

WS2105 EC30-EK51 用户手册

CREATE: 2010/06/01

UPDATE: 2010/11/24

Version 1.1

EC30-EK51 Version 1.1

GUTTA Ladder Editor Version 1.1

<http://www.plcol.com><http://www.visiblecontrol.com>

第 1 章 总体介绍.....	4
第 2 章 开发过程一览.....	4
第 3 章 硬件方案.....	8
芯片封装和管脚.....	8
3 种封装的管脚排列.....	8
3 种封装的尺寸图.....	10
运行条件.....	12
连接到计算机.....	13
使用 EC30-EKSTM32-USB 通讯线.....	14
使用 RS232 通讯器件.....	15
使用 RS485 通讯器件.....	16
系统.....	16
关键指标.....	16
内存使用.....	17
可用中断源 (ATCH/DTCH 指令).....	18
硬件参考.....	19
系统恢复.....	20
第 4 章 发布自己的 PLC 系统.....	23
第 5 章 系统配置说明.....	27
通讯端口 [PORT].....	27
密码 [PASSWORD].....	28
中断时间 [TIME].....	29
强制表 [FORCE].....	30
第 6 章 硬件配置说明.....	31
输入输出 [IO].....	31
简介.....	31
软件配置.....	32
特别说明.....	33
使用实例.....	33
模数转换 [AD].....	35
简介.....	35
软件配置.....	35
特别说明.....	36

使用实例	36
I2C 通讯 [I2C]	37
简介	37
软件配置	40
特别说明	41
使用实例	41
SPI 通讯 [SPI].....	43
简介	43
外部电路	45
软件配置	45
使用实例	46
1-Wire 通讯 [1W].....	50
简介	50
外部电路	53
软件配置	53
使用实例	54
脉冲密度 [SPD].....	58
简介	58
软件配置	59
特别说明	60
使用实例	60
脉冲输出 [PTO]	62
简介	62
软件配置	63
特别说明	64
使用实例	64
脉宽调制 [PWM].....	69
简介	69
软件配置	71
特别说明	72
使用实例	72
第 7 章 扩展配置说明.....	75
杂项 [MISC].....	75
简介	75
外部电路	75
软件配置	76
特别说明	77
使用实例	77
输入扫描 [SIN]	79
简介	79
外部电路	79
软件配置	80
使用实例	81
输出扫描 [SOUT]	82

简介	82
外部电路	82
软件配置	85
特别说明	86
使用实例	87
段码扫描 [SLED]	88
简介	88
外部电路	89
软件配置	91
特别说明	91
使用实例	92
驱动 [AT24C08B]	97
驱动 [PCF8563]	99
EBUS 通讯 [EBUS]	101

第 1 章 总体介绍

EC30-EK51 是一款基于梯形图编程的 8051 单片机开发工具（对应于 GUTTA Ladder Editor 1.1 下的一种特殊 PLC 类型）。其开发方式和 PLC 类似，采用梯形图的方式编写用户逻辑。和普通的 PLC 开发不一样，在使用 EC30-EK51 时，不但可以编写用户逻辑，还可以对单片机的硬件进行配置。例如指定单片机的管脚 P3.3 为 PLC 的输入 I0.0；指定单片机的管脚 P2.2 为 PLC 的输出 Q0.0 等。借助于 EC30-EK51，您不需要编写任何汇编或者是 C 代码，您只需要具备基本的电路知识，会梯形图编程，就能开发单片机项目。

简单的说，**EC30-EK51 是单片机硬件配置与梯形图编程这两种工具的结合。**

由于各个厂家的 8051 单片机外围硬件各不相同，片内 SRAM、FLASH 的大小也不一样，虽然都是 8051 内核，但是可以说各有千秋，各具特色。为了让用户在开发单片机的时候像开发 PLC 程序一样方便快捷，EC30-EK51 必须隐藏部分单片机驱动细节。这就意味着 EC30-EK51 只能支持某款特定的单片机，而不是所有的 8051 内核单片机。

目前 EC30-EK51 是专为宏晶（STC）的 **IAP12C5A60AD** 开发的。

考虑到用户程序的保密性和下载程序的方便性，GUTTA Ladder Editor 1.1 并不生成完整的固件代码，它只生成硬件配置数据以及 PLC 用户逻辑代码。因此单片机必须事先写入 PLC 系统，才能使用 GUTTA Ladder Editor 1.1 下载硬件配置数据以及 PLC 用户逻辑代码。程序的下载和 PLC 的在线监控都采用 GUTTA 通讯协议，而不是宏晶的 ISP 工具。如果需要使用 EC30-EK51，就必须购买预写了 GUTTA PLC 系统的 IAP12C5A60AD 芯片，同时电路板上必须具有 RS232 通讯接口。

芯片的具体价格可参考我们的网站：

<http://www.plcol.com/>

除了侧重成本的 8051 内核单片机，基于 Cortex-M3 内核的开发工具也已经正式发布，请留意我们的网站信息。

特点：

- 采用 PLC 梯形图或指令表编写控制逻辑，支持在线监控，便于开发。
- 根据使用者的情况，包括输入输出在内的几乎所有的 IAP12C5A60AD 硬件都可在软件中直接配置使用。
- 采用高性价比的 8051 内核芯片 IAP12C5A60AD（由于采购关系，有可能是使用功能更强的 IAP12C5A60S2）。
- 单芯片方案，无需外扩 SRAM 和 FLASH，减少产品尺寸。
- 灵活的供货形式，可提供 QFP44、PLCC44、PDIP40 等多种封装的芯片。
- 无偿提供编程软件 GUTTA Ladder Editor 及其配置工具。
- 软件可定制，基于 EC30-EK51 的控制产品除了可以自用，也可以作为专业 PLC 产品向外发布。

第 2 章 开发过程一览

一般说来，在工控行业，单片机控制器产品的开发，需要经过以下步骤：

1. 根据实际的控制对象以及控制工艺流程，确定控制器的基本需求，这些基本需求包括供电方式、数字量输入输出路数、模拟量输入输出路数及精度、抗干扰能力、高速输入输出路数、通讯接口种类、人机交互方式等。
2. 根据对控制器的基本需求，确定控制器的硬件方案，硬件方案的确定不但需要考虑

- 硬件本身的稳定性, 还需要考虑行业习惯、实现成本、安装尺寸、采购难易等因素。
3. 根据硬件方案完成样机后, 分模块测试硬件性能并完成各个模块的驱动程序。
 4. 编写具体的应用程序, 应用程序主要涉及具体的控制工艺。应用程序的检测往往需要控制器连接实际的设备在真实的环境中进行。应用程序程序的编写需要程序员对控制器所处的行业应用有清楚地认识, 有相关经验或者说是行业背景。
 5. 小批量的试制和试用, 通过一段时间的使用来评估硬件和软件。

我们提供的 EC30-EK51 系统芯片, 能够极大的简化步骤 3 和步骤 4。首先, 固件中已经包含了一些常用的外围驱动。这些驱动都经过了我们的严格测试并被大量用户所使用。这些驱动可以在上位机软件中通过图形化的控件灵活配置。在控件中配置不但不容易出错, 也大大的减少了驱动开发中的重复劳动。

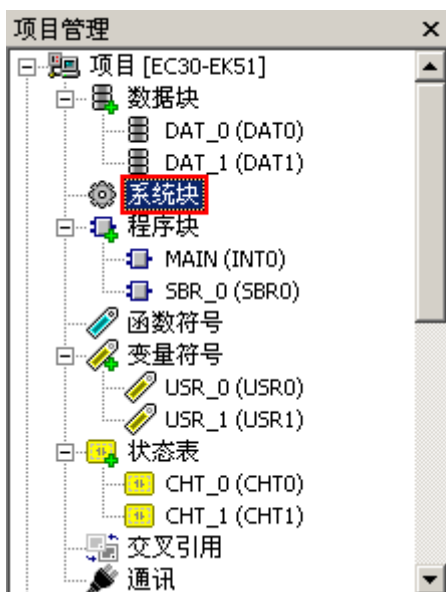
在 GUTTA Ladder Editor 软件中, 双击项目管理的项目:



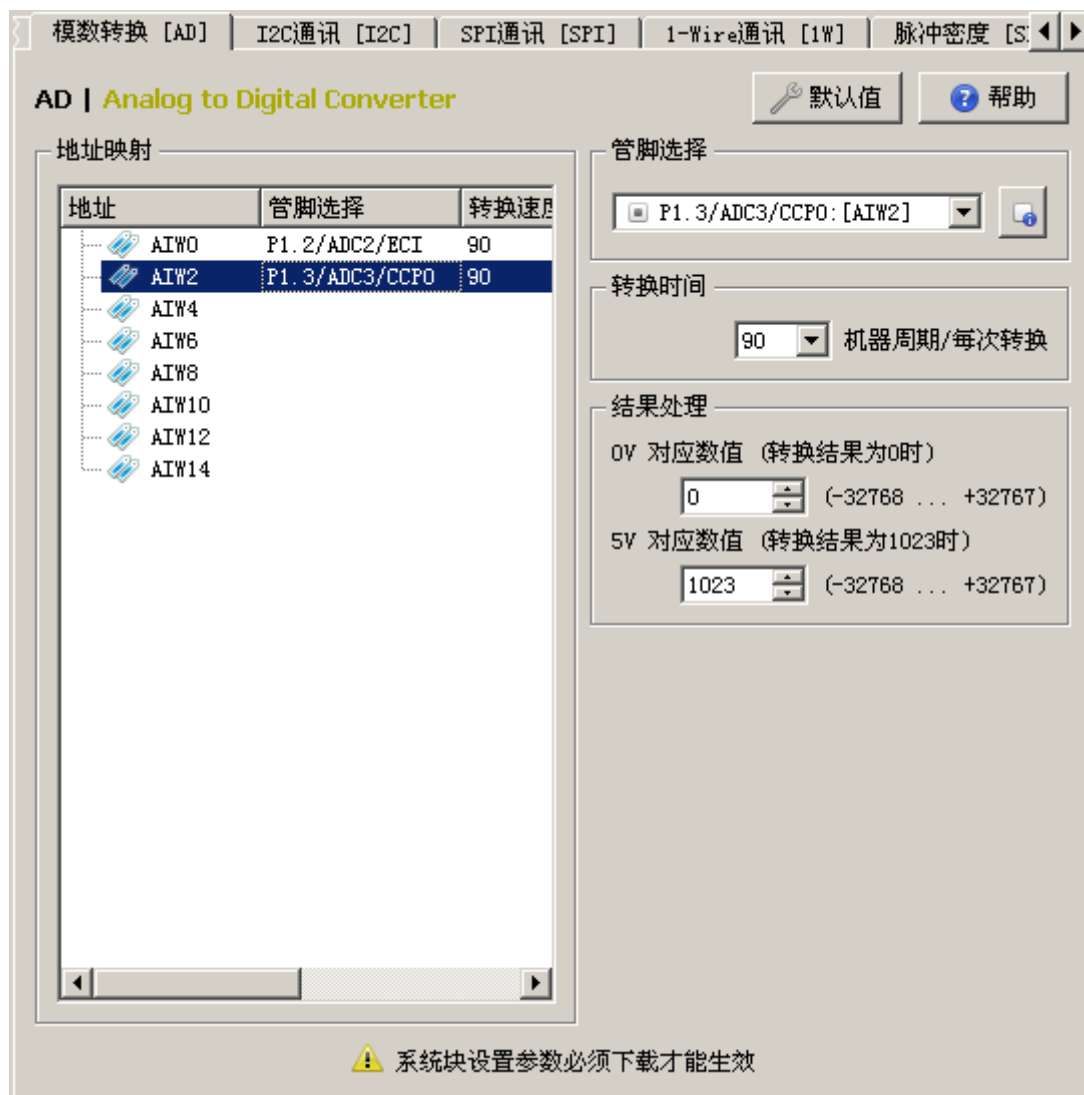
配置 PLC 类型为 EC30-EK51:



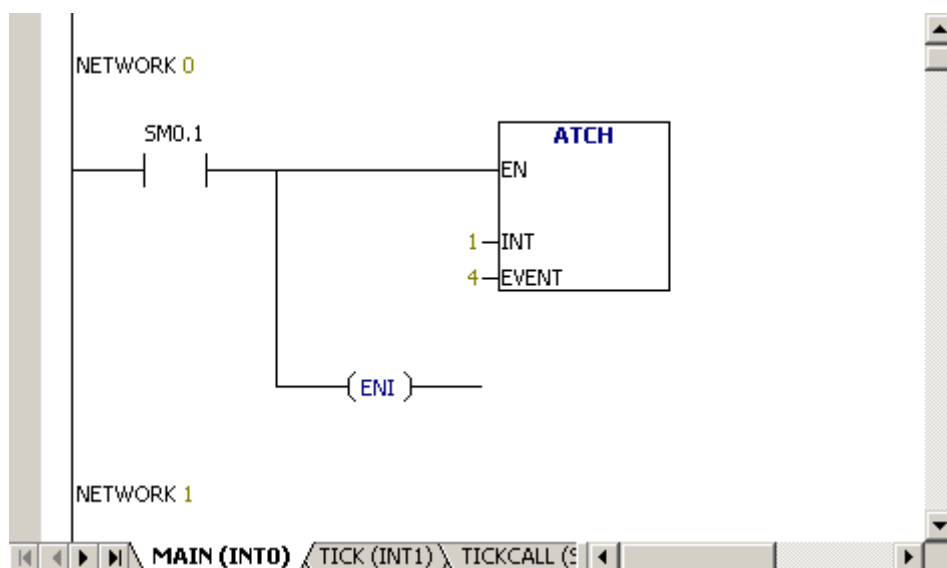
回车或点击确认，GUTTA Ladder Editor 会将当前项目转换成 EC30-EK51 项目。项目管理的项目后面也会跟随 EC30-EK51 字符串。这时双击项目管理的系统块，将会出现 EC30-EK51 的硬件配置对话框：



下图展示的是这个对话框，片上 A/D 转换器的配置。



应用逻辑的编写，借助于软件 GUTTA Ladder Editor 1.1，完全采用梯形图（或指令表）的方式进行编写。只要系统拥有一个 RS232 通讯口，您就能完成程序的下载和监控（EC30-EK51 不支持程序的上传）。



第 3 章 硬件方案

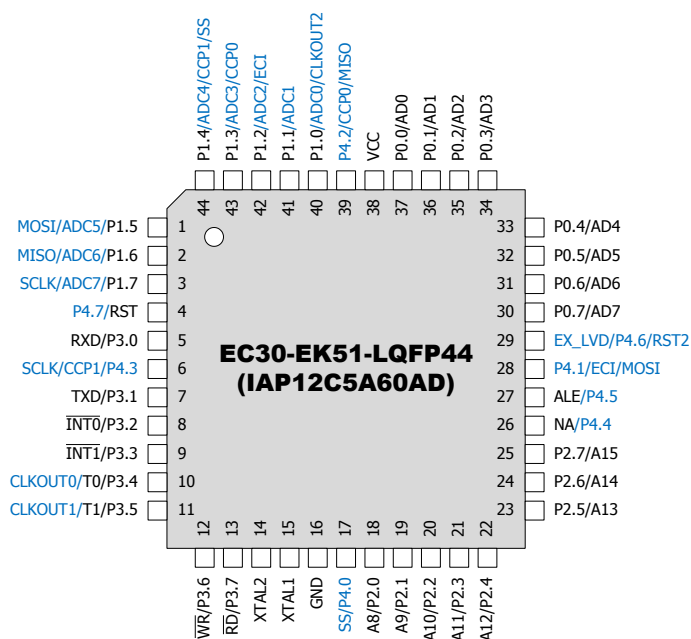
芯片封装和管脚

EC30-EKSTM32 提供三种封装形式的芯片，分别是：

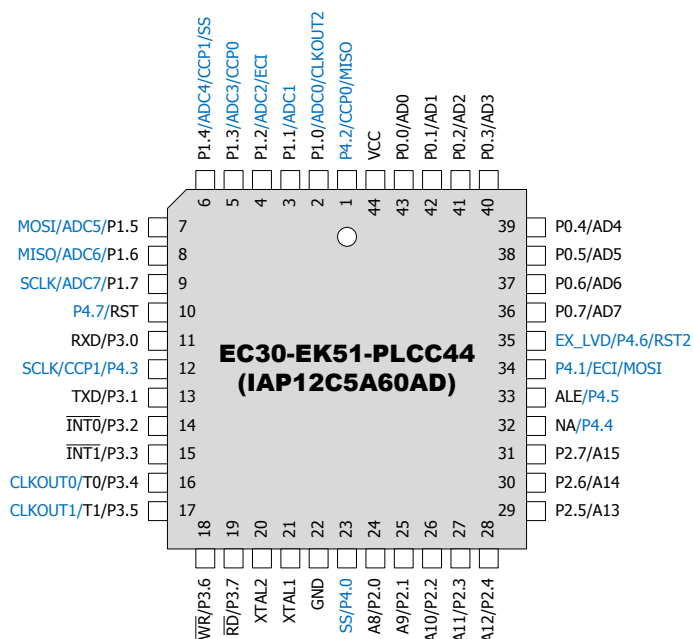
- **LQFP44** 订货号为 EC30-EK51-LQFP44。
- **PLCC44** 订货号为 EC30-EK51-PLCC44。
- **PDIP40** 订货号为 EC30-EK51-PDIP40。

3 种封装的管脚排列

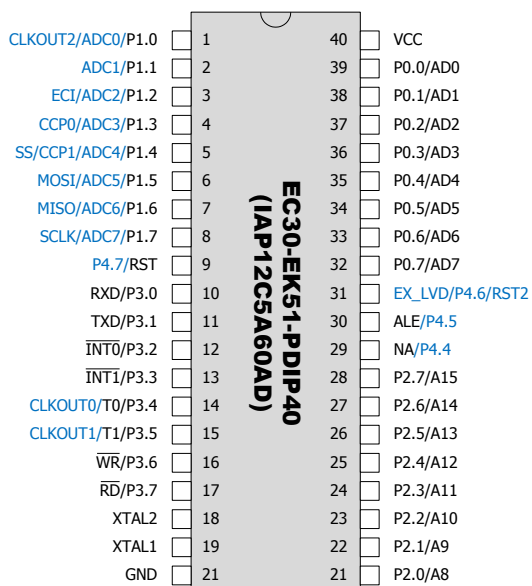
LQFP44 管脚排列



PLCC44 管脚排列



PDIP40 管脚排列

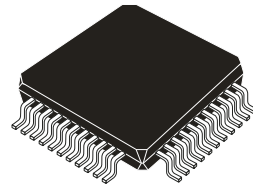


3 种封装的尺寸图

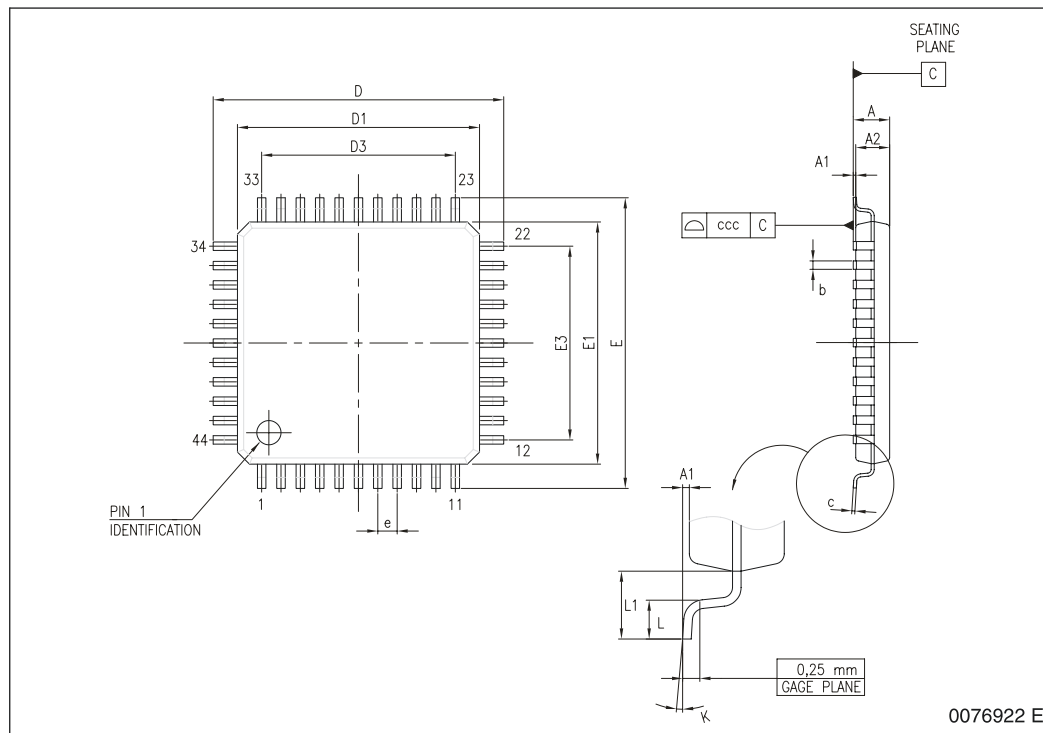
LQFP44 封装尺寸图

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			1.60			0.063
A1	0.05		0.15	0.002		0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
B	0.30	0.37	0.45	0.012	0.015	0.018
C	0.09		0.20	0.004		0.008
D	11.80	12.00	12.20	0.464	0.472	0.480
D1	9.80	10.00	10.20	0.386	0.394	0.401
D3		8.00			0.315	
E	11.80	12.00	12.20	0.464	0.472	0.480
E1	9.80	10.00	10.20	0.386	0.394	0.401
E3		8.00			0.315	
e		0.80			0.031	
L	0.45	0.60	0.75	0.018	0.024	0.030
L1		1.00			0.039	
k	0° (min.), 3.5° (typ.), 7° (max.)					
ccc			0.10			0.0039

OUTLINE AND MECHANICAL DATA



LQFP44 (10 x 10 x 1.4mm)



0076922 E

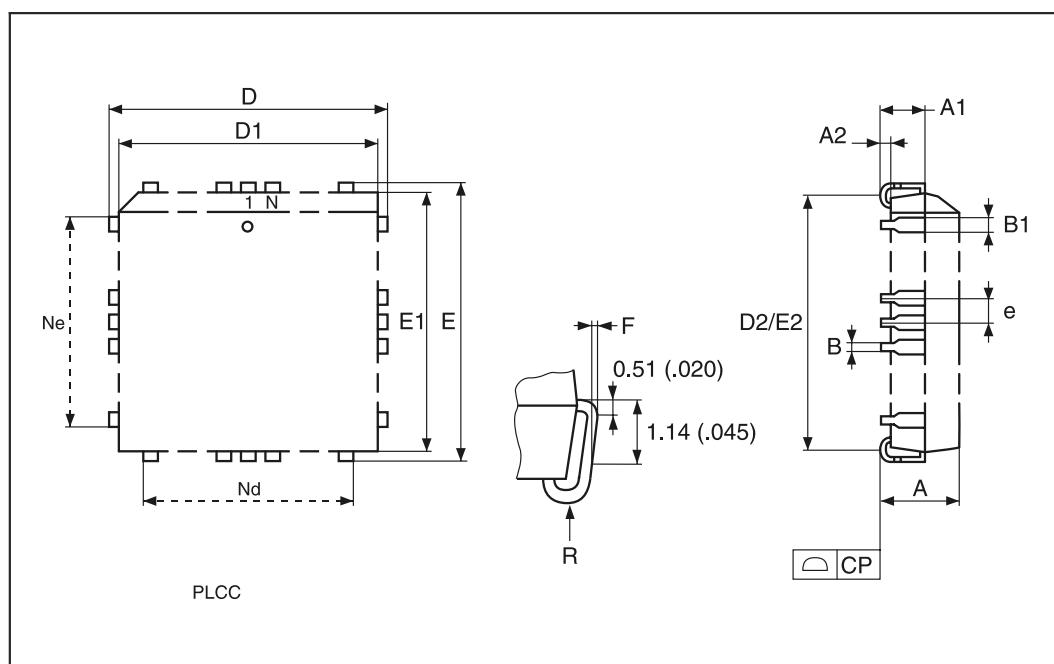
PLCC44 封装尺寸图

Symb	mm			inches		
	Typ	Min	Max	Typ	Min	Max
A		4.20	4.70		0.165	0.185
A1		2.29	3.04		0.090	0.120
A2		—	0.51		—	0.020
B		0.33	0.53		0.013	0.021
B1		0.66	0.81		0.026	0.032
D		17.40	17.65		0.685	0.695
D1		16.51	16.66		0.650	0.656
D2		14.99	16.00		0.590	0.630
E		17.40	17.65		0.685	0.695
E1		16.51	16.66		0.650	0.656
E2		14.99	16.00		0.590	0.630
e	1.27	—	—	0.050	—	—
F		0.00	0.25		0.000	0.010
R	0.89	—	—	0.035	—	—
N	44			44		
CP			0.10			0.004

44 pin Square Plastic Leaded Chip Carrier



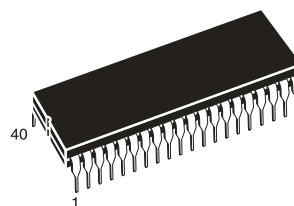
PLCC44



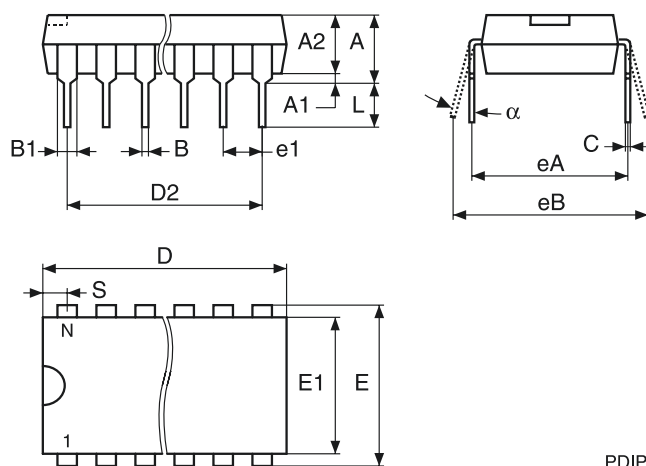
PDIP40 封装尺寸图

Symb	mm			inches		
	Typ	Min	Max	Typ	Min	Max
A	4.45	—	—	0.175	—	—
A1	0.64	0.38	—	0.025	0.015	—
A2		3.56	3.91		0.140	0.154
B		0.38	0.53		0.015	0.021
B1		1.14	1.78		0.045	0.070
C		0.20	0.31		0.008	0.012
D		51.78	52.58		2.039	2.070
D2	48.26	—	—	1.900	—	—
E		14.80	16.26		0.583	0.640
E1		13.46	13.99		0.530	0.551
e1	2.54	—	—	0.100	—	—
eA	15.24	—	—	0.600	—	—
eB		15.24	17.78		0.600	0.700
L		3.05	3.81		0.120	0.150
S		1.52	2.29		0.060	0.090
α		0°	15°		0°	15°
N	40			40		

40 pin Plastic DIP,
600 mils width



PDIP40



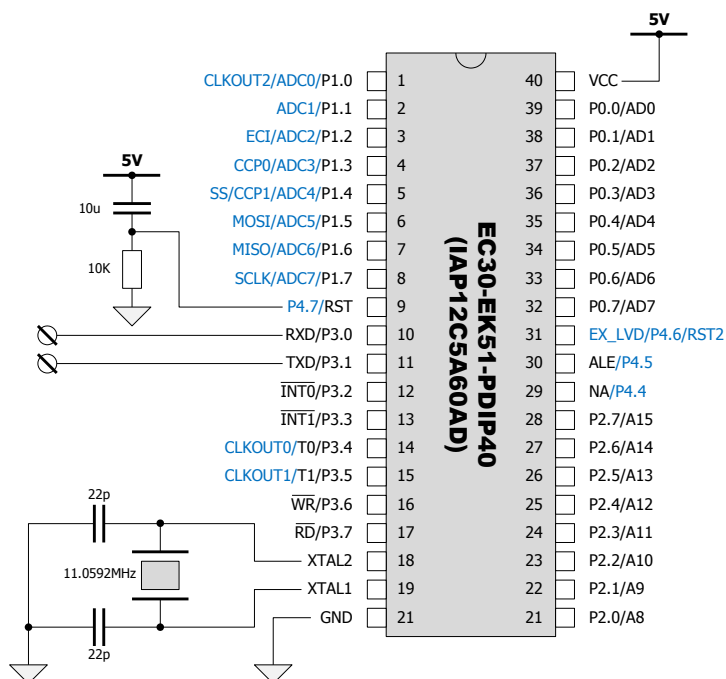
PDIP

运行条件

要让 EC30-EK51 正常工作，必须满足以下基本条件：

1. VCC 和 GND 之间存在稳定的 3.7 ~ 5V 供电电压。虽然 IAP12C5A60AD 的工作电压是 3.3 ~ 5V，但是 IAP 功能只在 3.7V 以上被使能。PLC 程序的下载需要芯片的 IAP 支持。
2. 复位电路。可以使用复位芯片或者是阻容回路，确保 NRST 管脚在上电后为高（复位状态），一段时间后转换为低（运行状态）。
3. 晶体振荡回路，晶体振荡器频率必须为 **11.0592MHz**。或者使用有源晶振，从 XTAL1 输入，XTAL2 保持悬空。EC30-EK51 整个系统基于 **11.0592MHz** 设计，不可变更，否则 PLC 定时器和通讯波特率会发生变化。

EC30-EK51 核运行的最小电路如下：



连接到计算机

将 EC30-EK51 的硬件配置信息和 PLC 程序由计算机下载到芯片，就必须建立计算机和芯片的通讯连接。EC30-EK51 系统在出厂时，串行通讯端口被配置为：

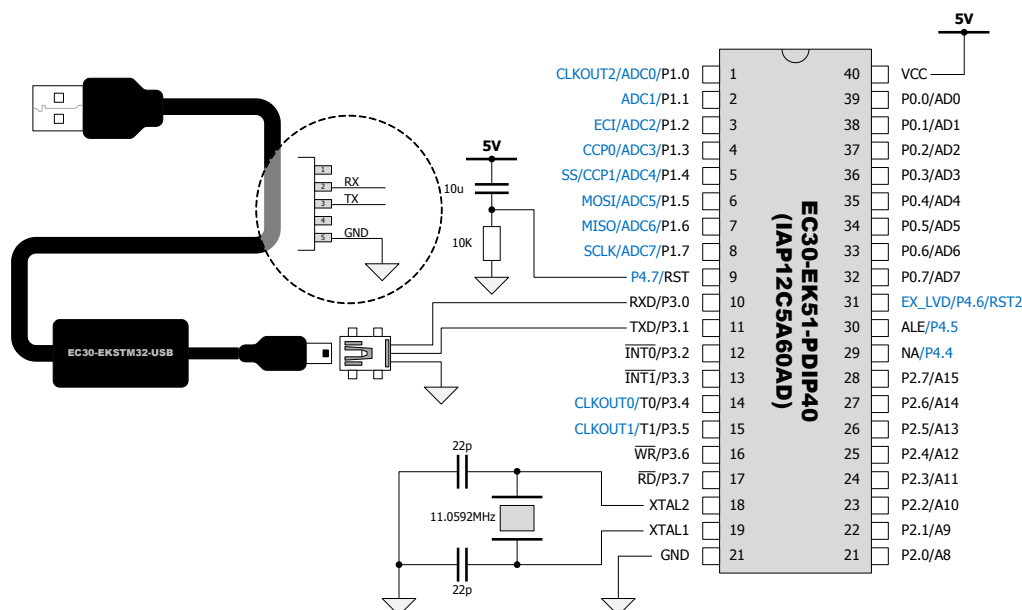
项	值
PLC 地址	1
波特率	38400 bps
数据位	8 (RTU)
奇偶校验	EVEN
停止位	1 Bit
响应超时	1000ms
帧间隔时间	10ms

第一次连接 EC30-EK51，GUTTA Ladder Editor 的通讯设置中，波特率为 38400 bps，奇偶校验为 EVEN。如果通讯线上只有 1 台 PLC，站号可以设置为 0。以后连接 EC30-EK51，需要根据上一次下载的 PLC 程序中，系统块的通讯设定来调整。

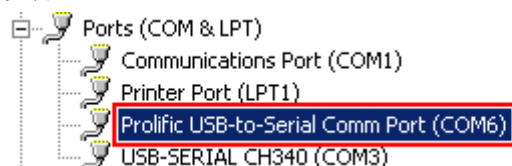
使用 EC30-EKSTM32-USB 通讯线

直接将 EC30-EK51 的 UART（异步串行）通讯管脚 RXD/P3.0 和 TXD/P3.1 从硬件线路板上引出而不做任何电器处理，是最节约成本的形式。计算机不能直接识别 RXD/P3.0 和 TXD/P3.1 上的 TTL 电平信号。如果需要通过 RXD/P3.0 和 TXD/P3.1 建立与计算机的连接，就需要设计专用的通讯电路。EC30-EKSTM32-USB 是通过计算机的 USB 接口，连接 EC30-EKSTM32 核心板的专用通讯线，它从电气和协议上实现了 USB 通讯到 UART 通讯的转换。基于这个功能，完全可以使用 EC30-EKSTM32-USB 通讯线，建立计算机与 EC30-EK51 的通讯。

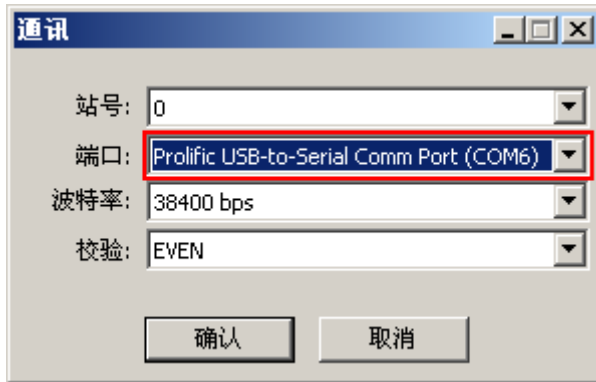
使用 EC30-EKSTM32-USB 通讯线连接到计算机：



EC30-EK51 的电路板上，需要焊接一个 Mini5-USB 插座，其中 2 脚连接 RXD/P3.0，3 脚连接 TXD/P3.1，5 脚连接到 5V 电源地。1 脚为 EC30-ESTM32-USB 的 3.3V 供电，保持悬空。将 EC30-ESTM32-USB 连接到计算机，安装好 USB 驱动后，便可在设备管理器中找到这个 USB 设备：



