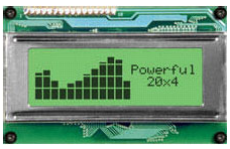


ARMexpress drives Serial LCD

Dr. Claus Kühnel

ARMexpress supports several serial interfaces. Using serial interfaces to peripheral devices helps in cases of low pin count microcontrollers. This article describes writing data to a LCD using the I²C interface.

1. I²C Bus

The I²C bus was developed for data exchange between different devices, as EEPROMs, RAMs, AD and DA converts, RTCs and microcontrollers in a networked environment.

Figure 1 shows all required connections in a typical I²C bus network. The lines SDA and SCL connect all members of the network. PullUp resistors connect these lines to the supply voltage V_{CC} and guarantees the Hi level.

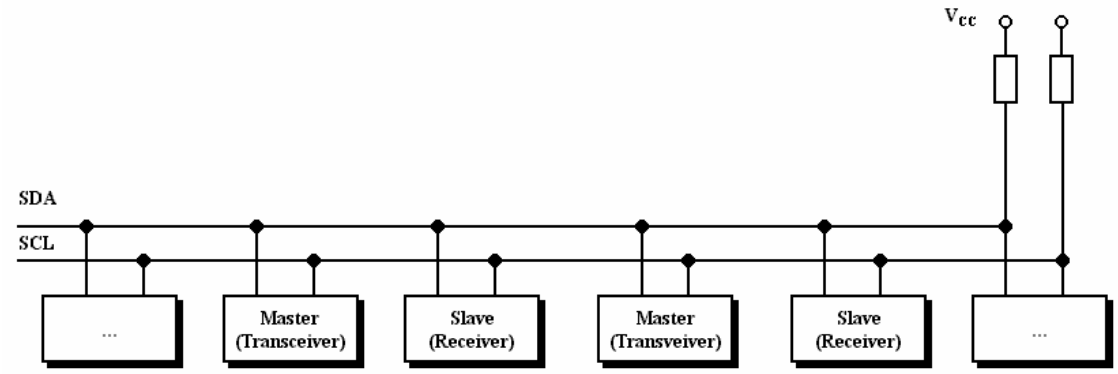
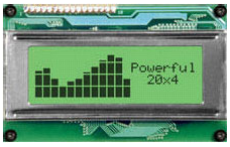


Figure 1 I²C Bus Network

In an I²C bus network several masters can be connected with several slaves (Multi-Master System). The I²C bus protocol addresses the members of the network.

The peripheral functions depend on the specific device. Beside EEPROMs and RAMs from numerous manufacturers there are a lot of further I²C bus devices, as I/O expander and LCD and LED driver devices etc.



1.1. Some Basics

For an interface built by such simple hardware the software protocol must secure the communication. Figure 2 shows the protocol for writing and reading of one byte.

There are further functions, as sequential write and read. We will concentrate us to the basic functions only.

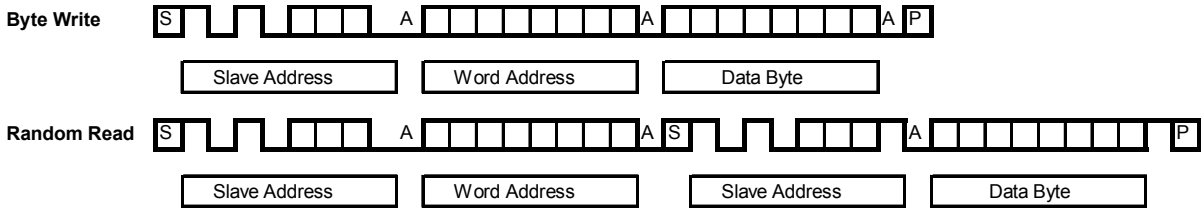


Figure 2 Writing and Reading of one Byte

For data exchange between I²C bus devices the master starts data exchange with a start condition (S) followed by an address byte containing the slave address. The structure of the slave address byte for a PCF8591 device as example is

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	1	A2	A1	A0	R//W

The bits Bit7-B4 characterize the I²C devices. The bits B3 to B1 are address selectors (sub addresses) and Bit0 defines read (B0=1) or write (B0=0) operation.

Due to this addressing structure it is possible to connect eight PCF8591 devices in one I²C bus network beside other I²C bus devices.

The individual byte sent by the master will be acknowledged (A) from the addressed I²C bus device to the master signalizing an error-free data exchange.

After the slave address a word address follows. This word address points to a register of the addressed I²C bus device. The data byte will be saved into this register or will be read from this register. The data exchange will be finished by a stop condition (P).

The start and stop conditions mentioned a characterized by a certain phase layer of the signals SCL and SDA. Figure 3 explains the details.

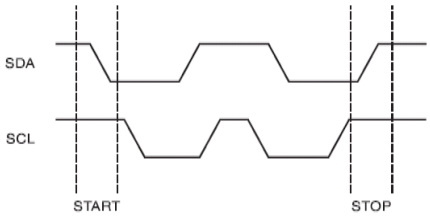
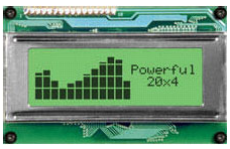


Figure 3 Start & Stop Condition



1.2. ARMexpress I²C Support

It is possible to organize data exchange between I²C bus devices by bit-banging the SCL and SDA lines. For BASIC Stamp microcontrollers (BS1 and BS2) we had examples we had examples in [1] and [2].

This way is possible for ARMexpress too. But, there are special instructions that hide all the details as start and stop condition, the acknowledge bit, and changing data direction of SDA line for the programmer.

For a write access to an I²C bus device we can use the instruction I2COUT:

I2COUT *Data pin, slaveADDR,* [OutputList]

Data pin defines the I/O pin used for SDA line. For SCL we have to use the I/O pin *Data pin+1* definitely. The slave address selects the I²C bus device for the following read or write access.

For a read access to an I²C bus device we can use the instruction I2CIN:

I2CIN *Data pin, slaveADDR,* [opt1, [... opt5,]] [InputList]

Before reading data from accessed I2C bus device into the InputList it is possible to send out up to five optional byte values.

The I2C bus is byte oriented, so each transaction will either send a byte value (0 to 255) or receive a byte to/from an I2C bus device. If a value in the *OutputList* is larger than 8 bits, the MSBs will be truncated.

Data is shifted out and in at 380 kbits/sec.

The program sample describing the use of an I²C bus driven LCD shows the use of instruction I2COUT in detail.

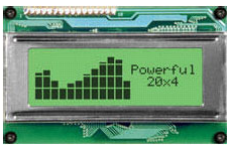
2. Character Display LCD2041

Matrix Orbital as manufacturer of the LCD2041 designed it as a display unit for an associated controller.

The LCD2041 provides a simple command structure to allow text and bar graphs to be displayed on the screen. Text fonts are built in, and use standard ASCII mapping. Provision is made for up to 8 user-defined characters.

The screen is backlit for low-light situations. Backlighting may be turned on or off under program control. Contrast is adjustable to compensate for differing lighting conditions and viewing angles.

A general-purpose output allows the controller to switch an electronic or electro-mechanical device by issuing commands to the display unit. This can be used for controlling LEDs, relays, etc.



The LCD2041 supports RS-232 and I^2C bus interface. Due to the possibility of networking, we use here the I^2C bus interface. Up to 16 LCD2041 modules can be connected on the same I^2C bus interface.

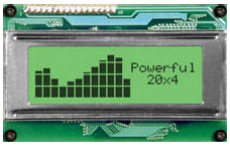
The slave address can be configured by jumpers J4-J1 on the LCD2041 module according to Table 1.

Slave Address	*J4*	*J3*	*J2*	*J1*
$50	Out	Out	Out	Out
$52	Out	Out	Out	In
$54	Out	Out	In	Out
$56	Out	Out	In	In
$58	Out	In	Out	Out
$5A	Out	In	Out	In
$5C	Out	In	In	Out
$5E	Out	In	In	In
$60	In	Out	Out	Out
$62	In	Out	Out	In
$64	In	Out	In	Out
$66	In	Out	In	In
$68	In	In	Out	Out
$6A	In	In	Out	In
$6C	In	In	In	Out
$6E	In	In	In	In

Table 1 Configuration of Slave Address

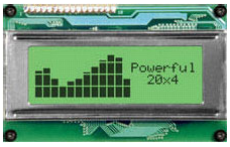
In Table 1 the configuration used in the program sample following is coloured. Table 2 shows the commands for programming the LCD2041.

COMMAND	*BYTES*	*NOTE*
Auto line wrap on	254 67	Enables line wrapping (not word wrap).
Auto line wrap off	254 68	Disables line wrapping.
Auto scroll on	254 81	Enables scroll at bottom of screen. Text will push display up one line to make room for new line.
Auto scroll off	254 82	Disables auto scroll. Text will wrap to top left and overwrite existing text.
Set cursor position	254 71 [col] [row]	Moves cursor to the specified column and row. The cursor marks the text insertion point in this and all commands.
Send cursor home	254 82	This command moves the cursor to the top left of the display area.
Underline cursor on	254 74	Turns on the underline cursor.
Underline cursor off	254 75	Turns off the underline cursor.
Block cursor on	254 83	Turns on the blinking block cursor.
Block cursor off	254 84	Turns off the blinking block cursor.
Cursor left	254 76	Moves the cursor one position to the left. If the cursor is already at the beginning of a line it will move to the end of the other line.



COMMAND	BYTES	NOTE
Cursor right	254 77	Moves the cursor one position to the right. If the cursor is already at the end of a line it will move to the beginning of the other line.
Initialize thick vertical bar graph	254 118	Initializes the user character set to make wide vertical bar graphs.
Initialize thin vertical bar graph	254 115	Initializes the user character set to make narrow vertical bar graphs.
Initialize horizontal bar graph	254 104	Initializes the user character set to make horizontal bar graphs.
Draw vertical bar graph	254 61 [col][length]	Draws a vertical bar graph at column [col] of length [length]. Length is measured in pixels (0x00 to 0x20). User must first use the 'v' or 's' command to initialize characters.
Draw horizontal bar graph	254 124 [c][r][d][length]	Draws a horizontal bar graph starting at column [c] on row [r] with direction [d] (0 is right, 1 is left) of length [length]. Length is measured in pixels (0x00 to 0x64 if starting in column 1). User must first use the 'h' command to initialize characters.
Initialize large digits	254 110	Initializes the user character set to make large digits.
Place large digits	254 35 [col] [digit]	Place large digit number [digit] in column [col] of the display. Cursor moves to bottom right of large digit. [digit] is 0x00 to 0x09, [col] is 0x01 to 0x12 (i.e. 1 to 18 decimal).
Define custom character	254 78 [c][8 bytes]	Defines one of 8 custom "user" characters. Character number is [c] between 0x00 and 0x07. The 8 bytes are described in section 4.1.8.
Clear display	254 88	Clears screen of text and graphics, places text cursor at top left.
Set contrast	254 80 [contrast]	Sets display contrast. Compensates for viewing angle. Contrast is a value between 0 and 255 (hex 0 to FF). Larger = darker.
Backlight on	254 66 [minutes]	Backlight will stay on for [minutes]. If [minutes]= 0 backlight will stay on permanently.
Backlight off	254 70	Turns off backlight.
Set Brightness	254 153 [brightness]	Sets display brightness. Brightness is a value between 0 and 255 (hex 0 to FF). Larger = brighter.
General purpose output off	254 86	Turns the general purpose output OFF.
General purpose output on	254 87	Turns the general purpose output ON.

Table 2 LCD2041 Commands



3. Program Sample I2C_LCD.BAS

The program sample listed in the next chapter explains using the I2COUT instruction for writing data to an I^2C driven text LCD. LCD2041 from Matrix Orbital was used. For detailed description of LCD2041 read the LCD2041 user manual.

The program samples shows some features of the LCD2041 as

- Output of text messages
- Output of big digits in combination with normal text
- Speedometer display
- Wide Vertical Bargraph
- Horizontal Bargraph

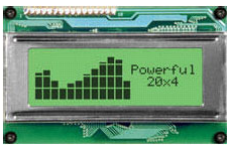

Medienclip

Double-click the media clip button on the right and you can see a short video clip of the display output. The video clips shows an ARMexpress module placed on an old BS2p demo board and the connected LCD2041 with its changing display content.

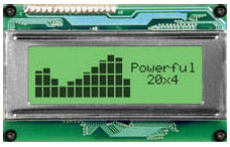
I think there are no further hints needed. Have a look to the source code of I2C_LCD.BAS and refer the commands and its parameters to Table 2.

4. Source code

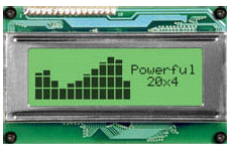
```
' -----[ Title ]-------------------------------------------------------
'
' File......  i2c_lcd.bas
' Purpose...  Character Output on LCD2041 by Matrix Orbital
' Author....  Claus Kuhnel
' Started...  2006-08-12
' Updated...
'
' -----[ Program Description ]-----------------------------------------
'
' This program explains using the I2COUT instruction for writing data to
' an I2C driven text LCD. LCD2041 from Matrix Orbital was used.
' For detailed description of LCD2041 read the LCD2041 user manual.
' Look at www.matrixorbital.ca/manuals/LCDVFD_series/LCD2041/
'
' -----[ Revision History ]---------------------------------------------
'
' -----[ CONSTants ]----------------------------------------------------
'
CONST wraddrLCD = $5C

CONST Instr = 254
CONST Autoscrollon = 81
CONST Autoscrolloff = 82
CONST Setcursor = 71                    ' [x] [y]
CONST Cursorhome = 72
```




```
CONST Cleardisplay = 88
CONST Initbigdigit = 110
CONST Placebigdigit = 35                ' [x] [digit]
CONST Initwvbargraph = 118
CONST Drawvbargraph = 61                ' [x] [height]
CONST Inithbargraph = 104
CONST Drawhbargraph = 124               ' [x] [y] [dir] [length]


CONST DataPin = 0


' -----[ Variables ]-------------------------------------------------
'
DIM txt$ (21)

DIM txt1$ (21)
DIM txt2$ (21)
DIM txt3$ (21)
DIM txt4$ (21)

' -----[ Initialization ]--------------------------------------------
'
wait (1000)

txt1$ = "ARMexpress made by"
txt2$ = "Coridium writes text"
txt3$ = "by I2C to LCD2041."
txt4$ = "www.coridiumcorp.com"

' -----[ Main Code ]-------------------------------------------------
'
print "Character Output on LCD2041 by Matrix Orbital"
do
      print "Textoutput"
      gosub TextOutput
      print "Big Digits"
      gosub BigDigit
      print "Tacho"
      gosub Tacho
      print "Wide Vertical Bargraph"
      gosub WideVBargraph
      print "Horizontal Bargraph"
      gosub HBargraph
loop

' -----[ Subroutines ]-----------------------------------------------
'
TextOutput:
      for i = 1 to 2
            I2COUT DataPin, wraddrLCD, [Instr, Cleardisplay]
            wait (500)
            tl = len(txt1$)
            I2COUT DataPin, wraddrLCD, [txt1$ \tl]
            I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 1, 2]
            tl = len(txt2$)
            I2COUT DataPin, wraddrLCD, [txt2$ \tl]
            I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 1, 3]
            tl = len(txt3$)
            I2COUT DataPin, wraddrLCD, [txt3$ \tl]
            I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 1, 4]
            tl = len(txt4$)
            I2COUT DataPin, wraddrLCD, [txt4$ \tl]
            wait (1000)
```




```
	next i
	return

BigDigit:
	pos0 = 2
	I2COUT DataPin, wraddrLCD, [Instr, Cleardisplay]
	I2COUT DataPin, wraddrLCD, [Instr, Initbigdigit]
	for i = 0 to 4
		pos = pos0
		Char = i
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		pos = pos+3
		char = i+1
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		pos = pos+3
		char = i+2
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		pos = pos+3
		char = i+3
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		pos = pos+3
		char = i+4
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		pos = pos+3
		char = i+5
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		wait (1000)
	next i
	return

Tacho:
	I2COUT DataPin, wraddrLCD, [Instr, Cleardisplay]
	I2COUT DataPin, wraddrLCD, [Instr, Initbigdigit]
	for i=0 to 111
		speed = i                  ' speed between 0 and 999
		sh = speed / 100
		st = speed / 10
		st = st - (sh * 10)
		ss = speed mod 10
		if (sh > 0) then
			pos = 5
			char = sh
			I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		endif
		if (sh > 0) OR (st > 0) then
			pos = 8
			char = st
			I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		endif
		pos = 11
		char = ss
		I2COUT DataPin, wraddrLCD, [Instr, Placebigdigit, pos , char]
		I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 16, 3]
		txt$ = "mph"
		tl = len(txt$)
		I2COUT DataPin, wraddrLCD, [txt$ \tl]
		I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 20, 4]
		wait (100)
	next i
	wait (1000)
	return

WideVBargraph:
	I2COUT DataPin, wraddrLCD, [Instr, Cleardisplay]
```



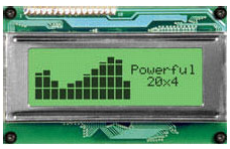

```
       I2COUT DataPin, wraddrLCD, [Instr, Initwvbargraph]
       For I = 1 To 20
       J = I * 32
       J = J / 20
       I2COUT DataPin, wraddrLCD, [Instr, Drawvbargraph , i, j]
       wait (10)
       Next i
       I2COUT DataPin, wraddrLCD, [Instr, Cursorhome]
       txt$ = "Bargraph"
       tl = len(txt$)
       I2COUT DataPin, wraddrLCD, [txt$ \tl]
       wait (3000)
       return

HBargraph:
       I2COUT DataPin, wraddrLCD, [Instr, Cleardisplay]
       I2COUT DataPin, wraddrLCD, [Instr, Inithbargraph]
       I2COUT DataPin, wraddrLCD, [Instr, Drawhbargraph, 1, 1, 0, 100]
       I2COUT DataPin, wraddrLCD, [Instr, Drawhbargraph, 1, 2, 0, 55]
       I2COUT DataPin, wraddrLCD, [Instr, Drawhbargraph, 1, 3, 0, 27]
       I2COUT DataPin, wraddrLCD, [Instr, Drawhbargraph, 1, 4, 0, 13]
       I2COUT DataPin, wraddrLCD, [Instr, Setcursor, 12, 4]
       txt$ = "Bargraph"
       tl = len(txt$)
       I2COUT DataPin, wraddrLCD, [txt$ \tl]
       wait (3000)
       return

' -----[ Data ]-----------------------------------------------------
'
```

Listing LCD Demo (I2C_LCD.BAS)

The source code listed here can be downloaded from ARMexpress User Group or author's homepage.

5. Links

Manuals for LCD2041	http://www.matrixorbital.ca/manuals/LCDVFD_series/LCD2041/
Coridium Homepage	http://www.coridiumcorp.com
ARMexpress User Group	http://groups.yahoo.com/group/ARMexpress
Author's Homepage	http://www.ckuehnel.ch



6. Literature

[1] Kühnel, C.; Zahnert, K.:
BASIC Stamp 2p
Commands, Features and Projects
Parallax: Rocklin (CA), 2003
ISBN 1-928982-19-0

[2] Kühnel, C.; Zahnert, K.:
BASIC Stamp 2nd Ed.
Newnes: Boston et al, 2000
ISBN 0-7506-7245-5