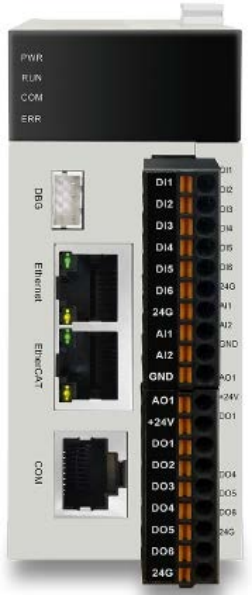

PWR
RUN
COM
ERR
DBG
Ethernet
EtherCAT
COM
DI1
DI2
DI3
DI4
DI5
DI6
24G
AI1
AI2
GND
AO1
+24V
DO1
DO2
DO3
DO4
DO5
DO6
24G

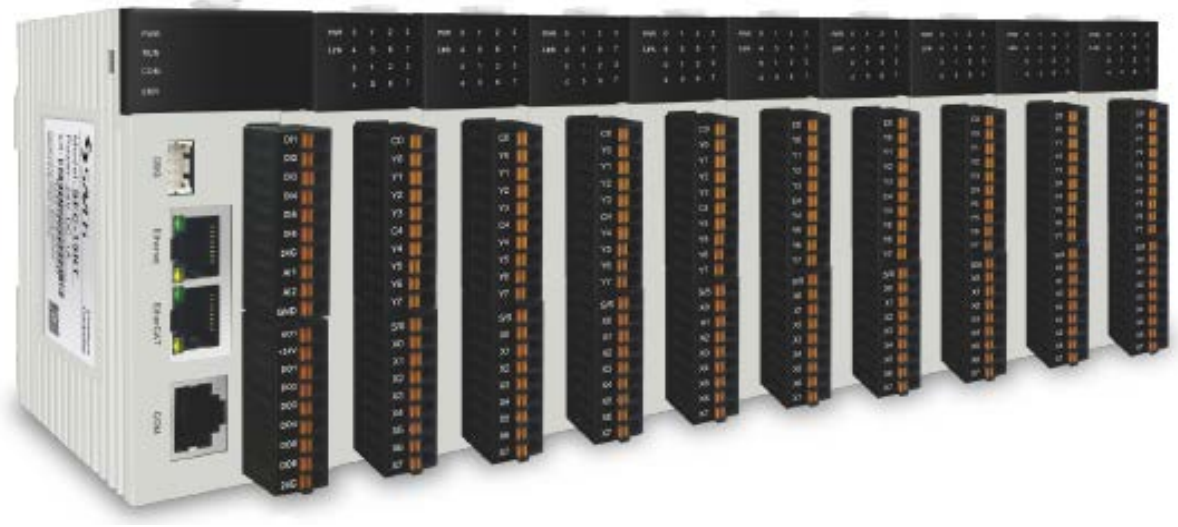
SFC产品介绍

目录

SFC产品介绍 之 目录

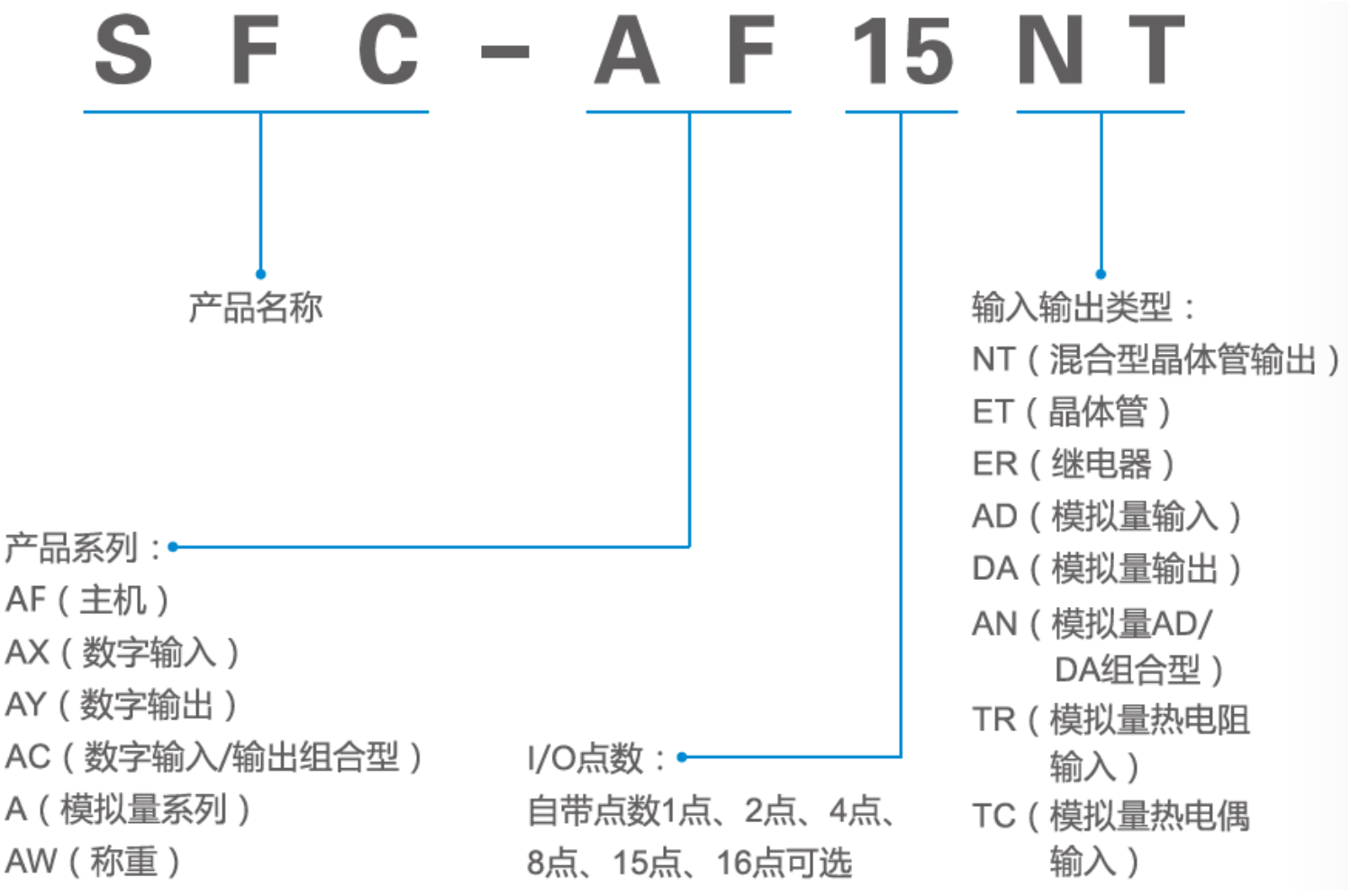
1.1、SFC介绍 之 SFC简介

SFC-AF15NT是三碁电气研发的一款以现场总线通信为核心的可编程控制器。双ARM处理器，总线通信与逻辑控制各司其职。支持EtherCAT、EtherNet、CAN、RS485等工业现场总线。以主机加扩展模块的方式进行硬件扩展，最大可扩展15块模块，极大的方便了控制器在不同点数需求、不同行业的系统中应用。C语言二次应用开发，效率高，保密性好。



