数字通道篇

BY_JC/HC

20220520

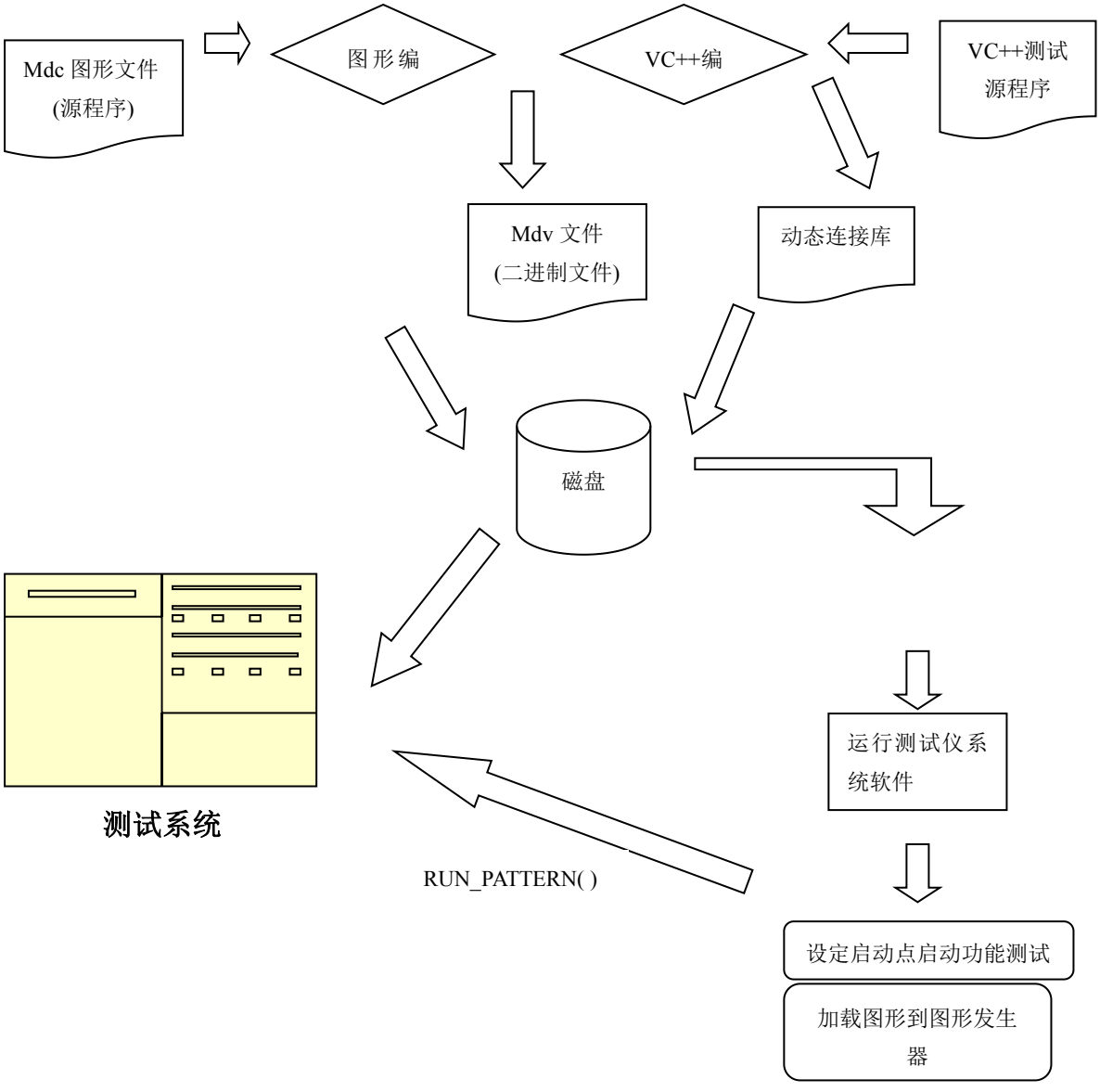
一． 图形文件编写方法

本系统使用 Visual C++ 6.0 作为开发测试程序的工具，测试程序以动态连接库的形式生成，由系统操作软件调用执行。

测试程序包括两部分：

- 图形文件：用于描述测试图形及图形顺序流向控制
- 测试程序：产生测试程序动态连接库文件

1．1 图形文件的生成、运行简介



图形文件是给定格式的文本文件，先在任意文本编辑器环境下编辑图形文件的 ASCII 源文件，然后执行 ComplieDlg.exe 程序编译源文件，转换成与测试系统硬件相适应的图形文件二进制目标文件。

