GreenPAK LCD Driver frees microcontroller pins

Author: Bilal Ahmed

Date: November 7, 2016

Introduction

In this application a GreenPAK SLG46620V is placed between the serial interface of an Arduino Mega 2560, and an LCD 2004A display. This approach now frees up 8 microcontroller pins for other uses in the design. Additionally, this circuit can be adapted to other LCD interfaces with minor changes in LCD initialization through the microcontroller.

Hardware Schematic

The top level block diagram schematic is shown in Fig. 1.

The Arduino mega 2560 TX1 pin is connected to SLG 46620V Pin 10. RW is grounded to use the LCD2004A for write only mode. The brightness of LCD can be adjusted through potentiometer. The connections between SLG46620V and LCD 2004A are tabulated below

Note that GreenPAK SLG46620V needs to be switched ON before the LCD display (2004A) and microcontroller (Arduino mega 2560). Otherwise the LCD 2004A display may not function properly.

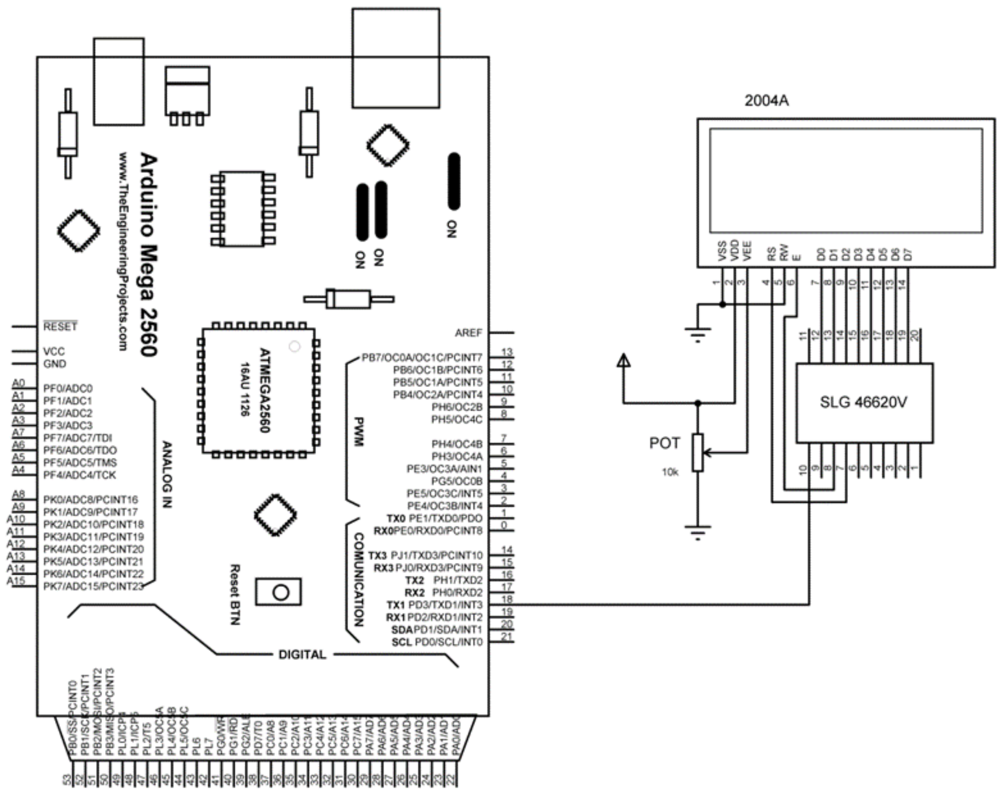


Figure 1. Hardware Schematic

Sr. No	SLG 46620V PIN	LCD 2004A PIN
1	7	RS
2	8	E
3	12	D0
4	13	D1
5	14	D2
6	15	D3
7	16	D4
8	17	D5
9	18	D6
10	19	D7

Table 1. LCD 2004A connection with SLG 46620V

GreenPAK Design Configuration

The serial data written on the TX1 pin of Arduino is received in the GreenPAK SLG466620V PIN 10, the data is sent to the SPI block MOSI input.

