ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN
Vi Xử Lý - Vi Điều Khiển

ĐỀ TÀI
Đồng hồ thời gian thực (RTC)

Giảng viên hướng dẫn: TS. Đoàn Duy

Sinh viên thực hiện:

Trần Minh Khải	23520679
Đỗ Xuân Khải	23520672
Huỳnh Vỹ Khang	23520688
Nguyễn Quang Khải	23520676
Trần Gia Huy	23520646

TP. Hồ Chí Minh, tháng … năm ….

LỜI CẢM ƠN

Nhóm chúng em xin chân thành cảm ơn Thầy **Đoàn Duy** – giảng viên hướng dẫn môn **Vi Xử Lý - Vi Điều Khiển**, lớp **CE103.P23** – đã tận tình giảng dạy, hướng dẫn và hỗ trợ nhóm trong suốt quá trình học tập và thực hiện báo cáo này.
Thông qua môn học, nhóm chúng em đã được tiếp cận và hiểu rõ hơn về kiến thức lý thuyết cũng như thực hành liên quan đến vi xử lý và vi điều khiển, đặc biệt là cách lập trình và ứng dụng trong các hệ thống nhúng. Những bài giảng và sự chỉ dẫn tận tâm của Thầy đã giúp nhóm nâng cao tư duy kỹ thuật và khả năng làm việc nhóm trong lĩnh vực này.
Nhóm cũng xin gửi lời cảm ơn đến các Thầy Cô trong Khoa và các bạn học lớp **CE103.P23** đã hỗ trợ, chia sẻ kiến thức và tạo điều kiện để nhóm hoàn thành tốt môn học này.
Mặc dù nhóm đã cố gắng hoàn thành báo cáo một cách nghiêm túc và đầy đủ nhất, nhưng chắc chắn vẫn không tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự góp ý của Thầy để cải thiện và hoàn thiện hơn trong những lần sau.
Nhóm chúng em xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, tháng … năm …

Nhóm sinh viên thực hiện

NHẬN XÉT CỦA GIẢNG VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

Tên viết tắt	Mô tả chi tiết
RTC	Real-Time Clock
GPIO	General-purpose input/output
VĐK	Vi điều khiển
CPU	Central Processing Unit
PWM	Pulse-width Modulation
UART	Universal Asynchronous Receiver/Transmitter
MSB	Most Significant Bit
LSB	Least Significant Bit
IDE	Integrated Development Environment
I2C	Inter-integrated Circuit

THÔNG TIN NHÓM THỰC HIỆN

- Nhóm:BigHK
- Lớp: CE103.P23

STT	Họ và Tên	MSSV
1	Đỗ Xuân Khải	23520672
2	Trần Minh Khải	23520679
3	Huỳnh Vỹ Khang (nhóm trưởng)	23520688
4	Trần Gia Huy	23520646
5	Nguyễn Quang Khải	23520676

CHƯƠNG 1: GIỚI THIỆU

1.1 TỔNG QUAN ĐỀ TÀI

1.1.1 Sơ lược về đồng hồ thời gian thực

- Đồng hồ thời gian thực (RTC - Real Time Clock) là một thiết bị điện tử hiển thị thông tin thời gian chính xác (giờ, phút, giây, ngày, tháng, năm) ngay cả khi hệ thống chính bị tắt nguồn. RTC thường được tích hợp vào các mạch điều khiển, máy tính, hệ thống nhúng và thiết bị điện tử cần theo dõi thời gian liên tục. Nhóm chúng em thực hiện đồng hồ này bằng linh kiện DS1307.

1.1.2 Sơ lược về cách hoạt động của mạch

- Mạch sử dụng IC RTC kết nối với vi điều khiển thông qua giao tiếp I2C. Vi điều khiển có nhiệm vụ giao tiếp, đọc dữ liệu thời gian từ IC RTC và hiển thị lên màn hình LCD. Thêm vào đó mạch còn hiển thị ngày tháng năm. Nguồn cấp cho IC RTC thường được tách biệt và có pin dự phòng để đảm bảo mạch vẫn hoạt động trong trường hợp mất điện

1.1.3 Mục đích nghiên cứu

- Đề tài chúng em nghiên cứu để phục vụ môn học tìm hiểu các phương thức giao tiếp của các vi xử lý-vi điều khiển . Xử lý các thao tác khi lập trình về thời gian thời gian thực , hiểu rõ về vi điều khiển 8051 là một trong những vi điều khiển nền tảng.

1.1.4 Ứng dụng thực tiễn

- **Đồng hồ thời gian thực được ứng dụng rộng rãi trong nhiều lĩnh vực:**
  + **Máy ghi dữ liệu (data logger):** Ghi nhận thời gian đo đạc trong các hệ thống theo dõi môi trường, công nghiệp.
  + **Máy tính, laptop, hệ điều hành:** Duy trì thời gian hệ thống chính xác dù máy bị tắt.
  + **Thiết bị IoT và nhà thông minh:** Tự động bật/tắt thiết bị theo giờ định sẵn.
  + **Máy chấm công, thiết bị điều khiển truy cập:** Ghi lại lịch sử sử dụng một cách chính xác.
  + **Báo thức, chuông trường học, hệ thống hẹn giờ:** Tự động hóa quy trình theo lịch biểu.

=> Việc sử dụng RTC giúp tăng độ tin cậy và tính tự động của hệ thống mà không cần phụ thuộc vào nguồn điện chính.

1.2 MỤC TIÊU ĐỀ TÀI

- Hiểu được phương thức giao tiếp I2C
- Tìm hiểu phương thức để thể hiện trên LCD

- Hiểu rõ nguyên lý hoạt động của IC đồng hồ thời gian thực.
- Lập trình trên mô phỏng và test thử mô phỏng.
- Lập trình vi điều khiển để giao tiếp và xử lý dữ liệu thời gian từ RTC.
- Hiển thị thông tin thời gian thực trên màn hình LCD hoặc các thiết bị tương ứng.
- Đảm bảo mạch có khả năng giữ thời gian ổn định khi mất nguồn chính.

1.3 CẤU TRÚC ĐỒ ÁN

- Đồ án được chia làm 6 chương như sau
    + **Chương 1**: Giới thiệu tổng quan đề tài, mục tiêu, cấu trúc và kế hoạch thực hiện.
    + **Chương 2**: Cơ sở lý thuyết về vi điều khiển, IC RTC, giao tiếp I2C và các linh kiện liên quan.
    + **Chương 3**: Thiết kế và xây dựng mạch phần cứng.
    + **Chương 4**: Phần mềm điều khiển và giao tiếp với RTC.
    + **Chương 5**: Kết quả thực nghiệm và đánh giá.
    + **Chương 6**: Kết luận và hướng phát triển.

1.4 KẾ HOẠCH TRIỂN KHAI

	Thời gian (Dự kiến)	Nội dung công việc
Tuần 1	01/04 – 07/04	Tìm hiểu về đồng hồ thời gian thực (RTC), các loại IC phổ biến
Tuần 2	08/04 – 14/04	Nghiên cứu giao tiếp I2C và vi điều khiển sử dụng trong đề tài
Tuần 3	15/04 – 21/04	Thiết kế sơ đồ nguyên lý, lựa chọn linh kiện
Tuần 4	22/04 – 28/04	Thiết kế mạch in (nếu có) và mô phỏng mạch trên phần mềm
Tuần 5	29/04 – 05/05	Lập trình đọc/ghi dữ liệu thời gian từ IC RTC

Tuần 6	06/05 – 12/05	Hiển thị dữ liệu thời gian lên màn hình LCD/LED
Tuần 7	13/05 – 19/05	Lắp ráp mạch thực tế, kiểm thử, khắc phục lỗi
Tuần 8	20/05 – 26/05	Hoàn thiện báo cáo, tài liệu, chuẩn bị thuyết trình và bảo vệ đề tài

Bảng 1.1: bảng kế hoạch

1.5 CÁC VẤN ĐỀ LIÊN QUAN

- Khi lúc bắt đầu nhóm còn gặp trục trặc thống nhất địa điểm làm.
- Cả nhóm quyết định làm trên github nhưng vẫn còn bị gặp nhiều lỗi (chưa biết repon, commit …)
- Đảm bảo vẫn hoạt động ổn khi hệ thống chính mất nguồn.
- Đảm bảo độ chính xác thời gian của RTC qua thời gian dài.
- Vấn đề về mất dữ liệu hoặc sai lệch thời gian khi mất nguồn.
- Tương thích giữa IC RTC và vi điều khiển sử dụng (xung nhịp, điện áp).
- Lỗi truyền nhận dữ liệu trong giao tiếp I2C.
- Hiển thị thời gian dễ hiểu, đúng định dạng và thân thiện người dùng

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 GIỚI THIỆU CÁC LINH KIỆN SỬ DỤNG TRONG MẠCH

2.1.1 Vi điều khiển AT89S51

Tổng quan vi điều khiển AT89S51

Hình 2.1.1.a Vi điều khiển AT89S51

- Vi điều khiển AT89S51 là một sản phẩm của hãng Atmel (nay thuộc Microchip Technology), thuộc dòng vi điều khiển 8-bit tương thích với kiến trúc MCS-51 của Intel. Đây là một trong những vi điều khiển được sử dụng phổ biến trong lĩnh vực nhúng và tự động hóa nhờ đặc tính đơn giản, dễ học, chi phí thấp và tài liệu phong phú.
- AT89S51 được xây dựng dựa trên kiến trúc Harvard với bộ nhớ chương trình và bộ nhớ dữ liệu tách biệt. Vi điều khiển này tích hợp sẵn 4KB bộ nhớ Flash có thể nạp lại nhiều lần (10000 chu kỳ đọc/ghi theo công bố của nhà sản xuất), 128 byte RAM nội, 32 GPIO chia thành 4 cổng I/O (mỗi cổng 8 bit), 2 bộ định thời (Timer/Counter), 1 cổng giao tiếp nối tiếp (UART), và 5 nguồn ngắt. Ngoài ra, AT89S51 còn có thể hoạt động với dao động thạch anh lên đến 33 MHz, giúp nâng cao tốc độ xử lý trong nhiều ứng dụng thực tế.
- Một điểm nổi bật của AT89S51 là khả năng lập trình trực tiếp thông qua chuẩn nạp ISP (In-System Programming), cho phép người dùng nạp chương trình vào vi điều khiển mà không cần tháo chip khỏi mạch. Điều này rất thuận tiện cho việc phát triển và thử nghiệm sản phẩm.
- Nhờ vào tính linh hoạt và cấu trúc phần cứng đơn giản, AT89S51 thường được sử dụng trong các hệ thống điều khiển tự động nhỏ như: điều khiển đèn, điều khiển động cơ, đo nhiệt độ, thiết bị báo động, giao tiếp với máy tính, v.v. Đây cũng là dòng vi điều khiển được lựa chọn phổ biến trong giáo dục để giảng dạy lập trình và thiết kế hệ thống nhúng.

Bộ xử lý trung tâm

- CPU là thành phần cốt lõi của VĐK, chịu trách nhiệm thực hiện các lệnh từ bộ nhớ chương trình. CPU của AT89S51 hoạt động theo kiến trúc Harvard 8-bit, nghĩa là dữ liệu và lệnh được lưu trữ riêng biệt trong các vùng bộ nhớ khác nhau. Tốc độ thực thi của CPU phụ thuộc vào tần số dao động ngoài và chu kỳ máy.

  Công thức tính chu kỳ máy:

$$T = \frac{12}{Oscillator\ frequency}$$

  T: Chu kì máy
  Oscillator frequency: Tần số xung clock cấp cho VĐK

- Để tính toán thời gian thực thi của một lệnh, ta cần xác định số chu kỳ máy cần thiết để thực thi lệnh đó rồi nhân với chu kỳ máy. Ví dụ, lệnh `ADD A,R0` cần 1 chu kỳ máy để thực thi và tần số thạch anh cung cấp cho VĐK là 24 MHz, ta tính được thời gian thực thi của lệnh này là 0.5 us. Xác định được thời gian thực thi của một lệnh giúp ta dễ dàng triển khai các ứng dụng đòi hỏi sự định thời chính xác. Ví dụ, tạo xung PWM, tạo độ trễ chính xác (Delay), tối ưu hiệu năng chương trình, v.v…

Bộ nhớ

- Bộ nhớ chương trình (Flash): 4KB, cho phép lưu trữ mã chương trình. Đây là bộ nhớ dạng EEPROM có thể ghi/xóa nhiều lần. Bộ nhớ dữ liệu (RAM): 128 byte RAM nội dùng để lưu trữ dữ liệu tạm thời trong quá trình thực thi chương trình. Bộ nhớ đặc biệt (SFR – Special Function Registers): gồm các thanh ghi điều khiển hoạt động của các khối ngoại vi như Timer, UART, ngắt, cổng I/O,…

Tên thanh ghi	Địa chỉ	Giá trị mặc định	Công dụng chính
ACC (A)	E0h	`00000000b`	Thanh ghi tích lũy, dùng trong các phép toán số học, logic, truyền dữ liệu.
B	F0h	`00000000b`	Thanh ghi phụ trợ dùng trong phép nhân và chia (MUL, DIV).
PSW	D0h	`00000000b`	Thanh ghi trạng thái chương trình (Program Status Word), chứa các cờ trạng thái như

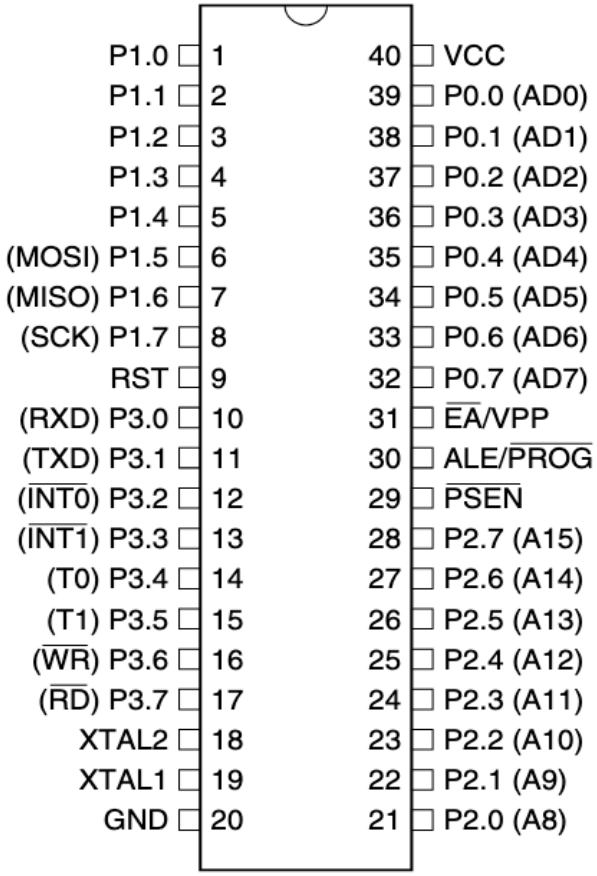
			carry, auxiliary carry, overflow, parity,…
SP	81h	00000111b	Stack Pointer – con trỏ ngăn xếp. Chỉ đến vị trí đầu của ngăn xếp.
DPTR	DP0L:82h DP0H: 83h DP1L: 84h DP1H: 85h	00000000b	Dùng để lưu địa chỉ 16-bit, thường dùng để truy xuất bộ nhớ mã hoặc truy cập dữ liệu ngoài.
P0	80h	11111111b	Port P0. Khi làm việc với bộ nhớ ngoài, P0 còn dùng cho bus dữ liệu.
P1	90h	11111111b	Port P1. Không có chức năng phụ.
P2	A0h	11111111b	Cổng vào/ra P2. Khi dùng bộ nhớ ngoài, P2 là phần cao của bus địa chỉ.
P3	B0h	11111111b	Cổng vào/ra P3. Ngoài chức năng I/O, từng chân còn có chức năng đặc biệt.
TMOD	89h	00000000b	Chọn chế độ hoạt động của Timer 0 và Timer 1.
TCON	88h	00000000b	Điều khiển Timer và ngắt ngoài. Chứa các bit: TF0, TR0, TF1, TR1, IT0, IE0, IT1, IE1.
TH0, TL0	TH0:8Ch TL0: 8Ah	00000000b	Thanh ghi cao/thấp của Timer 0.
TH1, TL1	TH1: 8Dh TL1: 8Bh	00000000b	Thanh ghi cao/thấp của Timer 1.
SCON	98h	00000000b	Điều khiển giao tiếp nối tiếp UART.
SBUF	99h	XXXXXXXXb	Dùng để truyền/nhận dữ liệu trong giao tiếp nối tiếp.
IE	A8h	0X000000b	Interrupt Enable – cho phép từng nguồn ngắt hoạt động.

IP	B8h	XX000000b	Interrupt Priority – thiết lập độ ưu tiên ngắt.
PCON	87h	0XXX0000b	Power Control – điều khiển chế độ tiết kiệm năng lượng (Idle, Power-down).

Bảng 2.1.1.a Các thanh ghi chức năng đặc biệt (SFR) của VĐK AT89S5

I/O Port

- AT89S51 có 4 Port song song 8 bit, gồm: P0, P1, P2 và P3. Mỗi cổng có thể được lập trình làm ngõ vào hoặc ngõ ra. Các cổng này còn đảm nhiệm các chức năng phụ như địa chỉ/ dữ liệu (cổng P0, P2), ngắt ngoài (P3.2, P3.3), truyền nhận UART (P3.0, P3.1), v.v.



Hình 2.1.1.b Sơ đồ chân VĐK AT89S51

Chân số	Tên chân	Chức năng chính	Chức năng phụ/Mô tả
1 – 8	P1.0 – P1.7	Cổng I/O P1 (8 bit)	I/O chung, không có chức năng đặc biệt ngoại trừ khi sử dụng SPI (P1.5–P1.7: MOSI, MISO, SCK).
9	RST	Reset	Đưa mức logic cao trong >2 chu kỳ máy để reset vi điều khiển.
10	P3.0 (RXD)	Cổng I/O	Nhận dữ liệu UART (serial).
11	P3.1 (TXD)	Cổng I/O	Truyền dữ liệu UART (serial).
12	P3.2 ($\overline{INT0}$)	Cổng I/O	Ngắt ngoài 0.
13	P3.3 ($\overline{INT1}$)	Cổng I/O	Ngắt ngoài 1.
14	P3.4 (T0)	Cổng I/O	Đầu vào xung bộ đếm/định thời Timer 0.
15	P3.5 (T1)	Cổng I/O	Đầu vào xung bộ đếm/định thời Timer 1.
16	P3.6 ($\overline{WR}$)	Cổng I/O	Ghi dữ liệu đến bộ nhớ ngoài.
17	P3.7 ($\overline{RD}$)	Cổng I/O	Đọc dữ liệu từ bộ nhớ ngoài.
18	XTAL2	Dao động thạch anh	Kết nối đến thạch anh hoặc mạch dao động RC.
19	XTAL1	Dao động thạch anh	Nguồn đầu vào cho bộ dao động nội.
20	GND	Nối đất	Nối đất (0V).
21 – 28	P2.0 – P2.7	Cổng I/O	Ngoài ra còn là byte địa chỉ cao khi dùng bộ nhớ ngoài (A8 – A15).
29	$\overline{PSEN}$	Program Store Enable	Cho phép đọc mã chương trình từ bộ nhớ ngoài.
30	ALE/$\overline{PROG}$	Address Latch Enable	Chốt địa chỉ byte thấp / Lập trình Flash.
31	$\overline{EA}$/VPP	External Access	Mức thấp: dùng bộ nhớ chương trình ngoài Mức cao: dùng bộ nhớ trong.

