the hold state waiting for a hardware ramp signal.

Each command can specifiy from 1 to 5 individual ramp setpoints. The setpoint defines the current at the end of the ramp and the time of the ramp.

The ramp time is specified by an unsigned 16 bit word. The units are 0.01 seconds for normal ramping or 0.05 for slow ramping, configured by the Daughter board EEPROM. Commands with zero ramp time(s) are rejected.

```
typedef struct          // setpoint
{
  float  Cur;           // Power supply current **
  WORD   Time;          // time in 1/100 second or 1/20 second (slow ramp)
} __attribute__ ((packed))  setp;

typedef struct          // 0xC1 and 0xC2 command structure
{
  BYTE   Ctype;         // 00 command type 0xC1 or 0xC2
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Num;           // 03 Number of setpoints (1 to 5)
  BYTE   Chan;          // 04 Channel number (must be zero)
  setp   Setpoint[5];   // 05-10, 16, 22, 26, 32, Setpoint 1 to 5
} __attribute__ ((packed))  Cmd_C1;

#define CMD_LEN_C1  5   // expected length of command message without setpoints
#define CMD_LEN_SP  6   // length of each setpoint

typedef struct          // 0xC1 and 0xC2 response structure
{
  BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
} __attribute__ ((packed))  Rsp_C1;

#define RSP_LEN_C1  6   // length of response message
```

** Negated when in reverse polarity

5.3 Read Ramp Setpoints (0xC3)

The 0xC3 command is used to read the ramp setpoints. The command specifies the number of setpoints to be returned in the response.

```
typedef struct          // 0xC3 command structure
{
  BYTE   Ctype;         // 00 command type 0xC3
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Num;           // 03 Number of setpoints (1 to 5)
  BYTE   Chan;          // 04 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_C3;

#define CMD_LEN_C3  5   // expected length of command message

typedef struct          // setpoint
{
  float  Cur;           // Power supply current
  WORD   Time;          // time in 1/100 second or 1/20 second (slow ramp)
} __attribute__ ((packed))  setp;

typedef struct          // 0xC3 response structure
{
  BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
  setp   Setpoint[5];   // 06-11, 17, 23, 29, 35, Setpoint 1 to 5
} __attribute__ ((packed))  Rsp_C3;

#define RSP_LEN_C3  6   // length of response message without setpoints
#define LEN_SP      6   // length of each setpoint
```

5.4 Interlock Reset (0xC4)

The 0xC4 command sends a reset signal to the power supply and sets all internal interlocks to the unlatched state.

Internal interlocks are initialized to an unlatched (transparent) state. When the power supply is turned on, the interlocks are set to latch fault conditions. The interlocks are returned to the unlatched condition by an power supply OFF command (0xC5) or the interlock reset command (0xC4).

```
typedef struct          // 0xC4 command structure
{
  BYTE   Ctype;         // 00 command type 0xC4
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, set by sender, returned unchanged
  BYTE   Chan;          // 03 Channel, must be zero
} __attribute__ ((packed))  Cmd_C4;

#define CMD_LEN_C4  4   // expected length of command message

typedef struct          // 0xC4 response structure
{
  BYTE   Ctype;         // 00 command type 0xC4, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
} __attribute__ ((packed))  Rsp_C4;

#define RSP_LEN_C4  6   // length of response message
```

5.5 Power Supply OFF (0xC5)

This command turns off the power supply. After setting the supply to off, the command waits for the power supply status to indicate off. If the power supply status still indicates ON after two seconds, the controller indicates a command error and generates an error message.

The off command returns the power supply interlocks to the unlatched (transparent) state.

```
typedef struct          // 0xC5 command structure
{
  BYTE   Ctype;         // 00 command type 0xC5
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel, must be zero
} __attribute__ ((packed))  Cmd_C5;

#define CMD_LEN_C5  4   // expected length of command message

typedef struct          // 0xC5 response structure
{
  BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
} __attribute__ ((packed))  Rsp_C5;

#define RSP_LEN_C5  6   // length of response message
```

5.6 Power Supply ON (0xC6 and 0xC7)

The 0xC6 and 0xC7 commands turn on the power supply. The checks that the system is in remote mote and then resets the power supply and interlocks. If there are no faulted interlocks, the controller performs a self test of the DAC and ADCs. The DAC is set to zero and the power supply turned on. The processor waits for the power supply status to indicate that the supply is on before sending a response. If after 5 seconds, the supply has still not indicated that it is on, the controller turns the supply off and sends the response with the error flag set.

