Đại Học Bách Khoa Đà Nẵng

Báo Cáo Thực Tập

ĐỀ TÀI: HIỂN THỊ LCD DÙNG MSP430F47176 THÔNG QUA GIAO DIỆN I2C.

Công ty Takemoto Đà Nẵng, Việt Nam

Người hướng dẫn: Kĩ sư Trần Văn Tâm

Sinh viên: Nguyễn Duy Trùng Dương

Lớp 09 ECE, Trung Tâm Xuất Sắc, Đại học Bách Khoa Đà Nẵng

2014

Nội Dung

1. Giới thiệu

Trong thời đại phát triển công nghệ hiện thời, việc sử dụng I2C để tương tác giữa các thiết bị với nhau sẽ thúc đẩy phát triển các ứng dụng một cách thuận lợi hơn. Các công ty lớn hiện nay như Siemens AG, Texas Instrument, Motorola đều sử dụng các thiết bị có chức năng I2C trên thị trường để thiết kế sản phẩm của mình. Hơn nữa, việc sử dụng hệ thống hai đường dây của I2C không bắt buộc phải đăng kí bản quyền sản phẩm. Từ đó, các thiết bị cá nhân có thể thực hiện nhiều chức năng và phát triển hết tiềm năng của nó thông qua việc tối ưu sử dụng các chân cắm. Ngoài ra, hiệu quả của hệ thống I2C rất linh hoạt trong vai trò chủ/tớ của từng thiết bị. I2C sở hữu các thủ tục định địa chỉ cụ thể, chế độ đa chủ đa tớ và khả năng xác nhận gửi đúng địa chỉ nên hoạt động khá chính xác. Hơn nữa, chế độ gửi chính xác từng đơn vị dữ liệu trên từng xung chu kì khiến ta dễ kiểm soát thông tin được gửi chính xác hơn.

2. Tóm tắt

Trong đồ án này, chúng tôi sẽ tiến hành hiển thị thời gian theo (giờ /phút/ giây) trên LCD đa điểm. Thông qua sự tích hợp các thiết bị trên hệ thống I2C, vi điều khiển chính chỉ sử dụng 2 chân là chân truyền dữ liệu (SDA) và chân xung cấp xung đồng hộ (SCL). Vi điều khiển trong đồ án này sẽ dùng là chip của hãng Texas Instrument MSP430-F47176. LCD được sử dụng sẽ tích hợp với chip điều khiển LCD là PCF-8578. Chip PCF-8578 cũng sẽ sử dụng 2 đường truyền SDA và SCL, và nó đóng vai trò là tớ trong hệ thống. Thêm vào đó, vi điều khiển (chủ) sẽ lấy địa chỉ của PCF-8578 để tiến hành việc gửi các câu lệnh và gửi/nhận dữ liệu.

3. Nội dung và kế hoạch

Để tiến hành đồ án hiển thị LCD đa điểm dựa trên hệ thống I2C, trước tiên lí thuyết sẽ được nghiên cứu theo từng thiết bị và theo từng chi tiết chức năng. Sau khi tìm hiểu rõ về hoạt tính của từng thiết bị, công đoạn lập trình và nạp vào thiết bị sẽ được tiến hành theo từng giai đoạn từ nhỏ đến lớn. Cuối cùng sẽ thống kê kết quả thu được và đồng thời bắt đầu tiến hành báo cáo.

1.1 Tìm hiểu lí thuyết

1.1.1 Tìm hiểu về MSP430-F47176

Những điều cần biết về MSP430-F47176 gồm có:

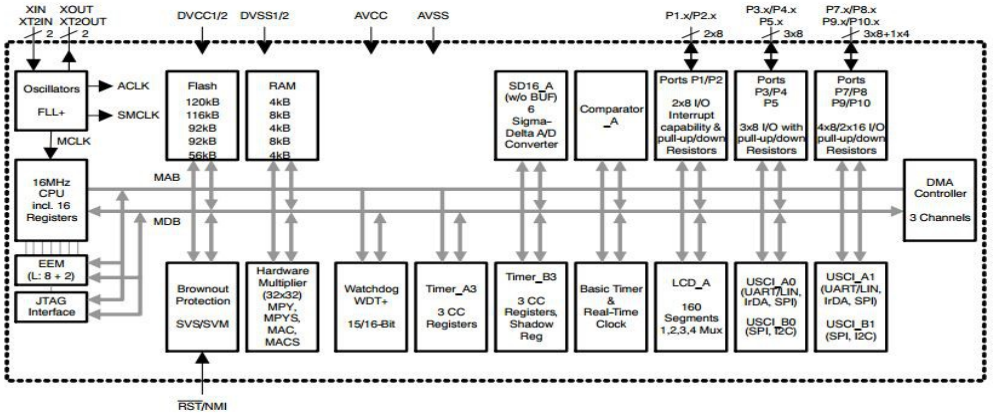
a) Tổng quan của vi điều khiển

b) CPU của vi điều khiển như: Tổ chức bộ nhớ, chức năng các chân, các thanh ghi đặc biệt…

c) Bộ định thời Timer_A
d) USCI sử dụng chế độ I2C

Phần chính tìm hiểu được

a) Sơ đồ khối



Mô hình câu trúc bộ nhớ:

Vùng bộ nhớ		Address
Bộ nhớ Interrupt vector Bộ nhớ code	Dung lượng Flash Flash	92kB 0xFFFF-0xFFC0 0x19FFF-0x03100
RAM (Tổng cộng) Mở rộng Nhân bản	Dung lượng Dung lượng Dung lượng	0x30FF-0x1100 (8kB) 0x30FF-0x1900 (6kB) 0x18FF-0x1100 (2kB)
Bộ nhớ thông tin	Dung lượng flash	0x10FF-0x1000(256Byte)
Bộ nhớ khởi động	Dung lượng flash	0x0FFF-0x0C00(2kB)
RAM (phần nhân bản từ địa chỉ 0x018FF-0x01100	Dung lượng	0x09FF-0x0200(2kB)
Thiết bị ngoại vi	16 bits 8 bits 8 bits SFR (Thanh ghi đặc biệt)	0x01FF-0x0100 0x0FF-0x010 0x0F-0x00

Hai chân SCL và SDA cho vi điều khiển:

P2.1/UCB1SIMO/UCB1SDA (Input/Output): Chân cắm tổng quát cho I/O / Đầu vào cho Slave hoặc đầu ra cho Master của USCI_B1 trong giao diện SPI (giao diện 4 dây) / Xung dữ liệu SDA của I2C trong chế độ I2C.

P2.2/UCB1SOMI/UCB1SCL (Input/Output): Chân cắm tổng quát cho I/O / Đầu ra của Slave và đầu vào của Master của USCI_B1 trong chế độ SPI (giao diện 4 dây) / Xung clock SCL của I2C trong chế độ I2C.

Các thanh ghi từ R0 đến R4 là những thanh ghi có chức năng cá nhân như:

- R0: Program Counter, bộ đếm chương trình, dùng để theo dõi địa chỉ của tập lệnh.
- R1: Stack Pointer, Con trỏ ngăn xếp, dùng để hỗ trợ PC trong việc thực hiện các chương trình con để sau khi hoàn thành quay lại vị trí cũ.
- R2: Status Register là thanh ghi chưa thông tin về trạng thái CPU. Thường liên quan đến interrupt và các cờ trạng thái báo hiệu kết quả sau 1 phép toán C, Z, N và V.
- R3: Constant Generator, thường cung cấp cho ta 6 biến số được sử dụng liên tục gần đây nhất để tiếp cận nhanh hơn khi cần.
- SFR: special function registers (SFR), được định vị ở ở vùng địa chỉ thấp nhất (ví dụ như vùng chứa địa chỉ 0x0000 đến 0x000F chẳng hạn trong phạm vi địa chỉ 16 bits) Được cơ cấu bởi thanh ghi sử dụng chế độ byte (là chế độ dịch chuyển từng đơn vị byte dữ liệu giữa CPU và thiết bị ngoại vi). SFR được tiếp cận bởi tập lệnh byte.

1. Thanh ghi interrupt enable 1 ở địa chỉ 0x00 dùng để kích hoạt chế độ interrupt cho Watchdog Timer/ Kích hoạt interrupt lỗi bộ dao động

2. Thanh ghi interrupt enable 2 ở địa chỉ 0x01:

3. Thanh ghi interrupt flag 1 ở địa chỉ 0x02: cờ hiệu báo lỗi

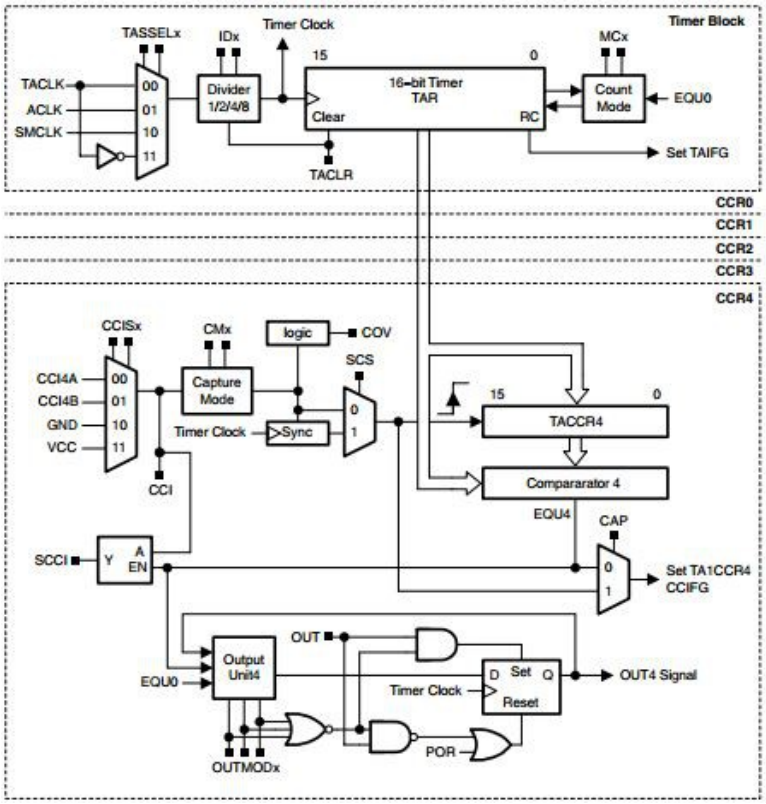
c) Bộ định thời Timer_A0

Bộ định thời có thanh ghi 16 bit TAR (Timer A Counter) sẽ được dùng hoặc đếm lên hoặc đếm xuống theo kích sườn lên của xung. Ngoài ra bộ Timer còn thực hiện ngắt theo nguyên lí tràn. Chức năng của TAR gồm có read/write.

(Lưu ý: khi muốn thay đổi TAR thì phải ngưng hoạt động trước)

Chúng ta có 4 lựa chọn trong xung đồng hồ của Timer thông qua phần bit TASSELx. Hơn thế nữa, ta có thể chia nhỏ xung ra theo mức 2, 4, 8 hay giữ nguyên tùy vào phần bit IDx.

Sơ đồ khối



TACTL, Timer_A Control Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw–(0)	rw–(0)	rw–(0)	rw–(0)	rw–(0)	w–(0)	rw–(0)	rw–(0)

2 phần bit trên được quản lí trong thanh ghi 16 bit TACTL:

TASSELx từ bit9-bit8:

00: TACLK 01: ACLK

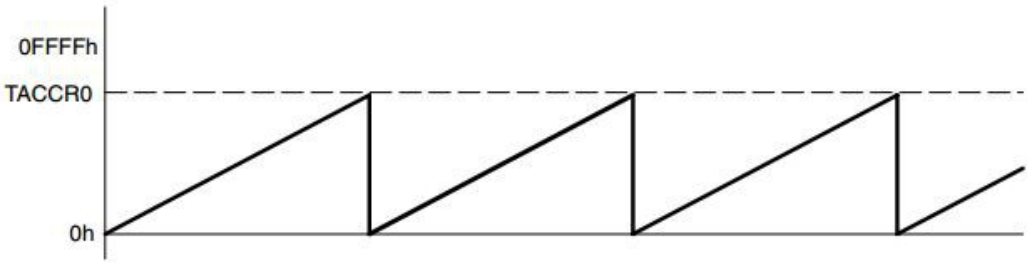
10: SMCLK 11: INCLK

IDx từ bit7-bit6:

00: giữ nguyên 01: chia cho 2

10: chia cho 4 11: chia cho 8.

Trong đồ án này ta sẽ sử dụng chế độ count UP cho timer. Trong chế độ này Timer sẽ so sánh TAR với thanh ghi TACCR0 (có giá trị nằm từ 0 đến 0xFFFF). Ứng vỗi mỗi xung đồng hồ sau khi chọn qua TASSELx và IDx, TAR sẽ đếm lên 1 xung bắt đầu từ giá trị 0 cho đến khi bằng TACCR0. Lúc đó, cờ ngắt CCIFG của TACCR0 sẽ được bật. Cờ ngắt timer TAIFG sẽ được bật sau khi TAR bằng TACCR0 và quay lại mức 0.



Lưu ý: việc điều chỉnh lại tần số(chu kì) của xung timer có thể thực hiện được bằng cách thay đổi giá trị của TACCR0 trong khi Timer đang chạy. Như vậy nếu giá trị mới không thấp hơn giá trị cũ thì TAR sẽ tiếp tục đến lên tiếp đến khi bằng thì thôi. Ngược lại thì TAR sẽ quay lại giá trị zero ngay lập tức hoặc trễ 1 chu kì xung đồng hồ rồi quay lại mức 0.

1.1.2 Tìm hiểu về chế độ I2C

Những điều cần được tìm hiểu về hệ thống I2C gồm có:

a) Giới thiệu tổng quan.

b) Cách thức hoạt động cụ thể.