32 – 39	P0.7 – P0.0	Cổng I/O	Ngoài ra là bus địa chỉ/dữ liệu khi dùng bộ nhớ ngoài (AD7–AD0).
40	VCC	Nguồn	Nguồn cấp điện (thường 5V).

Bảng 2.1.1.b mô tả chức năng các chân AT89S51

Bộ định thời (Timer/Counter)

- Vi điều khiển được trang bị 2 bộ định thời: Timer 0 và Timer 1, đều là các bộ đếm 16 bit. Chúng có thể hoạt động ở nhiều chế độ khác nhau như: chế độ định thời (Timer), chế độ đếm xung ngoài (Counter), và tạo ngắt định kỳ. Đây là thành phần quan trọng trong việc xây dựng các ứng dụng yêu cầu đo thời gian hoặc điều chế xung (PWM). Timer 0 và Timer 1 được điều khiển và cấu hình thông qua thanh ghi điều khiển bộ đếm thời gian (TCON) và thanh ghi cấu hình chế độ định thời (TMOD).

TCON Địa chỉ: 88H Giá trị mặc định: 00000000B

MSB LSB

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bảng 2.1.1.c Thanh ghi TCON

TF1: Cờ tràn Timer 1. Được set lên 1 khi Timer 1 chuyển từ toàn bộ số 1 sang số 0, xóa khi chương trình vector ngắt được thực thi tại địa chỉ 001BH.

TR1: Bit điều khiển Timer 1. Ghi 1 để khởi chạy Timer 1, ghi 0 để dừng.

TF0: Cờ tràn Timer 0. Được set lên 1 khi Timer 0 chuyển từ toàn bộ số 1 sang số 0, xóa khi chương trình vector ngắt được thực thi tại địa chỉ 000Bh.

TR0: Bit điều khiển Timer 0. Ghi 1 để khởi chạy Timer 0, ghi 0 để dừng.

IE1: Cờ ngắt ngoài 1. Được set lên 1 khi tín hiệu chuyển đổi từ mức cao sang mức thấp (cạnh xuống) trên chân 3.3 ($\underline{INT1}$), xóa khi chương trình vector ngắt được thực thi tại địa chỉ 0013H.

IT1: Bit cấu hình loại tín hiệu ngắt ngoài 1. Ghi 1 để cấu hình ngắt khi có cạnh

xuống, ghi 0 để cấu hình ngắt mức tín hiệu thấp.

IE0: Cờ ngắt ngoài 0. Được set lên 1 khi tín hiệu chuyển đổi từ mức cao sang mức thấp (cạnh xuống) trên chân 3.2 ($\underline{INT0}$), xóa khi chương trình vector ngắt được thực thi tại địa chỉ 0003H.

IT0: Bit cấu hình loại tín hiệu ngắt ngoài 0. Ghi 1 để cấu hình ngắt khi có cạnh xuống, ghi 0 để cấu hình ngắt mức tín hiệu thấp.

TMOD Địa chỉ: 89H Giá trị mặc định: 00000000B

MSB LSB

7	6	5	4	3	2	1	0
Gate1	C/$\underline{T}$1	M1_1	M1_0	Gate0	C/$\underline{T}$0	M0_1	M0_0
Timer 1				Timer 0			

Bảng 2.1.1.d Thanh ghi TMOD

Gate1: Điều khiển Timer 1 bởi chân $\underline{INT1}$ (P3.3) nếu set = 1 và TR1 = 1.Set = 0: bỏ qua $\underline{INT1}$.

C/$\underline{T}$1: Chọn chế độ hoạt động cho Timer 1:0 = Timer (đếm xung nội), 1 = Counter (đếm xung ngoài tại T1/P3.5).

M1_1: Chọn chế độ hoạt động của Timer 1.

M1_0: Chọn chế độ hoạt động của Timer 1.

Gate0: Điều khiển Timer 0 bởi chân $\underline{INT0}$ (P3.2) nếu set = 1 và TR0 = 1.Set = 0: bỏ qua $\underline{INT0}$.

C/$\underline{T}$0: Chọn chế độ hoạt động cho Timer 0:0 = Timer (đếm xung nội), 1 = Counter (đếm xung ngoài tại T0/P3.4).

M0_1: Chọn chế độ hoạt động của Timer 0.

M0_0: Chọn chế độ hoạt động của Timer 0.

M0	M1	Chế độ	Chế độ hoạt động
0	0	0	Chế độ định thời 13 bit: bộ định thời / bộ đếm 8 bit THx và bộ chia trước tần số 5 bit TLx.
0	1	1	Chế độ định thời 16 bit: bộ đếm định thời / bộ đếm 16 bit THx và TLx được xếp tầng; Không có bộ chia trước tần số.
1	0	2	Chế độ định thời 8 bit: bộ định thời / bộ đếm tự động tải lại 8 bit; THx giữ một giá trị sẽ được tải lại vào TLx mỗi khi nó tràn ra.
1	1	3	Chế độ định thời chia tách: Timer 0 được chia tách thành 2 timer 8 bit độc lập, Timer 1 bị vô hiệu hóa.

Bảng 2.1.1.e Các chế độ hoạt động của Timer trong VĐK AT89S51

Giao tiếp tuần tự (UART)

- Các máy tính phải có khả năng giao tiếp với nhau trong một hệ thống đa bộ xử lý phân tán hiện đại. Một cách hiệu quả về chi phí để giao tiếp là gửi và nhận dữ liệu bit một cách tuần tự. Giao tiếp tuần tự có nghĩa các bit được dịch chuyển thành một hàng ra một chân duy nhất. Ưu điểm của cách này là nó sử dụng ít chân trên VĐK, ít dây kết nối hơn từ đó tối ưu chi phí phần cứng. VĐK AT89S51 được trang bị một mạch giao tiếp tuần tự sử dụng thanh ghi SBUF để lưu trữ dữ liệu. Thanh ghi SCON kiểm soát hoạt động của giao tiếp tuần tự, thanh ghi PCON kiểm soát tốc độ truyền dữ liệu và chân RXD (P3.0) và chân TXD (P3.1) kết nối với mạng lưới dữ liệu tuần tự.
- Thanh ghi SBUF về mặt vật lý là hai thanh ghi riêng biệt. Một thanh ghi chỉ ghi và được sử dụng cho việc lưu trữ dữ liệu được gửi ra ngoài thông qua chân TXD. Thanh ghi còn lại chỉ đọc và lưu trữ dữ liệu nhận được từ bên ngoài thông qua chân RXD. Cả hai đều dùng địa chỉ 99H loại trừ lẫn nhau.
- AT89S51 có 4 chế độ giao tiếp tuần tự được cấu hình bằng cách ghi các bit SMx trong thanh ghi SCON. Baud rate được xác định bằng chế độ được chọn. UART truyền nhận dữ liệu từng bit một theo chuẩn **không đồng bộ** (không dùng xung clock chung), vì thế khi hai hay nhiều thiết bị muốn giao tiếp với nhau qua UART cần cấu hình Baud rate giống nhau.

SCON Địa chỉ: 98H Giá trị mặc định: 00000000B

MSB LSB

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Bảng 2.1.1.f Thanh ghi SCON

SM0: Chọn chế độ truyền (bit cao)

SM1: Chọn chế độ truyền (bit thấp)

SM2: Dùng trong chế độ đa xử lý

REN: Cho phép nhận dữ liệu (1: kích hoạt RX)

TB8: Bit thứ 9 truyền (chế độ 2, 3)

RB8: Bit thứ 9 nhận (chế độ 2, 3)

TI: Cờ truyền xong (set = 1 khi byte truyền xong)

RI: Cờ nhận xong (set = 1 khi nhận đủ 1 byte)

SM0	SM1	Mode	Mô tả
0	0	0	Shift register; baud = f/12
0	1	1	8-bit UART; baud = variable
1	0	2	9-bit UART; baud = f/32 hoặc f/64
1	1	3	9-bit UART; variable

Bảng 2.1.1.g Các chế độ truyền tuần tự

PCON Địa chỉ: 87H Giá trị mặc định: 0XXX0000B

MSB LSB

7	6	5	4	3	2	1	0
SMOD	-	-	-	GF1	GF0	PD	IDL

Bảng 2.1.1.h Thanh ghi PCON

SMOD: Bit cấu hình baud rate. Ghi 1 để nhân đôi baud rate Timer 1 ở chế độ 1, 2 và 3. Ghi 0 để sử dụng baud rate của Timer 1.

GF1: Bit cờ đa dụng 1 (General flag).

GF0: Bit cờ đa dụng 2 (General flag).

PD: Bit Power Down.

IDL: Bit chế độ Idle.

- Để các vi điều khiển có thể giao tiếp với nhau qua UART, cần thống nhất sử dụng chung một baud rate. Các baud rate tiêu chuẩn thường được sử dụng là 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, và 256000. Baud rate tiêu chuẩn phổ biến nhất là 9600 và đây cũng là baud rate mà các nhà phát triển thường sử dụng khi làm việc với dòng VĐK 8051. Trong vi điều khiển AT89S51 mỗi chế độ truyền tuần tự sẽ có cách tính baud rate cụ thể.

- Chế độ truyền tuần tự 0 (Shift register mode), cổng tuần tự trên VĐK AT89S51 hoạt động giống như một thanh ghi dịch, gửi và nhận dữ liệu thông qua chân RXD trong khi đó chân TXD có nhiệm vụ tạo xung clock. Tốc độ truyền được cấu hình cố định là $\frac{f}{12}$ với f là tần số xung clock cung cấp cho VĐK.

- Chế độ truyền tuần tự 1 (Standard UART), Timer 1 được sử dụng để tạo baud rate bằng cách sử dụng cờ tràn của timer để xác định tần số baud. Timer 1 thường được cấu hình ở timer mode 2 (timer tự nạp lại 8 bit) để tạo tần số baud với công thức như sau:

$$f_{baud} = \frac{2^{SMOD}}{32d} \times \frac{oscillator\ frequency}{12d \times [256d\ -\ (TH1)]}$$

f_{baud}: Tần số baud
- $SMOD$: Bit điều khiển trong thanh ghi PCON, có thể ghi 0 hoặc 1 để nhân đôi tần số baud.
$Oscillator\ frequency$: Tần số xung clock cấp cho VĐK
$TH1$: giá trị thanh ghi cao của Timer 1.

- Nếu Timer 1 không được cấu hình ở mode 2 thì tần số baud được xác định như sau:

$$f_{baud} = \frac{2^{SMOD}}{32d} \times f_{t1o}$$

f_{baud}: Tần số baud
- $SMOD$: Bit điều khiển trong thanh ghi PCON, có thể ghi 0 hoặc 1 để nhân đôi tần số baud.
f_{t1o}: Tần số tràn của Timer 1.

Ví dụ, đối với VĐK AT89S51 để cấu hình UART hoạt động với baud rate 9600, ta reset bit SMOD xuống 0, cấu hình timer 1 ở mode 2 và set giá trị thanh ghi TH1 = 253d và cấp cho VĐK tần số xung clock 11.0592 MHz.

- Chế độ truyền tuần tự 2 (Multiprocessor Mode) hoạt động giống mode 1 nhưng trong chế độ này 11 bit sẽ được trao đổi thay vì 8 bit: 1 bit bắt đầu, 9 bit dữ liệu và 1 bit kết thúc. Bit thứ 9 sẽ được sao chép từ bit TB8 trong thanh ghi SCON khi thực hiện gửi dữ liệu và được lưu trong bit RB8 của thanh ghi

SCON khi nhận dữ liệu. Bit bắt đầu và bit kết thúc bị loại bỏ.

Baud rate được xác định như sau:

$$f_{baud} = \frac{2^{SMOD}}{64d} \times oscillator\ frequency$$

f_{baud}: Tần số baud

- $SMOD$: Bit điều khiển trong thanh ghi PCON, có thể ghi 0 hoặc 1 để nhân đôi tần số baud.
  $Oscillator\ frequency$: Tần số xung clock cấp cho VĐK

- Chế độ truyền tuần tự 3 tương tự như chế độ 2 nhưng xác định tuần số baud thì giống chế độ 1, sử dụng timer 1 để tạo tần số giao tiếp.

Ngắt (Interrupt)

- AT89S51 có tổng cộng 5 nguồn ngắt: 2 ngắt ngoài ($\underline{INT0}$ và $\underline{INT1}$), 2 ngắt định thời (Timer 0 và 1) và ngắt cổng tuần tự. Các nguồn ngắt có thể được bật hoặc tắt một cách độc lập bằng cách ghi hoặc xóa các bit trên thanh ghi chức năng đặc biệt Interrupt Enable (IE). Ngoài ra, thanh ghi IE còn bao gồm một bit toàn cục EA, bit này có chức năng vô hiệu hóa tất cả ngắt cùng một lúc.

IE Địa chỉ: 0A8H Giá trị mặc định: 0X000000B

MSB LSB

7	6	5	4	3	2	1	0
EA	-	-	-	ES	ET1	ET0	EX0

Enable Bit = 1 → kích hoạt ngắt
Enable Bit = 0 → vô hiệu hóa ngắt

Bảng 2.1.1.i Thanh ghi IE

EA: Vô hiệu hóa tất cả các ngắt. EA = 0, không có ngắt nào được bật. EA = 1, các ngắt có thể được kích hoạt hoặc vô hiệu hóa một cách độc lập bằng cách ghi vào các bit tương ứng với ngắt đó.

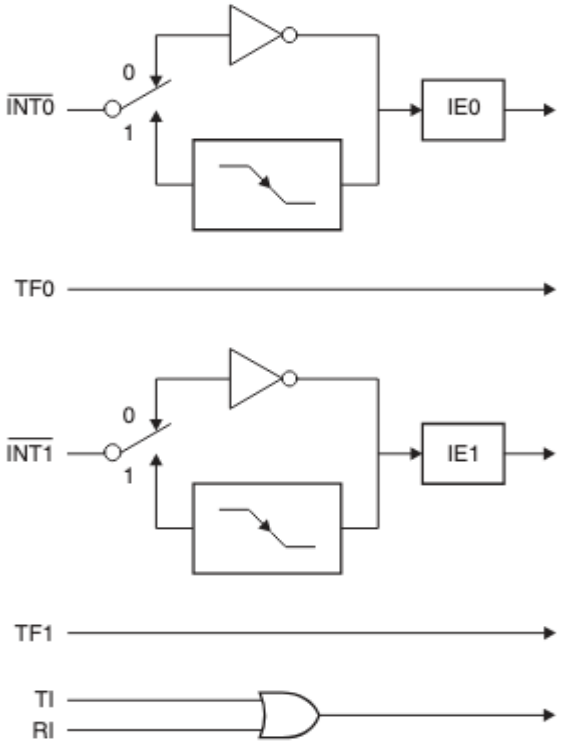
ES: Bit kích hoạt ngắt cổng tuần tự.

ET1: Bit kích hoạt ngắt Timer 1.

EX1: Bit kích hoạt ngắt ngoài 1.

ET0: Bit kích hoạt ngắt Timer 0.

EX0: Bit kích hoạt ngắt



Hình 2.1.1.c Sơ đồ nguồn ngắt của VĐK AT89S51

- Cơ chế ngắt giúp cho VĐK phản ứng nhanh và chính xác hơn với các sự kiện ngoại vi mà không cần kiểm tra liên tục (polling), từ đó tiết kiệm được chi phí xử lý và tài nguyên CPU. Khi sử dụng nhiều ngắt cần có cơ chế để xử lý trường hợp nhiều ngắt xảy ra cùng lúc. Độ ưu tiên của các ngắt được cấu hình thông qua thanh ghi Interrupt Priority (IP).

IP Địa chỉ: 0B8H Giá trị mặc định: XX000000B

MSB LSB

7	6	5	4	3	2	1	0
-	-	PT2	PS	PT1	PX1	PT0	PX0

Ghi chú: độ ưu tiên có thể là 1 (cao nhất) hoặc 0 (thấp nhất)

Bảng 2.1.1.j Thanh ghi IP

PT2: Dành cho mục đích sử dụng sau này.

PS: Độ ưu tiên của ngắt cổng tuần tự.

PT1: Độ ưu tiên ngắt tràn của Timer 1.

PX1: Độ ưu tiên ngắt ngoài 1.

PT0: Độ ưu tiên ngắt tràn của Timer 0.

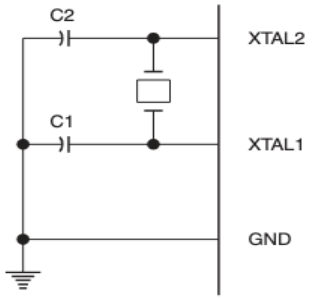
PX0: Độ ưu tiên ngắt ngoài 0.

- Khi có nhiều ngắt đồng thời xảy, thanh ghi IP giúp xác định ngắt nào sẽ được phục vụ trước bằng cách gán mức độ ưu tiên cao hơn (1) hoặc thấp hơn (0). Các chương trình ngắt có mức độ ưu tiên thấp hơn có thể bị ngắt bởi các ngắt có mức độ ưu tiên cao hơn. Khi nhiều ngắt có cùng mức độ ưu tiên xảy ra đồng thời, VĐK AT89S51 sẽ thực thi theo thứ tự mặc định: INT0 > T0 > INT1 > T1 > Serial.

Dao động hệ thống (Clock Oscillator)

- Chân XTAL1 và chân XTAL2 trên VĐK AT89S51 là input và output tương ứng của một bộ khuếch đại nghịch đảo bên trong VĐK, được sử dụng như bộ tạo dao động tích hợp trên chip (on-chip oscillator). Có thể kết nối hai chân này với thạch anh hoặc một mạch dao động RC. Để kết nối thiết bị với một nguồn xung clock bên ngoài ta cần thả nổi chân XTAL2 và kết nối nguồn dao động vào chân XTAL1. Có thể cung cấp tần số xung clock cho VĐK AT89S51 từ 0 đến 33 MHz. Tần số xung clock phổ biến thường được sử dụng là 11.0592 MHz để tương thích với chuẩn truyền thông nối tiếp. Trong đồ án này, tần số được lựa chọn là 24 MHz để thuận tiện cho việc triển khai giao thức I2C và định thời.

- Khi kết nối thạch anh hoặc một mạch dao động RC với VĐK ta cần thêm 2 tụ điện giúp lọc nhiễu, tăng độ chính xác của tần số và hỗ trợ dao động ổn định cho thạch anh. Giá trị khuyên dùng cho các tụ điện này là từ 20pF đến 40pF.



Hình 2.1.1.d Mạch dao động thạch anh của AT89S51

Tập lệnh

- Tập lệnh của một máy tính là thành phần rất quan trọng và không thể thiếu khi

muốn phát triển một ứng dụng thực tế. Để các nhà phát triển có thể tạo ra được các chương trình hiệu suất cao, cần phải nắm rõ tập lệnh của một máy tính, hiểu được chức năng của chúng và biết cách phối hợp các lệnh một cách có hiệu quả. Vi điều khiển AT89S51 sử dụng tập lệnh của họ 8051 do Intel phát triển. Tập lệnh này là ngôn ngữ máy (hoặc hợp ngữ - assembly) giúp vi điều khiển thực hiện các tác vụ cụ thể. Tập lệnh của họ 8051 thuộc vào loại CISC (Complex Instruction Set Computer), đặc điểm của tập lệnh này là có các lệnh đơn lẻ thực hiện nhiều tác vụ đồng thời. Tập lệnh của AT89S51 bao gồm khoản 111 lệnh và chia thành các nhóm như trong bảng.

