

---

## DSP 实践课程疑问

---

2016 年 5 月 5 日

1) 工程目录的完整路径为：

C:\GForge\starterware\dev\C6748\_StarterWare\_1\_20\_03\_03

导入工程的时候，以下两个选项都请不要勾选，特别是第二个选项。

- ☐ Copy projects into workspace
- ☐ Automatically import referenced projects

2) 导入工程或者编译时提示没有相应 compiler 版本的：

点击 CCS 菜单栏 Help->Install New Software...;

弹出 Install 窗口，在 Work with: 栏目中选择--All Available Site--

在出现的选项中找到 TI Compiler Updates，展开；

找到需要安装的编译器 Compiler，对于本课程用到的 C6748 的板子，我们需要安装的编译

器是 C6000 Compiler Tools，版本为 7.4.xx（高于等于 7.4.0 的应该都能支持），之后

一路 next 即可在线下载安装。

3) 查看函数功能和参数配置的参考文档为：

C:\GForge\starterware\dev\C6748\_StarterWare\_1\_20\_03\_03\docs\C6748\_StarterWare\_1\_20\_03\_03.chm

4) CCS 导入工程时选择的工程是以 LCDK 结尾的，EVM 结尾的工程是上一届使用的开发板。

两者有一些不同。

5) 该板为 DSP+ARM9 的双核心开发板，实验中只使用到其中的 DSP 核心->C6000

TMS320C6748，ARM9 核心不关心。该芯片适合于图像、音频、视频处理等，对于电气类的

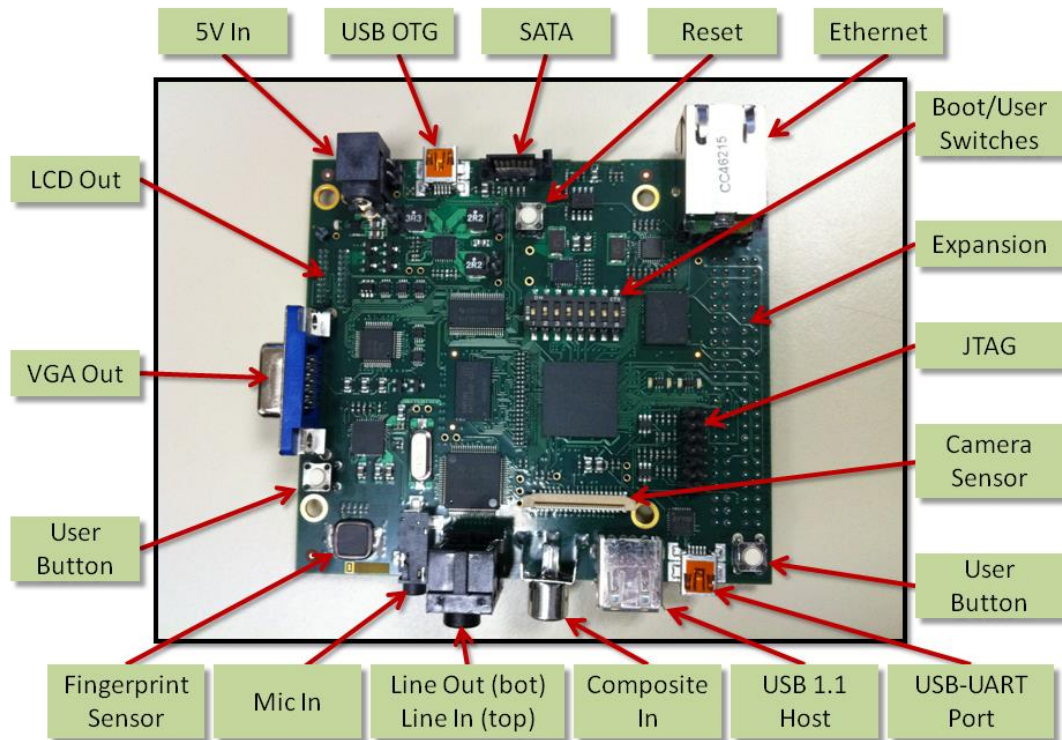
控制并不是很合适。电气类使用的较多的仍是 C2000 系列。使用 C6000 是系里要求。使用

到的外设与 C2000 基本是类似的。

6 )

[http://processors.wiki.ti.com/index.php/L138/C6748\\_Development\\_Kit\\_\(LCD\\_K\)](http://processors.wiki.ti.com/index.php/L138/C6748_Development_Kit_(LCD_K))

