Câu chuyện bắt đầu từ một cậu bé,
và một ý tưởng
có thể
làm thay đổi thế giới...

PAY IT FORWARD

Đó là khi bạn giúp đỡ 3 người bạn không quen biết,
dù là bằng thời gian,
hay công sức,
hay kinh nghiệm,
hay kiến thức,
hay tiền bạc, ...
của mình.

Mà không chờ đợi một sự báo ân nào.

Chỉ cần mỗi người trong 3 người đó,
lại đem những gì mình có, mà người khác cần,
tiếp tục giúp đỡ thêm 3 người nữa.

Chính những người-giúp-đỡ, và người-được-giúp-đỡ,
sẽ là những người góp phần thay đổi thế giới...

Một thế giới sẻ chia kiến thức - và yêu thương ...

PAY IT FORWARD ...

Chúng tôi không sáng tạo ra câu nói này.

Pay it forward...

Hãy tri ân người giúp mình bằng cách giúp đỡ người khác
Cho đi không phải để nhận lại.

TIMER

28/10/2014

payitforward.edu.vn

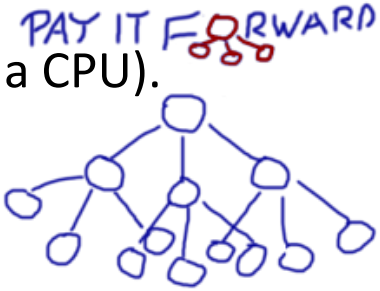
TIVA ARM Cortex-M4
TM4C123G Tutorial

- GIỚI THIỆU CHUNG
- CÁC CHẾ ĐỘ HOẠT ĐỘNG
- CHƯƠNG TRÌNH MẪU

GIỚI THIỆU CHUNG VỀ GENERAL PURPOSE TIMER MODULE (GPTM)

❖ **TIVA ARM Cortex-M4 TM4C123G** có:

- 6 timer 16/32 bit và 6 timer 32/64 bit.
- 12 chân capture/compare/PWM 16/32 bit và 12 chân capture/compare/PWM 32/64bit
- Counter đếm lên hoặc xuống.
- Tạo xung PWM đơn giản (không có deadband).
- Hỗ trợ đồng bộ timer, dừng lại trong quá trình debug.
- Có thể dùng trong lấy mẫu ADC (Analog- Digiatal Converter, chuyển tín hiệu tương tự sang tín hiệu số) hoặc truyền DMA (Direct Memory Access – cơ chế truyền dữ liệu trực tiếp giữa thiết bị I/O và RAM không cần qua CPU).