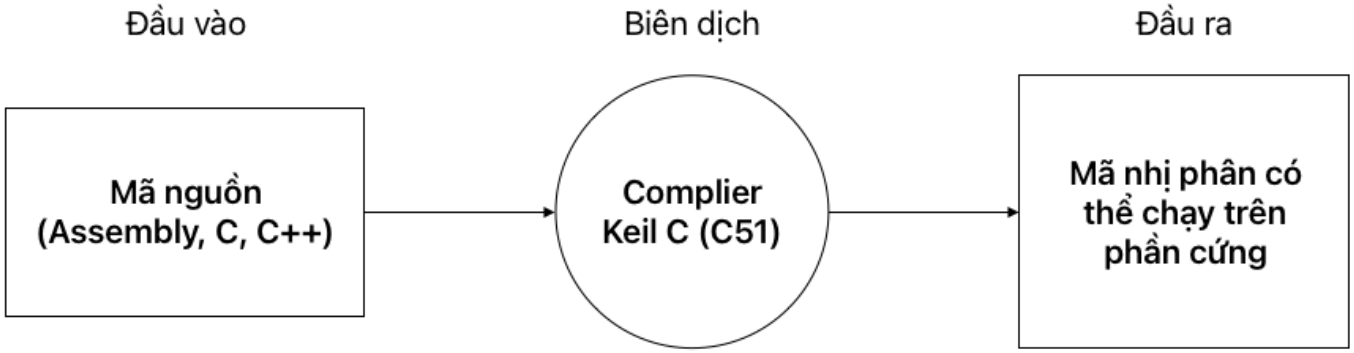
Nhóm lệnh	Mô tả	Ví dụ
Lệnh dịch chuyển dữ liệu	Di chuyển dữ liệu giữa các thanh ghi, bộ nhớ, I/O.	MOV, PUSH, POP, XCH
Lệnh xử lý số học	Cộng, trừ, nhân, chia,…	ADD, SUBB, MUL, DIV, INC, DEC
Lệnh logic	Thực hiện AND, OR, XOR, NOT,…	ANL, ORL, XRL, CPL, CLR
Lệnh rẽ nhánh	Thực hiện nhảy có điều kiện hoặc không điều kiện	SJMP, LJMP, JC, JNC, DJNZ, CALL, RET
Lệnh điều khiển bit	Xử lý từng bit riêng lẻ trong RAM hoặc SFR	SETB, CLR, CPL, ANL, ORL, RL, RLC, RR, RRC
Lệnh điều khiển hệ thống	Dùng để ngừng chương trình hoặc điều khiển trạng thái CPU	NOP, RET, RETI, ACALL

Bảng 2.1.1.k Các nhóm lệnh của VĐK AT89S51

Một số lệnh phổ biến	
Lệnh	**Chức năng**
MOV A, #55H	Gán hằng 55H vào thanh ghi tích lũy A
ADD A, R1	Cộng giá trị R1 vào A
SJMP LABEL	Nhảy đến nhãn LABEL (nhảy ngắn)
JNZ LABEL	Nếu A ≠ 0, nhảy đến LABEL
SETB P1.0	Đưa mức logic 1 lên chân P1.0
CLR C	Xóa cờ Carry
ACALL SUB	Gọi chương trình con tại SUB (nhảy tương đối ngắn)
RET	Kết thúc chương trình con

Bảng 2.1.1.l Một số lệnh phổ biến được dùng để lập trình VĐK AT89S51

- Để có thể biên dịch mã nguồn assembly hay bất kỳ ngôn ngữ cấp cao nào (thường là ngôn ngữ C) ta cần sử dụng một trình biên dịch để dịch mã nguồn sang mã máy có thể thực thi được trên phần cứng AT89S51. Các compiler phổ biến có thể kể đến như Keil C (C51), SDCC (Small Device C Compiler), MIDE-51 (MiniIDE), MCU 8051 IDE, IAR Embedded Workbench,... Trong đồ án này sử dụng trình biên dịch Keil C (C51) để biên dịch mã nguồn assembly sang mã máy. Trình biên dịch Keil C (C51) được tích hợp sẵn trong IDE (Integrated Development Environment) μVision rất thuận tiện để sử dụng cho việc phát triển chương trình chạy trên AT89S51 mà không cần tự cài đặt nhiều.



Hình 2.1.1.e Biên dịch mã nguồn sang mã máy

Lập trình AT89S51

- Vi điều khiển AT89S51 hỗ trợ hai chế độ lập trình: chế độ song song (Parallel Mode) và chế độ nối tiếp (Serial Mode). Chế độ lập trình nối tiếp còn gọi là ISP (In-System Programming) cho phép lập trình qua cổng SPI. Chế độ lập trình nối tiếp cũng là chế độ phổ biến được các nhà phát triển sử dụng để lập trình AT89S51 vì sự thuận tiện và dễ cài đặt, có thể sử dụng mạch ISP như USBISP để lập trình. Đồ án này sử dụng chế độ lập trình nối tiếp để nạp chương trình vào bộ nhớ Flash của VĐK AT89S51. Để có thể sử dụng chế độ lập trình nối tiếp cần tuân theo các bước sau:

1. Cấp nguồn cho vi điều khiển
   a. Cấp hiệu điện thế giữa hai chân VCC và GND.
   b. Kéo chân RST lên mức cao.
2. Kích hoạt lập trình tuần tự bằng cách gửi lệnh Programming Enable vào chân MOSI/P1.5. Tần số xung nhịp cấp cho chân SCK/P1.7 cần nhỏ hơn CPU clock chia cho 16.
3. Bộ nhớ Code được lập trình theo từng Byte kể cả trong Byte hay Page mode. Chu kỳ ghi thường nhỏ hơn 0.5 ms ở mức điện áp 5V.
4. Bất kỳ địa chỉ nào trong bộ nhớ Code đều có thể được xác thực tính toàn vẹn của dữ liệu (Verify), bằng cách dùng lệnh Read VĐK sẽ trả về

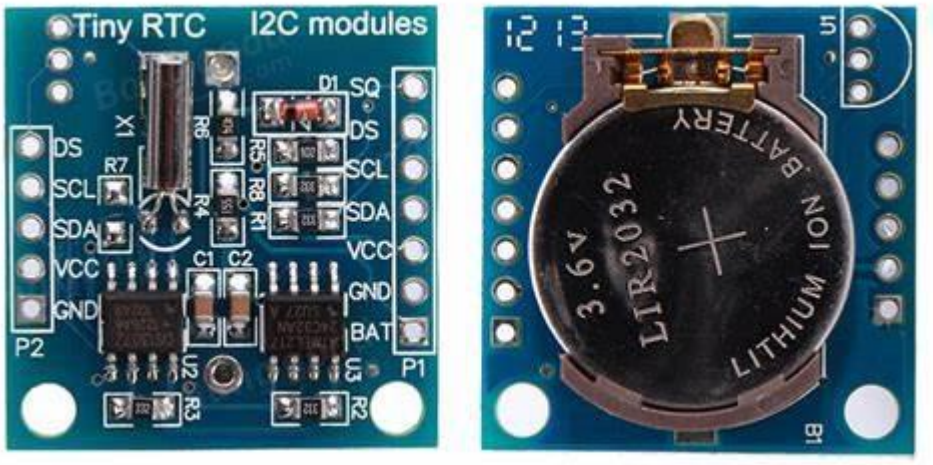
nội dung tại vùng nhớ có địa chỉ tương ứng thông qua chân MISO/P1.6.

5. Sau khi nạp code vào bộ nhớ Flash có thể kéo chân RST về mức thấp để đưa thiết bị về trạng thái hoạt động bình thường.

Hình 2.1.1.f Mạch nạp USBISP

2.1.2 IC thời gian thực DS1307

Tổng quan về IC DS1307



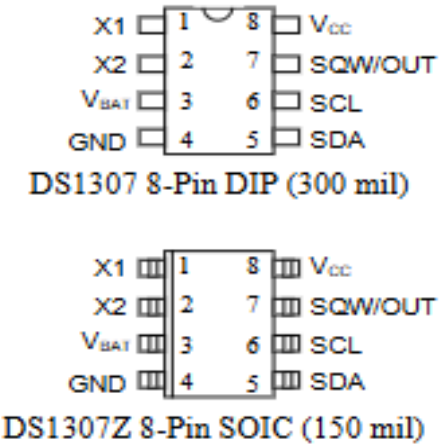
Hình 2.1.2.a Module DS1307

- **DS1307** là một IC thời gian thực (RTC – Real Time Clock) của hãng **Maxim Integrated (nay thuộc Analog Devices)**. Đây là một IC rất phổ biến dùng để giữ thời gian thực tế trong các hệ thống nhúng nhờ vào khả năng lưu trữ thời gian chính xác và hoạt động với nguồn pin dự phòng khi mất nguồn chính. DS1307 được sử dụng rộng rãi trong các ứng dụng như đồng hồ điện tử, hệ thống ghi dữ

liệu, thiết bị đo đạc, hệ thống báo giờ và các sản phẩm tự động hóa dân dụng và công nghiệp.

- **DS1307** sử dụng giao tiếp **I2C (Inter-Integrated Circuit)** để trao đổi dữ liệu với vi điều khiển, giúp việc kết nối đơn giản với chỉ hai đường tín hiệu: **SCL (Clock)** và **SDA (Data)**. IC này có thể lưu giữ **giờ, phút, giây, ngày, tháng, năm và thứ trong tuần**, với **tự động điều chỉnh năm nhuận** và định dạng thời gian **12/24 giờ**.
- Một ưu điểm nổi bật là DS1307 tích hợp **mạch dao động nội sử dụng thạch anh 32.768 kHz** và có khả năng hoạt động liên tục với **pin dự phòng (thường là pin CR2032 3V)** giúp duy trì thời gian ngay cả khi nguồn chính bị ngắt. Nhờ đó, DS1307 rất phù hợp trong các ứng dụng yêu cầu duy trì thời gian lâu dài

Cấu trúc chân DS1307 (8 chân)



Hình 2.1.2.a Sơ đồ cấu trúc chân DS1307

Chân	Ký hiệu	Mô tả
1	X1	Kết nối với đầu vào của thạch anh 32.768 kHz
2	X2	Kết nối với đầu ra của thạch anh 32.768 kHz
3	VBAT	Nối với pin dự phòng (thường là 3V)
4	GND	Nối đất
5	SDA	Dữ liệu I2C – truyền nhận dữ liệu
6	SCL	Xung clock I2C – đồng bộ dữ liệu
7	SQW/OUT	Tín hiệu đầu ra dạng xung vuông có thể lập trình (1Hz, 4kHz, 8kHz, 32kHz) hoặc ngõ ra logic
8	VCC	Nguồn cấp chính (2V – 5.5V)

Bảng 2.1.2.a Mô tả cấu trúc chân DS1307

Sơ đồ địa chỉ RTC và RAM

- Sơ đồ địa chỉ cho các thanh ghi RTC và RAM của DS1307 được minh họa qua bảng sau:

00H	Giây (Seconds)
01H	Phút (Minutes)
02H	Giờ (Hours)
03H	Ngày trong tuần (Day)
04H	Ngày trong tháng (Date)
05H	Tháng (Month)
06H	Năm (Year)
07H	Điều khiển (Control)
08H – 3FH	RAM 56 × 8 Byte

Bảng 2.1.2.b sơ đồ địa chỉ thanh ghi và RAM của DS1307

- Các thanh ghi thời gian thực nằm trong các địa chỉ từ 00h đến 07h. Các thanh ghi RAM nằm trong các địa chỉ từ 08h đến 3Fh. Trong một lần truy cập nhiều byte, khi con trỏ địa chỉ đạt đến 3Fh (cuối vùng RAM), nó sẽ quay lại vị trí 00h, là điểm bắt đầu của vùng thanh ghi thời gian.

CLOCK và CALENDAR

- Thông tin về thời gian và lịch được lấy bằng cách đọc các byte thanh ghi thích hợp. Các thanh ghi thời gian thực được minh họa trong hình sau:

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	X	10 MINUTES			MINUTES				00–59
	X	12/24	10 HR A/P	10 HR	HOURS				01–12 00–23
	X	X	X	X	X	DAY			1–7
	X	X	10 DATE		DATE				01–28/29 01–30 01–31
	X	X	X	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

Bảng 2.1.2.c thanh ghi thời gian thực

- Thời gian và lịch được thiết lập hoặc khởi tạo bằng cách ghi vào các byte thanh ghi tương ứng. Nội dung của các thanh ghi thời gian và lịch được lưu trữ dưới định dạng BCD (Binary-Coded Decimal - Thập phân Mã nhị phân). Bit 7 của thanh ghi 0 là bit Dừng Dao động (CH – Clock Halt). Khi bit này được đặt là 1, dao động bị vô hiệu hóa. Khi bit này được xóa về 0, dao động được kích hoạt.

  Lưu ý: trạng thái nguồn khởi tạo ban đầu của tất cả các thanh ghi là không xác định. Do đó, rất quan trọng phải bật dao động (bit CH = 0) trong quá trình khởi tạo ban đầu.

- DS1307 có thể hoạt động được ở chế độ 12 giờ hoặc 24 giờ. Bit 6 của thanh ghi giờ được định nghĩa là bit chọn chế độ 12 giờ hoặc 24 giờ. Khi bit này ở mức cao (1), chế độ 12 giờ được chọn. Trong chế độ 12 giờ, bit 5 là bit AM/PM với mức logic cao biểu thị PM. Trong chế độ 24 giờ, bit 5 là bit 10 giờ thứ hai (20–23 giờ).
- Khi bắt đầu bằng một tín hiệu START 2 dây, thời gian hiện tại được chuyển sang một tập hợp thanh ghi thứ hai. Thông tin thời gian sẽ được đọc từ các thanh ghi phụ này, trong khi đồng hồ vẫn tiếp tục chạy. Điều này giúp loại bỏ nhu cầu đọc lại các thanh ghi trong trường hợp thanh ghi chính được cập nhật trong quá trình đọc

Thanh ghi điều khiển

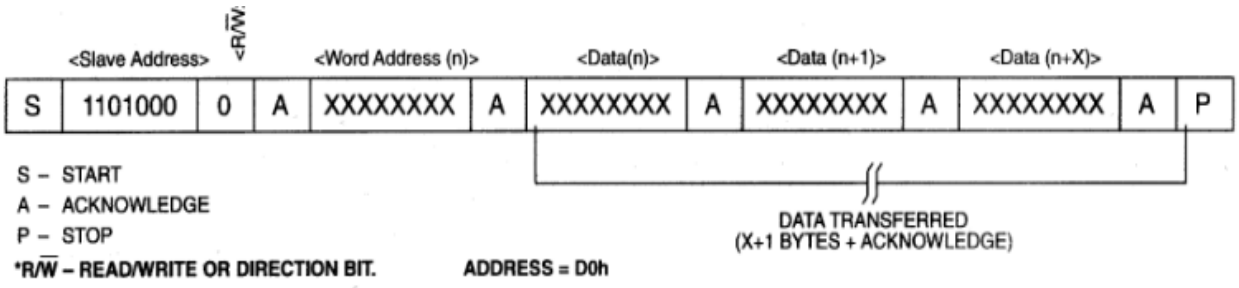
- Thanh ghi điều khiển của DS1307 được sử dụng để điều khiển hoạt động của chân SQW/OUT.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	X	X	SQWE	X	X	RS1	RS0

Bảng 2.1.2.d Thanh ghi điều khiển

- **OUT (Điều khiển đầu ra):**
  Bit này điều khiển mức logic tại chân **SQW/OUT** khi tín hiệu sóng vuông bị vô hiệu hóa.
  Nếu **SQWE = 0**, mức logic tại chân SQW/OUT là **1 nếu OUT = 1**, và là **0 nếu OUT = 0**.
- **SQWE (Kích hoạt Sóng Vuông - Square Wave Enable):**
  Bit này, khi được đặt bằng 1 (logic cao), sẽ bật đầu ra dao động (oscillator).
  Tần số của sóng vuông đầu ra phụ thuộc vào giá trị của hai bit **RS0** và **RS1**.
- **RS (Chọn Tốc Độ - Rate Select):**
  Hai bit này điều khiển tần số của sóng vuông đầu ra khi nó được bật.
  Bảng liệt kê các tần số sóng vuông có thể được chọn bằng các bit RS.

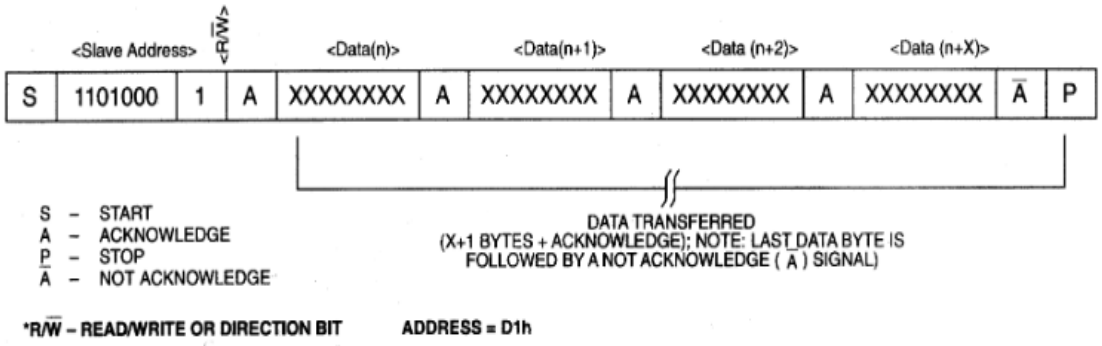
Ghi dữ liệu - Chế độ SLAVE RECEIVER



Hình 2.1.2.b mô tả ghi dự liệu của RTC

- Dữ liệu nối tiếp và xung đồng hồ được nhận thông qua các chân **SDA** và **SCL**. Sau mỗi byte được nhận, một bit xác nhận (**acknowledge**) sẽ được truyền. Các điều kiện **START** và **STOP** được nhận biết là điểm bắt đầu và kết thúc của một phiên truyền nối tiếp.Việc nhận biết địa chỉ được phần cứng thực hiện sau khi nhận được địa chỉ thiết bị và *bit chỉ hướng truyền* (direction bit) . Byte địa chỉ là byte đầu tiên được nhận sau khi điều kiện START được tạo bởi bộ điều khiển chính (master).Byte địa chỉ bao gồm địa chỉ 7-bit của DS1307 là **1101000**, theo sau là bit *chỉ hướng truyền (R/W)* với giá trị là **0** trong chế độ ghi.Sau khi nhận và giải mã byte địa chỉ, thiết bị xuất ra một tín hiệu **ACKNOWLEDGE** trên đường SDA.Sau khi DS1307 xác nhận (acknowledge) địa chỉ thiết bị cộng với bit ghi, master sẽ truyền **địa chỉ thanh ghi** đến DS1307. Điều này sẽ thiết lập **con trỏ thanh ghi (register pointer)** trong DS1307.Master sau đó sẽ bắt đầu truyền từng byte dữ liệu, và DS1307 sẽ xác nhận mỗi byte nhận được.Master sẽ tạo một điều kiện STOP để kết thúc quá trình ghi dữ liệu.

