```
s32 iNum = 0;

volatile UART_CON *pUartCon;
g_uOpClock = 0;

// 选择通道
printf("Note : [D] mark means default value. If you press ENTER key, default value is selected.\n");
printf("Select Channel(0~3) [D=0] : ");
cCh = (u8)GetIntNum();
if ( cCh>3 )     cCh = 0;                    // 默认 UART0
pUartCon = &g_AUartCon[cCh];

printf("\n\nConnect PC[COM1 or COM2] and UART%d of S3C6410 with a serial cable for test!!! \n", cCh);

//设置其他选项
printf("\nSelect Other Options\n 0. Nothing[D]    1.Send Break Signal    2. Loop Back Mode
        \n Choose : ");
switch(GetIntNum())
{
      default :
                        pUartCon->cSendBreakSignal = 0x0;
                        pUartCon->cLoopTest = 0x0;
                        break;
      case 1 :
                        pUartCon->cSendBreakSignal = 1;
                        return cCh;
      case 2 :
                        pUartCon->cLoopTest = 1;
                        break;
```

```
}

//设置奇偶模式
printf("\nSelect Parity Mode\n 1. No parity[D] 2. Odd 3. Even 4. Forced as '1' 5. Forced as '0' \n Choose : ");
switch(GetIntNum())
{
    default :
                pUartCon->cParityBit = 0;
                break;
    case 2 :
                pUartCon->cParityBit = 4;
                break;
    case 3 :
                pUartCon->cParityBit = 5;
                break;
    case 4 :
                pUartCon->cParityBit = 6;
                break;
    case 5 :
                pUartCon->cParityBit = 7;
                break;
}

//设置停止位的数量
printf("\n\nSelect Number of Stop Bit\n 1. One stop bit per frame[D] 2. Two stop bit per frame");
switch(GetIntNum())
{
    default :
                pUartCon->cStopBit = 0;
```

```
                    break;
        case 2 :
                    pUartCon->cStopBit = 1;
                    break;
}

//设置字长度
printf("\n\nSelect Word Length\n 1. 5bits 2. 6bits 3. 7bits 4. 8bits \n Choose : ");
switch(GetIntNum())
{
        case 1 :
                pUartCon->cDataBit = 0;
                break;
        case 2 :
                pUartCon->cDataBit = 1;
                break;
        case 3 :
                pUartCon->cDataBit = 2;
                break;
        default :
                pUartCon->cDataBit = 3;
                break;
}

//  设置操作时钟
printf("\n\nSelect Operating Clock\n 1. PCLK[D] 2. EXT_CLK0(pwm)     3.   EXT_CLK1(EPLL/MPLL)   \n
        Choose : ");
switch (GetIntNum())
{
```

```
case 2 :
    pUartCon->cOpClock = 1;
    // 连接 CLKOUT 和 UEXTCLK
    printf("\nInput PWM EXT_CLK by Pulse Generater\n");
    printf("How much CLK do you input through the pwmECLK?");
    printf("Mhz : ");
    g_uOpClock = GetIntNum()*1000000;
    GPIO_SetFunctionEach(eGPIO_F,eGPIO_13,2);
    break;
case 3 :
    pUartCon->cOpClock = 3;
    printf("\nSelect Clock SRC\n 1.EPLL    2.MPLL \n Choose: ");
        switch(GetIntNum())
            {
                case 1:
                    SYSC_SetPLL(eEPLL,32,1,1,0);      //EPLL=192Mhz
                    SYSC_ClkSrc(eEPLL_FOUT);
                    SYSC_ClkSrc(eUART_MOUTEPLL);
                    SYSC_CtrlCLKOUT(eCLKOUT_EPLLOUT,0);
                    g_uOpClock = CalcEPLL(32,1,1,0);
                    printf("EPLL = %dMhz\n",(g_uOpClock/1000000));
                    break;
                case 2:
                    SYSC_ClkSrc(eMPLL_FOUT);