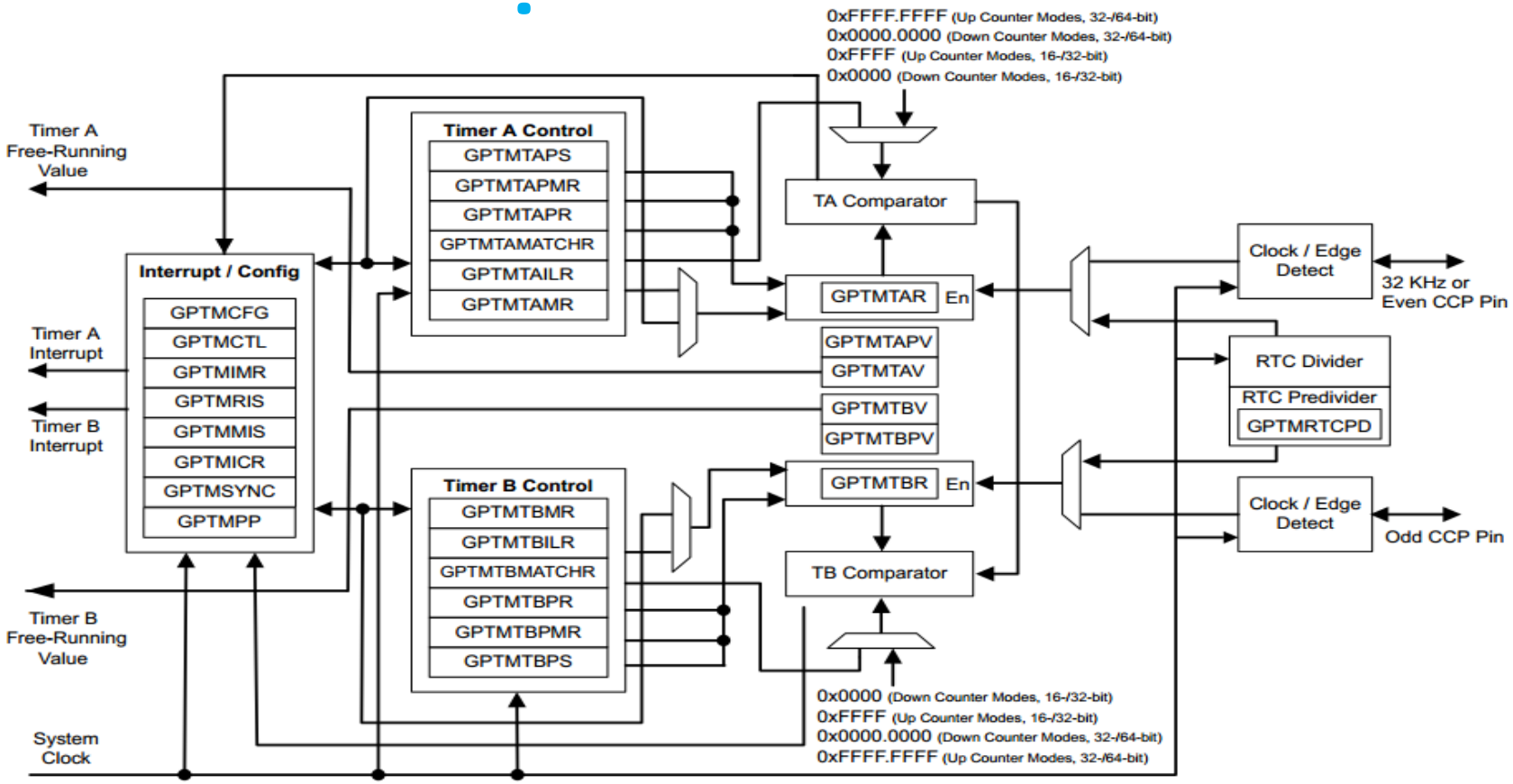


GIỚI THIỆU CHUNG VỀ GPTM

❖ **Các chế độ timer**:

- **One-shot**: Timer chạy 1 lần duy nhất.
- **Periodic:** Timer chạy tuần hoàn.
- **Input Edge count or time capture**: Đếm số cạnh xung của tín hiệu vào hoặc thực hiện chức năng capture.
- **PWM generation**: Tạo xung PWM.
- **Real-Time Clock**: Thời gian thực.

GIỚI THIỆU CHUNG VỀ GPTM



Sơ đồ khối của module GPTM

GIỚI THIỆU CHUNG VỀ GPTM

Mode	Timer Use	Count Direction	Counter Size		Prescaler Size[a]		Prescaler Behavior (Count Direction)
			16/32-bit GPTM	32/64-bit Wide GPTM	16/32-bit GPTM	32/64-bit Wide GPTM	
One-shot	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
Periodic	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Up), Prescaler (Down)
	Concatenated	Up or Down	32-bit	64-bit	-	-	N/A
RTC	Concatenated	Up	32-bit	64-bit	-	-	N/A
Edge Count	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
Edge Time	Individual	Up or Down	16-bit	32-bit	8-bit	16-bit	Timer Extension (Both)
PWM	Individual	Down	16-bit	32-bit	8-bit	16-bit	Timer Extension

a. The prescaler is only available when the timers are used individually

General Purpose Timer Capabilities

GIỚI THIỆU CHUNG VỀ GPTM

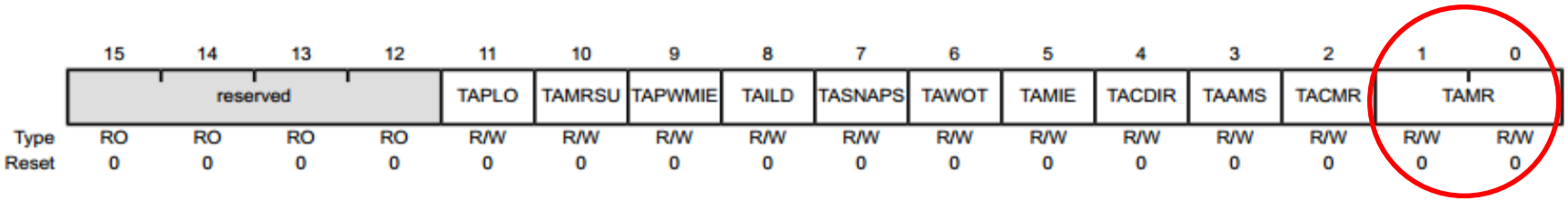
- Để cấu hình bằng phần mềm cho GPTM ta sử dụng các thanh ghi sau:
  - Thanh ghi GPTM Configuration – **GPTMCFG**
  - Thanh khi GPTM Timer A Mode – **GPTMTAMR**
  - Thanh ghi GPTM Timer B Mode – **GPTMTBMR**
- Trong thực tế, khi lập trình ta sẽ sử dụng các hàm API trong driverLib để tương tác thuận tiện hơn với các thanh ghi.

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.1. Chế độ One-Shot và Periodic:

- Việc chọn chế độ one-shot hay periodic được quyết định bởi giá trị trong vùng TnMR (gồm 2 bit) của thanh ghi GPTM Timer n Mode (**GPTMTnMR** – 'n' có thể là 'A' hoặc 'B').

❖ ***Thanh ghi GPTMTnMR:***



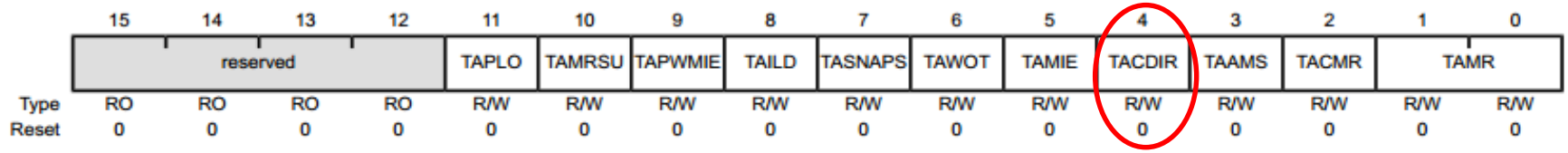
TnMR	Mode
0x0	Reserved
0x1	One-Shot Timer mode
0x2	Periodic Timer mode
0x3	Capture mode

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.1. Chế độ One-Shot và Periodic(tt):

- Việc chọn chế độ đếm lên hay đếm xuống được quy định bởi bit TnCDIR trong thanh ghi **GPTMTnMR .**

❖ ***Thanh ghi GPTMTnMR:***



TnCDIR	Count direction
0	Count down
1	Count up

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.1. Chế độ One-Shot và Periodic(tt):

- Mỗi khi chương trình set bit **TnEN** của thanh ghi GPTM Control (**GPTMCTL**) lên 1 thì timer bắt đầu đếm lên từ 0 hay đếm xuống từ giá trị đã đặt trước.

❖ ***Thanh ghi GPTMCTL:***

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Các giá trị đặt trước này được quy định trong thanh ghi **GPTMTnILR** và thanh ghi **GPTMTnPR** (GPTM Timer n Prescale).

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.1. Chế độ One-Shot và Periodic(tt):

- **Count down**: Khi timer đếm xuống và về 0, timer nạp lại giá trị từ 2 thanh ghi GPTMTnILR và GPTMTnPR trong chu kỳ kế tiếp.

- **Count up**: Khi timer đếm lên và gặp sự kiệm tràn (giá trị timer bằng giá trị lưu trong 2 thanh ghi GPTMnILR và GPTMnPR) thì timer nap lại giá trị 0.

- **One-shot**: khi sự kiện tràn timer xảy ra, bit TnEN trong thanh ghi GPTMCTL được xóa đi và dừng timer.

- **Periodic**: khi sự kiện tràn timer xảy ra, timer tiếp tục đếm lặp lại như cũ trong chu kỳ kế tiếp.

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.2. Chế độ Input Edge-Count – Đếm cạnh xung ngõ vào:

❖ ***Lưu ý:***

- Trong trường hợp đếm cạnh lên, thời gian ngõ vào ở mức cao phải là ít nhất 2 chu kỳ xung clock hệ thống.
- Tương tự cho trường hợp đếm cạnh xuống, ngõ vào phải ở mức thấp ít nhất 2 chu kỳ xung clock hệ thống.

