CP200X

COMMUNICATION PROTOCOL

1.- COMMUNICATION PROTOCOL.

The CP3000 transducers are equipped with a serial line in order to be able to communicate all the measured variables to any device with serial input capability. There are two different versions, one as RS232 standard, and the other as RS485. The first one can be used when the connection is point to point, and the second in multidrop connections. In this case, up to 32 devices, - as the standard specifies-, can be connected to the same communication line, to a maximum length of 1200 meters, allowing gathering data coming from any measurement unit.

Both versions are isolated from the input circuitry by means of transformers. This avoids problems in earth connection, breaking any grounding loops, and allows to connect the cable screen to earth in the best point of the installation.

Although the signal levels are different, the protocol used in both is the same, and complies with the RTU JBUS/MODBUS protocol. This is a master-slave protocol, very common in PLC's. The communication is always started by the master, and no spontaneous messages are allowed. The standard transmission rate is 9600 Bps, 8 bits, no parity, and one stop bit. Other speeds are possible, programming the device, from 300 to 19.200 Bps.

Each transaction consists of one request frame, generated by the master, and one reply frame, generated by the slave. In the case the master sends a writing command, the slave sends an acknowledge message. No provisions have been done for answering exception conditions. If any command is received in a correct way, it will be executed. If not, no answer will be generated.

Function codes accepted are:

03H	Read Holding Registers (3xxxx).
04H	Read Input Registers (4xxxx).
06H	Preset Single Register (6xxxx)
10H	Preset Multiple Registers

1.1. – FRAME DESCRIPTION.

1.1.1. - DATA POLLING (Commands 03h or 04H)

Each frame consists of:

- Identity number	one byte
- Command code	04H or 03H
- Data address	two bytes: H, L
- Number of registers to be read	two bytes: H, L
- CRC	two bytes: L, H

1.1.2. – WRITING COMMANDS.

Preset Single Register.

- Identity number	one byte
- Command code	06H
- Data address	two bytes: H, L
- Variable value	two bytes
- CRC	two bytes: L, H

Preset Multiple Registers

- Identity number	one byte
- Command code	10H
- Data address	two bytes: H, L
- Number of words to be written	two bytes: H, L
- Number of bytes to be written	one byte
- Variable value	four bytes
- .	.
- .	.
- Variable value	four bytes
- CRC	two bytes: L, H

1.1.3. – ANSWER FRAME

Each frame consists of:

- Identity	one byte
- Command code (the same as received)	04H, or 03H
- Number of bytes sent	one byte
- Variable value	four bytes
- .	.
- .	.
- .	.
- Variable value	four bytes
- CRC	two bytes: L, H

1.1.4. – ACKNOWLEDGE FRAME.

It consists of:

- Identity	one byte
- Command code (the same as received)	10H
- Data address (the same as received)	two bytes: H, L
- Number of words written (the same as received)	two bytes: H, L
- CRC	two bytes: L, H

1.2. - ADDRESS MAP.

The addressing mode for CP200X is rellocatable, as it uses a base register to get the actual register address. This base register is situated at the top of the map in absolute address 0000H, and is named BASE_ADD. The actual address for a variable is formed summing the content of this position to the offset address stated in the following tables. For instance, the address for the Network ID, (offset value = 205H), will be [BASE_ADD] + 205H. This applies not only to any variable to be read or written, but also to the BAS_ADD itself. Then it can be accessed directly through the absolute address 0000H, or using the same convention, through the [BASE_ADD] + 0000H. Any value can be programmed for the BASE_ADD, from 0 to 61440D, - 0 to 0F000H -, both inclusive. In this way, the map can be situated in any physical position, circumventing the problems we have found with some PLC's which are not able to address positions higher than 9999D.

In order to be free of problems sometimes found in certain PLC's regarding to register MODBUS definitions, two different reading and writing commands are accepted. Both 03H and 04H commands are valid for reading a value, and 06H and 10H commands are valid for writing. Command 06H can be used for writing 16 bit wide variables (WORD type), and command 10H can be used to write any single variable.

The addressing scheme fulfils totally the MODBUS requirements for registers length. The MODBUS protocol considers each register as a 16 bit word. Then, the difference between two consecutive address numbers is two bytes. If the variable is four bytes long, the difference will be four bytes, or two digits.