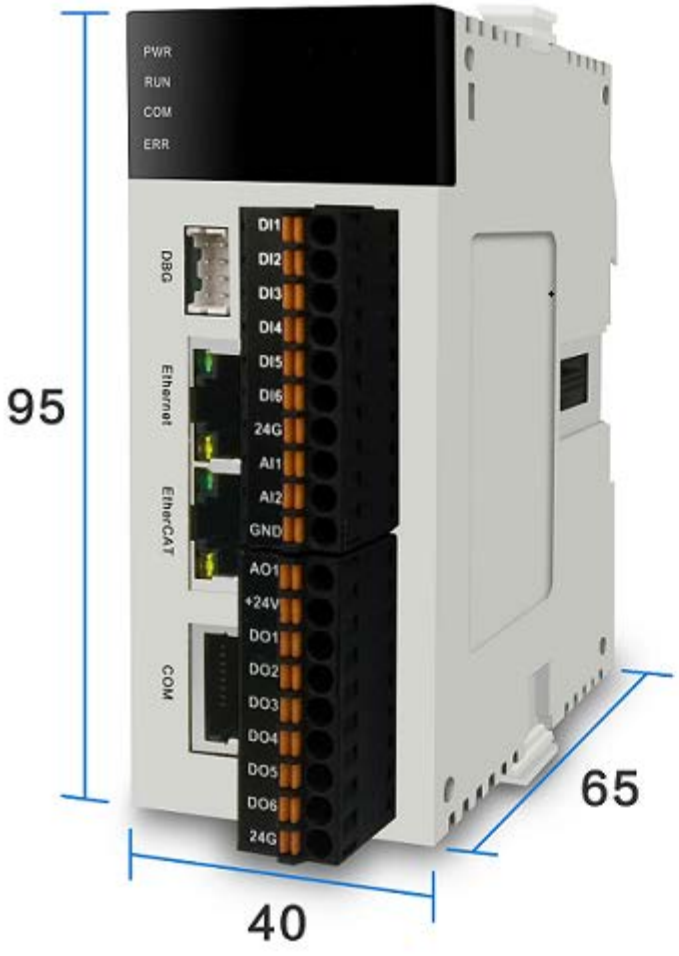
1.2、SFC介绍 之 产品信息

1.规格型号

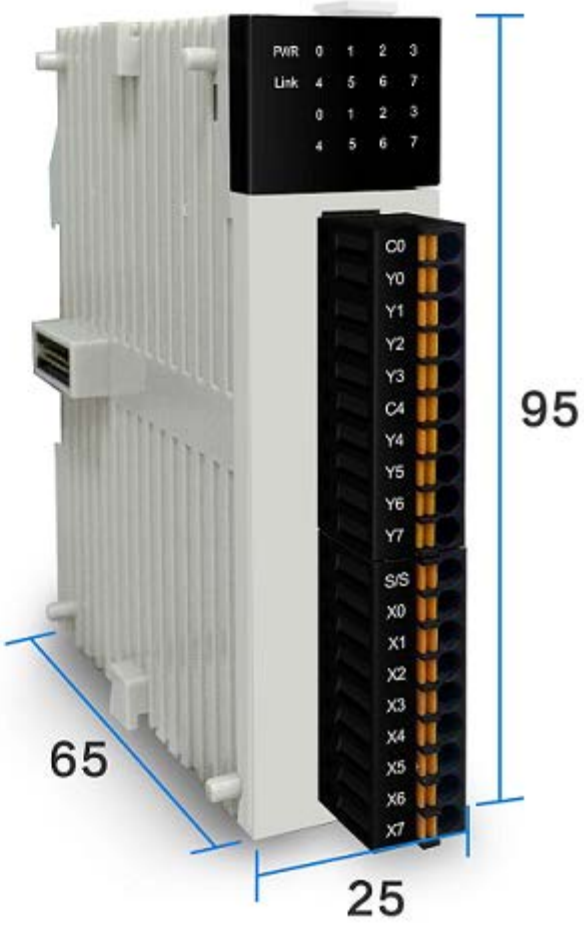


1.3、SFC介绍 之 外形尺寸

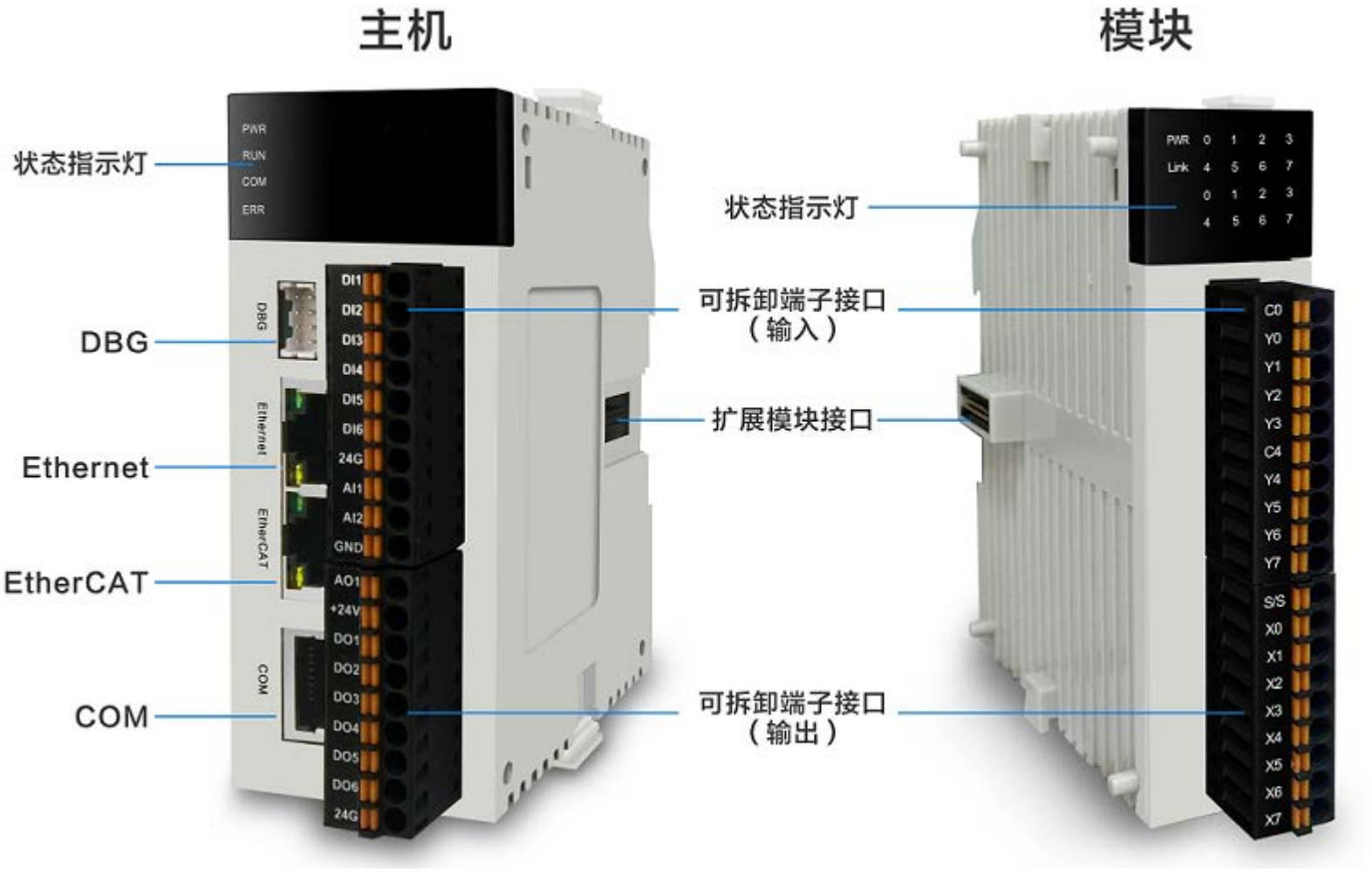
主机



模块



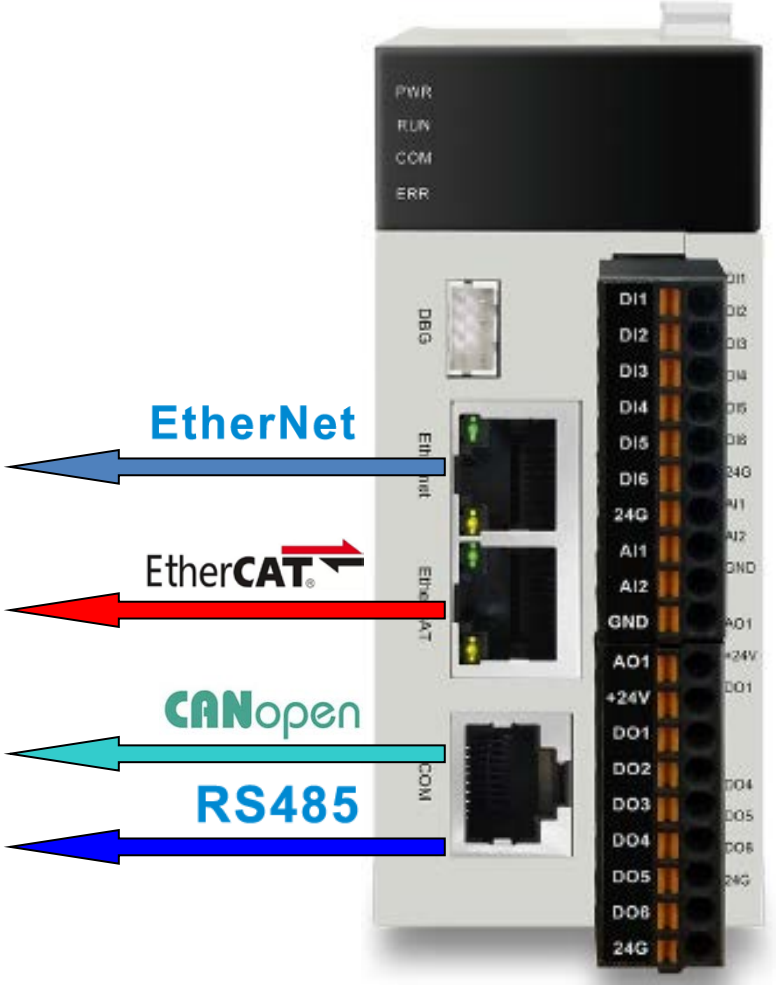
1.3、SFC介绍 之 外形尺寸



1.4、SFC介绍 之 接线图

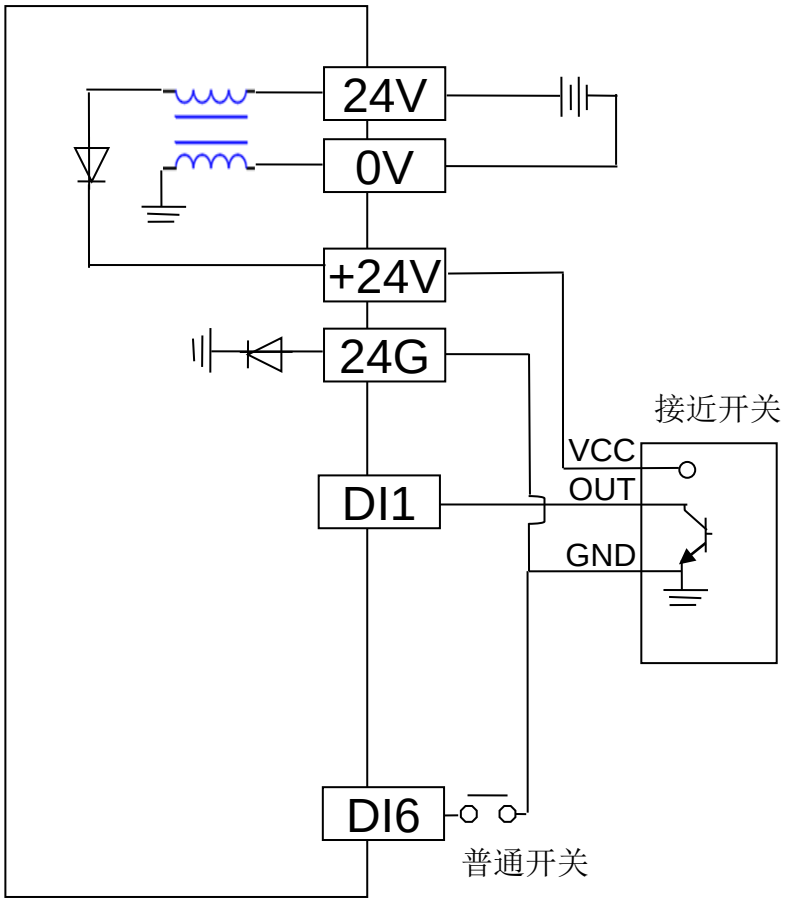
1.主机总线接线示意图

名称		PIN	总线	协议	主从站	连线
Ethernet		-	EtherNet	Modbus-Tcp	从站	普通网线
EtherCAT		-	EtherCAT	CoE、CiA402	主站	屏蔽双绞线
COM	1	CANH	CAN	CiA301、CiA402	主站	屏蔽双绞线
	2	CANL				
	3	GND	-	-	-	-
	4	DX-	RS485	Modbus-Rtu	主站	屏蔽双绞线
	5	DX+				
	6	GND	-	-	-	-
	7	SG-	RS485	Modbus-Rtu	从站	屏蔽双绞线
	8	SG+				

