5trfgyu875res` D

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

TRƯỜNG BÁCH KHOA

KHOA TỰ ĐỘNG HÓA

------🙢⯎🙠*------*



**TÀI LIỆU LẬP TRÌNH**

**VI ĐIỀU KHIỂN MSP430-G2553**

**TÁC GIẢ:**

**Nguyễn Nhựt Khánh; MSSV: B2204**

Ngành: Cơ Điện Tử– Khóa: 48

**Tháng 5/2024**

MỤC LỤC

[MỤC LỤC ii](#_Toc166422772)

[1. Xung Clock 3](#_Toc166422773)

[1.1 Cấu hình sử dụng nguồn xung clock từ DCO 3](#_Toc166422774)

[1.2 Cấu hình sử dụng nguồn xung clock từ DCO 3](#_Toc166422775)

[1.3 flsan 4](#_Toc166422776)

[1.4 afsf 4](#_Toc166422777)

[2. GPIO 4](#_Toc166422778)

[2.1 Khai báo ngõ vào, ngõ ra 4](#_Toc166422779)

[2.2 Đặt mức logic 4](#_Toc166422780)

[2.3 Đọc giá trị đầu vào (Input) 4](#_Toc166422781)

[2.4 Điện trở kéo lên và kéo xuống 4](#_Toc166422782)

[2.5 Tham khảo 5](#_Toc166422783)

[3. Ngắt (Interrupt) 5](#_Toc166422784)

[3.1 Interrupt Flag Registers P1IFG, P2IFG 5](#_Toc166422785)

[3.2 Interrupt Edge Select Registers P1IES, P2IES 5](#_Toc166422786)

[3.3 Interrupt Enable P1IE, P2IE 5](#_Toc166422787)

[4. ABC 6](#_Toc166422788)

[5. ABC 6](#_Toc166422789)

[TÀI LIỆU THAM KHẢO (Style “Cap 1”) 7](#_Toc166422790)

1. Xung Clock

Cần phải phân biệt hai khái niệm : **Clock source**  và **clock signal**

* **Clock source**: Nguồn cung cấp dao động đầu vào tạo xung nhịp  (Màu đỏ)

+  **Thạch anh ngoài** (32.768Khz)

+  **Dao động nội** **DCO**(tần số cao: 400Khz và 16Mhz), **VLO**(tần số thấp : 12Khz)

* **Clock signal** : Xung nhịp được tạo ở đầu ra cho CPU và ngoại vi

+ Nguồn Clock cấp cho CPU gọi là MCLK

+ Nguồn Clock cấp cho ngoại vi : nếu tần số cao  **SMCLK** (Lấy từ DCO). Nếu tần số thấp **ACLK**  (Lấy từ VLO hoặc thạch anh ngoài)

**NOTE**: Khi lập trình nếu chúng ta không đả động gì đến Clock thì nguồn DCO sẽ chọn tần số mặc định ~ **1.048 Mhz**

* 1. Cấu hình sử dụng nguồn xung clock từ DCO

Thư viện hỗ trợ người dùng chỉnh sẵn 4 tần số của nguồn dao dộng DCO 1Mhz , 8Mhz, 12Mhz, 16Mhz

**Cách cấu hình:**

**BCSCTL1** = CALBC1\_xMHZ; // Thiết lập tần số DCO tới xMHz

**DCOCTL** = CALDCO\_xMHZ; // Thiết lập hệ số đồng hồ DCO

// Cấu hình SMCLK để sử dụng DCO như là nguồn xung clock với tần số xMHz

**BCSCTL2** |= SELM\_0; // Chọn DCO làm nguồn xung cho SMCLK

**Note**: Giá trị x có thể chọn 1 trong những tần số, 1, 4, 8 or 16.

**Ví dụ:**

void Config\_Clock(void){

if ( CALBC1\_1MHZ == 0xFF) while(1);

DCOCTL = 0;

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

BCSCTL2 |= SELM\_0;