The 0xC6 and 0xC7 commands are the same except that the 0xC7 command sets the reverse polarity output. If the daughter board EEPROM does not indicate that there is a reversing switch, the command is rejected. When a reversing switch is used, the DAC and regulated transductor voltages remain positive, while the magnet current is negative. The sign of the setpoints and regulated transductor readback are negated when the power supply is turned on in reverse polarity. The auxiliary transductor should be after the reversing switch and the readback is not modified.

```
typedef struct          // 0xC6 and 0xC7 command structure
{ BYTE   Ctype;         // 00 command type 0xC6 or 0XC7
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel, must be zero
} __attribute__ ((packed))  Cmd_C6;

#define CMD_LEN_C6  4   // expected length of command message

typedef struct          // 0xC6 and 0xC7 response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
} __attribute__ ((packed))  Rsp_C6;

#define RSP_LEN_C6  6   // length of response message
```

5.7 Analog Status (0xC8)

This command returns the 8 analog readings available from the Bitbus chassis. The data is the last reading saved to memory by the ADC, which may be several seconds old.

The regulated transductor, auxiliary transductor, ground current, ripple current, and DAC have the ADC voltages scaled to amps by the coefficients in the daughter board EEPROM. The power supply voltage is also scaled to reflect the real output.

The temperature is returned in degrees F and the spare channel returns the voltage read by the ADC.

```
typedef struct          // 0xC8 command structure
{ BYTE   Ctype;         // 00 command type 0xC8
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_C8;

#define CMD_LEN_C8  4   // expected length of command message

typedef struct          // 0xC8 response structure
{ BYTE   Ctype;         // 00 command type, return unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, return unchanged
  BYTE   Chan;          // 03 Channel, return unchanged
  float  RegI;          // 04-07 PS current read from regulated transductor **
  float  AuxI;          // 08-11 PS current read from auxiliary transductor
  float  DacI;          // 12-15 DAC voltage (scaled to amps) **
  float  RipI;          // 16-19 PS current (reg xduct) ripple
  float  GndI;          // 20-23 PS ground current
  float  Temp;          // 24-27 Controller temperature (degrees F)
  float  PsV;           // 28-31 Power supply volts
  float  Spare;         // 32-35 Spare channel voltage
} __attribute__ ((packed))  Rsp_C8;
```

** Negated when in reverse polarity

5.8 Read Error Message (0xC9)

The controller provides error messages as 0-32 character ASCII strings. The strings are stored in a 16 string circular buffer. Up to 15 unread messages can be stored.

When there is one or more unread messages, the ERROR MESSAGE FLAG in status byte 1 is set. The 0xC9 command will return the oldest unread message. If all messages in the buffer have been read, the response will be "MESSAGE BUFFER EMPTY".

```
typedef struct          // 0xC9 command structure
{ BYTE   Ctype;         // 00 command type 0xC9
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_C9;

#define CMD_LEN_C9 4    // expected length of command message

typedef struct          // 0xC9 response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  char   ErrMsg;        // 04-35 Error message, 0 to 32 characters
} __attribute__ ((packed))  Rsp_C9;

#define RSP_LEN_C9   4  // length of response message without error message

#define MAX_LEN_C9  32  // maximum length of error message
```

