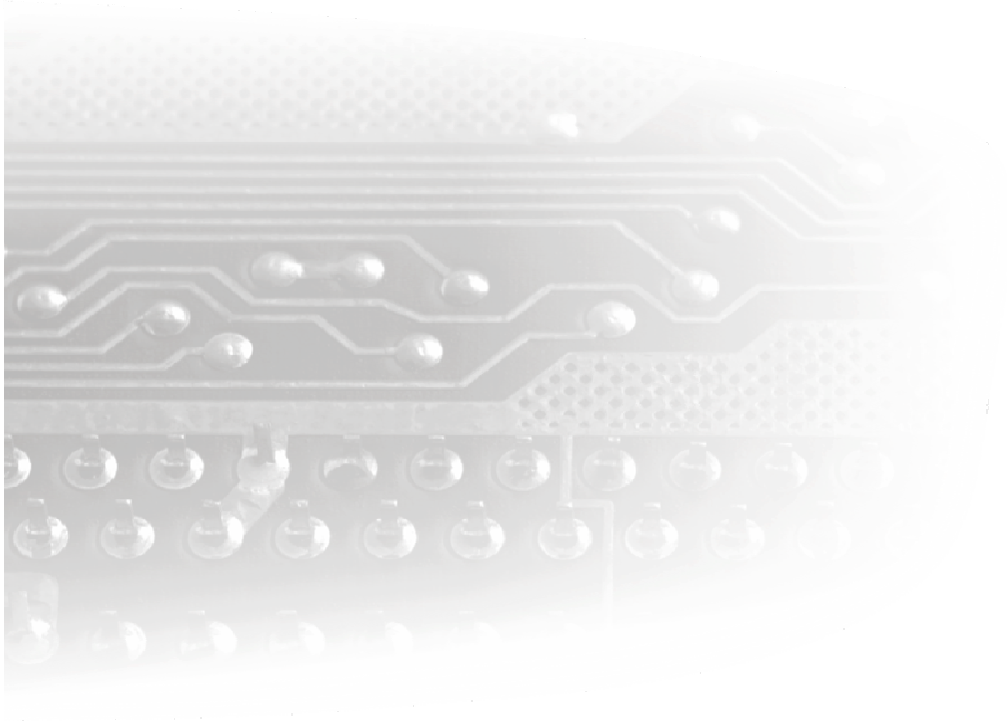
The OPEN CNC -

CNC46XX series CNC system

Development manual



Shenzhen to hing technology co., LTD

Address: shenzhen nanshan garden of the arts Tian Sha IC industrial park road 27-29 5 floor, zip code: 518052

Telephone: 0755-26722719 fax: 0755-26722718

E-mail：Adtech//21cn.comhttp://www.adtechcn.com

Version upgrade instructions

Item no.	first time toupload	version number	tal number of pages	Write an engineer	DTP engineers

Check record

date	Version/pages	results	confirm

Remark:

Shenzhen to xing technology co., LTD. Has this manual on the strict collating and check it carefully, but we cannot guarantee that this manual no errors and omissions.

Shenzhen to xing technology co., LTD., is committed to continuously improve the product function, improve the quality of service, thus keeping any products and software programs described in this manual as well as the contents of this manual to make changes without further notice in advance.

directory

The first paper CNC open source is introduced

- CNC table using unified way of addressing to resources calls and transfer.
- macro address is used to access in the G code, the type of macro variables default is 32 bit floating point registers, but some macro address points to the other types of variables, the other the value of the variable will be converted to 32-bit floating point values to G code for processing, should pay attention to when the value of this.
  - PLC&ModBus PLC address is used for internal or external devices through the Modbus visit the address of the operator, the PLC or the external devices are accessed through the address NC all resources; The address of the default points to the 16-bit integer registers, if point to a floating point or 32-bit integer data, will take up the address of two consecutive number to piece together into a 32-bit number. For example,
  - address R5042 represents the feed speed, for example, but the feed speed is a 32-bit integer data, then read back the occupied R5042, R5043 two 16-bit address to express a 32-bit integer;
  - R5042 represents the low 16 bits of data, 5043 represent high 16 bit data；
- if directly read R5043, is because the data are aligned and read failure, or returns null.
  - system variable types and definition of access:
  - INT8U unsigned 08bit integer data
  - INT8S signed 16bit integer data
  - INT16U unsigned 16bit integer data
  - INT16S signed 16bit integer data

- INT32U unsigned 32bit integer data
- INT32S signed 32bit integer data
- FP32 32 bit floating point data（32-bit IEEE-754 ）
- （W） Have write permissions（Write）
- （R） Have read permissions（Read）
- （RW） Have can read can write permissions（Read or Write）
- * Special type

1. CNCSpecial function register

Acer address	**PLCAddress**	**access Permissions**	**Variable types**	**Functional Description**
	500	Write	*special (W)	file access command register
	501	R/W	*special (RW)	file access data register
	502	Read	*special (R)	file access data length register
	503	R/W	*special (RW)	file access data position register
	504	R/W	*special (RW)	file access Error and statusregister
	505 ～ 519			reserve
	520	Write	*special (W)	Parameter Management command register
	521	Read	*special (R)	Parameter Managementdata register
	522	Read	*special (R)	Parameter Managementdata length register

	523	R/W	*special (RW)	Parameter Managementdata positionregister
	524	R/W	*special (RW)	Parameter ManagementError and statusregister
	525 ～ 539			reserve
	540	Write	*special (W)	Multi-register access command register
	541	R/W	*special (RW)	Multi-register access data register
	542	Read	*special (R)	Multi-register access data length register
	543	R/W	*special (RW)	Multi-register access data positionregister
	544	R/W	*special (RW)	Multi-register access Error and statusregister
	545 ～ 999			reserve

Note : CNCspecial operation register of access follow the same protocol Modbus format , in the access data, requires sequential access Multi- a register to complete a full operation. So this area is more suitable for high-level language programming register with complex logic operations using , for the PLC and touch screen access there may be some difficulties .

2. CNC run control bit register

CNC run control bit register is used to display the current status of the controller and the IO output port can be controlled to control other operations.

register List

Acer address	PLCAddress	access Permissions	Functional Description
	R0 ～ R499	Read	bit Address corresponding input port: 0：IN0 1：IN1 2：IN2 3：IN3 4：IN4 5：IN5 6：IN6 7：IN7 8：IN8 9：IN9 10：IN10 11：IN11 12：IN12 13：IN13 14：IN14 15：IN15 16：IN16 17：IN17 18：IN18 19：IN19 20：IN20 21：IN21 22：IN22 23：IN23

			24：Handwheel0.1 25：HandwheelHandwheelX 26：Handwheel0.01 27：HandwheelHandwheelY 28：Handwheel0.001 29：HandwheelHandwheelZ 30：HandwheelStart 31：HandwheelHandwheelA 32：Handwheel Pause 33：Handwheel scram 34：ServoXAlarm 35：ServoYAlarm 36：ServoZAlarm 37：ServoAAlarm Other bits Addressreserve unused
	R500 ～ R999	R/W	IO -bit Address corresponding output port 500：OUT0 501：OUT1 502：OUT2 503：OUT3 504：OUT4 505：OUT5 506：OUT6 507：OUT7 508：OUT8 509：OUT9 510：OUT10 511：OUT11 512：OUT12 513：OUT13 514：OUT14 515：OUT15 516：OUT16

			517：OUT17 518：OUT18 519：OUT19 520：OUT20 521：OUT21 522：OUT22 523：OUT23 540：RUNLight（Only forCNC8860） 541：ALARMLight（Only forCNC8860） Other bits Addressreserve unused
	R1000 ～ R1279	R/W	Bit Address corresponding panel LED output port: 1000：LED0 1001：LED1 1002：LED2 1003：LED3 1004：LED4 1005：LED5 1006：LED6 1007：LED7 1008：LED8 1009：LED9 1010：LED10 1011：LED11 1012：LED12 1013：LED13 1014：LED14 1015：LED15 Other bits Addressreserve unused
	R1280 ～ R7999		reserve

3. CNC operation control word register

CNC operation control register is used to display the current operational status of the implementation status of the controller and G codes , etc. , but also can be written on the part of the Address unit control operations .

register List

Acer address	PLCAddress	access Permissions	Variable types	Functional Description
1000～1367	1000～1367	Read	INT16U	IO corresponding input0～367, example: 1000：IN0 1001：IN1 … 1367：IN367
1370～1392	1370～1392	Read	INT16U	IO corresponding input0～367, each corresponding to 16 consecutive IO Address input , example: 1370：IN15～IN0 1371：IN31～IN16 … 1392：IN367～IN352
1400～1767	1400～1767	R/W	INT16U	IO corresponding output0～367, example: 1400：OUT0 1401：OUT1 … 1767：OUT367
1770～1792	1770～1792	R/W	INT16U	IO corresponding output0～

				367，Each Address correspondence successive 16 IO output ，example： 1770：OUT15～OUT0 1771：OUT31～OUT16 … 1792：OUT367～OUT352
1800～1975	1800～1975	R/W	INT16U	Corresponding LED output0～175，example： 1800：LED0 1801：LED1 … 1975：LED175
1980～1990	1980～1990	R/W	INT16U	Corresponding LED output0～175，Address correspondence to each of 16 consecutive LED output，example： 1980：LED15～LED0 1981：LED31～LED16 … 1990：LED175～LED160
2000	R2000	Read	128 byte array	Current Alarm message content , UTF8 character encoding format .
2064	R2064	R/W	128 byte array	Custom Alarm Information0 , External Alarm state corresponding section 0
2128	R2128	R/W	128 byte array	Custom Alarm Information1 , External Alarm state corresponding section 1
2192	R2192	R/W	128 byte	Custom Alarm Information2 ,

			array	External Alarm state corresponding section 2
2256	R2256	R/W	128 byte array	Custom Alarm Information3，External Alarm state corresponding section 3
2320	R2320	R/W	128 byte array	Custom Alarm Information4，External Alarm state corresponding section 4
2384	R2384	R/W	128 byte array	Custom Alarm Information5，External Alarm state corresponding section 5
2448	R2448	R/W	128 byte array	Custom Alarm Information6，External Alarm state corresponding section 6
2512	R2512	R/W	128 byte array	Custom Alarm Information7，External Alarm state corresponding section 7
2576	R2576	R/W	128 byte array	Custom Alarm Information8，External Alarm state corresponding section 8
2640	R2640	R/W	128 byte array	Custom Alarm Information9，External Alarm state corresponding section 9
2704	R2704	R/W	128 byte array	Custom Alarm Information10，External Alarm state corresponding section 10
2768	R2768	R/W	128 byte array	Custom Alarm Information11，External Alarm state corresponding section 11
2832	R2832	R/W	128 byte array	Custom Alarm Information12，External Alarm state corresponding section 12
2896	R2896	R/W	128 byte	Custom Alarm Information13，

			array	External Alarm state corresponding section 13
2960	R2960	R/W	128 byte array	SerialAlarmInformation14 ， External Alarm state corresponding section 14
3024	R3024	R/W	128 byte array	Warning bitAlarmInformation15， External Alarm state corresponding section 15
3088	R3088	Read	240 byte array	Tips Information stored breakpoint processing content , format UTF8 character encoding format .
3208	R3208	Read	128 byte array	The current G code to runInformation0
3272	R3272	Read	128 byte array	The current G code to runInformation1
3336	R3336	Read	128 byte array	The current G code to runInformation2
3400	R3400	Read	128 byte array	The current G code to runInformation3
3464	R3464	Read	128 byte array	The current G code to runInformation4
3528 ～ 3897	R3528～R3897			reserve
3898	R3898	Read	INT16U	Signal state mapping system functionsregister 0：Safety signals 1：Pressure signal 2：clip feed signal 3：The system pump 4：The knife cytometry 5: AlarmLight 6：Running Light 7：Lubrication output 8：Cooling output

				9: Spindle CW output 10: Reverse output shaft 11 : The system detects oil 12 : Spindle alarm detection 13 : Frequency alarm detection 14: on the knife blow Output 15: Detection limit on the knife
3899	R3899	Read	INT16U	Signal state mapping system functionsregister 0：External scram2 1：ExternalStart 2 2：External Pause2 3：Handwheel0.1 4：Handwheel0.01 5：Handwheel0.001 6：HandwheelHandwheelX 7：HandwheelHandwheelY 8：HandwheelHandwheelZ 9：HandwheelHandwheelA 10：Handwheel scram 11：Handwheel Pause 12：HandwheelStart
3900	R3900	Read	INT32U	External workpiece number
3902	R3902	Read	INT32U	The maximum number of workpiece machining
3904	R3904	Read	INT16U	The current system of internal keys
3905	R3905	Write	INT16U	System External buttons response (Note ①)
3906	R3906	Read	INT8U	The current system control mode 0 : Input

				1Automatic 2：Manual 3：Single -step / Handwheel 4：Zero
3907	R3907	R/W	INT8U	Interpolation magnification
3908	R3908	R/W	INT8U	Rapid traverse override
3909	R3909	R/W	INT8U	Spindle override
3910	R3910	R/W	FP32	Manual override
3912	R3912	Read	INT32U	Programming rate
3914	R3914	Read	INT32U	actual rate
3916	R3916	Read	INT32U	Manually rate
3918	R3918	Read	INT32S	Spindle status
3920	R3920	Read	INT16U	BIT15 ~ BIT8: Load state in loading stopped to load file; 0：Load Stop 1：Load Ready 2：Loading BIT7 ~ BIT0: loading progress , value range (0 to 100) .
3921	R3921	Read	INT16U	Processing file program number
3922	R3922	R/W	16 byte array	R: The current processing file name ; W: Loads the specified file as the current processing file. All machining program stored in a fixed directory controller.
3930	R3930	Read	INT32U	System Alarm No. (Note ②) When Alarm generation , the corresponding Alarm content is available to get through 1500Address, character encoding is

				UTF8 format.
3932	R3932	Read	INT16U	BIT15: single-stage status 0 : Continuous 1: single-stage BIT14 ~ BIT5: reserve BIT4 ~ BIT0: running 0 : Stop 1: Run 2: Pause 3 : single Pause
3933	R3933	R/W	INT16U	BIT15: single-stage status 0 : Continuous 1: single-stage BIT14 ~ BIT5: reserve BIT4 ~ BIT0: running 0 : Stop 1: Run 2: Pause 3 : single Pause
3934	R3934	Read	INT8U	HandwheelHandwheel 0 : not selected axis 1: X -axis 2: Y -axis 3: Z -axis 4: A shaft
3935	R3935	Read	FP32	Handwheel gear (ie Handwheel single-step incremental value)
3937	R3937	R/W	INT16U	BIT15 (read-only) : whether the

				breakpoint memory execution condition (0 - do not conform , 1 in line) ; BIT14 (Read Only) : Is able to perform breakpoint selection function (0- not , 1 can) ; BIT13 ~ BIT2: reserve BIT1 (read and write) : Determines BIT0 control effective (0 - inactive , 1- valid) ; BIT0 (read and write): BIT1 valid for 1:00 , (0 - do not run from the breakpoint , 1 run from the breakpoint) . Tips Information breakpoint execution can print their own narrative , can read the contents of Address2588register group print display 。
3938	R3938	R/W	INT16U	Processing mode 0 : The local file processing mode ; 1 : Online processing mode.
3939	R3939	W	INT16U	Online processing operations command
3940	R3941	R/W	INT16U	When performing online processing, storage and processing data written to the cache used . Reading : The remaining memory blocks ; Write: Write any value to empty the cache

