**PHẦN II : LÝ THUYẾT CƠ BẢN CỦA GIAO THỨC TRUYỀN THÔNG I2C**

**2.1 Khái niệm**

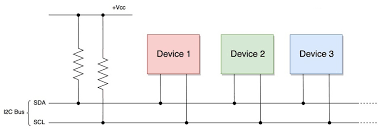
I2C – Inter- Intergrated Circuit là chuẩn truyền thông nối tiếp gồm 2 dây Clock và  Data

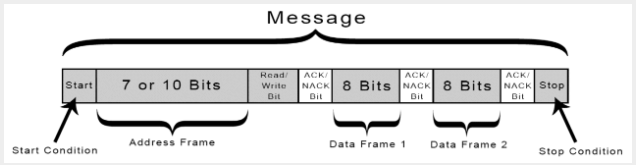
Đây là chuẩn truyền thông đồng bộ bằng xung clock. Tương ứng với 1 bit data là 1 xung clock.

Hai dây được nối với 2 điện trở kéo lên nhằm tránh trường hợp ngắn mạch .

**SDA:** Dây truyền data

**SCL:** Dây truyền xung clock





Tóm tắt quá trình chuyền như sau:

**1.2 Các thông số cơ bản trong truyền nhận dữ liệu I2C**

**START:** Điều khiện bắt đầu thông báo có một gói dữ liệu sắp truyền tới

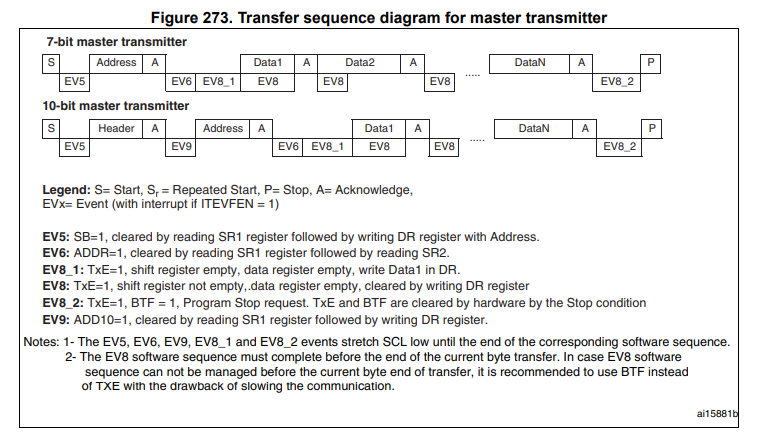
**ADDRESS FRAME**: Khối bit địa chỉ , để phân biệt các thiết bị với nhau, mỗi thiết bị sẽ được gắn 1 địa chỉ vật lý cố định riêng. [ hai kiểu 7 hoặc 10 bit].

**Read/Write Bit:** Dùng để xác định quá trình truyền nhận dữ liệu từ thiết bị Master, muốn Read hoặc Write dữ liệu.

**ACK/NACK:** Bit phản hồi từ Slave, so sánh địa chỉ vật lí của Slave có trùng khớp hay không nếu trùng khớp cho truyền nhận, nếu sai ngưng truyền nhận dữ liệu.

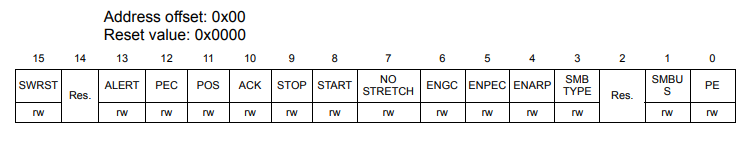
**DATA FRAME:** Sau khi trùng khớp với địa của thiết bị Slave, khối bit dữ liệu bao gồm dữ liệu từ thiết bị master gửi tới slave.

