CPP Configuration Upload/Download

1.0 General

The CPP supports a complete configuration upload and download. The upload sends the configuration set up from the CPP to a computer. The download receives data from a computer that configures the CPP.

The upload feature is a single command that commands a CPP to upload its entire configuration. The download is actually conducted in data groups. The data strings that are send from the CPP in an upload command, with the exception of the direction character, are identical to the strings that can be downloaded to the CPP to set up the configuration. Referencing section 2.0, below, setting a particular bit in the NNN field will result in the CPP returning the central direction character instead of the remote direction character. Therefore, the data strings defined in section 3.0, Upload Configuration Response below, also define the download strings.

Generic strings can be downloaded to any CPP. If the CPP receives a download string for a function that is not supported in its firmware, the string is ignored.

2.0 Upload Configuration Command

The CPP will transmit its entire configuration when the following command is received. The commas can also be spaces.

D_C,III,VVV,NNN,CC,<crlf>

Where;

D_C	is a > sign indicating that the command is from a central
III	is the CPP's station ID.
VVV	is the command code CFFH.
NNN	If the first N is a zero, then the CPP starts each string of data with a D_C character instead of a D_R character.

After having sent its entire configuration, the CPP sends an End of Message string. Although it is not of importance, strings from a CPP are sent in the order presented below. If a CPP does not support a function, then it is skipped.

The field portions of the upload/download strings are transferred as hexadecimal characters. Although some of the information in the strings could be transferred as ASCII characters, the overall majority of information is in a binary format. Therefore, using a hexadecimal transfer protocol accommodates both types of information, and results in consistent transmission and recovery schemes.

3.0 Upload Configuration Response

The CPP responds to the above command in the following format;

D_R,III,Cxx,NNN,[Field,]CC,<crlf>

Where;

Cxx is defined as follows;

C01	Station name/ID/Avg Int/Chn's En/Prt Frmt/Cart Rec/$ID/Modbus
C02	Channels Setup
C03	Data Validity
C04	I/O labels
C05	Sample delay
C06	Boolean Setup
C07	Alarm setup
C08	Auto prints/Chn Holds
C09	Digital calibration setup
C10	Serial calibration setup
C11	Sequencers
C12	Computed channels
C13	Manufacturer parameters
C14	External I/O/Temet
C15	External I/O Modbus
C16	Curve Fit Set UP
C17	Front panel/LCD
C18	Calibrator Setup
C19	Modem/Feedback
C20	Met Parameter Setup

3.1 Name/ID/Pswd/A I/Prt Format/NOCE/mm/dd or dd/mm/Cart Rec/$ID - C01 Poller Inactivity

D_R,III,C01,NNN,[Field,]CC,<crlf>

Where;

Field = 32 hex characters of station name<dl>
6 hex characters of station ID<dl>
12 hex characters of password<dl>
6 hex characters P, I, F averages<dl>
2 hex characters for print format<dl>
2 hex characters for number of channels enabled<dl>
2 hex characters for date format<dl>
6 hex characters for cartridge recordings<dl>
6 hex characters for $ID(01,H,h)<dl>
12 hex characters for Modbus setup<dl>
6 hex characters for Poller Inactivity<dl>

3.2 Channels Setup - C02

Channel setup download for the CPP is conducted one channel at a time.

Where;

Field = 2 hex characters indicating channel number<dl>
12 hex characters of channel name<dl>
12 hex characters of channel units<dl>
12 hex characters of channel full scale value<dl>
12 hex characters of channel zero value<dl>
10 hex characters for manufacturers & model #<dl>
22 hex characters for next 11 definitions<dl>

3.3 Data Validity Setup - C03

The data validity setup is sent in blocks of two channels.

Where:

Field = 2 hex characters with starting channel number<dl>
2 hex characters for setup or label<dl>
2 hex characters for NGOE<dl>
4*NGOE hexadecimal characters for next 2*NGOE digital input validity characters<dl>
4*NGOE hexadecimal characters for next 2*NGOE digital output validity characters<dl>
Repeated for one more channel

3.4 I/O Bit Labels - C04

I/O bit labels are send one bit at a time.

Where;

Field = 2 hex characters with bit number<dl>
2 hex character I<dl>
40 hex characters with input bit label<dl>
2 hex characters with bit number<dl>
2 hex character O<dl>
40 hex characters with output bit label<dl>

3.5 Sample Delay - C05

The CPP firmware supports an assembly selectable number of sample delays. The sample delay setups are transferred one delay setup at a time.