Đọc dữ liệu - Chế độ SLAVE TRANSMITTER



Hình 2.1.2.c mô tả đọc dự liệu của RTC

- Byte đầu tiên được nhận và xử lý tương tự như trong chế độ slave receiver. Tuy nhiên, trong chế độ này, bit *chỉ hướng truyền* sẽ cho biết rằng hướng truyền dữ liệu là ngược lại.Dữ liệu nối tiếp được truyền trên đường SDA bởi DS1307, trong khi tín hiệu xung clock được đưa vào từ SCL. Các điều kiện START và STOP được nhận diện là điểm bắt đầu và kết thúc của một phiên truyền nối tiếp .Byte địa chỉ là byte đầu tiên được nhận sau khi điều kiện START được tạo bởi master.

Byte địa chỉ chứa địa chỉ 7-bit của DS1307 là 1101000, theo sau là bit *chỉ hướng truyền (R/W)* có giá trị là 1 cho chế độ đọc.Sau khi nhận và giải mã byte địa chỉ, thiết bị xuất ra một tín hiệu acknowledge trên đường SDA.DS1307 sau đó bắt đầu truyền dữ liệu, bắt đầu từ địa chỉ thanh ghi được chỉ định bởi con trỏ thanh ghi (register pointer). Nếu con trỏ thanh ghi chưa được ghi trước khi bắt đầu chế độ đọc, thì địa chỉ đầu tiên được đọc sẽ là địa chỉ cuối cùng được lưu trong con trỏ thanh ghi. DS1307 phải nhận một tín hiệu Not Acknowledge (NACK) để kết thúc quá trình đọc.

Cách hoạt động của DS1307

- Khi được cấp nguồn chính (**VCC**), DS1307 hoạt động bình thường và liên tục cập nhật thời gian.Khi mất nguồn chính, nếu có kết nối pin dự phòng (**VBAT**), DS1307 sẽ tự động chuyển sang chế độ tiêu thụ năng lượng cực thấp và vẫn tiếp tục duy trì thời gian chính xác.Thời gian được lưu trữ ở dạng nhị phân thập phân BCD (Binary-Coded Decimal).Giao tiếp I2C cho phép vi điều khiển đọc/ghi dữ liệu thời gian hoặc RAM trong DS1307 thông qua địa chỉ I2C mặc định là **0x68** (7-bit).

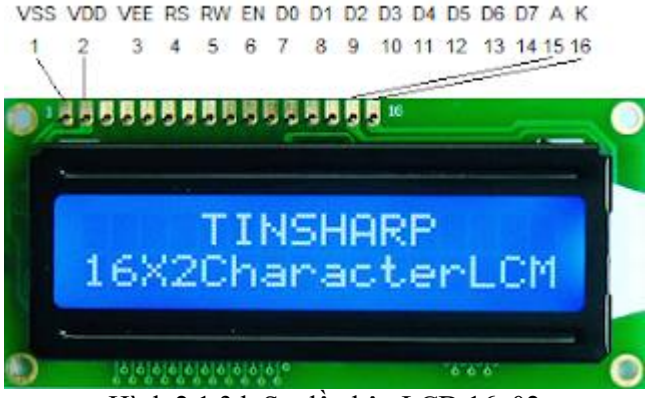
2.1.3 Màn hình LCD 16x02

Hình 2.1.3.a Màn hình LCD 16x02

Tổng quan về LCD 16x02

- **LCD 1602** (viết tắt của **Liquid Crystal Display 16x2**) là một loại màn hình hiển thị ký tự phổ biến, có thể hiển thị **16 ký tự trên 2 dòng**, thường được sử dụng trong các ứng dụng nhúng, hệ thống điều khiển tự động, thiết bị đo đạc, và dự án học tập với vi điều khiển. Màn hình này hoạt động ở chế độ **hiển thị ký tự ASCII**, tức là chỉ hiển thị văn bản mà không hỗ trợ đồ họa.
- LCD 1602 sử dụng **điều khiển bởi IC HD44780** hoặc các IC tương thích, cho phép giao tiếp với vi điều khiển thông qua **chế độ 4 bit hoặc 8 bit dữ liệu**. Với ưu điểm dễ sử dụng, giá thành rẻ, tiêu thụ năng lượng thấp và tài liệu phong phú, LCD 1602 là lựa chọn lý tưởng cho các dự án hiển thị cơ bản.

Cấu trúc chân LCD 16x02 (16 chân)



Hình 2.1.3.b Sơ đồ chân LCD 16x02

Chân	Tên	Mô tả chức năng
1	VSS	Nối đất (GND)
2	VDD	Nối nguồn 5V
3	V0	Điều chỉnh độ tương phản (qua biến trở ~10kΩ)
4	RS	Chọn thanh ghi (0: lệnh, 1: dữ liệu)
5	RW	Chọn chế độ đọc/ghi (0: ghi, 1: đọc)
6	E	Chân cho phép (Enable) – kích xung để LCD đọc dữ liệu
7-14	D0-D7	8 bit dữ liệu (nếu dùng chế độ 4 bit thì chỉ cần D4–D7)
15	LED+	Nguồn cho đèn nền (thường nối 5V qua điện trở)
16	LED-	GND của đèn nền

Bảng 2.1.3.a Mô tả chức năng LCD 16x02

Vị trí hiển thị	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Hàng 1	**00**	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
Hàng 2	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**4A**	**4B**	**4C**	**4D**	**4E**	**4F**

Bảng 2.1.3.b Mã địa chỉ ký tự hiển thị

Mô tả chức năng

- Mô-đun hiển thị LCD được tích hợp trong một bộ điều khiển LSI, bộ điều khiển này có hai thanh ghi 8-bit: một thanh ghi lệnh (IR) và một thanh ghi dữ liệu (DR).Thanh ghi IR lưu các mã lệnh, như xóa màn hình và di chuyển con trỏ, và thông tin địa chỉ cho bộ nhớ hiển thị DDRAM và bộ tạo ký tự CGRAM. IR chỉ có thể được ghi từ bộ xử lý trung tâm (MPU).DR tạm thời lưu trữ dữ liệu được ghi vào hoặc đọc từ DDRAM hoặc CGRAM. Khi thông tin địa chỉ được ghi vào IR, dữ liệu sẽ được lưu trữ trong DR từ DDRAM hoặc CGRAM.Dựa vào tín hiệu chọn thanh ghi (RS), hai thanh ghi này có thể được chọn như sau:

RS	R/W	Chức năng
0	0	Ghi vào IR như là một lệnh nội bộ (xóa hiển thị, v.v.)
0	1	Đọc cờ bận (DB7) và bộ đếm địa chỉ (DB0 đến DB7)
1	0	Ghi dữ liệu vào DDRAM hoặc CGRAM (DR đến DDRAM hoặc CGRAM)
1	1	Đọc dữ liệu từ DDRAM hoặc CGRAM (DDRAM hoặc CGRAM đến DR)

Bảng 2.1.3.c Mô tả chức năng RS và R/W

Cờ bận (BF)

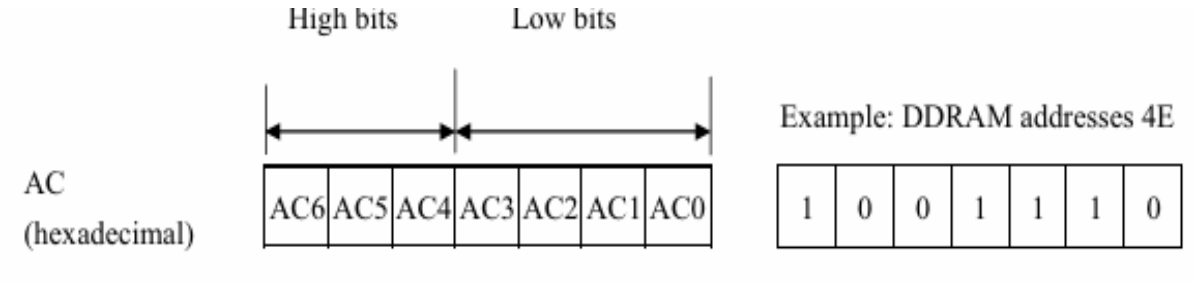
- Khi cờ bận bằng 1, bộ điều khiển LSI đang ở chế độ thực hiện lệnh nội bộ, và lệnh tiếp theo sẽ không được chấp nhận.Khi RS=0 và R/W=1, cờ bận sẽ được xuất ra chân DB7.Lệnh tiếp theo chỉ nên được ghi sau khi chắc chắn rằng cờ bận bằng 0.

Bộ đếm địa chỉ (AC)

- Bộ đếm địa chỉ (AC) gán địa chỉ cho cả DDRAM và CGRAM.

Bộ nhớ dữ liệu hiển thị

- DDRAM này dùng để lưu dữ liệu hiển thị, được biểu diễn bằng mã ký tự 8-bit. Dung lượng mở rộng của nó là 80x8 bit hoặc 80 ký tự. Hình bên dưới minh họa mối liên hệ giữa địa chỉ DDRAM và vị trí trên màn hình tinh thể lỏng:



Hình 2.1.3.b minh họa mối liên hệ giữa địa chỉ DDRAM và vị trí trên màn hình tinh thể lỏng

Upper 4 bit / Lower 4 bit	LLLL	LLLH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHLH	HHHL	HHHH
LLLL	CG RAM (1)															
LLLH	CG RAM (2)															
LLHL	CG RAM (3)															
LLHH	CG RAM (4)															
LHLL	CG RAM (5)															
LHLH	CG RAM (6)															
LHHL	CG RAM (7)															
LHHH	CG RAM (8)															
HLLL	CG RAM (1)															
HLLH	CG RAM (2)															
HLHL	CG RAM (3)															
HLHH	CG RAM (4)															
HHLL	CG RAM (5)															
HHLH	CG RAM (6)															
HHHL	CG RAM (7)															
HHHH	CG RAM (8)															

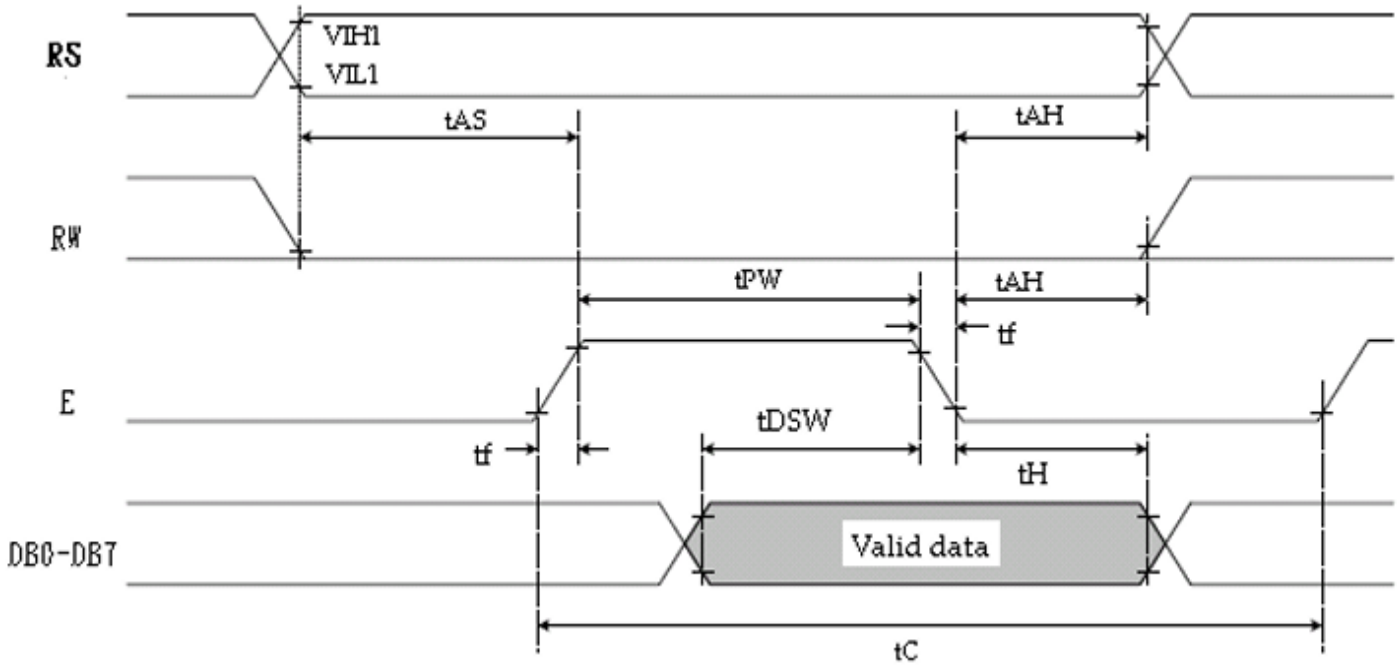
Bảng 2.1.3.d ký tự trong ROM (CGROM)

Lệnh	Mã lệnh										Mô tả	Thời gian thực thi
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Ghi “00H” vào DDRAM và đặt địa chỉ DDRAM về “00H” từ thanh ghi địa chỉ (AC)	1.53 ms
Return Home	0	0	0	0	0	0	0	0	1	-	Đặt địa chỉ DDRAM về “00H” từ AC và đưa con trỏ về vị trí gốc nếu nó đã bị dịch chuyển. Nội dung DDRAM **không thay đổi**.	1.53 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Gán hướng dịch chuyển của con trỏ và bật chức năng dịch chuyển toàn màn hình.	39 µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Thiết lập hiển thị (D), con trỏ (C) và trạng thái nhấp nháy (B) của con trỏ	39 µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Thiết lập chế độ dịch con trỏ hoặc màn hình và hướng dịch, **không thay đổi dữ liệu DDRAM**	39 µs
Function Set	0	0	0	0	1	DL	N	F	-	-	Thiết lập độ dài giao diện dữ liệu (DL: 8 bit/4 bit), số dòng hiển thị (N: 2 dòng/1 dòng), kiểu font hiển thị (F: 5×11/5×8 điểm ảnh)	39 µs
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Đặt địa chỉ CGRAM trong thanh ghi địa chỉ	39 µs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Đặt địa chỉ DDRAM trong thanh ghi địa chỉ	39 µs
Read Busy	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Kiểm tra trạng thái hoạt động nội bộ (bận hoặc	0 µs

Flag and Address											không) bằng cách đọc BF. Đồng thời có thể đọc địa chỉ trong thanh ghi địa chỉ.	
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Ghi dữ liệu vào RAM nội bộ (DDRAM hoặc CGRAM).	43 µs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Đọc dữ liệu từ RAM nội bộ (DDRAM hoặc CGRAM).	43 µs

Bảng 2.1.3.e lệnh LCD

Ghi dữ liệu (Write Operation)

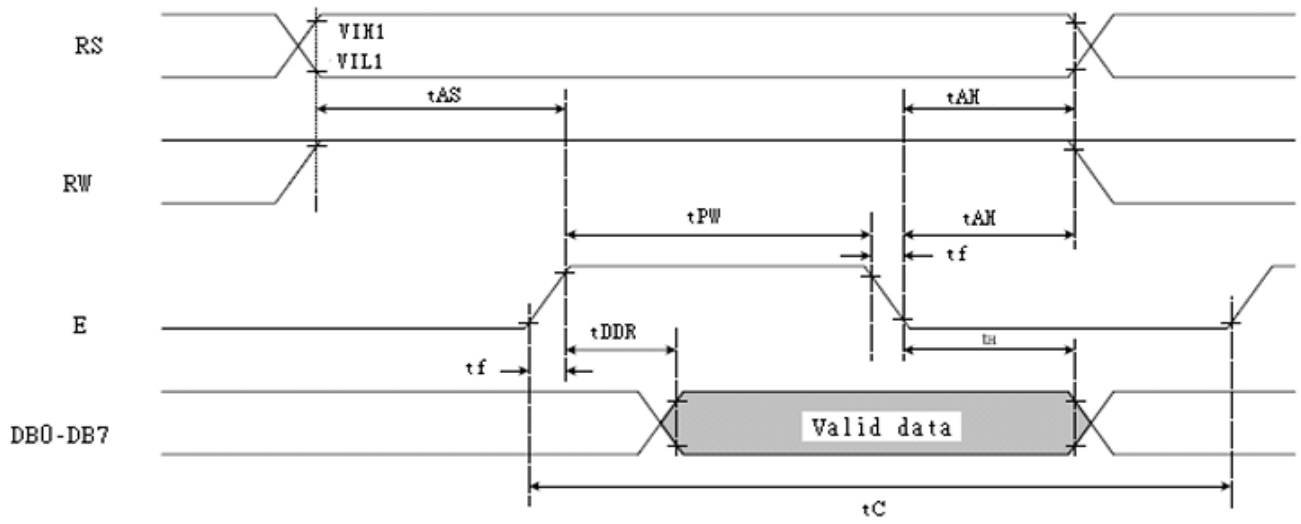


Hình 2.1.3.c mô tả ghi dữ liệu LCD

Thông số	Ký hiệu	Tối thiểu	Điển hình	Tối đa	Đơn vị
Chu kỳ Enable	tC	1200	-	-	ns
Độ rộng xung Enable	tPW	140	-	-	ns
Thời gian sườn lên/xuống của Enable	tR,tF	-	-	25	ns
Thời gian thiết lập địa chỉ	tAS	0	-	-	ns
Thời gian giữ địa chỉ	tAH	10	-	-	ns
Thời gian thiết lập dữ liệu	tDSW	40	-	-	ns
Thời gian giữ dữ liệu	tH	10	-	-	ns

Bảng 2.1.3.f thông số dữ liệu của ghi dữ liệu

Đọc dữ liệu (Read Operation)



Hình 2.1.3.d mô tả đọc dữ liệu LCD

Thông số	Ký hiệu	Tối thiểu	Điển hình	Tối đa	Đơn vị
Chu kỳ Enable	tC	1200	-	-	ns
Độ rộng xung Enable(mức cao)	tPW	140	-	-	ns
Thời gian sườn lên/xuống của Enable	tR,tF	-	-	25	ns
Thời gian thiết lập địa chỉ	tAS	0	-	-	ns
Thời gian giữ địa chỉ	tAH	10	-	-	ns
Thời gian trễ dữ liệu	tDDR	-	-	100	ns
Thời gian giữ dữ liệu	tH	10	-	-	ns

Bảng 2.1.3.g thông số dữ liệu của đọc dữ liệu

Quy trình hoạt động khi hiển thị một ký tự (truyền 1 lần 8-bit)

1. **Đặt RS = 1** để báo hiệu đang gửi dữ liệu ký tự (không phải lệnh).
2. **Đặt RW = 0** để chọn chế độ ghi vào LCD.
3. **Đặt toàn bộ 8 bit dữ liệu (mã ASCII của ký tự)** lên các chân D0–D7.
4. **Tạo xung E (Enable)**:
   a. Đưa E = 1
   b. Delay vài micro giây
   c. Đưa E = 0
5. **LCD chốt và lưu byte đó vào DDRAM** → ký tự tương ứng sẽ được hiển thị tại vị trí con trỏ.
6. **Delay ngắn (~40µs)** để chờ LCD xử lý xong.