5.9 Diagnostic Message 1 (0xCA)

This message returns the full 4 bytes of status (Status 0 to 3) along with various other diagnostic registers.

```
typedef struct          // 0xCA command structure
{ BYTE   Ctype;         // 00 command type 0xCA
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, set by sender, returned unchanged
  BYTE   Chan;          // 03 Chan number (must be zero)
} __attribute__ ((packed))  Cmd_CA;

#define CMD_LEN_CA  4   // expected length of command message

typedef struct          // 0xCA response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
  WORD   Status23;      // 06-07 Status bytes 2 and 3
  BYTE   RampState;     // 08 Ramp state
  float  CurSetPnt;     // 09-12 Current Setpoint of ramp (amps) **
  float  StrSetPtn;     // 13-16 Starting setpoint of ramp (amps) **
  DWORD  TimeRem;       // 17-20 remaining time in ramp, 0.01 sec / cnt
  SHORT  Adc2Off;       // 21-22 ADC2 offset correction +/-0x7FFF
  SHORT  Adc2Gain;      // 23-24 ADC2 gain factor +/-0x7FFF
  SHORT  DacOff;        // 25-26 DAC offset +/-7FFF
  SHORT  DacGain;       // 27-28 DAC Gain, NOT USED
  BYTE   LastRst;       // 29 last reset code
  BYTE   LastOff;       // 30 last PS turn off code
  BYTE   CalErr;        // 31 ADC and DAC Calibration error codes
  BYTE   SelfTst;       // 32 Self test code
} __attribute__ ((packed))  Rsp_CA;

#define RSP_LEN_CA  33  // length of response message
```

** Negated when in reverse polarity

5.10 Diagnostic Message 2 (0xCB)

This message returns with the chassis configuration and three 8 character ASCII strings. . The chassis configuration and MagnetID string come from the daughterboard EEPROM. The serial number string come from the motherboard EEPROM. The version string is a constant in firmware.

```
typedef struct          // 0xCB command structure
{ BYTE   Ctype;         // 00 command type 0xCB
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_CB;

#define CMD_LEN_CB  4   // expected length of command message

typedef struct          // 0xCB response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  BYTE   Config;        // 04 Chassis Configuration
  char   SerialNum[8];  // 05-12 Serial number 8 byte string
  char   Version[8];    // 13-20 Firmware Version 8 byte string
  char   MagnetId[8];   // 21-28 Magnet ID 8 byte string
} __attribute__ ((packed))  Rsp_CB;

#define RSP_LEN_CB  29  // length of response message
```

5.11 Diagnostic Message 3 (0xCC)

This message returns the conversion coefficients that convert from ADC reading to the actual units. The RegIva, AuxIva, GndIva convert the ADC reading from volts to amps. The RegIva is also used to convert current setpoints to DAC volts. The PsvIva is the conversion factor for the ADC reading to the power supply output voltage. These coefficients are stored in the daughterboard EEPROM.

The reference voltage (RefVlt) is the voltage of the LM399A reference used to calibrate both ADCs. The calibration date (CalDate) is an 8 character string. Both are stored in the motherboard EEPROM.

```
typedef struct          // 0xCC command structure
{ BYTE   Ctype;         // 00 command type 0xCC
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_CC;

#define CMD_LEN_CC  4   // expected length of command message

typedef struct          // 0xCC response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  float  RegIva;        // 04-07 Regulated transductor amp per ADC volt
  float  AuxIva;        // 08-11 Auxiliary transductor amp per ADC volt
  float  GndIva;        // 12-15 Ground current amp per ADC volt
  float  PsvIva;        // 16-19 PS output volt per ADC volt
  float  RefVlt;        // 20-23 Reference voltage
  char   CalDate[8];    // 24-31 Calibration date ASCII string (8 char)
} __attribute__ ((packed))  Rsp_CC;

#define RSP_LEN_CC  32  // length of response message
```

