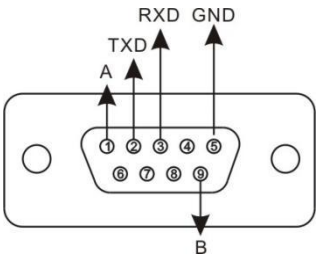


D2008 与 D12 串口通讯协议说明

第一节、仪表与电脑的连接使用



仪表 9 芯 PC 接口	电脑 9 芯串口
2	2
3	3
5	5

仪表的 PC 接口定义

仪表与电脑的 RS232 连接

第二节、设置

步 骤	操 作	显 示	解 释
1	按【设置】	PSt 00	
	按【1】	PSt 01	输入参数/功能类别号“01”
	按【输入】	C-CodE	
	按【888888】	C-----	输入标定密码，出厂初始密码为“888888”
2	按【输入】	Adr ***	显示当前通讯地址（指令方式时才有效）
3	按【1】	Adr 001	通讯地址（1~255）例如选 001
4	按【输入】	bt 600	串行通讯的波特率
5	按【←或→】	bt 9600	可选波特率 600、1200、2400、4800、9600 和 19200bps。例如选 9600
6	按【输入】	tF *	显示当前通讯方式
7	按【0】	tF 0	通讯方式选择。例如选 0 具体方式见附录 A， 21 为无线调秤模式;97 为 KL-MPLC 模式
8	按【输入】	Jn ****	当前校验方式
9	按【←或→】	Jn nonE	校验方式选择，选择项如下:例如选 nonE nonE: 无校验; odd 奇校验; EvEn: 偶校验; mArk: 标志; SPAcE: 空格;
10	按【输入】	bit *	数据位长度
11	按【←或→】	bit 8	可选 7 位和 8 位（默认）
12	按【输入】	CyC *	连续发送方式的时间间隔
13	按【←或→】	CyC 200	可选: 50、100, 200（默认）、250、300、400、500、600、1000ms
14	按【输入】	称重状态	

第三节、协议

(1). 连续方式(TF=0): 重量数据为毛重 (同 D39 连续方式 1)

连续方式(TF=17): 重量数据为净重

所传送的数据为仪表的当前重量。每帧数据由 12 组数据组成 。格式如下表所示:

第 X 字节	内容	注 解		举 例 (发送 +20.00)	
		内容	代码	内容	十六进制代码
1	开始	(XON)	02	XON	02
2	+或-	符号位	2B/2D	+	2B
3	称量数据	最高位	30~39	0	30
4			30~39	0	30
5			30~39	2	32
6			30~39	0	30
7			30~39	0	30
8		最低位	30~39	0	30
9	小数点位数	从右到左 (0~4)	30~34	2	32
10	异或校验	高四位		异或校验	31
11		低四位		=0x1B	'B'
12	结束	XOFF	03	XOFF	03

异或=2⊕3⊕……8⊕9。

注 1: 异或校验高、低 4 位的确定: 异或和高、低 4 位如果小于、等于 9, 则加上 30h, 成为 ASCII 码数字发送, 例如: 异或校验高 4 位为 6, 加 30h 后, 为 36h 即 ASCII 码的 6 发送; 异或和高、低 4 位如果大于 9, 则加上 37h, 成为 ASCII 码字母发送, 例如: 异或校验高 4 位为 B, 加 37h 后, 为 42h 即 ASCII 码的 B 发送。

(2). 连续方式(TF=2): (兼容 D2+) 重量数据为毛重 (同 D39 连续方式 2)

连续方式(TF=18): (兼容 D2+) 重量数据为净重

所有数据均为 ASCII 码, 每帧数据共有 8 字节组成 (包括小数点), 数据传送先低位后高位, 每帧数据间有一组是分隔符 “=”, 发送数据为当前称重值, 如当前称重重量为 188.5, 连续发送 5.88100=5.88100=……。如当前显示称重重量为-1885, 连续发送.58810=-.58810=……。

(3). 连续方式(TF=3): 重量数据为毛重 (同 D39 连续方式 1)

连续方式(TF=19): 重量数据为净重

所有数据均为 ASCII 码, 每帧数据共有 9 字节组成 (包括小数点), 数据传送先低位后高位, 每帧数据间有一组是分隔符 “=”, 发送数据为当前称重值, 如当前称重重量为 188.5, 连续发送 5.88100=5.881000=……。如当前显示称重重量为-1885, 连续发送.588100=-.588100=……。

(4). 连续方式(TF=4): (兼容托利多 T800) 不带校验和 , 重量数据前无效零, 用 0x30 表示 (同 D39 连续方式 4)