2.1.4 Pin CR2032

- **Pin CR2032** là loại **pin lithium dạng đồng xu (coin cell)**, rất phổ biến trong các thiết bị điện tử nhỏ gọn như: **đồng hồ, máy tính, thiết bị y tế, remote, cảm biến**, và đặc biệt là **cấp nguồn dự phòng (backup) cho các IC thời gian thực như DS1307** hoặc các thiết bị nhớ như SRAM.
- Tên gọi **CR2032** mang ý nghĩa:
  - **C**: Pin lithium mangan dioxide.
  - **R**: Hình tròn (Round).
  - **20**: Đường kính 20mm.
  - **32**: Độ dày 3.2mm.

Thông số	**Giá trị**
Loại pin	Lithium (Li-MnO_2)
Điện áp danh định	3.0V
Dung lượng	~220 mAh
Kích thước	20mm đường kính, 3.2mm độ dày

Dòng xả liên tục	~0.2 mA (tối đa đến 3 mA trong thời gian ngắn)
Trọng lượng	Khoảng 3 gram
Nhiệt độ hoạt động	Từ –20°C đến +60°C
Tuổi thọ lưu trữ	5–10 năm tùy điều kiện bảo quản
Không sạc lại	Không sạc được, là pin một lần dùng

Bảng 2.1.4 Thông số kỹ thuật chính của CR2032

2.1.5 Giao thức I2C

Tổng quan về giao thức I2C

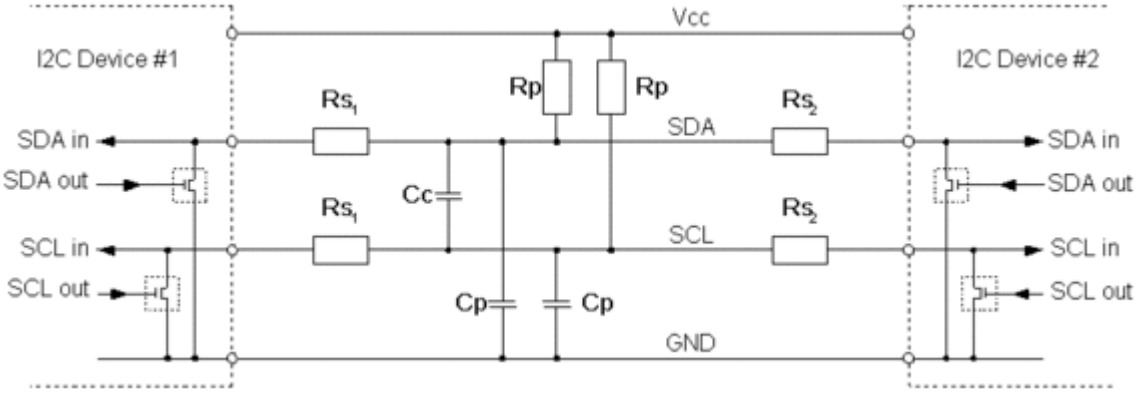
- Giao thức I2C được thế kế bởi Philips, một tập đoàn về điện tử đến từ Hà Lan vào đầu thập niên 80. Giao thức này cho phép nhiều thành phần điện tử nằm trên cùng một mạch điện giao tiếp dễ dàng với nhau. Vào năm 2006, Philips Semiconductor tách ra từ Philips Electronics và trở thành một công ty độc lập có tên NPX. Tên gọi I2C được dịch từ cụm từ "Inter IC" (liên kết mạch tích hợp). Trong một số tài liệu giao thức này còn được viết là IIC hay I^2C.

  -Tốc độ truyền tải dữ liệu tối đa ban đầu của giao thức I2C được xác định là 100Kbps và với đa số ứng dụng thì tốc độ này là vừa đủ và không yêu cầu cao hơn. Với những ứng dụng yêu cầu tốc độ truyền tải cao hơn, giao thức I2C hỗ trợ chế độ 400Kbps fastmode. Từ năm 1998, giao thức I2C hỗ trợ thêm chế độ high speed (tốc độ cao) với tốc độ truyền tải tối đa có thể đạt được 3.4Mbps. Phiên bản gần nhất, I2C hỗ trợ thêm chế độ fast mode +, cho phép tốc độ truyền tải tối đa đạt 1Mbps. Hơn thế nữa, một số ứng dụng yêu cầu tốc độ truyền tải cao hơn, để đáp ứng điều này I2C hỗ trợ thêm chế độ Ultra-Fast Mode, nhưng chế độ này loại bỏ một số tiêu chuẩn của I2C và chỉ cho phép ghi nên không được sử dụng rộng rãi.

- Giao thức I2C không chỉ giúp các linh kiện giao tiếp với nhau trên cùng một mạch một cách dễ dàng mà còn có khả năng kết nối các linh kiện với nhau thông qua dây dẫn. Đơn giản và linh hoạt là các đặc trưng nổi bật của giao thức I2C, khiến nó trở thành giao thức truyền thông lý tưởng trong nhiều ứng dụng. Một số tính năng quan trọng của giao thức I2C bao gồm:
  1. Chỉ yêu cầu hai dây kết nối
  2. Không yêu cầu nghiêm ngặt về tốc độ truyền, master tạo xung clock để đồng bộ.
  3. Quan hệ master/slave đơn giản giữa tất cả các linh kiện. Mỗi thiết bị kết nối vào đường tín hiệu I2C được định địa chỉ bằng phần mềm bởi một địa chỉ độc nhất.
  4. Đường tín hiệu I2C có thể gồm nhiều master và nó cung cấp khả năng phát hiện va chạm khi có nhiều thiết bị cùng lúc gửi dữ liệu.

Thiết lập I2C

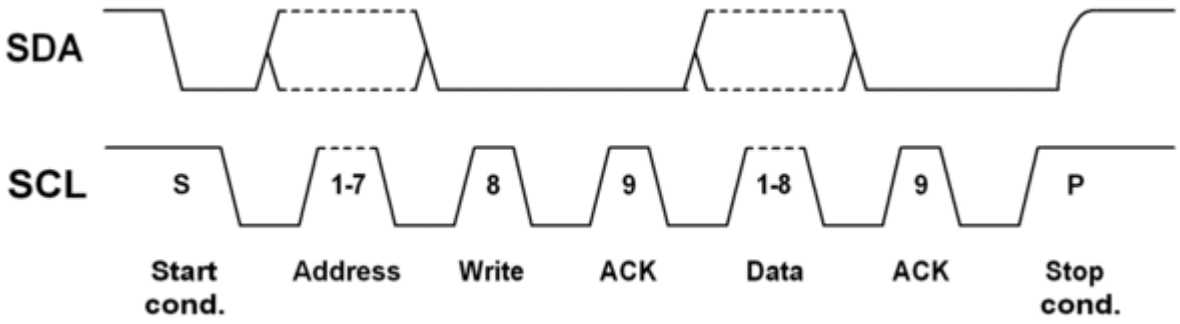
- Giao thức I2C yêu cầu hai cổng: SDA có vai trò truyền tải dữ liệu nối tiếp, SCL có vai trò đồng bộ xung clock giữa các thiết bị. Cả hai cổng được cấu hình ở trạng thái open-drain (thoát hở) vì thế các thiết bị chỉ có thể kéo tín hiệu ở hai cổng này xuống mức thấp hoặc để ở trạng thái thoát hở. Ta cần kết nối hai cổng SDA và SCL với điện trở kéo lên (pull-up resistor) để kéo tín hiệu lên mức Vcc khi không có thiết bị nào kéo tín hiệu xuống mức thấp



Hình 2.1.5.a Thiết lập I2C

Giao tiếp I2C

- Trong giao thức I2C, một phiên truyền thông tuân theo các bước như sau: (1) điều kiện bắt đầu (Start condition), (2) byte địa chỉ, (3) các byte dữ liệu, (4) điều kiện kết thúc (Stop condition). Ở giữa các byte đều sẽ có các ACK báo hiệu nhận dữ liệu thành công hoặc NoACK báo hiệu kết thúc nhận dữ liệu được gửi bởi master hoặc slave.



Hình 2.1.5.b Giao tiếp I2C

- Mỗi linh kiện kết nối với đường tín hiệu I2C đều sẽ có một địa chỉ duy nhất. Địa chỉ này có độ dài 7 bit, bit thứ 8 dùng làm bit điều khiển đọc ghi ($\underline{R}/W$). Khi bit điều khiển có giá trị 0 có nghĩa master đang yêu cầu dữ liệu từ slave, ngược lại, khi bit điều khiển có giá trị 1 có nghĩa master đang ghi dữ liệu lên slave.
-

7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	$\underline{R}/W$

Bảng 2.1.5.a Địa chỉ I2C của IC DS1307

Bit banging

- Bit Banging (đập bit) là thuật ngữ để chỉ một phương thức truyền thông dữ liệu số sử dụng GPIO thay vì phần cứng chuyên dụng. Phần mềm điều khiển (Controlling software) có trách nhiệm mô phỏng lại giao thức cụ thể để đáp ứng các yêu cầu của giao thức đó bao gồm cả việc định thời, công việc có thể gây tốn nhiều tài nguyên phần cứng và chiếm dụng nhiều thời gian CPU để thực hiện.
- Ngược lại, phần cứng chuyên dụng (ví dụ I2C, UART, SPI) có thể đáp ứng các yêu cầu của một giao thức cụ thể một cách dễ dàng, giảm thiểu thời gian chiếm dụng CPU và tối thiểu sử dụng tài nguyên phần cứng cần thiết.
- Phương thức Bit Banging cho phép một máy tính hỗ trợ các phương thức truyền thông với phần cứng giới hạn hoặc không can thiệp sâu vào phần cứng, do đó big banging có thể giảm thiểu chi phí vì trong đa phần các trường hợp, thay đổi phần mềm sẽ rẻ hơn thay đổi phần cứng.

Big banging	
Ưu điểm	Nhược điểm
Không cần phần cứng chuyên dụng	Tốn CPU (vì phải chờ và điều khiển từng bit)
Linh hoạt – có thể mô phỏng bất kỳ giao thức nào	Dễ sai nếu định thời không chính xác
Dễ dùng với vi điều khiển giá rẻ hoặc không đủ chân chuyên dụng	Không thích hợp cho ứng dụng tốc độ cao

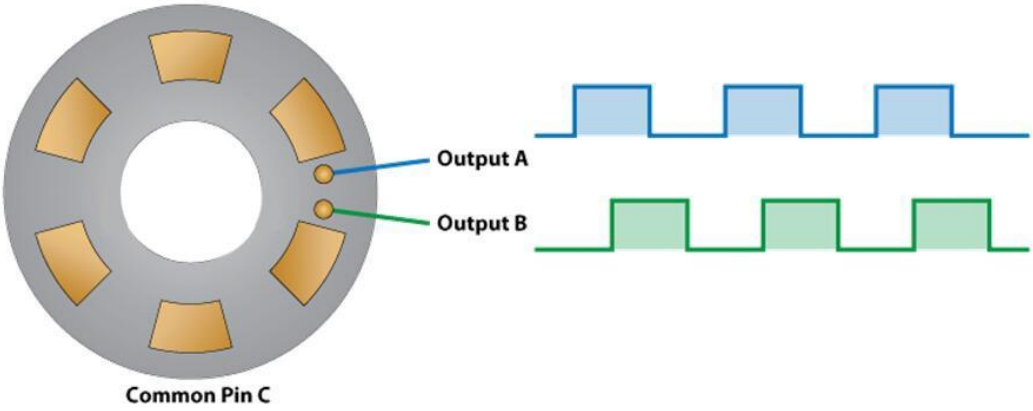
Bảng 2.1.5.b Ưu điểm và nhược điểm của Big banging

- Vi điều khiển AT89S51 không hỗ trợ sẵn giao thức I2C, vì thế cần áp dụng phương thức big banging để mô phỏng lại giao thức I2C. Trong đồ án này, để mô phỏng lại giao thức I2C trên VĐK AT89S51, nhóm sử dụng hai chân GPIO P3.6 và P3.7 làm cổng SDA và SCL tương ứng. Tốc độ đạt được là 100Kbps. Để có thể định thời chính xác, nhóm sử dụng lệnh NOP để xây dựng các hàm delay với khoảng thời gian delay chính xác lên đến 0.5μs (1 chu kỳ máy).

2.1.6 Rotary Encoder

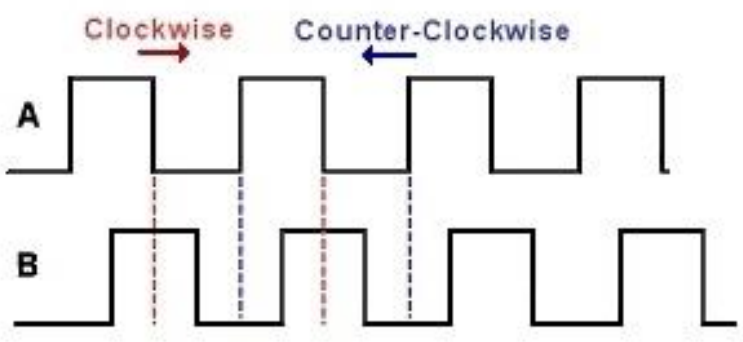
- Rotary encoder (cụ thể là **Incremental encoder** được sử dụng trong đồ án lần này) là một cảm biến được dùng để **đo vị trí góc**, **hướng quay** và **số vòng quay** của một trục hoặc núm xoay. Nó được sử dụng rất phổ biến trong các ứng dụng điều khiển

(như điều chỉnh âm lượng, vị trí motor, menu trên thiết bị điện tử…).



Hình 2.1.6.a Tín hiệu đầu ra A và B khi xoay rotary encoder

- Incremental encoder sử dụng trong đồ án này là mạch KY-040. Đầu ra tín hiệu gồm hai cổng CLK (A) và DT (B) , ngoài ra còn có thêm 1 cổng SW hoạt động như tín hiệu nút nhấn. Để xử lý được tín hiệu của rotary encoder ta cần hiệu được cách vận hành cũng như xung tín hiệu mà nó tạo khi xoay.



Hình 2.1.6.b Xung tín hiệu rotary encoder khi xoay theo hai chiều khác nhau

- Khi xoay rotary encoder theo một hướng nhất định, đĩa tiếp điểm (thành phần chứa các mặt dẫn điện) bên trong sẽ xoay theo cùng hướng, một trong hai tiếp điểm cố định sẽ chạm mặt dẫn điện trước tạo thành xung như hình. Nhờ vào đọc tín hiệu hai chân A và B ta có thể xác định điểm nào chạm mặt dẫn điện trước từ đó xác định được hướng quay của rotary encoder.

2.2 PHẦN MỀM KEIL C

- Keil C, hay còn gọi là Keil µVision IDE, là một môi trường phát triển tích hợp (IDE) mạnh mẽ do hãng Arm Keil phát triển, chuyên dùng để lập trình cho các dòng vi điều khiển như 8051, ARM, và Cortex-M. Đối với vi điều khiển AT89S51 – một đại diện điển hình trong họ MCS-51, Keil C đóng vai trò là công cụ lập trình phổ biến nhất hiện nay. Keil hỗ trợ cả ngôn ngữ lập trình C và hợp ngữ (Assembly) thông qua bộ biên dịch C51 và bộ hợp dịch A51. Giao diện Keil µVision cung cấp

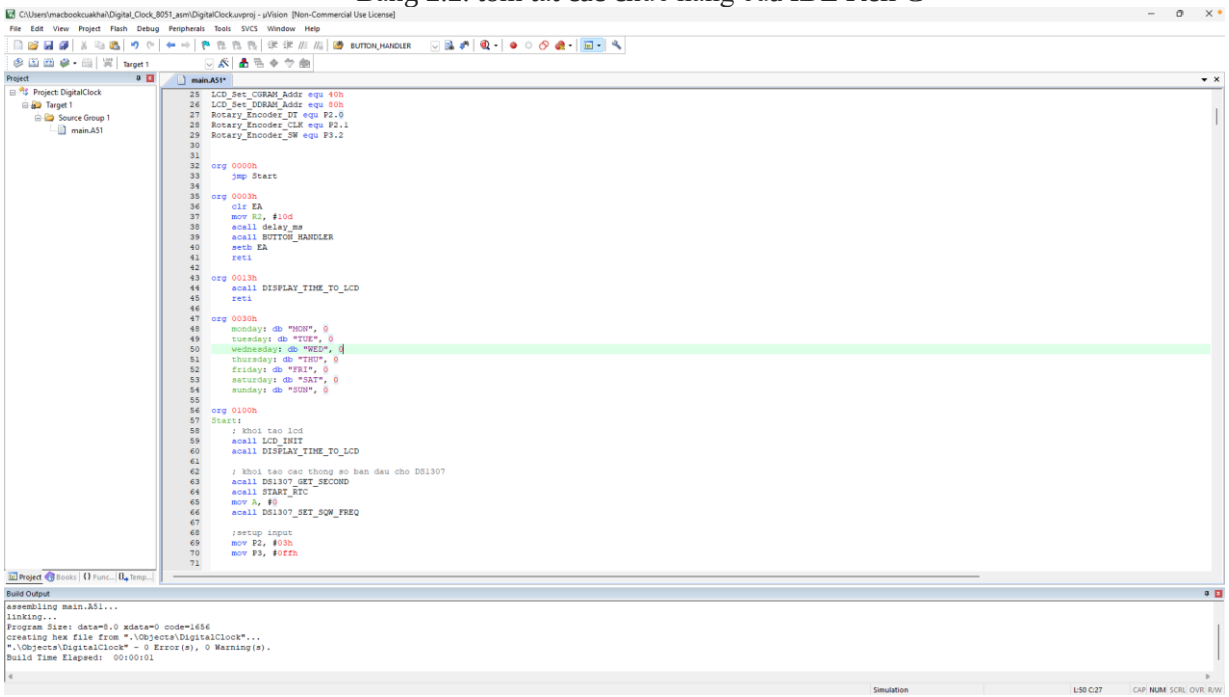
đầy đủ các công cụ cần thiết từ việc soạn thảo mã nguồn, biên dịch, mô phỏng cho đến tạo file nạp (.hex) một cách thuận tiện và dễ thao tác.

- Các thành phần chính của phần mềm bao gồm: µVision IDE dùng để quản lý và biên tập mã nguồn, trình biên dịch C51 để chuyển mã C sang mã máy, trình hợp dịch A51 dùng cho mã Assembly, linker để liên kết các tập tin chương trình và tạo ra file .hex, và cuối cùng là công cụ mô phỏng/debug giúp kiểm tra chương trình mà không cần vi điều khiển thật. Quy trình làm việc với Keil bắt đầu bằng việc tạo một project mới, chọn vi điều khiển phù hợp (AT89S51 hoặc AT89C51 – do chúng tương thích nhau), thêm tệp mã nguồn vào project, biên dịch chương trình, và cuối cùng là tạo file .hex để nạp vào chip bằng mạch nạp chuyên dụng như USBASP thông qua phần mềm ngoài như ProgISP hoặc SinaProg.
- Một điểm mạnh đáng kể của Keil là khả năng mô phỏng hoạt động của vi điều khiển, cho phép người lập trình kiểm tra logic chương trình trước khi triển khai thực tế. Ngoài ra, Keil cũng hỗ trợ tốt việc quản lý bộ nhớ, debug lỗi, và cung cấp thông tin chi tiết về ánh xạ bộ nhớ thông qua các tập tin .map, .lst, giúp việc phát triển ứng dụng nhúng trở nên hiệu quả và chính xác hơn. Một ví dụ đơn giản minh họa cho lập trình trong Keil là chương trình xuất dữ liệu dạng nhấp nháy lên cổng P1: P1 = 0x55;. Có thể nói, Keil C là một công cụ không thể thiếu đối với những ai đang học tập, nghiên cứu hoặc phát triển ứng dụng với vi điều khiển AT89S51 nói riêng và dòng 8051 nói chung.

