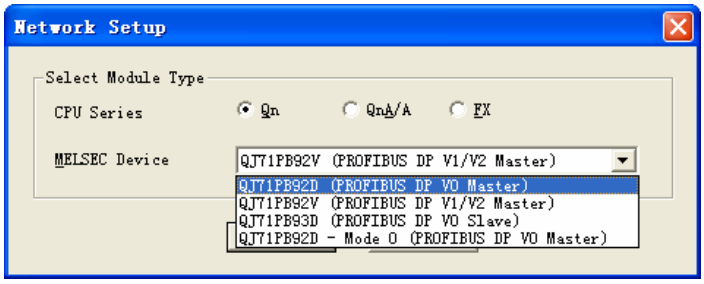
使用 MITSUBISHI GX-Configurator DP 配置 Profibus-DP 网络

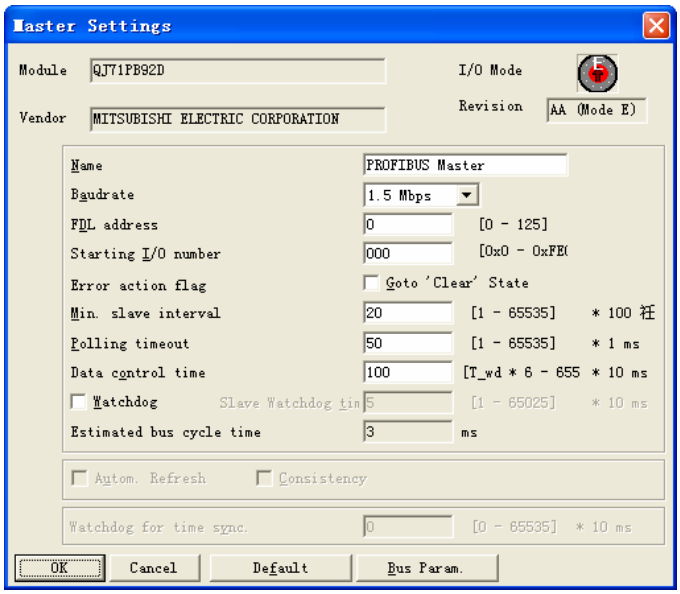
叶工，我升级了新软件，重新配置了一下文件，请见下面内容，如有问题，再联系我。 图尔克 张工