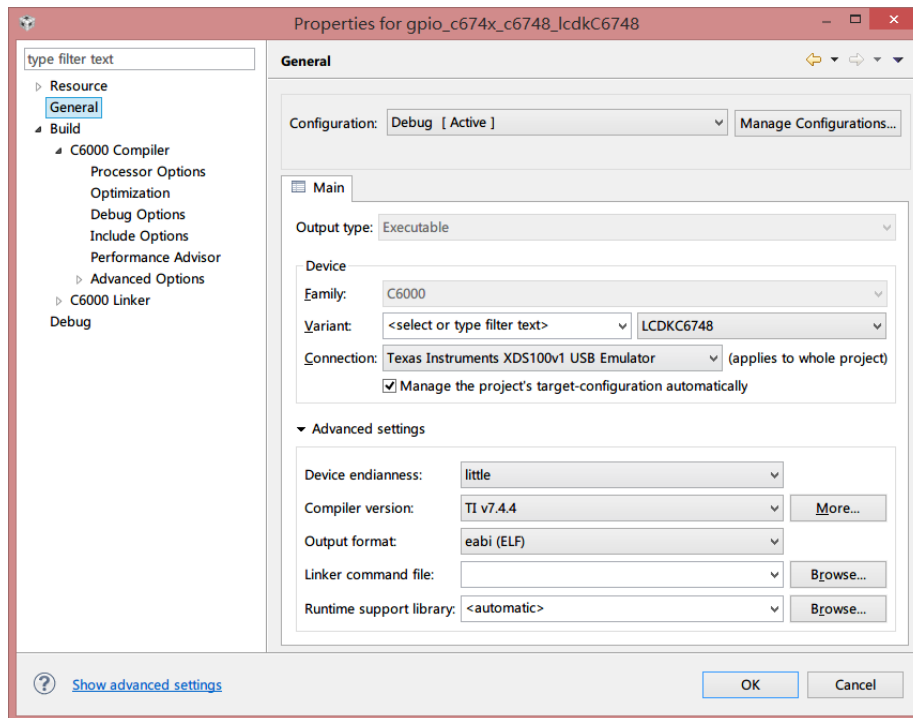
以上是 TI 官网对 LCDK 的概览介绍



7 ) 编译之前先设置工程属性，点击菜单栏 **Project-Properties**，如下图配置：

编译器版本问题的请参考条目 2 )

。 **Linker command file** 不要选择。使用工程中默认的 **.cmd** 文件即可。一个工程中只能同时编译一个 **.cmd** 文件。



8) 早上有一个 BUG 需要更正一下。LCDK 板上的 FT232R 芯片仅仅是 USB-SERIAL 转换芯片,没有仿真功能,此处疏忽了。下节课使用的时候需要使用 XDS100 系列的仿真器与 JTAG 口连接。

9) 修改过 VID\_PID 的同学,前期实验如果不使用串口,那么不影响使用。如果使用串口必须重新安装 FT232R 驱动。但是重新安装驱动会遇到找不到驱动的问题->因为修改了 VID\_PID,驱动安装的时候是需要配对 VID\_PID 的,修改了之后即使找到了正确的驱动也因为当前硬件设备 EEPROM 的 VID\_PID 不匹配而无法安装。解决该问题的办法是修改驱动文件。

