3196D 数模测试系统

编程手册

文件编号：XR8002-YH-02

版 本 号：2.2

编写：周卫、王斌 日期：2005-08-10

审核：周 卫 日期：2005-11-09

批准：周 卫 日期：2005-11-09

北京新润泰思特测控技术有限公司

目　　录

目　　录 - 1 -
第一章 系统类型说明 - 3 -
第二章 函数详细说明 - 5 -
2.1 高压PMU函数说明 - 5 -
2.2 通道PMU函数说明 - 7 -
2.2.1 操作方式 - 7 -
2.2.2 函数详细说明 - 8 -
2.3 数字部分函数说明 - 12 -
2.4 与被测器件管脚对应函数说明 - 20 -
2.5 控制位控制函数说明 - 22 -
2.6 TMU及高精度电压源/表函数说明 - 23 -
2.7 波形产生及波形分析函数说明 - 26 -
2.8 矩阵控制函数 - 27 -
第三章 算法图形及指令 - 28 -
3.1 算法图形指令含义说明 - 28 -
3.1.1 算法序列控制指令 - 28 -
3.1.2 XY地址AB寄存器算法指令 - 29 -
3.1.3 XY地址算术逻辑单元操作指令以及示例 - 29 -
3.1.4 XY地址输出以及算法数据控制组合指令 - 29 -
3.2 完整指令示例以及说明 - 30 -
3.2.1 清存储器（XY地址复用）512KB - 30 -
3.2.2 清存储器（XY地址非复用）512KB - 30 -
3.2.3 算法指令与算法序列指令的执行顺序 - 30 -
3.3 图形编辑界面图 - 31 -
第四章 数字芯片交流参数测试 - 32 -
4.1 交流参数测试的系统限制 - 32 -
4.2 器件输出延时Tpd的测试方法 - 33 -
4.2.1 测试Tpd的步骤 - 33 -
4.2.2 注意事项 - 34 -
4.2.3 测试实例 - 34 -
4.2.4 测试图形 - 35 -
4.3 器件建立保持时间的测试方法 - 36 -
4.3.1 建立时间的测试步骤 - 36 -
4.3.2 保持时间的测试步骤 - 37 -
4.3.3 注意事项 - 37 -
4.3.4 建立时间测试示意 - 37 -
4.3.5 建立时间测试代码 - 38 -
4.3.6 建立时间测试图形 - 38 -
4.3.7 保持时间Th测试示意图 - 39 -
4.3.8 保持时间测试Th测试代码 - 40 -
附录A:存储器测试图形说明 - 41 -

A.1 全 0 全 1①..... - 41 -
A.2 校验板（棋盘格图形，Checkerboard Pattern） ②..... - 41 -
A.3 地址互补图形③..... - 41 -
A.4 步进法（Marching pattern）④..... - 41 -
A.5 走步（walking pattern）⑤..... - 42 -
A.6 走步（simple walking pattern）⑤..... - 42 -
A.7 蝴蝶图案(butterfly pattern)⑥..... - 42 -
A.8 IMAG Pattern⑨..... - 42 -
A.9 乒乓（简化跳步图形）⑩..... - 43 -
A.10 行列乒乓（C/R Galloping pattern）⑧..... - 43 -

第一章 系统类型说明

（1）enum PINTYPE { PT_NF, PT_RTZ, PT_RTO, PT_SBC, PT_NRZ, PT_MCH, PT_ZRO, PT_ONE, PT_EDGE, PT_WINDOW};

管脚驱动比较类型定义。设置时序函数时使用。具体含义分别为：不格式化、归零格式化、归一格式化、环绕补码格式化、不归零格式化、曼彻斯特格式化、恒 1 格式化、恒 0 格式化、边沿比较格式、窗口比较格式。

（2）enum APGTYPE {AT_APGX, AT_APGY, AT_APGD, AT_PTN};

算法图形资源类型定义，分配算法图形资源到管脚时使用。具体含义分别为：算法 X 地址、算法 Y 地址、算法数据和普通图形通道。3196D 和 3168 系统不提供算法数据资源，即 APGD 无效。

（3）enum PINSTATE {PS_ADD,PS_NEW,PS_OFF};

管脚状态类型，增加连接管脚、新设连接管脚和断开管脚三个取值，所有连接通道或器件管脚的函数都必须使用。

（4）enum STOPTYPE {ST_HALT=(int)0,ST_FAIL=(int)1, ST_ADDR=(int)2, ST_STEP=(int)4,ST_BYPASS=(int)8};

图形停止方式类型定义，有仅指令停止（ST_HALT）、失效停止（ST_FAIL）、指定地址停止（ST_ADDR）、单步运行（ST_STEP）和自由运行（ST_BYPASS）方式。BYPASS 模式，图形被启动后自由运行，系统可以得到控制权。若要结束图形，使用 reset()命令即可。

（5）struct SYSRESULT

```
{
   unsigned short error ;
   unsigned short igroup[6];
   short          index[129];
   double         measval[128]
};
```

函数返回值结构定义，该结构包含了五个变量，用于返回函数的多种操作结果，具体说明如下：

- error 用于返回函数的操作正确或错误状态，函数操作正确，返回 0，否则为 1
- unsigned short igroup 用于返回整型字节数据数组
- short index[]用于索引测量值
- double measval[]用于返回多 pin 的测量值，如 CPMU 测流或测压值

使用该结构的函数，并不是使用结构的所有返回值，使用统一的结构主要是方便用户记忆，具体的含义由具体函数说明。

示例：

```
SYSRESULT  result;
result=pin_fimv("13,17-19,22",10,100);

for(i=0;i<5;i++)
   testvalue[i]=result.measval[result.index[i]];
   //读取测压值
```

（6）enum CIRANGE { MI_ AUTO=-1, MI_4uA=1, MI_40uA, MI_400uA, MI_4mA, MI_40mA, MI_400mA }

通道 PMU 电流量程选择：自动量程（MI_ AUTO）、最大电流 4uA（MI_4uA）、最大电流 40uA（MI_40uA）、最大电流 400uA（MI_400uA）、最大电流 4mA（MI_4mA）、最大电流 40mA（MI_40mA）、最大电流 400mA（MI_400mA）。

（7）enum HVMIRANGE { MI_ AUTO=0, MI_1A=1, MI_100mA=2, MI_10mA=3, MI_1mA=4, MI_100uA=5, MI_10uA=6, MI_1uA=7 }

高压 PMU 电流量程选择，自动量程（MI_ AUTO）、最大电流 1A（MI_1A）、最大电流 100mA（MI_100mA）、最大电流 10mA（MI_10mA）、最大电流 1mA（MI_1mA）、最大电流 100uA（MI_100uA）、最大电流 10uA（MI_10uA）、最大电流 1uA（MI_1uA）。

（8）enum HVMVRANGE { MV_ AUTO=0, MV_100V=1, MV_10V=2, MV_1V=3, MV_100mV=4 }

高压 PMU 电压量程、DVM 量程选择。具体为：自动量程（MV_AUTO）、最大测压 100V（MV_100V）、最大测压 10V（MI_10V）、最大测压 1V（MV_1V）、最大测压 100mV（MV_100mV）。

（9）enum TMUCMP { TMU_X=0X0000, TMU_ZERO=0X0004, TMU_ONE=0X0007, TMU_UPEDGE=0X0006, TMU_DOWNEDGE=0X0005 }

TMU 通道触发条件：不关心（TMU_X）、逻辑为 0 触发（TMU_ZERO）、逻辑为 1 触发（TMU_ONE）、上升沿触发（TMU_UPEDGE）、下降沿触发（TMU_DOWNEDGE）。

（10）struct TMUSET

```
{
    double dVoltage1;
    double dVoltage2;
    unsigned short nCount;
    unsigned long nDiv;
    TMUCMP eCondition1[2];
    TMUCMP eCondition2[2];
    unsigned short nInput;
};
```

TMU 工作模式设置结构定义该结构包含了七个变量，用于设置 TMU 的多种工作模式，具体说明如下：

- dVoltage1、dVoltage2 设置比较器的信号反转电平
- nCount 设置重复测试周期数
- nDiv 设置分辨率
- eCondition1[2] 设置 1 通道触发条件,[0]为起始条件,[1]为终止条件
- eCondition2[2] 设置 2 通道触发条件,[0]为起始条件,[1]为终止条件
- nInput 设置输入信号处理方式，即被测信号是否经过比较器调理

第二章 函数详细说明

用户函数分为 6 类：高压 PMU 控制函数、通道 PMU 控制函数、数字通道控制函数、控制位控制函数、TMU 控制函数和 WGWA 控制函数。

2.1 高压 PMU 函数说明

高压 PMU 控制函数共 7 个，详细介绍如下：

（1） short hvforce_v(unsigned short pmu,double volt,double clamp,HVMIRANGE range=MI_ AUTO)

函数用途：高压PMU施加设定电压值。

参数说明：unsigned short pmu，所选PMU序号1~10；

double volt，施加电压值,单位 V；

double clamp，设置最大允许输出电流（嵌位电流）<1A；

HVMIRANGE range，选择测流档位，缺省值 MI_1A，测流档位同时表示允许输出电流能力。缺省值 MI_AUTO 使用嵌位电流所在档位。

返回值：0，函数被正确执行；1，函数执行出错，通常有对话框弹出。

（2） short hvforce_i(unsigned short pmu,double current,double limit_v)

函数用途：高压PMU施加设定电流值。

参数说明：unsigned short pmu，所选PMU序号1~10；

doulbe current，施加电流值，单位 mA；

double limit_v，设置最大允许输出电压小于 30V；

返回值：0，函数被正确执行；1，函数执行出错，通常有对话框弹出。

（3） double hvfvmi(unsigned short pmu,double volt,double lim_i,
unsigned short delay=2,HVMIRANGE range=MI_AUTO)

函数用途：施加设定电压值，并测量电流

参数说明：unsigned short pmu，所选PMU序号1~10；

double volt，施加电压值,单位 V；

doulbe lim_i，设置最大允许输出电流（嵌位电流，小于 1A）；

unsigned short delay，设置测量前的延迟时间；

HVMIRANGE range，选择测流档位，缺省值 MI_AUTO 使用嵌位电流所在档位；

返回值：测量结果，单位mA

（4） double hvfimv(unsigned short pmu,double current,double lim_v,unsigned short delay=2)

函数用途：施加设定电流值，并测量电压

参数说明：unsigned short pmu，所选PMU序号1~10；

double current，施加电流值,单位 mA；

double lim_v，设置最大允许输出电压，小于 30V

unsigned short delay，设置测量前的延迟时间；

返回值：测量结果，单位V。

（5） *double hvmeasure_dvm(unsigned short pmu,HVMVRANGE range)*

函数用途：测量所选通道电压
参数说明：unsigned short pmu，所选PMU序号1~10；
HVMVRANGE range，选择测压量程，枚举类型；
返回值：测量结果，单位V。

（6） *double hvmeasure_diff(unsigned short pmu,HVMVRANGE range)*

函数用途：测量所选通道所在PMU板两通道间差分电压
参数说明：unsigned short pmu，所选PMU序号1~10；
HVMVRANGE mv_range 选择测压量程，枚举类型；
返回值：测量结果，单位V。

（7） *short hvpmu_rst(unsigned short pmu)*

函数用途：断开高压PMU的程控源输出，同时复位高压PMU。
参数说明：unsigned short pmu，所选PMU序号1~10；
返回值：无。

（8）*double hvadd_ripple(unsigned short pmu, double v_rms, unsigned short freq);*

函数用途：高压 PMU 输出纹波。该纹波叠加在高压 PMU 加的直流电压之上。
参数说明：unsigned short pmu，所选PMU序号1~10；
double v_rms：纹波电压的有效值，电压范围：0～1V
unsigned short freq：纹波频率，单位Hz，0Hz～4000Hz
返回值：0，函数操作成功，否则，启动失败。
函数说明：启动纹波必须在PMU的加压模式有效，加流时，不可以使用。设置纹波电压必须在施加直流电压之后进行。停止纹波使用hvclr_ripple（）函数或者hvpmu_rst（）函数。