➡ **Tần số tối đa ngõ vào để có thể phát hiện được cạnh xung là ¼ tần số hệ thống.**

- Trong chế độ đếm cạnh xung, timer có thể được cấu hình như là 1 counter 24 bit hoặc 48 bit đếm lên hoặc xuống phụ thuộc vào bộ chia (prescaler) và giá trị đếm lên lưu trong thanh ghi GPTMTnPR và bit thấp của thanh ghi GPTMTnR.

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.2. Chế độ Input Edge-Count – Đếm cạnh xung ngõ vào (tt):

- Để đưa timer vào chế độ đếm cạnh xung, ta phải đưa bit TnCMR trong thanh ghi GPTMTnMR xuống 0.

❖ ***Thanh ghi GPTMTnMR:***

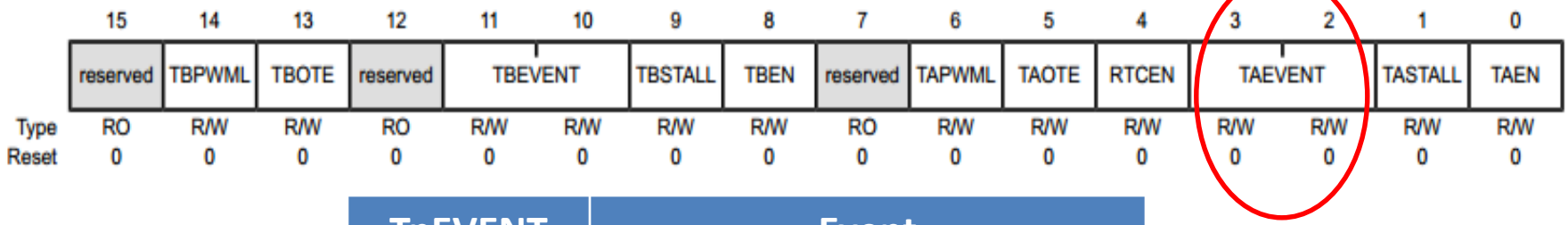
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TAPLO	TAMRSU	TAPWMIE	TAILD	TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CÁC CHẾ ĐỘ HOẠT ĐỘNG CỦA TIMER

2.2. Chế độ Input Edge-Count – Đếm cạnh xung ngõ vào(tt):

- Trong chế độ đếm cạnh xung, timer có 3 sự kiện: đếm cạnh lên, đếm cạnh xuống hoặc cả 2, để chọn sự kiện ta dùng 2 bit TnEVENT của thanh ghi GPTMCTL.

❖ ***Thanh ghi GPTMCTL:***



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TnEVENT	Event
0x0	Cạnh lên
0x1	Cạnh xuống
0x2	Reserved
0x3	Cạnh lên và cạnh xuống

CHƯƠNG TRÌNH MẪU

1. Chương trình mẫu:

- Chương trình sau đây dùng để chớp tắt led xanh dương với chu kỳ 1 giây dùng timer.

- ***Code mẫu:***

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
int main(void)
{
            uint32_t ui32Period;
            SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
            SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
            GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
            SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
            TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
            ui32Period = (SysCtlClockGet() / 10) / 2;
            TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period - 1);
            IntEnable(INT_TIMER0A);
            TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
            IntMasterEnable();
            TimerEnable(TIMER0_BASE, TIMER_A);
            while(1)
            {}
}
```

1. Chương trình mẫu (tt):

```
void Timer0IntHandler(void)
{
        // Clear the timer interrupt
        TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
        // Read the current state of the GPIO pin and
        // write back the opposite state
        if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
        {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
```

2. Phân tích chương trình:

2.1. Khai báo các thư viện và header cần thiết:

❖ ***Code:***

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
```

❖ ***Giải thích:***

- ***tm4c123gh6pm.h*:** header này chứa các định nghĩa về ngắt (interrupt) cũng như các thanh ghi trên board.

