GTEC UPS MODEL:

AP160N – ZP120N - MATRIX

ModBus Protocol
CMC General register maps

SERVICE DOCUMENT

Contents

➢GENERAL MESSAGE FORMAT

(1) RTU Transmission Mode

When devices communicate on a MODBUS serial line using the RTU (Remote Terminal Unit) mode, each 8–bit byte in a message contains two 4–bit hexadecimal characters.

•RTU Message Frame

Address	Function	Data	CRC
1 byte	1 byte	0 up to 2×252 byte(s)	2 bytes CRC Low | CRC Hi

Note：The maximum size of a MODBUS RTU frame is 256 bytes.

➢FUNCTION DESCRIPTION

🖉Read Function 0x03

• Master request message：8 bytes

◎RTU mode

Slave Address	Function	Starting Address		No. of Registers		CRC
		Hi	Lo	Hi	Lo	
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes CRC Low | CRC Hi

• Slave response message：

◎RTU mode

Slave Address	Function	Byte count	Data		Next data	CRC
			Hi	Lo		
2 chars	2 chars	2 chars	2 chars	2 chars	……	2 bytes CRC Low | CRC Hi

🖉Write Single Register Function 0x06

• Master request message：8 bytes

◎RTU mode

Slave Address	Function	Register Address		Register Value		CRC
		Hi	Lo	Hi	Lo	
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes CRC Low | CRC Hi

• Slave response message：

◎RTU mode

Slave Address	Function	Register Address		Register Value		CRC	
		Hi	Lo	Hi	Lo		
1 byte	1 byte	1 byte	1 byte	1 byte	1 byte	2 bytes	
						CRC Low	CRC Hi

•Data Addresses in PPC Modbus Messages：

All data addresses in PPC Modbus messages are referenced to zero. The first occurrence of a data item is addressed as item number zero. For example: The register known as ‘MeasurementData 1’ in a programmable controller is addressed as register 0000 in the data address field of a PPC Modbus message. Register 127 decimal is addressed as register 007E hex (126 decimal). The function code field already specifies a ‘holding register’ operation. Therefore the holding register 108 is addressed as register 006B hex (107 decimal).

•Error：

Error code	2 chars	0x83
Exception	2 chars	01 or 02 or 03 or 04

➢For error example：

The master addresses a query to slave device 1 (01 hex). The function code (03 hex) is for a Read Holding Registers operation. It requests the data of the holding registers at address (0250 hex). If the Holding Registers address is non–existent in the slave device, the slave will return the exception response with the exception code shown (02 hex). This specifies an illegal data address for the slave. For example, if the slave is a 2–7 with 8 holding registers, this code would be returned.

QUERY

Byte	Contents	Example
1.	Slave Address	01
2.	Function	03
3.	Starting Address Hi	02
4.	Starting Address Lo	50
5.	No. of Registers Hi	00
6.	No. of Registers Lo	03
7.	Error Check **(CRC)**	

EXCEPTION RESPONSE

Byte	Contents	Example
1.	Slave Address	01
2.	Function	83
3.	Exception Code	02
4.	Error Check **(CRC)**	

MODBUS Exception Codes

Code	Name	Meaning
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is non-configured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, a request with offset 96 and length 4 will be success, a request with offset 96 and length 5 will generate exception 02.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.

➢For read function (0x03) example：

The master query is a Read Holding Registers request to slave device address 01. The message requests data from two holding registers, 0x2AB through 0x2AC. Note that the message specifies the starting register address as 0683 (02AB hex).

The slave response echoes the function code, indicating this is a normal response. The ‘Byte Count’ field specifies how many 8–bit data items are being returned.

It shows the count of 8–bit bytes to follow in the data.

How to Use the Byte Count Field: When you construct responses in buffers, use a Byte Count value that equals the count of 8–bit bytes in your message data. The value is exclusive of all other field contents, including the Byte Count Field.