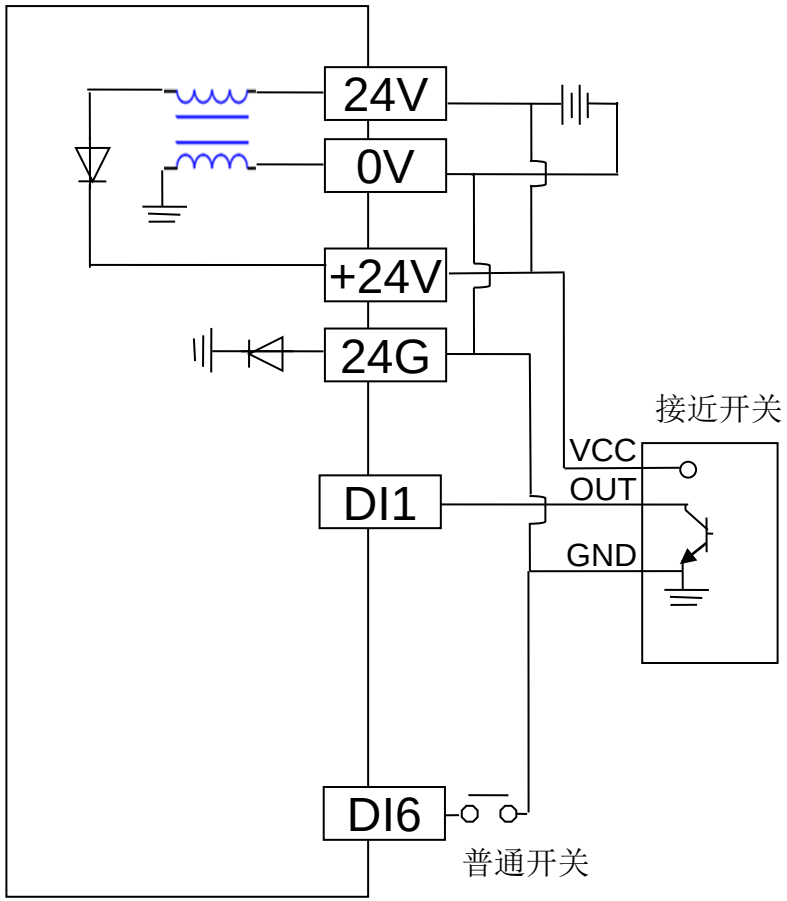


1.4、SFC介绍 之 接线图

1.主机数字量输入接线示意图



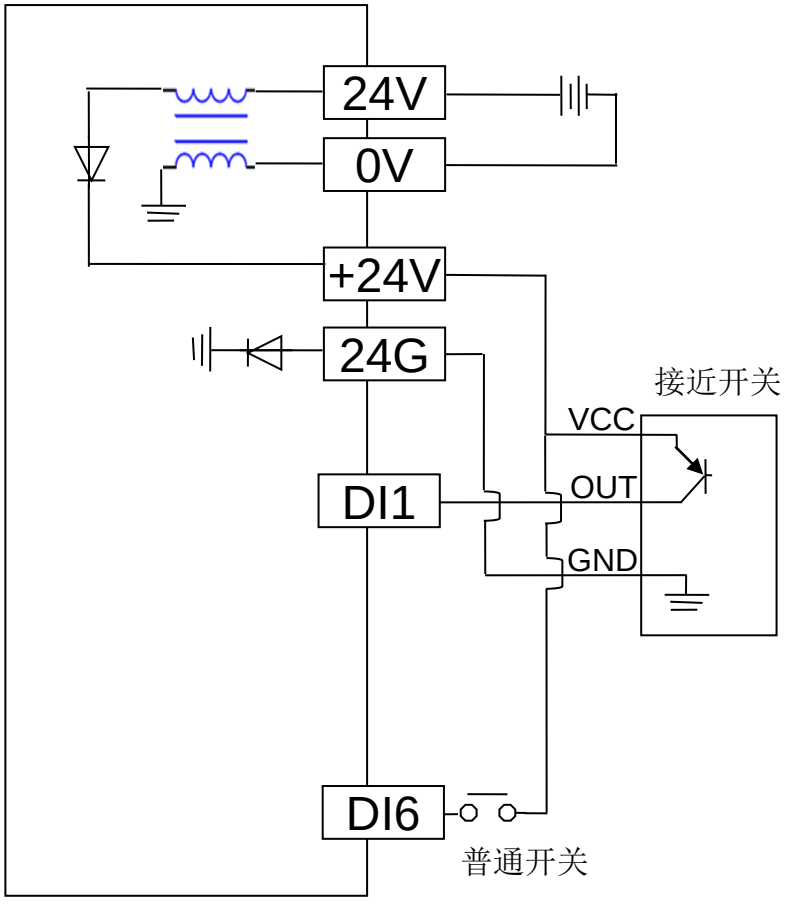
NPN内部供电



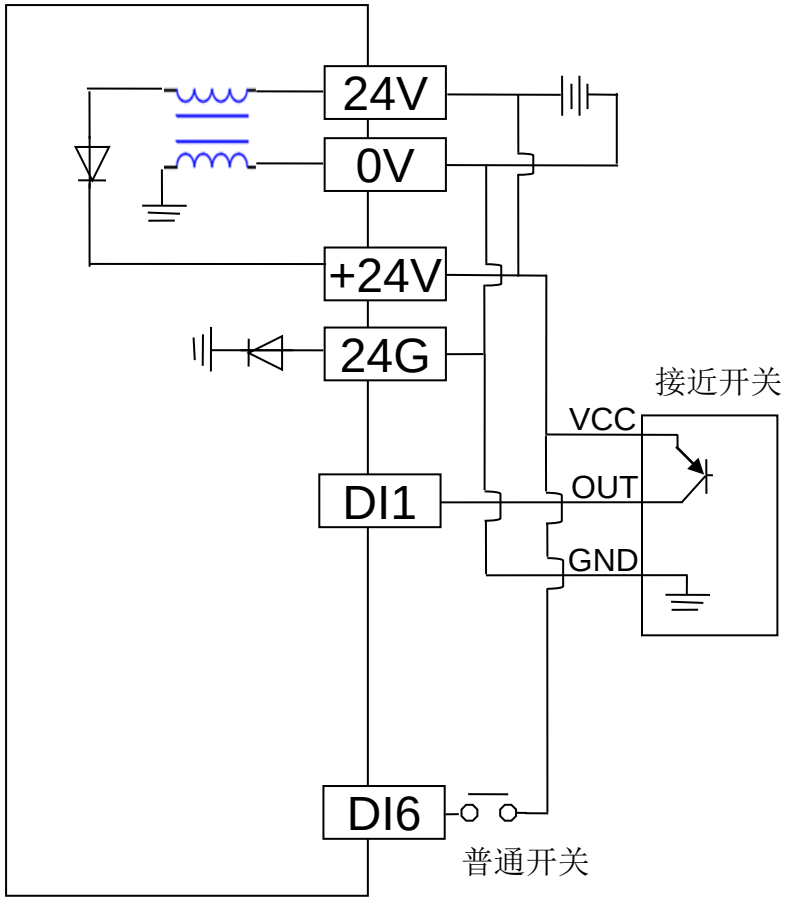
NPN外部供电

1.4、SFC介绍 之 接线图

1.主机数字量输入接线示意图



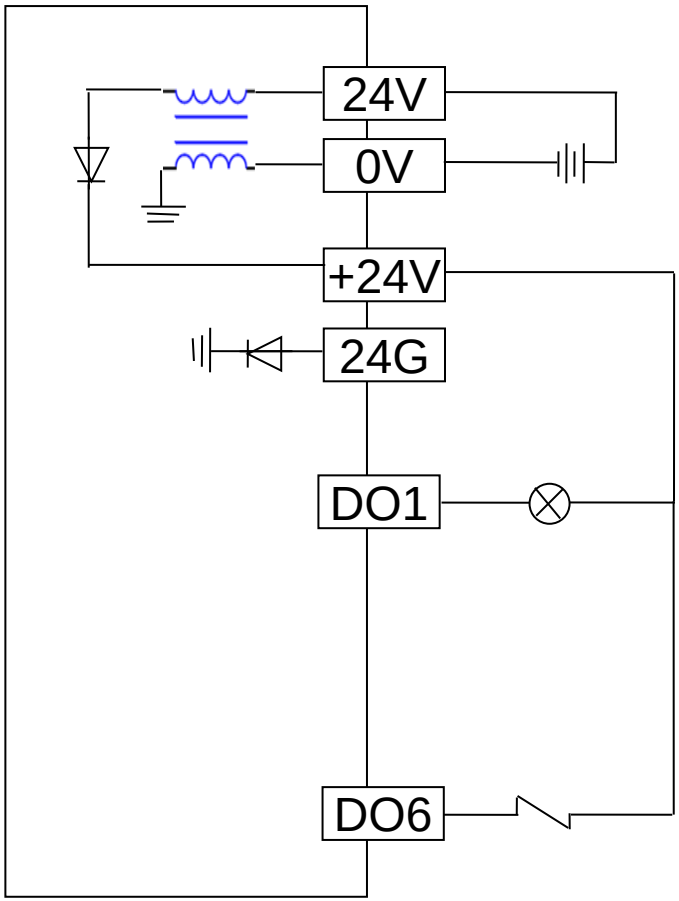
PNP内部供电



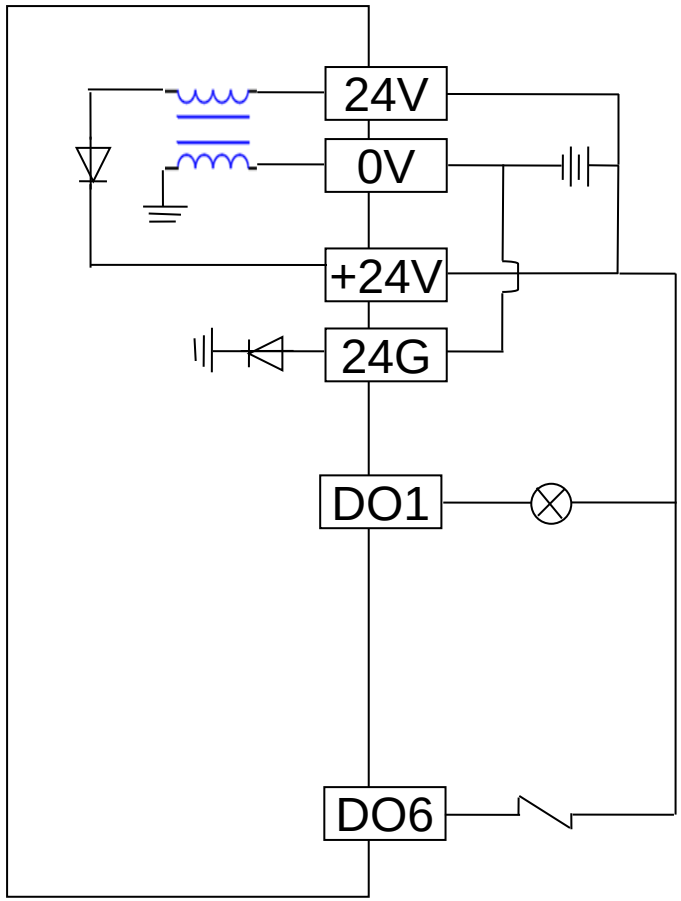
PNP外部供电

1.4、SFC介绍 之 接线图

2.主机数字量输出接线示意图



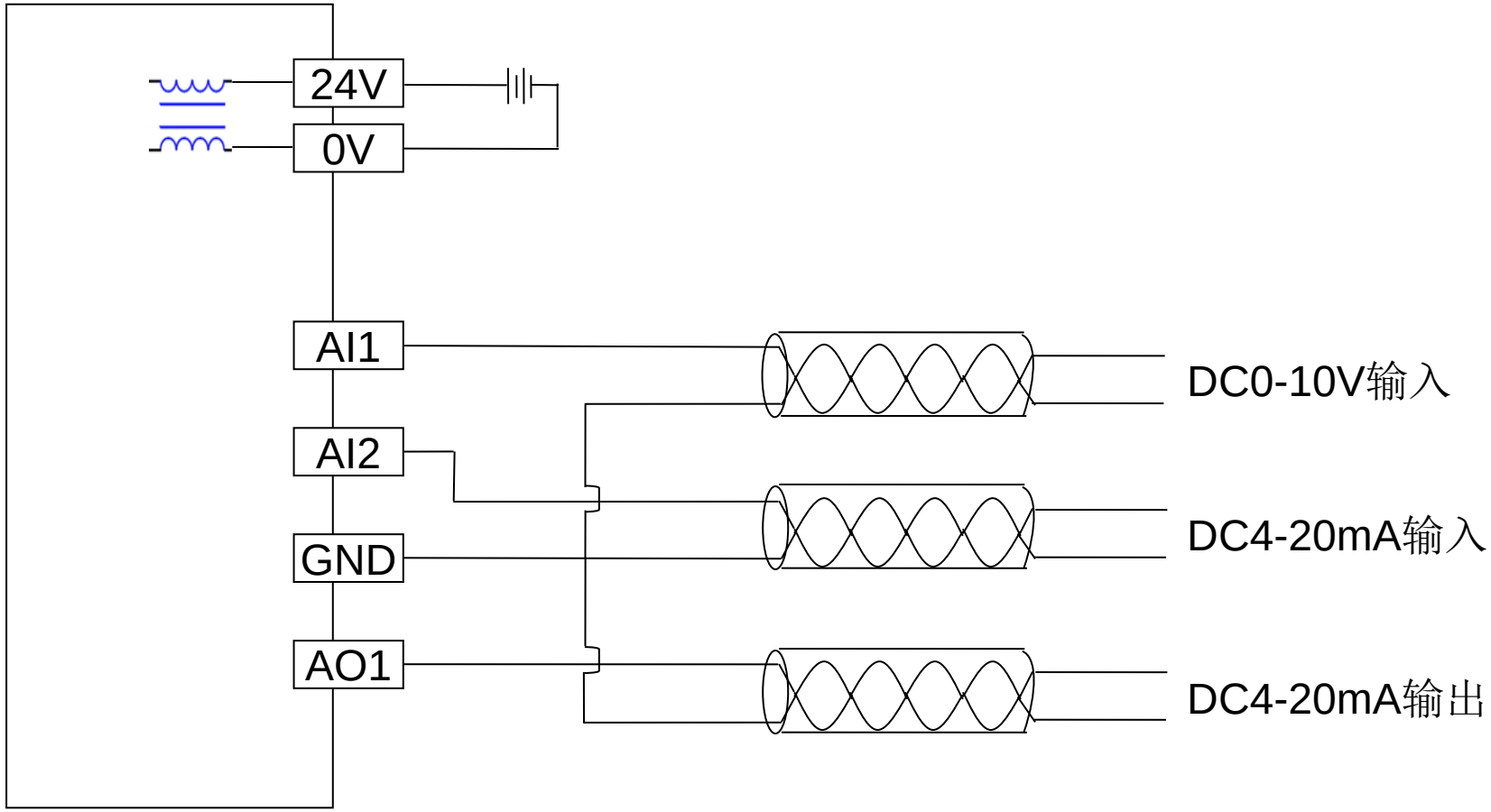
NPN晶体管输出内部供电



NPN晶体管输出外部供电

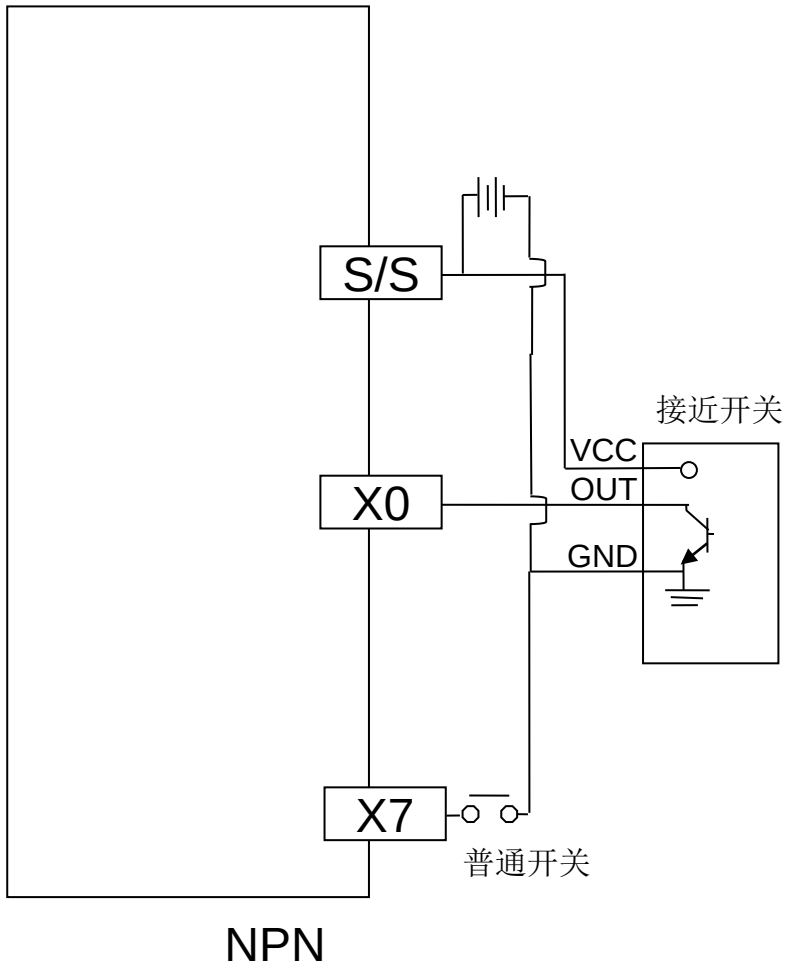
1.4、SFC介绍 之 接线图

3.主机模拟量接线示意图



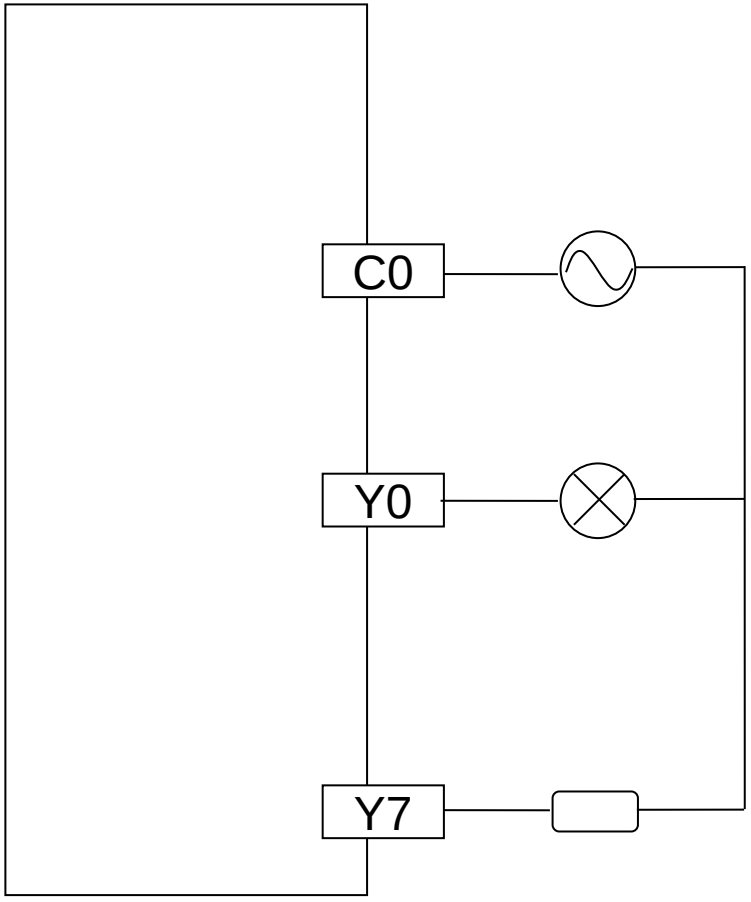
1.4、SFC介绍 之 接线图

4.扩展模块数字量输入接线示意图

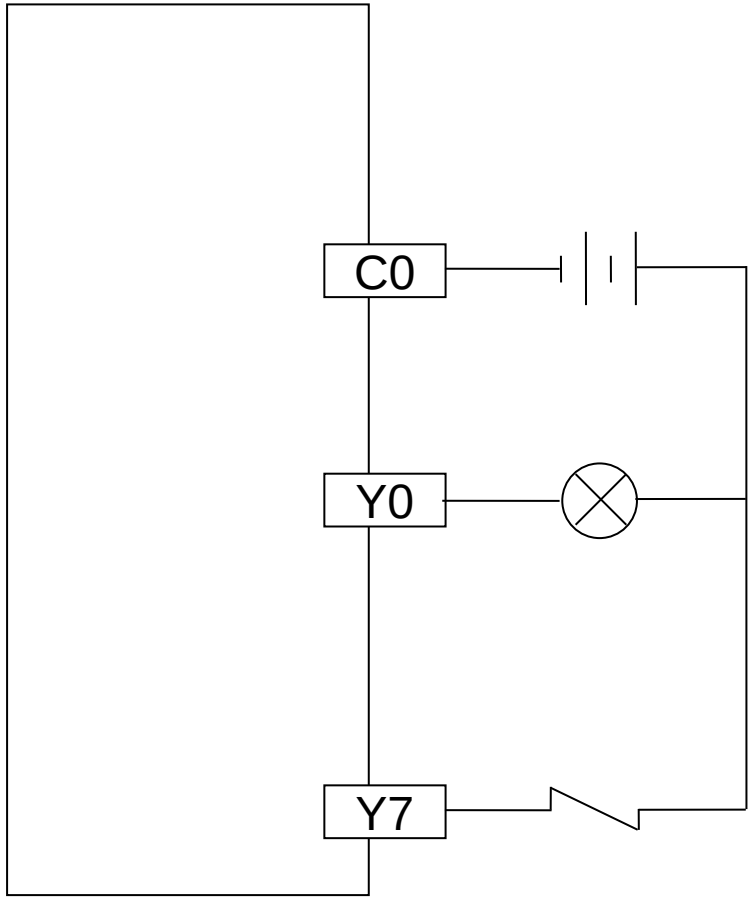


1.4、SFC介绍 之 接线图

5.扩展模块数字量输出接线示意图



