JDM PIC 编程器的原理、制作及应用

李 春 生

摘　要：

JDM 编程器原本是一款针对 PIC 系列 MCU 设计的代码固化工具，设计者十分巧妙地利用了 PC 机 RS232C 串行接口，不需外接电源即可烧写芯片。该编程器从串口窃电的方式不同于经典的设计，对串口的应用也不同寻常，因此对这款编程器进行深入的剖析会给我们带来很多启示。

本文详细讨论了 JDM 编程器的电路设计原理，同时给出了通过串口控制这款编程器的编程方法，以便大家能更好地利用这一经典设计。

关键词：

JDM　　编程器　　PIC　　串口窃电

一、JDM 编程器简介

JDM 编程器是一个非常精巧的设计，设计者十分巧妙地利用了 PC 机上的 RS232 串口，在不用外电源的情况下利用串口产生+5V/+13V 电压。另外，串口的工作方式也很不寻常，设计者充分利用了串口控制器的特性，把本来应工作在“UART”方式的串口改造成类似打印口的“GPIO”方式。

这款编程器原本为PIC系列MCU设计，Bonny Gijzen编写的IC prog软件更是令JDM编程器为广大PIC单片机开发者所熟知。IC prog不仅能支持JDM编程器，还可以支持其它一些编程器硬件，因而IC prog软件的应用不限于PIC系列MCU。该软件可以从www.ic-prog.com下载，目前最新的版本为 1.05D。软件的使用十分简单，本文将不再赘述。

二、PIC 芯片程序固化方法

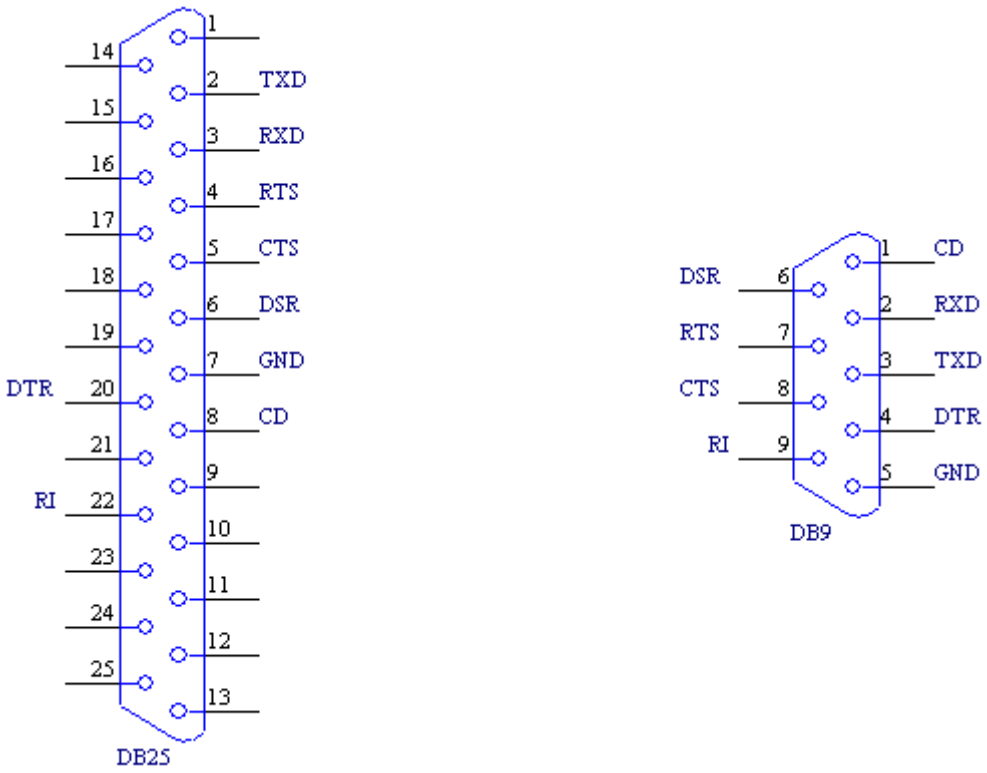
PIC 系列 MCU 为 MicroChip 公司推出的产品，该系列 MCU 具有“在电路串行编程（ICSP）”功能，固化程序时，数据从两个管脚（一般为 RB6/RB7）串行输入 MCU 内，这个特征使我们设计编程器时不必使用过多的 I/O 线。