}

* 1. Cấu hình sử dụng nguồn xung clock từ DCO

Cấu hình cho nguồn clock CPU (MCLK) chạy dùng nguồn VLO 12Khz

**Cách cấu hình:**

// Kích hoạt VLO

BCSCTL3 |= LFXT1S\_2; // Chọn VLO làm nguồn xung Low-Frequency

// Thực hiện chờ đợi cho VLO ổn định

\_\_bis\_SR\_register(SCG0); // Tắt tham số của ngoại vi để cấu hình

\_\_delay\_cycles(500); // Chờ đợi VLO ổn định

\_\_bic\_SR\_register(SCG0); // Cho phép tham số của ngoại vi

// Sử dụng VLO làm nguồn xung clock cho hệ thống

BCSCTL2 |= SELM\_3; // Chọn VLO làm nguồn xung cho MCLK

* 1. flsan
  2. afsf

1. GPIO
   1. Khai báo ngõ vào, ngõ ra

**Cú pháp:**

P<port>DIR |= <chân>; // Đặt chân <chân> của port <port> là chân output

P<port>DIR &= ~<chân>; // Đặt chân <chân> của port <port> là chân input

**Ví dụ** :

P1DIR |= BIT0; // Đặt chân P1.0 là chân output

P1DIR |= BIT0 + BIT6; // Đặt chân P1.0 và P1.6 là chân output

P2DIR &= ~BIT3; // Đặt chân P2.3 là chân input

* 1. Đặt mức logic

**Cú pháp:**

P<port>OUT |= <chân>; // Đặt lên mức HIGH

P<port>OUT &= ~<chân>; // Đặt xuống mức LOW

**Ví dụ:**

P1OUT |= BIT0; // Đặt chân P1.0 lên HIGH

P1OUT |= BIT0 + BIT6; // Đặt chân P1.0 và P1.6 lên HIGH

P2OUT &= ~BIT3; // // Đặt chân P1.3 lên LOW

Đọc giá trị chân INPUT:

* 1. Đọc giá trị đầu vào (Input)

**Cú pháp :**

<biến> = (P<port>IN & <chân>) != 0; // Đọc giá trị từ chân input và gán vào biến

**Ví dụ:**

*Đọc giá trị input từ chân P1.0*

If( P1IN & BIT0 ) == 0){   
 // code   
}

* 1. Điện trở kéo lên và kéo xuống

Mỗi bit trong mỗi thanh ghi **PxREN** enables or disables điện trở kéo lên hoặc kéo xuống của chân I / O tương ứng.

* Bit = 0: Pullup or pulldown resistor disabled
* Bit = 1: Pullup or pulldown resistor enabled

Bit tương ứng trong thanh ghi **PxOUT** chọn nếu pin được kéo lên hoặc kéo xuống

* Bit = 0: pulldown resistor
* Bit = 1: Pullup resistor

**Ví dụ**

P1DIR &= ~BUTTON\_PIN; // Thiết lập chân BUTTON là đầu vào

P1REN |= BUTTON\_PIN; // Bật điện trở treo trên chân BUTTON

P1OUT |= BUTTON\_PIN; // Đặt điện trở kéo lên

P1OUT &= ~BUTTON\_PIN; // Đặt điện trở kéo xuống

* 1. Tham khảo

<https://mlab.vn/bai-viet-ki-thuat/vi-dieu-khien-msp430/4900-hoc-msp430-voi-kit-msp430-launchpad-bai-2-gpio-va-clock-phan-1.html/>

1. Ngắt (Interrupt)

**Ngắt toàn cục**:  **\_BIS\_SR(GIE);           //** Cho phép ngắt toàn cục

Phải có câu lệnh này thì ngắt mới có tác dụng

3.1 Interrupt Flag Registers P1IFG, P2IFG

Khi có ngắt xảy ra bit tương ứng của chân ngắt sẽ tự động bật lên ‘1’ , và nó sẽ ko tự động xóa.

Vi xử lý sẽ căn cứ vào cờ ngắt này để nhảy vào chương trình phục vụ ngắt(ISR) . Nếu không xóa nó thì sẽ nhảy vào chương trình ngắt liên tục .  vì vậy sau mỗi lần vào ngắt phải xóa cờ ngắt đi để lần sau có thể nhảy vào được .

Bit = 0: Không có ngắt

Bit = 1: có một ngắt đang chờ

Ví dụ:

**P1IFG &=~ BIT3;**       // Xóa cờ ngắt chân P1.3

3.2 Interrupt Edge Select Registers P1IES, P2IES

PxIES : là thanh ghi cấu hình ngắt theo sườn

Bit = 0: Ngắt cạnh lên

Bit = 1: Ngắt cạnh xuống

**Ví dụ:**

**P1IES |= BIT3;**           // Ngắt theo sườn xuống

3.3 Interrupt Enable P1IE, P2IE

**PxIE :**  là thanh ghi cho phép ngắt của **Port x** (P1IE or P2IE)  gồm 8 bit tương ứng với 8 công của Port 1 : từ P1.0 đến P1.7.

Bit = 0: The interrupt is disabled.

Bit = 1: The interrupt is enabled.

**Ví dụ**

**P1IE |= BIT3;    //**cho phép ngắt chân P1.3

3.4 Chương trình phục vụ ngắt

Trong đó “PORT1\_VECTOR“ là vector ngắt của Port 1, tất cả các ngắt chân Input đếu nhảy vào vector ngắt này , tương tự với Port 2 có vector  ngắt “PORT2\_VECTOR”

 Chú ý :  phải xóa cờ ngắt sau mỗi lần nhảy vào trình phục vụ ngắt

**Cú pháp:**

#pragma vector=PORTx\_VECTOR

\_\_interrupt void Port\_x(void) {

if (PxIFG & BITy) { // Kiểm tra xem ngắt đến từ chân BUTTON hay không

// code xử lý ngắt

PxIFG &= ~BITy; // Xóa cờ ngắt

}

}

**Note**: x là Port 1 or 2, y là bit nhận tín hiệu ngắt ( 0 🡪 7)

1. TIMER và COUNTER

MSP430g2553 có 2 bộ timer:

* Timer\_A0 có 3 kênh: Channel 0 (TA0CCR0), Channel 1 (TA0CCR1) và Channel 2 (TA0CCR2).
* Timer\_A1 có 3 kênh: Channel 0 (TA1CCR0), Channel 1 (TA1CCR1) và Channel 2 (TA1CCR2).

