**《嵌入式系统及应用》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | | **2019计算机科学与技术卓越02** | | | **姓名** | **李燕琴 李放** |
| **实验题目** | **Lab7 SOS** | | | | | |
| **实验时间** | **2022-03-26** | | **实验地点** | **DS3 305** | | |
| **实验成绩** |  | | **实验性质** | **□验证性 √设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 一、实验目的  In Lab7 you will learn how to write software that reads from two switches, makes decisions, and outputs to an LED. You will learn and understand the steps required to initialize parallel ports.  Lab 8 is our first lab requiring you to build circuits on the breadboard and connect them to the LaunchPad. The purpose of this lab is to learn how to interface a switch and an LED. You will perform explicit measurements on the circuits in order to verify they are operational and to improve your understanding of how they work.  In Lab9 you will learn about time delays, arrays and functional debugging. The data you collect will be physical proof that the system operates within specifications. | | | | | | |
| 二、实验项目内容  The Lab7 starter project is the same as **C7\_SOS** example but includes the connections to the Lab7 grader. You will make three changes. **First**, you will modify the system so a yellow SOS is flashed instead of the green. Yellow is created by mixing red and green (PF3 high, PF2 low, and PF1 high). **Second**, make the SOS flash only if both switches SW1 and SW2 are pressed (this means both PF4 and PF0 inputs are 0). **Third**, you will decrease the time between SOS outputs from 5 to 4 seconds.  在本实验中，将构建一个实现正逻辑的开关接口以及构建一个实现正逻辑的 LED 接口。 您将此开关和 LED 连接到您的原型板（带有所有孔的白色部分），并将它们连接到您的 TM4C123。  该系统的整体功能与实验 6 类似，有五处更改：  1) 我们连接开关的引脚移至 PE0。  2) 您将不得不移除 PUR 初始化，因为不再需要上拉。  3）我们连接 LED 的引脚移动到 PE1。  4）开关从负逻辑变为正逻辑。  5）你应该减少延迟，使其闪烁约 5 Hz。  Lab9 入门项目与 C9\_Debugging 示例相同，但包括与 Lab9 分级机的连接。 您将进行三项更改： 1）、使 LED 以 10 Hz 的频率闪烁。 也就是说，让它开机0.05秒，然后关机0.05秒。  2）、如果按下开关 SW1 或 SW2，使 LED 闪烁（这意味着 PF4 或 PF0 为 0）。  3）每次输入改变或输出改变时记录PortF位4,1,0。 例如，如果您的系统检测到 PF4 或 PF0 输入发生变化，则记录 PortF 位 4、1、0。 如果您的系统导致 PF1 发生变化，请记录 PortF 位 4,1,0。  为了让评分引擎查看/评分您的数据，请保持调试数组的定义与初始项目中的完全相同。 | | | | | | |
| 1. 实验过程或算法（源程序） 2. 功能描述：   1) Make PF1, PF2, and PF3 outputs.  Make PF0 and PF4 inputs (enable PUR for PF0 and PF4).  2) If either SW1 or SW2 are off, the LEDs should be off.  If both SW1 and SW2 are on, the SOS is sent on the yellow LED  a) Send an ‘S’ as short short short pulses on the yellow LED  b) Send an ‘O’ as long long long pulses on the yellow LED  c) Send an ‘S’ as short short short pulses on the yellow LED  d) Wait 4 seconds  3) Repeat step 2 over and over.  1） 输出端口 PF1、PF2 和 PF3。 输入端口： PF0 和 PF4（为 PF0 和 PF4 启用 PUR）。  2） 如果 SW1 或 SW2 打开，则指示灯应熄灭。 