测试程序执行函数 LOAD_PATTERN()函数，把目标图形文件装载到测试系统的存储器，设定好启动点，执行 RUN_PATTERN()函数，启动测试。

2. 图形文件编写步骤

2.1 标准图形文件以.mdc 为扩展名，文件的第一行为“MEM_SOURCE_15;”。

- 定义管脚

  以“PINDEF”为开始标记，定义管脚及管脚组
- 定义管脚与通道对应关系

  以“PIN_TO_CHANNEL”为开始标记，定义管脚与通道的对应关系
- 编辑图形指令及数据

  以“MAIN_F”为开始标记，编辑测试图形的流程及图形
- 结束标志

  以“END.”为结束标记，结束图形文件的编辑
- 图形文件编译转换

  图形文件编辑完成后，在 CompileDlg.exe 程序环境下，填写或浏览（查找）要编辑的文件 -> 编译 -> 执行编译转换。转换成与源文件相同名称，扩展名为 .mdv 的二进制文件。如源文件有错误，提示错误行号，错误原因。

2.2 图形编译与查错程序

图形编译与查错程序 ComplieDlg.exe 文件是一个单独的执行程序，包括两个功能：

a) 编译图形文件，提示错误信息

b) 标记图形运行的失效行及失效通道

执行 ComplieDlg.exe 文件，弹出如下（图 3-1）对话框，具体功能如下：

图 3-1

2．2．1 编译图形

在文件名编辑框中填写要编译的图形文件的完全路径，或单击 ... ，弹出文件对话框，查找扩展名为. mdv 的图形文件。单击 编译(C) 执行。如出错，弹出信息窗口，提示出错行及错误信息。

2．2．2 标记失效位

在文件名编辑框中填写图形文件的完全路径，或单击 ... ，弹出文件对话框，查找扩展名为 . mdv 的图形文件。单击 标记失效位(M) ，弹出文件对话框，选择与图形文件对应的文件，扩展名为 . log 文件，选择后执行。执行结果在扩展名为 . mdc 的图形源文件中，在失效行的下一行标记失效标记“？”，对应通道标记通道失效标记。

2．2．3　设置路径

单击菜单设置－＞设置路径，弹出如下（图 3-2）窗口，选择默认的路径，单击 确定(O) 按钮。当选择图形文件时，自动到默认路径下。

图 3-2

3. 图形文件格式

源文件：ASCII 文件 .mdc 文件

3．1 第一行：

MEM_SOURCE_15; --------第一行必须以“MEM_SOURCE_15;”开始，作为图形文件的标志

3．2 管脚定义：

PINDEF: -------- 管脚定义开始。

〈管脚名称〉＝〈 I | O | IO 〉,〈 BIN 〉,(通道号)
-------- 用二进制方式编写图形的管脚定义

〈管脚组名称〉(数值..数值) ＝〈 I | O | IO 〉,〈 HEX 〉,(通道号)
------- 用十六进制方式编写图形的管脚定义

I： 表示为输入管脚
O： 表示为输出管脚
IO： 表示为输入/输出管脚

[范例]：

```
OUTP        = O , BIN , ( 9 )
IN1         = I , BIN , ( 10 )
DATA(0..7)  = IO, HEX , ( 8,7,6,5,4,3,2,1 )
ADDR(0..5)  = I , HEX , ( 12,13,14,18,19,20 )
```

