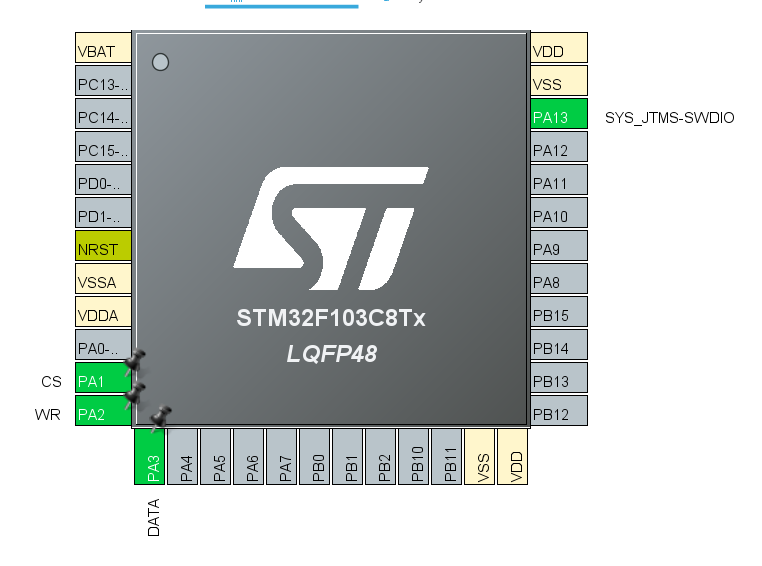
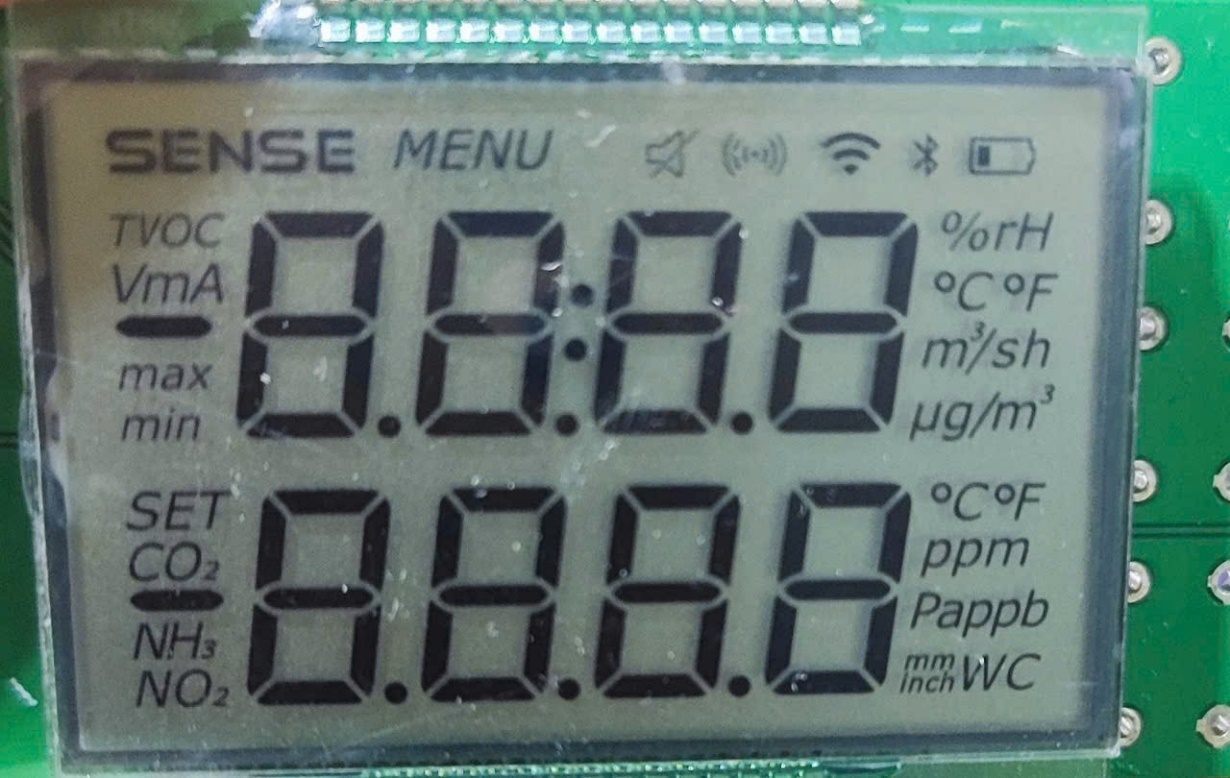
**HƯỚNG DẪN SỬ DỤNG HT1612**