**Timer của MSP430G2553 :**

* Timer 16bits.
* 4 chế độ hoạt động : Stop mode , Up mode , Up/Down mode , Continous Mode.
* Có chế độ đầu ra PWM (Pulse width Modulation )

**Các chế độ hoạt động của Timer**

* MC\_0 : Stop Mode ,  Timer không hoạt động
* MC\_1 : Up    Mode,   Timer đếm lặp đi lặp lại từ 0 > CCR0
* MC\_2 : Continuous,  Timer đếm lặp đi lặp lại từ 0 > 0xFFFF (65535) giá trị Max Timer
* MC\_3 : Up/Down   ,  Timer đếm lặp đi lặp lại từ 0 > CCR0 rồi CCR0

4.1 Chế độ đếm lên ( Mode up)

Ở mode này, Timer sẽ đếm liên tục từ 0 đến giá trị của register TACCR0

A black rectangular object with black text

Description automatically generated

4.2 Chế độ đếm liên tục( Continuous Mode)

Timer sẽ đếm từ 0 đến 0xFFFF và reset về 0

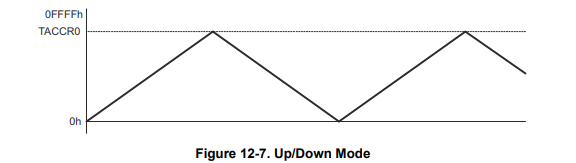
A rectangular object with black text

Description automatically generated

A diagram of a mode flow

Description automatically generated

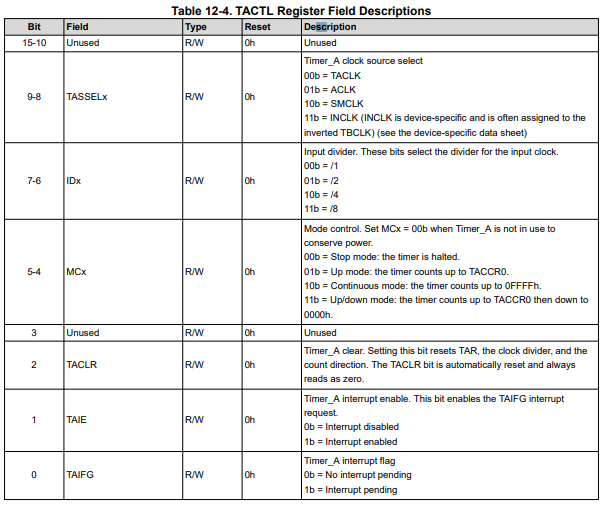
4.3 Chế độ Up/Down mode



A diagram of a computer

Description automatically generated

4.4 Thanh ghi cấu hình



A screenshot of a computer

Description automatically generated

4.5 Cách cấu hình các Mode Timer

**B1: Cấu hình xung clock MCU**

**B2: Cho phép ngắt toàn cục :** **\_BIS\_SR(GIE);**

**B3: Chọn xung clock**

|  |  |
| --- | --- |
| **TACTL ( bit 9 – 8)** | |
| TASSEL\_0 | Timer A clock source select: 0 - TACLK |
| TASSEL\_1 | Timer A clock source select: 1 - ACLK |
| TASSEL\_2 | Timer A clock source select: 2 - SMCLK |
| TASSEL\_3 | Timer A clock source select: 3 - INCLK |

**B4: Chọn bộ chia**

|  |  |
| --- | --- |
| **TACTL ( bit 7 - 6)** | |
| ID\_0 | Chia 0 |
| ID\_1 | Chia 2 |
| ID\_2 | Chia 4 |
| ID\_3 | Chia 8 |

**B5: Chọn chế độ**

|  |  |
| --- | --- |
| **TACTL ( bit 7 - 6)** | |
| MC\_0 | Stop |
| MC\_1 | Up to CCR0 |
| MC\_2 | Continous up |
| MC\_3 | Up/Down |

**Bước 5:**

Nếu ở mode Up or Mode Up/Down thì cần đặt giá trị cho **TACCR0**

**Bước 6: Cho phép ngắt Timer**

TAIE = 0: Interrupt disabled

TAIE = 1: Interrupt enabled

**Note:**

* Khi có ngắt timer **🡪** TAIFG được set lên
* Cờ ngắt tự động xóa
* TA0IV: timer overflow ( bảng 12.9 - tr393)

**Chương trình phục vụ ngắt**

#pragma vector = TIMERx\_Ay\_VECTOR

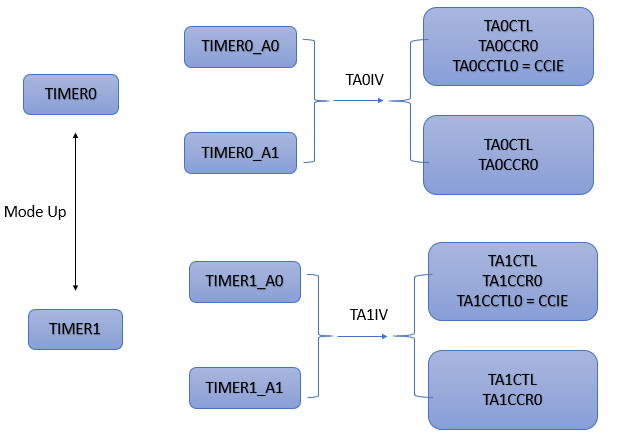
\_\_interrupt void Tên\_hàm\_ngắt(void) { // Timer over flow

If( TAxIV == 10){

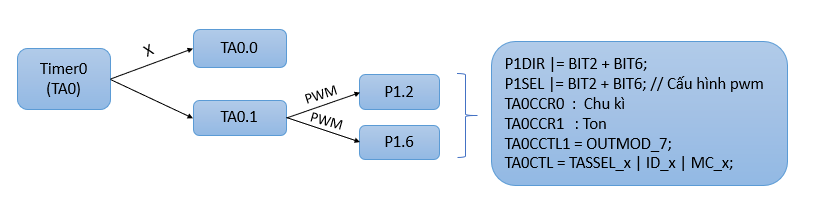
// Xử lý ngắt

{

|  |  |
| --- | --- |
| Vector ngắt | |
| TIMER0\_A0\_VECTOR | Timer0\_A CC0 |
| TIMER0\_A1\_VECTOR | Timer0\_A CC1, TA0 |
| TIMER1\_A0\_VECTOR | Timer1\_A CC0 |
| TIMER1\_A1\_VECTOR | Timer1\_A CC1, TA1 |



4.6. Mode PMW ( Compare Mode)



A diagram of a computer

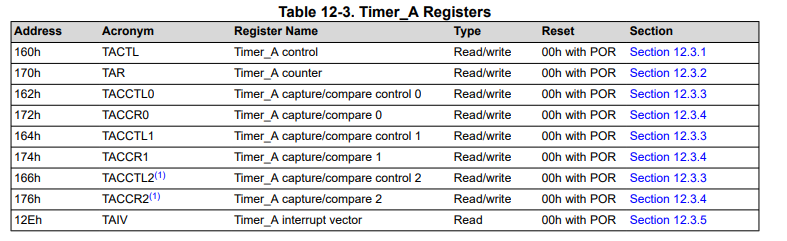
Description automatically generated

A diagram of a output example

Description automatically generated

TACCTLx Register : Thanh ghi cấu hình chức năng Capture/Compare

TAxCLT Register : Thanh ghi Cấu hình timer TAx



4.7. Input Capture

1. ADC

Bây giờ chúng ta đi tìm hiểu ADC MSP430G2553

* ADC 10 bit : giá trị số chuyển đổi  từ 0 à 1023
* Tốc độ lấy mẫu :200 ksps (200 kilo sample per second)
* Điện áp tham chiếu : có thể chọn  điện áp tham chiếu  nội 1.5V , 2.5V hoặc điện áp tham chiều bên ngoài MSP bằng  lập trình .
* 8 kênh đầu vào Analog  từ A0 à A7

A white background with black text

Description automatically generated

Có 4 kiểu chuyển đổi dữ liệu ADC

    + Single channel  : chuyển đổi đơn kênh

    + Repeat Single channel  : chuyển đổi liên tục một kênh

    + Sequence  : chuyển đổi lần lượt tất cả các kênh .

    + Repeat Sequence : chuyển đổi liên tục tất cả các kênh

5.1 Single-Channel Single-Conversion Mode

Chế độ này thực hiện chuyển đổi duy nhất trên một kênh ADC được chọn.

Sau khi quá trình chuyển đổi hoàn thành, ADC dừng và CPU có thể đọc giá trị kết quả.

**Cấu hình:**

ADC10CTL1 = INCH\_x; (Chọn kênh x)

ADC10CTL0 = SREF\_0 + ADC10ON + ADC10SHT\_3 + ENC + ADC10SC;

5.2 **Sequence-of-Channels Mode**

Chế độ này cho phép chuyển đổi liên tiếp trên một dãy các kênh ADC.

Quá trình chuyển đổi bắt đầu từ kênh thấp nhất và tiếp tục đến kênh cao nhất được chọn, sau đó dừng lại.

**Cấu hình:**

ADC10CTL1 = CONSEQ\_1 + INCH\_x; (Chọn chế độ sequence-of-channels, kênh x là kênh cuối cùng trong chuỗi)

ADC10CTL0 = SREF\_0 + ADC10ON + ADC10SHT\_3 + MSC;

5.3 Repeat-Single-Channel Mode

Chế độ này thực hiện chuyển đổi liên tục trên một kênh ADC duy nhất.

Sau khi hoàn thành một lần chuyển đổi, ADC sẽ tự động bắt đầu lần chuyển đổi tiếp theo.

**Cấu hình:**

ADC10CTL1 = CONSEQ\_2 + INCH\_x; (Chọn chế độ repeat-single-channel, kênh x)

ADC10CTL0 = SREF\_0 + ADC10ON + ADC10SHT\_3 + MSC;

5.4 Repeat-Sequence-of-Channels Mode

Chế độ này thực hiện chuyển đổi liên tục trên một dãy các kênh ADC.

Sau khi hoàn thành một chuỗi chuyển đổi, ADC sẽ tự động bắt đầu lại chuỗi từ đầu.

**Cấu hình:**

ADC10CTL1 = CONSEQ\_3 + INCH\_x; (Chọn chế độ repeat-sequence-of-channels, kênh x là kênh cuối cùng trong chuỗi)

ADC10CTL0 = SREF\_0 + ADC10ON + ADC10SHT\_3 + MSC;

1. UART

A black and white text and numbers

Description automatically generated

* Start bit : báo hiệu quá trình truyền dữ liệu
* Data Bits : dữ liệu cần giao tiếp , có thể gồm 5,6,7,8,9 bit, nhưng thông thường hay để định dạng dữ liệu 8 bits (1 byte).
* Parity bit : Bit kiểm tra chẵn lẻ , dùng khi muốn kiểm tra lỗi
* Stop bit : gồm 1 hoặc 2 bit
* IDLE : trạng thái nghỉ , phải ở mức cao

**Các bước cấu hình:**

* **B1:** Cấu hinh chân TX, RX ( P1.1, P1.2)
* **B2:** Cấu hình nguồn xung clock cho UART ( UCA0CTL1)
* **B3:** Cấu hình tốc độ Baud Rate ( UCA0BR0, UCA0BR1, UCA0MCTL)
* **B4:** Khởi tạo modul UART ( UCA0CTL1)
* **B5:** Cho phép ngắt TX, RX và ngắt toàn cục ( IE2, GIE)

**NOTE:**

* Khi sử dụng chân TX, RX giao tiếp với cảm biến thì cần gỡ 2 jump TX, RX trên board để có thể giao tiếp được
* Khi sử dụng cổng COM giao tiếp máy tính thì không cần gỡ 2 jump

1. I2C

7.1 Giới thiệu sơ lược về chuẩn giao tiếp I2C:

* I2C là một  loại bus ngoại vi được phát triển bởi hãng Philip
* Nguồn cung cấp cho giao tiếp I2C thường là 5v hoặc 3.3v
* I2C là giao tiếp được thực hiện trên hai đường dây: SCL và SDA
* SCL:Dây truyền xung clock từ master đến slave
* SDA:Dây truyền dữ liệu theo 2 chiều
* Do trên bus i2c chỉ có 2 dây mà có thể gắn kết nhiều thiết bị nên cần phân biệt các thiết bị bằng địa chỉ
* Trên bus i2c ko thể kết nối 2 thiết bị có cùng địa chỉ.

Phân loại thiết bị trên I2C

Các thiêt bị gắn trên bus i2c được chia ra làm 2 thành phần chính là master hoặc slave.

* + Thông thường trên bus i2c chúng ta sẽ có một chip vi điều khiển đóng vai trò master(đóng vai trò điều phối thông tin).
  + Trên bus i2c các cảm biến, bộ nhớ ngoài,adc,…. thường đóng vai trò là slave,trên bus i2c có thể co nhiều con slave

Chế độ hoạt động:

* Một master – Một slave.
* Một master – Nhiều slave.
* Nhiều master – Nhiều slave.

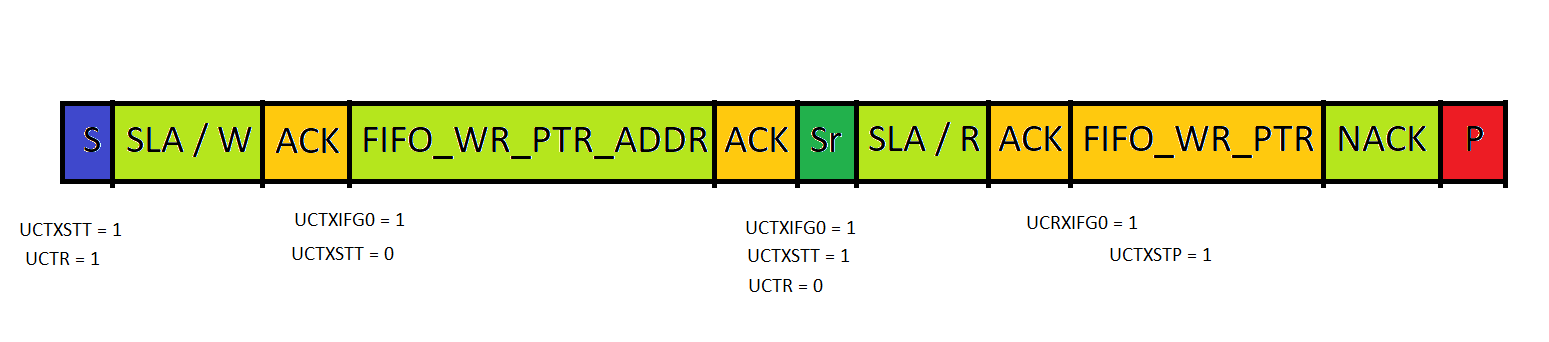
Tốc độ hoạt động:

* Chế độ chậm: 10 kbit/s.
* Chế độ cơ bản: 100 kbit/s.
* Chế độ nhanh 1: 400 kbit/s.
* Chế độ nhanh 2: 1 Mbit/s.
* Chế độ tốc độ cao: 3.4 Mbit/s.

Quá trình truyền dữ liệu trên bus i2c.

* Chế độ truyền dữ liệu:
  + Để bắt đầu truyền dữ liệu master tạo sườn xuống ở chân SDA trong khi SCL ở mức 1.
  + Sau khi SDA xuống mức 0 một khoảng thời gian ngắn thì SCL cũng xuống mức 0 và bắt đầu quá trình truyền dữ liệu.
  + Dữ liệu được truyền trên bus I2C theo từng bit  tại mỗi cạnh lên của xung Clock.
  + Để kết thúc truyền dữ liệu master tạo sườn lên ở chân SDA trong khi SCL ở mức 1





7.2 Module I2C USCI

Module I2C mình giới thiệu trong bài này thuộc loại USCI ( The universal serial communication interface) cung cấp giao tiếp truyền thông đa mode . Các module USCI hơn hẳn module USI ở khả năng lập trình mềm dẻo cho phép hoạt động với nhiều chức năng hơn.Một chip cũng có thể có nhiều module USCI cho phép hoạt động đồng thời ,được ký hiệu thành USCI\_A0 ,USCI\_Bx…

Chú ý các module:

-       USCI\_A0 hỗ trợ các module : UART ,SPI

-       USCI\_B0 hỗ trợ các module : I2C , SPI

Module I2C USCI cung cấp giao tiếp giữa MSP430 và các thiết bị dùng giao tiếp I2C theo chuẩn I2C 2 dây. Đặc điểm nổi bật của I2C USCI :

-          7-bit hoặc 10-bit địa chỉ giao tiếp

-          Giao tiếp qua 2 chân SDA và SCL

-          Chế độ giao tiếp đa Master

-          Chế độ Slaver truyền nhận

-          Hỗ trợ tốc độ giao tiếp từ 100kbps tới 400kbps

-          Thiết kế tối ưu hóa năng lượng

-          Tự động khởi động từ chế độ tiết kiệm pin LPMx khi  ở chế độ Slave

A diagram of a computer

Description automatically generated

Trong chip MSP430G2553

* Chân **SDA** là **UCB0SDA(P1.7)**
* Chân **SCL** là  **UCB0SCL(P1.6)**

7.3 Khởi tạo I2C

Module USCI reset khi set bit UCSWRST hoặc có lệnh reset hệ thông PUC(sẽ set bit UCSWRST) .Để chọn chức năng I2C bit UCMODEx=11 (UCBxCTL1) .Set bit UCSWRST sẽ gây ra :

-    Dừng giao tiếp I2C.

-    Pin SCL và SDA ở trạng thái cao trở

-    UCBxI2CSTAT,bit 6-0 clear

-    UCBxTXIE và UCBxRXIE clear

-    UCBxTXIFG và UCBxRXIFG clear

Khi muốn sử dụng I2C thì nhất thiết phải khởi tạo thông số cho các thanh ghi I2C . Các bước khi khởi tạo Module I2C:

-    Set bit UCSWRST =1 trong UCxCTL1 để chọn chức năng I2C cho Module USCI\_x

-    Chọn chức năng giao tiếp ngoại vi cho các Pin SCL và SDA ở thanh ghi PxSEL

-    Chọn địa chỉ của chip

-    Chọn tần số giao tiếp

-    Xóa bit UCSWRST 🡪0

-    Cho phép ngắt UCxRXIE hoặc/và UCxTXIE

7.4 Chế độ Master Transmitter

Các thủ tục khi viết chương trình :

* Master gửi tín hiệu start.
* Master gửi 7 bit địa chỉ thiết bị slave và bit write(bit 0) cuối cùng.( Không cần viết code vì msp430 tự động lấy địa chỉ trong thanh ghi địa chỉ Slave UCB0I2CSA ).
* Slave nhận được 8 bit ở trên thi gửi trả tin hiệu ACK(bit 0) .
* Master nhận được tín hiệu ACK thì master gửi di 8 bit địa chỉ của thanh ghi có trong con slave.
* Sau khi nhận 8 bit địa chỉ thi slave gửi trả bit ACK.
* Sau khi nhận bit ACK master gửi đi 8 bit dữ liều cần ghi.
* Sau khi nhận 8 bit dữ liệu slave sẽ gửi lại bit ACK để xác định đã nhận dữ liệu.
* Nếu muốn ghi tiếp thì master gửi tiếp 8 bit, còn nếu muốn kết thúc thì master gửi tín hiệu stop(SP).