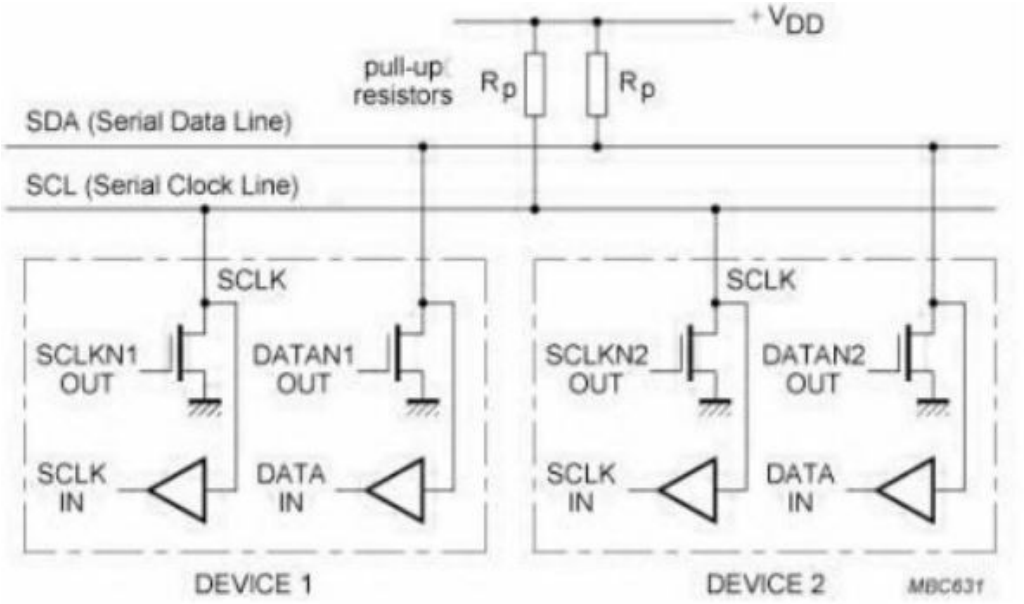
c) Phương án sử dụng với MSP430-F47176.

Thời gian dự kiến trong phần này sẽ là 2 ngày là ngày 18 và ngày 19/11/2013.

Phần chính tìm hiểu được

a) Giới thiệu tổng quan

I2C hay là ***Inter-Integrated-Circuit*** là bus để thực hiện giao tiếp giữa IC này với IC khác. Trong module này sử dụng 2 đường dây truyền chính đó là đường xung nhịp đồng hồ (SCL) và một đường dữ liệu (SDA).



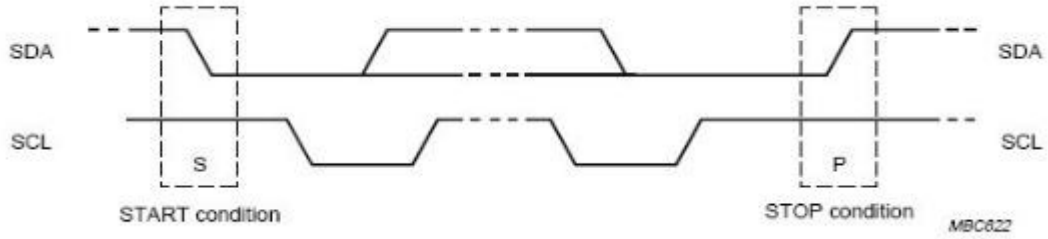
b) Cách thức hoạt động

- Hoạt động:

Đường truyền xung đồng hồ SCL là đường 1 hướng còn đường truyền dữ liệu SDA lại 2 chiều. Như vậy khi kết nối với thiết bị ngoại vi thì đường ra/vào dữ liệu sẽ nối với SDA của I2C và đường vào xung đồng hồ của ngoại vi sẽ nối với SCL.

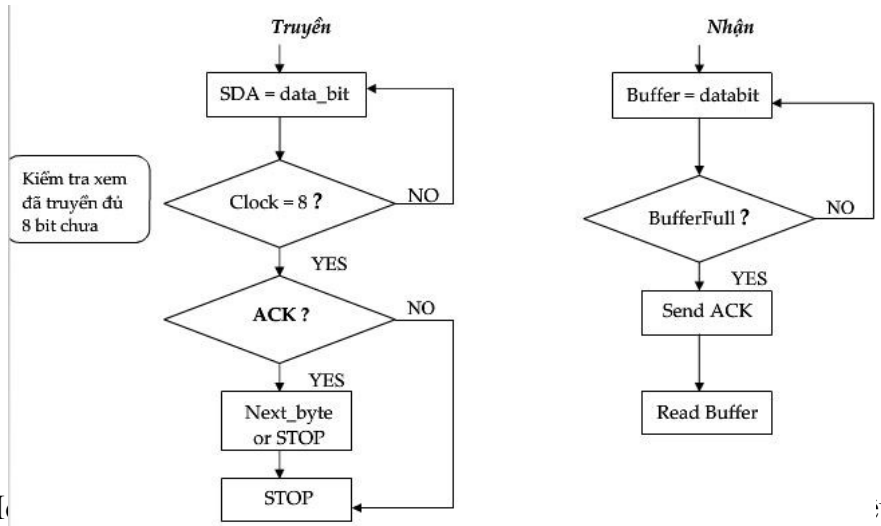
2 đường dây SDA và SCL của I2C sẽ được đấu với 1 điện trở kéo nối với đầu (+) của nguồn DC và thường có giá trị là 1k-4.7kOhm. Dù các thiết bị ngoại vi được sử dụng chung 1 đường dây nhưng sẽ không xảy ra tình huống nhầm lẫn tín hiệu giữa các thiết bị. Vì trong suốt cả quá trình, mỗi thiết bị được định hình với 1 địa chỉ duy nhất với một quan hệ master/slave. Và hoạt động truyền hay nhận của từng thiết bị phụ thuộc vào tính chất của nó là master hay slave. Trong quá trình giao tiếp, master sẽ là thiết bị phát ra xung đồng hồ và quản lí địa chỉ cũng như truyền xung đồng hồ và tín hiệu đến các thiết bị slave.

Trong giao tiếp của I2C, sẽ có 2 điều kiện trạng thái đó là START(bắt đầu giao tiếp/ I2C bus ở trạng thái ***busy***) và STOP(ngừng giao tiếp/ I2C bus ở trạng thái ***free***). Ban đầu, SDA=SCL =1, điều kiện của START đó là SCL =1 được giữ nguyên và SDA chuyển từ trạng thái "1" sang "0". Còn điều kiện STOP là SCL =1 được giữa nguyên và SDA chuyển từ "0" sang "1". Hai điều kiện này sẽ được phát ra bởi Master. Khi START xảy ra rồi và nếu 1 tín hiệu START nữa xảy ra thì vẫn tiếp tục trạng thái ***busy***.



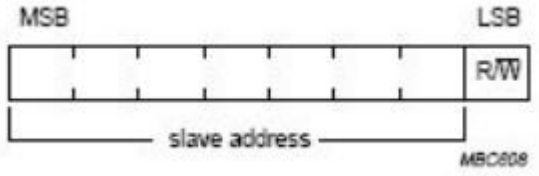
Dữ liệu được truyền trên bus I2C theo từng bit, bit dữ liệu được truyền đi tại mỗi sườn dương của SCL, quá trình thay đổi bit dữ liệu xay ra khi SCL=0.
Số lượng byte được truyền trong một lần là không hạn chế và mỗi byte có độ dài là 8 bits. Sau mỗi byte truyền là 1 bit ACK(Ackknowledge) để báo hiệu là đã nhận được dữ liệu. Trong mỗi byte, MSB sẽ được truyền trước và lần lượt các bit tiếp theo.

Sau 8 chu kì xung của SCL, I2C đã truyền đi được 8 bits dữ liệu và bên SDA sẽ tạo một xung thấp "0" tại chu kì thứ 9. Xung này chính là ACK để xác nhận là đã truyền hết 1 byte và báo cho thiết bị nhận để tiếp tục hoặc là kết thúc.



M u
khiển được 128 thiết bị. Khi thiết bị chủ muốn giao tiếp với ngoại vi nào

trên I2C, nó sửi 7 bit địa chỉ của thiết bị đó sau ngay xung START. Vậy Byte đầu tiên được gửi gồm 7 bit địa chỉ và bít thứ 8 điều khiển hướng R/W (đọc hoặc viết lên).



Mỗi thiết bị ngoại vị sẽ có địa chỉ nhất định do nhà sản xuất. Hơn nữa nó có thể là cố định hoặc thay đổi. Trong byte đầu tiên bit định hướng R/W sẽ định cho Master truyền dữ liệu khi R/W=0 còn ngoại vi truyền tới máy chủ khi R/W=1.

c) Điểu khiển I2C trong MSP430F47176:

Trong này LCD được kết nối với bộ I2C và bộ I2C này sẽ cắm 2 dây liên kết với vi điều khiển chính. Khi thiết kế phần cứng, vi điều khiển sẽ được kết nối với hệ thống I2C theo 2 chân đó là:

1. *Chân sô 81*: P2.2/UCB1SOMI/UCB1SCL: đầu ra cho slave/đầu vào cho Master cho SCL.
2. *Chân số 82*: P2.1/UCB1SIMO/UCB1SDA: đầu vào cho slave/ đầu ra cho master cho SDA.

1.1.3 Tìm hiểu về PCF-8578

Trong phần này, ta sẽ nghiên cứu về:
+ Sơ đồ khối của chip.
+ Các chân của chip.
+ Các kiểu hiển thị.
+ Display RAM.
+ Bộ tập lệnh sắp đặt chế độ hiển thị.
+ Cách cập nhập RAM.

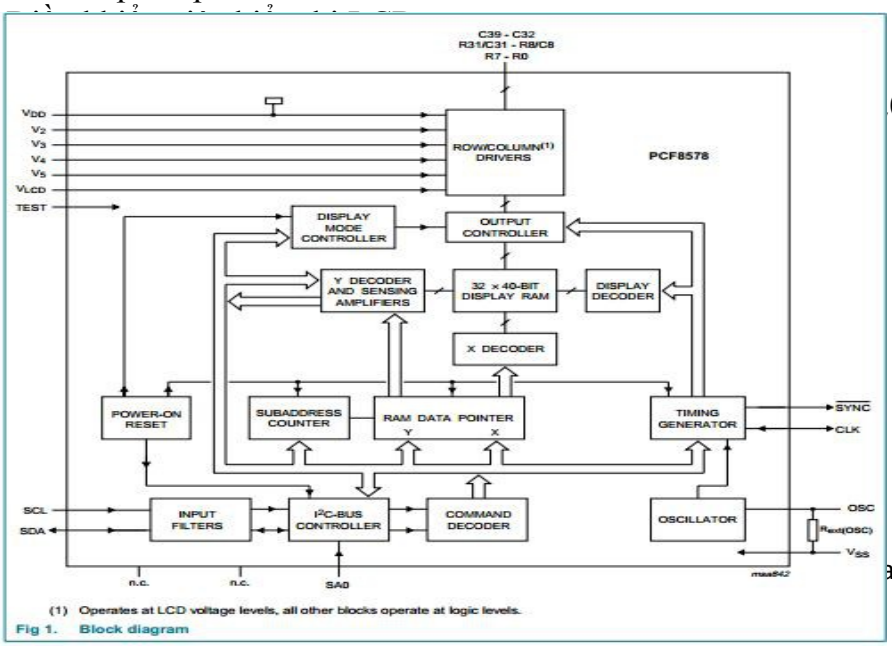


Fig 1. Block diagram

013.

Đặc điểm chân cắm:

Tên chân	Thứ tự pin	Mô tả
SDA	1	Cổng dữ liệu I2C
SCL	2	Cổng xung đồng hồ I2C
!SYNC	3	Đầu ra tầng kết nối đồng bộ
CLK	4	In/out của đồng hồ ở ngoài
Vss	5	MAX
TEST	6	Pin test
SA0	7	Đầu vào địa chỉ Slave của I2C
OSC	8	Đầu vào của bộ dao động
Vdd	9	Nguồn (+)
V2 tới V5	10 – 13	Đầu vào áp cho LCD
VLCD	14	Nguồn cấp cho LCD
Không cắm	15,16	
C39 – C32	17 – 2	Đầu ra cho cột LCD
R32/C31 – R8/C8	25 – 48	Đầu ra cho hàng/cột LCD
R7 – R0	49 - 56	Đầu ra cho hàng LCD

PCF8578 hoạt động theo 3 chế độ sau:

- Chế độ 1 mình, driver của hàng và cột cho hiển thị nhỏ (chế độ kết hợp mixed).

- Chế độ hàng và cột kết nối với PCF8579s (chế độ mixed).
- Chế độ chỉ hàng kết hợp PCF8579s (chế độ mixed và chế độ theo hàng).

Application	Multiplex rate	Mixed mode		Row mode		Typical applications
		Rows	Columns	Rows	Columns	
stand alone	1:8	8	32	-	-	small digital or alphanumeric displays
	1:16	16	24	-	-	
	1:24	24	16	-	-	
	1:32	32	8	-	-	
with PCF8579	1:8	8[1]	632[1]	8 × 4[2]	640[2]	alphanumeric displays and dot matrix graphic displays
	1:16	16[1]	624[1]	16 × 2[2]	640[2]	
	1:24	24[1]	616[1]	24[2]	640[2]	
	1:32	32[1]	608[1]	32[2]	640[2]	

[1] Using 15 PCF8579s.

[2] Using 16 PCF8579s.

Thiết bị PCF-8578 chứa 32x40 bit RAM tĩnh để chứa các dữ liệu hiển thị. Bộ RAM được chia làm 4 mục 40 bytes (tương đương 4 x 8 x 40 bits). Trong quá trình truy cập RAM, dữ liệu được truyền tới hoặc truyền từ RAM qua dây bus I2C. 8 cột dữ liệu đầu tiên (bit0-bit7) sẽ không dùng để hiển thị nhưng có thể trữ được dữ liệu tổng quát và cung cấp khả năng tương thích với PCF8579. Cơ chế định địa chỉ cho *Display RAM* được thực hiện bằng con trỏ Data. Từng đơn vị byte dữ liệu hoặc dãy byte dữ liệu sẽ được viết hoặc đọc trên RAM. Điều này được kiểm soát bởi các lệnh command gửi tới bus của I2C.

Bộ giải mã lệnh Command:
Bộ giải mã này dùng để phân loại các tập lệnh nhận được ở bus I2C. Có 5 dạng lệnh chính đều là 1 byte và có bit thứ 7 là được gọi là bit C hay là bit tiếp tục (continue). Nếu C bằng 0 thì xác nhận đây là byte điều khiển cuối cùng và byte tiếp theo là dữ liệu hiển thị. Ngược lại, nếu C bằng 1 thì byte tiếp theo vẫn là có thêm lệnh nữa.

Lênh Set-Mode
Phần này gồm có:

- Chế độ hiển thị: chế độ theo hàng, chế độ kiểu kết hợp.
- Lựa chọn trạng thái hiển thị: trống, bình thường, tất cả các đoạn đều bật và video ngược.