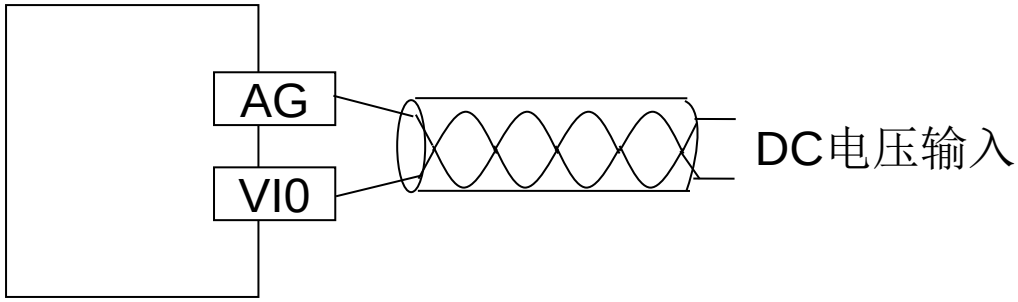
AC/DC继电器

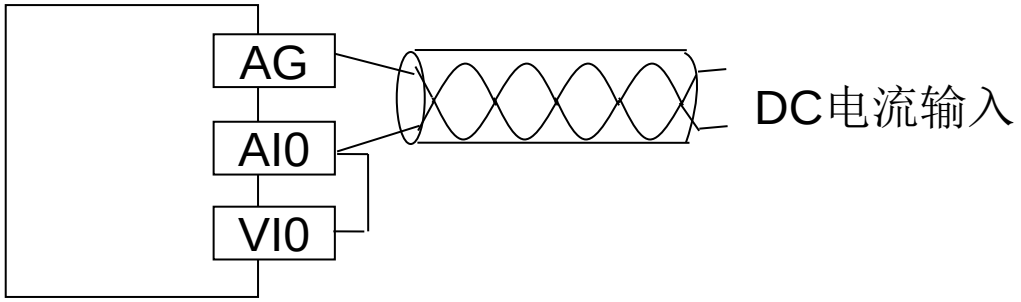


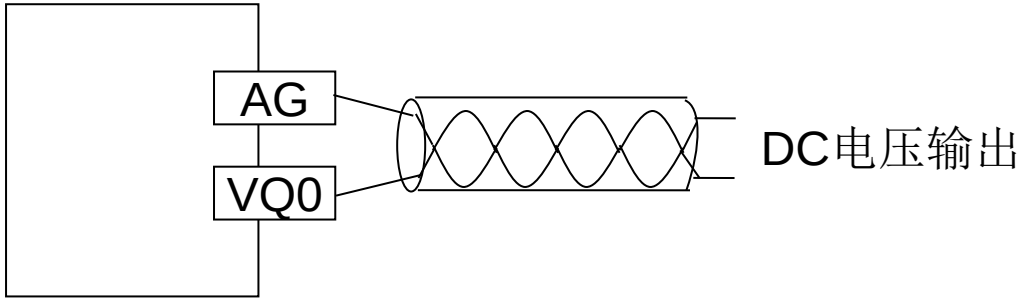
DC NPN晶体管输出

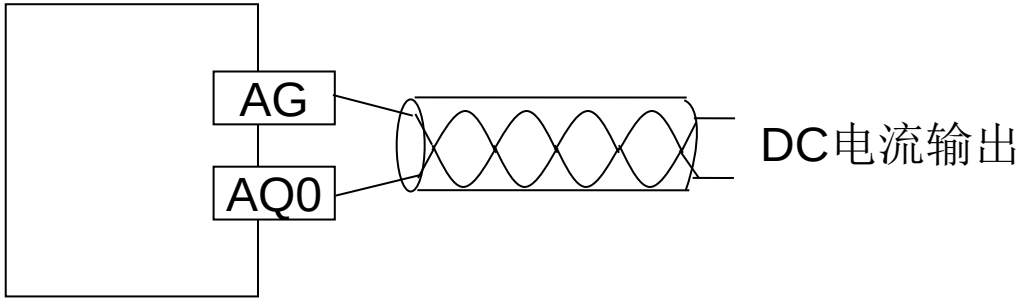
1.4、SFC介绍 之 接线图

6.扩展模块模拟量输入输出接线示意图









2.1、SFC特点 之 总线丰富

SFC**总线**控制器从命名就已经很清楚的知道，总线通信是控制器的一大特点。

★经典串行总线：RS485

低速串行通信总线，广泛应用于仪器仪表、驱动器等工业设备。

★控制器局域网：CAN

中速串行通信总线，可靠、安全，广泛应用于汽车、医疗、工业等行业。

★运动控制总线：Modbus-Tcp

高速串行通信总线，数据帧简单紧凑，数据传输量大，实时性好，在工业控制系统中广泛应用。

★运动控制总线：EtherCAT

高速串行通信总线，性能高，拓扑灵活，易用性、易实施性等特点，而受到广泛应用。

注：后续章节会对几种总线较详细介绍

2.2、SFC特点 之 执行效率高

★意法高性能MCU

SFC控制逻辑的CPU采用32位 ARM® Cortex®-M7 主频可达480MHz，双精度FPU，指令执行速度1027MDIPS(每秒10亿条指令)，支持DSP指令，执行速度快。