Where;

Field = 2 hex characters indicating the sample delay number<dl>
2 hex characters for setup or label<dl>
10 hex characters for setup<dl>
2 hex characters for NGOE<dl>
6*NGOE hex characters for input settings<dl>
6*NGOE hex characters for output settings<dl>

3.6 Boolean Setup - C06

The CPP firmware supports an assembly selectable number of Boolean functions. Each Boolean function has 10 terms for the IF statement and 10 terms for the THEN statement. The Boolean function setups are transferred one at a time, as are the labels.

Where;

Field = 2 hex characters indicating the Boolean function number<dl>
2 hex characters for setup or label<dl>
84 hex characters defining the function<dl>
120 hex characters for label<dl>

3.7 Alarm Setup - C07

The CPP supports an assembly selectable number of alarming channels. Alarm channel setup is transferred one channel at a time.

Where;

Field = 2 hex characters indicating alarm number<dl>
2 hex characters for setup or label<dl>
10 hex characters with setup<dl>
2 hex characters for NGOE<dl>
4*NGOE hex characters with alarm output settings<dl>
4*NGOE hex characters – inhibit alarm<dl>
42 hex characters, this alarm label<dl>

3.8 AutoPrints/Holds - C08

The automatic print setup, and channel holds are sent in a single string.

Where;

Field = 4 hex characters for auto prints<dl>
2 hex characters for NGOE<dl>
2+(4*NGOE) hex characters for preliminary<dl>
2+(4*NGOE) hex characters for interim<dl>
2+(4*NGOE) hex characters for final<dl>
2 hex characters for NOC<dl>
(NOC*2) hex characters for Holds<dl>

3.9 Digital Calibration Setup - C09

The CPP supports an assembly selectable number of calibration channels. The digital portion of the calibration setup is transferred one calibration channel at a time.

Where;

Field = 2 hex characters indicating the calibration channel number<dl>
4 hex characters for channel setup<dl>
2 hex characters for # of global spans<dl>
2 hex characters for NGOE<dl>
2*NGOE hex characters for inhibit bits<dl>
2*NGOE hex characters for alarm settings<dl>
2 hex characters indicating cal corrections<dl>
6+(2*NGOE) hex characters for zero<dl>
6+(2*NGOE) hex characters for span 1<dl>
6+(2*NGOE) hex characters for span 2<dl>
6+(2*NGOE) hex characters for span 3<dl>
6+(2*NGOE) hex characters for span 4<dl>
6+(2*NGOE) hex characters for span 5<dl>

6+(2*NGOE) hex characters for span 6<dl>

3.10 Serial Calibration Setup - C10

The CPP serial calibrations are triggered from and tied to the settings of the output bits. Each true or false transition of a bit can initiate up to four independent commands. There are 24 bytes associated with each output bit. The serial calibration setup is downloaded output bit at a time.

Where;

Field = 2 hex characters indicating which output bit<dl>
24 hex characters for the low going transition<dl>
24 hex characters for the high going transition<dl>

3.11 Sequencers - C11

The CPP supports an assembly selectable number of sequencers. Each sequencer requires four string transfers.

Where for T1;

Field = 2 hex characters indicating the sequencer number<dl>
2 hex characters indicating T1<dl>
2 hex characters for NGOE<dl>
112 hex characters for setup parameters<dl>
32+(16*NGOE)

Where for T2;

Field = 2 hex characters indicating the sequencer number<dl>
2 hex characters indicating T2<dl>
2 hex characters for NGOE<dl>
130 hex characters for setup parameters<dl>
(6+(4*NGOE))*5

Where for T3;

Field = 2 hex characters indicating the sequencer number<dl>
2 hex characters indicating T3<dl>
2 hex characters for NGOE<dl>
112 hex characters for setup parameters<dl>
8+(6+(4*NGOE))*4

Where for T4:

Field = 2 hex characters indicating the sequencer number<dl>
2 hex characters indicating T4<dl>
84 hex characters for sequence name<dl>
SEQLS

3.12 Computed Channels Setup - C12

The computed channel setup is transferred channel at a time.

Where;

Field = 2 hex characters for channel number<dl>
2 hex characters for setup or label<dl>
40 hex characters for setup<dl>

3.13 Manufacturer Parameters - C13

Any special setup parameters associated with a given manufacturer are provided here.

Where;

Field 2 hex characters for manufacturer<dl>
2 hex characters to identify parameter<dl>
hex characters as needed<dl>
R = R&P
01 = PM10
01 = 1st 20 chan's, 02 = 2nd chan's
commands sent = 1st = 20, 2nd = (NOC-20)<dl>
6 hex characters each<dl>

3.14 External I/O - C14

Setup parameters for DAC outputs devices, external digital I/O and external analog to digital conversion are provided here.