Master Query

QUERY

Field Name	Example	ASCII Characters	RTU 8-Bit Field
Header		: (colon)	None
Slave Address	01	0 1	0000 0001
Function	03	0 3	0000 0011
Starting Address Hi	02	0 2	0000 0010
Starting Address Lo	AB	A B	1010 1011
No. of Registers Hi	00	0 0	0000 0000
No. of Registers Lo	02	0 2	0000 0010
Error Check (LRC / CRC)		4 D (2 chars)	1011 0100 (16 bits) 0101 0011
Trailer		CR LF	None
	Total Bytes :	17	8

◎RTU mode

Slave Address	Function	Starting Address		No. of Registers		CRC	
		Hi	Lo	Hi	Lo		
0x01	0x03	0x02	0xAB	0x00	0x02	0xb4	0x53

Slave Response

RESPONSE

Field Name	Example	ASCII Characters	RTU 8-Bit Field
Header		: (colon)	None
Slave Address	01	0 1	0000 0001
Function	03	0 3	0000 0011
Byte Count	04	0 4	0000 0110
Data Hi	23	2 3	0010 0011
Data Lo	05	0 5	0000 0101
Data Hi	23	2 3	0010 0110
Data Lo	00	0 0	0000 0000
Error Check (LRC / CRC)		A D (2 chars)	1111 1000 (16 bits) 1000 0110
Trailer		CR LR	None
	Total Bytes :	19	9

◎RTU mode

Slave Address	Function	Byte count	Data		Data		CRC	
			Hi	Lo	Hi	Lo		
0x01	0x03	0x04	0x23	0x05	0x23	0x00	0xf8	0x86

➢For write single register function (0x06) example：

The master query is a Writing Single Holding Registers request to slave device address 01. The message requested preset value which is specified in the data field into single holding registers 0193 (00C0 hex).

The normal response is an echo of the request, returned after the register contents have been written.

Master Query

QUERY Field Name	Example	ASCII Characters	RTU 8-Bit Field
Header		: (colon)	None
Slave Address	01	0 1	0000 0001
Function	06	0 6	0000 0110
Register Address Hi	00	0 0	0000 0000
Register Address Lo	C0	C 0	1100 0000
Register value Hi	09	0 9	0000 1001
Register value Lo	60	6 0	0110 0000
Error Check (LRC / CRC)		D 0 (2 chars)	1000 1111 (16 bits) 1000 1110
Trailer		CR LF	None
	Total Bytes :	17	8

◎RTU mode

Slave Address	Function	Register Address		Register Value		CRC	
		Hi	Lo	Hi	Lo		
0x01	0x06	0x00	0xC0	0x09	0x60	0x8f	0x8e

Slave Response

QUERY

Field Name	Example	ASCII Characters	RTU 8-Bit Field
Header		: (colon)	None
Slave Address	01	0 1	0000 0001
Function	06	0 6	0000 0110
Register Address Hi	00	0 0	0000 0000
Register Address Lo	C0	C 0	1100 0000
Register value Hi	09	0 9	0000 1001
Register value Lo	60	6 0	0110 0000
Error Check (LRC / CRC)		D 0 (2 chars)	1000 1111 (16 bits) 1000 1110
Trailer		CR LF	None
	Total Bytes :	17	8

◎RTU mode

Slave Address	Function	Register Address		Register Value		CRC	
		Hi	Lo	Hi	Lo		
0x01	0x06	0x00	0xC0	0x09	0x60	0x8f	0x8e

➢GENERAL TABLE DATA AREA DEFINITION

Data	Based address index	[1]Length in Word	Format	Information	MODBUS/JBUS Functions
IDENTIFIERS	0x00	18	WORD	See related chapter	3 (read)
STATES	0x30	2	BIT	32 States	3 (read)
FAUTL CODE[2]	0x40	9	BIT	144Fault codes	3 (read)
WARNINGS[2]	0x50	6	BIT	96Warnings	3 (read)
MEASUREMENTS	0x60	48	WORD	48Measurements	3 (read)
CONFIGURATIONS	0xA0	16	WORD	16Configurations	3 (read)
COMMAND	0xC0	1	WORD	1 Command	3(read)、6(write)
ERD Data	0xE0	32	WORD	32Values	3(read)

Note：[1]Effective range of the No. Of Registers must be less than the length in word.
[2]Fault code and warnings data are only in support of 3C3 series.

➢VECTOR DATA AREA DETAIL

⌦ DENTIFIERS FRAME

Base address index = 0x00

WORD 0-14	WORD 15	WORD 16	WORD 17
UPS name	Slave address	Protocol number	UPS firmware Version

Description	Unit	Data representation	Explanation
UPS name	*char	###.....###	ASCII characters
Slave address (UPS ID)	Integer	##	
Protocol number	Integer	##.##	For example：03.30
UPS firmware Version	Integer	##.##	For example：09.90

▪ General UPS name

Byte No.	Description
Byte0-24	UPS name will form into sequence.
Byte25	Input source “ “(Empty)：means only one input source “ 2 “ ：means two input sources. (Bypass utility source and line utility source). If get this character (“2”), PC side need add to send the BPS command then get bypass utility data.
Byte26	Always empty.
Byte27	Input phase number.
Byte28	Always “/”.
Byte29	Output phase number.

Note：* UPS name are represented by character. A character with an ASCII code less than ASCII 32(space) or greater than ASCII 123 (“z”), is not valid.

If mater requests data in this field，the LRC check regard one character as one byte.

▪ Gripower UPS name

UPS Name: TTTTTTTTTTTTTTTTTTTTTTTTTTTTTT (30 Char)

E.g.

T29	T28	T27	T26	T25	T24	T23	T22	T21	T20
3	A	3		U	P	S			
T19	T18	T17	T16	T15	T14	T13	T12	T11	T10
T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

T29-T0: UPS name

⌦ STATES DATA AREA

Base address index = 0x30

Code	Description (UPS status / System mode)	Explanation
S00	0：Bat silence(only for Centralion)	•UPS Status is composed of <S00> to <S07>. •UPS System Mode is composed of <S00> to <S07> and <S16> to <S19> •<S00>……. <S07> is binary number “0” or “1”. For Centralion <S00>is defined as Bat silence (0), besides it is reserved (always 0).
S01	1：Shutdown active/Shutdown mode	
S02	1：Test in process/BatTest mode	
S03	0：Online UPS 1：Offline UPS	
S04	1：UPS failed/Fault mode	
S05	1：Bypass/Boost active/Bypass mode	
S06	1：Battery low warning	
S07	1：Utility fail (immediate)/Battery mode	
S08	1：Idle	•Status of Battery Test is composed of <S08> to <S15>. •<S08>……. <S15> is binary number “0” or “1”.
S09	1：Processing	
S10	1：Result：no failure	
S11	1：Result：failure/warning	
S12	1：Not possible or inhibit	
S13	1：Test cancel	
S14	1：Reserved	
S15	1：Other values	
S16	1：Standby mode	•<S16>……. <S19> is binary number “0” or “1”.
S17	1：Line mode	
S18	1：Converter mode	
S19	1：ECO mode	
S20-31	Reserved	

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
S07							S00	**Byte 0**
S15							S08	**Byte 1**
S23							S16	**Byte 2**
S31							S24	**Byte 3**

Example of data sequence:

WORD 0		WORD 1	
High	Low	High	Low
S15.....S08	S07.....S00	S31.....S24	S23.....S16

⌦ FAULT CODE AREA

Base address index = 0x40

Code	Description	故障表
F00	Temperature in converter too high	轉換器過溫
F01	Internal fault 34	內部故障
F02	Incorrect parameter(Inverter MLFB)	錯誤參數
F03	Inverter contactor defective	輸出電磁接觸器損壞
F04	Multiple inverter cut off as a result of over-current	變流器過電流切斷
F05	Failure power supply electronic	工作電源不正常
F06	Over-voltage in the intermediate circuit	內部電路過電壓
F07	External Quick Shutdown active	Not used
F08	Electronic defective(Signal Processor defective)	控制器損壞
F09	Defect in the Power Electronic(Rectifier)	整流器電力元件損壞
F10	Defect in the Power Electronic(UCE-supervision)	電力元件損壞
F11	Over current cutoff	過電流切斷
F12	False parameter input during (H/W Init.)	錯誤參數輸入
F13	UPS output out of tolerance	輸出超出規格
F14	Output overload(i2t-supervision)	輸出過載
F15	Not used	Not used
F16	Not used	Not used
F17	Bypass defective	旁路損壞
F18	Bypass defective(during Transfer)	旁路轉換期間損壞
F19	Bypass overload	旁路過載
F20	Not used	Not used
F21	Not used	Not used
F22	Electronic defective(EEPROM Inverter)	控制器損壞
F23	Communication with Battery Manager defective	電池管理通訊異常
F24	Electronic defective(Check-sum EPROM)	控制器損壞
F25	Environment temperature smaller than 0 grad or measurement defective	量測溫度異常
F26	Optional module failed or not put in	外加模組訊號異常
F27	Parallel Bypass failed	並聯旁路失敗
F28	Signal fault in the Parallel Module	併聯版異常
F29	Not used	Not used
F30	Not used	Not used
F31	Not used	Not used
F32	Not used	Not used
F33	Not used	Not used
F34	Not used	Not used
F35	Not used	Not used

F36	Not used	Not used
F37	Not used	Not used
F38	Not used	Not used
F39	Not used	Not used
F40	Not used	Not used
F41	Not used	Not used
F42	Not used	Not used
F43	Not used	Not used
F44	Not used	Not used
F45	Charger over charging	電池電壓過充
F46	Not used	Not used
F47	Not used	Not used
F48	Not used	Not used
F49	Not used	Not used
F50	Not used	Not used
F51	Not used	Not used
F52	Not used	Not used
F53	Not used	Not used
F54	Not used	Not used
F55	Reserved for led test(no fault)	Not used
F56	Not used	Not used
F57	Psdr to bus fuse open	功率板與BUS間Fuse損壞
F58	Load unbalance over 50%	負載不平衡超過50%
F59	Not used	Not used
F60	Not used	Not used
F61	Profibus fault	(PPC不會發生)
F62	System frequency out of tolerance	系統頻率超出容忍　(PPC不會發生)
F63	System voltage out of tolerance	系統電壓超出容忍　(PPC不會發生)
F64	Not used	Not used
F65	Not used	Not used
F66	LCD with Cudsmc Communication failure	內部通訊異常
F67-F79	Reserved	
F80	Reserved	Not used
F81	cBusSoftTimeOut	BUS软启动超时
F82	cBusOver	BUS高压Fault
F83	cBusUnder	BUS低压Fault
F84	cBusUnbalance	BUS不平衡Fault
F85	cBusShort	Bus短路故障
F86	cInvSoftTimeOut	逆变软启动超时
F87	cInvVoltHigh	逆变电压高压Fault
F88	cInvVoltLow	逆变电压低压Fault
F89	cOPVoltShort	输出电压短路
F90	cRInvVoltShort	R相逆变电压短路
F91	cSInvVoltShort	S相逆变电压短路
F92	cTInvVoltShort	T相逆变电压短路

F93	cRSInvVoltShort	RS相线电压短路
F94	cSTInvVoltShort	ST相线电压短路
F95	cTRInvVoltShort	TR相线电压短路
F96	cInvNegPow	负功Fault
F97	cInvRNegPow	R相负功Fault
F98	cInvSNegPow	S相负功Fault
F99	cInvTNegPow	T相负功Fault
F100	cTotalInvNegPow	三相总负功Fault
F101	cReactPowFault	不均流Fault
F102	cInvRlyOpenFault	INV Rly 无法闭合
F103	cInvRlyStickFault	INV Rly 粘死
F104	cLineSCRFault	市电输入SCR故障
F105	cBatScrFault	电池输入SCR故障
F106	cByPassScrFault	旁路输入SCR故障
F107	cWiringFault	输入输出接线错误
F108	cCommLineLoss	通讯线未连
F109	cHostlineFault	主机线故障
F110	cCanFault	CAN通讯线故障
F111	cSynLineFault	同步信号线故障
F112	cAllFansLockedFault	所有风扇全故障
F113	cOCCoreFault	DSP异常
F114	cChgOpSoftTimeOut	充电器输出软启动超时
F115	cUpsAllFault	UPS模块全故障
F116	cLineInNtcOpenFault	UPS市电输入NTC开路故障
F117	cLineInFuseOpenFault	市电输入fuse开路故障
F118	cCoherencyFault	输入不一致故障
F119	cEepromFault	Eeprom数据丢失
F120	cLinesupportFail	市电支援失效
F121	cPowerBreakDown	电源失效
F122	cSysOverCapacity	系统过容
F123	cADS7869Fault	ADS7869 Error
F124	cSTSHardwareFault	Bypass Mode No OP
F125	cOpBreakerOffFault	O/P Breaker OFF in Parallel Mode
F126	cNTCAbnormal	NTC故障
F127	cBatAbnormal	电池故障
F128	cWC1_ForbidComeBackFromByp	频繁过流故障
F129	cEpoFault	EPO故障(最高优先级故障，此故障一旦置位就不能被其它故障码覆盖)
F130-F143	Not used	Not used

Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**	
F07							F00	**Byte 0**
F15							F08	**Byte 1**
F23							F16	**Byte 2**
F31							F24	**Byte 3**
F39							F32	**Byte 4**

F47							F40	**Byte 5**
F55							F48	**Byte 6**
F63							F56	**Byte 7**
F71							F64	**Byte 8**
F79							F72	**Byte 9**
F87							F80	**Byte 10**
F95							F88	**Byte 11**
F103							F96	**Byte 12**
F111							F104	**Byte 13**
F119							F112	**Byte 14**
F127							F120	**Byte 15**
F135							F128	**Byte 16**
F143							F136	**Byte 17**

Example of data sequence:

WORD 0		WORD 1		.....	WORD 8	
High	Low	High	Low		High	Low
F15.....F08	F07.....F00	F31.....F24	F23.....F16		F143.....F136	F135.....F128

⌦ WARNING CODE AREA

Base address index = 0x50

Code	Description	故障表
A00	Over-temperature inverter	變流器過溫
A01	Bypass mains failure	市電異常
A02	Rectifier mains failure	市電異常
A03	Load too high(i2-t-Supervision)	過載
A04	Under-voltage intermediate circuit	電路低電壓
A05	Over load	過載
A06	Phase sequence incorrect in Bypass	輸入相序錯誤
A07	Service bypass is on	Not used
A08	Battery operation	電池供電
A09	Battery rest time exceeded	供電時間太少
A10	Battery under-voltage	電池低電壓
A11	Operating condition commissioning, converter	執行設定模式
A12	Battery switch not engaged	電池未接
A13	Ventilator lifetime exceeded	建議更換風扇
A14	Connection to charger lost	充電器未接
A15	Internal warning 16	內部警告
A16 | A21	Reserved	Reserved
A22	Battery charger communicate failure	充電器無通訊
A23	General battery charger failure	充電器異常

A24 | A29	Reserved	Reserved
A30	Load unbalance	負載不平衡
A31	Internal warning 32	內部警告
A32	cWC1_EpoActive	EPO开关未接
A33	cWC1_ModuleUnLock	模块未锁
A34	cWC1_IPNLoss	输入中线丢失
A35	cWC1_SiteFail	L、N反接
A36	cWC1_BypassLoss	旁路异常
A37	cWC1_ByPassPhaseErr	旁路相序错误
A38	cWC1_OverChg	电池过充
A39	cWC1_BatReverse	电池反接
A40	cWC1_MaintainOn	维修旁路盖板打开
A41	cWC1_ ErrorLocation	物理位置错误
A42	cWC1H_cTurnOnAbornaml	不满足开机条件，无法开机
A43	cWC1H_cRedundantLoss	冗余丢失
A44	cWC1_ModuleHotSwapActive	模块未插紧
A45	cWC1_BatteryInform	电池维护时间到
A46	cWC1_InspectionInform	巡检维护时间到
A47	cWC1_GuaranteeInform	过保维护时间到
A48	cWC1_TempLow	温度过低
A49	cWC1_BatOverTemp	电池过温
A50	cWC1_BusCapMainInform	BUS电容维护时间到
A51	cWC1_SysOverCapacity	系统过容
A52	cWC1_HighExternalWarning	外部告警高32位
A53-A63	reserved	保留
A64	cOuterWarningCode2L_ModuleFault	模块故障
A65	cOuterWarningCode2L_BypassNLoss	旁路N线丢失
A66	cOuterWarningCode2L_BatNLoss	电池N线断开
A67	cOuterWarningCode2L_ExChgFail	EXCHG故障
A68	cOuterWarningCode2L_BatTempLow	电池温度过低
A69	cOuterWarningCode2L_BatUnbalance	电池不平衡
A70	cOuterWarningCode2L_ADCalibrationFail	AD校准失败
A71-A95	Reserved	保留

Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**	
A07							A00	**Byte 0**
A15							A08	**Byte 1**
A23							A16	**Byte 2**
A31							A24	**Byte 3**
A39							A32	**Byte 4**
A47							A40	**Byte 5**
A55							A48	**Byte 6**
A63							A56	**Byte 7**
A71							A64	**Byte 8**
A79							A72	**Byte 9**
A87							A80	**Byte 10**
A95							A88	**Byte 11**

Example of data sequence:

WORD 0		WORD 1		.....	WORD 5	
High	Low	High	Low		High	Low
A15.....A08	A07.....A00	A31.....A24	A23.....A16		A95.....A88	A87.....A80

⌦ MEASUREMENT DATA AREA

Base address index = 0x60

Address Index	**Description**	**Unit**	**Data representation**	**Explanation**
0x60	I/P voltage R	V	###.#	
0x61	I/P voltage S	V	###.#	
0x62	I/P voltage T	V	###.#	
0x63	I/P fault voltage (R or S or T)	V	###.#	
0x64	Bypass I/P voltage R	V	###.#	
0x65	Bypass I/P voltage S	V	###.#	
0x66	Bypass I/P voltage T	V	###.#	
0x67	I/P frequency	Hz	##.#	
0x68	Bypass I/P frequency	Hz	##.#	
0x69	O/P voltage R	V	###.#	
0x6A	O/P voltage S	V	###.#	
0x6B	O/P voltage T	V	###.#	
0x6C	O/P current R	%(A)	###	This is a percent of maximum current for old UPS. This is an absolute value for new UPS type
0x6D	O/P current S	%(A)	###	
0x6E	O/P current T	%(A)	###	
0x6F	O/P current R	A	###.#	
0x70	O/P current S	A	###.#	
0x71	O/P current T	A	###.#	
0x72	O/P power R	KW	###.#	
0x73	O/P power S	KW	###.#	

0x74	O/P power T	KW	###.#	
0x75	Total power	KW	###.#	
0x76	O/P complex power R	KVA	###.#	
0x77	O/P complex power S	KVA	###.#	
0x78	O/P complex power T	KVA	###.#	
0x79	Total complex power	KVA	###.#	
0x7A	O/P load	%	###	W% is a percent of maximum real power. VA% is a percent of maximum VA.
0x7B	Battery cell voltage	V	#.##	
0x7C	Battery voltage/ Positive battery voltage	V	###.#	
0x7D	Temperature	℃	##.#	
0x7E	Charge in Status/ Battery capacity percentage	%	###	Provided range for the program： CCC=000.....to.....133
0x7F	Recharge time to 95%	hours	###	
0x80	Estimated run-time/ Remain battery backup time	min	#####	##### in seconds.
0x81		sec		
0x82	Hold time at 100% Load	min	###	###in minutes.
0x83		sec	.##	.## in seconds.
0x84	Hold time at 50% Load	min	###	###in minutes.
0x85		sec	.##	.## in seconds.
0x86	O/P frequency	HZ	##.#	
0x87	Negative battery voltage	V	###.#	
0x88 ｜ 0x8F	Reserved			

⌦ CONFIGUREATION DATA AREA

Base address index = 0xA0

Address Index	Description	Unit	Data representation	Explanation
0xA0	Nominal mains voltage	V	###	
0xA1	Nominal mains frequency	Hz	##.##	
0xA2	Nominal cell voltage	V	##.##	
0xA3	Maximum mains voltage	V	###	
0xA4	Minimum mains voltage	V	###	
0xA5	Maximum mains frequency	Hz	##.##	
0xA6	Minimum mains frequency	Hz	##.##	
0xA7	Maximum mains ambient temperature	℃	##	
0xA8	Minimum mains ambient temperature	℃	##	
0xA9	Battery cell number	Integer	#	
0xAA	Number of battery cells in series	Integer	###	
0xAB	Input transformer type Y or Delta	Boolean	Y	•Y=1, Input transformer is Y type, LCD display phase voltage. •Y=0, Input transformer is Delta type, LCD

				display Line voltage.
0xAC	LCD display output voltage Line or Phase	Boolean	O	•O=1, LCD display output voltage is Phase. •O=0, LCD display output voltage is Line.
0xAD | 0xAF	Reserved			

⌦ COOMMAND DATA CODE

Base address index = 0xC0

Address Index	Description	Unit	Data representation	Explanation
0xC0	Getting / Setting baud rate	Hz	This value can be only-- 1200 / 2400 / 4800 / 9600 / 19200	•Using function code 0x03 can get baud rate at present. •Using function code 0x06 can set baud rate at present.

⌦ ERD DATA AREA

Base address index = 0xE0

Address Index	Description	Unit	Data representation	Explanation
0xE0	Input Voltage	V	###	
0xE1	Input Current	A	###	
0xE2	Output Voltage	V	###	
0xE3	Output Current	A	###	
0xE4	Output Frequency	Hz	##.#	
0xE5	Output Watt	KW	##	## in integer
0xE6			.###	.### in decimal
0xE7	Output VA	KVA	##	## in integer
0xE8			.###	.### in decimal
0xE9	Total Battery Voltage	V	###	
0xEA	Battery discharge current	A	###	
0xEB	Temperature	℃	##.#	
0xEC	Load Percent	%	###	
0xED	UPS States	Bit		UPS States is composed of <S00> to <S07>
0xEE | 0xEF	Reserved			
0xF0	Load Type	A	###	
0xF1	Load Value or Current Value	KW or KA	##	## in integer
0xF2			.###	.### in decimal
0xF3	Battery Low Value	V	###	
0xF4	Battery High Value	V	###	
0xF5	System Permit Low Voltage	V	###	
0xF6	System Permit High Voltage	V	###	

0xF7	System Permit Low Frequency	Hz	###	
0xF8	System Permit High Frequency	Hz	###	
0xF9	Battery Capacity	%	###	
0xFA | 0xFF	Reserved			

➢ CMC Connection

✎ General CMC

1. The two pins of JP17 should be short as one pin together via a jumper part, the pins of JP18 should be also short as one pin via a jumper part.

2. Let the pin3(R+) of CMC RJ45 connect the pin (T/R+) of Converter, and pin4(R-) of CMC RJ45 connect the pin (T/R-) of Converter. Or let the pin6(R+) of CMC RJ45 connect the pin (T/R+) of Converter, and pin5(R-) of CMC RJ45 connect the pin (T/R-) of Converter.

Notice: Please use the RS232/RS485 Converter that can’t receive its echo data, and make sure that the pins connection between CMC and the converter is correct.

➢CRC check

The RTU mode includes an error–checking field that is based on a Cyclical Redundancy Checking (CRC) method performed on the message contents. The CRC field checks the contents of the entire message. It is applied regardless of any parity checking method used for the individual characters of the message.

The CRC field contains a 16–bit value implemented as two 8–bit bytes. The CRC field is appended to the message as the last field in the message. When this is done, the low–order byte of the field is appended first, followed by the high–order byte. The CRC high–order byte is the last byte to be sent in the message.

The CRC is started by first preloading a 16–bit register to all 1's. Then a process begins of applying successive 8–bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits and the parity bit, do not apply to the CRC.

During generation of the CRC, each 8–bit character is exclusive ORed with the register contents. Then the result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position. The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value. If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next 8–bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final content of the register, after all the characters of the message have been applied, is the CRC value.

A procedure for generating a CRC is:

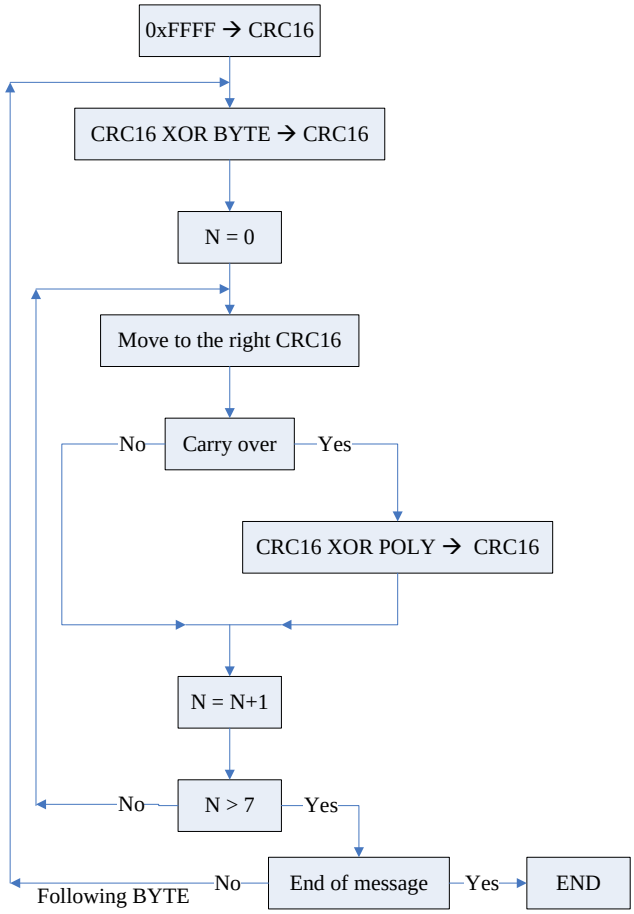
1. Load a 16–bit register with FFFF hex (all 1's). Call this the CRC register.
2. Exclusive OR the first 8–bit byte of the message with the low–order byte of the 16–bit CRC register, putting the result in the CRC register .
3. Shift the CRC register one bit to the right (toward the LSB), zero–filling the MSB. Extract and examine the LSB.
4. (If the LSB was 0): Repeat Step 3 (another shift).
   (If the LSB was 1): Exclusive OR the CRC register with the polynomial value 0xA001 (1010 0000 0000 0001).
5. Repeat Steps 3 and 4 until 8 shifts have been performed. When this is done, a complete 8–bit byte will have been processed.
6. Repeat Steps 2 through 5 for the next 8–bit byte of the message. Continue doing this until all bytes have been processed.
7. The final content of the CRC register is the CRC value.
8. When the CRC is placed into the message, its upper and lower bytes must be swapped as described below.