The serial data (10 bits total, one start bit, 8 data bits and one stop bit) that reaches the SPI block is converted to parallel bits. Set the SCLK of the SPI block so that it matches the 9600bps baudrate. SCLK needs to be started when the serial data is transmitted to GreenPAK. When no data is received there is a continuous high signal at the SPI.

Detecting the start of data packet

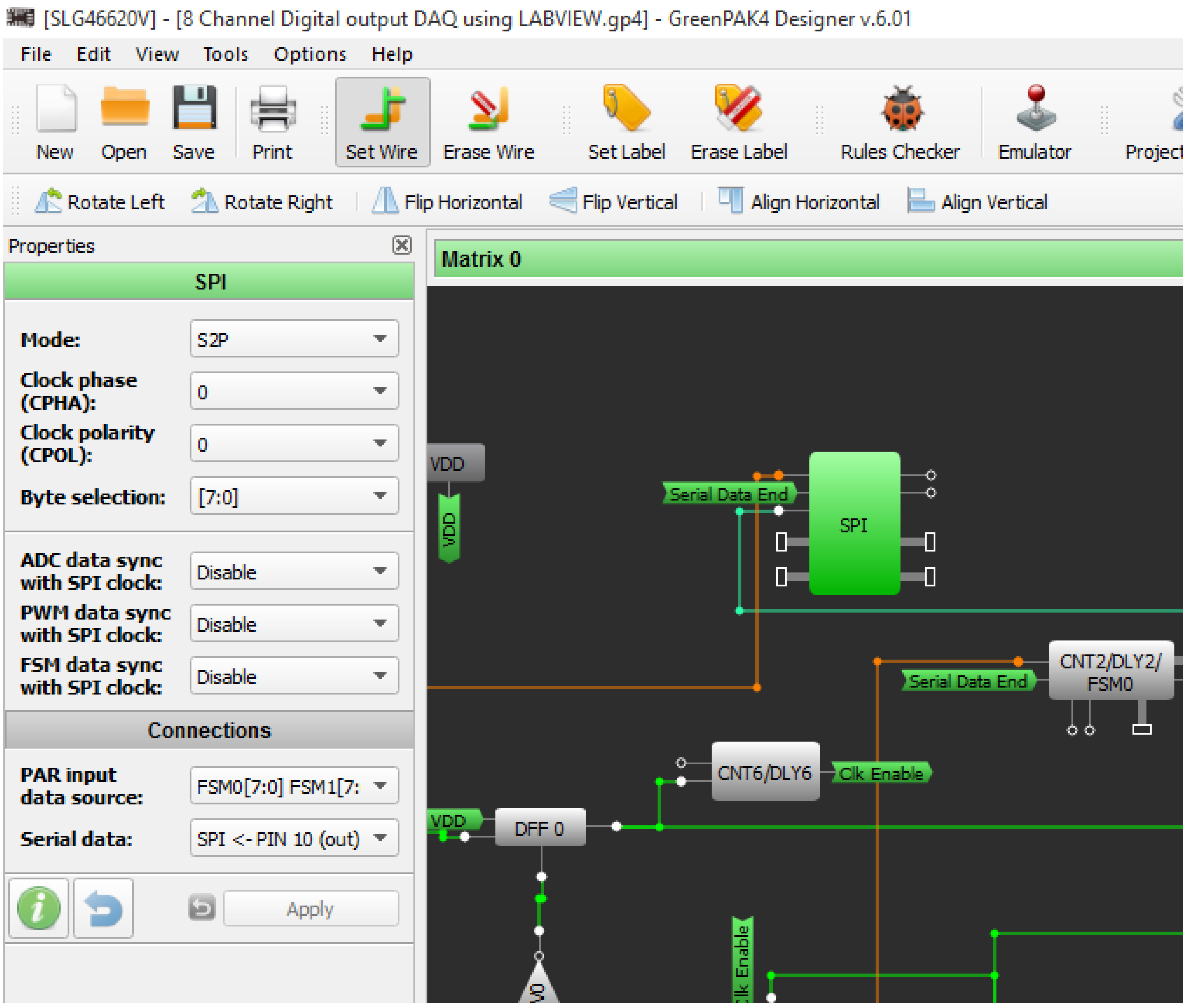
Whenever the data is sent from Arduino, the start bit, which is at the start of the packet is a low bit, arrives first.

