

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA ĐÀ NẴNG
KHOA ĐIỆN TỬ - VIỄN THÔNG



BÁO CÁO



THỰC TẬP TỐT NGHIỆP

ĐƠN VỊ THỰC TẬP:

Công Ty TNHH Takemoto Việt Nam

Sinh viên thực hiện : Nguyễn Quang Huy

Lớp :17DT1

MSSV :106170027

Người hướng dẫn : K.s Đỗ Thanh Tuyền

: K.s Trần Đình Lợi

Giảng viên hướng dẫn: K.s Lê Hồng Nam

BÁO CÁO THỰC TẬP TỐT NGHIỆP

TRƯỜNG ĐẠI HỌC BÁCH KHOA CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
KHOA ĐIỆN TỬ - VIỄN THÔNG Độc lập - Tự do - Hạnh phúc

ĐỀ CƯƠNG THỰC TẬP TỐT NGHIỆP

Sinh viên thực tập: Nguyễn Quang Huy

Lớp: 17DT1

Chuyên ngành học: Kỹ Thuật Điện Tử

Người hướng dẫn: K.s Đỗ Thanh Tuyền

I. MỤC ĐÍCH YÊU CẦU:

1. Về chính trị tư tưởng:

Rèn luyện đạo đức, tác phong đề xứng đáng với cương vị của người kỹ sư khi ra trường.

Nâng cao ý thức tổ chức, tính kỷ luật, ý thức chấp hành nội quy tại cơ quan thực tập, cũng như nơi làm việc sau khi ra trường.

2. Về chuyên môn:

Tiếp cận các lĩnh vực ứng dụng chuyên ngành Kỹ Thuật Điện Tử để nắm bắt được ứng dụng của lý thuyết vào thực tế sản xuất và xu hướng phát triển của ngành trong giai đoạn mới.

II. THỜI GIAN VÀ ĐỊA ĐIỂM:

1. Thời gian: - Từ ngày: 12/07/2021

- Đến ngày: 20/08/2021

2. Địa điểm thực tập: 1254 Xô Viết Nghệ Tĩnh, Phường Hoà Cường Nam, Quận Hải Châu, Thành phố Đà Nẵng

III. NỘI DUNG THỰC TẬP:

- Giới thiệu tổng quan về Công ty TNHH TAKEMOTO Việt Nam – chi nhánh Đà Nẵng.

- Tổng quan hệ thống được sử dụng trong công ty: Sơ đồ hệ thống, nguyên lý làm việc của hệ thống, lý do sử dụng các hệ thống đó.

- Đề tài thực hiện: Hiện thị thời gian lên đồng hồ đo.

- Tìm hiểu về thiết bị : Tìm hiểu về thiết bị phần cứng, quá trình thực hiện đề tài, nguyên lý hoạt động của thiết bị, cuối cùng là kết quả.

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

- Phân tích các nhược điểm: Nêu ra các nhược điểm chưa xử lý được, dựa kiến thức đã học và tài liệu của công ty để phân tích các nguyên nhân gây ra nhược điểm và đưa ra giải pháp khắc phục.

Đà Nẵng, ngày.... tháng.... năm 2021

Giảng viên hướng dẫn

ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN TỬ - VIỄN THÔNG

Địa chỉ : 54 Nguyễn Lương Bằng, Quận Liên Chiểu, Tp Đà Nẵng

Phone : (0511) – 384128 DD : 0905888809

Email : lehongnam@gmail.com

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

BẢNG ĐÁNH GIÁ KẾT QUẢ THỰC TẬP TỐT NGHIỆP

Họ và tên sinh viên : Nguyễn Quang Huy

Lớp: 17DT1

Cơ quan thực tập : Công ty TNHH TAKEMOTO Việt Nam – chi nhánh Đà Nẵng

Thời gian thực tập : từ 12/07/2021 đến 20/08/2021

Người trực tiếp hướng dẫn (tại cơ quan thực tập) :K.s Đỗ Thanh Tuyền

I. ĐÁNH GIÁ VỀ CÁC PHẨM CHẤT CỦA SINH VIÊN THỰC TẬP

A - Khả năng trí tuệ	Tốt	Khá	Trung bình	Yếu
Thông minh, trí tuệ, khả năng sáng tạo		✓		
Khả năng thực hành		✓		
Hoài bão, khát vọng	✓			
B - Tính chất con người	Tốt	Khá	Trung bình	Yếu
Khả năng truyền đạt và tiếp nhận thông tin (Kỹ năng thông tin)	✓			
Quan hệ trong tập thể	✓			
Tính thân thiện, năng động	✓			

II. ĐÁNH GIÁ VỀ KẾT QUẢ THỰC TẬP

A - Các công việc của sinh viên thực hiện trong đợt thực tập	Tốt	Khá	Trung bình	Yếu
Khả năng làm việc nhóm	✓			
Giờ giấc làm việc	✓			
Kiến thức tổng quát		✓		
Phương pháp làm việc		✓		
Khối lượng công việc	✓			
Khả năng tổng kết công việc		✓		
B - Bảng báo cáo thực tập	Tốt	Khá	Trung bình	Yếu
Sự chuẩn bị báo cáo		✓		

BÁO CÁO THỰC TẬP TỐT NGHIỆP

Cấu trúc bản báo cáo		✓		
Cách diễn đạt		✓		
Nội dung báo cáo		✓		
Khả năng phát triển		✓		

III. CÁC ĐÁNH GIÁ KHÁC:

Đã hoàn thành chương trình thực tập. Trong quá trình thực tập đã có nỗ lực và tiến bộ theo thời gian. Tuy nhiên cần cố gắng hơn nữa để trao dồi kiến thức nhiều dòng chip khác nhau, ngôn ngữ lập trình C cũng như trình bày thuật toán dễ hiểu. Đánh giá cao tính thân thiện và quan hệ tốt trong môi trường tập thể

.....

.....

.....

.....

.....

.....

.....

.....

Xác nhận của Cơ quan/ Đơn vị thực tập
(Ký, ghi rõ họ tên và đóng dấu)



NGƯỜI HƯỚNG DẪN
(Tại cơ quan thực tập)
(Ký và ghi rõ họ tên)

ĐỖ THANH TUYỀN

NHẬN XÉT CỦA THẦY/CÔ HƯỚNG DẪN THỰC TẬP

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên hướng dẫn

BÁO CÁO THỰC TẬP TỐT NGHIỆP

LỊCH THỰC TẬP

Tuần	Thứ	Nội dung thực tập	Thực hiện	Ghi chú
1 (12/07/2021)	2	Tìm hiểu tổng quan về công ty	Đại diện phía công ty giới thiệu chung về công ty	
	3	Nhận tài liệu và nhiệm vụ	Mỗi thành viên tự tìm hiểu thông qua tài liệu công ty cung cấp Sinh viên thực tập đưa ra các thắc mắc để công ty giải đáp	
	4	Tiến hành phân chia công việc	Cả nhóm tiến hành họp online để chia nhóm thực tập Thảo luận nhóm về công việc được giao, tìm kiếm tổng quan về công việc được giao và phân chia công việc	
	5	Nghiên cứu tài liệu và các mảng công ty đang làm	Mỗi người tự tìm hiểu chi tiết về lý thuyết của phân việc đã chia dựa theo tài liệu công ty cung cấp	
	6			
2 (19/07/2021)	2	Triển khai các nội dung cần có trong báo cáo	Làm việc nhóm và công ty hướng dẫn triển khai các nội dung	
	3	Người hướng dẫn kiểm tra nội dung thực tập	Người hướng dẫn kiểm tra lại các nội dung lý thuyết mà sinh viên đã tìm hiểu	
	4	Tìm hiểu về phần cứng của thiết bị.	Sinh viên tự nghiên cứu, tự học và tự làm bài tập thông qua các tài liệu công ty giao	
	5			
	6	Phân tích cách thức hoạt động, cách cấu hình	Cả tổ tiến hành họp online trên MS Teams để thảo luận cùng người hướng dẫn	
3 (26/07/2021)	2	Tìm hiểu về vi điều khiển MSP430.	Sinh viên tự nghiên cứu, tự học và tự làm bài tập thông qua các tài liệu công ty giao	
	3			
	4	Tìm hiểu về chip PCF8578 và giao tiếp i2c		
	5			
	6	Phân tích chi tiết thiết bị và cách hiển thị lên màn hình LCD	Cả tổ tiến hành họp online trên MS Teams để thảo luận cùng người hướng dẫn	
4 (02/08/2021)	2	Giảng viên hướng dẫn và kiểm tra nội dung thực tập theo đề cương và quy định	Báo cáo sơ bộ về tiến độ thực hiện cho giảng viên	
	3	Tìm hiểu và tiến hành giao tiếp i2c giữa vi điều khiển MSP430 và chip PCF8578	Sinh viên tự nghiên cứu, tự học và tự làm bài tập thông qua các tài liệu công ty giao	

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

	4	Tìm hiểu chân GPIO và cấu hình timer		
	5			
	6	Tiến hành hiển thị số ngẫu nhiên lên màn hình LCD	Cả tổ tiến hành họp online trên MS Teams để thảo luận cùng người hướng dẫn	
5 (09/08/2021)	2	Tiến hành coding lắp ghép các hàm để hoàn thành chương trình	Sinh viên thực hiện công việc của cá nhân được phân và tiến hành viết báo cáo thực tập	Đề tài: Hiển thị thời gian lên màn hình LCD
	3			
	4			
	5	Phân tích các nhược điểm của chương trình và đưa ra giải pháp cải tiến	Sinh viên tiến hành làm việc nhóm để thảo luận	
	6	Demo cho người hướng dẫn đánh giá	Từng nhóm báo cáo dự án cho người hướng dẫn	
6 (16/08/2021)	2	Giảng viên kiểm tra nội dung thực tập tốt nghiệp	Từng nhóm báo cáo nội dung thực tập cho giảng viên	
	3	Điều chỉnh lại báo cáo sao cho phù hợp với nhận xét của người hướng dẫn	Cá nhân tiến hành thực hiện điều chỉnh báo cáo	
	4			
	5	Công ty nhận xét và cho điểm thực tập tốt nghiệp	Tiến hành vấn đáp	
	6	Giảng viên kiểm tra vấn đáp các nhóm	Kiểm tra trực tuyến qua MS Teams	

LỜI NÓI ĐẦU

Với mong muốn hoàn thiện cho bản thân kiến thức để đáp ứng tốt cho công việc trong tương lai và cần phải hiểu biết cả lý thuyết lẫn thực hành. Để đáp ứng tốt những yêu tố về kiến thức thực tế em đã viết bài báo cáo thực tập tốt nghiệp này.

Được sự phân công của khoa Điện tử – Viễn thông trường Đại Học Bách Khoa Đà Nẵng và sự đồng ý của Công ty TNHH Takemoto, chúng em được thực tập tại đây trong vòng 6 tuần. Với sự chỉ bảo tận tình của các anh chị trong công ty trong quá trình thực tập em đã tự rút ra cho mình những kinh nghiệm thực tế của môi trường làm việc, nắm được sơ lược về hoạt động của các thiết bị mà công ty sản xuất cũng như cách vận hành của các thiết bị đồng hồ đo điện . Điều quan trọng hơn hết là rèn luyện được ý thức về tác phong, đạo đức và tính kỷ luật, và cách làm việc điều đó rất có ích cho bản thân em trong khoảng thời gian sắp tới khi bước vào môi trường làm việc thực thụ.

Em xin chân thành cảm ơn các thầy cô trong khoa và anh chị tại Công ty Takemoto đã tạo điều kiện thuận lợi cho em có thể hoàn thành việc thực tập trong 6 tuần vừa qua và đặc biệt là hai anh Đỗ Thanh Tuyền và anh Trần Đình Lợi đã giúp đỡ em hoàn thành được đề tài mà công ty đã đề ra.

Trong quá trình thực tập cũng như học tập ở trường để vận dụng vào kiến thức đã học, em cảm thấy còn nhiều thiếu sót, đặc biệt là trong quá trình ứng dụng những kiến thức đã học vào quá trình thực tập, vì vậy trong báo cáo vẫn còn nhiều thiếu sót, một vài kiến thức vẫn chưa được hoàn thiện tuyệt đối, kính mong thầy cô thông cảm và bổ sung cho em để em có thể hoàn thành báo cáo một cách đầy đủ nhất.

Em xin chân thành cảm ơn.

Sinh viên

Nguyễn Quang Huy

SVTH: NGUYỄN QUANG HUY

MỤC LỤC

LỊCH THỰC TẬP.....	6
LỜI NÓI ĐẦU	8
Chương 1	11
GIỚI THIỆU CÔNG TY TNHH TAKEMOTO VIỆT NAM	11
1.1 Giới thiệu chung.....	11
1.2 Các hệ thống của công ty.	11
1.3 Giải pháp cung cấp.....	11
Chương 2	13
PHÂN TÍCH HỆ THỐNG ĐO LƯỜNG VÀ CÁC THIẾT BỊ ĐO LƯỜNG	13
1 Hệ thống giám sát lượng điện tiêu thụ trong tòa nhà.....	13
1.1 Vấn đề đặt ra	13
1.2 Giải pháp công ty đang thực hiện	13
2. Hệ thống thực hiện theo dõi nhu cầu và kiểm soát điều hòa không khí với đài LoRa!.....	15
2.1 Vấn đề đặt ra	15
2.2 Giải pháp công ty thực hiện.	15
2.3 Tổng quan về các thiết bị	18
CHƯƠNG 3	20
TÌM HIỂU PHẦN CỨNG CỦA THIẾT BỊ ĐỒNG HỒ ĐO XS-96	20
1. Tìm hiểu về chip MSP430f47176.	20
1.1 MSP430F47176	20
1.2 Các chế độ hoạt động.	21
1.3 Điện áp, dòng điện và tần số hoạt động của vi điều khiển ở từng chế độ.	22
1.4 Timer_A.	23
1.5. Giao tiếp I2C.....	26
1.6 Cấu hình giao động để vdk có thể hoạt động.....	28

BÁO CÁO THỰC TẬP TỐT NGHIỆP

2. Tìm hiểu về chip PCF8578	30
2.1 Tổng quan:	30
2.2 Các chân và chức năng.....	30
2.3 Điều khiển hiển thị.....	37
Chương 4.....	38
THỰC HIỆN ĐỀ TÀI HIỂN THỊ THỜI GIAN LÊN ĐỒNG HỒ ĐIỆN XS-96....	38
1. Tóm tắt các yêu cầu đặt ra	38
2. Thực hiện đề tài.....	38
2.1 Sơ đồ khối tổng quan của đề tài.....	38
2.2 Sơ đồ minh họa cho việc kết nối trên thiết bị phần cứng	39
2.3 Các bước thực hành cho việc viết chương trình.	40
2.3.1 Cấu hình bộ giao động.	40
2.3.2 Khởi tạo bộ I2C.....	41
2.3.4 Tiến hành hiển thị những kí tự theo nguyên vọng	44
2.3.5 Khởi tạo bộ TIMER A0	49
2.3.6 Thao tác nút bấm.....	51
2.3.7 Hàm xử lí thời gian	53
Chương 5	56
KẾT LUẬN.....	56
1. Những kinh nghiệm và kiến thức học được trong quá trình thực tập.....	56
2. Các khó khăn khi gặp phải.....	57
Tài liệu tham khảo.....	57

Chương 1

GIỚI THIỆU CÔNG TY TNHH TAKEMOTO VIỆT NAM

1.1 Giới thiệu chung

Công ty TNHH TAKEMOTO Việt Nam, tên quốc tế **TAKEMOTO VietNam Company Limited** và tên đăng ký là **Công ty TNHH TAKEMOTO Việt Nam**, đã hoạt động hơn 8 năm trong lĩnh vực kinh tế Tư vấn máy vi tính, quản trị hệ thống máy vi tính, Công ty có địa chỉ: 1254 Xô Viết Nghệ Tĩnh, Phường Hoà Cường Nam, Quận Hải Châu, Thành phố Đà Nẵng. Là công ty con của công ty Hakaru+.

1.2 Các hệ thống của công ty

- + *Hệ thống đo lường*: Phát triển và sản xuất các thiết bị đo để phục vụ các lĩnh vực trong đời sống.
- + *Hệ thống bê tông trộn sẵn*: Đưa ra các giải pháp, thiết bị ứng dụng vào lĩnh vực xây dựng giúp giảm thiểu tai nạn lao động, giám sát tự động.
- + *Hệ thống cân*: Tùy chỉnh và hỗ trợ thiết kế và sản xuất hệ thống cân và chiết rót tối ưu dựa trên các thông số kỹ thuật, môi trường và cách bố trí cần thiết cho các nguyên liệu thô dạng bột và lỏng.
- + *Thiết bị chăm sóc y tế*: Phát triển và sản xuất các thiết bị kết nối giữa nhân viên chăm sóc và bệnh nhân, các thiết bị chăm sóc đặc biệt cho bệnh nhân.

1.3 Giải pháp cung cấp

Công ty phát triển và cung cấp nhiều loại sản phẩm khác nhau của Hakaru cho nhiều thị trường khác nhau như đo lường, thiết bị đo đạc (bê tông), cân và thiết bị phục vụ cho y tế .

Ngoài những công nghệ trên, công ty đã tiếp thu những công nghệ mới và đang làm việc chăm chỉ mỗi ngày để đáp ứng sự mong đợi của khách hàng.

Các lĩnh vực mà công ty phát triển:

- + *Đo đạc* : Thiết kế và cung cấp các thiết bị đo công suất đo tần số, các cảm biến nhiệt độ, cảm biến gia tốc , thiết bị cảm biến sóng siêu âm ...vv
- + *Mạch điện* : Thiết kế mạch điện tương tự, mạch đầu ra âm thanh, mạch tăng áp, mạch nguồn, mạch hiển thị.

BÁO CÁO THỰC TẬP TỐT NGHIỆP

- + Sự thi công: công nghệ thiết kế chống nhỏ giọt, thiết kế vỏ/ bộ phận kim loại, thiết kế vỏ nhựa cho các thiết bị.
- + Phần mềm nhúng : kiểm soát máy vi tính, kho , kiểm soát đầu ra.
- + Thiết bị cung cấp: Máy khuấy, thiết kế thiết bị ngăn chặn sự kết dính, cân xe đẩy di động, thang đo nền tảng kiểu đòn bẩy, máy đo cân nặng.
- + Phần mềm kiểm soát: thiết kế bảng điều khiển phần mềm cho các thiết bị.
- + Liên lạc: Giao tiếp PLC, truyền thông nối tiếp, các thiết bị không dây, giao tiếp đường dây điện, giao tiếp quang học, IoT LPWA, GPS/GNSS.
- + Y tế: Máy thu di động, phát triển một cơ chế sử dụng "cảm biến siêu âm" làm cảm biến chuyển động trong các cơ sở y tế và chăm sóc dài hạn yêu cầu "theo dõi" và thương mại hóa nó như một hệ thống phát hiện rời giường.