5.12 Configuration Summary Message (0xCE)

This message returns the all of the static configuration data from the motherboard and daughterboard EEPROMs. It is new for the Ethernet controller and is not supported in the Bitbus controllers.

```
typedef struct          // CE command structure
{ BYTE   Ctype;         // 00 command type 0xCE
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_CE;

#define CMD_LEN_CE  4   // expected length of command message

typedef struct          // CE response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Chan;          // 03 Channel, returned unchanged
  //daughterboard EEPROM data
  char   MagnetId[8];   // 04-11 Magnet ID 8 byte string
  DWORD  ip_adr;        // 12-15 Ethernet IP address
  DWORD  ip_mask;       // 16-19 Ethernet IP mask
  DWORD  ip_gate;       // 20-23 Ethernet IP gateway
  DWORD  ip_dns;        // 24-27 Ethernet IP Domain Name Server
  WORD   EnetConfig;    // 28-29 Ethernet configuration
  WORD   Config;        // 30-31 Chassis Configuration
  float  RegIva;        // 32-35 Regulated transductor amp per ADC volt
  float  AuxIva;        // 36-39 Auxiliary transductor amp per ADC volt
  float  GndIva;        // 40-43 Ground current amp per ADC volt
  float  PsvIva;        // 44-47 PS output volt per ADC volt
  float  DigitalGain;   // 48-51 Digital Regulation Gain
  float  DigitalTC;     // 52-55 Digital Reg time constant
  float  DigitalErrLim; // 56-59 Digital Reg error voltage limit
  //motherboard EEPROM data
  char   SerialNum[8];  // 60-67 Serial number 8 byte string
  char   Version[8];    // 68-75 Firmware date 8 byte string
  WORD   XilinxVersion; // 76-77 Xilinx version
  float  RefVlt;        // 78-81 Reference voltage
  char   CalDate[8];    // 82-89 Calibration date ASCII string (8 char)
  float  a1k1;          // 90-93 ADC1 linearity coefficient K1
  float  a1k2;          // 94-97 ADC1 linearity coefficient K2
  float  a2k1;          // 98-101 ADC2 linearity coefficient K1
  float  a2k2;          // 102-105 ADC2 linearity coefficient K2
  WORD   EnetAddr[3];   // 106-111 Ethernet MAC address (added 4/12/06)
} __attribute__ ((packed))  Rsp_CE;

#define RSP_LEN_CE  112 // length of response message
```

5.13 Diagnostic Message (0xCF)

This message returns all dynamic data for the controller, except for error messages. It is new for the Ethernet controller and is not supported in the Bitbus controllers.

```
typedef struct          // 0xCF command structure
{ BYTE   Ctype;         // 00 command type 0xCF
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Chan;          // 03 Channel number (must be zero)
} __attribute__ ((packed))  Cmd_CF;

#define CMD_LEN_CF  4   // expected length of command message

typedef struct          // 0xCF response structure
{ BYTE   Ctype;         // 00 command type, return unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, return unchanged
  BYTE   Chan;          // 03 Channel, return unchanged
  WORD   Status01;      // 04-05 Status bytes 0 and 1
  WORD   Status23;      // 06-07 Status bytes 2 and 3

  float  RegI;          // 08-11 PS current read from regulated transductor **
  float  AuxI;          // 12-15 PS current read from auxiliary transductor
  float  DacI;          // 16-19 DAC voltage (scaled to amps) **
  float  RipI;          // 20-23 PS current (reg xduct) ripple
  float  GndI;          // 24-27 PS ground current
  float  Temp;          // 28-31 Controller temperature (degrees F)
  float  PsV;           // 32-35 Power supply volts
  float  Spare;         // 36-39 Spare channel voltage
  float  AGnd;          // 40-43 Absolute round current
  float  Klix0;         // 44-47 Klixon 0 conductance
  float  Klix1;         // 48-51 Klixon 1 conductance
  float  VP15U;         // 52-55 Unregulated +15V
  float  VM15U;         // 56-59 Unregulated -15V
  float  VP15R;         // 60-63 Regulated +14.5
  float  VM15R;         // 64-67 Regulated -14.5
  float  A10V;          // 68-71 Unregulated +15V
  float  A5V;           // 72-75 Regulated +14.5
  float  VDD;           // 76-79 Digital 5.0 volts
  float  V3D3;          // 80-83 Digital 3.3 volts
  float  V2D5;          // 84-87 Digital 2.5 volts
  float  V1D2;          // 88-91 Digital 1.2 volts
  SHORT  FanRPM;        // 92-93 Fan RPM

  SHORT  Adc2Off;       // 94-95 ADC2 offset correction +/-0x7FFF
  SHORT  Adc2Gain;      // 96-97 ADC2 gain factor +/-0x7FFF
  SHORT  Adc1Off;       // 98-99 ADC1 offset correction +/-0x7FFF
  SHORT  Adc1Gain;      // 100-101 ADC1 gain factor +/-0x7FFF
  SHORT  DacOff;        // 102-103 DAC offset +/-7FFF
  SHORT  DacGain;       // 104-105 DAC Gain, NOT USED

  BYTE   LastRst;       // 106 last reset code
  BYTE   LastOff;       // 107 last PS turn off code
  BYTE   CalErr;        // 108 ADC and DAC Calibration error codes
  BYTE   SelfTst;       // 109 Self test code

  BYTE   RampState;     // 110 Ramp state
  BYTE   NumSetP;       // 111 Number of setpoints
  float  CurSetPnt;     // 112-115 Current Setpoint of ramp (amps) **
  float  StrSetPtn;     // 116-119 Starting setpoint of ramp (amps) **
```

```
  DWORD  TimeRem;       // 120-123 remaining time in ramp, 0.01 sec / cnt ***
  float  Cur1;          // 124-127 Ramp #1 Power supply current
  WORD   Time1;         // 128-129 Ramp #1 time, 0.01 sec / cnt ***
  float  Cur2;          // 130-133 Ramp #2 Power supply current
  WORD   Time2;         // 134-135 Ramp #2 time, 0.01 sec / cnt ***
  float  Cur3;          // 136-139 Ramp #3 Power supply current
  WORD   Time3;         // 140-141 Ramp #3 time, 0.01 sec / cnt ***
  float  Cur4;          // 142-145 Ramp #4 Power supply current
  WORD   Time4;         // 146-147 Ramp #4 time, 0.01 sec / cnt ***
  float  Cur5;          // 148-151 Ramp #5 Power supply current
  WORD   Time5;         // 152-153 Ramp #5 time, 0.01 sec / cnt ***
} __attribute__ ((packed))  Rsp_CF;
```