- ***#include "driverlib/interrupt.h"*:** header này định nghĩa và thay thế cho các tác vụ ngắt của "driverLib". Header này bao gồm các hàm API (Application Programming Interface – mục đính chính là cung cấp khả năng truy xuất đến các hàm hay dùng) bao gồm hàm cho phép ngắt "IntEnable" hoặc thiết lập thứ tự ưu tiên cho ngắt "IntPrioritySet".

- ***#include "driverlib/timer.h"*:** header này chứa các định nghĩa và thay thế cho các hàm Timer API của "driverLib". Header này bao gồm các hàm API như cấu hình thanh ghi "TimerConfigure" và "TimerLoadSet".

2. Phân tích chương trình(tt):

2.2. Chương trình chính:

2.2.1. Hàm main():

❖ ***Code:***

```
int main(void)
{
        uint32_t ui32Period;
}
```

❖ ***Giải thích:***

- Đầu tiên ta tạo 1 timer mới với thời gian delay được lưu trong biến đặt tên là "ui32Period" (kiểu unsigned int 32 bit).

2.2. Chương trình chính (tt):

2.2.2. Cấu hình clock:

❖ ***Code:***

```
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
```

❖ ***Giải thích:***

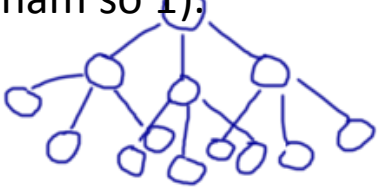
- **Tham số đầu tiên** "**SYSCTL_SYSDIV_5**" và thứ 2 "**SYSCTL_USE_PLL**":

- Nhằm tính toán tần số dao động của Processor Clock.
- Processor Clock có thể lấy từ OSC hoặc PLL, **tham số thứ 2** sẽ quyết định điều này.

  + OSC: **SYSCTL_USE_OSC**

  + PLL: **SYSCTL_USE_PLL**

- Tính toán Processor Clock: **Processor Clock = (tần số OSC hoặc PLL)/ x** với x được xác định từ tham số thứ 1.
- Lưu ý là nếu dùng PLL thì tần số PLL là 400MHz, trong clock path mặc định thêm 1 bộ chia 2, do đó trong trường hợp PLL thì Processor Clock = 200MHz/(tham số 1).



2.2. Chương trình chính (tt):

2.2.2. Cấu hình clock (tt):

- **Tham số thứ 3** "**SYSCTL_XTAL_16MHZ**":
  - PLL chỉ dao động tại 400MHz nhưng có thể được lái (chia nhỏ) bởi thạch anh dao động trong khoảng 5 đến 25MHz, do đó ta có thể dùng XTAL_16MHz để lái. Nếu thạch anh dao động ko trong khoảng này thì tham số đầu **SYSCTL_SYSDIV_5** không thể chia nhỏ tần số PLL được.

- **Tham số thứ 4** "**SYSCTL_OSC_MAIN**":
  - Nguồn dao động là main osc.
  - Kết hợp với tham số thứ 3, system clock dùng thạch anh 16MHz lấy từ main osc.

2.2. Chương trình chính (tt):

2.2.3. Cấu hình GPIO:

❖ ***Code:***

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
```

❖ ***Giải thích:***

Phần này ta cấu hình chân GPIO nối với LED là output như phần GPIO đã học.

2.2. Chương trình chính (tt):

2.2.4. Cấu hình Timer:

❖ ***Code:***

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
```

❖ ***Giải thích:***

- **Dòng 1:** Việc đầu tiên cần làm trước khi gọi bất kì ngoại vi nào từ hàm driverLib, ta phải enable clock ứng với ngoại vi đó, nếu không chương trình sẽ báo lỗi Fault ISR (lỗi địa chỉ).
- **Dòng 2:** Tiếp theo ta cấu hình Timer 0 là 1 timer 32 bit với periodic mode. Lưu ý ở đây timer 0 32bit bao gồm 2 timer 16bit nhỏ hơn là timer 0A và timer 0B và "TIMER0_BASE" là địa chỉ bắt đầu của thanh ghi Timer0.

2.2. Chương trình chính (tt):

2.2.5. Tính toán thời gian delay cho timer:

- Trong bài này ta muốn GPIO toggle với tần số 10Hz và chu kỳ nhiệm vụ là 50%, do đó ta cần tạo một ngắt tại ½ chu kỳ. Chu kỳ nhiệm vụ ở đây ta sẽ tính toán và vào biến ui32Period khai báo ờ đầu hàm main.