The variables used in the CP200x are explained as follows.

1.2.1. – CALIBRATION VARIABLES.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
400	VR100	LONG	R	NO	YES	NO
402	VS100	LONG	R	NO	YES	NO
404	VT100	LONG	R	NO	YES	NO
406	PR100	LONG	R	NO	YES	NO
408	PS100	LONG	R	NO	YES	NO
410	PT100	LONG	R	NO	YES	NO
412	QR100	LONG	R	NO	YES	NO
414	QS100	LONG	R	NO	YES	NO
416	QT100	LONG	R	NO	YES	NO
418	IR100	LONG	R	NO	YES	NO
420	IS100	LONG	R	NO	YES	NO
422	IT100	LONG	R	NO	YES	NO
424	IR0	BYTE	R	NO	YES	NO
425	VR0	BYTE	R	NO	YES	NO
426	IS0	BYTE	R	NO	YES	NO
427	VS0	BYTE	R	NO	YES	NO
428	IT0	BYTE	R	NO	YES	NO
429	VT0	BYTE	R	NO	YES	NO

These variables are used for hardware calibration. The customer must not use them, as they affect to the full scale input range. Prior to be changed, the access code must be sent. If code is not sent, when asked, the CP200x will answer the calibration value stored previously.

1.2.2. – CONFIGURATION VARIABLES.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
1	ESCALAV	IEEE	R/W	YES	NO	NO
5	ESCALAI	IEEE	R/W	YES	NO	NO
9	REF_ENER	IEEE	R/W	YES	NO	NO
19	OUT_LOW0	IEEE	R/W	YES	NO	NO
21	OUT_HIGH0	IEEE	R/W	YES	NO	NO
23	VAR_LOW0	IEEE	R/W	YES	NO	NO
25	VAR_HIGH0	IEEE	R/W	YES	NO	NO
31	OUT_LOW1	IEEE	R/W	YES	NO	NO
33	OUT_HIGH1	IEEE	R/W	YES	NO	NO
35	VAR_LOW1	IEEE	R/W	YES	NO	NO
37	VAR_HIGH1	IEEE	R/W	YES	NO	NO
43						
45						
47						
49						
15						
17						
75	FREC_NOM	IEEE	R/W	YES	NO	NO
27	OUT_MED0	IEEE	R/W	YES	NO	NO
29	VAR_MED0	IEEE	R/W	YES	NO	NO
39	OUT_MED1	IEEE	R/W	YES	NO	NO
41	VAR_MED1	IEEE	R/W	YES	NO	NO
51						
53						

These variables are used to configure the device, programming in it:

ESCALAV: Means the primary nominal voltage value. If voltage transformers are not used, this value corresponds to the hardware nominal voltage.

ESCALAI: Means the primary nominal current value. If current transformers are not used, this value corresponds to the hardware nominal current, (i.e., 1 or 5 A).

REF_ENER: Means the energy value to produce an energy pulse by the digital output. This value affects equally to the Active or Reactive energy. Its value can be any, but is recommended to use a multiple of ten, related with the power nominal value, in order to not generate a very high or very low pulse rate. (For instance, if primary power was 200 Kw, a possible value for this variable could be 1Kwh, which gives 200 pulses by hour. A value of .1 Kwh gives a very high rate. A value of 10 Kwh is also adequate).

OUT_LOWx, OUT_HIGHx : They mean the lowest and highest output values. The output must be chosen among the different ranges (voltage or current, and 1-5-20 mA, or 1-5-10 V). Inside this range, the beginning, and ending values can be programmed. For instance, when 5 mA is selected, the lowest output value can be programmed to be –2,5 mA, and the highest value to +2,5 mA. These values correspond to X0 and X2 in the commercial description, and must be written in absolute values, that is, for instance, 2 mA in floating point format.

VAR_LOWx, VAR_HIGHx : They mean the lowest and highest input values, associated to the output. They correspond to Y0 and Y2 in the commercial description. Any value inside the nominal values is allowed, but if the input range is reduced, the input precision is reduced too. Then, it is not

recommended to choose a value for Y2-Y0 lower than half scale. These values are written in percentage, also in floating point format, for instance, 80.00 for 80%.

OUT_MEDx, are the middle points for kinked characteristic output. It must be located between the OUT_LOW and OUT_HIGH values.