COM 端口号随计算机不同会分配不同的值。使用 GUTTA Ladder Editor 软件下载硬件配置和 PLC 程序时，在通讯设置中，选用这个端口即可。



使用 RS232 通讯器件

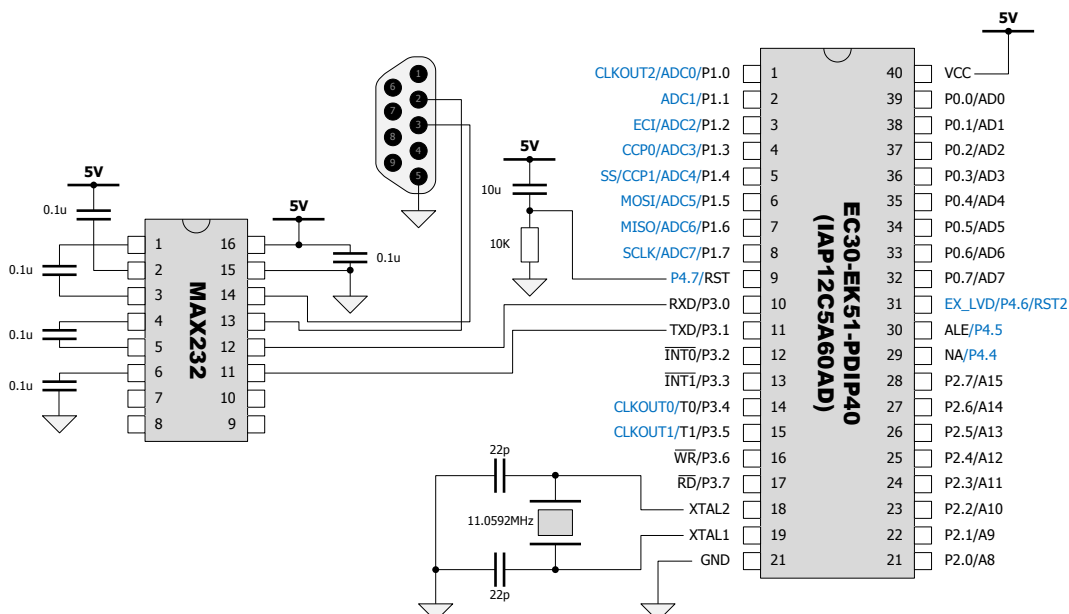
若 EC30-EK51 的硬件系统本身具备 RS232 通讯电路，产品在正常工作的时候需要通过 RS232 和其他设备通讯（例如触摸屏），我们可以借用这个通讯口作为 PLC 程序的下载口。

若 EC30-EK51 硬件产品被设计为 MODBUS 通讯子站，那么在任何时候都可以通过这个通讯口下载 PLC 程序。因为 PLC 程序的下载使用 GUTTA 通讯协议，GUTTA 通讯协议通过 MODBUS 的 13 号功能封装，与标准的 MODBUS 协议完全兼容。

若 EC30-EK51 硬件产品被设计为 MODBUS 通讯主站，只要 EXCH 指令被调用，EC30-EK51 的通讯口将作为主站主动访问其他子站，这个时候，通讯口无法作为 PLC 程序的下载端口。如果需要灌入新的硬件配置信息和 PLC 程序，有两种可行操作。

- 有条件的进行 EXCH 指令的调用，例如设备在停止状态时，不执行 EXCH 指令操作。如果 EXCH 指令维护的通讯队列为空，EC30-EK51 自动恢复为 MODBUS 通讯子站。
- 利用 GUTTA Flash Utility 软件重新初始化 EC30-EK51。

使用 RS232 通讯器件连接到计算机：



RS232 的 DB9 通讯头建议使用公头。因为计算机也是公头，而 2、3 线交叉的双母头连接线是一个标准的型号易于购买。

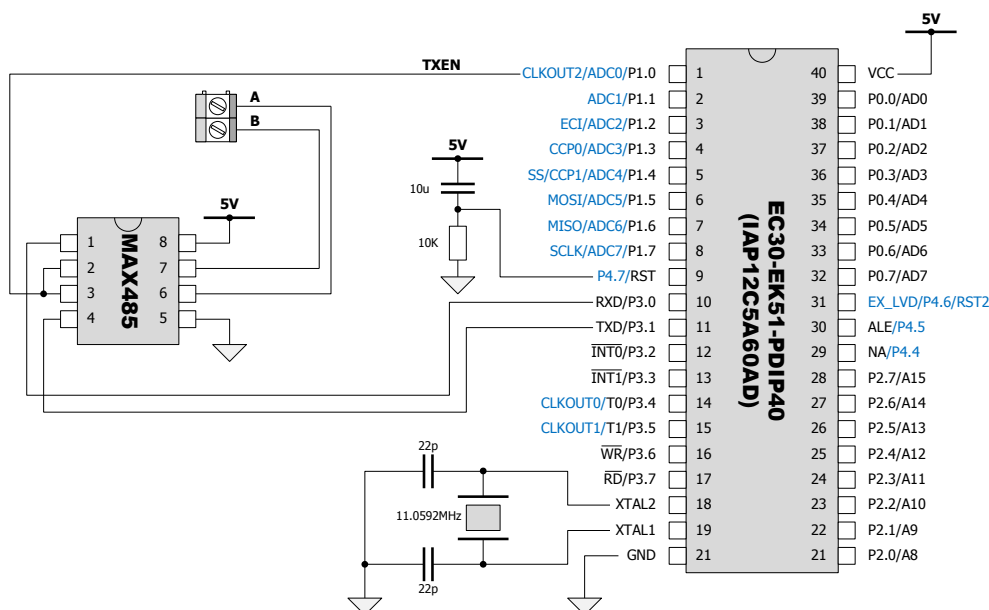
使用 RS485 通讯器件

若 EC30-EK51 的硬件系统本具备 RS485 通讯电路，产品在正常工作的时候需要通过 RS485 和其他设备通讯（例如触摸屏），我们可以借用这个通讯口作为 PLC 程序的下载口。

使用标准的 RS485 收发器，除了 RXD/P3.0 和 TXD/P3.1，还需要额外配置一个发送使能管脚 TXEN。EC30-EK51 系统没有强制规定哪个管脚用于 TXEN，需要在系统块中自行配置（参考 MISC 模块说明）。系统块未配置之前，RS485 通讯是无法工作的，这就要求能够通过其他方式将这份配置信息下载到 EC30-EK51 中。以后再通过 RS485 下载这份程序时，只要新的程序依然正确的包含了 TXEN 的配置，就和使用 RS232 没有区别。

EC30-EK51 做主站时 PLC 程序的下载可参考使用 RS232 通讯器件的相关内容。

使用 RS485 通讯器件连接到计算机：



系统

关键指标

核心	8051
频率	11.0592MHz
SRAM	1.25K (1024+256)
FLASH	60K
PLC 名称	EC30-EK51
PLC 信息	EC30-EK51
系统页大小（字节）	355
数据页数量	16
数据页数据项数量	16

中断程序个数	8
子程序个数	8
中断程序参数个数	32
子程序参数个数	32
常数区大小（字节）	256
指令区大小（字节）	15360 (15K)
通讯包有效数据长度（字节）	64
最大程序嵌套层数	4
是否支持单步调试	否

编译系统 PLC 程序的代码密度随 PLC 指令的不同略有变化。EC30-EK51 一共分配了 15K FLASH 空间用于存放 PLC 程序，实际对应的 PLC 指令约为 1200 条。

内存使用

EC30-EK51 变量分区

MODBUS 地址	槽号	区域标识	区域说明	变量偏移单位	位访问	字节访问	字访问	双字访问	取地址	取值	取指针
输入线圈(1x)	0	I	数字量输入	BYTE	√	√	√	√		√	
保持线圈(0x)	1	Q	数字量输出	BYTE	√	√	√	√		√	
输入寄存器(3x)	2	AI	模拟量输入	BYTE		√	√	√		√	
保持寄存器(4x)	3	AQ	模拟量输出	BYTE		√	√	√		√	
	4	M	普通内存	BYTE	√	√	√	√	√	√	√
	5	T	定时器专用	WORD			√			√	
	6	C	计数器专用	WORD			√			√	
	7	SM	系统内存	BYTE	√	√	√	√	√	√	
	8	J	流程控制专用	BYTE		√				√	
常数区域	9	K	常数区域	BYTE	√	√	√	√		√	
临时区域	10	L	临时区域	BYTE	√	√	√	√	√	√	√

EC30-EK51 变量分区大小

区域	MODBUS 地址开始	MODBUS 地址结束	长度（字节）	范围
I	100001	100064	8	IB0~IB7
Q	000001	000064	8	QB0~QB7

AI	300001	300012	24	AIB0~AIB23
AQ	400001	400012	24	AQB0~AQB23
M	400013	400140	256	MB0~MB255
T	400141	400172	64	T0~T31
C	400173	400180	16	C0~C7
SM	400181	400188	16	SMB0~SMB15
J	400189	400196	16	J0~J15
K			256	--
L			32	LB0~LB31

2010/09/20 日对 EC30-EK51 的变量分区大小作了调整, 主要有:

- AI、AQ 由以前的 16 个字节调整到了 24 个字节。
- M 由以前的 128 个字节调整到了 256 个字节。
- T 由以前的 32 个字节(16 个可用定时器)调整到了 64 个字节(32 个可用定时器)。
- K 由以前的 128 个字节调整到了 256 个字节。

编译型 PLC 一旦修改了变量分配, 需要同步更新 GUTTA Ladder Editor 软件的 PLC 类型描述, 因此在 2010/09/20 以后购买固件的用户, 请同步升级 GUTTA Ladder Editor 软件。

EC30-EK51 可用系统内存

变量名	数据类型	说明
SM0.0	BIT	该位始终为 1。
SM0.1	BIT	该位在首次扫描时为 1, 用途之一是调初始化子程序。
SM0.4	BIT	该位提供了一个时钟脉冲, 30 秒为 1, 30 秒为 0, 周期为 1 分钟。它提供了一个简单易用的延时, 或 1 分钟的时钟脉冲。
SM0.5	BIT	该位提供了一个时钟脉冲, 0.5 秒为 1, 0.5 秒为 0, 周期为 1 秒钟。它提供了一个简单易用的延时, 或 1 秒钟的时钟脉冲。
SM0.6	BIT	该位为扫描时钟, 本次扫描时置 1, 下次扫描时置 0。可用作扫描计数器的输入。
SM3.5	BIT	PTO 脉冲发送完成标志。
SMD4	UDINT	PTO 当前已发送的脉冲个数。

可用中断源 (ATCH/DTCH 指令)

EVENT	NAME	EC30-EK51
0	INT_EVENT_INPUTP0	EXT0
1	INT_EVENT_INPUTP1	EXT1
2	INT_EVENT_TIMER0	TIMER1_OVF
3	INT_EVENT_TIMER1	TIMER1_OVF
4	INT_EVENT_TIMER_SECOND	×
5	INT_EVENT_INPUTP2	×

6

INT_EVENT_INPUTP3



硬件参考

试验板：CPU-EC20 (8051)



EC30-EK51 的大部分驱动都可以在 CPU-EC20 (8051) 试验板上进行测试。CPU-EC20 (8051)试验板本身是一款面向 PLC 学习、单片机开发、PLC 开发的教学产品。随着 EC30-EK51 的推出，目前购买 CPU-EC20 (8051)试验板，**出厂时默认系统是 EC30-EK51**。原 CPU-EC20 (8051)系统固件随光盘提供。您可以在 CPU-EC20 (8051)上测试 EC30-EK51 的绝大部分模块。若需要学习单片机，可以通过 STC-ISP 写入自己的固件文件。若需要学习 PLC 编程，可以通过 STC-ISP 写入光盘提供的 CPU-EC20 (8051)系统固件。EC30-EK51 系统固件被冲掉，将不可恢复，此时试验板降级为 CPU-EC20 (8051)。

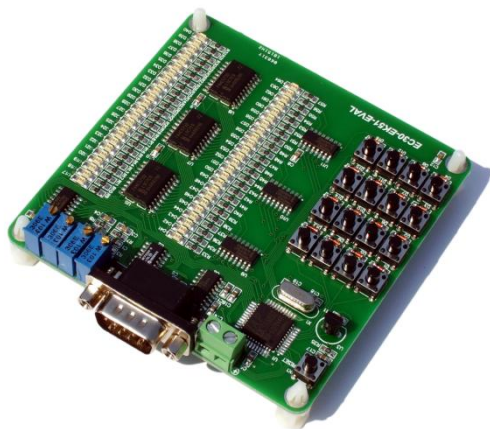
试验板：CPU-EC20 (8051)的详细介绍：

<http://www.plcol.com/products/boardof8051.htm>

试验板：CPU-EC20 (8051)的购买地址：

<http://www.plcol.com/products/buy.htm#cpuec208051>

试验板：EC30-EK51-EVAL



由于 CPU-EC20 (8051)并不能完整的测试所有的驱动模块，为此我们专门设计了试验板：EC30-EK51-EVAL，用于测试一些在 CPU-EC20 (8051)中无法测试的驱动模块。由于试验板 EC30-EK51-EVAL 只被设计为测试驱动模块，网站并不销售此试验板，但是提供完整的电路原理图下载，以供用户参考。

系统恢复

EC30-EK51 为编译型系统，EC30-EK51 运行时，会直接调用 PLC 用户程序。PLC 系统固件不能保证 PLC 用户程序的安全性。如果 PLC 用户程序发生崩溃，将会导致整个系统崩溃。由于 PLC 完成启动配置后就会运行主扫描循环 (INT0)，然后执行错误程序而崩溃。这个时候通讯往往已经不能使用，您此时无法通过 GUTTA Ladder Editor 清除程序。PLC 因为一旦启动就总是崩溃了。

基于以上原因，EC30-EK51 在启动后，并不立即运行。而是等待一段时间 (200ms 左右)，观察通讯端口是否有指定的通讯数据流。如果有，进入 PLC 的高级配置模式，然后可进行 PLC 程序的清除 (同时清除了致命错误)。其实也就是在单片机崩溃之前，给了一个清除错误的机会。我们可以使用软件 GUTTA Flash Utility 来发起这个通讯。

系统恢复步骤：

1. 连接计算机和 EC30-EK51 芯片。

连接方式可以是上面描述的

使用 EC30-EKSTM32-USB 通讯线、

使用 RS232 通讯器件。

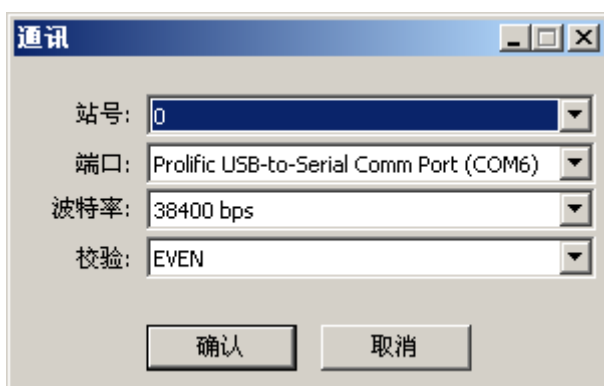
RS485 的半双工通讯方式不能满足 GUTTA Flash Utility 的要求。

2. 运行 GUTTA Flash Utility 软件。



选择启动工具。

在通讯设置中设置好计算机到 EC30-EK51 的通讯端口：



通讯参数使用默认的 0、38400 bps、EVEN。

3. 启动 GUTTA Flash Utility 数据发送。



点击“擦除编译”按钮，让 GUTTA Flash Utility 进入数据发送状态。此时可以看到软件下方的进度条发生移动，表明 GUTTA Flash Utility 正在向连接 EC30-EK51 的通讯端口发送数据请求。

4. 复位 EC30-EK51 芯片。

在 GUTTA Flash Utility 发送数据请求时(进度条在移动时),复位 EC30-EK51 芯片。若目标硬件无 Reset 按键，给整个目标硬件重新上电也可以。EC30-EK51 系统复位后，会响应 GUTTA Flash Utility 的发送数据：

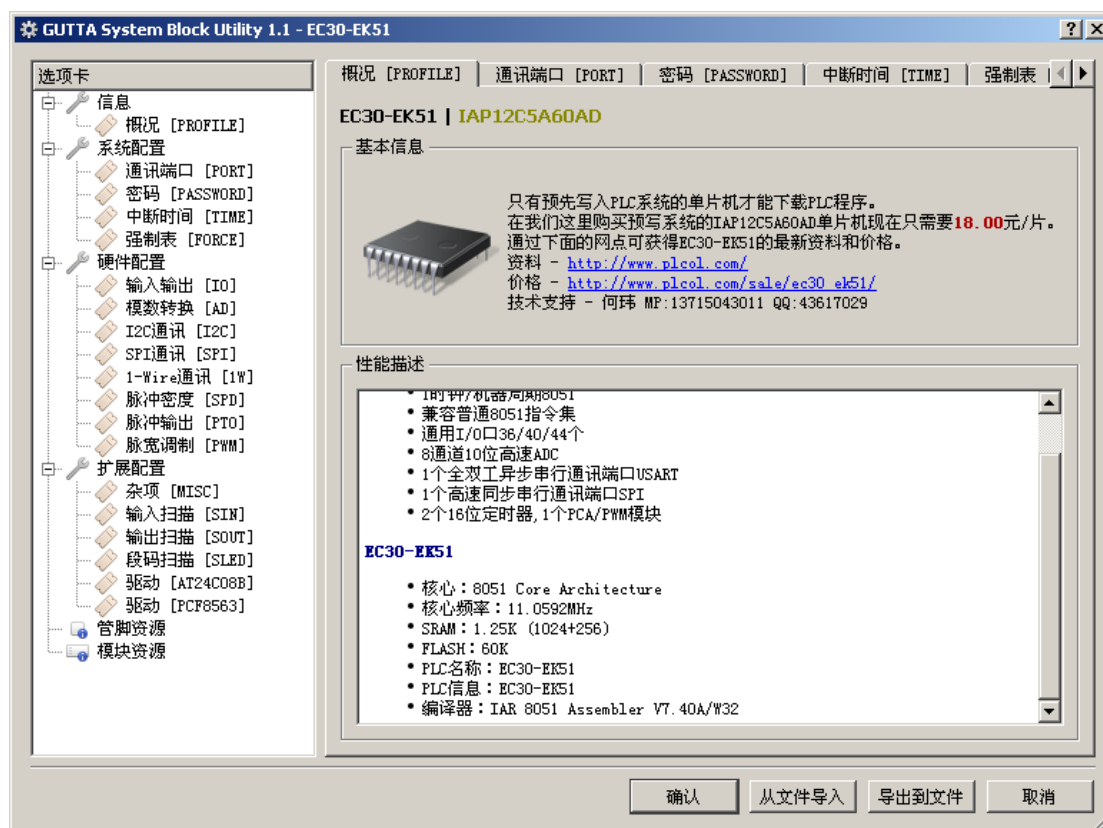


若 Gutta Flash Utility 显示出以上信息，则说明当前用户程序被清除，EC30-EK51 恢复成空程序状态。

第 4 章 发布自己的 PLC 系统

使用 EC30-EK51 来开发基于 51 内核单片机的 PLC 系统，您可以将上述开发过程的第 4 点完全交由控制器的使用者来处理。您可以将更多的精力放在硬件的设计和和生产上，而不需要过多的关注用户具体的控制逻辑和工艺流程。这就意味着，您在销售产品的同时必须向用户提供 Gutta Ladder Editor 软件。基于 EC30-EK51 内核，您向用户发布 Gutta Ladder Editor 软件不需要支付额外的费用，但是您可能不希望向用户开放硬件配置信息，或者您希望拥有一份 Gutta Ladder Editor 的特别版本，这些我们都提供相关支持。

一个不产生额外费用且简单易行的办法就是重新配置 SystemBlockDll.dll 程序。这个程序负责 EC30-EK51 系统块的编辑。在 Gutta Ladder Editor 软件的安装目录中，存在一个 GuttaLad 文件夹。此文件夹保存了所有的 PLC 类型信息。在 EC30-EK51 子文件夹中，可以找到这个 SystemBlockDll.dll 程序。如何配置 SystemBlockDll.dll 程序呢？为了生成配置文件，我们必须按照常规方式运行 Gutta Ladder Editor 软件，选择 PLC 类型为 EC30-EK51。编辑项目的系统块，便可以看到标准版本的系统配置界面：



按照常规的开发流程，在 GUTTA System Block Utility 中配置好单片机的硬件后，便可以开始编写梯形图逻辑了。但是不希望用户每次都执行同样的硬件配置工作，毕竟一款 PLC 产品稳定后，硬件设计在相当长的一段时间内都是保持不变的。为了避免重复劳动，在对话框中可以点击“导出到文件”按钮，将硬件配置保存到额外的文件中。我们将这个文件取名为 **SystemBlockUtility.vcb**，并让这个文件和 **SystemBlockDll.dll** 位于同一文件夹中（即 **EC30-EK51** 文件夹）。

重新运行 GUTTA Ladder Editor 软件（此操作不可省略），新建一个 EC30-EK51 工程，编辑项目的系统块，看看这个时候的硬件配置，是否和上次保存的硬件配置一致呢？是的，只要将硬件配置导出到 **EC30-EK51** 文件夹中且文件名为 **SystemBlockUtility.vcb**，那么 GUTTA Ladder Editor 软件在启动时，便会将这份硬件配置作为默认的 EC30-EK51 硬件配置，这样您的最终用户就不用每次都配置这份相同的硬件了。

更进一步，您甚至不希望您的用户看到这些硬件配置。毕竟这些东西比较复杂而且让人迷惑。万一被好奇的用户不正确的修改将会造成 PLC 产品出现错误。幸运的是，您可以选择隐藏这些配置。用记事本打开 **SystemBlockUtility.vcb** 文件，您应该可以看到下面格式的文本：


```
1 1:0001050102000A0A0001050102000A0A
2 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
3 1:C8C8
4 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
5 1:0004FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
6 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
7 1:FFFFFFF
8 1:FFFFFFF
9 1:FFFFFFF
10 1:FF
11 1:FFFF
12 1:FFFFFFF
13 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
14 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
15 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
16 1:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
17 1:FFFF
18 1:FF
```

可以看到，每行都是以“1:”开始，将每行开头所有的“1”都修改为“0”，保存这个文件并重新运行 GUTTA Ladder Editor 软件，新建一个 EC30-EK51 工程，编辑项目的系统块，您会看到配置对话框变成下面这个样子：



除了概况页，其余的配置页全部被隐藏了。SystemBlockUtility.vcb 每行第一个数字“1”或者是“0”决定了此配置页是否用户可见。基于这个机制，您可以任意选择哪些配置页是用户可见，哪些是用户不可见的。您应该在这仅有的概况页看到了 EC30-EK51 的介绍信息和本人的联系方式，也许您不希望这份信息出现在您的产品当中。将这些文字替换成您期望的信息将是非常有帮助的。实现的方法也很简单，用记事本创建一个 XML 文本文件 SystemBlockUtility.LabelLink.xml，并让这个文件和 SystemBlockDll.dll 位于同一文件夹中（即

EC30-EK51 文件夹)。然后在这个 XML 文件中编辑如下信息：

```
<b>ARDLABS</b>
<hr/>
<ul>
  <li>地址：北京市海淀区长春桥路5号新起点嘉园10号楼2107室</li>
  <li>电话：010-82561399，82561362，82561363</li>
  <li>传真：010-82561320，82561321</li>
  <li>邮编：100089 </li>
  <li>信箱：postmaster@ardlabs.com </li>
</ul>
```

保存这个文件并重新运行 GUTTA Ladder Editor 软件，新建一个 EC30-EK51 工程，编辑项目的系统块，您会看到配置对话框变成下面这个样子：



EC30-EK51 硬件配置是针对工业控制的一般性需求来设计的，因此无法涵盖所有可能的应用，而只是选择性的实现了一部分常用应用的驱动。如果您的应用比较复杂，建议使用 EC30-EKSTM32。EC30-EKSTM32 采用更高性能的 Cortex-M3 32 位处理器核心（72MHz、20K SRAM）。虽然价格略高，却在运算速度、内存容量、外设能力上高出 EC30-EK51 一个数量级，具有很高的性价比。更重要的是 EC30-EKSTM32 提供了一个开放的二次开发接口。您除了可以用软件配置的方式使用常用的工业控制驱动、用梯形图编写控制逻辑之外，您还可以用 C 语言编写您自己的驱动模块、应用逻辑并添加自己特殊的 PLC 指令。EC30-EKSTM32 提供完整的 IAR 或 KEIL 开发工程模板。在此模板上，您可以完成其它常规单片机开发可以完成的应用，并结合片上已经预先写入的 PLC 系统，将产品发布给最终使用者。

EC30-EKSTM32 的详细信息，请参考网站 <http://www.plcol.com/>。

第 5 章 系统配置说明

通讯端口 [PORT]

通讯端口 PORT 用于配置单片机的异步通讯串口 UART。由于用户 PLC 程序的下载和监控都依赖于这个串口，因此作为 PLC 系统，必须支持至少一个串口通讯。IAP12C5A60AD 只有一个硬件串口 UART。和绝大多数 51 单片机一样，P3.0/RXD 用于异步串行通讯口的接收管脚；P3.1/TXD 用于异步串行通讯口的发送管脚。和标准 51 内核不同的是，IAP12C5A60AD 的 UART 具有一个独立的波特率发生器（可选）。EC30-EK51 使用了这个波特率发生器，这样就将定时器/计数器 0 解放出来并用于其它功能（高数计数或者是硬件 PWM 的时钟源，可参考 SPD、PWM 模块）。

由于串口是必须使用的（否则根本不能下载单片机配置信息和用户逻辑 PLC 程序），PORT 模块自动配置单片机的 UART 硬件，同时自动配置 P3.0/RXD 和 P3.1/TXD 这两个管脚。

可通过**模块资源**对话框查看模块使用情况：

	配置	共享
☒ UART	PORT	
☐ TIMER/COUNTER0		
☒ TIMER/COUNTER1	SYSTEM	
☐ PCA/PWM		

可通过**管脚资源**对话框查看管脚使用情况：

	LQFP-48	QFP-44	PLCC-44	PDIP-40	配置
☒ RST/P4.7	05	04	10	04	
☒ P3.0/RXD	06	05	11	05	PORT. RX
☐ P4.3/CCP1/SCLK	07	06	12		
☒ P3.1/TXD	08	07	13	06	PORT. TX
☐ P3.2/INT0	09	08	14	07	

UART 通讯的物理层建议为 RS232，这样就可以很方便的与计算机通讯。也可使用 UART 到 USB 的转换芯片并通过连接线和计算机的 USB 相连，这对于目前很多不具备 COM 通讯口的笔记本来说是一个好事。PORT 通讯口除了用于 PLC 程序的下载和监控，还可以用于 MODBUS 通讯。详细情况可以参考文档《UM4001 GUTTA 通讯协议》和《AN2002 通讯系统的应用》。

UART 通讯的物理层可以为 RS485，但是必须保证能够有其他方式将第一份程序下载到单片机中，这个所谓的第一份程序包含了 RS485 通讯使能管脚的配置信息。有了这份配置信息，才能够正常使用 RS485。详细情况可以参考 MISC 模块的相关内容。

在 GUTTA System Utility 中，PORT 模块的配置窗口如下：

默认值 帮助

	端口0	端口1	
通讯协议	Modbus	Modbus	
PLC地址	1	1	(范围 0... 255)
波特率	38400 bps	38400 bps	
数据位	8 (RTU)	8 (RTU)	
奇偶校验	EVEN	EVEN	
停止位	1 Bit	1 Bit	
响应超时 (100ms)	10	10	(范围 1... 255)
帧间隔时间 (1ms)	10	10	(范围 1... 255)

⚠ 系统块设置参数必须下载才能生效

- 端口 0：对应单片机的第 1 个 UART 模块。
- 端口 1：对应单片机的第 2 个 UART 模块，IAP12C5A60AD 无此硬件。
- 通讯协议：EC30-EK51 只能为 MODBUS。
- PLC 地址：通讯端口 PORT 默认支持 MODBUS 通讯协议的 01、02、03、04、05、06、15、16 子站功能。PLC 地址就是 PLC 的 MODBUS 子站站号。此通讯口也可以被 PLC 梯形图程序编程为主站，此时站号被忽略。
- 波特率：1200 ~ 38400 BPS。
- 数据位：EC30-EK51 只能为 8 (RTU)。
- 奇偶校验：可以是无校验 (NONE)、奇校验 (ODD)、偶校验 (EVEN)。
- 停止位：标志一个字节数据发送的结束，用于异步串口通讯的同步。
- 响应超时：通讯端口 PORT 可以通过 PLC 编程为主站，这个时间是主站发送请求后，等待从站响应的最长时间。
- 帧间隔时间：通讯端口 PORT 用于判断当前帧通讯结束和下一帧通讯开始的总线静寂时间。

密码 [PASSWORD]

在 GUTTA PLC 系统中，为了保护用户 PLC 程序不被恶意读取，PLC 有两种状态：非登

录状态和登陆状态。在非登陆状态，通讯口 PORT 自动作为 MODBUS 通讯子站，也可以通过 PLC 程序编程为 MODBUS 通讯主站，但是几乎所有的 GUTTA 通讯指令都不支持，这就意味着不能进行 PLC 程序的下载和上传，运行过程中的监控等。

在 GUTTA Ladder Editor 软件中，如果要进入在线监控、下载程序、上传程序，都需要通过一个密码输入框输入一个登陆密码（EC30-EK51 为了节约 PLC 系统固件的空间，没有实现 PLC 程序的上传协议）：



输入一个密码，点击确认后，GUTTA Ladder Editor 就会发送一段包含这个密码的通讯数据尝试登陆 PLC。PLC 系统比较登陆请求数据中的密码和系统块 PASSWORD 中的密码是否一致，如果一致，就进入登陆状态，否则返回一个错误的通讯代码并保持非登陆状态。一旦进入登陆状态，所有的 GUTTA 通讯协议都得到支持（只要这个 PLC 子系统实现了这个通讯功能号）。登陆后，用户可以进入在线监控模式，可以下载新的 PLC 程序。新的 PLC 程序可能在系统块中包含新的密码，那么下次登陆时，就必须用这个新的密码。

为了防止多个通讯口同时下载程序或者是监控 PLC 造成逻辑上的冲突，GUTTA 协议不允许重复登陆 PLC。也就是说 PLC 一旦被登陆，重复的登陆通讯指令将会返回一个错误代码（GUTTA Ladder Editor 软件提示接收代码错误）。因此在监控模式下，一旦发生意外掉线，必须复位 PLC，重新输入密码，才能继续建立连接。

在 GUTTA System Utility 中，PASSWORD 模块的配置窗口如下：



- **密码：**登陆密码，密码默认为空。
- **验证：**为了防止密码输入错误，需要重复输入一次。

中断时间 [TIME]

在 GUTTA PLC 系统中，有一套独立于单片机中断系统的中断管理系统。这套中断管理系统可以打断当前扫描任务的执行去执行中断任务，在中断任务执行结束后返回到当前扫描任务。一个低优先级的中断也可以被一个高优先级的中断抢占。和单片机硬件中断不同的是，

PLC 中断最小的执行单元是 PLC 指令，而不是单片机指令。只有在一条 PLC 指令执行结束后，下一条 PLC 指令开始执行时，才会进行中断的检查和切换。

GUTTA PLC 系统中有两个时间触发中断，用于激活一些需要定时扫描的周期性任务，和主循环不同，只要 CPU 有空闲，主循环 MAIN 总是在执行。在 TIME 模块中，可以设置这两个中断源激活的时间间隔。

在 GUTTA System Utility 中，TIME 模块的配置窗口如下：

- **时间中断 0**：设置中断源 0 的激活时间间隔。
- **时间中断 1**：设置中断源 1 的激活时间间隔。

强制表 [FORCE]

在 GUTTA Ladder Editor 的状态表中，可以在连线模式下修改 I/O 位变量的值，同时可以强制 I/O 位变量为 1 或者 0。一个数字量输入一旦被强制，实际的输入管脚电平被忽略，数字量输入的值始终为强制值。一个数字量输出一旦被强制，数字量输出被忽略，实际的输出管脚始终为强制值。在连线方式下强制某个变量是临时性的，一旦 PLC 被复位，当前强制都会被取消（主要用于在线调试）。如果需要长期性的对数字量输入输出变量进行强制，可以使用系统块的强制表。

在 GUTTA System Utility 中，FORCE 模块的配置窗口如下：

默认值 帮助

离散量输入

	0	1	2	3	4	5	6	7
I0.x	☒	☒	☒	☒	☒	☒	☒	☒
I1.x	☒	☒	☒	☒	☒	☒	☒	☒
I2.x	☒	☒	☒	☒	☒	☒	☒	☒
I3.x	☒	☒	☒	☒	☒	☒	☒	☒

离散量输出

	0	1	2	3	4	5	6	7
Q0.x	☒	☒	☒	☒	☒	☒	☒	☒
Q1.x	☒	☒	☒	☒	☒	☒	☒	☒
Q2.x	☒	☒	☒	☒	☒	☒	☒	☒
Q3.x	☒	☒	☒	☒	☒	☒	☒	☒

⚠ 系统块设置参数必须下载才能生效

- **离散量输入：**对数字量输入进行配置，中间状态表示不强制，选择表示强制为 1，不选择表示强制为 0。

控件状态	含义
☒	中间状态，不强制
☒	选择，强制为 1
☐	不选择，强制为 0

- **离散量输出：**对数字量输出进行配置，中间状态表示不强制，选择表示强制为 1，不选择表示强制为 0。

控件状态	含义
☒	中间状态，不强制
☒	选择，强制为 1
☐	不选择，强制为 0

第 6 章 硬件配置说明

输入输出 [IO]

简介

PLC 的数字量输入（DI，Digital Input）和数字量输出（DO，Digital Output）可以通过单片机的 I/O 管脚直接实现。IAP12C5A60AD 的绝大部分管脚都可以配置为数字量输入或数字量输出。输入管脚用于读取电平状态（高电平为 0、低电平为 1），输出管脚用于驱动外部器件（光耦或晶体管）。