- Chọn chế độ chạy của LCD: 1:8 mux (8 hàng), 1:16 mux, 1:24 mux và 1:32mux.

Table 11. Set-mode - command bit description

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6, 5	-	10	fixed value
4	T		display mode
		0	row mode
		1	mixed mode
3, 2	E[1:0]		display status
		00	blank
		01	normal
		10	all segments on
		11	inverse video
1, 0	M[1:0]		LCD drive mode
		01	1:8 MUX (8 rows)
		10	1:16 MUX (16 rows)
		11	1:24 MUX (24 rows)
		00	1:32 MUX (32 rows)

Table 12. Set-start-bank - command bit description

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6 to 2	-	11111	fixed value
1, 0	B[1:0]		start bank pointer (see Figure 20)[1]
		00	bank 0
		01	bank 1
		10	bank 2
		11	bank 3

[1] Useful for scrolling, pseudo-motion and background preparation of new display content.

4 bit đầu dùng đê cho bộ đếm địa chỉ phụ (Subaddress Counter) cho việc xác định 16 địa chỉ phần cứng phụ (Subaddress Hardware).

Table 13. Device-select - command bit description

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6 to 4	-	110	fixed value
3 to 0	A[3:0]	0 to 15[1]	hardware subaddress; 4 bit binary value; transferred to the subaddress counter to define one of sixteen hardware subaddresses

[1] Values shown in decimal.

Lệnh RAM-Access

- Lựa chọn chế độ truy cập RAM, xác định hành vi tự tăng của địa chỉ trong truy cập RAM: theo kí tự, nữa đồ thị, toàn đồ thị và không sử dụng được,
- Lựa chọn địa chỉ hàng của RAM: xác định 1 trong 4 hàng của RAM

Table 14. RAM-access - command bit description

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6 to 4	-	111	fixed value
3, 2	G[1:0]		RAM access mode; defines the auto-increment behavior of the address for RAM access (see Figure 18)
		00	character
		01	half-graphic
		10	full-graphic
		11	not allowed[1]
1, 0	Y[1:0]	0 to 3[2]	RAM row address; two bits of immediate data, transferred to the Y-address pointer to define one of four display RAM rows (see Figure 17)

[1] See operation code for set-start-bank in Table 12.

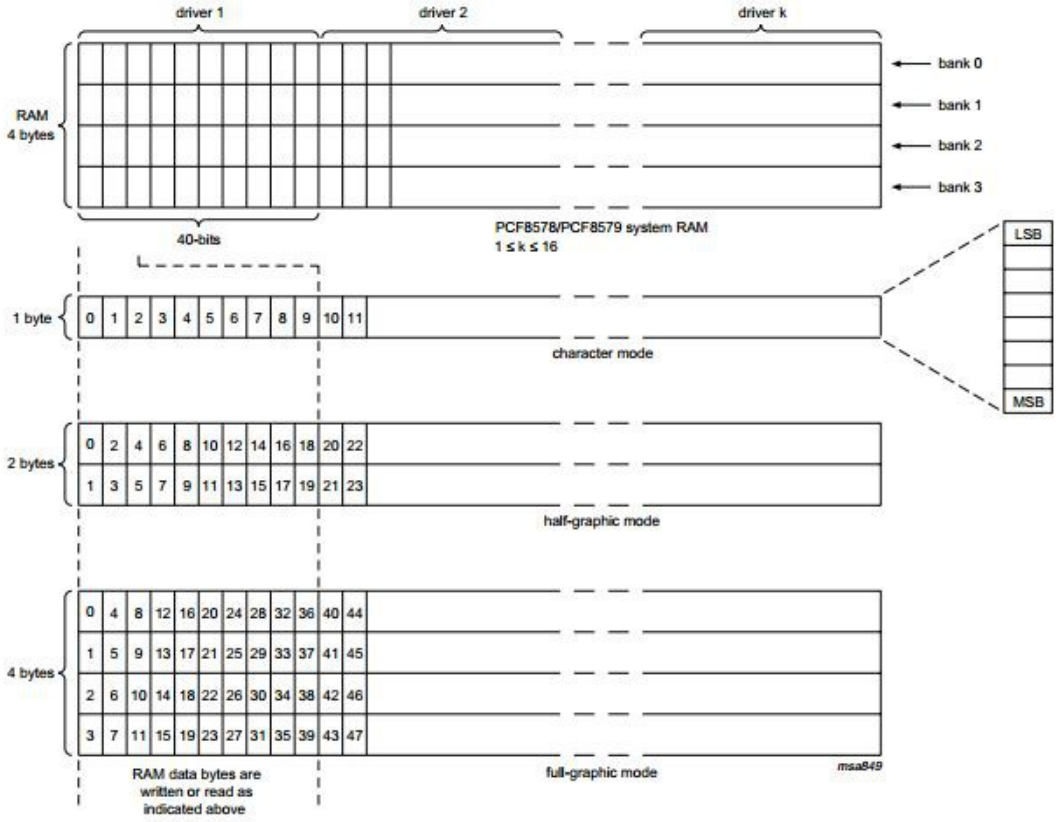
[2] Values shown in decimal.

Lệnh Load-X-Address

Dùng để xác định 1 trong 40 ngăn theo cột của Display RAM

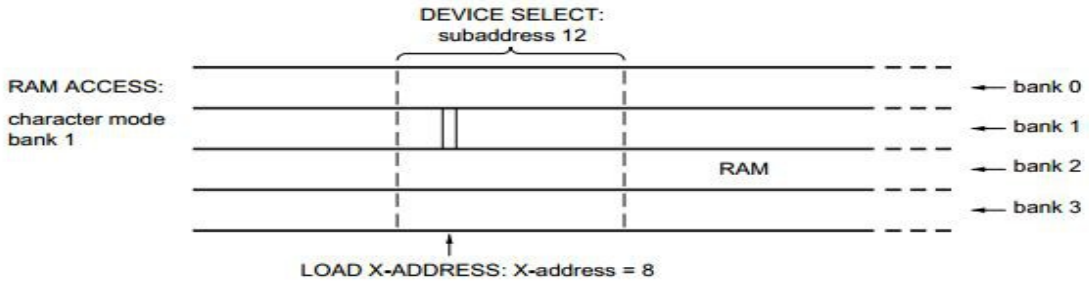
Table 15. Load-X-address - command bit description

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6	-	0	fixed value

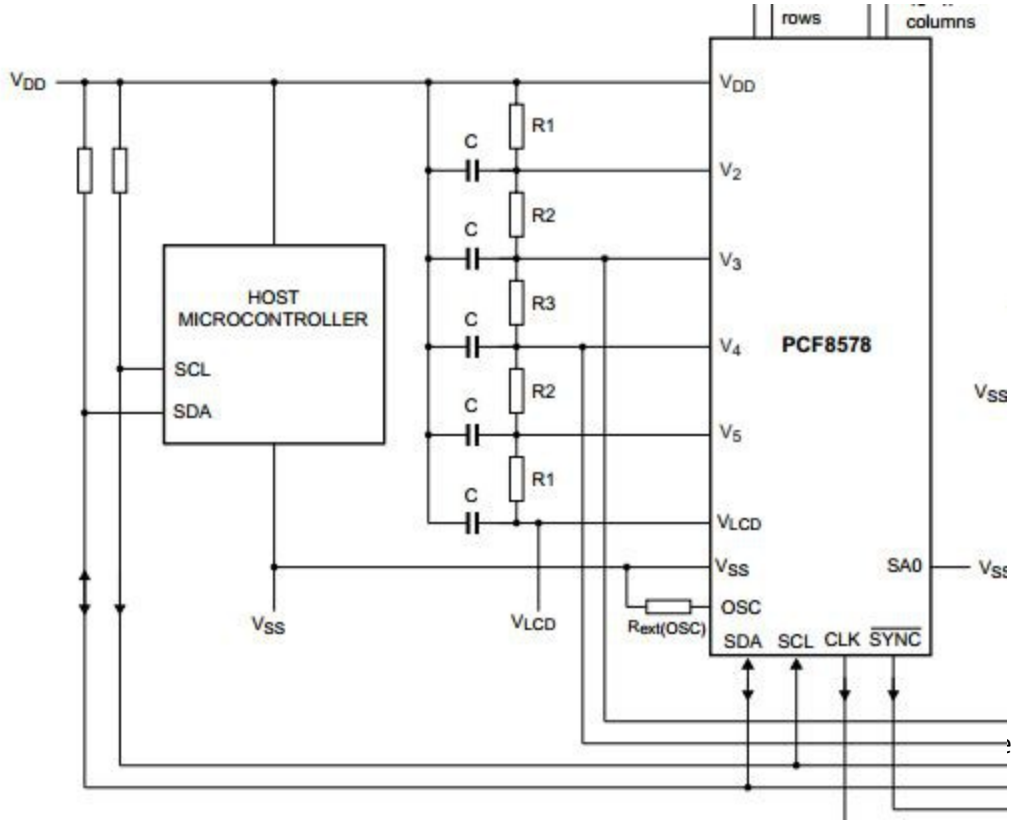


Để lưu trữ dữ liệu RAM, người dùng phải chỉ cụ thể vị trí mà byte đầu tiên sẽ được tải, gồm có:

- Xác định subaddress trong RAM
- Xác định vị trí X trong các cột
- Xác định vị trí ngân Y

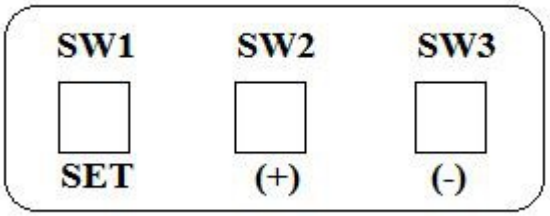


MASTER coi PCF 8578 là SLAVE:



1.1.4 Thao tác nút bấm

Phần nút bấm sẽ có chức năng để điều chỉnh thời gian theo ý muốn của người sử dụng. Tổng số nút bấm gồm có: nút SET, nút (+) và nút (-).



Chức năng của từng nút:

- Nút SET(chân P2.7): Nút này có mục đích như một nút điều khiển 5 chế độ của đồng hồ. Khi đồng hồ đang chạy bình thường, nếu ta nhấn nút SET:

  + Lần thứ 1: Đồng hồ đang chạy sẽ ngưng. (mode 2)

  + Lần thứ 2: Phần giờ sẽ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị từ các nút (+) và (-). (mode 3)

  + Lần thứ 3: Phần phút sẽ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị từ các nút (+) và (-). (mode 4)

  + Lần thứ 4: Phần giây sẽ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị từ các nút (+) và (-). (mode 5)

  + Lần thứ 5: Đồng hồ chạy lại bình thường (mode 1)

  (Theo sơ đồ hình 3.2.1.2.2 ở phần sau trên trang 20)

- Nút (+) (chân P3.4): Nút này để tăng giá trị của vị trí đang nhấp nháy. Chỉ có tác dụng khi đồng hồ đang nhấp nháy một trong 3 vị trí (giờ / phút / giây).

  + Khi bấm 1 lần rồi thả: giá trị nhấp nháy tại vị trí hiện thời tăng lên 1 đơn vị.

  + Khi bấm giữ hơn nửa giây: giá trị tại vị trí nhấp nháy sẽ tăng 1 đơn vị mỗi nửa giây.

- Nút (-) (chân P3.3): Nút này để giảm giá trị của vị trí đang nhấp nháy. Chỉ có tác dụng khi đồng hồ đang nhấp nháy một trong 3 vị trí (giờ / phút / giây).

  + Khi bấm 1 lần rồi thả: giá trị nhấp nháy tại vị trí hiện thời giảm xuống 1 đơn vị.

  + Khi bấm giữ hơn nửa giây: giá trị tại vị trí nhấp nháy sẽ giảm 1 đơn vị mỗi nửa giây.

Một số tiêu chuẩn về nút bấm: Khi có sự nhấn nút xảy ra đi kèm theo sau là thả nút thì hệ thống sẽ tính là một lần bấm nút. Nếu việc thực hiện nút bấm như vậy mà hệ thống nhận dạng ra trên 1 lần nhấn thì chứng tỏ thao tác nút bấm bị lỗi và chưa hoàn thiện.

1.2 Thực hành

- ***Giới thiệu đề tài thực hành***

1.2.1 Nội dung chính

Đề tài này nhằm mục đích sử dụng vi điều khiển MSP430-F47176 để hiển thị thời gian (giờ / phút / giây) lên màn hình LCD đa điểm. Điều chính ở đây là sử dụng hệ thống I2C 2 đường truyền dẫn để giảm bớt chân cắm cho vi điều khiển. LCD sẽ được tích hợp với thiết bị trung gian là PCF-8578. Trong đó vi điều khiển là thiết bị chủ (master) và PCF-8578 là thiết bị tớ (slave).

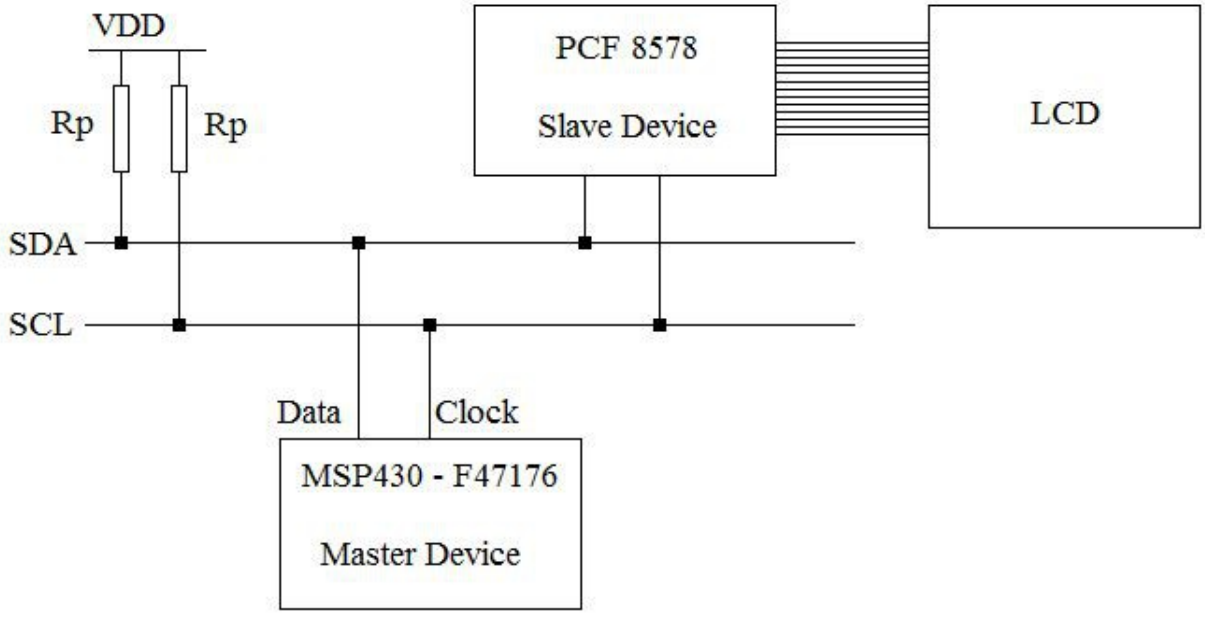
Trong phần cứng, đã được thiết kế sẵn, thì vi điều khiển sẽ sử dụng thạch anh với tần số là 14.7456 MHz. Vì vậy, thiết bị chủ, MSP430-F47176, ở đây sẽ sử dụng xung đồng hồ này để khởi tạo chế độ I2C nằm trong chức năng USCI cho việc truyền dữ liệu vào thiết bị tớ, PCF-8578.