★C语言开发

采用库+应用工程方式进行芯片级开发。工程编译后直接是CPU可识别的机器执行码，CPU直接执行。不需要像PLC等解析型控制器对其他脚本语言进行解码、再执行，执行效率高。

2.3、SFC特点 之 保密性强

SFC在对程序代码特别是用户关键技术、知识产权的保密非常重视。因此我们提供了双重代码保护方案：应用级保护及芯片级保护。使用户不必担心程序被破解。

★应用级保护

本级保护为库函数提供相应的API对每片CPU的唯一特性及用户自己设置的密码进行相关算法计算得到的密钥进行保护。即使执行码从芯片被读取出来，也无法在另一片芯片上应用。

★芯片级保护

芯片厂家提供的增强型安全功能：读保护、扇区写保护等。读保护用于保护FLASH用户区，以防止不可信代码进行读操作。扇区写则是避免FLASH的非易失性代码或数据被意外修改。

2.4、SFC特点 之 易用、易拓展

SFC应用现场总线通信控制伺服等设备，相对于传统的脉冲型控制器，接线更少、更简单方便，设备的拓展更加容易、灵活。

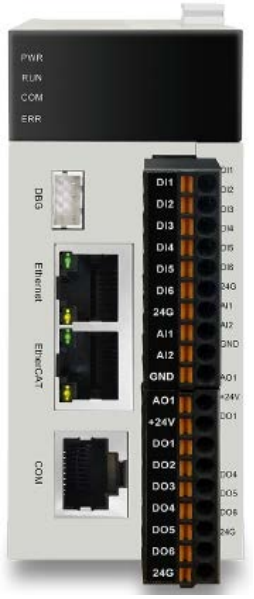
例如在现有系统的基础上，再增加一轴伺服驱动，则出现下列结果

脉冲型
可扩展

脉冲型
不可扩展

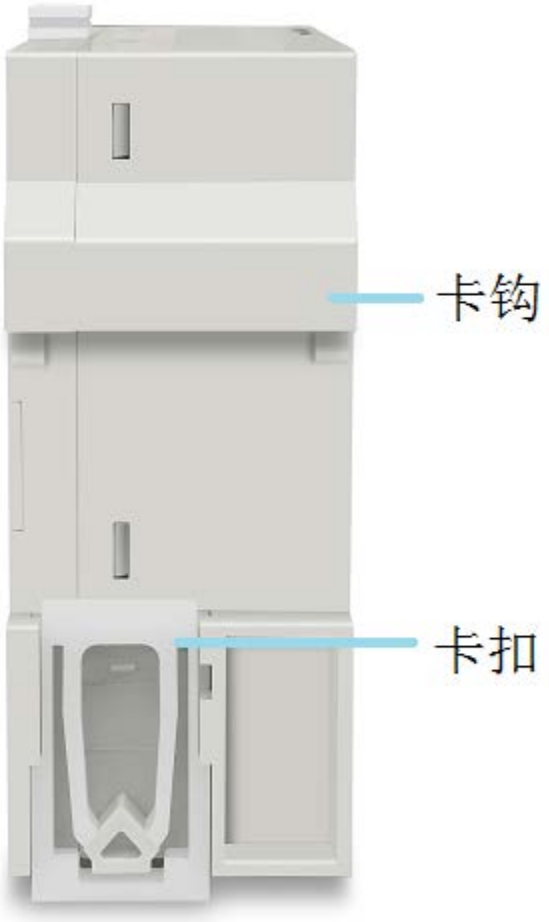
2.4、SFC特点 之 易用、易拓展

SFC
总线控制型



2.5、SFC特点 之 迷你安装

SFC采用卡片式设计，轨道式安装。体积小，安装与拆卸简易灵活，拓展方便，配线简单。有效减少配盘空间、控制柜体积。



3.1、现场总线简介 之 现场总线

现场总线是一种应用于现场，在现场设备之间、现场设备与控制装置之间实行双向、串行、多节点数字通信的技术。具有如下特点：

★实现分布式控制功能

★实现数据化、双向通信

★控制设备信息扩展

★降低安装成本

★互操作性

3.1、现场总线简介 之 现场总线

常见的现场总线及协议有以下几类：

1、Modbus

Modbus是一种串行通信协议，已经成为工业领域通信协议的业界标准，并且现在是工业电子设备之间常用的连接方式。

2、Modbus TCP

Modbus TCP是Modbus协议在Ethernet-TCP/IP之上应用。 将串行链路通信中的主从模式的概念演变为客户端与服务器。

3、CANopen

CANopen是一种架构在控制局域网络（CAN）上的高层通信协议，包括通信子协议及设备子协议，常在嵌入式系统中使用，也是工业控制常用到的一种现场总线。

4、DeviceNet

DeviceNet是一种低成本的通信连接也是一种简单的网络解决方案，有着开放的网络标准。直接互联性不仅改善了设备间的通信，而且提供了相当重要的设备级阵地功能。

3.1、现场总线简介 之 现场总线

5、PROFIBUS

Profibus 作为一种快速总线，被广泛应用于分布式外围组件(PROFIBUS-DP)。

6、PROFINET

PROFINET 是一种由 PNO(PROFIBUS 用户组织)针对开放式工业以太网制定的标准：国际上订立的一种针对通讯的 IT 标准(如 TCP/IP 协议)。

7、EtherCAT

EtherCAT(Ethernet for Control Automation Technology，控制和自动化技术的以太网)是一种用于工业自动化的实时以太网解决方案，性能优越，使用简便。

8、Lightbus

经过验证的 Beckhoff 光纤总线系统具有极为优秀的抗 EMI 性能，易于安装，数据流快速、循环且具有确定性。

3.1、现场总线简介 之 现场总线

9、CC-Link

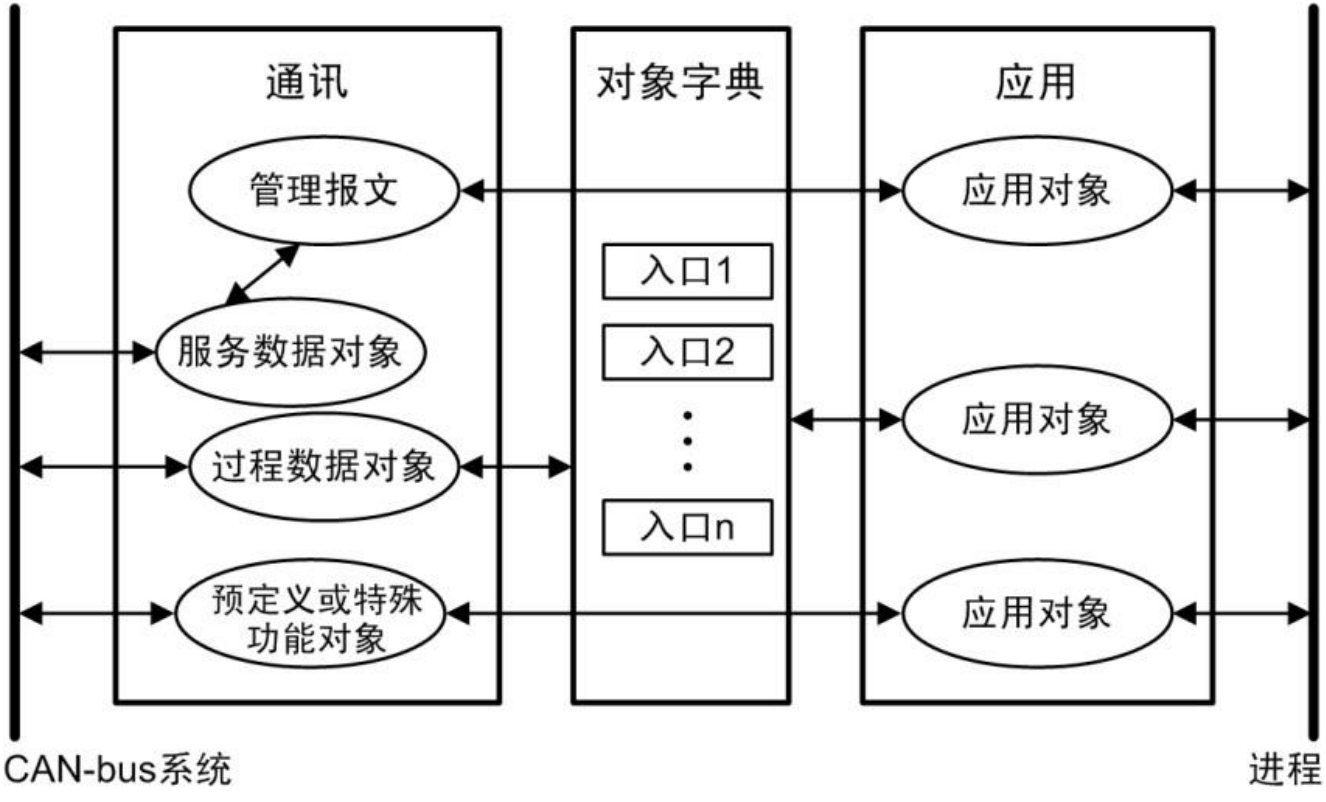
CC-Link是Control&Communication Link（控制与通信链路系统）的缩写，在其系统中，可以将控制和信息数据同是以10Mbit/s高速传送至现场网络，具有性能卓越、使用简单、应用广泛、节省成本等优点。

10、Ethernet POWERLINK

Ethernet POWERLINK 是一项在标准以太网介质上，用于解决工业控制及数据采集领域数据传输实时性的最新技术。本文介绍它的基本原理、相关特性如冗余、直接交叉通信、拓扑结构、安全性设计，并定义其物理层与介质等内容。

3.2、现场总线简介 之 CANopen

CANopen是基于CAN总线上实现的应用层，由非营利组织CiA（CAN in Automaion）进行标准的起草及审核工作，遵循基本的设备及通信子协议定义在CiA301标准中，用于运动控制的协议定义在CiA402标准中，其通信速率在5kbps 至1Mbps之间，最大节点数为127。



3.2、现场总线简介 之 CANopen

CANopen最为核心的是对象字典，描述了应用对象和 CANopen 报文之间的关系。

设备的启动及重置由状态机(state machine)控制。状态机需包括以下的几个状态：Init, Pre-OP, OP 及 Stop。

CANopen通信对象如下表所示，各对象的启动与关闭也由状态机控制。

通信对象	说明
网络管理(NMT)	用于CANopen网络管理,主机配置从机节点状态
服务数据对象(SDO)	用于传输非时间关键数据，如CANopen通信参数的配置
过程数据对象(PDO)	用于传输时间关键过程数据，如位置模式的位置数据等的传输
同步对象(SYNC)	用于同步各个节点
紧急对象(EMCY)	用于传输驱动器报警数据
错误控制(Error Control)	用于监控节点的生命状态