A new packet arriving is identified by detecting the falling edge of the incoming data at PIN 10.

P DLY0 block is configured to detect the Falling edge of the input. Whenever the data packet arrives the P DLY0 block sets its output high for one cycle which then set the output of DFF 0 high.

The output of DFF 0 is reset after a time of 924 microseconds (approximately 10cycles of 9600bps).

The Serial Data End signal low enables the SPI block nCSB input. After the 9th cycle of incoming data the signal goes high and SPI block is disabled.


[SLG46620V] - [8 Channel Digital output DAQ using LABVIEW.gp4] - GreenPAK4 Designer v.6.01
File Edit View Tools Options Help
New Open Save Print Set Wire Erase Wire Set Label Erase Label Rules Checker Emulator Project
Rotate Left Rotate Right Flip Horizontal Flip Vertical Align Horizontal Align Vertical
Properties
Matrix 0
VDD
Serial Data End
SPI
CNT2/DLY2/ FSM0
CNT6/DLY6
Clk Enable
DFF 0
Apply

SPI

Mode: S2P
Clock phase (CPHA): 0
Clock polarity (CPOL): 0
Byte selection: [7:0]
ADC data sync with SPI clock: Disable
PWM data sync with SPI clock: Disable
FSM data sync with SPI clock: Disable

Connections

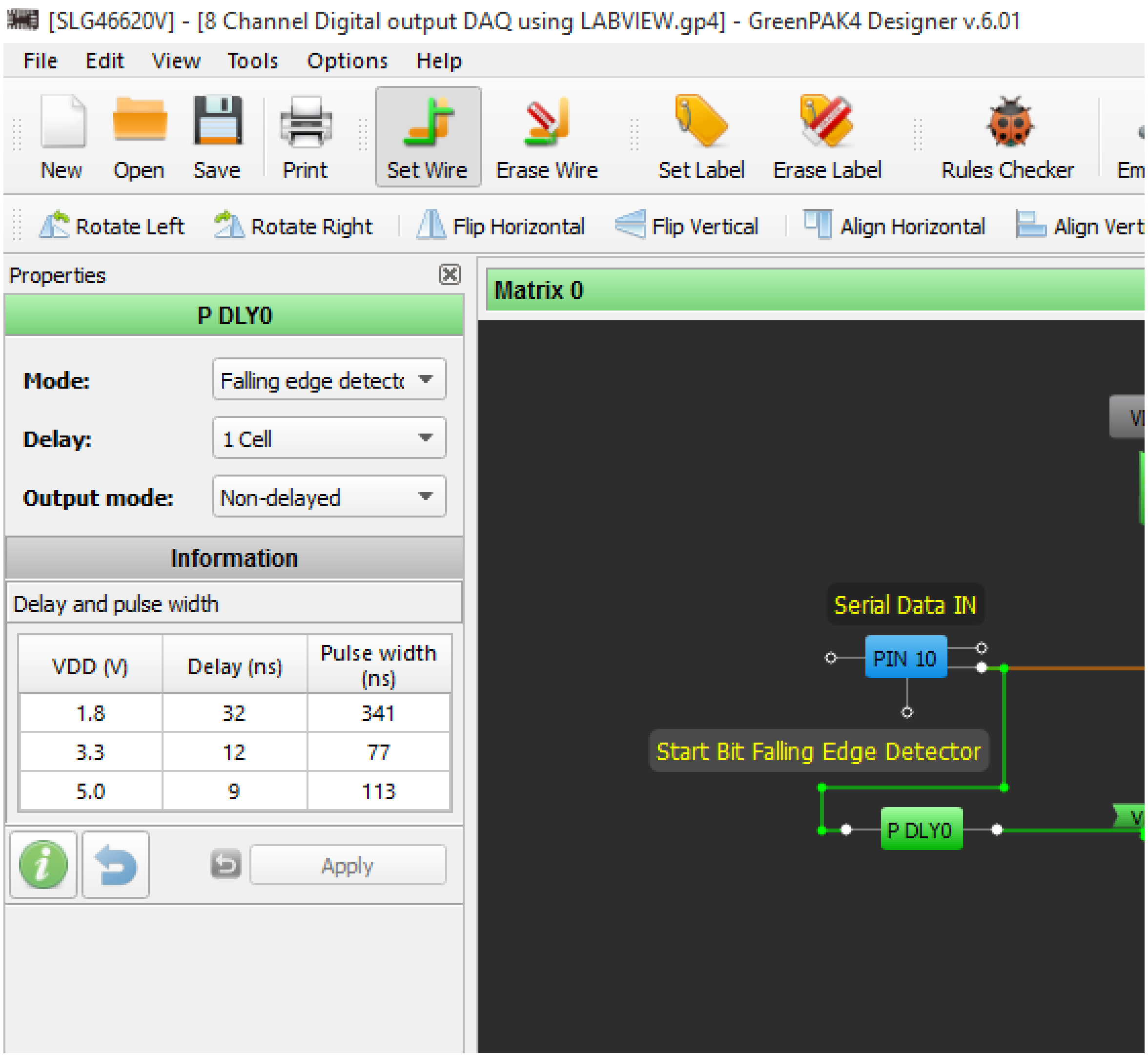
PAR input data source: FSM0[7:0] FSM1[7:
Serial data: SPI <- PIN 10 (out)