另外，该系列 MCU 中部分芯片在固化程序时要使用一个 12V – 14V 左右的高电压，这个高压要加到芯片的 MCLR 管脚上才行。这样一来再加上芯片正常供电所用的 Vcc/Vss 线，为该芯片编程就一共要 5 根连线。

ICSP 特性允许产品的生产者预先把空白的芯片焊到电路板上，在发货之前将软件“烧写”到芯片中，从而保证分发出去的产品一定运行了最新的软件。

三、RS232C 串口信号定义

桌面 PC 机通常都有 1 或 2 个串行接口，该接口遵循 RS232C 规范。早期的 PC 机一般使用一个 25 管脚的 D 型连接座，现在的 PC 通常使用 9 管脚的 D 型连接座。串口管脚的定义如图（1）所示。



图（1）RS232C 串口引脚定义

在 9 个信号中，GND 线是公共地，TXD 线用于输出串行数据，DTR 线和 RTS 线是用于控制 MODEM 的。除此之外，其它管脚都是用于输入的。由此可见，PC 的串口并不足以产生 PIC 芯片编程时所需的 5 根连线。

RS232C 规范所规定的逻辑电平与我们通常接触的 TTL 逻辑电平或 CMOS 逻辑电平有很大不同：TXD 线输出的逻辑“1”的电平相对 GND 线为-5V 至-15V 之间；逻辑“0”的电平为+5V 至+15V 之间。通常我们并不将 TXD 线上的称为逻辑“0”和“1”，而是将其称为“SPACE”和“MARK”。DTR 和 RTS 线上的逻辑电平和 TXD 线相反，逻辑“1”的电平相对 GND 线为+5V 至+15V 之间，逻辑“0”则为-5V 至-15V 之间。因此将 TXD 线上的逻辑状态称为“SPACE”和“MARK”也是为与 MODEM 控制信号相区分。

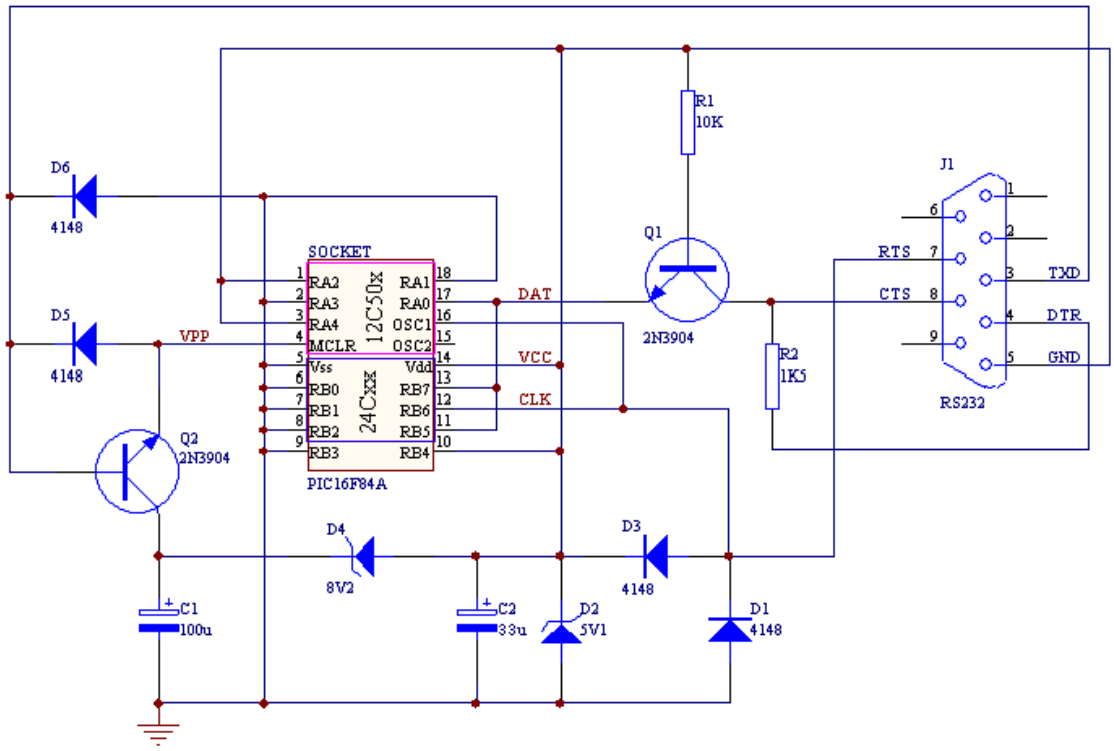
对于输入线，比如 CTS 线，逻辑 0 的输入电平为-3V 至-15V 之间，逻辑 1 的输入电平为+3V 至+15V 之间，比输出信号要更宽一些。

