ARM 实验（Cortex-M4）

实验一　时钟选择与 GPIO 实验

一. 实验目的

熟悉 ARM 的集成开发环境 KEIL uVision5，能够自行建立一个实验用工程项目。
理解 CPU 的时钟信号，了解不同时钟对电源消耗的不同。
掌握 GPIO 的工作原理，能够结合 GPIO 的输入与输出功能进行实验。

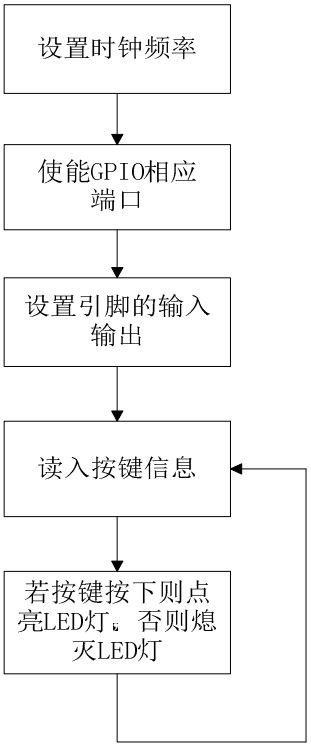
二. 实验要求

1. 分别使用内部 16M 的 HSI 时钟 ，外部 25M 的 HSO 时钟，以及 PLL 时钟进行 GPIO-PF0 的闪烁。参阅实验 1.1 的程序。并观察电流表显示的电流变化。
   用示波器观察 PF0 的频率在不同的时钟频率下是否有变化。
   进入 DEBUG 模式，设置断点，观察设置时钟后时钟返回值的大小是否符合设定。
   示例程序为正常时，慢闪 PF0，当按下 USR_SW1 时，快闪 PF0。

2. 当按下 USR_SW1 键时，点亮 LED_M0，放开时，熄灭 LED_M0
   当按下 USR_SW2 键时，点亮 LED_M1，放开时，熄灭 LED_M1
3. 当第一次短按 USR_SW1 键时，闪烁 LED_M0，
   当第二次短按 USR_SW1 键时，熄灭 LED_M0
   当第三次短按 USR_SW1 键时，闪烁 LED_M1
   当第四次短按 USR_SW1 键时，熄灭 LED_M1
   如此循环往复

三. 实验框图

实验要求 2：



四. 实验结果

应能现场演示实验的效果。

五. 讨论

1. 人为修改内部时钟或外部时钟，如将内部时钟改为 8M，或将外部时钟改为 30M，会有什么结果？

2. 能否将 PLL 时钟调整到外部时钟的频率以下？如将 25M 外部时钟用 PLL 后调整为 20M？

3. 将 PLL 后的时钟调整为最大值 120M，LED 闪烁会有什么变化？为什么？

4. GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0)此函数中，每个函数项的意义。第三个函数项为 GPIO_PIN_0，如果改为 1 或改为 2，或其他值，分别有什么现象？

5. 结合硬件说明 GPIOPinConfigure 行的作用。如果此行注释，在 WATCH 窗口中观察 key_value 值会有什么变化。

实验二 I2C GPIO 扩展及 SYSTICK 中断实验

一. 实验目的

了解 I2C 总线标准及在 TM4C1294 芯片的调用方法
掌握用 I2C 总线扩展 GPIO 芯片 PCA9557 及 TCA6424 的方法
能够通过扩展 GPIO 来输出点亮 LED 及动态数码管
熟悉 SYSTICK 中断调用方式，掌握利用软定时器模拟多任务切换的方法

二. 实验要求

1. 例程为两个芯片的初始化，PCA9557 点亮所有 LED；TCA6424 控制数码管显示 0 在第一位；阅读并理解。

2. 进行 LED 的跑马灯实验，当 LED 在某位点亮时，同时在数码管的某位显示对应的 LED 管号。如 LED 跑马灯时，从左到右依次点亮 LED1~LED8，此时在数码管上依次显示 1~8。