（9） *double hvclr_ripple(unsigned short pmu);*

函数用途：停止纹波输出。不影响直流电压输出。
参数说明：unsigned short pmu，所选PMU序号1~10；
返回值：无。

2.2 通道 PMU 函数说明

通道 PMU，可简称 CPMU，每通道板（16 数字通道）拥有一个 CPMU，它可直接对本板对应的 16 通道进行直流测试。各不同 CPMU 测试可同时进行。

2.2.1 操作方式

CPMU 的操作方式分三类：

以 CPMU 号（即板号）操作；

以数字通道号操作；

以被测器件引脚（由测试界面设置器件引脚与数字通道的映射关系）操作。

下面分类将函数功能简要说明：

- 以通道 PMU 号操作的函数：

short cpmu_rst(unsigned short pmu)；

//通道 PMU 复位

short cpmu_to_pin(unsigned short pmu, unsigned short pin, PINSTATE pinstate=PS_ADD)；

//设置通道 PMU 与通道继电器连接状态

short cforce_v(unsigned short pmu, double volt, double clamp,CIRANGE range=CMI_AUTO)；

//设置通道 PMU 加压

double cmeasure_i(unsigned short bpmu)；

//通道 PMU 测流

short cforce_i(unsigned short pmu, double current, double clamp)；

//通道 PMU 加流

double cmeasure_v(unsigned short pmu)；

//通道 PMU 测压

- 以通道顺序操作的函数：

SYSRESULT pin_fvmi(char pingrp, double volt, double limit_i, unsigned short delay=1, CIRANGE range=CMI_AUTO)；*

//以通道顺序多通道加压测流

SYSRESULT pin_fimv(char pingrp, double current, double limit_v, unsigned short delay=1)；*

//以通道顺序多通道加流测压

- 以被测引脚顺序操作的函数：

SYSRESULT dut_fvmi(char dutgrp, double volt, double limit_i, unsigned short delay=1, CIRANGE range=CMI_AUTO)；*

//以被测器件引脚顺序多通道加压测流

SYSRESULT dut_fimv(char dutgrp, double current, double limit_v, unsigned short delay=1)；*

//以被测器件引脚顺序多通道加流测压

2.2.2 函数详细说明

SYSRESULT dut_fimv(char dutgrp, double current, double limit_v, unsigned short delay＝1)；*

函数功能：按被测器件管脚顺序加流测压；

参数说明：char* dutgrp：管脚字符串数组，需加双引号；单管脚时可省略双引号；

double current：加流值， -360.0－360.0mA；

double limit_v：限压值，0－10V；

unsigned short delay：测试电路稳定延时，即管脚加压后与测量间隔；单位为毫秒，默认为 1ms；

返回值：系统结构变量；

若出错，则 result.error= -1； 否则为 0；

result.index[]为被测器件管脚索引，即 dutgrp 数组；

result.measval[]为按上索引对应的测量结果。

例：*res = dut_fimv(“1-5,6,20-26,34”, 5.0, 6.0, 20)；*

加流测压 1－5、6、20－26、34 管脚，加流值为 5mA，限压 6V，稳定延时为 20ms；

res = dut_fimv(1, 5.0, 6.0, 20)；

单管脚加流测压，省略双引号，其他同上。

SYSRESULT pin_fimv(char pingrp, double current, double limit_v, unsigned short delay＝1)；*

函数功能：按通道顺序加流测压

参数说明：char* pingrp：通道字符串数组，需加双引号；单通道时可省略双引号；

double current：加流值， -360.0－360.0 mA；

double limit_v：限压值，0－10V；

unsigned short dlelay：测试电路稳定延时，管脚加压后与测量间隔；单位为毫秒，默认为 1ms；

返回值：系统结构变量；

若出错，则 result.error= -1； 否则为 0；

result.index[]为被测通道索引，即 pingrp 数组；

result.measval[]为按上索引对应的测量结果。

例：*res = pin_fimv(“1-5,6,20-26,34”, 5.0, 6.0, 20)；*

加流测压 1－5、6、20－26、34 通道，加流值为 5mA，限压 6V,稳定延时为 20ms；

res = pin_fimv(1, 5.0, 6.0, 20)；

单通道加流测压，省略双引号，其他同上。

SYSRESULT dut_fvmi(char dutgrp, double volt, double limit_i, unsigned short delay=1, CIRANGE range=CMI_AUTO)；*

函数功能：按被测器件管脚顺序加压测流；

参数说明：char* dutgrp：管脚字符串数组，需加双引号；单管脚时可省略双引号；

doulbe volt：加压值， -10.0－10.0V；

double limit_i：限流值，0－360mA；

unsigned short delay：测试电路稳定延时，即管脚加压后与测量间隔；单位为毫秒，默认为 1ms；

返回值：系统结构变量；

若出错，则 result.error= -1，否则为 0；

result.index[]为被测器件管脚索引，即 dutgrp 数组；

result.measval[]为按上索引对应的测量结果。

说明：

限流值最好按负载情况来设置，这样减少 PMU 切换量程继电器次数，从而提高测试速度。

例：*res = dut_fimv(“1-5,6,20-26,34”, 5.0, 30., 20)；*

加压测流 1－5、6、20－26、34 管脚，加压值为 5V，限流 50mA, 量程为 360mA，稳定延时为 20ms；

res = dut_fimv(1, 5.0, 3.0, 20)；

单管脚加压测流，其他同上。

SYSRESULT pin_fvmi(char pingrp, double volt, double limit_i, unsigned short delay=1, CIRANGE range=CMI_AUTO)；*

函数功能：按被测通道顺序加压测流

参数说明：char* pingrp：通道字符串数组，需加双引号；单通道时可省略双引号；

double volt：加压值， -10.0－10.0；

double limit_i：限流值，0－360mA；

unsigned short delay：测试电路稳定延时，即通道加压后与测量间隔；单位为毫秒，默认为 1ms；

返回值：系统结构变量；

若出错，则 result.error= -1；否则为 0；

result.index[]为被测通道索引，即 pingrp 数组；

result.measval[]为按上索引对应的测量结果。

说明：限流值最好按负载情况来设置，这样减少 PMU 切换量程继电器次数，从而提高测试速度。

例： *res = pin_fimv(“1-5,6,20-26,34”, 5.0, 50.0, 20)；*

加压测流 1－5、6、20－26、34 通道，加压值为 5V，限流 50mA，稳定延时为 20ms；

res = pin_fimv(1, 5.0, 50.0, 20)；

单通道加压测流，其他同上。

short cpmu_rst(unsigned short pmu)；

函数功能：通道 PMU 复位；断开本通道 PMU 对应的通道继电器；PMU 工作模式恢复为默认；

参数说明：unsigned short pmu：通道 PMU 号，从 1 开始，每通道 PMU 对应 16 通道；

返回值： 0；

说明： 以通道板顺序操作通道 PMU，当完成功能后，必须加此函数复位。

例：*cpmu_rst(1)；* 复位通道 PMU 1；

short cpmu_to_pin(unsigned short pmu, unsigned short pin, PINSTATE pinstate=PS_ADD)；

函数功能： 连接/断开通道 PMU 与通道继电器

参数说明：unsigned short pmu：通道 PMU 号；

unsigned short pin：通道号，通道 PMU 对应的 16 个通道；

PINSTATE pinstate：设置通道继电器状态，枚举类型，PS_ADD 连接，PS_OFF 断开，默认为 PS_ADD。

返回值：参数设置错误返回 1；

说明： 参数设置错误弹出对话框提示。

例：*cpmu_to_pin(1, 1, PS_ADD)；*

连接通道 PMU 1 与通道 1 继电器；

cpmu_to_pin(1, 1, PS_OFF)；

断开通道 PMU 1 与通道 1 继电器；

short cforce_v(unsigned short pmu, double volt, double clamp, CIRANGE range=CMI_AUTO)；

函数功能： 通道 PMU 加压

参数说明：unsigned short pmu：通道 PMU 号；

double volt：加压值，-10.0 —10.0V；

double clamp：限流值，0 — 360mA；

返回值：参数设置错误返回-1；

说明：参数设置错误弹出对话框提示。

例：*cforce_v(1, 5.0, 10.0)；*

通道 PMU 1 加压 5V，限流 10mA。

double cmeasure_i(unsigned short pmu)；

函数功能： 通道 PMU 测流

参数说明：unsigned short pmu：通道 PMU 号；

返回值： 测试结果，单位 mA；

说明：

不能单独使用，与 cforce_v()配合使用，量程在该函数内按限流值设置，为了提高测试速度，应该一次设好。

例： *current = cmeasure_i (1)；*

通道 PMU 1 测流；

short cforce_i(unsigned short pmu, double current, double clamp)；

函数功能： 通道 PMU 加流

参数说明：unsigned short pmu：通道 PMU 号；

double current ：加流值，-360.0 —360.0mA；

double clamp：限压值，0－10V；

返回值： 参数设置错误返回-1；

例：*cforce_i(1, 5.0, 6.0)；*

通道 PMU 1 加流 5mA，限压 6.0V。

double cmeasure_v(unsigned short pmu)；

函数功能： 通道 PMU 测压

参数说明：unsigned short pmu：通道 PMU 号；

返回值：测试结果；

例： *volt = cmeasure_v (1)；*

通道 PMU 1 测压；

2.3 数字部分函数说明

数字部分操作函数共 31 个，比较多，但是对于初学者和多数的测试程序只需要 3－4 个常用函数即可以完成全部的操作。大部分的通道设置可以在图形编辑时通过参数设置对话框完成设置。只有当某个参数在测试过程中需要更改时，才需要用户在测试程序中使用用户函数完成设置。以下详细叙述数字部分操作函数，对于常用函数将使用蓝色底色标出。

（1） double set_clock_frequency(double freq,unsigned short cmpstart, unsigned short ptnstart=40);

函数用途：设置系统时钟设置频率，当启动硬图形时，必须先设置工作频率。

参数说明：double freq，系统工作频率，范围 33MHz～0.0016MHz。

unsigned short cmpstart：比较时钟时钟沿，默认为 30。

在用户单独编写 tpd 测试函数时应设置 cmpstart 为 20，测试后请重新设置为默认值 30。

返回值：返回实际的时钟值。系统时钟运行频率不是连续可调的，而是以 10ns 的分辨率可调，因而设定值与实际值会有误差。函数默认值为 1MHz，例如设置运行频率为 1MHz 可以使用以下两种写法：

set_clock_frequency(1);

set_clock_frequency();

（2） double set_clock9(double freq,unsigned short start,unsigned short stop, int runnow=0);

函数用途：设置时钟 9 的频率，前沿和后沿，以及是否单独启动。时钟 9 是一路特殊的用户时钟，它既可以和其他 8 路时钟一样配置，也可以单独配置和启动。时钟 9 的输出连接到了测试头，输出阻抗 50 欧姆，TTL 电平。

参数说明：double freq，预设的工作频率，范围 33Mhz～1.6KHz，分辨率 5ns

unsgined short start，设定时钟 9 前沿的位置，单位 ns

unsgined short stop，设定时钟 9 后沿的位置，单位 ns。前沿和后沿时间要小于一个时钟周期且大于等于 10ns。

int runnow，是否立即启动，runnow=1 则立即启动时钟，若取值为 0，则和其他时钟同时启动，runnow 默认值为 0。

返回值：实际设定的时钟值

说明：如果不需要单独使用时钟 9，则可以不使用该函数。

示例：`set_clock9(10,15,35,1);//`设置格式化时钟 9，工作频率 `10Mhz`，
`//`前沿开始时间为 `10ns`，高电平时间 `20ns`，立即运行。

（3） short set_ffmat_clock(clocksource, double start,double stop=0);

函数用途：设置格式化时钟的参数。时钟初始态均为 0，start指上升沿的开始时间，stop则指高电平的结束时间。如果stop＝0，则默认比较格式化时钟的占空比为 50％。格式化的参数必须在 10ns<stop-start<（T_{period}-10）ns区间内。

参数说明：Clocksource：欲设置的时钟。1－12 路可编程时钟。表示方法有两个：

1、数字，unsigned short clocksource

2、字符串法，参数类型 char*。例如：”1-4,8”

double start：时钟上升沿的起始施加，以施加图形时刻为 0 点

double stop：时钟高电平结束的施加，以施加图形时刻为 0 点，默认值为 0，此时时钟占空比为 50%

返回值： 返回值为操作结果，若返回 0，操作成功；返回 1，操作失败

说明：stop-start 时间必须小于测试时钟的周期值－10，并大于 10ns。否则出错。

示例：（1）`set_ffmat_clock("1-5",20,50)//`格式化时钟 `1－5` 延时为 `20ns`，高电平 `30ns`

（2）`set_ffmat_clock(6,40);//`格式化 `6` 延时为 `40`，占空比为 `50%`

（4） short set_fcmp_clock(clocksource,double cmpstart, double cmpstop=0);