Chức năng	Mô tả
Soạn thảo mã nguồn (Code Editor)	Cung cấp môi trường viết code với hỗ trợ tô màu cú pháp, gợi ý mã.
Tạo Project	Cho phép tạo và quản lý nhiều dự án với cấu trúc file rõ ràng.
Chọn vi điều khiển	Cho phép lựa chọn dòng vi điều khiển cụ thể (ví dụ: AT89S51, AT89C51).
Biên dịch chương trình (Compiler)	Dịch mã C sang mã máy nhờ trình biên dịch C51.
Hợp dịch (Assembler)	Hợp dịch mã Assembly sang mã máy với A51 Assembler.
Liên kết mã (Linker)	Liên kết các tệp mã thành tệp thực thi .hex để nạp vào vi điều khiển.
Mô phỏng chương trình (Debugger)	Cho phép chạy mô phỏng chương trình để kiểm tra và debug lỗi.
Gỡ lỗi (Debug)	Bước qua từng dòng mã, quan sát thanh ghi, bộ nhớ, cổng vào/ra…
Quản lý bộ nhớ	Cho biết cách sắp xếp và sử dụng bộ nhớ trong chương trình.

Hiển thị tập tin ánh xạ (.map)	Cung cấp thông tin ánh xạ các biến và hàm trong bộ nhớ chương trình.
Xuất file HEX	Tạo tệp .hex dùng để nạp vào vi điều khiển qua mạch nạp.
Tích hợp mô phỏng ngoại vi	Cho phép mô phỏng I/O như LED, UART, Timer, ngắt…
Hỗ trợ debug nâng cao	Xem thanh ghi CPU, giá trị biến, trạng thái cổng… theo thời gian thực.

Bảng 2.2. tóm tắt các chức năng của IDE Keil C



Hình 2.2 Giao diện phần mềm Keil C

2.3 PHẦN MỀM PROTEUS

- Proteus là phần mềm **EDA (Electronic Design Automation)**, do **Labcenter Electronics (Anh)** phát triển. Nó cho phép:
  - Vẽ **sơ đồ nguyên lý (schematic)** của mạch điện tử.
  - Thiết kế **bo mạch in PCB**.
  - **Mô phỏng hoạt động mạch điện** (đặc biệt mạnh với vi điều khiển).
  - **Kiểm thử chương trình điều khiển** ngay trên máy tính mà không cần phần cứng thực.

Các thành phần chính của proteus

ISIS – Vẽ mạch và mô phỏng

- Cho phép kéo linh kiện từ thư viện và nối dây tạo thành mạch.
- Hỗ trợ mô phỏng vi điều khiển thời gian thực (nạp file `.hex` hoặc `.elf`).
- Có thể chạy mô phỏng như thực tế: đèn LED nhấp nháy, LCD hiển thị, tín hiệu cảm biến...

ARES – Thiết kế mạch in (PCB)

- Biến sơ đồ nguyên lý (ISIS) thành bố trí mạch in 2D.
- Có thể tạo file Gerber để đặt hàng in mạch ngoài thực tế.

VSM (Virtual System Modelling)

- Cho phép mô phỏng cả phần mềm và phần cứng song song.
- Có thể gắn logic analyzer, oscilloscope, signal generator, v.v.

Giao diện người dùng Proteus (ISIS)

Thành phần	**Vai trò**
Toolbar trái	Chọn linh kiện, dây nối, text, chân đất (GND), nguồn (VCC)...
Component Mode	Tìm và chèn linh kiện từ thư viện (`P` để mở thư viện).
Mạch làm việc (workspace)	Kéo thả linh kiện và kết nối các chân để tạo mạch.
Simulation Control (Play,Stop,Reset..)	Dùng để mô phỏng – giống như chạy thử mạch ngoài đời.

Property Panel (phải)	Cài đặt thông số linh kiện: điện trở, tần số thạch anh, tên file `.hex`,...

Bảng 2.3.a bảng giao diễn người dùng proteus

Tính năng mạnh của Proteus

Tính năng	Mô tả cụ thể
Mô phỏng thời gian thực	Quan sát kết quả như thật khi chạy chương trình điều khiển LED, LCD, relay, DS1307...
Hỗ trợ nhiều vi điều khiển	8051, AVR, Arduino, PIC, STM32, ESP32…
Công cụ đo lường ảo	Oscilloscope, logic analyzer, voltmeter, ammeter... giúp kiểm tra tín hiệu trong mạch.
Thiết kế PCB hoàn chỉnh	Hỗ trợ tạo bo mạch in 1 lớp, 2 lớp hoặc nhiều lớp chuyên nghiệp
Thư viện linh kiện phong phú	Có sẵn IC, LCD, cảm biến, relay, nguồn, LED, nút nhấn,...
Tùy chỉnh linh kiện	Gán mã hex, chọn loại IC, cài đặt thông số thời gian, nguồn, điện áp,...

Bảng 2.3.b tính năng proteus

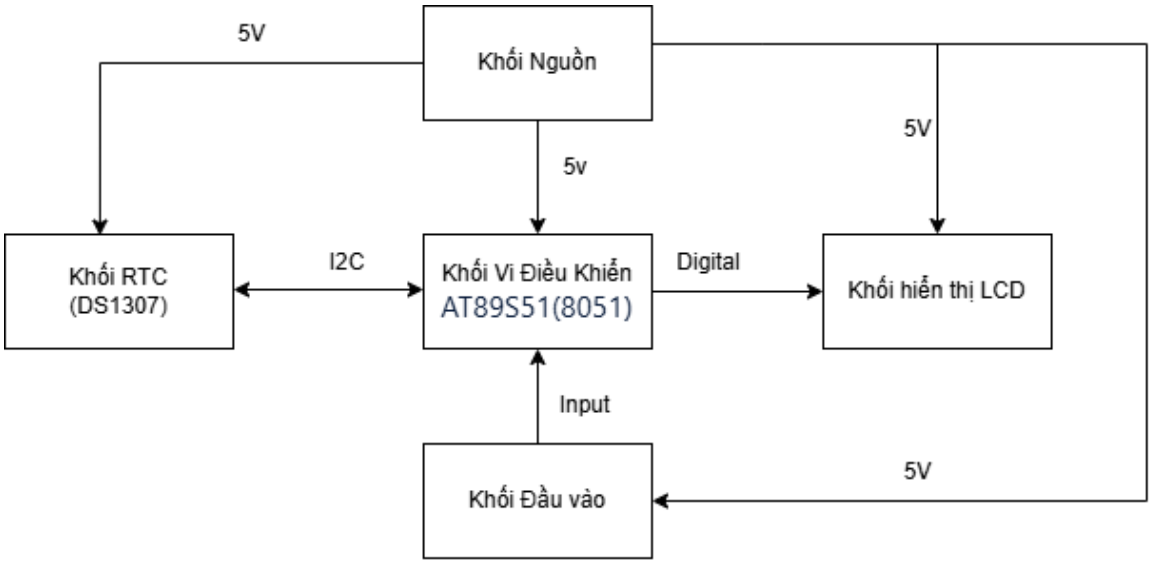
Ứng dụng của Proteus

Mục đích	Ứng dụng thực tế
Học lập trình VĐK	Kiểm tra chương trình trên 8051, AVR, Arduino... mà không cần mạch thật.
Thử nghiệm phần cứng	Xây dựng và kiểm tra mạch logic, điều khiển động cơ, cảm biến...
Kiểm tra tín hiệu	Dùng oscilloscope/logic analyzer để bắt tín hiệu và debug
Thiết kế mạch in	Làm bo mạch và tạo file Gerber cho sản xuất

Bảng 2.3.c ứng dụng proteus

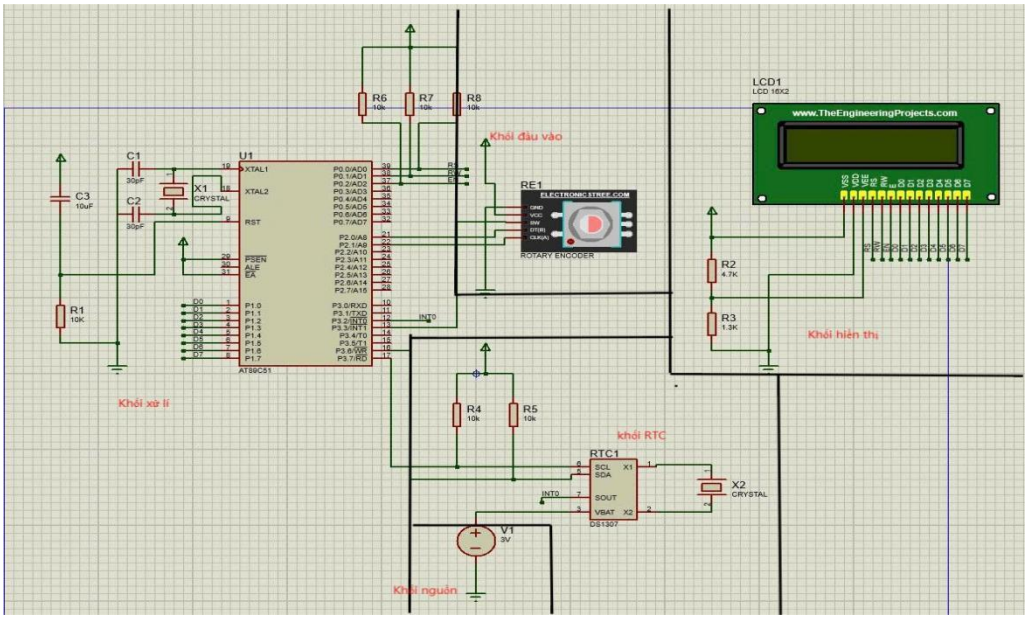
CHƯƠNG 3: SƠ ĐỒ KHỐI VÀ THIẾT KẾ MẠCH

3.1 SƠ ĐỒ KHỐI



Hình 3.1 sơ đồ khối

3.1.1 Nguyên lý hoạt động của mạch



Hình 3.1.1 sơ đồ nối dây của mạch

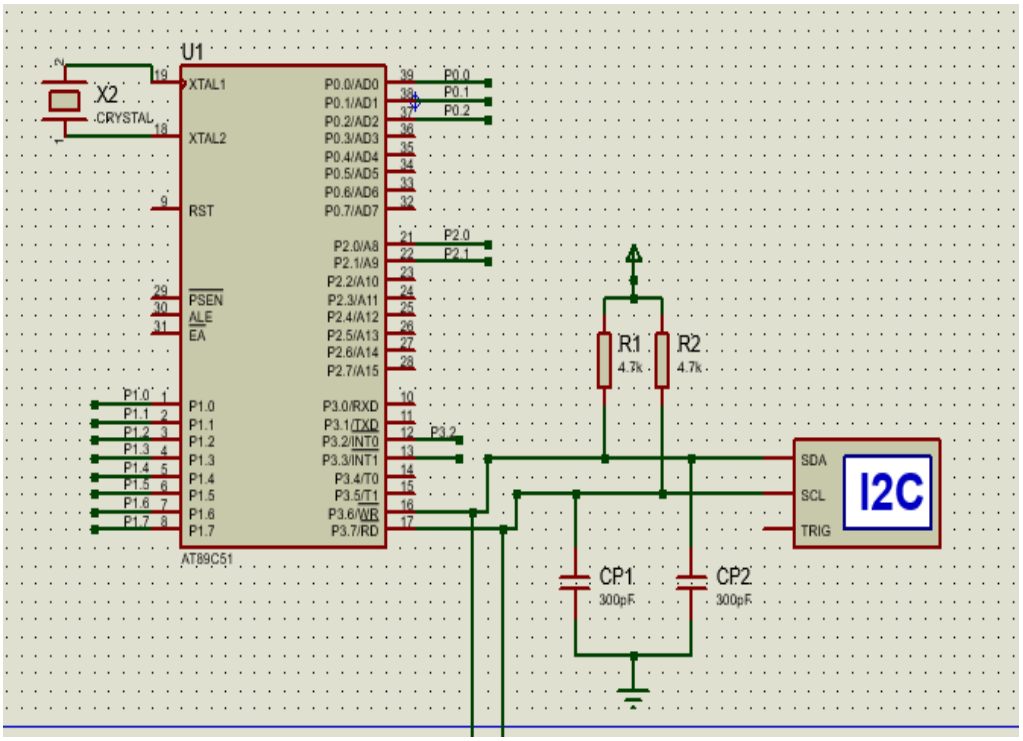
- **Khối nguồn** cung cấp điện cho toàn bộ hệ thống.
- **Khối vi điều khiển (AT89S51)** thực hiện giao tiếp I2C để gửi lệnh và yêu cầu đến **khối RTC (DS1307)**.
  Khối RTC nhận các yêu cầu từ vi điều khiển và thực hiện theo yêu cầu đó.
- **Khối đầu vào** gửi tín hiệu điều chỉnh dữ liệu thời gian đến **vi điều khiển**, từ đó dữ liệu sẽ được cập nhật vào **khối RTC**.
- **Khối hiển thị** nhận dữ liệu thời gian từ **vi điều khiển** và hiển thị thông tin lên màn hình

3.2 SƠ ĐỒ NGUYÊN LÝ

3.2.1 Khối nguồn

- Khối nguồn sử dụng nguồn cắm điện trực tiếp 5v để cấp cho vi điều khiển 8051, bộ DS1307 và các linh kiện như LCD16x2.

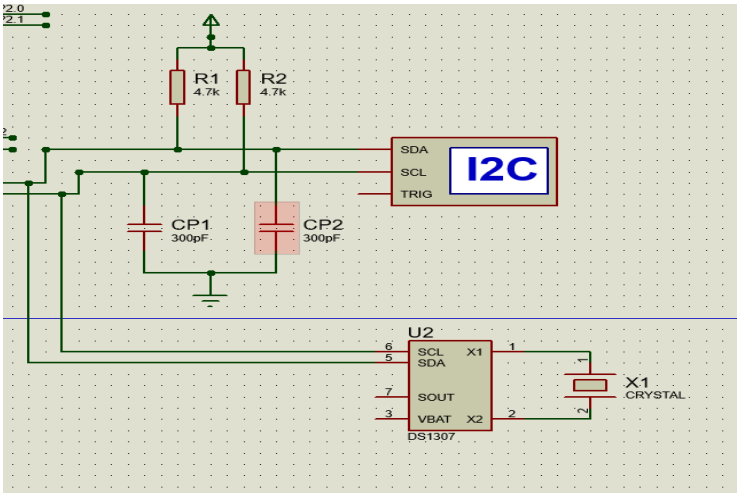
3.2.2 Khối vi điều khiển (xử lý)



Hình 3.2.2 Khối vi điều khiển được mô phỏng trên proteus

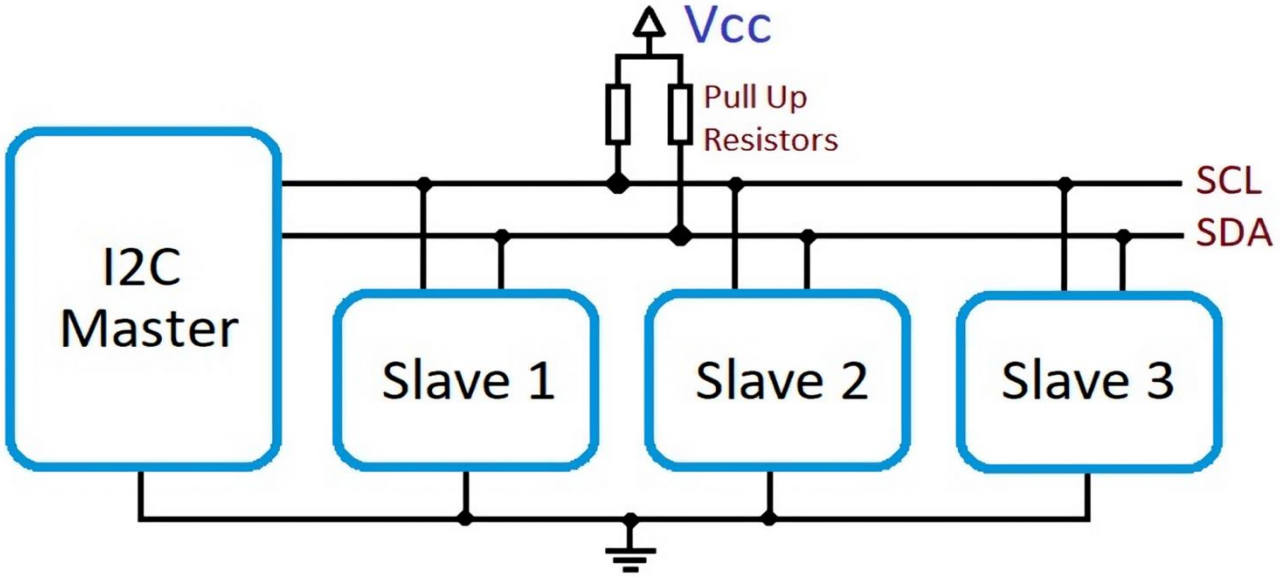
- Khối xử lý có chức năng lấy dữ liệu và ghi dữ liệu vào DS1307 bằng giao thức I2C. Tiến hành xử lý dữ liệu và hiển thị ra màn hình LCD 16x2 theo định dạng và đầu ra mong muốn.

3.2.3 Khối RTC (DS1307)



Hình 3.2.3.a Khối RTC được mô phỏng trên proteus

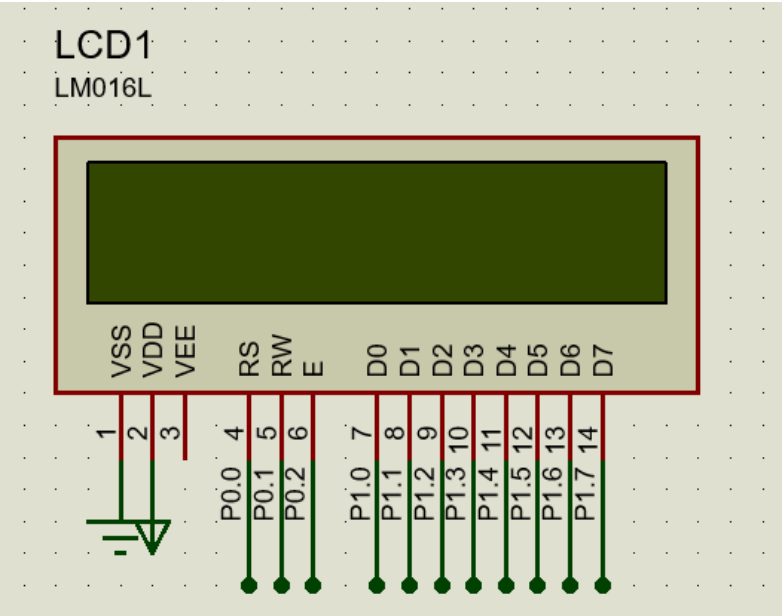
- Có chức năng là lưu trữ và cung cấp thời gian thực. Kể cả khi mất điện thì vẫn có nguồn dự phòng để giữ thời gian liên tục.
- 8051 sẽ đọc dữ liệu giờ, phút, giây, ngày, tháng, năm, thứ từ RTC bằng giao thức I2C.