3941	R3941	Write	240 byte array	Write online processing data register, a data up to 240 bytes. Wherein the first byte is a sequence number for each write to a data of the number plus 1 ; the second byte data length; the rest of the content data . That data frame format is: NO + length N + N data points
3942	R3942	Read	INT32U	The current code is running position (DNC processing use only)
3944	R3944	Read	INT16U	Zero zero mark and the current status : high eight zero mark in turn corresponds to the axis , 0 indicates no successful zero, a zero indicates success too ; lower 8 sequentially corresponding axis zero state 0 zero operation is completed , 1 shows the action being executed zero
3945 ～ 3999	R3945～R3999			reserve
4000	R4000	Read	INT16U	G CODE0modal value
4001	R4001	Read	INT16U	G CODE1modal value
4002	R4002	Read	INT16U	G CODE2modal value
4003	R4003	Read	INT16U	G CODE3modal value
4004	R4004	Read	INT16U	G CODE4modal value
4005	R4005	Read	INT16U	G CODE5modal value
4006	R4006	Read	INT16U	G CODE6modal value
4007	R4007	Read	INT16U	G CODE7modal value
4008	R4008	Read	INT16U	G CODE8modal value
4009	R4009	Read	INT16U	G CODE9modal value

4010	R4010	Read	INT16U	G CODE10modal value
4011 ～ 4105	R4011～R4105			reserve
4106	R4106	Read	INT32U	Tool radius compensation number D
4108	R4108	Read	INT32U	Feed rate F
4110	R4110	Read	INT32U	Tool length compensation No. H
4112	R4112	Read	INT32U	Auxiliary function M code
4114	R4114	Read	INT32U	The current program number O field No
4116 ～ 4117	R4116～R4117			reserve
4118	R4118	Read	INT32U	Spindle speed S
4120	R4120	Read	INT16U	Tool function number T
4121	R4121	RW	INT16U	T code format
4122	R4122	RW	INT32S	ExternalCoordinate compensationX
4124	R4122	RW	INT32S	ExternalCoordinate compensationY
4126	R4126	RW	INT32S	ExternalCoordinate compensationZ
4128	R4128	RW	INT32S	ExternalCoordinate compensationA
4130	R4130	RW	INT32S	ExternalCoordinate compensationB
4132	R4132	RW	INT32S	ExternalCoordinate compensationC
4133	R4133			reserve
4134	R4134	RW	INT32S	G54Workpiece coordinate systemX
4136	R4136	RW	INT32S	G54Workpiece coordinate systemY
4138	R4138	RW	INT32S	G54Workpiece coordinate systemZ
4140	R4140	RW	INT32S	G54Workpiece coordinate systemA

4142	R4142	RW	INT32S	G54Workpiece coordinate systemB
4144	R4144	RW	INT32S	G54Workpiece coordinate systemC
4146	R4146	RW	INT32S	G55Workpiece coordinate systemX
4148	R4148	RW	INT32S	G55Workpiece coordinate systemY
4150	R4150	RW	INT32S	G55Workpiece coordinate systemZ
4152	R4152	RW	INT32S	G55Workpiece coordinate systemA
4154	R4154	RW	INT32S	G55Workpiece coordinate systemB
4156	R4156	RW	INT32S	G55Workpiece coordinate systemC
4158	R4158	RW	INT32S	G56Workpiece coordinate systemX
4160	R4160	RW	INT32S	G56Workpiece coordinate systemY
4162	R4162	RW	INT32S	G56Workpiece coordinate systemZ
4164	R4164	RW	INT32S	G56Workpiece coordinate systemA
4166	R4166	RW	INT32S	G56Workpiece coordinate systemB
4168	R4168	RW	INT32S	G56Workpiece coordinate systemC
4170	R4170	RW	INT32S	G57Workpiece coordinate systemX
4172	R4172	RW	INT32S	G57Workpiece coordinate systemY
4174	R4174	RW	INT32S	G57Workpiece coordinate

				systemZ
4176	R4176	RW	INT32S	G57Workpiece coordinate systemA
4178	R4178	RW	INT32S	G57Workpiece coordinate systemB
4180	R4180	RW	INT32S	G57Workpiece coordinate systemC
4182	R4182	RW	INT32S	G58Workpiece coordinate systemX
4184	R4184	RW	INT32S	G58Workpiece coordinate systemY
4186	R4186	RW	INT32S	G58Workpiece coordinate systemZ
4188	R4188	RW	INT32S	G58Workpiece coordinate systemA
4190	R4190	RW	INT32S	G58Workpiece coordinate systemB
4192	R4192	RW	INT32S	G58Workpiece coordinate systemC
4194	R4194	RW	INT32S	G59Workpiece coordinate systemX
4196	R4196	RW	INT32S	G59Workpiece coordinate systemY
4198	R4198	RW	INT32S	G59Workpiece coordinate systemZ
4200	R4200	RW	INT32S	G59Workpiece coordinate systemA
4202	R4202	RW	INT32S	G59Workpiece coordinate systemB
4204	R4204	RW	INT32S	G591Workpiece coordinate systemC
4206	R4206	RW	INT32S	G591Workpiece coordinate systemX
4208	R4208	RW	INT32S	G591Workpiece coordinate

				systemY
4210	R4210	RW	INT32S	G591Workpiece coordinate systemZ
4212	R4212	RW	INT32S	G591Workpiece coordinate systemA
4214	R4214	RW	INT32S	G591Workpiece coordinate systemB
4216	R4216	RW	INT32S	G591Workpiece coordinate systemC
4218	R4218	RW	INT32S	G592Workpiece coordinate systemX
4220	R4220	RW	INT32S	G592Workpiece coordinate systemY
4222	R4222	RW	INT32S	G592Workpiece coordinate systemZ
4224	R4224	RW	INT32S	G592Workpiece coordinate systemA
4226	R4226	RW	INT32S	G592Workpiece coordinate systemB
4228	R4228	RW	INT32S	G592Workpiece coordinate systemC
4230	R4230	RW	INT32S	G593Workpiece coordinate systemX
4232	R4232	RW	INT32S	G593Workpiece coordinate systemY
4234	R4234	RW	INT32S	G593Workpiece coordinate systemZ
4236	R4236	RW	INT32S	G593Workpiece coordinate systemA
4238	R4238	RW	INT32S	G593Workpiece coordinate systemB
4240	R4240	RW	INT32S	G593Workpiece coordinate systemC

4242	R4242	RW	INT32S	G594Workpiece coordinate systemX
4244	R4244	RW	INT32S	G594Workpiece coordinate systemY
4246	R4246	RW	INT32S	G594Workpiece coordinate systemZ
4248	R4248	RW	INT32S	G594Workpiece coordinate systemA
4250	R4250	RW	INT32S	G594Workpiece coordinate systemB
4252	R4252	RW	INT32S	G594Workpiece coordinate systemC
4254	R4254	RW	INT32S	G595Workpiece coordinate systemX
4256	R4256	RW	INT32S	G595Workpiece coordinate systemY
4258	R4258	RW	INT32S	G595Workpiece coordinate systemZ
4260	R4260	RW	INT32S	G595Workpiece coordinate systemA
4262	R4262	RW	INT32S	G595Workpiece coordinate systemB
4264	R4264	RW	INT32S	G595Workpiece coordinate systemC
4266	R4266	RW	INT32S	G596Workpiece coordinate systemX
4268	R4268	RW	INT32S	G596Workpiece coordinate systemY
4270	R4270	RW	INT32S	G596Workpiece coordinate systemZ
4272	R4272	RW	INT32S	G596Workpiece coordinate systemA
4274	R4274	RW	INT32S	G596Workpiece coordinate systemB

4276	R4276	RW	INT32S	G596Workpiece coordinate systemC
4278	R4278	RW	INT32S	G597Workpiece coordinate systemX
4280	R4280	RW	INT32S	G597Workpiece coordinate systemY
4282	R4282	RW	INT32S	G597Workpiece coordinate systemZ
4284	R4284	RW	INT32S	G597Workpiece coordinate systemA
4286	R4286	RW	INT32S	G597Workpiece coordinate systemB
4288	R4288	RW	INT32S	G597Workpiece coordinate systemC
4290	R4290	RW	INT32S	G598Workpiece coordinate systemX
4292	R4292	RW	INT32S	G598Workpiece coordinate systemY
4294	R4294	RW	INT32S	G598Workpiece coordinate systemZ
4296	R4296	RW	INT32S	G598Workpiece coordinate systemA
4298	R4298	RW	INT32S	G598Workpiece coordinate systemB
4300	R4300	RW	INT32S	G598Workpiece coordinate systemC
4302	R4302	RW	INT32S	G599Workpiece coordinate systemX
4304	R4304	RW	INT32S	G599Workpiece coordinate systemY
4306	R4306	RW	INT32S	G599Workpiece coordinate systemZ
4308	R4308	RW	INT32S	G599Workpiece coordinate

				systemA
4310	R4310	RW	INT32S	G599Workpiece coordinate systemB
4312	R4312	RW	INT32S	G599Workpiece coordinate systemC
4314 ～ 4999	R4314～R4999			reserve
5000	R5000	Read	FP32	X -axis absolute coordinate
5002	R5002	Read	FP32	Y -axis absolute coordinate
5004	R5004	Read	FP32	Z -axis absolute coordinate
5006	R5006	Read	FP32	A -axis absolute coordinate
5008	R5008	Read	FP32	B -axis absolute coordinate
5010	R5010	Read	FP32	C -axis absolute coordinate
5012 ～ 5019	R5012～R5019			reserve
5020	R5020	Read	FP32	X -axis relative coordinates
5022	R5022	Read	FP32	Y -axis relative coordinates
5024	R5024	Read	FP32	Z -axis relative coordinates
5026	R5026	Read	FP32	A -axis relative coordinates
5028	R5028	Read	FP32	B -axis relative coordinates
5030	R5030	Read	FP32	C -axis relative coordinates
5032 ～ 5039	R5032～R5039			reserve
5040	R5040	Read	FP32	X -axis mechanicalposition
5042	R5042	Read	FP32	Y -axis mechanicalposition
5044	R5044	Read	FP32	Z -axis mechanicalposition
5046	R5046	Read	FP32	A -axis mechanicalposition
5048	R5048	Read	FP32	B -axis mechanicalposition
5050	R5050	Read	FP32	C -axis mechanicalposition
5052 ～ 5059	R5052～R5059			reserve
5060	R5060	R/W	INT32S	X axis feedbackposition
5062	R5062	R/W	INT32S	Y axis feedbackposition
5064	R5064	R/W	INT32S	Z axis feedbackposition
5066	R5066	R/W	INT32S	A axis feedbackposition
5068	R5064	R/W	INT32S	B axis feedbackposition
5070	R5066	R/W	INT32S	C axis feedbackposition

5068 ～ 5079	R5068～R5079			reserve
5080	R5080	Read	FP32	X-axis tool position compensation amount
5082	R5082	Read	FP32	Y-axis tool position compensation amount
5084	R5084	Read	FP32	Z-axis tool position compensation amount
5086	R5086	Read	FP32	A-axis tool position compensation amount
5088 ～ 6099	R5088～R6099			reserve
5100	R5100	R/W	INT16S	X-axisExternal manual Write positive : Xaxis positive direction runninge Write negative : Xaxis negative direction run Write zero value: X -axis immediately stop
5101	R5101	R/W	INT16S	Y-axisExternal manual Write positive : Yaxis positive direction runninge Write negative : Yaxis negative direction run Write zero value: Y -axis immediately stop
5102	R5102	R/W	INT16S	Z-axisExternal manual Write positive : Zaxis positive direction runninge Write negative : Zaxis negative direction run Write zero value: Z -axis immediately stop
5103	R5103	R/W	INT16S	A-axisExternal manual Write positive : Aaxis

				positive direction runninge Write negative : Aaxis negative direction run Write zero value: A -axis immediately stop
5104	R5104	R/W	INT16S	B-axisExternal manual Write positive : Baxis positive direction runninge Write negative : Baxis negative direction run Write zero value: B -axis immediately stop
5105	R5105	R/W	INT16S	C-axisExternal manual Write positive : Caxis positive direction runninge Write negative : Caxis negative direction run Write zero value: C -axis immediately stop
6100 ～ 14999	R6100～R14999			Please refer to the region register function "CNCParameter Managementregister" section
20000～22047	R20000～R22047	Read	INT16U	Directory / file list Information Output 0 ~ 255, example: 20006 ~ 20007 : BIT31: Contents 0 / file 0 distinction 0 : Directory 1: file BIT30 ~ BIT0: file 0 size (Unit : Byte) 20000 to 20005 : Directory 0 / file 0

				Name 20014 ~ 20015 : BIT31: Contents 1 / file 1 distinction 0 : Directory 1: file BIT30 ~ BIT0: file 1 Size (Unit : Byte) 20008 ~ 20013 : Contents 1 / file 1 name ... 22046 ~ 22047 : BIT31: Catalog 255 / file 255 distinction 0 : Directory 1: file BIT30 ~ BIT0: file 255 Size (Unit : Byte) 22040 ~ 22045 : Catalog 255 / file 255 name
23100～23227	R23100～R23227	R/W	INT16U	R: The current directory path W: enter the selected directory, where D is the root directory for the "\", U root directory as "usb: \"
23228	R23228	Read	INT32U	total number of directories and file in the current path
23238～23245	R23238～R23245	Write	INT16U	processing of file names to be deleted
23246～23253	R23246～R23253	Write	INT16U	processing of file names to be copied
23254～23261	R23254～R23261	Write	INT16U	ile name you want to paste

				processing
24000	R24000	Read	INT32U	Enter edit mode G code number of lines
24002	R24002	Read	INT32U	current editor window starting line number
24004	R24004	Read	INT32U	current editor window end of the line number
24006	R24006	R/W	INT32U	Edit window of the current line number from 0 to 15
24008	R24008	Write	INT16S	Editing window operation command 1 : cursor left 2 : cursor right 3 : Move the cursor 4 : cursor down 5: The current line Enter key 6 : Page Up 7 : Down 8 : Delete the current line 9 : Save the edited G
24500～24563	R24500～R24563	R/W	INT16U	G CODE EDITING CONTENT0
24564～24627	R24564～R24627	R/W	INT16U	G CODE EDITING CONTENT1
24628～24691	R24628～R24691	R/W	INT16U	G CODE EDITING CONTENT2
24692～24755	R24692～R24755	R/W	INT16U	G CODE EDITING CONTENT3
24756～24819	R24756～R24819	R/W	INT16U	G CODE EDITING CONTENT4
24820～24883	R24820～R24883	R/W	INT16U	G CODE EDITING CONTENT5
24884～24947	R24884～R24947	R/W	INT16U	G CODE EDITING CONTENT6
24948～25011	R24948～R25011	R/W	INT16U	G CODE EDITING CONTENT7
25012～25075	R25012～R25075	R/W	INT16U	G CODE EDITING CONTENT8
25076～25139	R25076～R25139	R/W	INT16U	G CODE EDITING CONTENT9
25140～25203	R25140～R25203	R/W	INT16U	G CODE EDITING CONTENT10
25204～25267	R25204～R25267	R/W	INT16U	G CODE EDITING CONTENT11
25268～25331	R25268～R25331	R/W	INT16U	G CODE EDITING CONTENT12