附件中提供了两个驱动文件夹。

第一个文件夹是“ft232r usb uart 驱动 ( oroginal )”,是 FT232R 官方的驱动未经修改。

第二个是“ft232r usb uart 驱动 ( oroginal )”,是我在原版驱动基础上修改过的。

没有修改过 VID\_PID 的同学,连接板子后,如果在设备管理器中的通用串行总线控制器没

有 USB Serial Converter 设备或者设备驱动没有正确安装的 ,请直接手动安装原版驱动。



修改过 VID\_PID 的同学，需要重新安装驱动，请安装修改后的 ft232r 驱动。

特别注意：如果是 WIN8 或者 WIN8.1 系统，安装修改后的驱动，会提示错误“文件的哈希值

不在指定的目录文件中”，那么必须禁用驱动程序强制签名后再安装驱动。

禁用驱动程序强制签名请看以下教程（或者自行上网搜索）。

<http://jingyan.baidu.com/article/7c6fb42879543380642c9036.html>

10) 下节课开始做实验，包括 GPIO 和 TIMER 中断。

**2016 年 5 月 7 日**

1) 掌握 GPIO 的主要（寄存器）配置，包括引脚复用、使能、数据读写；

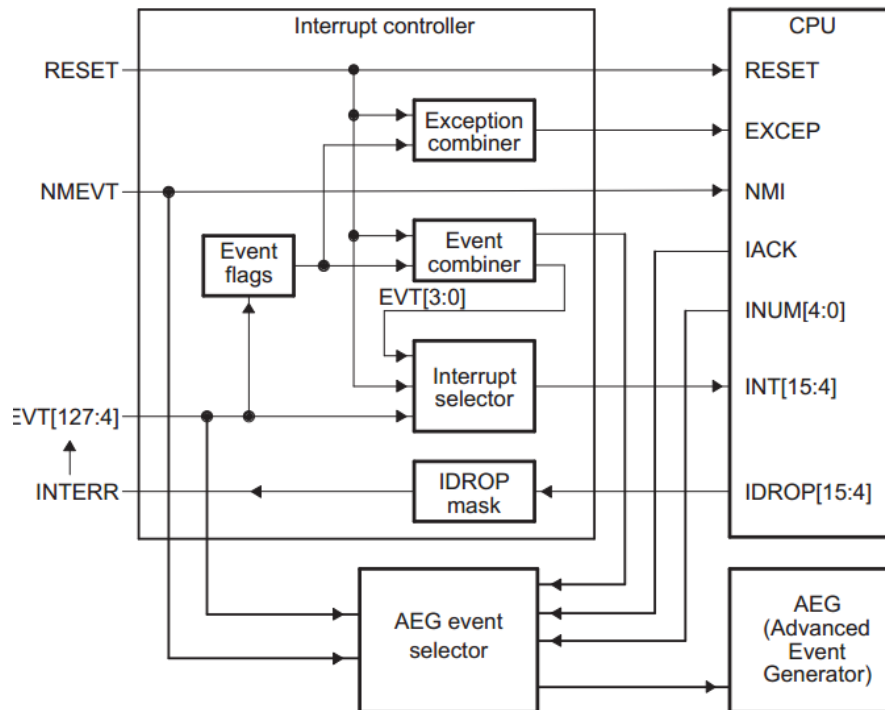
2) 了解系统时钟的使能与初始化，了解每个外设的时钟入口与其对应频率，参考

《TMS320C6748 DSP System Reference Guide.pdf》第六章和第七章；

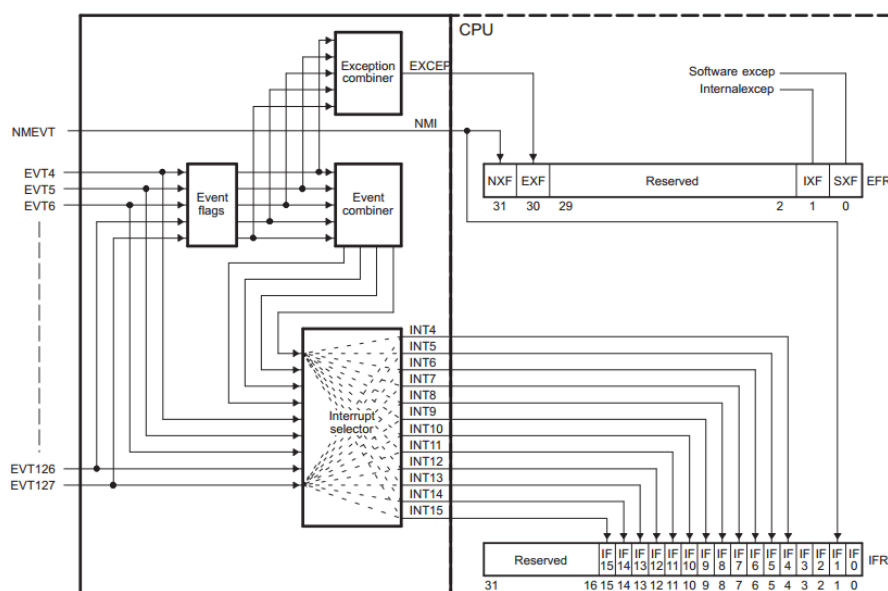


- 3) 掌握 TIMER 计时器的配置，区分时钟和计数器时基；
- 4) 掌握 TIMER 中断映射，掌握系统中断工作响应流程，参考《TMS320C674x DSP Megamodule Reference Guide.pdf》第七章；

**Figure 7-1. C674x Megamodule Interrupt Controller Block Diagram**



**Figure 7-14. CPU Event Routing Diagram**



- 5) 了解 DSP 操作系统最简化启动流程：bootloader 引导内核（简单初始化，主要是内存模块等）->内核启动（系统初始化）->用户初始化（入口为\_main，用户操作完成后控