通常情况下，PC 机串口各输出线对 GND 线的输出电压都在±10V 左右，很少有能达到±15V 的，所以想产生 PIC 芯片所要求的+12V 到+14V 之间的编程高压就需要一点技巧——不使用 GND 线做电压基准，而是使用一根输出线做基准，当这根输出线相对 GND 线输出-10V 时，另一根相对 GND 输出+10V 的线相对这个新的电压基准就输出了+20V 电压。同时，GND 线相对这个基准输出了+10V 电压。

DTR 和 RTS 线的输出可以通过软件来设置为逻辑“0”或“1”，TXD 线在不发送数据时为“MARK”状态，TXD 线是可以通过软件设为“SPACE”状态，但设置方法与 DTR/RTS 有所不同。

四、JDM 编程器电路设计原理

了解了 RS232C 串口的特性，我们就可以分析 JDM 编程器的硬件设计原理了。图（2）是 JDM 编程器的硬件原理图。图中的 Socket 是一个 18 脚的 IC 插座，可以插一片 PIC16F84，1 – 4 脚和 15 – 18 脚可以插 PIC12C508/9，5 – 8 脚和 11 – 14 脚可以插 24Cxx 系列 EEPROM，因此 JDM 编程器其实还是一个多功能编程器，并不仅限于对 PIC 系到 MCU 编程。



图（2）JDM 编程器电路原理图

（1）供电电路原理分析

从图（2）中大家可以看到 PC 串口的 GND 线连接到芯片插座的 Vdd 线上，芯片插座的 Vss 线则经二极管 D6 接 TXD 线，同时经 D1 接 RTS 线。由此我们可以看出，PC 串口的 GND 线实际为芯片插座提供+5V 电源，而芯片插座的 Vss 则由 TXD 和 RTS 供给。

通常情况下 TXD 线为“MARK”状态，也就是相对 GND 线输出约-10V 电压，这样一来加在芯片插座的 Vdd 和 Vss 间的供电电压就是+10V。5．1V 稳压管 D2 的作用也就显而易见了。此时 8．2V 稳压管 D4 处在正向导通状态，电容 C1 的+端充了约 4．5V 的电压，这样来看三极管 Q2 的集电极为 4．5V，二极管 D5 短路的基极和发射极，Q1 处在截止状态，芯片插座的 MCLR 线经 D5 连在 TXD 线上，不具有编程所需的高压。

现在我们再来看 RTS 线，如果 RTS 线相对 GND 线为+10V，那么二极管 D3 将导通，RTS

线实际被拑位在 5．7V 左右。如果 RTS 线相对 GND 线为-10V，那么它与同样为-10V 的 TXD 线一样。但如果我们控制 RTS 线为-10V，而令 TXD 线输出+10V 电压，此时三极管 Q2 的两个 PN 结都是正偏，+10V 电压通过 Q2 的 bc 结加到电容 C1 上，使得 C1 两端电压提升到+15V。这样一来 8．2V 稳压管 D4 就工作在稳压状态了，它与 5．1V 稳压管 D2 相串连，令 C1 的+端的电压稳定在+13．3V（相对 Vss），此高压经 Q2 的 be 结加到芯片插座的 MCLR 脚上。

当 TXD 线用于产生编程高压后，芯片插座的 Vss 端就只能依靠 RTS 线供电了。而 RTS 线本身还要用作编程时钟线，所以对 RTS 的利用就有特殊之处，我们在下一小标题讨论这个问题。

（2）时钟/数据产生电路原理分析

芯片插座的 RB6 脚是编程时钟输入端，时钟由串口的 RTS 线产生。我们已经分析过当 TXD 线为-10V 时，RTS 线如果输出+10V 电压，则二极管 D4 将其输出拑位在 5．7V 左右，而 RTS 线输出为-10V 时则等同于 TXD 线。

然而当 TXD 线为产生编程高压而输出+10V 时，我们需要换个角度去理解 RTS 线的作用：通常我们习惯把芯片的 Vss 脚做电路参考点，此时串口的 GND 线为芯片的 Vdd 脚供应+5V 电源电压；现在我们需要把芯片的 Vdd 脚做为电路的参考点，这时我们会看到串口的 RTS 线在产生编程时钟的同时，经二极管 D1 做半波整流，再经电容 C2 滤波 D2 稳压，为芯片的 Vss 脚供应-5V 电源电压。

芯片编程所需的串行数据由 DTR 线产生，注意芯片的 RB7 脚是双向口，编程数据从 RB7 写入片内，校验时片内数据也从 RB7 输出。从片内读取数据的原理比较好理解：三极管 Q2 的基极已然经电阻 R1 连接在+5V 上了，因此只要令 DTR 线输出+10V，三极管 Q1 的工作状态就受 RB7 脚的控制。当 RB7 脚输出逻辑 0 时 Q1 导通，集电极被拉到与 Vss 脚相同的电位，串口的 CTS 线就收到相对 GND 线约-5V 的电压；当 RB7 脚输出逻辑 1 时 Q1 截止，DTR 线输出的+10V 经电阻 R2 加到 CTS 线上。

当 RB7 脚用做输入时三极管 Q1 的工作不大好理解，我们不妨将 Q1 看做是串联稳压电源中的电压调整管，当 DTR 线相对 GND 线输出+10V 电压时，Q1 会将此电压降到约 5V 供给 RB7 脚；而当 DTR 线相对 GND 线输出-10V 电压时，Q1 发射级也会有一个相对 GND 线的负压输出。这个负压比电容 C1 的负极还要低一些，足以当作逻辑 0 了。

五、JDM 编程器的制作

（1）元件选择

两个三极管可以选择 2N3904 等常用的 NPN 小功率管，两个稳压管可以选择功率为 0.5W 的，建议选择 1W 的以提高可靠性。100u 的电容 C1 耐压应选 25V 以上，电阻选普通的 1/4W 电阻即可。

（2）电路调试

电路焊接无误后即可连接到 PC 机串口上，连接好后用万用表测量电容 C2 两端电压应该在 5.1V 左右，电容 C1 两端电压应该在 4.5V 左右。将编程器从串口上断开，这时 C2 两端电压应降到 1V 以下，而 C1 由于缺乏放电回路，其两端电压将维持在 4.5V。

我们可以利用 ic prog 软件进一步调试电路。Ic prog 下载后不用执行安装程序即可使用。如果是第 1 次启动 ic prog，我们会首先看到一个配置介面，选择“JDM Programmer”，并选择好串口。注意在 WIN2000/XP 系统下“Interface”一项必须“Windows API”，否则软件无法正常工作。

软件主窗口打开后我们可以从“Settings”菜单栏中选择“Hardware Check”一项，此时会有一个小对话档显示，选择“Enable MCLR”项后用万用表测电容 C1 两端电压应约为 13.3V，取消“Enable MCLR”项后这个高压仍然会维持一段时间。