3.3、现场总线简介 之 EtherCAT

将计算机网络中的以太网技术应用于工业自动化领域构成的工业控制以太网，简称工业以太网或以太网现场总线。它是当前工业控制现场总线技术的一个重要发展方向。

与传统技术的现场总线相比，以太网现场总线具有以下优点：

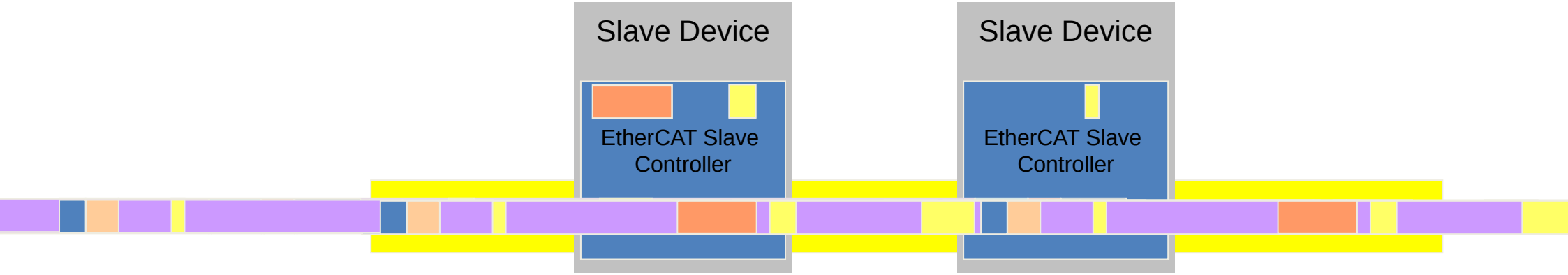
★传输速度快，数据包容量大，传输距离长

★使用通用以太网元器件，性价比高

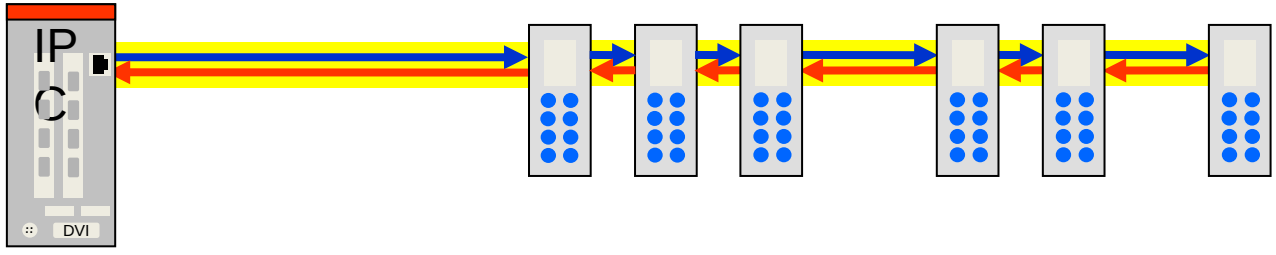
★可以接入标准以太网端口

3.3、现场总线简介 之 EtherCAT

"On the fly"传输方式



数据的提取由协议芯片实现,只提取,在提取后不立即对数据进行处理,而是将数据继续传递.



3.3、现场总线简介 之 EtherCAT

EtherCAT底层数据交换完全基于纯硬件机制。

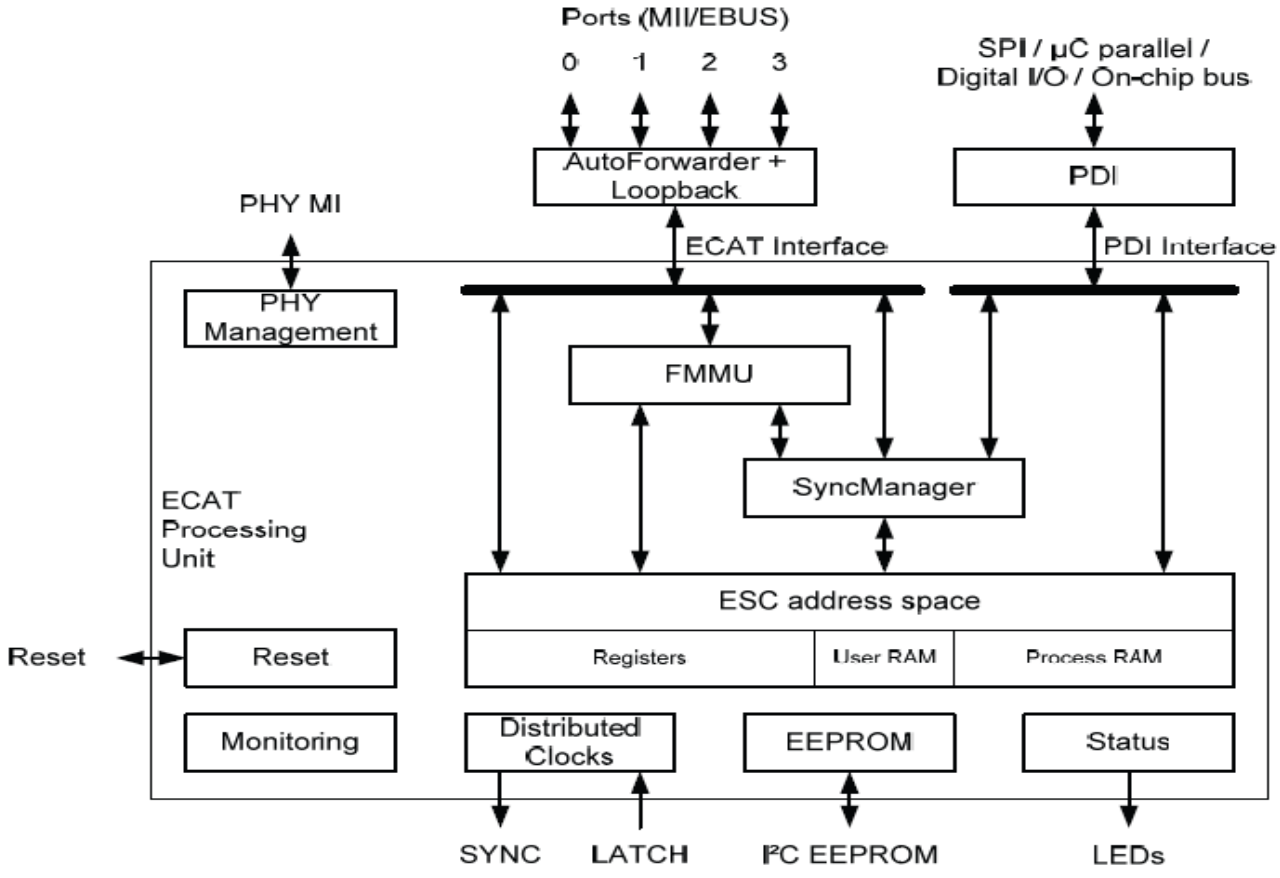
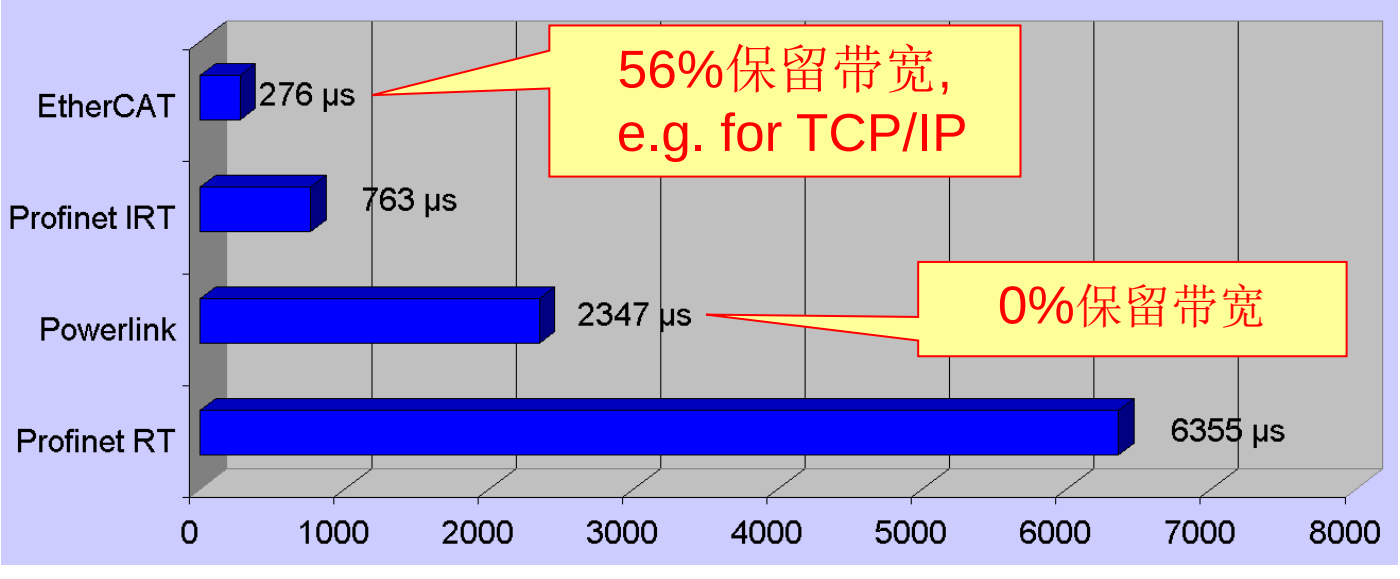


Figure 1: EtherCAT Slave Controller Block Diagram

3.3、现场总线简介 之 EtherCAT

EtherCAT性能比较：
1.40 轴 (每轴20字节输入/ 输出数据)
2.50 I/O 站，总共560个 EtherCAT 总线端子
3.2000 数字量 + 200 模拟量I/O, 总线长度 500 m

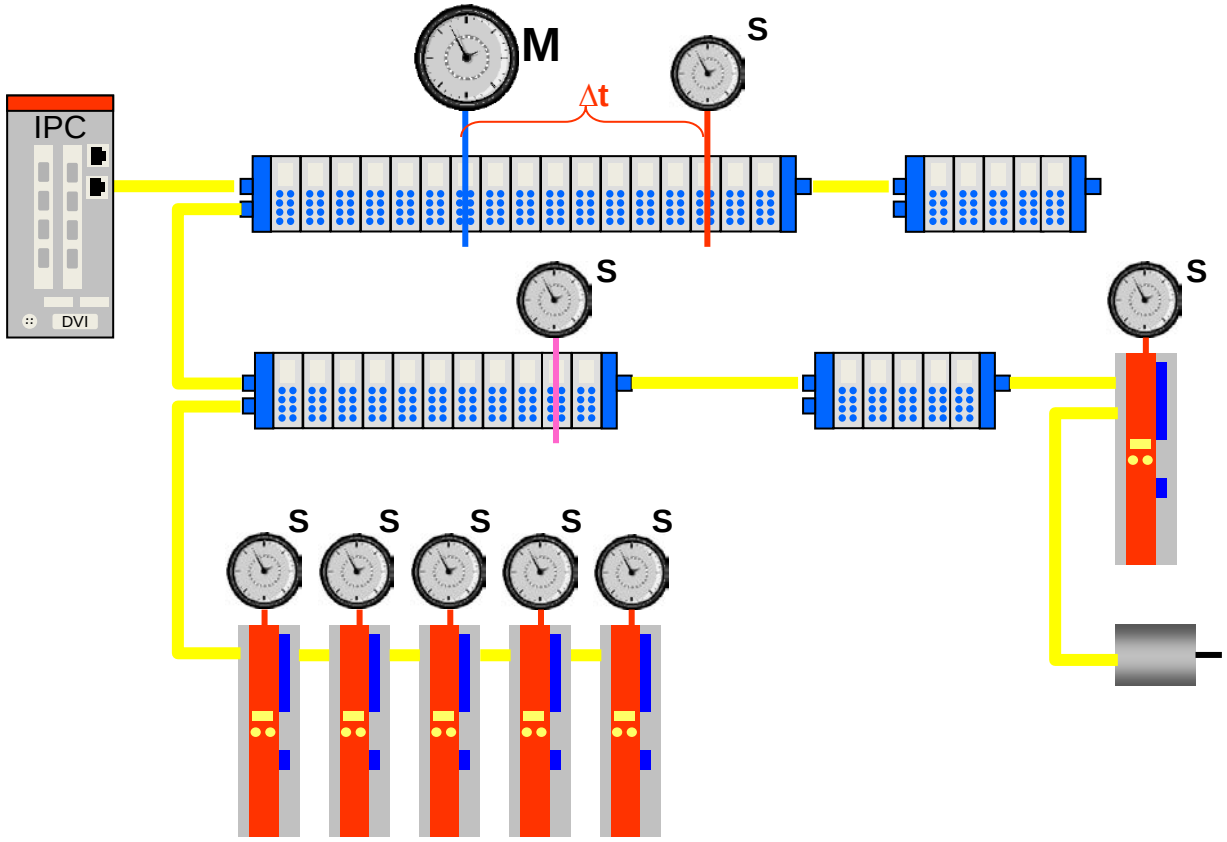
EtherCAT性能：循环时间 276µs，at 44% 总线负载, 报文长度 122µs
Profinet IRT **763 µs**, Powerlink V2 **2347µs*,** Profinet RT **6355 µs**