3．3 管脚到通道定义

PIN_TO_CHANNEL: --------从第一列开始，在下一行开始编辑管脚与通道对应关系

[范例]：

1 = 7

2..10 = 48..40

管脚与通道对应关系语句由两部分组成，第一部分是管脚号，第二部分是通道号，用 “=”号分开，可以有两种格式，如范例。

注意：必须一一对应，管脚数与通道数要相等。

3．4 图形指令及数据段标记

MAIN_F: -------- 必须从第一列开始，图形指令及数据段开始标志，表示下面的语句是指令和数据。

图形指令由多个指令段组成，每段由 START_INDEX()开始，HALT（图形）结束。

图形指令有四种格式。

	指令	（图形）
标号	指令	（图形）
	指令，参数	（图形）
标号	指令，参数	（图形）

注： 无标号时第 1 列必须为空格

标号必须从第 1 列开始

指令与图形之间、参数与图形之间必须空格

3．5 起始点定义

START_INDEX（起始点标号） -------- 必须从第一列开始，起始点标号可为 0 至 47 间的一个数字。起始点标号用于在执行图形时，标明执行哪一段图形。在 Visual C++中编写测试程序时，在调用函数 RUN_PATTERN( )时，第 1 个参数就是起始点标号，与此定义相同，相互对应使用。

3．6　图形结束

END.	--------　必须从第 1 列开始，标志图形文件结束。

3．7　注释

{ }	--------　注释一行
//	--------　注释一行或在一条指令的结尾注释

4.　图形指令及数据格式说明

4.1 图形指令

INC	顺次走一步，执行一次图形
RPT,n	重复送该图形 n+1 次，n 最大不能超过 4095
LDC,n	为 LOOP 循环指令定义循环次数，n 是 LOOP 指令的循环次数，最大值不能超过 4095，最多可嵌套装三次 n 值
LOOP,标号	如 LDC 装入的 n 值不等于 0:　n 减 1,跳转到标号. 如 LDC 装入的 n 值等于 0:　顺次往下走,LOOP 指令支持 3 层嵌套.
GOTO,标号	跳转到标号
GONP,标号	从标号开始，如失效：跳转到标号，如不失效,到下一图形。
LDF	动态测量时的标志，与 JMP 配合使用，
JMP,标号	动态测量时，从标号开始，到 JMP 语句循环执行。
HALT	送该图形后停止,图形发生器工作结束,每个起始点 START_INDEX 对应 1 个 HALT.

注意：图形指令的第一列必须是空格
　　标号必须从第一列开始，不能空格

2．数据格式

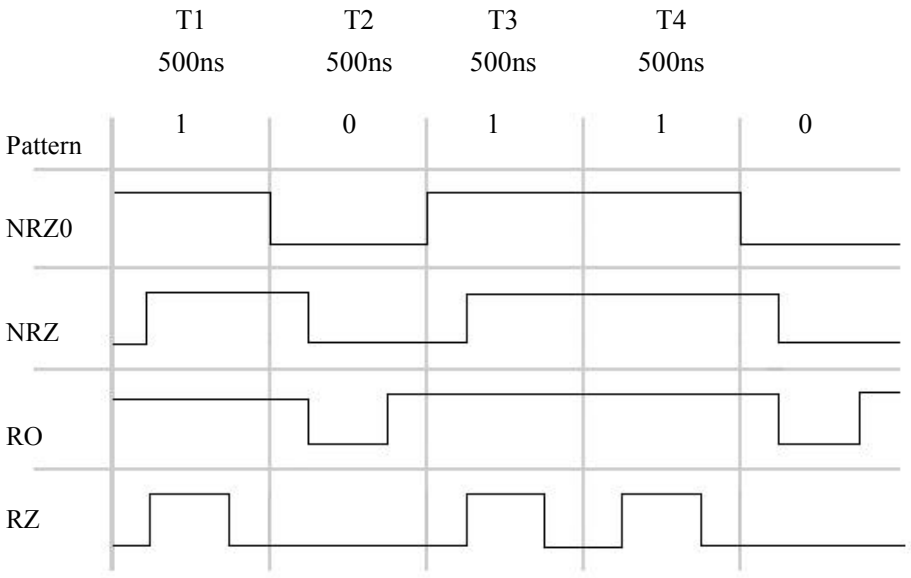
0、1:	二进制方式，0 为输入低，1 为输入高
L、H:	二进制方式， L 为输出低，H 为输出高
T:	十六进制方式，表示后面数据为输出
X:	表示对应的通道不驱动，不测试

5、 波形驱动格式说明

JC-5600 集成电路测试系统的数字通道板可以对每一个通道设置驱动波形，当设置某一个通道时，不改变其他通道的驱动格式。JC-5600 测试系统支持四种波形驱动格式：

NRZ	不归零方式
NRZ0	满周期
RO	归一格式
RZ	归零格式

详细波形格式如下图：



[图形文件范例]