❖ ***Code:***

```
ui32Period = (SysCtlClockGet() / 10) / 2;
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

❖ ***Giải thích:***

- **Dòng 1**: Hàm **SysCtlClockGet()** sẽ cho ta số chu kỳ xung clock cần để tạo được 10Hz. Sau đó chia cho 2 ta ra được chu kì nhiệm vụ cần tìm, tức là thời gian timer cần delay
- **Dòng 2**:

+ 2 tham số đầu tiên khai báo là “TIMER0_BASE” và “TIMER_A” vì ta sẽ sử dụng timer 0A 16bit.

+ Tham số cuối cùng là thời gian delay của timer. Lưu ý phải trừ đi 1 do chương trình ngắt bắt đầu thực hiện khi timer về 0.

2.2. Chương trình chính (tt):

2.2.6. Cho phép ngắt:

❖ ***Code:***

```
IntEnable(INT_TIMER0A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
IntMasterEnable();
```

❖ ***Giải thích:***

- Ta cho phép ngắt, ở đây ta phải enable cả ngắt của timer và ngắt toàn cục NVIC (Nested Vector Interrupt Controller).

  IntEnable(): Cho phép vector ngắt gắn với timer 0A.

  TimerIntEnable(): Enable sự kiện ngắt gắn với timer, trong trường hợp này là chương trình ngắt sẽ được thực hiện khi timerA tràn.

  IntMasterEnable(): Enable ngắt toàn cục.

2.2. Chương trình chính (tt):

2.2.7. Cho phép Timer:

❖ ***Code:***

```
TimerEnable(TIMER0_BASE, TIMER_A);
```

❖ ***Giải thích:***

- Cuối cùng ta enable timer, sau lệnh này timer bắt đầu đếm và sẽ thực hiện chương trình ngắt khi timer tràn.

2.2. Chương trình chính (tt):

2.2.8. Vòng lặp vô tận while(1):

❖ ***Code:***

```
While(1)
{}
```

❖ ***Giải thích:***

- Ở đây ta thêm while(1) vì việc toggle chân GPIO chỉ xảy ra trong chương trình ngắt.

2.3. Tác vụ ngắt Timer – Timer Interruppt Handler:

❖ ***Code:***

```
void Timer0IntHandler(void)
{
        // Clear the timer interrupt
        TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
        // Read the current state of the GPIO pin and
        // write back the opposite state
        if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
        {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
        }
        else
        {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
        }
}
```

❖ ***Giải thích:***

- Việc đầu tiên khi vào chương trình ngắt chính là xóa cờ ngắt timer, việc còn lại là thực hiện những phần mình muốn. Trong bài này đơn giản ta cho chớp tắt LED xanh dương.

3. Sửa file tm4c123gh6pm_startup_ccs.c

❖ Tiếp theo ta cần làm sửa chữa file **tm4c123gh6pm_startup_ccs.c** trước khi chạy chương trình:

- Mở file tm4c123gh6pm_startup_ccs.c

- Ta sử dụng bao nhiêu tác vụ ngắt thì phải khai báo sau dòng **"extern void _c_int00(void);"**

- Trong trường hợp này ta sẽ thêm dòng màu đỏ sau:

```
extern void _c_int00(void);
extern void Timer0IntHandler(void); //Them dong nay
```

3. Sửa file tm4c123gh6pm_startup_ccs.c (tt):

- Cẩn thận tìm đúng dòng chú thích phù hợp với ngắt ta đang xài và thay thế chữ "IntDefaultHandler" bằng tên chương trình ngắt của ta.
- Ví dụ trong bài này, ta dùng ngắt timer 0 nên sẽ tìm dòng chú thích **"//Timer 0 subtimer A**" và thay thế như sau:

IntDefaultHandler, // Timer 0 subtimer A

Timer0IntHandler, // Timer 0 subtimer A

- Đến đây ta đã cấu hình và lập trình đầy đủ cho timer và hoàn toàn có thể chạy chương trình.

Tài liệu tham khảo

[1] Tiva tm4c1233h6pm Datasheet
[2] TM4C123G_LaunchPad_Workshop_Workbook

PAY IT FORWARD

payitforward.edu.vn