VAR_MEDx are the variable values corresponding to the middle point kinked output. They must be located also between the VAR_LOW and VAR_HIGH values, but they must not be negative. The value corresponds to the percentage of the nominal output value.

The selection between linear and kinked output is programmed by means of register CONF_OUT (see point 6.2.4.-). The bits used are bit 0 for output 1, and bit 4 for output 2.

FREC_NOM: It means the base frequency for analog output and alarm function. As the device can be operated at any frequency, this value defines only the base point around which the alarm and output characteristic will be calculated.

1.2.3. – OUTPUT HARDWARE CALIBRATION VARIABLES.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
430	DAC0_0_I20	WORD	R/W	NO	YES	NO
431	DAC0_I20	WORD	R/W	NO	YES	NO
432						
433						
434						
435						
436	DAC1_0_I20	WORD	R/W	NO	YES	NO
437	DAC1_I20	WORD	R/W	NO	YES	NO
438						
439						
440						
441						
442						
443						
444						
445						
446						
447						

These variables are used to calibrate the different output ranges. The transducer is equipped with mA or V outputs, and then, only three different scales are used. Hardware adjustments are nor made by means of potentiometers, but by software commands, and storing their values in EEPROM. To change these values, the access code must be sent first. The user must not try to change these values.

1.2.4. – ANALOG OUTPUT AND ALARM CONFIGURATION.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
217	CONF_OUT	WORD	R/W	YES	NO	NO
224						
225						
218	CONF_DAC0	BYTE	R/W	YES	NO	NO
219	CONF_DAC1	BYTE	R/W	YES	NO	NO
220						
213						
214						
222	CONF_HARD	BYTE	R/W	YES	NO	NO
223	CONF_MANUAL	BYTE	R/W	YES	NO	NO

These variables are used for configuring the analog outputs mode.

CONF_OUT: It is a double byte variable and it is used as follows:

Bit 7	Volt/Current 2	1 : Voltage output
Bit 6	DAC Adjust 2	1 : Allows output adjust
Bit 5	Disable output 2	1: Reset output to 0
Bit 4	Kinked output 2	1: Kinked
Bit 3	Volt/current 1	1: Voltage output
Bit 2	DAC Adjust 1	1 : Allows output adjust
Bit 1	Disable output 1	1: Reset output to 0
Bit 0	Kinked output 1	1: Kinked

Bits Volt/current tell the microprocessor if the output is a voltage or a current value. This is needed as the V/I selection is done by switches, and the microprocessor must know the mode and its range. The range selected is written in the position CONF_HARD.

Bits DAC Adjust allow to change the HW settings for the D/A converters. If they are set to 1, the variables DACx_x_x can be modified. This belongs to the calibration process, and then, these bits must not be changed by the user.

Bits Disable output set each output to 0 value.

Bits Kinked select the kinked output for each one.

Byte CONF_DACx:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	X	X	X	X	X

This byte defines which variable is associated to the analog output, as the table:

Value written in CONF_DAC (XX)	Variable
0H	Vfr
1H	Vfs
2H	Vft
3H	Vrs
4H	Vst
5H	Vtr
6H	Pfr
7H	Pfs
8H	Pft
9H	Qfr
AH	Qfs
BH	Qft
CH	Ifr
DH	Ifs
EH	Ift
FH	Sfr

10H	Sfs
11H	Sft
12H	Cosr
13H	Coss
14H	Cost
15H	Prst
16H	Qrst
17H	Srst
18H	Cos
19H	Sinr
1AH	Sins
1BH	Sint
1CH	Sin
1DH	Freq

Byte CONF_MANUAL:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	-	X	X

This byte defines the manual mode for any analog output. If set to 1, the value sent through the output is the value represented by the AN_OVERx variable. This allows to simulate the output to check if the signal is being sent to the receiver.

CONF_HARD:

This byte allows to define the output range, according to the bits Ox1, Ox0:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	O21	O20	O11	O10

Ox1	Ox0	Range
1	1	-
1	0	1V or 1 mA
0	1	5 V or 5 mA
0	0	10 V or 20 mA

1.2.5. – READING VARIABLES.