**STOP:** Thống báo điều khiện kết thúc quá trình chuyền data.



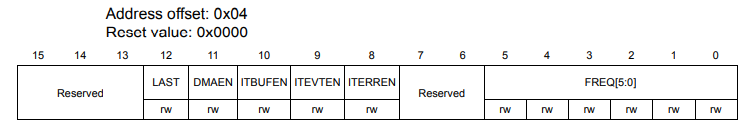
**2.2 Một số thanh ghi quan trọng**

**I2C Control register 1 (I2C\_CR1)**



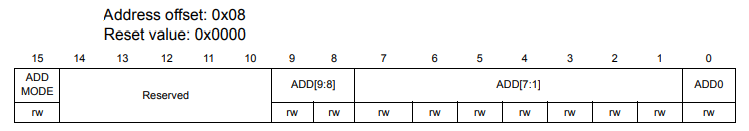
* ACK : Cho phép nhận hoặc không nhận tín hiệu ACK return từ thiết bị được truyền(master).
* STOP : Sinh ra tín hiệu stop kết thúc quá trình giao tiếp.
* START : Sinh ra tín hiệu start để bắt đầu quá trình giao tiếp.
* PE: Peripheral enable – bit này được bật lên bằng 1 khi quá trình giao tiếp I2C đang được thực hiện và kết thúc khi có tín hiện End communication.

**I2C Control register 2 (I2C\_CR2)**



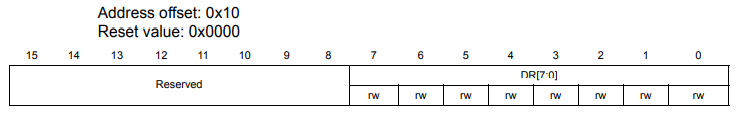
* ITBUFEN : Cho phép hoặc không cho phép xảy ra **ngắt khi có Data** trong truyền/nhận.
* ITEVTEN : Cho phép **ngắt** khi xảy ra các **sự kiện** sau trên các bit sau: SB, ADDR, ADD10, STOPF, BTF , TxE, RxE.
* ITERREN : Cho phép hoặc không cho phép xảy ra **ngắt khi có lỗi xảy ra**.
* FREQ[5:0] : Bộ chia clock tần số được tính từ nhánh clock hệ thống chia cho bộ I2C.

**I2C Own address register 1 (I2C\_OAR1)**



* ADDMODE : Cài đặt số bit địa chỉ của slave là 7 hay 10 bit.
* ADD[9:8] : Chỉ sử dụng khi mode địa chỉ là 10 bit.
* ADD[7:1] : Bit địa chỉ của thiết bị slave.
* ADD0 : Chỉ sử dụng khi mode địa chỉ là 10 bit.

**I2C Data register (I2C\_DR)**

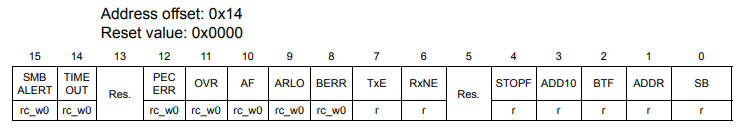


Thanh ghi này gồm 8 bit, chứa data của quá trình truyền hoặc nhận.