Chương 2

PHÂN TÍCH HỆ THỐNG ĐO LƯỜNG VÀ CÁC THIẾT BỊ ĐO LƯỜNG

1 Hệ thống giám sát lượng điện tiêu thụ trong tòa nhà.

1.1 Vấn đề đặt ra

Nhằm nhu cầu giám sát điện năng cùng một lúc trong các tòa nhà các cơ sở hạ tầng và việc thực hiện này có thể thực hiện được trên một máy tính duy nhất hay không. Để giám sát điện năng một cách chi tiết mọi thông số và thông tin sẽ được hiển thị trên máy tính, tối ưu hóa và tiết kiệm năng lượng và giảm thiểu thấp nhất rủi ro có thể xảy ra về các vấn đề an toàn điện.

1.2 Giải pháp công ty đang thực hiện

Ở thời điểm hiện tại công ty đã đề xuất thiết bị giám sát nhu cầu CSA-109-T và bộ chuyển đổi đa năng công suất TWPM làm công cụ đo lường và lưu dữ liệu TPS-10 làm phần mềm giám sát công suất.

Vì thiết bị theo dõi có nhu cầu giám sát từ xa nên được trang bị chức năng máy chủ web nên có thể được giám sát đơn giản bằng cách kết nối thiết bị với máy tính cá nhân.

Ngoài ra có thể thống kê và đưa ra các biểu đồ để đánh giá việc tiêu thụ điện năng và đưa ra các biểu mẫu báo cáo hàng tháng.

Dưới đây sơ đồ của hệ thống:



Các thiết bị được sử dụng trong hệ thống :

Đồng hồ đa năng TWPM	
Đồng hồ đa chức năng đo 5 kênh (5CT - bộ biến dòng), dùng chung VT-bộ biến áp.	
Máy chì báo nhu cầu CSA-109-D	
Thiết bị giám sát nhu cầu CSA-109-T	

1.3 Tổng quan về các thiết bị

Đồng hồ đo đa năng TWPM dùng trong hệ thống giám sát năng lượng dữ liệu chủ yếu được giám sát qua phần mềm nên chỉ cần hiển thị LED 7 đoạn.

Đồng hồ đa chức năng đo 5 kênh loại đo lường công suất TWP5M nó đo các dữ liệu khác nhau của mạch điện như điện áp, dòng điện và năng lượng điện của 5 bộ cấp nguồn của cùng một hệ thống, và gửi nó đến máy tính phía trên bằng giao tiếp RS-485 hoặc giao tiếp Modbus. Màn hình tinh thể lỏng 20 ký tự x2 cột được sử

BÁO CÁO THỰC TẬP TỐT NGHIỆP

dụng cho màn hình. Bạn có thể hiển thị giá trị đo của tối đa 3 phần tử cùng lúc trên một màn hình.

Thiết bị giám sát nhu cầu CSA-109-T thiết bị này nhận tín hiệu xung từ wattmeter, đo công suất nhu cầu và thông báo cho người dùng khi nhu cầu sắp bị vượt quá. Bằng cách kết hợp với màn hình nhu cầu tùy chọn (CSA-109-D) và thiết bị cảnh báo nhu cầu (TDD3ZB-R), đầu ra cảnh báo có thể được thực hiện thông qua dòng đèn hiện có. Ngoài ra, khi kết hợp với bộ điều khiển điều hòa (TDD8EPT), có thể tự động điều khiển dàn nóng khi vượt quá giá trị nhu cầu sử dụng.

- Có thể giảm chi phí xây dựng bằng cách sử dụng các đường dây đèn và đường dây điện hiện có.
- Bổ sung chức năng máy chủ WEB
- Bằng cách sử dụng chức năng giao tiếp Ethernet, bạn có thể kiểm tra dữ liệu đo lường và trạng thái cảnh báo của thiết bị từ trình duyệt web của máy tính cá nhân hoặc thiết bị đầu cuối máy tính bằng thông qua mạng LAN hiện có.
- Bổ sung chức năng xuất tệp vào bộ nhớ USB Bằng cách kết nối bộ nhớ USB với thiết bị chính, dữ liệu được ghi bởi thiết bị chính có thể được xuất ra dưới dạng tệp.

Máy chỉ báo nhu cầu CSA-109-D : Thiết bị hiển thị Demand, hiển thị và cảnh báo sớm nếu năng lượng dùng có xu hướng quá mức thiết lập

2. Hệ thống thực hiện theo dõi nhu cầu và kiểm soát điều hòa không khí với đài LoRa!

2.1 Vấn đề đặt ra

Nhằm mục đích tiết kiệm năng lượng giảm thiểu điện năng tiêu thụ các nhà máy và tự động điều khiển các thiết bị máy điều hòa không khí.

2.2 Giải pháp công ty thực hiện.

Sử dụng công nghệ lora nhằm mục đích tiết kiệm năng lượng, là kết nối không dây giảm chi phí lắp đặt ban đầu.

Thiết bị giám sát nhu cầu và thiết bị điều khiển điều hòa không khí được liên kết với tín hiệu giao tiếp không dây với đài LoRa .

BÁO CÁO THỰC TẬP TỐT NGHIỆP

Đầu tiên, tín hiệu RS-485 (Modbus) của thiết bị theo dõi nhu cầu được lắp đặt trong thiết bị nhận và chuyển đổi nguồn được thực hiện không dây bởi đài LoRa và được giám sát bởi màn hình hiển thị nhu cầu được lắp đặt trong văn phòng .

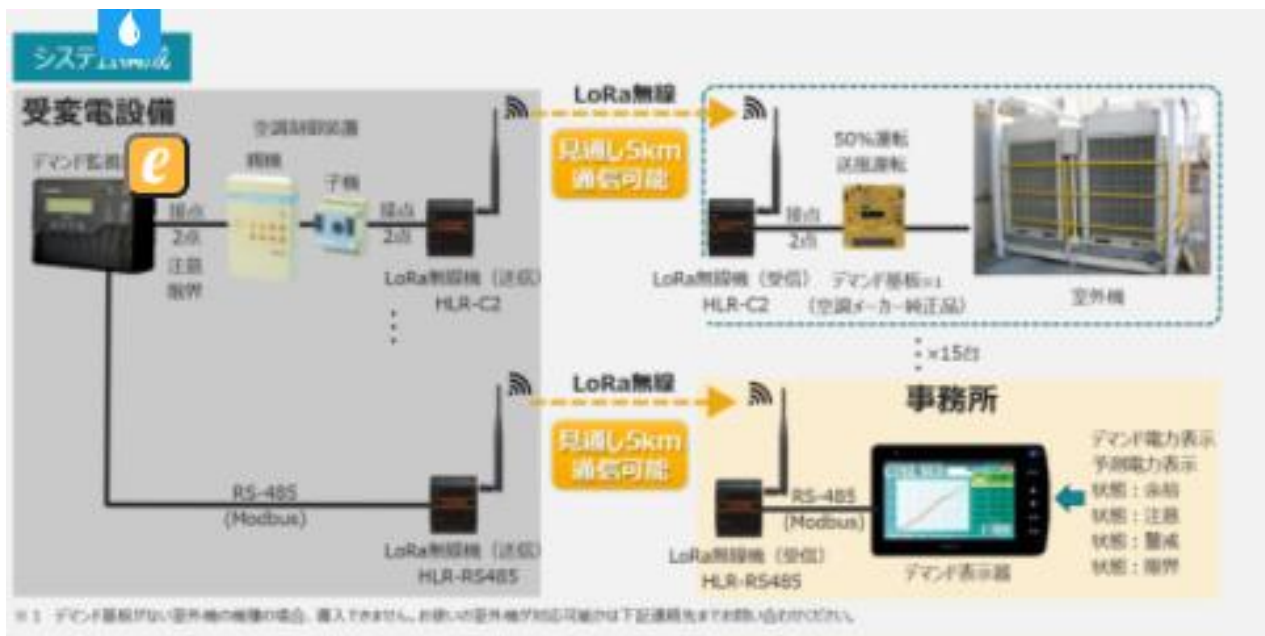
Hiển thị công suất nhu cầu, hiển thị công suất dự đoán, hiển thị trạng thái (biên, giới hạn) được hiển thị dưới dạng các mục giám sát và trạng thái nhu cầu có thể được theo dõi trong thời gian thực.

Ngoài ra, đầu ra cảnh báo nhu cầu (thận trọng / giới hạn) từ thiết bị giám sát nhu cầu được đưa vào thiết bị chính của thiết bị điều khiển máy điều hòa không khí và tín hiệu điều khiển được gửi đến máy điều hòa không khí tại chỗ trong chế độ truyền không dây của thiết bị vô tuyến LoRa. (Vận hành / vận hành quạt gió) được gửi để thực hiện hoạt động tiết kiệm năng lượng của máy điều hòa không khí .

Do đó, thiết bị giám sát nhu cầu thiết bị điều khiển điều hòa không khí có thể được kết nối mà không cần nối dây bằng cách sử dụng đài LoRa.






Ngoài ra, bằng cách liên kết thiết bị giám sát nhu cầu và thiết bị điều khiển điều hòa, có thể giảm hóa đơn tiền điện qua từng năm.


Sơ đồ của hệ thống:



BÁO CÁO THỰC TẬP TỐT NGHIỆP

Các thiết bị của hệ thống:

Mô hình tiếp điểm hai điểm HLR-C2	
Kiểu giao tiếp RS485 HLR-RS485	
Thiết bị giám sát nhu cầu CSA-109-T	
Máy chỉ báo nhu cầu CSA-109-D	
Thiết bị một chiều (thiết bị chính) TDD8EP-T	

Thiết bị một chiều (đơn vị phụ) TDD2EP-R	
--	---

2.3 Tổng quan về các thiết bị

Tiếp điểm vô tuyến LoRa kiểu 2 điểm HLR-C2: có thể thực hiện đầu vào / đầu ra tiếp xúc 2 điểm với một thiết bị và có thể thực hiện truyền dữ liệu đến cổng kết nối IoT và giám sát tín hiệu liên lạc qua sóng vô tuyến LoRa. (Đếm đếm / trạng thái / thời gian BẬT). Ngoài ra, HLR-C2 có thể truyền tín hiệu đầu vào của tiếp điểm tới HLR-C2 bằng đài LoRa bằng cách sử dụng chế độ truyền không dây và có thể xuất tiếp điểm tới HLR-C2 ở một nơi xa. Ngoài ra, chức năng này có thể truyền tín hiệu đầu vào tiếp xúc với một HLR-C2 đến nhiều thiết bị (lên đến 50) cùng một lúc bằng không dây LoRa.. Đối với giao tiếp không dây, LoRa là một trong những công nghệ không dây cho IoT, được sử dụng. Do đó, có thể giao tiếp với tầm xa tối đa là 5km cho đến cổng kết nối IoT HLR-GW, là thiết bị chính và HLR-C2, là một cặp để liên lạc vô tuyến.

Thiết bị không dây LoRa Kiểu giao tiếp RS-485 HLR-RS485 có thể chuyển đổi giao tiếp RS-485 (tương thích với Modbus) sang không dây LoRa và truyền dữ liệu đến hệ thống chủ bằng giao tiếp không dây. Nó có thể được kết nối với các thiết bị sau của công ty theo tiêu chuẩn và nhiều dữ liệu khác nhau có thể được truyền qua mạng không dây LoRa và được giám sát bởi cổng IoT.

- Đồng hồ vạn năng điện tử: XM2 / XS2
- Bộ chuyển đổi năng lượng điện: TWPS
- Bộ chuyển đổi đa nguồn: TWPM / TWP5M
- Bộ chuyển đổi đầu vào xung: TWPP
- Bộ chuyển đổi đầu vào tiếp xúc: TWP8C

Ngoài ra, bằng cách sử dụng thiết bị này như một cặp trong chế độ truyền không dây, tín hiệu RS-485 (tương thích với Modbus) có thể được truyền không

BÁO CÁO THỰC TẬP TỐT NGHIỆP

dây bằng LoRa và tín hiệu thu được có thể được chuyển đổi thành RS-485 và được kết nối với một thiết bị như PLC. Đối với giao tiếp không dây, LoRa là một trong những công nghệ không dây cho IoT, được sử dụng. Bộ đàm có thể tiếp cận thậm chí ở những nơi không thể thực hiện được và được khách hàng đánh giá rất cao.

Thiết bị điều khiển điều hòa một chiều (thiết bị chính) TDD8EP-T thiết bị điều khiển điều hòa là một hệ thống điều khiển hỗ trợ tiết kiệm năng lượng sử dụng chức năng theo yêu cầu của dàn nóng điều hòa (do nhà sản xuất điều hòa không khí sản xuất).

Hai phương pháp điều khiển, cài đặt ưu tiên và điều khiển xoay, giúp tiết kiệm năng lượng trong khi duy trì môi trường điều hòa không khí thoải mái.

Công nghệ giao tiếp đường dây điện (PLC) được áp dụng giao tiếp giữa thiết bị chính và thiết bị phụ sử dụng đường dây điện hiện có. Vì không cần lắp đặt thiết bị liên lạc mới, chi phí xây dựng và đi dây có thể giảm đáng kể.

Kết hợp với thiết bị theo dõi nhu cầu hiện có là được cũng có thể sử dụng kết hợp với thiết bị theo dõi nhu cầu đã được lắp đặt sẵn (sản phẩm của các công ty khác cũng có thể). Bằng cách tự động hóa điều khiển điều hòa không khí cùng với việc theo dõi nhu cầu, có thể đạt được mức tiết kiệm năng lượng và tiết kiệm lao động một cách đáng tin cậy hơn.

- Có 6kHz và 9kHz.
- Có 200V và 100V

Thiết bị điều khiển điều hòa Một chiều (đơn vị phụ) TDD2EP-R cảnh báo do thiết bị điều khiển điều hòa thiết bị chính nhận được từ thiết bị giám sát nhu cầu dưới dạng tín hiệu liên lạc bằng giao tiếp đường dây điện.

Bộ phận phụ đưa tín hiệu này vào bảng cầu như một tiếp điểm. Bảng cầu điều khiển đầu ra của dàn nóng theo tiếp điểm này, giúp tiết kiệm điện năng hoạt động. Vì đầu ra được điều khiển thay vì BẬT / TẮT, máy điều hòa không khí trong phòng có thể tiết kiệm năng lượng một cách thoải mái và hiệu quả.

Con lại hai thiết bị Thiết bị giám sát nhu cầu CSA-109-T và Chỉ báo nhu cầu CSA-109-D chứng năng tương tự như hệ thống ở trên.

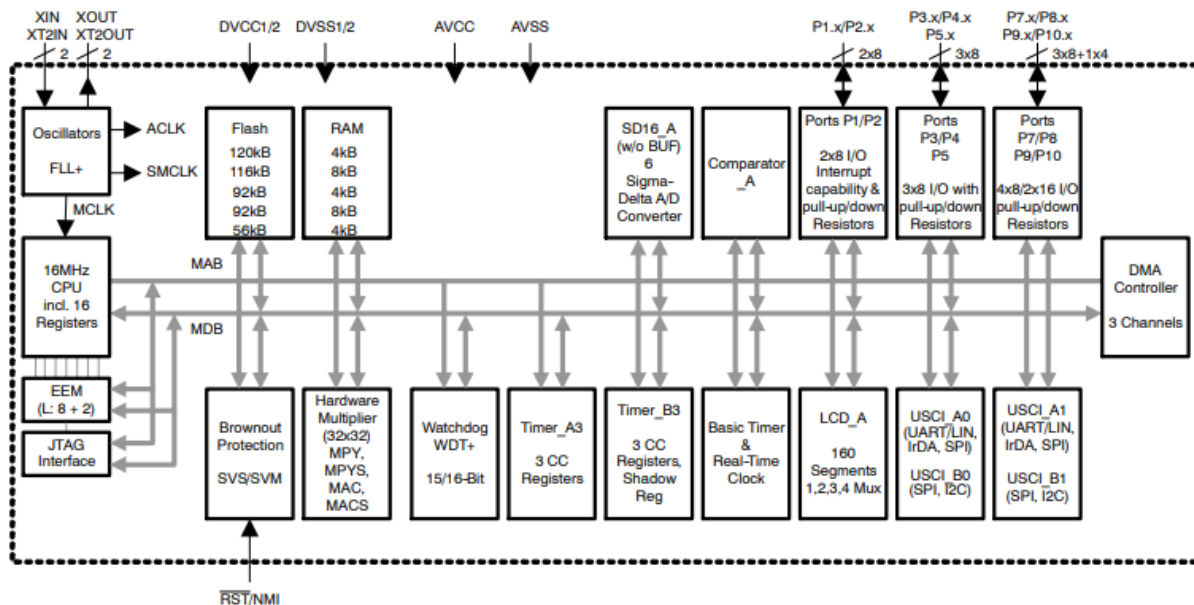
CHƯƠNG 3

TÌM HIỂU PHẦN CỨNG CỦA THIẾT BỊ ĐỒNG HỒ ĐO XS-96

1. Tìm hiểu về chip MSP430F47176.

1.1 MSP430F47176: 16-Bit RISC Architecture, 62.5-ns Instruction Cycle Time (RISC là một phương pháp thiết kế các bộ vi xử lý (VXL) theo hướng đơn giản hóa tập lệnh, trong đó thời gian thực thi tất cả các lệnh đều như nhau. Hiện nay các bộ vi xử lý RISC phổ biến là ARM, SuperH, MIPS, SPARC, DEC Alpha, PA-RISC, PIC, và PowerPC của IBM). Bộ nhớ 92KB Flash, 8KB RAM, 6 Sigma-Delta ADCs (6 chân analog).

functional block diagram, MSP430F471x6



Sơ đồ khối của vi điều khiển

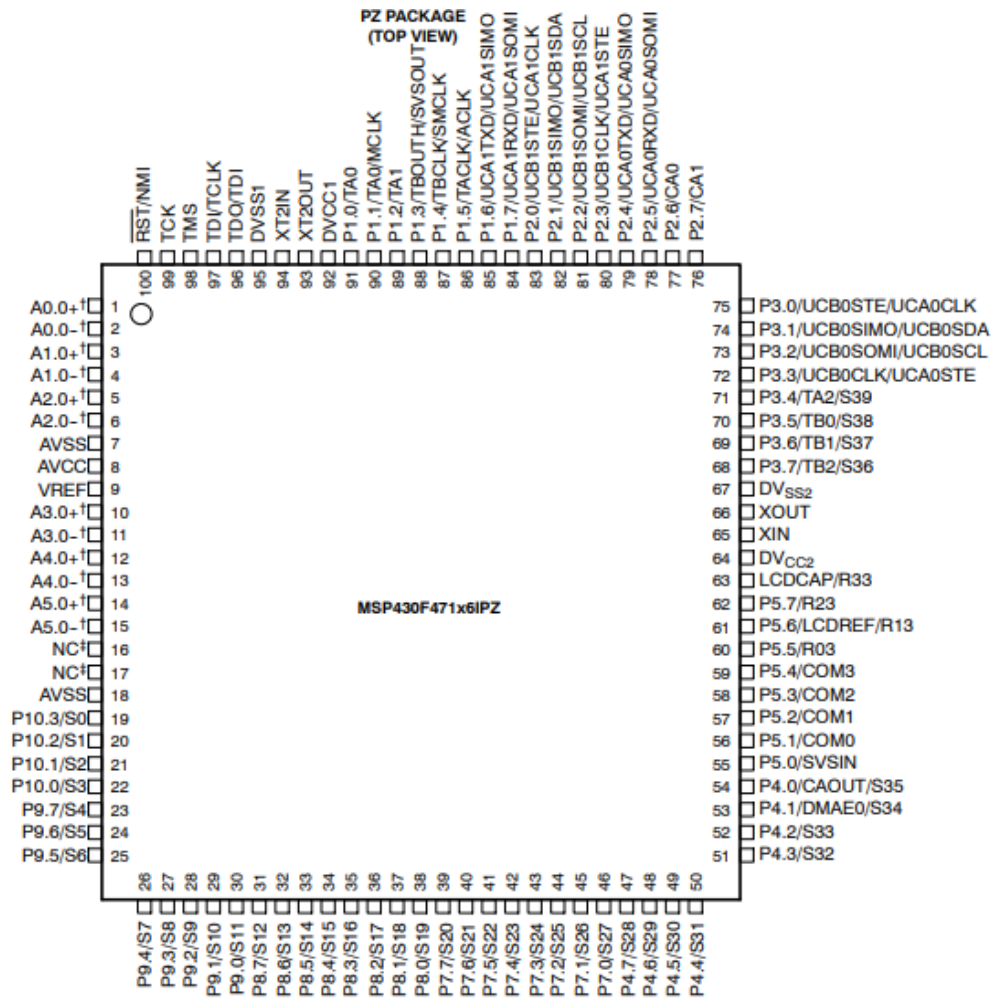
+ Dòng MSP430F471xx là các cấu hình vi điều khiển nhằm mục tiêu đến các đồng hồ đo điện một pha và nhiều pha với ba, sáu bộ chuyển đổi A/D 16 bit sigma-delta. Mỗi kênh có một cặp đầu vào vi sai và độ lợi đầu vào có thể lập trình được.