These variables can be read at any moment. Its meaning is self explanatory. The variable SEQUENCE when =0, indicates the correct connection of the three voltage phases. Any value different from zero indicates wrong sequence. The sequence test is done only after a reset, and is intended as help for the installation. It is visualized also by means of the Enable LED, which in the case of wrong sequence, flashes during about three seconds.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R
120	VFR	IEEE	R	YES	NO	YES
122	VFS	IEEE	R	YES	NO	YES
124	VFT	IEEE	R	YES	NO	YES
126	VRS	IEEE	R	YES	NO	YES
128	VST	IEEE	R	YES	NO	YES
130	VTR	IEEE	R	YES	NO	YES
132	PFR	IEEE	R	YES	NO	YES
134	PFS	IEEE	R	YES	NO	YES
136	PFT	IEEE	R	YES	NO	YES
138	QFR	IEEE	R	YES	NO	YES
140	QFS	IEEE	R	YES	NO	YES
142	QFT	IEEE	R	YES	NO	YES
144	IFR	IEEE	R	YES	NO	YES
146	IFS	IEEE	R	YES	NO	YES
148	IFT	IEEE	R	YES	NO	YES
150	SFR	IEEE	R	YES	NO	YES
152	SFS	IEEE	R	YES	NO	YES
154	SFT	IEEE	R	YES	NO	YES
156	COSR	IEEE	R	YES	NO	YES
158	COSS	IEEE	R	YES	NO	YES
160	COST	IEEE	R	YES	NO	YES
162	PRST	IEEE	R	YES	NO	YES
164	QRST	IEEE	R	YES	NO	YES
166	SRST	IEEE	R	YES	NO	YES
168	COSENO	IEEE	R	YES	NO	YES
170	FREC_RED	IEEE	R	YES	NO	YES
172	SENR	IEEE	R	YES	NO	YES
174	SENS	IEEE	R	YES	NO	YES
176	SENT	IEEE	R	YES	NO	YES
178	SENO	IEEE	R	YES	NO	YES
216	SEQUENCE	BYTE	R	YES	NO	NO

1.2.6. – IDENTIFICATION VARIABLES.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
200	SER.NUMBER	STRING 10	R	NO	YES	NO
205	ID	BYTE	R/W	YES	NO	NO
206	TIPO	STRING 6	R	YES	NO	NO

SERIAL NUMBER is a string of ten characters, specifying the number stored during manufacturing.

ID represents the identity number. It is a binary number, and its value can be from 0 to 255. Identity 199 must be reserved, and can not be used, as it is a generic address. This means that any device will answer to this number, as it was theirs. Special care must be take when the transducers are connected forming a network, as every write command will be understood by all of them.

1.2.7. – CONTROL VARIABLES.

OFFSET (DEC)	VARIABLE (NAME)	TYPE	R/W	USER	CODE	BLOCK R/W
209	CONFIG_DIG	BYTE	R/W	YES	NO	NO
210	OUT_DIG	BYTE	R/W	YES	NO	NO
454	CODE_ACC	BYTE	W	YES	NO	NO
211	TIPO_PROT	BYTE	R/W	YES	NO	NO
212	PROG_VEL	BYTE	R/W	YES	NO	NO
13	ESCALAP	IEEE	R	YES	NO	NO
67	AN_OVER0	IEEE	R/W	YES	NO	NO
69	AN_OVER1	IEEE	R/W	YES	NO	NO

OUT_DIG:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	RL1	RL0

Writing a 1 in the position marked sets the corresponding relay. A 0 resets the relay.

CONFIG_DIG: This variable allows to select the working mode of the digital outputs, according to the value written.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	0	X	X

0 0 Pulse output mode
0 1 SW controlled mode

CODE_ACC: The access code must be written in this position in order to be able of modifying the values affected by it.

TIPO_PROT:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	-	-	X

This is a one bit only variable. When =1 means MODBUS. When =0, means JBUS.

A special remark must be done on this subject. The difference is only in the order in which floating point format data are sent. In mode JBUS, data is sent as

S+EXP Mantissa H Mantissa M Mantissa L

In mode MODBUS data is sent as

Mantissa M Mantissa L S+EXP Mantissa H

ESCALAP: This read only variable express the nominal power. Its value allows to confirm if the current and voltage programming has been correctly done. Its answer will be equal to

ESCALAV * ESCALAI * √3

AN_OVERx: This variable allows to produce an analog value in its respective output. The format is IEEE, and its range is from –100% to +100%. The actual output depends on the output range previously defined, that is 1, 5, or 20 mA, or 1, or 10 V. NOTE: When 4-20 mA has been selected, is possible to get an analog output outside the 4-20 mA range.