3. 接程序 2，当按键 USR_SW1 按下时，停止跑马灯，但 LED 及数码管显示维持不变；当按键松开后，继续跑马灯。

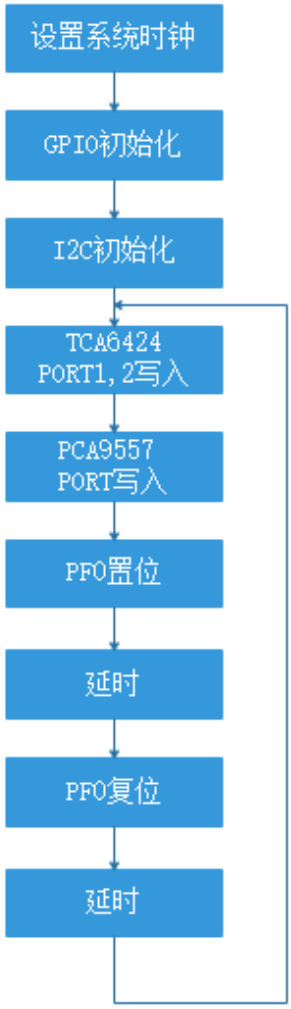
4. 根据中断例程，在中断程序中实现实验 2，3。

5. 选做。PF0 闪烁作为任务 1，LED 跑马灯及数码管循环显示作为任务 2，在 SYSTICK 的时间调度下，两个任务优先级相同，按时间片在运行。现在增加任务 3，按下 USR_SW1 按键时，PN0 常亮，松开按键熄灭。任务 3 的优先级高于任务 1，2，即任务 3 在执行时（按下 USR_SW1），任务 1，2 均不执行。