                    SYSC_ClkSrc(eUART_DOUTMPLL);
                    Delay(100);
                    g_uOpClock = (u32)g_MPLL/2;
                    printf("MPLL = %dMhz\n",(g_uOpClock/1000000));
                    break;
```

```
                default:
                    SYSC_ClkSrc(eMPLL_FOUT);
                    SYSC_ClkSrc(eUART_DOUTMPLL);
                    Delay(100);
                    g_uOpClock = (u32)g_MPLL/2;
                    printf("MPLL = %dMhz\n",(g_uOpClock/1000000));
                    break;
            }
        break;
default :
        pUartCon->cOpClock = 0; // PCLK
        break;
}

//  选择 UART 或 IrDA 1.0
printf("\n\nSelect External Interface Type\n 1. UART[D]      2. IrDA mode\n Choose : ");
if (GetIntNum() == 2)
        pUartCon->cSelUartIrda = 1;             // IrDA 模式
else
        pUartCon->cSelUartIrda = 0;             // URAT 模式

//  设置波特率
printf("\n\nType the baudrate and then change the same baudrate of host, too.\n");
printf(" Baudrate (ex 9600, 115200[D], 921600) : ");
pUartCon->uBaudrate = GetIntNum();
if ((s32)pUartCon->uBaudrate == -1)
        pUartCon->uBaudrate = 115200;

//  选择 UART 操作模式
```

```
printf("\n\nSelect Operating Mode\n 1. Interrupt[D]    2. DMA\n Choose : ");
if (GetIntNum() == 2)
{
    pUartCon->cTxMode = 2;              // DMA0 模式
    pUartCon->cRxMode = 3;              // DMA1 模式
}
else
{
    pUartCon->cTxMode = 1;              // Int 模式
    pUartCon->cRxMode = 1;              // Int 模式
}

// 选择 UART FIFO 模式
printf("\n\nSelect FIFO Mode (Tx/Rx[byte])\n 1. no FIFO[D]  2. Empty/1  3. 16/8  4. 32/16  5. 48/32
    \n Choose : ");
iNum = GetIntNum();
if ( (iNum>1)&&(iNum<6) )
{
    pUartCon->cEnableFifo = 1;
    pUartCon->cTxTrig = iNum -2;
    pUartCon->cRxTrig = iNum -2;
}
else
{
    pUartCon->cEnableFifo = 0;
}

// 选择 AFC 模式使能/禁用
printf("\n\nSelect AFC Mode\n 1. Disable[D]    2. Enable\n Choose : ");
```

```
    if (GetIntNum() == 2)
    {
        pUartCon->cAfc = 1;                    // AFC 模式使能
        printf("Select nRTS trigger level(byte)\n 1. 63[D]    2. 56    3. 48    4. 40    5. 32    6. 24    7. 16
              8. 8\n Choose : ");
        iNum = GetIntNum();

        if ( (iNum>1)&&(iNum<9) )
            pUartCon->cRtsTrig = iNum -1;
        else
            pUartCon->cRtsTrig = 0;        //  默认 63 字节
    }
    else
    {
        pUartCon->cAfc = 0;                    // AFC 模式禁用
    }
#if 1
    printf("SendBreakSignal=%d\n",pUartCon->cSendBreakSignal);
    printf("Brate = %d\n,          SelUartIrda = %d\n,          Looptest= %d\n,          Afc = %d\n,
    EnFiFO = %d\n,          OpClk = %d\n,          Databit = %d\n,          Paritybit = %d\n,          Stopbit  =
%d\n,          Txmode = %d\n,          TxTrig = %d\n,          RxMode = %d\n,          RxTrig = %d\n,
    RtsTrig = %d\n,          SendBsig = %d\n",pUartCon->uBaudrate
        ,pUartCon->cSelUartIrda,
        pUartCon->cLoopTest,
        pUartCon->cAfc,
        pUartCon->cEnableFifo,
        pUartCon->cOpClock,
        pUartCon->cDataBit,
        pUartCon->cParityBit,
```

```
            pUartCon->cStopBit,
            pUartCon->cTxMode,
            pUartCon->cTxTrig,
            pUartCon->cRxMode,
            pUartCon->cRxTrig,
            pUartCon->cRtsTrig,
            pUartCon->cSendBreakSignal);
#endif
        return cCh;
}
```