PROG_VEL defines the baud rate for the serial communication line, according to:

Value	Speed (Bps)
0	9600
1	300
2	600
3	1200
4	2400
5	4800
6	19200

1.3. - COMMANDS.

1.3.1. – READING COMMANDS.

Any reading command consists of

ID---Command type---Address---Number of words---CRCL---CRCH

ID is the identity number. Command type accepted are 03H or 04H. Address corresponds to the summation of Base Reg. content plus the offset. Number of words is the number of registers to be read, and has two byte length.. If only one byte is needed, also the number of words written is 1. Every data is accessible by a specific command to its respective address. However, the habitual reading values can be accessed by means of block commands, with the following limitations:

The values read must be all IEEE variables.

The total reading must not exceed 12 variables.

In the following examples is assumed the Base Reg. content is 1000D (3E8H), and the protocol is MODBUS (“Big-endian ” mode).

1.3.2. – EXAMPLES OF READING COMMANDS.

Serial number reading.

Q.:	01H	04H	04H	0B0H	00H	05H	30H	0DEH	
			----------		----------				
	ID	CMD	ADDRESS		No. WORDS		CRCL	CRCH	
A.:	01H	04H	0AH		53H	41H	43H	49H	31H	30H
	ID	CMD	NºBYTES		S	A	C	I	1	0
	31H	32H	35H	41H	0BEH	0F7H				
	1	2	5	A	CRCL	CRCH				

Nominal Voltage. (ESCALAV)

Q.:	01H	04H	03H	0E9H	00H	02H	0A0H	07BH
	ID	CMD	ADDRESS		No.WORDS		CRCL	CRCH
A.:	01H	04H	04H		0	0	43H	0C8H	0CBH	22H
	ID	CMD	No.BYTES		----------	400 V	-------	CRCL	CRCH	

Multiple reading

Q.: 01H 04H 04H 60H 00H 18H 0E0H 48H

ID CMD ADDRESS NO.WORDS CRCL CRCH

A.: 01H 04H 30H 5H 0H 43H 5EH

ID CMD N.BY ---------- Vfr=222 V ---------

2BH 0H 43H 5EH 0AH 0H 43H 5EH

----------Vfs=222,2 V ---- ----------Vft=222 V -------------

55H 0H 43H 0C0H 57H 0H 43H 0C0H

---------- Vrs= 384,7------- ---------Vst=384,7----------

47H 0H 43H 0C0H 0BEH 0H 44H 31H

----------Vtr= 384,6--------- ----------Pfr= 710,9 --------------

0AAH 0H 44H 31H 9BH 0H 44H 31H

------------Pfs= 710,6 ------- ----------Pft= 710,4 --------------

0ABH 0H 0C2H 86H 68H 0H 0C2H 89H

--------- Qfr= - 67,3 ---------- ---------Qfs= -68,7 --------------

2H 0H 0C2H 80H 13H 0EEH

----------Qft= -64,0 -------- CRCL CRCH

1.3.3. – WRITING COMMANDS.

The writing commands consists of
ID---Command type---Address---Number of words---Number of bytes---CRCL---CRCH

ID is the identity number. Command type is 06H or 10H. Number of words is two byte long. Number of bytes is one byte long.
No block commands are allowed. Each variable must be written by a single command.

1.3.4. – EXAMPLES OF WRITING COMMANDS.

Writing of AN_OVER0

Q.:	01H	10H	4H	2BH	00H	02H	04H
	ID	CMD	ADDRESS		No.WORDS		No.BYTES
	0H	0H	42H	48H	0B2H	52H	
	---------- VALUE ----------				CRCL	CRCH	

A.: Acknowledge frame

01H	10H	4H	2BH	00H	02H	30H	0F0H
ID	CMD	ADDRESS		NO.WORDS		CRCL	CRCH

1.4. DATA FORMATS.

The following data formats are used in the data transmission from the CP3000.

ASCII: As characters, for serial number, etc. They are sent in the order specified.

BYTE: Eight bits. For status or control parameters.

WORD: Two bytes. They are sent as MSB-LSB.

