数字反相器数据手册 DigInv V 1.5

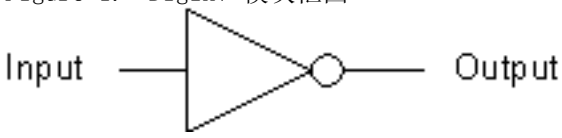
资源	PSoC® 模块			API 存储空间 （字节）		引脚 （每个外部 I/O）
	数字	模拟 CT 模块	模拟 SC 模块	Flash （闪存）	RAM	
CY8C29/27/24/22/21xxx, CY8C23x33, CY8CLED02/04/08/16, CY8CLED0xD, CY8CLED0xG, CY8C21x45, CY8C22x45, CY8CTMA140, CY8CTMA30xx, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xxx						
	1	0	0	57	0	1
CYWUSB6953	1	0	0	57	0	1

功能和概述

- 输出引脚是经过数字反相的输入引脚
- 仅需一个数字模块
- 可用于在输入信号的下降沿来产生一个中断。

DigInv 用户模块是一个简单的数字反相器。 输出信号是输入信号的逻辑非。

Figure 1. DigInv 模块框图



功能描述

DigInv 模块是一个单输入的数字反相器。 可将其映射至任何数字 PSoC 模块。 API 能启动和停止 DigInv 用户模块、启用或禁用它的中断功能。 当 DigInv 模块是停止状态时，输出信号会保持在低电平。

直流和交流电气特性

Table 1. DigInv 模块的直流和交流电气特性

参数	条件和注意事项	典型值	限定	单位
来自 Global Bus 的输入 F_{max}			12	MHz
来自内部连接的输入 F_{max}			48	MHz
输入到输出转换		< 25		ns
输出到 Global Bus 的 F_{max}			12	MHz
输出到内部连接的 F_{max}			48	MHz

时序

当输入信号或输出信号与 global bus 连接时，DigInv 用户模块转换速率限定在 12 MHz 。 要达到更高转换速率，可将 DigInv 用户模块放置在能提供输入或接收输出的用户模块的附近，使 Table 1 的其他连接选项也可以被选择。 将 DigInv 放置在 PSoC 模块 DBA03 中，使输出可以作为输入连接到任何其他数字 PSoC。

放置

可以将 DigInv 模块放置在任何 PSoC 的数字模块中。

参数和资源

输入

从 16 个源之一选择输入。 这些源包括 48 MHz 振荡器、从 24 MHz 系统时钟细分出来的较低频率（24V1 与 24V2）、其他 PSoC 模块，以及通过全局输入和输出 (global inputs and outputs) 连接的外部输入。

输出

可以经过四个全局输出 (global output) 信号之一来输出。

ClockSync

如果要将 DigInv 用户模块的输出信号用作 PSoC 内其他模块的输入信号或一个时钟，我们建议使输入与内部系统时钟同步。 时钟的选择取决于输出信号在内部的路径，在下表中进行详述。

ClockSync 值	使用说明
Sync to SysClk（同步到系统时钟）	在输出信号经由使用 24 MHz (SysClk)、或使用二分频或多分频的 SysClk 时钟源的模块时，使用该设置。 例如有 VC1、VC2、VC3 （当 VC3 由 SysClk 驱动），32KHz。
Sync to SysClk*2	在输出信号经由使用 48 MHz (SysClk*2) 、或使用基于 SysClk*2 的时钟的模块时，使用该设置。
Unsynchronized	在需要非同步的输入信号时使用该设置。 通常，只有在将输出信号直接连接到一个引脚、或为了产生中断时，才建议使用。

中断产生的控制

当选中 PSoC Designer 中的 “ 中断产生的控制 ” 复选框时，有两个附加参数变为可用。 **中断产生的控制** 复选框时，会有一个附加参数变为可用。 可以在以下菜单下找到此复选框： **项目 > 设置 > 芯片编辑器** . 当外覆层的多个用户模块所共享的中断用于多个外覆层时，中断生成控制非常重要：

- 中断 API
- IntDispatchMode

InterruptAPI

InterruptAPI 参数允许有条件生成用户模块的中断处理程序和中断矢量表条目。 选择 “ 启用 ”(Enable) 可生成中断处理程序和中断向量入口。 选择 “ 禁用 ”(Disable) 可避免生成中断处理程序和中断向量入口。 在那些拥有多个外覆层而且有多个外覆层使用同一块资源的项目中，特别推荐要正确地选择是否要生成中断 API。 仅在必要时选择生成中断 API，这样可以避免生成中断处理的代码，从而减少开销。

IntDispatchMode

IntDispatchMode 参数用于指定中断请求的处理方式，这些中断由同一块不同外覆层中的多个用户模块共享。 选择 “ActiveStatus” 会导致固件在为共享的中断请求提供服务之前测试哪一个外覆层正处于活动状态。 每次请求共享中断时，都会进行此测试。 这会增加延迟，还会产生不确定的为共享中断请求提供服务的过程，但是不需要任何 RAM。 选择 “OffsetPreCalc” 参数会导致固件仅在最初加载重叠层时计算共享中断请求的来源。 这种计算可减少中断延迟，并产生为共享中断请求提供服务的确定过程，但会占用一个字节的 RAM 空间。

