在 CCSv5.1 中利用 MSP430 的代码示例开发 MSP430

制作小组成员：刘阳　　电信 0905

蔡世滨　电信 0905

谢超凡　电信 0905

I、MSP-430USCI 模块使用技巧及实例---IrDA

㈠USCI 介绍

- Up to Four Universal Serial Communication Interfaces
  - USCI_A0, USCI_A1, USCI_A2, and USCI_A3 Each Supporting
    - Enhanced UART supporting Auto-Baudrate Detection
    - IrDA Encoder and Decoder
    - Synchronous SPI
  - USCI_B0, USCI_B1, USCI_B2, and USCI_B3 Each Supporting
    - I^2C^{TM}
    - Synchronous SPI

msp430f541x、msp430f543x 多达 4 个通用串行通信接口（USCI）模块，支持多种串行通信模式，不同的 USCI 模块支持不同的模式。

USCI_Ax 模块支持：

UART 模式；

IrDA 通信的脉冲整形；

LIN 通信的自动波特率检测；

SPI 模式；

USCI_Bx 模块支持：

IIC 模式；

SPI 模式；

UART 模式：

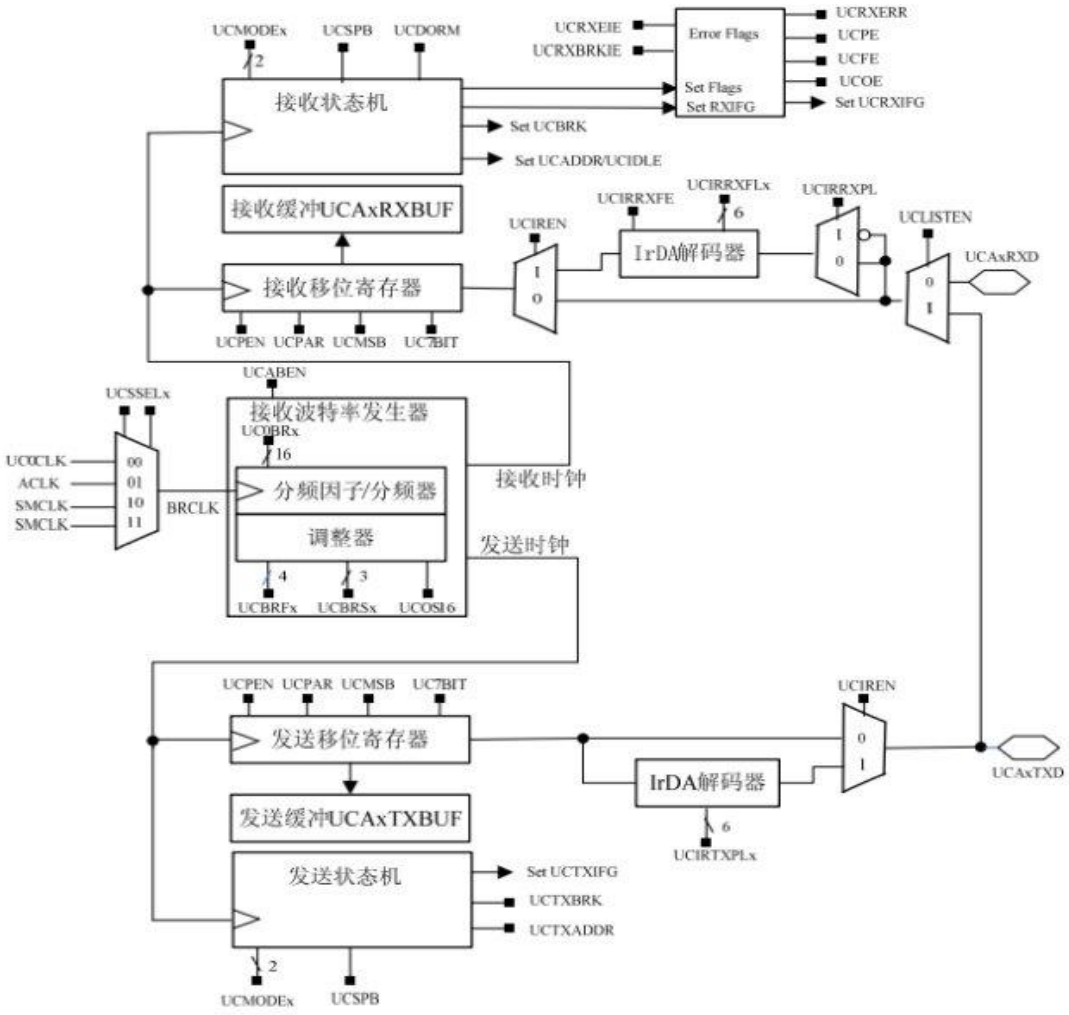


图 15-1 USCI_Ax 模块图：UART模式 (UCSYNC=0)

在异步模式下，USCI_Ax 模块通过两个外部引脚 UCAxRXD 和 UCAxTXD 将芯片连接到外部系统。当 UCSYNC
位清零时，选择 UART 模式。

USCI_Ax 控制寄存器 0(UCAxCTL0)

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

UCSYNC Bit0 使能同步模式.
0 异步模式
1 同步模式

UART 模块特征包括：
带奇校验、偶校验或非奇偶校验的 7 或 8 位数据；
独立的发送和接收移位寄存器；
独立的发送和接受缓冲寄存器；
发送和接收的独立中断能力；

最低位优先或最高位优先的数据发送和接收；
多处理器系统的内置空闲线路和地址位通信协议；
用于自动从 LPMx 模式唤醒的接收机起始边沿检测；
波特率可编程控制，支持小数波特率调制；
用于错误检测和抑制的状态标志；
用于地址检测的状态标志；
一、USCI 初始化和复位

PUC 或置位 UCSWRST，可以使 USCI 复位。PUC 后，UCSWRST 位自动置位，这使 USCI 保持在复位状态。UCSWRST 位置位，将使 UCRXIE，UCTXIE，UCRXIFG，UCRXERR，UCBRK，UCPE，UCOE，UCFE，UCSTOE 和 UCBTOE 位复位，UCTXIFG 位置位。清除 UCSWRST 将释放 USCI，使其进入操作状态。

USCI_Ax 控制寄存器 1（UCAxCTL1）

7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTxBRK	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