制权转交操作系统，有 while 死循环除外）->事件响应。

C6000 和 C2000 系列的操作系统为 TI-RTOS（老版本为 SYS/BIOS 和 DSP/BIOS）实时操作系统。

实时操作系统 RTOS：TI-RTOS、UcosII、vXWorks、一些协议栈如 Zigbee 协议栈等；

分时操作系统：linux、windows、unix，一些协议栈如 TCP/IP 协议栈等；

6) 学会寄存器操作，包括 CPU 和外设的。

7) 前四周做简单的外设实验，后四周大作业 FFT。

**2016 年 5 月 12 日**

1) 本周二任务：使用 GPIO 寄存器控制 4 个 LED 灯(连续的,位置位于 JTAG 烧写口附近)。

本周二早上鉴于连接 JTAG 等出现了各种问题，因此任务本周四下课前验收。

任务具体要求：控制 LED 等实现跑马灯功能。假设从左到右分别为 LED0, LED1, LED2, LED3。

1、基本要求：按以下流程重复（时间间隔**大约**是 500ms）：

LED0 亮（其他全灭，以下默认），LED1 亮，LED2 亮，LED3 亮，LED2 亮，LED1 亮，LED0 亮（此处回到最开始状态，循环往复）；

2、加分项 1：按以下流程重复（时间间隔**大约**是 500ms）：

LED0 亮（其他全灭），LED1 亮（其他全灭），LED2 亮（其他全灭），LED3 亮（其他全灭），

LED2 亮（其他全灭），LED1 亮（其他全灭），LED0 亮（其他全灭）；

LED0 LED1 亮（其他全灭），LED0 LED1 LED2 亮（其他全灭），全亮；

LED0 灭（其他全亮），LED1 灭（其他全亮），LED2 灭（其他全亮），LED3 灭（其他全亮），

LED2 灭（其他全亮），LED1 灭（其他全亮），LED0 灭（其他全亮），全灭。

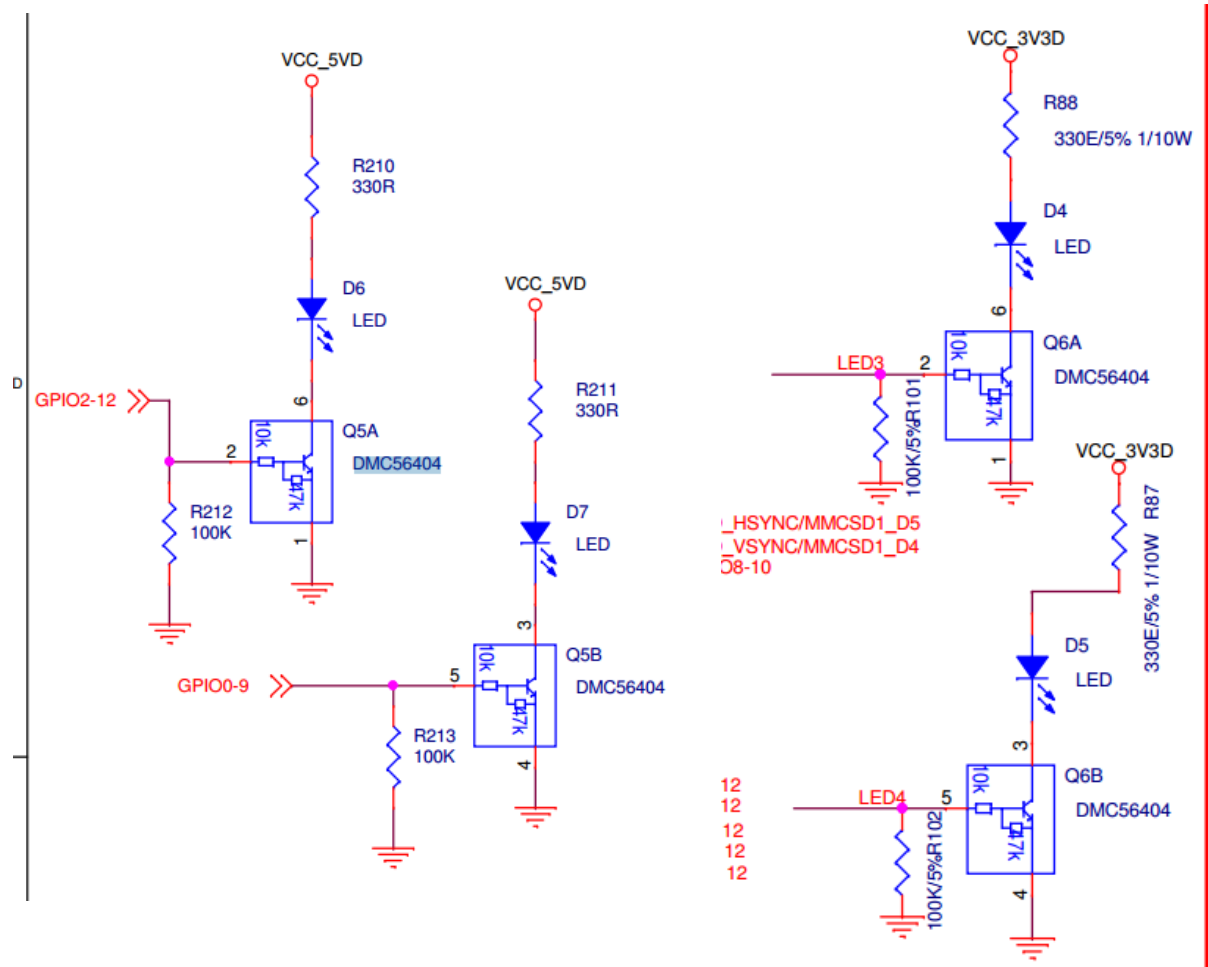
回到第一个状态。

3、加分项 2：按加分项 1 的流程重复（时间间隔为精确的 500ms）。

提示：使用位操作，移位。

2）上述讲到的 LED0，LED1，LED2，LED3 分别对应哪个 GPIO？

附件有一份文档《OMAP-L138\_C6748 LC Dev Kit Ver A6a.pdf》是该开发板的 SCH 图。



具体的 GPIO 请参照本文档，周四课上再讲解。

2016 年 5 月 14 日

1）当前课时未完成任务，下一课时可以再检查。

2）宏定义

```
#define _HWREG(x) (*(volatile Uint32*)(x))
#define PINMUX0 _HWREG(0x01C14120)
#define PINMUX5 _HWREG(0x01C14134)
#define PINMUX13 _HWREG(0x01C14154)
```



HWREG(x), x 为寄存器地址, 32 位, 先强制转换为 unsigned int 的指针, 再取出该指针指向的地址的内容。

PINMUXx, 定义了几个用到的寄存器的地址。

### 3) 函数实现

先清空相应的位, 再写入设置值。之前给的程序有点遗漏, 清空应该是使用 PINMUXx &=

~(0xf << 12), 这样才能连续清空四位。

```
void _GPIOBank6Pin12PinMuxSetup()
{
    PINMUX13 &= ~(0xf << 12);
    PINMUX13 |= (0x8 << 12);
}
```

```
void _GPIOBank6Pin13PinMuxSetup()
{
    PINMUX13 &= ~(0xf << 8);
    PINMUX13 |= (0x8 << 8);
}
```

```
void _GPIOBank2Pin12PinMuxSetup()
{
    PINMUX5 &= ~(0xf << 12);
    PINMUX5 |= (0x8 << 12);
}
```

```
void _GPIOBank0Pin9PinMuxSetup()
{
    PINMUX0 &= ~(0xf << 24);
    PINMUX0 |= (0x8 << 24);
}
```

### 4) 程序简单讲解: GPIO使用前需要先设置引脚复用。一个PIN可能有多种功能, 作为外设引

脚或者简单的GPIO引脚使用。使用SYSCFG Registers下的PINMUXx寄存器进行修改。

详细寄存器设置内容参考《TMS320C6748 DSP System Reference Guide.pdf》

设置好复用为GPIO后, 设置GPIO方向(0输出1输入), 即可修改输出电平了。

### 5) 最简单的延迟使用了delay, 但是很没有效率, 也不准确。精准一些的延迟使用timer进行

延时。参考timer例程。timer的周期，sysclock的配置在前面的课时已经讲解。

6) 出现以下bug的：

```
Debug Assertion Failed! File: isctype.c Line: 56 Expression (unsigned)
(c+1) <=256
```