** Negated when in reverse polarity
*** 0.05 sec / cnt when slow ramp set

5.14 Communications Check Message (0xE1)

This message allows checking for communication between the master and the controller. The controller returns the message as sent with the data byte set to 0xFF.

```
typedef struct          // 0xE1 command structure
{ BYTE   Ctype;         // 00 command type 0xE1
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID
  BYTE   Data;          // 03 Data byte, (don't care)
} __attribute__ ((packed))  Cmd_E1;

#define CMD_LEN_E1  4   // expected length of command message

typedef struct          // 0xE1 response structure
{ BYTE   Ctype;         // 00 command type, returned unchanged
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Data;          // 03 Data byte, set to 0xFF
} __attribute__ ((packed))  Rsp_E1;

#define RSP_LEN_E1   4  // length of response message
```

5.15 Reset Controller (0xE3)

The command is used to reset the controller. A soft reset (0x00) only resets the MOD5282 processor. The power supply state and interlocks are not effected. If a ramp is in progress, it will continue after the processor is reset.

A hard reset (0x01) resets the processor and the Xilinx FPGA. This turns off the power supply and clears all status information. The hard reset is not allowed when the controller is in local mode.

There is no response message for the reset command.

```
typedef struct          // E3 command structure
{ BYTE   Ctype;         // 00 command type 0xE3
  BYTE   Rsp;           // 01 response code
  BYTE   Tid;           // 02 Task ID, returned unchanged
  BYTE   Type;          // 03 Reset type, 0x00 is soft reset, 0x01 is hard
reset
} __attribute__ ((packed))  Cmd_E3;

#define CMD_LEN_E3 4    // expected length of command message
```

6.0 Description of Data

Status byte 0

D0 Command OK
D1 Command Error
D2 Power Supply Off
D3 Ramp On (set current request 0xC1)
D4 Ramp On (synchronized ramp 0xC2)
D5 Ramp Ready (0xC1 or 0xC2 in hold state)
D6 Reverse polarity (set by 0xC7 command)
D7 Local Mode (set by local control board)

The controller will always set either the Command OK (D0) or the Command Error (D1) bit in the response message. Command OK indicates that the request was completed and Command error indicates that the command was not completed successfully.

The Power Supply OFF bit (D2) is set high when the power supply is off and low when the power supply is on. The state is the value that the controller is requesting of the power supply, not the value returned by the power supply.

Status byte 1
D0 Error Message Flag (message available)
D1 ADC Failure (ADC1 or ADC2 not responding)
D2 Calibration Fault
D3 Auxiliary Transductor Fault (status only)
D4 Interlock Fault (system not ready)
D5 Spare
D6 Spare
D7 Spare

The error message flag indicates that at least 1 (15 maximum) error message has not been read using the 0xC9 command.

