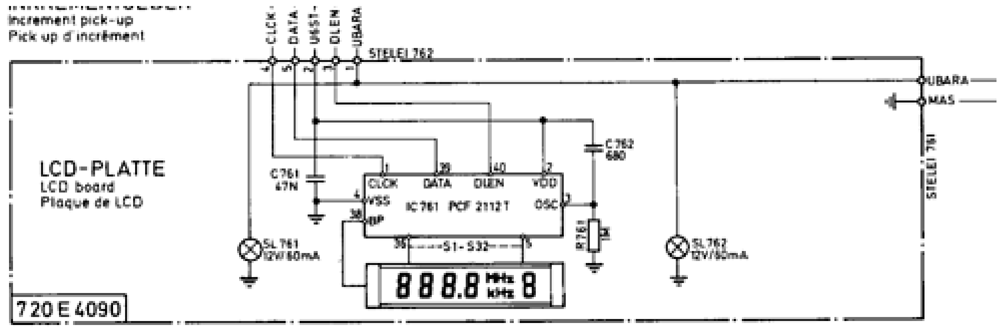
"C-Bus"? 7 segment LCD with Arduino

DaniTD 1 March 25, 2019, 7:58pm

Hello!

My name is Daniel and I'm a newcomer to this forum, and I think my first post will be a bit complicated.
I have done basic things with Arduino and now I wanted to salvage an old car stereo 7 segment LCD. I would like to use an Arduino to drive it. I thought it would be pretty straightforward but it seems to be a bit "old".



The LCD comes from a Bavaria Electronic II unit manufactured by BECKER. It has 5 characters, one decimal point and two custom indicators (Mhz and Khz). As you can see in the radio schematic extract I've attached, it is driven by a PCF2112T LCD driver, that relies on something called C-Bus. Problem is, I have no idea what C-Bus is, and I don't know if I would need a library to use it.

I'm not very used to use communication protocols with Arduino, but I'm open to all of your ideas to help me understand it and maybe get it to work.

In the datasheet I attached, and on the schematic too, it seems that it has one 6V pin, one ground, and DLEN, CLB and DATA. But I don't understand how to manage the 3 wires needed to communicate to the LCD.

Again, all help would be appreciated as I would like to learn as much as I can.

Thank you

PCF21XXC_FAM.pdf (254 KB)

DaniTD 2 March 25, 2019, 9:48pm

Well, I tried to connect SDA to DATA, SCL to CLB and 5v to Vdd. Result: some segments illuminate and stay static. I tried an I2C scanner sketch but no luck. Somewhere I read I2C can be connected to CBUS receivers? But I don't know how to wire the DLEN pin or how to connect to the IC via Arduino.

Regards

DaniTD 3 March 25, 2019, 10:02pm

As per the I2C bus specification from 1995:

"**9.1.3 CBUS compatibility**

CBUS receivers can be connected to the I2C-bus. However, a third bus line called DLEN must then be connected and the acknowledge bit omitted. Normally, I2C transmissions are sequences of 8-bit bytes; CBUS compatible devices have different formats.In a mixed bus structure, I2C-bus devices must not respond to the CBUS message. For this reason, a special CBUS address(0000001X) to which no I2C-bus compatible device will respond, has been reserved. After transmission of the CBUS address, the DLEN line can be made active and a CBUS-format transmission(Figure 20) sent. After the STOP condition, all devices are again ready to accept data.Master-transmitters can send CBUS formats after sending the CBUS address. The transmission is ended by a STOP condition, recognised by all devices.

NOTE: If the CBUS configuration is known, and expansion with CBUS compatible devices isn't foreseen, the designer is allowed to adapt the hold time to the specific requirements of the device(s) used."

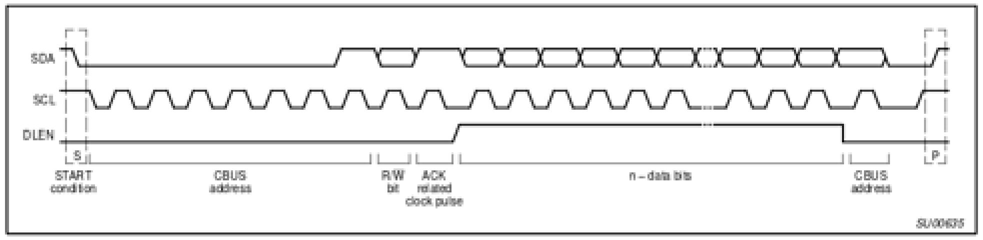


Figure 20. Data format of transmissions with CBUS transmitter/receiver

Rintin 4 March 26, 2019, 7:55am

I think you need to implement this

NXP Semiconductors **PCF21xxC family**

LCD drivers

7. Functional description

An LCD segment or LED output is activated when the corresponding DATA bit is HIGH; see Figure 7.