函数用途：设置比较时钟参数。对于边沿比较，cmpstart有效，对于窗口比较cmpstart和cmpstop均有效。如果cmpstop=0，则默认比较时钟的占空比为 50%。比较时钟参数必须在 10ns<stop-start<（T_{period}-10）ns区间内。

参数说明：Clocksource：可编程时钟源，一次可以同时设置多个时钟；范围 1-12，两种表示方法：

1、数字，unsigned short clocksource

2、字符串法，参数类型 char*。例如：”1-4,8”

double start：时钟上升沿的起始施加，以比较周期开始时刻为 0 点

double stop：时钟高电平结束的施加，以比较周期开始时刻为 0 点。默认值为 0，即设置占空比为 50%的时钟

返回值： 返回值为操作结果，若返回 0，操作成功；返回 1，操作失败

（5） void set_external_clock();

函数用途：启动外部时钟，硬图形运行时有效。启动外部时钟时，系统不支持格式化，用户也不用设置管脚的比较时间，系统将按默认时间运行，同时只支持边沿比较。

参数和返回值：无

（6） short set_pin_timing(PinNum,unsigned short fmat_clock, unsigned short cmp_clock=1);

函数用途：用于设置通道的时序，如果多个通道时序相同，可以同时设置。数字通道的每一个管脚都可以设置为输入、输出或双向，用户必须设定管脚的格式化时钟和比较时钟。

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个相同的管脚时序模式，表示方式有两种：

1、数字，参数类型：unsigned short pin

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

unsigned short fmat_clock：预选择的格式化时钟

unsigned short cmp_clock：预选择的比较时钟，默认为时钟 1

返回值：设置成功，返回 0；失败返回 1

函数说明：当管脚为输入时，比较时钟被忽略；当管脚为输出时，格式化时钟被忽略，只有当管脚为双向时，两种均被使用。

（7） *short set_pin_mode(PinNum, PINTYPE pintype=PT_NF, PINTYPE cmptype=PT_EDGE);*

函数用途：设置管脚或管脚组的驱动模式或比较模式，即管脚的格式。

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个相同的管脚驱动模式，表示方式有两种：

1、数字，参数类型：unsigned short pin；1-64 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

PINTYPE pingype，管脚格式化类型，常数类型，可取值为：驱动模式：PT_NF、PT_RZ、 PT_RTZ、 PT_RTO、 PT_SBC、 PT_MCH、 PT_ONE 和 PT_ZRO 八个值

比较模式：PT_EDGE 和 PT_WINDOW

返回值：设置成功，返回 0；失败返回-1。

示例：

```
set_pin_mode("1-25",NRZ);//1－25 通道，不归零码
set_pin_mode("30,32-35,38",EDGE);//30，32－35，38 边沿比较模式
set_pin_mode(40,SBC,WINDOW);//40，驱动为环绕补码，比较为窗口模式
set_pin_mode("50-64"); //通道 50－64 不格式化
```

（8） *short apg_to_pin(unsigned short pin, APGTYPE apgtype=AT_PTN, unsigned short apgpin=0);*

函数用途：分配 APG 资源到管脚，每次只可以设置 1 个管脚

参数说明：unsigned short pin，管脚 1－系统中安装的管脚数

APGTYPE apgtype，APG 资源的类型，可取值：AT_APGX、AT_APGY、AT_APGD、AT_PTN，默认值为 AT_PTN

unsigned short apgpin，APG 资源；XY 地址可取值 0－13；D 可取值 0－7，默认值为 0

返回值：0，操作成功；1，操作失败。

示例：

```
apg_to_pin(4,APGX,0);//设置通道 4 为 APGX0
apg_to_pin(4);          //清通道 4 算法图形资源，恢复为图形通道
```

（9） *short apg_to_pin(char* pingrp, APGTYPE apgtype=AT_PTN, char* apgpin="0");*

函数用途：分配 APG 资源到管脚，每次可以同时分配多个管脚到 APG 资源上，上电默认状态，系统分配图形到所有的管脚上，因此不使用算法图形，则可以不调用该函数。

参数说明：char* pingrp，管脚组，有效管脚范围

APGTYPE apgtype，APG 资源的类型，可取值：AT_APGX、AT_APGY、AT_APGD、AT_PTN，默认值为 AT_PTN

char* apgpin，APG 资源数组；要求与管脚组对应，默认值为"0"

返回值：0，操作成功；1，操作失败。

函数说明：管脚数组和 APG 资源数组大小可以不相同，系统会从做到右自动匹配，所以管脚数组可以大于 APG 数组

示例：

```
apg_to_pin("35,23,27-29,40", AT_APGX,"0-5");//APG 资源到管脚
apg_to_pin("6-11,17,15,12", AT_APGY,"0-7");
```

（10）short w_xywidth(unsigned short x_used_width,unsigned short y_used_width);

函数用途：设定 XY 地址的使用个数。在测试存储器时或其他使用时，不一定所有的 XY 地址都会被使用，用户必须从 0 开始分配 xy 地址，并将 x 和 y 地址的分配数写入系统。否则系统会出错。

参数说明：unsigned short x_used_width，x 地址使用个数，0～14 为有效值；
unsigned short y_used_width，y 地址使用个数，0～14 为有效值；

返回值：0，操作成功；1，操作失败

函数说明：该函数要在分配完 APG 资源后执行，而且必须执行。

（11） *void start_capture();*

函数用途：启动捕捉存储器；当要启动捕捉存储器，捕捉图形时，要首先调用该函数。

参数说明：无

返回值：无

（12）void stop_capture();

函数用途：禁止或停止捕捉存储器；当要读取捕捉存储器数据时，首先调用该函数。

参数说明：无

返回值：无

（13）void load_capaddr(long addr,unsigned short board=0);

函数用途：设置捕捉存储器地址。与 inc_capaddr()共同使用，读取捕捉图形。

参数说明：long addr：捕捉存储器地址
unsigned short board ，通道数 1－安装的通道板号。默认参数为 0，即设置所有通道板的捕捉地址。

返回值：无。

函数说明：每个通道板可以独立设置捕捉存储器的地址，也可以一起设置。

（14）void inc_capaddr(unsigned short board=0);

函数用途：设置捕捉存储器地址。捕捉存储器地址＋1

参数说明：unsigned short board ，通道数 1－安装的通道板号。默认参数为 0，即设置所有通道板的捕捉地址。

返回值：无。

函数说明：每个通道板可以独立设置捕捉存储器的地址，也可以一起设置。捕捉存储器地址加一的执行效率比装载地址效率高！当连续读取大量捕捉数据时推荐使用该函数。

（15）void set_capsaddr(long saddr=-1,unsigned short board=0);

函数用途：设置捕捉存储器启动地址。

参数说明：long addr：捕捉存储器启动地址
unsigned short board ：通道数 1－安装的通道板号。默认参数为 0，即设置所有通道板的捕捉地址。

返回值：无。

函数说明：每个通道板可以独立设置捕捉存储器的地址，也可以一起设置。由于捕捉存储器的空间小于图形存储器的空间，因此不是所有图形的执行结果都可以被捕捉和存储的，因此用户可以选择从哪些地址开始启动捕捉存储器。

（16）SYSRESULT r_capture(char chanel)*

函数用途：读取捕捉存储器。与 inc_capaddr()和 load_capaddr()共同使用，读取捕捉图形。

参数说明：char* chanel，通道数 1－安装的通道板号。可以是数字也可要是字符数组

返回值：当输入值为数字时，返回一个通道的捕捉结果，返回值为 16 位无符号数；当参数说明为字符串时，返回值为 SYSRESULT 结构，其中 error=1，函数操作失败，error=0 函数操作成功

igroup 返回捕捉数据，数组的每一位代表一个通道，数组的顺序与 chanel 一一对应。

（17）unsigned long r_pin_capture(char pingrp,long addr)*

函数用途：读取捕捉存储器，按管脚顺序返回捕捉数据。

参数说明：char* pingrp，通道数 1－安装的通道板号。

long addr，捕捉存储器的地址

返回值：为 long 型数据，每一位对应一个管脚的捕捉值

示例：

```
r_pin_capture("7,5,11,14",1); //读取地址 1 的通道 7、5、11、14 的捕捉值。
```

（18）short pattern_stop_type(STOPTYPE stoptype, long stop_address=-1);

函数用途：硬图形运行方式设定。

参数说明：STOPTYPE stoptype：枚举变量，可取值 ST_FAIL、ST_ADDR、ST_HALT、ST_ALL 和 ST_STEP

分别含义为：失效停止、指定地址停止，仅指令停止，地址＋失效停止，单步运行，当设为指定地址和地址＋失效停止时，需设置停止地址。

long stop_addresss：停止地址，默认值为-1。

使用示例：

```
pattern_stop_type(ST_FAIL);
pattern_stop_type(ST_ADDR,100);
```

（19）short run_pattern(unsigned long start_addr , short stoptype=-1, long stop address=-1, unsigned short wait_time=1000);

函数用途：运行图形函数。

参数说明：unsigned long start_addr：运行图形的起始地址，可以是图形文件任意一行
short stoptype：运行图形的停止方式，取值为系统定义常数，如 ST_FAIL
long stop_address：停止地址，只有使用 ST_ADDR 方式时需要设置。
unsigned short wait_time：运行图形的等待时间，单位为毫秒；默认值等待 1s。即图形运行超过等待时间，就自动退出运行。

返回值：图形运行 Pass，返回 0；图形运行失效，返回 1；图形中断退出返回-1。如果不是失效方式停止，函数返回值只能为 0 或-1。

函数说明：硬图形是可以在任意一行以指定运行方式进行运行，并可以在任意指定地址停止。该函数有多种调用格式，方便用户使用。连续调用 run_pattern 函数系统会保存上次运行后的状态，如果希望清除上次的运行状态，在启动图形前调用 reset()函数即可。

停止方式有以下 5 种基本方式： ST_HALT 、 ST_FAIL 、 ST_ADDR 、 ST_STEP 和 ST_YPASS。

用户可以使用单一方式，也可以采用复合方式。停止方式之间可以做“或”运算。例如如果希望图形运行到指定行停止，并在此之前遇失效停止，可以使用：

ST_FAIL|ST_ADDR //失效或指定地址。

（1）如果使用 ST_ADDR 停止方式，必须指定停止地址。

（2）通常不用指定等待时间，除非图形运行时间很长或当图形不能自动退出时，等待时间有效。

函数示例：

不指定停止方式（按预设停止方式），运行图形

```
run_pattern(10);//启动图形，从第 10 行开始运行，使用默认等待时间
```

指定停止方式，运行图形

```
run_pattern(10, ST_FAIL);// 失效停止方式运行图形
run_pattern(10, ST_ADDR,100);指定停止地址，运行图形
```

指定复合停止方式和等待时间运行图形

```
run_pattern(10,ST_FAIL|ST_ADDR,50,100);//设定地址＋失效停止方式运行图形，等待 100ms，自动退出
```

单步运行方式，步骤如下

```
run_pattern(10,ST_STEP);// 单步方式运行图形，起始地址 10，运行一步
run_pattern();//运行一步图形
```

也可以使用：

```
run_pattern(10,ST_STEP);// 运行一步；由于单步已经启动，10 不起作用
run_pattern();//运行一步图形
```

单步运行模式结束方式有：遇到停止指令、使用 reset 结束或启动其他运行方式。

单步运行方式，并返回失效值

```
run_pattern(10,ST_STEP|ST_FAIL);//每运行一步，返回一次失效值
```

启动图形，使其自由运行：

```
run_pattern(10,ST_BYPASS);//图形自由运行，直到自行停止，或用户干预停止。该方法可以使系统产生特定的激励信号，也可以进行动态电流测试。
```

（20）SYSRESULT r_hfail(unsigned short board);