**I/ Sơ đồ đấu dây thiết bị HT1612 và STM32F103C8T6**

**II/ Hiển thị số, chữ theo mong muốn:**

Bước 1: Khởi động LCD sử dụng lệnh Init\_1621();

Bước 2: Kiểm tra LCD có sáng đầy đủ sau đó tắt LCD:

 HT1621\_all\_on(32);

HT1621\_all\_off(32); // tắt toàn bộ màn hình

Bước 3: Sau khi hoàn thành 2 bước trên thì bước vào hàm main để hiển thị lên LCD theo yêu cầu của bạn:

* Để hiển thị các giá trị mong muốn ta sử dụng hàm:
  + Hiển thị kí tự đặc biệt:  **Write\_1621**(uchar addr, uchar sdata);
    - Ví dụ: Write\_1621(0x0B,0x01<<0);
  + Hiển thị trên các thanh led: **WriteMulti\_1621**(uint8\_t startAddr, **const** uint8\_t \*pData, uint8\_t length,uint8\_t dot);
    - Ví dụ: WriteMulti\_1621(0x18, number\_line2[1], 2 ,DOT\_LINE2);
* Bảng địa chỉ và giá trị đưa vào để hiển thị kí tự đặc biệt:
  + Write\_1621(Địa chỉ bắt đầu, Giá trị);
  + Ví dụ: Write\_1621(0x0B,0x01<<0);
    - * Nhìn vào bảng dưới ta thấy tại địa chỉ **0x0B** thì **bit 0** được kích lên 1 nên màn hình sẽ hiện giá trị **℃**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Addr | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0x0A | inch | mm | Pa | ppm |
| 0x0B | WC | ppb | ℉ | ℃ |
| 0x0C |  |  |  |  |
| 0x0D | Bluetooth | Dấu phẩy |  | : |
| 0x0E | Pin | Độ | m/ | / |
| 0x0F | % | C | S | M |
| 0x10 | rH | ℉ | h | Dấu phẩy |
| 0x11 | MENU | SENSE | TVOC | v |
| 0x12 | min | max | Dấu trừ hàng 1 | mA |
| 0x13 | Lập Phương | Dấu trừ hàng 1 | NH3 | NO2 |

* Bảng địa chỉ và giá trị hiển thị trên các thanh led:
  + **WriteMulti\_1621**( Địa chỉ bắt đầu, Giá trị, Số giá trị ,Dấu chấm);
  + **Vị trí led** hàng 1 từ trái sang phải có địa chỉ: 0x02,0x04,0x06,0x08
    - Số : WriteMulti\_1621(**Vị trí led**, number\_line1[Số] , 2, 0);

Ví dụ: WriteMulti\_1621(**0x02**, number\_line1[0] , 2, 0);

* + - * Hiện số 0 ở vị trí led đầu tiên.

Ví dụ: WriteMulti\_1621(**0x02**, number\_line1[0] , 2, DOT\_LINE1);

* + - * Hiện số 0 ở vị trí led đầu tiên và có dấu chấm.
  + Tương tự với Led hàng 2 có địa chỉ: 0x14,0x16,0x18,0x1A

 Để hiện như hình sau:

* Code như sau:

Hiện số hàng 1:

WriteMulti\_1621(0x04, number\_line1[6] , 2, 0);

WriteMulti\_1621(0x06, number\_line1[4] , 2, DOT\_LINE1);

WriteMulti\_1621(0x08, number\_line1[3] , 2, 0);

Hiện số hàng 2:

WriteMulti\_1621(0x16, number\_line2[3] , 2, 0);

WriteMulti\_1621(0x18, number\_line2[1], 2 ,DOT\_LINE2);

WriteMulti\_1621(0x1A, number\_line2[4] , 2, 0);

Các giá trị hiện lần lượt như sau: Độ C, %,rH,SENSE

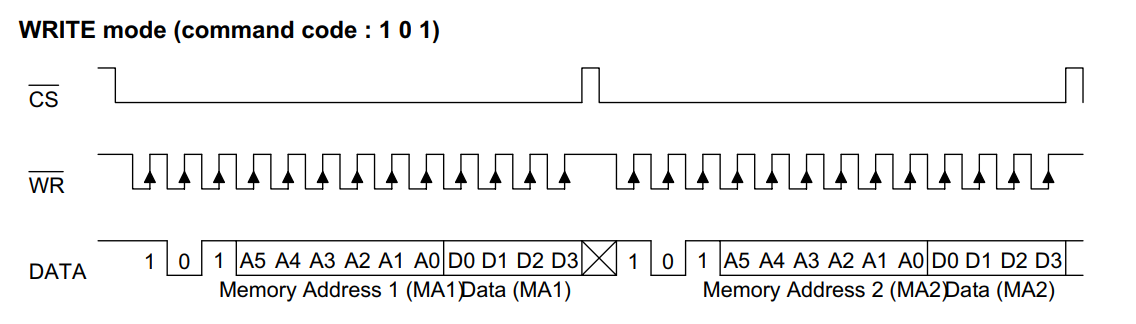
Write\_1621(0x0B,0x01<<0);