一、使用 MITSUBISHI GX-Configurator DP 配置 Profibus-DP 网络

1、运行 GX-Configurator DP 软件，新建一个项目，选择主站模块。

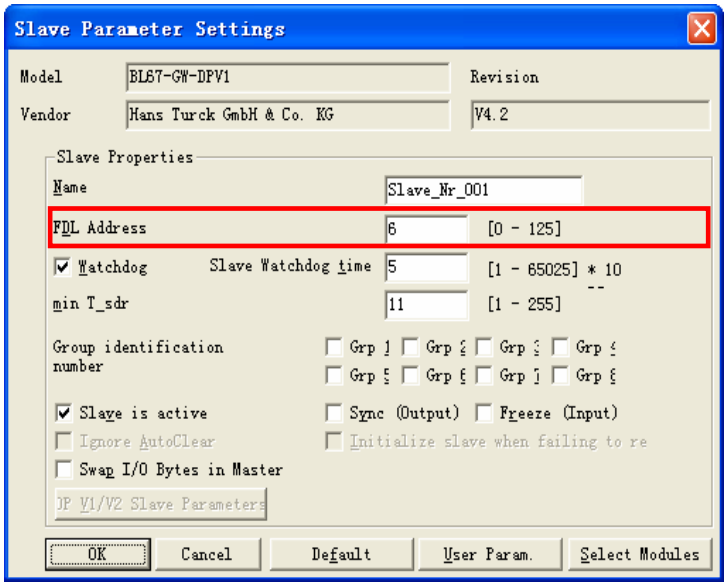


2、双击出现的主站图标，在出现的“Master Settings”对话框中可进行相关参数设置，如总线波特率等。

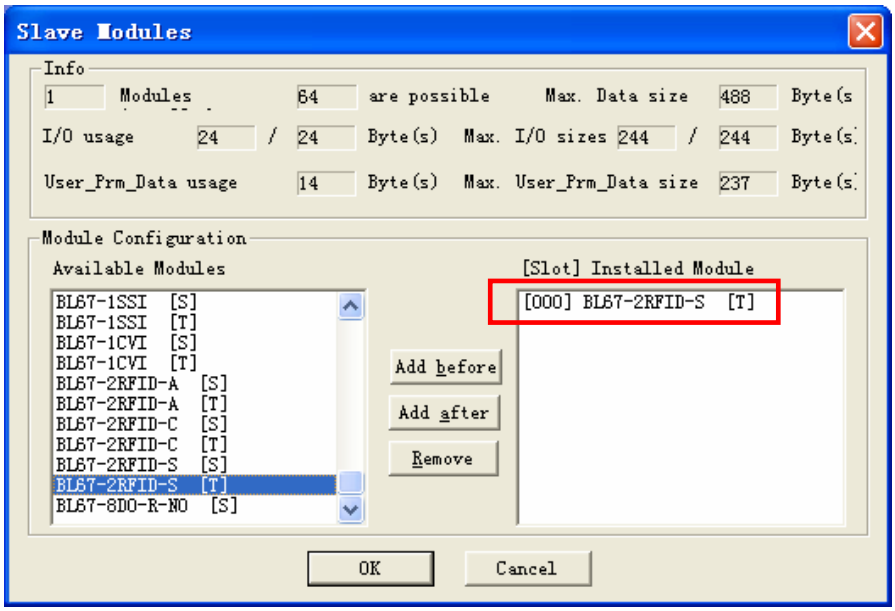


3、在出现的“Slave Parameter Settings”对话框中对该从站进行相关设置。

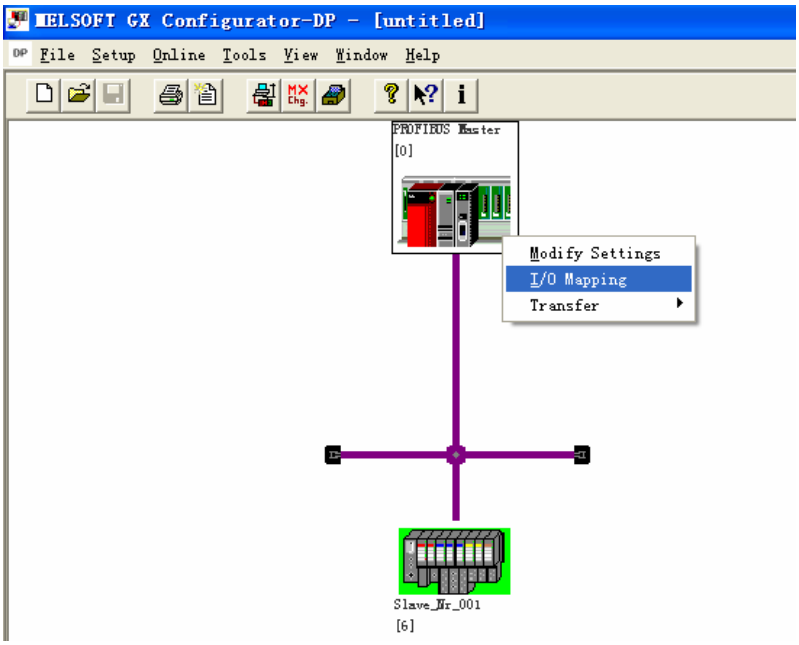
FDL Address：从站地址。

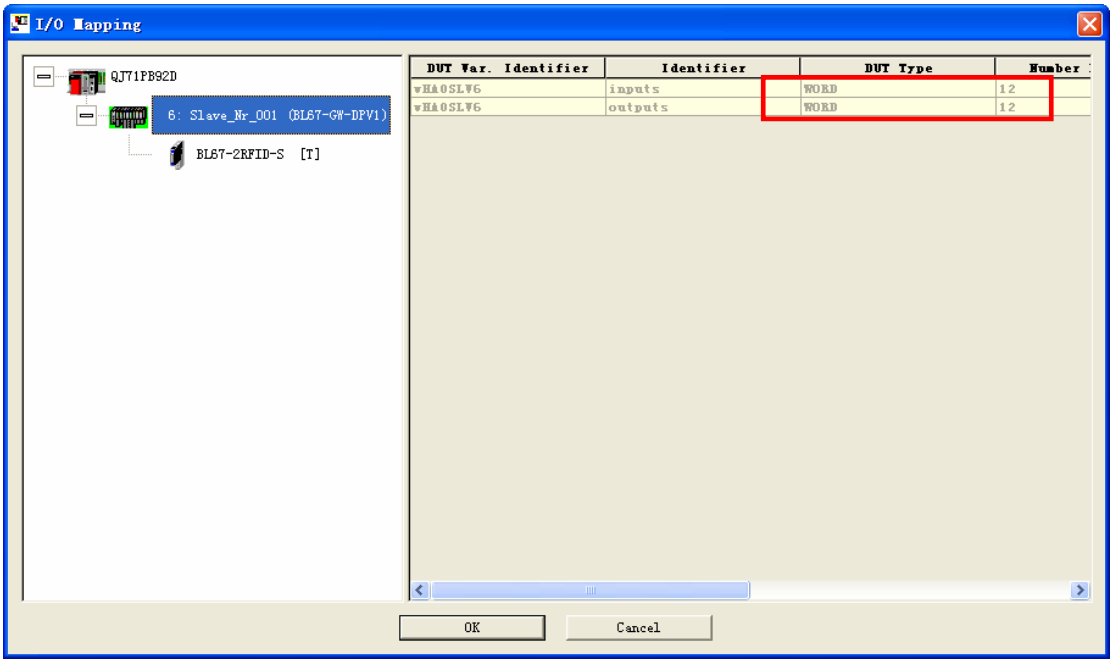


4、点击“Slave Parameter Settings”对话框中“Select Modules”，在出现的“Select Modules”对话框中选择相应的I/O模块进行添加。完成从站模块配置后点击“OK”返回“Slave Parameter Settings”对话框。