Figure 2. Serial Data received in GreenPAK

Loading serial data in the SPI block

The latch (DFF 0) high output is also used to load serial data at the MOSI input of the SPI block. For storing serial data near the center of SCLK, it is necessary to input a delay of half cycle of 9600 bps for SCLK. The time corresponds to half cycle of 9600 bps is approximately 52 microseconds. This time delay is achieved by passing the output of DFF 0 through CNT6/DLY6.

The Clock Enable signal makes sure that counter CNT2/DLY2 won't start counting until the start bit arrives. The CNT2/DLY2 output then clocks signals for the SPI block (SCLK). It helps load the serial data to SPI block, and once the 10 bit cycle completes, the SPI block converts the serial data to the SPI parallel output block ports.


[SLG46620V] - [8 Channel Digital output DAQ using LABVIEW.gp4] - GreenPAK4 Designer v.6.01

P DLY0

Mode: Falling edge detect

Delay: 1 Cell

Output mode: Non-delayed

Information

Delay and pulse width

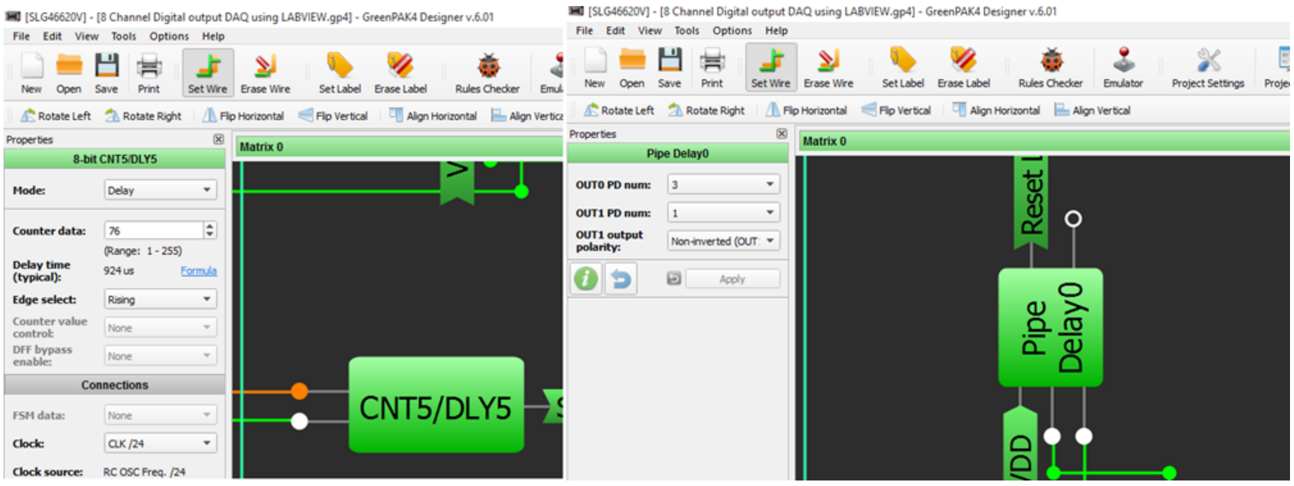
VDD (V)	Delay (ns)	Pulse width (ns)
1.8	32	341
3.3	12	77
5.0	9	113