25332～25395	R25332～R25395	R/W	INT16U	G CODE EDITING CONTENT13
25396～25459	R25396～R25459	R/W	INT16U	G CODE EDITING CONTENT14
25460～25523	R25460～R25523	R/W	INT16U	G CODE EDITING CONTENT15
30000~30074	R30000~R30074	R/W	FP32	XStandby
30100~30174	R30100~R30174	R/W	FP32	YStandby
30200~30274	R30200~R30274	R/W	FP32	Z Tool length compensation # 30200 # 3020 ...
30300~30374	R30300~R30374	R/W	FP32	AStandby
30400~30474	R30400~R30474	R/W	FP32	BStandby
30500~30574	R30500~R30574	R/W	FP32	CStandby
30600~30999	R30600~R30999	R/W	FP32	Standby
31000~31074	R31000~R31074	R/W	FP32	Tool radius compensation # 31000 # 31002 ...
31100~31174	R31100~R31174	R/W	FP32	Lathe cutting diameter test # 31100 # 31102 ...
31200~31274	R31200~R31274	R/W	FP32	Turning diameter compensation # 31300 # 31302 ...…
31300~31374	R31300~R31374	R/W	FP32	Lathe test cut length # 31200 # 31202
31400~31474	R31400~R31474	R/W	FP32	Turning length compensation # 31400 # 31402 ...
31500~31574	R31500~R31574	R/W	FP32	Lathe R compensate # 31500 # 31502 ...
31600~31674	R31600~R31674	R/W	FP32	Lathe T compensate # 31600 # 31602 ...2…
32000~32074	R32000~R32074	R/W	FP32	XOffset...#32000 #32002…
32100~32174	R32100~R32174	R/W	FP32	YOffset...#32100 #32102…
32200~32274	R32200~R32274	R/W	FP32	ZOffset...#32200 #32202…
32300~32374	R32300~R32374	R/W	FP32	AOffset...#32300 #32302…
32400~32474	R32400~R32474	R/W	FP32	BOffset...#32400 #32402…
32500~32574	R32500~R32574	R/W	FP32	COffset...#32500 #32502…

Note ①: CNC4xxxx system External buttons key features are as follows :

Key..（Dec）46Series	**Key..（Hex）46Series**	**Key..（Dec）49Series**	**Key..（Hex）49Series**	Valid state	**Functional Description**
0	0x00	0	0x00	Any valid state	Empty Key..
139	0x8b	41	0x29	Stop state effective	Automatic [Automatic mode]
140	0x8c	42	0x2A	Stop state effective	Manual [Manual mode]
141	0x8d	45	0x2D	Stop state effective	Entry [Input Mode]
142	0x8e	46	0x2E	Stop state effective	Handwheel / single step [Handwheel / single-step mode]
143	0x8f	51	0x33	Stop state effective	Zero [Zero mode]
49	0x31	16	0x10	Under the zero mode is active	X-【XAxis zero operation】
48	0x30	23	0x17	Under the zero mode is active	Y-【YAxis zero operation】
46	0x2e	24	0x18	Under the zero mode is active	Z-【ZAxis zero operation】
45	0x2d	22	0x16	Under the zero mode is active	A-【AAxis zero operation】
167	0xa7	47	0x2F	Under the zero mode is active	Start 【 all Axis zero operation】
133	0x85	40	0x28	Start spindle effective	Right 【Speed decrement operation

					】
134	0x86	38	0x26	Start spindle effective	Left【Self- plus operating speed 】
135	0x87	44	0x2C	Automatic mode is active	Down 【 Magnification decrement operator]
136	0x88	39	0x27	Automatic mode is active	up【Self- plus operating ratio 】
49	0x33	18	0x12	Manual mode is active	X+ 【 X-axis positive movement】
51	0x31	16	0x10	Manual mode is active	X- 【 X-axis negative direction】
53	0x35	11	0x0B	Manual mode is active	Y+ 【 Y-axis positive movement】
48	0x30	23	17	Manual mode is active	Y- 【 Y-axis negative direction】
52	0x34	10	0x0A	Manual mode is active	Z+ 【 Z-axis positive movement】
46	0x2e	24	18	Manual mode is active	Z- 【 Z-axis negative direction】
54	0x36	12	0x0C	Manual mode is active	A+ 【 A-axis positive movement】
45	0x2d	22	16	Manual mode is active	A- 【 A-axis negative direction】
160	0xa0				Forward [Forward]
161	0xa1				Reverse [reversed]
162	0xa2				Cool [cool]
163	0xa3				Lubrication [Lubrication]
164	0xa4				Jigging [Jigging]
165	0xa5	49	0x31		Single-stage[single-segment

166	0xa6	48	0x30		Pause【 Pause】
167	0xa7	47	0x2F		Start 【Start 】
145	0x91	52	0x34		Reset[Reset]
		17	0x11	Manual mode is active	B+【 5 】
		29	0x1D	Manual mode is active	B-【 0 】
		28	0x1C	Manual mode is active	C+【 - 】
		30	0x1E	Manual mode is active	C-【 . 】

Note ②: No Alarm system defines as follows :

系统 AlarmNo.	Description
0x0000	No Alarm (system default)
0x0001	End of program
0x0002	Not specified motion functions
0x0003	No G-code to obtain lines of code function
0x0004	ATC failure
0x0005	The tool is invalid
0x0006	G block repeat mistakes
0x0007	Block Program number error
0x0008	G7x8x complex instruction code does not work properly
0x0009	Specify the port number error
0x000a	Abnormal program termination error
0x000b	M01 code specifies procedures Pause
0x000c	Program number specified does not exist
0x000d	M98 is malformed
0x000e	Call campaign execution failed
0x000f	This paragraph does not require compensation
0x0010	G Block invalid format
0x0011	G program rerun , M99 command , this is a normal movement Alarm Alarm, used to refresh the count
0x0012	Motion abnormalities Alarm

0x0013	Illegal characters
0x0014	mment character formatting error or no symmetry comment symbol
0x0015	Illegal G code
0x0016	G code number or value error compensation
0x0017	Undefined radius compensation G code error
0x0018	Arc programming error
0x0019	Illegal specified plane , G17, G18, G19 plane outside
0x001a	M98 calling errors that may exceed the maximum
0x001b	Spindle axis number specified hardware error
0x001c	M code execution error
0x001d	Specify spindle failure
0x001e	Movement repeat request
0x001f	Specifies the arc does not exist
0x0020	noXInstruction error
0x0021	noYInstruction error
0x0022	noZInstruction error
0x0023	noAInstruction error
0x0024	noBInstruction error
0x0025	noCInstruction error
0x0026	noDInstruction error
0x0027	noRInstruction error
0x0028	noFInstruction error
0x0029	noTInstruction error
0x002a	noSInstruction error
0x002b	noPInstruction error
0x002c	noMInstruction error
0x002d	noGInstruction error
0x002e	noIInstruction error
0x002f	noJInstruction error
0x0030	noKInstruction error
0x0031	noQInstruction error
0x0032	Repeat pitch value specified error
0x0033	An alarm system abnormal exit

0x0034	Human intervention exit
0x0035	no G code specified parameters Source
0x0036	no program number is specified G code table storing Address
0x0037	Macro function call error
0x0038	Writing abnormal macro expression
0x0039	Macro variables Address error
0x003a	Illegal variable values
0x003b	Jump statement error
0x003c	Macro loop pairing error
0x003d	Macro loop nesting error
0x003e	Macro loops nested calls over Multi-, beyond the most Multi- layers
0x003f	Get Address macro variable is not defined function
0x0040	Information output macro definition , not Alarm
0x0041	Macro definitions Alarm output
0x0042	Constant reference error
0x0043	On some tracks i_gcode property value error
0x0044	Under some tracks i_gcode property value error
0x0045	Starting compensation point arc entering
0x0046	End compensation point arc exit
0x0047	On some tracks in the compensation process starting position and end position equal
0x0048	Starting position and the end of the next paragraph trajectory equal position in the compensation process
0x0049	Compensation value is greater than the G0203 programmed radius compensation R value
0x004a	Encountered during radius compensation G code can not be switched
0x004b	NURBS nodes over Multi-
0x004c	NURBS parameter error
0x004d	Number of segments composite program memory overflow error program too Multi-
0x004e	Programs exist expressions composite error
0x004f	noUInstruction error
0x0050	noWInstruction error

0x0051	Multi- heavy G code expressions error
0x0052	Multi- weight M code expressions error
0x0400	Undefined zero
0x0401	Soft limitA-
0x0402	Soft limitA+
0x0403	Soft limitZ-
0x0404	Soft limitZ+
0x0405	Soft limitY-
0x0406	Soft limitY+
0x0407	Soft limitX-
0x0408	Soft limitX+
0x0409	Hard LimitA-
0x040a	Hard LimitA+
0x040b	Hard LimitZ-
0x040c	Hard LimitZ+
0x040d	Hard LimitY-
0x040e	Hard LimitY+
0x040f	Hard LimitX-
0x0410	Hard LimitX+
0x0411	Tight scram only
0x0412	XServoAlarm
0x0413	YServoAlarm
0x0414	ZServoAlarm
0x0415	AServoAlarm
0x0416	Axis number redefinition error
0x0417	Spindle not zero error
0x0418	Die clip unlocked error
0x0419	Safety signals are not in place to detect
0x041a	Relocatable code
0x041b	Insufficient system pressure
0x041c	System Folder signal no material effect Alarm
0x041d	Hydraulic Alarm System

0x041e	Spindle an alarm
0x041f	Drive an alarm
0x0420	Failed to put the knife
0x0421	Failure to grasp the knife
0x0422	Knife Tocumen detect errors
0x0423	Bayonet detect errors
0x0424	Pine knife detect errors
0x0425	Failed to put the knife
0x0426	Failed to put the knife
0x0427	Failed to put the knife
0x0428	Failed to put the knife
0x0429	Failed to put the knife
0x042a	Failed to put the knife
0x042b	Failed to put the knife
0x042c	Failed to put the knife
0x042d	Failed to put the knife
0x042e	Failed to put the knife
0x042f	Failed to put the knife
0x0430	On the knife Limit Alarm
0x0431	Additional panel is not working
0x0432	Pretreatment External abort program execution
0x1000	Custom macro to repeat the number of calls overflowed
0x1001	Processing code in operation to detect the coordinates programmed ultra Soft limit value set

4. CNCParameter Managementregister

- parameter register of all the variables used to store the current parameters of the CNC , these variables or parameters may be peripheral devices inside the PLC access, access can also be called the macro code .
- PLC is 6100 ~¬Address range

Parametersregister List

Acer address	PLCAddress	access Permissions	Variable types	Functional Description
6100～6299	6100 ～ 6299	R/W	FP32	Corresponding # 100 to # 199 macro variable region , the macro area after a power
6300～6499	6300 ～ 6499			reserve
6500～7499	6500 ～ 7499	R/W	FP32	# 500 to # 999 corresponding macro variable region , the macro area after a power outage , data is not lost.
7500～7999	7500 ～ 7999			reserve
8000～8499	8000 ～ 8499			Comprehensive parameter area
8500～8999	8500 ～ 8999			Axis configuration parameters
9000～9499	9000 ～ 9499			Management parameters
9500～9999	9500 ～ 9999			Spindle parameters
10000～ 10499	10000 ～ 10499			Port Configuration
10500～14999	10500 ～ 14999			reserve
30000~39999	30000~39999			length, radius , Offset ..., trial cutting , try to cut length, diameter compensation , length compensation parameters

5. PMC axis control parameter register

2. register Address60100 ~ 60299 PMC axis control used for control , including 60,100 ~ 60,199 for one axis parameter control Address District , 60200 ~ 60299 for the two-axis parameter control Address area. Each control axis can be selected from 1 to 6 axes in any one axis use , PMC axis control can not be interpolated motion , only single-axis motion control, single-axis motion does not affect the operation of any other axis . example as CNC8860 the former 4 -axis CNC programming done using 5,6 PMC axis can be used for programming.

3. Axis control parameters

register List：

Address(i：0,1)	**access Permissions**	**Variables Type**	**The default value**	**Range**	**Description**
60100+i*100	R/W	INT16U	0	0～6	Axis number 0 indicates that the axis of no use . CNC8860 controller should be set to 5,6 axis number.
60101+i*100	R/W	INT16U	9	0～15	Axis characteristics and pulse mode setting (modified restart valid) BIT3: Axis properties 0 rotation ; a linear ; BIT2: pulse direction logic 0 direction output positive logic ; One direction output negative logic ; BIT1: Pulse Logic 0 positive logic pulse ; A negative logic pulse ; BIT0: pulse mode 0 pulse + pulse ;

					A pulse + direction . Axis hardware limit mode settings BIT2: 0 positive limit no effect ; A positive limit effective ; BIT1: 0 Negative limit no effect ; A negative limit effective ; BIT0: 0 Low Stop ; A high level stops.
60102+i*100	R/W	INT16U	6	0～7	Axis characteristics and pulse mode setting (modified restart valid) BIT3: Axis properties 0 rotation ; a linear ; BIT2: pulse direction logic 0 direction output positive logic ; One direction output negative logic ; BIT1: Pulse Logic 0 positive logic pulse ; A negative logic pulse ; BIT0: pulse mode 0 pulse + pulse ; A pulse + direction . Axis hardware limit mode settings BIT2: 0 positive limit no effect ; A positive limit effective ;

					BIT1: 0 Negative limit no effect ; A negative limit effective ; BIT0: 0 Low Stop ; A high level stops.
60103+i*100	R/W	INT16U	0	0～3	Axis stop signal0 mode settings BIT1: 0 Hardware origin signal no effect ; A hardware origin valid signal. BIT0: 0 active low ; An active high ;
60104+i*100	R/W	INT16U	0	0～3	Axis stop signal1 mode settings BIT1: 0 Hardware origin signal no effect ; A hardware origin valid signal. BIT0: 0 active low ; An active high ;
60105+i*100	R/W	INT16U	4	0～7	Axis configuration BIT2: generate Alarm axis stop 0 does not stop a stop BIT1: Axis Reset active level 0 Low ; 1 high ; (Servo) BIT0: Axis Alarm active level 0 Low ; 1 high. (Servo)
60106+i*100	R/W	INT16U	11	0～31	Enable and direction of the axis zero configuration

					BIT4: 0ServoZ phase zero active low ; 1ServoZ with 0:00 active high ; BIT3: 0 ServoZ phase zero prohibited ; 1 ServoZ phase zero is enabled. BIT2: 0 External zero active low ; 1 External zero active high ; BIT1: 0 External zero prohibited ; 1 External Zero enabled ; BIT0: 0 positive direction to zero ; A negative direction zero.
60107+i*100	R/W	INT16U	i*2+20	0～69	Hardware axis positive limit input port number
60108+i*100	R/W	INT16U	i*2+21	0～69	Axis hardware negative limit input port number
60109+i*100	R/W	INT16U	0	0～69	External zero input port number
60110+i*100	R/W	INT16U	i+39	0～69	Servo axis Alarm input port number
60111+i*100	R/W	INT16U	i+23	0～23	Servo axis reset output port number
60112+i*100	R/W	INT32U	500	1～2000000	Axis initial speed
60114+i*100	R/W	INT32U	10000	1～32000	Axis acceleration
60116+i*100	R/W	INT32U	10000	1～2000000	Axis rapid traverse
60118+i*100	R/W	INT32U	5000	1～2000000	Shaft speed manual
60120+i*100	R/W	INT32U	10000	1～32000	Axis zero acceleration
60122+i*100	R/W	INT32U	20000	1～2000000	Axis zero speed
60124+i*100	R/W	INT32U	5000	1～2000000	Zero speed reducer shaft

60126+i*100	R/W	INT32U	1000	1～2000000	Zero Zero axis scanning speed
60128+i*100	R/W	INT32U	0	-2147483648 ～ 2147483647	Axis zero pulse Offset ...
60130+i*100	R/W	INT32U	20000	1～32000	Axis biggest constraint acceleration
60132+i*100	R/W	INT32S	100000	1～2000000	The maximum speed of the shaft constraint
60134+i*100	R/W	INT32S	2147483647	-2147483648 ～ 2147483647	Axis positive Soft limit the number of pulses
60136+i*100	R/W	INT32S	-2147483648	-2147483648 ～ 2147483647	Axis negative Soft limit to the number of pulses
60138+i*100	R/W	INT32S	9999	0～9999999	ROUND axis setpoint (as the rotation axis to use, modify restart valid)

Axis Control State and control register List

:

Address(i：0,1)	**access Permissions**	**Variables Type**	**Range**	**Description**
60170+i*100	R/W	INT32S	-2147483648 ～ 2147483647	Axis actual position pulses
60172+i*100	R/W	INT32S	-2147483648 ～ 2147483647	Axis logical position pulses
60174+i*100	R	INT32U	0～2000000	Axis current speed
60176+i*100	R	INT16U	0～1	Axis current state BIT0: Axis operation status 0 stopped 1 running state
60177+i*100	R	INT16U	0～63	Axis AlarmInformation: Alarm generated

				after all axis motion command will no effect until the Alarm Axis AlarmInformation: Alarm generated after all axis motion command will no effect until the fault is cleared after the Alarm , and then perform cleanup AlarmInformation instruction , axis motion control to return to normal . BIT5: negative effective to Soft limitAlarm 0 no effect 1 BIT4: Forward Soft limitAlarm 0 no effect a valid BIT3: negative effective to Hard LimitAlarm 0 no effect 1 BIT2: Forward Hard LimitAlarm 0 no effect a valid BIT1: ServoAlarm 0 no effect a valid BIT0: Axis not zero Alarm 0 no effect a valid
60178+i*100	R/W	INT16U	0～1	Axis Zero flag : when the power controller , BIT0 bit defaults to 1 , then if the other axis motion command execution except " zero" command will produce the "Axis not zero Alarm". After the successful execution of the next instruction is zero , BIT0 bit is automatically set to 0 , indicating that the shaft has been successfully zeroed abnormal running , the user can according to different anomalies of BIT0 set, the request before the next run of the axis must first zero. Also be based on the need to control BIT0 cleared , the requirements do not need to perform before the zero axis running operation .

				BIT0: Axis zero case 0 zero 1 no zero
60179+i*100	W	INT16U	0～7	Axis control command 0x00: Clear AlarmInformation; 0x01: stop immediately ; 0x02: Deceleration stop ; 0x03: positive direction manual operation , the speed set by the manual speed ; 0x04: negative direction manual operation , the speed set by the manual speed ; 0x05: the positive direction of the continuous operation , the speed set by the drive speed ; 0x06: the negative direction of the continuous operation , the speed set by the drive speed ; 0x07: Axis zero operation.
60180+i*100	W	INT32S	-2147483648 ～ 2147483647	Quantitative driving pulses , the axis in the stopped state , write a non-zero data starts to run , write the operating state data no effect.

The first chapter reviews the PC level language programming

1. PMC development of library functions Detailed

data TypeThe following statement

:

```
typedef unsigned char   INT8U;    //8Bitunsigned Integer
typedef char            INT8S;    //8Bit signed Integer
typedef unsigned short  INT16U;   //16Bitunsigned Integer
```

```
typedef short          INT16S;   //16Bit signed Integer
typedef unsigned int   INT32U;   //32Bitunsigned Integer
typedef int            INT32S;   //32Bit signed Integer
typedef float          FP32;     //32BitFloating PointInteger
typedef double         FP64;     //64BitDouble-precision
typedef int            BOOL;     //Booleandata
```

1.1. Communications Manager Functions category

```
//***************Communicationportconfiguration*********************************
#define    COMM_PORT_NUM          6     // MODBUS transmission port number (tentative 6 , UART0, UART1, UART2, UART3, TCP, UDP)


#define    COMM_INVALID_PORT      -1    //no effect port number
#define    COMM_UART_PORT0        0     //UART0
#define    COMM_UART_PORT1        1     //UART1
#define    COMM_UART_PORT2        2     //UART2
#define    COMM_UART_PORT3        3     //UART3
#define    COMM_TCP_PORT          4     //TCPNetwork Interface
#define    COMM_UDP_PORT          5     //UDPNetwork Interface

//***************SerialCommunicationsMode*************************************
**********
#define    COMM_MODE_RTU          0x00  //RTUMode
#define    COMM_MODE_ASCII        0x01  //ASCIIMode

//****************CommunicationsManagementFunctions*****************************
*******************
#define    COMM_NO_ERR            0x00  // Communication no error
#define    COMM_PORT_ERR          0x01  // Communications port number error

#define    COMM_DEV_BUSY          0x02  // Communications equipment busy

#define    COMM_PERFORM_FAIL      0x03  // Execution fails
```

1.1.1. COMM_LibVer( )

Function Name	FLOAT _stdcall COMM_LibVer(void);
Functional Description	Communication library version , you can get through the current library version of this function , easy upgrade and maintenance .
Input Parameters	no
Output parameters	no
return value	Library version number, the format X.XX
Precautions	no

example：

```
voidmain( )
{
    FLOAT fVer;

    fVer = COMM_LibVer();

printf("Library version %.2f", fVer);    // Print library version
    Information
}
```

1.1.2. COMM_ParaInit ( )

Function Name	void _stdcall COMM_ParaInit(void);
Functional Description	Communication parameters initialization function。
Input Parameters	no
Output parameters	no

The return value	no
Precautions	**This function is called to be the first and only called once。**

1.1.3. COMM_UartInit ( )

Function Name	INT8U _stdcall COMM_UartInit(INT8U UartPort, INT32U UartBaud, INT8U DataBit, INT8U StopBit, INT8U Parity);
Functional Description	Initialization open the serial port , serial port communication configuration。
Input Parameters	UartPort: serial number (0,1,2,3) UartBaud: Set the corresponding serial port baud rate DataBit: Serial data Bit (5,6,7,8Bit four ways) StopBit: Serial Stop Bit (1,2Bit two ways) Parity: serial parity Bit (no parity, odd parity, even parity three ways)
Output parameters	no
The return value	COMM_NO_ERR: Normal
Precautions	no

示 example：

```
voidmain( )
{
    //Open the serial port 0 , the baud rate is 115200,8Bitdata Bit, 1Bit Stop Bit, no parity.
if (COMM_UartInit (COMM_UART_PORT0, CBR_115200, 8, ONESTOPBIT, NOPARITY) == COMM_NO_ERR)
{
printf (" Open serial success ! " ) ;
}
}
```

1.1.4. COMM_NetConnect ( )

Function Name	INT8U _stdcall COMM_NetConnect(INT8U NetPort, INT32S iDevNum, INT32U uiConIp, INT16U nConPort);
Functional Description	Make TCP / UDP connections , and the corresponding mapping equipment number and IPAddress
Input Parameters	NetPort: Network port number for the TCP port specified 4 , 5 to UDP port iDevNum: Network communication device number corresponds , Range （0 ～ 255） uiConIp: Connected to the server IPAddress nConPort: Server communication port number of the connection (usually fixed communications 502)
Output parameters	no
The return value	COMM_NO_ERR: Normal
Precautions	no

示 example：

```
voidmain( )
{
    INT32U uiConIp;

    //The integer value is converted to a string IP
uiConIp = htonl (inet_addr ("192.168.0.100"));
//Conduct UDP network connections , the assigned device number is 1 , the corresponding IP: 192.168.0.100, port number: 502
if (COMM_NetConnect (COMM_UDP_PORT, 1, uiConIp, 502) == COMM_NO_ERR)
{
printf (" Connection successful ! " ) ;
}
}
```

1.1.5. COMM_CloseNetConn ( )

Function Name	INT8U _stdcall COMM_CloseNetConn(INT8U NetPort, INT32S iDevNum);
Functional Description	Close the connection to the corresponding device , Only forTCP / UDP connections use
Input Parameters	NetPort: Network port number for the TCP port specified 4 , 5 to UDP port iDevNum: Network communication device number corresponds , Range （0 ～ 255）
Output parameters	no
The return value	COMM_NO_ERR: Normal
Precautions	This function is mainly used for the network device , the use of the corresponding device is turned off is connected , does not affect the operation of other network devices.

示 example：

```
void main( )
{
// Off connected devices under a UDP connection
if (COMM_CloseNetConn (COMM_UDP_PORT, 1) == COMM_NO_ERR)
{
printf (" Connection closed successfully ! " ) ;
}
}
```

1.1.6. COMM_ClosePort ( )

Function Name	INT8U _stdcall COMM_ClosePort(INT8U Port);
Functional Description	Close the corresponding communication port for network communication, connection with the closure of its corresponding devices are also will be closed.

Input Parameters	Port: communication port number , Range0 ~ 5 (0,1,2,3 which is serial ; 4 for TCP network ; 5 for UDP network)
Output parameters	no
The return value	COMM_NO_ERR: Normal
Precautions	**The function call will make this port is disconnected from all connected communications .。**

示 example：

```
void main( )
{
    COMM_ClosePort(COMM_UART_PORT0);   //Close the serial port.0
    COMM_ClosePort(COMM_UART_PORT1);   //Close the serial port.1
    COMM_ClosePort(COMM_UART_PORT2);   //Close the serial port.2
    COMM_ClosePort(COMM_UART_PORT3);   //Close the serial port.3

    COMM_ClosePort(COMM_TCP_PORT);     //Close the NetworkTCPConnection
    COMM_ClosePort(COMM_UDP_PORT);     //Close the NetworkUDPConnection
}
```

1.1.7. COMM_SetTimesOut ( )

Function Name	BOOL _stdcall COMM_SetTimesOut(INT8U port, INT16U timeouts, INT8U repeat_times);
Functional Description	Set the corresponding port communications timeout and abnormal number of retransmissions
Input Parameters	port: Communications port number timeouts: Timeout(Unit:ms) repeat_times: Abnormal number of retransmissions
Output parameters	no
The return value	TRUE: Normal
Precautions	no

1.1.8. COMM_GetTimesOut ( )

Function Name	BOOL _stdcall COMM_GetTimesOut(INT8U port, INT16U *timeouts, INT8U *repeat_times);
Functional Description	Get the corresponding port CommunicationsTimeout and Abnormal number of retransmissions
Input Parameters	port: Communications port number
Output parameters	*timeouts: Timeout(Unit:ms) *repeat_times: Abnormal number of retransmissions
The return value	TRUE: Normal
Precautions	no

示 example：

```
void main( )
{
    INT8U  repeat_times;
    INT16U timeouts;

    //Set COMM_UART_PORT0 serial Timeout for 1000ms, most Multi- retransmission timeout 4 times
COMM_SetTimesOut (COMM_UART_PORT0, 1000, 4);

//Get Timeout and most Multi- serial number of retransmissions COMM_UART_PORT0 set
COMM_GetTimesOut (COMM_UART_PORT0, & timeouts, & repeat_times);

//Print acquired Timeout and most Multi- number of retransmissions
printf ("timeouts =% d, repeat_times =% d \ n", timeouts, repeat_times);
}
```

1.1.9. COMM_ByteToAscii ( )

Function Name	void _stdcall COMM_ByteToAscii(INT8U data, INT8U *ch, INT16U *offset);
Functional Description	Bytesdata Turn ASCII characters
Input Parameters	data: To convertdata
Output parameters	*ch: Array to store the converted ASCII value * offset: an array of data in the position (will point to the next data position after data access)
The return value	no
Precautions	no

1.1.10. COMM_AsciiToByte ( )

Function Name	void _stdcall COMM_AsciiToByte(INT8U *data, INT8U *ch, INT16U *offset);
Functional Description	ASCIICharacters turnBytesdata
Input Parameters	*ch: Convert ASCII value of data stored in an array *offset: position will be an array of data points to the next data after (data access position)
Output parameters	*data: Converteddata
The return value	no
Precautions	no

示 example：

```
void main( )
{
    INT8U data;
    INT16U offset;
    INT8U ch[5];
```

```
    offset = 0;
COMM_ByteToAscii (0x12, ch, & offset); // convert the 0x12 character ' a ', ' two ' .
COMM_ByteToAscii (0x34, ch, & offset);
ch [4] = '\ 0'; // add terminator
printf ("ch:% s, offset:% d", ch, offset); // print the results : ch: 1234, offset: 4
```

```
offset = 0;
    COMM_AsciiToByte(&data, ch, &offset);
    printf("data=0x%x, offset=%d", data, offset);
    COMM_AsciiToByte(&data, ch, &offset);
    printf("data=0x%x, offset=%d", data, offset);
}
```

1.1.11. COMM_Send ( )

Function Name	INT16U _stdcall COMM_Send(INT8U Port, INT8U *sbuff, INT16U length);
Functional Description	Serial port to send data to the specified function
Input Parameters	Port: Communications port number，Only use the serial port *sbuff: Senddata Storage area length: Senddata length
Output parameters	no
The return value	Returns Send the data length
Precautions	Send the data to the function directly through the appropriate serial port.

1.1.12. COMM_Recv ( )

Function Name	INT16U _stdcall COMM_Recv(INT8U Port, INT8U *rbuff, INT16U length);
Functional Description	Serial read data function

Input Parameters	Port: Communications port number length: Specify the read data length
Output parameters	*rbuff: Readdata Storage area
The return value	Returns the actual received data length
Precautions	The function receives data directly from the corresponding serial port.

1.2 Processing Management function classes

```
//Process management operations command
#define  WORK_NOCMMD              0x00    /Invalidcommand
#define  WORK_RESET               0x01    //Reset command
#define  WORK_WRITEADDR           0x02    // Write discontinuous register Address and the corresponding length
 command
#define WORK_WRITEADDREXT         0x03    // Extended Read Write discontinuous register Address and the corresponding length command (data length does not exceed the maximum frame agreement )
#define  WORK_READDATAEXT         0x04    //ReadMulti-Discontinuousregister data
#define  WORK_WRITEDATAEXT        0x05    //WriteMulti-个 Discontinuousregister data


//ErrorTypeDefinition
#define  WORK_NO_ERR              0x00    //OperationnoError
#define  WORK_INVALID_CMMD        0x01    /No effect command
#define  WORK_INVALID_DATA        0x02    //Readnoeffectdata
#define  WORK_CMMDLEN_ERR         0x03    // command length Error
#define WORK_CMMDREAD_ERR         0x04    //Read command Error
#define WORK_CMMDWRITE_ERR        0x05    //Write command Error
#define  WORK_DATALEN_OVER        0x06    //data  length Beyond the maximum cache Range
#define  WORK_DATALEN_ERR         0x07    //data  length Error
```

```
#define  WORK_READDATA_ERR        0x08    //Readdata    Error
#define  WORK_WRITEDATA_ERR       0x09    //Writedata    Error

#define  WORK_NULLADDR_ERR        0x0a    //EmptyAddressError
#define  WORK_DEVICE_ERR          0x0b    //EquipmentError
#define  WORK_TIMEOUT_ERR         0x0c    //Communication timeoutError
```

1.2.1. WORK_SelectComm ( )

Function Name	INT8U _stdcall WORK_SelectComm(INT8U port, INT32S iDevNum, INT8U mode, BOOL bEnable);
Functional Description	For data access processing section , select the appropriate communication settings Connection.
Input Parameters	port: Communications port number iDevNum: Slave EquipmentNo. mode: Select RTU or ASCIIMode for serial communication bEnable: Communication is enabled (TRUE: Allow communication FALSE: ban Communications)
Output parameters	no
The return value	WORK_NO_ERR: Normal
Precautions	no

1.2.2. WORK_GetCommInfo ( )

Function Name	INT8U _stdcall WORK_GetCommInfo(INT8U *port, INT32S *iDevNum, INT8U *mode, BOOL *bEnable);
Functional Description	For data access processing part , to get the current state of communications Connection .
Input Parameters	no

Output parameters	port: Communications port number iDevNum: SlaveEquipmentNo. mode: Select RTU or ASCIIMode for serial communication bEnable: Communication is enabled (TRUE: Allow communication FALSE: ban Communications)
The return value	WORK_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    INT8U port;
    INT32S iDevNum;
    INT8U  mode;
    BOOL   bEnable;

    // Select COMM_UART_PORT0 serial, station No. is 1, RTU Communication Mode, communications enabled.
WORK_SelectComm (COMM_UART_PORT0, 1, COMM_MODE_RTU, TRUE);
// Get Communications Information Processing
    WORK_GetCommInfo(&port, &iDevNum, &mode, &bEnable);
    printf("port=%d, iDevNum=%d, mode=%d, bEnable=%d\n",
        port, iDevNum, mode, bEnable);
}
```

1.2.3. WORK_ReadInBit ( )

Function Name	INT8U _stdcall WORK_ReadInBit(INT16U nAddr, INT8U *pBuff, INT16U nNum);
Functional Description	Read one or Multi- consecutive input Bitregister Address state.
Input Parameters	nAddr: Bitregister Home Address nNum: Bit access to the number o

Output parameters	pStatus: Read out stored Bit state array
The return value	WORK_NO_ERR: Normal
Precautions	no

1.2.4. WORK_ReadBit ( )

Function Name	INT8U _stdcall WORK_ReadBit(INT16U nAddr, INT8U *pBuff, INT16U nNum);
Functional Description	Read one or Multi- consecutive Bitregister Address state.
Input Parameters	nAddr: Bitregister Home Address nNum: Bit access to the number of
Output parameters	pBuff: Read out stored Bit state array
The return value	WORKNO_ERR: Normal
Precautions	no

1.2.5. WORK_WriteBit ( )

Function Name	INT8U _stdcall WORK_WriteBit(INT16U nAddr, INT8U *pBuff, INT16U nNum);
Functional Description	WriteOne orMulti-ConsecutiveBitregister AddressState。
Input Parameters	nAddr: Bitregister Home Address pBuff: Write array to store incoming BitState nNum: Bit access to the number o
Output parameters	no
The return value	WORK_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    int i;
    INT8U wSta[10];
    INT8U  rSta[10];

    memset(rSta, 0, 10);
    //ReadBitAddress 100 started Consecutive10 months only Read input State
if(WORK_ReadInBit(0, &rSta, 10) != WORK_NO_ERR)
    {
        printf("Read in bit Error!\n");
        return;
    }
    for(i=0; i<10; i++)
    {
        printf("%d, ", rSta[i])       //Print inputState
    }
    printf("\n")

    for(i=0; i<10; i++)
    {
        wSta[i] = ((i%4)==0) ? 1 : 0;
    }

    // 100 began to BitAddress Consecutive10 output points Write the State
    if(WORK_WriteBit(100, &wSta, 10) != WORK_NO_ERR)
    {
        printf("Write bit Error!\n");
        return;
    }

    memset(rSta, 0, 10);
    // Read the BitAddress output points to 100 Consecutive10 State begins
```

```
    if(WORK_ReadBit(100, &rSta, 10) != WORK_NO_ERR)
    {
        printf("Read bit Error!\n");
        return;
    }

    for(i=0; i<10; i++)
    {
        printf("%d, ", rSta[i]);          //Print outputState
    }
    printf("\n")
}
```

1.2.6. WORK_ReadReg ( )

Function Name	INT8U _stdcall WORK_ReadReg(INT16U nAddr, INT8U *pBuff, INT16U nSize);
Functional Description	ReadOne orMulti-Consecutive word of data storage Address
Input Parameters	no
Output parameters	no
The return value	WORK_NO_ERR: Normal
Precautions	no

1.2.7. WORK_WriteReg ( )

Function Name	INT8U _stdcall WORK_WriteReg(INT16U nAddr, INT8U *pBuff, INT16U nSize);
Functional Description	Write the One orMulti-Consecutive word Storage Address of data.

Input Parameters	nAddr:	word register Home Address
	pBuff:	Write register data to be stored into an array
	nSize:	To Write into the register data Bytes size
Output parameters	no	
The return value	WORK_NO_ERR: Normal	
Precautions	no	

示 example：

```
void main( )
{
    int i;
    INT8U  err;
    INT32   pos = 0;

    //SelectCOMM_UART_PORT0Serial ， 站 No.is 1 ， RTUCommunicationsMode ，Communications Enable。
    WORK_SelectComm(COMM_UART_PORT0, 1, COMM_MODE_RTU, TRUE);

    //example X axis feedbackposition register StartingAddress is 5060，data Type is 32Bit signed Integer。
    err = WORK_ReadReg(5060, (INT8U*)&pos, 4);   //ReadX axis feedbackposition
    if(err != WORK_NO_ERR)
    {
        printf(“Read register err!\n”);
        return;
    }
    printf(“pos1 = %d\n”, pos);
    pos = 1000;
    err = WORK_WriteReg(5060, (INT8U*)&pos, 4);   //WriteX axis feedbackposition is 1000
    if(err != WORK_NO_ERR)
    {
```

```
        printf("Write register err!\n");
        return;
    }
    pos = 0;
err = WORK_ReadReg(5060, (INT8U*)&pos, 4);    // ReadX axis feedbackposition again

    if(err != WORK_NO_ERR)
    {
        printf("Read register err!\n");
        return;
    }
    printf("pos2 = %d\n", pos);
}
```

1.2.8. WORK_ReadDiscReg ( )

Function Name	INT8U _stdcall WORK_ReadDiscReg(const INT16U *pRegBuff, INT16U nBuffNum, INT8U *pData);
Functional Description	ReadOne orMulti- a Discontinuous word of data storage Address
Input Parameters	pRegBuff: Store Discontinuous Storage Address Readdata length of the array and the corresponding nBuffNum: pRegBuff contents of the array length
Output parameters	pData: Read the Discontinuousregister data stored in an array
The return value	WORK_NO_ERR: Normal
Precautions	no

1.2.9. WORK_WriteDiscReg ( )

Function Name	INT8U _stdcall WORK_WriteDiscReg(const INT16U *pRegBuff, INT16U nBuffNum, INT8U *pData);

Functional Description	Write a Discontinuous word into One orMulti- of data storage Address 。
Input Parameters	pRegBuff: Store Discontinuous Storage Address Readdata length of the array and the corresponding nBuffNum: pRegBuff contents of the array length pData: To Write an array to store the incoming Discontinuousregister data
Output parameters	no
The return value	WORK_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    int     i;
    INT8U   err;
    INT8U   pData[10];
    INT16U  nBuffNum;
    INT16U  pRegBuff[10];

// Select COMM_UART_PORT0 serial port, station No. is 1, RTU Communication Mode, COMM
    。
    WORK_SelectComm(COMM_UART_PORT0, 1, COMM_MODE_RTU, TRUE);

    // example , such as X, Z axis feedbackpositionregister starting Address respectively 5060,5064 . Where X axis occupancy rooms 5060,5061 word AddressEmpty, Z axis occupancy rooms 5064,5065 word AddressEmpty.
    nBuffNum = 0;
    pRegBuff[nBuffNum++] = 5060;
    pRegBuff[nBuffNum++] = 2;
    pRegBuff[nBuffNum++] = 5064;
```

```
    pRegBuff[nBuffNum++] = 2;

    err = WORK_ReadDiscReg (pRegBuff, nBuffNum, pData);  //ReadX , Z axis feedbackposition
    if(err != WORK_NO_ERR)
    {
        printf("Read register err!\n");
        return;
    }
    printf("posx1= %d, posz1=%d\n", *(INT32S*)pData, *(INT32S*)(pData+4));

    *(INT32S*)pData = 1000;
    *(INT32S*)(pData+4) = 2000;

// To the X, Z axis feedbackposition Write into 1000 and 2000 , respectively .
err = WORK_WriteDiscReg (pRegBuff, nBuffNum, pData);
if (err! = WORK_NO_ERR)
    {
        printf("Write register err!\n");
        return;
    }
    *(INT32S*)pData = 0;
    *(INT32S*)(pData+4) = 0;

err = WORK_ReadDiscReg (pRegBuff, nBuffNum, pData);  // Again ReadX, Z axis feedbackposition
    if(err != WORK_NO_ERR)
    {
        printf("Read register err!\n");
        return;
    }

    printf("posx2= %d, posz2=%d\n", *(INT32S*)pData, *(INT32S*)(pData+4));
}
```

Parameter Management Function classes

1.1. //access command Definition

```
#define  PARA_NOCMMD              0x00    /Invalidcommand
#define  PARA_READLIST            0x01    //ReadParametersAll Information
#define  PARA_READINFO            0x02    //ReadParametersInformation

//ErrorTypeDefinition
#define  PARA_NO_ERR              0x00    //OperationnoError
#define  PARA_INVALID_CMMD        0x01    /No effect command
#define  PARA_INVALID_DATA        0x02    //Readnoeffectdata
#define  PARA_READLIST_ERR        0x03    //ReadListError
#define  PARA_READINFO_ERR        0x04    //ReadParametersInformationError
#define  PARA_READVALUE_ERR       0x05    //Readdata  Error
#define  PARA_WRITEVALUE_ERR      0x06    //Writedata  Error
#define  PARA_READPOS_ERR         0x07    //ReadpositionError
#define  PARA_WRITEPOS_ERR        0x08    //WritepositionError
#define  PARA_NULLADDR_ERR        0x09    //EmptyAddressError
#define  PARA_DEVICE_ERR          0x0a    //EquipmentError
#define  PARA_TIMEOUT_ERR         0x0b    //Communication timeoutError
#define  PARA_LOADFILE_ERR        0x0c    //Load Parametersfile  Error
#define  PARA_SAVEFILE_ERR        0x0d    //Save Parametersfile  Error

#define  PARAINFO_SIZE            80      //ParametersInformationBytes Value

//Parameters Properties Definition
#define  REG_RO                   (0<<7)  //只 Readregister
#define  REG_RW                   (1<<7)  //ReadWriteregister
#define  REG_USER                 (0)     // General Customer Permissionsregister
#define  REG_SUPER                (1)     //Super User  Permissionsregister

#define  RO_USER                  (REG_RO|REG_USER)
#define  RW_USER                  (REG_RW|REG_USER)
#define  RO_SUPER                 (REG_RO|REG_SUPER)
#define  RW_SUPER                 (REG_RW|REG_SUPER)
```

```
//data  TypeDefinition，Type 4Bit used to represent the value of the high number VariablesBytes
#define   DT_NULL              0x00
#define   DT_INT8S             0x11
#define   DT_INT8U             0x12
#define   DT_INT16S            0x23
#define   DT_INT16U            0x24
#define   DT_INT32S            0x45
#define   DT_INT32U            0x46
#define   DT_PPS               0x47
#define   DT_FLOAT             0x48
#define   DT_DOUBLE            0x89
#define   DT_STR               0x0a
#define   DT_VER               0x0b
#define   DT_TITLE             0x0c
#define   DT_END               0x0d


typedef struct _PARA_TAB_
{
    INT8U  Popedom;                 //ParametersOperationPermissions
    INT8U  DataType;                //Parametersdata   Type
    INT16U PlcAddr;                 //Corresponding   PLCAddress
    INT8S  pText[PARAINFO_SIZE-16];// Parameters descriptor , fixed 64 Bytes