UCSWRST Bit0 软件复位使能

0 禁止。USCI 复位释放操作。

1 使能。USCI 逻辑保持在复位状态。

推荐使用以下过程，进行初始化或重新配置：
1.置位 UCSWRST (BIS.B #UCSWRST,&UCAxCTL1)；
2.2 设置 UCSWRST=1，初始化所有的 USCI 寄存器（包括 UCAxCTL1）；
3.配置端口；
4.软件清除 UCSWRST(BIC.B #UCSWRST,&UCAxCTL1)；
5.通过 UCRXIE 和/或 UCTXIE 使能中断（可选）；
例：串口助手发什么就返回什么.

```
#include "msp430x54x.h"
// ACLK = REFO = 32768Hz, MCLK = SMCLK = default DCO/2 = 1048576Hz
// P3.4，5——USCI_A0 TXD/RXD；P9.4,5——USCI_A2 TXD/RXD；P10.4,5——USCI_A3 TXD/RXD；
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;              // Stop WDT
  P5SEL = 0xc0;                        // P5.6,7 = USCI_A1 TXD/RXD
  UCA1CTL1 |= UCSWRST;                   // **Put state machine in reset**
  UCA1CTL1 |= UCSSEL_2;                  // SMCLK
  UCA1BR0 = 9;                         // 1MHz 115200 (see User's Guide)
  UCA1BR1 = 0;                         // 1MHz 115200
  UCA1MCTL |= UCBRS_1 + UCBRF_0;         // Modulation UCBRSx=1, UCBRFx=0
  UCA1CTL1 &= ~UCSWRST;                  // **Initialize USCI state machine**
  UCA1IE |= UCRXIE;                    // Enable USCI_A1 RX interrupt
  __bis_SR_register(LPM0_bits + GIE);      // Enter LPM0, interrupts enabled
```

```
}
// Echo back RXed character, confirm TX buffer is ready first，发送数据之前确定发送缓存准备好
#pragma vector=USCI_A1_VECTOR
__interrupt void USCI_A1_ISR(void)
{
  switch(__even_in_range(UCA1IV,4))
  {
  case 0:break;                          // Vector 0 - no interrupt
  case 2:                               // Vector 2 - RXIFG
   while (!(UCA1IFG&UCTXIFG));          // USCI_A1 TX buffer ready?
    UCA1TXBUF = UCA1RXBUF;               // TX -> RXed character
    break;
  case 4:break;                          // Vector 4 - TXIFG
  default: break;
  }
}
```

// UCTXIFG=0x02，UCA1IFG&UCTXIFG，当 UCA1IFG 的 UCTXIFG 位为 1 时，说明 UCA1TXBUF 为空，跳出 while 循环循环；当 UCTXIFG 位为 0 时 UCA1TXBUF 不为空，停在循环。

USCI_Ax 中断标志寄存器（UCAxIFG）

7	6	5	4	3	2	1	0
保留						UCTXIFG	UCRXIFG
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

保留	Bits 7-2	保留。
UCTXIFG	Bit1	发送中断标志位。当 UCAxTXBUF 为空时 UCTXIFG 置位。 0 没有中断挂起 1 中断挂起
UCRXIFG	Bit0	接收中断标志位。当 UCAxRXBUF 已经收到一个完整的字符，UCRXIFG 置位。 0 没有中断挂起 1 中断挂起

二、USCI 中断

USCI 只有一个发送和接收共用的中断向量，USCI_Ax 和 USC_Bx 不共用中断向量。

2.1 USCI 发送中断操作

发射机置位 UCTXIFG 中断标志，这表明 UCAxTXBUF 已经准备好接收另一个字符（即 UCAxTXBUF 为空）如果 UCTXIE 和 GIE 也置位的话，将产生中断请求。如果将字符写入，UCAxTXBUF、UCTXIFG 将自动复位而无需软件复位。PUC 之后或 UCSWRST = 1 时，UCTXIFG 置位、UCTXIE 复位。

2.2 USCI 接收中断操作

每接收到 1 个字符并将其载入到 UCAxRXBUF 时，UCRXIFG 中断标志置位，如果 UCTXIE 和 GIE 也置位的话，将产生中断请求。UCRXIFG 和 UCRXIE 可以通过系统复位 PUC 信号或 UCSWRST = 1 复位。当读取 UCAxRXBUF 时，UCRXIFG 自动复位。

USCI_Ax 中断使能寄存器（UCAxIE）

7	6	5	4	3	2	1	0
保留						UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

保留	Bits 7-2	保留。
UCTXIE	Bit1	发送中断使能。 0 禁止中断 1 使能中断
UCRXIE	Bit0	接收中断使能。 0 禁止中断 1 使能中断

USCI_Ax 中断向量寄存器(UCAxIV)

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
r0	r0	r0	r0	r0	r0	r0	r0

7	6	5	4	3	2	1	0
0	0	0	0	0	UCIVx		0
r0	r0	r0	r0	r0	r-0	r-0	r0

UCIVx Bits 15-0 USCI 中断向量值

UCAxIV值	中断源	中断标志	中断优先级
000h	无中断		
0002h	数据接收	UCRXIFG	最高
0004h	发送缓存空	UCTXIFG	最低

2.3 UCAxIV，中断向量发生器

USCI 中断标志具有一定的优先次序，组合使用一个中断向量。中断向量寄存器 UCAxIV 用于确定产生中断的标志。使能的具有最高优先级的中断在 UCAxIV 寄存器内产生一个数值，该数值可以加到程序计数器上，从而自动跳转到相应的软件子程序处。禁止中断不会影响 UCAxIV 的值。

对 UCAxIV 寄存器的任何访问，读或写，将会自动复位挂起的优先级最高的中断标志。如果另一个中断标志置位，在响应完第一个中断后，立即产生另一个中断。

三、寄存器

3.1

USCI_Ax 控制寄存器 0(UCAxCTL0)

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB	UCMODEx		UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