The ADC failure flag indicates that 1 or both ADCs are not providing data. The power supply cannot be turned on with this flag set. It will not turn the supply off if it faults after turn on.

The calibration fault flag is set when one of the ADC or DAC offset or gain coefficients has reached its maximum allowable value. The coefficients are limited to a range of about +/- 0.1% of the full scale of the device.

The auxiliary transductor fault indicates the unit is not functioning properly. The transductor is for diagnostic purposes only and the fault does not prevent the power supply from operating.

The interlock fault indicates that one or more interlocks are faulted. A low indicates that the power supply is ready for turn on.

Status byte 2

D0	Magnet Interlock 0 Fault and Klixon 0 Fault
D1	Magnet Interlock 1 Fault and Klixon 1 Fault
D2	Magnet Interlock 2 Fault
D3	Magnet Interlock 3 Fault
D4	Power Supply Fault (supply not ready)
D5	Regulated Transductor Fault
D6	Ground Fault
D7	Controller Fault

D0 and D1 are shared between two inputs. Only one of the two inputs should be used. Both inputs need to be faulted to indicate a fault.

The controller fault indicates that Xilinx FPGA is not reading the interlock status properly. The interlock status is read serially through external parallel to serial chips. The last 2 bits in the chain are supplied by the Xilinx and are used to confirm the data integrity.

Status byte 3

D0	Fault Latch ON
D1	Power Supply Status 0
D2	Power Supply Status 1
D3	Power Supply Status 2
D4	Power Supply Status 3
D5	Power Supply On Status
D6	Spare
D7	Spare

Fault latch ON indicates that all interlocks will latch faults. The fault must clear and the fault latch tuned off (0xC4 or 0xC5) before the power supply can be turned on.

The power supply status flags are read from the power supply. The controller only reports the state of the flags and takes no action based on their state. Note that most of the supplies used at SLAC do not provide status. Most large supplies (>30KW) do provide status but the definition of the flags changes between power supply families.

Calibration Error Byte (CalErr)

This byte contains flags that indicate an internal calibration has reached its limit. This typically indicates a hardware problem in the controller. The flags (bits) have the following assignments;

0x01 ADC1 offset Error
0x02 ADC1 gain Error
0x04 ADC2 offset Error
0x08 ADC2 gain Error
0x10 DAC offset Error
0x20 Digital regulation error

Hardware Fault Byte (HrdFlt)

The hardware fault register indicates serious controller hardware faults. Power supply turn on is inhibited when a fault flag is set.

0x01 ADC1 read fault
0x02 ADC2 read fault
0x04 Motherboard EEPROM checksum error
0x08 Daughter board EEPROM checksum error
0x10 Xilinx fault (failed write/readback test)

Last Reset Byte (LastReset)

The last reset byte indicates the source of the last reset.

0x01 Power On reset
0x02 Software reset
0x04 UART (Diagnostic) reset
0x08 Local control reset
0x10 Watchdog reset

Last Power Supply Turn Off (LastOff)

This byte indicates the source of the last power power turn off. The power supply off command (0xC5) set the byte to zero. Any interlock that trips the power supply sets a flag in the byte.

0x01 Magnet Interlock 0
0x02 Magnet Interlock 1
0x04 Magnet Interlock 2
0x08 Magnet Interlock 3
0x10 Power Supply Ready Interlock
0x20 Regulated Transductor Interlock
0x40 Ground Current Interlock
0x80 Serial data fault

Mother Board and Daughter Board EEPROMs

The mother board and daughter board EEPROMs are 1024 bit serial EEPROMs organized as 64 (16 bit) words. Data is saved in the first 32 words and a redundant copy in the upper 32 words. The last word is set to insure the checksum is 0xA5A5.

The EEPROMs can be programmed using a ASCII terminal emulator connected to UART0 or UART2. UART0 is located in the 15 pin D connector on the front panel. It operates at 115200 baud, 8 bit, no parity, and no flow control. The port is also the default port for the processor and can be used to download programs. The second port (UART2) is compatable with the Bitbus chassis. It is available on the 9 pin D connector on the local control board. It operates at 9600 baud, 8 bit, no parity, and no flow control.