如果 SW1 和 SW2 均闭合，则 SOS 在黄色 LED 上发送  a） 发送一个"S"作为黄色 LED 上的短脉冲 （3次）  b） 发送一个"O"作为黄色 LED 上的长脉冲 （3次）  c） 发送一个"S"作为黄色 LED 上的短脉冲 （3次）  d） 等待 4 秒钟  3） 一遍又一遍地重复步骤 2。     1. 环境设置   要运行 Lab 7 评分器，必须执行两项操作。首先，执行project→options，然后选择Debug选项卡。调试参数字段必须包含 –dedXLab7。其次，必须将 edXLab7.dll 文件添加到本地Keil\ARM\BIN 文件夹中。    3、注意事项  1）适当修改程序，确保LED黄灯闪烁。  2）适当修改程序，确保当两个开关都按下时，才会执行SOS闪烁过程。   1. 更改 SOS 输出软件中的逻辑，使其在输出之间延迟 4 秒，而不是 5 秒。   4、上板时注意  1） 在 Keil 中，执行**Project->OptionsForTarget**.。在**Target**选项卡中，选择**Operating System**下拉菜单中的"TExaS"。在**Debug**选项卡中，单击右侧的Use radio，然后选择 Stellaris ICDI。  2） 通过执行 **Project->BuildProject**  3） 通过执行 **Flash->Download**下载将目标代码下载到 Flash EEPROM  4） 通过执行 **Debug->Start/StopDebuggingSession**调试会话  5） 通过执行 **Debug->OSsupport->TExaSGrader2.0** 展示得分板  6） 通过**Debug->Run**运行执行的程序  7） 在尝试评分之前调试软件， 请先调试您的软件。当您准备好运行刻度器时，请按重置，然后运行。通过单击**Grade**按钮调用评分器（按照ActionMsg窗口中的说明进行操作）  5、源程序   1. // 0.Documentation Section 2. // Lab7\_SOS, main.c 4. // Runs on LM4F120 or TM4C123 LaunchPad 5. // Input from PF4(SW1) and PF0(SW2), output to PF3 (Green LED) 6. // Pressing SW1 starts SOS (Green LED flashes SOS). 7. //    S: Toggle light 3 times with 1/2 sec gap between ON....1/2sec....OFF 8. //    O: Toggle light 3 times with 2 sec gap between ON....2sec....OFF 9. //    S: Toggle light 3 times with 1/2 sec gap between ON....1/2sec....OFF 10. //    4 second delay between SOS 11. // Pressing SW2 stops SOS 13. // Authors: Daniel Valvano, Jonathan Valvano and Ramesh Yerraballi 14. // Date: July 15, 2013 16. // 1. Pre-processor Directives Section 17. #include "TExaS.h" 19. // Constant declarations to access port registers using 20. // symbolic names instead of addresses 21. #define GPIO\_PORTF\_DATA\_R       (\*((volatile unsigned long \*)0x400253FC)) 22. #define GPIO\_PORTF\_DIR\_R        (\*((volatile unsigned long \*)0x40025400)) 23. #define GPIO\_PORTF\_AFSEL\_R      (\*((volatile unsigned long \*)0x40025420)) 24. #define GPIO\_PORTF\_PUR\_R        (\*((volatile unsigned long \*)0x40025510)) 25. #define GPIO\_PORTF\_DEN\_R        (\*((volatile unsigned long \*)0x4002551C)) 26. #define GPIO\_PORTF\_LOCK\_R       (\*((volatile unsigned long \*)0x40025520)) 27. #define GPIO\_PORTF\_CR\_R         (\*((volatile unsigned long \*)0x40025524)) 28. #define GPIO\_PORTF\_AMSEL\_R      (\*((volatile unsigned long \*)0x40025528)) 29. #define GPIO\_PORTF\_PCTL\_R       (\*((volatile unsigned long \*)0x4002552C)) 30. #define SYSCTL\_RCGC2\_R          (\*((volatile unsigned long \*)0x400FE108)) 31. // 2. Declarations Section 32. //   Global Variables 33. // unsigned long SW1; // input from PF4 34. // unsigned long SW2; // input from PF0 35. unsigned **long** myIn; 37. //   