NV32F100x PIT 模块编程示例

第一章 库函数简介

库函数列表

void PIT_Init(uint8_t u8Channel_No, PIT_ConfigType *pConfig)；
PIT 初始化函数
void PIT_Enable(void)；
使能 PIT 模块
void PIT_Disable(void)；
禁用 PIT 模块
void PIT_SetDebugFreeze(void)；
设置 PIT 在调试模式下禁止运行
void PIT_SetDebugOn(void)；
设置 PIT 在调试模式下继续运行
void PIT_ChannelEnable(uint8_t u8Channel)；
PIT 模块通道使能
void PIT_ChannelDisable(uint8_t u8Channel)；
PIT 模块通道禁止
void PIT_ChannelEnableInt(uint8_t u8Channel)；
PIT 模块通道中断使能
void PIT_ChannelDisableInt(uint8_t u8Channel)；
PIT 模块通道中断禁用
void PIT_ChannelEnableChain(uint8_t u8Channel)；
PIT 链模式使能，PIT1 连接到 PIT0
void PIT_ChannelDisableChain(uint8_t u8Channel)；
PIT 链模式禁止
uint8_t PIT_ChannelGetFlags(uint8_t u8Channel)；
获取 PIT 中断标志位
void PIT_ChannelClrFlags(uint8_t u8Channel)；
清除 PIT 中断标志位
void PIT_DeInit(void)；
复位 PIT 模块
void PIT_SetLoadVal(uint8_t u8Channel, uint32_t u32loadvalue)；
设置对应 PIT 定时器的加载值
void PIT_SetCallback(uint8_t u8Channel_No, PIT_CallbackType pfnCallback)；
设置 PIT 中断回调函数
void PIT_ChnIsr(void)；
对应 PIT 中断服务子函数

1.1 模块初始化

相关配置寄存器存下：

PIT 模块控制寄存器（PIT_MCR)

字段	描述
31-3 保留	此字段为保留字段 此只读字段为保留字段且值始终为 0.
2 保留	保留
1 MDIS	模块禁用 -（PIT 部分） 禁用标准定时器。必须在执行任何其他设置之前使能该字段。 0 标准 PIT 定时器的时钟使能。 1 标准 PIT 定时器的时钟禁用。
0 FRZ	冻结 当器件进入调试模式时，允许停止定时器。 0 定时器在调试模式下继续运行。 1 定时器在调试模式下停止运行。

定时器控制寄存器（PIT_TCTRLn)

字段	描述
31-3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 CHN	链模式 激活时，定时器 n-1 需先到期，定时器 n 才能递减 1. 不能链接定时器 0. 0 定时器不链接 1 定时器链接到前一定时器。例如，对于通道 2，如该字段置位，则定时器 2 链接到定时器 1。
1 TIE	定时器中断使能 某个中断挂起或 TFLGn[TIF]置位时，使能该中断将立即引起中断事件。为避免这种情况，必须先清零相关的 TFLGn[TIF]。 0 定时器 n 的中断请求禁用 1 只要 TIF 置位，就会请求中断
0 TEN	定时器使能 使能或禁用定时器 0 定时器 n 禁用 1 定时器 n 使能

函数名	PIT_Init
函数原形	PIT_Init(uint8_t u8Channel_No, PIT_ConfigType *pConfig)
功能描述	以配置结构体 pConfig 来初始化 PIT 模块
输入参数	通道号 u8Channel_No，配置结构体 PIT_ConfigType
输出参数	无
返回值	无
先决条件	无
函数使用实例	先设置配置结构体，PIT_Init(0, &PIT0_Config);

```
/*****************************************************************************//*!
*
* @初始化 PIT 模块.
*
* @输入        u8Channel_No        通道号
* @输入        pConfig             指向配置的结构体
*
* @无返回
*
*****************************************************************************/
void PIT_Init(uint8_t u8Channel_No, PIT_ConfigType *pConfig)
{
    SIM->SCGC |= SIM_SCGC_PIT_MASK;        /* 选通 PIT 模块 */

    if (pConfig->bFreeze)
    {
        PIT_SetDebugFreeze();//定时器在调试模式下停止运行
    }

    if (pConfig->bModuleDis == 0)
    {
        PIT_Enable();          /* 标准 PIT 定时器的时钟使能 */
    }

    PIT_SetLoadVal(u8Channel_No, pConfig->u32LoadValue);   //加载对应通道的定时器起始值

    if (pConfig->bInterruptEn)
    {
        if (u8Channel_No)
        {
            NVIC_EnableIRQ(PIT_CH1_IRQn); //开启对应通道的 IRQ 中断
        }
        else
        {
            NVIC_EnableIRQ(PIT_CH0_IRQn);
```

```
        }
        PIT_ChannelEnableInt(u8Channel_No);//开启对应通道的中断请求
    }
    else
    {
        NVIC_DisableIRQ(PIT_CH0_IRQn);//禁止通道 0 的 IRQ 中断
    }

    if (pConfig->bChainMode)
    {
        PIT_ChannelEnableChain(u8Channel_No); //定时器链接到前一定时器
    }

    if (pConfig->bETMerEn)
    {
        PIT_ChannelEnable(u8Channel_No);//定时器使能
    }

}
```