74LS138.mdc 图形文件

```
MEM_SOURCE_15;

PINDEF
   A       = I, BIN, (2)
   B       = I, BIN, (3)
   C       = I, BIN, (4)
   GA      = I, BIN, (5)
   GB      = I, BIN, (6)
   G1      = I, BIN, (7)
   Y7      = O, BIN, (8)
   Y6      = O, BIN, (9)
   Y5      = O, BIN, (10)
   Y4      = O, BIN, (11)
   Y3      = O, BIN, (12)
   Y2      = O, BIN, (13)
   Y1      = O, BIN, (14)
   Y0      = O, BIN, (15)

MAIN_F

START_INDEX(0)   {FUN}
      inc           (XXXXXXXXXXXXXX)
      inc           (111111HHHHHHHH)
      inc           (000000HHHHHHHH)
      inc           (000001HHHHHHHL)
      inc           (100001HHHHHHLH)
      inc           (010001HHHHHLHH)
      inc           (110001HHHHLHHH)
      inc           (001001HHHLHHHH)
      inc           (101001HHLHHHHH)
      inc           (011001HLHHHHHH)
      inc           (111001LHHHHHHH)
      inc           (000010HHHHHHHH)
      inc           (111001LHHHHHHH)
      inc           (010101HHHHHHHH)
      inc           (011001HLHHHHHH)
      inc           (111010HHHHHHHH)
      inc           (101001HHLHHHHH)
```

```
        inc             (101000HHHHHHHHH)
        inc             (101001HHLHHHHH)
        inc             (010011HHHHHHHHH)
        inc             (001001HHHLHHHH)
        inc             (100001HHHHHHLH)
        inc             (001001HHHLHHHH)
        inc             (000001HHHHHHHL)
        inc             (XXXXXXXXXXXXXX)
        inc             (100000XXXXXXXX)
        inc             (010000XXXXXXXX)
        inc             (001000XXXXXXXX)
        inc             (000100XXXXXXXX)
        inc             (000010XXXXXXXX)
        inc             (000001XXXXXXXX)
        inc             (XXXXXXXXXXXXXX)
        halt            (XXXXXXXXXXXXXX)
START_INDEX(1)
        INC             (XXXXXXXXXXXXXX)
        INC             (111111XXXXXXXX)
        HALT            (111111XXXXXXXX)
START_INDEX(2)
        inc             (XXXXXXXXXXXXXX)
        inc             (111111HHHHHHHH)
        inc             (000000HHHHHHHH)
        inc             (000100XXXXXXXX)
        inc             (000010XXXXXXXX)
        inc             (000001XXXXXXXX)
        inc             (XXXXXXXXXXXXXX)
        halt            (XXXXXXXXXXXXXX)
START_INDEX(3)
        INC             (XXXXXXXXXXXXXX)
        INC             (000000XXXXXXXX)
        HALT            (000000XXXXXXXX)
START_INDEX(4)  VOL
        INC             (XXXXXXXXXXXXXX)
        INC             (000001HHHHHHHL)
        halt            (000001HHHHHHHX)
START_INDEX(5)
        INC             (XXXXXXXXXXXXXX)
        INC             (100001HHHHHHLH)
        halt            (100001HHHHHHXH)
START_INDEX(6)
        INC             (XXXXXXXXXXXXXX)
        INC             (010001HHHHHLHH)
```

```
      halt          (010001HHHHHHXHH)
START_INDEX(7)
      INC           (XXXXXXXXXXXXXX)
      INC           (110001HHHHLHHH)
      halt          (110001HHHHXHHH)
START_INDEX(8)
      INC           (XXXXXXXXXXXXXX)
      INC           (001001HHHLHHHH)
      halt          (001001HHHXHHHH)
START_INDEX(9)
      INC           (XXXXXXXXXXXXXX)
      INC           (101001HHLHHHHH)
      halt          (101001HHXHHHHH)
START_INDEX(10)
      INC           (XXXXXXXXXXXXXX)
      INC           (011001HLHHHHHH)
      halt          (011001HXHHHHHH)
START_INDEX(11)
      INC           (XXXXXXXXXXXXXX)
      INC           (111001LHHHHHHH)
      halt          (111001XHHHHHHH)
END.
```

24C02.mdc 图形文件(节选)

```
MEM_SOURCE_15;

PINDEF
A2       =I,BIN,(3)
A1       =I,BIN,(2)
A0       =I,BIN,(1)
SCL      =I,BIN,(6)
SDA      =IO,BIN,(5)
WP       =I,BIN,(7)

//         SDA---10K---VIS2

START_INDEX(21)   { iccw }
                   {AAACDW}
                   {210LAP}

        INC         (000000)
        INC         (000100)
```

```
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100) //ADDRESS 1--8

        INC         (0000L0)
        INC         (0001L0)
        LDF         (0001L0)

JW      INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //DATA 1010101010

        INC         (0000L0)
        JMP,JW      (0001L0)

        INC         (0000L0)
        HALT        (0001L0)

START_INDEX(22)   { iccw }
                  {AAACDW}
```

```
                    {210LAP}

        INC         (000010)
        LDF         (000010)
JWW     INC         (000010)
        INC         (000100)
        INC         (000100) //START
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //1010
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)//000
        INC         (000000)
        INC         (000100) //WRITE

        JMP,JWW     (0000L0)
        INC         (0001L0)//ACK
        HALT        (0000L0)

START_INDEX(23)   { iccr }
                    {AAACDW}
                    {210LAP}

        INC         (000010)
        INC         (000010)
        INC         (000100)
        INC         (000100) //START
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //1010
```

```
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)//000
        INC         (000000)
        INC         (000100) //WRITE

        INC         (0000L0)
        INC         (0001L0)//ACK

        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100) //ADDRESS 1--8

        INC         (0000L0)
        LDC,7       (0001L0)

JR1     INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
```

```
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //DATA 1010101010

        INC         (0000L0)
        LOOP,JR1    (0001L0)

        INC         (000000)
        INC         (000000)
        INC         (000110)
        INC         (000110)//STOP

        LDC,1299    (000000)
JR2     INC         (000100)
        LOOP,JR2    (000000)

        INC         (000010)
        INC         (000010)
        INC         (000100)
        INC         (000100) //START
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //1010
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)//000
        INC         (000000)
        INC         (000100) //WRITE

        INC         (0000L0)
        INC         (0001L0)//ACK

        INC         (000000)
        INC         (000100)
        INC         (000000)
```

```
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100) //ADDRESS  1--8

        INC         (0000L0)
        INC         (0001L0)

        INC         (000010)
        INC         (000010)
        INC         (000100)
        INC         (000100) //START
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //1010
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)//000
        INC         (000010)
        INC         (000110) //READ

        INC         (0000L0)
        INC         (0001L0)
        INC         (0001L0)

JR3     INC         (0000X0)
        INC         (0001X0)
```

```
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0)
        INC         (0000X0)
        INC         (0001X0) //DATA 1010101010

        INC         (0000L0)
        JMP,JR3     (0001L0)

        INC         (000000)
        HALT        (000100)

START_INDEX(24)     { vol }
                    {AAACDW}
                    {210LAP}
        INC         (000010)
        INC         (000010)
        INC         (000100)
        INC         (000100) //START
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100)
        INC         (000010)
        INC         (000110)
        INC         (000000)
        INC         (000100) //1010
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)
        INC         (000000)
        INC         (000100)//000
        INC         (000000)
        INC         (000100) //WRITE
```

```
        INC         (0000L0)
        HALT        (0001L0)//ACK
START_INDEX(25)   { IIL }
                    {AAACDW}
                    {210LAP}

        INC         (000000)
        INC         (000000)
        INC         (000000)
        INC         (000000)
        HALT        (000000)

START_INDEX(26)   { IIH }
                    {AAACDW}
                    {210LAP}

        INC         (000111)
        INC         (000111)
        INC         (000111)
        INC         (000111)
        HALT        (000111)

START_INDEX(27)   { ILO-ISB}
                    {AAACDW}
                    {210LAP}

        INC         (0000X0)
        INC         (0000X0)
        INC         (0000X0)
        INC         (0000X0)
        HALT        (0000X0)


START_INDEX(28)   { IHO }
                    {AAACDW}
                    {210LAP}

        INC         (0001X1)
        INC         (0001X1)
        INC         (0001X1)
        INC         (0001X1)
        HALT        (0001X1)
END.
```