+ Ngoài ra còn được tích hợp hai bộ định thời 16 bit (Timer_A, Timer_B), bốn giao diện truyền thông nối tiếp đa năng (USCI_A0 and USCI_A1 và USCI_B0 and USCI_B1), DMA, và một trình điều khiển tinh thể lỏng (LCD) với điều khiển độ tương phản tích hợp.

+ Five Power-Saving Modes (5 chế độ tiết kiệm năng lượng).

BÁO CÁO THỰC TẬP TỐT NGHIỆP

- + Wake-Up From Standby Mode in Less Than 6 μ s (khoảng thời gian chuyển từ trạng thái ngủ sang chế độ hoạt động chỉ trong vòng 6us),
- + Low Supply-Voltage Range: 1.8 V to 3.6 V
- + MSP430F47176 có 68 chân GPIO pin, trong đó chín cổng I/O 8 bit được triển khai cổng P1 đến P5 và P7 đến P10.
- + Có 6 cặp chân ADC từ chân (1-6 và từ 10-17) trên vi điều khiển .
- + Có 4 bộ giao tiếp SPI, 2 bộ giao tiếp I2C, 2 bộ giao tiếp UART.



Các chân trên vi điều khiển

1.2 Các chế độ hoạt động.

Có tất cả 6 chế độ hoạt động gồm :

BÁO CÁO THỰC TẬP TỐT NGHIỆP

- + Chế độ AM : lúc này tất cả các bộ tạo xung clock hoạt động.
- + Chế độ năng lượng thấp 0: CPU ngắt kết nối, ACLK và SMCLK hoạt động MCLK không hoạt động, FLL + Điều khiển vòng lặp vẫn hoạt động.
- + Chế độ năng lượng thấp 1: CPU ngắt kết nối, ACLK và SMCLK hoạt động MCLK không hoạt động, FLL + Điều khiển vòng lặp không hoạt động.
- + Chế độ năng lượng thấp 2: CPU ngắt kết nối, MCLK và FFL + điều khiển vòng lặp và DCOCLK không hoạt động, bộ tạo giao đồng nội DCO's vẫn hoạt động, ACLK vẫn hoạt động.
- + Chế độ năng lượng thấp 3: CPU ngắt kết nối, MCLK và FLL + điều khiển vòng lặp và DCOCLK không hoạt động, bộ tạo giao đồng nội DCO's không hoạt động, ACLK vẫn hoạt động.
- + Chế độ năng lượng thấp 4: CPU ngắt kết nối, ACLK ngắt kết nối, MCLK và FLL + điều khiển vòng lặp và DCOCLK không hoạt động, DCO's không hoạt động. Bộ giao đồng thạch anh tạm dừng.

1.3 Điện áp, dòng điện và tần số hoạt động của vi điều khiển ở từng chế độ.

		MIN	NOM	MAX	UNIT
Supply voltage during program execution, V_{CC} ($AV_{CC} = DV_{CC} = V_{CC}$) (see Note 1)		1.8		3.6	V
Supply voltage during program execution, SVS enabled, PORON = 1, V_{CC} ($AV_{CC} = DV_{CC} = V_{CC}$) (see Notes 1, 2)		2.0		3.6	V
Supply voltage during program/erase flash memory, V_{CC} ($AV_{CC} = DV_{CC} = V_{CC}$) (see Note 1)		2.2		3.6	V
Supply voltage, V_{SS}			0		V
Operating free-air temperature range, T_A		-40		85	°C
Processor frequency f_{SYSTEM} (Maximum MCLK frequency) (see Notes 3, 4 and Figure 1)	$V_{CC} = 1.8$ V, Duty cycle = 50% \pm 10%	dc		4.15	MHz
	$V_{CC} = 2.2$ V, Duty cycle = 50% \pm 10%	dc		7.5	MHz
	$V_{CC} = 2.7$ V, Duty cycle = 50% \pm 10%	dc		12	MHz
	$V_{CC} \geq 3.3$ V, Duty cycle = 50% \pm 10%	dc		16	

PARAMETER	TEST CONDITIONS	V _{CC}	MIN	TYP	MAX	UNIT
I _(AM) Active mode, (see Note 1) f _(MCLK) = f _(SMCLK) = 1 MHz, f _(ACLK) = 32768 Hz XTS = 0, SELM = (0,1) (Program executes from flash)	T _A = -40°C to 85°C	2.2 V		350	400	μA
		3 V		500	560	
I _(LPM0) Low-power mode, (LPM0) (see Notes 1, 4)	T _A = -40°C to 85°C	2.2 V		45	70	μA
		3 V		75	110	
I _(LPM2) Low-power mode, (LPM2), f _(MCLK) = f _(SMCLK) = 0 MHz, f _(ACLK) = 32768 Hz, SCG0 = 0 (see Notes 2, 4)	T _A = -40°C to 85°C	2.2 V		11	14	μA
		3 V		17	22	
I _(LPM3) Low-power mode, (LPM3) f _(MCLK) = f _(SMCLK) = 0 MHz, f _(ACLK) = 32768 Hz, SCG0 = 1 Basic Timer1 and RTC enabled, ACLK selected LCD_A enabled, LCDCPEN = 0: (static mode, f _{LCD} = f _(ACLK) /32) (see Notes 2, 3, and 4)	T _A = -40°C	2.2 V		0.7	2.0	μA
	T _A = 25°C			0.8	2.0	
	T _A = 60°C			2.0	3.5	
	T _A = 85°C			5.0	9.5	
	T _A = -40°C	3 V		1.1	3.0	μA
	T _A = 25°C			1.2	3.0	
	T _A = 60°C			2.5	4.0	
	T _A = 85°C			6.0	10.0	
I _(LPM3) Low-power mode, (LPM3) f _(MCLK) = f _(SMCLK) = 0 MHz, f _(ACLK) = 32768 Hz, SCG0 = 1 Basic Timer1 and RTC enabled, ACLK selected LCD_A enabled, LCDCPEN = 0: (4-mux mode, f _{LCD} = f _(ACLK) /32) (see Notes 2, 3, and 4)	T _A = -40°C	2.2 V		3.5	5.5	μA
	T _A = 25°C			3.5	5.5	
	T _A = 60°C			5.5	7.0	
	T _A = 85°C			11.0	17.0	
	T _A = -40°C	3 V		4.0	6.5	μA
	T _A = 25°C			4.0	6.5	
	T _A = 60°C			6.0	8.0	
	T _A = 85°C			13.0	20.0	
I _(LPM4) Low-power mode, (LPM4) f _(MCLK) = 0 MHz, f _(SMCLK) = 0 MHz, f _(ACLK) = 0 Hz, SCG0 = 1 (see Notes 2 and 4)	T _A = -40°C	2.2 V		0.1	1.0	μA
	T _A = 25°C			0.2	1.0	
	T _A = 60°C			1.0	2.5	
	T _A = 85°C			4.5	8.5	
	T _A = -40°C	3 V		0.1	2.0	μA
	T _A = 25°C			0.2	2.0	
	T _A = 60°C			1.5	3.0	
	T _A = 85°C			5.0	9.0	

1.4 Timer_A.

- + Timer_A là một bộ định thời / bộ đếm 16 bit với ba thanh ghi lưu / so sánh.
- + Timer_A có thể hỗ trợ nhiều lưu /so sánh, đầu ra PWM và định thời gian.
- + Timer_A cũng có khả năng ngắt rộng rãi. Ngắt có thể được tạo ra từ bộ đếm trong điều kiện tràn và từ mỗi thanh ghi lưu /so sánh.

+ Bộ định thời có thể được lấy nguồn từ ACLK, SMCLK hoặc bên ngoài thông qua TACLK hoặc INCLK. Nguồn xung nhịp được chọn với các bit TASSELx. Nguồn xung nhịp đã chọn có thể được chuyển trực tiếp đến bộ định thời hoặc chia cho 2, 4 hoặc 8 bằng cách sử dụng các bit IDx. Bộ chia tần được đặt lại khi TACLK được đặt.

BÁO CÁO THỰC TẬP TỐT NGHIỆP

Nguồn xung Clock: Timer A có thể hoạt động với 1 trong 4 nguồn xung clock sau, tùy thuộc vào yêu cầu của ứng dụng cần thực hiện.

- TACLK (Timer A clock): Là một nguồn xung clock ngoại
- ACLK (Auxiliary clock): Là một nguồn clock bổ sung cho Timer A, chỉ được cấp nguồn từ LFXTCL, thường dùng cho hệ thống phụ với clock hoạt động thấp để tiết kiệm năng lượng.
- SMCLK (Submaster clock): thường dùng cho ngoại vi, lấy nguồn từ DCO hoặc XT2.

+ Các chế độ đếm của timer: chế độ dừng, đếm lên, liên tục, và cuối cùng là lên và xuống .

MCx	Mode	Description
00	Stop	The timer is halted.
01	Up	The timer repeatedly counts from zero to the value of TACCR0.
10	Continuous	The timer repeatedly counts from zero to 0FFFFh.
11	Up/down	The timer repeatedly counts from zero up to the value of TACCR0 and back down to zero.

+ Stop: tắt Timer A.

+ Up: Chế độ đếm lên của Timer A. Giá trị của thanh ghi TAR (Timer A register) sẽ tăng liên tục từ 0 cho đến một giá trị được định sẵn trong thanh ghi TACCR0 rồi reset về giá trị 0, quá trình hoạt động cứ tiếp tục như vậy.

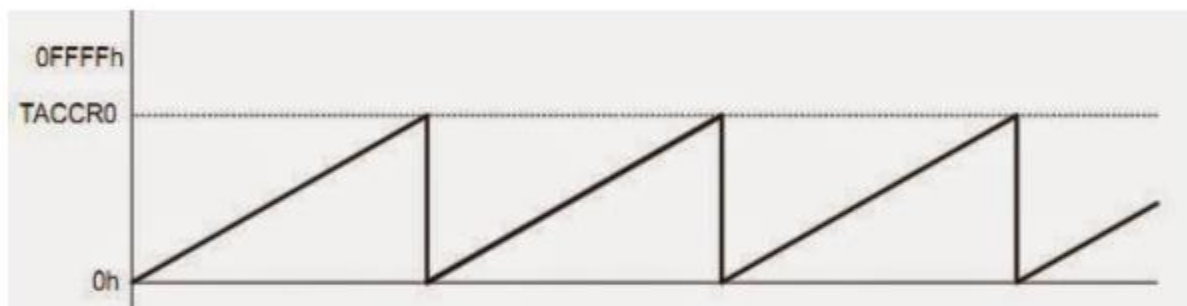
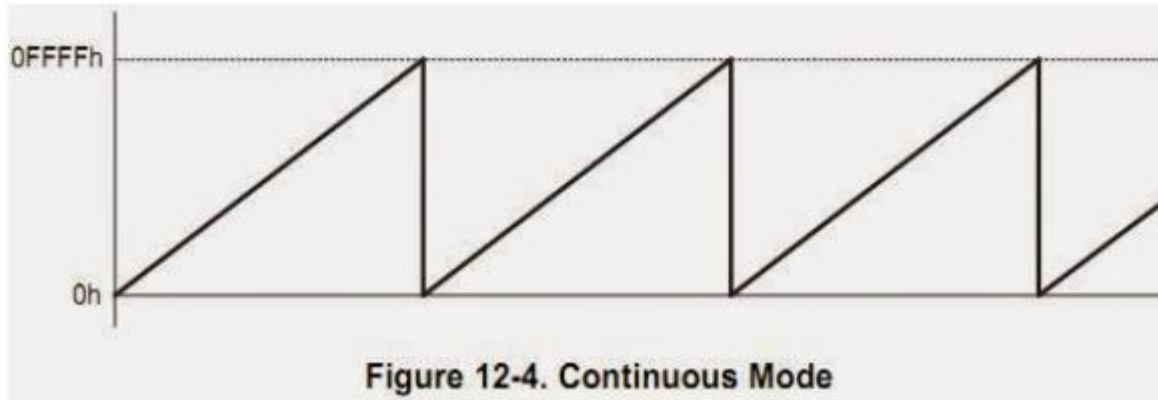


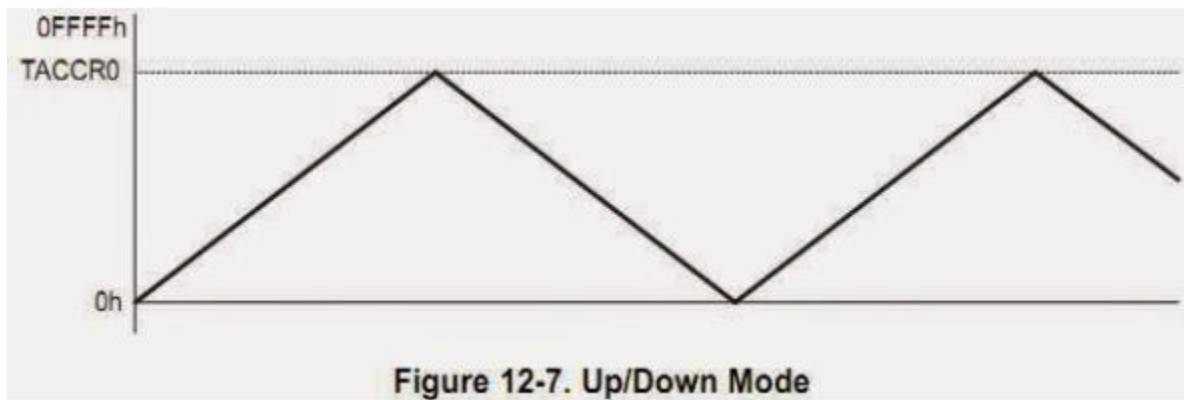
Figure 12-2. Up Mode

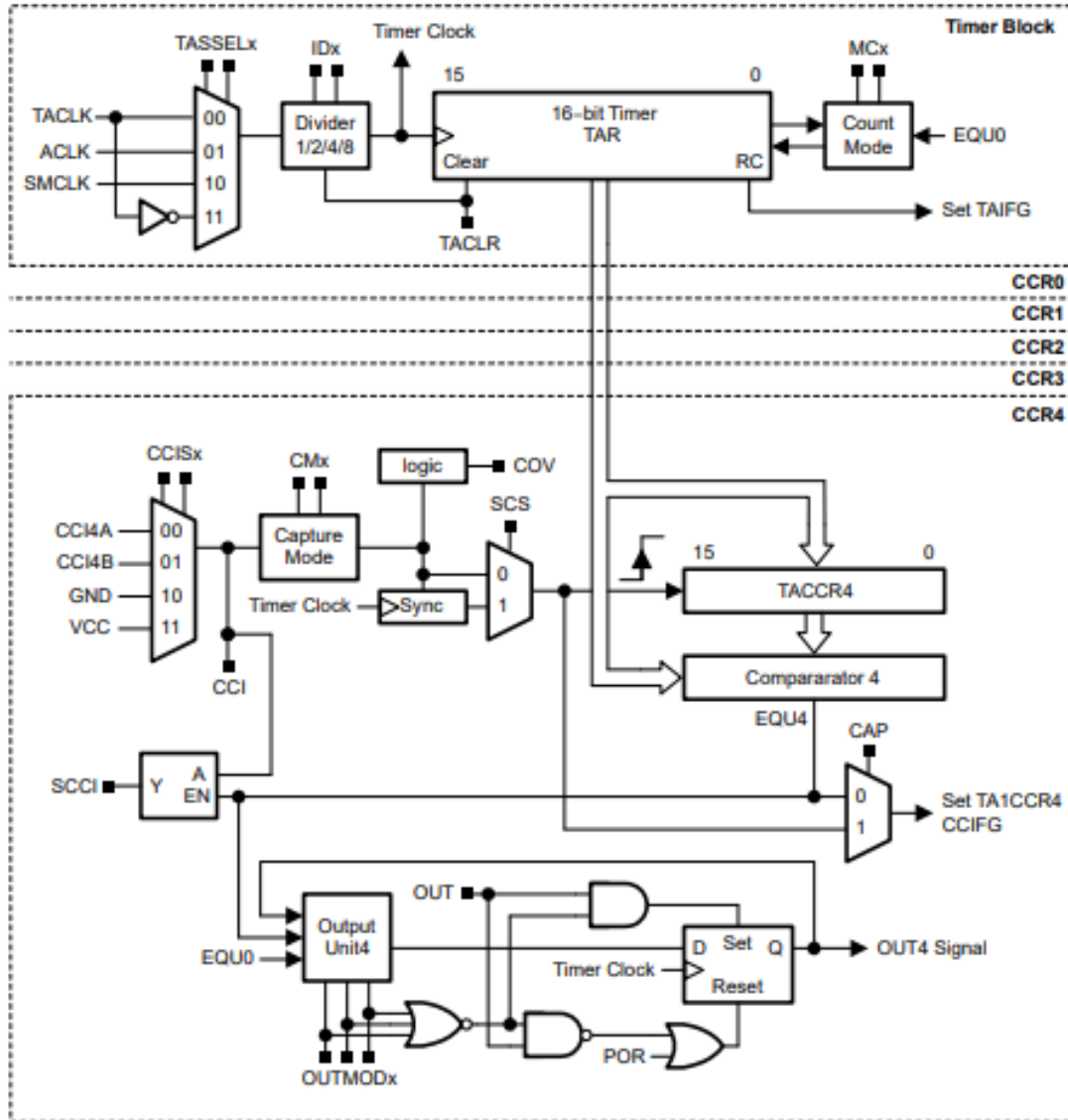
BÁO CÁO THỰC TẬP TỐT NGHIỆP

+ Continuous: Chế độ đếm lên liên tục của Timer A. Trong một chu kỳ hoạt động giá trị của thanh ghi TAR sẽ tăng liên tục từ 0 cho đến 0FFFFh rồi reset về giá trị 0.