UCPEN	**Bit7**	奇偶检验使能 0 禁止奇偶校验。 1 使能奇偶校验。（UCAxTXD）产生奇偶校验位，（UCAxRXD）接收到奇偶校验位。在地址位多处理器模式下，地址位参与奇偶校验计算。
UCPAR	**Bit6**	选择奇偶校验。当禁止奇偶校验时，不使用 UCPAR。 0 奇校验 1 偶校验
UCMSB	**Bit5**	选择高位优先。控制发送和接收移位寄存器的方向。 0 低位优先 1 高位优先
UC7BIT	**Bit4**	字符长度。选择 7 位或 8 位字符长度。 0 8 位数据 1 7 位数据
UCSPB	**Bit3**	选择停止位。停止位的个数。 0 1 个停止位 1 2 个停止位
UCMODEx	**Bits2-1**	USCI 模式。当 UCSYNC = 0 时，UCMODEx 选择异步模式。 00 UART 模式 01 空闲线路多处理器模式 10 地址位多处理器模式 11 自带动波特率检测的 UART 模式

此寄存器主要是定义数据通信的字符格式，UART 的字符格式包括一个起始位，7 或 8 位数据位，一个奇/偶/非奇偶校验位，地址位（地址位模式），以及 1 或 2 个停止位，UCMSB 位控制传送方向，选择低位或高位优先，UART 通讯的典型选择是低位优先。

PUC 之后全为 0，即 字符长度 8、1 个停止位、无奇偶校验、低位优先，UART 模式。

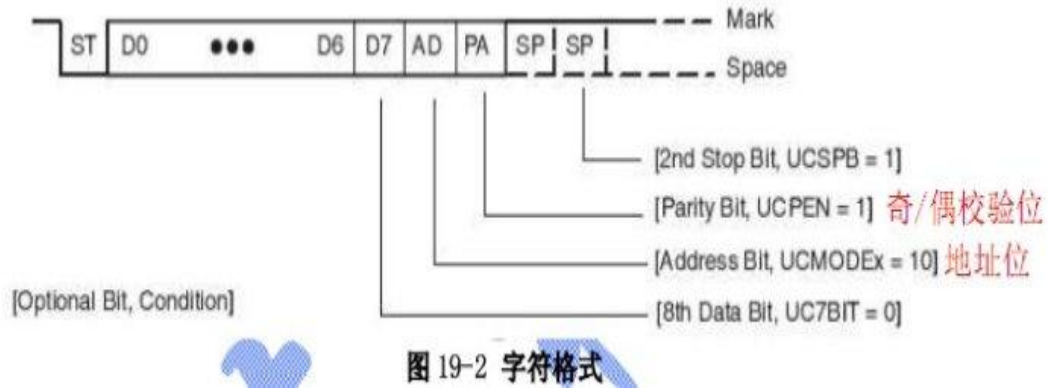


图 19-2 字符格式

UCMODEx Bits2_1 位：

两个芯片进行异步通信时，对协议来说，不需要多处理器格式。当 3 个或更多个芯片通信时，USCI 支

持线路空闲和地址位多处理器通信格式。

3.11 线路空闲多处理器模式（待续）

3.12 地址位多处理器模式（待续）

3.13 自动波特率检测（待续）

3.2

USCI_Ax 控制寄存器 1（UCAxCTL1）

7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTxBRK	UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

UCSSELx Bits7-6 USCI 时钟源选择。这些位选择 BRCLK 时钟源。
00 UCLK（外部 USCI 时钟）
01 ACLK
10 SMCLK
11 SMCLK

UCRXEIE Bit5 接收错误字符中断使能。
0 不接收错误字符且不置位 UCRXIFG
1 接收错误字符，置位 UCRXIFG

UCBRKIE Bit4 接收打断字符中断使能
0 接收的打断字符不置位 UCRXIFG
1 接收的打断字符置位 UCRXIFG

UCDORM Bit3 睡眠。使 USCI 进入睡眠模式。
0 不睡眠。所有接收的字符置位 UCRXIFG。
1 睡眠。只有空闲线路或地址位作为前导的字符置位 UCRXIFG。带自动波特率检测的 UART 模式下，只有打断和同步字段的组合可以置位 UCRXIFG。

UCTXADDR Bit2 发送地址。根据选择的多处理器模式，发送的下一帧为地址。
0 发送的下一帧是数据
1 发送的下一帧是地址

UCTxBRK Bit1 发送打断。发送带发送缓冲器写入操作的打断。在自动波特率检测的 UART 模式下，为了产生需要的中断/同步字段，必须将 055h 写入 UCAxTXBUF。否则，必须将 0h 写入发送缓冲器。
0 发送的下一帧不是打断
1 发送的下一帧是打断或打断/同步

此寄存器主要是配置 USCI，PUC 后时钟选择外部时钟，所以初始化时除了置位 UCSWRST 位外还需配置时钟源。其它的默认就行。

四、低功耗 UART 模式下使用 USCI 模块

USCI 模块提供低功耗模式下的自动时钟激活功能。当 USCI 时钟源由于设备处于低功耗模式不活动时，无论时钟源的控制位如何设置，USCI 模块会在需要时激活时钟源，时钟将保持活动状态直到 USCI 模块返回空闲状态。USCI 模块返回到闲状态后，将反转时钟源控制

位的设置。

eg.

```
void InitUARTA1(void)
{
 UCA1CTL1 |= UCSWRST;// PUC 后，UCSWRST 位自动置位，这使 USCI 保持在复位状态
 UCA1CTL0 = 0x00;
 UCA1CTL1 |= UCSSEL_2;                  // SMCLK
 UCA1BR0 = 216;                         // 24MHz 115200
 UCA1BR1 = 0;                           // 24MHz 115200
 UCA1MCTL = UCBRS_2 + UCBRF_0;          // 0x04+0x00
 P5SEL = 0xC0;                          // P5.6/7 = USCI_A0 TXD/RXD
 UCA1CTL1 &= ~UCSWRST;   // **Initialize USCI state machine**,。清除 UCSWRST 将释放
USCI，        UCA1IE |= UCRXIE;                    // Enable USCI_A1 RX interrupt
}
#pragma vector = USCI_A1_VECTOR
__interrupt void USCI_A1_ISR(void)
{
 switch (__even_in_range(UCA1IV,4))
 {
 case 0:break;
 case 2:
  g_uartBufA[g_bufALen] = UCA1RXBUF;
  if (g_uartBufA[g_bufALen]==0xFF)
  {
   }
  if (g_uartBufA[g_bufALen++]==0xFD) //判断 PC 机发送的命令帧是否已完
  {
    g_bufALen=0;
    g_uartReceive = 1;//  置位
  }
 break;
 case 4:break;                          // Vector 4 - TXIFG
 default: break;
 }
}
void USciSend( )
{
 unsigned char i;
 for (i = 0; i < g_bufALen; i++)
 {
   while (!(UCA1IFG & UCTXIFG));
   UCA1TXBUF = g_uartBufA[uartBuf1];
 }
}// UCA1MCTL 是 UCA1 的调制控制寄存器
```