Figure 3. P DLY0 block Properties

Data at output pins

The serial data is presented at SPI Parallel output block pins. It then appears on the output pins after passing through Latches.

Since there are two bytes of data arriving from Arduino, we need to parse the data to send it to different GreenPAK pins. For doing this, a mod 2 synchronous counter is used whose output toggles for every cycle (at start bit) of the serial data. The first packet of serial data at the parallel pins of SPI parallel output block appears on PIN 7 (RS) and PIN 8 (E). The second packet of serial data appears on PIN12, 13…19 (D0, D1…D8). The data from the next incoming cycles will update on these pins after every two cycles of serial data.


[SLG46620V] - [8 Channel Digital output DAQ using LABVIEW.gp4] - GreenPAK4 Designer v.6.01

Properties — 8-bit CNT5/DLY5

Mode: Delay
Counter data: 76
(Range: 1 - 255)
Delay time (typical): 924 us
Edge select: Rising
Counter value control: None
DFF bypass enable: None

Connections

FSM data: None
Clock: CLK /24
Clock source: RC OSC Freq. /24

Properties — Pipe Delay0

OUT0 PD num: 3
OUT1 PD num: 1
OUT1 output polarity: Non-inverted (OUT

Figure 4. CNT5/DLY5 and Pipe Delay 0 properties

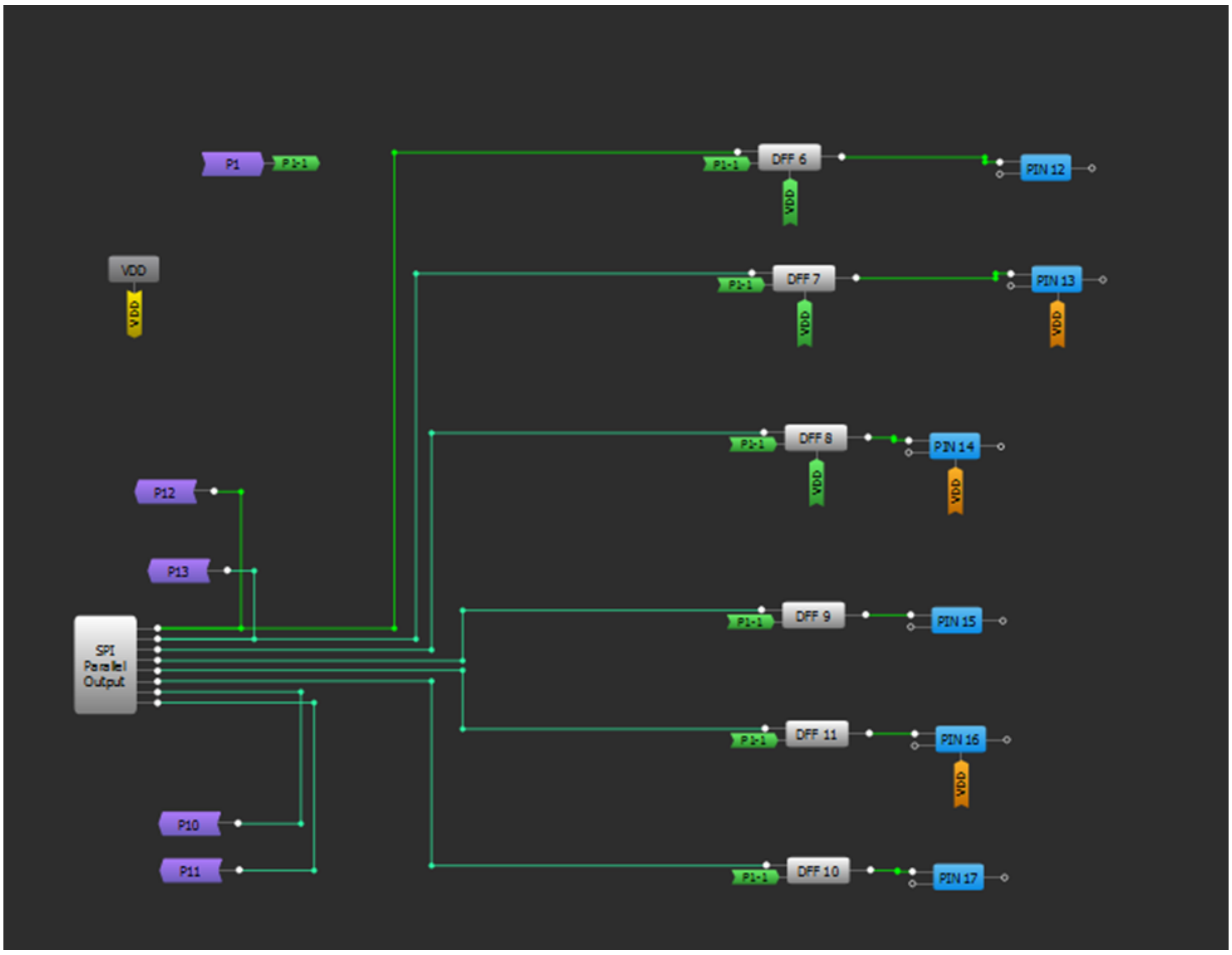


Figure 5. SPI Parallel Output Block Pins

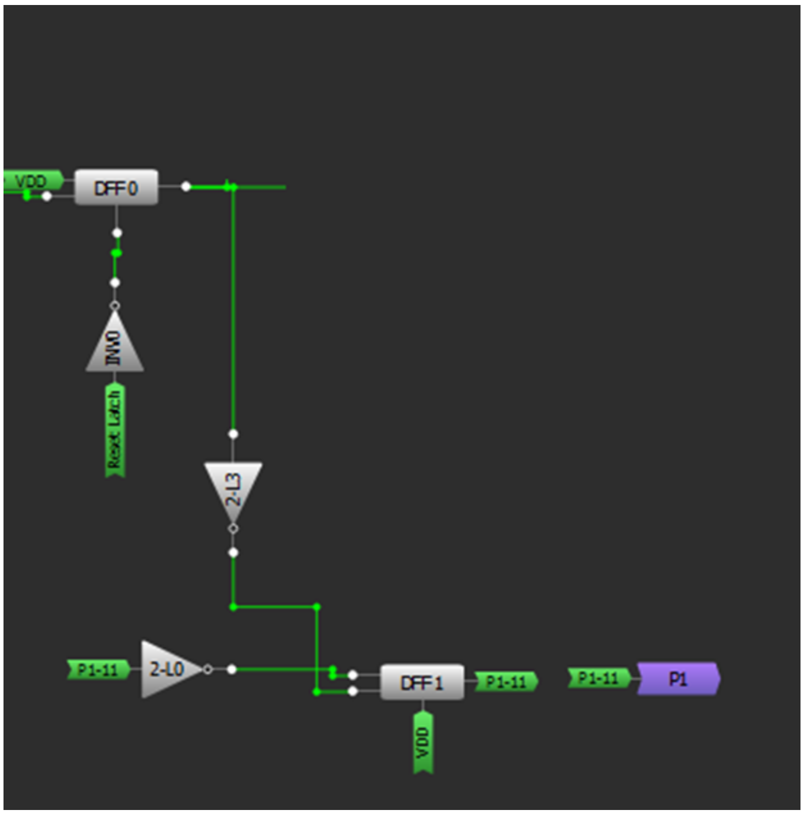


Figure 6. Mod 2 Counter for data Parsing

LCD Module LCD2004A

LCD2004A has 16 pins. This is a basic 20 character by 4 line display. It utilizes the HD44780 parallel interface chipset.