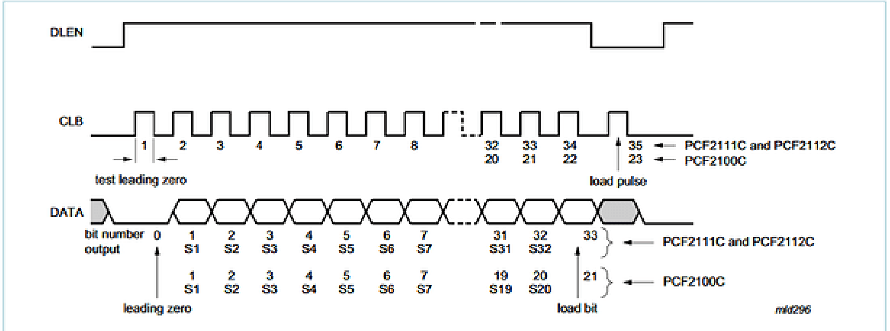


Fig 7. C-bus data format

Can be found in the datasheet you attached on page 6

DaniTD 5 March 26, 2019, 3:54pm

Hi Rintin, thank you for pointing that. I saw it before but I'm not sure how to implementate that, since I usually use libraries or some example code.
CLB should be the clock pin? Would SCL be adequate?
DATA should be connected to SDA pin? And DLEN to what? And as I can see, CBUS doesn't have an address for the device...
I think this is maybe too complicated for my current abilities with the Ardu, >:(

Regards

Rintin 6 March 26, 2019, 9:31pm

I wouldn't do it via i2c. Just with pinMode() and a bunch of digitalWrite().

Something like this:

```
#define PIN_DLEN 3
#define PIN_CLB  4
#define PIN_DATA 5

#define DATA_COUNT 4
byte displayData[DATA_COUNT];

void setup() {
  pinMode(PIN_DLEN, OUTPUT);
  digitalWrite(PIN_DLEN, LOW);
  pinMode(PIN_CLB , OUTPUT);
  digitalWrite(PIN_CLB, LOW);
  pinMode(PIN_DATA, OUTPUT);
  digitalWrite(PIN_DATA, LOW);
}

void shiftBit(byte value){
  digitalWrite(PIN_CLB, HIGH);
  digitalWrite(PIN_DATA, value);
  digitalWrite(PIN_CLB, LOW);
}

void shiftByte(byte value){
  byte tmp = value;
```

DaniTD 7 March 27, 2019, 10:49am

Thank you Rintin! I'll check that code and try to understand what it does, and then try to make it work.

What does the loop logic?

At least I tried to upload the code to the Arduino and the LCD responds, showing all kind of random segments.

Thank you for all your help.

Regards

Rintin 8 March 27, 2019, 6:05pm

Loop() does 2 things

1.

```
updateDisplay();
delay(30);
```

When I got the datasheet right, the controller needs a regular update to get the voltages for the LCD right.
updateDisplay() does this update by sending displayData to the display.
The delay() is for getting the timing right.

You should move the updateDisplay() to a timer interrupt and get rid of the delay().

2.

```
if(millis() - 1000 > last){
  last = millis();
  displayData[0] += 1;
  displayData[1] = displayData[0];
  displayData[2] = displayData[0];
  displayData[3] = displayData[0];
}
```

The timeing is a bit messed up by the delay() but this should change displayData every second.
The visual result should be the following:
You have 8 groups with 4 segments.
Every group is blinking.
The 1. group blinks every second.
The 2. group blinks every 2 seconds.
The 3. group blinks every 4 seconds.
The delay for the following groups is doubled.