Nhận dữ liệu TxE= 1

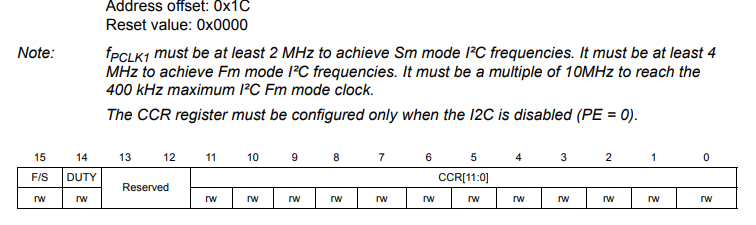
Truyền dữ liệu RxNE=1

**I2C Status register 1 (I2C\_SR1)**



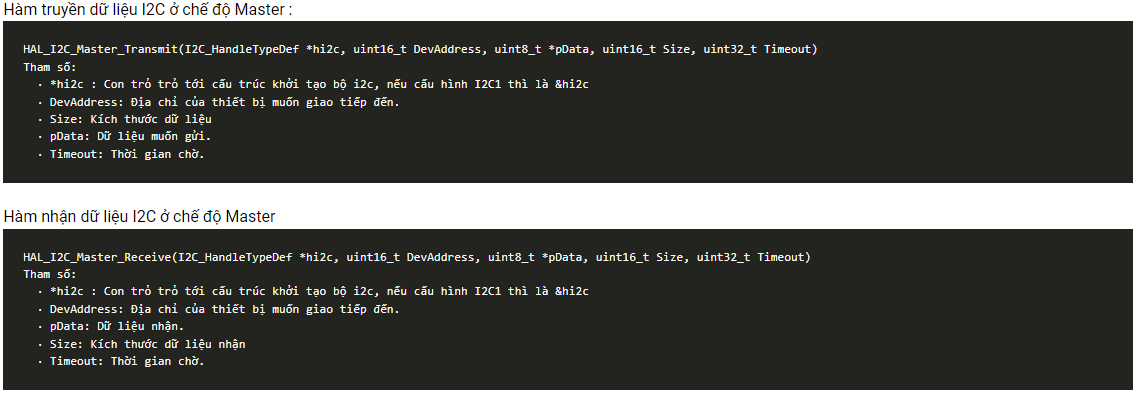
* TIME OUT: Báo hết thời gian đợi quá trình truyền nhận data.
* PECERR: Chấp nhận lỗi PCE và phản hồi lại quá trình tín hiệu lỗi.
* OVR : Báo quá trình overrun/ underrun.
* AF: Báo tín hiệu ACK bị lỗi hay không.
* TxE : Báo buffer truyền có trống hay không.
* RxE : Báo buffer nhận có trống hay không.
* STOPF : Cờ báo có kiểm tra quá trình stop hay không.
* BTF: Báo truyền byte dữ liệu đã xong hay chưa.
* ADDR: Báo địa chỉ đã được gửi(mode master) hay không tương thích (mode slave).
* SB: Có sử dụng bit start hay không.

**I2C Clock control register (I2C\_CCR)**



* F/S : Sử dụng mode Fast(400khz) hay Standard(100Khz).
* DUTY: Chỉ sử dụng cho Fast mode với 2 chế độ .
* CCR[11:0] : Thanh ghi chứa Clock control cho bộ , công thức chia tham khảo thêm trong reference manual.

**Hàm sử dụng truyền nhận dữ liệu mà thư viện Hal cung cấp:**



**Ưu điểm**

Sử dụng chỉ hai dây

Hỗ trợ nhiều máy chủ chính và nhiều máy chủ nô lệ

Bit ACK / NACK xác nhận rằng mỗi khung hình đã được truyền thành công

Phần cứng không phức tạp như UART

Giao thức nổi tiếng và được sử dụng rộng rãi

**Nhược điểm**.

Tốc độ truyền chậm hơn SPI

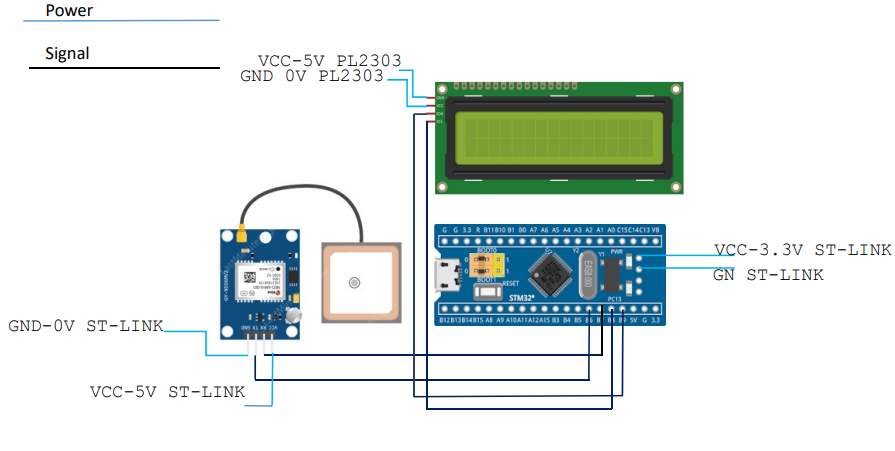
Kích thước của khung dữ liệu được giới hạn trong 8 bit