7.5 Chế độ Master Receiver

Giải thích khung truyền :

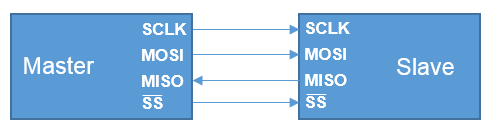
* Master gửi tín hiệu start.
* Master gửi 7 bit địa chỉ thiết bị slave và bit write(bit 0) cuối cùng .( Không cần viết code vì msp430 tự động lấy địa chỉ trong thanh ghi địa chỉ Slave UCB0I2CSA).
* Slave nhận được 8 bit ở trên thi gửi trả tin hiệu ACK (bit 0).
* Master nhận được tín hiệu ACK thì master gửi đi 8 bit địa chỉ của thanh ghi có trong slave.
* Sau khi nhận 8 bit địa chỉ thì slave gửi trả bit ACK.
* Sau khi nhận bit ACK master gửi tin hiệu restart rồi gửi lại địa chỉ thiết bị một lần nữa cùng bit read(bit 1).
* Sau khi nhận 8 bit trên thi slave gửi trả bit ACK và ngay sau đó gửi 8 bit data.
* Nếu master muốn đọc tiếp dữ liệu thì sẽ gửi tiếp bit ACK và slave sẽ gửi tiếp 8 bit luôn mà ko cần tín hiệu ACK ở trước nữa.
* Nếu ko muốn đọc dữ liệu nữa thì master sẽ gửi bit NACK(bit 1) sau đó gửi tín hiệu stop.
  + - [LINK THAM KHẢO:](https://thuanbk55.blogspot.com/2014/05/bai-8msp430basic-i2c-usci.html)

1. SPI

MSP430G2553 có 2 bộ SPI

- USCI\_A0 control SPI 1

- USCI\_B0 control SPI 2



8.1 Cấu tạo của giao tiếp SPI

**SCLK** (Serial Clock): Tín hiệu xung đồng hồ điều khiển cho việc đồng bộ truyền dữ liệu.

**MOSI** (Master Out Slave In): Tín hiệu dữ liệu được truyền từ thiết bị master tới các thiết bị slave.

**MISO** (Master In Slave Out): Tín hiệu dữ liệu được truyền từ các thiết bị slave về thiết bị master.

**SS/CS** (Slave Select/Chip Select): Tín hiệu được sử dụng để chọn thiết bị slave nào sẽ tham gia truyền thông.

8.2 Nguyên lý hoạt động cuả giao tiếp SPI

Trong giao tiếp SPI, thiết bị master điều khiển quá trình truyền dữ liệu. Nó tạo ra xung clock trên chân SCK (Serial Clock) để đồng bộ hóa truyền dữ liệu giữa master và slave. Thiết bị master cũng chịu trách nhiệm kích hoạt chân Slave Select (SS) của từng thiết bị slave để chọn thiết bị nào sẽ giao tiếp.

A diagram of a computer system

Description automatically generated

Quá trình truyền dữ liệu diễn ra theo nguyên tắc “bit-shift” (dịch bit). Thiết bị master và slave truyền và nhận dữ liệu theo cùng một thứ tự từ bit cao nhất đến bit thấp nhất. Mỗi bit được truyền qua kênh MOSI (Master Out Slave In) từ master đến slave và qua kênh MISO (Master In Slave Out) từ slave về master.

8.3 Master Mode MSP430

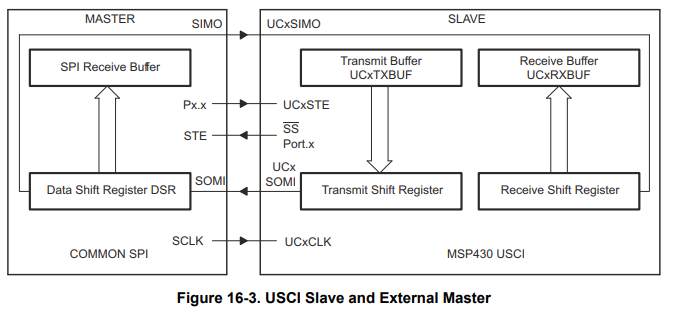
A diagram of a computer hardware system

Description automatically generated

USCI bắt đầu truyền dữ liệu khi dữ liệu được chuyển đến bộ đệm dữ liệu truyền UCxTXBUF. Dữ liệu UCxTXBUF được chuyển đến thanh ghi TX khi thanh ghi TX trống, bắt đầu truyền dữ liệu trên UCxSIMO bắt đầu bằng bit quan trọng nhất hoặc ít quan trọng nhất tùy thuộc vào cài đặt UCMSB. Dữ liệu trên UCxSOMI được chuyển vào thanh ghi dịch RX ở cạnh của xung clock. Khi nhận được ký tự, dữ liệu nhận được chuyển từ thanh ghi dịch RX rồi chuyển sang bộ đệm UCxRXBUF và cờ ngắt nhận, UCxRXIFG được đặt, cho biết thao tác RX / TX đã hoàn tất.