LONG: Four bytes. They are sent as MSB-___-___-LSB.

IEEE: Four bytes. They are sent as S+EXP-Mh-Mm-ML. (See annex).

ANNEX 1. CRC ALGORITHM TYPE "CRC16".

1. - GENERATOR POLYNOMIAL.
El polynomial used is:

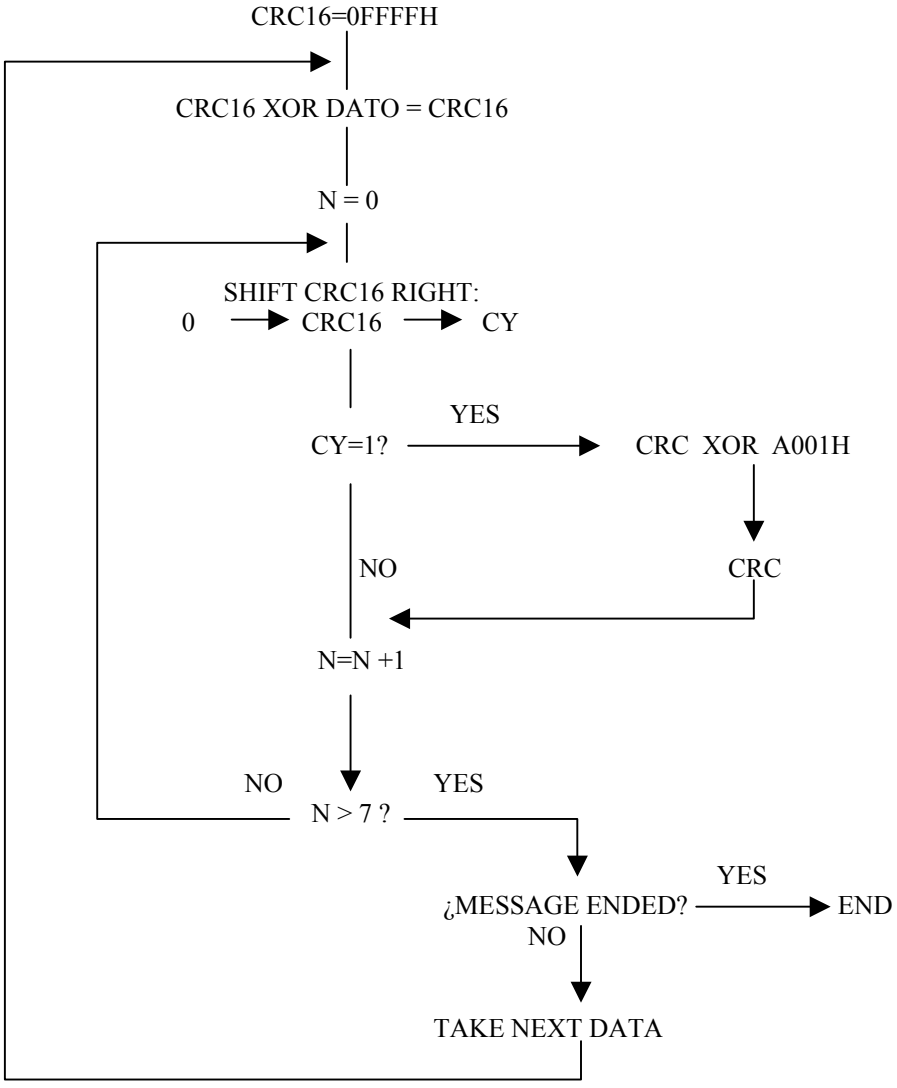
$$X^{16} + X^{15} + X^{2} + 1 = 18005H$$

To get the CRC, this polynomial must be inverted, omitting the least significative bit.

CRC16 POLYNOMIAL: 1 1000 0000 0000 0101 = 18005H

WORKING POLYNOMIAL : 1010 0000 0000 0001 = A001H.

2. - ALGORITMH.



"DATA" is the byte received/or to be transmitted.
"CRC16" is a 16 bits word. The result is left in CRC16. If the received CRC characters are included in the algorithm, the final result will be zero.

ANNEX 2.IEEE NOTATION USED. (IEEE 754).