1.2 装载起始值

相关配置为定时器加载值寄存器（PIT_LDVALn），如下：

字段	描述
31-0 TSV	定时器起始值 设置定时器起始值。定时器将倒计时至 0，然后生成一次中断并再次加载该寄存器值。将新值写入寄存器不会重启定时器，定时器到期后会加载新值。要中指当前周期并用新值开始一个定时器周期，必须先禁用该定时器然后再将其使能。

函数名	PIT_SetLoadVal
函数原形	PIT_SetLoadVal(uint8_t u8Channel, uint32_t u32loadvalue)
功能描述	设置对应通道的模加载值
输入参数	通道号 u8Channel_No，加载数值 u32loadvalue
输出参数	无
返回值	无
先决条件	无
函数使用实例	PIT_SetLoadVal(0, 1000)

```
/*****************************************************************************//*!
*
* @装载定时器起始值到加载值寄存器中.
*
* @输入          u8Channel       通道号
* @输入          u32loadvalue    加载数据到寄存器中
*
* @无返回
*
*****************************************************************************/
void PIT_SetLoadVal(uint8_t u8Channel, uint32_t u32loadvalue)

{
    PIT->CHANNEL[u8Channel].LDVAL = u32loadvalue;
}
```

1.3 设置回调函数

函数名	PIT_SetCallback
函数原形	Void PIT_SetCallback(uint8_t u8Channel_No, PIT_CallbackType pfnCallback)
功能描述	设置 PIT 中断回调函数入口
输入参数	通道号 u8Channel_No，中断回调函数地址 PIT_CallbackType
输出参数	无
返回值	无
先决条件	无
函数使用实例	PIT_SetCallback(1,PIT_Task)

```
/*****************************************************************************//*!
*
* @设置 PIT 模块回调函数
*
* @输入          u8Channel_No    通道号.
* @输入          pfnCallback     指向回调的地址.
*
* @无返回
*
*****************************************************************************/
void PIT_SetCallback(uint8_t u8Channel_No, PIT_CallbackType pfnCallback)
{
  PIT_Callback[u8Channel_No] = pfnCallback;
}
```

1.4 还原 PIT 模块设置

函数名	PIT_DeInit
函数原形	PIT_DeInit(void)
功能描述	复位还原 PIT 模块至初始化之前
输入参数	无
输出参数	无
返回值	无
先决条件	无
函数使用实例	PIT_DeInit()

```
/*****************************************************************************//*!
*
* @还原 PIT 模块的设置到出厂值
*
* @无输入
*
* @无返回
*
*****************************************************************************/
void PIT_DeInit(void)
{
    NVIC_DisableIRQ(PIT_CH0_IRQn);//禁止相应通道的 IRQ 中断
    NVIC_DisableIRQ(PIT_CH1_IRQn);
    PIT_SetLoadVal(0,0);//还原对应通道的数据加载值为 0
    PIT_SetLoadVal(1,0);
    PIT_ChannelDisable(0);//禁用对应通道
    PIT_ChannelDisable(1);
    PIT_ChannelDisableInt(0);//禁用对应定时器中断请求
    PIT_ChannelDisableInt(1);
    PIT_ChannelDisableChain(0);//定时器不链接
    PIT_ChannelDisableChain(1);
    PIT_ChannelClrFlags(0);//清楚定时器中断标志
    PIT_ChannelClrFlags(1);
    PIT_SetDebugOn();//定时器在调试模式下继续运行
    PIT_Disable();//禁用 PIT 定时器
    SIM->SCGC &= ~SIM_SCGC_PIT_MASK;//禁止 PIT 总线时钟
}
```

