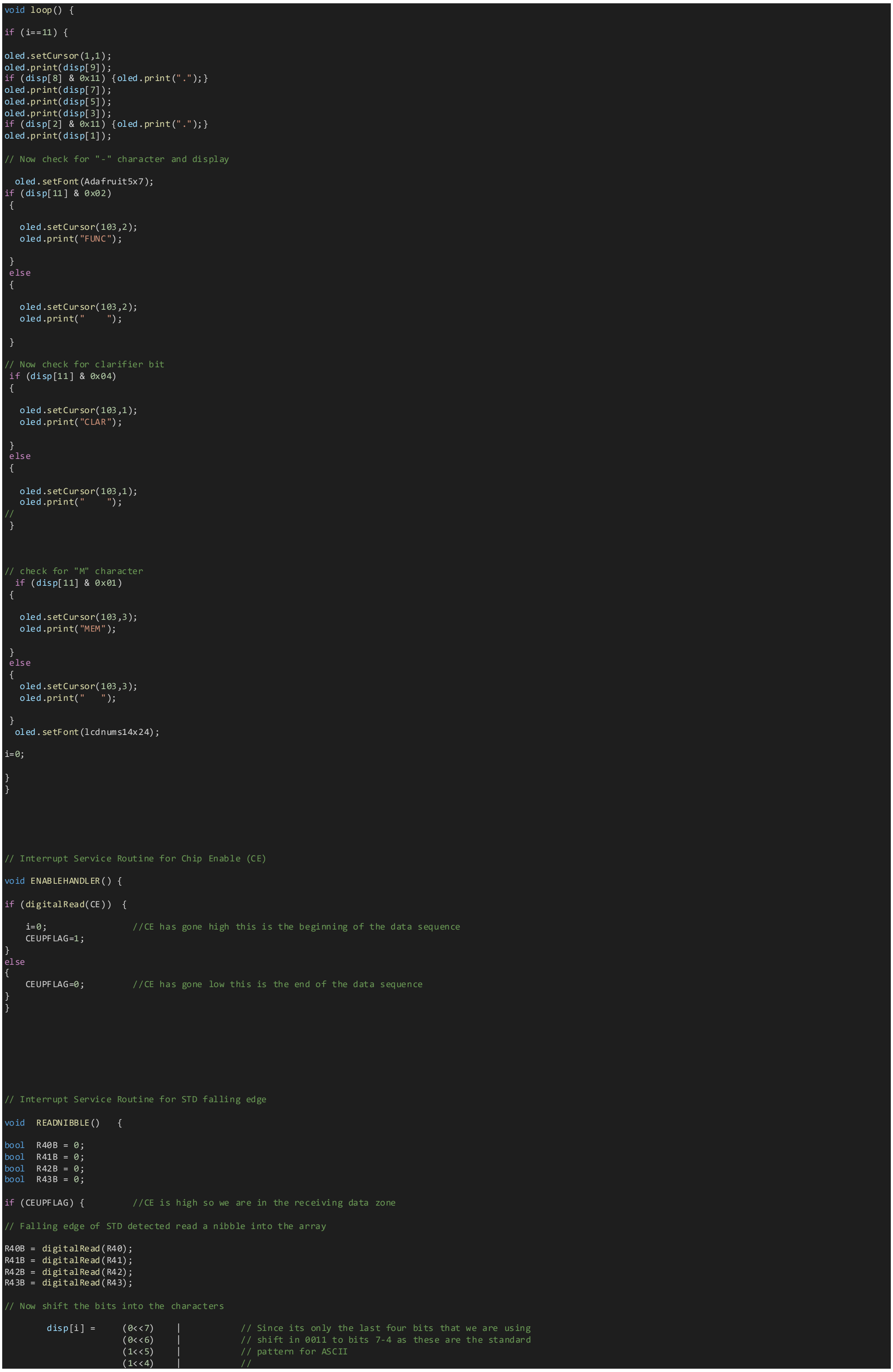
```
// An Arduino based FT290r Mk 1 display decoder.
// Nigel Stevens G6RZR
// 16th June 2022
// A simple Arduino Nano is powerful enough to read in the data from the FT290R
// and display it on an .96" OLED

#include <Wire.h>
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"

// 0X3C+SA0 - 0x3C or 0x3D
#define I2C_ADDRESS 0x3C

// Define proper RST_PIN if required.