Where;

Field = 2 hex characters for NOAO<dl>
2 hex characters to identify parameter<dl>
01 = #1 Internal/External DAC outputs<dl>
8 hex characters for DACIM<dl>
4 hex characters per DACRM<dl>
NOAO times
02 = External ADC input
03 = External Digital I/O
18 hex characters for EXOM (output)<dl>
18 hex characters for EXIM (input)<dl>
04 = Internal DAC<dl>
4 hex characters for OBDOS<dl>
10 = Temet Calcmet
24 hex characters for TTDSUM<dl>
11 = Temet Calcmet
(NOGE*8)*2 hex characters for TTDDIG<dl>
12 = Temet Calcmet
(NGOE*8)*2 hex characters for TTRDIG<dl>
13 = Temet Calcmet
16 hex characters for TTDSNM<dl>

3.1 5 Modbus Output's - C15

Setup parameters for the Modbus outputs are provided here.

Where;

Field = 2 hex characters for NOMO<dl>
2 hex characters to identify parameter<dl>
05 = Modbus Output
8 hex characters for MACIM<dl>
4 hex characters per MACRM<dl>
NOMO times

3.16 Curve Fit Setup - C16

The curve fit parameters are transferred one curve fit setup at a time.

Where:

Field = 2 hex characters for NCFC<dl>
2 hex characters for curve fit #<dl>
2 hex characters for MNDV<dl>
2*((MNDV*4)+10) hex characters<dl>

3.17 Front Panel/LCD - C17

The operating configuration set up for the front panel is transferred as given below. The bytes are transferred in order, including checksum. The checksum is also transferred with each screen.

Where:

Field = 2 hex characters for number of bytes that follow<dl>
(not counting this byte or <dl>)
= 2 hex characters for All or More or Screens<dl>
= A = All, M = More, S = Screens
= A & M = LCD Field defined above, More follows this field.
= For S = Screens
= hex characters in screen+2 for CS<dl>
= S for screens
= 2 hex characters for screen number<dl>
= Screen definition + CS (77+2)=158 hex characters<dl>

3.18 Calibrator Setup – C18

The setups for the calibrators are transferred as follows.

Where:

Field = 2 hex characters for Manu, A7 = API Mdl 700
2 hex characters for calibrator #, 1 or 2<dl>
2 hex characters for CPP channel #<dl>
8 hex characters for conc
2 hex characters for units, 1=ppb, 2=ppm, 3=ppt;
4=pct, 5=ug/m^3, 6=mg/m^3<dl>
Repeated 10 times (seq #1-10)

3.19 Modem/Feedback – C19

Where:

Field = 2 hex characters for modem =M<dl>
170 hex characters for string<dl>

Where:

Field = 2 hex characters for FB loop=F<dl>
2 hex characters for loop number<dl>
2 hex characters for bytes to follow<dl>
FBMOFS*2 hex characters<dl>
CS sent with last loop

3.20 Met Parameters Setup – C21

The meteorological parameters are transferred one channel at a time.

Where for T1;

Field = 2 hex characters indicating what follows<dl>
= C = channel assignments
2 hex characters indicating channel number<dl>
6 hex characters for MCAW<dl>
4 hex characters for HWAM<dl>
= R = rainfall set up
2 hex characters = R<dl>
4 hex characters per RF set up, separated with delimiters<dl>
back to back delimiters = end of RF set ups
=F = counter set up
2 hex characters = F<dl>
2 hex characters for counter number<dl>
30 hex characters for WSPCM<dl>

4.0 Configuration Download

To download data, the CPP is sent a command indicating that configuration strings are to be sent. The CPP clears all relevant configuration files, suspends data collection, and then sends an <,OK,<crlf> indicating that it is ready to accept the incoming strings. After all strings have been sent, the CPP calculates new checksums and restarts operation by generating a soft power reset. Either detecting the End Of Message string, or fifteen seconds of inactivity in the comm port will terminate the download sequence and the CPP returns to data collection.

The command string indicating that download strings are coming is given below. This command string is the first string sent upon receiving the upload configuration. This allows the strings received from a CPP to be stored in a file, and then turned around and downloaded to the CPP. Downloading data strings requires a proper checksum for the CPP to accept the string. In download, the CPP does not honor a carriage return in place of the checksum.

D_C,III,CF0,000,CC<crlf>

After clearing all existing configurations, the CPP returns a <,OK,<crlf> indicating that it is ready to accept the first download string. **Each string must be processed by the CPP, and an <,OK,<crlf> received before the next string is sent.** If the acknowledge message is not received in ten seconds, the CPP must have detected an error in the received string, and the string should be sent again. This process is repeated until all strings have been downloaded. Since the CPP's can be daisy chained, it cannot return a NAK when a reception error is detected. Upon detecting the EOT or a timeout, the CPP restarts to process the downloaded configuration.

The download is terminated with the following End of Message string.

D_C,III,CF0,EOT,CC<crlf>

After receiving this End Of Message string during a download, the CPP calculates new checksums, operating parameters, returns a download completed message and restarts.