3.3、现场总线简介 之 EtherCAT

EtherCAT 分布时钟

精确同步 (<< 1 µs!)，通过分布时钟的精确调整来实现！



4.1、SFC软件开发说明 之 应用框架

本应用工程采用嵌入式实时多线程操作系统RT-Thread，以库+二次开发工程的方式提供给用户使用。

用户的应用工程主要在SysAct.c文件内实现。主要由以下函数：本
void SysActInit(void)、void SysActIdle(void)、void SysActTimer(void)、
void X1EintIrq(void)、void X2EintIrq(void)、void X3EintIrq(void)、
void X4EintIrq(void)、void X5EintIrq(void)、void X6EintIrq(void)。

★void SysActInit(void)
初始化函数，用于开机后，程序正常运行前的准备工作，如参数初始化等。

★void SysActIdle(void)
循环函数，应用任务都放在此函数内一直执行。

★void SysActTimer(void)
1mS定时中断，每1mS执行一次，用于对时间或实时性要求比较高的任务。

★void X1EintIrq(void)
DI1外部边沿中断，用于执行需要快速响应DI1边沿变化的任务。

4.1、SFC软件开发说明 之 应用框架

★void X2EintIrq(void)
DI2外部边沿中断，用于执行需要快速响应DI2边沿变化的任务。

★void X3EintIrq(void)
DI3外部边沿中断，用于执行需要快速响应DI3边沿变化的任务。

★void X4EintIrq(void)
DI4外部边沿中断，用于执行需要快速响应DI4边沿变化的任务。

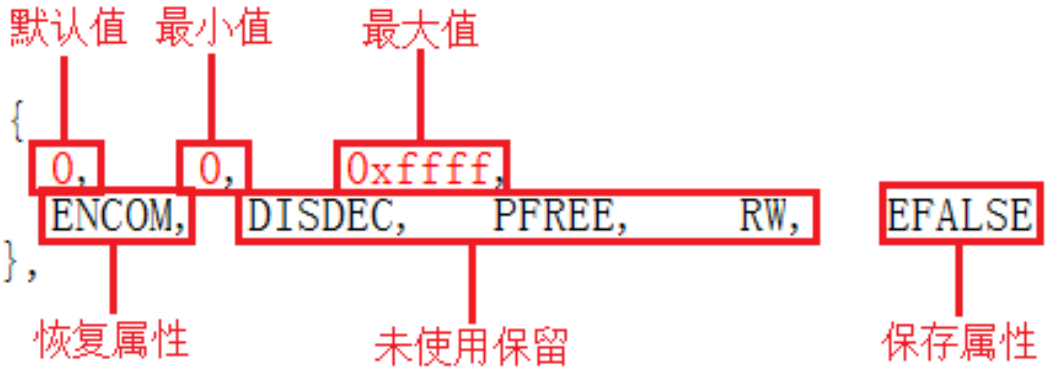
★void X5EintIrq(void)
DI5外部边沿中断，用于执行需要快速响应DI5边沿变化的任务。

★void X6EintIrq(void)
DI6外部边沿中断，用于执行需要快速响应DI6边沿变化的任务。

4.2、SFC软件开发说明 之 参数管理

基本参数就像是高楼大厦的基础，如果没有基础，就不会有高楼。一段程序，一个项目，一个工程都是如此，都要有一个基础，一个框架，而后通过不断地修改，不断的革新以达成自己想要的结果。为了方便用户使用，这里提供了两种参数管理方式：

1.每个参数拥有一个属性表，表包含参数最大值、最小值、默认值、恢复属性、保存属性等（其中位参数组，16个位共享一个属性表）。并对所有参数分为以下几组：位参数组2组PB、PKB，字参数组2组PW、PKW，系统参数组1组SYS(内核使用)。



4.2、SFC软件开发说明 之 参数管理

2.不配置参数属性表，只对参数的保存属性进行划分组，分别定义如下：位参数组PB(不保存)，位参数组PKB(掉电保存)，字参数组PW(不保存)，字参数组PKW(修改保存)，字参数组PDW(掉电保存)，系统参数组SYS(内核使用)

这些参数组又对应到Modbus-Rtu/Modbus-Tcp通信地址上。如下表：

序号	参数组	通信类型	起始地址	结束地址
1	PB	0x	0	PBNUM*16-1
2	PKB	0x	PBNUM*16	PBNUM*16+PKBNUM*16-1
3	PW	4x	0	PWNUM-1
4	PKW	4x	PWNUM	PWNUM+PKWNUM-1
5	PDW	4x	PWNUM+PKWNUM	PWNUM+PKWNUM+PDWNUM-1
6	SYS	4x	0x8000	0x8000+SYSNUM-1

4.2、SFC软件开发说明 之 参数管理

参数管理按参数组及通信地址方式提供两组管理函数：编号1为参数组方式；编号2为通信地址方式。

★位状态获取
1、UINT16 GetParamBit(UINT32 Gp,UINT32 Id)
2、UINT16 GetBit(UINT16 Addr)

★位状态设置
1、UINT8 SetParamBit(UINT32 Gp,UINT32 Id,UINT16 Val)
2、UINT8 SetBit(UINT16 Addr,UINT16 Val)

★字参数值获取
1、UINT16 GetParamWord(UINT32 Gp,UINT32 Id)
2、UINT16 GetWord(UINT16 Addr)

★字参数值设置
1、UINT8 SetParamWord(UINT32 Gp,UINT32 Id,UINT16 Val)
2、UINT8 SetWord(UINT16 Addr,UINT16 Val)

4.2、SFC软件开发说明 之 参数管理

★双字参数值获取

1、UINT32 GetParamDWord(UINT32 Gp,UINT32 Id)

2、UINT32 GetDWord(UINT16 Addr)

★双字参数值设置

1、UINT8 SetParamDWord(UINT32 Gp,UINT32 Id,UINT32 Val)

2、UINT8 SetDWord(UINT16 Addr,UINT32 Val)

Gp	参数组：PB、PKB、PW、PKW、PDW、SYS
Id	组成员：0-组成员个数-1
Addr	位地址：PBNUM*16+PKBNUM*16-1
	字地址：0-PWNUM+PKWNUM+PDWNUM-1
Val	位设置参数值：非零表示1
	字设置参数值

4.3、SFC软件开发说明 之 开关量操作

开关量(数字量)在每个控制系统中都是不可缺少的一部分，有输入开关量、输出开关量之分。输入开关量用于获取外部数字信号，输出开关量用于控制外部数字量。提供了如下开关量相关函数：

★获取输入开关量
UINT8 GetIoStu(UINT8 Cnl)

★获取输出开关量
UINT8 GetOutStu(UINT8 Cnl)

★获取输入开关量上升沿
UINT32 GetIoUp(UINT8 Cnl)

★获取输入开关量下降沿
UINT32 GetIoDw(UINT8 Cnl)

★获取输出开关量上升沿
UINT32 GetIoOutUp(UINT8 Cnl)

4.3、SFC软件开发说明 之 开关量操作

★获取输出开关量下降沿
UINT32 GetIoOutDw(UINT8 Cnl)

★设置输出开关量
void SetIoStu(UINT8 Cnl,UINT8 Val)

注：边沿获取函数仅应用于SysActIdle函数及其调用的子函数。

SFC主机与扩展模块开关量编号说明：

SFC+SMC-AX16ET+SMC-AC16ET+SMC-AY16ET

4.3、SFC软件开发说明 之 开关量操作

序号	型号	输入标识	输出标识	输入编号	输出编号
1	SFC	D1-DI6	DO1-DO6	0-5	0-5
2	SMC-AX16ET	X0-X15	-	8-23	-
3	SMC-AC16ET	X0-X7	Y0-Y7	24-31	8-15
4	SMC-AY16ET	-	Y0-Y15	-	16-31

注：扩展模块开关量的起始地址编号从8开始

4.3、SFC软件开发说明 之 开关量操作

SFC主机机体提供了3个可控的LED指示灯(RUN、COM、ERR)，供用户应用工程定义使用。提供的函数可对3个LED灯进行常灭、常亮、慢闪、慢闪等操作。

★LED操作函数

void SetLedMod(UINT8 Cnl,UINT8 Mode)

Cnl	
LED_RUN	设置灯通道
LED_COM	
LED_ERR	
Mode	
LED_MODE_OFF	设置灯状态
LED_MODE_ON	
LED_MODE_SLOW	
LED_MODE_QUICK	

4.4、SFC软件开发说明 之 模拟量操作

在控制系统中，我们有时也需要采集外部模拟信号如：电压、电流、温度、重量等需要的目标信息，同时加工这些信息，以输出电压模拟信号、电流模拟信号达到控制目标的目的。提供了如下模拟量相关函数：

★获取电压/电流型模拟量输入值
UINT16 GetAiVal(UINT8 Channel)

★获取热电阻模块输入值
UINT16 GetRCVal(UINT8 Channel)

★获取热电偶模块输入值
UINT16 GetTCVal(UINT8 Channel)

★获取温湿度模块温度输入值
UINT16 GetDTTVal(UINT8 Channel)

★获取温显度模块湿度输入值
UINT16 GetDTHVal(UINT8 Channel)

4.4、SFC软件开发说明 之 模拟量操作

★获取称重模块输入值
UINT16 GetDTHVal(UINT8 Channel)

★设置电压/电流模拟量输出值
void SetAoVal(UINT8 Channel,UINT16 Val)

SFC与扩展模块模拟量编号说明：

SFC+SMC-A04AD+SMC-A04DA
+SMC-A04TC+SMC-A04RC
+SMC-A04DT+SMC-AW02

4.4、SFC软件开发说明 之 模拟量操作

序号	型号	输入标识	输出标识	输入编号	输出编号
1	SFC	AI1-AI2	AO1	0-1	0
2	SMC-A04AD	AI0/VI0-AI3/VI3	-	2-5	-
3	SMC-A04DA	-	AQ1/VQ1-AQ4/VQ4	-	1-4
4	SMC-A04TC	TC0-TC3	-	0-3	-
5	SMC-A04RC	RC0-RC3	-	0-3	-
6	SMC-A04DT	DT0-DT3	-	0-3	-
7	SMC-AW02	W0-W1	-	0-1	-

注：扩展模块电压/电流型输入起始地址编号从2开始，电压/电流型输出起始地址编号从1开始，其他模块起始地址编号从0开始

4.5、SFC软件开发说明 之 总线通信

RS485总线Modbus-Rtu主站相关操作函数：

★RS485总线初始化

```
uint8_t FB_RS485_INIT(uint8_t par, uint8_t stopbits, uint32_t baudrate)
```

★线圈(位)读操作

```
uint8_t FB_MB_COIL_READ(uint8_t slv, uint16_t addr, uint8_t num,
                        uint8_t *pdat, uint16_t timeout)
```

★线圈(位)写操作

```
uint8_t FB_MB_COIL_WRITE(uint8_t slv, uint16_t addr, uint8_t num,
                         uint8_t *pdat, uint16_t timeout)
```

★寄存器(字)读操作

```
uint8_t FB_MB_REG_READ(uint8_t slv, uint16_t addr, uint8_t num,
                       uint8_t *pdat, uint16_t timeout)
```