5、设置 I/O 映射

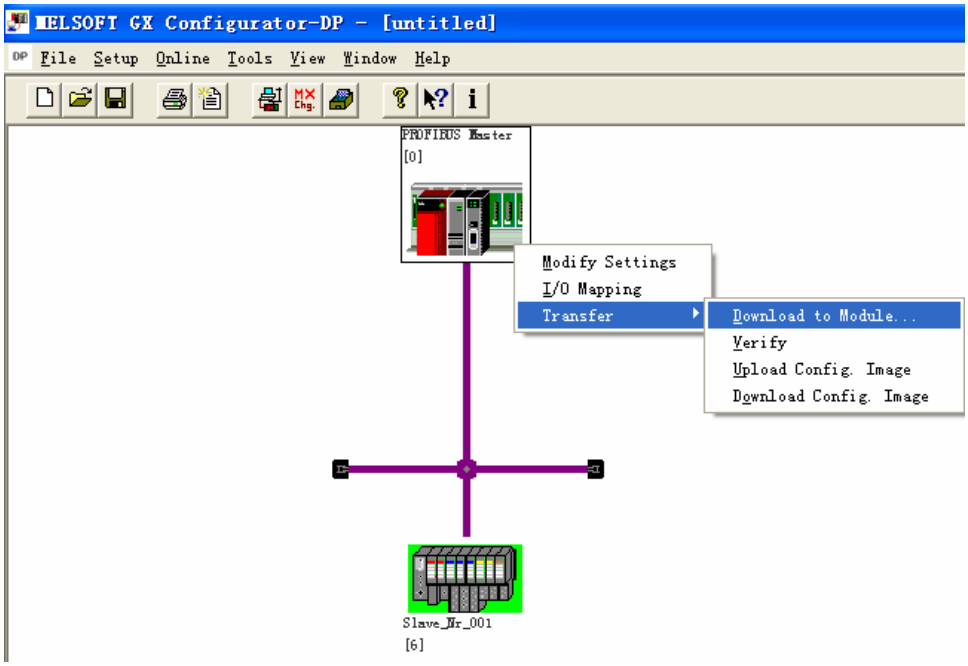




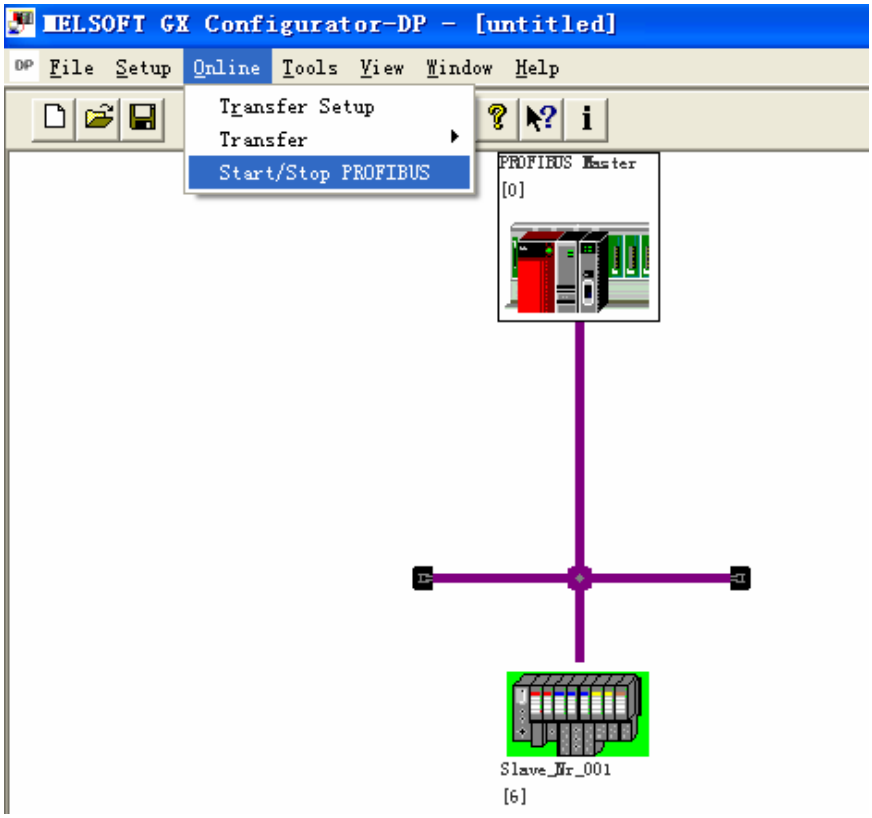
每个模块占用 12 WORD 输入和 12 WORD 输出。

6、保存项目

7、下载信息

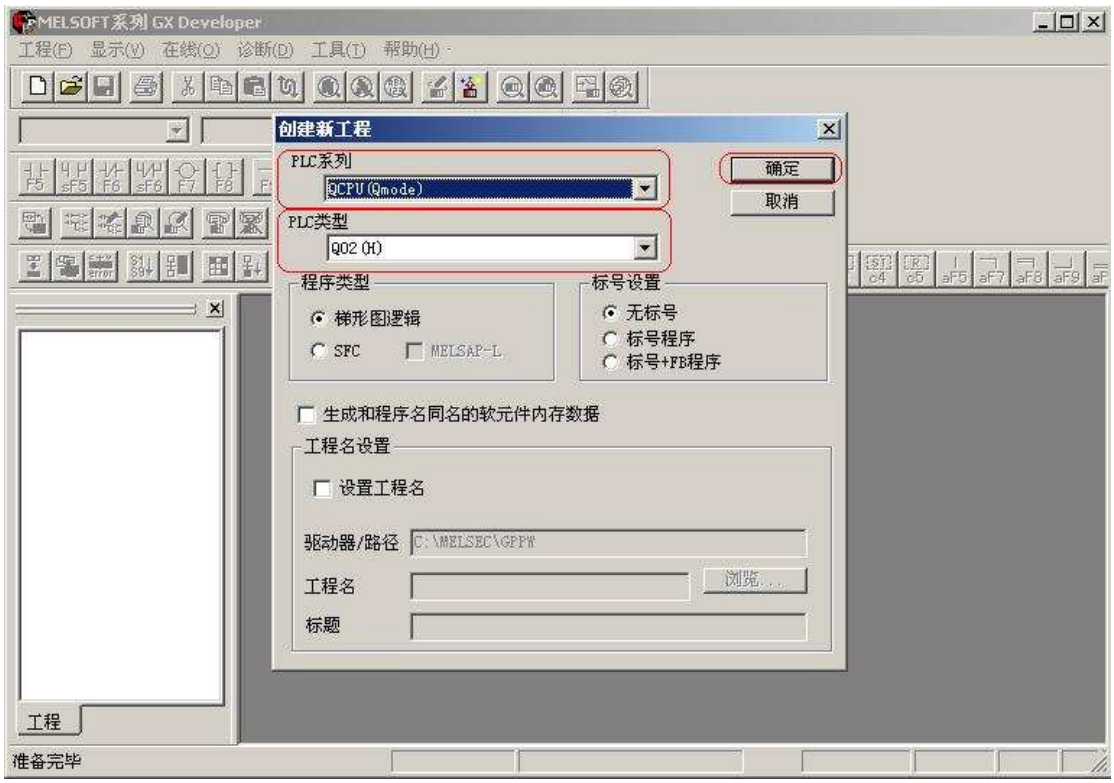


8、启动 Profibus 连接



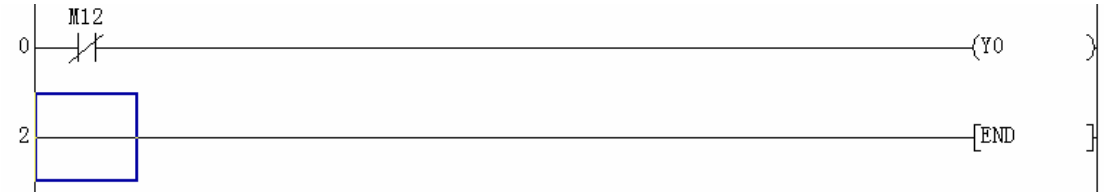
二、使用 GX-Developer 编写 QJ71PB92D 启动程序并监控从站数据