可能与路径中包含有中文有关。不仅仅是当前路径，还需要检查CCS安装路径，workspace路径，等。本人前两周遇到该问题，无法解决。修改用户名为英文最终解决了这个问题。。。

7) CCS请尽量使用管理员身份运行。

8) 其他编译上出现的问题，请查看前面章节中涉及到的配置，并且保证一些重要文件，

如.cmd，.ccxml无重复的情况下，**Rebuild Project!**

9) 其他编译上遇到的问题，需要大家自行上网搜索查找解决方案。问题实在太多了，但是大

部分还是因为修改了路径、修改了配置方案导致的。请严格按照上述提到的路径进行配

置。如果修改了路径，可能可以运行，但是运行之后无法通过Stop Into查看具体函数实现。

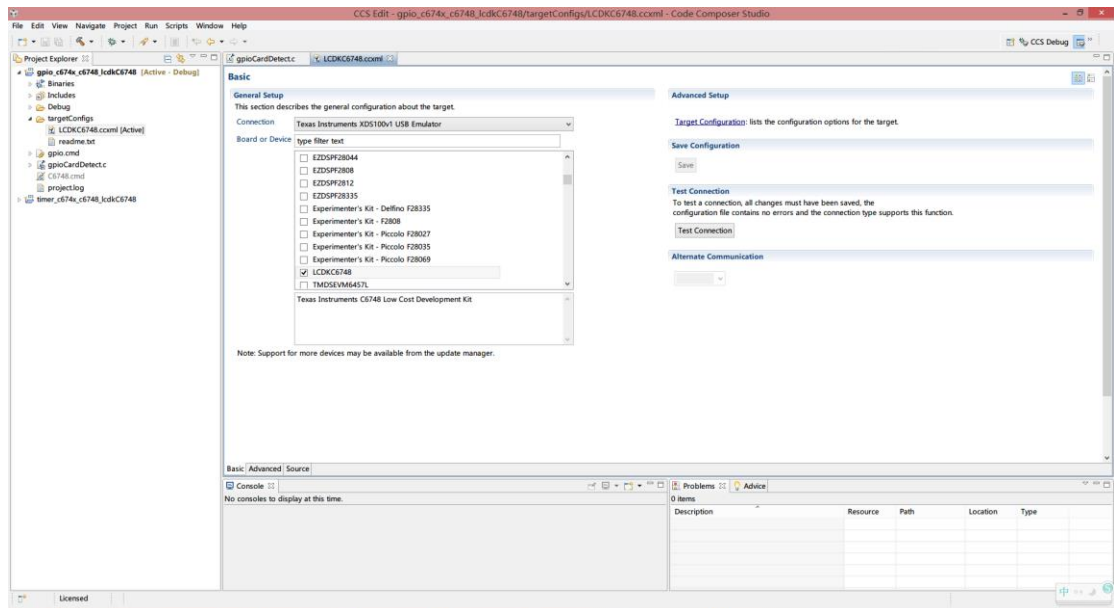
记住：本函数库非TI官方！TI官方的函数库是使用了直接寄存器操作，我想对大家理解上会比较困难，所以找到了这个代理商提供的函数库。因此，请严格按照前面章节提到的路径等进行配置。

10) 烧写时遇到error 151的，基本都是连接的问题。请查看设备管理器中有没有XDS100 A和

XDS100 B这两个设备！如果存在的话，打开工程中的.ccxml文件，点击Test Connection

进行测试，看最终的连接测试结构是否succeed！再连接不上请关掉CCS，拔掉JTAG仿真

器，再打开CCS然后连接仿真器，再烧写一次！



11) 下节课的内容包括：

1. RTC 实验，比较简单，了解一下 RTC 的工作原理和配置方式；
2. WATCHDOG 实验，Timer 的另一种配置；
3. UART 实验（第二个实验任务）：需要配合 GPIO 和 TIMER 完成简单的状态设置，尽量都能使用中断完成。

实验内容大致如下：

配置 UART 为输入中断，配置 TIMER 中断，TIMER 周期为 LED 状态切换间隔；

上位机 sscom 接收用户输入的 LED 状态字，发送到 UART，触发 UART 中断，用户读取状态字，实时将状态字转换为 LED3，LED2，LED1，LED0 的闪烁变化。

12) 第八节课可能会涉及到的外设内容有：

1. EPWM(会有实验任务三)

EPWM 实验内容暂定。

2. SPI（暂定）

3. EDMA（比较难，暂定）

4. I2C ( LCDKC6748 上的 I2C 例程实在太不直观，暂定 )

13 ) 最后的 6-8 节课做大作业。期间只提供答疑。

**2016 年 5 月 19 日**

1 ) 第二次实验要求 :

**基本要求 ( 70 分 ) :**

配置 UART 为输入中断，通过上位机 sscom 接收用户输入的 LED 状态字 ( 只需要一次接受一个 char 类型的数据即可 )，发送到 UART，触发 UART 中断，用户读取状态字，实时将状态字转换为 LED3，LED2，LED1，LED0 的闪烁变化。LED 的闪烁根据以下描述：