1、数字通道部分指令

1．1 SET_ CHANNEL_RELAY

[函数原形]

void SET_CHANNEL_RELAY (CString csRelay)

[功能]

闭合指定的通道继电器，其他通道继电器打开

[参数说明]

csRelay ：通道继电器号，为一串整数字符或一串由“，-”连接的整数范围，中间不能有空格。

[范例]

设置通道继电器 1，3，5，6，7，8 合上，其他通道继电器打开

SET_CHANNEL_RELAY (“1，3，5，6，7，8”)；或

SET_CHANNEL_RELAY (“1，3，5-8”)；

设置通道继电器 1，2 后，再设置通道继电器 1，3，5，6，7，8

SET_ CHANNEL_RELAY (“1，2”)； //1，2 通道继电器闭合

SET_ CHANNEL_RELAY (“1，3，5-8”)；//1，3，5-8 通道继电器闭合，2 通道继电器打开

1．2 CLOSE_ CHANNEL_RELAY

[函数原形]

void CLOSE_ CHANNEL_RELAY (CString csRelay)

[功能]

闭合指定通道继电器，其他通道继电器保持原状态

[参数说明]

csRelay ：通道继电器号，为一串整数字符或一串由“，-”连接的整数范围，中间不能有空格。

[范例]

闭合通道继电器 1，3，4，5，8，9，10，再闭合 12

SET_ CHANNEL_RELAY (“1，3-5，8-9”)；

CLOSE_ CHANNEL_RELAY (“12”)； //已经闭合的通道继电器 1，3，4，5，8，9，10 保持闭合状态，12 通道继电器闭合。

1．3 OPEN_ CHANNEL_RELAY

[函数原形]

void OPEN_ CHANNEL_RELAY(CString csRelay)

[功能]

打开指定通道继电器

[参数说明]

csRelay ：通道继电器号，为一串整数字符或一串由“，-”连接的整数范围，中间不能有空格。

[范例]
设置 2，3，7，8，9 通道继电器后，打开 3 通道继电器
SET_ CHANNEL_RELAY (“2，3，7-9”)； // 2，3，7，8，9 通道继电器闭合
OPEN_ CHANNEL_RELAY (“3”)； //2，7，8，9 通道继电器保持闭合状态,3 通道继电器打开。

1．4 CLEAR_ CHANNEL_RELAY

[函数原形]
void CLEAR_ CHANNEL_RELAY()
[功能]
打开所有通道继电器

1．5 SET_INPUT_LEVEL

[函数原形]
void SET_INPUT_LEVEL (double Vih , double Vil)
[功能]
设置输入比较电平值。
[参数]
Vih : 输入高电平, 单位 V，范围：0V~+7.8V
Vil : 输入低电平, 单位 V，范围：0V~+7.8V
[范例]
设置输入电平，低电平 0.8 V，高电平 3V
SET_INPUT_LEVEL (3，0.8)

1．6 SET_OUTPUT_LEVEL

[函数原形]
void SET_OUTPUT_LEVEL (double Voh, double Vol)
[功能]
设置输出比较电平。
[参数]
Voh : 输出高电平, 单位 V，范围：0V~+7.8V
Vol : 输出低电平, 单位 V，范围：0V~+7.8V
[范例]
设置输出比较电平，高电平 5V，低电平 3V
SET_OUTPUT_LEVEL (5，3)

1．7 SET_PERIOD

[函数原形]

void SET_PERIOD (unsigned int period)

[功能]

在执行图形时，设置一个测试周期时间长度。

[参数说明]

period ：时钟周期, 单位 ns，范围：大于 100ns ，小于 10ms

[范例]

设置始终周期为 500ns

SET_PERIOD (500)

1．8 SET_TIMING

[函数原形]

void SET_TIMING (double LeadEdge,double Fwidth,double Ctg)

[功能]

设置波形格式时间。

[参数说明]

LeadEdge ：前沿，设定波形格式开始时间，单位 ns，范围：大于 10ns

Fwidth ：脉宽，设定波形的宽度，单位 ns，范围：大于 20 ns

Ctg ：选通，设定测量取样时间，单位 ns，范围：大于 10ns

[范例]