The 10 connections with GreenPAK SLG46620V are listed in Table 2. The other connections for LCD are listed below

Sr. No	LCD 2004A PIN	SLG 46620V PIN
1	VSS	Ground
2	VDD	5V
3	RW	Ground
4	K	Ground
5	A	5V
6	VO/VEE	Pot middle pin

Table 2. LCD 2004A Connections

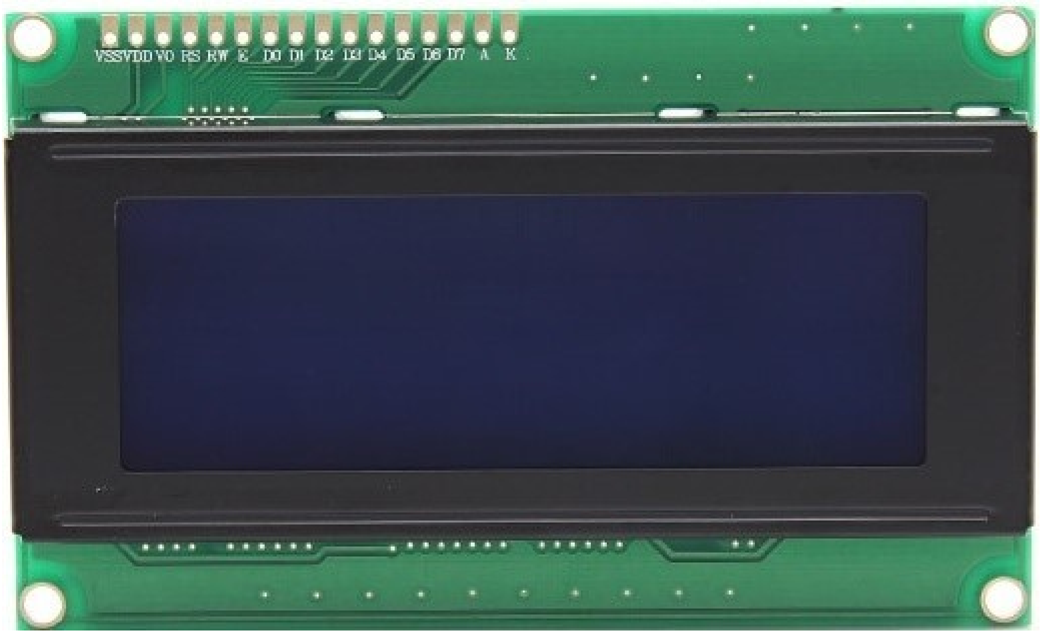


Figure 7. LCD 2004A Photo

Conclusion

The application note shows how a single pin of a microcontroller is used to interface with LCD2004A through a GreenPAK SLG46620V. The interface used for LCD2004A is 8 bit. Other LCD types can also be configured similarly.

Another important aspect of this design is that the LCD can be interfaced with any other device (Like PC) having a serial interface. The device must be capable of sending data bytes to configure the LCD module, so that LCD can be initialized and data can be written/clear on the LCD display. An application on PC such as: MATLAB, Hyper-terminal, Visual studio, LabVIEW, QT etc. can be used for this purpose.

Appendix

LCD Command Messenger

In order to send multiple commands from Arduino, command messenger library is used. The data format of sending commands from Arduino serial interface window is

1. 0, column no, row no;

   The number '0' at the start of the packet is identifier for row and column selection. Here the row number and column number must not exceed 3 and 19 respectively.

2. 1, character;

   After selecting the row and column, the character is written on the specified location. The number '1' at the start of the packet is identifier for character write. After character is written the cursor automatically moves to next location.

   Here it must be noted that the character written by user is first converted into ASCII code and then it is sent to LCD through GreenPAK SLG 46620V.

3. 2, clear;

   In order to clear the LCD the packet send will start with 2 (the identifier). The text 'clear' written after the identifier then clears the LCD.