函数用途：回读硬图形运行失效时的失效值

参数说明：unsigned short chanel，安装的通道板号，1－8（最多）。可以为数字或字符串

返回值： 当参数说明为数字时，返回值为 16 位无符号数，代表该通道板各个通道的失效情况；若参数说明为字符串时，返回值为 SYSRESULT 结构，返回每个通道的失效情况。igroup 的每个数与 chanel 一一对应，数据的每个数据的每一位对应一个通道的失效情况，0 为 pass，1 为失效。

（21）long r_address(unsigned short fail=0);

函数用途：回读硬图形运行结束的停止地址。

参数说明：unsigned short fail：0，则读停止地址，1 则读失效地址。

返回值： 图形的停止地址。

（22）unsigned short force_sptn(char* pattern);

函数用途：施加设定的软图形

参数说明：char* pattern，图形字符串。有效值为：“1”，“0”，“L”，“H”，“X”，“Z”。

返回值：若软图形执行 pass，返回 0，软图形运行失效，则返回 1。-1，则操作失败。

使用说明：使用该函数一次可以向全部通道施加图形，示例如下：

```
force_sptn("1100LLHHXX[18]LLHH[25]HHLL[30]1100");
```

上述函数，第一个 1，代表 1 通道驱动为 1。依次向下，索引[18]代表从 18 通道开始图形，18 通道之前没有说明的通道，默认图形为 Z。

下面的使用为非法：

```
force_sptn("1100LLHHXX[18]LLHH[20]HHLL[30]1100");
```

因为索引 20 和 18 之间有四个图形位，所以这个索引至少要大于 21。

对于输入管脚必须使用图形“1”“0”“Z”；对于输出管脚必须使用图形“L”“H”“X”否则出错。

（23）SYSRESULT r_sfail(unsigned short board);

函数用途：回读，软图形运行失效时的失效值

参数说明：unsigned short chanel，安装的通道板号，1－8（最多）。可以为数字或字符串

返回值： 当参数说明为数字时，返回值为 16 位无符号数，代表该通道板各个通道的失效情况；若参数说明为字符串时，返回值为 SYSRESULT 结构，返回每个通道的失效情况。igroup 的每个数与 chanel 一一对应，数据的每个数据的每一位对应一个通道的失效情况，0 为 pass，1 为失效。

（24）unsigned long r_pin_sfail(char* pingrp)

函数用途： 按指定通道读取软图形失效值。返回值与管脚顺序有关。

参数说明：char* pingrp，通道数 1－安装的通道板号。

返回值：为 long 型数据，每一位对应一个管脚的软图形失效值

函数说明：该函数是转为读取管脚输出状态设计。如果希望回读的失效值就是实际的管脚电平状态，比较图形请设置为“L”。

示例：

```
r_pin_sfail("7,5,11,14",1); //读取地址 1 的通道 7、5、11、14
```
的软图形失效值。

（25）double measure_tpd(unsigned short cmpclock,int start,double freq=1, int wait=1000)

函数用途： 测试数字器件的输出延迟。

参数说明：unsigned short cmpclock，用于延时测试的比较时钟，可以选择 1－12 时钟；其中 10－12 时钟为高精度时钟，分辨率为 0.3125ns，其他时钟为 5ns。

int start：用于 Tpd 测试的图形的起始地址。

double freq：测试 TPD 时使用的测试频率，默认为 1MHz。

int wait： 测试图形运行的等待时间，默认为 1000ms。

返回值：double 型，为器件的延时

函数说明：该函数测试延时必须配合参考电平的重新设置，以及输出管脚的时序设置。函数设置的时钟频率，在函数执行完毕后，系统自动恢复原来的测试频率。

示例：

测试 74ls00 的输出延时：

```
set_dut_timing("3,6,8,11",15,10);
set_dut_voh("3,6,8,11",0.35);
tpd=measure_tpd(10,20);
```

（26）short ptn_to_pin(PinNum, PINSTATE state=PS_ADD);

函数用途：连接图形通道到管脚或断开图形通道到管脚。在使用图形通道之前必须将图形通道连接到管脚，同时断开连接的其他资源。

参数说明：PinNum，1－安装的所有管脚，同时可以连接多个图形通道到管脚，管脚表示方式有两种：

1、数字，参数类型：unsigned short pin；1-64 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

PINSTATE state：管脚状态，PS_ADD，连接；PS_OFF 断开；默认值为 PS_ADD

返回值：连接/断开成功，返回 0，连接/断开失败返回 1；

函数示例：

连接图形到通道，可以使用：

```
ptn_to_pin(1);
```

也可以使用：`ptn_to_pin(1,PS_ADD);`

断开图形与通道的连接，使用

```
ptn_to_pin(1,PS_OFF);
```

（27）short set_pin_vih(PinNum, double volt=0);

函数用途：设置图形通道驱动电平，高电平。

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个图形通道驱动电平，表示方式有两种：

1、数字，参数类型：unsigned short pin；1-PIN_NUMBER 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

double volt：欲设置的驱动电平值-8V～+8V 之间有效

返回值：驱动电平设置成功，返回 0，设置失败返回 1。

（28）short set_pin_vil(PinNum, double volt=0);

函数用途：设置图形通道驱动电平，低电平。

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个图形通道驱动电平，表示方式有两种：

1、数字，参数类型：unsigned short pin；1-PIN_NUMBER 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

double volt：欲设置的驱动电平值-8V～+8V 之间有效

返回值：驱动电平设置成功，返回 0，设置失败返回 1。

（29）short set_pin_voh(PinNum, double volt=0);

函数用途：设置图形通道比较电平低电平

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个图形通道比较电平，表示方式有两种：

1、数字，参数类型：unsigned short pin；1-PIN_NUMBER 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

double volt：欲设置的比较电平值-7V～+7V 之间有效

返回值：设置成功，返回 0，设置失败返回 1；

（30）short set_pin_vol(PinNum , double volt=0);

函数用途：设置图形通道比较电平低电平

参数说明：PinNum，1－安装的所有管脚，同时可以设置多个图形通道比较电平，表示方式有两种：

1、数字，参数类型：unsigned short pin；1-PIN_NUMBER 有效

2、字符串法，参数类型 char* pingrp。例如：”1-4,8,25-33”

double volt：欲设置的比较电平值-7V～+7V 之间有效

返回值：设置成功，返回 0，设置失败返回 1；

2.4 与被测器件管脚对应函数说明

在编辑图形文件时，已经将被测器件管脚（dut）与系统的通道（pin）进行了一一对应的分配，这时如果采用与 dut 对应的函数将会方便用户操作，因此系统也提供了这些映射函数供用户使用，他们每一个均与相应的带“pin”的函数相对应，调用格式和含义完全相同，这些函数如下：

（1） set_dut_timing(dutNum, unsigned short fmat_clock,
unsigned short cmp_clock=1)

（2） set_dut_mode(dutNum,, PINTYPE pintype=PT_NF,
PINTYPE cmptype=PT_EDGE);

（3） apg_to_dut（dutNum，APGTYPE apgtype=AT_PTN,unsigned short apgpin=0）

（4） r_dut_ sfail(char dutgrp);*

（5） r_dut_capture(char dutgrp,long address);*

（6） ptn_to_dut(dutNum， PINSTATE pinstate=PS_ADD);

（7） set_dut_vih(dutNum , double volt=0);

（8） set_dut_vil(dutNum , double volt=0);

（9） set_dut_voh(dutNum , double volt=0);

（10）set_dut_vol(dutNum , double volt=0);

使用这些函数时，系统可以帮助用户判断管脚操作是否错误。如果被设置的是电源或地或闲置管脚，系统会自动报错。

2.5 控制位控制函数说明

（1）short set_rcbit(CbitNum);

函数用途：置继电器控制位。继电器控制位为 32 位，输出方式为 MOS 管开漏输出，用户使用必须接上拉电阻到用户电源。用户电源可选+5，+12 或+15V。

参数说明：CbitNum，1－32，同时可以设置多个控制位，表示方式有两种：

1、数字，参数类型：unsigned short rcbit；1－32 有效。

2、字符串法，参数类型 char* rcbit_grp。例如：”1-4,8,25-33”

返回值：无。

（2）short clr_rcbit(CbitNum);

函数用途：清继电器控制位。继电器控制位为 32 位，输出方式为 MOS 管开漏输出，用户使用必须接上拉电阻到用户电源。用户电源可选+5，+12 或+15V。

参数说明：CbitNum，1－32，同时可以设置多个控制位，表示方式有两种：

1、数字，参数类型：unsigned short rcbit；1－32 有效。

2、字符串法，参数类型 char* rcbit_grp。例如：”1-4,8,25-33”

返回值：无。

2.6 TMU 及高精度电压源/表函数说明

（1） void force_vref (double voltage)

函数用途：设置高精度电压源电压。

参数说明：double voltage：欲设置的输出电压值-10V～+10V 之间有效；

返回值：无。

（2） void disconn_vref ()

函数用途：断开高精度电压源

参数说明：无

返回值：无

（3） double measure_dvm(HVMVRANGE range,int ms=5)

函数用途：使用高精度电压表测压。

参数说明：HVMVRANGE range，选择测压量程；

int ms，设置测量前的延迟时间；

返回值：测量结果，单位 V。

（4） void set_tmu_ctrl(TMUSET sTmuSet)

函数用途： TMU 测量回路的工作状态。

参数说明： TMUSET sTmuSet，TMU 工作模式

返回值：无。

（5） void set_tmu_start()

函数用途：启动 TMU 测量。

返回值：无。

（6） double measure_time(int source)

函数用途：读取所选回路的测量结果。

参数说明：int source，模式为 1

返回值：测量结果，单位 s，返回值小于 0 时表示测试未完成。

（7） double measure_plus(int type,double voltage,int ms,int div,int count)

函数用途：测量脉冲宽度；

参数说明：int type，选择测试方法 范围 0~3；

0、测量第一通道正脉冲宽度

1、测量第一通道负脉冲宽度

2、测量第二通道正脉冲宽度

3、测量第二通道负脉冲宽度

double voltage，设置比较器反转电平

int ms，测量时间

int div，分辨率

int count，测量的总脉冲数

返回值：count 个脉冲宽度的平均值，单位 ns，返回值小于 0 时测试未完成。

（8） *double measure_period(int type,double voltage,int ms,int div,int count)*

函数用途：测量周期。

参数说明：int type，选择测试方法 范围 0~3；

0、测量第一通道信号

1、测量第一通道反向信号

2、测量第二通道信号

3、测量第二通道反向信号

double voltage，设置反转电压，高频模式无意义；

int ms，测量时间

int div，分辨率

int count，测量的总脉冲数

返回值：count 个周期的平均值，单位 ns，，返回值小于 0 时测试未完成

（9） struct TMUSET

```
{
    double dVoltage1;
    double dVoltage2;
    unsigned short nCount;
    unsigned long nDiv;
    TMUCMP eCondition1[2];
    TMUCMP eCondition2[2];
    unsigned short nInput;
};
```

详细说明：

1）dVoltage1、dVoltage2:两个通道的信号反转电平，将输入信号转变为逻辑信号，电压值高于反转电平时比较器输出逻辑高，电压值低于反转电平时比较器输出逻辑低，见（图 2-6-1）