4.5、SFC软件开发说明 之 总线通信

★寄存器(字)写操作

```
uint8_t FB_MB_REG_WRITE(uint8_t slv, uint16_t addr, uint8_t num,
                        uint8_t *pdat, uint16_t timeout)
```

4.5、SFC软件开发说明 之 总线通信

EtherCAT总线/CANopen总线主站相关操作函数：

★端口扫描
uint8_t FB_CONFIG(uint8_t type)

★获取总线从设备节点数
uint8_t FB_NODE_COUNT(uint8_t type)

★启动总线通信
uint8_t FB_START(uint8_t type)

★设置同步周期
uint8_t FB_SYNC_CYCLE(uint32_t cycle)

★获取总线状态
uint8_t FB_ECAT_STATUS(void)

4.5、SFC软件开发说明 之 总线通信

★SDO对象字典获取

uint32_t FB_SDO_READ(uint8_t slave, uint16_t idx, uint8_t subidx,
uint8_t *len, uint8_t *dat)

★SDO对象字典设置

uint32_t FB_SDO_WRITE(uint8_t slave, uint16_t idx, uint8_t subidx,
uint8_t len, uint8_t *dat)

4.5、SFC软件开发说明 之 总线通信

基于总线通信的伺服控制相关操作函数：

★设置伺服轴类型及轴编号
uint8_t MC_AXIS_TYPE(uint8_t axis, int32_t src, uint8_t fb_no)

★设置伺服PDO文件
uint8_t MC_PDO_CONFIG(uint8_t axis, int32_t src)

★伺服轴使能
uint8_t MC_ENABLE(uint8_t axis, uint8_t on_off)

★控制单位当量设置
uint8_t MC_UNITS(uint8_t axis, double UNITS)

★伺服轴操作模式设置
uint8_t MC_OPERATION_MODE(uint8_t axis, int8_t mod)

4.5、SFC软件开发说明 之 总线通信

★伺服轴空闲状态查询
uint8_t MC_IDLE(uint8_t AxisSel)

★等待伺服轴空闲
void MC_WAIT_IDLE(uint8_t AxisSel)

★移动轴至指定位置(增量式)
uint8_t MC_MOVE(uint8_t axis, int32_t Length)

★移动轴至指定位置(绝对式)
uint8_t MC_MOVEABS(uint8_t axis, int32_t Length)

★控制轴做连续运动
uint8_t MC_MOVE_CO(uint8_t axis, int8_t dir)

★取消轴运动
uint8_t MC_CANCEL(uint8_t axis, int32_t mode)

4.5、SFC软件开发说明 之 总线通信

★设置轴加速度
uint8_t MC_ACCEL(uint8_t axis, uint32_t Accel)

★设置轴减速度
uint8_t MC_DECEL(uint8_t axis, uint32_t Decel)

★设置轴速度
uint8_t MC_SPEED(uint8_t axis, int32_t Speed)

★定义轴当前位置
uint8_t MC_DEF_POS(uint8_t axis, int32_t pos)

★获取轴反馈位置
int64_t MC_GET_FBK_POS(uint8_t axis)

4.5、SFC软件开发说明 之 总线通信

RS485总线Modbus-Rtu从站相关操作函数：

★设置从站波特率
void SetModbusBtr(UINT32 Ptr)

★设置从站站号
void SetModbusAddr(UINT8 Addr)

Modbus-Tcp从站相关操作函数：

★IP地址设置
void SetEthernetIpByNum(UINT32 Ip)

4.6、SFC软件开发说明 之 线程操作

在实际应用中可能SysAct的这个任务无法满足开发需求，需要进行多线程任务的开发。在此就需要应用到与线程相关的函数。

★线程创建函数

```
rt_thread_t create_user_thread(const char *name,
                               void (*entry)(void *parameter),void *parameter)
```

★线程启动函数

```
rt_err_t rt_thread_startup(rt_thread_t thread)
```

★线程休眠

```
rt_err_t rt_thread_mdelay(rt_int32_t ms)
```

★线程恢复

```
rt_err_t rt_thread_resume (rt_thread_t thread)
```

★线程删除

```
rt_err_t rt_thread_delete(rt_thread_t thread)
```

4.7、SFC软件开发说明 之 代码保护

对于应用开发人员来说，为了防止别人恶意窃取自己的劳作成果，将自己的代码在控制器上进行保护是非常有必要的。我们提供了两层的代码保护：应用层保护、芯片层保护。

★应用层密钥安装函数
void SetupAppKeyPrt(UINT32 UserKey)

★应用层密钥判断函数
UINT8 DetAppKeyPrt(UINT32 UserKey)

★芯片层保护
void SetFlashReadprotection(void)

4.8、SFC软件开发说明 之 其他说明

SFC自带6路高速输入，6路高速输入都可对端口边沿脉冲进行计数或是由DI1-DI2，DI4-DI5组成的两组AB相计数端口。

★获取端口脉冲计数值
UINT32 GetXinPul(UINT8 Cnl)

★设置端口脉冲计数值
void SetXinPul(UINT8 Cnl,UINT32 Val)

★获取AB相端口脉冲值
UINT32 GetEncPul(UINT8 Cnl)

★设置AB相端口脉冲值
void SetEncPul(UINT8 Cnl,UINT32 Val)

4.9、SFC软件开发说明 之 应用示例

应用示例1：将输入端口信息放置于PB、PW参数组内，并将PB、PW参数组内的数值传递到输出端口上。

```
//主机数字输入宏定义
#define IN_DI1_ADDR     PB,0          //主机DI1状态存入的PB地址
#define IN_DI2_ADDR     PB,1          //主机DI2状态存入的PB地址
#define IN_DI3_ADDR     PB,2          //主机DI3状态存入的PB地址
#define IN_DI4_ADDR     PB,3          //主机DI4状态存入的PB地址
#define IN_DI5_ADDR     PB,4          //主机DI5状态存入的PB地址
#define IN_DI6_ADDR     PB,5          //主机DI6状态存入的PB地址

//主机数字输出宏定义
#define OUT_DO1_ADDR    PB,8          //主机DO1受控的PB地址
#define OUT_DO2_ADDR    PB,9          //主机DO2受控的PB地址
#define OUT_DO3_ADDR    PB,10         //主机DO3受控的PB地址
#define OUT_DO4_ADDR    PB,11         //主机DO4受控的PB地址
#define OUT_DO5_ADDR    PB,12         //主机DO5受控的PB地址
#define OUT_DO6_ADDR    PB,13         //主机DO6受控的PB地址
```

4.9、SFC软件开发说明 之 应用示例

```
#define AI1_VIN_ADDR     PW,4              //主机AI1模拟量值存入的PW地址
#define AI2_IIN_ADDR     PW,5              //主机AI2模拟量值存入的PW地址

#define AO1_IOUT_ADDR    PW,6              //主机AO1模拟量受控的PW地址

//示例代码

SetParamBit(IN_DI1_ADDR,GetIoStu(0));     //获取DI1的状态存入对应的PB地址
SetParamBit(IN_DI2_ADDR,GetIoStu(0));     //获取DI2的状态存入对应的PB地址
SetParamBit(IN_DI3_ADDR,GetIoStu(0));     //获取DI3的状态存入对应的PB地址
SetParamBit(IN_DI4_ADDR,GetIoStu(0));     //获取DI4的状态存入对应的PB地址
SetParamBit(IN_DI5_ADDR,GetIoStu(0));     //获取DI5的状态存入对应的PB地址
SetParamBit(IN_DI6_ADDR,GetIoStu(0));     //获取DI6的状态存入对应的PB地址

//GetAiVal获取CPU端口的采样值
//GetInVol将采样值转化成对应的电压值(AI1出厂默认电压型输入)
//将电压值存于PW对应地址
SetParamWord(AI1_VIN_ADDR,GetInVol(GetAiVal(0)));
```

4.9、SFC软件开发说明 之 应用示例

```
//GetAiVal获取CPU端口的采样值
//GetInCir将采样值转化成对应的电压值(AI2出厂默认电流型输入)
//将电流值存于PW对应地址
SetParamWord(AI2_IIN_ADDR,GetInCir(GetAiVal(1)))

//获取输出对应PB地址内的值，控制相应输出
SetIoStu(0,GetParamBit(OUT_DO1_ADDR) ? 1 : 0);
SetIoStu(1,GetParamBit(OUT_DO2_ADDR) ? 1 : 0);
SetIoStu(2,GetParamBit(OUT_DO3_ADDR) ? 1 : 0);
SetIoStu(3,GetParamBit(OUT_DO4_ADDR) ? 1 : 0);
SetIoStu(4,GetParamBit(OUT_DO5_ADDR) ? 1 : 0);
SetIoStu(5,GetParamBit(OUT_DO6_ADDR) ? 1 : 0);

//GetOutCir将输出的电流值转为CPU端口输出值
//驱动CPU端口输出
SetAoVal(0,GetOutCir(GetParamWord(AO1_IOUT_ADDR)));
```

4.9、SFC软件开发说明 之 应用示例

应用示例2：简单的通过总线控制单个伺服轴运动。

```
//总线扫描
FB_CONFIG(0);

//总线节点获取(伺服轴数)
AxisNumCur = FB_NODE_COUNT(0);

//设置轴类型
MC_AXIS_TYPE(1,0,1);

//设置伺服PDO功能
MC_PDO_CONFIG(1,0);

MC_WAIT(100);

//设置单位脉冲当量
MC_UNITS(1,1);
```

4.9、SFC软件开发说明 之 应用示例

```
//设置加速度
MC_ACCEL(1,20000000);

//设置减速速度
MC_DECEL(1,20000000);

//设置当前位置
MC_DEF_POS(1,0);

//设置速度
MC_SPEED(1,20000);

//启动总线
FB_START(0);

//设置轴操作模式
MC_OPERATION_MODE(1,8);
```

4.9、SFC软件开发说明 之 应用示例

```
//复位伺服轴报警
MC_ALARM_RESET(1);

//使能伺服
MC_ENABLE(1,1);

//移动伺服轴
MC_MOVE(1,100000);

//等待移动完成
MC_WAIT_IDLE(1);
```

4.9、SFC软件开发说明 之 应用示例

应用示例3：代码保护简要说明。这里建议用户将用户密钥的安装与应用分别两个工程。以达到，应用中无入口进行密钥安装，保护更彻底。

```
//用户密钥安装

#define APP_USER_KEY    0x1a2b3c4d       //用户密钥32bit，用户自己设定

//密钥安装状态判断
//正确安装则3个LED灯快速闪烁
if(DetAppKeyPrt(APP_USER_KEY))
{
        SetLedMod(LED_RUN,LED_MODE_QUICK);

        SetLedMod(LED_COM,LED_MODE_QUICK);

        SetLedMod(LED_ERR,LED_MODE_QUICK);
}
```

4.9、SFC软件开发说明 之 应用示例

```
//安装失败，则3个LED熄灭
else
{

    SetLedMod(LED_RUN,LED_MODE_OFF);

    SetLedMod(LED_COM,LED_MODE_OFF);

    SetLedMod(LED_ERR,LED_MODE_OFF);

}
//密钥安装
SetupAppKeyPrt(APP_USER_KEY);
```

4.9、SFC软件开发说明 之 应用示例

```
//应用工程中，密钥应用

#define APP_USER_KEY    0x1a2b3c4d       //用户密钥与密钥安装要一致

//密钥安装状态判断，并将用户关键代码放些判断内。
//密钥不一致时，不运行关键代码。
//也可以将此密钥判断函数的结果，参与条件判断的逻辑运算，达到隐性判断。
if(DetAppKeyPrt(APP_USER_KEY))
{
      //用户关键代码
}
或
if(DetAppKeyPrt(APP_USER_KEY)&&运行条件1&&运行条件2)
{
      //代码运行
}
```

5、SFC应用场合

纺织机械

包装机械

印刷机械

建材机械

木工机械

食品机械

塑料机械

自动化生产线


SAVCh®