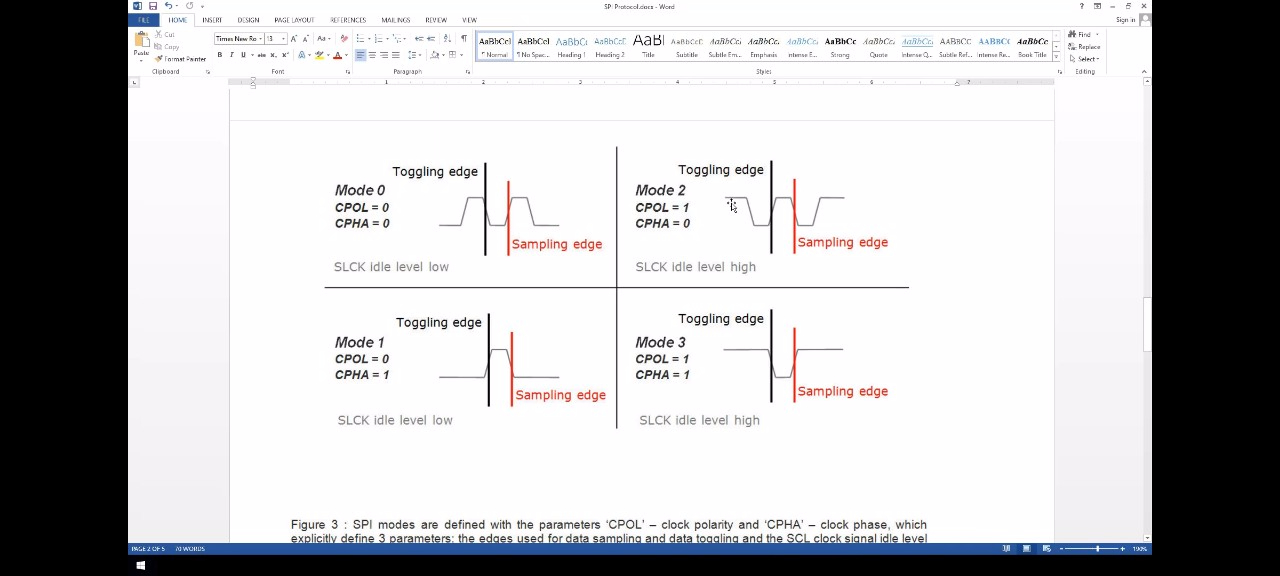
Cờ ngắt truyền UCxTXIFG được đặt, cho biết dữ liệu đã chuyển từ UCxTXBUF sang thanh ghi dịch TX và UCxTXBUF đã sẵn sàng cho dữ liệu mới. Nó không cho biết hoàn thành RX / TX.

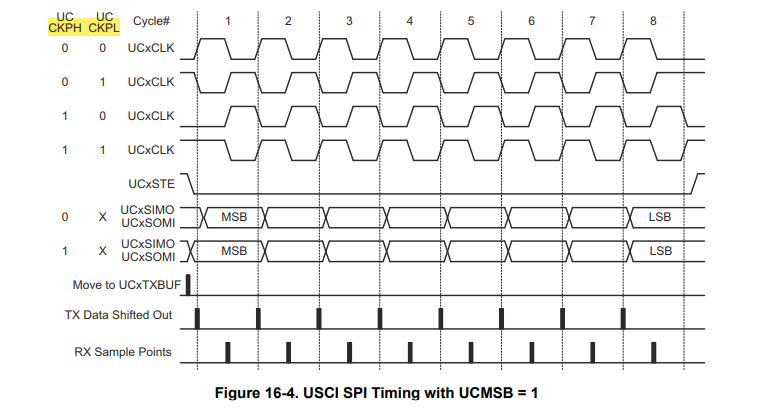
8.4 Slave Mode MSP430



UCxCLK được sử dụng làm xung clock đầu vào cho SPI và phải được cung cấp bởi Master. Tốc độ truyền dữ liệu được xác định bởi xung clock này chứ không phải bởi bộ tạo xung nhịp bên trong của Slave. Dữ liệu được ghi vào UCxTXBUF và chuyển đến thanh ghi dịch TX trước khi bắt đầu UCxCLK được truyền trên UCxSOMI. Dữ liệu trên UCxSIMO được chuyển vào thanh ghi dịch nhận ở cạnh xung clock của UCxCLK và được chuyển đến UCxRXBUF khi nhận được số bit đã đặt. Khi dữ liệu được chuyển từ thanh ghi dịch RX sang UCxRXBUF, cờ ngắt UCxRXIFG được đặt, cho biết rằng dữ liệu đã được nhận. Bit lỗi tràn, UCOE, được đặt khi dữ liệu nhận được trước đó không được đọc từ UCxRXBUF trước khi dữ liệu mới được chuyển sang UCxRXBUF.

8.4 Xung Clock PHASE and Polarity





TÀI LIỆU THAM KHẢO (Style “Cap 1”)

1. Style “TLTK”
2. Style “TLTK”