+ Up/Down: Chế độ đếm lên/xuống của Timer A. Trong một chu kỳ hoạt động giá trị của thanh ghi TAR sẽ tăng liên tục từ 0 cho đến TACCR0 rồi giảm dần về giá trị 0.





Sơ đồ khối bộ timer A

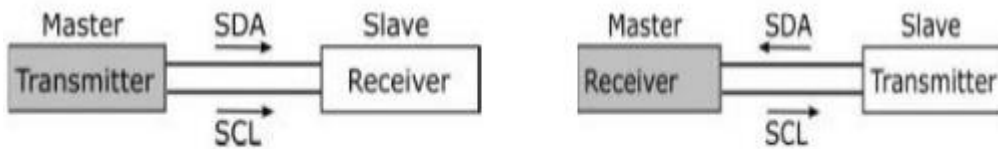
1.5. Giao tiếp I2C

+ 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).

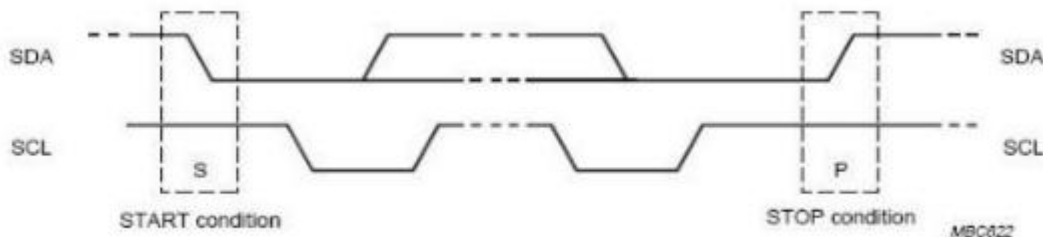
+ Đường truyền xung đồng hồ SCL là đường 1 hướng còn đường truyền dữ liệu SDA lại 2 chiều. Như vậy khi kết nối với thiết bị ngoại vi thì đường ra/vào dữ liệu sẽ nối với SDA của I2C và đường vào xung đồng hồ của ngoại vi sẽ nối với SCL. 2 đường dây SDA và SCL của I2C sẽ được đấu với 1 điện trở kéo nối với đầu (+) của nguồn DC và thường có giá trị là 1k-4.7kOhm. Dù các thiết bị ngoại vi được

BÁO CÁO THỰC TẬP TỐT NGHIỆP

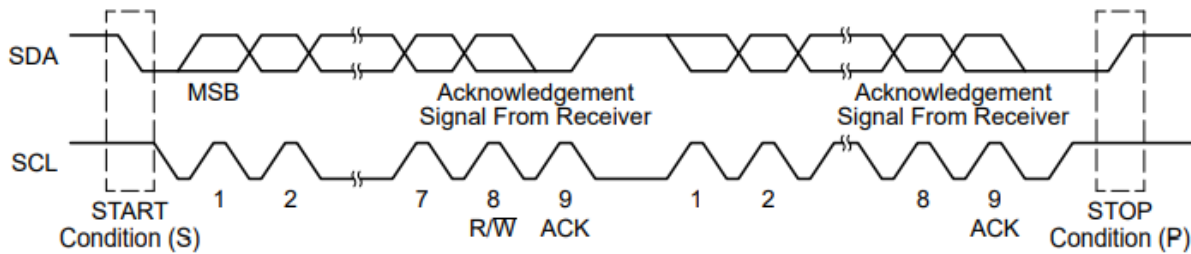
sử dụng chung 1 đường dây nhưng sẽ không xảy ra tình huống nhầm lẫn tín hiệu giữa các thiết bị. Vì trong suốt cả quá trình, mỗi thiết bị được định hình với 1 địa chỉ duy nhất với một quan hệ master/slave. Và hoạt động truyền hay nhận của từng thiết bị phụ thuộc vào tính chất của nó là master hay slave. Trong quá trình giao tiếp, master sẽ là thiết bị phát ra xung đồng hồ và quản lý 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 giữ 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 xảy 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(Acknowledge) 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 kỳ xung của SCL, I2C đã truyền đi được 8 bits dữ liệu và bên SDA sẽ tạo một xung thấp “0” tại chu kỳ 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.



Quá trình truyền dữ liệu

1.6 Cấu hình giao động để vdk có thể hoạt động.

+ LFXT1CLK: Bộ dao động tần số thấp / tần số cao có thể được sử dụng với giao động tần số thấp 32768 Hz hoặc bộ giao động tiêu chuẩn hoặc bộ cộng hưởng trong phạm vi 450 kHz đến 8 MHz.

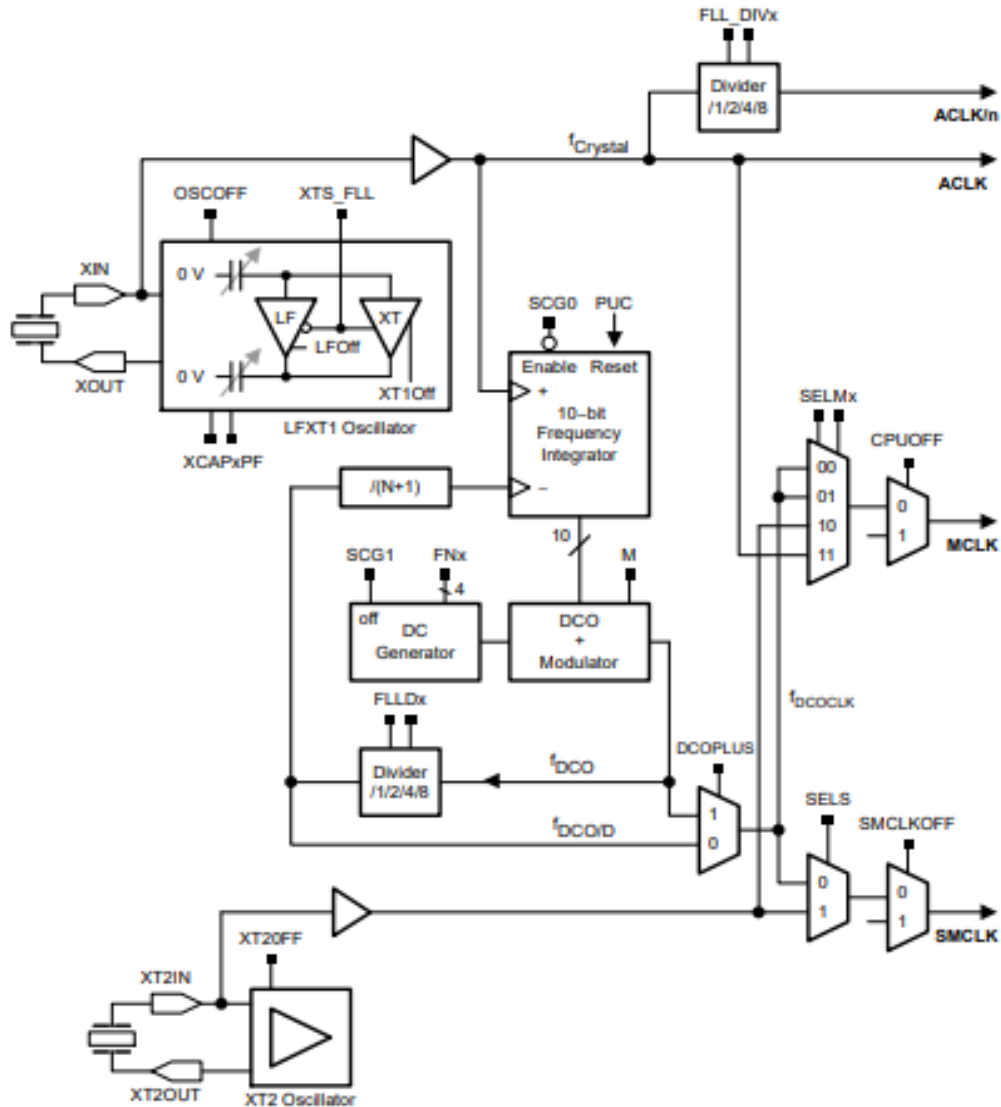
+ XT2CLK: Bộ dao động tần số cao tùy chọn có thể được sử dụng với các bộ giao động thạch anh, bộ cộng hưởng hoặc nguồn xung nhịp bên ngoài trong dải tần 450 kHz đến 8 MHz. Trong thiết bị MSP430F47x3/4 và MSP430F471xx, giới hạn trên là 16 MHz.

+ DCOCLK: Bộ dao động điều khiển kỹ thuật số bên trong (DCO) với các đặc tính kiểu RC, được ổn định bởi FLL.

+ ACLK: Xung clock phụ. ACLK là phần mềm có thể chọn làm LFXT1CLK hoặc VLOCLK làm xung clock nguồn. ACLK có thể lựa chọn cho các mô-đun ngoại vi riêng lẻ.

+ MCLK: Xung clock chính. MCLK là phần mềm có thể lựa chọn như LFXT1CLK, VLOCLK, XT2CLK (nếu có) hoặc DCOCLK. MCLK có thể được chia cho 1, 2, 4 hoặc 8 trong khối FLL. MCLK được sử dụng bởi CPU và hệ thống.

+ SMCLK: Xung clock chính phụ. SMCLK là phần mềm có thể chọn là XT2CLK (nếu có) hoặc DCOCLK. SMCLK là phần mềm có thể lựa chọn cho các mô-đun ngoại vi riêng lẻ.



Sơ đồ khối bộ tạo giao động

+ Sau khi PUC, MCLK và SMCLK được lấy từ DCOCLK, gấp 32 lần tần số ACLK. Bộ giao động tạo tần số 32768 Hz được sử dụng cho ACLK, MCLK và SMCLK ổn định thành 1.048576 MHz. Các bit điều khiển thanh ghi trạng thái SCG0, SCG1, OSCOFF và CPUOFF định cấu hình chế độ hoạt động MSP430 và bật hoặc tắt các thành phần của mô-đun đồng hồ FLL + Ngắt và Chế độ thực thi. Các thanh ghi SCFQCTL, SCFI0, SCFI1, FLL_CTL0 và FLL_CTL1 cấu hình mô-đun clock. FLL+ có thể được cấu hình hoặc cấu hình lại bằng phần mềm bất kỳ lúc nào trong quá trình thực thi chương trình.

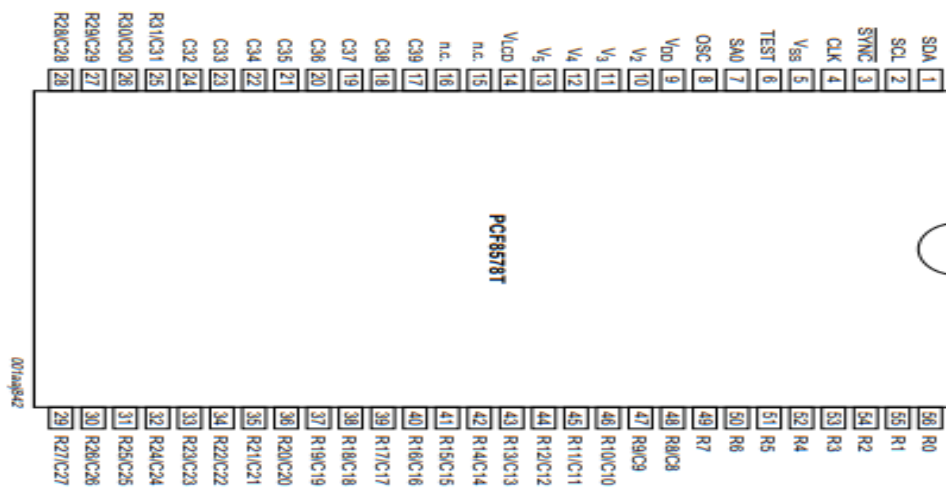
ACLK	Bộ tạo xung clock phụ
MCLK	Bộ tạo xung clock chính
SMCLK	Hệ thống tạo xung clock chính phụ DC
DCO	Bộ tạo giao động điều khiển kĩ thuật số
FFL	Vòng lặp tần số đã khóa
SCG	Hệ thống xung clock chung
PUC	Bật lại hệ thống
WDT	Bộ hẹn giờ cho hệ thống.

2. Tìm hiểu về chip PCF8578

2.1 Tổng quan:

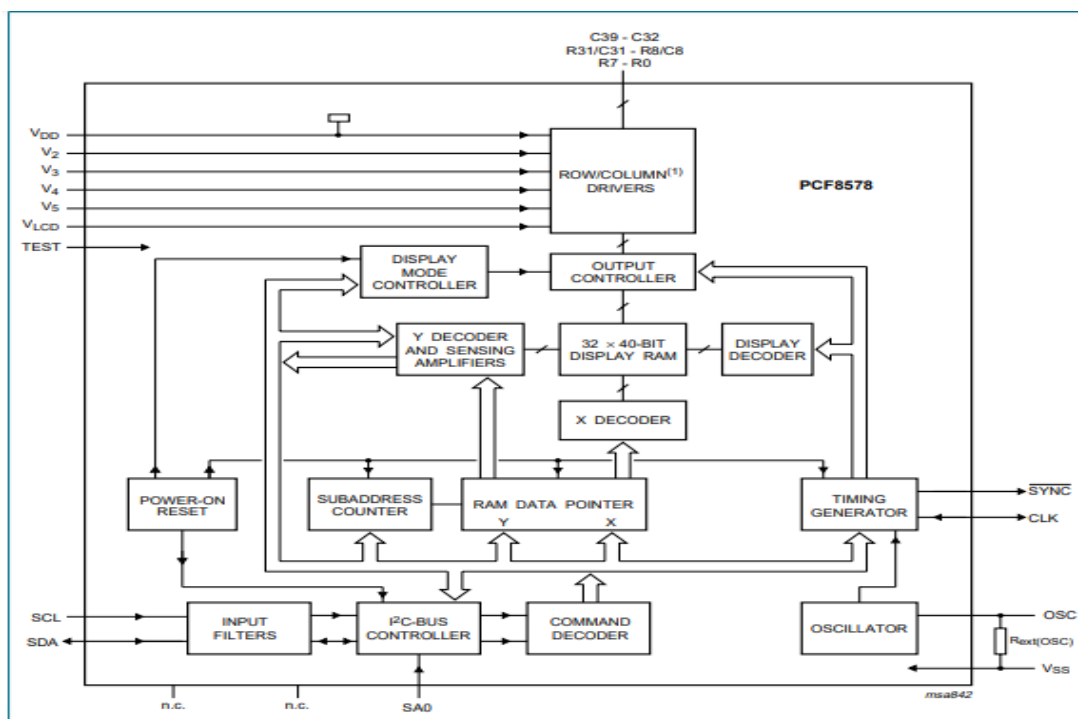
+ **PCF8578**: là trình điều khiển hàng và cột màn hình LCD công suất thấp, được thiết kế để điều khiển màn hình đồ họa ma trận điểm ở tỷ lệ ghép kênh 1: 8, 1:16, 1:24 hoặc 1:32. Thiết bị có 40 đầu ra, trong đó 24 đầu ra có thể lập trình và cấu hình cho các tỷ lệ hàng / cột sau: 32/8, 24/16, 16/24 hoặc 8/32. PCF8578 có thể hoạt động như một bộ điều khiển LCD độc lập và trình điều khiển để sử dụng trong các hệ thống nhỏ. Đối với các hệ thống lớn hơn, nó có thể được sử dụng kết hợp với tối đa 32 PCF8579 mà nó đã được tối ưu hóa. Hai thiết bị này kết hợp với nhau tạo thành một bộ chip điều khiển ma trận điểm LCD đa năng, có khả năng điều khiển màn hình lên đến 40960 điểm. PCF8578 tương thích với hầu hết các bộ vi điều khiển và giao tiếp qua bus hai chiều hai dòng (I2C-bus). Chi phí giao tiếp được giảm thiểu bởi một RAM màn hình với chức năng định địa chỉ tự động tăng dần và chuyển đổi bank hiển thị.

2.2 Các chân và chức năng.



BÁO CÁO THỰC TẬP TỐT NGHIỆP

Tên chân	Thứ tự pin	Mô tả
SDA	1	Cổng dữ liệu I2C
SCL	2	Cổng xung đồng hồ I2C
ISYNC	3	Đầu ra tăng 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



Sơ đồ khối của chip PCF8578

BÁO CÁO THỰC TẬP TỐT NGHIỆP

PCF8578 hoạt động theo 3 chế độ sau:

- + Chế độ 1 mình, driver của hàng và cột cho hiển thị nhỏ.
- + 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.

- + Ở chế độ hỗn hợp, thiết bị hoạt động như một trình điều khiển hàng và cột. Nó có thể được sử dụng trong các ứng dụng độc lập nhỏ hoặc cho các màn hình lớn hơn với tối đa 15 PCF8579 (31 PCF8579 khi hai địa chỉ phụ được sử dụng).
- + Ở chế độ hàng, thiết bị hoạt động như một trình điều khiển hàng với tối đa 32 đầu ra hàng và cung cấp tín hiệu đồng hồ và đồng bộ hóa cho PCF8579. Thông thường có thể xếp tầng lên đến 16 PCF8579 (32 khi hai địa chỉ phụ được sử dụng).
- + Display RAM: Hiển thị PCF8578 chứa RAM tĩnh 32×40 -bit để lưu trữ dữ liệu hiển thị. RAM được chia thành 4 bank 40 byte ($4 \times 8 \times 40$ bit). Trong quá trình truy cập RAM, dữ liệu được truyền đến và đi từ RAM thông qua bus I2C. Tám cột dữ liệu đầu tiên (0 đến 7) không thể được hiển thị nhưng có sẵn để lưu trữ dữ liệu chung và cung cấp khả năng tương thích với PCF8579. Có sự tương ứng trực tiếp giữa địa chỉ X và số đầu ra của cột. 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

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

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.