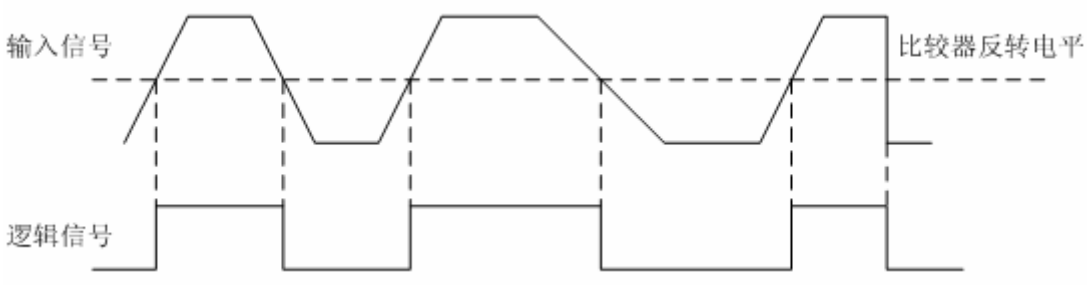


图 2-6-1

2）nCount: 设置信号的测量次数，见（图 2-6-2）

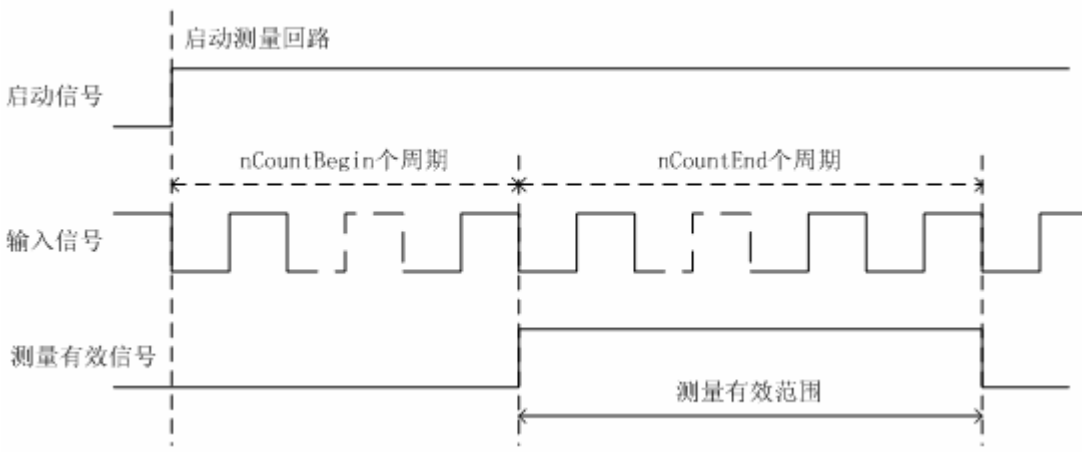


图 2-6-2

3）nDiv：设置测量分辨率，公式为

分辨率(ns)=(nDiv+1)*10

4）nInput：设置输入信号处理方式，固定设置为 0

5）eCondition1[2]、eCondition2[2]：设置通道的触发条件，见（图 2-6-3）

eCondition1[0] ：1 通道起始触发条件

eCondition2[0] ：2 通道起始触发条件

eCondition1[1] ：1 通道终止触发条件

eCondition2[1] ：2 通道终止触发条件

触发条件 ：

TMU_X	不关心状态
TMU_ZERO	逻辑零
TMU_ONE	逻辑一
TMU_UPEDGE	上升沿
TMU_DOWNEDGE	下降沿

2.7 波形产生及波形分析函数说明

（1）short wg_start (int wg ,int num)

函数说明：启动音频源，并发出指定序号的音频信号

参数说明：int wg：选择要启动音频源序号，取值 1 or 2

int num：要启动的波形序号，1～设置波形的最大值。

返回值： 启动成功，返回 0；启动失败返回-1。

（2）short wg_start_all (int num)

函数说明：同时启动两路音频源，启动指定序号的音频信号

参数说明：int num：要启动的波形序号；1～设置波形的最大值

返回值： 启动成功返回 0，启动失败返回 1。

（3）void wg_stop (int wg)；

函数说明：停止音频源信号输出。

参数说明：int wg：选择要停止的音频源序号，取值 1 或 2

返回值： 无返回值。

（4）void wg_stop_all ( )；

函数说明：停止全部的音频源信号输出。

参数说明：无参数

返回值： 无返回值。

（5）void sample (double p0, int sample_dot, double WA_ARRAY[])

函数说明：波形分析器开始采样，将采样结果放到用户指定的数组，采样结束后自动停止波形分析器。

参数说明：int p0：采样频率设置，其含义是系统最高采样频率。系统采样频率范围为 300KHz--0.625KHz。

int sample_dot：采样点数，最多 65000 点

double WA_VAL[]：采样结果存放数组。数组大小应大于等于采样点个数

返回值： 无

注意事项：采样周期的设定与被测的频率有关，最好采样频率是被测频率的 10 倍左右为佳。

（6）void creat_wafile(char fname[], unsigned int sample_dot, double WA_ARRAY[])

函数说明：建立采样点文档。

参数说明：char fname[]：保存路径以及文档名称。

int sample_dot：保存点数，最多 65000 点

double WA_VAL[]：采样结果存放数组。

返回值： 无

（7）void set_single_ch (int ch， int gain)；

函数说明：设置波形分析器通道序号

参数说明：int ch，波形分析器通道序号，1 或 2。

int gain，对被测波形的增益倍数。取值：1，10，100，1000

返回值： 无返回值

（8）double measure_thd ( )

函数说明：测量信号的失真度

参数说明：无参数

返回值： 返回被测信号的失真度，百分比单位

（9）double measure_rms (int S_Frq=300,int S_Dot=3000)；

函数说明：音频电压表测量交流信号有效值

参数说明：int S_Frq：采样频率。取值范围：300KHz--1KHz。

int S_Dot：采样点数。

返回值： 返回被测信号的有效值，单位 V。

（11）double measure_avg (int S_Frq=300,int S_Dot=3000)；

函数说明：音频电压表测量交流信号平均值

参数说明：int S_Frq：采样频率。取值范围：300KHz--1KHz。

int S_Dot：采样点数。

返回值： 返回被测信号的有效值，单位 V。

2.8 矩阵控制函数

（9）void hvpmu_to_pin (int pmu,PINUM pins,PINSTATE ps=PS_ADD)；

函数说明：高压 PMU 连接到通道

参数说明：int pmu：高压 PMU 号，1－8

PinNum pins：要连接的通道，可以使用数字，也可以使用字符串

高压 PMU1、3、5、7 仅可以连接到 1－8 通道；高压 PMU2、4、6、8 仅可以连接到 33－40 通道。

PINSTATE ps：继电器连通状态，可取 PS_ADD，PS_OFF 和 PS_NEW

返回值： 无返回值。

（11）void gnd_to_pin (PINUM pins,PINSTATE ps=PS_ADD)；

函数说明：模拟地连接到通道。

参数说明：PinNum pins：要连接的通道，可以使用数字，也可以使用字符串

模拟地可以连接到 1－8 通道和 33－40 通道。

PINSTATE ps：继电器连通状态，可取 PS_ADD，PS_OFF 和 PS_NEW

返回值： 无返回值。

第三章 算法图形及指令

算法图形发生器包括：算法序列发生器、14 位 X 地址发生器、14 位 Y 地址发生器。后两个发生器用于存储器或其他测试，每个地址发生器包括两个寄存器和一个算术逻辑单元。

3.1 算法图形指令含义说明

3.1.1 算法序列控制指令

图形指令	含义
INC	执行当前行图形，然后转到下一行图形。
LDC,n	加载循环计数器，该行为循环起始行。同时执行当前行图形，之后转到下一行图形。循环次数 n 最大 65535。
LOOP	循环语句。执行当前行图形后，开始循环 LDC 到 LOOP 语句之间的图形 n 次。循环满后转到下一行图形。
RPT,n	重复执行当前行图形 n 次，然后跳转到下一行图形。重复次数最多 65535 次。RPT,1 执行当前图形 2 次，即重复 1 次。
MCH,n	匹配图形命令。如果执行当前行图形未通过，则重复执行当前行图形，直到满 n 次，如果 n 次都没有通过，图形失效；如果某次执行为 pass，则执行下一行图形。
JZ,line	执行当前行图形，若 x 或 y 地址为 0 则跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
JNZ,line	执行当前行图形，若 X 地址和 Y 地址都不为 0，跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
JYNZ,line	执行当前行图形，若 Y 地址不为 0 跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
JYZ,line	执行当前行图形，若 Y 地址为 0 跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
JO,line	执行当前行图形，若 x 和 y 地址奇偶校验为奇数，则跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
JE,line	执行当前行图形，若 x 和 y 地址奇偶校验为偶数，则跳转到指定行，否则继续执行下一行图形。跳转范围为当前行前后各 32767 行。
HALT	停止命令。同时转到下一行图形

说明：

①、循环、重复、匹配最多为 65536 次和跳转最多 32767 行（前后各 32767）

②、跳转指令专为算法地址发生器提供。普通图形编辑不需要使用。

3.1.2 XY 地址 AB 寄存器算法指令

A:<14bit 数>	B:<14bit 数>	装载 Load，可同时装载
A++	B++	A 、B 寄存器加 1
A--	B--	A、B 寄存器减 1
A	B	无操作
A:X		A 寄存器装载 X 地址
A>>	A>>	A 寄存器右移 1 位
A<<	A<<	A 寄存器左移 1 位

3.1.3 XY 地址算术逻辑单元操作指令以及示例

示例(使用说明)	指令	含义
X＝A++, B:100	A	选择 A
X=A:X, B++	B	选择 B
X=B--, A--	A~	选择 A，A 取反
X=A+B,A++,B++	B~	选择 B，B 取反
X=A-B,A:10 Y=A+B,B--	A+B	选择 A＋B 的和
	A-B	选择 A－B 的差

3.1.4 XY 地址输出以及算法数据控制组合指令

WRITE	写存储器，x，y 地址同时输出
READ	读存储器，x，y 地址同时输出
WRITEX	写存储器，x 地址输出
WRITEY	写存储器，y 地址输出
READX	读存储器，x 地址输出
READY	读存储器，y 地址输出
NOP	空操作，保持上一拍施加地址状态(不允许在第一行使用)

X、Y 指令之间以分号“;”相隔，内部分指令操作使用“,”分隔。写入顺序任意，图形编辑界面将帮助用户完成自动更正和调整顺序。

复用模式时，XY 地址不能同时输出，XY 地址均由 X 地址位输出。XY 输出由指令 READX、READY、WRITEX、WRITEY 控制；非复用模式 X 地址输出 X 地址位，Y 地址位输出 Y 地址。

本系统不提供算法数据发生器，因此 WRITE、READ、NOP 仅注释用。

3.2 完整指令示例以及说明

3.2.1 清存储器（XY 地址复用）512KB

```
INC          WRITEX,   X=A:0,B:0;     Y=A:0,B:0;        D=A:0,B:0xff;
LDC,512      WRITEY,   X=A;           Y=A,B++;          D=A,B>>;
INC          WRITEX,   X=A;           Y=A;              D=A;
RPT 511      WRITEX,   X=A++;         Y=A ;             D=A;
LOOP          NOP,     X=A:0;         Y=A++;           D=A;
```

3.2.2 清存储器（XY 地址非复用）512KB

```
INC          WRITE;    X=A:0;      Y=A:0;      D=A: 0
LDC,512      WRITE;    X=A++;      Y=A;        D=A
RPT 511      WRITE;    X=A++;      Y=A;        D=A
LOOP          WRITE;   X=A:0;      Y=A++;      D=A
```

3.2.3 算法指令与算法序列指令的执行顺序

首先执行算法指令，然后执行图形指令和测试图形。例如：

```
INC      WRITEX,   X=A:0,B:0; Y=A:10,B:10;      D=A,B:0xff;
```

系统首先将X地址发生器的A、B寄存器寄存器赋值0，Y地址发生器A、B地址寄存器装载10；算法数据A寄存器保持，B装载0xff；然后将X地址输出，D输出A；最后执行当前行图形，并将图形、指令地址增加1。

A:X或A:Y指令是A寄存器装载X地址或Y地址；该指令装载的是上一拍的数据，而不是当前算出的数据，需要注意。

例如

```
INC          WRITEX,   X=A,B:100;     Y=A;              D=A;
JZ,line      WRITEX,   X=B++,A:X;     Y=A ;             D=A;
```

其中第二行A装载的值是100，而不是101

3.3 图形编辑界面图

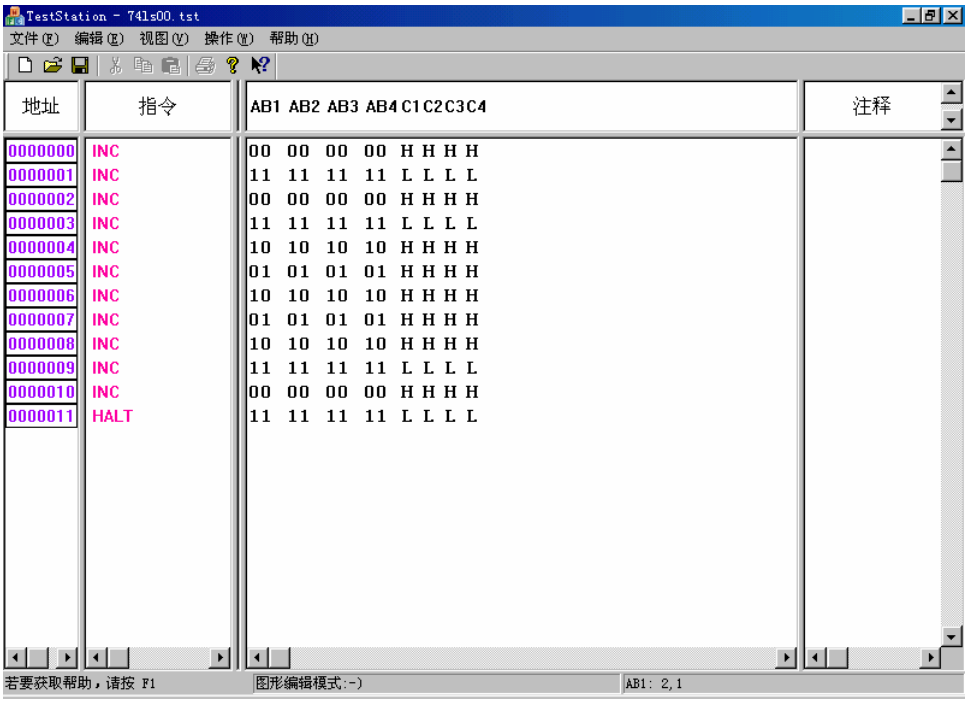


图 3-1 图形编辑界面

图形编辑界面是数字测试图形和算法图形编写以及调试的多功能界面。创建测试程序后，在TestShell界面单击编辑图形按钮，就可以打开图形编辑界面。详细说明参加“软件 编程手册”。

图形编辑界面从做到右共5列。最左侧为地址，不可以修改，但双击行号可以进行该行图形的注释和恢复；添加图形、修改图形时会行号将自动修改。第二列为指令，用于编写算法序列指令。第三列算法图形编辑窗口，用于编写算法指令。若测试图形中不包含算法图形的程序，该列自动隐藏。第四列为图形编辑窗口，用户可以编写自定义的图形。第五列为注释窗口，用于编写用户注释，每行注释最多255个字符。

第四章 数字芯片交流参数测试

3196D 数模混合测试系统和 3168 数字测试系统可以对数字芯片的部分交流参数进行近似测试和分析，其测试结果基本接近国外测试系统的测试结果，可以作为芯片设计验证测试和产品合格测试。3196D 和 3168 系统可以测试的数字芯片的交流参数主要有以下几种：

（1）芯片输出延时（Tpd）测试，具体包括 TPLH、TPHL、TPHZ、TPLZ、TPZH 和 TPZL 等类似参数。对于 Tpd 测试的精度为 1ns，最小可以测试 2～3ns 的器件延时。

（2）建立、保持时间测试。对于该项指标测试，要求器件的建立或保持时间大于 2ns。

（3）存储器的读写访问时间 Tw 和 Tr 等。

（4）可以对器件输出的脉冲宽度、频率等参数进行测试。

使用本系统进行交流参数测试不需要额外添加测试电路。若对激励信号的上升和下降时间有要求，可以对激励信号并联电容改变上升、下降时间。本系统输出驱动信号 3V 摆幅的上升、下降时间为 2ns，5V 摆幅为 3ns，10V 摆幅为 4ns。

4.1 交流参数测试的系统限制

由于本系统设计初并没有考虑交流参数的测试，器件输出到比较器回路电容稍大，因此不能为交流参数测试提供理想的负载条件。以 54AC00 为例，该器件输出管脚的上升下降时间小于 3ns（图 4-1），接入测试系统后，器件输出管脚的上升、下降时间为 10ns（图 4-2），此时已经无法为器件的交流参数测试搭接外围测试电路。图 4-1 和 4-2 显示了器件输出管脚连接到测试系统前、后下降时间的变化。

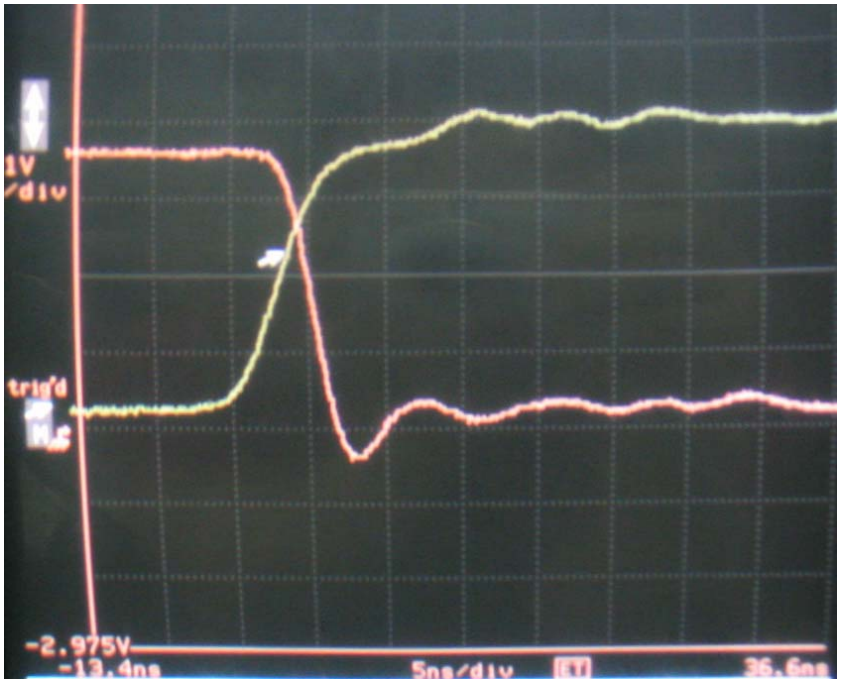


图 4-1 54AC00 TPHL 波形图

图 4-1 所示为器件经负载板接入测试系统，由测试系统提供激励信号，器件输出管脚悬空时，示波器探头测试到的输入、输出信号波形。正跳变信号为系统输出的激励信号，负跳

变为器件输出信号。施加条件和测试仪器如下：

- 使用测试仪：XR3196D/XR3168
- 使用负载板：74 通用适配器
- 使用示波器：Tektronix DSA602，300MHz 通道（通道 1【黄色】比通道 2【红色】超前 2ns。）
- 示波器以及测试探头的上升时间为 4ns。
- 交流波形图中，通道 1 为测试系统产生的激励波形，通道 2 为器件输出管脚波形。
- 测试系统输出方波频率 1KHz，VIH＝4.0V，VIL＝0V
- 器件供电电压为 4.5V。

当器件输出引脚连接到测试系统时的波形如图 4-2 所示。

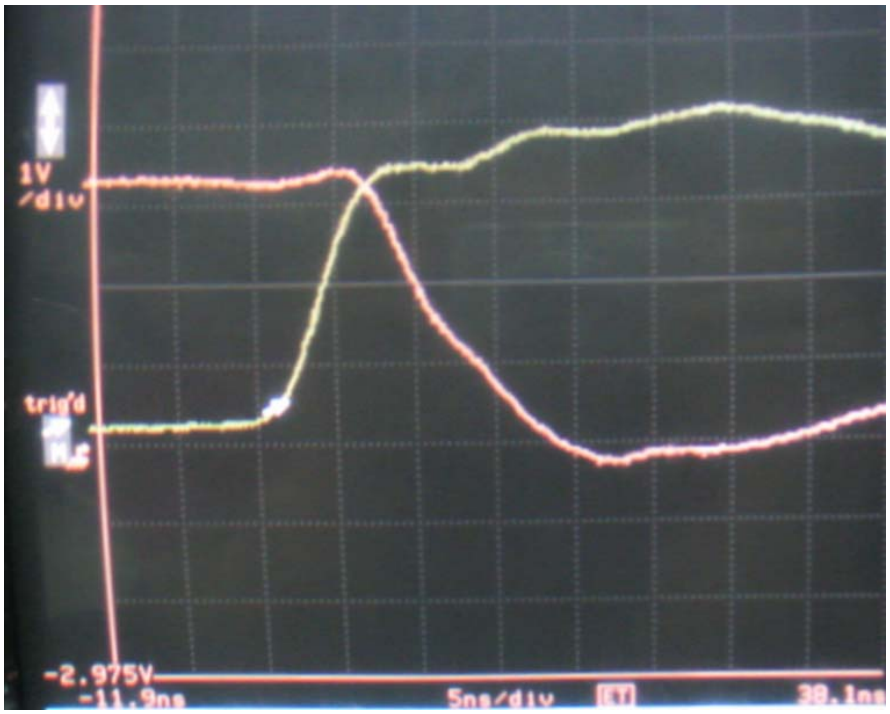


图 4-2 器件接入测试系统后的 TPHL 波形

从图中可以看出器件输出管脚的下降时间变长，这是由器件输出引脚到系统比较器的信号回路电容引起的。

对比图 4-1 和 4-2 可以看出，虽然输出信号的下降变缓了，但是激励信号跳变起始点与器件输出管脚信号的跳变起始点之间的延时并没有变化，我们可以认为该时间为器件的输出延时 TPD。我们使用该方法对大量器件的输出延时测试，并与国外的测试仪（泰瑞达 J750）的测试结果比对得出：该方法的测试结果基本接近实际测试值，是可信的。而使用 3196D 和 3168 测试仪测试的成本很低。

4.2 器件输出延时 Tpd 的测试方法

4.2.1 测试 Tpd 的步骤

器件输出延时的测试步骤如下：

（1）测试器件输出低到高延时（Tplh）：设置器件输出管脚比较电平 VOH 为器件输出低电平＋0.3V；器件输出 VOL 为典型值，如 0.4V；测试图形中不关心器件输出低的图形矢量。

（2）测试器件输出高到低延时（Tphl）：设置器件输出管脚比较电平 VOL 为器件输出高电平－0.3V；器件输出 VOH 保持为典型值，如 2.4V；测试图形中不关心器件输出为高的矢量。

（3）按照交流参数的测试条件设置输入激励的 VIH 和 VIL。

（4）按照测试条件设置器件的供电电压。

（5）设置芯片输出管脚的时序模式为：格式化时钟为 15；比较时钟为 10。

（5）调用 Tpd 测试函数 measure_tpd，测出 Tpd。

（6）恢复所修改的 VOH 或 VOL 电平。

4.2.2 注意事项

（1）跳变电压设置不能与器件输出电平太接近，太近会由于输出信号的微小波动引起测试结果错误；太远则测试结果误差偏大。一般 TPLH 选取比器件输出低电平多 300～400mV 的电压为宜；TPHL 选取比器件输出高电平低 300～400mV 为宜。

（3）电源要接滤波电容，一般 0.1uF 左右即可，电容尽量与器件电源和地引脚接近。

（4）测某个输出管脚的延时时，尽量使器件的其他逻辑和时序电路保持不工作或稳定状态。器件内部 MOS 管或晶体管的开关工作会影响输出信号的质量，影响测试结果。所有的交流测试必须使用这个原则。

（5）测试 TPLH 时器件输出低电平不比较，测试 TPHL 时器件输出高电平不比较。

（6）对于 TPD 小于 5 的最好使用格式化方式测试。由于系统的比较时钟有 5ns 的死区，对于小于 5ns 的延时无法准确测出。格式化的方法是：对输入激励使用 NRZ 格式，使激励延迟 20～30ns，这样比较时钟就可以避过测试死区。

4.2.3 测试实例

下面以测试 54AC00 器件的 TPHL 为例说明如何测试器件的输出延时。首先测试器件的输出高电平 VOH，然后将 VOH-0.3 作为器件高到低的跳变电压，即设置 vol＝VOH-0.3，然后运行图形，移动比较时钟测试 Tpd，具体步骤如下：（以 54AC00 的 TPHL 为例）

（1）设置器件电源电压为 4.5V.