Xung đồng hồ cho SCL của I2C sẽ được lấy từ xung đồng hồ của thạch anh và ước tính là khoảng 99.63 KHz (~100 KHz).

Nhiệm vụ của thiết bị chủ là xác định địa chỉ của thiết bị tớ và thiết lập các chế độ để điều khiển màn hình LCD. Các chế độ thiết lập này chính là bộ tập lệnh sắp đặt chế độ hiển thị của PCF-8578. Thiết bị chủ sẽ xử lí bên trong, sử dụng bộ Timer_A0 để tạo các byte để thiết bị chủ gửi cho thiết bị tớ. Các byte dữ liệu này sẽ giúp LCD hiển thị và cập nhập thời gian lên màn hình theo chu kì chính xác là 1 giây.

1.2.2 Sơ đồ minh họa

Sơ đồ I2C sẽ gồm có thiết bị chủ là vi điều khiển, thiết bị tớ là chip PCF-8578 và LCD. Hai đường truyền SDA và SCL sẽ được nối với trở kéo lên Pull Up Resistor vào nguồn Vdd.

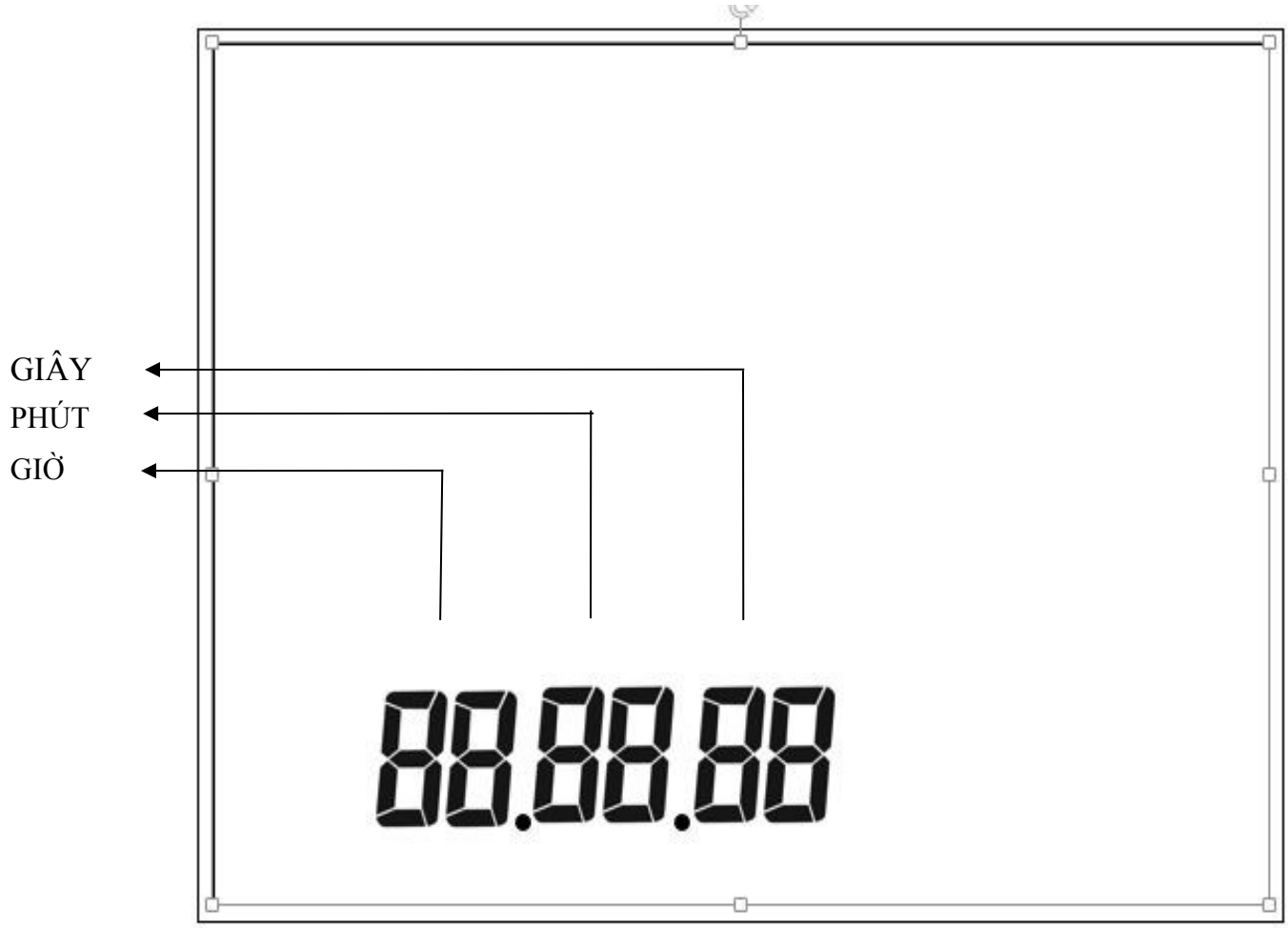


Trong đó SCL sẽ là đường xung đồng hồ và SDA sẽ là đường truyền của tín hiệu. SCL sẽ đi theo xung một chiều còn SDA sẽ đi theo xung hai chiều. Thiết bị chủ vi điều khiển sẽ gửi các tín hiệu điều khiển đến thiết bị tớ PCF – 8578.

Thiết bị tớ PCF – 8578 sẽ đọc các tín hiệu từ vi điều khiển và thực hiện thao tác hiển thị trên mà hình LCD. Các tín hiệu hiển thị sẽ gồm có 5 tín hiệu set-up các chế độ và sau đó là các tín hiệu thông tin chi tiết về các kí tự được hiển thị trên màn hình.

Kết quả hiển thị dự tính trên màn hình LCD sẽ là như hình dưới đây theo thứ tự sau:

1. Giờ
2. Phút
3. Giây



Hình 3.2.1.2.2 Dự kiến của đồng hồ LCD giờ / phút / giây

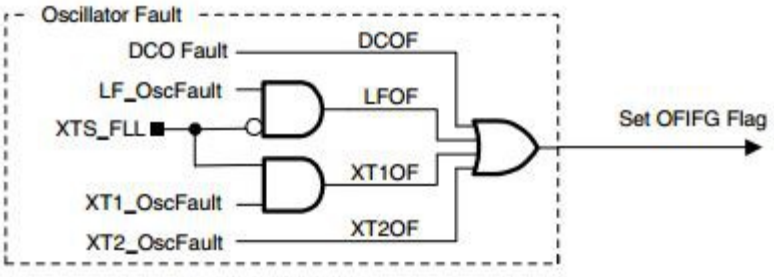
- ***Phương pháp tiến hành và nội dung chương trình thực hiện***

1.2.3 Các bước thực hành chính

(a) Khởi tạo xung đồng hồ từ thạch anh cho vi điều khiển

Trong phần cứng của toàn hệ thống, chúng ta sẽ sử dụng thạch anh là nguồn cấp xung chính thông qua cổng XT2. Để kích hoạt bộ phần này, trước hết ta phải tắt cờ ngắt hoạt động của XT1 và các thanh ghi trạng thái liên quan tới XT1. Thêm vào đó, ta cũng cần ngắt bộ điều biến DCO.

Một trong những cờ có thể ảnh hưởng đến việc khởi tạo XT2 là OFIFG. Đây là cờ báo lỗi khi 1 trong các bộ dao động tại XT1(tần số cao), XT2, LFOF(tần số thấp XT1) và DCO có vấn đề xảy ra.



Nếu cờ OFIFG còn bật thì không thể nào sử dụng được XT2. Tuy đã tắt XT1 và không sử dụng DCO, nhưng trong quá trình khởi tạo thì vẫn chưa sử dụng được XT2 (Ta cần dùng thạch anh để tạo ra xung chính xác cho đồng hồ). Lí do là tốc độ CPU của vi điều khiển chưa thể cập nhập lên tốc độ của XT2 được (14.7456 MHz). Vì vậy, bộ xử lí trung tâm của vi điều khiển sẽ cần một khoảng thời gian để đạt được tốc độ của thạch anh. Ngoài ra, để tránh cờ OFIFG còn bật, các cờ XT1OF, XT2OF, LFOF, DCOF bắt buộc phải tắt đồng thời.

(b) Khởi tạo bộ I2C

Để khởi tạo các thiết lập cho I2C của MSP430, trước tiên phải để ở chế độ reset. Sau đó ta chọn cổng cho SDA và SCL là thuộc USCI_B1. Vi điều khiển sẽ được cài đặt ở chế độ chủ (master) và đồng bộ. Chính vì vi điều khiển là thiết bị chủ nên nó sẽ là thiết bị cấp xung đồng hồ cho dây SCL. Với xung đồng hồ lấy từ SMCLK từ thạch anh 14.7456 MHz, thì ta

sẽ chia tần số này với hệ số là 148 để có tần số xung cấp cho I2C là 99.63 kHz (gần 100 kHz).

Để gửi đúng chính xác tới PCF-8578, thiết bị chủ (MSP430) sẽ cần địa chỉ của PCF-8578. Địa chỉ của chip này mặc định sẽ là 0111 100. Trước khi tiến hành gửi các byte thông qua I2C thì cần phải tắt reset của I2C. Sau đó muốn I2C tiến hành gửi các byte dữ liệu thì ta phải cho phép cờ ngắt GIE được bật.

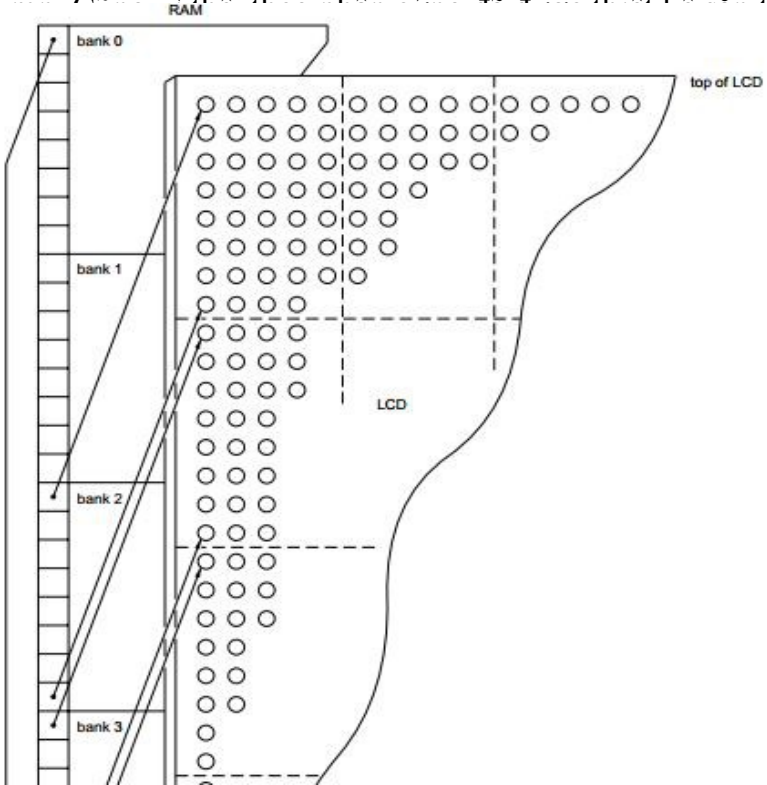
Đặc điểm của hệ thống I2C trong đề tài này là sử dụng vi điều khiển như bộ truyền (transmitter). Để thực hiện điều này, cho phép ngắt truyền của vi điều khiển để bắt đầu truyền dữ liệu.Vi điều khiển sẽ tiếp tục gửi từng byte dữ liệu đến khi hết và sẽ tạo lệnh STOP để ngưng hoạt động I2C.

(c) Thử nghiệm và lựa chọn chế độ điều khiển cho việc hiển thị trên màn hình LCD

Phần này liên quan đến việc cấu hình Display RAM trước khi gửi các byte hiển thị đến PCF-8578. Sẽ gồm có 5 byte được gửi trước, trong đó 4 byte đầu có MSB = 1 và byte cuối có MSB = 0 để xác nhận byte tiếp theo là dữ liệu hiển thị. Những điều cần lưu ý thêm đó là phần cứng thì LCD được kết nối với PCF-8578 theo 16 chân row và 24 chân column.

Có tổng cộng 5 lệnh lựa chọn chế độ điều khiển.

+ Set mode: ở phần này ta cần lựa chọn chế độ sử dụng driver, chế độ hiển thị và tỉ lệ của driver. Vì ta chỉ sử dụng 1 mình con chip PCF-8578 nên bắt buộc phải chọn mix mode, có nghĩa là kết hợp giữa 2 driver row



Hình 3.2.2.1.1 Hình dạng của Display RAM

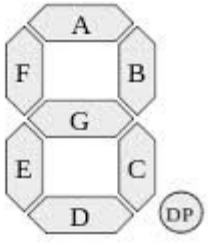
+ Device select: Ở phần này ta phải chọn subaddress của PCF-8578 và giá trị của nó là 0000.

+ RAM access: Trong phần này ta sẽ lựa chọn 1 trong 4 bank, hay là địa chỉ Y, của Display RAM để bắt đầu trữ các byte dữ liệu từ vi điều khiển qua I2C. Hơn nữa ta phải chọn chế độ trữ dữ liệu cho các byte dữ liệu đầu vào. Các byte chứa các dữ liệu hiển thị chỉ nằm ở bank 0 và bank 1 nên ta sẽ chọn chế độ ghi vào Display RAM là half-graphic, tức là byte đầu tiên sẽ ở được tải vào của địa chỉ X bank 0, byte tiếp theo ở địa chỉ X bank 1 và byte tiếp theo là ở địa chỉ X+1 tiếp theo của bank 0 và cứ lặp lại quá trình này.

+ Load X address: Ở phần này chúng ta sẽ lựa chọn địa chỉ X cho Display RAM. Vì lí do phần cứng được thiết kế sẵn, các kí tự hiển thị bắt đầu từ địa chỉ 22 đến 39 nên ta chọn X-address là 22 cho tổng quát và sau này có thể phát triển thêm cho LCD một cách toàn diện.

(d) Tiến hành hiển thị những kí tự theo nguyện vọng

Ở phần này, nhiệm vụ đặt ra là phải hiển thị được 1 số tự nhiên bé hơn 100 trên 3 vùng (giờ / phút / giây) một cách độc lập. Có một điều cần lưu ý ở đây là các số này sẽ hiển thị theo dạng led 7 đoạn. Trong LCD sẽ có 18 vị trí hiển thị số được đánh dấu index từ 1 đến 18. Như vậy, ví dụ số tại ví trí số 7 sẽ có các bit hiển thị như 7A,7B,7C,7D,7E,7F,7G,7P. Từ đó nếu ta muốn hiển thị một số bất kì ta chỉ việc hiển thị số hàng đơn vị và hàng chục theo nguyên lí bit bằng "1" là sáng còn bit bằng "0" là tắt. Và ta có thể hiển thị bất cứ số có 2 chữ số mà ta mong muốn.



Hình 3.2.2.1.2 số hiển thị kiểu led 7 đoạn

Tuy nhiên, trong Display RAM của LCD thì bit hiển thị của số ở 2 vị trí khác nhau có thể nằm trên cùng một byte. Vì vậy, nếu tác động theo từng byte gửi đi qua I2C thôi thì sẽ gây ảnh hưởng đến kết quả cuối cùng trên màn hình LCD. Điều này đòi hỏi một thuật toán khác có thể giải quyết từng con số một cách độc lập.

(e) Phát triển phương pháp hiển thị tương thích theo Display RAM của PCF-8578

Trong phần này ta sẽ phát triển theo hướng tác động BIT của từng BYTE trong Display RAM để việc hiển thị từng số sẽ không ảnh hưởng tới nhau.

	Bank 0								Bank 1							
X=22	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
X=23	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
X=24	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
.								.	.							
.								.	.							
.								.	.							
.								.	.							
X=38	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
X=39	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7

Về kết cấu của Display RAM của PCF-8578, thì chỉ có X address từ địa chỉ 22 tới 39 của bank 0 và bank 1 là chứa các bit hiển thị trên LCD. Do đó, hình dáng Display RAM cũng tương tự như một ma trận mà mỗi phần tử là một bit chứa thông tin hiển thị. Nên ta cần phải tạo một bộ ma trận có kích cỡ (2x18), trong đó mỗi phần tử là một byte và ta cũng đồng thời có thể điều khiển các bit trong byte đó.

Với ý tưởng chính như vậy, giải pháp được đưa ra ở đây chính là sử dụng union để kết hợp 2 type lại với nhau. Trong Union được tạo sẽ có 2 phần tử. Trong đó, phần tử thứ nhất là dạng "char" tức là 1 byte. Còn dạng còn lại là một đối tượng được tạo bởi lệnh struct gồm có 8 bit từ bit 0 đến bit 7. Và vì 2 dạng này được union với nhau nên chúng chia sẻ cùng bộ nhớ. Vì vậy khi ta thay đổi phần tử này cũng sẽ làm tác động đến phần tử kia. Từ đó, ta sở một dạng mới là *union_byte* như là một array 8 phần tử. Ta có thể tác động từng phần tử một và đồng thời tác động lên tất cả các phần tử cùng một thời điểm. Từ đó ta có thể tạo ra một ma trận (2x18) mà mỗi phần tử là dạng *union_byte*.

Quay lại với bài toán hiển thị số, ta sẽ cần một thư viện để hiển thị các số từ 1 tới 9. Hơn nữa, các phần tử trong thư viện này sẽ phải tác động đến các phần tử trong ma trận union để hiển thị số trên LCD. Mỗi số sẽ gồm có 7 bit như led 7 đoạn. Vì vậy ta sẽ tạo một hàm số để chia ra 6 trường hợp cho 6 con số cần hiển thị như hình 3.2.1.2.2. Điều cốt lõi là ta sẽ dùng một biến *union_byte* trung gian để lấy các giá trị bit theo led 7 đoạn từ thư viện. Rồi từ đó ta lấy biến trung gian này để thay đổi các bit cần thay đổi trong ma trận (2x18).

Ví dụ, số 7 sẽ cần đoạn a, b và c của led 7 đoạn. Ta gán cho đoạn a tương ứng với bit 0, đoạn b tương ứng với bit 1… và đoạn g tương ứng với bit 7. Để hiển thị số 7 ta cần đoạn a, b và c sáng, hay là bit 0, bit 1 và bit 2 ở giá trị cao. Khi đổi ra hệ hexa ta sẽ có 0x07 = 00000111 (binary). Như vậy biến union_byte trung gian sẽ có giá trị là 0x07(Lưu ý là thư viện hiển thị sẽ có 10 phần tử được đánh dấu từ 0 tới 9, vậy khi ta muốn lấy số 7 ra ta chỉ việc lấy phần tử thứ 7). Nếu ta muốn hiển thị số 7 ở hàng chục của giây thì ta chỉ việc gán các bitA, bitB, bitC…bitG của hàng chục của giây tương ứng với bit 0, bit 1… bit 6 của biến trung gian union_byte. Từ đó ta có thể hiển thị số 7 ở hàng chục của giây như mong muốn.

Code minh họa

```
//----------------- Số được hiển thị theo kiểu led 7 đoạn từ 0 đến 9 ------------
const unsigned char led_seg[] ={
  0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F //
//  0       1       2       3       4       5       6       7       8       9
};
//-------------------------------------------------------------------------------
//------------Tạo ma trận tương tác giữa BIT và BYTE Display
RAM------------
```

```
struct bitDATA {
  unsigned char B0:1 ;  //cac bien B0...B7 duoc hieu la BIT bang cach ":1"
  unsigned char B1:1 ;  //su dung bit cuoi cung
  unsigned char B2:1 ;
  unsigned char B3:1 ;
  unsigned char B4:1 ;
  unsigned char B5:1 ;
  unsigned char B6:1 ;
  unsigned char B7:1 ;
};
// ket hop giua BYTE va BIT voi nhau
typedef union UN_BYTE {
  unsigned char BYTE;
  struct bitDATA BIT;
}LCD_BYTE;

  LCD_BYTE UNBYTE[2][18];   //Tao ra ma tran cho DISPLAY RAM
  LCD_BYTE TRANS_BIT;       //Tao ra BYTE de truyen du lieu vao cac
 phan tu trong ma tran UNBYTE
//---------------Ham xu ly BIT cho Matrix hien thi cua Display RAM
 -------------
void Display_Function(unsigned char NUM,unsigned char POS){
  TRANS_BIT.BYTE = led_seg[NUM];   // Biến trung gian TRANS_BIT
  switch(POS){

  case 1:   //Hang chuc cua gio
    UNBYTE[0][4].BIT.B7 = TRANS_BIT.BIT.B0;
    UNBYTE[0][3].BIT.B7 = TRANS_BIT.BIT.B1;
    UNBYTE[0][2].BIT.B7 = TRANS_BIT.BIT.B2;
    UNBYTE[0][1].BIT.B7 = TRANS_BIT.BIT.B3;
    UNBYTE[1][2].BIT.B0 = TRANS_BIT.BIT.B4;
    UNBYTE[1][4].BIT.B0 = TRANS_BIT.BIT.B5;
    UNBYTE[1][3].BIT.B0 = TRANS_BIT.BIT.B6;
    break;
  case 2:   //Hang don vi cua gio
    UNBYTE[1][4].BIT.B3 = TRANS_BIT.BIT.B0;
    UNBYTE[1][3].BIT.B3 = TRANS_BIT.BIT.B1;
    UNBYTE[1][2].BIT.B3 = TRANS_BIT.BIT.B2;
    UNBYTE[1][1].BIT.B3 = TRANS_BIT.BIT.B3;
    UNBYTE[1][2].BIT.B2 = TRANS_BIT.BIT.B4;
    UNBYTE[1][4].BIT.B2 = TRANS_BIT.BIT.B5;
    UNBYTE[1][3].BIT.B2 = TRANS_BIT.BIT.B6;
    UNBYTE[1][1].BIT.B4 = 1;
```

```
    break;
  }
}
```

(f) Khởi tạo bộ định thời TIMERA0

Bộ định thời timer sẽ kích hoạt TimerA0, để ở chế độ count up và để thanh ghi giá trị CCR0 = 0x2400. Chọn xung đồng hồ vào là submain clock tương đương với 14.7456 MHz. Lí do cho những thiết lập này sẽ được giải thích rõ ở phần sau.

(g) Phân tích và thiết kế giải thuật để hiển thị thời gian giờ/phút/giây trên màn hình LCD

Để hiển thị thời gian một cách chính xác, đòi hỏi phải có một thuật toán để lấy thời gian chính xác từ Timer và cập nhập thời gian theo (giờ / phút / giây) cho I2C truyền dữ liệu.

Trong bài toán này, Timer sẽ được tận dụng để cập nhập thời gian. Theo đó ta gọi một biến second (giây) để cập nhập theo giây. Như vậy, khi biến second đạt được giá trị 60 thì ta sẽ tăng giá trị của phút lên một đơn vị. Tương tự, khi biến phút tăng lên đến giá trị 60 thì ta sẽ tăng giá trị của giờ lên một đơn vị. Khi thời gian đạt đến giá trị 24 thì tự động điều chỉnh lại bằng 0. Như vậy ta đã có một hệ thống quản lí 3 biến thời giạn giờ / phút / giây và cho phép chúng chạy theo chuẩn thời gian của một ngày.

Độ chính xác của hệ thống là mục tiêu tiếp theo. Chính vì SMCLK có giá trị rất lớn, một xung của bộ đồng hồ này sẽ rất nhỏ so với một giây khi ta muốn cập nhập theo giây trực tiếp trên timer. Xung đồng hồ được lấy từ thạch anh nên độ chính xác rất cao. Lí do ta chọn CCR0 = 0x2400 = 9216 (thập phân) vì khi đó mỗi xung ngắt của Timer sẽ có chu kì bằng (9216 Hz / 14.7456 MHz) = 0.625 ms. Khi ta thu được một chu kì chính xác như vậy, để đạt được 1 giây một cách chính xác thì chỉ cần cho Timer ngắt 1600 lần. Hệ quả là ta sẽ sử dụng một biến count để đếm trong Timer sao cho mỗi khi count = 1600 thì ta sẽ bắt đầu tăng biến second (giây).

Cuối cùng nhưng cũng quan trọng không kém, đó là nếu để việc cập nhập liên tục theo mỗi xung Timer thì sẽ rất tốn kém. Vì vậy một cờ cập nhập sẽ được bật trong Timer mỗi lần biến count đếm đến 1600. Và trong hàm chính sẽ cho phép cập nhập thời gian, hay là truyền dữ liệu trên I2C, nếu cờ ngắt này được bật. Sau khi gửi hết các byte cập nhập thời gian thì

cờ ngắt này sẽ được tắt. Theo lối đó, hệ thống sẽ tự động cập nhập thời gian theo từng giây.

(h) Thao tác nút nhấn

Các nút bấm được nối với các chân P2.7, P3.3 và P3.4. Trong chương trình chính các chân này sẽ được khai báo là INPUT (đầu vào).

Chức năng nút bấm

Nút SET (P2.7): với những tính năng được mô tả trong phần tìm hiểu lí thuyết, ta sẽ sử dụng một biến để đếm để theo dõi số lần nhấn nút SET, gọi là uc_set_count.

Nếu giá trị của uc_set_count:

Là 1: Tức là đồng hồ sẽ không chạy.
Là 2: Đồng hồ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị ở vị trí giờ.
Là 3: Đồng hồ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị ở vị trí phút.
Là 4: Đồng hồ nhấp nháy theo chu kì 1 giây và chờ tín hiệu thay đổi giá trị ở vị trí giây.
Là 5: Đồng hồ chạy bình thường trở lại với các giá trị giờ / phút / giây đã được điều chỉnh. Hơn nữa, biến uc_set_count sẽ được reset lại bằng 0.

Nút (+) (P3.4): với những tính năng được mô tả trong phần tìm hiểu lí thuyết, ta sẽ sử dụng một biến đếm để tính số lần nhấn nút (+) và biến này gọi là PLUS. Cứ mỗi lần nhấp nháy xong một chu kì thì biến này được reset.

Có tổng cộng 2 cách thực hiện nút nhấn:

Cách thứ nhất: nhấn nút nhưng không giữ và thả nút ra. Trong trường hợp này thì hệ thống sẽ tính là một lần bấm và tăng giá trị của vị trí đang bị nhấp nháy l đơn vị.
Cách thứ hai : nhấn nút nhưng tiếp tục giữ. Tính cho đến lúc thả nút ra thì hệ thống sẽ cứ tăng giá trị của vị trí đang bị nhấp nháy lên 1 đơn vị mỗi nửa giây. Hơn thế nữa, ta sẽ thay đổi chu kì nhấp nháy còn nửa giây.

Nút (-) (P3.3): với những tính năng được mô tả trong phần tìm hiểu lí thuyết, ta sẽ sử dụng một biến đếm để tính số lần nhấn nút (+) và biến này gọi là MINUS. Cứ mỗi lần nhấp nháy xong một chu kì thì biến này được reset. Cách thức thực hiện nút bấm cũng tương tự như nút (+) chỉ là thay vì tăng giá trị thì giảm giá trị.

Nhấp nháy

Khi mà uc_set_count đang nằm trong các giá trị {3;4;5} thì đồng nghĩa với việc nhấp nháy xảy ra. Chu kì nhấp nháy dự định là 1 giây, nên cứ nửa giây là ta sẽ cho chữ số hiện ra rồi nửa giây sau là biến mất. Việc này có thể thực hiện bằng cách cho Timer A0 đếm 800 chu kì (tương ứng với 0.5 giây). Mỗi lần đếm xong thì ta cho LCD lần lượt hiển thị và xóa đi tại vị trí thời gian đang được quan tâm. Để việc hiển thị và xóa đi thực hiện được trong hàm chính, ta thêm một cờ xóa hiển thị số trong hàm hiển thị Display_Function và chỉ việc đảo cờ đó trong hàm chính để thực hiện việc nhấp nháy.