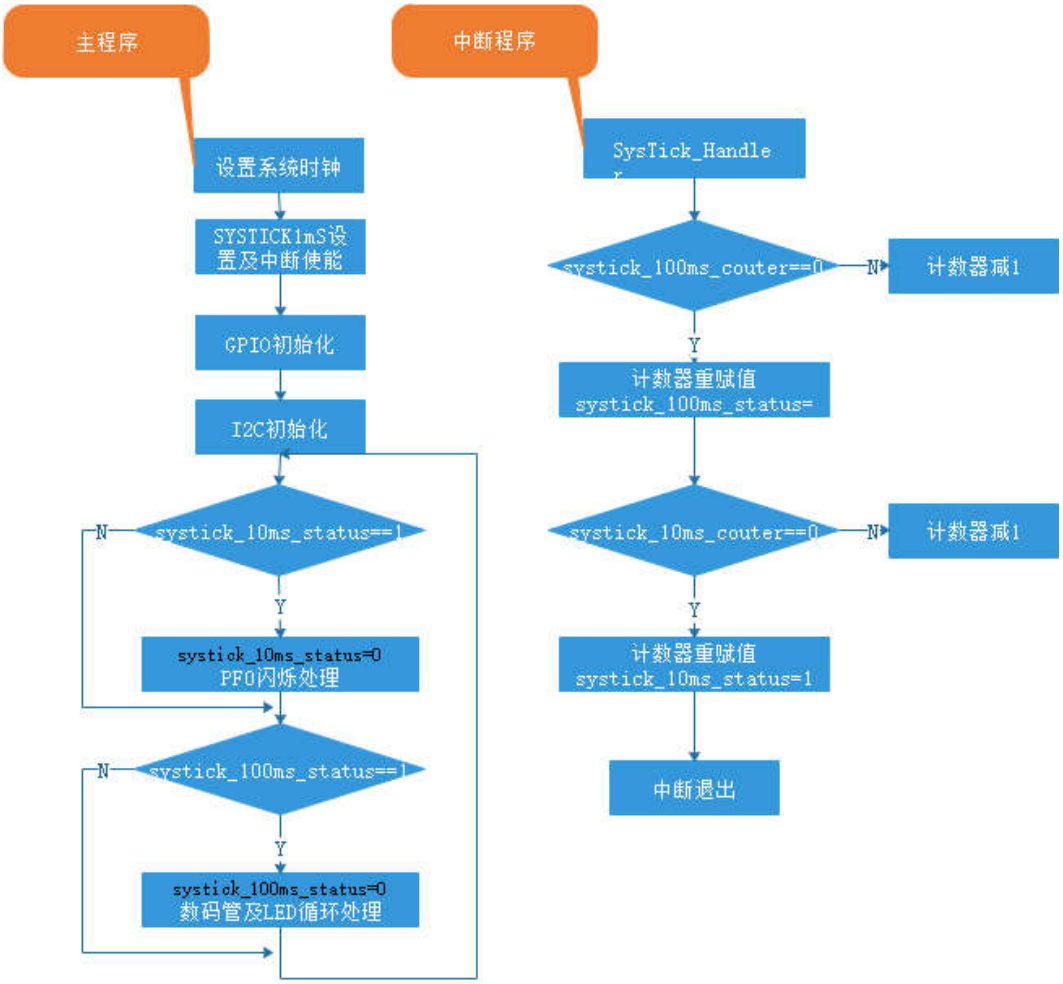
注：当有 I2C 器件时，系统时钟不宜超过 20M。

三.实验框图

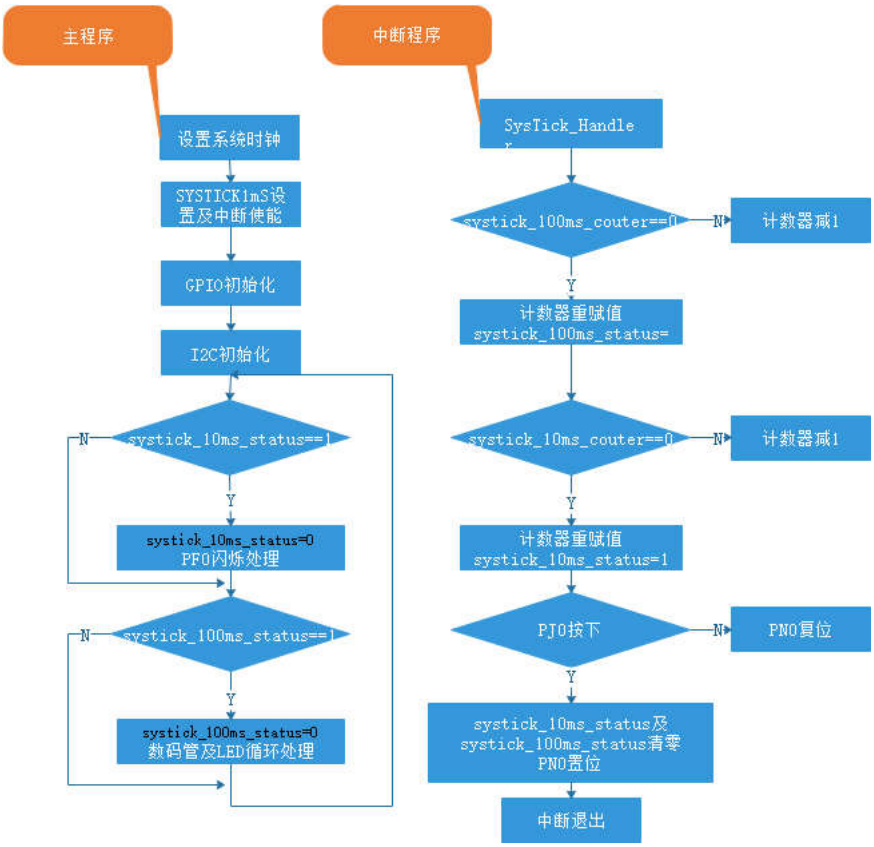
实验 1：


设置系统时钟
GPIO初始化
I2C初始化
TCA6424
PORT1,2写入
PCA9557
PORT写入
PFO置位
延时
PFO复位
延时

实验 4：


主程序
设置系统时钟
SYSTICK1mS设置及中断使能
GPIO初始化
I2C初始化
systick_10ms_status==1
N
Y
systick_10ms_status=0
PF0闪烁处理
systick_100ms_status==1
N
Y
systick_100ms_status=0
数码管及LED循环处理
中断程序
SysTick_Handle
r
systick_100ms_couter==0
N
计数器减1
Y
计数器重赋值
systick_100ms_status=
systick_10ms_couter==0
N
计数器减1
Y
计数器重赋值
systick_10ms_status=1
中断退出

实验 5



四.实验结果

五. 讨论

1. 在使用 PIOSC 及 MOSC 时，能否生成非晶振频率的系统频率？如内部晶振为 16M，或外部晶振为 25M，能否生成系统频率为 1M 或 10M.

2. 在使用 PLL 时，系统频率最小值及最大值分别为多少？

3. 如果跑马灯要求为 2 位跑马，例：当显示为 1 时，跑马灯点亮 LED8，LED1，当显示为 2 时，跑马灯点亮 LED1，LED2，如此循环，如何实现？

4. 在 3 基础上，数码管显示也改为 2 位显示。例

第一步　显示 1，2
　　跑马灯显示 LED1,2
第二步　显示 2，3
　　跑马灯显示 LED2，3
。。。
第 8 步　显示 8，1
　　跑马灯显示 LED8，1
第 9 步　回到第一步

5. 用 USR_SW1 控制跑马灯的频率，

按第 1 下，间隔为 1S
按第 2 下，间隔为 2S

按第 3 下，间隔为 0.2S

按第 4 下，回到上电初始状态，间隔 0.5S

以 4 为模，循环往复

6. 请编程在数码管上实现时钟功能，在数码管上最左端显示分钟+秒数，其中分钟及秒数均为 2 位数字。如 12:00，共 5 位。

每隔一秒，自动加 1，当秒数到 60 时，自动分钟加 1，秒数回到 00，分钟及秒数显示范围 00~59。

当按下 USR_SW1 时，秒数自动加 1

当按下 USR_SW2 时，分钟自动加 1

当按下以上一个或两个按键不松开时，对应的显示跳变数每隔 200mS 自动加 1。即如下按下 USR_SW1 1S，则显示跳变秒数加 5

实验三 UART 串行通讯口及中断优先级实验

二. 实验目的

了解 UART 串行通讯的工作原理
掌握在 PC 端通过串口调试工具与实验板通过 UART 通讯的方法
掌握 UART 的堵塞式与非堵塞式通讯方法

二. 实验要求

1. 例程为 UART0 的初始化，实验板在初始化完成后向主机发送“HELLO,WORLD!”字符串。阅读 3-1.c 并理解。

2. 在实验 1 的基础上，通过 PC 端发送字符串，实验板收到后并原样返回。称为 UART ECHO。阅读 3-2.c 并理解。

3. 将实验 2 改写为非堵塞式方式，即中断方式进行发送与接收。当进行数据接收时，点亮 PN1。阅读 3-3.c 并理解。

4. 请编程实现一个虚拟 AT 指令集：

当 PC 端发来 AT+CLASS 后，实验板回以 CLASS######，其中#####为你的班级号

当 PC 端发来 AT+STUDENTCODE 后，实验板回以 CODE######，其中#####为你的学号

5. 选做。将 4 改为大小写均能适应。