IAP12C5A60AD 所有的 I/O 管脚均可以分别配置为以下 4 种工作类型的一种：

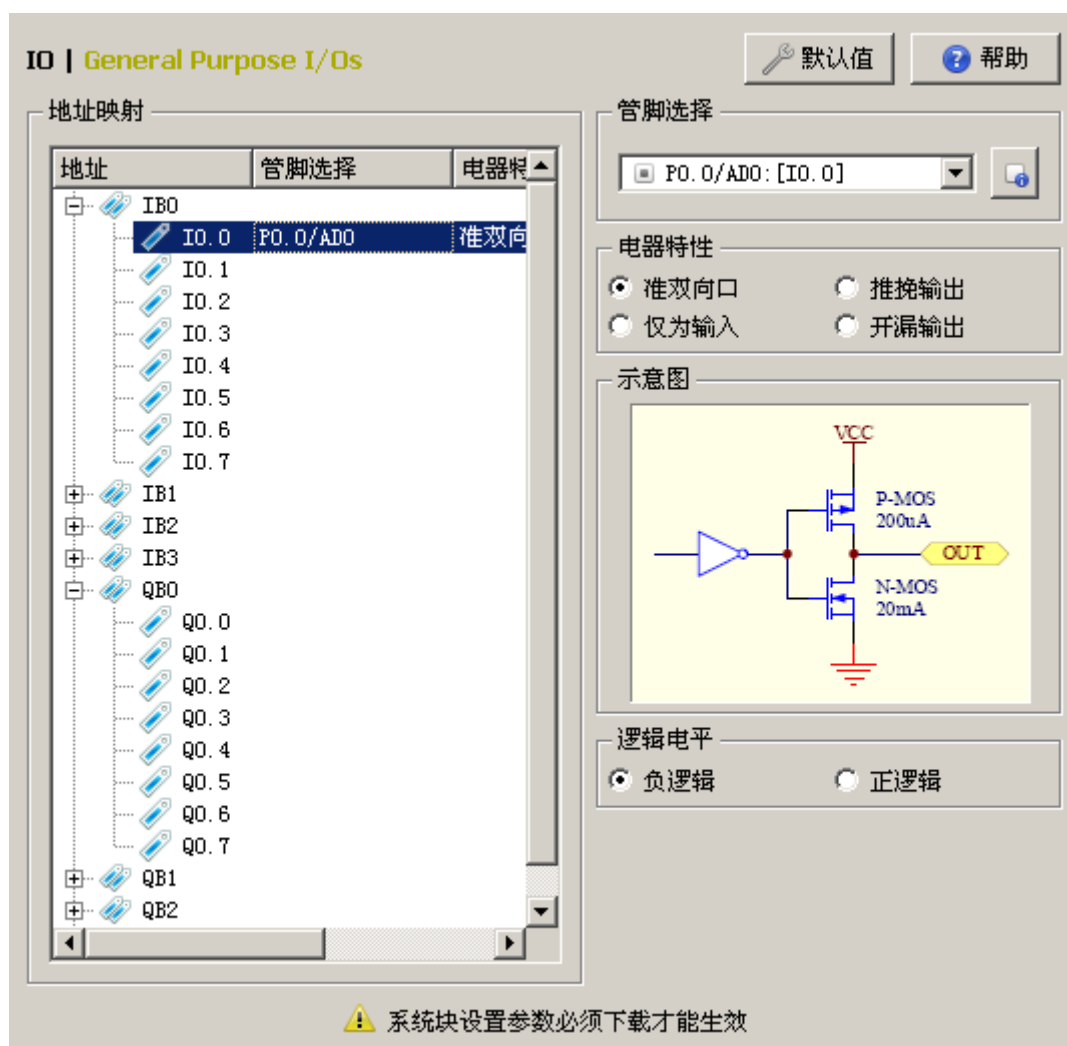
- 准双向口（8051 默认模式）

- 推免输出
- 仅为输入（高阻）
- 开漏输出。

IAP12C5A60AD 上电复位后管脚都是准双向口模式，在作为输入时，2V 以上为高电平，0.8V 以下时为低电平。4 种模式下管脚的详细电器特性请参考宏晶的《STC12C5A60S2 系列单片机器件手册》。

软件配置

在 GUTTA System Utility 中，IO 模块的配置窗口如下：



- **地址映射：**可以配置 IB0 ~ IB3 共 32 个数字量输入；可以配置 QB0 ~ QB3 共 32 个数字量输出。在地址映射控件中选择一个 PLC 输入（输出），例如图中的输入映射 IO.0，便可以在右边的控件中配置这个输入（输出）的具体特性。
- **管脚选择：**设置地址映射控件中选择的 PLC 输入（输出）对应的单片机管脚。IAP12C5A60AD 的绝大部分管脚都可以配置为数字量输入或者是数字量输出，除了 P3.0/RX0、P3.1/TX0、RST/P4.7、XTAL1、XTAL2、VCC、GND 这 7 个管脚。
- **电器特性：**设置地址映射控件中选中的 PLC 输入（输出）对应的单片机管脚的电器特性。

电器特性和 I/O 口的工作模式对应，即准双向口（8051 默认模式）、推免输出、仅为输入（高阻）、开漏输出这 4 种工作模式。每种工作模式下的电器参数请参考宏晶的《STC12C5A60S2 系列单片机器件手册》。

- **逻辑电平**：在单片机中，管脚寄存器位 1 一般表示管脚为高电平，寄存器位 0 一般表示管脚为低电平。但是由于输入隔离、采样电路或者是输出隔离、驱动电路的不同，实际的逻辑可能是翻转的。就输入光藕隔离来说，一般认为输入有电流（触点闭合）为逻辑值 1。输入有电流代表光藕输入端发光，光藕驱动端导通。若平时单片机管脚是上拉，光藕导通实际上是拉地的话，那么逻辑刚好是翻转的（触点闭合管脚反而是拉地即低电平）。这种情况下，可以在这里配置逻辑电平为负逻辑以符合使用习惯。

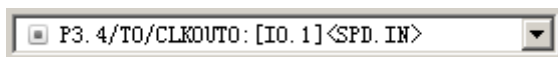
特别说明

在 GUTTA System Utility 的单片机模型中，每个管脚有配置属性和共享属性。一个被使用的管脚必须有一个配置属性；可以没有、有一个甚至多个共享属性。为了保证管脚的配置不发生冲突，通常一个管脚配置属性有且只能有一个。共享属性表示在 PLC 服务程序具体如何使用这个管脚，同一个管脚由于在不同的时候用于不同的功能。一个管脚允许被多个功能共享使用。

在管脚的下拉对话框中，每个管脚的描述格式是：

管脚名:[配置]<共享功能 1|共享功能 2|...|共享功能 N>

例如下图中的官脚表示：



管脚 P3.4/T0/CLKOUT0 被配置为 PLC 的输入 IO.1（定义了电器特性），同时被频率计模块 SPD 共享使用，作为频率计的输入端 IN。

使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道，单片机有 6 个管脚用于自锁按键的输入，有 4 个管脚用于轻触按键的输入，有 6 个管脚用于驱动 LED 作为试验板的输出。

根据输入输出的电路图，对应的软件配置如下：

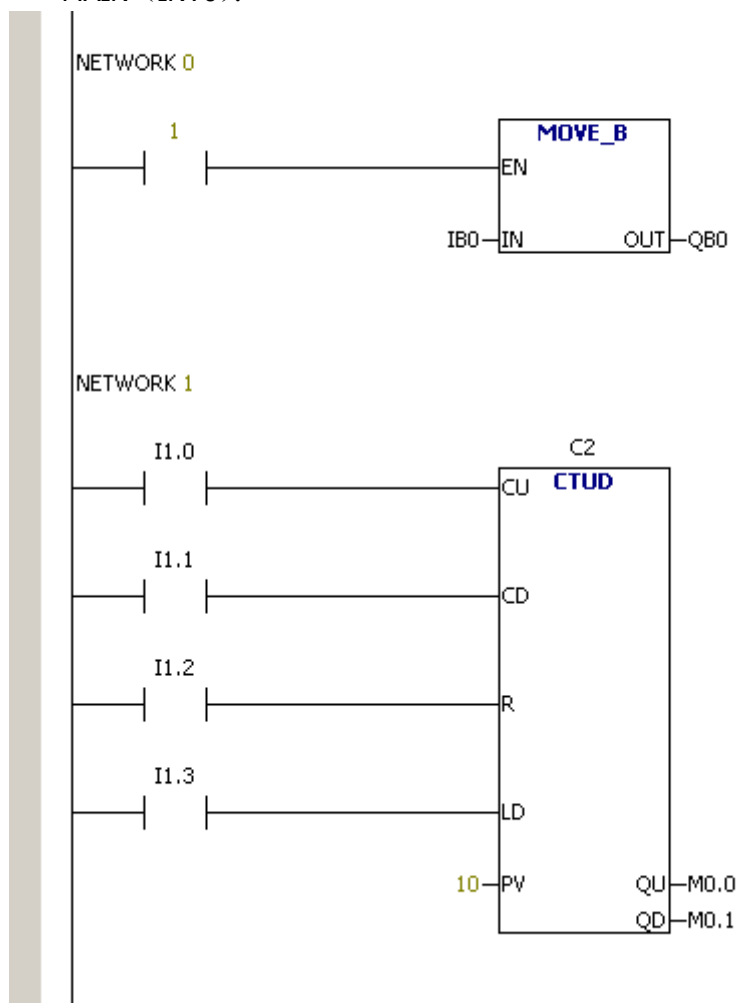
地址	管脚选择	电器特性	逻辑电平
I0.0	P3.3/INT1	仅为输入	负逻辑
I0.1	P3.4/T0/CLKOUT0	仅为输入	负逻辑
I0.2	P0.0/AD0	仅为输入	负逻辑
I0.3	P0.1/AD1	仅为输入	负逻辑
I0.4	P0.2/AD2	仅为输入	负逻辑
I0.5	P0.3/AD3	仅为输入	负逻辑
I1.0	P2.0/A8	仅为输入	负逻辑
I1.1	P2.1/A9	仅为输入	负逻辑
I1.2	P2.2/A10	仅为输入	负逻辑
I1.3	P2.3/A11	仅为输入	负逻辑

Q0.0	P2.4/A12	准双向口	负逻辑
Q0.1	P2.5/A13	准双向口	负逻辑
Q0.2	P2.6/A14	准双向口	负逻辑
Q0.3	P2.7/A15	准双向口	负逻辑
Q0.4	P0.5/AD5	准双向口	负逻辑
Q0.5	P0.4/AD4	准双向口	负逻辑

配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**04-EK51-IO.vcw**。

MAIN (INT0):



首先，通过 MOV B IB0, QB0 指令，将输入 I0.0 ~ I0.7 和 Q0.0 ~ Q0.7 一一对应。下载程序后，分别按下输入 I0.0 ~ I0.7，观察对应的 Q0.0 ~ Q0.7 是否输出。

然后，用 I1.0 ~ I1.3 这 4 个按键控制加减计数器 C2。I1.0 为加计数信号，I1.1 为减计数信号，I1.2 为复位信号，I1.3 为加载信号。下载程序后，进入监控状态，分别按下 I1.0 ~ I1.3 这个四个轻触按键，观察 C2 值的变化。

模数转换 [AD]

简介

IAP12C5A60AD 单片机自带的 A/D 转换口在 P1 口 (P1.0 ~ P1.7)，具有 8 路 10 位高速 A/D 转换器，速度可达到 250KHz (25 万次/秒)。在 GUTTA System Utility 中，可以对每一路 A/D 转换进行配置。

在 EC30-EK51 PLC 中，模拟量输入映射为 IW0、IW2、...、IW14。和数字量输入一样，PLC 服务程序在进入主循环扫描 (MAIN) 之前，都会先读取输入管脚的值。将数字量输入结果写入映像区域 I；将模拟量输入结果写入映像区域 AI；以供主循环扫描程序 MAIN 来使用。

软件配置

在 GUTTA System Utility 中，AD 模块的配置窗口如下：

AD | Analog to Digital Converter

默认值 帮助

地址映射

地址	管脚选择	转换速度
AIW0	P1.0/ADCO/CLK...	90
AIW2		
AIW4		
AIW6		
AIW8		
AIW10		
AIW12		
AIW14		

管脚选择

P1.0/ADCO/CLKOUT2: [AIW0]

转换时间

90 机器周期/每次转换

结果处理

0V 对应数值 (转换结果为0时)

0 (-32768 ... +32767)

5V 对应数值 (转换结果为1023时)

1023 (-32768 ... +32767)

⚠ 系统块设置参数必须下载才能生效

- **地址映射：**可以配置 AIW0、AIW2、...、AIW14 共 16 个模拟量输入。在地址映射控件中选择一个 PLC 模拟量输入，例如图中的输入映射 AIW0，便可以在右边的控件中配置这个模拟量输入的具体特性。
- **管脚选择：**设置地址映射控件中选择的 PLC 模拟量输入对应的单片机管脚。IAP12C5A60AD 的 P1 口（P1.0 ~ P1.7）共 8 个管脚都可以配置为模拟量输入。
- **转换时间：**PLC 服务程序在进入主循环扫描前，会按照 AIW0 到 AIW14 的顺序依次启动 A/D 转换读取管脚的电压。在 IAP12C5A60AD 单片机中，有几种可选的转换时钟，转换时钟决定了转换速度，可供选择的转换速度有 90、180、360、540 机器周期每次转换。
- **结果处理：**模拟量一般对应于一个实际的物理量。这个物理量可能是温度、压力、位置、速度等。在实际的应用中，传感器将这些物理量转换成标准的电流或电压信号，到了单片机这，就必须是 0 ~ 5V 范围内的一个电压。0 ~ 5V 电压通过 A/D 模块读取到寄存器后，又对应于一个 0 ~ 1023 的数字量（10 位数字）。这个数字量代表了外部物理量的大小，但往往单位和零点是不正确的。这个时候可以采用 AD 自带的结果处理运算模块对 0 ~ 1023 进行处理从而标准化为实际的常用单位。

特别说明

由于单片机自带的 A/D 转换模块，采样电压和单片机是共地的，这就给模拟量的隔离带来了困难。噪声比较严重的模拟量信号，会对单片机的运行会产生影响，严重的时候甚至会让单片机程序跑飞。由于目前模拟信号的隔离实现比较困难，要求比较严格的模拟量信号，建议采用专用的 A/D 芯片，然后通过数字接口和单片机通讯交换数据。数字通讯接口的隔离难度要小于模拟量的隔离。EC30-EK51 自带的 SPI 通讯模块完全可以胜任大多数 A/D 转换芯片的数据交换，详情请参考 SPI 模块的说明。

使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道，试验板上有 2 个可调的旋钮电位器，可直接输出 0 ~ 5V 电压。电压信号经过一个 RC 滤波回路后就连接到了单片机的 P1.2、P1.3。

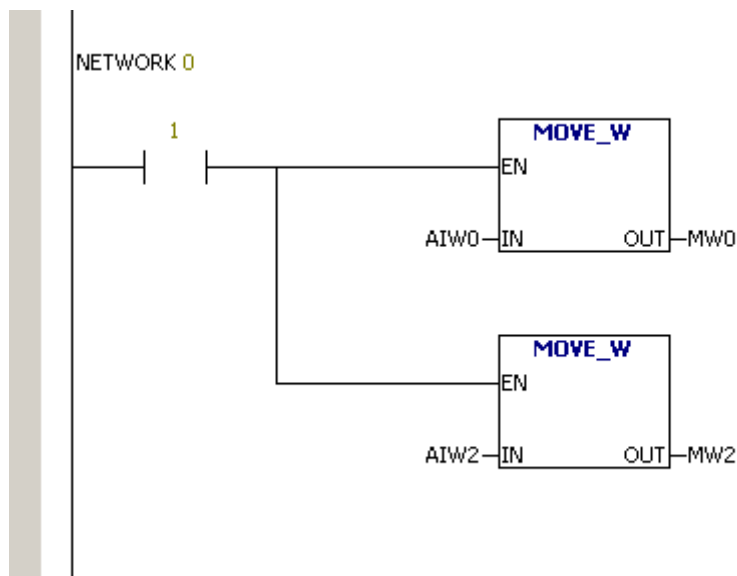
根据模拟量输入的电路图，对应的软件配置如下：

地址	管脚选择	转换速度	0V 对应值	5V 对应值
AIW0	P1.2/AD2/ECI	180	-400	1600
AIW2	P1.3/ADC3/CCP0	180	-400	1600

配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**05-EK51-AD.vcw**。

MAIN (INT0):



下载程序后，进入监控状态，调整 CPU-EC20 (8051)试验板上两个电位器，观察变量 AIW0 和 AIW2 的变化。

I2C 通讯 [I2C]

简介

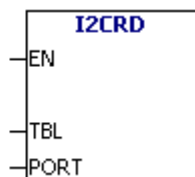
I2C 是一种串行通讯总线，使用单主多从架构，由飞利浦公司在 1980 年代为了让主板、嵌入式系统或手机连接低速周边设备而发展。I2C 的正确读法为“I-squared-C”，而“I-two-C”则是另一种错误但被广泛使用的读法，在国内则多以“I 方 C”称之。截至 2006 年 11 月 1 日为止，使用 I2C 协定不需要为其专利付费，但制造商仍然需要付费以获得 I2C 从属设备地址。

I2C 使用两条通讯线来连接设备（控制线 SCL 和数据线 SDA）。不论是主设备还是从设备，都需要将对应的 SCL 和 SDA 连接到总线上。设备的 SCL 和 SDA 管脚必须是开漏输出。所有设备的 SCL 管脚通过总线，共用一个 10K 左右的上拉电阻；所有设备的 SDA 管脚也通过总线，共用一个 10K 左右的上拉电阻。I2C 的这种连接方式使得所有设备的 SCL（SDA）是“或”的关系，即只要任意一个设备的 SCL（SDA）管脚拉地，总线 SCL（SDA）就被拉地。

IAP12C5A60AD 没有 I2C 硬件支持，故我们只能用普通的 I/O 管脚来模拟。由于 I2C 的通讯时钟是完全由主设备来控制的，可以用查询等待的方式来实现 I2C 主设备。至于 I2C 从设备，在没有硬件支持的情况下，单片机是难以实现的。EC30-EK51 的 I2C 模块只实现了主设备。EC30-EK51 实现了最基本的通讯支持，通讯的具体数据由用户通过 PLC 编程来决定。EC30-EK51 支持两条 I2C 通讯指令：I2CRD 和 I2CWR。

I2CRD

I2CRD 指令用于向指定的 I2C 从设备读取若干数据，I2CRD 指令有两个操作数，TBL 和 PORT。



TBL 操作数接受一个字节数据，标识一个表格的开始。

（假设 Na = 2; Nd = 3）

偏移	数据	类型	说明
0	BYTE	输入	设备地址
1	USINT	输入	子地址长度（Na）
2	USINT	输入	数据长度（Nd）
3	SINT	输出	返回代码
4	BYTE	输入	子地址第 0 个字节
5	BYTE	输入	子地址第 1 个字节
6	BYTE	输出	接收数据的第 0 个字节
7	BYTE	输出	接收数据的第 1 个字节
8	BYTE	输出	接收数据的第 2 个字节

返回代码有以下几种可能的值：

- Nd - 操作成功。
- -1 - 操作数 PORT 错误，不存在的 I2C 模块（PORT 必须为 0 或 1）。
- -2 - I2C 模块未被配置。
- -3 - Nd 数据数量不正确。
- -4 - 发送从设备地址后，从设备无应答。
- -5 - 发送从子地址后，从设备无应答。
- -6 - 接收数据出错。
- -7 - 其他错误。

PORT 指示从哪个 I2C 模块读取数据。

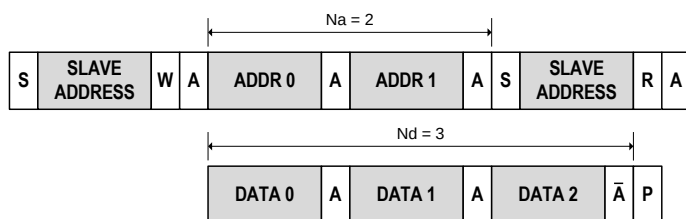
PLC 指令系统采用等待查询的方式执行 I2CRD 指令，只有在全部数据接收完毕或者是出错后，才从 I2CRD 指令返回并继续执行后面的指令。I2CRD 指令按照下面的步骤执行操作：

Na = 0 且 Nd > 0	
1. [START]	发送 I2C 总线通讯开始标志。
2. [SLA+R]	发送 I2C 设备地址，包含读数据标志。
3. [DATA]	接收若干数据（数量由 TBL 中的数据长度 Nd 确定）。
4. [STOP]	发送 I2C 总线通讯结束标志。

Nd = 3									
S	SLAVE ADDRESS	R	A	DATA 0	A	DATA 1	A	DATA 2	Ā P

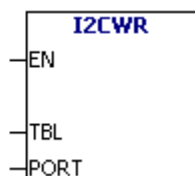
Na > 0 且 Nd > 0	
1. [START]	发送 I2C 总线通讯开始标志。
2. [SLA+W]	发送 I2C 设备地址，包含写数据标志。
3. [DATA]	发送若干数据（数量由 TBL 中的数据长度 Na 确定）。
4. [RESTART]	发送 I2C 总线通讯重新开始标志。

5. [SLA+R] 发送 I2C 设备地址，包含读数据标志。
6. [DATA] 接收若干数据（数量由 TBL 中的数据长度 Nd 确定）。
7. [STOP] 发送 I2C 总线通讯结束标志。



I2CWR

I2CWR 指令用于向指定的 I2C 从设备写入若干数据，I2CWR 指令有两个操作数，TBL 和 PORT。



TBL 操作数接受一个字节数据，标识一个表格的开始：
(假设 Na = 2; Nd = 3)

偏移	数据	类型	说明
0	BYTE	输入	设备地址
1	USINT	输入	子地址长度 (Na)
2	USINT	输入	数据长度 (Nd)
3	SINT	输出	返回代码
4	BYTE	输入	子地址第 0 个字节
5	BYTE	输入	子地址第 1 个字节
6	BYTE	输入	发送数据的第 0 个字节
7	BYTE	输入	发送数据的第 1 个字节
8	BYTE	输入	发送数据的第 2 个字节

返回代码有以下几种可能的值：

- Nd - 操作成功。
- -1 - 操作数 PORT 错误，不存在的 I2C 模块 (PORT 必须为 0 或 1)。
- -2 - I2C 模块未被配置。
- -3 - Na 与 Nd 数据数量不正确。
- -4 - 发送从设备地址后，从设备无应答。
- -5 - 发送从子地址后，从设备无应答。
- -6 - 发送数据出错。
- -7 - 其他错误。

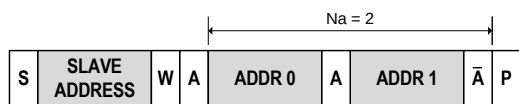
PORT 指示从哪个 I2C 模块写入数据。

PLC 指令系统采用等待查询的方式执行 I2CWR 指令，只有在全部数据发送完毕或者是出错后，才从 I2CWR 指令返回并继续执行后面的指令。I2CWR 指令按照下面的步骤执行 I2C

操作:

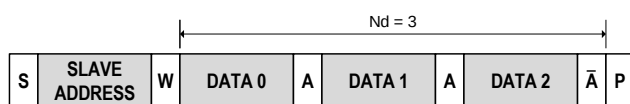
Na > 0 且 Nd = 0

1. [START] 发送 I2C 总线通讯开始标志。
2. [SLA+W] 发送 I2C 设备地址，包含写数据标志。
3. [DATA] 发送若干数据（数量由 TBL 中的数据长度 Na 确定）。
4. [STOP] 发送 I2C 总线通讯结束标志。



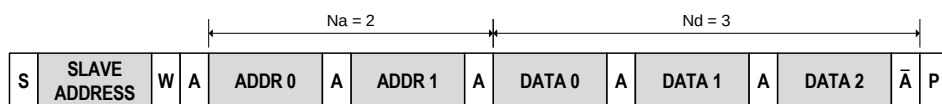
Na = 0 且 Nd > 0

5. [START] 发送 I2C 总线通讯开始标志。
6. [SLA+W] 发送 I2C 设备地址，包含写数据标志。
7. [DATA] 发送若干数据（数量由 TBL 中的数据长度 Nd 确定）。
8. [STOP] 发送 I2C 总线通讯结束标志。



Na > 0 且 Nd > 0

1. [START] 发送 I2C 总线通讯开始标志。
2. [SLA+W] 发送 I2C 设备地址，包含写数据标志。
3. [DATA] 发送若干数据（数量由 TBL 中的数据长度 Na 确定）。
4. [DATA] 发送若干数据（数量由 TBL 中的数据长度 Nd 确定）。
5. [STOP] 发送 I2C 总线通讯结束标志。



软件配置

在 GUTTA System Utility 中，I2C 模块的配置窗口如下：

I2C | Inter-Integrated Circuit

默认值帮助

☐ 配置I2C0模块

SCL控制线
SDA数据线

☒ 配置I2C1模块

SCL控制线
SDA数据线

 系统块设置参数必须下载才能生效

EC30-EK51 中一共有 2 个 I2C 模块，I2C0 和 I2C1。

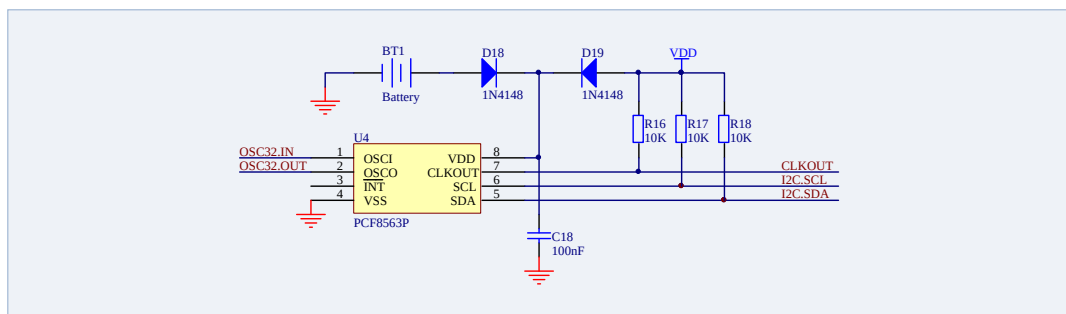
每个 I2C 模块需要配置两个单片机管脚，一个作为控制线 SCL，一个作为数据线 SDA。这两个管脚会在 PLC 上电复位后配置为开漏输出。一旦在 PLC 程序中使用了 I2CRD 或 I2CWR 指令，PLC 指令系统就会驱动 SCL 和 SDA 对应的管脚接收或发送数据。

特别说明

在某些特定的应用中：例如利用 I2C 通讯协议和 EEPROM 芯片 AT24C08B 交换数据；利用 I2C 协议和时钟芯片 PCF8563 交换数据，可以不必使用 I2CRD 或 I2CWR 指令。在 EC30-EK51 中还提供了 PCF8563 配置模块和 AT24C08B 配置模块。这两个模块共享使用 I2C0 或者是 I2C1 模块，管脚也由 I2C0 或者是 I2C1 配置。不同的是用户不必关心其具体的通讯细节，直接使用 TODR、TODW、ERB、EWB 指令即可。详情可参考 PCF8563 模块和 AT24C08B 模块的相关章节以及指令说明。

使用实例

可以在试验板 CPU-EC20 (8051) 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道，板上支持 I2C 通讯的芯片有 PCF8563P 和 AT24C04B，这里我们编写一个实例通过 I2CRD 指令读取 PCF8563P 中的时钟信息。



通过电路图可以知道,PCF8563P 的 SCL 和单片机的 P1.1 相连接,SDA 和单片机的 P1.0 相连接。根据以上信息,对应的软件配置如下:



配置部分完成后,为了测试驱动,写一小段梯形图程序如下:

程序文件名: **06-EK51-I2C.vcw**。

MAIN (INT0):

NETWORK 0

```
LD I0.0
MOVB 16#A2, MB10
MOVB 16#01, MB11
MOVB 16#07, MB12
MOVB 16#00, MB13
MOVB 16#02, MB14
I2CRD MB10, 1
```

这里采用指令表（STL）格式来表达。

程序很简单，在主循环 MAIN 中：

若检测到 I0.0 的上升沿，初始化 MB10 开始的数据表格：

```
MOV B 16#A2, MB10  填入设备地址，PCF8563P 的设备地址就是 16#A2。
MOV B 16#01, MB11  填入子地址长度，使用 1 字节子地址。
MOV B 16#07, MB12  填入数据长度，读取 7 个字节的数据，即 MB15 ~ MB21。
MOV B 16#00, MB13  初始化返回代码，可省略。
MOV B 16#02, MB14  子地址数据，从 PCF8563P 手册中获得，时间寄存器在
PCF8563P 内部的地址偏移为 2。
```

并：

```
I2CRD MB10, 1      执行 I2C 读操作。
```

下载好程序后，连线进入调试状态，按下 I0.0（需要在 IO 模块中配置），观察 MB15 ~ MB21 数据的改变（注意和 TODR、TODW 指令不同，读取的值为未经过处理的 BCD 码，含义请参考 PCF8563P 数据手册）。

SPI 通讯 [SPI]

简介

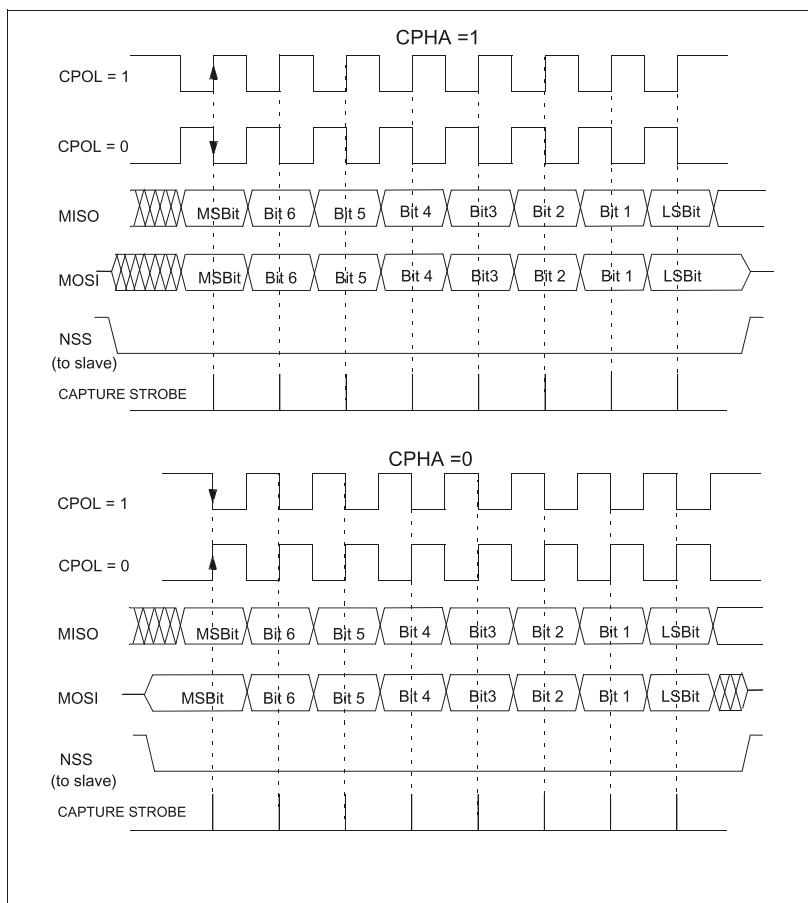
SPI 是英语 Serial Peripheral Interface 的缩写，故名思义就是串行外围设备接口。是 **Motorola** 首先在其 MC68HCXX 系列处理器上定义的。SPI 接口主要应用在 EEPROM、FLASH、实时时钟、A/D 转换等外围器件和单片机的通讯上。SPI 是一种高速的、全双工、同步通讯总线。使用 SPI 通讯总线，总共需要 4 个通讯管脚。

- MOSI - 主设备数据输出，从设备数据输入
- MISO - 主设备数据输入，从设备数据输出
- SCLK - 时钟信号，由主设备产生
- NSS - 从设备使能信号，由主设备控制。

其中 NSS 是控制芯片是否被选中的，只有片选信号 NSS 为低电位时，对此芯片的操作才有效。这就允许在同一总线上连接多个 SPI 设备成为可能。

接下来就负责通讯的 3 根线了。通讯是通过数据交换完成的，这里先要知道 SPI 是串行通讯协议，也就是说数据是一位一位的传输的。这就是 SCLK 时钟线存在的原因，由 SCLK 提供时钟脉冲，MOSI, MISO 则基于此脉冲完成数据传输。数据输出通过 MOSI 线，数据在时钟 SCLK 上升沿或下降沿时改变，在紧接着的下降沿或上升沿被读取。完成一位数据传输，输入也使用同样原理。这样，在至少 8 次时钟信号的改变（上沿和下沿为一次），就可以完成 8 位数据的传输。

要注意的是，SCLK 信号线只由主设备控制，从设备不能控制信号线。同样，在一个基于 SPI 的设备中，至少有一个主控设备。这样的传输方式有一个优点，与普通的串行通讯不同，普通的串行通讯一次连续传送至少 8 位数据，而 SPI 允许数据一位一位的传送，甚至允许暂停，因为 SCLK 时钟线由主控设备控制，当没有时钟跳变时，从设备不采集或传送数据。也就是说，主设备通过对 SCLK 时钟线的控制可以完成对通讯的控制。SPI 还是一个数据交换协议：因为 SPI 的数据输入和输出线独立，所以允许同时完成数据的输入和输出。不同的 SPI 设备的实现方式不尽相同，主要是数据改变和采集的时间不同，在时钟信号上沿或下沿采集有不同定义：

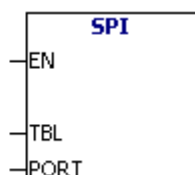


如上图所示，SPI 通讯根据数据采样的时机分为两种模式。如果数据在 SCLK 的前沿发送，在 SCLK 的后沿采样，则为 CPHA=1 模式（图中的上半部分）。如果数据在 SCLK 上一周期的后沿发送，在 SCLK 的前沿采样，则为 CPHA=0 模式（图中的下半部分，这种模式下，首个数据位必须在 NSS 的下降沿给出，虽然此时 SCLK 还没有开始）。

IAP12C5A60AD 具有一个 SPI 硬件。即可作为 SPI 通讯主设备，也可以作为 SPI 通讯从设备。考虑到单片机和外围器件通讯一般为主设备，EC30-EK51 的 SPI 模块只实现了主设备。

SPI

SPI 指令用于在指定的 SPI 总线上与从设备交换 N 个字节的数据。SPI 指令有 2 个操作数，TBL 和 PORT。



TBL 操作数接受一个字节数据，标识一个表格的开始：

偏移	类型	说明
0	字节	需要交换的数据长度 (N)
1	字节	返回代码

2	字节	发送（接收）缓冲区的第 0 个字节
3	字节	发送（接收）缓冲区的第 1 个字节
4	字节	发送（接收）缓冲区的第 2 个字节
5	字节	发送（接收）缓冲区的第 3 个字节
...		...
N+2	字节	发送（接收）缓冲区的第 N 个字节

返回代码有以下几种可能的值：

- 0 - 操作成功。
- -1 - 操作 SPI 错误，不存在的 SPI 模块。
- -2 - SPI 模块未被配置。
- -3 - SPI 从设备无响应。

PORT 指示操作哪个 SPI 模块。

外部电路

IAP12C5A60AD 的 SPI 使用 P1 口：

- NSS - P1.4/ADC4/CCP1/SS
- MOSI - P1.5/ADC5/MOSI
- MISO - P1.6/ADC6/MISO
- SCLK- P1.7/ADC7/SCLK

也可以重映射到 P4 口：

- NSS - P4.0/SS
- MOSI - P4.1/ECI/MOSI
- MISO - P4.2/CCP0/MISO
- SCLK- P4.3/CCP1/SCLK

由于主设备可以采用软件 NSS 控制，故 NSS 管脚可以是 P1.4、P4.0 以外的管脚。

软件配置

在 GUTTA System Utility 中，SPI 模块的配置窗口如下：

SPI | Serial Peripheral Interface Bus

默认值 帮助

☒ 配置SPI模块

时钟速率

CPU_CLK/128 =86.4kHz (晶振为11.0592MHz时)