Thực hiện phần cứng phức tạp hơn SPI

**PHẦN 3: TRUYỀN NHẬN DỮ LIỆU GPS LÊN LCD**

**3.1 Cấu hình Cube MX**

**3.2 Sơ đồ kết nối chân**



**Kết nối GPS NEO 6M :**

**VCC-5v** và **GND-0v** cấp từ mạch nạp ST-Link

Chân RX nối chân TX **PB6** của board STM32F103C8

Chân TX nối chân RX **PB7** của board STM32F103C8

**Hiển thị lên máy tính:**

Chân RX nối chân TX của board PL2303

Chân TX nối chân RX của board PL2303

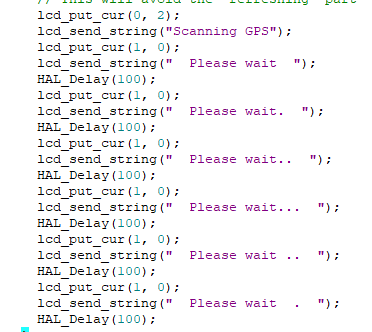
**Kết nối LCD có tích hợp IIC:**

**VCC-5v** và **GND-0v** cấp từ mạch chuyển đổi USB qua UART **PL2303**

Chân **SDA** kết nối với **PB9** của board STM32F103C8

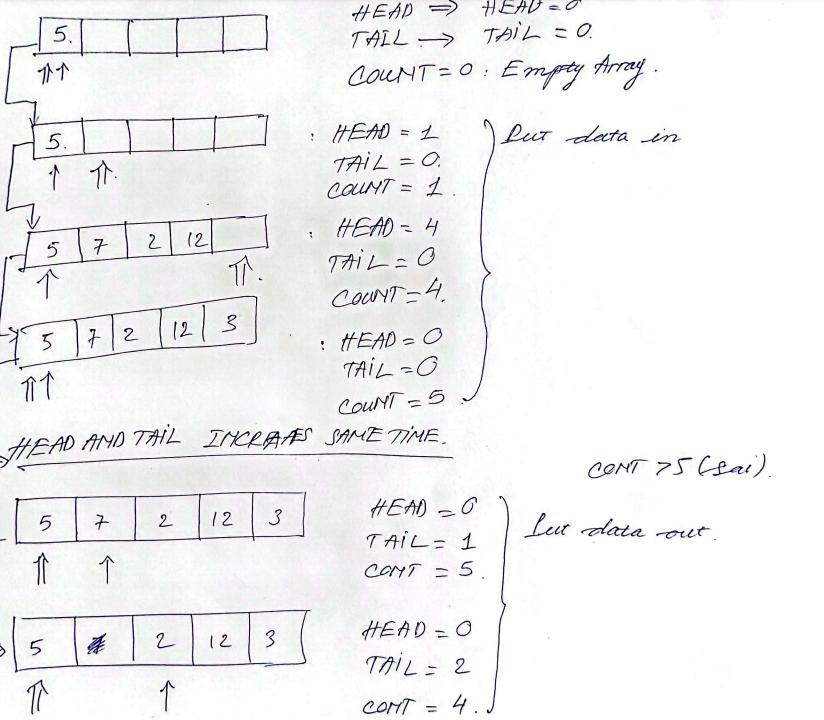
Chân **SCL** kết nối với **PB8** của board STM32F103C8

**3.3 Thư viện i2c lcd**

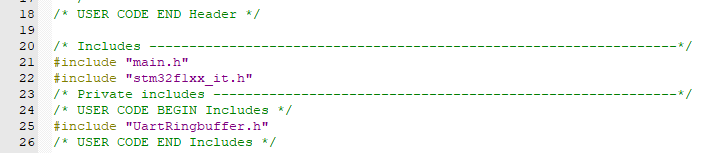


**3.4 Thư viện UARTRingbuffer**

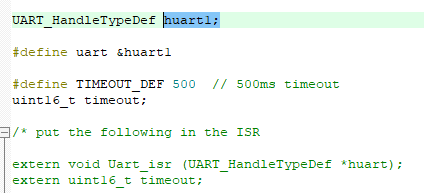
Có thể hiểu 1 cách đơn giản Ring\_buffer là 1 bộ đệm hình tròn (hình nhẫn) dùng để chứa dữ liệu, nó cũng như 1 bộ đệm thẳng nhưng điểm đặc biệt của bộ đệm này là có thể chứa vô số dữ liệu nếu ta biết phối hợp việc đọc, ghi dữ liệu bộ đêm 1 cách nhịp nhàng với số ô nhớ giới hạn, còn bộ đệm thẳng thì chứa càng nhiều dữ liệu thì sẽ càng tốn bộ nhớ.



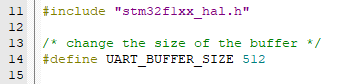
**3.5 Thực hiện viết chương trình lấy dữ liệu GPS**



Tiếp theo điều chỉnh thành **huart** bạn muốn sử dụng.Ở đây mình dùng UART1 cấu hình từ CubeMX

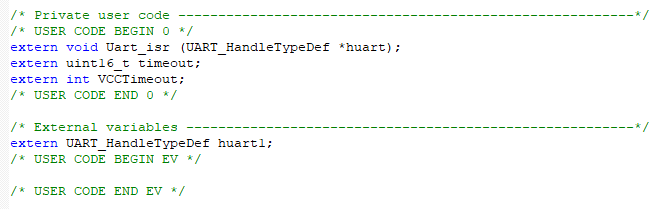


Bên cạnh đó, để đảm bảo có thể đọc hết và không bị thiếu dữ liệu trả về từ GPS ta nên chỉnh lại bộ đệm Rx UART\_BUFFER\_SIZE thành 512 ở trong file **UARTRingbuffer.h** bit vì bộ đệm sử dụng dữ liệu khá lớn.

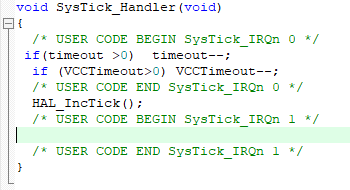


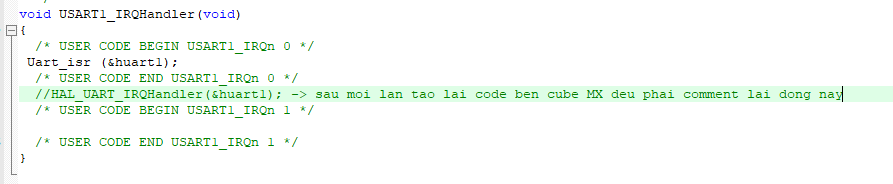
Sau đó extern các hàm sau vào từ file **UARTRingbuffer.c**

vào /\* USER CODE BEGIN 0 \*/ của file **stm32f1xx\_it.c**



**Giảm giá trị**

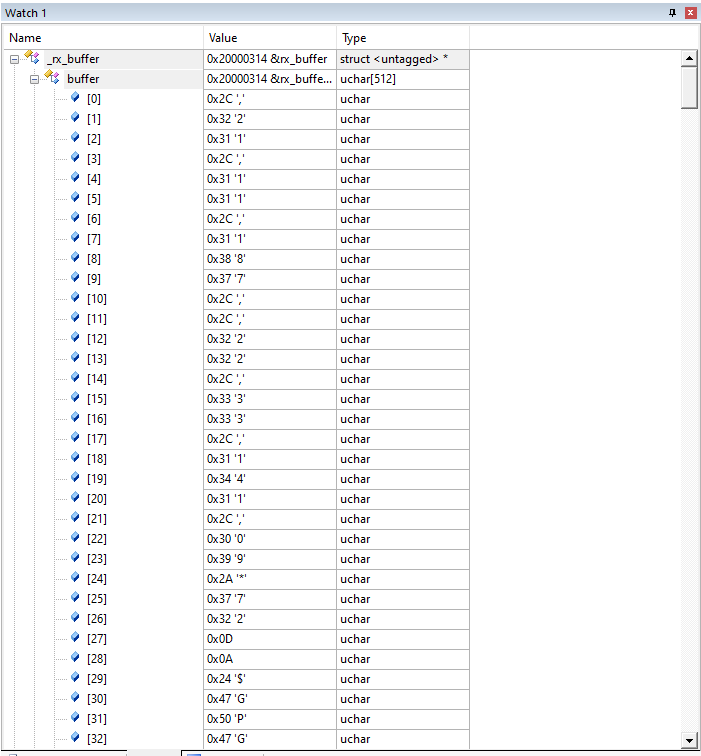




**Debug chương trình:**

Rx\_buffer: **Add rx\_buffer** vào **watch table 1:**

Nếu ta thấy các chuỗi xuất hiện và thay đổi cóp nghĩa chương trình của chúng ta đã chạy đúng