1、在 GX-Developer 软件中新建一个工程，选择所使用的 PLC 类型。



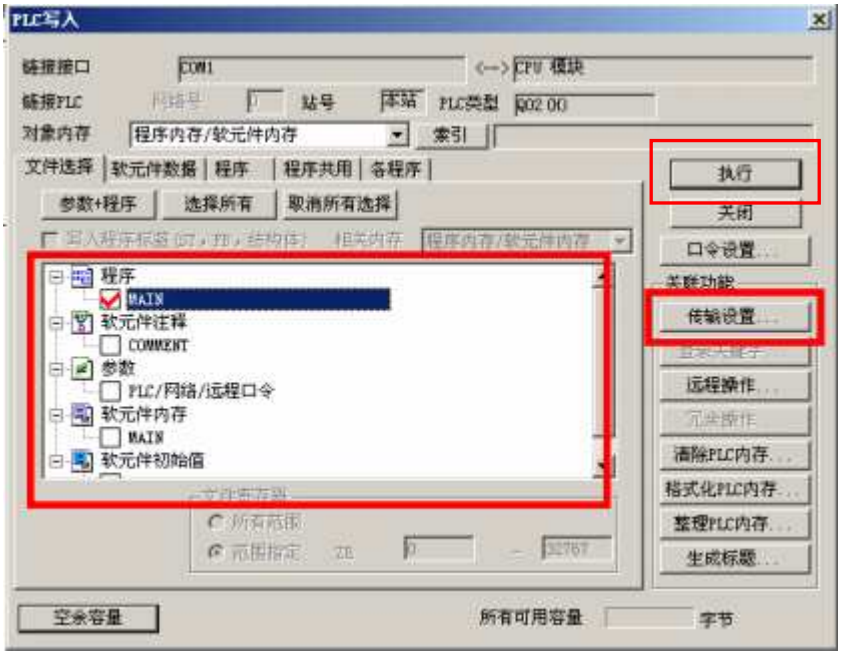
2、在主程序中添加如下程序段，用于启动 QJ71PB92D

注：QJ71PB92D 本身占用 32bit 的 I/O 数据，用于 CPU 对该模块的控制。本例中 QJ71PB92D 为 CPU 机架上的第一个模块，则该模块的 32bit 的 I/O 数据为 X0-X31/Y0-Y31。其中 Y0 置 1 用于启动 PROFIBUS 总线。



3、编写程序后，点击“变换”菜单下的“变换”命令进行程序编译。

4、点击“在线”菜单下“PLC 写入”命令，打开“传输设置”对话框。在该对话框中选择需要下载的选项，点击“执行”。本例中只是下载程序，所以只选中“Main 程序”。



5、应用

PLC 的 DP 模块中的接收缓冲区地址为 0H – 3BFH

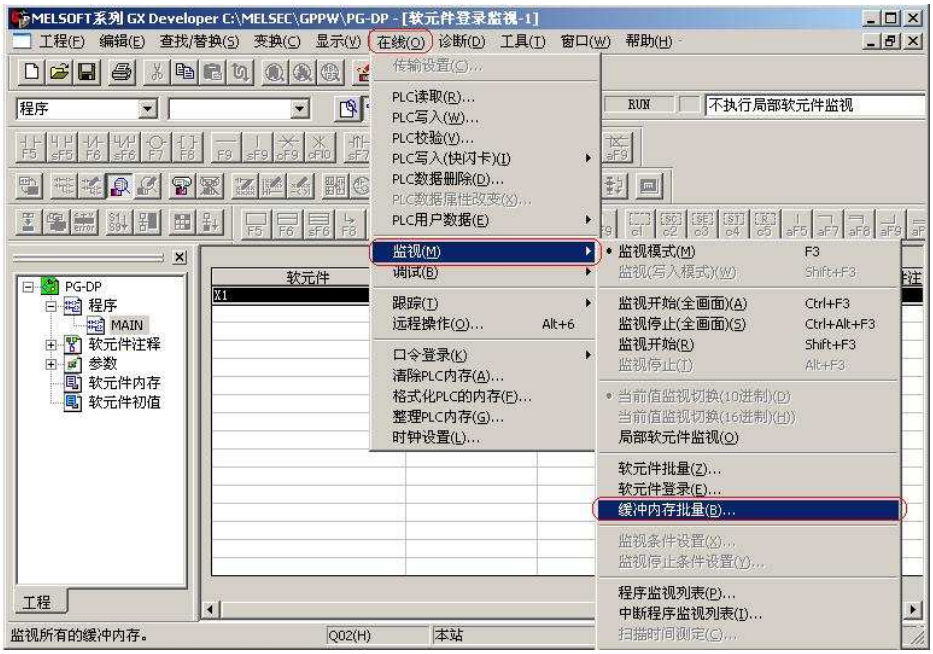
PLC 的 DP 模块中的发送缓冲区地址为 3C0H – 77FH

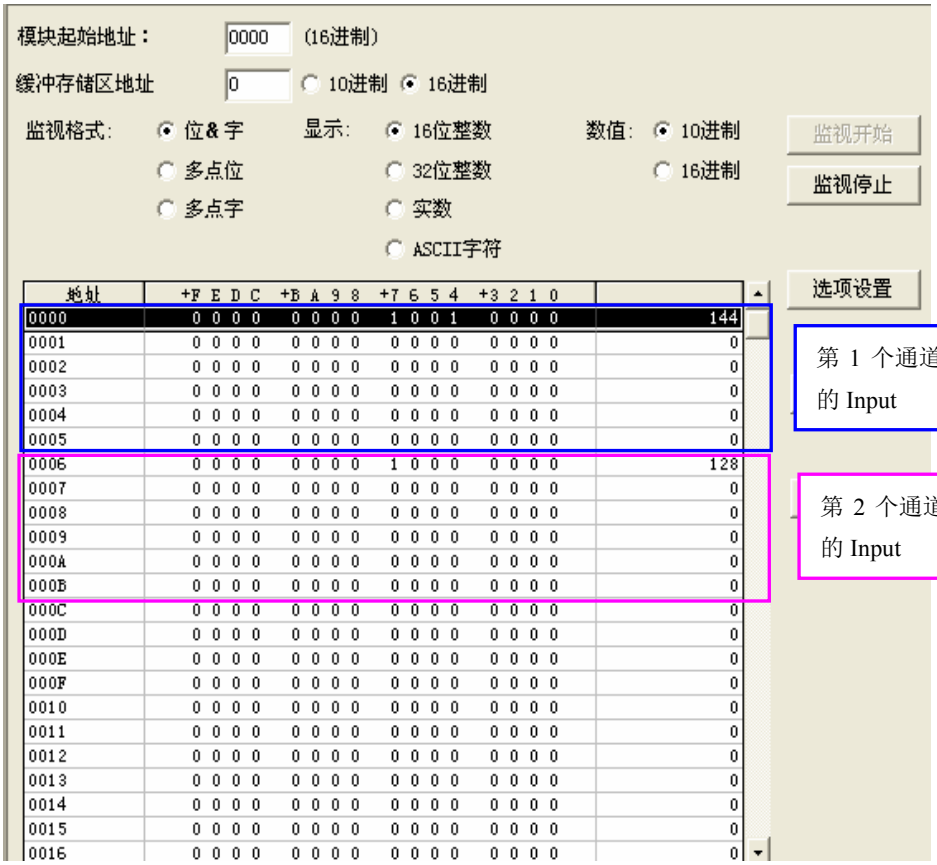
每个 BL67-2RFID-S 模块占用 12 WORD 输入和 12 WORD 输出。