设置时间前沿为 30ns，脉冲宽度 100ns，选通为 120ns

SET_TIMING (30,100,120)

1．9 FORMAT

[函数原形]

void FORMAT (BYTE fmt，Cstring csChannel)

[功能]

格式化通道，在驱动图形方式时，选用此格式驱动通道。当前设置只影响设置的通道，不改变其他通道的格式化方式，可选择四种格式，具体波形请参照图形说明。

[参数]

fmt ：格式化模式

NRZ ------ 非归零

RO ------- 归一

RZ ------- 归零

NRZ0 -- 非归零

CsChannel: 通道号数据, 是一串包含整数，“，-”分隔符的字符串，最多 64 个通路, 通道编号 1-64

[范例]

格式化通道 1，2，3，4，7 为 NRZ0 方式，再格式化通道 1，3，6 为 RZ 方式

FORMAT (NRZ0，“1-4,7”) //格式化 1-4，7 通道为 NRZ0 方式

FORMAT (RZ，“1,3,6”) //将 1，3 通道格式化方式改为 RZ 方式

1．10 BOOL RUN_PATTERN

[函数原形]

BOOL RUN_PATTERN (int start_idx, int get_fail, int apgen , int time_range, soft_bin)

BOOL RUN_PATTERN (CString csItem , int start_idx, int get_fail, int apgen，int time_range, int soft_bin)

[功能]

运行图形，返回 PASS 或 FAIL。

第一种格式：只运行图形，不在显示设备上显示

第二种格式：用于测试功能，测试完成后，显示

[参数]

csItem ：测试项目名称，用于功能测量时使用

start_idx ：运行图形的索引号，指定执行哪一段图形

get_fail ：测试模式选择

0 或 GO ------- 失效后继续执行

1 或 NOGO ------- 失效返回

apgen ：固定为 0.

time_range ：设置图形运行的时间长度，单位 ms ，超时自动退出，返回测试失效。用于 GONP 执行匹配时使用。0 表示时间范围不限制。

soft_bin ：软件分箱号。

[返回值]

返回失效值　　0：对应通道 FAIL

1：对应通道 PASS

[范例]

执行图形，索引号为 0，失效返回，不使用 APGEN 方式，不设运行时间范围

RUN_PATTERN (0，1，0，0，1)

执行功能测量，测试项名称“Function”，索引号为 0，失效返回，不使用 AGPEN 方式，不设运行时间范围

RUN_PATTERN (“Function” 0，1，0，0，1)

注：在运行图形前，必须先设时钟周期，前沿，后沿，选通，否则不能运行

1．11　READ_PC

[函数原形]

DWORD　　READ_PC(　)

[功能]

读当前图形指针 PC 的值

1．12　PMU_CONDITIONS

[函数原形]

void PMU_CONDITIONS (unsigned int Mode, double Value, unsigned int Value_Unit,double Clamp_Value,unsigned int Clamp_Unit);

[功能]

设置数字通道精密测量单元 PMU 的测试条件，在使用 PMU 测量前先设置好，选用不同的模式，决定 PMU 测量方法。箝位值作为限流或限压的保护值，并且根据箝位值自动选择测量量程。

[参数]

Mode ：　测量模式,有 2 种选择:　FVMI ----- 加压测流

FIMV ----- 加流测压

Value：　施加值范围　电压：±8V

电流：±250mA

Value_Unit：　施加单位　FVMI 方式，单位可以选 V 或 MV

FIMV 方式，单位可以选 MA 或 UA

Clamp_Value： 箝位值范围 电压：0 ~8V

电流：0 ~250mA

Clamp_Unit： 箝位单位 FVMI 方式，电流箝位, 单位可以选 MA 或 UA

FIMV 方式，电压箝位, 单位可以选 V 或 MV

[范例]

设置 PMU 测量模式为加压测流方式，施加电压 5V，箝位电流 10mA。

PMU_CONDITIONS (FVMI，5，V，10，mA)

1．13 PMU_MEASURE

[函数原形]

double PMU_MEASURE (unsigned int pin,unsigned int tDelay);

[功能]

使用 PMU 测量直流参数。直接测量单一通道，返回测量值

[参数]

pin ：通道号，范围：1 ~ 32

tDelay ：测量延迟时间, 单位毫秒

[返回值] 测量值,电压单位为 V，电流单位为 A.