Notation IEEE754 is followed in floating point numbers. As extremely high precision is not required, the mantissa least significative byte is always zero. This could produce some disagreements between for instance, the value written and read for a high number, as 220,000 V, which can be read as 219,987.

BYTE1	BYTE2	BYTE3	BYTE4
SIGN+	MANTISSA	MANTISSA	MANTISSA
EXPONENT	HIGH	MEDIUM	LOW

SIGN: sign of the number.
0H means a positive number.
1H means a negative number.

MANTISSA: FRACTION 0, XXX XXXX XXXX XXXX XXXX XXXX

M1 M23

EXPONENT: Number exponent, with offset 127.
0: 127. (7FH)
1: 128. (80H)
-1: 126. (7EH)

To find the value:

$$VALUE: \quad (-1)^{S} * 2^{EXP-127} * (1+ FRACTION)$$

$$FRACTION: \quad \sum_{i=1}^{i=23} 2^{-i} * M(i)$$

THE BYTE CONFIGURATION IS AS FOLLOW:

BYTE1:

7	6	5	4	3	2	1	0
SIGN	-------		EXPONENT			----------	
	E7	E6	E5	E4	E3	E2	E1

BYTE2:

7	6	5	4	3	2	1	0
EXP	---------		MANTISSA HIGH			--------	
E0	M1	M2	M3	M4	M5	M6	M7

BYTE3:

7	6	5	4	3	2	1	0
----------			MANTISSA MIDDLE			-------	
M8	M9	M10	M11	M12	M13	M14	M15

BYTE4: (Always null).

7	6	5	4	3	2	1	0
----------			MANTISSA LOW			--------	
M16	M17	M18	M19	M20	M21	M22	M23

ANNEX 3. IEEE FLOATING POINT AND LONG INTEGERS DATA TRANSMISSION FORMAT.

IEEE data are sent in the following order:

1. SIGN + EXPONENT BYTE
2. HIGH MANTISSA
3. MEDIUM MANTISSA
4. LOW MANTISSA (ALWAYS ZERO)

This mode of transmission is referred as JBUS mode.

In certain applications, data is required in the following order:

1. MEDIUM MANTISSA
2. LOW MANTISSA (ALWAYS ZERO)
3. SIGN + EXPONENT BYTE
4. HIGH MANTISSA

This mode of transmission is referred as MODBUS mode.

Both formats are supported in the CP3000 protocol. The standard way is the JBUS mode.

This is applicable also to the long integer format. They are sent as MSB.....LSB in JBUS mode, and as LSB.....MSB in MODBUS mode.

To select the protocol type, the one byte variable, named TIPO_PROT, must be used.

00H select the mode to JBUS.

01H selects the mode to MODBUS

ANNEX 4.CRC CALCULATION.

Example of CRC calculation in BASIC

```
function crc16 (txt, lon) AS INTEGER
        DIM flag AS LONG
        DIM crc AS LONG
        DIM car AS INTEGER
        DIM bit AS INTEGER
        CRC= &HFFFF&
        FOR car =1 TO LON
                crc = crc XOR ASC(MID$ ( txt, car, 1))
                FOR bit = 0 TO 7
                        flag = crc AND 1&
                        crc = crc\ 2&
                        IF flag = 1 THEN
                                crc= crc XOR &HA001&
                END IF
        NEXT bit
        NEXT car
        crc16 = INT (crc AND &HFFFF&)
END FUNCTION
```

En C:

```
        Void Saci_CalculoCRC ( unsigned char *Mensaje, int NumeroDeElementos)
{
        long flag, crcx;
        int car,bit;
        unsigned char v1,v2;
        crcx= 0xffff;
        for (car=0; car < NumeroDeElementos; car++)
        {
                crcx = crcx ^ Mensaje[car];
                for (bit=0; bit <8; bit++)
                {
                        flag= crcx & 1
                        crcx = crcx >>1 ;
                        if ( flag== 1 ) { crcx = crcx ^ 0xa001; }
                }
        }
        crcx= crcx & 0xffff;
        v1 = ( unsigned char)  abs (crcx / 256 );
        v2 = ( unsigned char ) crcx – (v1*256);
        Mensaje [NumeroDeElementos] = v2;
        Mensaje [NumeroDeElementos+1] = v1;
```

To check the message, calculate the CRC, including the CRC bytes received. If the result is zero, the message has been correctly received.