32　PWM 定时器

该 S3C6410　RISC 微处理器由五个 32 位定时器组成。这些计时器用来产生内部中断到 ARM 子系统。此外，定时器 0，1，2 和 3 包含一个 PWM 功能（脉宽调制），它可以驱动外部的 I/O 信号。定时器 0 上的 PWM 能够产生一个可选的死区发生器。它可能被用来支持大量的通用装置。定时器 4 只是一个没有输出引脚的内部定时器。

该定时器的时钟频率通常是 APB-PCLK 分频的版本。定时器 0 和 1 共享一个可编程的 8 位预定标器，它提供第一层为 PCLK。计时器 2，3 ，4 共享不同的 8 位预定标器。每个定时器拥有它自己的时钟分频器，个别时钟分频器提供有一个二级时钟版本（预定标器由 2，4，8 或 16 进行分频）。另外，定时器从外部引脚选择时钟来源。定时器 0 和 1 可以选择外部时钟 TCLK0。定时器 2，3 和 4 可以选择外部时钟 TCLK1。每个定时器有自己的 32 位向下计数器，被定时器时钟驱动。下数计数器是最初被加载来自定时器计数缓冲寄存器（TCNTBn）。当下数计数器达到零，定时器产生中断请求通知 CPU，定时器操作完成。当定时器下数计数器达到零，相应的 TCNTBn 能被自动重新进入下数计数器的下一个周期的开始。然而，如果定时器停止，例如，在定时器运行模式下，通过清除定时器的使能位 TCONn ，TCNTBn 的值将不被加载到计数器中。

脉冲宽度调制功能（PWM）使用 TCMPBn 寄存器的值，当下数计数器的值匹配于定时器控制器逻辑中的比较寄存器的值时，定时器控制器逻辑改变输出标准。因此，比较寄存器决定一个 PWM 输出的打开时间（或关闭时间）。TCNTBn 和 TCMPBn 寄存器是双缓冲，允许定时器参数在循环中更新。直到目前的定时器周期完成，新的值才能生效。

一个 PWM 的周期的简单的例子如图 32-1 所示。

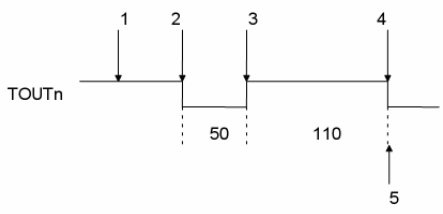

图 32-1　PWM 周期方框图的简单例子

（1）初始化带有 160（50+110）的 TCNTBn 和带有 110 位的 TCMPBn。

（2）通过设置起始位启动定时器和设置手动更新位关闭。

TCNTBn 的 160 值载入下数计数器，输出为低电平。

（3）当下数计数器的计数下降到 TCMPBn 寄存器的值（110）时，输出从低电平到高电平。

（4）当下数计数器达到零，产生中断请求。

（5）同时，通过 TCNTBn（重新启动循环）下数计数器是自动重新载入。

如图 32-2 所示，描述的是 PWM 定时器的结构框图。

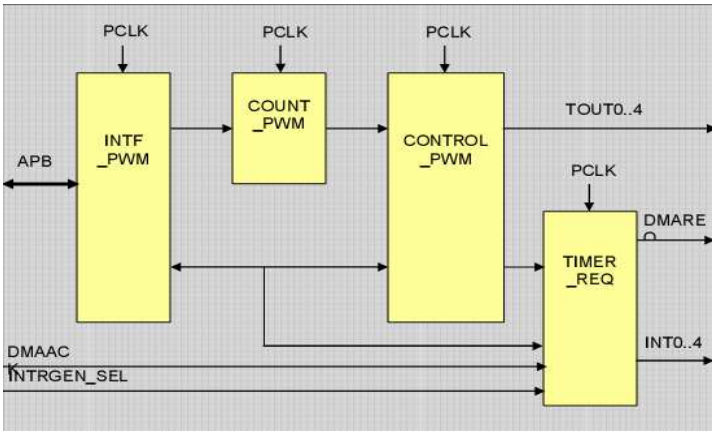

图 32-2 PWM 定时器的结构框图

如图 32-3、32-4 所示，描述个别的 PWM 通道的时钟控制的产生。