（2）设置器件输入波形的高电平 VIH=4.0,VIL=0；

（3）设置器件输出管脚比较低电平 VOL＝4.25//VOH 为 4.49V

（4）设置器件输出管脚比较时钟为 10（10－12 为精密延时可调时钟）

（5）运行图形，测试 Tpd

具体程序如下：（选用 74 通用适配器，54AC00 测试程序）

```
  //测试项名： Tphl_45
MYDLLAPI double Tphl_45(double* grpTestValue,short* pingrp)
{
  hvforce_v(3,4.5,200);
  set_dut_vih(“1,2,4,5,9,10,12,13”,4.0);
  set_dut_vil(“1,2,4,5,9,10,12,13”,0);
  set_dut_voh("3,6,8,11",2.4);
  set_dut_vol("3,6,8,11",4.25) ;
  set_dut_timing("3,6,8,11",15,10);
  ptn_to_dut("1-6,8-13") ;
  delay_ms(10);
  double tpd=measure_tpd(10,16);  //时钟 10 测试，从图形 16 行运行。
```

```
  set_dut_timing("3,6,8,11",15,1);//恢复原时序设置
  set_dut_vol("3,6,8,11",0.4) ;    //恢复标准参考电平
  hvpmu_rst(3);
}
```

若使用格式化测试，其代码如下：

```
  //测试项名： Tphl_45
MYDLLAPI double Tphl_45(double* grpTestValue,short* pingrp)
{
  hvforce_v(3,4.5,200);
  set_dut_vih("1,2,4,5,9,10,12,13",4.0);
  set_dut_vil("1,2,4,5,9,10,12,13",0);
  set_dut_voh("3,6,8,11",2.4) ;
  set_dut_vol("3,6,8,11",4.25) ;
  set_dut_timing("1,2,4,5,9,10,12,13",2);
  set_dut_mode("1,2,4,5,9,10,12,13",NRZ);//输入引脚格式化设置
  set_ffmat_clock(2,20) ;          //设置格式化时钟前沿为 20ns
  set_dut_timing("3,6,8,11",15,10);
  ptn_to_dut("1-6,8-13") ;
  delay_ms(10);
  double tpd=measure_tpd(10,16)-20;//时钟 10 测试，从图形 16 行运行。
  set_dut_timing("3,6,8,11",15,1);
  set_dut_vol("3,6,8,11",0.4);
  set_dut_mode("1,2,4,5,9,10,12,13",NF) ;
  hvpmu_rst(3);
}
```

4.2.4 测试图形

54AC00的TPHL测试图形如图4-3所示。

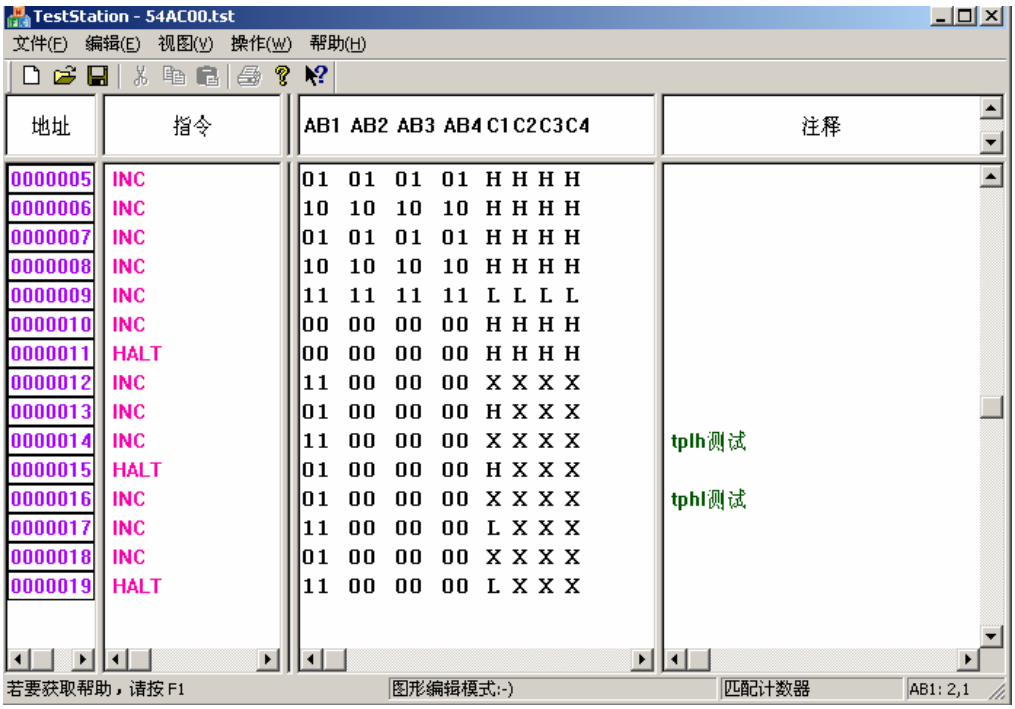


图 4-3 54AC00 TPHL测试图形

4.3 器件建立保持时间的测试方法

对于触发器类的器件，交流参数还包括数据对时钟的建立和保持时间的测试。

一般器件的建立和保持时间都很短，例如54AC273的时钟对触发器D端的建立时间不超过4.5ns，保持时间不超过2ns。为了能够测试如此小的交流参数，必须使用定时精度很高的时钟才能完成。

3168和3196D测试系统提供了12路精密程控时钟，总体的定时精度为1ns ，但这个精度还不能满足建立、保持时间测试。但两个系统的时钟是分为3组产生的：时钟1－4为第一组；时钟5－9为第二组；时钟10－12为第三组。每组内部的时钟之间的定时精度非常高，不超过200ps，因此可以使用同一组之内的几个时钟配合完成该参数的测试。

4.3.1 建立时间的测试步骤

- 设置触发器数据输入端为不归零（NRZ）格式化，选择时钟10作为格式化时钟。
- 设置触发器时钟端为归零或归一格式化（根据是上升沿触发还是下降沿触发），选择格式化时钟为12。电平触发也进行格式化设置，高电平触发设置为归零格式，低电平触发设置为归一格式。
- 设置触发器输出端的时序模式为边沿比较，比较时钟延时设置要足够大，保证可以稳定的采样到器件的输出电平。
- 设置时钟10的初始时间要小于时钟12，两时钟前沿时间差要大于建立时间。
- 运行被测图形，若图形通过，则**增加时钟10的延时**，直到图形运行失效，此时时钟10与时钟12的差值就是建立时间。

4.3.2 保持时间的测试步骤

- 设置触发器数据输入端为不归零（NRZ）格式化，选择时钟10作为格式化时钟。
- 设置触发器时钟端为归零或归一格式化(根据时钟是上升沿触发还是下降沿触发)，选择格式化时钟为12。电平触发也进行格式化设置，高电平触发设置为归零格式，低电平触发设置为归一格式。
- 设置触发器输出端的时序模式为边沿比较，比较时钟延时要足够大，保证可以稳定的采样到器件的输出电平。
- 设置时钟10的初始时间要大于时钟12，两时钟前沿时间差要大于保持时间。
- 运行被测图形，若图形通过，则**减小时钟10的延时**，直到图形运行失效，此时时钟10与时钟12的差值就是保持时间。

4.3.3 注意事项

- 器件的供电按测试条件给出，并在电源与地之间接去耦电容
- 输入信号激励按测试要求提供。
- 输出管脚比较电平为典型值。
- 每次只能测试一个数据端与时钟端的建立或保持时间。

4.3.4 建立时间测试示意

以下各例均以54AC273为例测试建立时间和保持时间。

建立时间测试的示意图如图 4-4 所示。

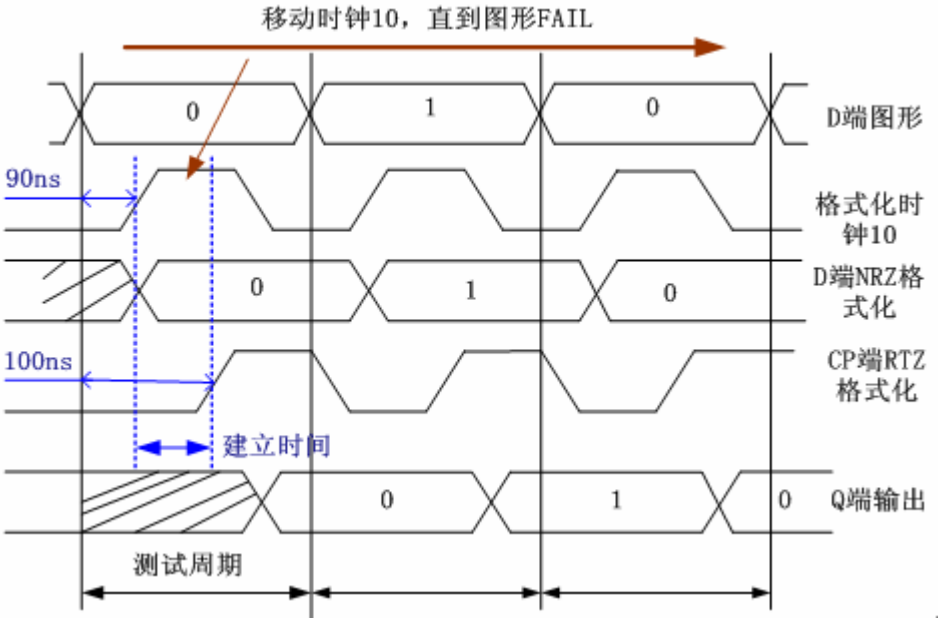


图 4-4 建立时间测试示意

图 4-4 中 D 端为 273 的任意一个 D 触发器的输出端，Q 端为相应 D 触发器的输出端。D 端使用时钟 10 作为格式化时钟，设置为 NRZ 格式，时钟 10 起始延时为 90ns。时钟端使用时钟 12 为格式化时钟，设置为 RTZ 格式，时钟 12 起始延时为 100ns。运行测试图形，如图 4-5 所示，从 34 行开始运行。若图形通过，则增加时钟 10 的延时，直到运行图形失效。这时时钟 12 与时钟 10 的延时的差值就是建立时间。

4.3.5 建立时间测试代码

```
//测试项名： Tsu_45
MYDLLAPI double Tsu_45(double* grpTestValue,short* pingrp)
{
   double Tsu=0 ,tclk=100,tData=90 ;
   hvforce_v(3,4.5,150);
   set_dut_vih("1,3,4,7,8,11,13,14,17,18",4.0);//参考电平设置
   set_dut_vil("1,3,4,7,8,11,13,14,17,18",0.0);
   set_dut_voh("2,5,6,9,12,15,16,19",2.4);
   set_dut_vol("2,5,6,9,12,15,16,19",0.4);
   set_dut_timing("2,5,6,9,12,15,16,19",15,1);//Q 端比较时钟为 1
   set_cmp_clock( "1",400) ;    //Q 端比较时钟 400
   set_dut_mode("11",PT_RTZ) ;  //时钟格式为归零
   set_dut_timing("11",12) ;    //使用时钟 12
   set_ffmat_clock("12",tclk) ; //时钟前沿为时间周期+100ns
   set_dut_mode("3,4,7,8,13,14,17,18",PT_NRZ) ;//D 端格式化
   set_dut_timing("3,4,7,8,13,14,17,18" ,10) ; //格式化时钟
   set_ffmat_clock("10",tData) ; //D 端数据初值
   ptn_to_dut("1-9,11-19");
   delay_ms(20);
   while(1)
  {
     short fail=run_pattern(34,ST_FAIL) ;
     Tsu=tclk-tData ;
     if(fail) break ;
     tData +=0.5 ;   //时间步进为 0.5ns
     set_ffmat_clock("10",tData) ;
     if(Tsu<0)  break ; //防止死循环
  }
  return Tsu ;
  }
```