Multiple commands can be sent by separating them with semicolon (;). The data is written on serial monitor window of Arduino IDE by the user according to format listed above. Based on this data the commands are sent to GreenPAK SLG46620V through Serial1 TX1 pin of Arduino mega 2560.

For more details about command messenger please visit the link.

http://playground.arduino.cc/Code/CmdMessenger

Example Application

An example application of data write, cursor set and LCD clear is shown in the video attached with the application note.

Step 1:

```
0 , 2 , 0 ; 1 , * ; 1 , * ; 1 , * ; 1 , * ; 1 , W ; 1 , E ; 1 , L ; 1 , C ; 1 , O ; 1 , M ; 1 , E ; 1 , * ; 1 , * ; 1
, * ; 1 , * ;
```

Write the above line in the serial window of Arduino IDE. It will print

******WELCOME******

In the center of first line of LCD.

Step 2:

```
0 , 5 , 2 ; 1 , s ; 1 , I ; 1 , l ; 1 , e ; 1 , g ; 1 , o ; 1 , . ; 1 , c ; 1 , o ; 1 , m ;
```

Write the above line in the serial window of Arduino IDE. It will print:

silego.com

in the center of third line of LCD.

Step 3:

2 , clear ;

Write above line in the serial window of Arduino IDE. It will clear the LCD.

Arduino Mega 2560 programming

The Arduino controller is programmed to send two bytes to SLG 46620V. The GreenPAK chip will then parse the data and use it for initialization, cursor setting, LCD write and LCD clear.

1ST Packet	Start Bit	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Stop Bit
	0	RS	E	x	x	x	x	x	x	1
2nd Packet	Start Bit	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Stop Bit
	0	D0	D1	D2	D3	D4	D5	D6	D7	1

Table 3. Data structure from Arduino to SLG 46620V

The two data packets keep on repeating themselves. The baudrate is set to 9600bps for both Arduino and SLG46620V.

The three main functions used in Arduino code are

1. **LCD Initialization**

The LCD2004A is initialized by keeping RS and E bits Low, then sending data bytes for D0, D1..D8. The E pin is then set high and then low to set the data.

The commands sent from Arduino are in order given below. These are for 8-bit initialization of LCD 2004A.

- RS Low , serial1.write (0x30) , E High , E low
- Serial1.write (0x30) , E High , E low
- Serial1.write (0x30) , E High , E low
- Serial1.write (0x38) , E High , E low
- Serial1.write (0x08) , E High , E low
- Serial1.write (0x01) , E High , E low
- Serial1.write (0x06) , E High , E low

- Serial1.write (0x0C) , E High , E low

For more details on LCD2004A initialization (8-bit mode) please visit the link.

LCD Initialization (alfredstate.edu)

2. LCD Clear

For clearing the LCD display, write 0x01 on data bits. The E pin is then set high and low to clear the LCD.

3. LCD write

For writing a particular character on LCD display, set the RS pin high first. Then send corresponding ASCII code on data bits (the Arduino IDE take cares of ASCII code you have to just write the character). The E pin is then set high and low to write the LCD.

4. Cursor Set

For cursor setting some certain numbers (shown in table below) are sent to data bits. The E pin is then set high and low to set the cursor.

	Row 0	Row 1	Row 2	Row 3
Col 0	128	192	148	212
Col 1	129	193	149	213
Col 2	130	194	150	214
Col 3	131	195	151	215
Col 4	132	196	152	216
Col 5	133	197	153	217
Col 6	134	198	154	218
Col 7	135	199	155	219
Col 8	136	200	156	220
Col 9	137	201	157	221
Col 10	138	202	158	222
Col 11	139	203	159	223
Col 12	140	204	160	224
Col 13	141	205	161	225
Col 14	142	206	162	226
Col 15	143	207	163	227
Col 16	144	208	164	228
Col 17	145	209	165	229
Col 18	146	210	166	230
Col 19	147	211	167	231

Table 4. Cursor setting for LCD

The other functions of LCD2004A like auto scroll, cursor blink etc. can be similarly implemented from its datasheet.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks