Catalogue

1 Brief introduction of the function

1.1 functional overview

GCAN PLC Core - M7 is a kind of embedded PLC core modules. The module has been integrated with PLC control kernel, allows users to use ec61131 1-3 standard five programming languages for its programming, including: FBD, SFC, LD, ST, IL and support online debugging features include watching and set variables, single cycle, breakpoints and step, and other functions.

GCAN PLC Core - M7 process is very suitable for signal processing, centralized monitoring, intelligent network nodes, using distributed control system, also can be used as a big system of special components, can also be used as the core of the motion controller PLC.

GCAN PLC Core - M7 core modules already contains the whole running environment and part of the PLC storage peripherals, and its derivation will also chip in various resources, to provide up to 4 road UART, 2 road CAN, a road 100 m Ethernet, AD 8 road, DA 2 road, 40 digital input and output, two high-speed counter 2 road PWM output. Users can use more resources to develop specific procedures to collect and control.The module can also be customized as required.

1.2 PLC core board resources

- CPU:ARM32-bit cortex-M7 kernel,processor frequency216Mhz;
- 4 road UART， 2 road CAN,a road 100m Ethernet;
- 8 road AD,2 road DA;
- 40 digital input and output;
- 2 high-speed counter and 2 road PWM output;
- Double 60 pin socket(at bottom),convenience access to all kinds of back plates;
- A 5V＆3.3V test point,support external power supply or output power to external;
- A power light,a status light;
- A reset button,can use to reset MCU and LCD

1.3 typical application

- Industrial grade PLC developing;
- CAN industrial auto control system developing.

2 GCAN-PLCcore-M7 pin definition and function description

Pin definition and function of GCAN-PLCcore-M7 are explained in detail in the table below.Among them,IO means that it can use as DI and DO,X means that no use,the pin hangs in the air.

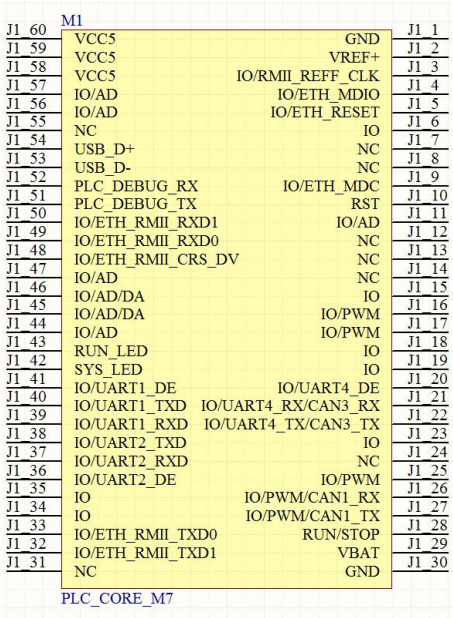
Numerical order	Pin number	Function definition	The default mapping	Numerical order	Pin number	Function definition	The default mapping
CON1				CON2			
1	P1	GND		1	P61	IO/ETH_RMII_TX_EN	
2	P2	VREF+		2	P62	IO/PWM	I0.7
3	P3	IO/RMII_REFF_CLK	ETH	3	P63	IO	I0.6
4	P4	IO/ETH_MDIO	ETH	4	P64	IO/CAN2 RX	CAN2
5	P5	IO/ETH_RESET	ETH	5	P65	IO/CAN2 TX	CAN2
6	P6	IO	Q2.0	6	P66	IO/PWM	I0.5
7	P7	x		7	P67	IO/PWM	I0.4
8	P8	x		8	P68	IO/PWM	I0.3
9	P9	IO/ETH_MDC	ETH	9	P69	IO	I0.2
10	P10	RST		10	P70	IO/PWM	I0.1
11	P11	IO/AD		11	P71	IO/PWM	I0.0
12	P12	x		12	P72	IO/PWM	I1.5
13	P13	x		13	P73	IO	Q2.7
14	P14	x		14	P74	IO	I1.6
15	P15	IO	Q2.3	15	P75	IO/UART3_DE	I2.2
16	P16	IO/PWM	I1.0	16	P76	IO/UART3_TX	I2.4
17	P17	IO/PWM	I1.1	17	P77	IO/UART3_RX	I2.3
18	P18	IO	Q2.4	18	P78	IO/PWM	I1.7
19	P19	IO	Q2.5	19	P79	IO/PWM	I2.0
20	P20	IO/UART4_DE	I2.5	20	P80	IO/PWM	I2.1
21	P21	IO/UART4_RX/CAN3_RX	Q1.0	21	P81	x	
22	P22	IO/UART4_TX/CAN3_TX	Q1.1	22	P82	x	
23	P23	IO	Q2.6	23	P83	x	
24	P24	X		24	P84	IO	Q2.2
25	P25	IO/PWM	I1.3	25	P85	IO	Q2.4
26	P26	IO/PWM	I1.2	26	P86	x	
27	P27	IO/PWM	I1.4	27	P87	x	
28	P28	RUN/STOP		28	P88	IO	Q2.5

29	P29	VBAT		29	P89	IO/CAN_RX	CAN1 RX
30	P30	GND		30	P90	IO/CAN_TX	CAN1 TX
31	P31	x		31	P91	GND	
32	P32	IO/ETH_RMII_TXD1	ETH	32	P92	IO	Q2.3
33	P33	IO/ETH_RMII_TXD0	ETH	33	P93	IO	Q2.1
34	P34	IO	Q1.6	34	P94	IO	Q0.0
35	P35	IO	Q1.7	35	P95	IO	Q0.1
36	P36	IO/UART2_DE	RS485 EN	36	P96	IO	Q0.2
37	P37	IO/UART2_RXD	RS485 RX	37	P97	IO	Q0.3
38	P38	IO/UART2_TXD	RS485 TX	38	P98	IO	Q0.4
39	P39	IO/UART1_RXD	RS232 RX	39	P99	IO	Q0.5
40	P40	IO/UART1_TXD	RS232 TX	40	P100	IO	Q0.6
41	P41	IO/UART1_DE	Q1.2	41	P101	IO	Q0.7
42	P42	SYS_LED	SYS LED	42	P102	IO/AD	I9.0 AD
43	P43	RUN_LED	RUN LED	43	P103	x	
44	P44	IO/AD	I7.0 AD	44	P104	x	
45	P45	IO/AD/DA	I13.0 AD	45	P105	x	
46	P46	IO/AD/DA	I11.0 AD	46	P106	x	
47	P47	IO/AD	I5.0 AD	47	P107	x	
48	P48	IO/ETH_RMII_CRS_DV	ETH	48	P108	x	
49	P49	IO/ETH_RMII_RXD0	ETH	49	P109	x	
50	P50	IO/ETH_RMII_RXD1	ETH	50	P110	x	
51	P51	PLC_DEBUG_TX	PLC DEBUG TX	51	P111	x	

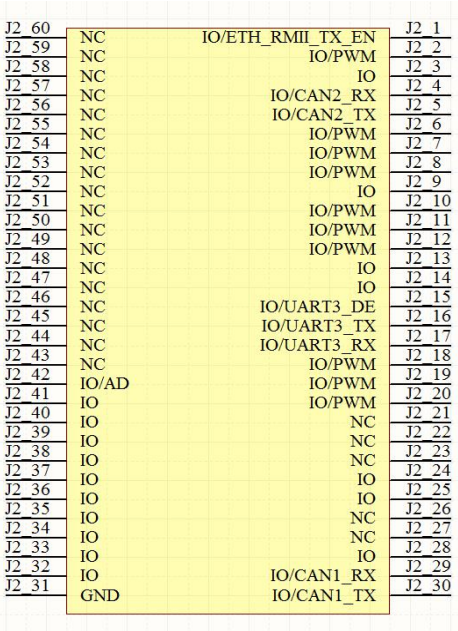
52	P52	PLC_DEBUG_RX	PLC DEBUG RX	52	P112	x	
53	P53	IO/CAN1_RX		53	P113	x	
54	P54	IO/CAN2_TX		54	P114	x	
55	P55	x		55	P115	x	
56	P56	IO/AD	I5.0	56	P116	x	
57	P57	IO/AD	I3.0 AD	57	P117	x	
58	P58	VCC5		58	P118	x	
59	P59	VCC5		59	P119	x	
60	P60	VCC5		60	P120	x	

GCAN-PLC core-M7 core board use“ 3710F terminal”type connector connection to baseboard,if guest want to development baseboard,the baseboard refer to schematic diagram and pin definition as below picture,specific connection schematic diagram and packing can be recieved from technical support engineer.

CON1:



CON2：



3 GCAN-PLCcore-M7 start strategy

By default, GCAN PLC core - M7 when power on or restart to load all the necessary firmware program, will begin to run after the PLC program. Therefore, GCAN PLC core - M7 is suitable for the control system used in independent model. In power under the condition of collapse, this system can resume execution PLC program, without user intervention. The following illustration shows the details of the system boot:

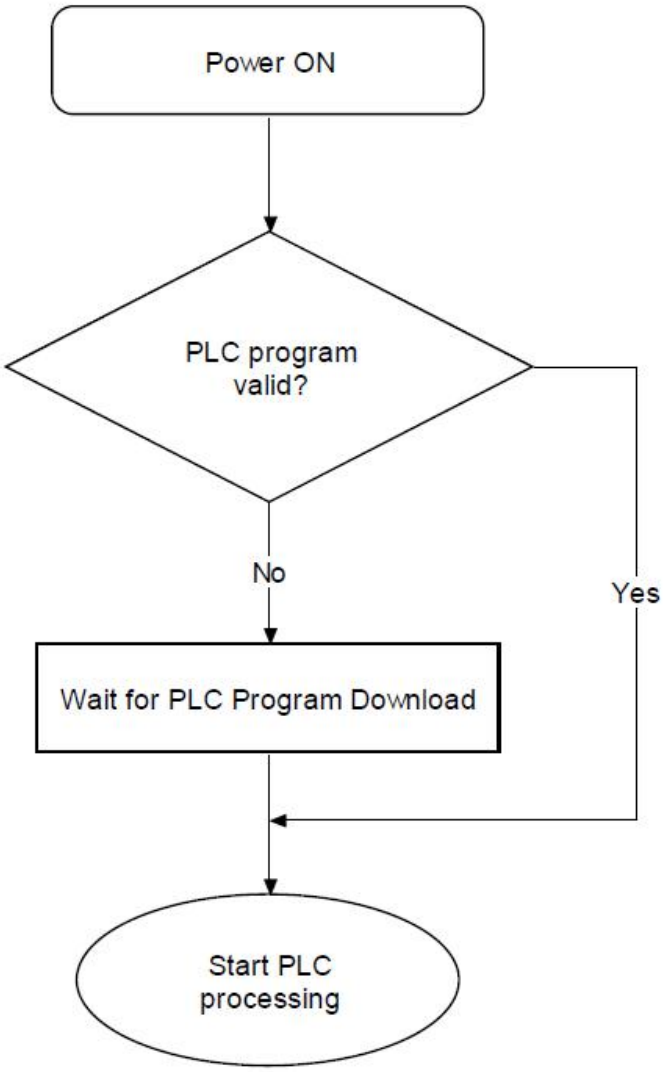


Figure 4.1 GCAN-PLCcore-M7 start strategy

If the firmware in the nonvolatile memory found effective control program, the program will restart and processing. Otherwise will activate the program, enter GCAN PLC core - M7 stop mode.

4 GCAN-PLCcore-M7 core board argument list

Controller character	
controller	STM32F7
core	ARM® 32-bit Cortex®-M7
Processor frequency	216MHz
Procedure memory space	32M bytes
RAM	32M bytes
Data memory space	128M bytes
Interface character	
Ethernet	1 road，10/100Mbps
RS232/RS485	4 roads，600bps~115200bps
CAN connector	2 roads，follow ISO 11898 standard，support CAN2.0A/B
Digital quantity input/output	50roads DI/DO（reuse）
Analog input	6 roads（reuse）
Analog output	2 roads（reuse）
The onboard interface	Double 60pin outlet，0.8mm lead pin pitch
software	Internal integration IEC61131-3real time kernel
Programmatic interface	RS232、TCP（choosable）、CAN（choosable）
RTC	On-board real-time clock, external clock battery is needed
power supply	
supply voltage	+5V DC（±5%）
supply current	200mA
environment test	
operating temperature	-40℃~+85℃
operating humidity	15%~90%RH，without condensation
EMC test	EN 55024:2011-09 EN 55022:2011-12
evel of protection	IP 20
basic information	
boundary	65mm *45mm

dimension	
weight	10g

5 PLCCore-EDV-M7 development kits

PLCcore EDV - M7 development kit is a set of high-performance low-cost development kit. Based on GCAN PLCcore - M7 core board, development kit PLCcore EDV - M7 gives users fast spread, network compatible automation project. In addition, it helps users understand based on IEC 61131-3 PLC programming advantages compared with the traditional programming language.

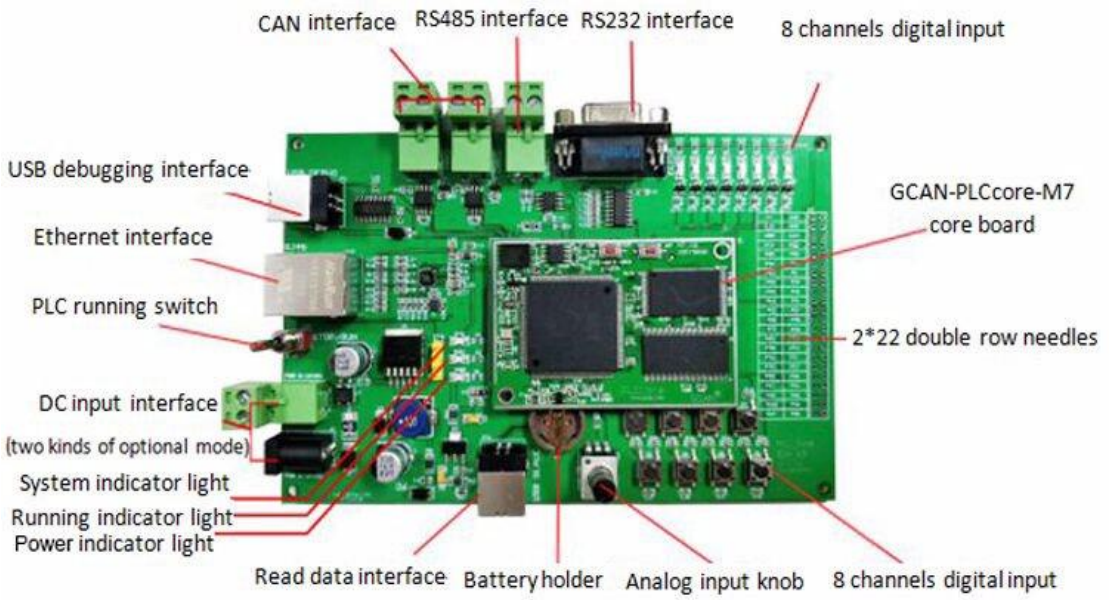


Figure 6.1 plccore-edv-m7 development suite

GCAN PLC Core - M7 core plate supporting development board floor on-board resources are as follows:

- 1 core board interface, support GCAN PLC Core - M7 core board;
- One power light, two status indicator light;
- 2 way CAN interface, Ethernet interface 1 road;
- 1 road RS232 interface, RS485 interface 1 road;
- Road on the 8th of road digital quantity input, digital output;
- USB serial port 1 road, can be used to download PLC program and debug the code;
- 1 USB Slave, can be used in the PLC user data read and write;
- A dc power input interface (input voltage range: DC9-30 v);
- RTC back-up battery holder;
- A switch for PLC RUN and STOP;
- 1 set of 2 x22 double row needle, used to draw out the rest of the pin inside the core plate;
- Lead to 2 x22 interface, including power supply and 10 interface, meet various application requirements;
- Dimension is 150 mm * 100 mm.

6 PLCCore-EDV-M7 Development suite parameters

PLCCore EDV - M7 development kit (GCAN - PLCCore - M7 core plate supporting plate) technical parameters as below figure:

Core plate characteristics	
core-board	GCAN-PLCCore-M7
core	ARM® 32-bit Cortex®-M7
Interface characteristics	
Ethernet	1 road，10/100Mbps
RS232	1 road，600bps~115200bps
RS485	1 road，600bps~115200bps
CAN connector	2 roads，follow ISO 11898 standard，support CAN2.0A/B
CAN Baud rate	1000K、500K、250K、200K、125K、100K、50K、20K
digital input	8 roads DI，keystroke
Number quantity input	8 roads DO，LED
The onboard interface	Double 60 pin outlet，0.8mm lead pin pitch
programmatic interface	RS232
RTC	On-board real-time clock, external clock battery is needed
power supply	
service voltage	+9-30V DC
environment test	
operating temperature	-40℃~+85℃
operating humidity	15%~90%RH，without condensation
EMC test	EN 55024:2011-09 EN 55022:2011-12
level of protection	IP 20
essential information	
boundary dimension	150mm *100mm
weight	100g

7 PLC Core EDV - M7 development suite basic routines

PLC Core EDV - M7 development kit provides the following the code routines, through these routines users can quickly familiar with GCAN - PLC Core - the use of M7 and its features, users can quickly develop their own projects.
Experiment 1 entertaining diversions experiment
Experiment 2 input/output
Experiment 3 serial experiments
Experiment 4 can communication
Experiment 5 TCP Server

7.1 6 Modbus TCP experiments

This experiment used to implement GCAN - PLC Core Modbus TCP from station function, the function of the involved mainly contains 0203 04 '05, Modbus TCP data format can be found in the appendix "table b. 2" Modbus TCP response data format.
A. Download the program
Please refer to "9.3.4 set debug connection" and "9.3.5 download program and debug program download.After downloading the program, you need to click RESET button above the GCAN - PLC Core or back to GCAN - PLC Core development board to electricity, make the new program to take effect.
B. The main function block This experiment used in the function block has: LANINIT, LANASCILTOINET, Modbus TCPSLAVE INIT, Modbus SLAVE CTRL four function block. Their functions are shown in table below. Function block prototype, operand definition and description information, please refer to "GCAN - PLC extended gong Ming book".

functional device	function
LAN_INIT	Used to initialize the Ethernet interface.
LAN_ASCII_TO_INET	Converts the ASCII form of the IP address to the integer form of the IP address.
Modbus_TCP_SLAVE_INIT	Used to initialize the Modbus TCP slave interface.
Modbus_SLAVE_CTRL	Used to execute control instructions for the Modbus TCP slave station interface.

Table 8.1 functional blocks used in Modbus TCP experiment

C.Program description
Open the test.st file and see the following code：

```
if lanInit=false then

mIP := LAN_ASCII_TO_INET('192.168.1.31');
mNetMask := LAN_ASCII_TO_INET('255.255.0.0');
```

```
mGateWay := LAN_ASCII_TO_INET('192.168.1.1');
    inst0_LAN_INIT(
            ENABLE :=true ,
            HOSTNAME := 'PLCCore',
            IP := mIP,
NETMASK :=mNetMask,
            GATEWAY :=mGateWay,
            NETNUMBER:=1|mConfirm:= CONFIRM,
            mError:= ERROR,
            mErrorinfo:= ERRORINFO
             );
    lanInit:=true;
    else
        if mModbusInitOK=false then

            inst0_MODBUS_TCP_SLAVE_INIT(
            ENABLE := true,
            MODE := 1,
            ADDRESS :=1 ,
            NETNUMBER :=1   |   mConfirm:= CONFIRM,
            mError:= ERROR,
            mErrorinfo:= ERRORINFO
            );
            mModbusInitOK:=true;
        end_if;
    end_if;
```

This code is used to initialize the Ethernet interface and Modbus TCP interface from the station. Users can set the IP, NETMASK. These three parameters, GATEWAY Ethernet connection is established. Through the MODE is set to 1 defines the Port as the Modbus protocol and make the deal, by putting the MODE is set to 0 defines the Port as PPI and banned the Modbus protocol; Through the ADDRESS parameter setting the site ADDRESS on TCP.

```
VAR
lanInit:bool:=false;
inst0_LAN_INIT:LAN_INIT;
mConfirm:bool;
mError:usint;
mErrorinfo:usint;
mIP:udint;
mNetMask:udint;
mGateWay:udint;
mSocket:INT;
inst0_MODBUS_TCP_SLAVE_INIT:MODBUS_TCP_SLAVE_INIT;
mModbusInitOK:bool:=false;
```

```
modbusDOBuf : ARRAY[0..5] OF byte;
DO_Ptr : POINTER;
modbusDIBuf : ARRAY[0..5] OF byte;
DI_Ptr : POINTER
modbusAIBuf : ARRAY[0..10] OF int;
AI_Ptr : POINTER;
modbusRegBuf : ARRAY[0..127] OF int;
mPtr : POINTER;
xDONE:BOOL;

inst4_MODBUS_TCP_SLAVE_CTRL:MODBUS_TCP_SLAVE_CTRL;
xError:usint;
mDI at %I0.0:byte;
mAI at %I1.0:int;
mDO     at %Q0.0:byte;
END_VAR
 mDO:=modbusDOBuf[0];
modbusDIBuf[0]:=mDI;
modbusAIBuf[0]:=mAI;
modbusRegBuf[0]:=byte_to_int(mDI);      (*The input value of modbus digital
                                          quantity is stored in*)
                                              (*Modbus Keep the
                                              address in the register*)
modbusRegBuf[1]:=mAI;                   (*Analog input value*)
modbusRegBuf[2]:=byte_to_int(mDO);      (*digital quantity*)

mPtr:=&modbusRegBuf;
DO_Ptr:=&modbusDOBuf;
DI_Ptr:=&modbusDIBuf;
AI_Ptr:=&modbusAIBuf;

if mModbusInitOK=true then
    inst4_MODBUS_TCP_SLAVE_CTRL(
        NETNUMBER :=1,
        DO_ENABLE :=1 ,
        DO_PTR :=DO_Ptr ,
        DO_LENGTH :=5 ,                 (*Please note the length limitation*)
        DI_ENABLE :=1 ,
        DI_PTR :=DI_Ptr ,
        DI_LENGTH :=5 ,
        AI_ENABLE :=1 ,
        AI_PTR :=AI_Ptr ,
        AI_LENGTH :=10 ,
        REG_ENABLE :=1 ,
```

```
            REG_PTR :=mPtr ,
            REG_LENGTH :=127
            | xDONE := DONE,
            xError := ERROR);
    end_if;
```