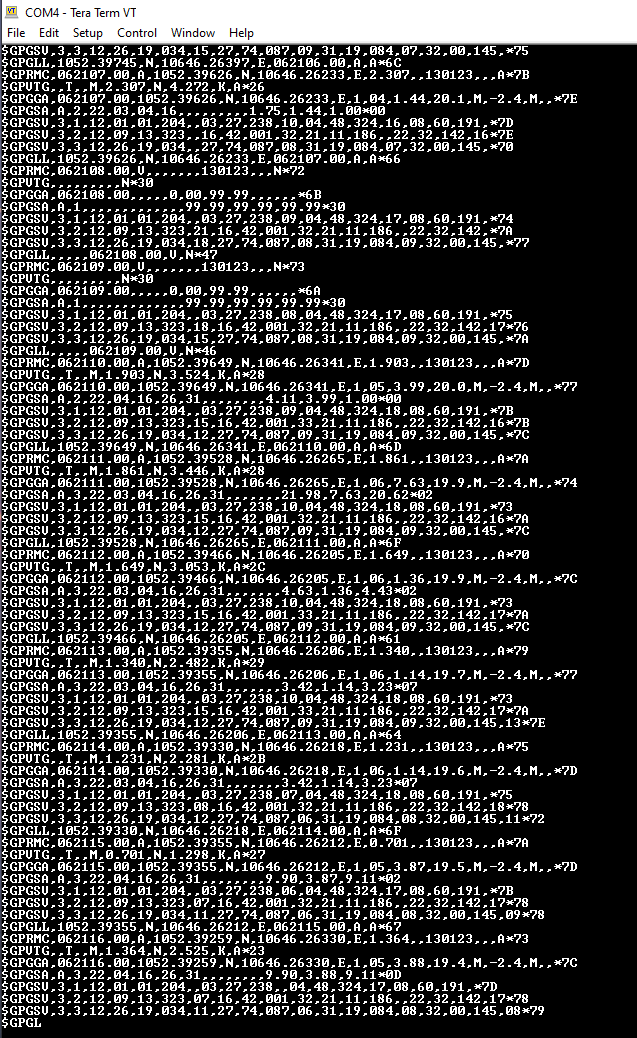


Các chuỗi hiển thị trên Rx\_buffer ta có thể quan sát nó qua phần mềm Tera Term hoặc Hercules đếu được

Để hiển thị lên máy tính:

Chân RX nối chân TX của board PL2303

Chân TX nối chân RX của board PL2303



Để lấy giá trị gồm thời gian, ngày tháng năm kinh độ, vĩ độ , tính toán vận tốc, số lượng vệ tinh ta cần chú ý 2 chuỗi sau từ dữ liệu trả về của GPS:

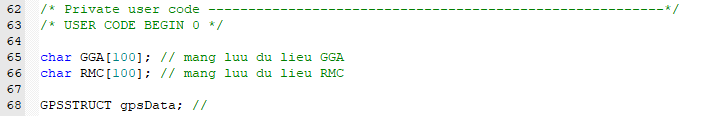


$GPRMC, thời gian, kiểm tra dữ liệu,Vĩ độ, hướng Bắc (North), Kinh độ, hướng Đông(East),…., Ngày/tháng/Năm, kiểm tra dữ liệu

$GPRMC, thời gian UTC, Vĩ độ, hướng Bắc (North), Kinh độ, hướng Đông(East),

**FILE MAIN.C**

Tạo mảng lưu dữ liệu từ GGA và RMC:



Hàm decode của GGA và RMC được lấy từ thư viện NMEA:



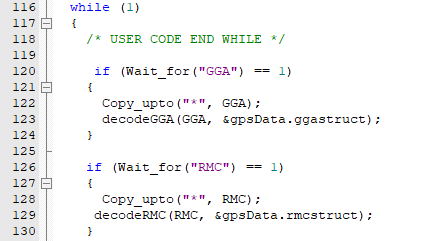
**Hàm int Wait\_for (char \*string)**

Để đề phòng trường hợp không phát hiện chuỗi, hàm sẽ được đạt một thời gian chờ nhất định nếu hết thời gian chờ sẽ thực thi câu lệnh tiếp theo.

**Hàm int Copy\_upto (char \*string, char \*buffertocopyinto)**

Nếu chuỗi được phát hiện

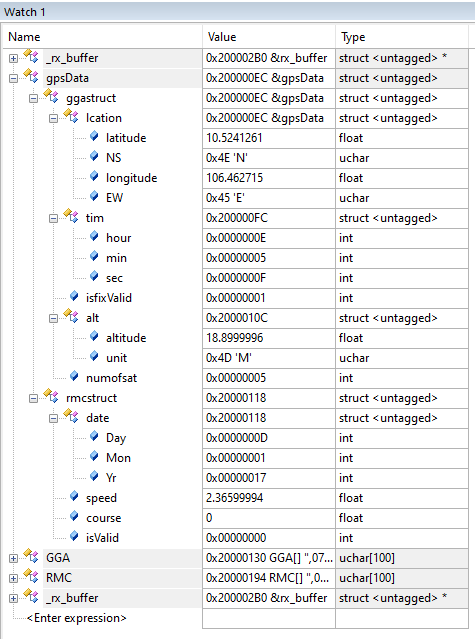
Sẽ copy dữ liệu từ GGA hoặc RMC từ đầu đến vị trí dấu “\*”