假设 sscom 上用户输入了 '1' 字符，UART 接受到的该字符为 char 类型，ASCII 码对应的 16 进制数为 0x31，即 0b00110001，占据了一个 BYTE 的内存空间。我们只有 4 个 LED 灯，无法一次显示 8bit，因此分两次显示：

第一次显示低 4bit，即 0b0001，此时 LED3，LED2，LED1 灭，LED0 亮；

第二次显示高 4bit，即 0b0011，此时 LED3，LED2 灭，LED1，LED0 亮。

两次显示切换间隔使用 delay 函数，延迟大概 0.5s 即可。

**扩展要求一 ( 85 分 ) :**

在基本要求的基础上使用 Timer 进行精确延迟。

**扩展要求二 ( 100 分 ) :**

在扩展要求一的基础上，修改程序，使得程序可以一次接受最多 100 个字符，并以此使用 LED 显示其状态。此时 n 个字符的 LED 显示需要 n 秒。

**提示：**

配置 UART 和 TIMER 中断的时候，有一个地方需要大家特别注意：

Timer 的一个配置函数 static void TimerIntrSetup(void) 中有以下语句：

```
IntDSPINTCInit();
IntRegister(C674X_MASK_INT4, TimerIsr);
IntEventMap(C674X_MASK_INT4, SYS_INT_T64P2_TINTALL);
IntEnable(C674X_MASK_INT4);
IntGlobalEnable();
```

UART 的中断配置函数包括了：

**static void SetupInt(void)**，包含以下语句：

```
IntDSPINTCInit();
IntGlobalEnable();
```

**static void ConfigureIntUART(void)**，包含了以下语句：

```
IntRegister(C674X_MASK_INT4, UARTIsr);
IntEventMap(C674X_MASK_INT4, SYS_INT_UART2_INT);
IntEnable(C674X_MASK_INT4);
```

这三个函数的配置需要有一定的先后顺序，配置的时候请按照以下顺序：

```
IntDSPINTCInit();
IntGlobalEnable();
IntRegister(C674X_MASK_INT5, UARTIsr);
IntEventMap(C674X_MASK_INT5, SYS_INT_UART2_INT);
IntEnable(C674X_MASK_INT5);
IntRegister(C674X_MASK_INT4, TimerIsr);
IntEventMap(C674X_MASK_INT4, SYS_INT_T64P2_TINTALL);
IntEnable(C674X_MASK_INT4);
```

让IntDSPINTCInit()这个函数运行之后再去配置其他的中断映射。同时注意到上面的两

个中断对应的中断映射也被修改了，为了避免两个中断公用一个CPU中断入口。

配置好两个外设的中断之后，后面需要做的就是逻辑方面的整理了。该部分不会有太大难度。

扩展二提到的一次接受最多100个字符，可以使用一个数组来完成，但是要注意数组的更新。必须考虑到接收字符的时候和显示LED的时候都需要更新该数组。请做好两者之间的同步。

2) 实验二的检查最迟可以到下周四上课。下周二讲解最后的模块EPWM。之后分配大作业。修

改一下进度，使用三周的时间完成大作业。

2016年5月26日

## 1) Shadow Register?

Shadow Register的概念，其实是个和硬件有关的概念。

有些register是2层的，第一层是供CPU访问，第二层供Hw访问。

### CPU访问

### Hw访问

其中CPU访问的这层register称之为Shadow Register。

CPU在写Register的时候，会先写在上层的Shadow Register，随后硬件update之后才会会在下层供Hw访问的Register开始执行。

这是同一个Register，不是2个Register，只不过分了2层。形象的讲上层是下层的Shadow。

因为真正生效的执行Hw动作的是下面这层，而上面这层只是将CPU（也就是将软件）的信息获取到，等下个硬件周期才会执行。

在PWM模块里面经常用到影子寄存器，一般有周期寄存器，占空比寄存器，相位寄存器等会有影子寄存器。

具体的作用是让用户可以选择立即更新这些寄存器，还是在某个特定的时间点再去更新这个寄存器。

举占空比的例子：

如果选择立即更新模式，当你在任何时刻写占空比寄存器，都会立即生效。

如果选择shadow模式，如果一个PWM周期期间写占空比寄存器，写到真正的占空比寄存器中立即生效，而是保存在影子寄存器中。在下个周期开始的时候再从影子寄存器中装载到占空比寄存器。