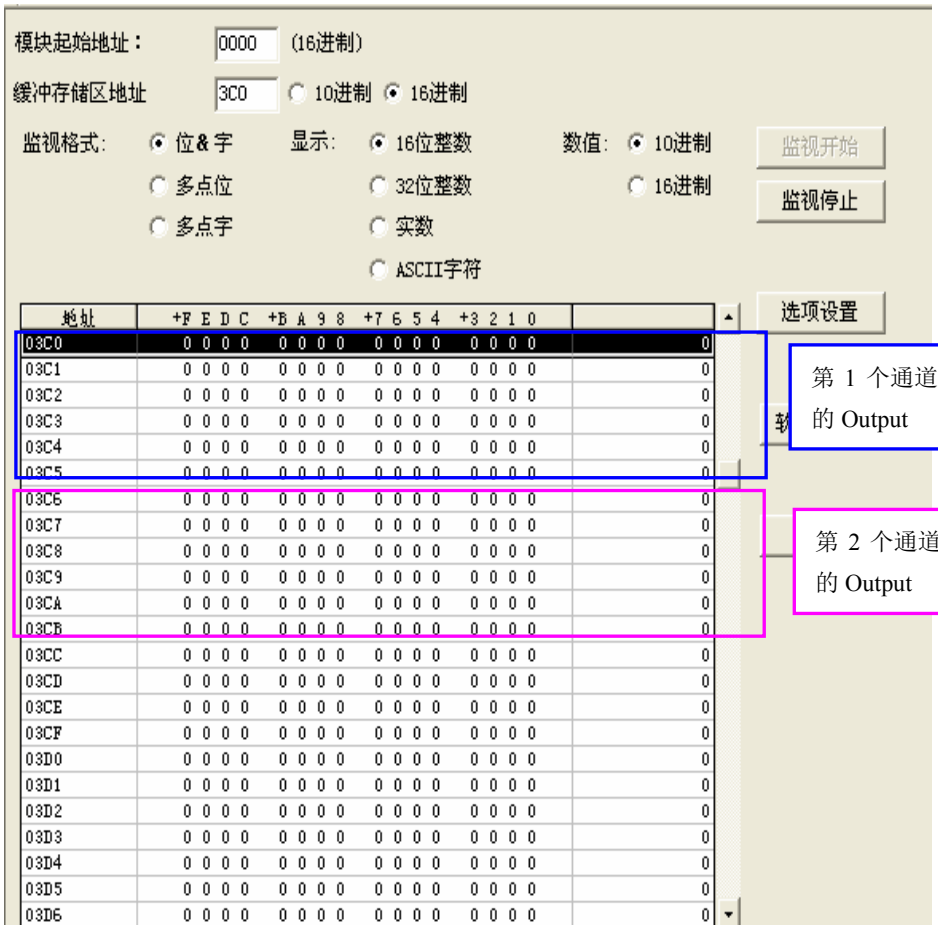
此处所有数据都要从 DP 模块的缓冲区中读取，写入。所以为了方便易懂，只监控缓冲区。如果想要操控 CPU 的地址单元，需自己写一段程序，才可以将该缓冲区的地址单

元 move 到 CPU 的地址单元。

1、原始状态：






模块起始地址：
0000
(16进制)
缓冲存储区地址
3C0
10进制
16进制
监视格式:
位& 字
多点位
多点字
显示:
16位整数
32位整数
实数
ASCII字符
数值:
10进制
16进制
监视开始
监视停止
选项设置
地址
+F E D C
+B A 9 8
+7 6 5 4
+3 2 1 0
03C0
03C1
03C2
03C3
03C4
03C5
03C6
03C7
03C8
03C9
03CA
03CB
03CC
03CD
03CE
03CF
03D0
03D1
03D2
03D3
03D4
03D5
03D6
第 1 个通道的 Output
第 2 个通道的 Output

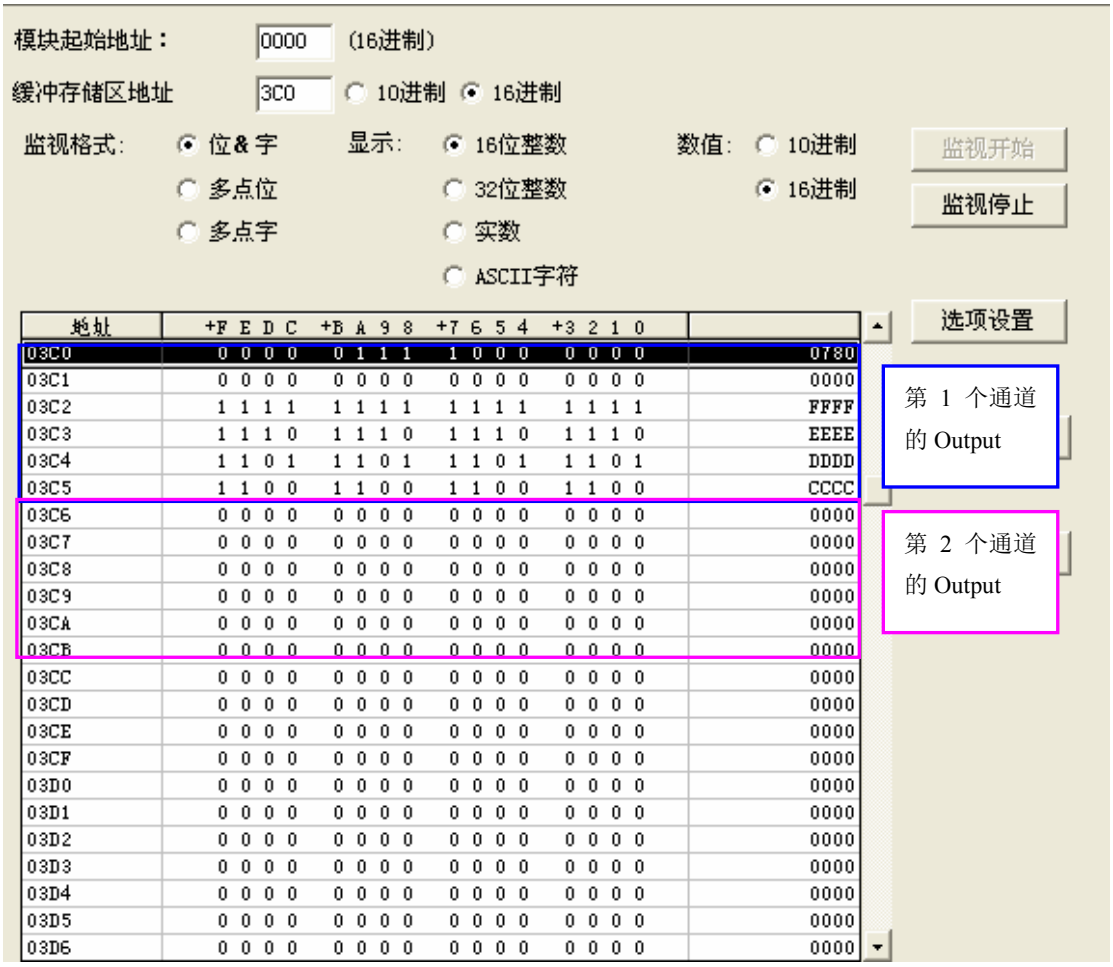
2、设置读写数据长度，例如 8 bytes ，RESET 操作

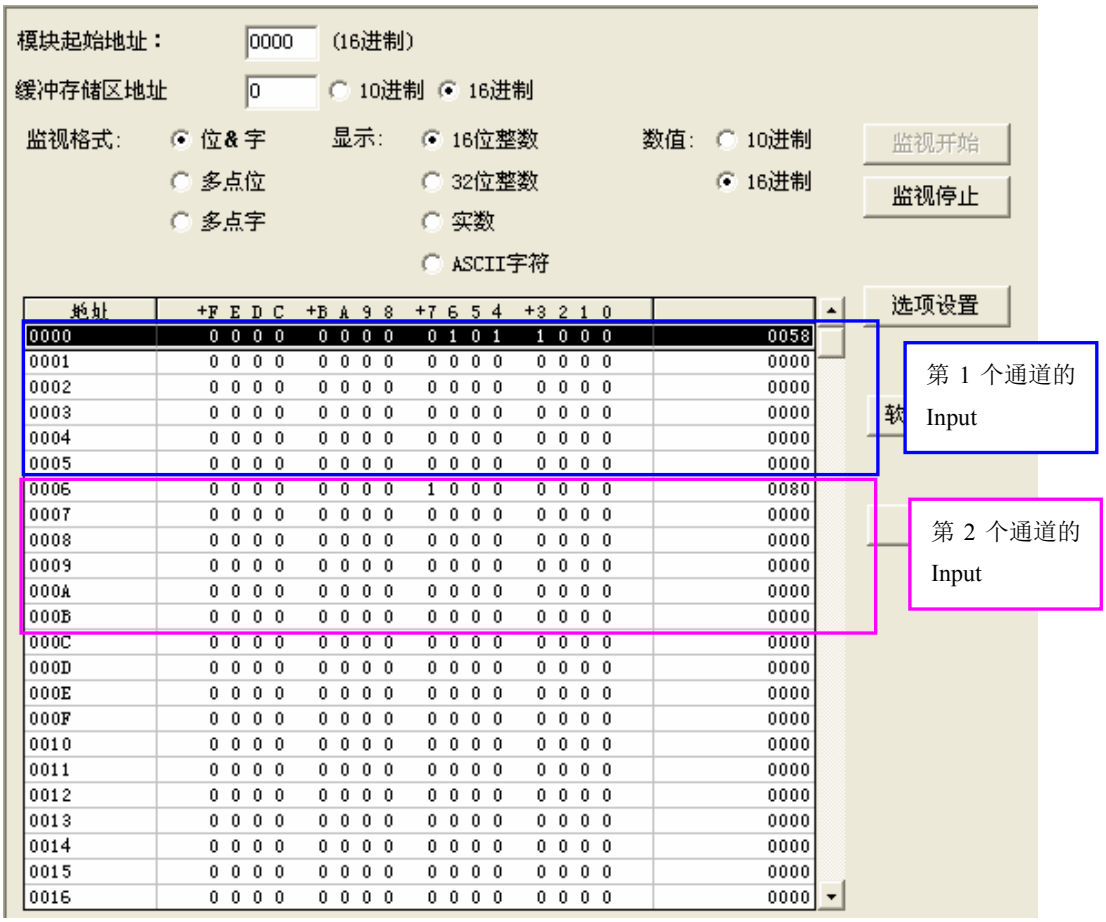
3C0=“0781H”为 Reset 启动，3C0=“0780H”为 Reset 完成，如下图。

地址	+F E D C	+B A 9 8	+7 6 5 4	+3 2 1 0	
0000	0 0 0 0	0 0 0 0	1 0 0 1	1 0 0 0	0098
0001	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0002	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0003	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0004	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0005	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0006	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0080
0007	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0008	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0009	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000A	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000B	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000C	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000D	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000E	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000F	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0010	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0011	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0012	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0013	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0014	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0015	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0016	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000