Table 9. Definition of PCF8578 commands

Command	Operation code								Reference
Bit	7	6	5	4	3	2	1	0	
set-mode	C	1	0	T	E[1:0]		M[1:0]		Table 11
set-start-bank	C	1	1	1	1	1	B[1:0]		Table 12
device-select	C	1	1	0	A[3:0]				Table 13
RAM-access	C	1	1	1	G[1:0]		Y[1:0]		Table 14
load-X-address	C	0	X[5:0]						Table 15

- Lệnh Set-mode:

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)

- Lệnh Set-start-bank:

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

- Lệnh Device-select.

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

- Lệnh RAM-access:

Table 14. RAM-access - command bit description

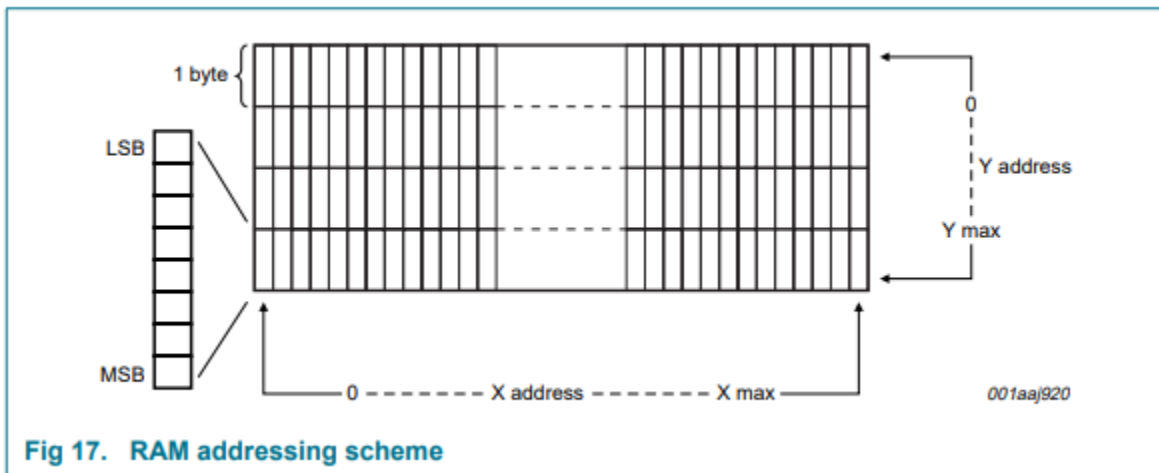
Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6 to 4	-	111	fixed value
3, 2	G[1:0]		RAM access mode; defines the auto-increment behavior of the address for RAM access (see Figure 18)
		00	character
		01	half-graphic
		10	full-graphic
		11	not allowed ^[1]
1, 0	Y[1:0]	0 to 3 ^[2]	RAM row address;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

- Lệnh Load-X-address

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

Bit	Symbol	Value	Description
7	C	0, 1	see Table 10
6	-	0	fixed value
5 to 0	X[5:0]	0 to 39 ^[1]	RAM column address; six bits of immediate data, transferred to the X-address pointer to define one of forty display RAM columns (see Figure 17)

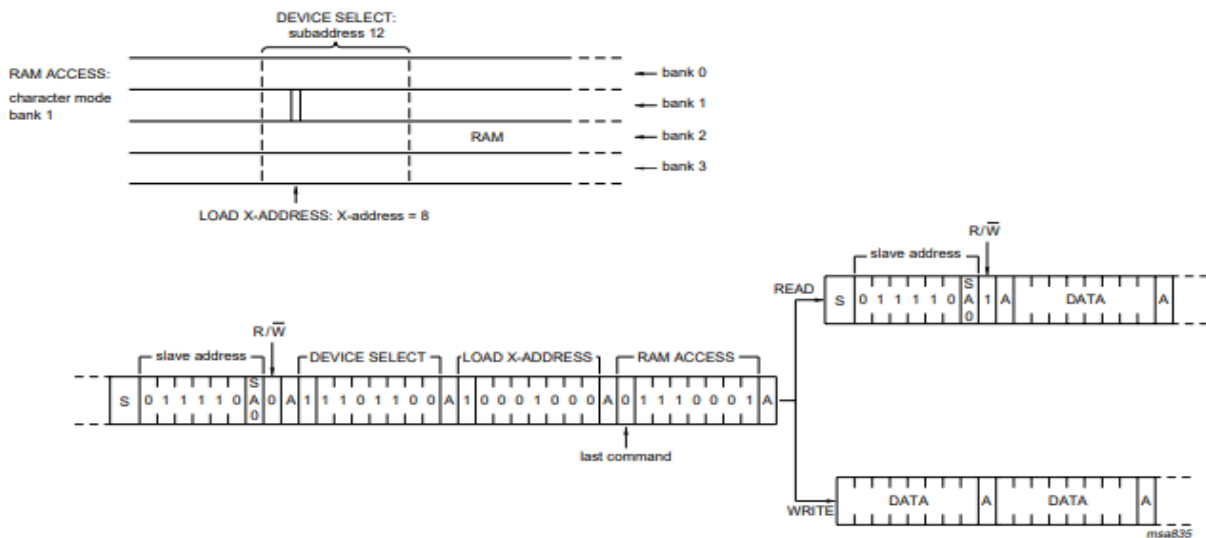
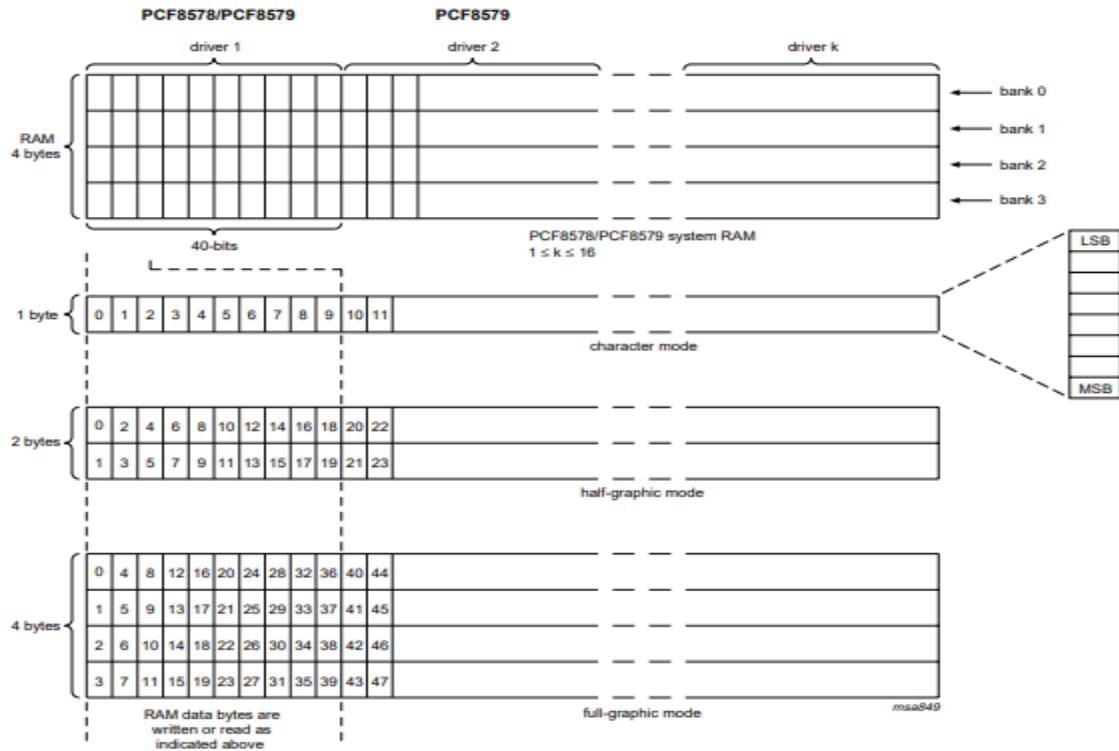
Truy cập RAM.



- + Hoạt động của RAM chỉ có thể thực hiện được khi PCF8578 ở chế độ hỗn hợp. Trong trường hợp này, địa chỉ con phần cứng của nó được cố định nội bộ ở 0000 và địa chỉ con phần cứng của bất kỳ PCF8579 nào được sử dụng cùng với PCF8578 phải bắt đầu từ 0001.
- + Có ba chế độ truy cập RAM: Ký tự, nửa đồ họa, toàn đồ họa.
- + Các chế độ này được chỉ định bởi các bit G [1: 0] của lệnh truy cập RAM. Lệnh truy cập RAM kiểm soát thứ tự dữ liệu được ghi vào hoặc đọc từ RAM.
- + Để lưu trữ dữ liệu RAM, người dùng chỉ định vị trí mà byte đầu tiên sẽ được tải.
 - Địa chỉ con của thiết bị (được chỉ định bởi lệnh chọn thiết bị)
 - Địa chỉ X RAM (được chỉ định bởi các bit X [5: 0] của lệnh tải-địa chỉ X)
 - Bank RAM (được chỉ định bởi các bit Y [1: 0] của lệnh truy cập RAM)

BÁO CÁO THỰC TẬP TỐT NGHIỆP

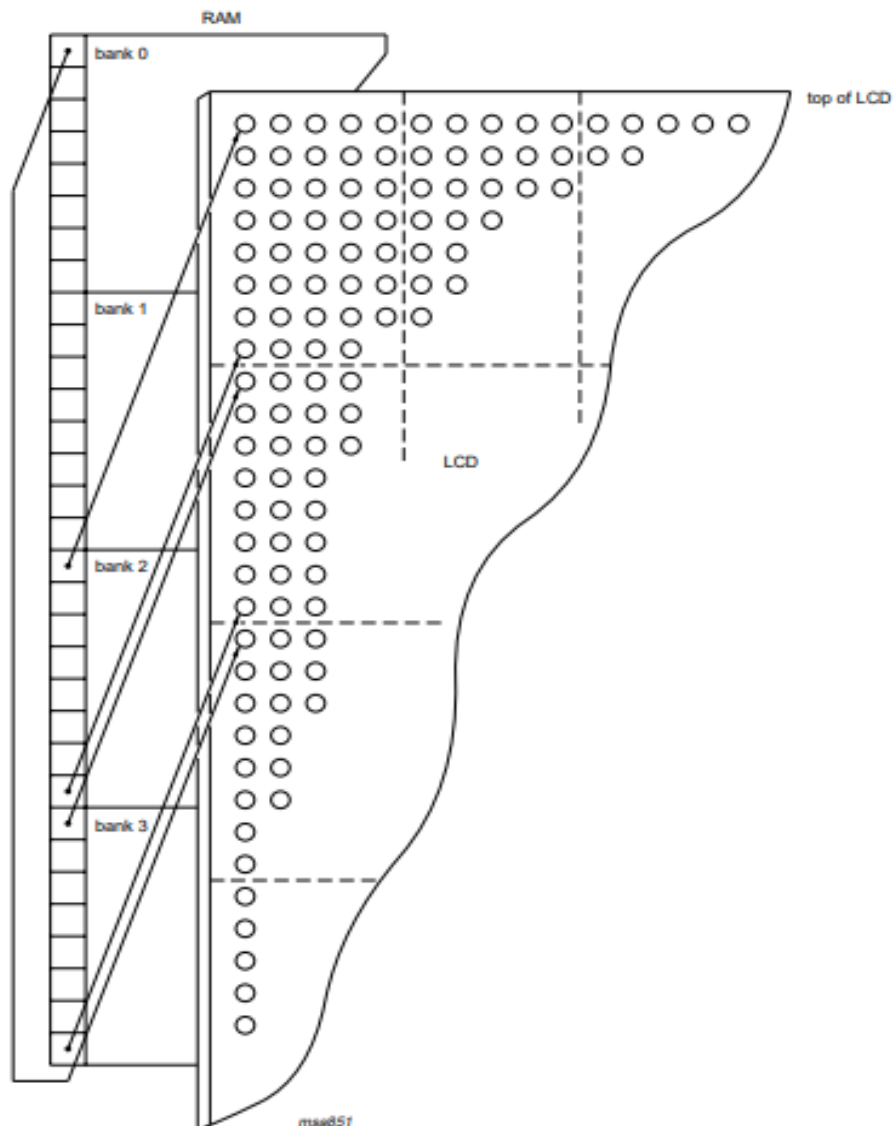
+ Các byte dữ liệu tiếp theo sẽ được ghi hoặc đọc theo chế độ truy cập RAM đã chọn. Các địa chỉ con của thiết bị được tăng tự động giữa các thiết bị cho đến khi đạt được thiết bị cuối cùng. Nếu thiết bị cuối cùng có địa chỉ con 15, việc truyền dữ liệu hiển thị thêm sẽ dẫn đến việc bao quanh địa chỉ con là 0.



2.3 Điều khiển hiển thị.

+ Màn hình được tạo ra bằng cách dịch chuyển liên tục các hàng dữ liệu RAM sang màn hình LCD ma trận điểm thông qua các đầu ra cột. Số hàng được quét phụ thuộc vào tốc độ ghép kênh được đặt bởi các bit M [1: 0] của lệnh set-mode.

+ Trạng thái hiển thị (tắt cả các dấu chấm bật hoặc tắt và video bình thường hoặc đảo ngược) được đặt bởi các bit E [1: 0] của lệnh set-mode. Đối với chuyển mạch bank, bank RAM tương ứng với phần trên cùng của màn hình được đặt bởi các bit B [1: 0] của lệnh set-start-bank. Điều này được thể hiện trong hình. Tính năng này rất hữu ích khi cuộn trong các ứng dụng chữ và số.



Chương 4

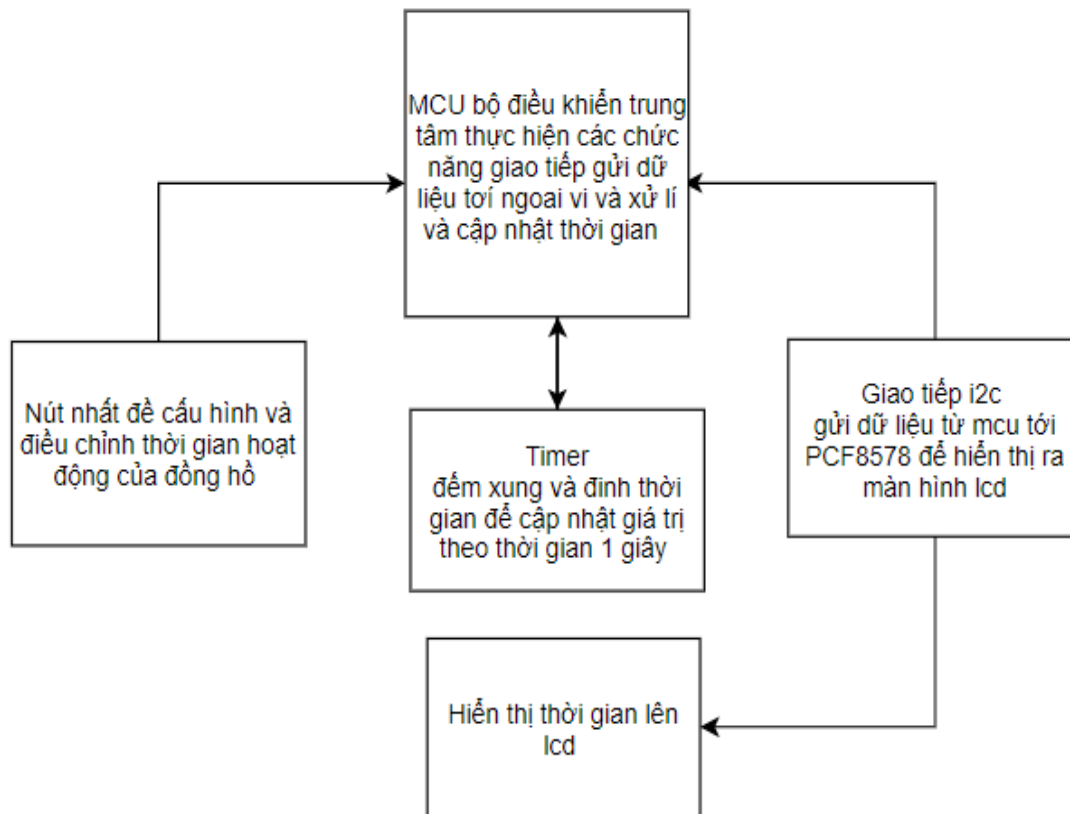
THỰC HIỆN ĐỀ TÀI HIỂN THỊ THỜI GIAN LÊN ĐỒNG HỒ ĐIỆN XS-96

1. Tóm tắt các yêu cầu đặt ra

Trong đồ án này, sinh viên sẽ tiến hành hiển thị thời gian theo (giờ /phút/ giây) trên LCD đa điểm. Thông qua sự tích hợp các linh kiện trên thiết bị, vi điều khiển chính chỉ sử dụng 2 chân là chân truyền dữ liệu (SDA) và chân xung cấp xung đồng hồ (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.