[范例]

加压测流，测量通道 3 上器件管脚的电流，PMU_CONDITIONS 设置 PMU 测量条件，PMU_MEASURE 函数测量通道 3 上器件管脚的电流，延时 10ms，函数返回值为测量值。

double Result;

PMU_CONDITIONS (FVMI，5，V，10，mA)；

Result = PMU_MEASURE (3,10)；

注：在使用次函数前，必须先设 PMU_CONDITIONS 函数

1．14 BOOL RUN_PATTERN_P

[函数原形] BOOL RUN_PATTERN_P(int start_idx,int get_fail,int apgen,int time_range,double *fValue)

[功能]

运行图形，返回 PASS 或 FAIL。

[参数]

csItem ：测试项目名称，用于功能测量时使用。

start_idx ：运行图形的索引号，指定执行哪一段图形。

get_fail ：测试模式选择

0 或 GO ------- 失效后继续执行

1 或 NOGO ------- 失效返回

apgen ：固定为 0。

time_range ：设置图形运行的时间长度，单位 ms ，超时自动退出，返回测试失效。用于 GONP 执行匹配时使用。0 表示时间范围不限制。

soft_bin ：软件分箱号。

double *fValue：失效数据返回指针，返回值大于 1 为失效，否则为合格。

[返回值]

返回失效值 >1：对应通道 FAIL

1：对应通道 PASS

[范例]

执行图形，索引号为 0，失效返回，不使用 APGEN 方式，不设运行时间范围

```
double  func1[4];
RUN_PATTERN_P(0,1,0,0,&func1[0]);
if(!SHOW_RESULT("FUN",func1,"_",1,No_LoLimit,5))
    return;
```

软件分箱号（上例为“**5**”）在显示指令里设置。

注：在运行图形前，必须先设时钟周期，前沿，后沿，选通，否则不能运行

1．15 PMU_CONDITIONS_P ( )

[函数原形]

void PMU_CONDITIONS_P (unsigned int Mode, double Value, unsigned int Value_Unit,double Clamp_Value,unsigned int Clamp_Unit);

［功能］设置数字通道精密测量单元 PMU 的多 SITE 测试条件，在使用 PMU 测量前先设置好，选用不同的模式，决定 PMU 测量方法。箝位值作为限流或限压的保护值，并且根据箝位值自动选择测量量程。

［参数］

Mode ： 测量模式,有 2 种选择: FVMI ----- 加压测流

FIMV ----- 加流测压

Value： 施加值范围 电压：±8V

电流：±250mA

Value_Unit： 施加单位 FVMI 方式，单位可以选 V 或 MV

FIMV 方式，单位可以选 MA 或 UA

Clamp_Value： 箝位值范围 电压：0 ~8V

电流：0 ~250mA

Clamp_Unit： 箝位单位 FVMI 方式，电流箝位, 单位可以选 MA 或 UA

FIMV 方式，电压箝位, 单位可以选 V 或 MV

[范例]

设置 PMU 测量模式为加压测流方式，施加电压 5V，箝位电流 10mA。

PMU_CONDITIONS_P (FVMI，5，V，10，mA)

1．16 PMU_MEASURE_P ( )

[函数原形]

double PMU_MEASURE_P (unsigned int pin,unsigned int tDelay);

[功能]使用数字通道精密测量单元 PMU 双 SITE 测量直流参数。双 SITE 分别测量各自板上的同一通道，返回测量值。测量值要从 GET_DATA 函数获得。测量值,电压单位为 V，电流单位为 A..

[参数]

pin ：通道号，范围：1 ~ 16

tDelay ：测量延迟时间, 单位毫秒

[返回值] 无

[范例]

加压测流，测量通道 3 上器件管脚的电流，PMU_CONDITIONS 设置 PMU 测量条件，PMU_MEASURE_P 函数同时测量两个通道 3 上器件管脚的电流，延时 10ms，函数返回值从 GET_DATA 函数获得。

```
double  ret [10];
int i;
PMU_CONDITIONS_P (FVMI，5，V，10，mA);
PMU_MEASURE (3,10);
for(i=0;i<iNum;i++)
        ret [i]=GET_DATA(i)*1e3;
```

注：在使用次函数前，必须先设 PMU_CONDITIONS_P 函数