Function Prototypes 38. **void** PortF\_Init(**void**); 39. **void** FlashSOS(**void**); 40. **void** delay(unsigned **long** halfsecs); 41. **void** EnableInterrupts(**void**);  // Enable interrupts 43. // 3. Subroutines Section 44. // MAIN: Mandatory for a C Program to be executable 45. **int** main(**void**){ 46. TExaS\_Init(SW\_PIN\_PF40, LED\_PIN\_PF321);  // activate grader and set system clock to 80 MHz 47. PortF\_Init(); // Init port PF4 PF2 PF0 48. EnableInterrupts();           // enable interrupts for the grader 49. **while**(1){ 50. /\* 51. do{ 52. SW1 = GPIO\_PORTF\_DATA\_R&0x10; // PF4 into SW1 53. }while(SW1 == 0x10); 54. do{ // pf4 press one 55. SW2 = GPIO\_PORTF\_DATA\_R&0x01; // PF0 into SW2 56. }while(SW2 == 0x01); 57. FlashSOS(); 58. \*/ 59. myIn = GPIO\_PORTF\_DATA\_R&0x11; // read PF4 and PF0 into In 60. **if**(myIn == 0x00){              // SW1 & SW0 all pressed on 61. FlashSOS(); 62. } 63. } 64. } 65. // Subroutine to initialize port F pins for input and output 66. // PF4 is input SW1 and PF2 is output Blue LED 67. // Inputs: None 68. // Outputs: None 69. // Notes: ... 70. **void** PortF\_Init(**void**){ **volatile** unsigned **long** delay; 71. SYSCTL\_RCGC2\_R |= 0x00000020;     // 1) F clock 72. delay = SYSCTL\_RCGC2\_R;           // delay 73. GPIO\_PORTF\_LOCK\_R = 0x4C4F434B;   // 2) unlock PortF PF0 74. GPIO\_PORTF\_CR\_R |= 0x1F;           // allow changes to PF4-0 75. GPIO\_PORTF\_AMSEL\_R &= 0x00;        // 3) disable analog function 76. GPIO\_PORTF\_PCTL\_R &= 0x00000000;   // 4) GPIO clear bit PCTL 77. GPIO\_PORTF\_DIR\_R = 0x0E;          // 5.1) PF4,PF0 input 78. // GPIO\_PORTF\_DIR\_R |= 0x11;          // 5.2) PF1,PF2,PF3 output that is 01110 79. GPIO\_PORTF\_AFSEL\_R &= 0x00;        // 6) no alternate function 80. GPIO\_PORTF\_PUR\_R |= 0x11;          // enable pullup resistors on PF4,PF0 81. GPIO\_PORTF\_DEN\_R |= 0x1F;          // 7) enable digital pins PF4-PF0 82. } 83. // Color    LED(s) PortF 84. // dark     ---    0 85. // red      R--    0x02 86. // blue     --B    0x04 87. // green    -G-    0x08 88. // yellow   RG-    0x0A 89. // sky blue -GB    0x0C 90. // white    RGB    0x0E 92. // Subroutine to Flash a green LED SOS once 93. // PF3 is green LED: SOS 94. //    S: Toggle light 3 times with 1/2 sec gap between ON....1/2sec....OFF 95. //    O: Toggle light 3 times with 2 sec gap between ON....2sec....OFF 96. //    S: Toggle light 3 times with 1/2 sec gap between ON....1/2sec....OFF 97. // Inputs: None 98. // Outputs: None 99. // Notes: ... 