1.5 通道中断服务

函数名	PIT_Ch0Isr
函数原形	PIT_Ch0Isr(void)
功能描述	设置 PIT 中断服务函数
输入参数	无
输出参数	无
返回值	无
先决条件	无
函数使用实例	PIT_Ch0Isr(void)

```
/*****************************************************************************//*!
*
* @函数为通道 0 中断服务子函数.
*
* @无输入
*
* @无返回
*
*****************************************************************************/
void PIT_Ch0Isr(void)
{
    PIT_ChannelClrFlags(0);   //清楚中断标志位

     if (PIT_Callback[0]    )//回调函数
     {
          PIT_Callback[0]();
     }
}
```

第二章 样例程序

2.1 链接定时器配置

```
/*****************************************************************************
*
* @PIT 样例程序.
*
* 本例程设置定时器 1 链接至定时器 0 产生每秒 1 次的中断. 定时器 0 触发 1,000,000 个周期,
* 定时器 1 触发 40 次，总周期为 10,000,000 个时钟周期，总线时钟在本例程中为 40MHZ，
* 所以说，时钟周期为 25ns ,产生一秒的中断需要 40,000,000 个时钟周期
* 设定回调任务为 LED 灯闪烁.
*****************************************************************************/
#include "common.h"
#include "uart.h"
#include "pit.h"
#include "sysinit.h"

int main (void);
void PIT_Task(void);
int main (void)
{
    uint8_t             u8Ch;
    uint32_t            u32LoadValue0, u32LoadValue1;
    PIT_ConfigType   sPITConfig0, sPITConfig1;
    PIT_ConfigType   *pPIT_Config1    =&sPITConfig1;
    PIT_ConfigType   *pPIT_Config0    =&sPITConfig0;

    /* 系统初始化 */
    sysinit();

    printf("\nRunning the PIT_demo project.\n");

    LED0_Init();

    /* PIT 时钟源为总线时钟 */
    /* 通道 0 装载值为 =(1000000-1),通道 1 装载值为 =(40-1) */
    u32LoadValue0    = 0xF423F;                    /* 通道 0 装载值 */
    u32LoadValue1    = 0x27;                       /* 通道 1 装载值 */

    /* 配置通道 1 为链接模式,开启中断并且使能 */
    pPIT_Config1->u32LoadValue          = u32LoadValue1;
```

```
    pPIT_Config1->bFreeze                = FALSE;      //定时器在调试模式下继续运行
    pPIT_Config1->bModuleDis             = FALSE;     //使能定时器模块
    pPIT_Config1->bInterruptEn           = TRUE;       //开启对应通道的 IRQ 中断
    pPIT_Config1->bChainMode             = TRUE;      //定时器链接到前一定时器
    pPIT_Config1->bETMerEn               = TRUE;     //定时器使能

    /* 配置通道 0, 仅仅使能 */
    pPIT_Config0->u32LoadValue           = u32LoadValue0;
    pPIT_Config0->bFreeze                = FALSE;       //定时器在调试模式下继续运行
    pPIT_Config0->bModuleDis             = FALSE;       //使能定时器模块
    pPIT_Config0->bInterruptEn           = FALSE;
    pPIT_Config0->bChainMode             = FALSE;
    pPIT_Config0->bETMerEn               = TRUE;       //定时器使能

    PIT_Init(PIT_CHANNEL0, pPIT_Config0); //初始化 PIT 模块通道 0
    PIT_Init(PIT_CHANNEL1, pPIT_Config1); //初始化 PIT 模块通道 1

    PIT_SetCallback(PIT_CHANNEL1, PIT_Task);//设置通道 1 中断回调函数

    /* 回应从终端所发的字符 */
    while(1)
    {
        u8Ch = UART_GetChar(TERM_PORT);
        UART_PutChar(TERM_PORT, u8Ch);
    }
}
/*****************************************************************************//*!
*
* @PIT 模块任务函数
*
* @无输入
*
* @无返回
*
*****************************************************************************/
void PIT_Task(void)
{
    LED0_Toggle();
}
```

本例程提供了一个进行链接定时器配置提供 1 秒的中断间隔，为用户提供了一个基本的 PIT 编程框架。
该样例工程在 nv32_pdk\build\keil\NV32\PIT_demo 下