## 2) EPWM时基同步EHRPWMTIMEBASESYNC

时基同步用于多个EPWM模块需要保持一定的相位差的情况下。

3) 掌握计数模式（向上计数，向下计数，上下计数），Action Qualifier，CMPA、CMPB在

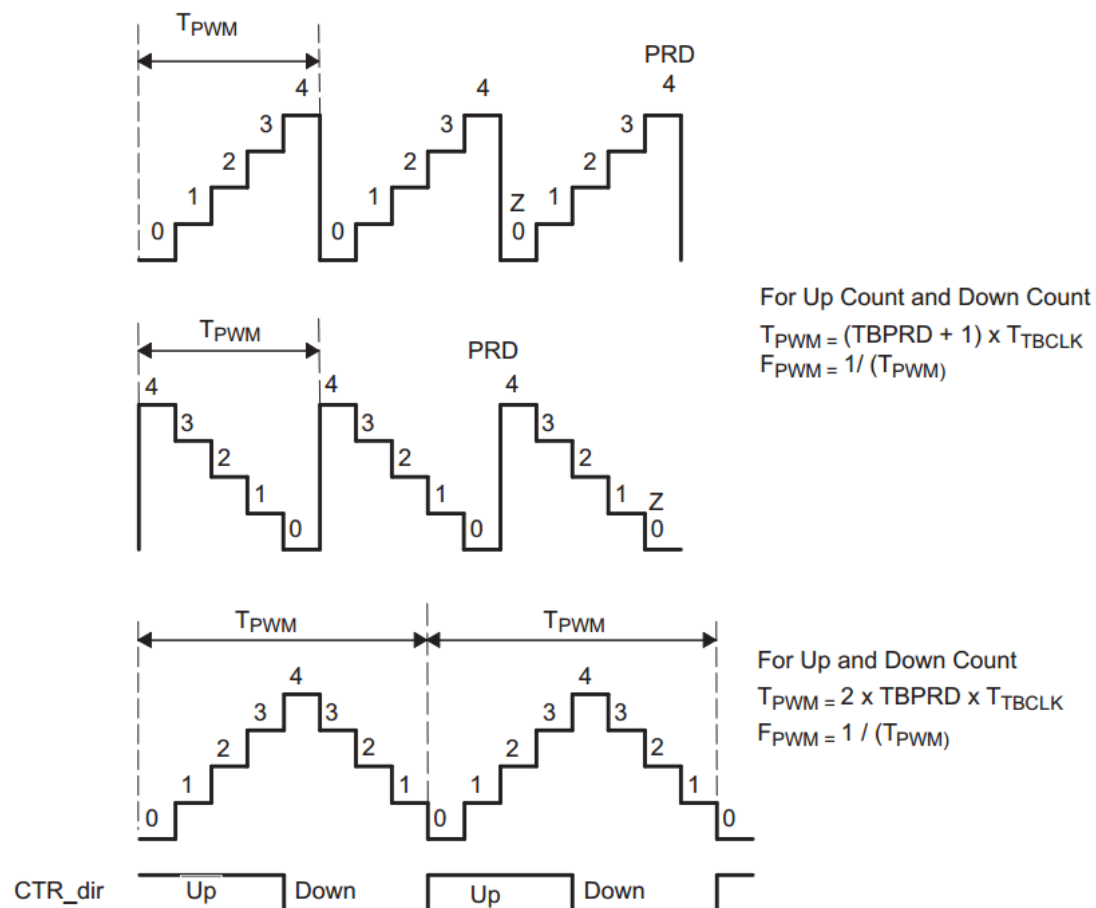
不同计数模式下的作用。

计数模式：

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):





















- **Up-Down-Count Mode:** In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.
- **Up-Count Mode:** In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.
- **Down-Count Mode:** In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

**Figure 6. Time-Base Frequency and Period**



Action Qualifier:

**Figure 20. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
					Do Nothing
					Clear Low
					Set High
					Toggle

- 4) 了解TripZone (一般用于保护), TZ引脚检测外部电平; TripZone事件发生后, 会按照事先配置好的动作对EPWM的输出做出改变: 强制拉低, 强制拉高或高阻。
- 5) 了解EPWM中断配置: 可以配置相同的某个事件发生N次后触发一次EPWM中断, 如计数器周期重复三次后出发一次中断等等。

```
void EHRPWMETIntPrescale ( unsigned int  baseAddr,
    unsigned int  prescale )
    This API prescales the event on which interrupt is to be generated.
```

**Parameters:**

baseAddr            Base Address of the PWM Module Registers.  
prescale            prescalar value

**Returns:**

None

```
void EHRPWMETIntSourceSelect ( unsigned int  baseAddr,
    unsigned int  selectInt )
```

This API selects the interrupt source.

**Parameters:**

baseAddr    Base Address of the PWM Module Registers.  
selectInt    Event which triggers interrupt. The possible source can



be,

- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUZERO
- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUPRD
- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUCMPAINC
- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUCMPADEC
- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUCMPBINC
- EHRPWM\_ETSEL\_INTSEL\_TBCTREQUCMPBDEC

**Returns:**

None

6) 掌握死区设置；

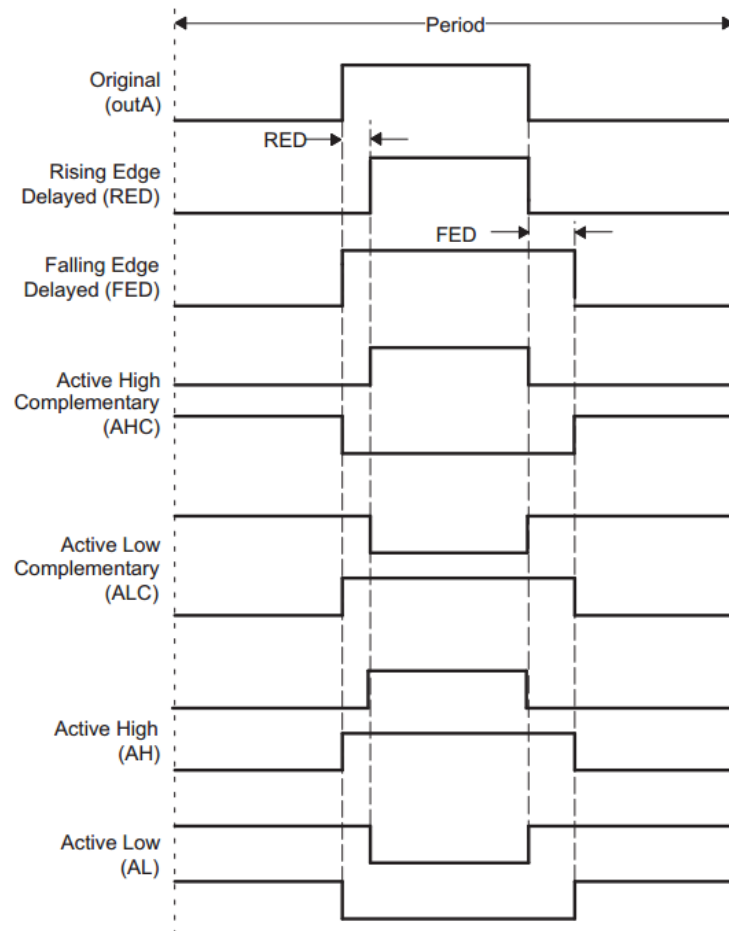
死区可以设置上升沿死区，下降沿死区；可以分别设置以EPWMxA或者EPWMxB作为基准；可

以设置死区输出（只输出上升沿、下降沿死区的其中一个或者两个都输出或者两个都不输

出）；

死区输出极性四种模式：

**Figure 30. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)**

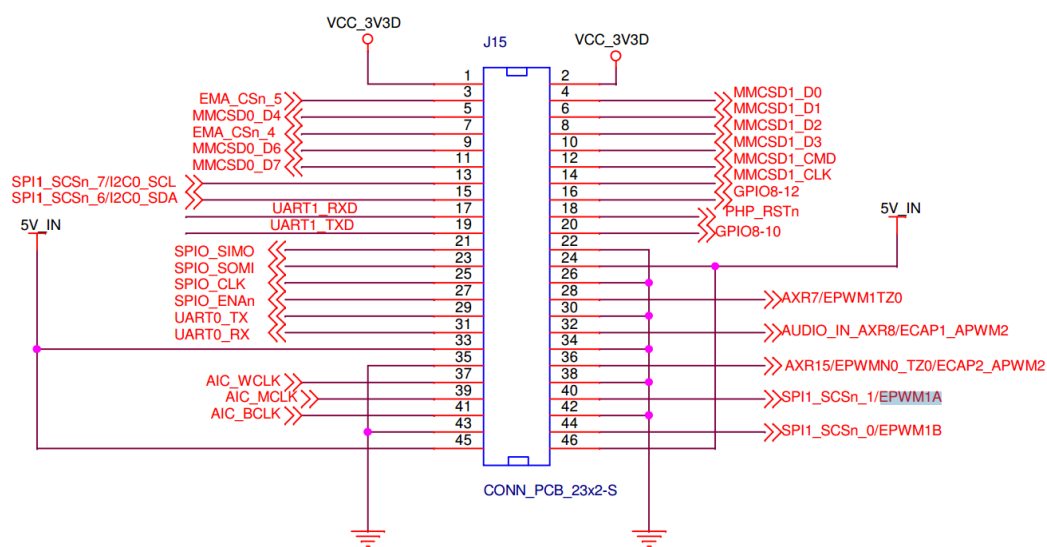


7) 第三次实验要求>>EPWM配制：输出两路EPWM1A和EPWM1B，频率为15KHz，EPWM1A占空比为25%；EPWM1B与EPWM1A互补；设置死区为1us，以EPWM1A为上升沿、下降沿参考源；死

区上升沿、下降沿均有输出，使用Active Low Complementary极性模式输出。

其他要求：TZ禁止，斩波模式禁止，中断源选择为EHRPWM\_ETSEL\_INTSEL\_TBCTREQUPRD，两次时间发生后触发一次EPWM中断。

EPWM1A和EPWM1B的引脚如下图（接线的时候请注意，可以使用杜邦线引出）：



在开发板的J15排针的40和44引脚，GND引脚参见上图。

8) 第三次实验在学期结束前任何时候都可以检查。由于使用示波器查看EPWM波形较为困难，需要引线等，因此本次实验可以只查看重要代码。

9) 大作业要求：FFT变换

实验要求

对一段离散数据进行FFT分析，并将其输出结果呈现在输出窗口中，以增加对算法的认识。

其具体要求如下：

1. 无示例程序
2. 熟悉并编写FFT算法

3. 自行生成至少 $2^n$  ( $n$ 为自然数,  $n \geq 7$ ) 点的数据, 并进行FFT分析

4. 输出结果显示在图像窗口

10) 实验报告: 实验任务一、实验任务二、实验任务三、大作业的实验原理、重要代码解释、实验收获等, 没有格式要求。

11) 大作业检查截止到最后一周周四; 实验报告提交截止到最后一周结束后下一周周五 (17周周五?)。

12) 按许老师要求, 最终的评分由平时出勤+三个实验任务评分+大作业+实验报告组成, 后两部分占比较大, 具体由许老师评定。其中, 三个实验任务和大作业的代码都是一个小队一份代码, 共同完成, 按照小组成员具体分配和完成的任务评定分数。

**2016 年 5 月 28 日**

1) 纠正一点儿笔误: 第三次实验要求中的“TZ 进制, 斩波模式进制”改成“TZ 禁止, 斩波模式禁止”。

2) 从下周 (第 14 周) 起, 每周来智能电网大楼 5 楼上课的课时只有 2 课时, 初定在周四。有变动会提前通知大家。