

**ĐẠI HỌC QUỐC GIA TPHCM TRƯỜNG ĐẠI HỌC BÁCH KHOA**



**KHOA ĐIỆN – ĐIỆN TỬ**

**BÁO CÁO BÀI TẬP**

**THIẾT KẾ HỆ THỐNG NHÚNG NÂNG CAO**

*Giảng viên hướng dẫn:* **PGS.TS. Trương Quang Vinh**

*Học viên thực hiện:*

|  |  |
| --- | --- |
| **Huỳnh Phạm Nhất Triều** | **2370014** |

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

Mục lục

[1 Bài 1 3](#_Toc148805522)

[1.1 Đề bài 3](#_Toc148805523)

[1.2 Các bước thực hiện 3](#_Toc148805524)

[1.3 Kết quả thực hiện 4](#_Toc148805525)

[1.4 Source code 5](#_Toc148805526)

[2 Bài 2 6](#_Toc148805527)

[2.1 Đề bài 7](#_Toc148805528)

[2.2 Các bước thực hiện 7](#_Toc148805529)

[2.3 Kết quả thực hiện 8](#_Toc148805530)

[2.4 Source code 8](#_Toc148805531)

[3 Bài 3 10](#_Toc148805532)

[3.1 Đề bài 10](#_Toc148805533)

[3.2 Các bước thực hiện 10](#_Toc148805534)

[3.3 Kết quả thực hiện 12](#_Toc148805535)

[3.4 Source code 12](#_Toc148805536)

[4 Tài liệu tham khảo 15](#_Toc148805537)

# Bài 1

## Đề bài

Write a program to generate a clock signal 1Hz at PF1, and a clock signal 2Hz at PF2.

## Các bước thực hiện

- Cấu hình GPIO cho Port F, Pin 1, 2 ở mode output.

A close up of a text

Description automatically generated

- Viết hàm tạo xung Clock 2 Hz cho chân PF2, ứng với chu kỳ 0.5s bằng cách thay đổi mức ở ngõ ra liên tục

+ Tạo mức 0 ở ngõ ra

+ Delay 0.5s

+ Tạo mức 1 ở ngõ ra

+ Delay 0.5s

A screenshot of a computer code

Description automatically generated

- Tương tự cho Clock 1 Hz cho chân PF1, ứng với chu kỳ 1s:

+ Tạo mức 0 ở ngõ ra.

+ Gọi hàm **generateClock1()** (tương đương delay 1s).

+ Tạo mức 1 ở ngõ ra.

+ Gọi hàm generateClock1 (tương đương delay 1s).

A screen shot of a computer code

Description automatically generated

- Trong hàm main() gọi hàm configureGPIO() và trong loop **while(1)** gọi hàm **generateClock2()** để tạo ra 2 xung như yêu cầu của đề bài.

A screenshot of a computer program

Description automatically generated

## Kết quả thực hiện

[*https://drive.google.com/file/d/1J460tjMUmTZB02ye6URO7urZaKSanUON/view?usp=sharing*](https://drive.google.com/file/d/1J460tjMUmTZB02ye6URO7urZaKSanUON/view?usp=sharing)

## Source code

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/fpu.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/sysctl.h"

**#define** PIN\_HIGH 0xFF

**#define** PIN\_LOW 0x00

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Configure the GPIO and its pins. This must be called before generateClock().

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **configureGPIO**(**void**){

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2);

}

**void** **generateClock1**(**void**){

// High pin

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, PIN\_HIGH);

// Delay 0.5s

**SysCtlDelay**(**SysCtlClockGet**() / 6);

// Low pin

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, PIN\_LOW);

// Delay 0.5s

**SysCtlDelay**(**SysCtlClockGet**() / 6);

}

**void** **generateClock2**(**void**){

// High pin

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, PIN\_HIGH);

// Delay 1s

generateClock1();

// Low pin

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, PIN\_LOW);

// Delay 1s

generateClock1();

}

**int** **main**(**void**)

{

//

// Set the clocking to run directly from the crystal.

//

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ |

SYSCTL\_OSC\_MAIN);

// Initialize GPIO

configureGPIO();

**while**(1)

{

//

// Generate clock as required

//

generateClock2();

}

}

# Bài 2

## Đề bài

Write a program to count a 8bit number and display on 8 single LEDs at port PB[7:0].

## Các bước thực hiện

- Cấu hình GPIO cho Port B, Pin 1, 2, …, 7 ở mode output.

A black text on a white background

Description automatically generated

- Viết hàm xuất giá trị của biến đếm ra các LED

A close up of a code

Description automatically generated

- Trong hàm main(), khởi tạo biến count tương trưng cho số 8-bit, gọi hàm configureGPIO() và trong loop while(1):

+ Kiểm tra biến **count** có nằm ngoài giá trị yêu cầu hay không (0-255).

+ Gọi hàm **exportToLEDs()** và truyền giá trị biến count vào để xuất ra các LED.

+ Delay 1s.

+ Tăng biến đếm **count** lên 1.

A screenshot of a computer program

Description automatically generated

## Kết quả thực hiện

[*https://drive.google.com/file/d/11n01tT18E5S8mQsmz9Zw6Hl5SKWknmz0/view?usp=sharing*](https://drive.google.com/file/d/11n01tT18E5S8mQsmz9Zw6Hl5SKWknmz0/view?usp=sharing)

## Source code

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/fpu.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/sysctl.h"

**#define** PIN\_HIGH 0xFF

**#define** PIN\_LOW 0x00

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Configure the GPIO and its pins. This must be called before generateClock().