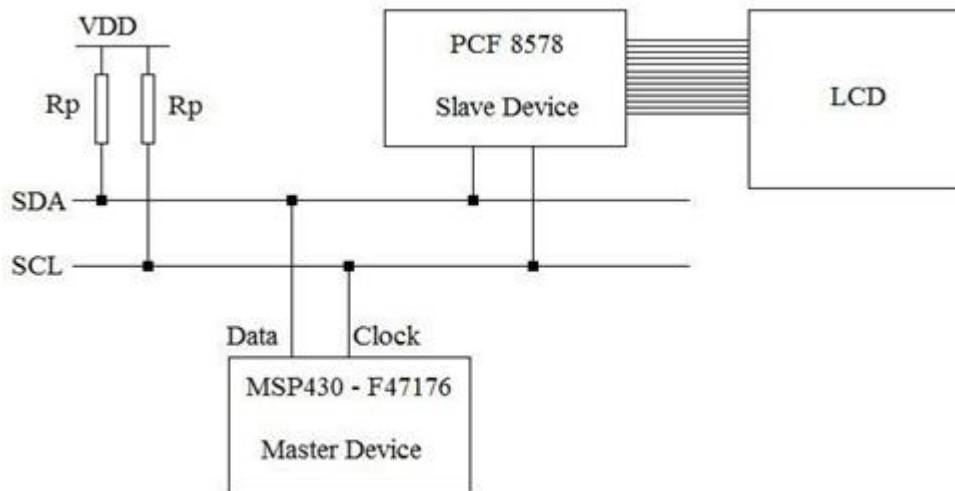
2. Thực hiện đề tài

2.1 Sơ đồ khối tổng quan của đề tài.



2.2 Sơ đồ minh họa cho việc kết nối trên thiết bị phần cứng

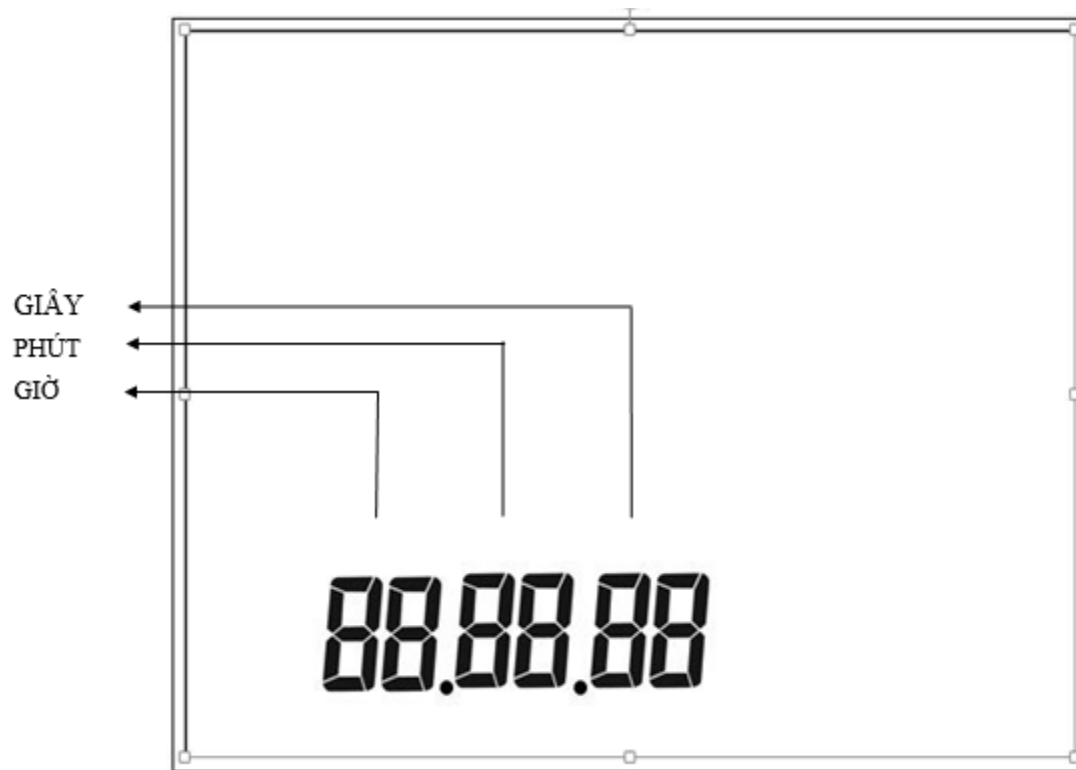
Sơ đồ I2C sẽ gồm có thiết bị chủ là vi điều khiển MSP430, thiết bị tớ là chip PCF- 8578 và LCD. Hai đường truyền SDA và SCL sẽ được nối với trở kéo lên Pull Up Resistor vào nguồn Vdd.



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 PCF8578.

Thiết bị tớ PCF 8578 sẽ đọc các tín hiệu từ vi điều khiển và thực hiện thao tác hiển thị trên màn 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 ký tự được thiết lập các vị trí cụ thể trên màn hình và hiển thị theo mong muốn lên LCD.

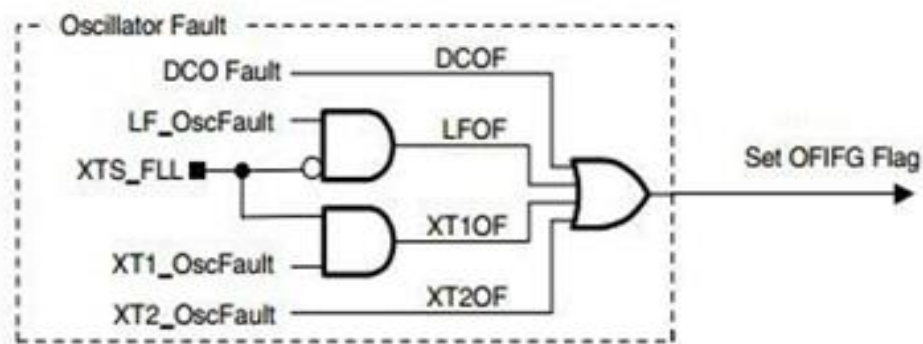
Kết quả hiển thị dự tính trên màn hình LCD sẽ là như hình dưới đây theo thứ tự (Giờ / Phút / Giây).



2.3 Các bước thực hành cho việc viết chương trình.

2.3.1 Cấu hình bộ giao động.

Đầu tiên để cho một vi điều khiển có thể hoạt động được thì chúng ta cần phải cấu hình bộ giao động để cho vi điều khiển có thể hoạt động trong suốt quá trình. Trong phần cứng của toàn hệ thống chúng ta sẽ sử dụng thạch anh ngoại là nguồn cung cấp xung chính thông qua bộ giao động 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.



2.3.2 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.

```
void setup_I2CB1(){  
    P2SEL |= BIT1+BIT2;  
    UCB1CTL1 |= UCSWRST;  
    UCB1CTL0 = UCMST + UCMODE_3 + UCSYNC;  
    UCB1CTL1 = UCSSEL_2 + UCSWRST;  
    UCB1BR0 = 148;  
    UCB1BR1 = 0;  
    UCB1I2CSA = 0x3C;  
    UCB1CTL1 &=~ UCSWRST;  
    UC1IE |= UCB1TXIE;  
}
```

2.3.3 Thử nghiệm và lựa chọn chế độ điều khiển cho việc hiển thị trên màn hình LCD.

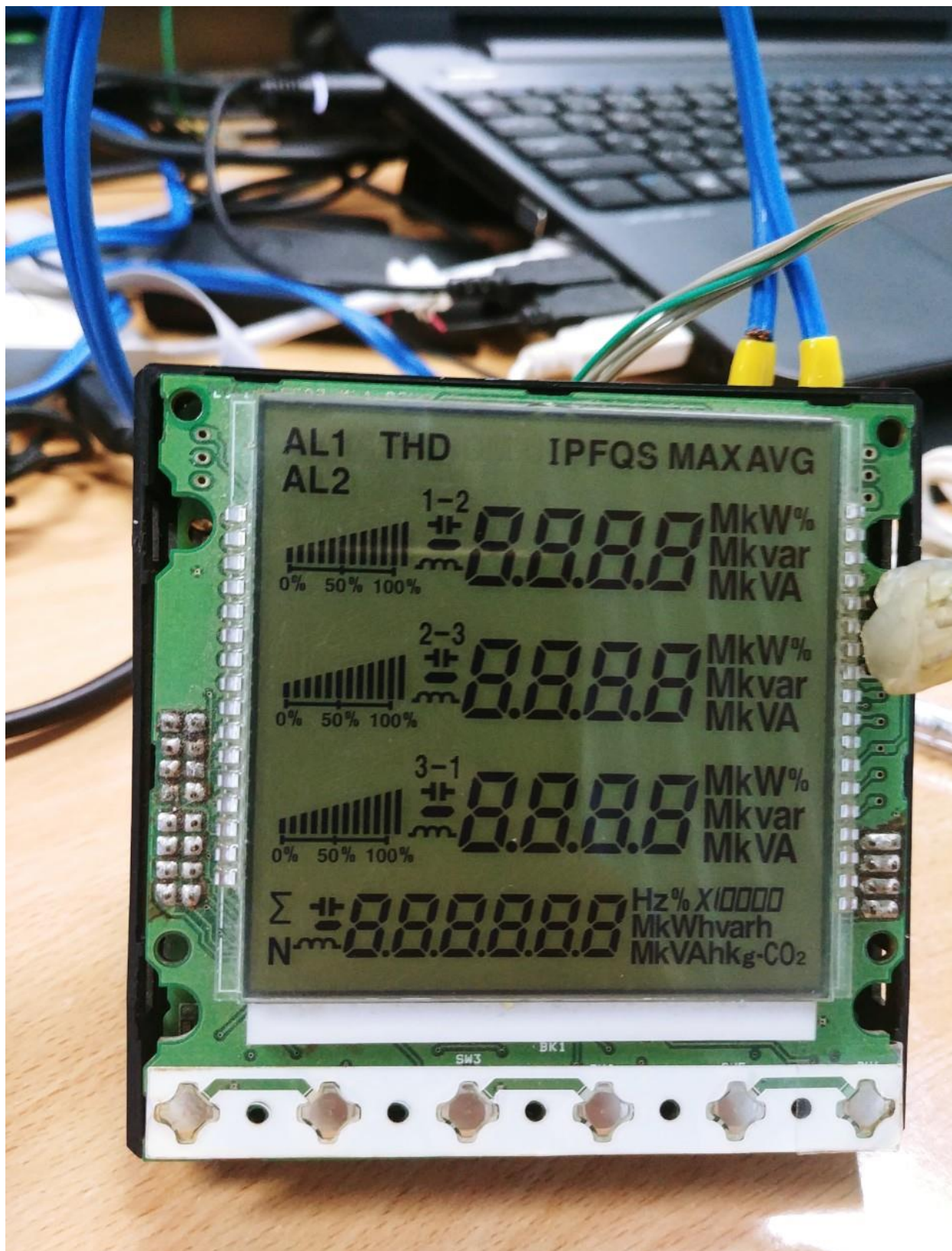
Phần này liên quan đến việc cấu hình Display RAM trước khi gửi các byte hiển thị đến PCF8578. Sẽ gồm có 5 byte được gửi trước, trong đó 4 byte đầu có MSB =

BÁO CÁO THỰC TẬP TỐT NGHIỆP

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 và column. Cũng vì thế, theo phần cứng đã được thiết kế sẵn thì sẽ phải chọn tỉ lệ 1:16, tức là 16 chân row và 24 chân column. Ở mục lựa chọn chế độ hiển thị, ta sẽ chọn chế độ normal để có thể hiển thị kí tự mong muốn.
- Set start bank: ở phần này ta có 4 bank của Display RAM để khởi tạo. Để chương trình mang tính tổng quát, ta sẽ lựa chọn bank 0.
- 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 lưu trữ các byte dữ liệu từ vi điều khiển qua I2C. Hơn nữa ta phải chọn chế độ lưu 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 7 đến 15 nên ta chọn X-address là 7 cho tổng quát và sau này có thể phát triển thêm cho LCD một cách toàn diện.
- Dữ liệu được gửi qua và bật hết tất cả các bit hiển thị trên màn hình LCD.

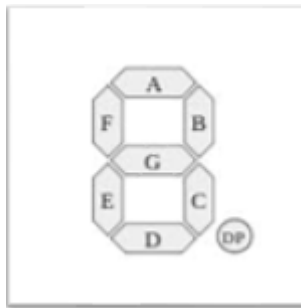


Bật hết tất các các bút

SVTH: NGUYỄN QUANG HUY

2.3.4 Tiến hành hiển thị những kí tự theo nguyên vọng

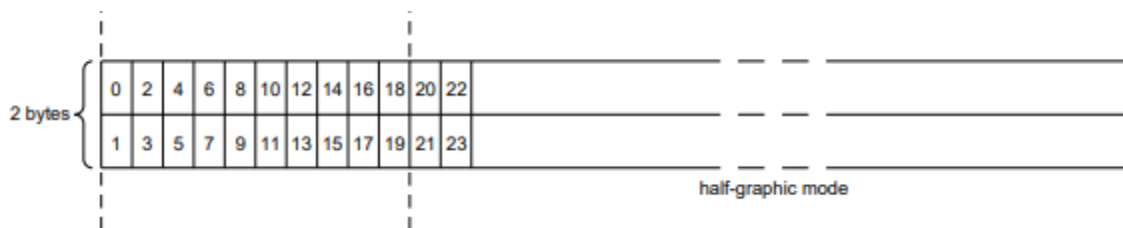
Ở phần này, nhiệm vụ đặt ra là phải hiển thị được 1 số tự nhiên bé hơn 100 trên 3 vùng (giờ / phút / giây) một cách độc lập. Có một điều cần lưu ý ở đây là các số này sẽ hiển thị theo dạng led 7 đoạn. Giả sử trong LCD sẽ có 18 vị trí hiển thị số được đánh dấu index từ 1 đến 18. Như vậy, ví dụ số tại vị 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.



Tuy nhiên, trong Display RAM của LCD thì bit hiển thị của số 1 nằm ở 4 vị trí địa chỉ khác nhau. 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.

Từ 5 lệnh điều khiển chế độ hiển thị ở trên thì lúc này ta sẽ thực hiện việc tác động từng BIT trong 1 Byte dữ liệu được ghi vào trong Display RAM để có thể điều khiển việc hiển thị. Dựa trên phần cứng được thiết kế sẵn việc thiết lập truyền cấu hình ban đầu cho LCD thì ở lệnh RAM access chọn chế độ half-graphic.

Thì lúc này các ô nhớ trong chip PCF8578 sẽ được minh họa như sau:



Từ hình ảnh minh họa trên công với việc chúng ta hiển thị tất cả các kí tự lên LCD sau đó tắt dần các kí tự không phù hợp để lựa các ô nhớ đúng với yêu cầu đặt ra thì ta rút ra được như sau .

Bank0	x =0	x =2	x =4	x =38	x =40
Bank1	x =1	x =3	x =5	x =39	

BÁO CÁO THỰC TẬP TỐT NGHIỆP

Trong đó địa chỉ các ô địa chỉ từ x0 – x4 là lưu các lệnh cấu hình lên LCD. Các ô địa chỉ từ x7, x9, x11, x13 của bank 1 sử dụng để hiển thị các số ở vị trí giây và hàng chục của vị trí giờ. Các ô x8, x10, x12, x14 sẽ sử dụng để hiển thị số ở các vị trí còn lại trên màn hình LCD.

Với việc phân cứng và cấu hình như trên thì bây giờ ta phải làm thế nào để lựa chọn ra từng bit trong từng ô địa chỉ để có thể tiến hành quá trình bật tắt từng bit và có thể hiển thị ra giá trị từ 0-9. Để làm điều đó thì ta cần tạo ra một mảng vs 41 phần tử để truyền vào các ô nhớ để đưa ra màn hình hiển thị, trước khi gửi dữ liệu vào mảng thì chúng ta cần tách các bit của số bất kì và gán các bit đó vào đúng vị trí của bit của ô địa chỉ của mảng đó. Các số sẽ được hiển thị theo kiểu led 7 thanh từ 0 đến 9.

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_kethop 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_kethop. 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
```

```
};
```

Để việc gán các BIT của 1 số vào 1 bit trong 1 Byte dữ liệu của địa chỉ của phần tử mảng thì ta dùng 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 có một

BÁO CÁO THỰC TẬP TỐT NGHIỆP

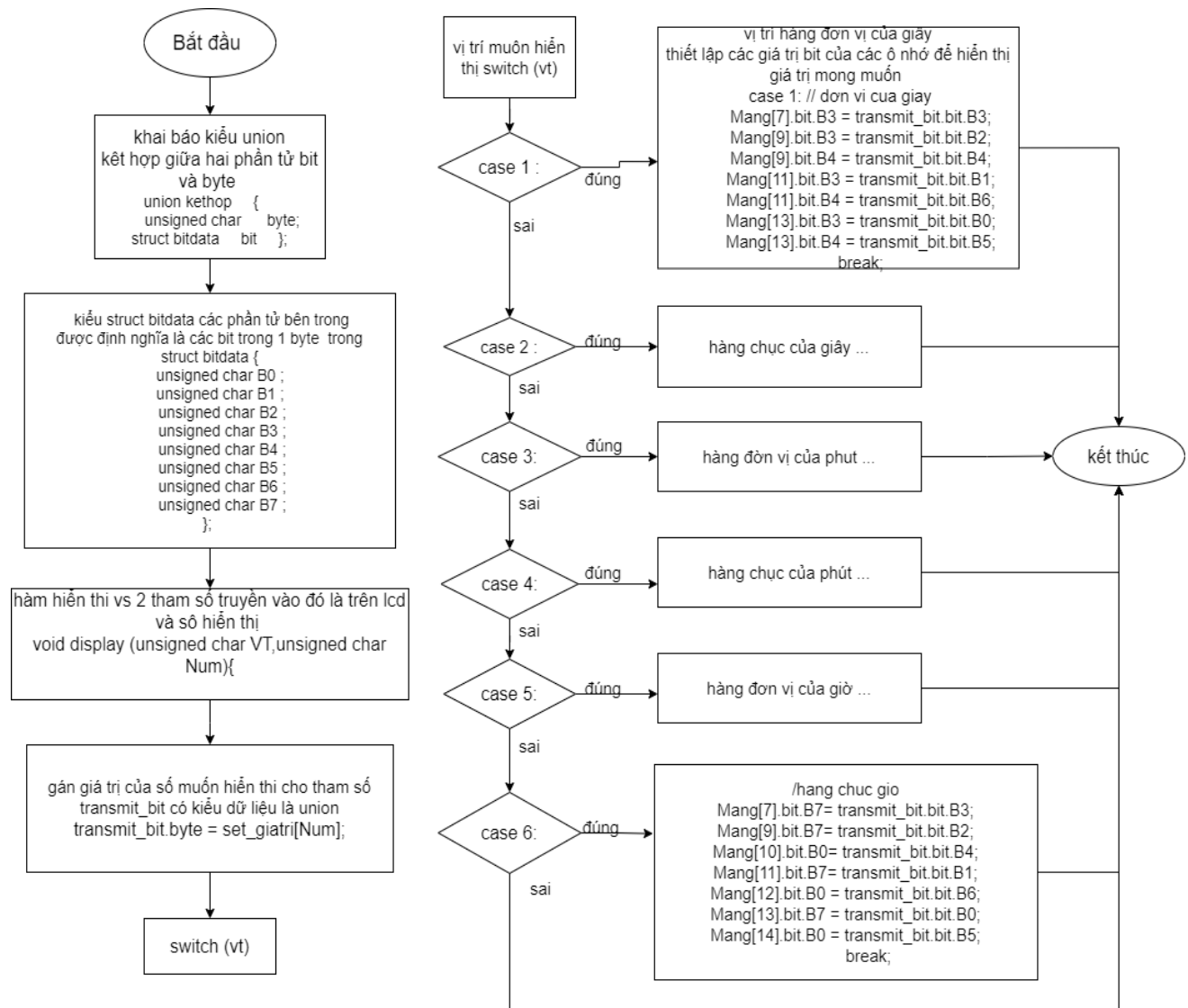
dạng mới là *union_kethop* như là một array 8 phần tử. Ta có thể tác động từng phần tử một và đồng thời tác động lên tất cả các phần tử cùng một thời điểm.

Hiển thị số tại khoảng giây.



SVTH: NGUYỄN QUANG HUY

Sơ đồ khối:



Code minh họa

struct bitdata {

unsigned char B0:1 ; //các biến B0...B7 được hiệu là BIT bằng cách ":1"

unsigned char B1:1 ; //sử dụng bit cuối cùng

unsigned char B2:1 ;

unsigned char B3:1;

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
    unsigned char B4:1 ;
    unsigned char B5:1 ;
    unsigned char B6:1 ;
    unsigned char B7:1 ;
};

union kethop{
    unsigned char byte;
    struct bitdata bit;
};

union kethop Mang[41]={0xD6,0xFC,0xF4,0xE0,0x16,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
    0x00,0x00,0x00,0x00,0x00,0x00};

union kethop transmit_bit;

void display (unsigned char VT,unsigned char Num){
    transmit_bit.byte = set_giatri[Num];
    switch (VT){
    case 1: // don vi cua giay
        Mang[7].bit.B3 = transmit_bit.bit.B3;
        Mang[9].bit.B3 = transmit_bit.bit.B2;
        Mang[9].bit.B4 = transmit_bit.bit.B4;
        Mang[11].bit.B3 =transmit_bit.bit.B1;
        Mang[11].bit.B4 = transmit_bit.bit.B6;
        Mang[13].bit.B3 = transmit_bit.bit.B0;
```

```
Mang[13].bit.B4 = transmit_bit.bit.B5;  
break;  
case 2:// hang chuc giay  
Mang[7].bit.B6=1;  
Mang[7].bit.B5= transmit_bit.bit.B3;  
Mang[9].bit.B5 = transmit_bit.bit.B2;  
Mang[9].bit.B6 = transmit_bit.bit.B4;  
Mang[11].bit.B5 = transmit_bit.bit.B1;  
Mang[11].bit.B6 = transmit_bit.bit.B6;  
Mang[13].bit.B5 = transmit_bit.bit.B0;  
Mang[13].bit.B6 = transmit_bit.bit.B5;  
break;
```

2.3.5 Khởi tạo bộ TIMER A0

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 hàm second (giây) để cập nhập theo giây. Như vậy, khi hàm second đạt được giá trị 60 thì ta sẽ tăng giá trị của phút lên một đơn vị. Tương tự, khi hàm 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 gian giờ / phút / giây và cho phép chúng chạy theo chuẩn thời gian của một ngày.