Hình 3.2.3.b Giao thức I2C

- Khi dùng khối RTC thì dữ liệu chính xác hơn và không phải dùng bộ định thời để tính giờ vì khối RTC là một khối hoạt động độc lập. Lúc này sẽ giảm tải những thứ mà 8051 phải xử lý, tập trung vào xử lý hiển thị dữ liệu ra màn hình, các thao tác điều chỉnh giờ.

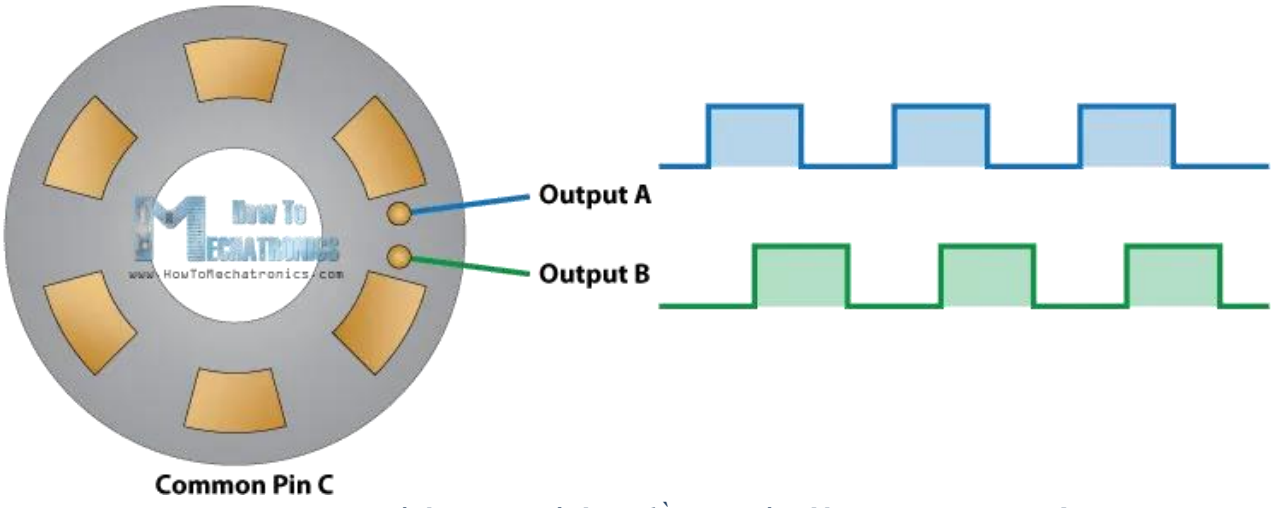
3.2.4 Khối hiển thị



Hình 3.2.4 Khối Hiển thị được mô phỏng trên proteus

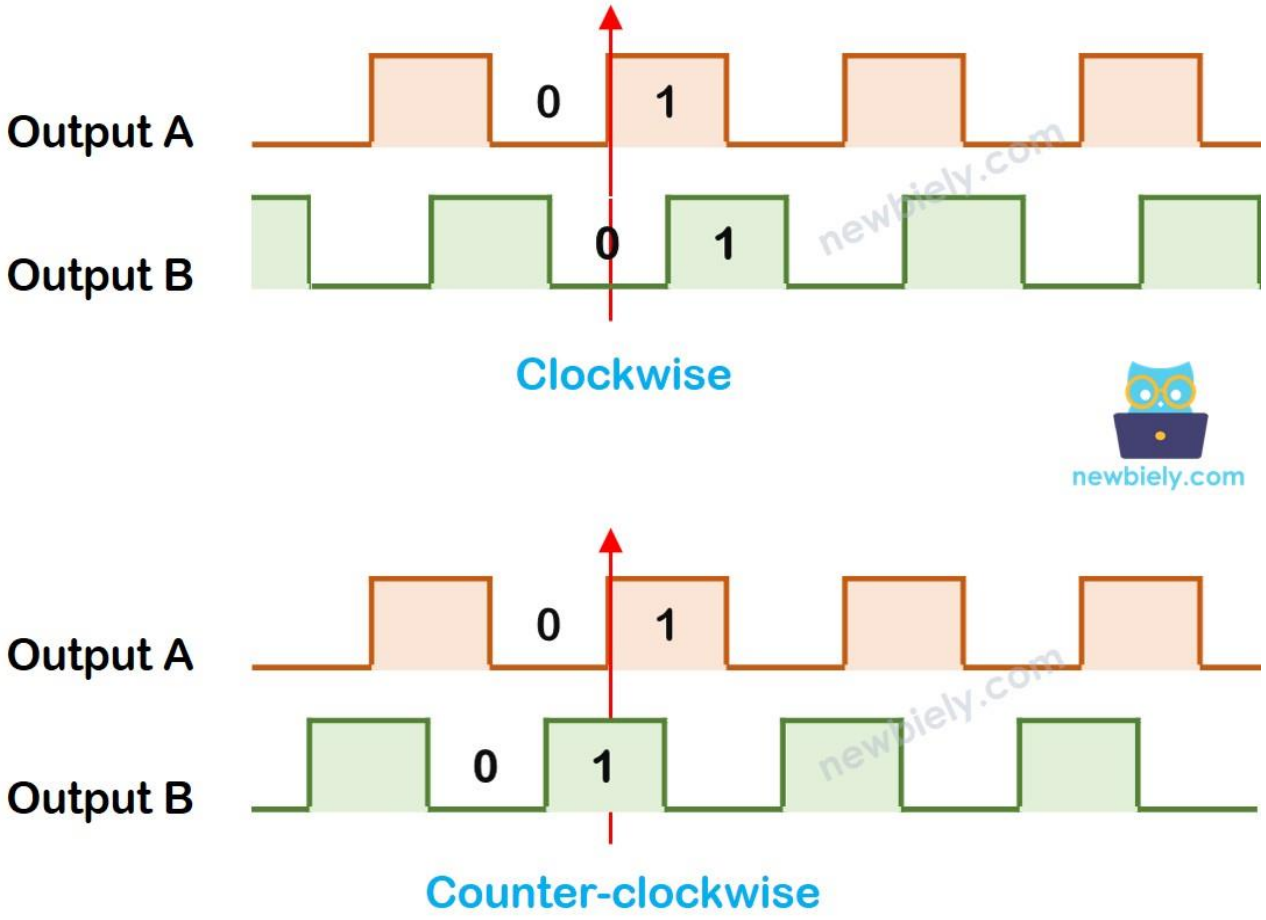
- Khối hiển thị sẽ dùng dữ liệu đã qua xử lý (vi điều khiển) để hiển thị dữ liệu ra màn hình.

3.2.5 Khối đầu vào



Hình 3.2.5.a Tín hiệu đầu ra A và B khi xoay rotary encoder

- Khối đầu vào gồm nút nhấn và rotary encoder khối này có tác dụng dùng điều chỉnh giờ cho đúng múi giờ mà người sử dụng mong muốn.



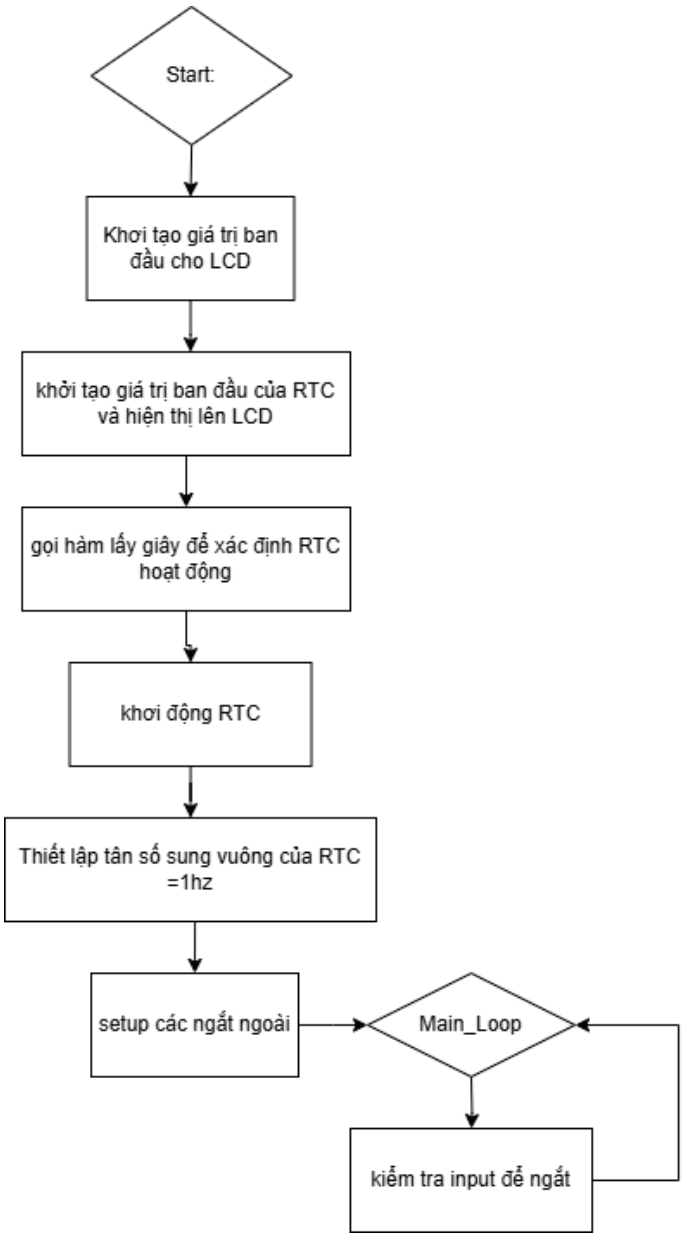
Hình 3.2.5.b tín hiệu điều khiển từ rotary encoder

- Rotary encoder sẽ có có hai đầu output A và B để tạo ra tín hiệu, hai tín hiệu này sẽ lệch pha nhau 90 độ. Nếu quay thuận chiều kim đồng hồ thì tín hiệu A sẽ có trước B lúc này tín hiệu lấy ra là (0 1) và ngược chiều kim đồng hồ thì tín hiệu B sẽ có trước A lúc này tín hiệu sẽ là (1 0). Vì vậy lúc này chúng ta có thể xem xét tín hiệu để nhận biết và lập trình hướng xoay và đọc bộ mã hóa.
- Trên module đã có tích hợp một nút nhấn vì vậy cách hoạt động của khối này là nhấn nút nhấn để bật chế độ, vặn nút xoay để điều chỉnh giờ.

CHƯƠNG 4: CHƯƠNG TRÌNH ĐIỀU KHIỂN

4.1 LƯU ĐỒ GIẢI THUẬT

4.1.1 Lưu đồ giải thuật chương trình chính (main)



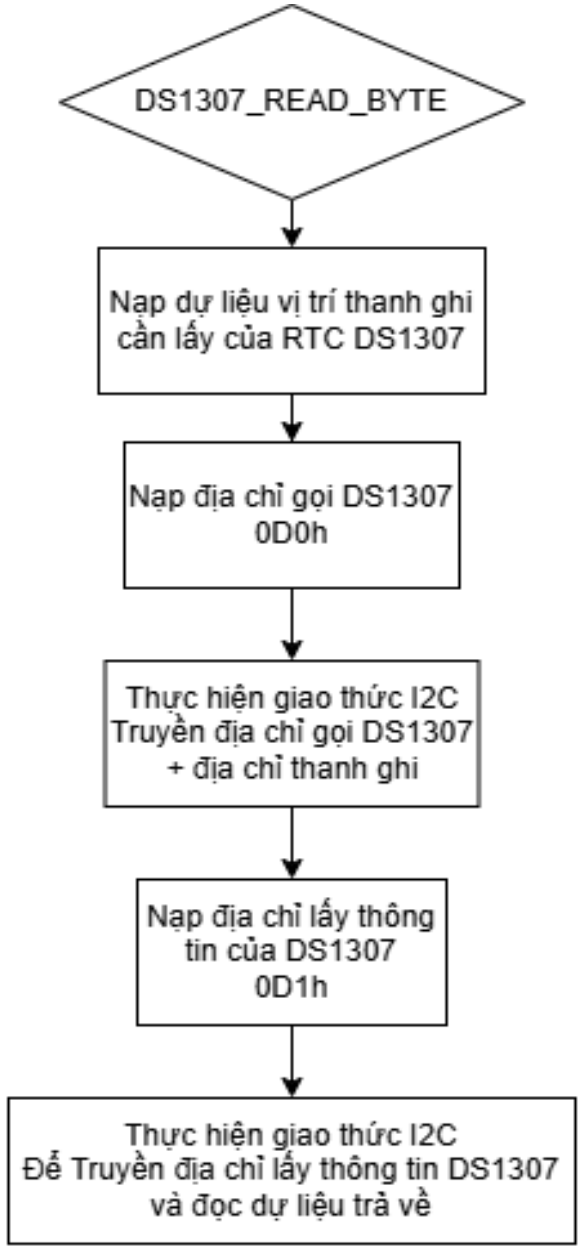
Hình 4.1.1: Lưu đồ giải thuật chương trình chính (main)

- Chương trình này có nhiệm vụ khởi tạo các thiết bị, cấu hình các ngắt ngoài và thiết lập vòng lặp chính (Main_Loop) để liên tục kiểm tra và xử lý khi có ngắt xảy ra.

```
Start:
        acall LCD_INIT
        acall DISPLAY_TIME_TO_LCD
        acall DS1307_GET_SECOND
        acall START_RTC
        mov A, #0
        acall DS1307_SET_SQW_FREQ
        mov P2, #03h
        mov P3, #0ffh
        SETB IT0    ;
        SETB EX0    ;
        SETB EA     ;
        setb IT1    ;
        setb ex1    ;
        mov 38h, #0ffh

MainLoop:
        acall CHECK_ENCODER
        acall delay_1ms
        jmp MainLoop
```

4.1.2 Lưu đồ giải thuật đọc dữ liệu từ IC DS1307

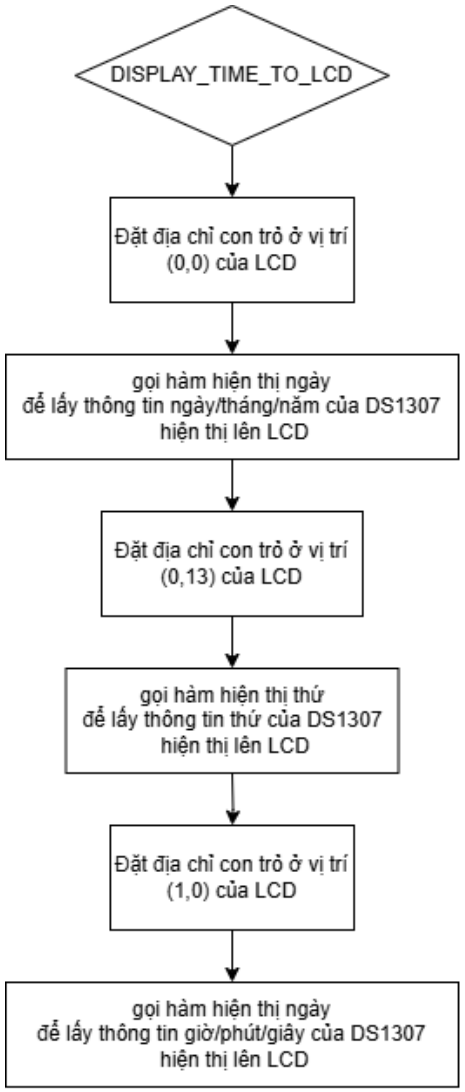


Hình 4.1.2 Lưu đồ giải thuật đọc dữ liệu từ IC DS1307

Chương trình này được thiết kế để yêu cầu RTC (DS1307) gửi dữ liệu từ thanh ghi được chỉ định và nhận dữ liệu đó thông qua giao tiếp I2C
dùng push để đẩy dữ liệu vào stack và pop để đẩy dữ liệu lên lại

```
DS1307_READ_BYTE:
    PUSH Acc
    mov A, #DS1307_Address
    acall I2C_Start
    acall I2C_Send_Byte
    acall I2C_Read_ACK
    pop Acc
    acall I2C_Send_Byte
    acall I2C_Read_ACK
    acall I2C_Stop
    mov A, #DS1307_Address
    orl A, #1
    acall I2C_Start
    acall I2C_Send_Byte
    acall I2C_Read_ACK
    acall I2C_Read_Byte
    acall I2C_Send_NoAck
    acall I2C_Stop
   RET
```

4.1.3 Lưu đồ giải thuật hiển thị thông tin lên LCD



Hình 4.1.3 Lưu đồ giải thuật hiển thị thông tin lên LCD

- Chương trình sử dụng con trỏ điều khiển vị trí hiển thị trên LCD và gọi các hàm Display_... để truy xuất dữ liệu thời gian từ DS1307. Mỗi hàm đảm nhận việc hiển thị một thành phần cụ thể ví dụ như DISPLAY_HOURS_TO_LCD để hiển thị giờ.

```
DISPLAY_TIME_TO_LCD:
        MOV R0, #0
        MOV R1, #0
        ACALL LCD_SETCURSOR
        ACALL DISPLAY_DATE_TO_LCD
        MOV R0, #0
        MOV R1, #13
        ACALL LCD_SETCURSOR
        ACALL DISPLAY_DAY_TO_LCD

        MOV R0, #1
        MOV R1, #0
        ACALL LCD_SETCURSOR
        ACALL DISPLAY_HOURS_TO_LCD
        ACALL DISPLAY_MINUTES_TO_LCD
        ACALL DISPLAY_SECONDS_TO_LCD
        RET
```

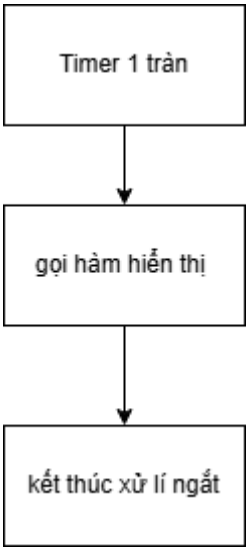
4.2 LƯU ĐỒ GIẢI THUẬT CỦA CHƯƠNG TRÌNH CON

4.2.1 Lưu đồ giải thuật xử lý ngắt

1. Xử lý ngắt ngoài.

Hình 4.2.1.a Xử lý ngắt ngoài

2. Xử lý ngắt timer 1.



Hình 4.2.1.b xử lý ngắt timer 1

- Chương trình sử dụng ngắt ngoài 0 gọi hàm delay_1ms để không bị dội phím tín hiệu sau đó gọi hàm BUTTON_HANDLER để xử lí điều chỉnh giờ

```
org 0003h
        acall delay_1ms
        acall BUTTON_HANDLER
        reti

org 0013h
        acall DISPLAY_TIME_TO_LCD
        reti
```

4.2.3 Chương trình minh họa

link github: babayaga87/Digital_Clock_8051_asm

Hình 4.2.3 QR code dẫn đến github

CHƯƠNG 5: KIỂM THỬ

5.1 PHÂN TÍCH TÍN HIỆU GIAO TIẾP I2C BẰNG LOGIC ANALYZER

Để có thể ghi hay đọc dữ liệu từ IC DS1307, cần sử dụng giao thức I2C. Vì vi điều khiển AT89S51 không có sẵn phần cứng chuyên dụng cho giao thức I2C, nên cần triển khai một phần mềm điều khiển (controlling software) để mô phỏng lại giao thức này. Phần mềm điều khiển mô phỏng giao thức I2C được chia làm 7 hàm nhỏ với chức năng được mô tả trong bảng sau.