时钟相位

☐ 数据在时钟后沿采样 (CPHA=1)
☒ 数据在时钟前沿采样 (CPHA=0)

时钟极性

☐ CLK空闲时为高电平 (CPOL=1)
☒ CLK空闲时为低电平 (CPOL=0)

数据顺序

☐ LSB最先发送 (DORD=1)
☒ MSB最先发送 (DORD=0)

模块选择

使用SPI模块 ☒ SPI: [SPI]

管脚选择

☐ 将SPI从P1口切换到P4口

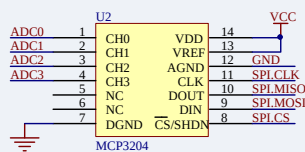
MOSI ☐ P1.5/ADC5/MOSI: [SPI.MOSI]
MISO ☐ P1.6/ADC6/MISO: [SPI.MISO]
SCLK ☐ P1.7/ADC7/SCLK: [SPI.SCLK]

系统块设置参数必须下载才能生效

- **配置 SPI 模块：** 选择表示使能 SPI 模块，不选择表示禁用 SPI 模块。
- **时钟速率：** 其分频系数决定了 SCLK 的频率，其值不能超过 SPI 从设备允许的最高频率。
- **时钟相位：** 确定数据在时钟的前沿还是后沿采样 (CPHA=0 或 CPHA=1)。
- **时钟极性：** 确定 SCLK 空闲时为底电平还是为高电平 (CPOL=0 或 CPOL=1)。
- **数据顺序：** 确定低位 (LSB) 最先发送还是 (MSB) 最先发送。
- **模块选择：** 这里必须选择硬件 SPI。
- **管脚选择：** 可以重映射 SPI 管脚，以方便硬件电路的设计。需要注意的是，SPI 模块不自动操作 NSS 通讯脚，需要用户手动操作！

使用实例

可以在实验板 EC30-EK51-EVAL 上测试本模块。通过电路图我们知道，在 EC30-EK51-EVAL 试验板上，有一个支持 SPI 通讯协议的 A/D 转换芯片 MCP3204。



根据 MCP3204 的硬件手册，做出下面的软件配置：

SPI | Serial Peripheral Interface Bus

默认值
帮助

☒ 配置SPI模块

时钟速率

CPU_CLK/128

=86.4kHz (晶振为11.0592MHz时)

时钟相位

☐ 数据在CLK的当前时钟前沿驱动,当前时钟后沿采样 (CPHA=1)

☒ 数据在CLK的前一时钟后沿驱动,当前时钟前沿采样 (CPHA=0)

时钟极性

☐ CLK空闲时为高电平,CLK的时钟前沿为下降沿,时钟后沿为上升沿 (CPOL=1)

☒ CLK空闲时为低电平,CLK的时钟前沿为上升沿,时钟后沿为下降沿 (CPOL=0)

数据顺序

☐ 数据字的LSB (最低位)最先发送 (DORD=1)

☒ 数据字的MSB (最高位)最先发送 (DORD=0)

管脚选择

☐ 将SPI从P1口切换到P4口

MOSI

P1.5/ADC5/MOSI: [SPI. MOSI]

MISO

P1.6/ADC6/MISO: [SPI. MISO]

SCLK

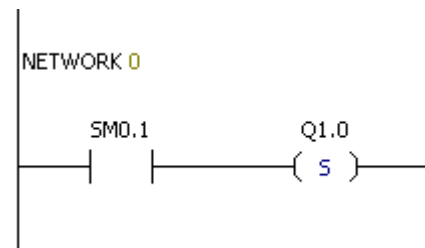
P1.7/ADC7/SCLK: [SPI. SCLK]

系统块设置参数必须下载才能生效

配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**07-EK51-SPI (EVAL).vcw**。

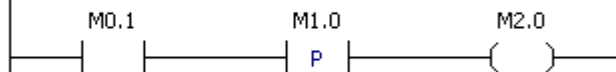
MAIN (INT0):



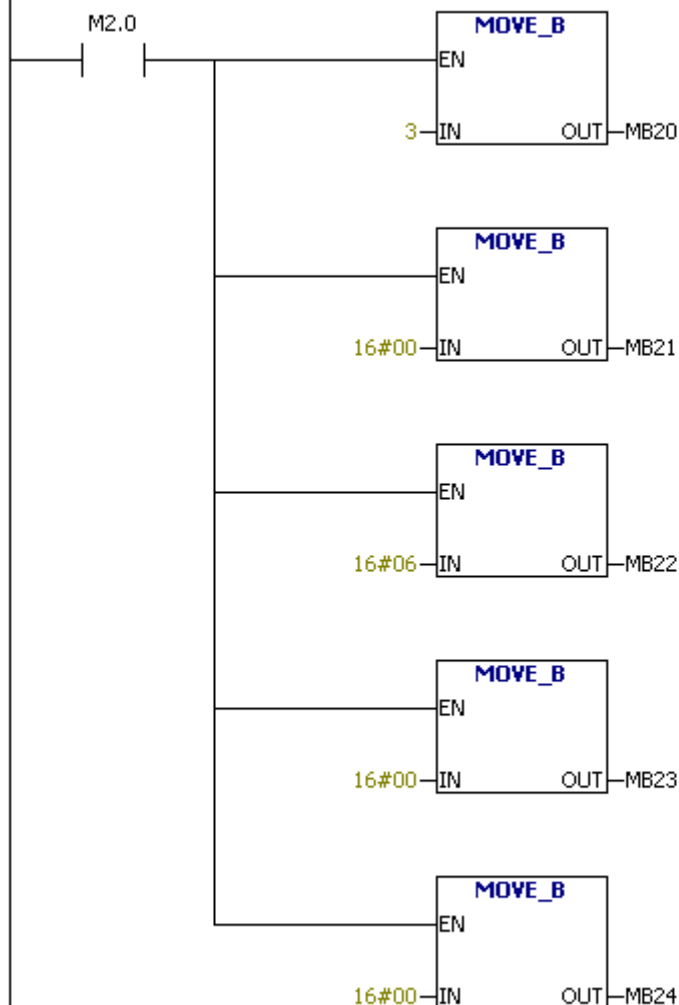
NETWORK 1

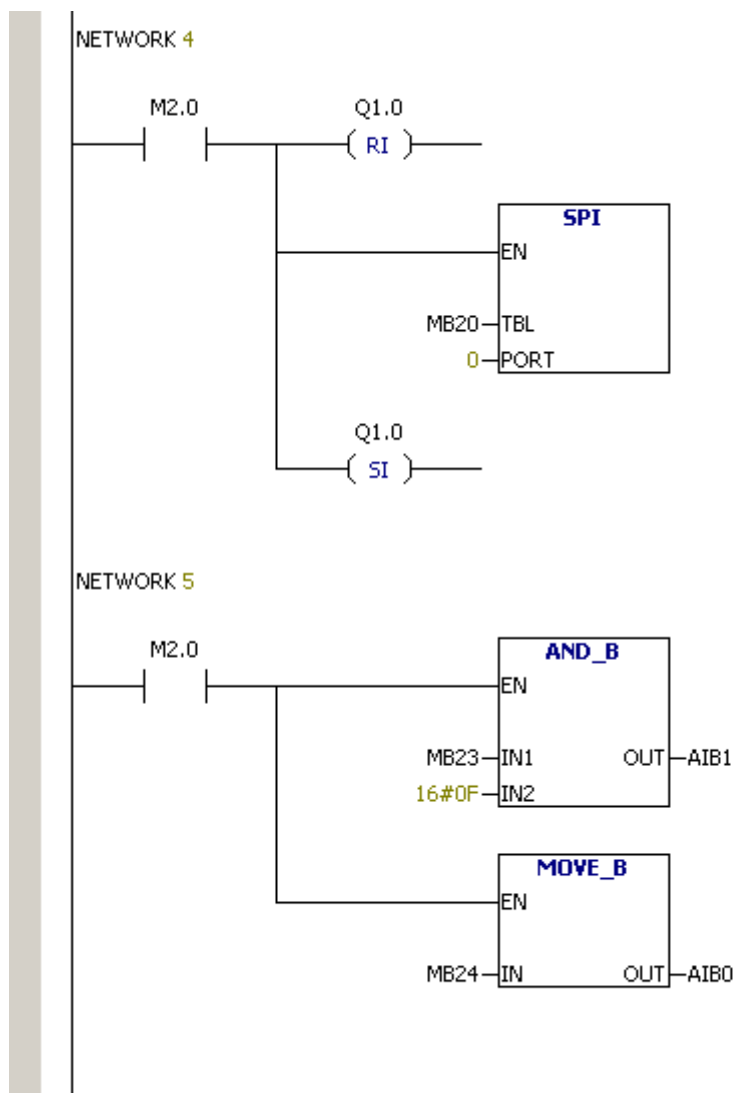


NETWORK 2



NETWORK 3

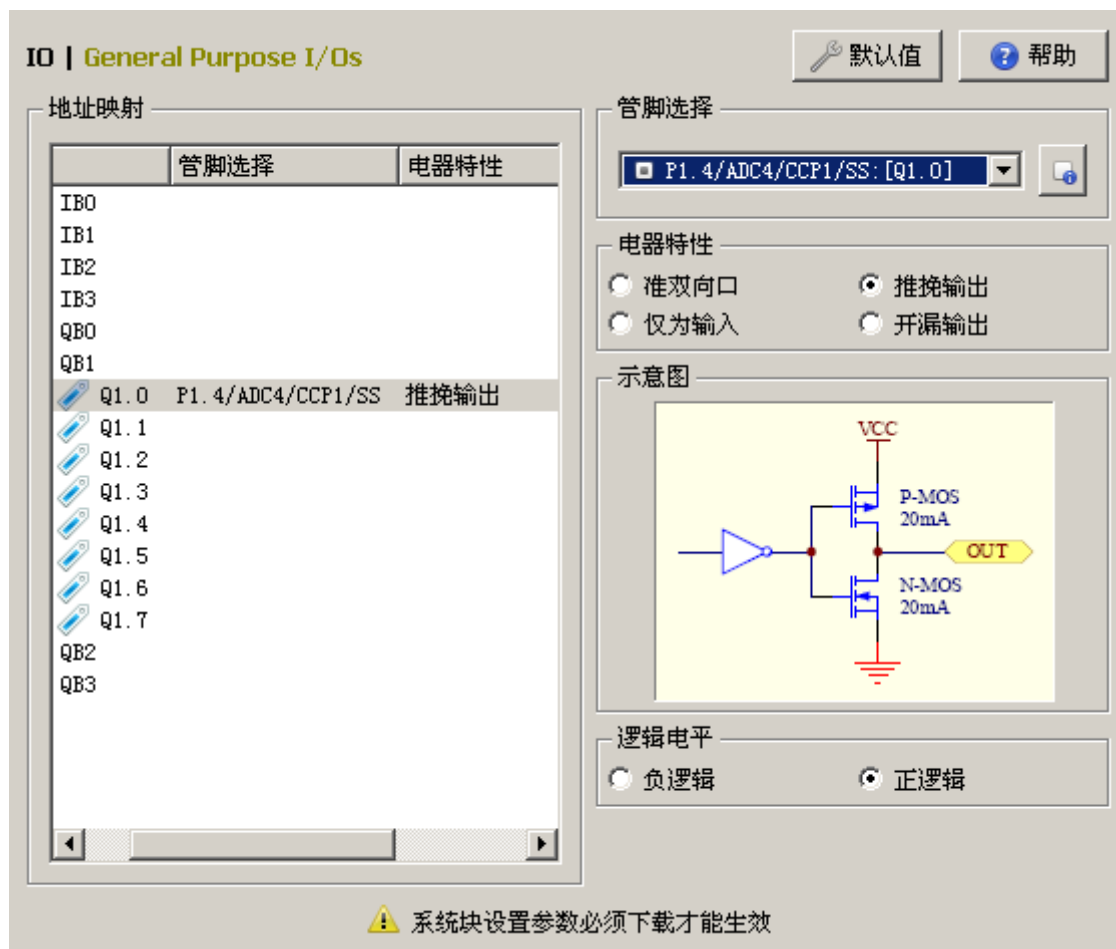




在监控状态下设置 M0.0 为 1，则 T2 开始工作且每过 500mS 驱动 M2.0 发出一个正脉冲。若 M2.0 为 1，则执行下面操作：

1. 初始化 MB20 开始的数据表格：
MB20 = 3、MB21 = 0、MB22 = 6、MB23 = 0、MB24 = 0。
2. 设置 Q1.0 为 0，即 NSS = 0，开始 SPI 通讯。
3. 执行 SPI 通讯，与 MCP3204 交换若干数据。
4. 设置 Q1.0 为 1，即 NSS = 1，结束 SPI 通讯。
5. 对结果 MB23、MB24 进行数据处理，并将模拟量存储到 AIW0 中。

由于 SPI 模块不自动处理 NSS 管脚，这里使用立即复位（RI）和立即置位（SI）指令来手动控制 NSS 管脚。这就需要在 IO 模块中定义 Q1.0：



将程序下载到 EC30-EK51-EVAL 试验板中，进入监控模式，置 M0.0 为 1，旋转试验板上的电位器 R1，便可观察到 AIW0 值的变化。

1-Wire 通讯 [1W]

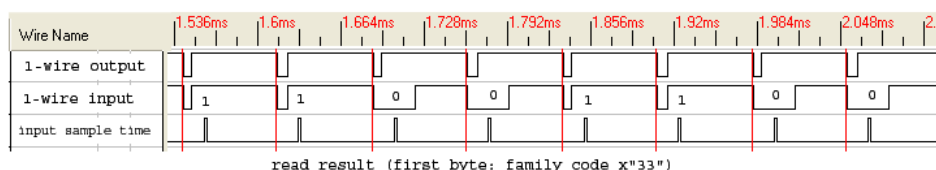
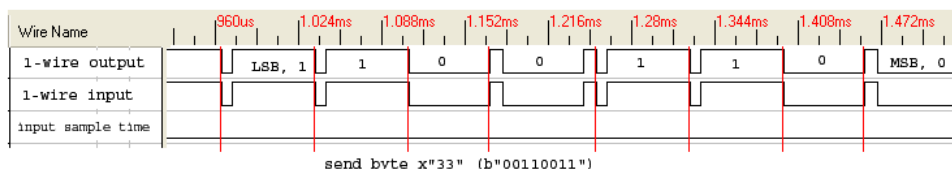
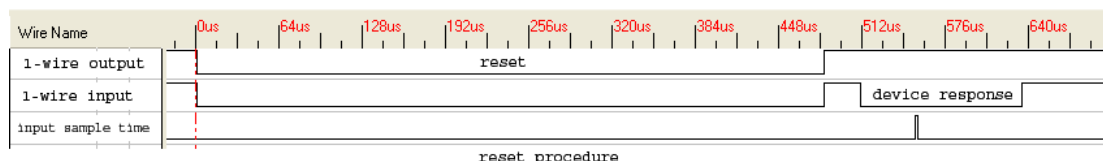
简介

1-Wire 是 **MAXIM** 子公司 **DALLAS** 半导体的专利技术，采用单一信号线，却可以像 I2C、SPI 一样，同时传输时钟（CLK）又传输数据（DAT），而且数据传输是双向的。1-Wire 使用较低的数据传输速率，通常是在板上用来和小型器件进行通讯，例如温度计。

在具体的器件连接上，唯一的信号线 DQ 即被用于传输时钟，又被用于传输数据。1-Wire 从器件必须和 1-Wire 主器件共用同一个供电线 VDD 和同一个地线 VSS。由于总线空闲时 DQ 是上拉到 VDD 的，对于功率很小的 1-Wire 器件，可以省去 VDD 的连接，器件在通讯空闲时从 DQ 上“窃电”即可保证正常工作。

DQ 工作的具体情况可以参考下图，分别给出了复位、写字节、读字节的时序。由于 1-Wire 协议对从器件的响应速度要求很严格，EC30-EK51 的 1W 模块只实现了 1-Wire 主器件在总线上的复位、写字节、读字的基本操作。

1 Wire reset, write and read example with DS2432



1-Wire 总线支持一主多从的总线结构。每个 1-Wire 从器件都有一个唯一的 64bit 序列号。在一主多从的总线连接中，1-Wire 主器件发起通讯后（执行复位操作），可以发送 55H 指令，表示接下来会主动给出需要与之通讯器件的序列号，发送序列号之后，与之匹配的从器件响应后面的通讯，否则不响应，直到接收到下个复位操作。

可以判断，如果使用 55H 指令，主器件必须先知道与之相连从器件的序列号，这个序列号是出厂时确定的，我们如何获得这个序列号呢？

一种办法就是确保 1-Wire 总线上只有一个从器件，让主器件发送 33H 指令。从器件在接收到 33H 指令后，会无条件的发送自己的序列号给主器件。由于 1-Wire 总线上只有一个从器件，因此不会存在通讯冲突。

另一种办法就是使用 1-Wire 的器件查找命令 F0H。器件接收到 F0H 指令后，会先发送序列号 64bit 的第 1 个 bit，然后发送第 1 个 bit 的反码，紧接着从主器件接收一个 bit。如果这个 bit 和开始发送的 bit 一致，表示被选择。紧接着开始发送第 2 个 bit，然后是第 2 个 bit 的反码，依此类推，直到序列号的第 64 个 bit。64 个 bit 中只要有一个接收到的 bit 不符合自己的序列号，就停止通讯，回到通讯复位操作前的状态。

这种办法充分利用了 DQ 总线上拉，开漏驱动的特性。也就是说只要有一个器件的 DQ 为输出，DQ 就必然为低电平。这种或的关系配合从器件每个位的两次发送（一次是位，一次是位的反码），使得主器件能够根据 DQ 的两个值，做出下面的判断：

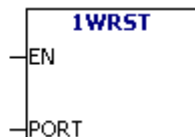
- 总线上不存在器件。（两次读都为高电平）
- 总线上存在此位为 0 的器件。（第 1 次读为低电平，第 2 次读为高电平）
- 总线上存在此位为 1 的器件。（第 1 次读为高电平，第 2 次读为低电平）
- 总线上存在此位为 0 和 1 的器件。（两次读都为低电平）

那么，接下来主器件发送的位相当于根据上述信息完成的一道选择题：“我要选择此位为何值的器件呢？”。主器件根据每次的选择条件，完成这 64 道选择题后，就最终确定了与之通讯的从器件。

EC30-EK51 的 1W 模块目前无法完成器件的查找，故 1W 总线上只能有一个从器件，除非事先知道每个从器件的序列号。

1WRST

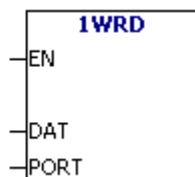
1WRST 指令用于向指定的 1-Wire 总线上发送通讯复位信号。1WRST 指令有 1 个操作数，PORT。



- PORT 用于指定 1-Wire 总线。
- 指令执行结束后，会更新 SMB1 的值：
- -1: 不存在的 1W 模块。
 - -2: 1W 模块未被配置。
 - 0: 执行操作后，未接收到从器件的应答。
 - 1: 执行操作后，已接收到从器件的应答。

1WRD

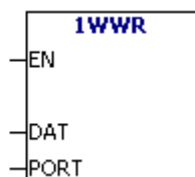
1WRD 指令用于从指定的 1-Wire 总线上接收 1 个字节的数据。1WRD 指令有 2 个操作数，DAT 和 PORT。



- DAT 用于存放接收的字节。
 - PORT 用于指定 1-Wire 总线。
- 指令执行结束后，会更新 SMB1 的值：
- -1: 不存在的 1W 模块。
 - -2: 1W 模块未被配置。
 - 0: 正确执行。

1WWR

1WWR 指令用于向指定的 1-Wire 总线上发送 1 个字节的数据。1WWR 指令有 2 个操作数，DAT 和 PORT。



- DAT 用于存放发送的字节，可以是常数。
- PORT 用于指定 1-Wire 总线。

指令执行结束后，会更新 SMB1 的值：

- -1: 不存在的 1W 模块。
- -2: 1W 模块未被配置。
- 0: 正确执行。

外部电路

由于 1-Wire 通讯只需要 1 根线即可，外部电路的连接十分简单。IAP12C5A60AD 本身不带硬件 1-Wire 通讯模块，EC30-EC51 的 1W 模块是采用 I/O 管脚模拟的方式来实现主器件通讯协议的，故 IAP12C5A60AD 的任何通用 I/O 管脚都可用做 DQ。

软件配置

在 GUTTA System Utility 中，1W 模块的配置窗口如下：



EC30-EK51 可以配置两个 1W 模块，1W0 和 1W1。

每个 1W 模块需要配置一个单片机管脚作为 1-Wire 总线的 DQ。指定的管脚会在 PLC 上电复位后自动为准双向口（弱上拉）。一旦在 PLC 程序中使用 1WRST、1WRD、1WWR 指令，PLC 指令系统就会驱动 DQ 管脚执行通讯操作。

使用实例

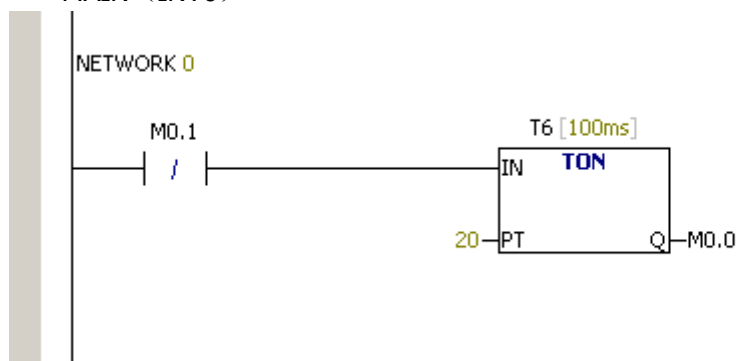
可以在实验板 EC30-EK51-EVAL 上测试本模块。通过电路图我们知道，在 EC30-EK51-EVAL 试验板上，有一个支持 1-Wire 通讯协议的温度测量芯片 DS18B20。DS18B20 和单片机共用电源和地，并且总线 DQ 和单片机的 P3.2/INT0 相连接。根据以上信息，对应的软件配置如下：

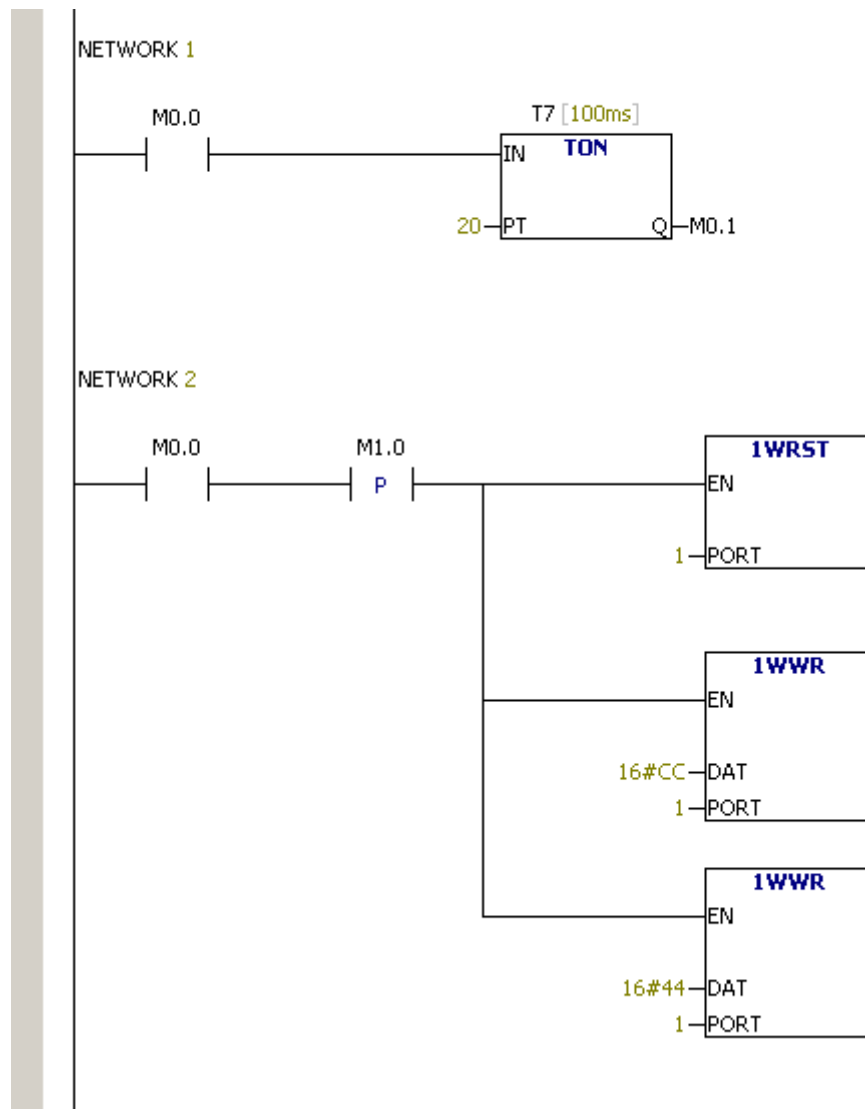


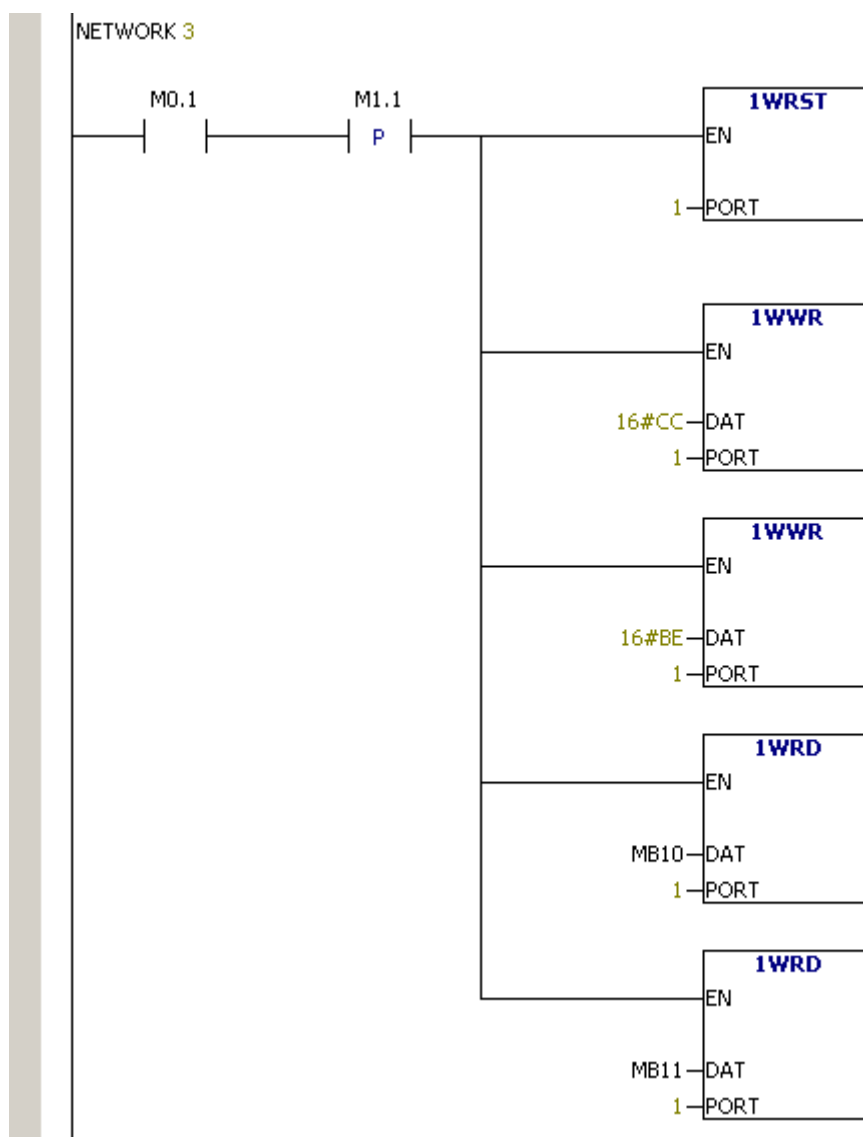
配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**08-EK51-1W (EVAL).vcw**。

MAIN (INT0)



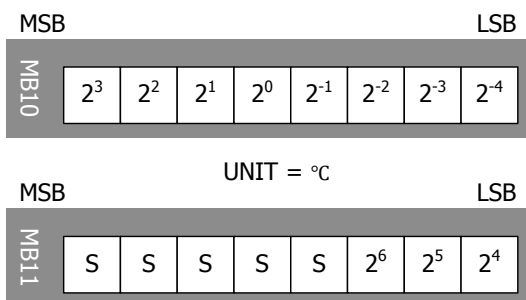




程序很简单，在主循环中，通过级连的两个定时器 T6 和 T7，得到 M0.0 和 M0.1 的两个脉冲信号。PLC 上电复位后 T6 工作，M0.0 在 2 秒后置位导致 T7 工作，第 4 秒时 M0.1 置位然后马上复位 M0.0 和自己，依此循环。

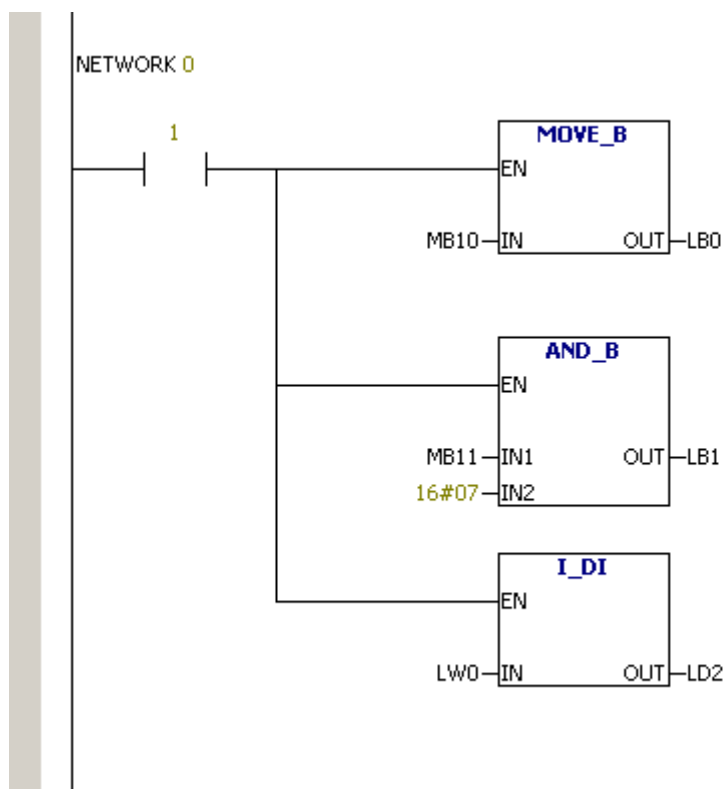
程序在监测到 M0.0 的上升沿转换时，发送 16#CC、16#44 两个字节的数据，启动 DS18B20 的温度转换。2 秒后，检测到 M0.1 的上升沿转换，发送 16#CC、16#BE 两个字节的数据，通知器件，将会读取两个字节的温度值。这里读取的两个字节的温度值不能直接使用，具体格式和转换方法可以参考 DS18B20 的数据手册。

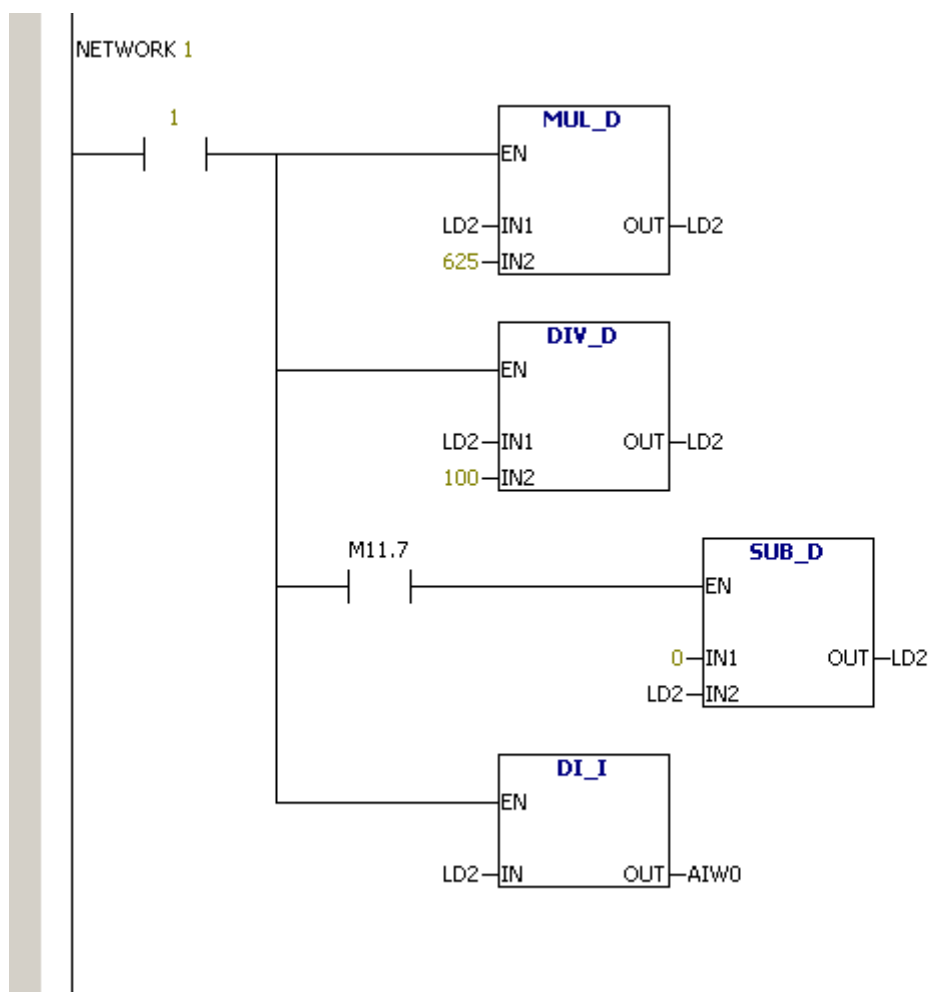
在线监控时，可以尝试着改变 DS18B20 的温度（例如用手捏住温度 IC），观察 MB10 和 MB11 的变化。如果需要得到实际可以使用的温度值，需要对 MB10 和 MB11 进行处理。根据 DS18B20 的数据手册我们知道，MB10 和 MB11 这两个字节的实际含义是：



通过上面的图可以知道，MB10 的 8 位、MB11 低 3 位组成的 11 位数给出了当前温度。MB11 的高 5 位给出了温度的符号。温度的单位为℃，分辨率为 2^{-4}°C ，即 0.0625°C 。为了方便使用，我们希望使用 10 进制的℃单位，例如 0.01°C 单位。转换很简单，用这 11 位数乘上 0.0625 便得到了实际的温度值，然后除以 0.01 便得到了 0.01°C 单位的数值。由于 EC30-EK51 不支持浮点运算，为了减小精度损失，我们这里先用这 11 位数乘上 625，然后除以 100。具体的转换程序如下：