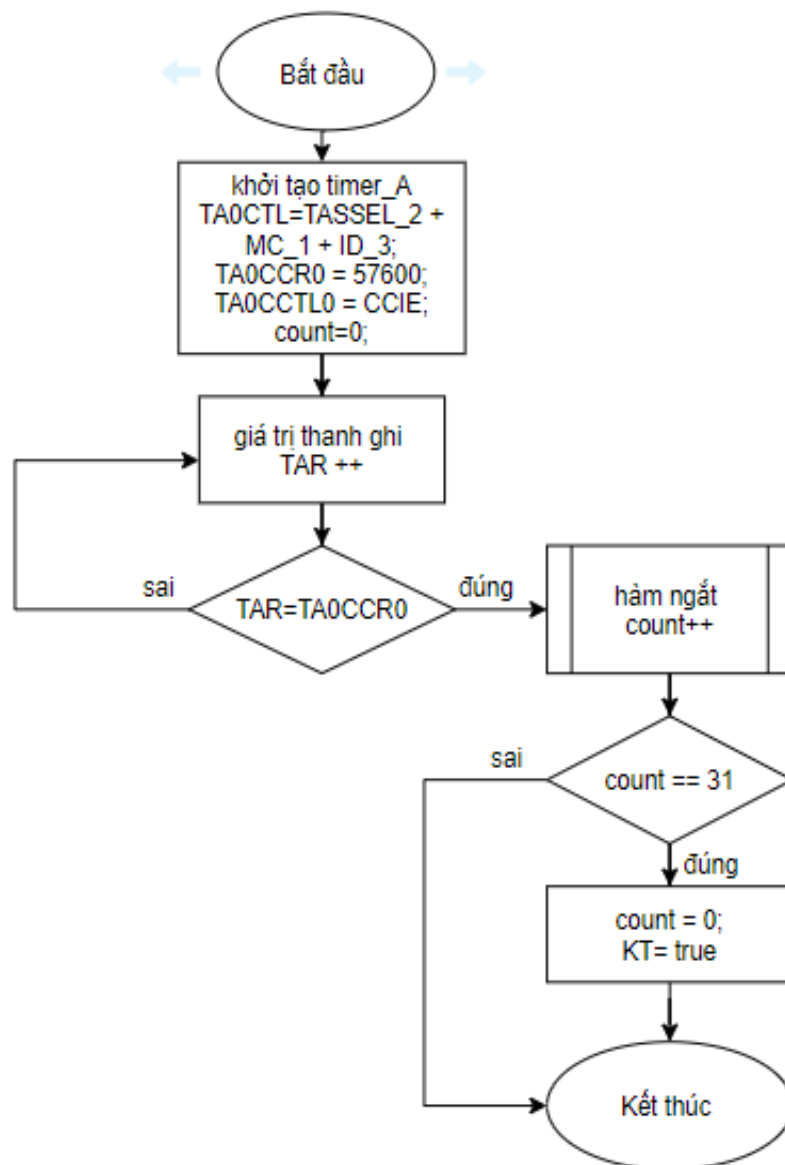
Độ chính xác của hệ thống là mục tiêu tiếp theo. Chính vì SMCLK có giá trị rất lớn, một xung của bộ đồng hồ này sẽ rất nhỏ so với một giây khi ta muốn cập nhập theo giây trực tiếp trên timer. Xung đồng hồ được lấy từ thạch anh nên độ chính xác rất cao. Lí do ta chọn $CCR0 = 57600$ vì khi đó mỗi xung ngắt của Timer sẽ có chu kì bằng $(57600 \text{ Hz} / 14.7456 \text{ MHz}) = 3.906 \text{ ms}$. 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

BÁO CÁO THỰC TẬP TỐT NGHIỆP

ngắt 32 lần. Hệ quả là ta sẽ sử dụng một biến count để đếm trong Timer sao cho mỗi khi count = 31 thì ta sẽ bắt đầu tăng second (giây).

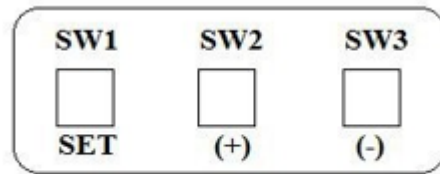
Cuối cùng nhưng cũng quan trọng không kém, đó là nếu để việc cập nhập liên tục theo mỗi xung Timer thì sẽ rất tốn kém. Vì vậy một cờ cập nhập sẽ được bật trong Timer mỗi lần biến count đếm đến 31. Và trong hàm chính sẽ cho phép cập nhập thời gian, hay là truyền dữ liệu trên I2C, nếu cờ ngắt này được 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.

Sơ đồ khối



2.3.6 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 0.5 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 0.5 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 0.5 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)

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).

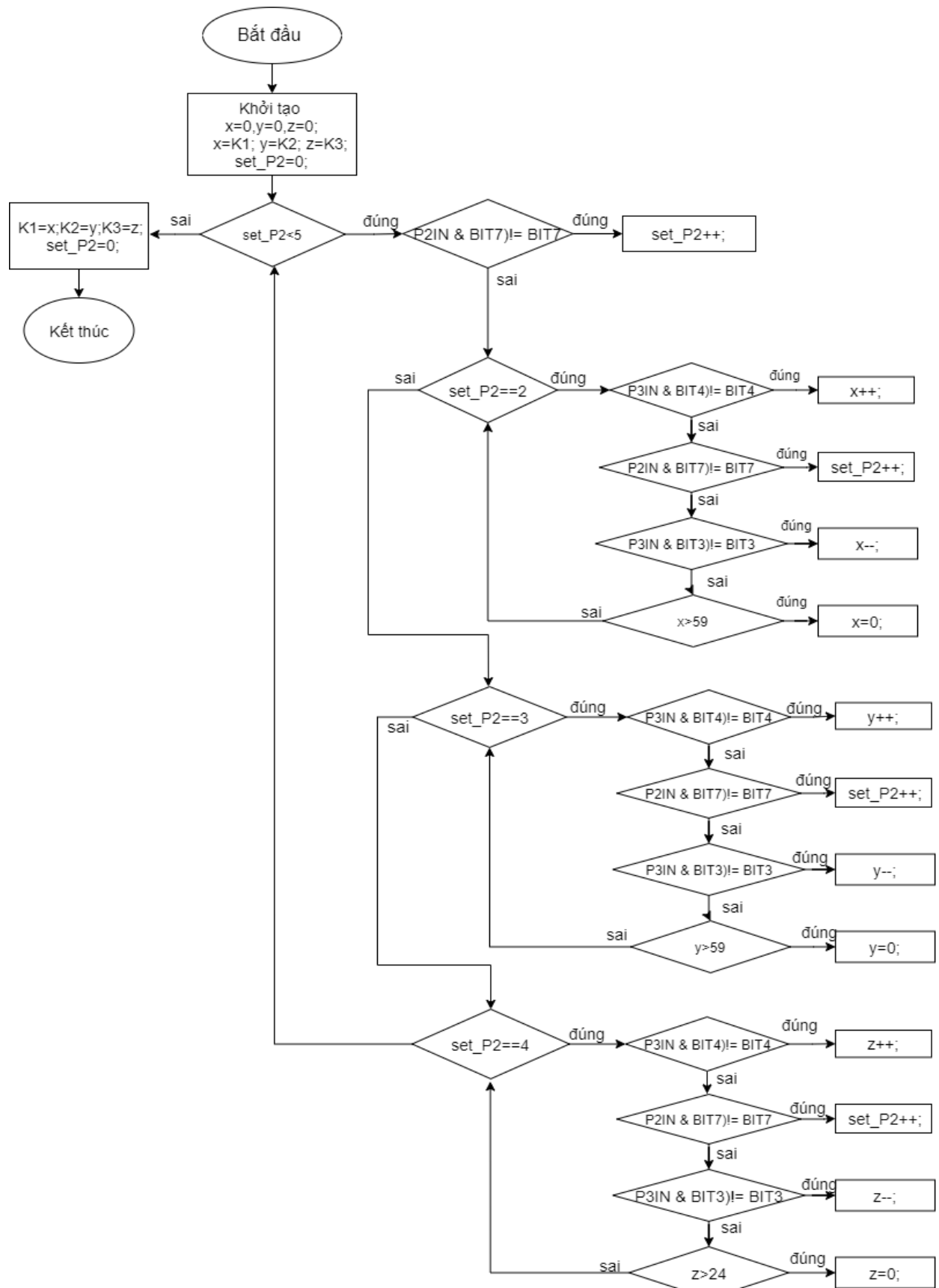
- + Khi bấm 1 lần rồi thả: giá trị nhấp nháy tại vị trí hiện thời tăng lên 1 đơn vị.

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.

Sơ đồ thuật toán

BÁO CÁO THỰC TẬP TỐT NGHIỆP



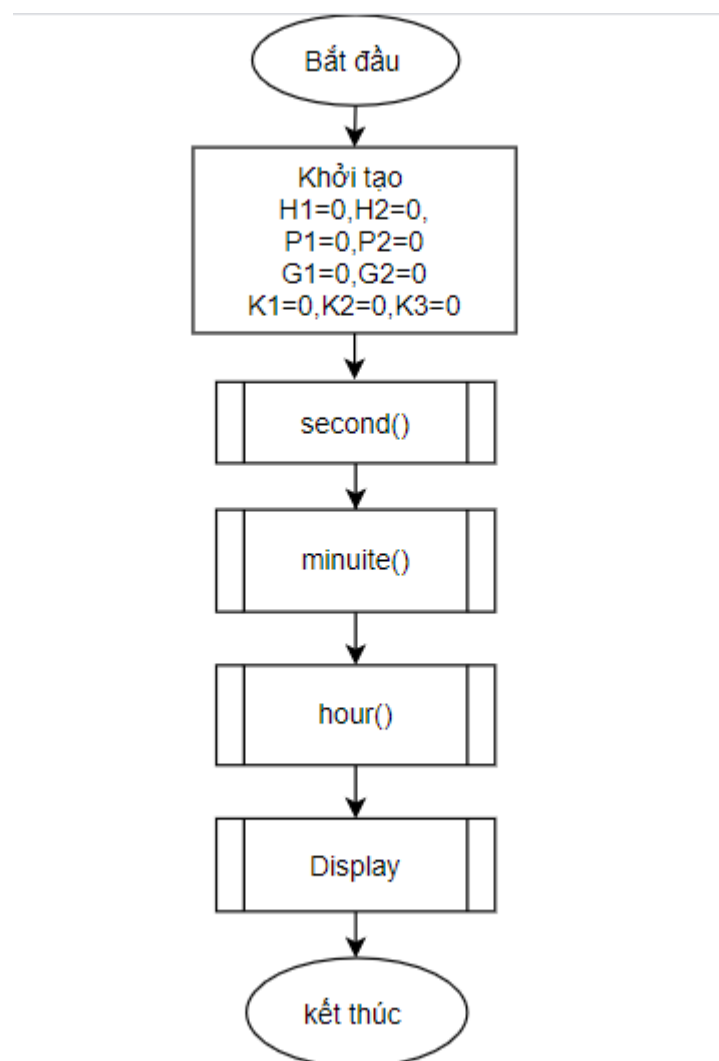
SVTH: NGUYEN QUANG HUY

2.3.7 Hàm xử lý thời gian

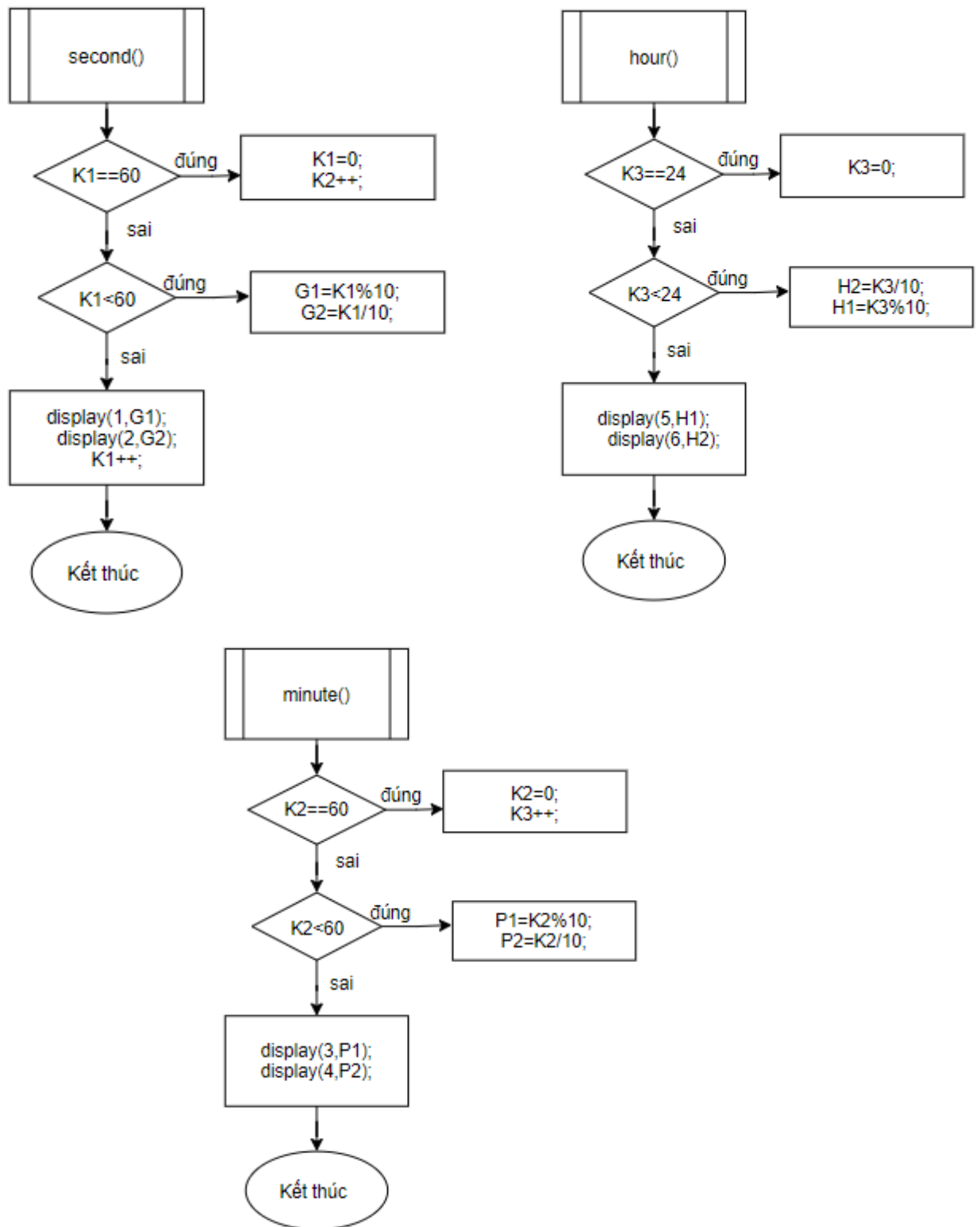
Hàm xử lý thời gian sẽ đảm bảo cho giờ nằm trong vùng giá trị nguyên từ 0 đến 23, phút nằm trong vùng giá trị nguyên từ 0 đến 59 và giây nằm trong vùng giá trị nguyên từ 0 đến 59.

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.