Hàm	Mô tả chức năng
I2C_Start	Tạo tín hiệu bắt đầu giao tiếp (Start condition)
I2C_Send_Byte	Gửi một byte từ master đến slave
I2C_Read_Ack	Đọc ACK
I2C_Read_Byte	Nhận một byte từ slave gửi đến master
I2C_Send_Ack	Gửi ACK
I2C_Send_NoAck	Gửi No ACK
I2C_Stop	Tạo tín hiệu kết thúc giao tiếp (Stop condition)

Bảng 5.1 Các hàm I2C

```
I2C_Start:
        setb SDA      ; ensure SDA is high
        setb SCL      ; ensure SCL is high
        clr SDA       ; Pull SDA to low
        ret
```

Triển khai hàm I2C_Start

```
I2C_Send_Byte:
        mov R7, #8;                 (1 cycle)
        send_loop:
        clr SCL       ; ready for data transfer     (1 cycle)
        rlc A ;                     (1 cycle)
        mov SDA, C ;                      (2 cycles)
        acall delay_3us ;                 (6 cycles)
        setb SCL  ;                 (1 cycle)
        acall delay_3us;                  (6 cycles)
        nop     ;                   (1 cycle)
        DJNZ R7, send_loop ;              (2 cycles)
        clr SCL       ;                   (1 cycle)
        ret;                        (2 cycles)
```

Triển khai hàm I2C_Send_Byte

```
I2C_Read_ACK:
        acall delay_2us;                  (4 cycles)
        setb SDA      ;                   (1 cycle)
        setb SCL ;                  (1 cycles)
        mov C, SDA;                       (1 cycle)
        nop;                        (1 cycle)
        nop;                        (1 cycle)
        ret ;                       (2 cycles)
```

Triển khai hàm I2C_Read_ACK

```
I2C_Read_Byte:
        mov R7, #9; counter               (1 cycle)
        clr C
        clr A
        read_loop:
        setb SCL;                   (1 cycle)
        mov C, SDA; read SDA's state            (1 cycle)
        rlc A; shift read bit to A register   (1 cycle)
        acall delay_3us;                  (6 cycles)
        nop;                        (1 cycle)
        clr SCL;                    (1 cycle)
        acall delay_3us;                  (6 cycles)
```

```
        nop;                    (1 cycle)
        DJNZ R7, read_loop;             (2 cycles)
        setb SCL;               (1 cycle)
        ret;                    (2 cycles)
```

Triển khai hàm I2C_Read_Byte

```
I2C_Send_Ack:
        clr SDA;                (1 cycle)
        acall delay_2us;                (4 cycles)
        setb SCL;               (1 cycle)
        ret;                    (2 cycles)
```

Triển khai hàm I2C_Send_Ack

```
I2C_Send_NoAck:
        nop;                    (1 cycle)
        acall delay_2us;        (4 cycles)
        clr SCL;                (1 cycle)
        acall delay_4us;        (8 cycles)
        setb SDA;               (1 cycle)
        setb SCL;               (1 cycle)
        ret;                    (2 cycles)
```

Triển khai hàm I2C_Send_NoAck

```
I2C_Stop:
        nop ;                   (1 cycle)
        nop ;                   (1 cycle)
        clr SCL       ; Push SCL to high        (1 cycle)
        clr SDA;                (1 cycle)
        acall delay_4us;                (8 cycles)
        setb SCL;               (1 cycle)
        setb SDA      ;stop condition           (1 cycle)
        ret;                    (2 cycle)
```

Triển khai hàm I2C_Stop

5.1.1 Đọc dữ liệu từ IC DS1307

Các hàm I2C triển khai cho vi điều khiển AT89S51 không hoạt động độc lập mà phụ thuộc lẫn nhau. Để tạo phiên giao tiếp chuẩn I2C ta cần gọi các hàm theo đúng thứ tự nhất định gọi là thủ tục (procedure). ví dụ để ghi một byte dữ liệu vào IC DS1307 ta cần tuân theo thủ tục sau:

Start condition
↓
gửi địa chỉ DS1307
↓
Đọc Ack
↓
gửi địa chỉ thanh ghi
↓
Đọc Ack
↓
gửi byte dữ liệu
↓
Đọc Ack
↓
Stop condition

```
DS1307_WRITE_BYTE:
        push ACC
        mov A, #DS1307_Address
        acall I2C_Start
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        pop ACC
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        mov A, B
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        acall I2C_Stop
        ret
```

Thủ tục ghi byte dữ liệu vào DS1307

5.1.2 Ghi dữ liệu vào IC DS1307

Thủ tục đọc dữ liệu từ DS1307:

Gọi hàm ghi dữ liệu để set địa chỉ thanh ghi mong muốn

↓

Start condition

↓

Gửi địa chỉ DS1307

↓

Đọc Ack

↓

Đọc byte dữ liệu

↓

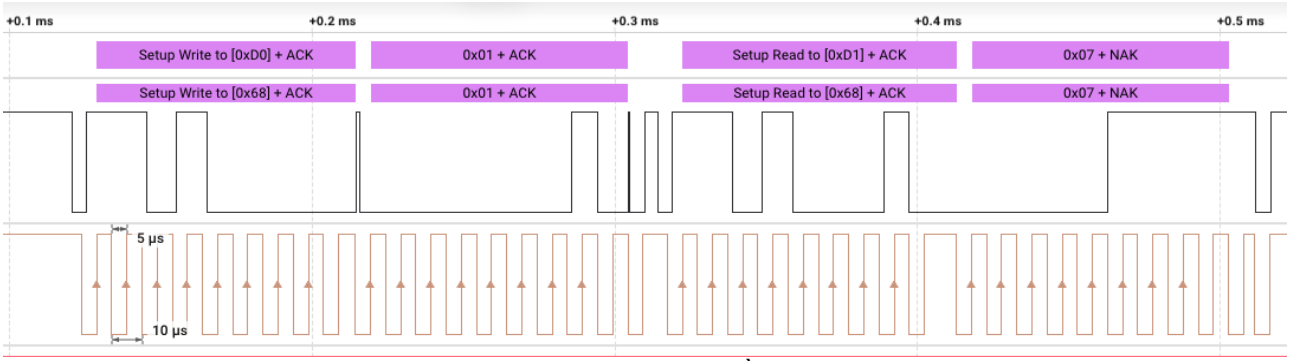
Gửi No Ack

↓

Stop condition

```
DS1307_READ_BYTE:
        PUSH Acc
        mov A, #DS1307_Address
        acall I2C_Start
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        pop Acc
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        acall I2C_Stop
        mov A, #DS1307_Address
        orl A, #1
        acall I2C_Start
        acall I2C_Send_Byte
        acall I2C_Read_ACK
        acall I2C_Read_Byte
        acall I2C_Send_NoAck
        acall I2C_Stop
        RET
```

Thủ tục đọc byte dữ liệu vào DS1307



Hình 5.1.2 Phân tích tín hiệu I2C bằng Logic Analyzer

5.2 HIỂN THỊ VĂN BẢN LÊN LCD

Hình 5.2 hiện thị văn bản lên LCD

Hiển thị giờ/phút/giây ngày/tháng/năm

```
DISPLAY_HOURS_TO_LCD:
        ACALL DS1307_GET_HOURS
        MOV A, 32H  ; hours
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
        ACALL LCD1602_Send_Data
        MOV A, #03Ah ; hienthi dau :
        ACALL LCD1602_Send_Data
        RET

DISPLAY_MINUTES_TO_LCD:
        ACALL DS1307_GET_MINUTES
        MOV A, 31H  ; minutes
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
        ACALL LCD1602_Send_Data
        MOV A, #03Ah
        ACALL LCD1602_Send_Data
        RET

DISPLAY_SECONDS_TO_LCD:
        ACALL DS1307_GET_SECOND
        MOV A, 30H ; seconds
        anl A, #7fh
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
```

```
        ACALL LCD1602_Send_Data
        RET

DISPLAY_DATE_TO_LCD:
        ACALL DS1307_GET_DATE    ; -> 34H
        ACALL DS1307_GET_MONTH   ; -> 35H
        ACALL DS1307_GET_YEAR    ; -> 36H

        MOV A, 34H ; date
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
        ACALL LCD1602_Send_Data
        MOV A, #02Fh
        ACALL LCD1602_Send_Data


        MOV A, 35H ; month
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
        ACALL LCD1602_Send_Data
        MOV A, #02Fh
        ACALL LCD1602_Send_Data


        MOV A, 36H ; year
        ACALL CONVERT_BCD_TO_ASCII
        MOV A, R3
        ACALL LCD1602_Send_Data
        MOV A, R4
        ACALL LCD1602_Send_Data

        RET
```

Mã code hiển thị giờ/phút/giây ngày/tháng/năm

Thuật toán

Ý tưởng: để hiển thị lên màn hình LCD ta phải biết rằng kí tự để hiển thị lên màn hình LED ở dưới định dạng ASCII vậy nên ta cần phải chuyển mã từ BCD sang ASCII trước sau đó mới truyền dữ liệu lần lượt là hàng chục và hàng đơn vị lên LCD

1. Gọi hàm đọc giờ/phút/giây ngày/tháng/năm từ DS1307

```
ACALL DS1307_GET_HOURS
ACALL DS1307_GET_MINUTES
ACALL DS1307_GET_SECOND
ACALL DS1307_GET_DATE
ACALL DS1307_GET_MONTH
ACALL DS1307_GET_YEAR
```

2. Lấy giá trị giờ/phút/giây ngày/tháng/năm từ thanh RAM

```
MOV A, 32H  ; hours
MOV A, 31H  ; minutes
MOV A, 30H ; seconds
MOV A, 34H ; date
MOV A, 35H ; month
MOV A, 36H ; year
```

3. Chuyển từ định dạng BCD sang ASCII bằng cách gọi hàm `CONVERT_BCD_TO_ASCII`

```
ACALL CONVERT_BCD_TO_ASCII
```

4. Riêng với hàm DISPLAY_SECONDS_TO_LCD trước đó ta cần phải làm mất bit CH (Clock Halt)

```
anl A, #7fh
```

5. Cuối cùng gửi dữ liệu hàng chục và hàng đơn vị lên LCD thêm kí tự ":" để hiển thị dưới dạng dd:mm:yy

```
MOV A, R3
ACALL LCD1602_Send_Data
MOV A, R4
ACALL LCD1602_Send_Data
MOV A, #02Fh
ACALL LCD1602_Send_Data
```

Hiển thị thứ lên LCD

```
DISPLAY_DAY_TO_LCD:
        ACALL DS1307_GET_DAY     ; -> 33H
        MOV A, 33H       ; A = thứ (1..7)
        DEC A           ; Đổi về 0..6
       mov B, #4
       mul AB
       add A, #30h
       mov DPL, A
       mov DPH, #0
       acall LCD_SEND_STRING
       ret
```

Hàm hiển thị thứ

Thuật toán

- Ý tưởng: Để hiển thị tên thứ (Mon, Tue …) lên màn LCD ta phải biết rằng DS1307 cung cấp thứ trong tuần ở dạng từ 1 → 7 (1: Chủ nhật, 2: Thứ hai …) vậy nên ta cần chuyển đổi số thứ này thành chuỗi ASCII tương ứng

1. Đọc thứ từ DS1307

```
ACALL DS1307_GET_DAY     ; -> 33H
MOV A, 33H       ; A = thứ (1..7)
```

2. Chuyển về chỉ số mảng 0-6

```
DEC A            ; Đổi về 0..6
```

3. Tính địa chỉ chuỗi cần hiển thị vì mỗi chuỗi (Mon, Tue, Wed …) dài 3 byte nên ta lưu địa chỉ của DPL và DPH sao cho dưới định dạng DPL: 0011, DPH:0000

```
mov B, #4
mul AB
add A, #30h
mov DPL, A
mov DPH, #0
```

4. Gọi hàm hiển thị chuỗi để hiển thị lên LED

```
acall LCD_SEND_STRING
```

Lưu địa chỉ hiển thị và hiển thị lên LCD

```
DISPLAY_TIME_TO_LCD:
        MOV R0, #0
        MOV R1, #0
        ACALL LCD_SETCURSOR
        ACALL DISPLAY_DATE_TO_LCD
        MOV R0, #0
        MOV R1, #13
        ACALL LCD_SETCURSOR
        ACALL DISPLAY_DAY_TO_LCD

        MOV R0, #1
        MOV R1, #0
```

```
    ACALL LCD_SETCURSOR
    ACALL DISPLAY_HOURS_TO_LCD
    ACALL DISPLAY_MINUTES_TO_LCD
    ACALL DISPLAY_SECONDS_TO_LCD
    RET
```

Hàm lưu địa chỉ hiển thị và hiển thị lên LCD

Thuật toán

Ý tưởng hàm này ta chỉ gọi lại các hàm cho nên chỉ có chọn địa chỉ hiển thị và đặt con trỏ hiển thị ngay tại đó qua hàm `LCD_SETCURSOR` và gọi các hàm cần hiển thị tại vị trí đó ra với thanh ghi R0 là dòng và thanh ghi R1 là cột

1. Hiển thị ngày/tháng/năm tại dòng 0, cột 0

```
MOV R0, #0
MOV R1, #0
ACALL LCD_SETCURSOR
ACALL DISPLAY_DATE_TO_LCD
```

2. Hiển thị thứ tại dòng 0, cột 13

```
MOV R0, #0
MOV R1, #13
ACALL LCD_SETCURSOR
ACALL DISPLAY_DAY_TO_LCD
```

3. Hiển thị giờ/phút/giây tại dòng 1 cột 0

```
MOV R0, #1
MOV R1, #0
ACALL LCD_SETCURSOR
ACALL DISPLAY_HOURS_TO_LCD
ACALL DISPLAY_MINUTES_TO_LCD
ACALL DISPLAY_SECONDS_TO_LCD
```

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 KẾT QUẢ ĐẠT ĐƯỢC

6.1.1 Ứng dụng

- Sau quá trình nghiên cứu và thực hiện, đề tài "Thiết kế và xây dựng mạch đồng hồ thời gian thực sử dụng IC DS1307 đã được hoàn thành .
- Hệ thống cho phép hiển thị thời gian thực (giờ, phút, giây, ngày , tháng , năm) lên màn hình LCD 16x2, dữ liệu được duy trì ổn định nhờ IC DS1307 kết hợp với nguồn cắm trực tiếp.
- Ngoài ra, người dùng có thể hiệu chỉnh thời gian bằng nút nhấn và Rotary Encoder một cách dễ dàng.
- Có thể thay thế cho những đồng hồ treo tường hoặc để bàn vì giá thành thấp nhưng đảm khả năng hiện giờ làm góc học tập .

6.1.2 Ưu điểm

- Mạch hoạt động ổn định, tiêu thụ điện năng thấp.
- Đồng hồ giữ thời gian chính xác ngay cả khi mất nguồn nhờ sử dụng pin CR2032 cho IC DS1307.
- Giao diện hiển thị rõ ràng qua màn hình LCD 16x2.
  Sử dụng giao tiếp I2C đơn giản và hiệu quả.
- Có thể mở rộng thêm chức năng như báo thức, hiển thị ngày tháng hoặc đo nhiệt độ.

6.1.3 Hạn chế

- Mạch về giao diện vẫn chưa gọn gàng .
- Hiển thị chưa được đẹp.
- Vẫn còn đi dây chưa tích hợp Mạch PCB vào trong mạch
- Vẫn còn phải sử dụng nguồn trực tiếp.
- Còn hạn chế nhiều chức năng : Hẹn giờ , đổi múi giờ.
- Chưa xử lý tốt được tín hiệu từ nút nhấn của rotary encoder.

6.2 HƯỚNG PHÁT TRIỂN

- Đồ án đã xây dựng thành công sản phẩm đồng hồ RTC bằng module DS1307 dựa trên kế hoạch ban đầu. Phần cứng hiện tại đã chạy ổn định và chính xác. Tuy nhiên đây là lần đầu đồ án đã được triển khai nên có nhiều hạn chế trong cách vận hành đội nhóm, chưa thêm được nhiều chức năng cho đồng hồ, gặp vấn đề về nguồn điện cung cấp cho thiết bị.
- Mạch còn đang được gia công trên bảng mạch đục lỗ sẵn, cần thiết kế được PCB để tối ưu được kích thước của mạch càng nhỏ gọn càng tốt.

- Nâng cao chất lượng hiển thị bằng cách sử dụng màn hình oled để dữ liệu được hiển thị rõ ràng hơn đẹp mắt hơn.
- Thêm nhiều chức năng để hỗ trợ người dùng như báo thức khi đến giờ thì đồng hồ sẽ phát ra tín hiệu, cảnh báo nhiệt độ, thời tiết, la bàn, thêm các chế độ hiển thị ra màn hình 24h format hay 12h format. Tạo ra thêm nhiều ký tự đặc biệt để hiển thị đẹp hơn.
- Cải thiện khối nguồn có pin, vừa có thể sử dụng nguồn trực tiếp vừa có thể sử dụng nguồn từ pin dự phòng. Tích hợp bộ sạc bằng năng lượng mặt trời để tiết kiệm năng lượng.
- Đổi sang linh kiện nhúng khác mạnh hơn như STM32, ESP32 để tích hợp WIFI, màn hình đồ họa và việc lập trình, thêm tính năng được hiệu quả hơn rất nhiều.
- Đổi cách thức hiển thị để đa dạng hóa hình thức và tăng tính thẩm mỹ.
- Nghiên cứu thêm các giải pháp về phần cứng và phần mềm để chống nhiễu tín hiệu.

TÀI LIỆU THAM KHẢO

[1] **A. Corporation**, "8-bit Microcontroller with 4K Bytes Flash," 1996. [Online]. Available: https://ww1.microchip.com/downloads/en/devicedoc/doc0265.pdf.

[2] **A. Industries**, "Using DS1307 RTC with Microcontrollers," 2013. [Online]. Available: https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit.

[3] **H. Ltd.**, "HD44780U LCD Controller/Driver," 1983. [Online]. Available: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf.

[4] **K. Ayala**, *THE 8051 MICROCONTROLLER*, Chicago: United States of America, 2005.

[5] **M. Integrated**, "DS1307: I2C Real-Time Clock," 2008. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS1307.pdf.

[6] **MikroElektronika**, "Interfacing LCD with Microcontrollers," 2010. [Online]. Available: https://www.mikroe.com/.

[7] **N. Semiconductors**, "I2C-bus specification and user manual (UM10204)," 2000. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[8] **S. Electronics**, "LCD 16x2 Tutorial," [Online].