CONVERT (SBR0)





程序根据 M11.7 决定 LD2 的符号,最后将 LD2 转换成 16 位整数存放到 AIW0 中。AIW0 的单位是 0.01℃。例如我在测试时 AIW0 为 2906, 表示当前的开发板温度为 29.06℃。

脉冲密度 [SPD]

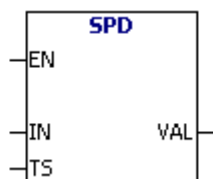
简介

脉冲密度 SPD 模块用于统计指定时间周期内, 脉冲发生的个数(频率计)。脉冲密度 SPD 模块必须配合 SPD 指令来使用。在 GUTTA System Utility 中配置好 SPD 模块后, SPD 模块自动配置并占用 IAP12C5A60AD 的硬件定时器/计数器 0, 并共享使用定时器/计数器 0 的外部计数管脚 P3.4/T0/CLKOUT0。因此需要将这个管脚在 IO 模块中配置为数字量输入。在使用 SPD 指令时, 这个数字量输入必须给出以便指令确认是哪一路高速计数(虽然目前 EC30-EK51 只能配置一路硬件高速计数, 但是考虑到以后的扩展和兼容, 这个数字量输入目前必须正确给出, SDP 指令才真正启动 SPD 模块计数)。

SPD

SPD 模块通过 SPD 指令来驱动, 当 SPD 指令的使能端 EN 为 1 时, 模块统计每个 TS

周期中输入信号 IN 发生的次数，并将结果 VAR 存入 SPD 指令的参数中。



IN 为计数对象，如果在 IO 模块中将 SPD 的计数管脚 P3.4/T0/CLKOUT0 配置为 IO.1，这里计数对象就必须为 IO.1。

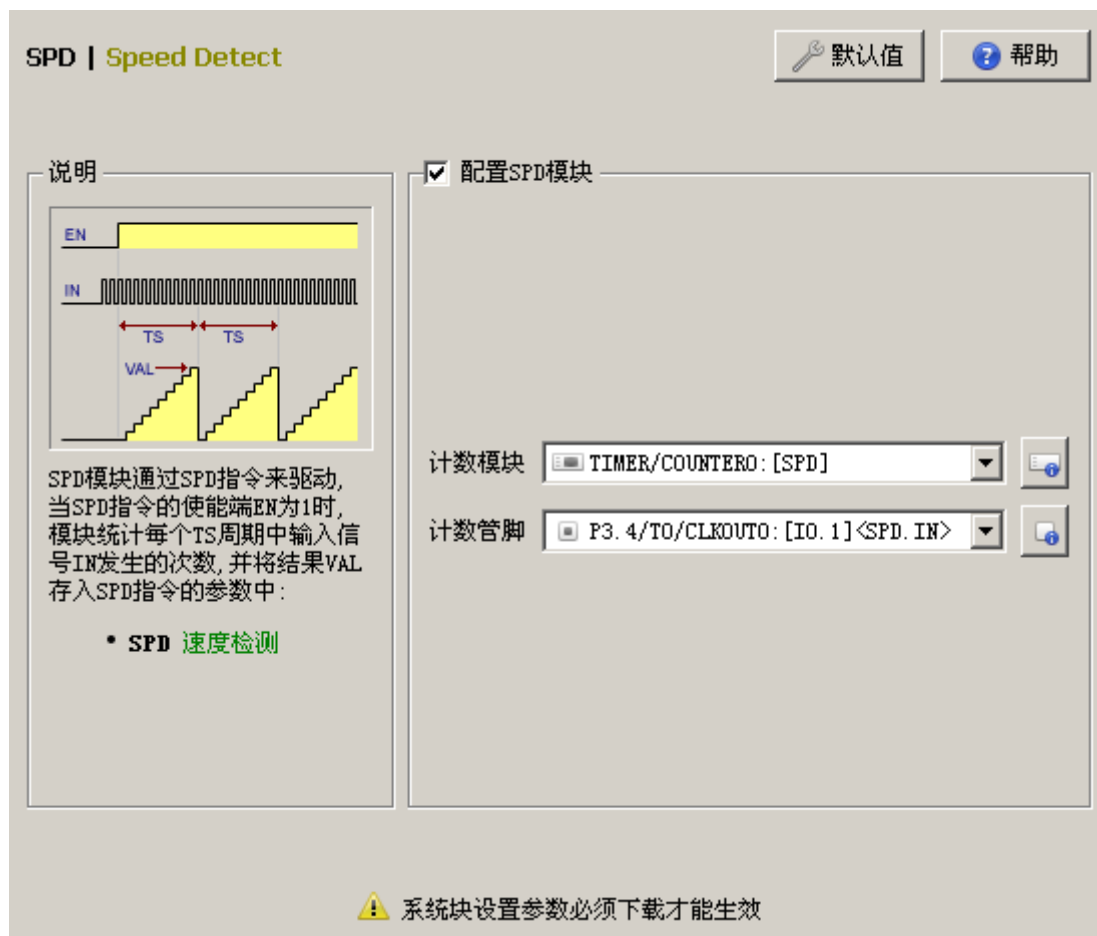
TS 为采样周期，单位为 ms。

VAL 为指令输出的第一个字。若 VAL 为 MW20，指令输出就为 MW20 开始的连续三个字，即 MW20、MW22、MW24：

- MW20 为统计结果。指令在每个 TS 周期结束时，将计数器的当前值存入 MW20。
- MW22 为计数器的当前值。
- MW24 为当前 TS 周期剩下的时间，单位为 ms。

软件配置

在 GUTTA System Utility 中，SPD 模块的配置窗口如下：



- **配置 SPD 模块：**选择是否使能脉冲密度 SDP 模块，如果使能，SDP 模块将配置单片机的计数模块 TIMER/COUNTER0 并共享使用计数管脚 P3.4/T0/CLKOUT0。

特别说明

用单片机实现高速计数一般有两种方式，一种是使用单片机的硬件定时器/计数器，另一种方式使用中断系统。使用硬件定时器/计数器这种方式时，每次的脉冲发生单片机系统不产生中断，即不消耗单片机 CPU 的时间，故可以做到很高的速度。频率计 SDP 模块的计数范围，很大程度上取决于外部电路的速度（例如高速光藕的性能）。不过由于 51 单片机的定时器/计数器是 16 位的，每到 65535 后会发生溢出，如果计数对象的频率很高，应该适当缩短采样时间。例如 100KHz 左右的脉冲信号，如果采样时间为 1000ms，那么脉冲个数将是 100K 即 10 万个，这个数值已经大于 65535，计数器在一个采样周期内就会发生溢出。如果采样时间修改为 100ms，那么脉冲的个数将是 10K 即 1 万个，这个数值小于 65535，SDP 模块能够得到正确的结果。

全硬件方式的频率计 SPD 模块有其局限性。首先，单片机的硬件定时器/计数器数量是有限的（IAP12C5A60AD 只有两个标准定时器/计数器），其中 TIMER/COUNTER1 被用于 PLC 系统时钟，能够供 SPD 使用的就只有 TIMER/COUNTER0 了。其次，计数管脚也被限定为 P3.4/T0/CLKOUT0，在需要高速计数的场合，在电路图绘制阶段必须规划好高速计数管脚。

使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道，单片机的 P3.4/T0/CLKOUT0 和一个自锁按键相连，由于没有高速信号发生器，这里我们只简单的测试一段时间内按键被按下的次数，以此来演示如何使用脉冲密度 SPD 模块。

SPD | Speed Detect

默认值
帮助

说明

SPD模块通过SPD指令来驱动,当SPD指令的使能端EN为1时,模块统计每个TS周期中输入信号IN发生的次数,并将结果VAL存入SPD指令的参数中:

- **SPD 速度检测**

☒ 配置SPD模块

计数模块 TIMER/COUNTER0: [SPD]

计数管脚 P3.4/T0/CLKOUT0: [I0.1] <SPD.IN>

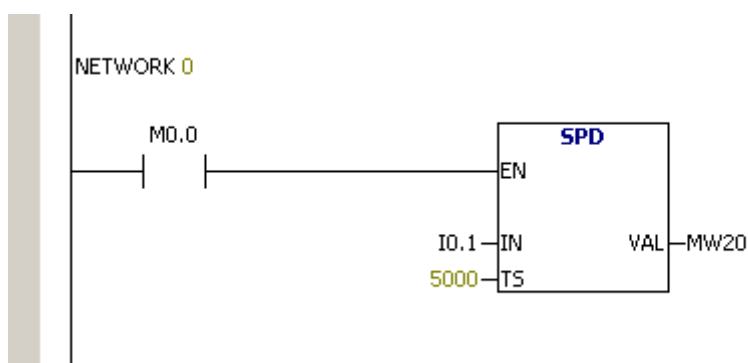
系统块设置参数必须下载才能生效

除了配置软件的 SPD 模块,还需要配置 P3.4/T0/CLKOUT0 管脚为数字量输入 I0.1, 配置办法请参考 IO 模块的相关章节。

配置部分完成后,为了测试驱动,写一小段梯形图程序如下:

程序文件名: **09-EK51-SPD.vcw**。

MAIN (INT0):



程序很简单,下载后进入监控状态,在状态表中设置 M0.0 为 1,运行 SPD 指令。观察 MW20、MW22、MW24 三个变量值的变化。同时反复按下试验板上的 I0.1 按键,继续观察这三个变量值的变化。

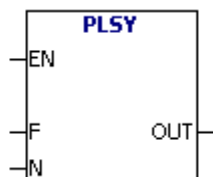
脉冲输出 [PTO]

简介

脉冲输出 PTO 模块根据用户的需要，以指定的频率将一定数量的脉冲通过特定的管脚发送出去。脉冲输出 PTO 模块必须配合 PLSY 或 PLSR 指令来使用。在 GUTTA System Utility 中配置好 PTO 模块后，PTO 模块自动占用 IAP12C5A60AD 的 PCA/PWM 硬件，并共享使用 PCA/PWM 的输出管脚 P1.4/ADC4/CCP1/SS（或重映射为 P4.3/CCP1/SCLK）。因此需要将这个管脚在 IO 模块中配置为数字量输出。在使用 PLSY 或 PLSR 指令时，这个数字量输出必须给出以便指令确认是哪一路高速输出（虽然目前 EC30-EK51 只能配置一路硬件高速输出，但是考虑到以后的扩展和兼容，这个数字量输出目前必须正确给出，PLSY 或 PLSR 指令才真正启动 PTO 模块输出）。

PLSY

PTO 模块可以通过 PLSY 指令驱动，当 PLSY 指令的使能端 EN 由 0 变为 1 时，PLSY 指令初始化 PTO 硬件并将数量为 N 的脉冲以 F 频率输出到输出管脚 OUT 上。



F 为输出脉冲的频率，单位为 Hz。

N 为输出脉冲的数量。

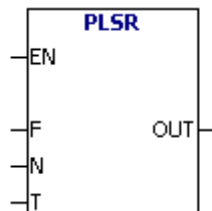
OUT 为输出脉冲的管脚。

- 频率 F 的范围是 10 ~ 40K (Hz)。频率 F 可以在脉冲发送过程中改变，正在发送的脉冲频率也随之改变。
- 脉冲数量 N 的范围是 0 ~ 2147483647，若 N 为 0，表示忽略脉冲数量，PLSY 模块只要 EN 为 1，始终发送脉冲。N 在 PLSY 指令的使能端 EN 由 0 变为 1 时读取，发送过程中改变 N 的值不起作用，N 改变后会在下一次脉冲发送生效。
- 在脉冲发送过程中如果 PLSY 指令的使能端 EN 变为 0，脉冲发送停止。EN 由 0 变为 1，PLSY 指令重新初始化 PTO 硬件开始发送脉冲，之前被中断的脉冲发送被忽略。
- 脉冲发送的占空比为 50%ON、50%OFF，脉冲的发送完全由 PCA/PWM 的硬件中断处理，不受扫描周期的影响。
- 若 N 不为 0，当前脉冲发送的数量可以通过 SMD4 读取，数量为 N 的脉冲发送完毕后，特殊寄存器 SM3.5 被置 1。

PLSR

PTO 模块可以通过 PLSR 指令来驱动，当 PLSY 指令的使能端 EN 由 0 变为 1 时，PLSR

指令初始化 PTO 硬件并将数量为 N 的脉冲以 F 频率输出到输出管脚 OUT 上。和 $PLSY$ 指令不同，考虑到实际脉冲驱动设备的物理惯性（例如有负载的步进电机）， $PLSR$ 指令输出的脉冲会由一个较低的频率（ $F/16$ ）在指定时间 T 内，平稳加速到 F 。同样的，当脉冲输出将要结束时， $PLSR$ 指令对输出的脉冲频率进行平稳的减速，到一个较低的频率（ $F/16$ ）时，刚好发送完所有的脉冲。

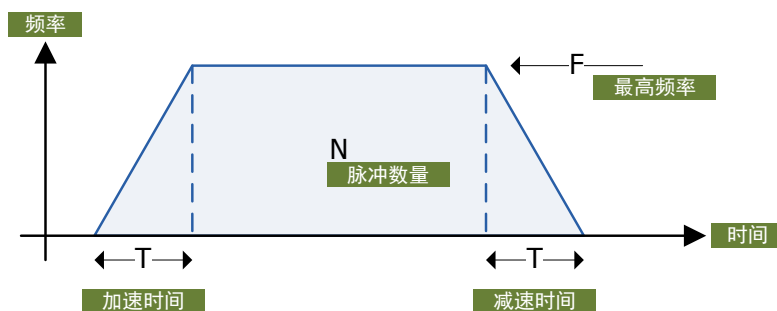


F 为输出脉冲的最高频率，单位为 Hz。

N 为输出脉冲的数量。

T 为加速和减速的时间，单位为 ms。

OUT 为输出脉冲的管脚。



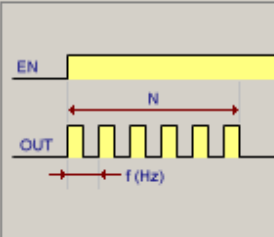
- 频率 F 的范围是 $10 \sim 40K$ (Hz)。和 $PLSY$ 指令不一样，所有的参数在 $PLSR$ 指令的使能端 EN 由 0 变为 1 时读取，脉冲输出过程中改变参数对当前输出不起作用，在下一次脉冲发送生效。
- 脉冲数量 N 的范围是 $0 \sim 2147483647$ ，若 N 为 0，表示忽略脉冲数量， $PLSR$ 模块只要 EN 为 1，始终发送脉冲（加速时间依然有效）。
- 在脉冲发送过程中如果 $PLSR$ 指令的使能端 EN 变为 0，脉冲发送停止。 EN 由 0 变为 1， $PLSR$ 指令重新初始化 PTO 硬件开始发送脉冲，之前被中断的脉冲发送被忽略。
- 脉冲发送的占空比为 50%ON、50%OFF，脉冲的发送完全由 PCA/PWM 的硬件中断处理，不受扫描周期的影响。
- 若 N 不为 0，当前脉冲发送的数量可以通过 **SMD4** 读取，数量为 N 的脉冲发送完毕后，特殊寄存器 **SM3.5** 被置 1。

软件配置

在 GUTTA System Utility 中，PTO 模块的配置窗口如下：

PTO | Pulse Train Output

说明



PTO模块通过PLSY指令来驱动, 当PLSY指令的使能端EN为1时, 模块将数量为N的脉冲以指定频率输出到OUT管脚上 (使用PLSR指令可带加减速).

- PLSY 脉冲输出指令
- PLSR 可调脉冲输出指令

☒ 配置PTO模块

☐ 将PCA/PWM从P1口切换到P4口

计数模块

PCA/PWM: [PTO]

?

输出管脚

P1.4/ADC4/CCP1/SS: [Q1.0] <PTO.OUT>

?

系统块设置参数必须下载才能生效

- **配置 PTO 模块:** 选择是否使能脉冲输出 PTO 模块, 如果使能, PTO 模块将配置单片机的 PCA/PWM 并共享使用 PCA/PWM 的输出管脚 P1.4/ADC4/CCP1/SS, 或者通过选择将 PCA/PWM 从 P1 口切换到 P4 口共享使用 P4.3/CCP1/SCLK。

特别说明

IAP12C5A60AD 除了标准的两个 51 定时器 T0 和 T1, 还提供了一个 PCA/PWM 模块。PCA/PWM 模块有自己独立的定时器/计数器。PCA/PWM 模块在高速输出模式时, 一旦发生比较匹配, 便自动翻转输出管脚。输出管脚默认是 P1.4/ADC4/CCP1/SS, 可以通过管脚的重映射功能配置到 P4.3/CCP1/SCLK。

在 PTO 模块被配置后, 一旦使用 PLSY 或者是 PLSR 指令, 指令会根据参数初始化 PCA/PWM 硬件, PCA/PWM 的定时器/计数器开始工作, 并在溢出时自动翻转输出管脚并在中断中判断是否继续发送, 发送的时间间隔是多少 (即脉冲频率)。

使用实例

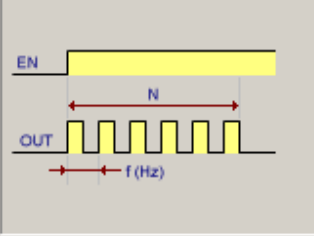
可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道, 单片机的 P4.3/CCP1/SCLK 和 MAX485 的 RE、DE 相连, 在不使用 MAX485 通讯时, 可以使用这个管脚作为脉冲输出信号, 并用示波器捕捉波形。单片机的 P1.4/ADC4/CCP1/SS 用于驱动 LED 段码, 我们可以利用 SOUT 模块, 给 LED 段码的 74HC164 发送特定数据, P1.4/ADC4/CCP1/SS 的电平, 便可以在 LED 段码上体现出来。

PTO 模块的配置:

PTO | Pulse Train Output

默认值
帮助

说明



PTO模块通过PLSY指令来驱动,当PLSY指令的使能端EN为1时,模块将数量为N的脉冲以指定频率f输出到OUT管脚上(使用PLSR指令可带加减速).

- **PLSY** 脉冲输出指令
- **PLSR** 可调脉冲输出指令

☒ 配置PTO模块

☐ 将PCA/PWM从P1口切换到P4口

计数模块 PCA/PWM: [PTO] ?

输出管脚 P1.4/ADC4/CCP1/SS: [Q1.0] <PTO.OUT> ?

系统块设置参数必须下载才能生效

SOUT 模块的配置:

SOUT | Scan Output

默认值

帮助

配置SOUT模块

电路参考

1: 并出, 573锁存

3: 串出, 573锁存

2: 串出, 不锁存

4: 串出, 595锁存

基本设定

映射地址

20

M区间偏移

数据线数量

8

片选线数量

1

数据取反后发送

发送数据

并行方式

串行方式

发送数据

CLK的上升沿

CLK的下降沿

刷新数据

片选正脉冲

片选负脉冲

管脚选择

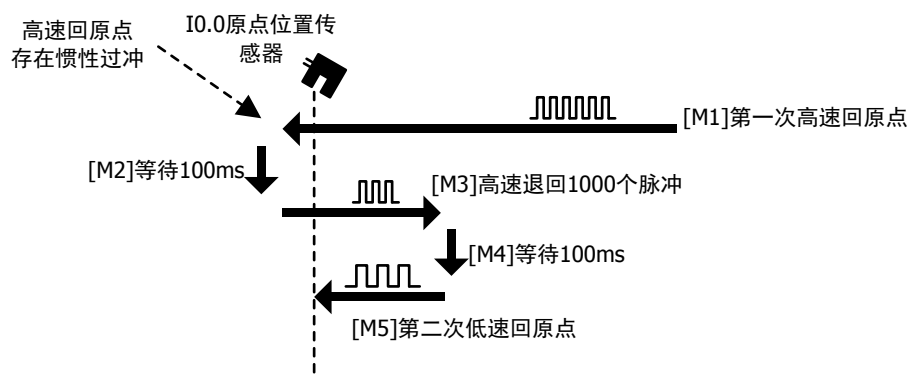
地址	管脚选择
Qn	
DS	P0. 6/AD6
CP	P0. 7/AD7
CS	
CS0	

P0. 6/AD6: [SOUT.DS]

系统块设置参数必须下载才能生效

除了配置软件的 PTO 模块, 还需要配置 P1.4/ADC4/CCP1/SS 管脚为数字量输出 Q1.0, 配置办法请参考 IO 模块的相关章节。

配置部分完成后, 为了测试驱动, 写一小段梯形图程序。这里我们用 PLSY 指令实现二次回原点。二次回原点能够解决大部分回原点的精度问题。



如上图, I0.0 为原点位置传感器, 按下 I0.1, 进入回原点流程。为了节约时间, 程序首先采用较高的速度驱动 Q1.0 发送脉冲, 运动机构向 I0.0 运动, 由于速度较快, 到达原点, 会存在一定的惯性过冲。在过冲位置等待 100ms 后, 退回 1000 个脉冲, 保证运动机构退回到原点前。等待 100ms 后, 采用较低的速度第二次回原点。由于这次速度较低, 能够达到较好的回原点精度。这里采用步进流程编写梯形图, 具体程序如下:

程序文件名: **10-EK51-PTO.vcw**。

MAIN (INT0):

NETWORK 0

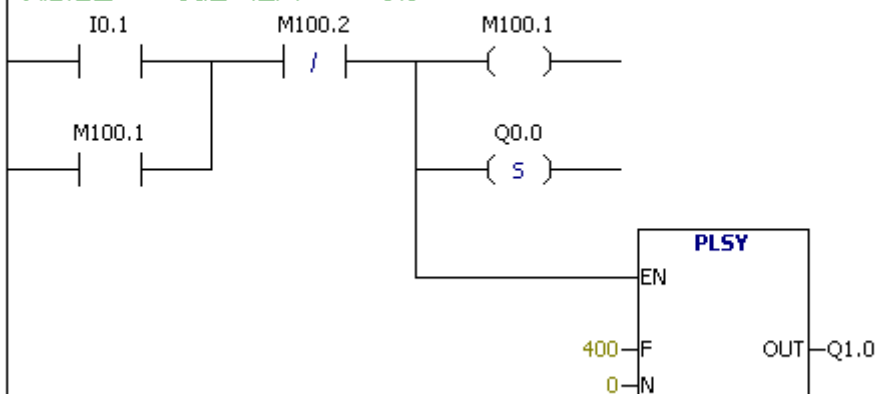
I0.0 原点位置传感器
I0.1 启动信号
Q1.0 脉冲输出
Q0.0 脉冲方向

[M100.1] 第一次高速回原点
[M100.2] 等待100ms
[M100.3] 高速退回1000个脉冲
[M100.4] 等待100ms
[M100.5] 第二次低速回原点
[M100.6] 结束

NETWORK 1

[M100.1] 第一次高速回原点

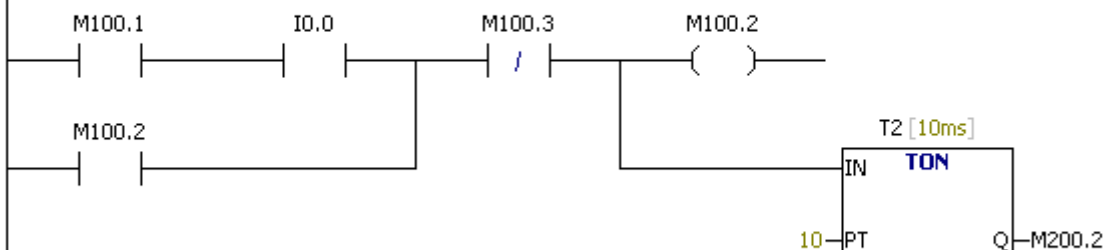
状态进入：按下I0.1
状态退出：I0.0到达，进入M100.2等待



NETWORK 2

[M100.2] 等待100ms

状态进入：M100.1时，I0.0到达
状态退出：延时到达，进入M100.3退回状态

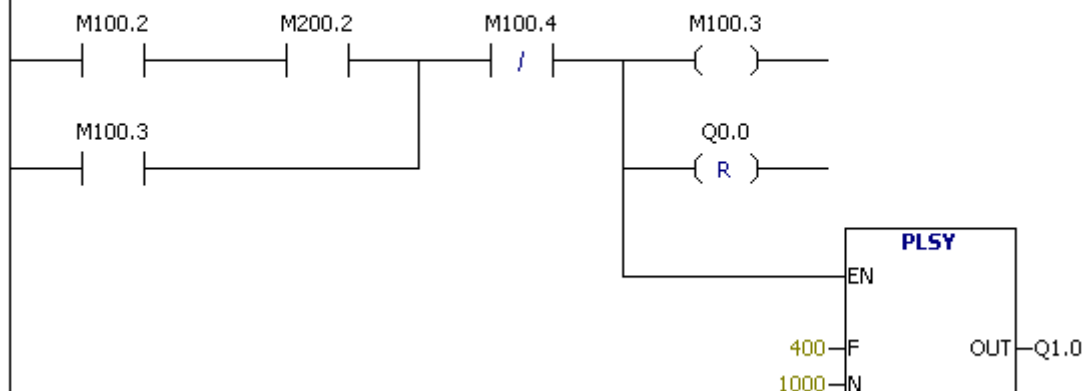


NETWORK 3

[M100.3] 高速退回1000个脉冲

状态进入：M100.2时，T2延时到达

状态退出：1000个脉冲发送完成，进入M100.4等待

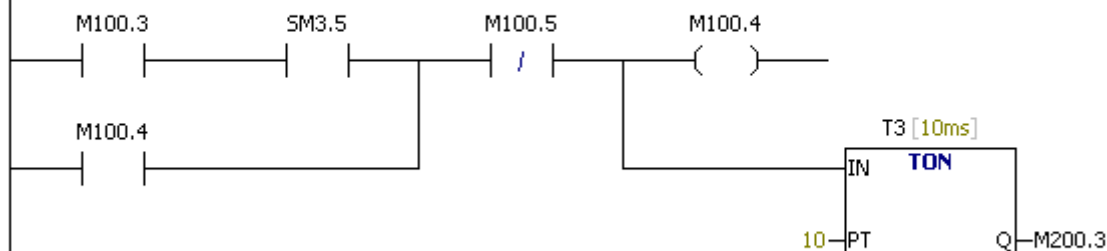


NETWORK 4

[M100.4] 等待100ms

状态进入：M100.3时，发送完成标志SM191.1置位

状态退出：延时到达，进入M100.5低速回原点

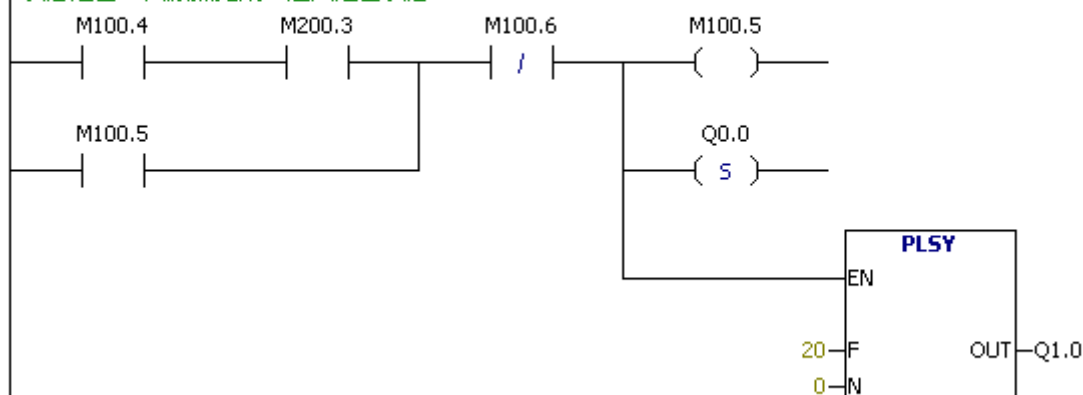


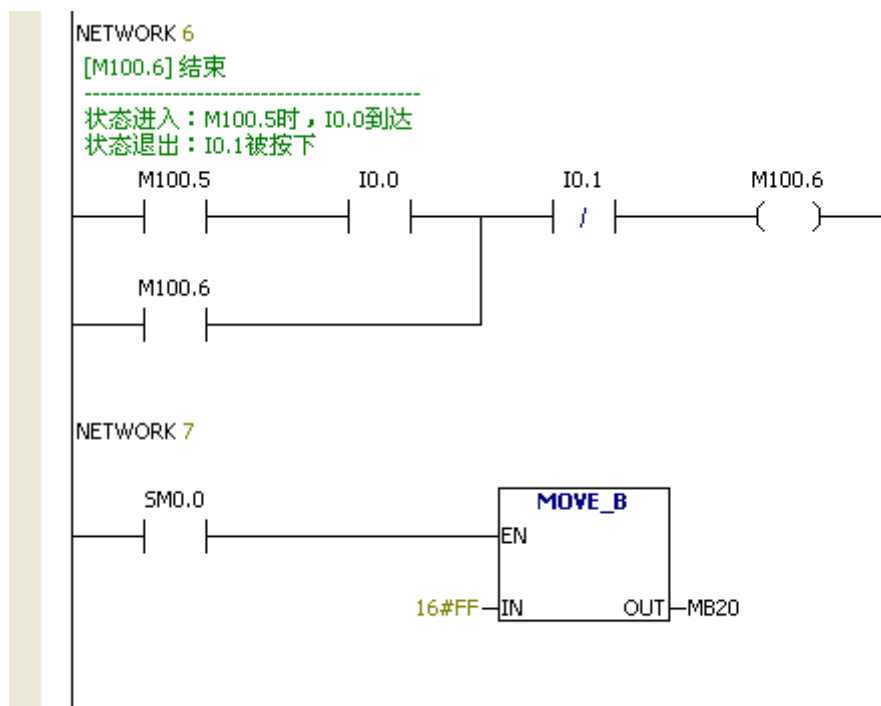
NETWORK 5

[M100.5] 第二次低速回原点

状态进入：M100.4时，T3延时到达

状态退出：回原点完成，进入退出状态





程序很简单，下载后进入监控状态。

按下 IO.1，开始第一次回原点。此时 Q1.0 以 400Hz 的频率发送脉冲，同时方向控制 Q0.0 亮。按下 IO.0 模拟原点到达，Q1.0 停止脉冲发送。等待 100ms 后，Q0.0 灭，Q1.0 以 400Hz 的频率发送脉冲。退回 1000 步后（2.5 秒），Q1.0 脉冲停止。等待 100ms 后，Q1.0 以 20Hz 的频率发送脉冲，同时方向控制 Q0.0 亮。此时再按下 IO.0 模拟原点到达，Q1.0 停止脉冲发送，二次回原点完成。

脉宽调制 [PWM]

简介

同样是高速脉冲输出，PWM 模块和 PTO 模块不一样，PTO 模块以指定的频率发送一定数量的脉冲，但是占空比始终是 50%ON、50%OFF。PWM 模块可以指定发送脉冲的频率和占空比，但是不能指定脉冲数量。PTO 模块一般用于运动控制。PWM 模块一般用于信号量的传递。一个最简单的应用就是用 PWM 模块来驱动一个滤波电路得到一个和 PWM 占空比相关的模拟量，这也是一种常用的 D/A 转换。

PWM 模块依然使用 IAP12C5A60AD 单片机自带的 PCA/PWM 模块来产生 PWM 脉冲。在 GUTTA System Utility 中配置好 PWM 块后，PWM 模块自动占用 IAP12C5A60AD 的 PCA/PWM 硬件，并共享使用 PCA/PWM 的输出管脚 P1.4/ADC4/CCP1/SS（或重映射为 P4.3/CCP1/SCLK）。因此需要将这个管脚在 IO 模块中配置为数字量输出。在使用 PWM 指令时，这个数字量输出必须给出以便指令确认是哪一路高速输出（虽然目前 EC30-EK51 只能配置一路硬件高速输出，但是考虑到以后的扩展和兼容，这个数字量输出目前必须正确给出，PWM 指令才真正启动 PWM 模块的输出）。PWM 模块可以配置为两种工作模式，中断模式 PWM 和硬件模式 PWM。

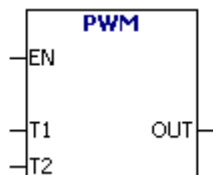
中断模式 PWM 的实现和 PTO 模块类似，PCA/PWM 模块依然配置为高数输出。每次 PCA/PWM 的定时器/计数器发生溢出时，PCA/PWM 就翻转脉冲输出管脚并产生中断，中断

根据当前脉冲是高电平还是低电平载入正确的定时器/计数器初始化值。

硬件模式 PWM 的实现和中断模式略有不同，PCA/PWM 模块配置为 PWM 工作模式，16 位的 PCA/PWM 定时器计数器被分为低 8 位和高 8 位的两个部分。低 8 位作为 8 位的定时器/计数器工作，在 8 位定时器/计数器发生比较匹配或溢出后，单片机不产生中断，而是翻转脉冲输出管脚。全硬件的 PWM 由于不需要频繁产生中断，可以产生很高频率的 PWM 输出且不占用 CPU 时间。

中断模式 PWM

PWM 模块可以通过 PWM 指令来驱动，当 PWM 指令的使能端 EN 由 0 变为 1 时，PWM 指令初始化 PWM 硬件并持续发送高速脉冲。在一个脉冲周期中，输出管脚 OUT 在 T_1 时间内为 1，在 $T_2 - T_1$ 时间内为 0。也就是说，高速脉冲的占空比为 $T_1 / (T_2 - T_1)$ 、脉冲的周期是 T_2 、脉冲频率为 $1/T_2$ 。



T_1 为一个脉冲周期输出为 1 的时间，单位是 10 μ s。

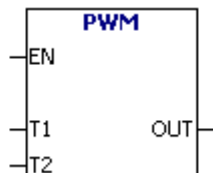
T_2 为一个脉冲周期的时间，单位是 10 μ s。

OUT 为输出脉冲的管脚。

- T_2 的范围是 2 ~ 4000 (20 μ s ~ 40ms)。
- T_1 必须小于或等于 T_2 。 T_1 如果为 0，不输出脉冲，脉冲输出管脚始终为 0。 T_1 如果等于 T_2 ，不输出脉冲，脉冲输出管脚始终为 1。
- 脉冲发送过程中可以改变 T_1 和 T_2 的值，脉冲输出的占空比和周期随之改变。
- 在脉冲发送过程中如果 PWM 指令的使能端 EN 变为 0，脉冲发送停止。EN 由 0 变为 1，PWM 指令重新初始化 PWM 硬件开始发送脉冲。

硬件模式 PWM

PWM 模块可以通过 PWM 指令来驱动，当 PWM 指令的使能端 EN 由 0 变为 1 时，PWM 指令初始化 PWM 硬件，以及可能的 PWM 时钟源定时器/计数器 0，并持续发送高速脉冲。输出管脚 OUT 在一个周期的 $T_1/256$ 时间内为 1，在一个周期的 $(256 - T_1)/256$ 时间内为 0。



T_1 为一个脉冲周期输出为 1 的时间，单位是一个周期的 256 分之 1。

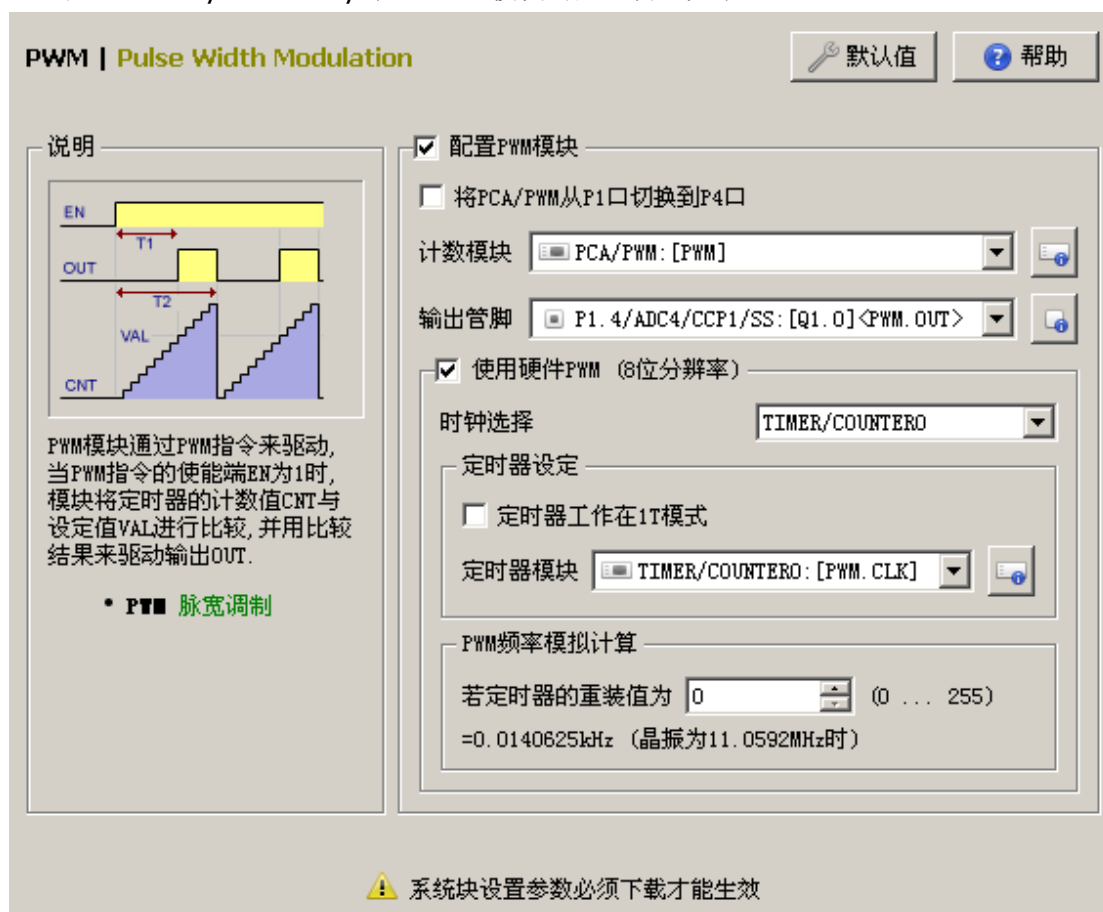
T_2 为定时器/计数器 0 的重装值，如果 PCA/PWM 模块采用系统时钟，脉冲周期只和分频有关系， T_2 无意义。

OUT 为输出脉冲的管脚。

- 脉冲频率的范围和 PCA/PWM 模块采用的时钟源有关，最高为 43.2KHz（直接使用 FOSC_CLK），最低为 0.0140625Hz（使用定时器 T0 的溢出作为时钟源，T0 工作在 51 标准模式，重装值为 0）。
- 脉冲发送过程中可以改变 T1 和 T2 的值，脉冲输出的占空比和周期随之改变。
- 在脉冲发送过程中如果 PWM 指令的使能端 EN 变为 0，脉冲发送停止。EN 由 0 变为 1，PWM 指令重新初始化 PWM 硬件开始发送脉冲。

软件配置

在 GUTTA System Utility 中，PWM 模块的配置窗口如下：



- **配置 PWM 模块：**选择是否使能脉宽调制 PWM 模块，如果使能，PWM 模块将配置单片机的 PCA/PWM 并共享使用 PCA/PWM 的输出管脚 P1.4/ADC4/CCP1/SS，或者通过选择将 PCA/PWM 从 P1 口切换到 P4 口共享使用 P4.3/CCP1/SCLK。
- **使用硬件 PWM 模块：**不选择为中断 PWM 工作模式，选择为硬件 PWM 工作模式。
- **时钟选择：**选择 PCA/PWM 的工作时钟，可供选择的时钟有：
 - FOSC_CLK
 - FOSC_CLK/2
 - FOSC_CLK/4
 - FOSC_CLK/6
 - FOSC_CLK/8
 - FOSC_CLK/12
 - TIMER/COUNTER0

如果选用 FOSC_CLK ~ FOSC_CLK/12, 指令 PWM 的重装值 T2 操作数被忽略。如果使用 TIMER/COUNTER0, PCA/PWM 模块采用 TIMER/COUNTER0 的溢出作为时钟信号, TIMER/COUNTER0 必须工作在 8 位自动重装模式, 运行过程中 PWM 指令的 T2 操作数可以动态修改这个重装值以便输出过程中改变脉冲频率。同时, TIMER/COUNTER0 可以选择为工作在 1T 模式。在 1T 模式下, TIMER/COUNTER0 的当前计数每一个晶振周期加 1, 而不是每一个机器周期加 1, 我们知道 51 单片机中 12 个晶振周期为 1 个机器周期, 故采用 1T 模式, 脉冲的实际输出频率会比标准模式快 12 倍。

如果是采用硬件 PWM 模式, 实际的脉冲频率会在 PWM 频率模拟计算中给出。

特别说明

中断模式下, 管脚电平受输出管脚 IO 配置正负逻辑的影响。正逻辑 1 为高电平; 0 为低电平。负逻辑 1 为低电平; 0 为高电平。硬件模式下, 管脚电平不受输出管脚 IO 配置正负逻辑的影响, 而是始终采用负逻辑。

硬件模式的 PWM 和中断模式的 PWM 不一样, 硬件模式的 PWM 一旦使能, 会修改管脚的电器特性 (原 IO 模块中定义的管脚电器特性), 根据宏晶的《STC12C5A60S2 系列单片机器件手册》的 PWM 相关资料我们知道, 若 PCA/PWM 工作在 PWM 模式下, 会对原管脚配置做如下修改:

PWM 之前的管脚配置	PWM 时的管脚配置
准双向口 (弱上拉)	强推挽输出
强推挽输出	强推挽输出
仅为输入 (高阻)	PWM 无效
开漏	开漏

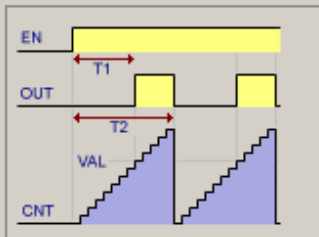
使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道, 单片机的 P4.3/CCP1/SCLK 和 MAX485 的 RE、DE 相连, 在不使用 MAX485 通讯时, 可以使用这个管脚作为脉冲输出信号, 并用示波器捕捉波形。单片机的 P1.4/ADC4/CCP1/SS 用于驱动 LED 段码, 我们可以利用 SOUT 模块, 给 LED 段码的 74HC164 发送特定数据, P1.4/ADC4/CCP1/SS 的电平, 便可以在 LED 段码上体现出来。

PWM 模块的配置:

PWM | Pulse Width Modulation

说明



PWM模块通过PWM指令来驱动, 当PWM指令的使能端EN为1时, 模块将定时器的计数值CNT与设定值VAL进行比较, 并用比较结果来驱动输出OUT.

- **PWM** 脉宽调制

☒ 配置PWM模块
默认值 帮助

☐ 将PCA/PWM从P1口切换到P4口

计数模块 PCA/PWM: [PWM] ⓘ

输出管脚 P1.4/ADC4/CCP1/SS: [Q1.0] <PWM.OUT> ⓘ

☒ 使用硬件PWM (8位分辨率)

时钟选择 TIMER/COUNTER0 ⓘ

定时器设定

☐ 定时器工作在1T模式

定时器模块 TIMER/COUNTER0: [PWM.CLK] ⓘ

PWM频率模拟计算

若定时器的重装值为 0 (0 ... 255)

=0.0140625kHz (晶振为11.0592MHz时)

⚠ 系统块设置参数必须下载才能生效

SOUT 模块的配置:

SOUT | Scan Output

默认值
帮助

☒ 配置SOUT模块

电路参考

1: 并出, 573锁存

3: 串出, 573锁存

2: 串出, 不锁存

4: 串出, 595锁存

基本设定

映射地址: M区间偏移

数据线数量:

片选线数量:

☐ 数据取反后发送

发送数据: ☐ 并行方式 ☒ 串行方式

发送数据: ☒ CLK的上升沿 ☐ CLK的下降沿

刷新数据: ☒ 片选正脉冲 ☐ 片选负脉冲

管脚选择

地址	管脚选择
Qn	
DS	P0. 6/AD6
CP	P0. 7/AD7
CS	
CS0	

☒ P0. 6/AD6: [SOUT. DS]

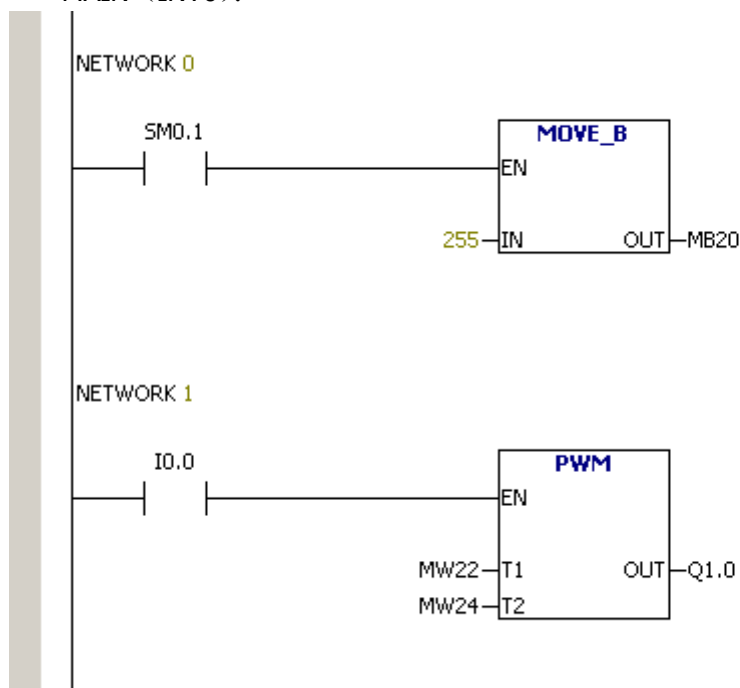
系统块设置参数必须下载才能生效

除了配置软件的PWM模块, 还需要配置 P1.4/ADC4/CCP1/SS 管脚为数字量输出 Q1.0, 配置办法请参考 IO 模块的相关章节。

配置部分完成后, 为了测试驱动, 写一小段梯形图程序如下:

程序文件名: **11-EK51-PWM.vcw**。

MAIN (INT0):



将程序下载后进入监控状态, 在状态表中将 MW22 的值设置为 127, 将 MW22 的值设

置为 255，用示波器观察 PLC 数字量输出 Q1.0（单片及管脚 P1.4/ADC4/CCP1/SS）的波形是否为占空比为 50%ON、50%OFF 的方波，频率是否为 3.6KHz。然后将 MW22 修改为 16，将 MW24 修改为 254，观察波形和频率的变化。

第 7 章 扩展配置说明

杂项 [MISC]

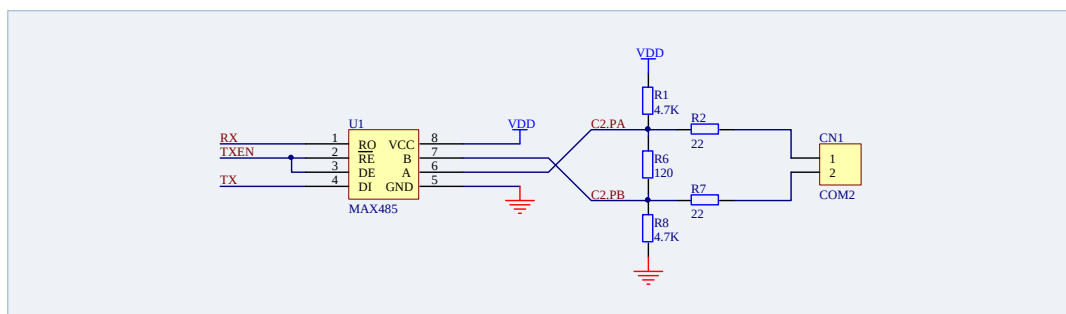
简介

通过 MISC 模块，我们可以配置以下项目：

1. **INT0、INT1** 两个外部中断。
2. 串口 **PORT0** 发送使能管脚 **TXEN**。
3. 输出使能 **OE**，可用于防止上电跳继电器。
4. 运行状态灯 **RUN** 的管脚配置和电平逻辑。
5. 通讯状态灯 **COM** 的管脚配置和电平逻辑。
6. 错误状态灯 **ERR** 的管脚配置和电平逻辑。

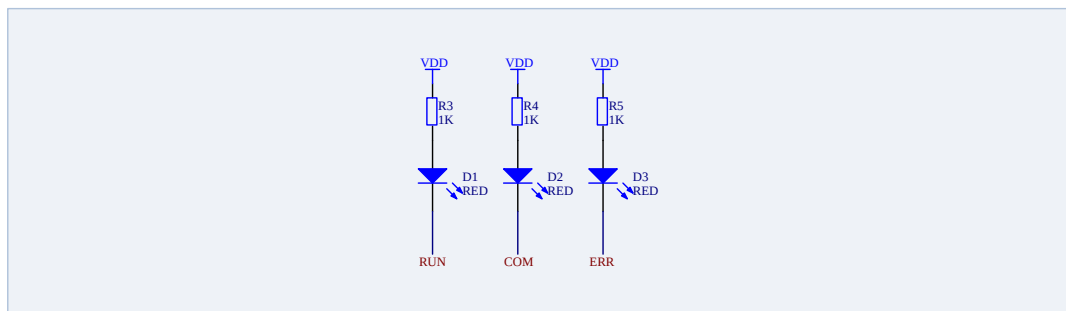
外部电路

1. **INT0**、**INT1** 中断使能后，**INT0** 对应的管脚 **P3.2** 和 **INT1** 对应的管脚 **P3.3** 所采用的是共享使用。这也就意味着，必须在其它地方对 **P3.2** 和 **P3.3** 进行配置才能使用这两个管脚的外部中断。**P3.2** 和 **P3.3** 一般情况下被配置为输入管脚（采样外部中断事件），被配置为输出管脚并不影响单片机根据电平产生中断。**INT0**（**INT1**）被配置后，51 单片机的 **IT0**（**IT1**）和 **EX0**（**EX1**）寄存器位在 **PLC** 复位后被置 1。若 **P3.2**（**P3.3**）的电平发生了翻转，则产生中断。**PLC** 系统检测到这个中断后，会立即打断当前主循环扫描的执行，开始执行中断任务（如果中断用 **ATTCH** 指令挂接了任务同时用 **ENI** 指令打开了中断允许标志位）。
2. 串口（**PORT0**）发送使能管脚一旦被配置，**PLC** 复位后，**TXEN** 对应的管脚被拉低以进入接收串口数据的状态。**PLC** 系统只有在接收到数据需要发送数据回应主站时，或者是作为主站在指令的驱动下主动发起通讯请求时，才会拉高 **TXEN** 管脚的电平。如果外部使用 **MAX485** 芯片，典型的连接就是这个样子：



3. 输出使能（OE）一般直接连接到锁存芯片的 OE 端，详细情况可参考 SOUT 模块。

4. RUN、COM、ERR 三个状态灯用于指示当前 PLC 的工作状态。由于 IAP12C5A60AD 芯片的管脚差不多可以直接驱动一个 LED，因此也可以不加而外的驱动电路直接连接。



软件配置

在 GUTTA System Utility 中，MISC 模块的配置窗口如下：

MISC | Miscellany 默认值 帮助

☐ 配置外部中断INT0

☒ 配置外部中断INT1

☒ 配置PORT0发送使能TXEN

☐ 配置输出使能OE

管脚选择

电器特性

示意图

逻辑电平

系统状态指示

地址	管脚选择	电器特性
RUN	P3. 5/T1/CLKOUT1	准双向口
COM	P3. 6/WR	准双向口
ERR	P3. 7/RD	准双向口

系统块设置参数必须下载才能生效

- **配置外部中断 INT0**：选择是否使能 51 单片机外部中断 INT0。如果使能，会自动共享使用 P3.2/INT0 管脚，一般将 P3.2/INT0 配置为数字量输入。

- **配置外部中断 INT1:** 选择是否使能 51 单片机外部中断 INT1。如果使能, 会自动共享使用 P3.3/INT1 管脚, 一般将 P3.3/INT1 配置为数字量输入。
- **配置 PORT0 发送使能 TXEN:** 选择是否使能 PORT0 发送使能 TXEN, 如果使能, 需要为 TXEN 功能选择一个单片机管脚, 被选择的管脚会在 PLC 复位后配置为准双向口(弱上拉), 并在 PLC 的 PORT0 发送数据前置 1(弱上拉), 在发送数据结束后置 0(下拉)。
- **配置输出使能 OE:** 选择是否使能输出使能 OE, 如果启用, 需要为 OE 功能选择一个单片机管脚, 被选择的管脚会在 PLC 复位后配置为弱上拉, 直到第一次输出扫描结束后下拉, 然后一直保持低电平。
- **系统状态指示:** 分别配置 RUN、COM、ERR 三个状态灯的驱动管脚, 选择管脚后, 需要根据具体的外部电路确定对应管脚的电器特性和逻辑电平。

特别说明

虽然可以在 GUTTA System Utility 中配置 PORT0 的发送使能管脚 TXEN, 但是硬件系统必须预留 RS232 通讯(或者是全双工的 RS422)。这是因为, 在 PORT0 的 TXEN 管脚未被配置前, 如果直接采用 RS485 通讯(半双工), 是不能下载 TXEN 配置信息的。

一种可行的办法就是事先通过 RS232 通讯下载一个包含 TXEN 管脚配置信息的空程序, 再用跳线转为 RS485 通讯。这个时候用 RS485 下载程序时, 只要新下载的程序依然正确配置了 TXEN 管脚, 就不会有任何问题。如果新程序没有配置 TXEN 管脚, 或者配置错误, 新程序依然可以通过 RS485 下载, 当最后一条通讯指令(强制 PLC 复位)执行后, RS485 通讯就不能正常工作了。

总之, 可以用 RS485 进行程序的下载、监控和 MODBUS 通讯, 但是前提是必须先用 RS232 通讯配置好 PORT0 的 TXEN。

使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道, 单片机的 INT0 和 PCF8563T 的 CLKOUT 相连接, INT1 和试验板的第一个按键相连, 为了测试方便, 我们只测试按键触发的中断, 即只配置 INT1。试验板 CPU-EC20 同时提供一块 MAX485CSA 芯片, 可以通过跳线连接到单片机的 P3.0、P3.1。MAX485CSA 的发送控制管脚 RE、DE 和单片机的 P4.3 相连接。CPU-EC20 试验板同时提供了 3 个状态灯, 分别用于 3 个运行状态的指示, 它们分别由单片机的 P3.5、P3.6、P3.7 管脚来驱动。根据以上信息, 对应的软件配置如下:

MISC | Miscellany

默认值
帮助

☐ 配置外部中断INT0

☒ 配置外部中断INT1

☒ 配置PORT0发送使能TXEN

如果PORT0的电器协议为RS485, 需要额外配置一个发送使能管脚

☐ 配置输出使能OE

系统状态指示

地址	管脚选择	电器特性
RUN	P3. 5/T1/CLKOUT1	准双向口
COM	P3. 6/WR	准双向口
ERR	P3. 7/RD	准双向口

管脚选择

电器特性

☒ 准双向口
 ☐ 推挽输出

☐ 仅为输入
 ☐ 开漏输出

示意图

逻辑电平

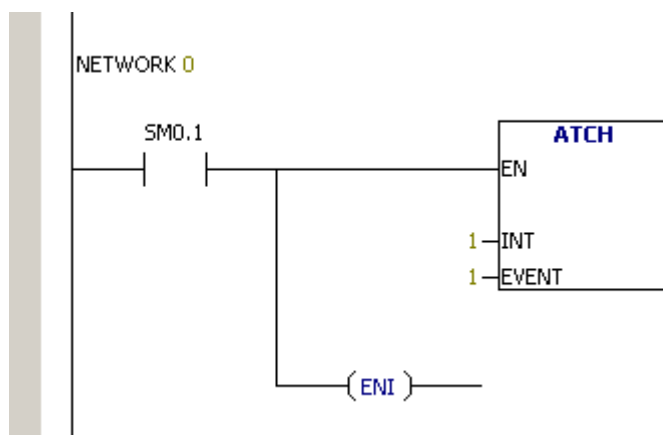
☒ 负逻辑
 ☐ 正逻辑

⚠ 系统块设置参数必须下载才能生效

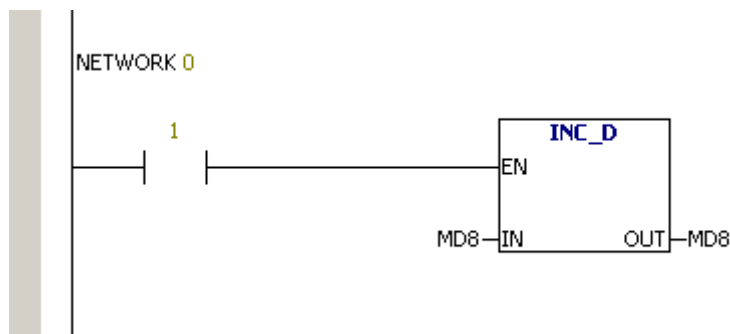
配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**12-EK51-MISC.vcw**。

MAIN (INT0):



INT_1 (INT1):



程序很简单，首先在主循环 MAIN 中，根据第一次扫描标志位 SM0.1，将中断 INT1 和中断事件 1（外部中断 1）挂接，然后通过 ENI 指令使能中断。INT1 将 MD8 自加，用以记录进入中断 INT1 的次数。

下载好程序后，连线进入调试状态，观察 RUN 灯是否长亮（运行状态），观察 COM 灯是否闪烁（调试的时候必须不断通讯）。然后按下 I0.0，观察 MD8 值的变化。

可以改变试验板上 JP6 跳线的位置，采用 RS485 通讯后，检验在 RS485 通讯时，是否依然可以进行程序的监控和下载。

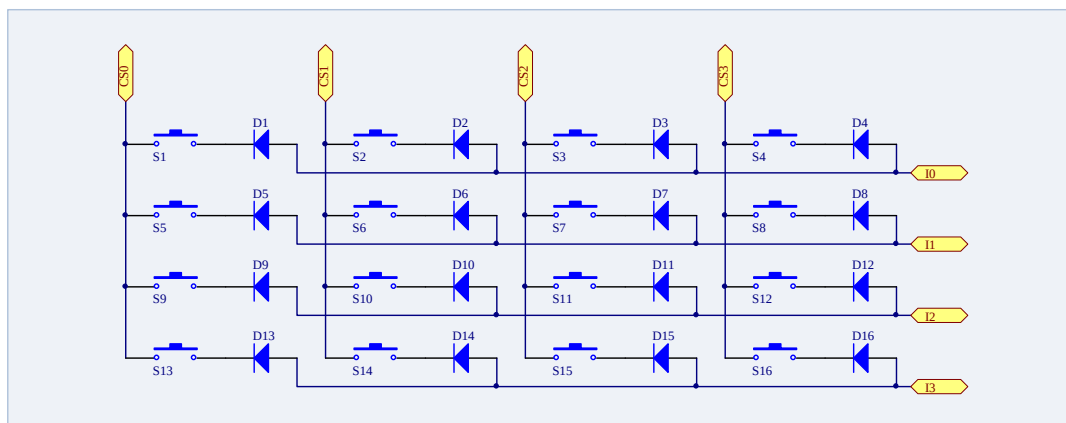
输入扫描 [SIN]

简介

若数字量输入采样速度要求不高，采样点数却很多，可以采用分组扫描的方式读取数字量输入。这样做可以节省单片机的管脚资源，同时也能简化布线。EC30-EK51 支持最大 8×8 的输入矩阵。每组 8 个，总共 8 组，即 64 个数字量输入，却只需要 16 个单片机管脚。输入扫描最典型的应用就是用于处理用户按键，按键的数量一般较多但对处理速度没有很高的要求。

外部电路

使用 EC30-EK51 自带的输入扫描 SIN 模块，推荐的电路图如下：

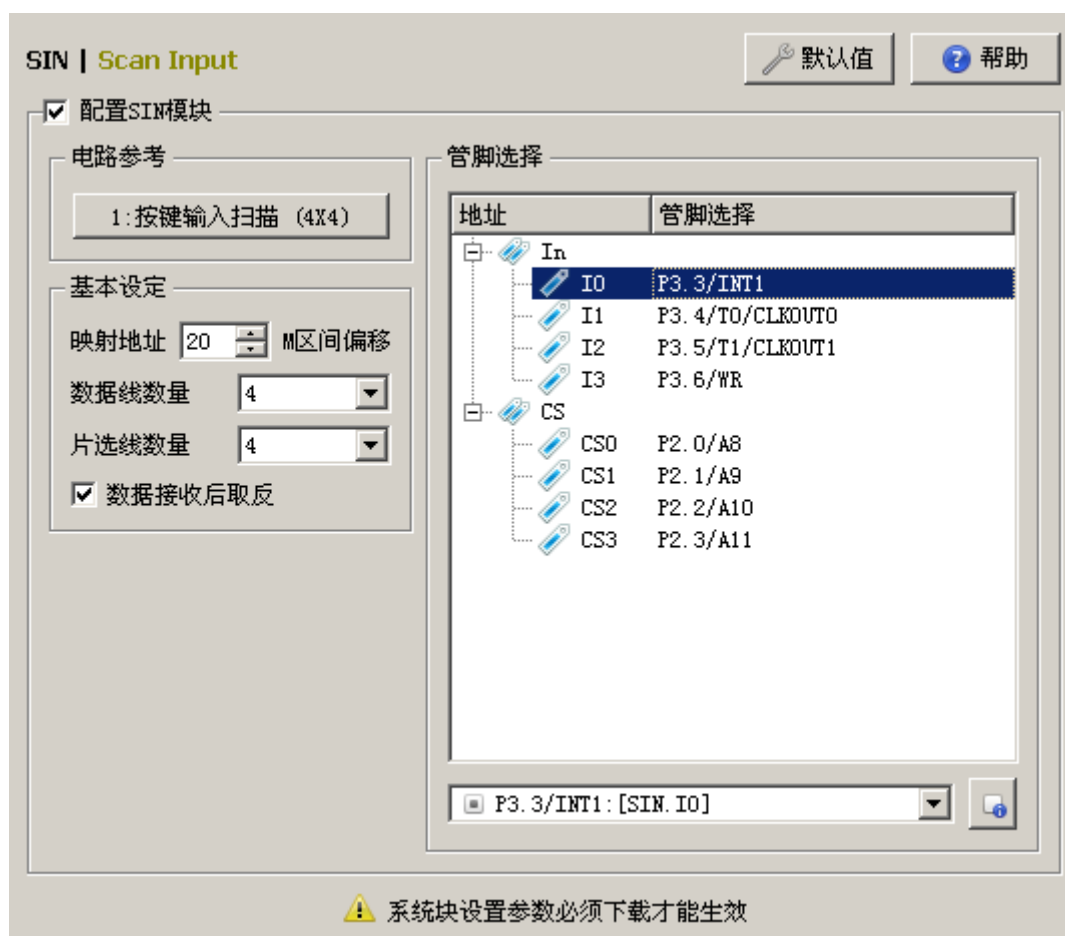


S1 ~ S16 可以是机械按键，也可以是其它电子开关（例如光藕）。D1 ~ D16 为单向导通的二极管，用于防止多个按键同时按下后互相干扰。SIN 模块在 PLC 复位后，将 I0 ~ I3

都设置为弱上拉（因此外部可以省去上拉电阻），同时配置 CS0 ~ CS3 为开漏输出，并置 1（对地不导通）。在 SIN 模块需要读取数据时，先置 CS0 为 0（对地导通），读取 I0 ~ I3 的值，I0 ~ I3 的值此时就反映了 S1、S5、S9、S13 的通断情况，读取结束后置 CS0 为 1（对地不导通）。然后置 CS1 为 0（对地导通），读取 I0 ~ I3 的值，I0 ~ I3 的值此时就反映了 S2、S6、S10、S14 的通断情况，读取结束后置 CS1 为 1（对地不导通）。如此循环，单片机就能够读取所有开关 S1 ~ S16 的通断状态。

软件配置

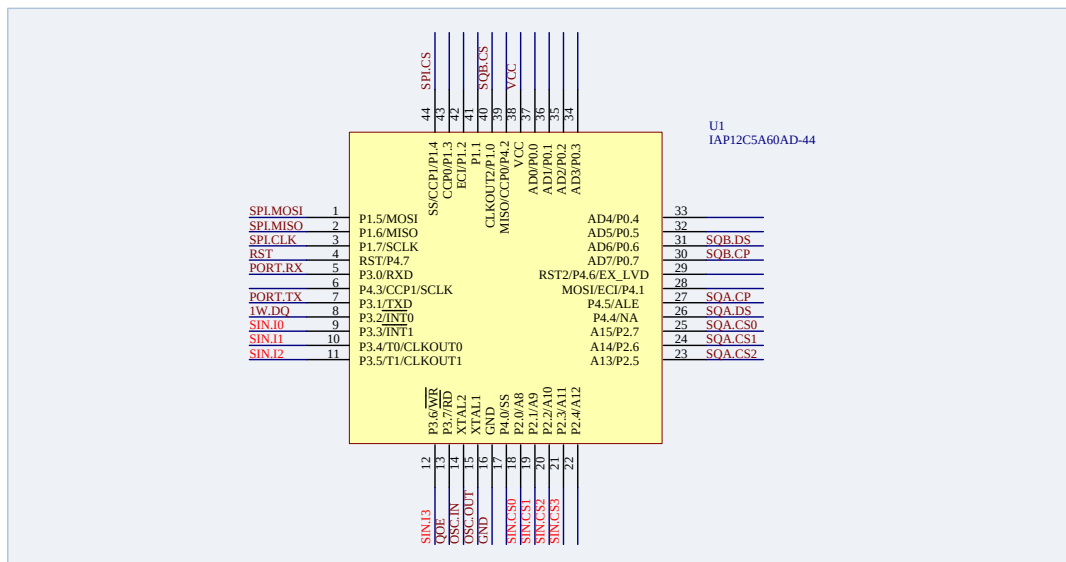
在 GUTTA System Utility 中，SIN 模块的配置窗口如下：



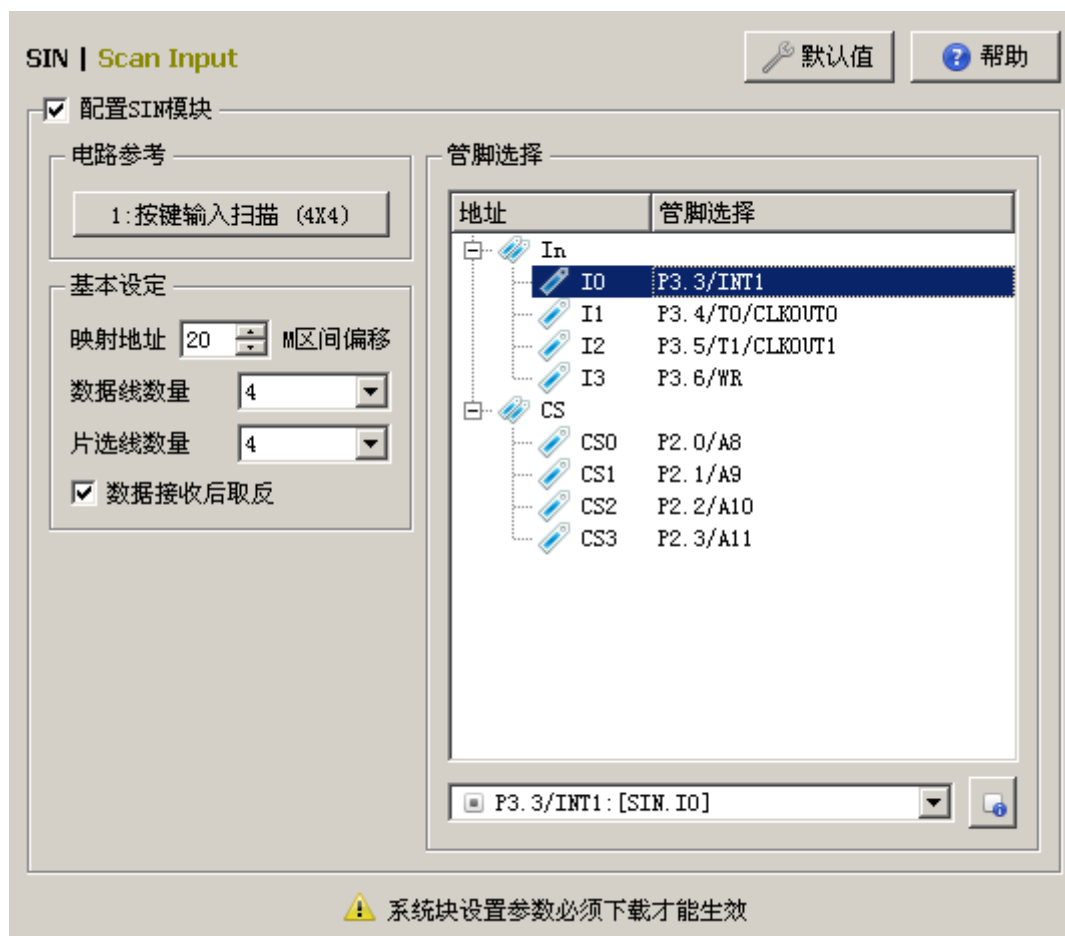
- **配置 SIN 模块：**选择表示使能 SIN 模块，不选择表示禁用 SIN 模块。
- **映射地址：**SIN 读取数据的存放地址。例如上面设置 M 区间偏移为 20，那么通过 CS0 读取的开关值放置在 MB20 中，通过 CS1 读取的开关值放置在 MB21 中，依此类推。
- **数据线数量：**在按键扫描中，单次扫描读取的开关数量，即 In 输入管脚的个数。
- **片选线数量：**在按键扫描中，总共的扫描次数，即 CSn 驱动管脚的个数。
- **数据接收后取反：**在电路图中，所有的 In 都是单片机弱上拉，CSn 驱动，对地导通。如果开关 Sn 断开（对地断开），对应的 In 线依然保持弱上拉，故为高电平 1。如果开关 Sn 接通（对地接通），对应的 In 线下拉到地，故为低电平 0。电平的高低与开关的通断刚好是相反的，即负逻辑。为了符合使用习惯，可以在这里选择数据接收后取反，那么最后在 MB 中，0 就代表开关断开，1 就代表开关接通。