连续方式(TF=5): (兼容托利多 T800) 带校验和 , 重量数据前无效零, 用 0x30 表示 (同 D39 连续方式 5)

连续方式(TF=14): (兼容托利多 T800) 不带校验和, 重量数据前无效零, 用 0x20 表示

连续方式(TF=15): (兼容托利多 T800) 带校验和 , 重量数据前无效零, 用 0x20 表示

每字节数据由 10 位组成, 第 1 位为起始位, 第 10 位为停止位, 中间 8 位为数据位;连续输出每帧数据为 18 个字节。

连续输出格式 2																	
StX	A	B	C	X	X	X	X	X	X	X	X	X	X	X	X	CR	CKS
1	2			3				4				5		6			

其中 :

1. <StX> ASCII 起始符 (02H) 。
2. 状态字 A,B,C 。
3. 显示重量 , 可能是毛重也可能是净重 .6 位不带符号和小数点的数字 。
4. 皮重 , 6 位不带符号和小数点的数字 。
5. <CR> ASCII 回车符 (0DH) 。
6. <CKS>校验和。

状态字 A			
Bits 0 , 1 , 2			
0	1	2	小数点位置
0	0	0	KGKG00
1	0	0	KGKGX0
0	1	0	KGKGKG
1	1	0	KGKGX.X
0	0	1	KGKG.KG
1	0	1	KGX.KGX
0	1	1	KG.KGKG
1	1	1	X.KGKGX

Bits 3, 4		分 度 值 因 子
3	4	
1	0	X1
0	1	X2
1	1	X5
Bit 5		恒 为 1
Bit 6		恒 为 0

状 态 字 B	
Bits	功 能
Bit 0	毛 重 = 0, 净 重 = 1
Bit 1	符 号 : 正 = 0, 负 = 1
Bit 2	超 载 (或 小 于 零) = 1
Bit 3	动 态 = 1
Bit 4	单 位 : kg = 1
Bit 5	恒 为 1
Bit 6	仪 表 上 电 时 为 1

状 态 字 C	
Bit 0	恒 为 0
Bit 1	恒 为 0
Bit 2	恒 为 0
Bit 3	有 打 印 命 令 = 1
Bit 4	扩 展 显 示 (X10) = 1
Bit 5	恒 为 1
Bit 6	恒 为 0

(5). 连续方式(TF=6): (同 D39 连续方式 6)

每个字节数据位 8 位, 校验位可选, 停止位 1 位。

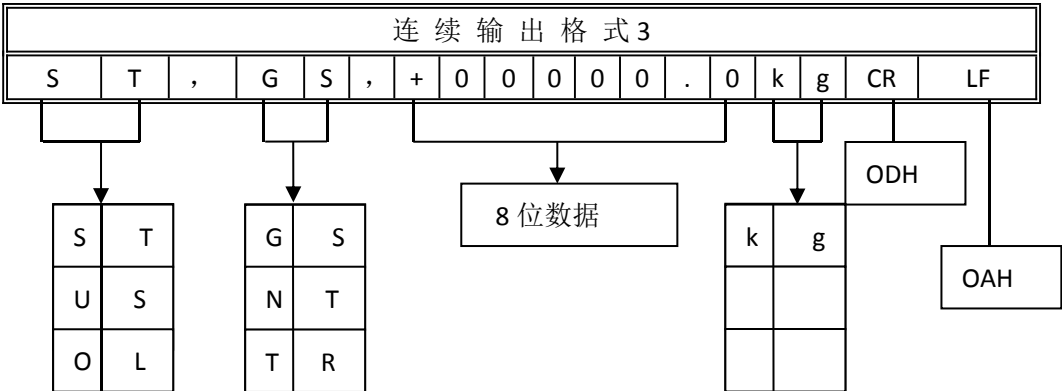
为了减少传输字节数, 将状态用一状态字节来描述, 数据压缩为三个字节 BCD 码, 一帧数据加一个标志字节 FF (HEX), 共五个字节构成。即: FF(HEX) 状态字 BCD1 BCD2 BCD3

其中状态字节定义如下:

D7	D6	D5	D4	D3	D2	D1	D0
溢出	稳定	符号	单位	毛/净	小数点位置		
1 溢出	1 不稳	1 负	1 吨	1 毛重	000	001	010
0 正常	0 稳定	0 正	0 千克	0 净重	X.	.X	.XX
							011
							100
							.XXX
							.XXXX

(6). 连续方式(TF=7): (同 D39 连续方式 7)