Code minh họa: (Trong vòng while(1))
```
clear ^= BIT0;      //đảo bit
Display_Function(uc_hour % 10,Dig_Hour,clear);
Display_Function(uc_hour / 10,Dec_Hour,clear);
```

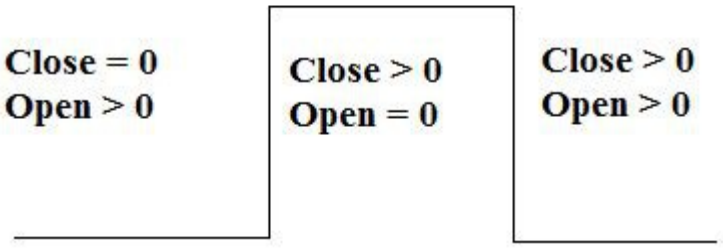
Xử lí nút bấm

Độ chính xác của nút bấm (+) và (-) đòi cần phải có những đặc điểm sau:
- Sau mỗi lần nhấn và thả ra ngay tức khắc được tính là một lần bấm.
- Nếu nhấn giữ thì cứ nửa giây sẽ tăng hoặc giảm 1 đơn vị.

Đối với nút SET thì việc tính 1 lần xảy ra khi có thao tác nhấn và thả nút bất kể khi nhấn có giữ phím hay không.

Phương án:
Gọi 2 biến close và open để theo dõi quá trình nhấn nút. Mỗi chu kì nếu có nhấn nút thì biến close tăng 1 đơn vị, còn nếu không có thì biến open tăng 1 đơn vị. Tuy nhiên, khi biến close tăng 1 đơn vị thì biến open sẽ được điều chỉnh về 0.

Đối với nút SET, khi mà cả 2 giá trị của open và close đều lớn hơn 0 thì được tính là một lần nhấn.



Hình 3.2.2.1.3

Đối với 2 nút nhấn (+) và (-), thuật toán cũng tương tự. Chỉ cần bổ sung thêm tính năng là nếu giữ nút lâu thì tốc độc nhấp nháy sẽ tăng gấp đôi và giá trị của vị trí đang được quan tâm sẽ tăng hoặc giảm 1 đơn vị mỗi nửa giây. Chính vì chức năng này, nên ta cần một biến để thay đổi thời gian nhấp nháy. Gọi biến đó là toggle_value. Nếu nhấp nháy bình thường thì toggle_value = 800 (tức là 0.5 giây). Nếu đang nhấp nháy bình thường mà có hiện tượng nhấn giữ nút tăng hoặc giảm thì toggle_value = 400 (tức là 0.25 giây). Khi mà giá trị của uc_set_count nằm trong vùng {3;4;5} thì ta cho phép Timer A0 đếm số lượng xung theo toggle_value để thực hiện việc nhấp nháy.

Hàm xử lí thời gian

Hàm xử lý thời gian sẽ đảm bảo cho giờ nằm trong vùng giá trị nguyên từ 0 đến 23, phút nằm trong vùng giá trị nguyên từ 0 đến 59 và giây nằm trong vùng giá trị nguyên từ 0 đến 59.

Ngoài ra, khi giây tăng từ 59 lên 60 thì tăng phút lên 1 đơn vị và giây bằng 0. Ngược lại, khi giây đang ở 0 mà giảm 1 đơn vị thì phút cũng giảm 1 đơn vị và giây bằng 59.

Khi phút tăng từ 59 lên 60 thì tăng giờ lên 1 đơn vị và phút bằng 0. Ngược lại, khi phút đang ở 0 mà giảm 1 đơn vị thì giờ cũng giảm 1 đơn vị và phút bằng 59.

(i) Tổng hợp thành chương trình chính

Trong chương trình chính thì sẽ có bộ Timer để cập nhập thời gian theo (giờ / phút / giây) và cứ mỗi giây như vậy hàm chính sẽ dùng bộ xử lý thời gian để thay đổi các bit trên Display RAM để hiển thị ra các con số tương ứng và dùng I2C để gửi các byte thiết lập cũng như byte dữ liệu hiển thị.

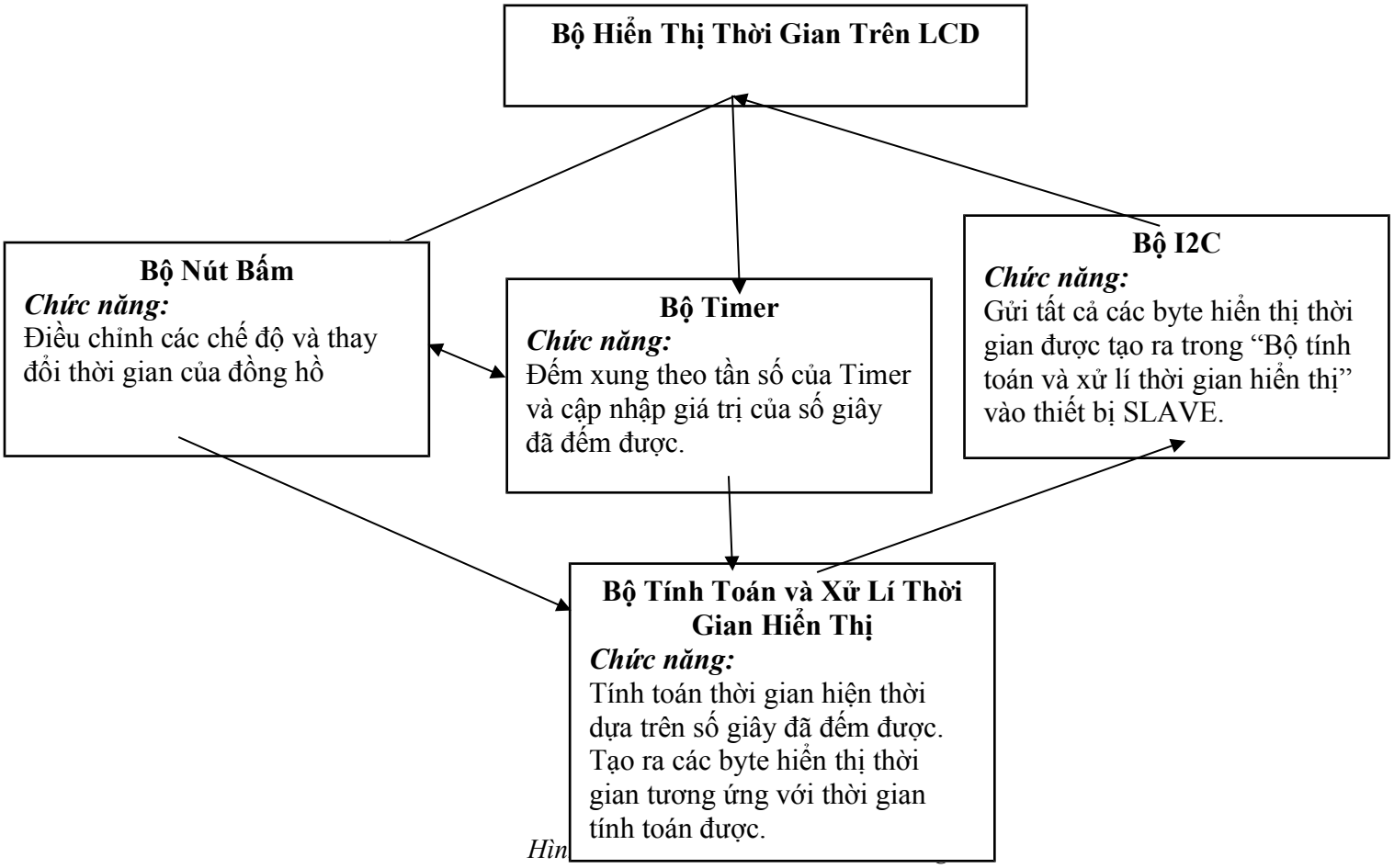
(j) Kiểm tra

Để kiểm tra, đồng hồ điện tử được chạy để so sánh với một đồng hồ chuẩn khác trong một buổi chiều từ 14 giờ 50 đến 17 giờ 26 phút và kết quả cuối cùng của 2 đồng hồ vẫn giống nhau. Việc sử dụng thạch anh mang lại hiệu quả chính xác cao.

1.2.4 Chương trình chính

(a) Sơ đồ khối

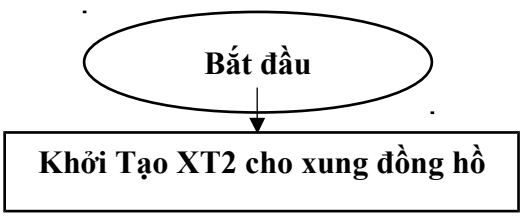
Trong sơ đồ khối gồm có các khối chính như: Bộ tính toán và xử lí thời gian hiển thị, bộ I2C, bộ Timer và bộ hiển thị thời gian trên LCD.

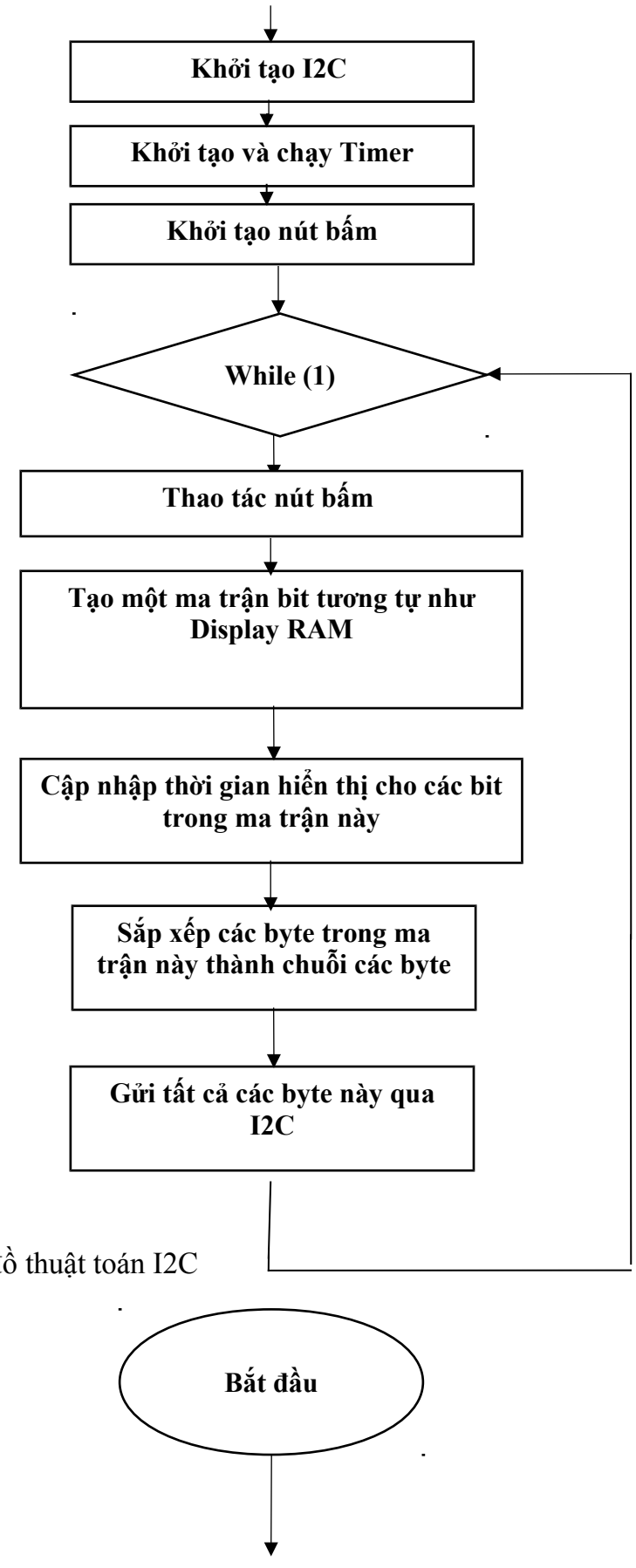


Hìn

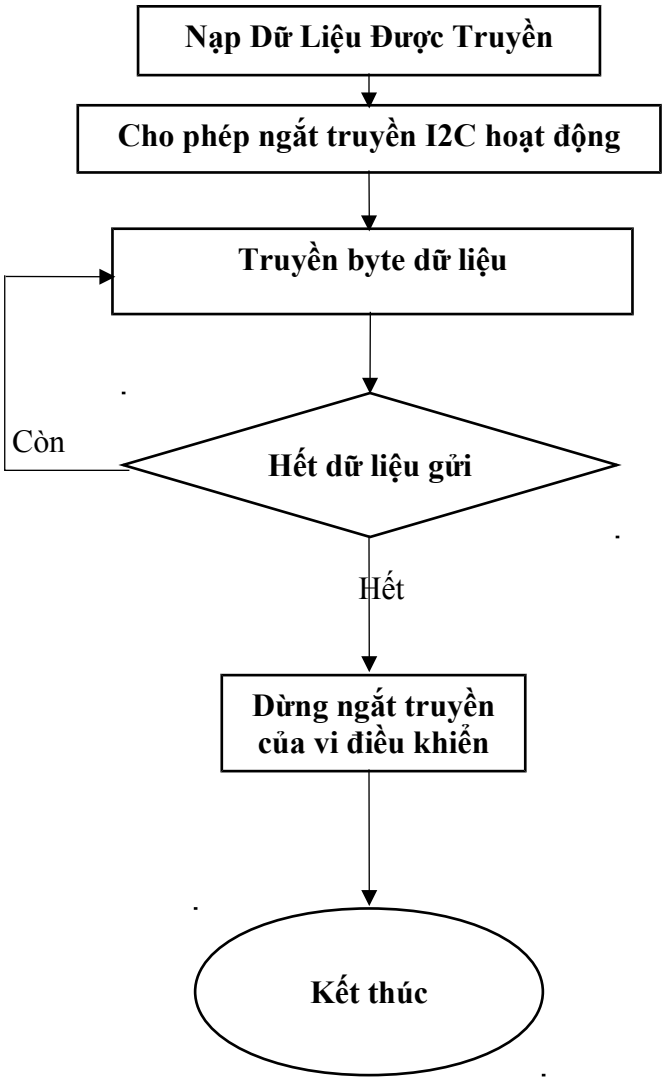
(b) Sơ đồ thuật toán

Thuật toán chính

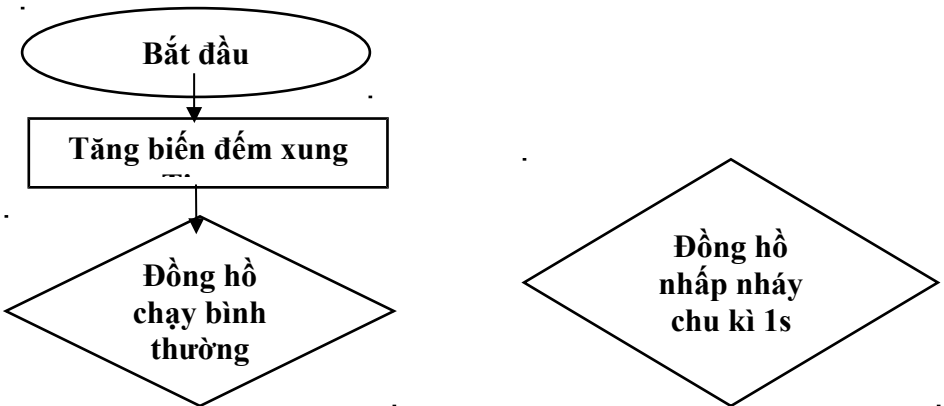


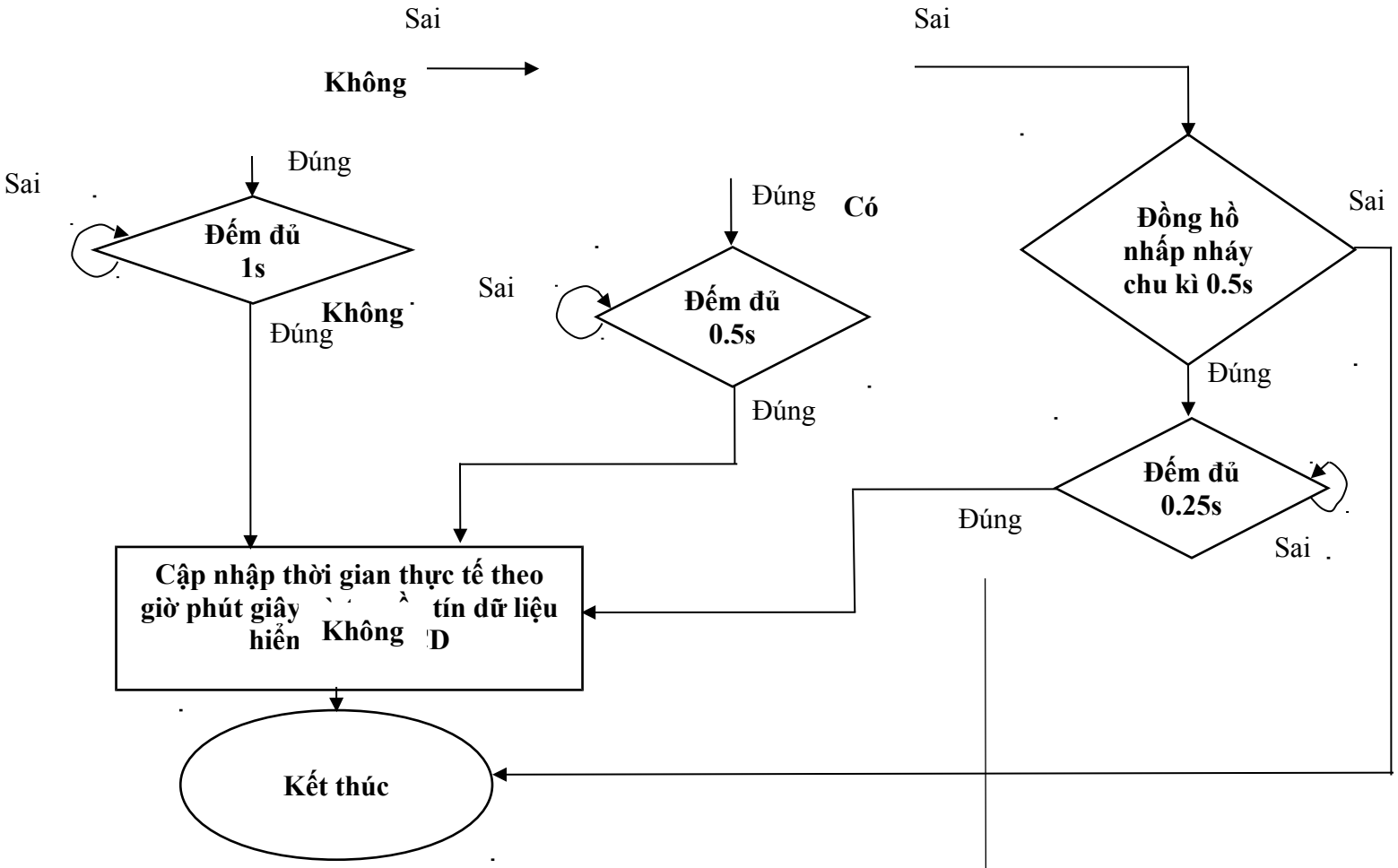


Sơ đồ thuật toán I2C



Sơ đồ thuật toán Timer cho đồng hồ chạy bình thường

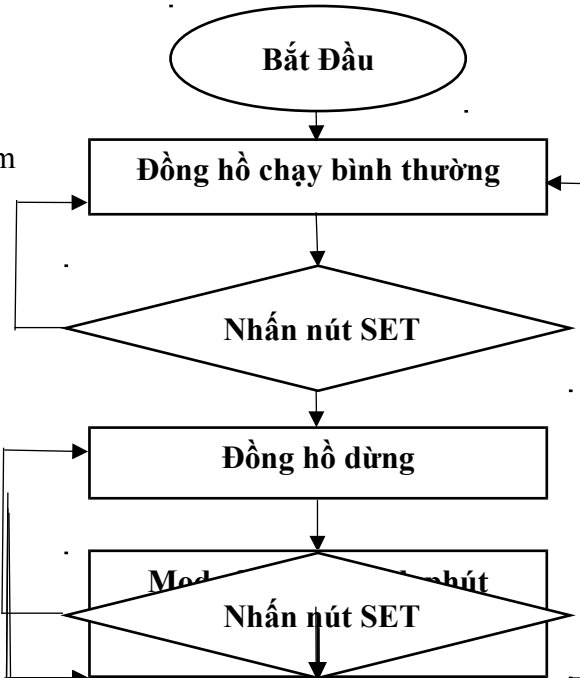


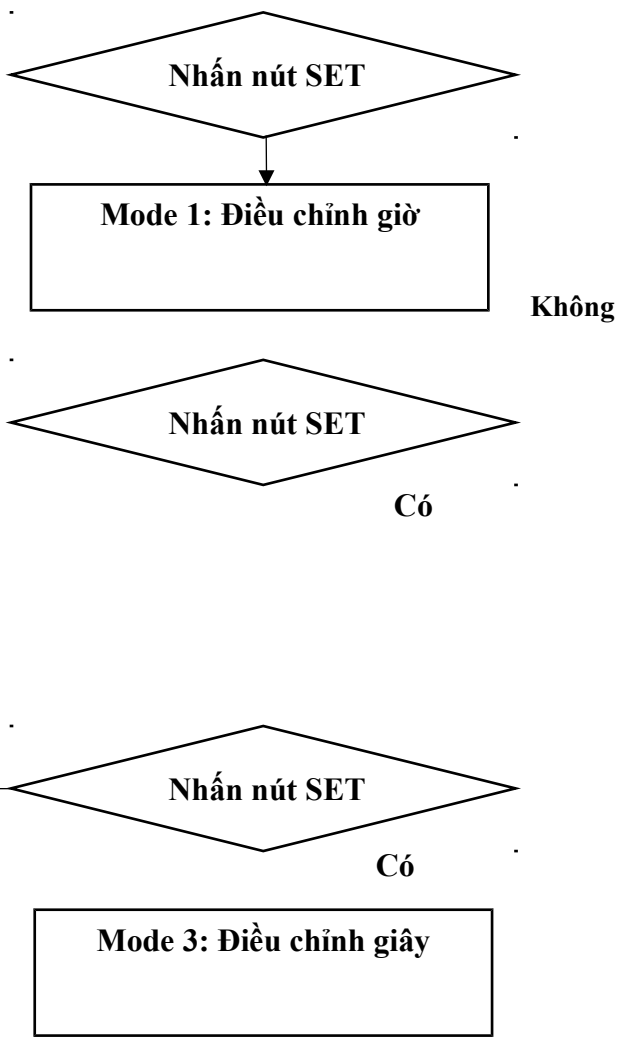


Chú thích Khi Timer đếm đủ 1600 xung tức là 1 giây đã trôi qua.

Không

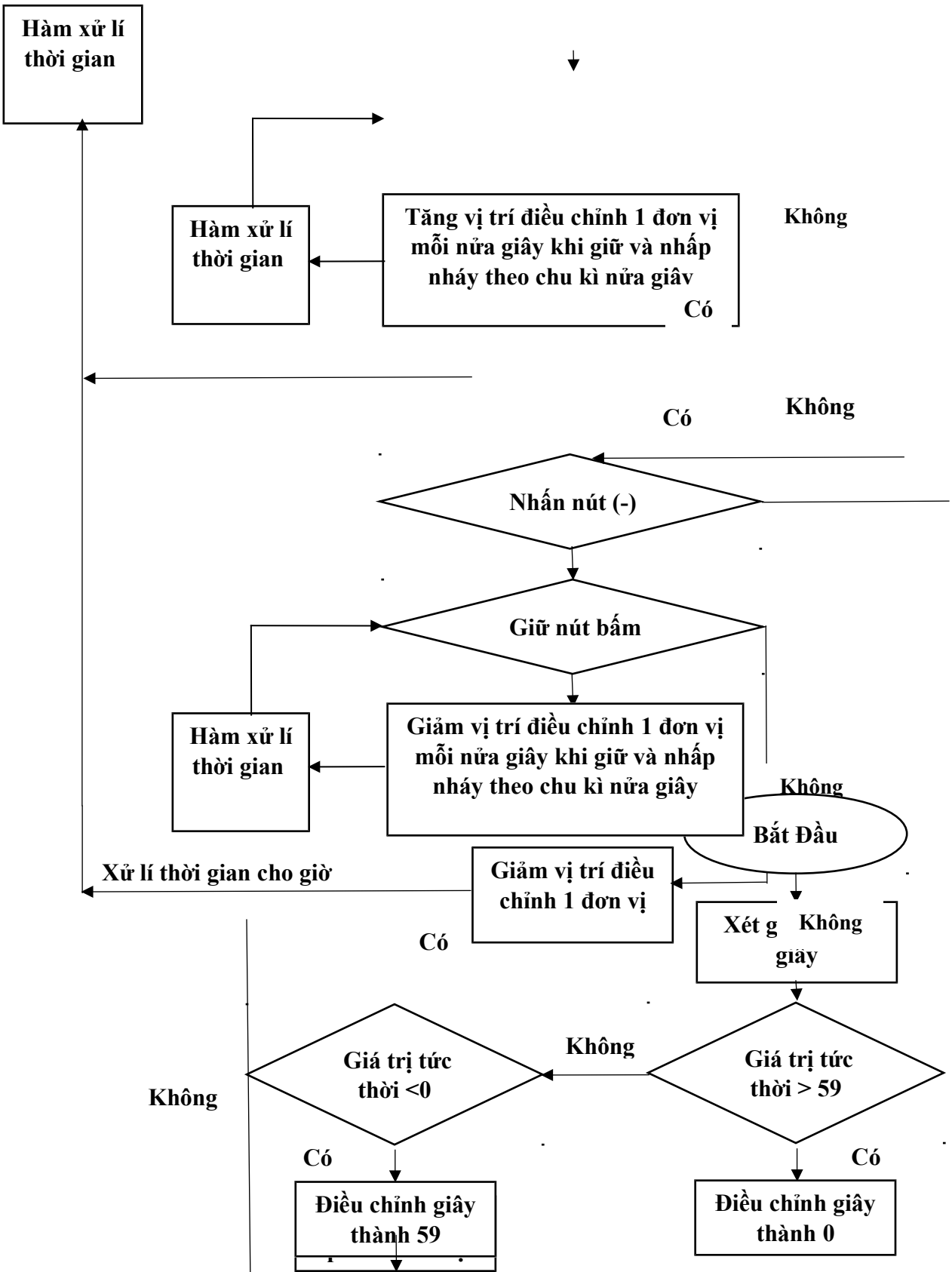
Sơ đồ thuật toán nút bấm

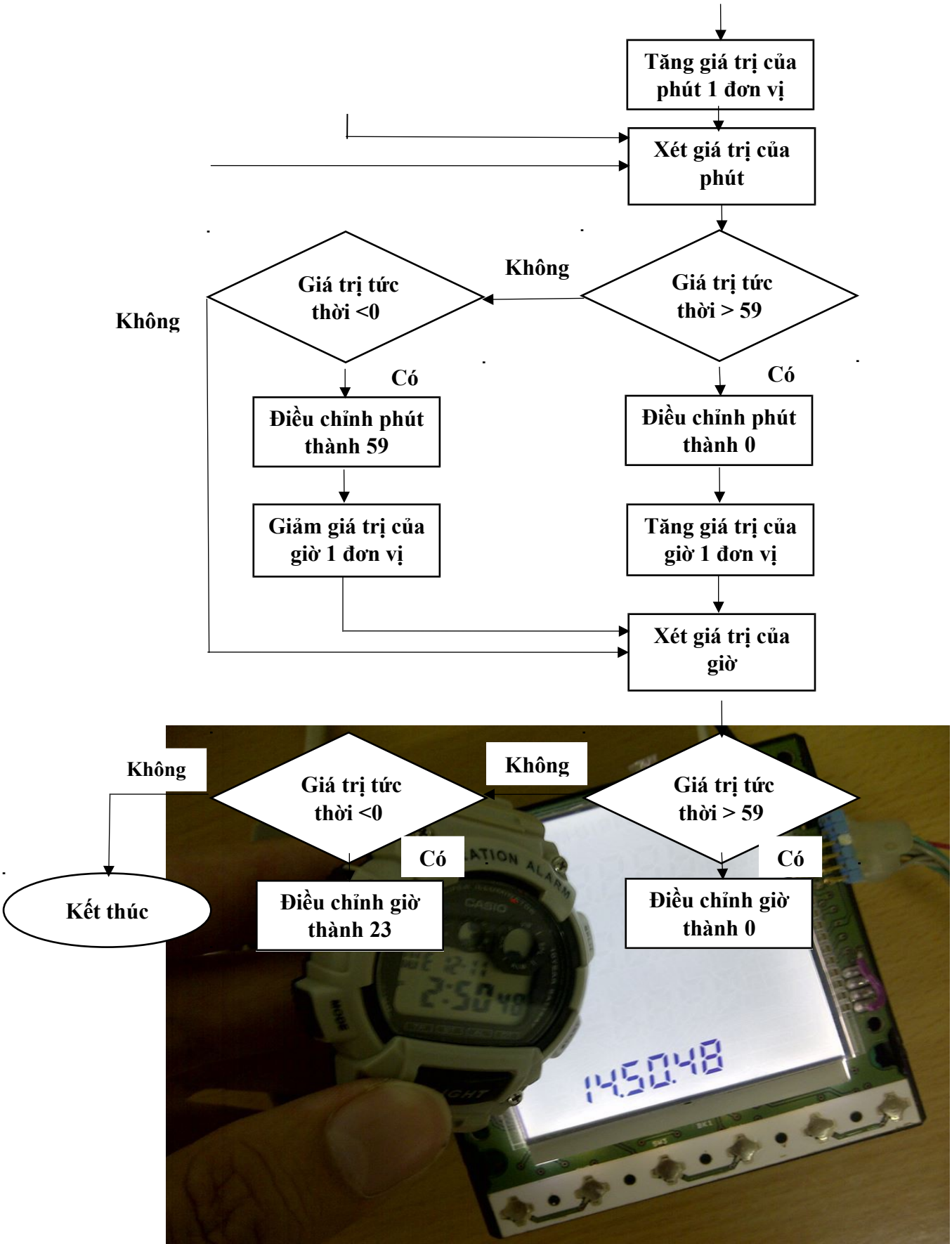




Sơ đồ thuật toán các mode:

Mode 1: Điều chỉnh về giờ
Mode 2: Điều chỉnh về phút
Mode 3: Điều chỉnh về giây


Hàm xử lí thời gian
Hàm xử lí thời gian
Tăng vị trí điều chỉnh 1 đơn vị mỗi nửa giây khi giữ và nhấp nháy theo chu kì nửa giâv
Không
Có
Có
Không
Nhấn nút (-)
Giữ nút bấm
Hàm xử lí thời gian
Giảm vị trí điều chỉnh 1 đơn vị mỗi nửa giây khi giữ và nhấp nháy theo chu kì nửa giây
Không
Bắt Đầu
Xử lí thời gian cho giờ
Giảm vị trí điều chỉnh 1 đơn vị
Có
Xét g Không
giây
Giá trị tức thời <0
Không
Giá trị tức thời > 59
Không
Có
Có
Điều chỉnh giây thành 59
Điều chỉnh giây thành 0


Tăng giá trị của phút 1 đơn vị
Xét giá trị của phút
Giá trị tức thời > 59
Không
Giá trị tức thời <0
Không
Có
Có
Điều chỉnh phút thành 0
Điều chỉnh phút thành 59
Tăng giá trị của giờ 1 đơn vị
Giảm giá trị của giờ 1 đơn vị
Xét giá trị của giờ
Giá trị tức thời > 59
Không
Giá trị tức thời <0
Không
Có
Có
Điều chỉnh giờ thành 0
Điều chỉnh giờ thành 23
Kết thúc

Hình 3.2.3.1 Ảnh đồng hồ lúc đầu trong lúc kiểm tra

Khi thử các nút SET, hiệu ứng của việc nhấn nhả và thả ngay lập tức cũng giống như việc nhấn giữ nút và thả ra. Đó là đồng hồ nhảy chế độ 1 lần.

Khi đồng hồ đang ở các chế độ 3,4,5 thì các nút (+) và (-) mới có tác dụng. Hiệu ứng khi nhấn và thả liền giúp điều chỉnh thời gian (giờ / phút / giây) tăng giảm bình thường. Tuy nhiên, khi có sự kiện nhấn giữ thì thời gian tăng theo tốc độ gấp đôi.

4. Biểu đồ kế hoạch thời gian

Thời gian dự kiến để thực hiện đồ án sẽ là 3 tháng.
Trong đó các nhiệm vụ cụ thể là:

Nhiệm vụ 1: Tìm hiểu về chip MSP430-F47176.
Nhiệm vụ 2: Tìm hiểu khởi tạo I2C.
Nhiệm vụ 3: Thuật toán đồng hồ và phương thức hiển thị.
Nhiệm vụ 4: Code Programming và Debug.
Nhiệm vụ 5: Báo Cáo Thực nghiệm Phần điều khiển LCD.
Nhiệm vụ 6: Thuật toán và phương pháp thao tác nút.
Nhiệm vụ 7: Code Programming & Debug.
Nhiệm vụ 8: Báo cáo Thực Nghiệm Phần điều khiển nút.

Tháng	11																		12				
Nội Dung	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5
	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5
Nhiệm vụ 1																							
Nhiệm vụ 2																							
Nhiệm vụ 3																							
Nhiệm vụ 4																							

Tháng	12																						
Nội Dung	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7
	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Nhiệm vụ 4																							
Nhiệm vụ 5																							
Nhiệm vụ 6																							
Nhiệm vụ 7																							

Tháng	12			1																			
Nội Dung	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2
	29	30	31	1	2	3	4	5	6	6	8	9	10	11	12	13	14	15	16	17	18	19	20
Nhiệm vụ 6																							
Nhiệm vụ 7																							

Tháng	1											2											
Nội Dung	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4	T5	T6	T7	CN	T2	T3	T4
	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12
Nhiệm vụ 7																							
Nhiệm vụ 8																							

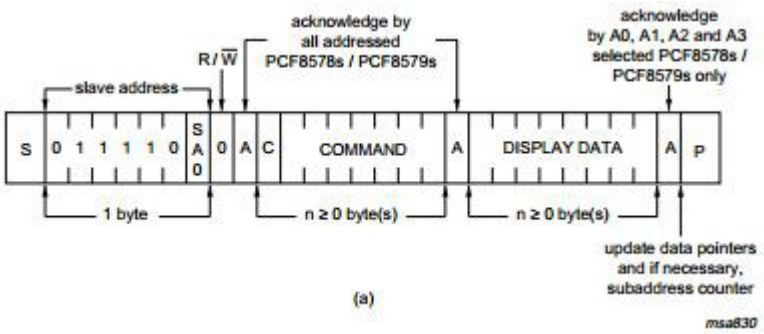
5. Kết luận

1.1 Những kinh nghiệm và kiến thức học được trong quá trình thực tập

Trong giai đoạn đầu của quá trình thực tập, thông qua việc tìm hiểu về lí thuyết, tôi đã tích lũy cho mình được một số kiến thức quan trọng cho công đoạn thực hành sau này. Trong đó nhưng kiến thức cần lưu ý nhất đó là tổng quan và phương thức hoạt động của I2C; và chức năng của PCF-8578 trong việc hiển thị LCD.

Trong quá trình thực hành, một trong những kinh nghiệm đáng lưu ý nhất đó là việc khởi tạo xung đồng hồ cho vi điều khiển bằng thạch anh. Để thực hiện được điều này, việc đầu tiên cần làm sẽ là tắt toàn bộ các hệ thống cấp nguồn xung khác. Ví dụ trong trường hợp của đồ án này, để dùng nguồn cấp từ XT2 thì phải tắt tất cả các nguồn từ XT1 và DCO. Tuy nhiên, tần số dao động của thạch anh rất lớn nên để CPU của vi điều khiển đạt được tốc độ đó một cách nhanh chóng thì không thể được. Chính vì vậy, phải cho CPU của vi điều khiển thêm thời gian để đạt được tốc độ đó.

Trong quá trình tìm hiểu lí thuyết về I2C, các chu trình chọn thiết bị master, xác định địa chỉ slave là để giúp người đọc hiểu cách hoạt động. Tuy nhiên, việc sử dụng chúng dựa trên điều khiển phần mềm của vi điều khiển có một chút khác biệt. Đối với đồ án này nói riêng, chức năng của MSP430-F47176 là Master và truyền các tập byte dữ liệu vào thiết bị tớ đã được định địa chỉ. Việc thiết lập phần mềm trên vi điều khiển chỉ cần xác định địa chỉ của tớ và cho ngắt truyền dữ liệu (Transmitter) là vi điều khiển tự hiểu và gửi byte đầu tiên gồm địa chỉ và bit Write/Read.



Hình 5.1.1 Định dạng các byte dữ liệu được truyền trong I2C

Ngoài ra, trong đồ án này có một yếu tố không thể thiếu là độ chính xác của đồng hồ. Việc dùng thạch anh lấy xung cho Timer dẫn đến kết quả cho ta những xung rất chính xác về thời gian. Việc còn lại chỉ là làm sao để đếm số xung sao cho đạt được chính xác 1 giây. Xét thanh ghi CCR0 (vì ta sử dụng Timer_A0) có giá trị lớn nhất là 0xFFFF = 65535. Gọi N là giá trị của CCR0, như vậy $N \leq 65535$. Mỗi xung của Timer_A0 sẽ là (N/14745600) giây. Gọi M là số xung mà Timer_A0 phải đếm để được 1 giây thì $M = \frac{14745600}{N}$. Với M và N là số tự nhiên, nên ta phải lựa N lớn nhất có thể mà thỏa mãn M là số tự nhiên. Phân tích số 14745600 ra các thừa số nguyên tố và ta chọn được $N = 2^{12}.3^2 = 36864$. Vậy ta thu được M = 400. Có nghĩa là khi ta chọn CCR0 = 36864 = 0x9000 thì chỉ cần đếm Timer_A0 400 lần thì ta sẽ được 1 giây. Tuy nhiên, đây là thuật toán ban đầu khi chưa có thao tác nút bấm. Sau này khi, dùng cả thao tác nút bấm thì giá trị của CCR0 buộc phải giảm xuống 4 lần để việc kiểm soát chấ lượng nút bấm trở nên tốt hơn.

Ý tưởng của việc hiển thị đồng hồ là cứ mỗi giây I2C sẽ gửi tất cả các byte dữ liệu vào thiết bị PCF-8578 để điền vào tổng cộng 36 vị trí trên Display RAM. Chính vì thế nếu điều khiển được từng bit trên các byte dữ liệu được gửi sẽ tiện lợi hơn so với điều khiển chính byte đó. Một phương án giải quyết được đề cập ra chính là dùng union để tạo ra một ma trận *Union_Byte* đã được nêu ở phần thực hành.

Ngoài ra, khi được thực tập ở đây, em đã học được một số kĩ năng quản lí đồ án của mình. Điều quan trọng khi ta tiến hành một đồ án là luôn luôn lưu lại từng bước những gì mình đã làm được. Sau này mỗi khi gặp vấn đề em có thể luôn bắt đầu lại từ lần gần đây nhất. Ngoài ra còn có một số phương pháp trình bày và đặt tên biến trong code để cho tiện theo dõi.

1.2 Các khó khăn và hạn chế khi thực tập

Những khó khăn và hạn chế trong suốt quá trình thực tập đều xảy ra do tìm hiểu lí thuyết chưa kĩ hoặc do thiếu kiến thức thực tế mà ra.

Việc khởi tạo xung thạch anh cho vi điều khiển chính là một trong những khó khăn đó. Sau khi cố gắng tìm mọi cách để tắt cờ ngắt OFIFG như đã nêu ở phần thực hành, vi điều khiển vẫn chưa dùng được xung đồng hồ của thạch anh. Với sự giúp đỡ của người hướng dẫn, kĩ sư Trần Văn Tâm, em đã khởi tạo được thạch anh cho xung đồng hồ bằng cách cho CPU thêm thời gian để đạt tốc độ lớn của thạch anh.

Ngoài ra, tự bản thân em còn thấy mình thiếu kiến thức về an toàn điện và khả năng tự sửa lỗi trong lập trình của mình. Trong suốt quá trình thực hành, có nhiều công đoạn đã đạt gần đích. Tuy nhiên, khi xuất hiện những lỗi nhỏ thì vẫn chưa tự khắc phục được. Chứng tỏ cần phải tự rèn luyện thêm về mặt tư duy logic trong lập trình.

1.3 Dự định, mong muốn và nguyện vọng sau khi thực tập

Xuyên suốt quá trình thực tập, em tự thấy bản thân đam mê bên mảng lập trình nhúng hơn. Đặc biệt là khi chiếc đồng hồ được tạo ra đã đạt được những yêu cầu trong đồ án như độ chính xác, cách trình bày… Từ đó, em tự xác định mình muốn được tiếp tục phát triển bản thân bên mảng lập trình nhúng để có thể tạo ra các sản phẩm có ích cho xã hội.

Hơn thế nữa, môi trường làm việc vô cùng quan trọng. Việc được thực tập trong môi trường chuyên nghiệp và thân thiện. Được các anh chị đội ngũ cán bộ trong công ty nhiệt tình hướng dẫn và quan tâm giúp đỡ nên em có thể thoải mái phát triển các ý tưởng của mình. Mặc dù còn nhiều sai sót và thiếu trách nhiệm, nhưng các anh chị vẫn tạo cho em điều kiện tốt nhất có thể. Nên em rất mong muốn về lâu về dài được tiếp tục đồng hành cùng công ty.

1.4 Các cảm nghĩ khác của bản thân

Việc hiển thị thời gian trên LCD vẫn thuộc một phần nhỏ có thể thực hiện được từ các tài nguyên của đồ án này. Chính vị vậy, bản thân em vẫn mong muốn mình có thể có cơ hội để được phát triển thêm một vài chức năng khác cho thiết bị tổng quan dưới sự cho phép và hướng dẫn của anh Trần Văn Tâm.

6. Mục tham khảo

[1] Texas Instrument, *MSP430x4xx Hardware – SLAS626C*, 2011.

[2] Texas Instrument, *MSP430x4xx Family User's Guide – SLAU056J*, 2010.

[3] NXP, *PCF-8578 LCD row/column driver for dot matrix graphic display*, 2009.

[4] Nguyễn Chí Linh, *"Giới thiệu giao tiếp I2C"*.
http://ebook.ringring.vn/xem-tai-lieu/gioi-thieu-giao-tiep-i2c/107932.html