使用实例

试验板 EC30-EK51-EVAL 有一个 4×4 的按键输入矩阵，实际电路和上面给出的电路图完全一致，而具体连接到单片机如下图：



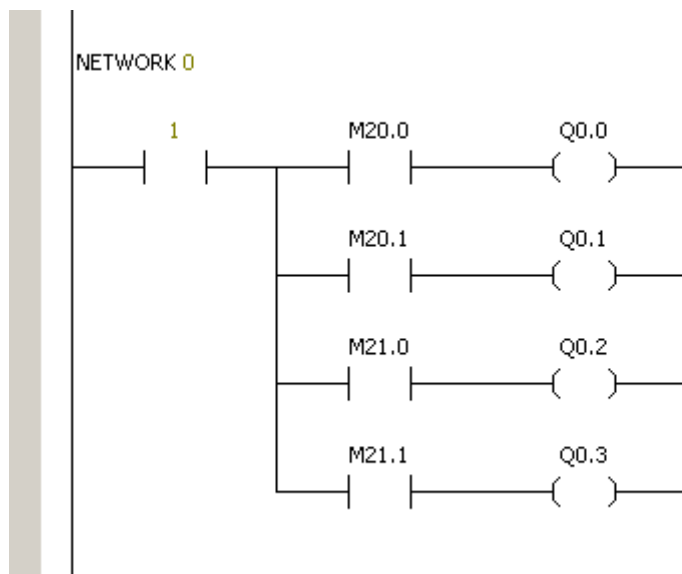
对应的软件配置如下：



配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名: **13-EK51-SIN (EVAL).vcw**。

MAIN (INT0):



虽然我们没有配置输出，但是我们可以通过监控观察 Q0.0 ~ Q0.3 的变化。将程序下载到实验板 EC30-EK51-EVAL 上并进入监控模式，分别按下输入 S1、S2、S5、S6，观察输出 Q0.0 ~ Q0.3 的变化。

输出扫描 [SOUT]

简介

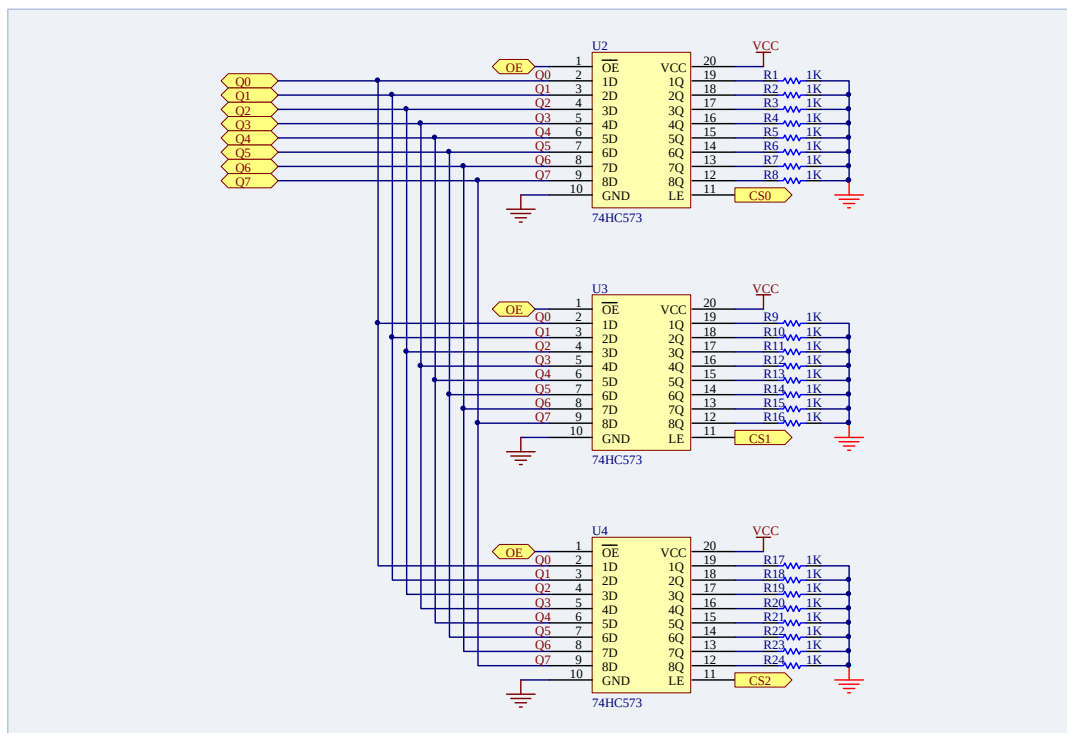
若数字量输出速度要求不高，可以采用串转并或者分组扫描的方式设置数字量输出。这样做可以节省单片机的管脚资源，同时也能简化布线。在 EC30-EK51 的 SOUT 模块中，支持最大 8×8 的输出矩阵。每组 8 个，总共 8 组，即 64 个数字量输出。和输入扫描 SIN 类似，SOUT 模块在不同的时间分别扫描输出 8 组数据，和 SIN 模块不同的是，由于输出状态必须保持（输入只需要读一次状态即可），SOUT 模块必须使用具有锁存能力的逻辑 IC 来保持输出状态。

在 SOUT 模块的模型中，可以定义 Q0 ~ Q7 共 8 条输出线，CS0 ~ CS7 共 8 条片选线。同时 Q0 ~ Q7 也可以采用串转并的方式设定，即可以用 DS（数据线）、CP（时钟线）两根线代替。

外部电路

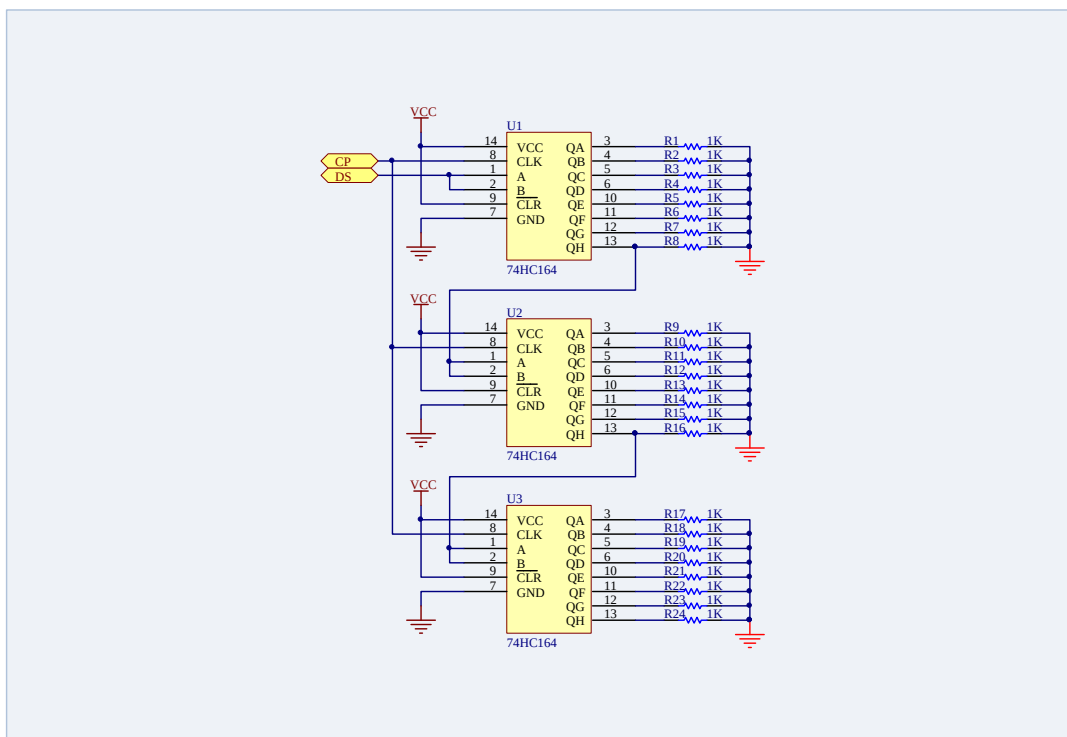
SOUT 模块具有较高的配置能力，支持的外部电路也有很多种。这里我们举例给出几种基本的使用方式：

并出 573 锁存



单片机采用并行的方式发送数据 Q0 ~ Q7，每次发送数据结束后，驱动当前数据锁存芯片对应的 CS 线更新数据。

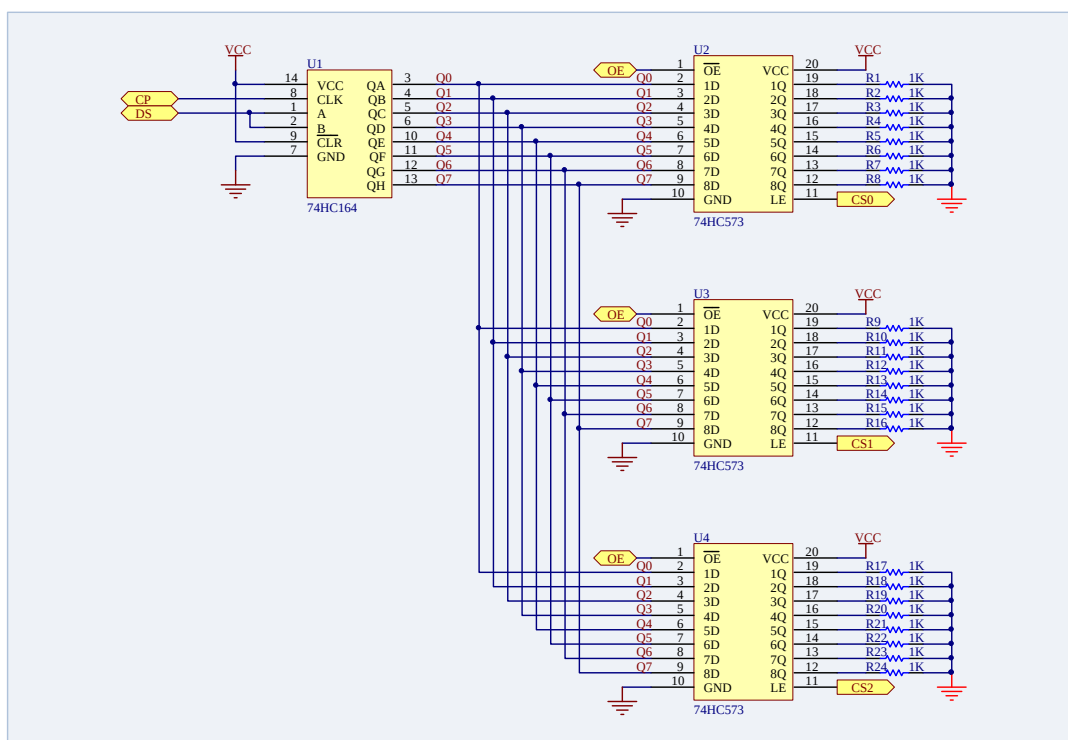
串出不锁存



单片机采用串行的方式通过 DS、CP 发送数据 Q0 ~ Q7，一组数据发送结束后，SOUT 模块会尝试驱动当前数据对应的 CS 线，这种情况下 CS 线不用设置，这个操作将被 SOUT 模块忽略，然后单片机继续通过 DS、CP 发送下一组数据。由于外部的 74HC164 采用级连的方式连接，最终第一组数据被发送至 U3，第二组数据被发送至 U2，依此类推。

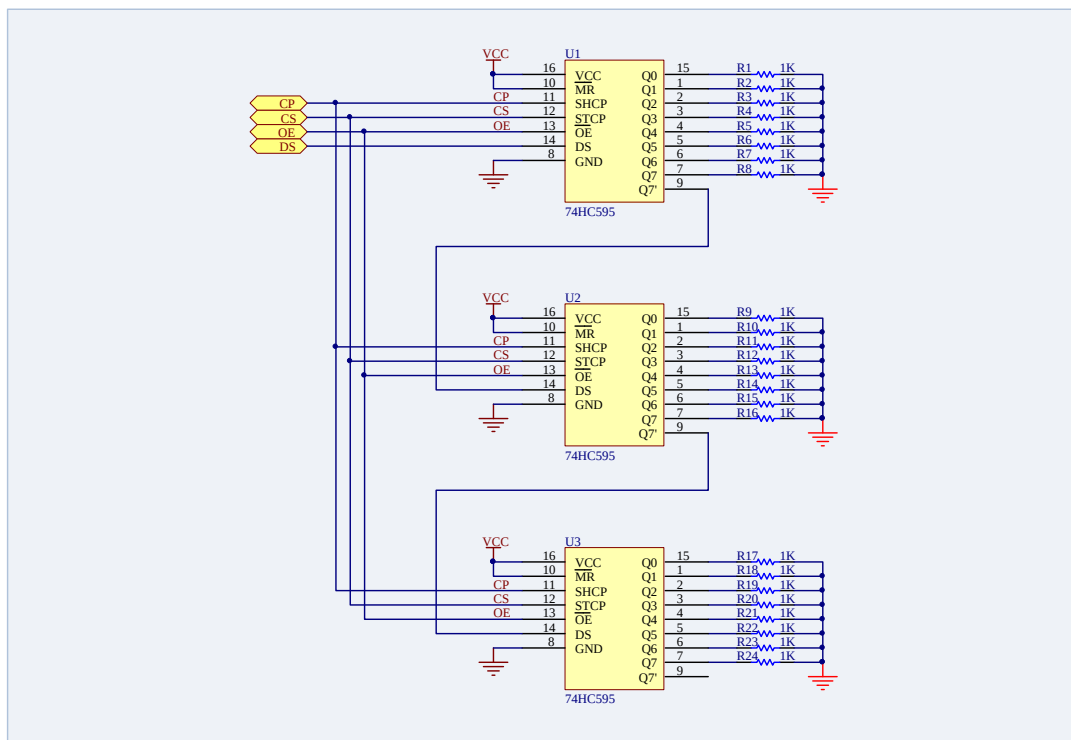
这种方式由于没有锁存功能，每次在更新时，输出状态会持续变化。这种连接一般只能在对输出动作瞬间变化不敏感的场所使用，例如驱动一组 LED 指示灯。在更新的瞬间 LED 指示灯会发生抖动，但是由于更新的时间很短，人眼是无法察觉的。

串出 573 锁存



单片机采用串行的方式通过 DS、CP 发送数据 Q0 ~ Q7，每次发送数据结束后，驱动当前数据对应的 CS 线更新数据。

串出 595 锁存



和 74HC164 不同，74HC595 除了具有移位寄存器（串转并），还自带了锁存器，这就能够防止单片机通过 DS、CP 更新数据时，输出发生抖动。在 SOUT 模块中，必须将连接 74HC595 STCP 的单片机管脚配置为最后一组数据的更新线 CS3，确保单片机在通过 DS、CP 发送完所有的数据后，数据才通过 CS3 一次性更新到所有 74HC595 的输出上。

由于外部的 74HC595 采用级连的方式连接，最终第一组数据被发送至 U3，第二组数据被发送至 U2，依此类推。

软件配置

在 GUTTA System Utility 中，SOUT 模块的配置窗口如下：

SOUT | Scan Output

默认值

帮助

☒ 配置SOUT模块

电路参考

1: 并出, 573锁存

3: 串出, 573锁存

2: 串出, 不锁存

4: 串出, 595锁存

基本设定

映射地址

20

M区间偏移

数据线数量

8

片选线数量

3

☐ 数据取反后发送

发送数据

☐ 并行方式

☒ 串行方式

发送数据

☒ CLK的上升沿

☐ CLK的下降沿

刷新数据

☒ 片选正脉冲

☐ 片选负脉冲

管脚选择

地址	管脚选择
Qn	
DS	NA/P4. 4
CP	ALE/P4. 5
CS	
CS0	P2. 7/A15
CS1	P2. 6/A14
CS2	P2. 5/A13

NA/P4. 4: [SOUT. DS]

系统块设置参数必须下载才能生效

- **配置 SOUT 模块：**选择表示使能 SOUT 模块，不选择表示禁用 SOUT 模块。
- **映射地址：**SOUT 驱动数据的存放地址。例如上面设置的 M 区间偏移为 20，那么第 1 组数据为 MB20 的 8 个位，第 2 组数据为 MB21 的 8 个位，依此类推。
- **数据线数量：**在分组输出数据时，每组数据位的个数。在并行输出模式下，即 Qn 输出管脚的个数。在串行输出模式下，即使每组数据位不足 8 位，SOUT 模块在串行输出时，依然会补足 8 位作为一个字节发送。
- **片选线数量：**在分组输出数据时，数据组的数量，即 CSn 驱动管脚个数。
- **数据取反后发送：**根据外部电路的不同，有时候实际的外部动作和数据值的对应关系是相反的，即负逻辑，为了符合使用习惯，可以在这里选择数据取反后发送。
- **并行还是串行方式发送数据：**每组数据位的发送方式。
- **正脉冲还是负脉冲刷新数据：**每组数据位锁存器件更新数据的方式（绝大多数逻辑 IC 为正脉冲更新锁存数据）。

特别说明

带锁存的逻辑 IC 一般都有输出允许管脚，即所谓的 OE 管脚。OE 管脚为高时，所有的输出都是高阻跟随状态，没有任何驱动能力。高电平、低电平、高阻这三种状态构成了严格意义上的三态门输出。PLC 上电复位后，锁存器件的实际锁存值在第一次更新前是无意义的。用这些值（甚至很有可能是随机值）去驱动外部电器将是很危险的。为了让 PLC 在上电复位后，第一次更新锁存值之前，输出器件就有一个确定的输出，则需要配置 MISC 模块中的

OE 管脚。OE 管脚在 PLC 上电后就保持高电平，直到第一次 PLC 主循环扫描结束，所有的锁存值已经被更新之后，才被 PLC 系统拉低。OE 被拉低后，锁存逻辑 IC 才真正具备驱动能力。

使用实例

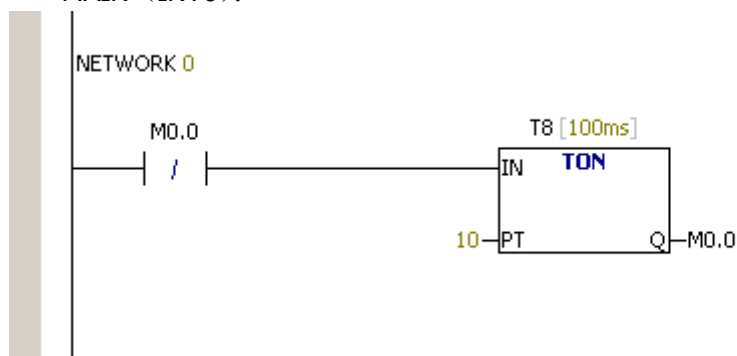
试验板 EC30-EK51-EVAL 有两路输出扫描电路。这里我们使用 1 片 74HC164 加 3 片 74HC573 组成的输出扫描电路（3×8）。电路图和上面给出的“串出 573 锁存”模式完全一致。根据单片机的管脚配置，得到对应的软件配置如下：

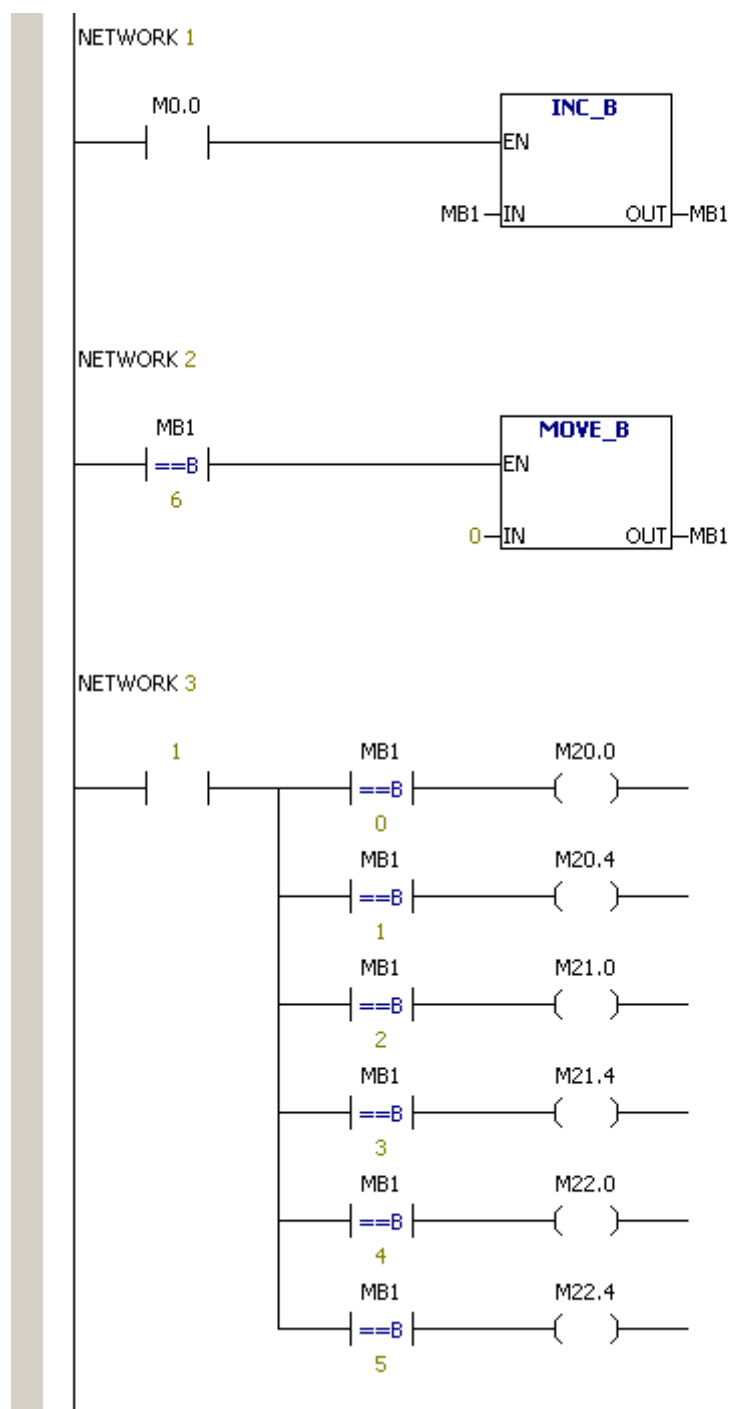
⚠ 系统块设置参数必须下载才能生效

配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

程序文件名：**14-EK51-SOUT (EVAL).vcw**。

MAIN (INT0):





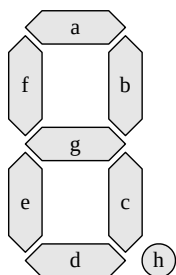
程序通过定时器 T8 得到一个每秒一次的脉冲信号，在这个脉冲信号中将 MB1 自加 1，并且每当自加到 6 时清 0。然后根据 MB1 的值分别驱动 M20.0、M20.4、...M22.4。将程序下载到实验板，就能够看到 M20.0、M20.4、...M22.4 这 6 个 LED 交替闪烁。

段码扫描 [SLED]

简介

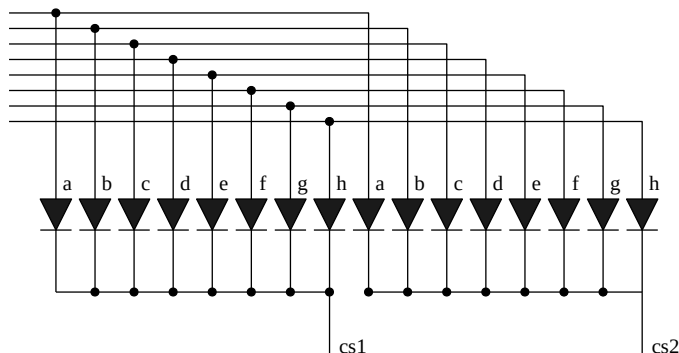
对于需要显示数字的场合，使用 LED（Light Emitting Diode，发光二极管）段码是一种

较为经济的方式。1 位数字的 LED 段码看上去是下面的样子：



其中 **a ~ h** 分别代表一个发光二极管，需要显示一个数字，驱动对应的发光二极管发光即可。一般说来，LED 段码的这些发光二极管在内部有两种连接方式：共阴或者共阳。如果是共阴的话，**a ~ h** 这 8 个发光二极管的阴极是连接在一起的，加上它们各自的阳极，封装引出的管脚总共是 9 个。如果是共阳的话，**a ~ h** 这 8 个发光二极管的阳极是连接在一起的，加上它们各自的阴极，封装引出的管脚总共也是 9 个。

由于多数场合需要显示不止一个数字，LED 段码也有多数字组合的形式。为了节省管脚资源，多数字的 LED 段码，不同数字的相同段位，驱动端也是连接在一起的，例如两数字共阴 LED 段码内部的电路图可能是下面的形式：

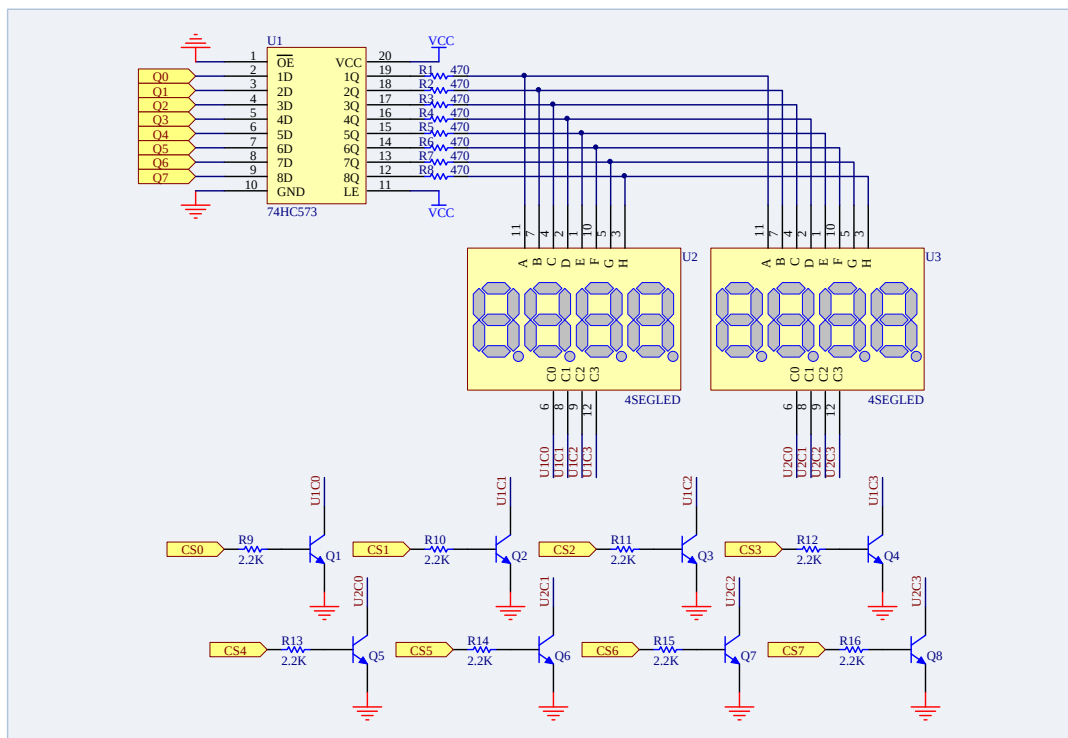


如果 2 个数字不一样，就需要分时来驱动了。先根据第 1 个数字的值确定 **a ~ h** 的电平，同时使 **cs1** 有效，那么第 1 个数字就得到了显示。然后根据第 2 个数字的值确定 **a ~ h** 的电平，同时使 **cs2** 有效，那么第 2 个数字就得到了显示。来回显示数字 1 和数字 2，只要交替的速度足够快，在人的眼睛看来，这两个数字就是同时显示的了。

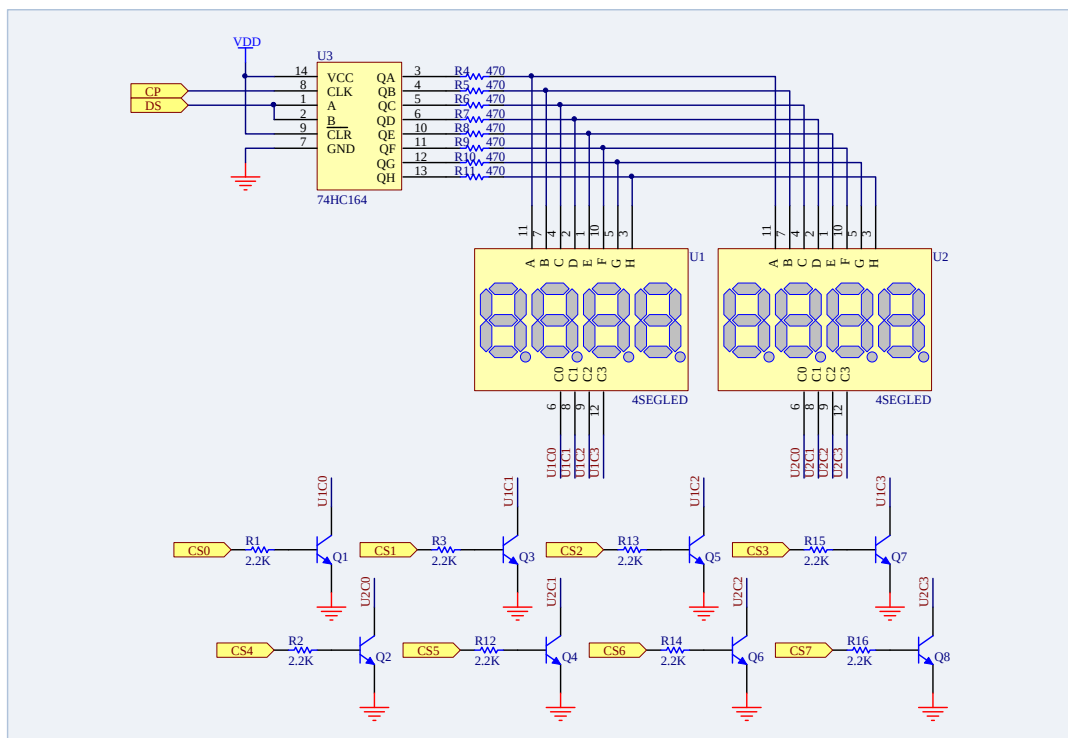
外部电路

使用 EC30-EK51 自带的 LED 段码驱动 **SLED** 模块，推荐下面两种外部电路图：

并行驱动 Q0 ~ Q7



串行驱动 Q0 ~ Q7

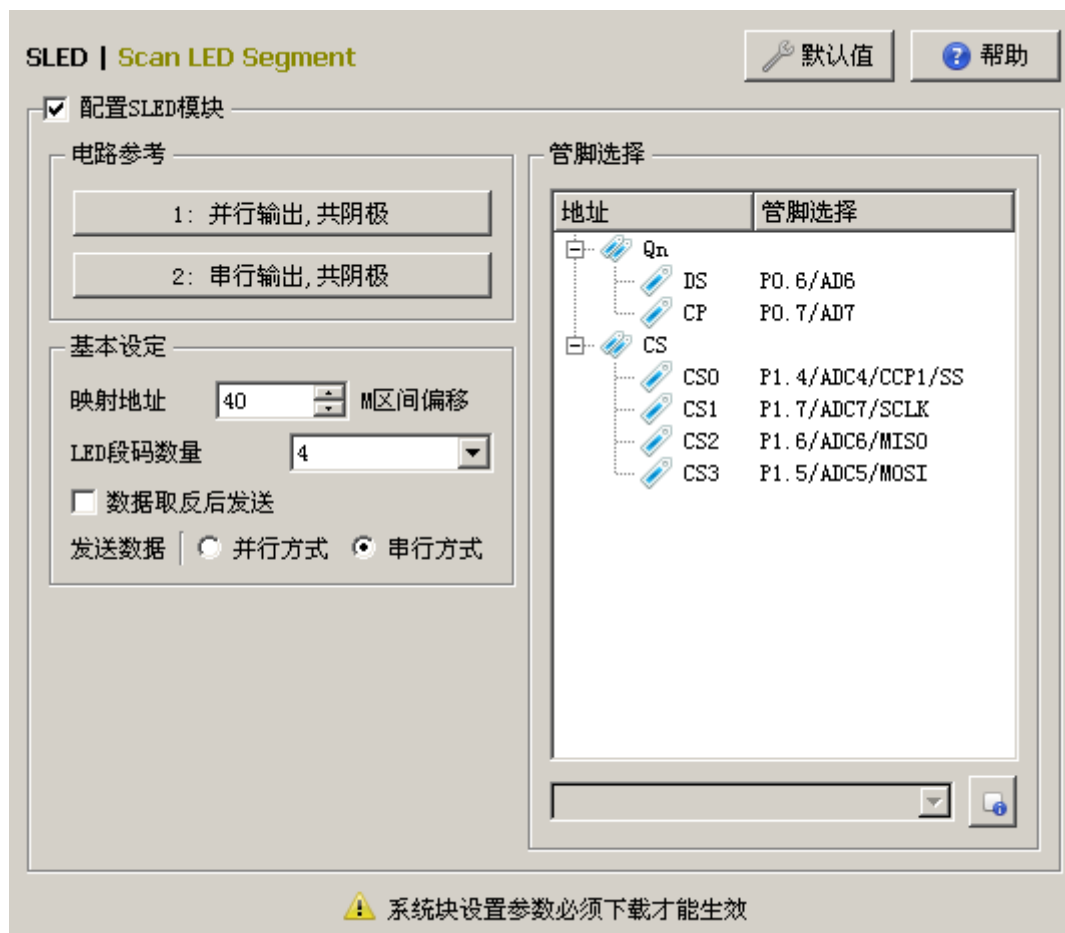


这两种电路都使用了两个 4 数字的共阴 LED 段码，组成了一共 8 个数字，这也是目前 SLED 模块驱动数字数量的极限。实际使用的时候，可以根据需要调整这个数量。在单片机

管脚资源比较紧张的时候，推荐使用 74HC164 来串行驱动 Qn（串行驱动）。

软件配置

在 GUTTA System Utility 中，SLED 模块的配置窗口如下：



- **配置 SLED 模块：**选择表示使能 SLED 模块，不选择表示禁用 SLED 模块。
- **映射地址：**LED 段码显示的内容。例如上面设置的 M 区间偏移为 40，那么第 1 个数码管的显示内容由 MB40 确定，第 2 个数码管的显示内容由 MB41 确定，依此类推。
- **LED 段码数量：**需要驱动的 LED 数字的数量。有几个 LED 数字，就有几个片选管脚(CSn)。
- **数据取反后发送：**选择表示 LED 数字的每个段都是低电平驱动。否则为高电平驱动。
- **并行还是串行方式发送数据：**每个段位 (Qn) 驱动的方式。
- **管脚选择：**根据前面的配置，需要使用的功能管脚将会在这里列出，点击选中功能管脚，便可以在下面的下拉框中为这个功能管脚确定一个具体的单片机管脚。