The hardware connectors are as follows

Signal	9 Pin	15 Pin
TX	2	9
RX	3	1
CTS	7	2
RTS	6	10
COM	5	5

Mother Board EEPROM Data

The mother board EEPROM contains data unique to the controller. This includes the chassis serial number, reference voltage, and calibration data. There are four optional ADC linearity coefficients that can be used to cancel most most of the non-linearity of the two ADCs.

```
typedef struct            // mother board EEPROM structure (32 words)
{ VWORD   SerialNum[4];   // 00->03 chassis serial number, 8 characters
  VWORD   CalDate[4];     // 04->07 calibration date, 8 characters
  float   ref;            // 08->09 reference voltage (LM399A)
  VSHORT  a1k1;           // 10     ADC1 linearity coefficient K1
  VSHORT  a1k2;           // 11     ADC1 linearity coefficient K2
  VSHORT  a2k1;           // 12     ADC2 linearity coefficient K1
  VSHORT  a2k2;           // 13     ADC2 linearity coefficient K2
  VWORD   pack00[17];     // 14->30 packing
  VWORD   CheckSum;       // 31     structure checksum, set for 0xA5A5
} mb_eep_struct;
```

Daughter Board EEPROM Data

The daughter board EEPROM contains data unique to a power supply and magnet. This allows a controller to be changed without any configuration. There are four main groups of data; Ethernet configuration, Power Supply Coefficients, digital regulation coefficients, and controller configuration. There is also an 8 character ASCII string that is programmed with the magnet (string) name.

The Ethernet configuration allows the IP address, IP mask, IP gateway, and default domain name server (DNS) to be specified. For the intended mode of operation

(master/slave UDP) only the IP address and mask are required. If the IP address is set to 0.0.0.0, the controller uses the dynamic host configuration protocol (DHCP) to set its IP address. If Ethernet defaults to auto-configuration but can be manually configured for 10 or 100 Mb/sec at either half or full duplex.

There are four coefficients that are used to convert between power supply units (amps) and ADC volts. The ADC input is intended to be +/-10 volts with a 20% over range.

A typical transductor provides a 10V signal for 150 amps, which is specified as 15 amps/volt. A typical ground sense resistor is 100 ohms, which is specified as 0.01 amps/volt.

The configuration byte is used to configure the controller for special modes such as bipolar operation or digital regulation.

0x01 Enable linear ramps
0x02 Enable slow ramp (time in 1/20 seconds)
0x04 Enable hardware Hold signal for all ramps
0x10 Digital Regulation Enabled
0x40 Power supply has reversing switch
0x80 Power supply is bipolar

At this time, the digital regulation function is not fully implemented and tested. A basic proportional-integrator (PI) with difference limit has been included. Additional diagnostic code need to be included to allow for testing of the coefficients with the power supply and magnet. To save EEPROM space for future additions, the digital regulation coefficients are floating point numbers with the lower 16 bits of the mantissa removed.

```
typedef struct            // daughter board EEPROM structure (32 words)
{ VDWORD  ip_adr;         // 00->01 Ethernet IP address
  VDWORD  ip_mask;        // 02->03 Ethernet IP mask
  VDWORD  ip_gate;        // 04->05 Ethernet IP gateway
  VDWORD  ip_dns;         // 06->07 Ethernet IP Domain Name Server
  float   reg_iva;        // 08->09 regulated transductor volts/amp
  float   aux_iva;        // 10->11 auxiliary transductor volts/amp
  float   gnd_iva;        // 12->13 ground current readback volts/amp
  float   psv_iva;        // 14->15 power supply voltage readback volts/volt
  VWORD   MagnetId[4];    // 16->19 magnet identification, 8 characters
  VWORD   Config;         // 20     power supply configuration flags
  VWORD   EnetConfig;     // 21     Ethernet configuration
  VWORD   DigitalGain;    // 22     Digital Regulation Gain, 16 bit float
  VWORD   DigitalTC;      // 23     Digital Reg time constant, 16 bit float
  VWORD   DigitalErrLim;  // 24     Digital Reg err voltage limit, 16 bit float
  VWORD   pack00[6];      // 25->30 packing
  VWORD   CheckSum;       // 31     structure checksum, set for 0x5A5A
} db_eep_struct;
```