The download completed message contains any error conditions that occurred during the download. The message is as presented below. Bytes E_1 through E_6 indicate errors in processing a particular parameter, as given below. With the exception of bit2 in E_6, which indicates that the download configuration contained a parameter not supported by the firmware installed in the CPP, all other bits should be zero for a proper download.

D_R,III,CF0,$E_1E_2E_3E_4E_5E_6E_7E_8$,CC<crlf>

Where;

E_1	E_2		E_3	E_4		E_5	E_6	
xxxx	xxx1	Name/Pswd	xxxx	xxx1	Digital cals	xxxx	xxx1	Met
xxxx	xx1x	Channel setup	xxxx	xx1x	Serial cals	xxxx	xx1x	Not Ours
xxxx	x1xx	Validity	xxxx	x1xx	Sequencers	1xxx	xxxx	Timeout
xxxx	1xxx	I/O labels	xxxx	1xxx	Comp Ch	xxxx	xxxx	Not
xxx1	xxxx	Sample Delay	xxx1	xxxx	Manu Param			Defined
xx1x	xxxx	Boolean	xx1x	xxxx	DAC,Ex I/O			
x1xx	xxxx	Alarm	x1xx	xxxx	Crv Fit			
1xxx	xxxx	AutoPrints	1xxx	xxxx	LCD			

For $E_7$and E_8 see section 5.0 below. E_7 and E_8 are the ASCIEC register.

5.0 CPP Error Indications

The CPP keeps an internal register that can be very useful when developing an interface using this protocol. Obviously the command string and checksum need to be correct before the CPP can respond. Should the CPP not respond, it loads this internal register with a code. This code can be reviewed with the M keyin. The register name is ASCIEC, and its absolute memory location can be found with the M keyin. Entering an M[ASCIEC Addresss- $9CF5_H$, e.g.] results in the CPP printing the contents of this register. The brackets are for clarity only an are not to be included in the key in. The various error conditions are given below in Table 5.0. Reported error conditions could also be the result of a missing delimiter.

Table 5.0
CPP Interface Error Conditions

Code	Indication	E7	E8
1	Not our address	0	1
2	Not defined	0	2
3	V field in error	0	3
4	N field in error	0	4
5	Unknown command	0	5
6	If B, then in error	0	6
7	If Z, then in error	0	7
8	If checksum, then in error	0	8
9	Error in Bxxx	0	9
A	Error in Zxxx	0	A
B	Error in final request	0	B
C	Error in interim request	0	C
D	Error in preliminary request	0	D
E	Error in field	0	E

6.0 Upload/Download Configuration To RamPack

The PCMCIA RamPack cards have a section that is reserved for storing CPP configurations. CPP configurations can be uploaded into this section, and likewise, configurations stored in this section can be downloaded into the CPP. Data is stored in the RamPack identical to that uploaded over a comm port. Therefore, this configuration stored in the RamPack can be further uploaded into a computer for archiving, and downloaded at a later date. Likewise, configurations stored in the computer can be loaded into RamPack's and taken to a remote site for configuration download.

6.1 Upload CPP Data To The RamPack

To upload configurations from the CPP into the RamPack, enter the following command;

CFGUL<cr>

The CPP will respond with;

This Will Overwrite Any RamPack Stored Configuration
Do You Want To Continue Y/N

Entering a Y will result in the CPP uploading its' configuration into the RamPack. Entering an E will result in the CPP recording the configuration in the on board EEPROM. This takes several seconds and the CPP will print the data strings being stored, and will output the End Of Message string when the upload is completed. Any other input aborts the upload. If a write protected RamPack is installed, the CPP will so advise and exit the upload sequence. If a RamPack is not installed, the CPP asks

Upload To Onboard EEPROM, Y/N =

Entering a Y will upload the configuration stored in the CPP ram into the non volatile EEPROM memory. Any other entry aborts the upload.

6.2 Download Data From The RamPack To The CPP

To download configurations stored in the RamPack, enter the following command;

CFGDL<cr>

The CPP will respond with;

This Will Overwrite Any CPP Setup
Do You Want To Continue Y/N

Entering a Y will result in the CPP downloading the configuration stored in the RamPack into the CPP. Any other input aborts the command.

If a RamPack is installed, but there is no configuration stored in the RamPack, the CPP will output a message and not attempt the configuration download. The setup parameters in the CPP will not be changed. If a RamPack is not installed, the CPP will check the onboard EEPROM for a stored configuration and if one is stored, the CPP downloads the configuration into the operating ram. If the EEPROM does not have a stored configuration, the CPP outputs a message. At the completion of the download, the CPP will output the results of the download. During the download, the CPP outputs the strings being loaded into the CPP.