100. **void** FlashSOS(**void**){ 101. //S 102. GPIO\_PORTF\_DATA\_R |= 0x0A;  delay(1); // 1\*0.5s 103. GPIO\_PORTF\_DATA\_R &= ~0x0A; delay(1); 104. GPIO\_PORTF\_DATA\_R |= 0x0A;  delay(1); 105. GPIO\_PORTF\_DATA\_R &= ~0x0A; delay(1); 106. GPIO\_PORTF\_DATA\_R |= 0x0A;  delay(1); 107. GPIO\_PORTF\_DATA\_R &= ~0x0A; delay(1); 108. //O 109. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(4); // 4\*0.5s 110. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(4); 111. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(4); 112. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(4); 113. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(4); 114. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(4); 115. //S 116. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(1); 117. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(1); 118. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(1); 119. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(1); 120. GPIO\_PORTF\_DATA\_R |= 0x0A; delay(1); 121. GPIO\_PORTF\_DATA\_R &= ~0x0A;delay(1); 122. delay(8); // Delay for 5 secs in between flashes  but the book is sayed 4ms 124. } 126. // Subroutine to delay in units of half seconds 127. // Inputs: Number of half seconds to delay 128. // Outputs: None 129. // simple delay function 130. // which delays time\*500 milliseconds 131. // assuming 80 MHz clock 132. **void** delay(unsigned **long** time){ 133. unsigned **long** i; 134. **while**(time > 0){ 135. i = 6666665; 136. **while**(i > 0){ 137. i = i - 1; 138. } 139. time = time - 1; 140. } 141. } 142. 功能描述：   1) 将 PE1 设为输出，将 PE0 设为输入。  2) 系统启动，LED 亮起（使 PE1 =1）。  3) 等待约 100 ms  4) 如果按下开关（PE0 为 1），则切换 LED 一次，否则打开 LED。  5) 一遍又一遍地重复步骤 3 和 4。   1. 电路图绘制      1. 环境设置   要运行 Lab 8 评分器，必须执行两项操作。首先，执行project→options，然后选择Debug选项卡。调试参数字段必须包含 –dedXLab8。其次，必须将 edXLab8.dll 文件添加到本地Keil\ARM\BIN 文件夹中。    3、注意事项  1）初始化PORTE端口。  2）保证delay时间为100ms。   1. 适当修改程序，使按下按键为灯泡闪烁，不按下按键灯泡常亮。   4、上板时注意  1） 在 Keil 中，执行**Project->OptionsForTarget**.。在**Target**选项卡中，选择**Operating System**下拉菜单中的"TExaS"。在**Debug**选项卡中，单击右侧的Use radio，然后选择 Stellaris ICDI。  2） 通过执行 **Project->BuildProject**  3） 通过执行 **Flash->Download**下载将目标代码下载到 Flash EEPROM  4） 通过执行 **Debug->Start/StopDebuggingSession**调试会话  5） 通过执行 **Debug->OSsupport->TExaSGrader2.0** 展示得分板  6） 通过**Debug->Run**运行执行的程序  7） 在尝试评分之前调试软件， 请先调试您的软件。当您准备好运行刻度器时，请按重置，然后运行。通过单击**Grade**按钮调用评分器（按照ActionMsg窗口中的说明进行操作）   1. 在面包板上搭建电路   为了构建电路，我们将使用无焊面包板，也称为原型板。原型板上的孔以系统的方式在内部连接。沿着原型板外侧的长排孔是电连接的。原型板有四排长（每边两排），而另一些只有两排长排（每边各一排）。 我们将长排称为动力总线。 如果您的原型板只有两排长排（每侧一个，我们将一排连接到+3.3V，另一排接地。如果您的原型板每侧有两排长排，那么两排接地，一排接地。 将是+3.3V。用黑色记号笔在每一行上标出电压。在原型板的中间，你会发现两组孔放置在一个0.1英寸的网格中。每排相邻的五个引脚是电连接的。 我们通常将元件插入这些孔中。IC 芯片放置在原型板上，这样两排引脚横跨中心谷。要连接到 TM4C123，我们可以从微控制器板的底部运行公-公实心线到 原型板。例如，假设我们希望将 TM4C123 PE1 输出连接到 470 欧姆电阻，如图 8.1 所示。