每字节数据由 10 位组成, 第 1 位为起始位, 第 10 位为停止位, 中间 8 位为数据位和校验位;



Header1 Header2
Header 1
ST 重量稳定 (Stable)
US 重量不稳定 (Unstable)
OL 超载 (Over Load)

Header 2
GS 毛重 (Gross data)

NT 净重 (Net data)

TR 扣重 (Tare data)

(7). 连续方式(TF=8): (同 D39 连续方式 8)

1、串行通信的数据格式如下

10 位: 1 位起始位, 8 位数据位, 1 位停止位。

2、串行通信发送的重量数据为 ASCII 码, 每次发送 12 个字节。

其定义如下:

第 1 个字节: 起始位 (02H)

第 2 个字节: 状态字 A

第 3 个字节: 状态字 B

第 4 个字节: 状态字 C

第 5 个字节: 6 位重量值的高位

第 10 个字节: 6 位重量值的低位

第 11 个字节: 回车 (0DH)

第 12 个字节: 换行 (0AH)

状态字 A

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0			

小数点:

	X	.X	.XX	.XXX	.XXXX	.XXXXX
D2 =	0	0	1	1	1	1
D1 =	0	1	0	0	1	1
D0 =	0	1	0	1	0	1

状态字 B

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1				0

D3 = 非动态为 0, 动态为 1。

D2 = 量程以内为 0, 超载为 1。

D1 = 重量值正为 0, 负为 1。

状态字 C = 20H

(8). 连续方式(TF=16):

(9).指令方式(TF=1):modbus RTU

具体协议详见如下:

仪表设置:pst 01-----中 TF = 1, D39 系列要选命令方式 2

一、老协议

兼容老协议

1.主机读仪表数据

等待	字节 0	1	2	3
四个字节的的时间	1~247	x03	起始地址高 8	低 8
	仪表地址	功能代码	所要读的寄存器开始地址	

4	5	6	7	等待
N/2 个字节高 8	低 8	CRC16 低 8	CRC16 高 8	三个字节的的时间
发送数量		CRC16 校验		

数据以 ASII 传送

2.仪表反馈

正常回应

字节 0	1
1~247	0x03
仪表地址	功能代码

2	3			3+n	3+n
N	数据 0		数据 N(偶数)	CRC16L	CRC16h
数据长度	数据				

1) 称重数据读取指令

	功能	所要读的寄存器开始地址 (地址高 8 位)	所要读的寄存器开始地址 (地址低 8 位)
1	读毛重	0x00	0x01
2	读皮重	0x00	0x02
3	读净重	0x00	0x03
4	读总内码	0x00	0x07
5	读第 i 号传感器内码 异常时得到-999999	0x00	0x0A+i(i=1~16)

发送

0	1	2	3	4
仪表地址	0x03	地址高 8 位	地址低 8 位	0x00

5	6	7
0x04	CRC16L	CRC16H

返回

0	1	2	3	5	6
仪表地址	0x03	0x08	符号-或 data6	Data5	Data4

7	8	9	10	11
Data3	Data2	Data1	Data0	小数点(右至左)

12	13
CRC16L	CRC16H

例子:

1、读取当前毛重

发送:01 03 00 01 00 04 15 C9

接收: 若当前毛重为 0

01 03 08 30 30 30 30 30 30 30 30 30 F8 2F (最后两个字节为 CRC16 校验码)

若当前毛重为 1240

01 03 08 30 30 30 31 32 34 30 30 85 96

2、远程置零

发送: 01 06 00 01 00 17 98 04

接收: 01 06 00 01 00 17 98 04

常见问题:

1、仪表连续发送(如 TF=0)时, 电脑串口能收到数据, 但用指令方式仍无响应

原因: 1) 仪表参数错误, D2008、D12 系列仪表 TF = 1; D39 系列仪表为命令方式 2

2) 仪表串口故障, 可通过 test 08 进行自检。

3) 电脑串口故障, 将接串口 2 和 3 短路, 然后通过串口工具发送数据, 看其能否收到自己发送的相同的数据。

4) 接线问题, 仪表 9 芯的 3 号脚 (RXD) 为接收脚, 需要与电脑的 2 号脚 (TXD) 连接

5) 仪表虽支持 modbus RTU 协议, 但 modbus 发指令时有要求, 如读毛重, 其寄存器地址为 40001. 此时必须一次性读 4 个字 (40001~40004), 否则无应答。