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**void** **configureGPIO**(**void**){

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOB);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTB\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|

GPIO\_PIN\_3|GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7);

}

**void** **exportToLEDs**(uint8\_t count){

**GPIOPinWrite**(GPIO\_PORTB\_BASE, GPIO\_PIN\_0|GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3|

GPIO\_PIN\_4|GPIO\_PIN\_5|GPIO\_PIN\_6|GPIO\_PIN\_7, count);

}

**int** **main**(**void**)

{

uint8\_t count = 0;

// Set the clocking to run directly from the crystal.

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_XTAL\_16MHZ |

SYSCTL\_OSC\_MAIN);

// Initialize GPIO

configureGPIO();

**while**(1)

{

// Check count value is out of range or not (0-255)

**if** (count > 255)

count = 0;

// Export count value to LED

exportToLEDs(count);

// Delay 1 second

**SysCtlDelay**(**SysCtlClockGet**() / 3);

// Increase count value

count ++;

}

}

# Bài 3

## Đề bài

Write a program to measure the temperature using the internal sensor, and transmit them to UART after every second.

## Các bước thực hiện

- Cấu hình UART 0 với tốc độ baund 115200, 8 bit dữ liệu, 1 bit stop, và không parity

A screenshot of a computer code

Description automatically generated

- Cấu hình ADC 0 với sample sequence number là 2 (cho phép lấy tối đa 4 mẫu), các step 0, 1, 2, 3 được cấu hình là **ADC\_CTL\_TS** ứng với giá trị của “**internal temperature sensor**” với step 3 được cấu hình cuối cùng (**ADC\_CTL\_END**) và raise ngắt khi hoàn thành (**ADC\_CTL\_IE**).

A screenshot of a computer program

Description automatically generated

- Viết hàm để truyền một chuỗi ký tự:

A computer code with text

Description automatically generated

- Trong hàm main(), gọi hàm configureUART(), configureADC() để khởi tạo các ngoại vi cần thiết và trong loop while(1):

+ Đo nhiệt độ của cảm biến nhiệt độ và lưu vào buffer (4 giá trị).

A black text on a white background

Description automatically generated+ Tính trung bình và convert từ giá trị ADC sang giá trị nhiệt độ theo công thức trong datasheet:

+ Convert giá trị từ uint32\_t sang chuỗi string.

+ Truyền chuỗi vừa convert qua UART sử dụng hàm UARTStringPut().

+ Delay 1s.

A screenshot of a computer program

Description automatically generated

## Kết quả thực hiện

[*https://drive.google.com/file/d/1Jnt1f585BgOl07N9KvLN6eJXNhhwz4VV/view?usp=sharing*](https://drive.google.com/file/d/1Jnt1f585BgOl07N9KvLN6eJXNhhwz4VV/view?usp=sharing)

## Source code

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <string.h>

**#include** <stdio.h>

**#include** <math.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/pin\_map.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/uart.h"

**#include** "driverlib/adc.h"

**void** **configureUART**(**void**){

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTConfigSetExpClk**(UART0\_BASE, **SysCtlClockGet**(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));

}

**void** **configureADC**(**void**){

//

// Enable the ADC0 module.

//

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

//

// Wait for the ADC0 module to be ready.

//

**while**(!**SysCtlPeripheralReady**(SYSCTL\_PERIPH\_ADC0))

{

}

//

// Enable the first sample sequencer to capture the value of channel 0 when

// the processor trigger occurs.

//

**ADCSequenceConfigure**(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 3, ADC\_CTL\_IE | ADC\_CTL\_END | ADC\_CTL\_TS);

**ADCSequenceEnable**(ADC0\_BASE, 2);

}

**void** **UARTStringPut**(uint32\_t ui32Base,**const** **char** \*str, size\_t count){

size\_t i;

**char** ch;

**for**(i = 0; i < count; i++){

ch = str[i];

**UARTCharPut**(ui32Base, (**char**) ch );

}

}

**int** **main**(**void**) {

uint32\_t ui32Temp[4];

**double** tempAvg;

**double** finalResult;

**char** tempBuffer[50] = {'\0'};

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

configureUART();

configureADC();

**while** (1)

{

/\* Measure temperature \*/

// Clear the interrupt flag which indicating the ADC conversion process is complete

**ADCIntClear**(ADC0\_BASE, 2);

// Trigger the sample sequence.

**ADCProcessorTrigger**(ADC0\_BASE, 2);

// Wait until the sample sequence has completed.

**while**(!**ADCIntStatus**(ADC0\_BASE, 2, false));

// Read the value from the ADC.

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32Temp);

// Calculate average value for 4 samples

tempAvg = (**double**) ((ui32Temp[0] + ui32Temp[1] + ui32Temp[2] + ui32Temp[3])/4);

// Convert ADC value to Temperature

finalResult = **round**((**double**)((1475 \* 4096 - (75 \* 33 \* tempAvg)) / 40960));

/\* Transmit to UART \*/

// Convert the unsigned integer ui32Temp value to a string tempBuffer.

**sprintf**(tempBuffer, "Current Temp: %u\r\n", (uint32\_t) finalResult);

// Transmit

UARTStringPut(UART0\_BASE, tempBuffer, **strlen**(tempBuffer));

/\* Delay 1s \*/

**SysCtlDelay**(**SysCtlClockGet**() / 3);

}

}

# Tài liệu tham khảo

[1] TivaWare™ Peripheral Driver Library for C Series User's Guide (Rev. E)

[2] Tiva™ C Series TM4C123GH6PM Microcontroller Data Sheet datasheet (Rev. E)