5.UCAxMCTL　USCI_Ax 调整控制寄存器

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

UCBRFx　选择第一个调整阶段。这些位决定了 UCOS16=1 时 BITCLK16 的调整位图。在 UCOS16=0 时忽略这些位。见表 15-3。

UCBRSx　选择第二个调整阶段。这些位决定 BITCLK16 的调整位图。见表 15-2。

UCOS16　过采样模式允许

0　禁止

1　允许

表 15-3 BITCLK16 调整位图

UCBRFx	NO. of BITCLK16 Clocks after last falling BITCLK edge															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00h	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02h	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
03h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
04h	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
05h	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1
06h	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1
07h	0	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1
08h	0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1
09h	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1
0Ah	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1
0Bh	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
0Ch	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1
0Dh	0	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
0Eh	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0Fh	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

表 15-2 BITCLK 调整位图

UCBRSx	Bit 0 (Start Bit)	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0
3	0	1	0	1	0	1	0	0
4	0	1	0	1	0	1	0	1
5	0	1	1	1	0	1	0	1
6	0	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1

五、波特率的产生

USCI 波特率发生器可以从非标准源频率中产生标准的波特率，可以通过 UCOS16 位选择系统提供的两种操作模式。波特率可以通过使用 BRCLK 产生，根据 UCSSELx 设置，BRCLK 可以作为外部时钟 UCAxCLK 或内部时钟 ACLK 或 SMCLK 的时钟源。

5.1 低频波特率

当 UCOS16=0 时选择低频模式。该模式允许从低频时钟源产生波特率（例如从 32768Hz 晶振产生 9600 波特）。通过使用较低的输入频率，可以降低模块的功耗。在高频和高分频设置下使用这种模式，将会使多数表决在逐渐缩小的窗口中执行，因此会降低多数表决法的优势（下面的例子都是这种模式）。

在低频模式下，波特率发生器使用 1 个预分频器和 1 个调制器产生位时钟时序。这种组合

下，产生波特率时支持小数分频；在这种模式下，最大的 USCI 波特率是 UART 源时钟频率 BRCLK 的 1/3 。

每一位的时序如图所示，对于接收的每一位，为了确定该位的值，采用多数表决法。这些采样点发生在 N/2-1/2，N/2 和 N/2 + 1/2 个 BRCLK 周期处，这里 N 是每个 BITCLK 时钟中 BRCLKs 的数值。

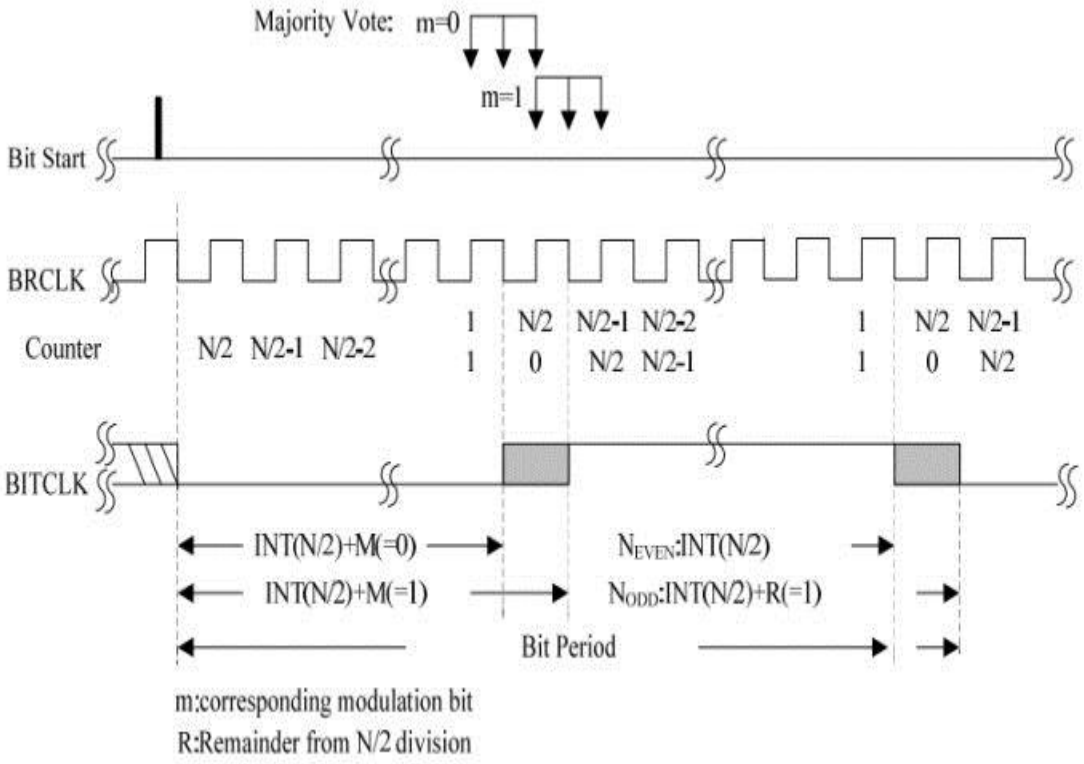


图 15-10 用UCOS16=0设定BITCLK 波特率定时

调制是建立在如表 15-2 所示的 UCBRSx 设置基础上的。表中 1 个 1 表示 m= 1，相应的 BITCLK 周期是一个 BRCLK 周期，它比 m=0 时的 BITCLK 周期长。调制在 8 位后进行，但以新的开始位重新启动。

5.2 过采样波特率的产生

当 UCOS16=1 时，选择过采样模式。该模式支持在较高输入时钟频率下对 UART 位流采样。在多数表决方法的结果总是一个位时钟周期的 1/16 位置。当使能 IrDA 编码器和解码器时，这种模式也支持带有 3/16 位时间的 IrDA 脉冲。

该模式使用一个预分频器和调制器产生 BITCLK16 时钟，该时钟比 BITCLK 快 16 倍。这种组合方式支持波特率产生时 BITCLK16 和 BITCLK 的小数分频。在这种情况下，最大的 USCI 波特率是 UART 源时钟频率 BRCLK 的 1/16。当 UCBRx 设置为 0 或 1 时，将忽略第一级分频器和调制器，BRCLK 等于 BITCLK16—在这种情况下 BITCLK16 没有调制，因此将忽略 UCBRFx 位。

BITCLK16 调制是建立在如表 15-3 所示的 UCBRFx 设置基础上的。表中 1 个 1 表示相应的 BITCLK16 周期一个 BRCLK 周期，它比 m=0 时的 BITCLK16 周期长。以每一个新位时序开始调制；BITCLK 调制是建立在如前所述的 UCBRSx 设置（见表 15-2）基础上的。

5.3 设置波特率

430 的波特率设置用三个寄存器实现：

UxBR0：波特率发生器分频系数低 8 位；

UxBR1：波特率发生器分频系数高 8 位；

UxMCTL：波特率发生器分频系数的小数部分实现；

对于给定的 BRCLK 时钟源，所使用的波特率将决定分频因子 N： N = fBRCLK/波特率。分频因子 N 通常不是一个整数值，因此至少需要一个分频器和一个调制器来尽可能接近分频因子，如果 N 值等于或大于 16，可以通过置位 UCOS16 选择过采样波特率产生模式。

在低频模式下，分频因子的整数部分通过预分频器实现 UCBRx = INT(N)；

小数部分由带有下面 nominal 公式的调制器实现： UCBRSx = round((N–INT(N))× 8)，（round 表示舍入）

UCBRSx 计数值增 1 或减 1，对任何给定的位给一个较小的最大比特误差。为了检测是不是这种情况，对于每个 UCBRSx 设置的每一位都必须经过详细的误差计算；

在过采样模式下，预分频器设置为：UCBRx = INT(N/16)，第一阶调制器设置为：UCBRFx = round(((N/16)– INT(N/16)) × 16)，当需要更高精度时，UCBRSx 调制器可以实现从 0 到 7 的值。对于给定位，为了找到最低的最大误码率设置，对于带有初始 UCBRFx 设置和增 1 或减 1 的 UCBRFx 设置的 UCBRSx 从 0 到 7 的所有设置，都必须经过详细的误差计算。

下面详解上例中的四条语句：

```
UCA1CTL1 |= UCSSEL_2;                  // SMCLK
UCA1BR0 = 216;                        // 24MHz 115200
UCA1BR1 = 0;                          // 24MHz 115200
UCA1MCTL = UCBRS_2 + UCBRF_0;         // 0x04+0x00
```

这里 SMCLK 已在时钟部分初始化，其时钟源为: Fdcoclkdiv = (760+1)*32768 = 24.936448 MHZ；

分频系数 N = 24936448/115200 = 216.462222,UCA1BR0 是分频系统整数部分的低 8 位、UCA1BR1 是高 8 位，所以..

UCA1MCTL 是波特率发生器分频系数的小数部分，由于是低频模式（UCOS16=0）, UCA1MCTL 寄存器中的 UCBRFx 位忽略，而 UCBRSx = round((N–INT(N))× 8) 即 UCBRSx = 0.46×8 四舍五入取为 4。

UCBRSx 的值也可以这么解释: 0.46*8=3.68 四舍五入为 4 个 1，把这 4 个 1 分成 8 位均匀排开 01010101 （LSB 在前），对照表 15-2 查得 UCBRSx=0x04。

㈡ IrDA 介绍

IrDA 是红外数据组织（Infrared Data Association）的简称，

目前广泛采用的 IrDA 红外连接技术就是由该组织提出的.

到目前为止，全球采用 IrDA 技术的设备超过了 5000 万部。 IrDA 已经制订出物理介质和协议层规格，以及 2 个支持 IrDA 标准的设备可以相互监测对方并交换数据。初始的 IrDA1.0 标准制订了一个串行，半双工的同步系统，传输速率为 2400bps 到 115200bps，传输范围 1 m，

传输半角度为 15 度到 30 度。最近 IrDA 扩展了其物理层规格使数据传输率提升到 4Mbps。PXA27x 就是使用了这种扩展了的物理层规格。

IrDA 协议分析

IrDA 数据协议由物理层，链路接入层和链路管理层三个基本层协议组成，另外，为满足各层上的应用的需要，IrDA 栈支持 IrLAP, IrLMP, IrIAS, IrIAP, IrLPT, IrCOMM,IrOBEX 和 IrLAN 等。

1、 IrDA 红外串行物理层协议：

IrPHY 定义了 4Mb/s 以下速率的半双工连接标准。在 IrDA 物理层中，将数据通信按发送速率分为三类：SIR、MIR 和 FIR。串行红外（SIR）的速率覆盖了 RS-232 端口通常支持的速率（9600bps～1152Kbps）。MIR 可支持 0.576Mbps 和 1.152Mbps 的速率；高速红外（FIR）通常用于 4Mbps 的速率，有时也可用于高于 SIR 的所有速率。4Mb/s 连接使用 4PPM 编码，1.152Mb/s 连接使用归零 OOK 编码，编码脉冲的占空比为 0.25。115.2kb/s 以及以下速率的连接使用占空比为 0.1875 的归零 OOK 编码。

2、 IrLAP 红外链路接入协议：

IrLAP 定义了链路初始化、设备地址发现、建立连接（其中包括比特率的统一）、数据交换、切断连接、链路关闭以及地址冲突解决等操作过程。它是从异步数据通信标准高级数据链路控制（HDLC）协议演化而来的。IrLAP 使用了 HDLC 中定义的标准祯类型，可用于点对点和点对多的应用。IrLAP 的最大特点是，由一种协商机制来确定一个设备为主设备，其他设备为从设备。主设备探测它的可是范围，寻找从设备，然后从那些相应它的设备中选择一个并试图建立连接。在建立连接的过程中，两个设备彼此协调，按照它们共同的最高通信能力确定最后的通信速率。以上所说的寻找和协调过程都是在 9.6kbps 的波特率下进行的。