• Placing the CRC into the Message

When the 16–bit CRC (two 8–bit bytes) is transmitted in the message, the low-order byte will be transmitted first, followed by the high-order byte. For example, if the CRC value is 1241 hex (0001 0010 0100 0001)：

Addr.	Func.	Data Count	Data	Data	Data	Data	CRC Lo	LRC Hi
							0x41	0x12

• Calculation algorithm of the CRC 16



XOR = exclusive or
N = number of information bits
POLY = calculation polynomial of the CRC 16 = 1010 0000 0000 0001
(Generating polynomial = $1 + X_2 + X_{15} + X_{16}$)
In the CRC16, the 1st byte transmitted Is the least significant one

CRC register initialization		1111	1111	1111	1111
XOR 1st character		0000	0000	0000	0000
		1111	1111	1111	1101
	Move1	0111	1111	1111	1110 | 1
		1010	0000	0000	0001
Flag to 1, XOR polynomial		1101	1111	1111	1111
	Move2	0110	1111	1111	1111 | 1
Flag to 1, XOR polynomial		1010	0000	0000	0001
		1100	1111	1111	1110
	Move3	0110	0111	1111	1110 | 0
	Move4	0011	0011	1111	1111 | 1
		1010	0000	0000	0001
		1001	0011	1111	1110
	Move5	0100	1001	1111	1111 | 0
	Move6	0010	0100	1111	1111 | 1
		1010	0000	0000	0001
		1000	0100	1111	1110
	Move7	0100	0010	0111	1111 | 0
	Move8	0010	0001	0011	1111 | 0
		1010	0000	0000	0001
		1000	0001	0011	1110
		0000	0000	0000	0111
XOR 2nd character		1000	0001	0011	1001
	Move1	0100	0000	1001	1100 | 1
		1010	0000	0000	0001
		1110	0000	1001	1101
	Move2	0111	0000	0100	1110 | 1
		1010	0000	0000	0001
		1101	0000	0100	1111
	Move3	0110	1000	0010	0111 | 1
		1010	0000	0000	0001
		1100	1000	0010	0110
	Move4	0110	0100	0001	0011 | 1
	Move5	0011	0010	0000	1001 | 1
		1010	0000	0000	0001
		1001	0010	0000	1000
	Move6	0100	1001	0000	0100 | 0
	Move7	0010	0100	1000	0010 | 0
	Move8	0001	0010	0100	0001 | 0
			Most significant		Least significant

Example: an example of a C language function performing CRC generation is shown below.
The function takes two arguments:
unsigned char *puchMsg; A pointer to the message buffer containing binary data to be used for generating the CRC
unsigned short usDataLen; The quantity of bytes in the message buffer.

▪ CRC Generation Function

```
unsigned short CRC16 ( puchMsg, usDataLen ) /* The function returns the CRC as a unsigned short type */
unsigned char *puchMsg ; /* message to calculate CRC upon */
unsigned short usDataLen ; /* quantity of bytes in message */
{
    unsigned char uchCRCHi = 0xFF ; /* high byte of CRC initialized */
```

```
    unsigned char uchCRCLo = 0xFF ; /* low byte of CRC initialized */
    unsigned uIndex ; /* will index into CRC lookup table */
    while (usDataLen--) /* pass through message buffer */
    {
        uIndex = uchCRCLo ^ *puchMsgg++ ; /* calculate the CRC */
        uchCRCLo = uchCRCHi ^ auchCRCHi[uIndex] ;
        uchCRCHi = auchCRCLo[uIndex] ;
    }
    return (uchCRCHi << 8 | uchCRCLo) ;
}
```

High-Order Byte Table

```
/* Table of CRC values for high–order byte */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01,
0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01,
0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81,
0x40
} ;
```

Low-Order Byte Table

```
/* Table of CRC values for low–order byte */
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7, 0x05, 0xC5, 0xC4,
```

```
0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD,
0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32, 0x36, 0xF6, 0xF7,
0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE,
0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1, 0x63, 0xA3, 0xA2,
0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB,
0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0, 0x50, 0x90, 0x91,
0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88,
0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80,
0x40
};
```