首先，剪下一根足够长的 24 规格实心线，以便从 PE1 和电阻的引脚 1 到达。接下来，剥去大约 每端 0.25 英寸。将电线的一端放入孔中PE1 和另一端位于由电阻器共用的 5 孔行中剩余的四个孔之一中。     1. 源程序 2. // \*\*\*\*\* 0. Documentation Section \*\*\*\*\* 3. // SwitchLEDInterface.c for Lab 8 4. // Runs on LM4F120/TM4C123 5. // Use simple programming structures in C to toggle an LED 6. // while a button is pressed and turn the LED on when the 7. // button is released.  This lab requires external hardware 8. // to be wired to the LaunchPad using the prototyping board. 9. // January 11, 2014 11. // Lab 8 12. //      Jon Valvano and Ramesh Yerraballi 13. //      November 21, 2013 15. // \*\*\*\*\* 1. Pre-processor Directives Section \*\*\*\*\* 16. #include "TExaS.h" 17. #include "tm4c123gh6pm.h" 19. // \*\*\*\*\* 2. Global Declarations Section \*\*\*\*\* 21. // FUNCTION PROTOTYPES: Each subroutine defined 22. **void** DisableInterrupts(**void**); // Disable interrupts 23. **void** EnableInterrupts(**void**);  // Enable interrupts 24. **void** PortE\_Init(**void**); 25. **void** delay1ms(unsigned **long** time); 26. // \*\*\*\*\* 3. Subroutines Section \*\*\*\*\* 27. unsigned **long** In; 28. unsigned **long** Out; 30. // PE0, PB0, or PA2 connected to positive logic momentary switch using 10 k ohm pull down resistor 31. // PE1, PB1, or PA3 connected to positive logic LED through 470 ohm current limiting resistor 32. // To avoid damaging your hardware, ensure that your circuits match the schematic 33. // shown in Lab8\_artist.sch (PCB Artist schematic file) or 34. // Lab8\_artist.pdf (compatible with many various readers like Adobe Acrobat). 35. **int** main(**void**) 36. { 37. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 38. // The following version tests input on PE0 and output on PE1 39. //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 40. TExaS\_Init(SW\_PIN\_PE0, LED\_PIN\_PE1); // activate grader and set system clock to 80 MHz 41. PortE\_Init(); 42. EnableInterrupts(); // enable interrupts for the grader 43. **while** (1) 44. { 46. In = GPIO\_PORTE\_DATA\_R & 0x01; // get PE0 input value 47. **if**(In==0x01){ 48. GPIO\_PORTE\_DATA\_R ^= 0x02; // close 49. } 50. **else** GPIO\_PORTE\_DATA\_R |= 0x02; // open 51. delay1ms(100); 52. // GPIO\_PORTE\_DATA\_R |= 0x02; // open 53. } 54. } 56. **void** PortE\_Init(**void**) 57. { 58. **volatile** unsigned **long** delay; 59. SYSCTL\_RCGC2\_R |= 0x00000010;    // 1) E clock 60. delay = SYSCTL\_RCGC2\_R;          // delay 61. GPIO\_PORTE\_LOCK\_R = 0x4C4F434B;  // 2) unlock PortF PF0 62. GPIO\_PORTE\_CR\_R |= 0x1F;         // allow changes to PF4-0 63. GPIO\_PORTE\_AMSEL\_R &= 0x00;      // 3) disable analog function 64. GPIO\_PORTE\_PCTL\_R &= 0x00000000; // 4) GPIO clear bit PCTL 65. GPIO\_PORTE\_DIR\_R = 0x02;         // 5) PE1out, PE0 in 66. GPIO\_PORTE\_AFSEL\_R &= 0x00;      // 6) no alternate function 67. // GPIO\_PORTE\_PUR\_R |= 0x01;        // enable pullup resistors on PE4,PE0 68. GPIO\_PORTE\_DEN\_R |= 0x03;        // 7) enable digital pins PF4-PF0 69. }  72. // Subroutine to wait 1ms 73. // Inputs: None 74. // Outputs: None 75. // Notes: ... 76. **void** delay1ms(unsigned **long** time){ 77. unsigned **long** i; 78. **while**(time > 0){ 79. i = 15913; 80. **while**(i > 0){ 81. i = i - 1; 82. } 83. time = time - 1; 84. } 85. } 86. 功能描述：   a) 在模拟器和真实板上运行启动代码。 您应该能够重复本章教学视频中演示的测量。  b) 部分首先，将振荡频率更改为 10 Hz，并使用现有的调试转储来证明它正在切换 0.05 秒开启和 0.05 秒关闭。 您应该在模拟和真实板上运行此修改。 您会注意到，简单地将等待计数器一分为二并不能得到精确的解决方案。 这是因为每次循环都包括等待、输出和转储。 将等待计数器分成两半只是将等待分开，而不会更改输出和转储的时间。 您只需使其在 10 Hz 的 10% 范围内运行。 模拟器并不完美，因此我们在模拟中接受 ±25% 的时间精度，但在实际电路板上需要 ±1% 的时间精度。  c) 部分接下来，添加仅在按下任一开关时读取输入和切换输出的功能。 模拟模式下的自动评分器应该会给你一些满足输入/输出/时间规格的分数。 您可能需要调整延迟功能，以便分级员以预期 10 Hz 的 ±25% 看到它。  d) 最后，添加将输入/输出信息转储到 Data[] 数组中的调试工具。 您应该使用 0x13 屏蔽端口 F 数据以仅选择位 4、1 和 0。   1. 环境设置   要运行 Lab 9 评分器，必须执行两项操作。首先，执行project→options，然后选择Debug选项卡。调试参数字段必须包含 –dedXLab9。其次，必须将 edXLab9.dll 文件添加到本地Keil\ARM\BIN 文件夹中。    3、注意事项  1）初始化PORTE端口。  2）保证delay时间为100ms。   1. 适当修改程序，使按下按键为灯泡闪烁，不按下按键灯泡常亮。   4、上板时注意  1） 在 Keil 中，执行**Project->OptionsForTarget**.。在**Target**选项卡中，选择**Operating System**下拉菜单中的"TExaS"。在**Debug**选项卡中，单击右侧的Use radio，然后选择 Stellaris ICDI。  2） 通过执行 **Project->BuildProject**  3） 通过执行 **Flash->Download**下载将目标代码下载到 Flash EEPROM  4） 通过执行 **Debug->Start/StopDebuggingSession**调试会话  5） 通过执行 **Debug->OSsupport->TExaSGrader2.0** 展示得分板  6） 通过**Debug->Run**运行执行的程序  7） 在尝试评分之前调试软件， 请先调试您的软件。当您准备好运行刻度器时，请按重置，然后运行。通过单击**Grade**按钮调用评分器（按照ActionMsg窗口中的说明进行操作）   1. 源程序 2. // \*\*\*\*\* 0. Documentation Section \*\*\*\*\* 3. // main.c for Lab 9 4. // Runs on LM4F120/TM4C123 5. // In this lab we are learning functional debugging by dumping 6. //   recorded I/O data into a buffer 7. // February 21, 2014 9. // Lab 9 10. //      Jon Valvano and Ramesh Yerraballi 12. // \*\*\*\*\* 1. Pre-processor Directives Section \*\*\*\*\* 13. #include "TExaS.h" 14. #include "tm4c123gh6pm.h" 16. // \*\*\*\*\* 2. Global Declarations Section \*\*\*\*\* 18. // FUNCTION PROTOTYPES: Each subroutine defined 19. **void** DisableInterrupts(**void**); // Disable interrupts 20. **void** EnableInterrupts(**void**);  // Enable interrupts 22. // \*\*\*\*\* 3. Subroutines Section \*\*\*\*\*   26. /\* 27. This Lab9 starter project is the same as C9\_Debugging example but 28. includes the connections to the Lab9 grader. You will make three changes. 29. First, make the LED flash at 10 Hz. In other words, make it turn on for 0.05 seconds, 30. and then turn off for 0.05 seconds. 31. Second, make the LED flash if either switch SW1 or SW2 are pressed 32. (this means either PF4 or PF0 is 0). 33. Third, record PortF bits 4,1,0 every time the input changes or the output changes. 34. For example, if your system detects a change in either PF4 or PF0 input, 35. record PortF bits 4,1,0. If your system causes a change in PF1, record PortF bits 4,1,0. 37. If both PF4 and PF0 switch are not pressed, the PF1 output should be low. 38. If either PF4 or PF0 switches is pressed, the output toggles at 10 Hz (±10%). 39. Information collected in the Data array matches the I/O on PortF. 40. 50 data points are collected only on a change in input or a change in output. 41. (i.e., no adjacent elements in the array are equal). 43. \*/  46. **void** PortF\_Init(**void**){ **volatile** unsigned **long** delay; 47. SYSCTL\_RCGC2\_R |= 0x00000020;     // 1) activate clock for Port F 48. delay = SYSCTL\_RCGC2\_R;           // allow time for clock to start 49. GPIO\_PORTF\_LOCK\_R = 0x4C4F434B;   // 2) unlock GPIO Port F 50. GPIO\_PORTF\_CR\_R = 0x1F;           // allow changes to PF4-0 51. // only PF0 needs to be unlocked, other bits can't be locked 52. GPIO\_PORTF\_AMSEL\_R = 0x00;        // 3) disable analog on PF 53. GPIO\_PORTF\_PCTL\_R = 0x00000000;   // 4) PCTL GPIO on PF4-0 54. GPIO\_PORTF\_DIR\_R = 0x0E;          // 5) PF4,PF0 in, PF3-1 out 55. GPIO\_PORTF\_AFSEL\_R = 0x00;        // 6) disable alt funct on PF7-0 56. GPIO\_PORTF\_PUR\_R = 0x11;          // enable pull-up on PF0 and PF4 57. GPIO\_PORTF\_DEN\_R = 0x1F;          // 7) enable digital I/O on PF4-0 58. } 60. // Initialize SysTick with busy wait running at bus clock. 61. **void** SysTick\_Init(**void**){ 62. NVIC\_ST\_CTRL\_R = 0;                   // disable SysTick during setup 63. NVIC\_ST\_RELOAD\_R = 0x00FFFFFF;        // maximum reload value 64. NVIC\_ST\_CURRENT\_R = 0;                // any write to current clears it 65. NVIC\_ST\_CTRL\_R = 0x00000005;          // enable SysTick with core clock 66. } 67. unsigned **long** Led; 68. **void** Delay(**void**){unsigned **long** **volatile** time; 69. time = 160000; // 0.1sec 70. **while**(time){ 71. time--; 72. } 73. } 75. **void** delay\_50ms(unsigned **long** n){unsigned **long** **volatile** time; 76. **while**(n--){ 77. time = 80000; // 0.1sec 78. **while**(time){ 79. time--; 80. } 81. } 82. } 84. // first data point is wrong, the other 49 will be correct 85. unsigned **long** Time[50]; 86. // you must leave the Data array defined exactly as it is 87. unsigned **long** Data[50]; 88. **int** main(**void**){ 89. unsigned **long** i,last,now; 90. unsigned **long** myIn,last4,last1,last0,now4,now1,now0; 91. TExaS\_Init(SW\_PIN\_PF40, LED\_PIN\_PF1);  // activate grader and set system clock to 16 MHz 92. PortF\_Init();   // initialize PF1 to output 93. SysTick\_Init(); // initialize SysTick, runs at 16 MHz 94. i = 0;          // array index 95. last = NVIC\_ST\_CURRENT\_R; 96. last4 = GPIO\_PORTF\_DATA\_R&0x10; 97. last1 = GPIO\_PORTF\_DATA\_R&0x02; 98. last0 = GPIO\_PORTF\_DATA\_R&0x01; 99. EnableInterrupts();           // enable interrupts for the grader 100. **while**(1){ 101. myIn = GPIO\_PORTF\_DATA\_R&0x11; // read PF4 and PF0 into In 102. **if**(myIn!=0x11){ // if any one press(which it is equal to 0) 103. Led = GPIO\_PORTF\_DATA\_R;   // read previous 104. Led = Led^0x02;            // toggle red LED 105. GPIO\_PORTF\_DATA\_R = Led;   // output 106. **if**(i<50){ 107. now = NVIC\_ST\_CURRENT\_R; 108. now4 = GPIO\_PORTF\_DATA\_R&0x10; 109. now1 = GPIO\_PORTF\_DATA\_R&0x02; 110. now0 = GPIO\_PORTF\_DATA\_R&0x01; 111. Time[i] = (last-now)&0x00FFFFFF;  // 24-bit time difference 112. **if**(last4!=now4 || last1!=now1 || last0!=now0){ 113. Data[i] = GPIO\_PORTF\_DATA\_R&0x13; // record PF4,1,0 114. } 115. last = now; 116. last4 = now4; 117. last1 = now1; 118. last0 = now0; 119. i++; 120. } 121. // Delay(); 122. delay\_50ms(1); // wait for 50 ms 123. GPIO\_PORTF\_DATA\_R &= ~0x0E;   // output to close the led 124. **if**(i<50){ 125. now = NVIC\_ST\_CURRENT\_R; 126. now4 = GPIO\_PORTF\_DATA\_R&0x10; 127. now1 = GPIO\_PORTF\_DATA\_R&0x02; 128. now0 = GPIO\_PORTF\_DATA\_R&0x01; 129. Time[i] = (last-now)&0x00FFFFFF;  // 24-bit time difference 130. **if**(last4!=now4 || last1!=now1 || last0!=now0){ 131. Data[i] = GPIO\_PORTF\_DATA\_R&0x13; // record PF4,1,0 132. } 133. last = now; 134. last4 = now4; 135. last1 = now1; 136. last0 = now0; 137. i++; 138. } 139. delay\_50ms(1); // wait for 50 ms 140. } 141. } 142. }  145. // Color    LED(s) PortF 146. // dark     ---    0 147. // red      R--    0x02 148. // blue     --B    0x04 149. // green    -G-    0x08 150. // yellow   RG-    0x0A 151. // sky blue -GB    0x0C 152. // white    RGB    0x0E 153. // pink     R-B    0x06 | | | | | | |
| 四、实验结果及分析和（或）源程序调试过程   1. 输入输出验证、程序功能验证        1. 逻辑分析仪结果      1. 程序得分      1. 上板结果   同时按下SW1和SW2，黄灯亮起：    不按SW1和SW2:     1. 输入输出验证、程序功能验证      1. 程序得分      1. 上板结果   不按按钮，红灯亮起：    按下按钮，红灯以100ms的频率闪烁:       1. 输入输出验证、程序功能验证      1. 程序得分     3、每当 PF0 或 PF4 为低电平时，逻辑分析仪输出显示 PF1 以 10 Hz 切换 | | | | | | |
| 五、实验心得  1、本实验其实和Lab6很相似，但是之前理解错了题意，没有看到将绿灯0E转为黄灯0A,一直处在60分的状态。找bug找了很久，果然要仔细读题。  2、Floating Point Hardware需选择Not Used    否则会出现错误    3、通过这次实验，我们小组成员进一步学习了延迟函数和GPIO通信的相关知识 | | | | | | |