Để có được thời gian đúng, ta cần phải chỉnh lại GMT biến thành +700, vì giờ GMT (Greenwich Mean Time) lệch với Việt Nam là 7 tiếng (GMT +7).



**Add thêm gpsData,GGA,RMC vào watch table để xem**



Trong đó :

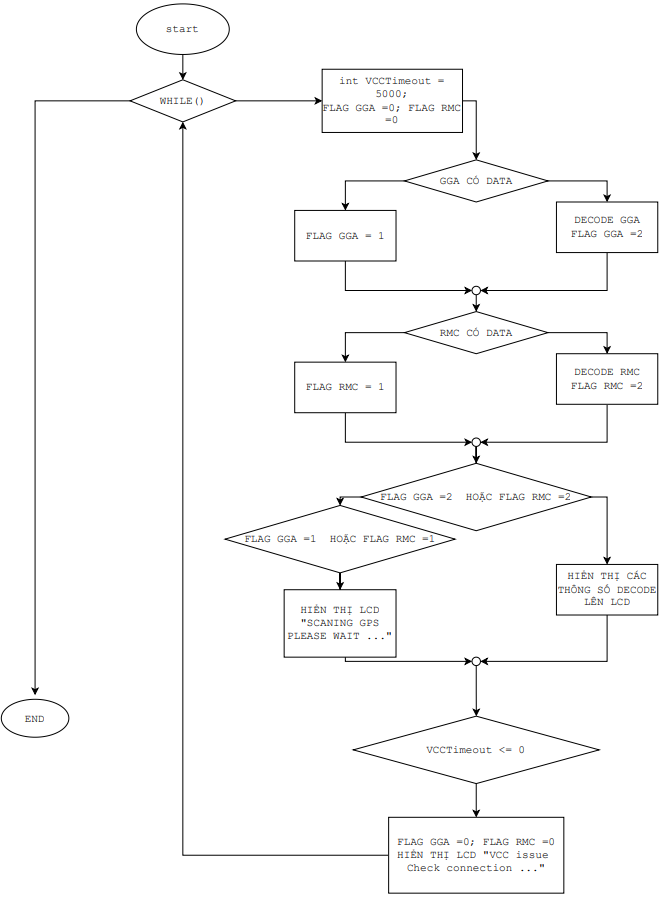
**Latitude**: Vĩ độ, **Longtitude**: Kinh độ**, NS**: North South, **EW**: East West, **Altitude**: Độ cao so với mặt nước biển v.v