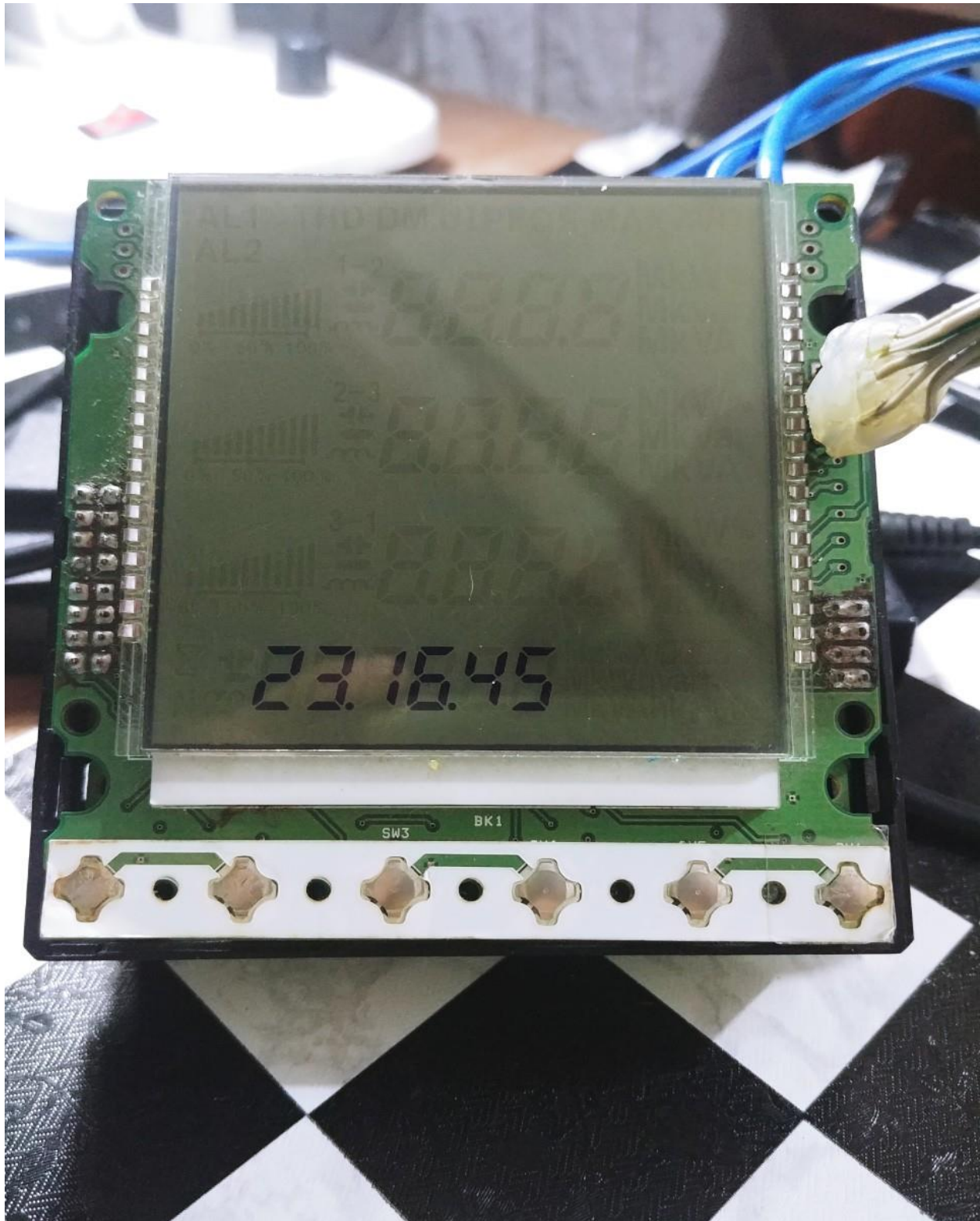


Sơ đồ khối của hàm hiển thị thời gian



BÁO CÁO THỰC TẬP TỐT NGHIỆP

Kết quả đạt được.



Hiển thị thời gian như trên đồng hồ

SVTH: NGUYỄN QUANG HUY

Chương 5

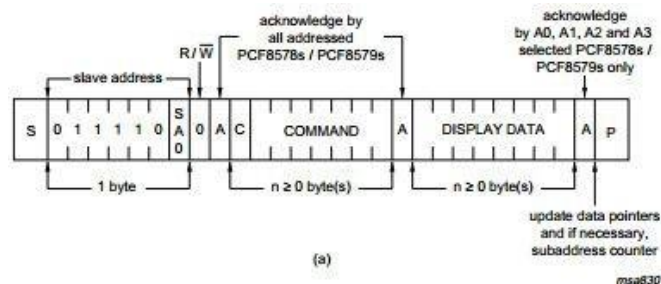
KẾT LUẬN

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 lý thuyết, em 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. Quá trình thực hiện coding thì trước tiên phải có sơ đồ khối và trình bày bài toán một cách rõ ràng để việc coding diễn ra nhanh hơn và tránh mất thời gian, có thể xử lý bài toán một cách tối ưu nhất. Không nên giải quyết bài toán theo cảm tính.

+ 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 lý 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.



BÁO CÁO THỰC TẬP TỐT NGHIỆP

+ Ngoài ra, trong đề 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. Từ đó dựa vào công thức các thông số đã cài đặt thì giá trị $N = 57600$ khi biến count đếm đến 31 lần sẽ thực hiện việc tăng thời gian lên 1s.

2. Các khó khăn khi gặp phải.

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 thực hiện sét các nút nhấn vẫn chưa hoạt động ổn định do chưa tối ưu code vẫn chưa hay và cần phải cải tiến, cần thực hành nhiều hơn.

Việc khởi tạo xung thạch anh cho vi điều khiển chính và việc gửi dữ liệu là hai khó khăn lớn nhất mà em gặp phải. 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 anh Tuyền và anh Lợi, 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, và có thể gửi dữ liệu chính xác từ vi điều khiển sang chip PCF8578 để có thể hiện thị lên LCD.

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 cộng vs đó là lối tư duy giải quyết bài toán vẫn còn hạn hẹp và chưa thực sự tốt. Chúng tôi cần phải tự rèn luyện thêm về mặt tư duy logic trong lập trình.

Tài liệu tham khảo

Texas Instrument, *MSP430x4xx Hardware – SLAS626C*, 2011.

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

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

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

Code của chương trình :

```
#include <msp430.h>
#include <stdbool.h>
unsigned char set_P2=0;
unsigned char H1=0,H2=0,P1=0,P2=0,G1=0,G2=0,K1=0,K2=0,K3=0;
unsigned char *Tx_data;
unsigned char Tx_byte;
unsigned char count=0;
_Bool KT = false;
unsigned char buffer[]={0xD6,0xFC,0xF4,0xE0,0x16,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00};

unsigned char set_giatri[]={
0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F // 0 1 2 3 4 5 6 7 8 9
};
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 ;
};
union kethop{
    unsigned char byte;
    struct bitdata bit;
};
union kethop Mang[41]={0xD6,0xFC,0xF4,0xE0,0x16,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                        0x00,0x00,0x00,0x00,0x00,0x00};
union kethop transmit_bit;

void display (unsigned char VT,unsigned char Num){
    transmit_bit.byte = set_giatri[Num];
    switch (VT){
        case 1: // don vi cua giay
            Mang[7].bit.B3 = transmit_bit.bit.B3;
            Mang[9].bit.B3 = transmit_bit.bit.B2;
            Mang[9].bit.B4 = transmit_bit.bit.B4;
            Mang[11].bit.B3 =transmit_bit.bit.B1;
            Mang[11].bit.B4 = transmit_bit.bit.B6;
            Mang[13].bit.B3 = transmit_bit.bit.B0;
            Mang[13].bit.B4 = transmit_bit.bit.B5;
            break;

        case 2:// hang chuc giay
            Mang[7].bit.B6=1;
```

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
Mang[7].bit.B5= transmit_bit.bit.B3;
Mang[9].bit.B5 = transmit_bit.bit.B2;
Mang[9].bit.B6 = transmit_bit.bit.B4;
Mang[11].bit.B5 = transmit_bit.bit.B1;
Mang[11].bit.B6 = transmit_bit.bit.B6;
Mang[13].bit.B5 = transmit_bit.bit.B0;
Mang[13].bit.B6 = transmit_bit.bit.B5;
break;

case 3:// don vi phut
Mang[8].bit.B7 = transmit_bit.bit.B3;
Mang[10].bit.B7= transmit_bit.bit.B2;
Mang[10].bit.B6= transmit_bit.bit.B4;
Mang[12].bit.B7= transmit_bit.bit.B1;
Mang[12].bit.B6= transmit_bit.bit.B6;
Mang[14].bit.B7= transmit_bit.bit.B0;
Mang[14].bit.B6= transmit_bit.bit.B5;
break;

case 4: // hang chuc phut
Mang[8].bit.B4 = 1;
Mang[8].bit.B5 = transmit_bit.bit.B3;
Mang[10].bit.B5= transmit_bit.bit.B2;
Mang[10].bit.B4= transmit_bit.bit.B4;
Mang[12].bit.B5= transmit_bit.bit.B1;
Mang[12].bit.B4= transmit_bit.bit.B6;
Mang[14].bit.B5= transmit_bit.bit.B0;
Mang[14].bit.B4= transmit_bit.bit.B5;
break;

case 5:// hang don vi gio
Mang[8].bit.B3 = transmit_bit.bit.B3;
Mang[10].bit.B3= transmit_bit.bit.B2;
Mang[10].bit.B2= transmit_bit.bit.B4;
Mang[12].bit.B3= transmit_bit.bit.B1;
Mang[12].bit.B2= transmit_bit.bit.B6;
Mang[14].bit.B3= transmit_bit.bit.B0;
Mang[14].bit.B2= transmit_bit.bit.B5;
break;

case 6://hang chuc gio
Mang[7].bit.B7= transmit_bit.bit.B3;
Mang[9].bit.B7= transmit_bit.bit.B2;
Mang[10].bit.B0= transmit_bit.bit.B4;
Mang[11].bit.B7= transmit_bit.bit.B1;
Mang[12].bit.B0 = transmit_bit.bit.B6;
Mang[13].bit.B7 = transmit_bit.bit.B0;
Mang[14].bit.B0 = transmit_bit.bit.B5;
break;
    }
}

void setup_SMCLK(){ // cau hinh chon bo giao dong ngoai smclk
    FLL_CTL0 |= XCAP11PF;
    FLL_CTL1 &=~XT2OFF;
```

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
FLL_CTL1 |= SELS;
FLL_CTL2 |= XT2S_2;
}
void setup_I2CB1(){ // thiết lập ucs1 chọn chức năng giao tiếp i2c
    P2SEL |= BIT1+BIT2;
    UCB1CTL1 |= UCSWRST;
    UCB1CTL0 = UCMST + UCMODE_3 + UCSYNC;
    UCB1CTL1 = UCSSEL_2 + UCSWRST;
    UCB1BR0 = 148;
    UCB1BR1 = 0;
    UCB1I2CSA = 0x3C;
    UCB1CTL1 &=~ UCSWRST;
    UC1IE |= UCB1TXIE;
}
#pragma vector = USCIAB1TX_VECTOR // ham ngat co chức năng gửi dữ liệu
__interrupt void USCIAB1TX_ISR(void)
{
    if (Tx_byte)
    {
        UCB1TXBUF = *Tx_data++;
        Tx_byte--;
    }
    else
    {
        UCB1CTL1 |= UCTXSTP;
        UC1IFG &=~ UCB1TXIFG;
    }
}
// khởi tạo vào bắt đầu gửi dữ liệu từ vdk sai pcf 8578
int transmit_i2c( unsigned char dataSend[], unsigned char count){
    Tx_data = (unsigned char *)dataSend;
    Tx_byte= count;
    UCB1CTL1 |= UCTR + UCTXSTT;
    __bis_SR_register(GIE);
    while (UCB1CTL1 & UCTXSTP);
    return 0;
}
void convert_data(){ // chuyển từ kiểu dữ liệu union sang unsigned char
    int i;
    for (i=0;i<41;i++){
        buffer[i]=Mang[i].byte;
    }
}

void setup_timer (unsigned char T){ // timer định thời gian vào khoảng 1s
    TA0CTL=TASSEL_2 + MC_1 + ID_3;
    TA0CCR0 = 57600;
    TA0CCTL0 = CCIE;
    _enable_interrupt();
}
#pragma vector = TIMER0_A0_VECTOR // vector ngat CCR0 của Timer
__interrupt void TimerInterruptCCR0(void){
    count ++;
    if (count== 31){
```

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
        KT= true;
        count=0;
    }
}
void hour(){ // ham hien thi gio va cho gio hoạt động
    if (K3==24){
        K3=0;
    }
    if (K3<24){
        H2=K3/10;
        H1=K3% 10;
    }
    display(5,H1);
    display(6,H2);
}
void minutes(){ // ham hien thi phut va cho phut hoạt động
    if(K2==60){
        K2=0;
        K3++;
    }
    if (K2<60){
        P1=K2%10;
        P2=K2/10;
    }
    display(3,P1);
    display(4,P2);
}
void second (){ //ham hien thi giây cho giây hoạt động
    if (K1==60){
        K1=0;
        K2++;
    }
    if(K1<60){
        G1=K1% 10;
        G2=K1/10;
    }
    display(1,G1);
    display(2,G2);
    K1++;
}
void Time(){ // ham cho thời gian chạy tự động
    second();
    minutes();
    hour();
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
}
// ham sử dụng để clear các bit tại các vị trí để thực hiện việc nhập nhay
void no_display(unsigned char VT){
    switch (VT){
        case 1:
            Mang[7].bit.B3 = 0;
            Mang[9].bit.B3 = 0;
            Mang[9].bit.B4 = 0;
            Mang[11].bit.B3 =0;
            Mang[11].bit.B4 = 0;
```

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
Mang[13].bit.B3 = 0;  
Mang[13].bit.B4 = 0;
```

```
Mang[7].bit.B6=1;  
Mang[7].bit.B5= 0;  
Mang[9].bit.B5 = 0;  
Mang[9].bit.B6 = 0;  
Mang[11].bit.B5 = 0;  
Mang[11].bit.B6 = 0;  
Mang[13].bit.B5 = 0;  
Mang[13].bit.B6 = 0;
```

```
break;
```

```
case 2:// don vi phut
```

```
Mang[8].bit.B7 = 0;  
Mang[10].bit.B7= 0;  
Mang[10].bit.B6= 0;  
Mang[12].bit.B7= 0;  
Mang[12].bit.B6= 0;  
Mang[14].bit.B7= 0;  
Mang[14].bit.B6= 0;
```

```
Mang[8].bit.B4 = 1;  
Mang[8].bit.B5 = 0;  
Mang[10].bit.B5= 0;  
Mang[10].bit.B4= 0;  
Mang[12].bit.B5= 0;  
Mang[12].bit.B4= 0;  
Mang[14].bit.B5= 0;  
Mang[14].bit.B4= 0;
```

```
break;
```

```
case 3:// hang don vi gio
```

```
Mang[8].bit.B3 = 0;  
Mang[10].bit.B3= 0;  
Mang[10].bit.B2= 0;  
Mang[12].bit.B3= 0;  
Mang[12].bit.B2= 0;  
Mang[14].bit.B3= 0;  
Mang[14].bit.B2= 0;
```

```
Mang[7].bit.B7= 0;  
Mang[9].bit.B7= 0;  
Mang[10].bit.B0= 0;  
Mang[11].bit.B7= 0;  
Mang[12].bit.B0 = 0;  
Mang[13].bit.B7 = 0;  
Mang[14].bit.B0 = 0;
```

```
break;
```

```
}
```

```
}
```

```
// ham nay dung de nhap nhay cac vi tri khi vao che do thiet lap .
```

```
void blink_number(unsigned char set_time, unsigned char TD){  
    unsigned char A1,A2;
```

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
A1=TD%10;
A2=TD/10;
if (set_time==2){
    no_display(1);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
    display(1,A1);
    display(2,A2);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
} else if (set_time==3){
    no_display(2);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
    display(3,A1);
    display(4,A2);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
} else if(set_time==4){
    no_display(3);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
    display(5,A1);
    display(6,A2);
    convert_data();
    transmit_i2c(buffer,sizeof (buffer));
    _delay_cycles(150000);
}
}

void setup_button(){
    int x=0,y=0,z=0;
    x=K1; y=K2; z=K3; // gán các giá trị ở thời trước khi dùng chạy của ham Time cho việc set thời gian .
    while(set_P2<5){
        if ((P2IN & BIT7)!= BIT7){ // kiểm tra xem set_P2 có được nhận 1 lần nữa
            while ((P2IN & BIT7)!= BIT7) // tiếp tục tăng giá trị set_P2 . nếu set_P2 = 2,3,4 ứng với các vị trí giây
            phút giây.
        }
        while (set_P2==2){ // Ứng vs vị trí giây
            if ((P2IN & BIT7)!= BIT7){ // kiểm tra xem Set_P2 có được nhận
                while ((P2IN & BIT7)!= BIT7) ;
                set_P2++;
            }
        }
        if((P3IN & BIT4)!= BIT4){ // kiểm tra xem P3.4 có được nhận để tăng giá trị giây thiết lập lên 1
            while ((P3IN & BIT4)!= BIT4) ;
            x++;
        }
        if ((P3IN & BIT3)!= BIT3){ // kiểm tra xem P3.3 có được nhận để giảm giá trị giây thiết lập xuống
            while ((P3IN & BIT3)!= BIT3) ;
        }
    }
}
```


BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
x--;
    }
    if(x>59){ // kiểm tra giá trị thiếp lập nếu lớn hơn 59 thì reset về 0
        x=0;
    }
    blink_number(2,x); // hàm thực hiện việc nhấp nháy
}
while (set_P2==3){ // ứng với vị trí phut set nút như vị trí giày

    if ((P2IN & BIT7)!= BIT7){
        while ((P2IN & BIT7)!= BIT7) ;
        set_P2++;
    }

    if((P3IN & BIT4)!= BIT4){
        while ((P3IN & BIT4)!= BIT4) ;
        y++;}
    if ((P3IN & BIT3)!= BIT3){
        while ((P3IN & BIT3)!= BIT3) ;
        y--;
    }
    if(y>59){
        y=0;
    }
    blink_number(3,y);
}
while (set_P2==4){ // ứng với vị trí giày cùng xet nút như vị trí giày
    if ((P2IN & BIT7)!= BIT7){
        while ((P2IN & BIT7)!= BIT7) ;
        set_P2++;
    }
    if((P3IN & BIT4)!= BIT4){
        while ((P3IN & BIT4)!= BIT4) ;
        z++;
    }
    if ((P3IN & BIT3)!= BIT3){
        while ((P3IN & BIT3)!= BIT3) ;
        z--;
    }
    if(z>24){
        z=0;
    }
    blink_number(4,z);
}
}
K1=x;K2=y;K3=z; // gán các giá trị đã thiếp lập lại cho các giá trị của hàm chạy tự động
set_P2= // reset giá trị set_P2 lại bằng 0.
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;
    P2DIR &=~ BIT7; // khởi tạo nút nhấn
    P3DIR &=~ BIT3; // khởi tạo nút nhấn
    P3DIR &=~ BIT4; // khởi tạo nút nhấn
    P5DIR |= BIT0;
    setup_SMCLK();
}
```

SVTH: NGUYỄN QUANG HUY

BÁO CÁO THỰC TẬP TỐT NGHIỆP

```
setup_I2CB1();
setup_timer(32);
while (1) {
    if (KT==true){
        Time();
        KT=false;
    }
    if ((P2IN & BIT7)!= BIT7){ // kiểm tra xem nút set P2.7 có được nhận
        while ((P2IN & BIT7)!= BIT7): // chống rung nút nhấn cho tới khi dứt nha ra
            set_P2++;
            setup_button();
        }
    }
}
```