2、数据如何解释

以下为读毛重

地址	内容	举例 (重量 1234567)	举例 (重量-23456.7)
40001	符号-或 data6, Data5	0x31,0x32	0x2d,0x32
40002	Data4, Data3	0x33,0x34	0x33,0x34
40003	Data2, Data1	0x35,0x36	0x35,0x36
40004	Data0, 小数点+0x30	0x37,0x30	0x37,0x31

以下为读净重

地址	内容	举例 (重量 1234567)	举例 (重量-23456.7)
40003	符号-或 data6, Data5	0x31,0x32	0x2d,0x32
40004	Data4, Data3	0x33,0x34	0x33,0x34
40005	Data2, Data1	0x35,0x36	0x35,0x36
40006	Data0, 小数点+0x30	0x37,0x30	0x37,0x31

以下是西门子 200 smart 的示例:

读毛重:

二、新协议

只有以下版本支持:

仪表: D2008 型(DF)、D12-YF--V09 版本 D2008-W ---V13 版本, KL-MPLC 的 com2 协议 1

寄存器地址		说明
40060		状态 Bit0: 开机零点确认中 0: 已确认 1: 正在确认中 Bit1: 超载 0: 正常 1: 超载 Bit2: 稳定 0: 不稳定 2: 稳定 Bit3: 去皮 0: 没有去皮 1: 去皮状态 Bit4: 零点 0: 没在零点 1: 零位区 Bit5: 计量数据有效 0: 无效 1: 有效 Bit6: 传感器出错 0: 通讯正常 1: 异常 Bit7: 备用 Bit8~15: 传感器个数
40061		传感器状态 0:正常 1: 异常 Bit16: 1 号 Bit17: 2 号 ... Bit31: 16 号
40062-40063		毛重: float
40064-40065		皮重: float
40066-40067		净重: float
40068-40069		第 1 个传感器内码: float
40070-40071		第 2 个传感器内码: float
...		...
40098-40099		第 16 个传感器内码: float
40100		传感器状态 0:正常 1: 异常 Bit16: 17 号 Bit17: 18 号 ... Bit31: 32 号
40101-40102		第 17 号传感器内码: float
40103-40104		第 18 号传感器内码: float
...		...
40131-40132		第 32 号传感器内码: float

约定:

当秤台计量数据无效: 仪表的净重为-999999,如果小数点为 3 位, 则为-999.999

置零: 向地址 40001 写 0x0017;

2、Modbus 主机配置:

第一项:选择为 RTU

第二项:

Read/Write Definition

Slave ID: 2 OK

Function: 03 Read Holding Registers (4x) Cancel

Address: 60 Protocol address. E.g. 40011 -> 10

Quantity: 40

Scan Rate: 100 [ms] Apply

Disable

☐ Read/Write Disabled

☐ Disable on error

View

Rows

☒ 10 ☐ 20 ☐ 50 ☐ 100 ☐ Fit to Quantity

☐ Hide Alias Columns ☐ PLC Addresses (Base 1)

☐ Address in Cell ☐ Enron/Daniel Mode

按所需内容设定
字长度，如只读
毛重时，就1个
字

如:

Read/Write Definition

Slave ID: 2 OK

Function: 03 Read Holding Registers (4x) Cancel

Address: 66 Protocol address. E.g. 40011 -> 10

Quantity: 2

Scan Rate: 100 [ms] Apply

Disable

☐ Read/Write Disabled

☐ Disable on error

View

Rows

☒ 10 ☐ 20 ☐ 50 ☐ 100 ☐ Fit to Quantity

☐ Hide Alias Columns ☐ PLC Addresses (Base 1)

☐ Address in Cell ☐ Enron/Daniel Mode

第三项: float 显示格式设置:

Modbus Poll - [Mbpoll1]

File Edit Connection Setup Functions Display View Window Help

Tx = 20240: Err = 995: ID = 1: F = 03: SR = 100ms

	Alias	00040	Alias	00050
0				12.485
1				...
2				
3				
4				
5				
6				
7				
8		12.345		
9				

Format

- Read/write Definition... F8
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Select All Ctrl+A
- Foreground Color Ctrl+F
- Background Color Ctrl+B
- Font
- Signed Alt+Shift+S
- Unsigned Alt+Shift+U
- Hex Alt+Shift+H
- Binary Alt+Shift+B
- Long AB CD
- Long CD AB
- Long BA DC
- Long DC BA
- Float AB CD
- Float CD AB
- Float BA DC
- Float DC BA

For Help, press F1.

Port 1: 9600-8-N-1

第三项: 全部显示例子:

	Alias	bit7--0,15--8	00060	Alias	00070	Alias	00080	Alias	00090
0		0010 0100 0000 0100			6519.5		0		0
1		0xFFFF0			--		--		--
2		12.45			6098.3		0		0
3		--			--		--		--
4		0			16679.2		0		0
5		--			--		--		--
6		12.45			0		0		0
7		--			--		--		--
8		2618.1			0		0		0
9		--			--		--		--

3、相关解码代码：接收字节转化为浮点型

定义数据结构:

```
typedef union noneTYPE32DATABYTE
{
```

```
    struct{
        INT32U onebyte0:8;
        INT32U onebyte1:8;
        INT32U onebyte2:8;
        INT32U onebyte3:8;
    }onebyte;
    INT32U One32data;
    FP32    FP32data;
}TYPE32DATABYTE;
```

代码:

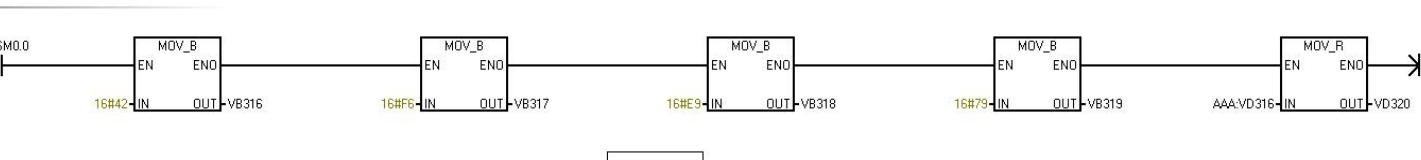
原如接收 4 个字节 (按接收顺序)--Rdata[0],Rdata[1],Rdata[2]和 Rdata[3]

```
float    FP32data;
TYPE32DATABYTE P_temp;
P_temp.onebyte.onebyte1 = Rdata[0];
P_temp.onebyte.onebyte0 = Rdata[1];
P_temp.onebyte.onebyte3 = Rdata[2];
P_temp.onebyte.onebyte2 = Rdata[3];
F_temp = P_temp.FP32data; //F_temp 就是结果
```

在西门子 200smart 中字节转浮点型操作如下:

例: 浮点数 123.456 对应[0]=0x79 0xe9,0xf6,[3]=0x42

PLC 梯形图转化:



效果:



常用例子 1:读毛重

仪表显示毛重 68，皮重为 0

发: 01 03 00 42 00 02 64 1F

收: 01 03 04 00 00 42 88 CA F5

常用例子 2：远程置零

发送: 01 06 00 01 00 17 98 04

接收: 01 06 00 01 00 17 98 04

CRC16 校验码算法:

```
/* CRC 高位字节值表 */
```

```
const unsigned char auchCRCHI[] = {
```

0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,

 $\};$

```
/* CRC 低位字节值表*/
```

```
const unsigned char auchCRCLo[] = {
```

```
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
```

```
};
```

```
/******
```

函数名: INT16U crc16(unsigned char *puchMsg, INT16U usDataLen)

功能: 根据字符串 puchMsg 及长度产生一个 CRC16

输入: unsigned char *puchMsg 字符串 usDataLen 长度

输出: INT16U CRC16

```
*****/
```

INT16U crc16(unsigned char *puchMsg, INT16U usDataLen) //根据字符串 puchMsg 及长度产生一个 CRC16

```
{
```

```
    unsigned char uchCRCHi = 0xFF; /* 高 CRC 字节初始化 */
```

```
    unsigned char uchCRCLo = 0xFF; /* 低 CRC 字节初始化 */
```

```
    unsigned long uIndex; /* CRC 循环中的索引 */
```

```
    while (usDataLen--) /* 传输消息缓冲区 */
```

```
    {
```

```
        uIndex = uchCRCHi ^ *puchMsg++; /* 计算 CRC */
```

```
        uchCRCHi = uchCRCLo ^ uchCRCHi[uIndex];
```

```
        uchCRCLo = uchCRCLo[uIndex];
```

```
    }
```

```
    return (uchCRCHi << 8 | uchCRCLo);
```

```
}
```

```
//=====
```

应用: 假设要发送读毛重指令 (16 进制)

01 03 00 01 00 04

产生校验码的方式

//TXD_C_temps[]存了 01 03 00 01 00 04 (6 个字节)

INT16U crcData;

ByteCount = 6;//数据为 6 字节

crcData = crc16(TXD_C_temps,ByteCount);

TXD_C_temps[byteCount] = crcData >> 8;

byteCount++;

TXD_C_temps[byteCount] = crcData & 0xff;

//此时字符串

01 03 00 01 00 04 15 C9

putchars2_8th(TXD_C_temps,byteCount+1);//通过串口发送