特别说明

EC30-EK51 会在系统的 1ms 定时中断中处理 LED 段码的扫描。也就是说每过 1ms，将会刷新一个 LED 段码的数字。如果数字数量为 8 个（最大值），那么就是每 8ms 完成所有的刷新。这就意味着 1 秒钟可以刷新 125 次，这个数值远远高出了人眼的分辨能力。

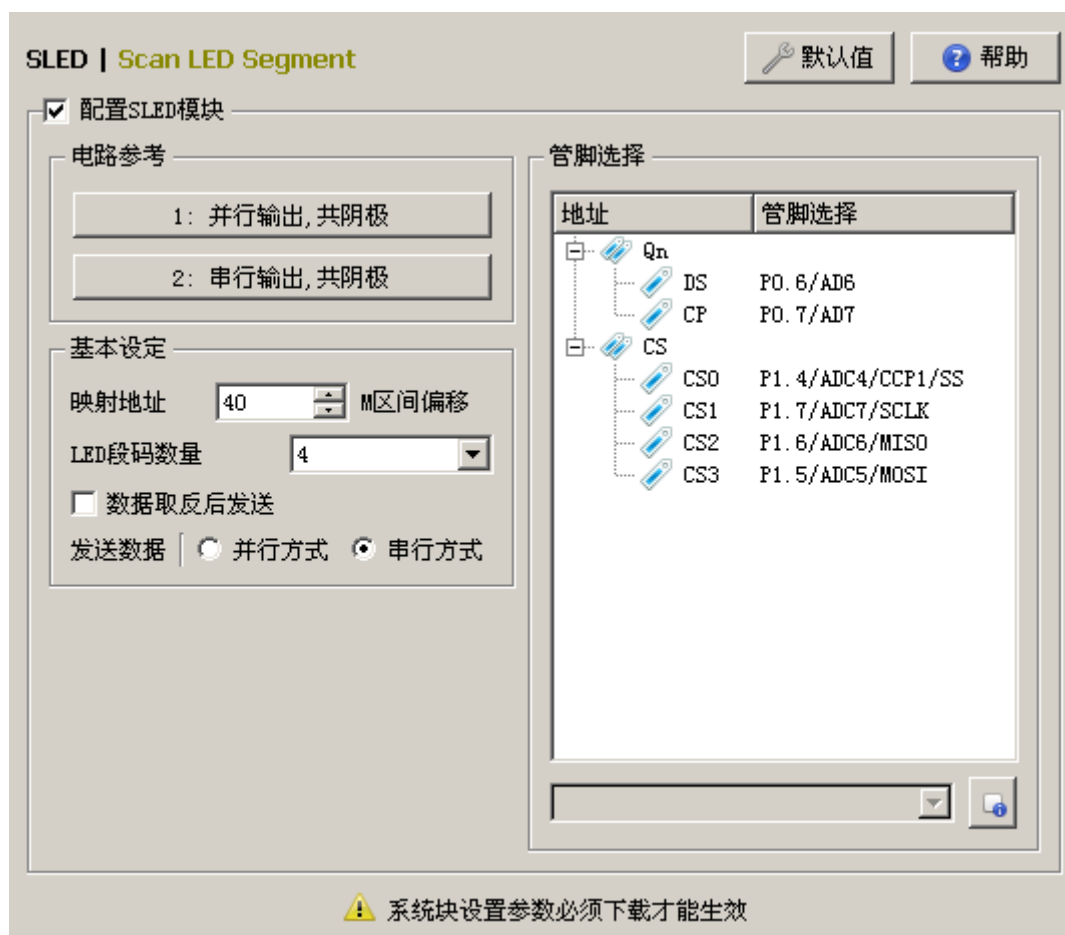
一旦 SLED 被使能，相应的管脚配置如下：

1. 串行发送数据：DS、CP 配置为推挽输出。
2. 并行发送数据：Qn 配置为推挽输出。
3. 片选：CSn 配置为推挽输出。

虽然 Qn 被配置为推挽输出且具有一定的驱动能力，但是不建议直接使用单片机的管脚来驱动 LED 段码。特别是在数字比较多的情况下。由于 LED 段码的几个数字是分时驱动的，如果一共有 8 个数字，那么每个段码的驱动时间实际上只有一个周期的 1/8，为了达到一定的亮度，往往需要加大驱动电流，而这样的电流可能会让单片机比较难以承受。因此推荐使用逻辑门电路来驱动（例如电路 1 中使用的 74HC573）。

使用实例

可以在试验板 CPU-EC20 上测试本模块。通过《WS2102 CPU-EC20 (8051) PLC 仿真学习套件用户手册》我们知道：CPU-EC20 使用的是串行驱动方式；数码管数字为 4。对应的软件配置如下：

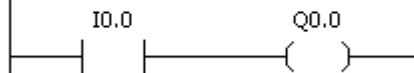


配置部分完成后，为了测试驱动，写一小段梯形图程序如下：

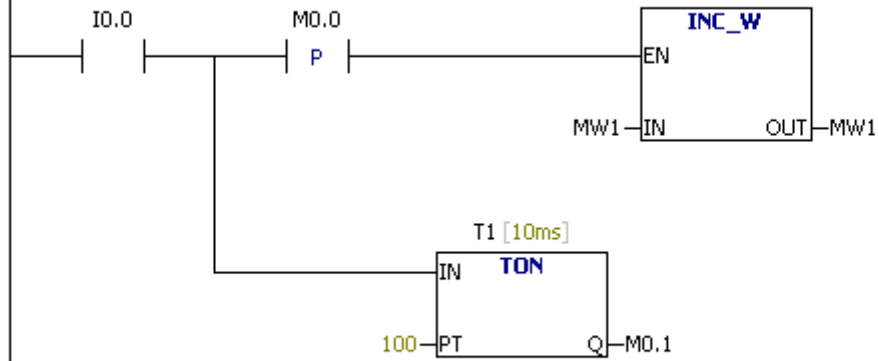
程序文件名：**15-EK51-SLED.vcw**。

MAIN (INT0):

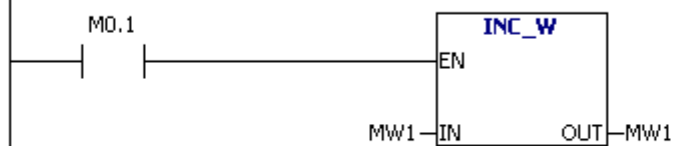
NETWORK 0



NETWORK 1



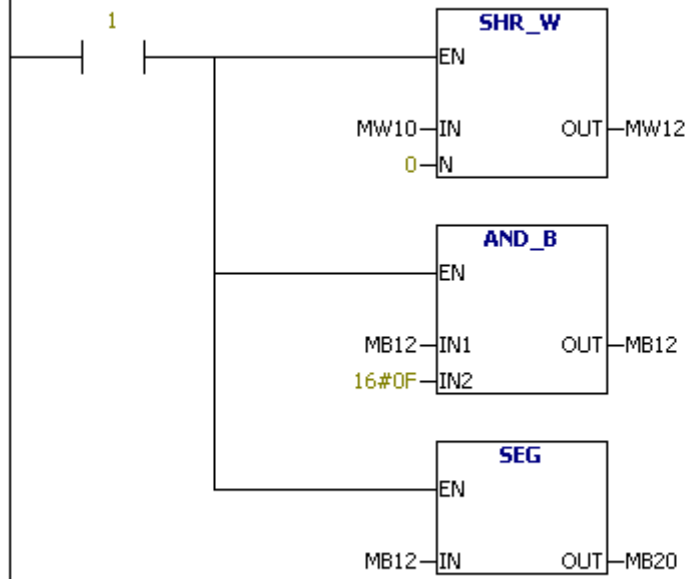
NETWORK 2



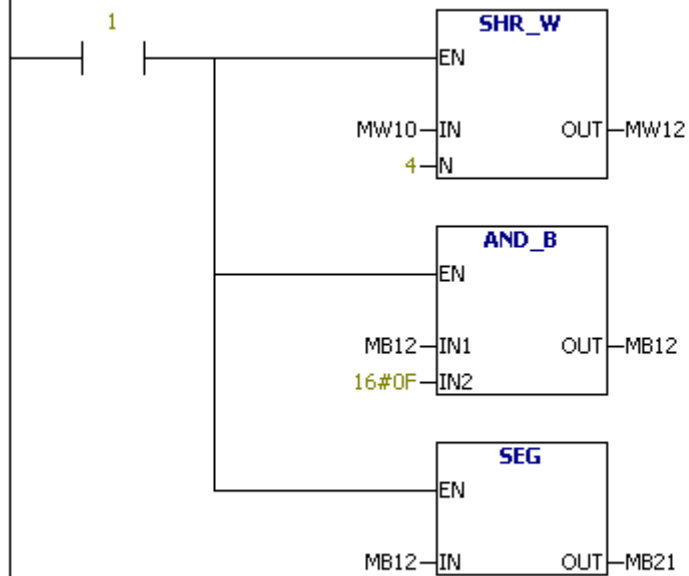
NETWORK 3



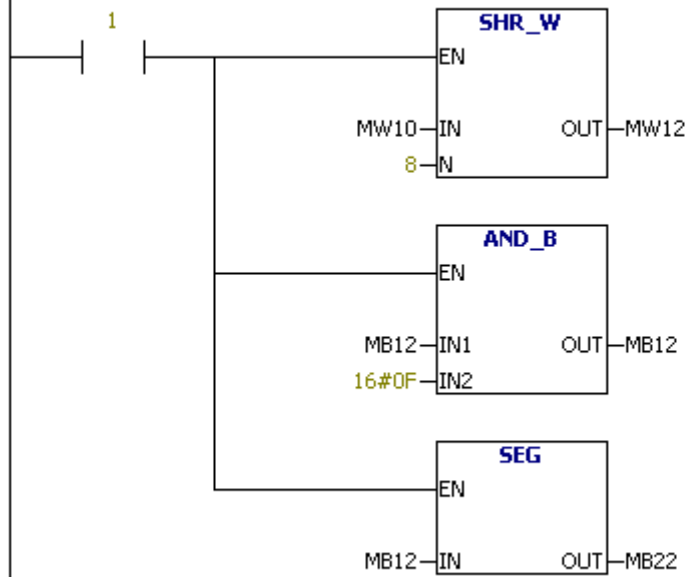
NETWORK 4



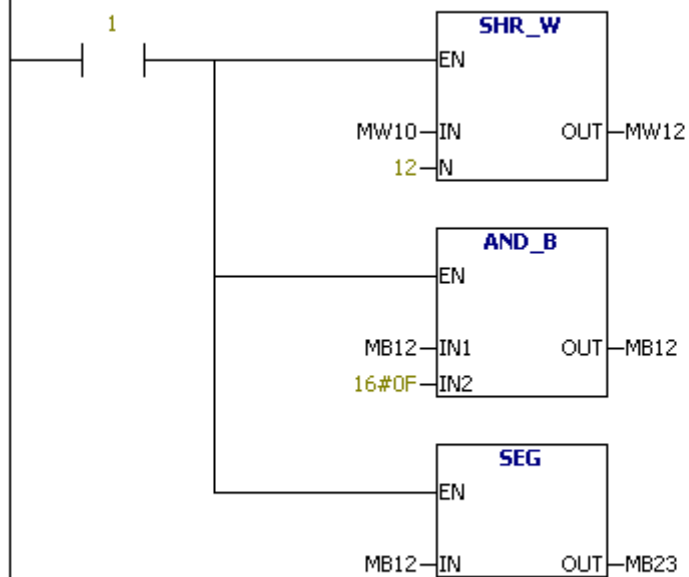
NETWORK 5

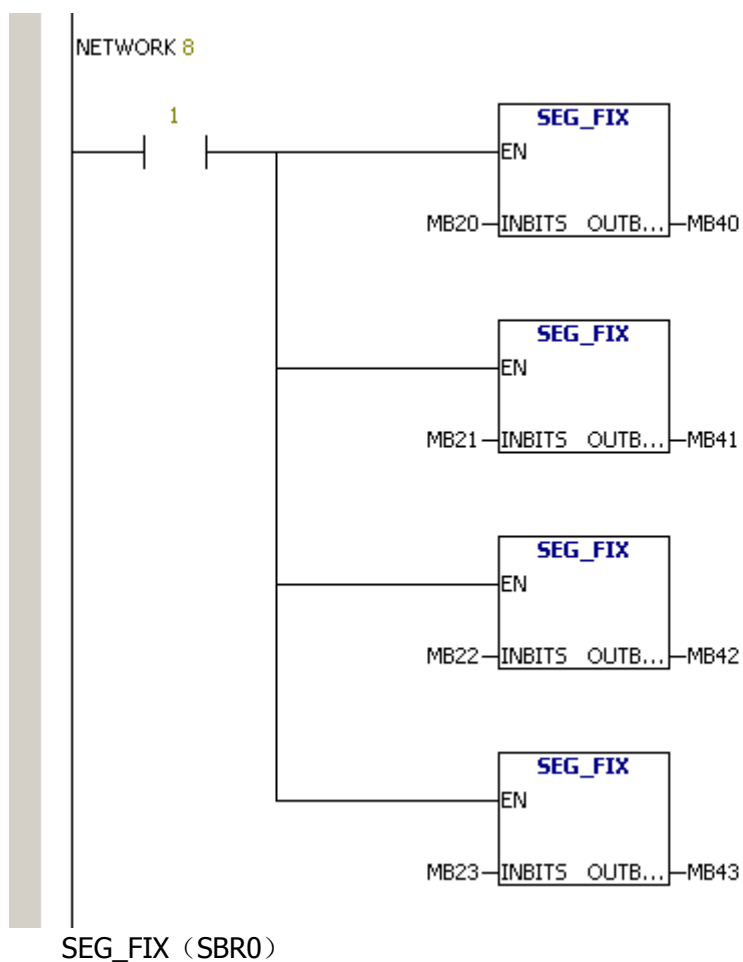


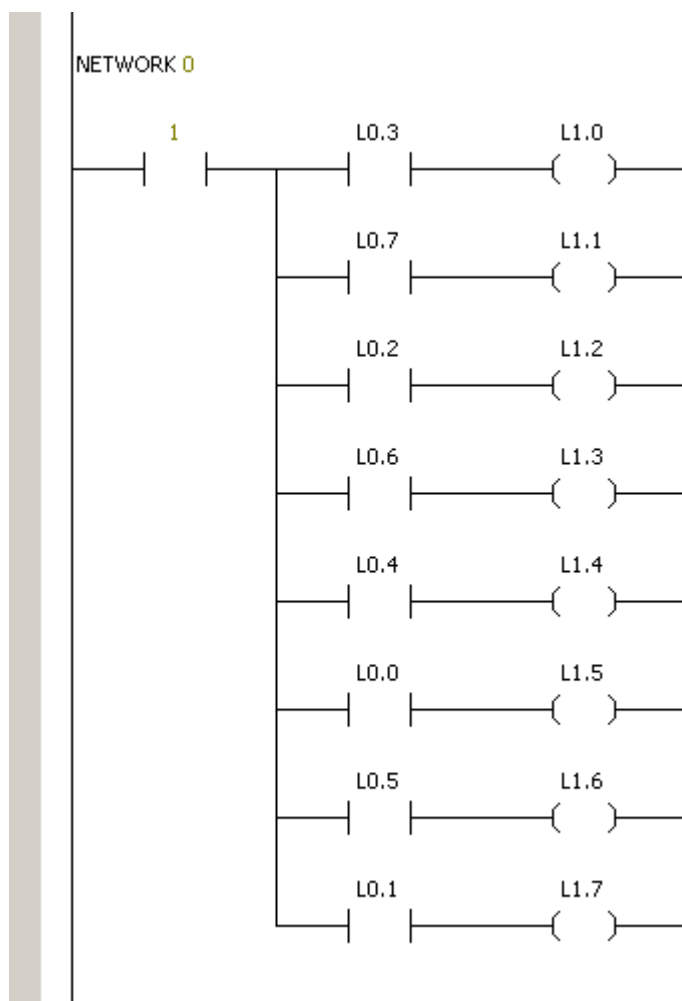
NETWORK 6



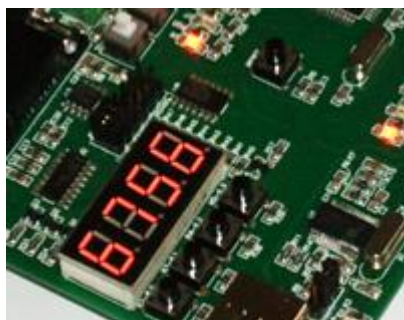
NETWORK 7







由于程序中使用了 I0.0，在 GUTTA System Utility 中，我们还需要配置 IO 模块。配置的具体办法参照 IO 模块的相关章节。由程序我们知道，只要按下 I0.0，MW1 变量就加 1。若持续按下 I0.0 超过 1 秒（长按），则每个扫描周期 MW1 变量都加 1。同时，我们将 MW1 的 BCD 码转换到 MW10 中，并将 MW10 的 4 个 BCD 码通过 SEG 指令分别写入 MB20 ~ MB23。在 EC30-EK51-EVAL 的具体电路中，由于驱动芯片 74HC164 出来不是按照 a、b、c、...、h 的顺序连接 LED 数码管的，故这里用一个子程序 SEG_FIX 来做位映射，并将最后结果保存到 MB40 ~ MB43。由 SLED 模块的配置我们知道，LED 的显示的内容，就是 MB40 ~ MB43 的值。最终，MW1 这个数据就被显示在开发板上了。



驱动 [AT24C08B]

AT24C08B 是 ATMEL 生产的 EEPROM 存储芯片，采用 I2C 总线与单片机相连接。

EC30-EK51 的 AT24C08B 模块封装了 I2C 通讯的具体细节。硬件上只要确认 AT24C08B 和单片机 IAP12C5A60AD 进行了正确的电器连接,配置好 AT24C08B 模块之后,就可以在 PLC 程序中使用 ERB、EWB 指令保存或提取 EEPROM 数据。由于 AT24C08B 采用 I2C 通讯协议,配置 AT24C08B 之前,必须事先配置好一个 I2C 模块,由于这条 I2C 总线上可能还有其他 I2C 设备,故 AT24C08B 只是共享使用 I2C 模块,以确保不和其他设备冲突。

ERB、EWB 指令的详细说明请参考指令手册。

在 GUTTA System Utility 中,AT24C08B 模块的配置窗口如下:

AT24C08B | Two-wire Serial EEPROM

☒ 配置AT24C08B模块

AT24C08B模块由指令ERB和EWB来操作

使用I2C模块: I2C0: [I2C0] <AT24C08B>

地址配置

- 若为AT24C08B, 电路中芯片的A0和A1不用连接, 这里必须设置为接地。A2根据电路的实际情况设置。
- 若为AT24C04B, 电路中芯片的A0不用连接, 这里必须设置为接地。A1和A2根据电路的实际情况设置。
- 若为AT24C02B/AT24C01B, A0、A1、A2根据电路的实际情况设置。

连接A0到: GND

连接A1到: GND

连接A2到: VCC

I2C地址为16#A8

⚠ 系统块设置参数必须下载才能生效

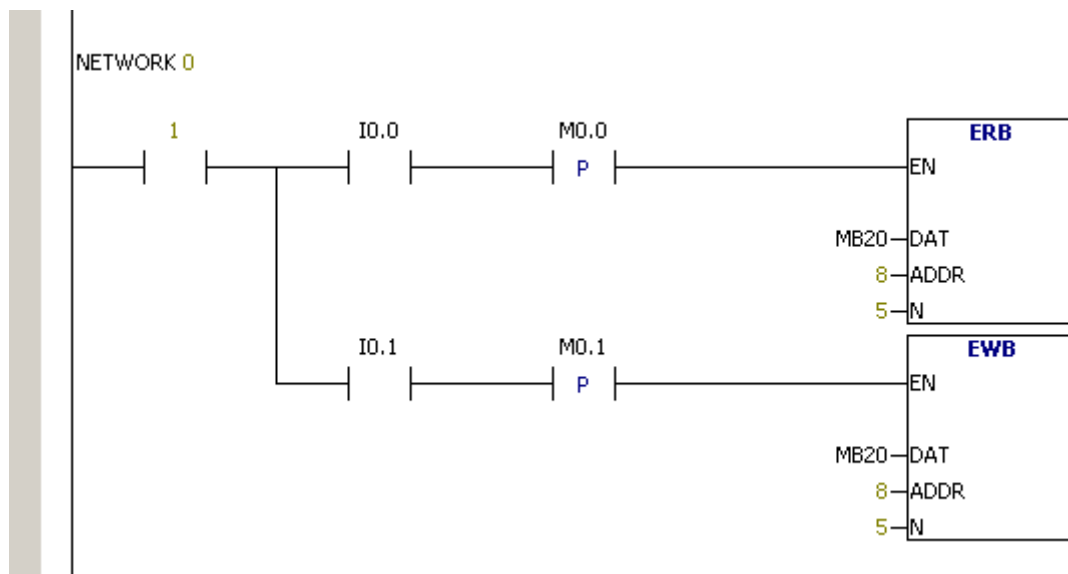
- 配置 AT24C08B 模块: 选择表示使能 AT24C08B 模块, 不选择表示禁用 AT24C08B 模块。
- 连接 A0/A1/A2 到: 这里决定了 AT24C08B 使用的地址。详细阅读 AT24C08B 的芯片数据手册我们就知道, 芯片根据 A0/A1/A2 电平来决定自己的实际 I2C 通讯地址。同时, AT24C01B ~ AT24C08B 这几种芯片由于容量不一样, 对 A0/A1/A2 的使用也不一样。

通过前面关于 I2C 模块的介绍我们知道, 在试验板 CPU-EC20 (8051) 上可以测试 I2C 通讯, 给出的例子虽然是与板上的 PCF8563 通讯, 但是由于板上 PCF8563 和 AT24C04B 使用同一条 I2C, 我们可以照搬例子中的 I2C 模块的设置, 并在 AT24C08B 模块中使能, 选用这条总线后, 就能够使用 ERB、EWB 指令读取或保存 EEPROM 数据了。由于 AT24C04B 的默认地址是 16#A0, PCF8563 的默认地址是 16#A2。AT24C04B 有 4Kbit 可以被访问, 也就是 512 个字节, 如果需要访问高 256 字节的数据, 使用的 I2C 通讯地址也是 16#A2, 这就与 PCF8563 的地址发生了冲突。为了避免这个冲突, 在电路上, 我们将 AT24C04B 的 A2 拉高

到 VCC，让 AT24C04B 的地址从 16#A8 开始。

程序文件名：16-EK51-AT24C08B.vcw。

MAIN (INT0):



程序非常简单，若捕捉到 IO.0 的上升沿转换，就将 MB20 ~ MB24 存入 EEPROM 芯片，芯片内部地址为 8；若捕捉到 IO.1 的上升沿转换，就将 EEPROM 芯片从地址 8 开始，恢复到 MB20 ~ MB24 这 5 个字节中。I2C 的配置完全照抄 I2C 模块说明给出的例子，然后在 AT24C08B 模块中使能，共享使用已经配置好的 I2C 模块，将 A2 设置位为上拉到 VCC 即可。

2010/9/20 日之后的固件对 ERB 和 EWB 指令做了优化。ERB 和 EWB 指令的 EEPROM 地址 ADDR 和数据数量 N 不再受 AT24C08B 分页机制的限制。如果进行了跨页操作，ERB 和 EWB 指令会自动发起多次 I2C 通讯以完成任务。与之同时，EWB 指令通过 I2C 写数据之前会先通过 I2C 读取数据，如果发现数据一致，将自动忽略写操作。

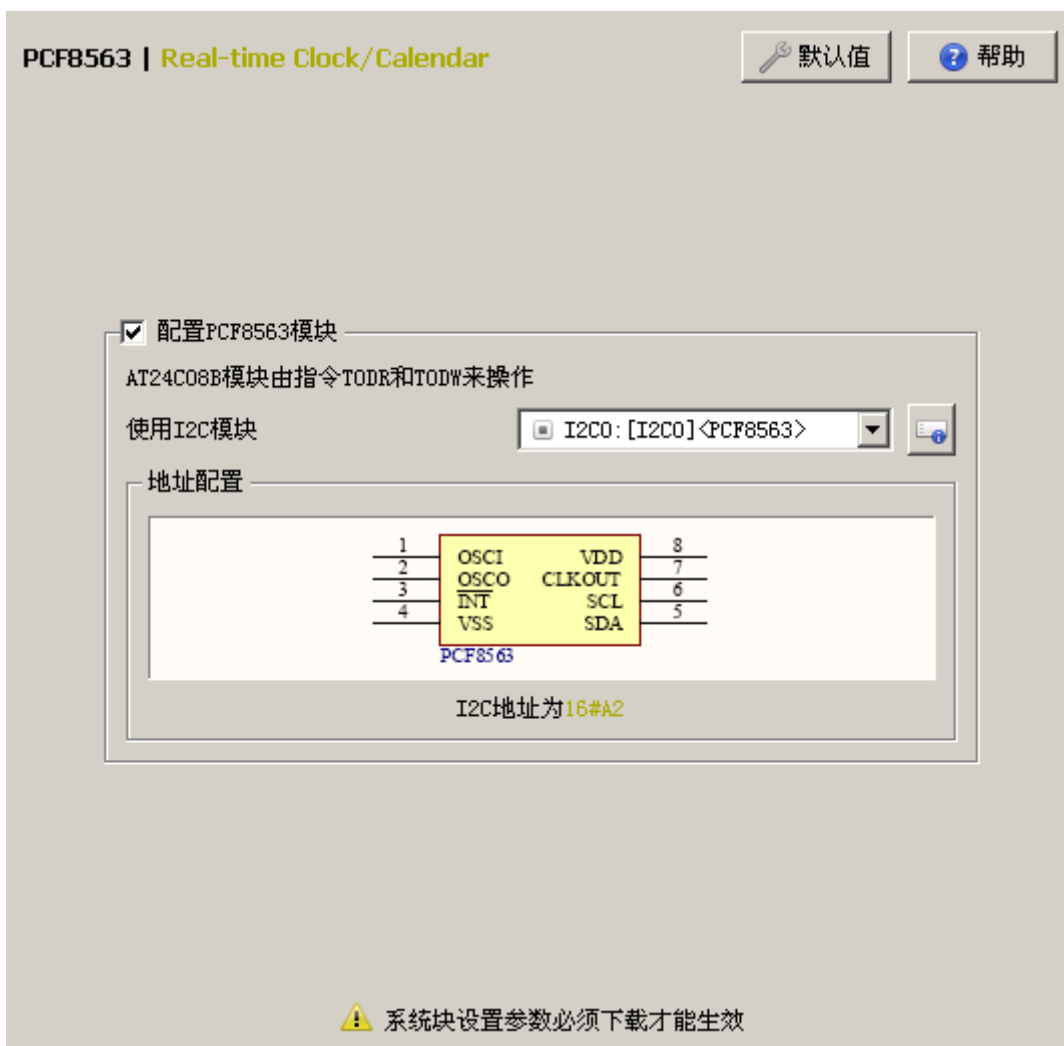
通过以上改进，EC30-EK51 可以采用很简单的机制保存用户参数。可以设定某段 M 区域为用户参数区。在 PLC 上电后，利用 SMB0.1 调用 ERB 指令，将 EEPROM 中的用户参数恢复到 M 区域。同时，可以周期性的调用 EWB 指令，PLC 系统一旦发现用户参数发生了改变（例如触摸屏操作了这些数据），便发起通讯同步更新 EEPROM 中对应的数据。

驱动 [PCF8563]

PCF8563 是 NXP 生产的时钟芯片，采用 I2C 总线与单片机相连接。EC30-EK51 的 PCF8563 模块封装了 I2C 通讯的具体细节和格式的转换。硬件上只要确认 PCF8563 和单片机 IAP12C5A60AD 进行了正确的电器连接，配置好 PCF8563 模块之后，就可以在 PLC 程序中使用 TODR、TODW 指令读取或设置当前时钟。由于 PCF8563 采用 I2C 通讯协议，配置 PCF8563 模块之前，必须事先配置好一个 I2C 模块，由于这条 I2C 总线上可能还有其他 I2C 设备，故 PCF8563 只是共享使用 I2C 模块，以确保不和其他设备冲突。

TODR、TODW 指令的详细说明请参考指令手册。

在 GUTTA System Utility 中，PCF8563 模块的配置窗口如下：



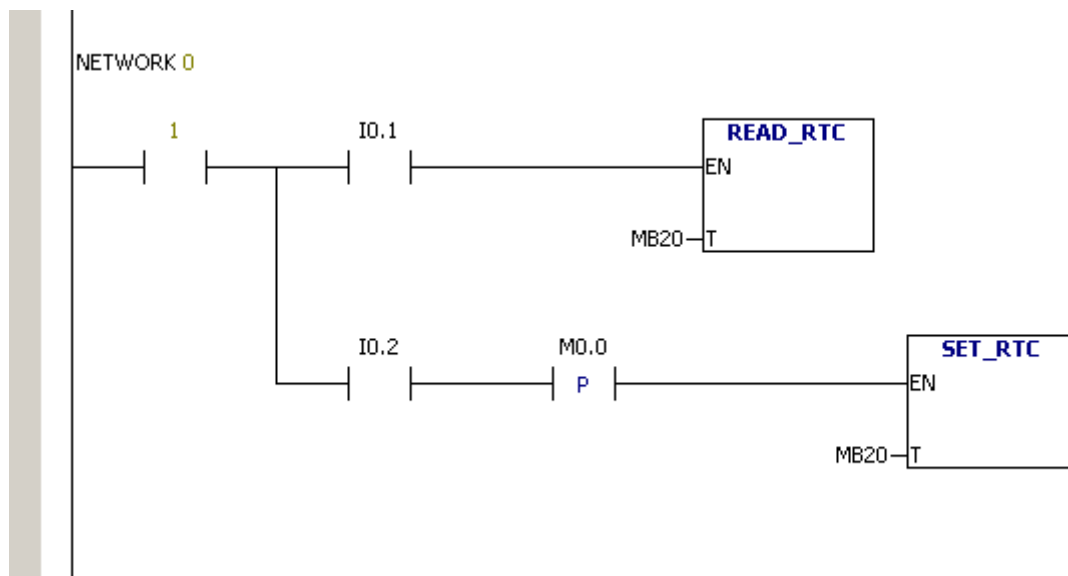
● 配置 PCF8563 模块：选择表示使能 PCF8563 模块，不选择表示禁用 PCF8563 模块。

● 使用 I2C 模块：PCF7563 使用哪条 I2C 总线与 PCF8563 通讯。

通过前面关于 I2C 模块的介绍我们知道，在试验板 CPU-EC20 (8051)上可以测试 I2C 通讯，并且给出的例子就是与板上的 PCF8563 通讯。只不过例子中采用最基本的 I2CRD 和 I2CWR 指令在 I2C 总线上发送或接收数据。由于 PCF8563 模块包装了这些通讯细节，有了 PCF8563 模块，我们现在可以直接通过 TODR、TODW 指令来读写时钟了。

程序文件名：**17-EK51-PCF8563.vcw**。

MAIN (INT0):



程序非常简单，若 IO.1 接通，就不停的读取时间信息，放到 MB20 开始的 7 个字节中。如果 IO.2 接通，在上升沿将 MB20 开始的 7 个字节作为时间信息，写入 PCF8563。至于系统块配置部分，I2C 的配置完全照抄 I2C 模块说明给出的例子，然后在 PCF8563 模块中使用，共享使用已经配置好的 I2C 模块即可。

EBUS 通讯 [EBUS]

EBUS 是 Efficient Serial Bus for PLC Module 的简称。通过 EBUS，您可以将最多 7 个 PLC 扩展模块连接到使用 EC30-EK51 的 PLC 本体上。EBUS 模块自动识别有多少个模块连接到了总线上，判断其连接顺序后给每个模块分配输入/输出变量（I、Q、AI、AQ）。EBUS 不需要特别的初始化流程，即使在 PLC 运行中调整了 PLC 扩展模块的位置，EBUS 能够实时重新分配输入/输出变量，而不需要复位 PLC。

除了运行中实时的模块识别排序，变量分配，在 EBUS 上通讯的数据都有严格的数据校验，这将使干扰造成 I/O 误动作的可能性降到了最低。如果有强干扰造成 EBUS 通讯数据不正确，PLC 可能在得到输入或者是刷新输出的速度上有所延时，但几乎不可能误动作。

我们会陆续推出支持 EBUS 协议的标准扩展模块，包括数字量输入输出扩展模块，4~20mA/0~5V 标准工业信号模拟量扩展模块，PT100 热电阻扩展模块等。详情请关注我们网站的 EC30-EK51 板式 PLC 的详细介绍。

在 GUTTA System Utility 中，EBUS 模块的配置窗口如下：

EBUS | Efficient Serial Bus for PLC Module

默认值 帮助

配置EBUS模块

起始地址

离散量输入 (IBx) 2
离散量输出 (QBx) 1
模拟量输入 (AIBx) 4
模拟量输出 (AQBx) 0

模块地址线

地址线管脚 (A2) P1.4/ADC4/CCP1/SS: [E
地址线管脚 (A1) P1.3/ADC3/CCP0: [EBUS
地址线管脚 (A0) P1.2/ADC2/ECI: [EBUS

模块选择

使用SPI模块

SPI: [EBUS]

管脚选择

将SPI从P1口切换到P4口

MOSI P1.5/ADC5/MOSI: [EBUS.MOSI
MISO P1.6/ADC6/MISO: [EBUS.MISO
SCLK P1.7/ADC7/SCLK: [EBUS.SCLK

系统块设置参数必须下载才能生效

- **配置 EBUS 模块：**选择表示使能 EBUS 模块，不选择表示禁用 EBUS 模块。
- **起始地址：**给模块分配 I/O 变量的开始地址。I/O 变量的偏移单位是字节。例如 EC30-EK51-D20 板式 PLC 一共有 12 路数字量输入，本体的这些输入分配为 I0.0 ~ I0.7、I1.0 ~ I1.3。由于 IB1 已经被本体使用，即使有未使用的位，也无法分配给扩展模块，扩展模块的数字量输入必须从 IB2 开始。
- **模块地址线：**需要给 EBUS 分配 3 根地址线，用于 PLC 扩展模块的选择。
- **模块选择：**EBUS 的数据通讯采用 SPI 协议（2MHz 左右的通讯速度），这里必须选择硬件 SPI。
- **管脚选择：**可以重映射 SPI 管脚，以方便硬件电路的设计。需要注意的是，SPI 模块不自动操作 NSS 通讯脚，由前面的 A2 ~ A0 三根地址线来完成 PLC 扩展模块的选择。

有关 EBUS 的详细使用请参考 EC30-EK51-D20 板式 PLC 的相关说明。