    INT8U   pData[4];               // Parameters descriptor , fixed 64 Bytes
    FP32    fLmtVal[2];             // Parameters RMS boundaries : [ 0 ] minimum ; [ 1 ] max
}PARA_TAB;


typedef struct _PARA_SORT_
{
    int            iCount;          //Parameters Quantity
    PARA_TAB *pTitle;               //ParametersTypeName
```

```
    PARA_TAB *pParam;          // Parameters list Home Address
}PARA_SORT;
```

1.3.1. PARA_SelectComm ( )

Function Name	INT8U _stdcall PARA_SelectComm(INT8U port, INT32S iDevNum, INT8U mode, BOOL bEnable);
Functional Description	For data access Parameters section , Select set the appropriate CommunicationsConnection.
Input Parameters	port: Communications port number iDevNum: SlaveEquipmentNo. mode: For SerialCommunicationsSelectRTU or ASCIIMode bEnable: yes/no EnableCommunications(TRUE:Allow Communications FALSE: Ban Communications)
Output parameters	no
The return value	PARA_NO_ERR: Normal
Precautions	no

1.3.2. PARA_GetCommInfo ( )

Function Name	INT8U _stdcall PARA_GetCommInfo(INT8U *port, INT32S *iDevNum, INT8U *mode, BOOL *bEnable);
Functional Description	Data access ForParameters part,Get the current CommunicationsConnectionState。
Input Parameters	no
Output parameters	port: Communications port number iDevNum: SlaveEquipmentNo. mode: ForSerialCommunicationsSelectRTU or ASCIIMode bEnable: yes/no EnableCommunications(TRUE:Allow Communications FALSE: Ban Communications)
The return value	PARA_NO_ERR: Normal

Precautions	no

1.3.3. PARA_GetParaVerify ( )

Function Name	INT16U _stdcall PARA_GetParaVerify(PARA_TAB *pTab, int iTabNum);
Functional Description	Calculate the checksum value of the specified number ParametersList
Input Parameters	pTab: PARA_TABType array Home Address iTabNum: The number ParametersList
Output parameters	no
The return value	Checksum calculation
Precautions	no

1.3.4. PARA_GetParaInfo ( )

Function Name	INT8U _stdcall PARA_GetParaInfo(INT16U *pParaVer, INT16U *pVerify, INT32U *pParaSize);
Functional Description	Get Remote EquipmentParametersVersion , Check value and total ValueInformation
Input Parameters	no
Output parameters	pParaVer: StoreParametersVersion pointer pVerify: Check value pointer StoreParametersList pParaSize: StoreParametersList Total Value pointer
The return value	PARA_NO_ERR: Normal
Precautions	no

1.3.5. PARA_ReadParaTab ( )

Function Name	PARA_TAB* _stdcall PARA_ReadParaTab(int *piTabNum, INT16U *pVersion, INT8U *pErr);
Functional Description	Read Remote Equipment ParametersListContent
Input Parameters	no
Output parameters	piTabNum: StoreParametersList number of pointers pVerify: StoreParametersListPointer check value pErr: StoreFunctionsOperationErrorInformation pointer
The return value	Non-Empty：PARA_TABTypePointer NULL：Get failure
Precautions	no

1.3.6. PARA_TabToSort ( )

Function Name	PARA_SORT* _stdcall PARA_TabToSort(PARA_TAB *pTab, int iTabNum, int *piSortNum);
Functional Description	将 ParametersListContentClassification，Converted to PARA_SORTTypedata
Input Parameters	pTab: PARA_TABType array Home Address iTabNum: ParametersList Number
Output parameters	piSortNum: StoreParametersType number of pointers
The return value	Non-Empty：PARA_TABTypePointer NULL：Get failure
Precautions	no

1.3.7. PARA_Destroy ( )

Function Name	void _stdcall PARA_Destroy(PARA_SORT *pSort);
Functional Description	Get Bit Machine Parameters under or Read local matching Parameters residence and create ParametersList.

Input Parameters	pSort: PARA_SORTTypePointer
Output parameters	no
The return value	no
Precautions	no

示 example：

```
void mian( )
{
    int   iTabNum;
    INT8U err;
    INT16U      nVersion1, nVersion2;
    INT16U      nVerify1, nVerify2;
    INT32U      uiParaSize;
    PARA_TAB*     pParaTab;
    PARA_SORT*    pParaSort;

    iTabNum = 0;

    // Start a local file in ReadParameters

    pParaTab = LoadFileToParaTab(_T(“para.dat”), &iTabNum, &err);
    if(pParaTab == NULL)
    {
        // Read directly from the local file after the failure of a remote EquipmentRead
pParaTab = PARA_ReadParaTab (& iTabNum, & nVersion1, & err);

        // Save from a remote EquipmentRead of ParametersListInformation to the local file
SaveParaTabToFile (_T ("para.dat"), pParaTab, iTabNum, nVersion1)
;
    }

    if(pParaTab != NULL)
```

```
    {
        // Get ParametersList checksum

        nVerify1 = PARA_GetParaVerify(pParaTab, iTabNum);
        printf("nVersion1=%d, nVerify1=%d\n", nVersion1, nVerify1);

        //Get Remote EquipmentParametersVersion，Check value and total ValueInformation
        err = PARA_GetParaInfo(&nVersion2, &nVerify2, &uiParaSize);
        if(err == PARA_NO_ERR)
        {
            printf("nVersion2=%d, nVerify2=%d, uiParaSize=%d\n",
                nVersion2, nVerify2, uiParaSize);
        }

        // The ParametersListContentClassification conversion
        pParaSort = PARA_TabToSort(pParaTab, iTabNum, &iParaSort);
    }

    PARA_Destroy(pParaSort);          // Finally released to create Parameters
}
```

1.3.8. PARA_ReadValue ( )

Function Name	INT8U _stdcall PARA_ReadValue(PARA_TAB *pTab);
Functional Description	Bit machine Read values from the Parameters of
Input Parameters	no
Output parameters	pTab: PARA_TABTypePointer, after successful implementation , members of the Parameters value pTab will have to be updated.
The return value	PARA_NO_ERR: Normal
Precautions	no

1.3.9. PARA_WriteValue ( )

Function Name	INT8U _stdcall PARA_WriteValue(PARA_TAB *pTab, void *pData);
Functional Description	Bit Machine Write down the value of the Parameters after the execution of successful local Parameters value does not get updated immediately , to be under the corresponding Parameters value Bit machines by PARA_ReadValueFunctionsRead.
Input Parameters	pTab: PARA_TABTypePointer, Store the Parameters All Information. pData: Store To Write into the Parameters of Parameters value.
Output parameters	no
The return value	PARA_NO_ERR: Normal
Precautions	no

example：

```
void main( )
{
    PARA_TAB Tab;
    INT16U    data;
    Tab.Popedom = RW_USER;
    Tab.DataType = DT_INT16U;
    Tab.PlcAddr = 8000;
    strcpy (Tab.pText, "001, X -axis command multiplier
<●>");
    Tab.fLmtVal[0] = 1.0;
    Tab.fLmtVal[1] = 65535.0;
    data = 10;

    if(PARA_WriteValue(&Tab, &data) != PARA_NO_ERR)
```

```
    {
        printf("Write parameter value err!\n");
        return;
    }

    if(PARA_ReadValue(&Tab) != PARA_NO_ERR)
    {
        printf("Read parameter value err!\n");
        return;
    }

    printf("Read value : %u, data=%u\n", *(INT16U*)(Tab.pData), data);
}
```

1.3.10. PARA_ReadMultValue ( )

Function Name	INT8U _stdcall PARA_ReadMultValue(PARA_TAB *pTab, int iNum);
Functional Description	From the values of Bit machine ReadConsecutiveMulti- months Parameters 。
Input Parameters	pTab: PARA_TABTypePointer，Multi-ParametersPointerHome Address。 iNum: ConsecutiveParameters number .
Output parameters	pTab: PARA_TABTypePointer, after successful implementation , pTab for ConsecutiveMulti- a Home Address Parameters member value will be updated.
The return value	PARA_NO_ERR: Normal
Precautions	no

example：

```
void main( )
{
    int     i;
    PARA_TAB Tab[8];
```

```
    for(i=0; i<8; i++)
    {
        Tab.Popedom = RW_USER;
        Tab.DataType = DT_INT16U;
        Tab.PlcAddr = 8000 + i;        //Because data Type are no breaks 16BitInteger, so direct plus
i
        Tab.fLmtVal[0] = 1.0;
        Tab.fLmtVal[1] = 65535.0;
    }

    if(PARA_ReadMultValue (Tab, 8) != PARA_NO_ERR)
    {
        printf("Read parameter value err!\n");
        return;
    }

    for(i=0; i<8; i++)
    {
        printf("<%d> Read value : %u\n", i, *(INT16U*)(Tab.pData));
    }
}
```

file Management Functions category

1.2.

```
//access     command Definition
#define  FS_NOCMMD          0x00    /Invalid command
#define  FS_READDIR         0x01    //Read directory
#define  FS_CREATEDIR       0x02    //Creating directory
#define  FS_DELETEDIR       0x03    //Deletedirectory
#define  FS_DELETEFILE      0x04    //Deletefile
#define  FS_RENAMEFILE      0x05    //file   Renaming
#define  FS_READFILE        0x06    // Open and Readfile
#define  FS_WRITEFILE       0x07    // Open and Writefile
```

```
#define  FS_CLOSEFILE          0x08     //Shut downfile

//ErrorTypeDefinition
#define  FS_NO_ERR             0x00     //file    OperationnoError
#define  FS_INVALID_CMMD       0x01     /No effect command
#define  FS_FILE_UNOPEN        0x02     //file      no Turn on
#define  FS_FILE_UNCLOSE       0x03     //file      no Shut down
#define  FS_PATH_ERR           0x04     // Path Error
#define  FS_READDIR_ERR        0x05     //ReaddirectoryListError
#define  FS_CREATDIR_ERR       0x06     //CreatingdirectoryError
#define  FS_DELDIR_ERR         0x07     //DeletedirectoryError
#define  FS_DELFILE_ERR        0x09     //Deletefile    Error
#define  FS_OPENFILE_ERR       0x0a     //Turn onfile    Error
#define  FS_CLOSEFILE_ERR      0x0b     //Shut downfile    Error
#define  FS_READDATA_ERR       0x0c     //Readdata    Error
#define  FS_WRITEDATA_ERR      0x0d     //Writedata    Error
#define  FS_READPOS_ERR        0x0e     //ReadpositionError
#define  FS_WRITEPOS_ERR       0x0f     //WritepositionError
#define  FS_INVALID_DATA       0x10     //Readnoeffectdata
#define  FS_PATHNAME_ERR       0x11     // Path    name    Error
#define  FS_NULLADDR_ERR       0x12     //EmptyAddressError
#define  FS_DEVICE_ERR         0x13     //EquipmentError
#define  FS_TIMEOUT_ERR        0x14     //Communication timeoutError

#define  FS_READ               0x01     //only Read
#define  FS_WRITE              0x02     //only Write

#define MAX_LIST               0x100    //dwithin irectory contains the largest number of file
directory

#define  MAX_NAMEPATH  200        //Maximumfile    /directory Path Byteslength

#define FS_FNULL               0xff
#define  FS_FDISK              0x00     //Disk
```

```
#define  FS_FDIR              0x01     //directory
#define  FS_FDOC              0x02     //Documentation

typedef struct _FS_DIRENT_
{
    INT8U       Attr;              //file   Properties(0:Disk  1:directory      2:Documentation)
    INT8S       Name[15];          //file    name   (8+3Format)

    INT32U      Size;              //file   Value/DiskTotalValue
    INT32U      FreeSize;          //Disk remainingBytesValue
}FS_DIRENT;// Unit Value24 Bytes

typedef struct _CMM_DIR_
{
    INT32U      uiDirent;                 //directoryList Number

    FS_DIRENT pDirent[MAX_LIST];//directoryListInformation
}FS_DIR;

typedef struct _FS_FILE_
{
    INT32S      filepos;  //file   access   position
    INT32S      size;          //file   Value

    INT8U       mode;          //file   ReadWriteOperation
    INT8U       err;       //file   ErrorInformation
}FS_FILE;
```

1.4.1. FS_SelectComm ( )

Function Name	INT8U _stdcall FS_SelectComm(INT8U port, INT32S iDevNum, INT8U mode, BOOL bEnable);
Functional Description	Data access Forfile part , Select set the appropriate CommunicationsConnection.

Input Parameters	port: Communications port number iDevNum: SlaveEquipmentNo. mode: ForSerialCommunicationsSelectRTU or ASCIIMode bEnable: yes/no EnableCommunications(TRUE:Allow Communications FALSE: Ban Communications)
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.2. FS_GetCommInfo ( )

Function Name	INT8U _stdcall FS_GetCommInfo(INT8U *port, INT32S *iDevNum, INT8U *mode, BOOL *bEnable);
Functional Description	Data access Forfile part , to get the current CommunicationsConnectionState.
Input Parameters	no
Output parameters	port: Communications port number iDevNum: SlaveEquipmentNo. mode: ForSerialCommunicationsSelectRTU or ASCIIMode bEnable: yes/no EnableCommunications(TRUE:Allow Communications FALSE: Ban Communications)
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.3. FS_FReadDir ( )

Function Name	INT8U _stdcall FS_FReadDir(const INT8S *pDirName, FS_DIR *pDir);
Functional Description	DirectoryInformation Read remote file system.

Input Parameters	pDirName: To Read the remote Equipmentfile system directory name. Does not know the remote file in the path of the case , through the ReadEmpty string "" to Get Remote file system's root directoryInformation. Get the directory structure of the residence where the Store in pDir storage area.
Output parameters	pDir: Structure StoredirectoryInformation of Pointer.
The return value	FS_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    FS_DIR pDir;

    err = FS_FReadDir(“”, pDir);      // Empty Path said Read root directoryInformation (DiskInformation)
    if(err != FS_NO_ERR)
    {
        printf ("Read the root directory failed!
”)
    }

    err = FS_FReadDir(“C:\\”, pDir);  //ReadCDiskdirectoryInformation
    if(err != FS_NO_ERR)
    {
        printf(“ReadCDiskRootdirectoryInformationFailure！”)
    }

    err = FS_FReadDir(“D:\\ADT\\”, pDir);     //ReadDDisk 下 ADTdirectoryInformation
    if(err != FS_NO_ERR)
    {
        printf(“ReaddirectoryFailure！”)
    }
```

}

1.4.4. FS_FOpen ( )

Function Name	FS_FILE* _stdcall FS_FOpen(const INT8S *pFileName, const char *pMode);
Functional Description	Turn on remote file
Input Parameters	pFileName: To Turn on the remote file Path, all file name are absolutely Path. pMode: file Turn onMode('r':ReadOperation 'w':WriteOperation)
Output parameters	no
The return value	FS_FILE: file structure Pointer, after performing successful return Non-0 value 。
Precautions	When access remote file, while only Turn on a remote file. The Functions and FS_FClose () FunctionsCorresponding use , after the success of Turn onfile, ReadWrite must Shut downfile After data.