#define RST_PIN -1

//Define FT290R display connections

#define CE   3       //Chip enable is connected to D3
#define STD  2       //STD is connected to D2
#define R40  5       //R40 is connected to D5
#define R41  6       //R41 is connected to D6
#define R42  7       //R42 is connected to D7
#define R43  8       //R43 is connected to D8

#define LEDLIGHT  9     // Meter led

static int i=0;
static char disp[12]={0,0,0,0,0,0,0,0,0,0,0,0}; // Table to hold the 12 nibbles of character data
static bool CEUPFLAG = 0;                       // Global flag which indicates that we are in a chip enabled state

SSD1306AsciiWire oled;
//------------------------------------------------------------------------------




void setup() {
  Wire.begin();
  Wire.setClock(400000L);

// Define inputs from FT290

pinMode(CE, INPUT);            // set pin to input
pinMode(STD, INPUT);           // set pin to input
pinMode(R40, INPUT);           // set pin to input
pinMode(R41, INPUT);           // set pin to input
pinMode(R42, INPUT);           // set pin to input
pinMode(R43, INPUT);           // set pin to input

// Define an output for the LED that lights the meter
pinMode(LEDLIGHT, OUTPUT);



// set up interrupt pins

attachInterrupt(digitalPinToInterrupt(CE), ENABLEHANDLER, CHANGE);  // Chip enable interupt handler
attachInterrupt(digitalPinToInterrupt(STD), READNIBBLE, FALLING);   // 4 bits of data are clocked into the arduino on the falling edge of STD

// set up OLED display

#if RST_PIN >= 0
  oled.begin(&Adafruit128x32, I2C_ADDRESS, RST_PIN);
#else // RST_PIN >= 0
  oled.begin(&Adafruit128x32, I2C_ADDRESS);
#endif // RST_PIN >= 0

// Display an initial message

  oled.setFont(Adafruit5x7);
  oled.set2X();
  oled.setCursor(1,2);
  oled.println("Hi G6RZR");
  oled.set1X();
  delay(2000); // leave the message up for a couple of seconds

// Turn on the meter light

  digitalWrite(LEDLIGHT,HIGH);

// Change the font for one that looks like an LCD

  oled.setFont(lcdnums14x24);
  oled.clear();

}



//------------------------------------------------------------------------------
// Main loop. The simple function of the loop is to update the display
// The structure of the received data is as follows:
//
// disp[9]  is the first character of the display
//
// e.g. for 144.250 MHz display would read 4.250.0
// so disp[9] would be '4'
//
// disp[8] and disp[2] carry the decimal point
//
// disp[7], disp[5], disp[3] and disp[1] are the remaining digits
//
// Finally disp[11] holds bits indicating CLAR, FUNC and MEM
//
// so 00110001   Should display MEM
//    00110010   Should display FUNC
//    00110100   Should display CLAR
//    These could also all be displayed at once with the sequence
//    00110111   Would display MEM, FUNC and CLAR
//
//------------------------------------------------------------------------------
```



```
void loop() {

if (i==11) {

oled.setCursor(1,1);
oled.print(disp[9]);
if (disp[8] & 0x11) {oled.print(".");}
oled.print(disp[7]);
oled.print(disp[5]);
oled.print(disp[3]);
if (disp[2] & 0x11) {oled.print(".");}
oled.print(disp[1]);

// Now check for "-" character and display

  oled.setFont(Adafruit5x7);
if (disp[11] & 0x02)
 {

   oled.setCursor(103,2);
   oled.print("FUNC");

 }
 else
 {

   oled.setCursor(103,2);
   oled.print("    ");

 }

// Now check for clarifier bit
 if (disp[11] & 0x04)
 {

   oled.setCursor(103,1);
   oled.print("CLAR");

 }
 else
 {

   oled.setCursor(103,1);
   oled.print("    ");
//
 }



// check for "M" character
  if (disp[11] & 0x01)
 {

   oled.setCursor(103,3);
   oled.print("MEM");

 }
 else
 {
   oled.setCursor(103,3);
   oled.print("   ");

 }
  oled.setFont(lcdnums14x24);

i=0;

}
}





// Interrupt Service Routine for Chip Enable (CE)

void ENABLEHANDLER() {

if (digitalRead(CE))  {

    i=0;                //CE has gone high this is the beginning of the data sequence
    CEUPFLAG=1;
}
else
{
    CEUPFLAG=0;         //CE has gone low this is the end of the data sequence
}
}








// Interrupt Service Routine for STD falling edge

void  READNIBBLE()   {

bool  R40B = 0;
bool  R41B = 0;
bool  R42B = 0;
bool  R43B = 0;

if (CEUPFLAG) {         //CE is high so we are in the receiving data zone

// Falling edge of STD detected read a nibble into the array

R40B = digitalRead(R40);
R41B = digitalRead(R41);
R42B = digitalRead(R42);
R43B = digitalRead(R43);

// Now shift the bits into the characters

        disp[i] =     (0<<7)    |           // Since its only the last four bits that we are using
                      (0<<6)    |           // shift in 0011 to bits 7-4 as these are the standard
                      (1<<5)    |           // pattern for ASCII
                      (1<<4)    |           //
                      (R43B<<3) |           // Shift in the actual data from the FT290 processor
                      (R42B<<2) |           // Helpfully they used an ASCII-like scheme
                      (R41B<<1) |
                       R40B;

i++;
}

}
```