Write\_1621(0x0F,0x01<<3);

Write\_1621(0x10,0x01<<3);

Write\_1621(0x11,0x01<<2);

III/ Giải thích code:

1. Write mode

* Theo datasheet ta thấy: để gửi 1 data đi thì phải có 1 xung clock từ WR.
* Ta được hàm sau:

**void** **SendBit\_1621**(uchar sdata, uchar cnt)

{

uchar i;

**for**(i = 0; i < cnt; i++)

{

WR0;

**if**(sdata & 0x80)

{

DATA1;

}

**else**

{

DATA0;

}

WR1;

sdata <<= 1;

}

}

* Giải thích:
  + Nhận vào hai tham số:
    - sdata : Dữ liệu 8-bit cần gửi
    - cnt : Số lượng bit cần gửi
  + Các biến vòng lặp:
    - for(i = 0; i < cnt; i++) : lặp cnt lần để gửi từng bit sdata
  + Quá trình gửi từng bit:
    - WR0; : hạ mức tín hiệu trên chân ghi WR xuống mức thấp.
    - if(sdata & 0x80){

DATA1;}

else{

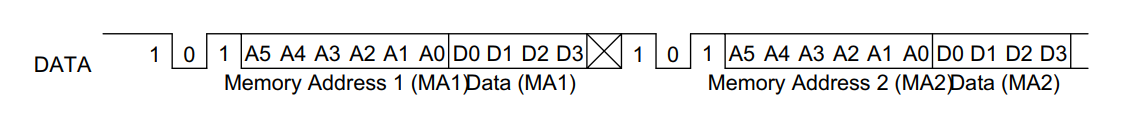
DATA0;}

Kiểm tra bit cao nhất (MSB - Most Significant Bit) của sdata bằng phép toán **sdata x 0x80**

* + - WR1; : nâng mức tín hiệu trên chân ghi WR lên mức cao.
    - sdata <<= 1;: Dịch trái sdata một bit để chuẩn bị gửi bit tiếp theo trong vòng lặp kế tiếp.
  + ví dụ: SendBit\_1621(0xA0, 3);

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vòng lặp i | sdata | Bit gửi(MSB | DATA | WR0 | WR1 | sdata <<= 1 |
| 0 | 10100000 | 1 | 1 | 0 | 1 | 01000000 |
| 1 | 01000000 | 0 | 0 | 0 | 1 | 10000000 |
| 2 | 10000000 | 1 | 1 | 0 | 1 | 00000000 |

* Frame truyền của write mode:

Gồm: 3 bit command code (101), 6 bit địa chỉ và 4 bit dữ liệu

* + Code như sau:

**void** **Write\_1621**(uchar addr, uchar sdata)

{

addr <<= 2;

CS0;

SendBit\_1621(0xA0, 3);

SendBit\_1621(addr, 6);

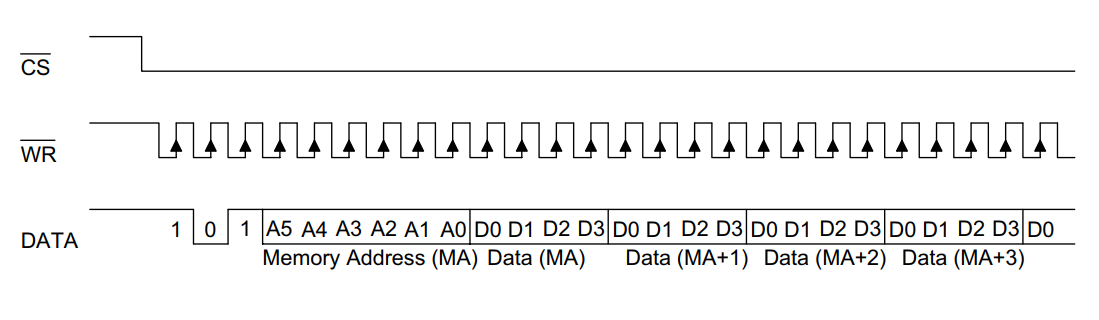
SendBit\_1621((sdata & 0x0F) << 4, 4);

CS1;