1.4.5. FS_FRead ( )

Function Name	INT8U _stdcall FS_FRead(FS_FILE *pFile, void *pData, INT32U size);
Functional Description	Read Remote file Content
Input Parameters	pFile: file structure Pointer. size: Readfile Content of Byteslength.
Output parameters	pData: StoreRead the file Content.
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.6. FS_FWrite ( )

Function Name	INT8U _stdcall FS_FWrite(FS_FILE *pFile, const void *pData, INT32U size);
Functional Description	Write Remote file Content
Input Parameters	pFile: file structure Pointer. pData: StoreWrite into the file Content. size: Writefile Content 的 Byteslength 。
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.7. FS_FClose ( )

Function Name	INT8U _stdcall FS_FClose(FS_FILE *pFile);
Functional Description	Shut down Remote file Content，与 FS_FOpenCorresponding Use。
Input Parameters	pFile: file Structure Pointer。
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.8. FS_Remove ( )

Function Name	INT8U _stdcall FS_Remove(const INT8S *pFileName);
Functional Description	Deletefile

Input Parameters	pFileName: Deletefile Path
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    int     size;
    char    Buff[100];
    INT8U   err;
    FS_FILE *pFile;

    pFile = FS_FOpen(“D:\\ADT\\TEST.TXT”, “r”);  // Read only way to Turn on Remote file
    if(pFile == NULL)
    {
        printf(“Turn on Remote file Failure!”);
        return;
    }
    printf(“File size = %d\n”, pFile->size);  // Print Turn onfile Value

    err = FS_FRead(pFile, Buff, 100);  //from Remote file Read100Bytesdata
    FS_FClose(pFile);  //Shut downTurn on file
    if(err != FS_NO_ERR)
    {
        printf(“Read Remote file Failure！”);
        return;
    }

    FS_Remove(“D:\\ADT\\TEST.TXT”);  //DeleteTEST.TXTfile
```

```
pFile = FS_FOpen("D:\\ADT\\ADT.TXT", "w");    //Turn on 一个可 Write Remote file ，
Not then Creating
        if(pFile == NULL)
    {
        printf("Turn on Remote file Failure!");
        return;
    }

    err = FS_FWrite(pFile, Buff, 100);          //from Remote file Write100Bytesdata
    FS_FClose(pFile);                           //Shut downTurn on 的 file
    if(err != FS_NO_ERR)
    {
        printf("Write Remote file Failure！");
        return;
    }

}
```

1.4.9. FS_FSeek ( )

Function Name	INT8U _stdcall FS_FSeek(FS_FILE *pFile, INT32S Offset);
Functional Description	定 Bitfile Operationposition。
Input Parameters	pFile: file StructurePointer。 Offset: Operationposition given Bitfile , relative to file start position.
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.10. FS_FTell ( )

Function Name	INT8U _stdcall FS_FTell(FS_FILE *pFile, INT32S *Offset);
Functional Description	GetCurrentfile Operationposition。
Input Parameters	pFile: file StructurePointer。
Output parameters	Offset: StoreCurrentfile Operation 的 position，With respect to thefile Beginposition。
The return value	FS_NO_ERR: Normal
Precautions	no

示 example：

```
void main( )
{
    int         i;
    char        Buff[10];
    INT32S      pos;
    FS_FILE     *pFile;

    //假设 TEST.TXTfile Content：0123456789。10 个 Bytesdata
pFile = FS_FOpen(“C:\\TEST.TXT”, “r”);        // With the only way to Turn on a Remote Read
     file
    if(pFile == NULL)
    {
        printf(“Turn on Remote file Failure!”);
        return;
    }

    for(i=0; i<10; i++)
    {
        Buff[i] = 0;
    }
```

```
FS_FSeek(pFile, 2);                    // Set Bitfile Operationposition the position in the first 2Bytes
    FS_FRead(pFile, Buff, 5);
    printf("Read str: %s\n", Buff);          // Print    data

    FS_FTell(pFile, &pos);                   //GetCurrentfile    Operation 的 position
    printf("Current file pos: %d\n", pos);

    FS_FClose(pFile);                  //Shut downfile
}
```

1.4.11. FS_MkDir ( )

Function Name	INT8U _stdcall FS_MkDir(const INT8S *pDirName);
Functional Description	Creatingdirectory
Input Parameters	pDirName: Creatingdirectory 的 Path
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.12. FS_RmDir ( )

Function Name	INT8U _stdcall FS_RmDir(const INT8S *pDirName);
Functional Description	Deletedirectory
Input Parameters	pDirName: Deletedirectory 的 Path
Output parameters	no

The return value	FS_NO_ERR: Normal
Precautions	no

example：

```
void main( )
{
    if(FS_MkDir(“C:\\PARA”) != FS_NO_ERR)
    {
        printf(“CreatingdirectoryFailure！\n”);
        return;
    }