Giá trị xem được khi chuyển đồi Hex sang Decimal

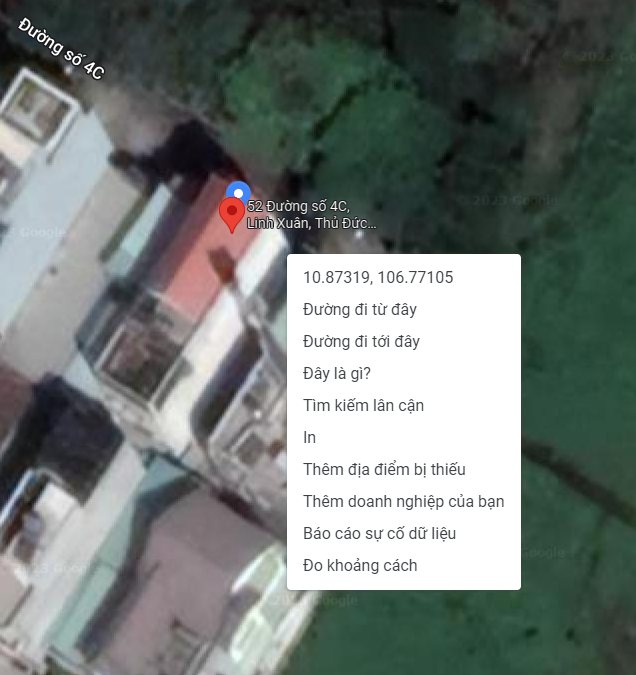
Ví dụ: Hour : 0x0000000E = 14 giờ

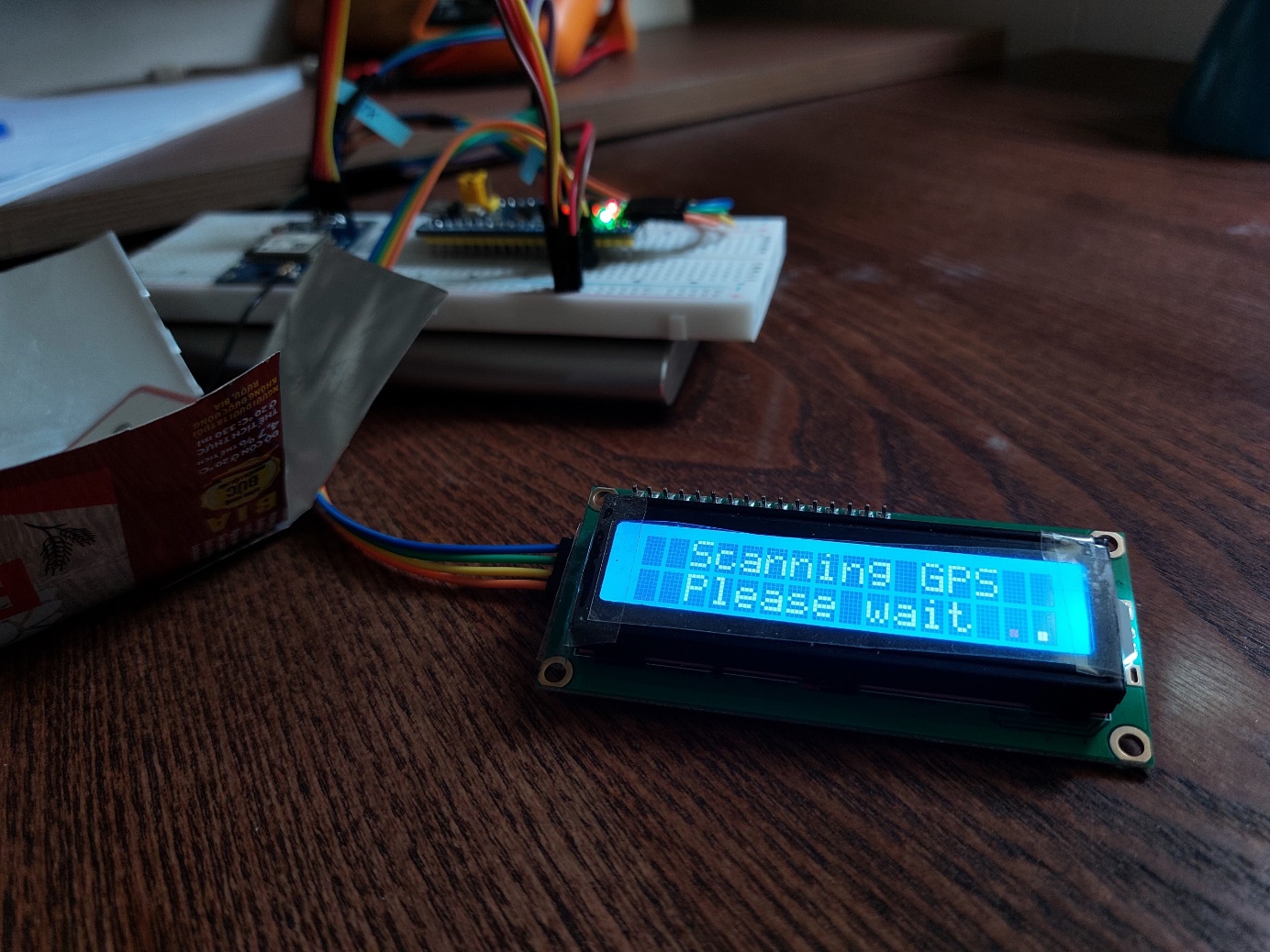
Min: 0x00000005 = 5 phút

Chương trình chính hoạt động như sau:



**PHẦN 4: KẾT QUẢ ĐẠT ĐƯỢC**



****