}

* + Giải thích:
    - addr <<= 2; : vì đọc bit từ MSB về LSB mà địa chỉ trong HT1621 là **6-bit**, nhưng vị trí lưu trữ lại nằm trong một khung 8-bit
    - SendBit\_1621(0xA0, 3); : HT1621 sử dụng **3-bit opcode** để chỉ định lệnh, trong đó:
      * 101 là lệnh ghi dữ liệu vào RAM.
      * Vì chỉ có **3 bit được gửi**, nó sẽ gửi 101 trước.
    - SendBit\_1621(addr, 6); : Địa chỉ đã được dịch trái 2 bit ở bước 1, giờ sẽ được gửi đi với độ dài **6 bit**.
    - SendBit\_1621((sdata & 0x0F) << 4, 4);:
      * sdata & 0x0F: Lấy 4 bit thấp của sdata
      * << 4 : Dịch trái 4 bit để đưa **4 bit dữ liệu** vào vị trí cao của byte, vì hàm SendBit\_1621 đọc dữ liệu theo dạng **MSB trước**.
    - CS1; : nâng mức tín hiệu trên chân ghi CSlên mức cao để hoàn tất quá trình giao tiếp
  + Ví dụ: Write\_1621(0x03, 0x05);
    - addr = 0x03 = 0b000011
      * Dịch trái 2 bit → addr = 0b001100
    - sdata = 0x05 = 0b00000101
      * Lấy 4 bit thấp 0b0101, dịch trái 4 → 0b01010000

|  |  |  |
| --- | --- | --- |
| Gửi dữ liệu | Nội dung | Số bit |
| 101 | |  | | --- | |  |  |  | | --- | | Lệnh ghi RAM | | 3 |
| 001100 | |  | | --- | | Địa chỉ đã dịch trái |  |  | | --- | |  | | 6 |
| 0101 | |  | | --- | | Dữ liệu đã dịch trái |  |  | | --- | |  | | 4 |

1. Write mode ( successive address writing)

Tương tự với write mode ở trên, tuy nhiên với chế độ này ta viết được nhiều gói data một lúc và các gói data này sẽ nằm ở địa chỉ liền kề nhau.

* Code như sau:

**void** **WriteMulti\_1621**(uint8\_t startAddr, **const** uint8\_t \*pData, uint8\_t length,uint8\_t dot)

{

startAddr <<= 2;

CS0;

SendBit\_1621(0xA0, 3);

SendBit\_1621(startAddr, 6);

**for** (uint8\_t i = 0; i < length; i++)

{

uint8\_t multidata = (pData[i] & 0x0F) << 4;

**if**(i==0)

{multidata |= dot;}

SendBit\_1621(multidata, 4);

}

CS1;

}

* Giải thích code:
  + startAddr <<= 2;: Dịch trái startAddr **2 bit** để phù hợp với cách lưu trữ địa chỉ 6-bit của **HT1621**.
  + CS0;: **Hạ CS xuống mức thấp** để bắt đầu giao tiếp
  + SendBit\_1621(0xA0, 3); : lệnh ghi dữ liệu vào RAM.
  + SendBit\_1621(startAddr, 6); : Gửi **địa chỉ đã dịch trái 2 bit** với **6-bit**.
  + for (uint8\_t i = 0; i < length; i++){

uint8\_t multidata = (pData[i] & 0x0F) << 4;

if(i == 0){

multidata |= dot;}

SendBit\_1621(multidata, 4);}

* **Duyệt qua từng phần tử của pData[]**, thực hiện các bước:
  + Lấy **4 bit thấp của dữ liệu** (pData[i] & 0x0F)
  + Dịch trái **4 bit** (<< 4) để đúng định dạng của **HT1621**.
  + **Nếu là byte đầu tiên (i == 0)**, kết hợp thêm dot (multidata|= dot;);)
  + Gửi **4 bit** này bằng SendBit\_1621(multidata, 4);
* Ví dụ:

uint8\_t data[] = {0x5, 0xA, 0x3};

WriteMulti\_1621(0x02, data, 3, 0x08);

* + startAddr = 0x02 = 0b000010
    - Dịch trái 2 → startAddr = 0b001000
  + pData[] = {0x5, 0xA, 0x3}
    - 0x5 = 0b00000101
    - 0xA= 0b00001010
    - 0x3 = 0b00000011
  + dot = 0x08 = 0b00001000

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vòng I | pData[] | Sau khi lấy 4 bit thấp | Dịch trái (<< 4) | Kết hợp dot (nếu i == 0) | Gửi đi |
| 0 | 00000101 | 00000101 | 01010000 | 01011000 thêm dot | 01011000 |
| 1 | 00001010 | 00001010 | 10100000 | 10100000 | 10100000 |
| 2 | 00000011 | 00000011 | 00110000 | 00110000 | 00110000 |