    if(FS_RmDir(“C:\\PARA”) != FS_NO_ERR)
    {
        printf(“DeletedirectoryFailure！\n”);
    }
}
```

1.4.13. FS_FError ( )

Function Name	INT8U _stdcall FS_FError(INT8U *pErr);
Functional Description	GetCurrentfile 系统的 ErrorState。
Input Parameters	no
Output parameters	pErr: StoreCurrentfile ErrorState system
The return value	FS_NO_ERR: Normal
Precautions	no

1.4.14. FS_ClearErr ( )

Function Name	INT8U _stdcall FS_ClearErr(void);

Functional Description	Clear Remote Equipmentfile ErrorState system.
Input Parameters	no
Output parameters	no
The return value	FS_NO_ERR: Normal
Precautions	no

1. PMC Development Library Use navigation

2.1 PMC Library Functions Overview

PMCFunctions library designed to meet a certain PC users to develop Use application software development capabilities , the user interface by calling Functions, that is likely to be completed with the next Communications Bit machine control functions.

Windows dynamic link library under "PMCLib.DLL" is based on the VC environment made knitting Write , currently supports Use in MFC programming environment.

Specific call as follows:

(1) Create a new project ;

(2) the development package "PMCLib.lib" and "PMCLib.h" file copied to the Path of new projects ;

(3) On the New Project " work area" of "file view", right-click the mouse Select "Add File to Project", the insert file dialog box , file TypeSelect for "Library Files (.lib)", found " PMCLib.lib "file and Select, click on " OK ", complete Load static library ;

(4) stated in the desired call source file or header file in the library Functions interface plus #include "PMCLib.h";

2.1 Programming Development Highlights

PMC Library Functions main achievement of Communications protocols and data access package , all

in the opening statement of APIFunctions "PMCLib.h" file in the conduct of the function and Parameters of Description.

COMM_ParaInit

The Functions must be called, which is used in the library Functions Variables used to initialize Operation.

Initialize and Turn onSerialFunctions:

COMM_UartInit

The Functions realize Serial initialization Operation, then through the Serial Connection of Equipment for the terminal data of Communicationsaccess after the call was successful.

Network ConnectionFunctions:

COMM_NetConnect

The Functions achieve TCP / UDP 's Connection of Operation, you can through the network to the Connection of Equipment terminal data of Communicationsaccess after the call was successful.

Shut downCommunications Port Functions:

COMM_ClosePort

Calling the Functions can be achieved on the Serial and network ports Shut down function , after all the success and the Communications Equipment Connection port is disconnected Connection, and the Functions and COMM_UartInit and COMM_NetConnectCorresponding of Use.

Close the NetworkEquipmentConnectionFunctions:

COMM_CloseNetConn

The network ConnectionFunctionsCOMM_NetConnect FunctionsOnly for pairing Use, which can be invoked by a single Shut down so as not to affect the network Connection Connection to other networks. And COMM_ClosePort there is a difference .

Timeout and the number of retransmissions Functions:

COMM_SetTimesOut

Because of Communications data Communications anomalies may occur , then the need to establish a data retransmission mechanism to ensure reliable transmission of data , since TCPConnection itself is a reliable Connection, it is not practical to TCPConnection the Functions of Communications process. If the program does not call CommunicationsTimeout the Functions for Current port and set the number of retransmissions , then the library Functions will follow the default Timeout2000ms and manage the number of retransmissions exception mechanism of three times.

CommunicationsConnection Set Functions

WORK_SelectComm

PARA_SelectComm

FS_SelectComm

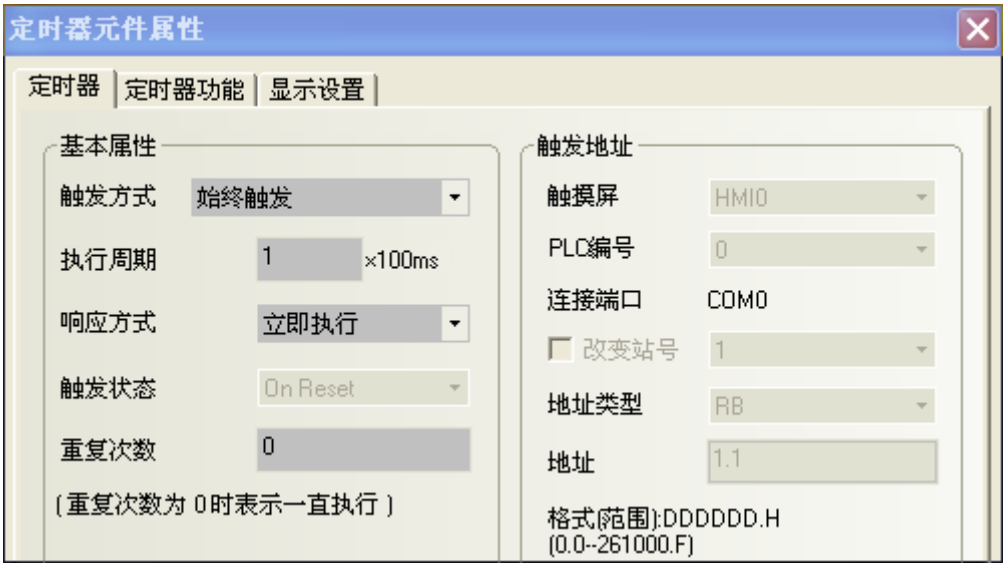
In order to facilitate the management of the register of Communications access is divided into three functional Type, processing TypeFunctions available to all Address of register data for independent access, including Consecutive, Non-Consecutive word Address, BitAddress of access. Functions of the function block can access other supports the standard Modbus protocol Equipment. ParametersTypeFunctions is extended out in a standard word Addressregister access data on the basis of an interaction mechanism through Functions area can easily be under Bit machine out ParametersInformationGet open to all , with Structure approach to storage management for each ParametersInformation, easy Use programming . file TypeFunctions with ParametersTypeFunctions as on file of ReadWriteOperation made some Functions package. ForParameters and file Type of FunctionsUse, currently limited to part time with the company 's products for data interaction Use

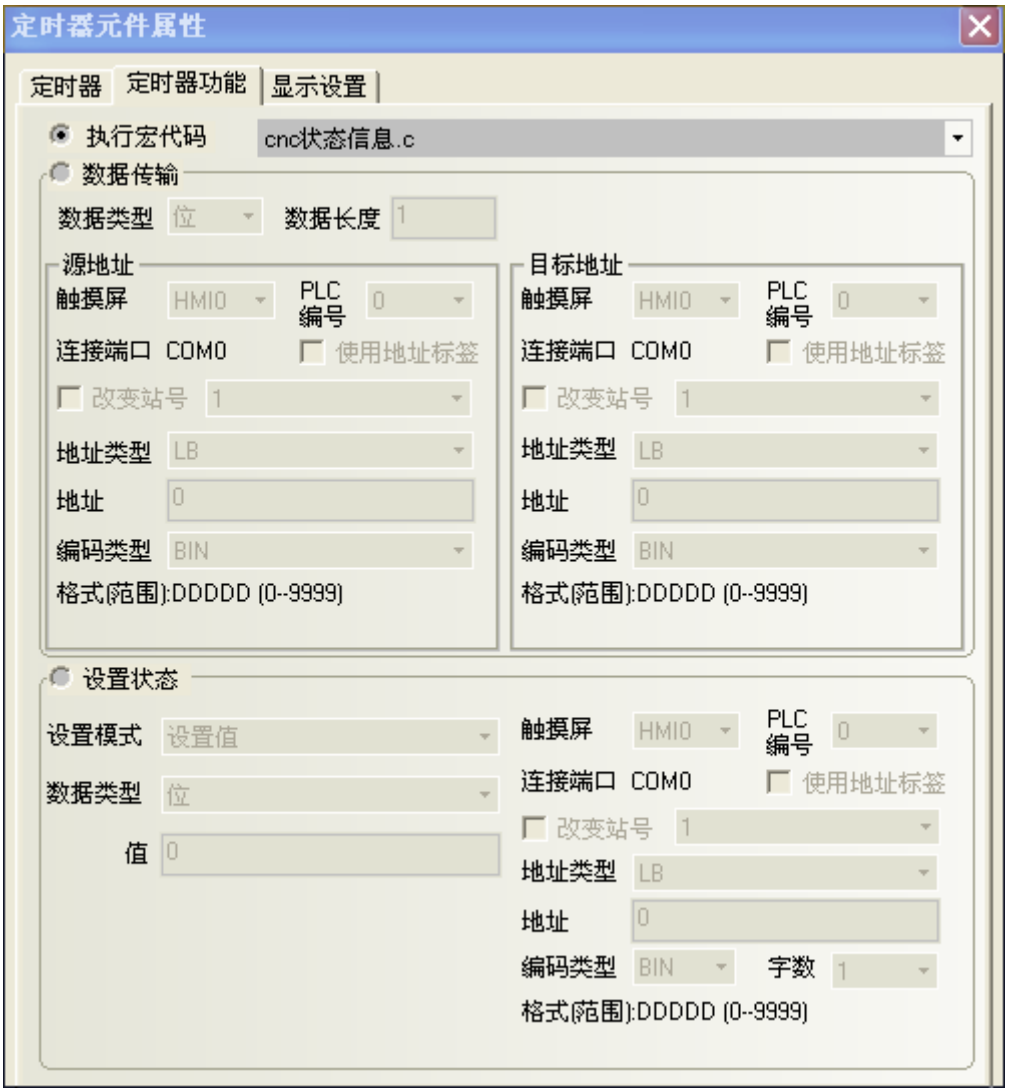
Functions can be three or more of its three Type Communications be administered separately , so that when constructing the computer program can For Bit of Shut down of a certain TypeFunctions ConnectionEnable, Communications in order to achieve effective utilization of resources .

The first chapter of industrial touch screen programming papers

1 Kinco real example of industrial touch screen

1.1 Timer control settings





Timer is to perform timing device , when the regular time to be executed after the corresponding function , scheduled to be completed macros and data transfer functions. Here timer function settings Select the macro code execution functions , macros, code program "cncStateInformation.c" code as follows:

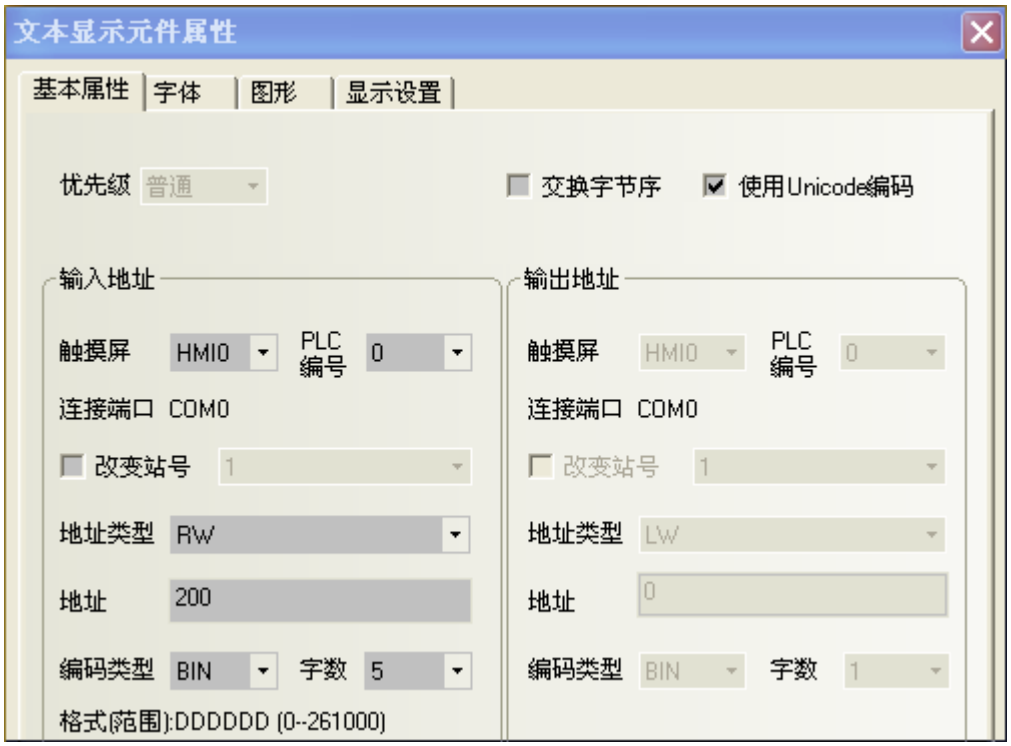
```
24
25  const short ModeSta[][5] = {
26      {0x5f55, 0x5165},  //录入
27      {0x81ea, 0x52a8},  //自动
28      {0x624b, 0x52a8},  //手动
29      {0x624b, 0x8f6e},  //手轮
30      {0x5f52, 0x96f6},  //归零
31      {0x7f16, 0x8f91},  //编辑
32      {0x624b, 0x8f6e, 0x7a7a, 0x8dd1}, //手轮空跑
33      {0x5355, 0x6b65},  //单步
34    };
35
36  const short RunSta[][5] = {
37      {0x505c, 0x6b62},                  //停止
38      {0x6b63, 0x5728, 0x8fd0, 0x884c},  //正在运行
39      {0x6682, 0x505c},                  //暂停
40      {0x5355, 0x6bb5, 0x505c, 0x6b62},  //单段停止
41    };
42
43  int MacroEntry()
44  {
45    if(CNCRunMode < 8)
46    {
47      WriteLocal("RW", 200, 5, (void*)ModeSta[CNCRunMode], 0);  //当前模式状态信息
48    }
49
50    if(CNCRunSta < 4)
51    {
52      WriteLocal("RW", 205, 5, (void*)RunSta[CNCRunSta], 0);     //当前运动状态信息
53
54      if(CNCRunSta != 1)
55      {
56        LedSta = 1;          //设置指示灯为停止状态
57      }
58      else
59      {
60        LedSta = 0;          //设置指示灯为运行状态
61      }
62    }
63
64    WriteLocal("RW", 112, 1, (void*)(&CNCProgNo), 0);       //加工程序号
65    WriteLocal("RW", 113, 8, (void*)(&CNCProgName[0]), 0); //加工文件名
66
67    return 0;
68  }
69
```

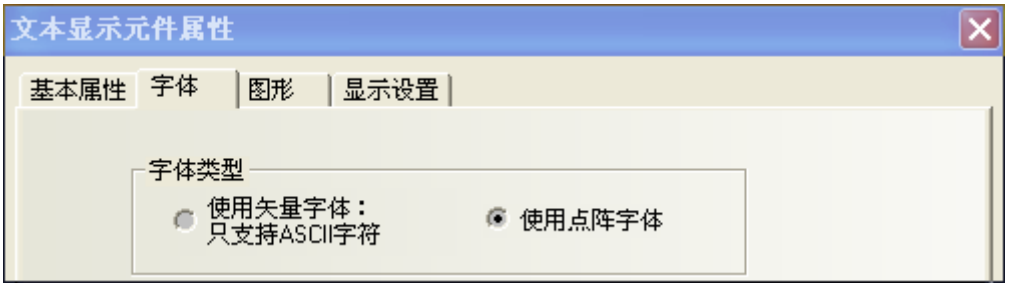
宏代码变量窗口[cnc状态信息.c]

数据类型	变量名	PLC编号	地址类型	地址	字长	操作属性	是否数组	数组长度
bit	LedSta		RB	50.0	1	读/写	否	
unsigned short	CNCRunMode	0	4X	3907	1	读	否	
unsigned short	CNCRunSta	0	4X	3933	1	读	否	
unsigned short	CNCProgName	0	4X	3923	1	读	是	8
unsigned short	CNCProgNo	0	4X	3922	1	读	否	

The macro program to achieve the right sport Mode, sports StateInformation and processing procedures No., processing file name data transmission , etc. is displayed . Variables window shows the macro code manually add the macro VariablesInformation, some macro Use of Variables need to be declared here Use.

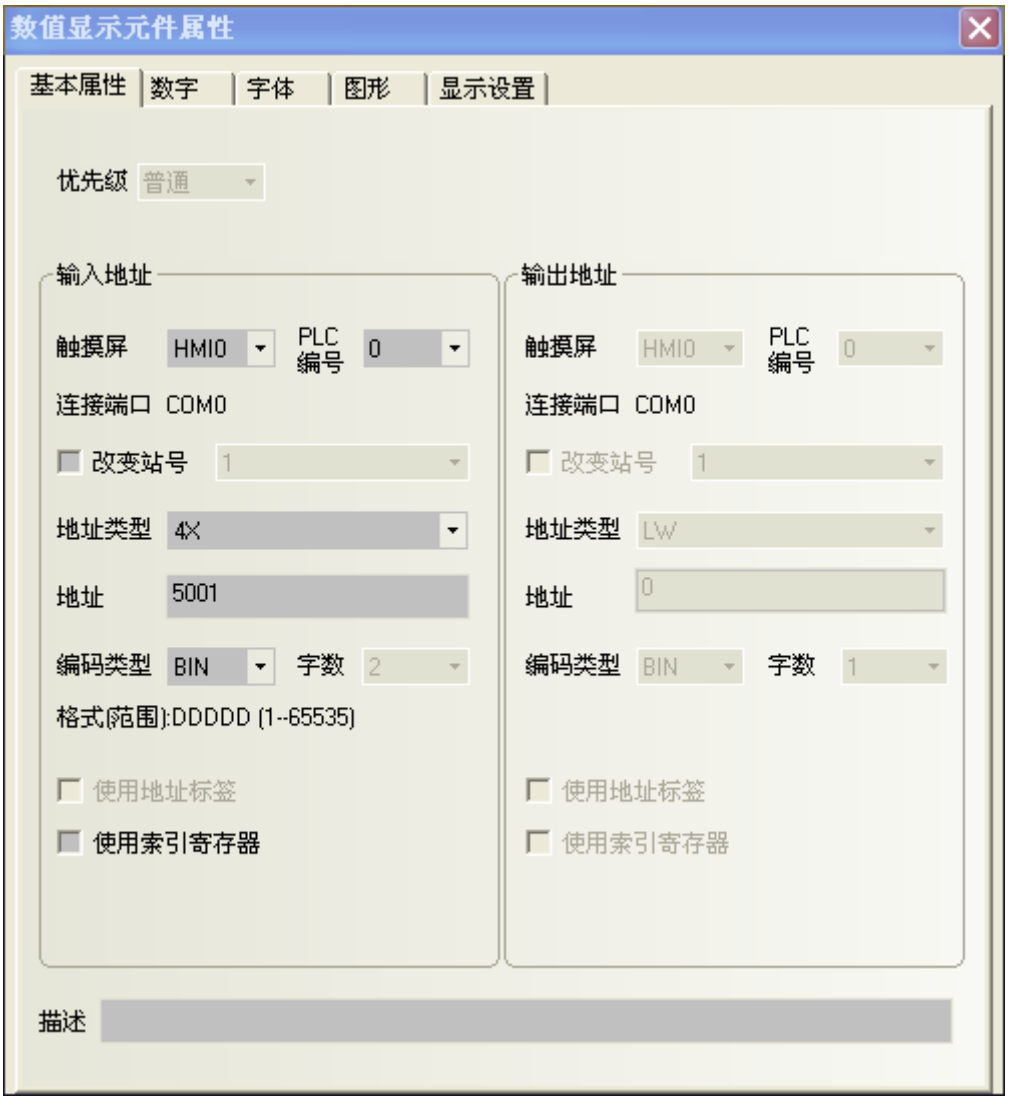
1.1. Information display element control settings

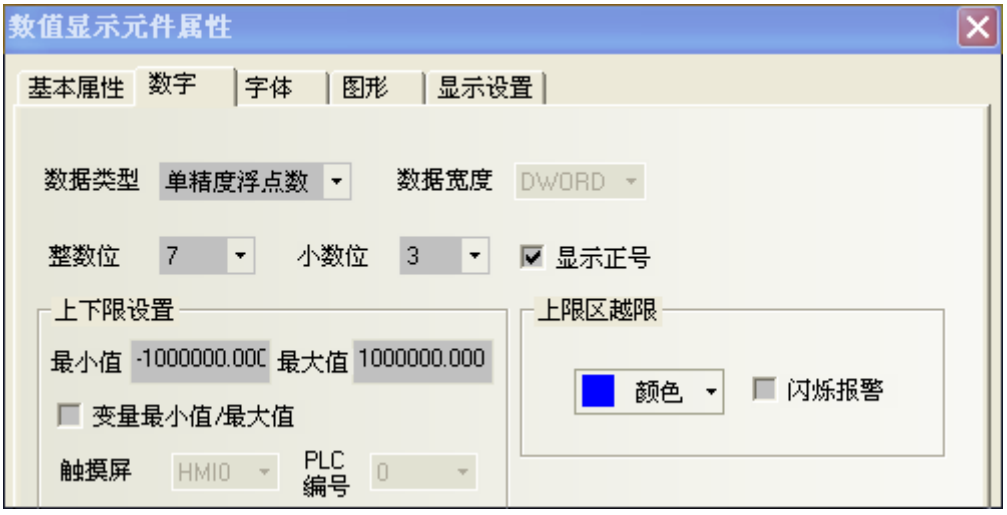




Run ModeInformation box to display ModeInformation Current running , InformationContent text Format, storage Value accounted for most Multi- 10 Bytes, so here Select "RW" AddressType, StartingAddress for 200 words Select5 months . UseUnicode coding, dot matrix character display. Among them , "cncStateInformation.c" macro in the first 47 lines of program execution , which is transmitted to the Address InformationContent of Operation.

1.1 -axis coordinate display control settings

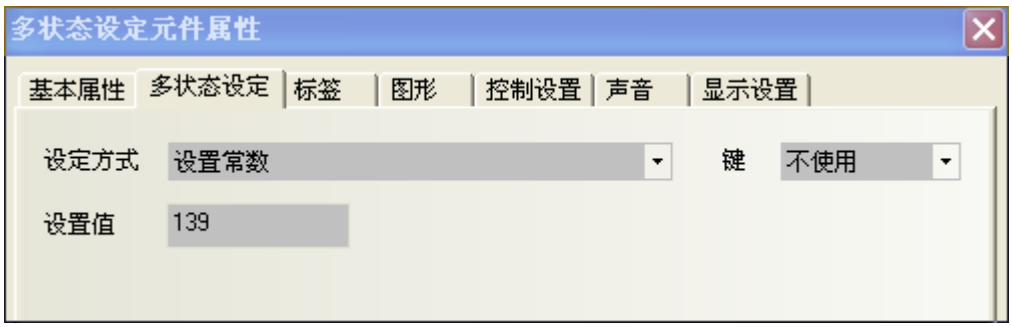




data display control settings are performed directly on the Slave ReadOperation 's , AddressTypeSelect4X, Address for 5001 (due to PLCAddress is 1Begin , so PLCAddress the agreement

to add an Address, ie, 5001) , single-precision Floating Point type data TypeSelect , the word length is 2 .

1.1. 按钮功能控件设置



Multi-State setting element setting is performed directly on the Slave WriteOperation 's , AddressTypeSelect3X ,, Address for 3906 , setting mode Select Set constant function , Variables value is pressing the button Corresponding Key ...

Notes:

About the Modbus transmission AddressType description:

■ 0X AddressRange1-65535, the motion controller for Bitregister be WriteOperation, via modbus protocol to transfer .

■ 1X AddressRange1-65535, the motion controller for Bitregister be ReadOperation, via modbus protocol to transfer .

■ 3X AddressRange1-65535, value register for the motion controller will be WriteOperation, via modbus protocol to transfer .

■ 4X AddressRange1-65535, value register for the motion controller will be ReadOperation, via modbus protocol to transfer .

Value Range AddressType located above are from 1Begin due , so when you add the Address value, the value is set to be added to the protocol Address 1.

Content is only part of the above controls were set cite exampleDescription, Properties set other controls are similar. The example source code implementation process Corresponding Store under " industrial touch screen real example package " directory. Kinco touch screen for programming Use, specifically refer to "EV5000Use Manual " Documentation, or visit the company website http://www.kinco.cn/ Kinco to understand .