4.3.6 建立时间测试图形

建立时间和保持时间测试图形，如图 4-5 所示。

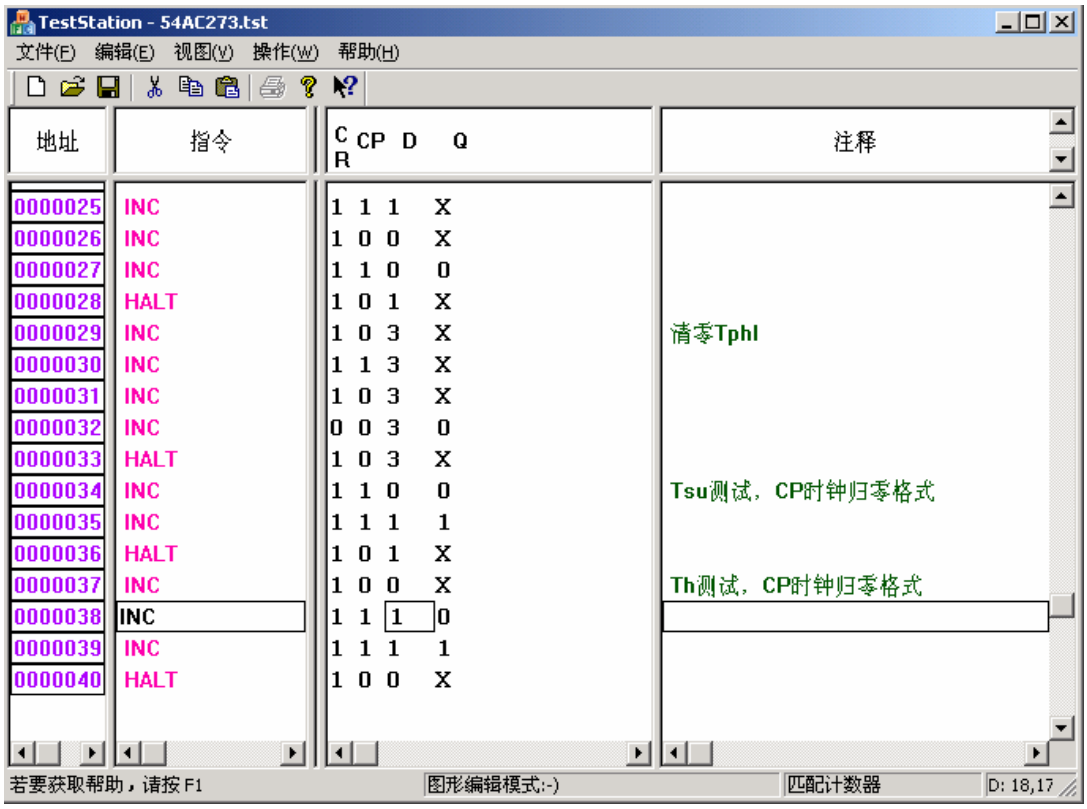


图 4-5 54AC273 建立保持时间测试图形

图中 34－36 行图形为建立时间测试图形，图中 37－40 行为保持时间测试图形。用户可以编写任意 D 端与时钟 CP 的的建立和保持时间。

4.3.7 保持时间 Th 测试示意图

保持时间测试示意图如图 4-6 所示。

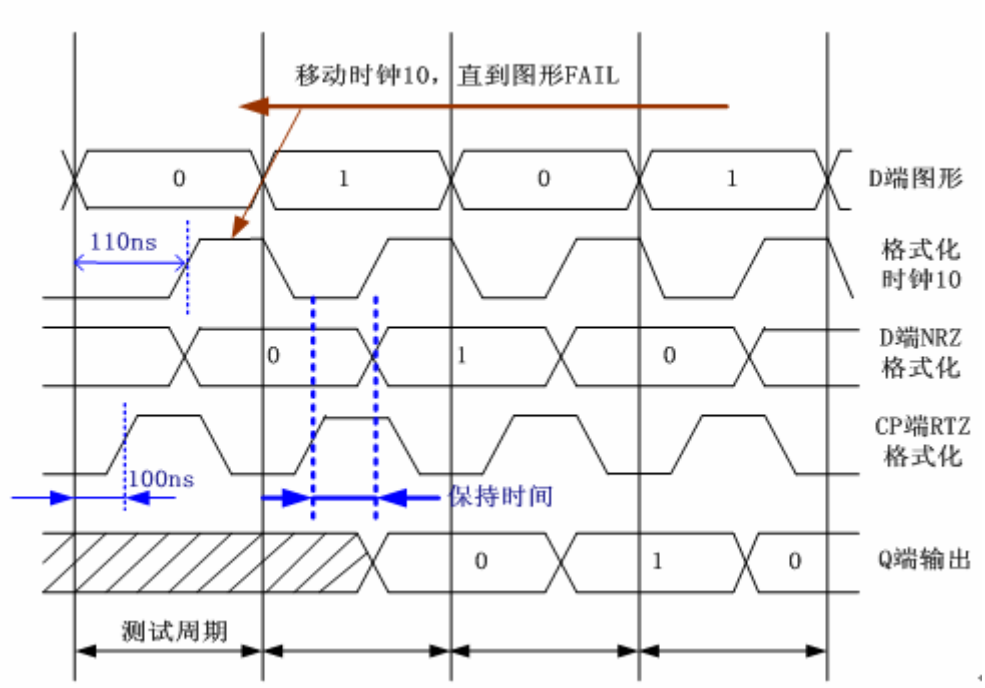


图 4-6 保持时间测试示意图

图 4-6 中 D 端为 273 的任意一个 D 触发器的输出端，Q 端为相应 D 触发器的输出端。D 端使用时钟 10 作为格式化时钟，设置为 NRZ 格式，时钟 10 起始延时为 110ns。时钟端使用时钟 12 为格式化时钟，设置为 RTZ 格式，时钟 12 起始延时为 100ns。运行测试图形，如图 4-5 所示，从 37 行开始运行。若图形通过，则减小时钟 10 的延时，直到运行图形失效。这时时钟 12 与时钟 10 的延时的差值就是建立时间。

4.3.8 保持时间测试 Th 测试代码

```
//测试项名： Th_45
MYDLLAPI double Th_45(double* grpTestValue,short* pingrp)
{
   double Th=0 ,tclk=100,tData=110 ;
   hvforce_v(3,4.5,150);
   set_dut_vih("1,3,4,7,8,11,13,14,17,18",4.0);//参考电平设置
   set_dut_vil("1,3,4,7,8,11,13,14,17,18",0.0);
   set_dut_voh("2,5,6,9,12,15,16,19",2.4);
   set_dut_vol("2,5,6,9,12,15,16,19",0.4);
   set_dut_timing("2,5,6,9,12,15,16,19",15,1);//Q 端比较时钟为 1
   set_cmp_clock( "1",400) ;    //Q 端比较时钟 400
   set_dut_mode("11",PT_RTZ) ;  //时钟格式为归零
   set_dut_timing("11",12) ;    //使用时钟 12
   set_ffmat_clock("12",tclk) ; //时钟前沿为时间周期+100ns
   set_dut_mode("3,4,7,8,13,14,17,18",PT_NRZ) ;//D 端格式化
   set_dut_timing("3,4,7,8,13,14,17,18" ,10) ; //格式化时钟
   set_ffmat_clock("10",tData) ; //D 端数据初值
   ptn_to_dut("1-9,11-19");
   delay_ms(20);
   while(1)
  {
     short fail=run_pattern(37,ST_FAIL) ;
     Tsu=tData -tclk ;
     if(fail) break ;
     tData -=0.5 ;   //时间步进为 0.5ns
     set_ffmat_clock("10",tData) ;
     if(Tsu<0)  break ; //防止死循环
  }
  return Tsu ;
```

其他交流参数的测试，用户可以参考这两个方法编写测试程序。

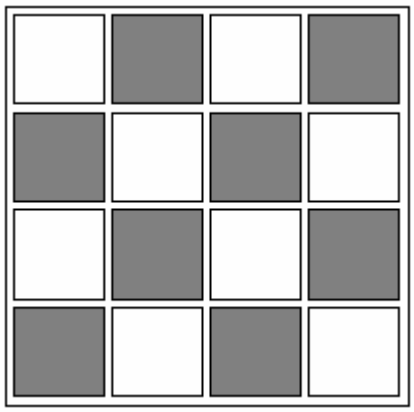
附录 A:存储器测试图形说明

A.1 全 0 全 1①

在存储器阵列中所有位都写入 1 或 0，然后全部读出

A.2 校验板（棋盘格图形，Checkerboard Pattern） ②

在存储器阵列中所有单元中交替写入 1 和 0，然后依次读出，然后再将原来位置取反（1 位置写 0，0 位置写 1）



A.3 地址互补图形③

（Address Complement Pattern），也称为多重地址选择

（1）在阵列所有单元中写入 0

（2）在（x,y）（x,y=0…n/2）读 0，写入 1，再在（~x,~y）单元中读 0 写 1，下一个单元，直到（row/2,col/2）

（3）在（x,y）（x,y=0…n/2）读 1，写入 0，再在（~x,~y）单元中读 1 写 0，下一个单元，直到（row/2,col/2）

A.4 步进法（Marching pattern）④

（1）在全部单元中写 0

（2）读A_0单元的 0，然后再改写为 1；然后读A_1的 0，再改写A_1为 1，直到A_{n-1}。这样所有单元均为 1

（3）然后从A_{n-1}到A_0做读 1 写 0 的操作

A.5 走步（walking pattern）⑤

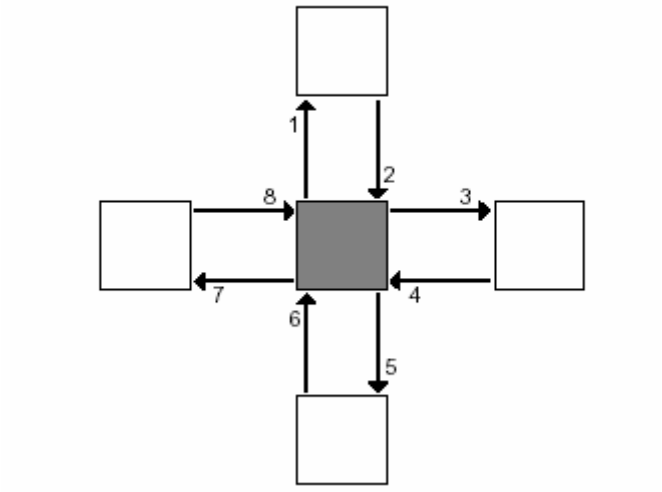
（1）把 0 写入全部地址单元中
（2）A0 写 1，从 A1 到 An-1 中读 0；
（3）从 A0 中读 1。检查其他地址连续读取对 A0 的影响。改写 A0 为 0
（4）A1 写 1，从 A2 到 An-1 中读 0；然后读 A1 中的 1，改写 A1 为 0
（5）重复上述步骤直到全部地址单元

A.6 走步（simple walking pattern）⑤

（1）把 0 写入全部地址单元
（2）在所有单元中执行以下操作
读 0，写 1，读 1 写 0

A.7 蝴蝶图案(butterfly pattern)⑥

（1）所有单元清 0
（2）将第一个单元写入 1，读相邻单元格的 0，再读第一个单元的 1，再清零
（3）完成后，将第一个单元写入 0，然后将第二个单元写入 1，重复第 2 步的过程，直到完成所有单元格的测试。



A.8 IMAG Pattern⑨

（1）全部单元写 0
（2）从（0，0）单元读 0，然后写 1，再写 0，再写 1
（3）从（0，1）单元读 0，然后写 1，再写 0，再写 1
（4）直到所有的单元格操作完毕
（5）然后再从（0,0）单元读 1，然后写 0，再写 1，再写 0

（6）再从（0，1）单元中读 1，然后写 0，写 1，再写 0

（7）直到全部单元

（8）然后从最后一个单元格（ROW，COL）读 0，写 1 再写 0

（9）反向操作直到（0，0）单元

***** 该方法比跳步快，而且可以检测出单元格之间的关联和耦合，对于大容量存储器测试非常有用。

A.9 乒乓（简化跳步图形）⑩

（1）将全部地址单元清 0

（2）A0 写 1，然后 A1 读 0，A0 读 1；然后 A2 读 0，A0 读 1；接着 A3 读 0，A0 读 1，直到全部单元

（3）将 A0 改写为 0，写 A1 为 1，重复以上步骤

A.10 行列乒乓（C/R Galloping pattern）⑧

与乒乓不同的是只进行测试该行或列上的单元具体如下：

（1）将所有单元格写入 0

（2）（R0，C0）单元写入 1，然后读该行的（R1,C0）的 0，读（R0，C0）的 1……直到该行的最后一个单元；然后读（R0,C0）的 1，再读第一列的（R1，C0）的 0 直到该列的最后一个单元；然后改写（R0，C0）单元为 0

（3）然后同样方法操作下一个单元格，直到全部单元格