应用程序编程接口

提供的应用程序编程接口 (API) 子程序作为用户模块的一部分，允许设计人员能够采用更高级的方式处理模块。 本节详细描述每个函数的接口，以及 “ 引用 ”(include) 文件所提供的相关常量。

Note

在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能通过调用 API 函数进行更改。 如果在调用后需要 A 和 X 的值，则调用函数负责在调用前保留 A 和 X 的值。 选择此 “ 寄存器易失 ” 策略是为了提高效率，自 PSoC Designer 1.0 版起已强制使用此策略。 C 编译器自动遵循此要求。 汇编语言程序员也必须确保其代码遵守这一策略。 尽管一些用户模块的 API 函数可以保留 A 和 X 的值不变，但是无法保证它们将来也会如此。

对于大容量存储器模块， 保存 CUR_PP、IDX_PP、MVR_PP 以及 MVW_PP 寄存器中的所有值也是调用函数的职责。 尽管部分寄存器现在可能不可修改，但是无法保证在将来的版本中也会如此。

以下是为 DigInv 模块提供的 API 编程子程序。

DigInv_Start

说明：

开始 DigInv 操作。

C 原型：

```
void  DigInv_Start(void);
```

汇编：

```
lcall  DigInv_Start
```

参数：

None

返回值：

None

副作用：

可通过该函数修改寄存器 A 和 X。 在调用 DigInv_Start 函数前，其标志将被设置为 FALSE，然后可以在 ISR 中去查看。 如果 ISR 发现标志的值为 TRUE，则它执行 ISR 代码；如果标志的值为 FALSE，则它将标志设为 TRUE 并退出，且不执行剩下的 ISR 代码。

DigInv_Stop

说明：

停止 DigInv 操作。 输出将保持低电平。

C 原型：

```
void  DigInv_Stop(void);
```

汇编：

```
lcall  DigInv_Stop
```

参数：

None

返回值：

None

副作用：

可通过该函数修改寄存器 A 和 X。

DigInv_EnableInt

说明：

启用中断模式操作。

C 原型：

```
void DigInv_EnableInt(void);
```

汇编：

```
lcall DigInv_EnableInt
```

参数：

None

返回值：

None

副作用：

可通过该函数修改寄存器 A 和 X。

DigInv_DisableInt

说明：

禁用中断模式运行。

C 原型：

```
void DigInv_DisableInt(void);
```

汇编：

```
lcall DigInv_DisableInt
```

参数：

None

返回值：

None

副作用：

可通过该函数修改寄存器 A 和 X。

固件参考源代码

以下汇编语言源代码说明了 API 的使用。

```
;-----------------------------------------------------
; Example assembly program using DigInv User Module
;-----------------------------------------------------

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

_main:

    lcall  DigInv_EnableInt  ; Use if interrupts desired
    lcall  DigInv_Start      ; Enable inverter

; Place user code here.

.terminate:
    jmp .terminate
```

同一代码用 C 语言表示如下：

```
//*****************************************
//  Example C program using DigInv User Module
//
//*****************************************
#include    "M8C.h"
#include    "PSoCAPI.h"

void main(void)
{
DigInv_EnableInt();  // Use if interrupts desired
   DigInv_Start();      // Enable Inverter

// Rest of User code
}
```

配置寄存器

下面几个表格中描述了用于配置 DigInv 用户模块的 PSoC 数字块。仅解释参数化的符号。

Table 2. DigInv 模块：寄存器函数

位	7	6	5	4	3	2	1	0
值	0	0	1	0	0	0	0	0

Table 3. DigInv 模块：寄存器输入

位	7	6	5	4	3	2	1	0
值	0	0	0	0	输入			

输入信号从 16 个源中的 1 个源选择输入，并在器件编辑器中进行设置。

Table 4. DigInv 模块：寄存器输出

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	启用输出 (Out Enable)	Out Sel	

“启用输出”(Output Enable) 标志用于表示输出已启用。Output Sel 标志指示 DigInv 模块的输出信号将连接何处。这两个参数都是在器件编辑器中设置的。

Table 5. DigInv 模块：计数寄存器 DR0

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	0	0	0

Table 6. DigInv 模块：周期寄存器 DR1

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	0	0	0

Table 7. DigInv 模块：数值比较 (CompareValue) 寄存器 DR2

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	0	0	0

Table 8. DigInv 模块：控制寄存器 CR0

位	7	6	5	4	3	2	1	0
值	0	0	0	0	0	0	0	使能 (Enable)

设置了 Enable，表示已启用 DigInv。使用 DigInv API 对其进行修改。

版本历史记录

版本	创作者	说明
1.5	DHA	添加了版本历史

Note PSoC Designer 5.1 在所有用户模块数据手册中都引入了 “ 版本历史 ”。 本数据表详细介绍了当前和先前用户模块版本之间的区别。