在取消“Enable MCLR”项后我们点击“Enable Clock”项，此时我们用万用表测串口插头的第 7 脚 RTS 线到电容 C2 负极的电压应该在 5.7V 左右，取消“Enable Clock”项 RTS 线到 C2 负极的电压应降至-0.6V。

点击“Enable Data Out”项后测晶体管 Q1 的发射极对 C2 负极的电压应该为 4.6V 左右，取消“Enable Data Out”项后这个电压应降至-1.6V。如果在编程插座中插上一片 PIC16F84，则这个电压则不会降到-1.6V。

六、JDM 编程器的二次开发

对 JDM 编程器进行二次开发，其重点是如何在 PC 上编写程序控制串口。我们将要讨论的串口编程都是基于 WIN32 平台的，在 DOS 下如何编程不在本文中讨论。

目前在 WINDOWS 平台上对串口编程主要是采用一些现成的控件或者调用 WINDOWS API。我们准备采用后一种方式，以避免集成在一些控件中的复杂功能影响主要矛盾。程序例（1）显示出如何通过 WINDOWS API 访问串口。

程序例（1）

```
#include <windows.h>
#include <stdio.h>

void main(void)
{
    HANDLE hPort;
    DWORD dwBytesTransferred;
    BYTE str[] = "Hello,world\n";

    hPort = CreateFile("COM1:",
                       GENERIC_READ | GENERIC_WRITE,
                       0,
                       NULL,
```

```
                        OPEN_EXISTING,
                        0,
                        NULL);
    if (hPort == INVALID_HANDLE_VALUE)
    {
        printf("Port COM1 Open Error!\n");
        return;
    }

    WriteFile(hPort, str, sizeof(str), &dwBytesTransferred, NULL);

    strcpy(str,"");
    ReadFile(hPort, str, sizeof(str), &dwBytesTransferred, NULL);
    printf(str);

    CloseHandle(hPort);
}
```

例（1）显示了串口的规范用法，也就是做为“UART”通过 TXD 线向外发送数据以及从 RXD 线接收数据。大家可以把 COM1 的 TXD 线（DB9 插座第 3 脚）和 RXD 线（DB9 插座的第 2 脚）直接短接，即可观察到例（1）执行时显示出的“自发自收”效果。

而要想控制 JDM 编程器，我们必须解决的问题是如何分别控制 TXD 线、RTS 线和 DTR 线分别输出 0 或 1，以及如何通过 CTS 线读取芯片发出的数据。对 RTS 线和 DTR 线的控制要使用 API 函数 SetCommState 以及一个名为 DCB 的数据结构。控制 TXD 线则要使用 API 函数 SetCommBreak 和 ClearCommBreak。要通过 CTS 线读取数据，则要使用 API 函数 GetCommModemStatus。我们首先讨论如何控制 RTS 线和 DTR 线。

DCB 是个比较大的数据结构，我们只对它的两个成员进行讨论。首先是“fDtrControl”，这个成员控制串口的 DTR 线。它有 3 个不同的取值，取“DTR_CONTROL_DISABLE”时 DTR 线对 GND 线输出为-10V，取“DTR_CONTROL_ENABLE”时 DTR 线对 GND 线输出+10V，取“DTR_CONTROL_HANDSHAKE”时 DTR 线用于串行通信的握手信号。

另一个成员是“fRtsControl”，它控制串口的 RTS 线，取值与“fDtrControl”类似，取“RTS_CONTROL_DISABLE”时 RTS 线对 GND 线输出-10V，取“RTS_CONTROL_ENABLE”时 RTS 线对 GND 线输出为+10V。

程序例（2）显示了如何令 DTR 线对 GND 输出-10V 和令 RTS 线对 GND 线输出+10V 的编程方法，注意在使用函数 SetCommState 设置 DCB 之前，应先调用 GetCommState 函数获得当前的 DCB 设置。

程序例（2）

```
#include <windows.h>
#include <stdio.h>

void main(void)
{
```

```
    HANDLE hPort;
    DCB PortDCB;

    hPort = CreateFile("COM1:",
                       GENERIC_READ | GENERIC_WRITE,
                       0,
                       NULL,
                       OPEN_EXISTING,
                       0,
                       NULL);
    if (hPort == INVALID_HANDLE_VALUE)
    {
        printf("Port COM1 Open Error!\n");
        return;
    }

    PortDCB.DCBlength = sizeof(DCB);
    GetCommState(hPort, &PortDCB);

    PortDCB.fDtrControl = DTR_CONTROL_DISABLE;
    PortDCB.fRtsControl = RTS_CONTROL_ENABLE;

    if (!SetCommState(hPort, &PortDCB))
    {
        printf("Unable to configure the serial port!\n");
        return;
    }

    CloseHandle(hPort);
}
```

控制 TXD 线的方式与 DTR/RTS 线不同，调用函数 SetCommBreak 函数可以令 TXD 线对 GND 线输出+10V，调用函数 ClearCommBreak 函数可以令 TXD 线输出-10V。程序例（3）显示了对 TXD 线的编程方法。

程序例（3）

```
#include <windows.h>
#include <stdio.h>

void main(void)
{
    HANDLE hPort;
    DCB PortDCB;

    hPort = CreateFile("COM1:",
```

```
                        GENERIC_READ | GENERIC_WRITE,
                        0,
                        NULL,
                        OPEN_EXISTING,
                        0,
                        NULL);
    if (hPort == INVALID_HANDLE_VALUE)
    {
        printf("Port COM1 Open Error!\n");
        return;
    }

    if (!SetCommBreak(hPort))
    {
        printf("Unable to set break!\n");
        return;
    }
    getch();
    if (!ClearCommBreak(hPort))
    {
        printf("Unable to clear break!\n");
        return;
    }

    CloseHandle(hPort);
}
```

想通过 CTS 线读取数据，我们必须先令 DTR 线相对 GND 线输出+10V，使三极管 Q2 有条件进入正常的放大状态。程序例（4）演示了如何通过 CTS 线读取芯片输出的数据，例（4）运行后会一直监示 CTS 线，当 CTS 线的状态发生改变后就在屏幕上显示读到的内容。因此我们运行例（4）之后可以从一短导线将 IC 插座的 17 脚与 5 脚重复短接几次，即可看到例（4）读出的内容。

程序例（4）

```
#include <windows.h>
#include <stdio.h>

void main(void)
{
    HANDLE hPort;
    DCB PortDCB;
    DWORD OldStatus;
    DWORD NewStatus;

    hPort = CreateFile("COM1:",
```

```
                        GENERIC_READ | GENERIC_WRITE,
                        0,
                        NULL,
                        OPEN_EXISTING,
                        0,
                        NULL);
if (hPort == INVALID_HANDLE_VALUE)
{
      printf("Port COM1 Open Error!\n");
      return;
}

PortDCB.DCBlength = sizeof(DCB);
GetCommState(hPort, &PortDCB);

// ENABLE or DISABLE? That's a problem.
PortDCB.fDtrControl = DTR_CONTROL_ENABLE;
PortDCB.fRtsControl = RTS_CONTROL_DISABLE;

if (!ClearCommBreak(hPort))
{
      printf("Unable to set break!\n");
      return;
}

GetCommModemStatus(hPort, &OldStatus);
printf("CTS is %s now\n", (OldStatus&MS_CTS_ON) ? "on":"off");

for(;;)
{
      GetCommModemStatus(hPort, &NewStatus);
      if (NewStatus != OldStatus)
      {
            Sleep(10);
            GetCommModemStatus(hPort, &NewStatus);
            if (NewStatus != OldStatus)
            {
                  printf("CTS is %s now\n",(NewStatus&MS_CTS_ON) ? "on":"off");
                  getch();
                  break;
            }
      }
}
```

```
    CloseHandle(hPort);
}
```

附：作者的联系方式

作者个人网站：www.nucstorm.com/geniekits/index.html

作者E-Mail：webmaster@nucstorm.com