6. 选做。请编程实现以下三个命令：

底板运行后自动实现 1S 计时。

a) PC 端发来绝对对时命令，如 SET12:56:03 或 12-56-03，自动将当前时间同步到 12:56:03，并回之以当前时间

b) PC 端发来相对对时命令，如 INC00:00:12，自动将当前时间加 12 秒，并回之以当前时间

c) PC 端发来查询命令，GETTIME，自动回之以当前时间

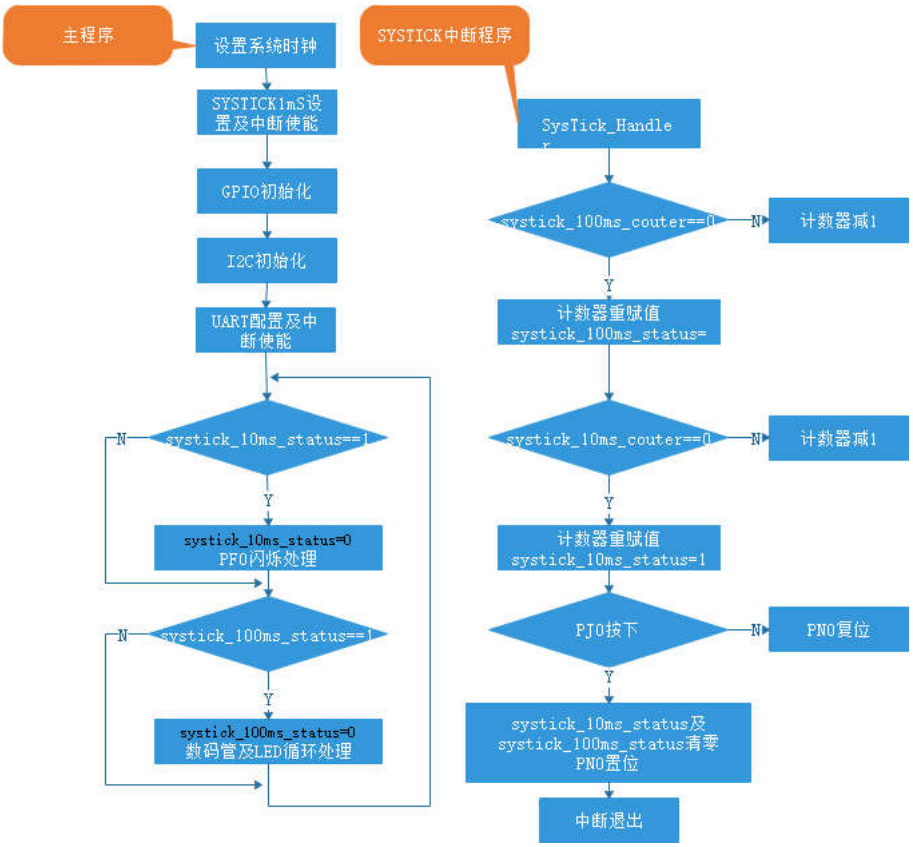
d) 当前时间格式统一为 TIME12:56:03，其中 TIME 为字符，后续为时间值

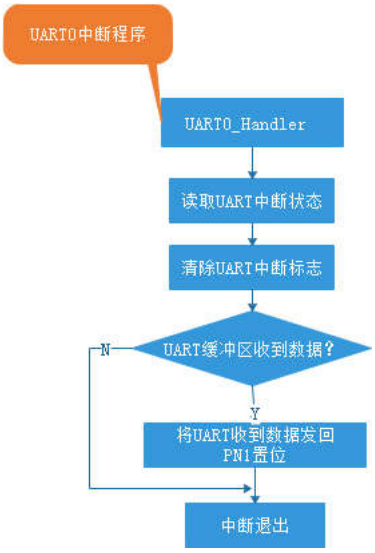
7. 优先级调整实验。基于实验 3，调整 UART0 的优先级，使之高于 SYSTICK 的优先级，并处于抢占式优先，这样当按下 USR_SW2 时，UART0 不退出，导致 SYSTICK 中断不能进入，整个系统停滞。显示不再跳变。请阅读 3-4.c 并理解。

改写程序，将 SYSTICK 的抢占式优先级高于 UART0，从而达到即使 USR_SW2 按下时，SYSTICK 中断仍然能进入。

三.实验框图

实验 3





四.实验结果

五. 讨论

1. 实验 3-2，if (UARTCharsAvail(UART0_BASE))此行程序的作用。如果没有此行，会导致什么问题？
2. 实验 3-3，void UART0_Handler(void)为什么没有在主函数声明？
3. 为什么 3-3 的中断中需要读取中断标志并清除，而 SYSTICK 不需要

4. 请根据上位机的命令，如“MAY+01”，格式为：
其中 MAY 为月份，（JAN,FEB,…DEC）均为三位。
+表示加运算符,-表示减运算符，均为 1 位。
01 表示增加或减少量，均为 2 位。范围 00-11
以上均为 ASCII 码，
MAY+01 应该回之以 JUNE
MAY-06 应该回之以 NOV

5. 请根据上位机的命令，如“14:12+05:06”，格式为：
其中 14:12 为分钟与秒，共 5 位，包括一个“:”。
+表示加运算符,-表示减运算符，均为 1 位。
05:06 为分钟与秒的变化量，共 5 位。包括一个“：”，范围 00:00~23:59
以上均为 ASCII 码，
14:12+05:06 回之以 19:18