3、 IrLMP 红外链路管理协议：

IrLMP 是 IrLAP 之上的一层链路管理协议，主要用于管理 IrLAP 所提供的链路连接中的链路功能和应用程序以及评估设备上的服务，并管理如数据速率、BOF 的数量（帧的开始）及连接转换向时间等参数的协调、数据的纠错传输等。

4、IrIAS,IrLPT,IrCOMM,IrOBEX,IrLAN 是建立在 IrLAP 之上的应用。

IrDA 建立连接的过程

当 IrDA 被建立时,它为自己设置下列目标：

“建立可互操作的，廉价的红外线资料互连标准能维持无连接的， 定向无线电传送的使用者模型，能适应活动的宽带的的要连接到外围设备和主机的应用。”IrDA 选择短射程的、无连接的、点对点定向的红外线通信模型有两主要的原因。

1. 第一,它初始的目标市场为支持 IrDA 的设备将是可移动的

2. 第二, IrDA 选择这个通信模型因为它最低的价格。

IrDA 建立连接通信分四个阶段

1. 设备发现和地址解析

发现过程是 IrDA 设备查明在通讯范围是否有其它设备的过程。 在此情况下，发现范围内所有设备的地址，也就是 IrLAP 操控的设备序号， 也有的是由 IrLMP 层指定的。哪个设备的发现程序占有时间槽， 那个设备就控制发现过程。当范围内有多个设备时，这种分槽的办法减少了冲突的可能性。

在等待 560ms 后（普通断开方式规则），初始设备在每个时间槽的头部开始发现过程，并广播帧标记。当听到初始发现槽时， 设备将随机选择一个响应。当设备接收到它选择槽的帧标记时，传送一个发现响应帧。 在发现过程中所有的帧都采用 HDLC 的无编号的交换标识（XID）类型。如果参加发现过程的设备有重复的地址，那就需启动地址解析过程。地址解析过程与发现过程相似，它用探测地址冲突来启动过程，仅解析有冲突的地址。 初始设备向冲突的地址传送地址解析 XID 命令，这个地址的设备选择另一个随机地址和槽响应。

初始这像以前一样传送槽标记，而原先地址冲突的设备选择恰当的槽响应。 一旦过程结束，每个设备将有唯一地址。如果仍有冲突，此过程反复进行。

2. 链接建立

一旦发现和地址解析过程完成后，应用层可以决定它希望连接到哪一个被发现的设备。应用层将发一个连接请求，它最终选择调用适当的 IrLAP 服务原语。IrLAP 层连接远程设备是采用发送带轮换查询位（poll bit）的设置正常响应模式（SNRM）的命令帧。 假设远程的设备能接受连接，它将发送一个带中止位的无编号应答响应帧， 指示连接已经被接受。在正常环境下，启动连接的设备（发送 SNRM）是主设备，其它设备是从设备。

3. 信息交换和链接复位

信息交换过程的操作实在主从模式下进行的，就是主设备控制从设备的访问。 主设备发出命令帧，从设备响应。为了保证在同一时间里只有一个设备能传送帧， 一个传送许可令牌在主、从设备间交换。一个传送许可令牌在主、从设备间交换。 主设备通过发送带轮换查询位的控制帧传递一个传送许可令牌给从设备， 从设备通过带结束位的响应帧返回令牌。传送数据时，从设备保留令牌， 一旦数据传输结束或达到最长转换时间，它必须将令牌返回主设备。 当然，主设备也受最长传送时间的限制，但没有数据传送时，主设备允许保留令牌。

4. 链接终止

一旦数据传输完，主、从设备之一将断开链接。如果主设备希望断开链接，它将发送带轮询位的断开命令给从设备。从设备返回带终止位的未编号确认帧应答。 两个设备将都处于正常断开模式，采用其参数（9600bps）。一旦两个设备处于正常中断模式，传输媒介对于任何设备都是空闲的，都可以开始设备发现，地址解析，连接建立过程。

IrLAP 协议分析

IrDA 提供的服务分为两大类，即面向连接的服务和无连接的服务。具体分为 4 种：

－Request 由上层协议送达，用来激活服务

－Indication 用于将服务初始化请求通知上层应用

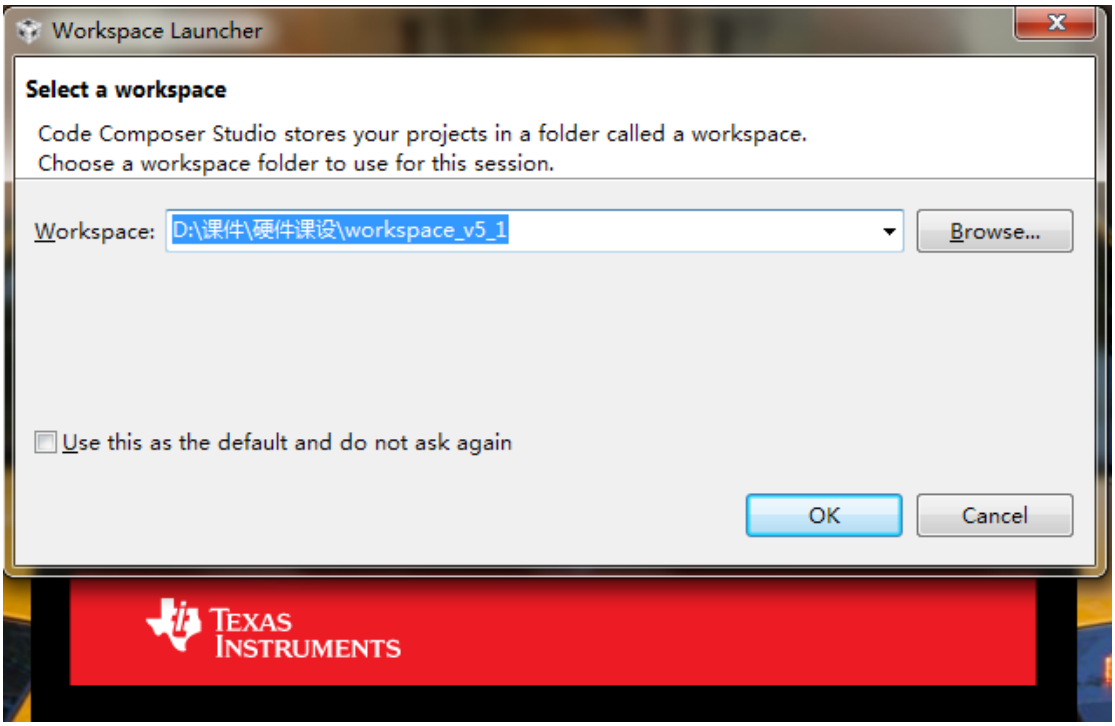
－Response 上层协议用于接受服务请求

－Confirm IrLAP 层报告服务结果

II、在 CCSv5.1 中利用 MSP430 的代码示例开发 MSP430

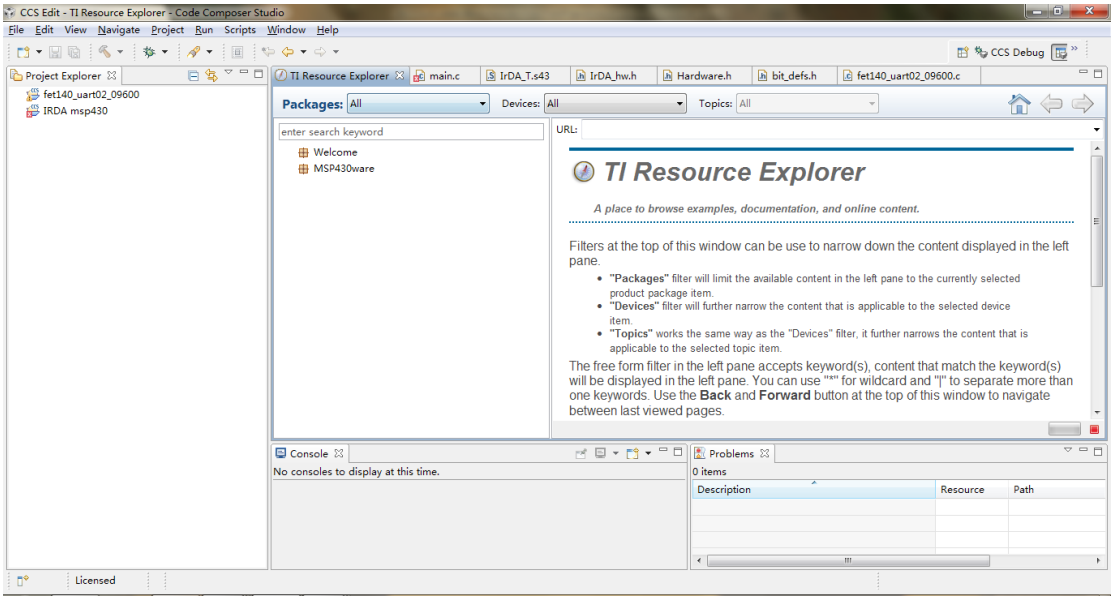
1、 安装好 ccs 5.1。得到如下图标。

2、 双击打开 ccs 5.1

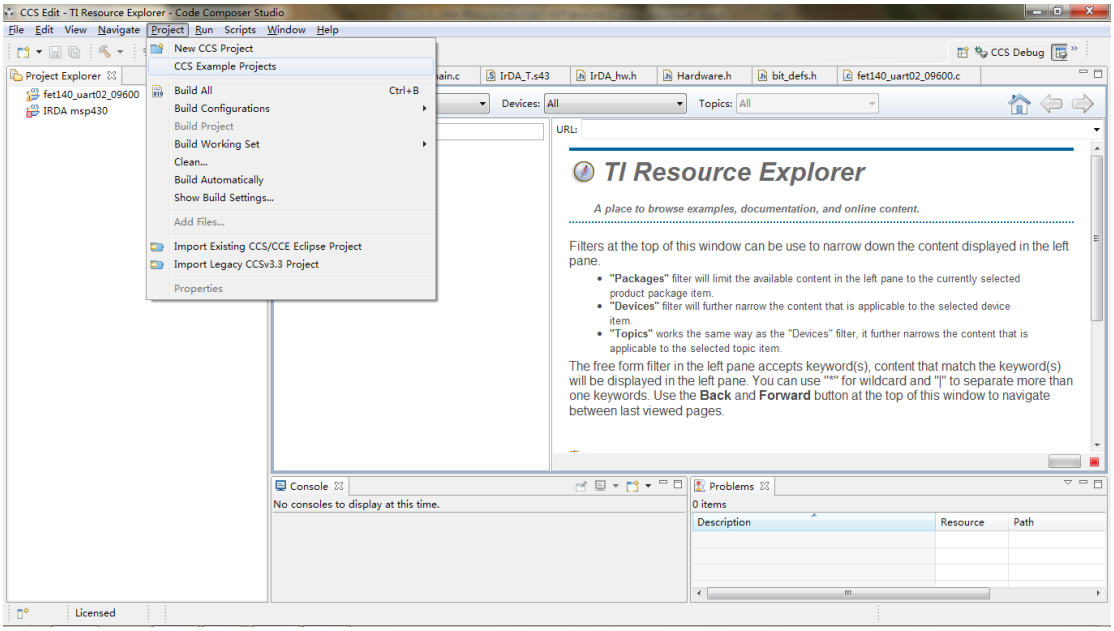


出现这个界面，选择一个你常用的开发储存地址

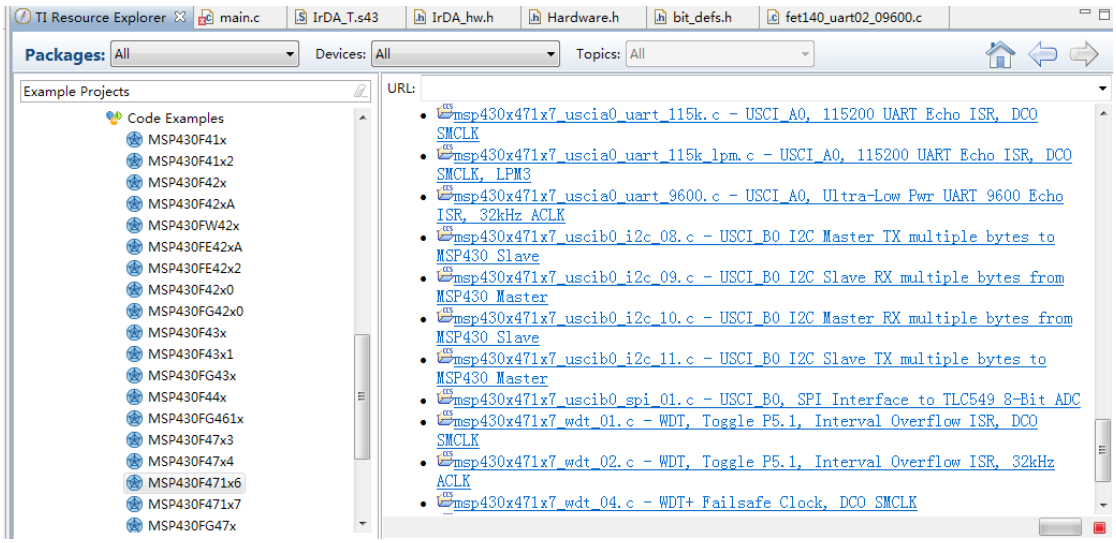
3、 出现 ccs 主界面



4、 由于我们进行样例调试，所以选择 **project** 菜单中的 **ccs example projects**



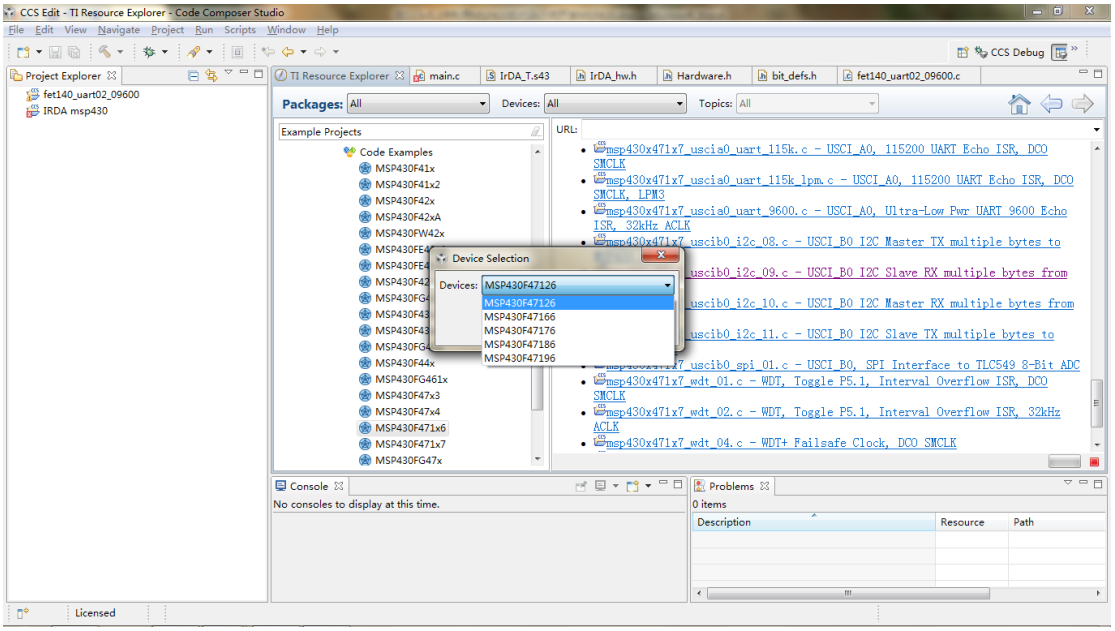
5、 在此处选择对应开发板芯片型号



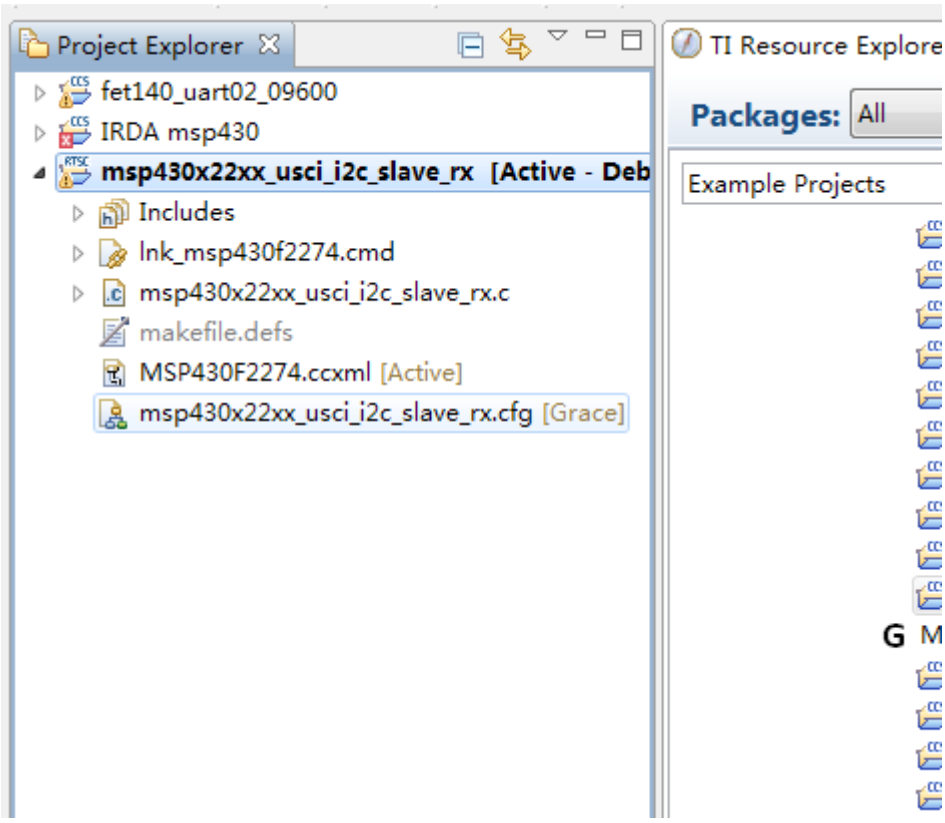
此处我们仅做示范，随便选择一款。

6、 我们选择这个

7、 会出现要你选择具体型号，我们选择默认



8、 左边出现相应的 **project**



9、 在相应工程上右键单击，在出现的菜单中选择 **build** 即可