DaniTD 9 March 27, 2019, 7:47pm

Great! First I did:

```
    last = millis();
    displayData[0] += 1;
    displayData[1] = 0;
    displayData[2] = 0;
    displayData[3] = 0;
```

Then I noticed it blinked 16 specific sets of segments, and then repeating again the same introducing another new segment for antoher 16 times and so... Making an array of 16x16 combinations of segments for displayData[0]. Tried with the others and got the same results, so combining the 4 Data sets I think you can make specific characters. I believe I have to map these combinations to know the values to show what I want on the LCD.

Seems it's progressing - thanks!

DaniTD 10 March 28, 2019, 7:51pm

Well, got it working! There are 4 groups of segments. I wrote in a paper which segments blinked on every group, from 0 to 255. That way, I got 3 16x16 matrix, and a 32*8 one. I numbered the elements of each one from 0 to 255.

To make a the first digit cames from the 32x18 matrix, the second one from combining this matrix with one 16x16, the third combining two 16x16... After "cleaning" the matrices from some combinations that gave "useless" characters, I defined the 4 matrices on the code, and made a little function that reads the number you want to display, and then filters it digit by digit through the matrices to display the correct number, and voila! Now it works!

Thank you very much Rintin for your help 🙂

Regards

Rintin 11 March 28, 2019, 8:18pm

Sounds complicated.

Maybe you want to attach something for your "future-me"... or that other poor soul with the same display who stumbles over this thread in a few years.

DaniTD 12 April 2, 2019, 2:31pm

I'll make a simplified example to explain how it works and upload it, and maybe that way it can be understandable.

Regards

DaniTD 13 April 2, 2019, 8:50pm

Rintin, for completeness sake and to try to understand how this works seeing the differences, how would you modify the code to make it work on two PCF2111 as specified on that datasheet, one as master and another as slave?

Regards

> Rintin:
> I wouldn't do it via i2c. Just with pinMode() and a bunch of digitalWrite().
>
> Something like this:
>
> ```
> #define PIN_DLEN 3
> ```

```
#define PIN_CLB  4
#define PIN_DATA 5

#define DATA_COUNT 4
byte displayData[DATA_COUNT];

void setup() {
  pinMode(PIN_DLEN, OUTPUT);
  digitalWrite(PIN_DLEN, LOW);
  pinMode(PIN_CLB , OUTPUT);
  digitalWrite(PIN_CLB, LOW);
  pinMode(PIN_DATA, OUTPUT);
  digitalWrite(PIN_DATA, LOW);
}

void shiftBit(byte value){
  digitalWrite(PIN_CLB, HIGH);
  digitalWrite(PIN_DATA, value);
  digitalWrite(PIN_CLB, LOW);
}

void shiftByte(byte value){
  byte tmp = value;
  for(byte c=0; c<8; c++){
    shiftBit(tmp & 1);
    tmp = tmp >> 1;
  }
}

void updateDisplay(){
  digitalWrite(PIN_DLEN, HIGH);
```

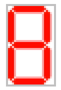
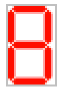
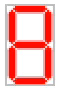
```
  // leading zero
  shiftBit(0);

  for(byte c=0; c<DATA_COUNT; c++){
    shiftByte(displayData[c]);
  }

  // load bit
  shiftBit(1);

  digitalWrite(PIN_DLEN, LOW);

// load pulse
  shiftBit(0);

}

unsigned long last = 0;

void loop() {
  if(millis() - 1000 > last){
    last = millis();
    displayData[0] += 1;
    displayData[1] = displayData[0];
    displayData[2] = displayData[0];
    displayData[3] = displayData[0];
  }

  updateDisplay();
  delay(30);
}
```

DaniTD 14 April 3, 2019, 10:39am

> DaniTD:
> I'll make a simplified example to explain how it works and upload it, and maybe that way it can be understandable.
>
> Regards

As a very simple example for the code shared by Rintin to control a PCF2112 series LCD driver, I think this should help to get the idea (every LCD can be wired on a different way, so it's trial and error):

Let's suppose we have a 3 digit, 7 segment LCD already embedded with this driver IC. For reference, we call each digit:

1st char.	2nd char.	3rd char.
		