3、Write 操作：

3C0=“0788H”为 Write 启动，3C0=“0780H”为 Write 完成，如下图。

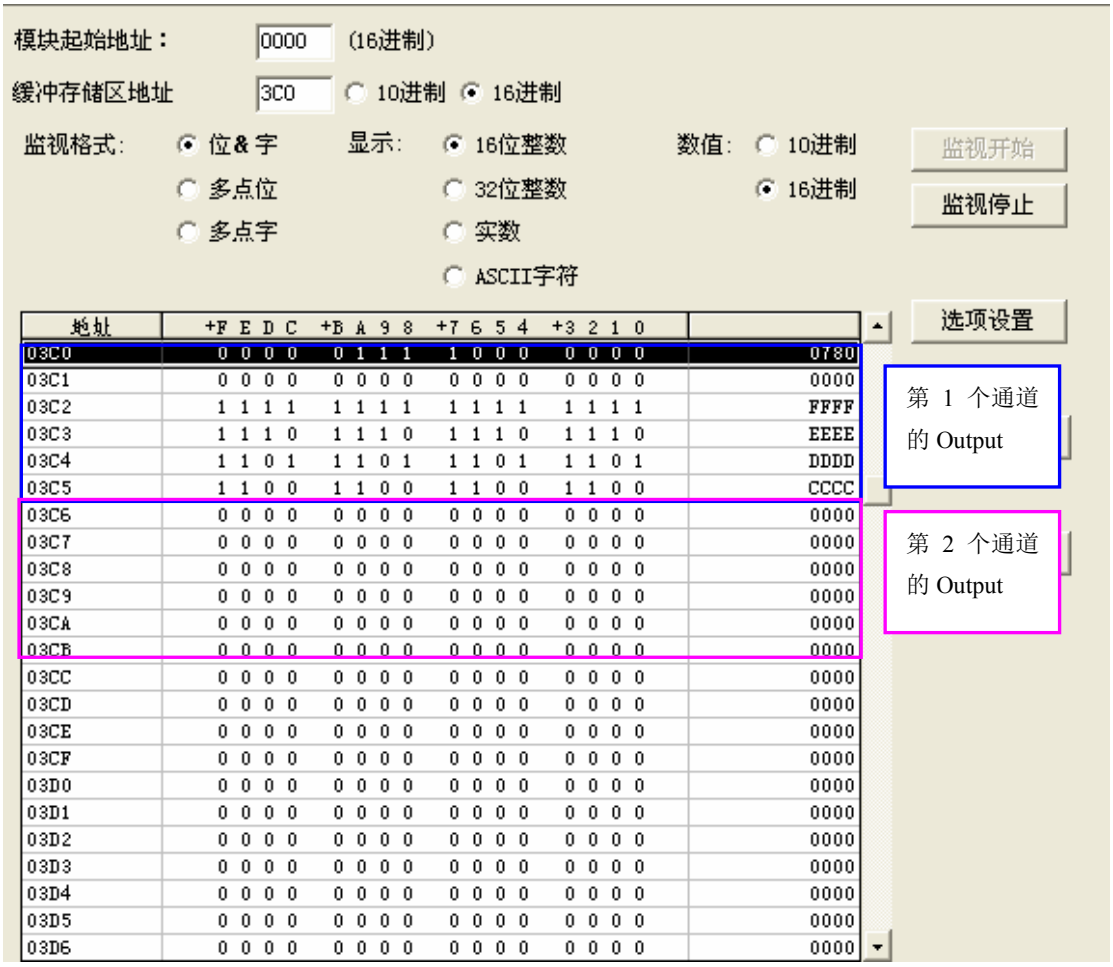




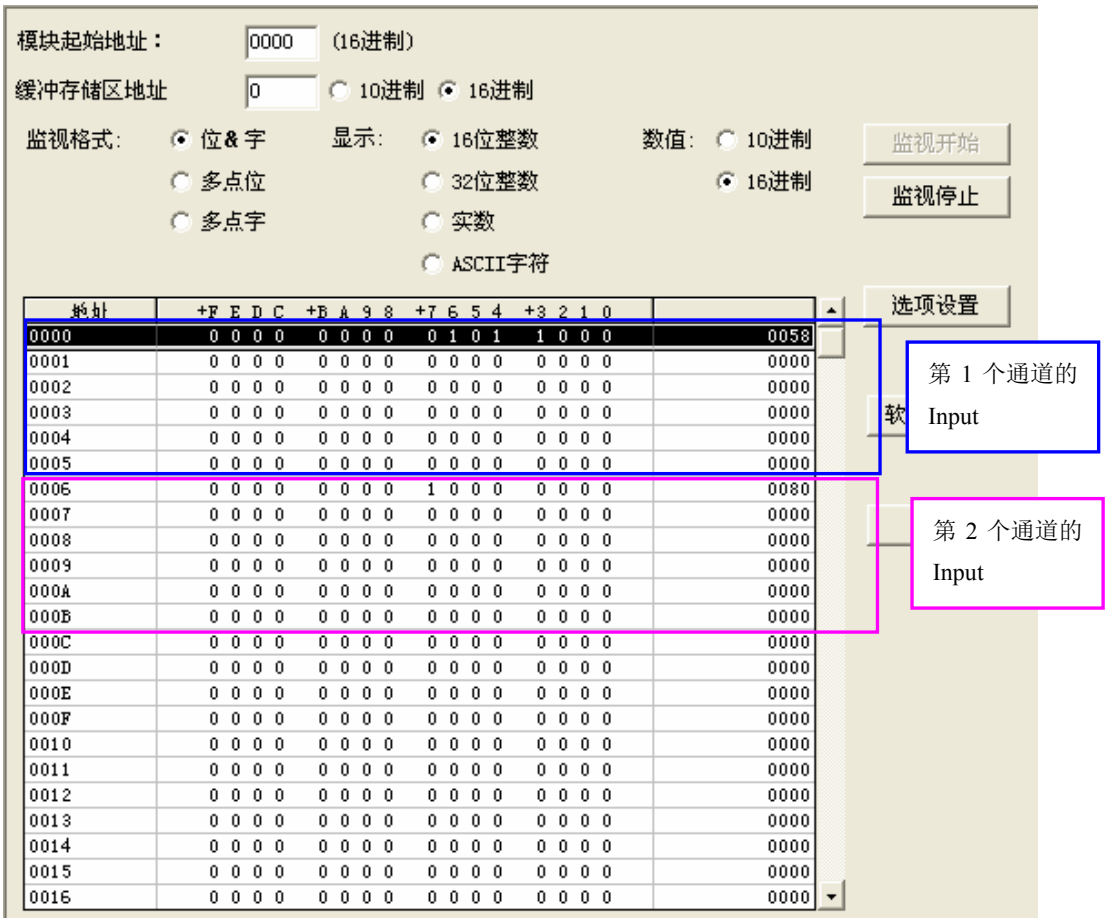
完成上述操作后，把载码体靠近读写头的读写区域，进行写数据到载码体操作。

4、Read 操作：

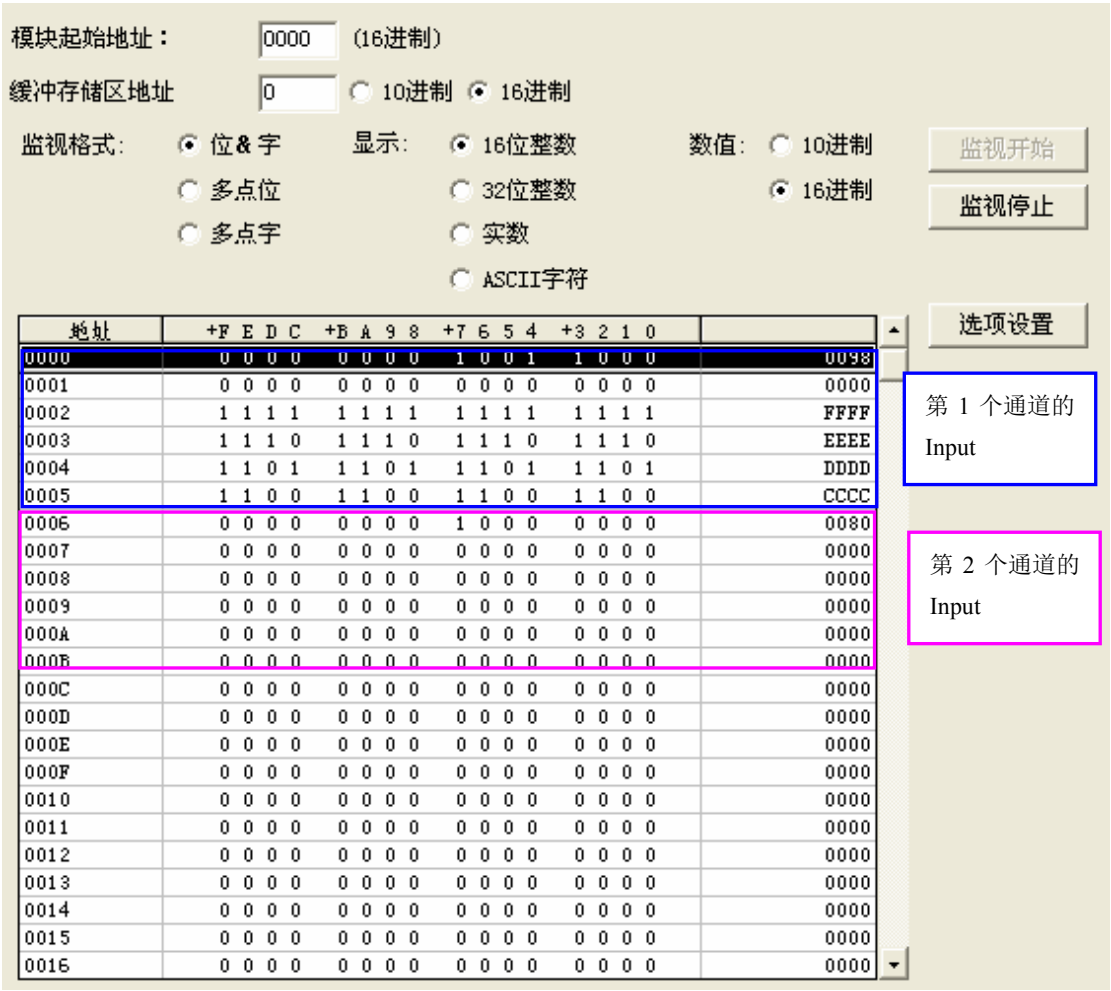
3C0=“0790H”为 Read 启动，3C0=“0780H”为 Read 完成，如下图。



地址	+F E D C	+B A 9 8	+7 6 5 4	+3 2 1 0	
03C0	0 0 0 0	0 1 1 1	1 0 0 0	0 0 0 0	0780
03C1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03C2	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	FFFF
03C3	1 1 1 0	1 1 1 0	1 1 1 0	1 1 1 0	EEEE
03C4	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	DDDD
03C5	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	CCCC
03C6	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03C7	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03C8	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03C9	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CA	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CB	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CC	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CD	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CE	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03CF	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D1	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D2	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D3	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D4	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D5	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
03D6	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000



完成上述操作后，把载码体靠近读写头的读写区域，进行从载码体中读数据操作。如下图，数据读写成功。



地址	+F E D C	+B A 9 8	+7 6 5 4	+3 2 1 0	
0000	0 0 0 0	0 0 0 0	1 0 0 1	1 0 0 0	0098
0001	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0002	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	FFFF
0003	1 1 1 0	1 1 1 0	1 1 1 0	1 1 1 0	EEEE
0004	1 1 0 1	1 1 0 1	1 1 0 1	1 1 0 1	DDDD
0005	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	CCCC
0006	0 0 0 0	0 0 0 0	1 0 0 0	0 0 0 0	0080
0007	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0008	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0009	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000A	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000B	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000C	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000D	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000E	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
000F	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0010	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0011	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0012	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0013	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0014	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0015	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000
0016	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0000

RFID 读写过程参照资料

常用参数说明：

INPUT DATA:

1.DONE　完成读写操作
2.BUSY　接收到读写指令，正在等待载码体
3.XCVR CON　读写头与模块正常通讯标志
4.XCVR ON　读写头处于激活状态
5.READ DATA　读出的数据（最多一次读 8Byte）

I/O Data Mapping

INPUT	BYTE	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel 0	0	DONE	BUSY	ERROR	XCVR CON	XCVR ON	TP	TFR	Reserved
	1	Error Code							
	2	Error Code 1							
	3	Reserved							
	4	READ DATA (8 Byte)							
	5								
	...								
	10								
	11								
Channel 1	12	DONE	BUSY	ERROR	XCVR CON	XCVR ON	TP	TFR	Reserved
	13	Error Code							
	14	Error Code 1							
	15	Reserved							
	16	READ DATA (8 Byte)							
	17								
	...								
	22								
	23								

OUTPUT DATA:

1.XCVR 激活读写头
2.NEXT 读写下一个载码体
3.TAG ID 读取载码体的 ID 号
4.READ 读指令
5.WRITE 写指令
6.RESET 复位指令
7.Byte Count 2.1.0 一次读写的字节数 000=1、001=2、010=3 … 111=8 （最多一次读写 8Byte）
8.Address high byte 读写载码体内数据的起始字节数（高 8 位）
9. Address low byte 读写载码体内数据的起始字节数（低 8 位）
10.WRITE DATA 要写入的数据（最多一次写 8Byte）

OUTPUT	BYTE	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel 0	0	XCVR	NEXT	TAG ID	READ	WRITE	TAG INFO	XCVR INFO	RESET
	1	Reserved					Byte Count 2	Byte Count 1	Byte Count 0
	2	Address high byte							
	3	Address low byte							
	4	WRITE DATA (8 Byte)							
	5								
	...								
	10								
	11								
Channel 1	12	XCVR	NEXT	TAG ID	READ	WRITE	TAG INFO	XCVR INFO	RESET
	13	Reserved					Byte Count 2	Byte Count 1	Byte Count 0
	14	Address high byte							
	15	Address low byte							
	16	WRITE DATA (8 Byte)							
	17								
	...								
	22								
	23								

1. 读数据（使用 Channel 0）：

首先激活读写头：将 XCVR 置 1；指定要读取的数据大小 Byte Count 2.1.0=111，一次读取 8byte；指定读取数据的起始字节数，Address high byte=0，Address low byte=0，从载码体的第 0 个字节开始读；将 READ 置 1，此时 BUSY 为 1，DONE 为 0，等待读取载码体数据；将载码体放入读写范围内，读完成后，BUSY 由 1 变 0，DONE 为 1，读取的 8 个 byte 数据存储到相应的存储区中。

2. 写数据（使用 Channel 0）：

首先激活读写头：将 XCVR 置 1；指定要写入的数据大小 Byte Count 2.1.0=111，一次读取 8byte；指定写入数据的起始字节数，Address high byte=0，Address low byte=0，从载码体的第 0 个字节开始写；将要写入的数据放入相应的存储区中 Address high byte，Address low byte；将 WRITE 置 1，此时 BUSY 为 1，DONE 为 0，等待写入载码体数据；将载码体放入读写范围内，写完成后，BUSY 由 1 变 0，DONE 为 1，写入的 8 个 byte 数据存储到载码体中。