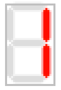
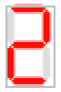
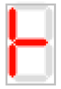
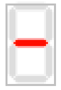
We don't know what numbers/letters it can display, so we upload the following code (before, we should declare outside the void displayData[0] = 0):

```
void loop() {
  if(millis() - 1000 > last){
    last = millis();
    displayData[1] = 0;
    displayData[2] = 0;
    displayData[3] = 0;
  }

  updateDisplay();
  delay(30);
  displayData[0] = displayData[0] + 1;
}
```

Varying displayData[0] value we will start see the LCD flash different segments in the 2nd and 3rd character. We should take note of the segment that illuminates on each sequence and give it a value from 0 to the number of combinations shown (they can go up to 255). Let's imagine we get the following sequence (as I said, very simplified):

3rd digit

2nd digit

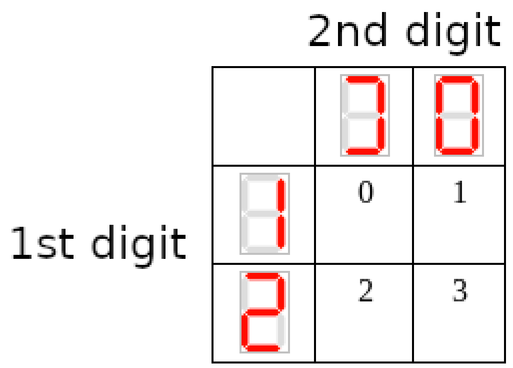
		
	0	1
	2	3

The values from 0 to 3 shown on the table would be the displayData[0] value and at first row and first column, the segments illuminated on 3rd and 2nd digit.

Now we do the same, but with displayData[1]:

```
void loop() {
  if(millis() - 1000 > last){
    last = millis();
    displayData[0] = 0;
    displayData[2] = 0;
    displayData[3] = 0;
  }

  updateDisplay();
  delay(30);
  displayData[1] = displayData[1] + 1;
}
```

Getting the following combinations:



Now, we should figure the possible combinations to give readable characters or numbers, and then creating a function that asks for the number you want to show, so it can select the proper displayData values.

As you can see this example is very limited, but proves that one character can depend on two displayData values. So, for example, show the number "132" you should use the values:

```
    displayData[0] = 0;
    displayData[1] = 3;
```

So displayData[1] draws the "1" in the first character and the outer segments of the second character, and displayData[0] draws the middle section of the second character and the "2" on the third.

Having complete tables from 0 to 255 values you should get all the possible combinations for numbers from 0 to 9, and some letters. Then, using an "if" statement, you could select which values to assign to the different displayData values to get the correct number.

Hope this helps to understand how to interface with this IC (if I've managed to not to mess it even more, I will be satisfied 🙂)

Regards

Rintin 15 April 3, 2019, 7:46pm

> DaniTD:
> Rintin, for completeness sake and to try to understand how this works seeing the differences, how would you

> modify the code to make it work on two PCF2111 as specified on that datasheet, one as master and another as slave?
>
> Regards

The 2111 support 2 backplanes (instead of one). You need to store the extra data for the segments (byte displayData[DATA_COUNT] -> byte displayData[2][DATA_COUNT])
Also you need to keep track of the active backplane and toggle between both (-> activeBP).

This should do it:

```
#define PIN_DLEN 3
#define PIN_CLB  4
#define PIN_DATA 5

#define DATA_COUNT 4
byte displayData[2][DATA_COUNT] = {{0,0,0,0},{0,0,0,0}};

void setup() {
  pinMode(PIN_DLEN, OUTPUT);
  digitalWrite(PIN_DLEN, LOW);
  pinMode(PIN_CLB , OUTPUT);
  digitalWrite(PIN_CLB, LOW);
  pinMode(PIN_DATA, OUTPUT);
  digitalWrite(PIN_DATA, LOW);
}

void shiftBit(byte value){
  digitalWrite(PIN_CLB, HIGH);
  digitalWrite(PIN_DATA, value);
  digitalWrite(PIN_CLB, LOW);
}

void shiftByte(byte value){
  byte tmp = value;
```

Rintin 16 April 3, 2019, 8:12pm

> DaniTD:
> As a very simple example for the code shared by Rintin to control a PCF2112 series LCD driver, I think this should help to get the idea (every LCD can be wired on a different way, so it's trial and error):

For some reason I expect that one bit controls only one segment. Can you check that the following values light up only one segment (and always the same)?
1
2
4
8
16
32
64
128

system Closed 17 May 6, 2021, 4:00am