

Serial LCD Interface (SLI)TM

Assembly Instructions and Technical Reference V1.5



1.0 - Product Description

The Serial LCD Interface (SLI)TM is a self contained display unit with TTL and RS232 compatible asynchronous serial inputs. The unit is designed to allow the engineer, technician, or hobbyist to integrate it into a circuit with a minimum of interfacing requirements. Jumperless control and automatic detection of data rates eliminates difficult interfaces and complex controlling code. Low cost and line-powering features make it suitable for use in commercial products. With both alphanumeric and hexadecimal display mode, SLITM is a powerful debugging and display tool for many applications.

Features:

- Jumperless Control/Operation
- Wide Baud Range, 100 to 125,000 bps
- RS232 and TTL Compatible
- RS232 Serial Powered *
- Auto-Baud Rate Detection
- ASCII and Hex Display Mode
- D Sub 9 Female Connector (DB-9) and IDC Connector
- Allows direct control of the LCD functions and features
- Designed for all 8x1 to 40x2 character HD44780 controlled LCDs with 14x1 pin connectors

* With some serial ports, notebooks in particular, the power available for SLITM may be low enough for the display to have a low contrast however it is usually readable. As SLITM has additional power inputs for unregulated and regulated 5 Volts, it is easy to connect a 9 Volt battery to the unit. Only one power source should be used at a time. Serial Power should not be enabled at the same time SLITM is being externally powered.

19-Feb-97

Copyright © Wirz Electronics, 1997

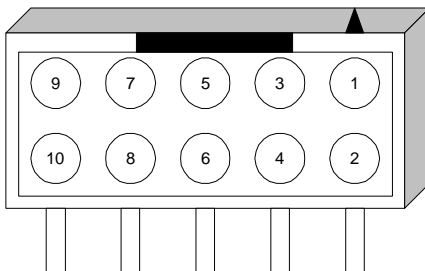
Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

2.0 - Product Specifications

Size: 1.4" x 1.9", 35.6 mm x 48.3 mm
Power Requirements: 50 mW (10 mA @ 5 Volts) including a 16x2 LCD
Data Speed Range: 100 bits per second (bps) to 125,000 bps
Data Voltage Range: -15V to +15V
Logic Low Voltage: -15V to +0.8V
Logic High Voltage: 2.5V to +15V
Data Format: 8 Data Bits, No Parity, 1 Stop Bit (8-N-1)
Data Buffer Size: 16 Bytes
Power-on Init. Time: 40 msec
Power Input: +5 VDC Regulated or > 6 VDC Unregulated
Regulated Power Output: 40 mA @ 5 V (May not be available in Serial Power Mode)
LCD Connection: 14x1 on 0.100" Centers
Pin 1 - Gnd
Pin 2 - +5V
Pin 3 - LCD Contrast - 0V to +5V
Pin 4 - R/S (Data/Instruction Select)
Pin 5 - R/W (Data Read/Write Select)
Pin 6 - E (Data/Instruction Strobe)
Pin 7 - Data Bit 0
Pin 8 - Data Bit 1
Pin 9 - Data Bit 2
Pin 10 - Data Bit 3
Pin 11 - Data Bit 4
Pin 12 - Data Bit 5
Pin 13 - Data Bit 6
Pin 14 - Data Bit 7
LCD Types: HD44780 Controlled 8x1 to 40x2 Liquid Crystal Displays
RS-232 Connector: D Sub 9 Female Connector (DB-9) set as Data Communications Equipment (DCE)
Pin 3 - Tx (Data)
Pin 4 - Dtr (Serial Power)
Pin 5 - Gnd
Pin 7 - Rts (Serial Power)
All other pins are not connected
IDC Connector: 5x2 on 0.100" centers
Mounted on Comp. Side



Gnd	-10	9 - Gnd
No Connect	- 8	7 - Serial In
+5V	- 6	5 - +5V
Unregulated V	- 4	3 - Unregulated V
_Reset	- 2	1 - LCD Contrast

Power Connector: 2x1 on 0.100" centers - Connected to Gnd and Unregulated V

Wirz Electronics

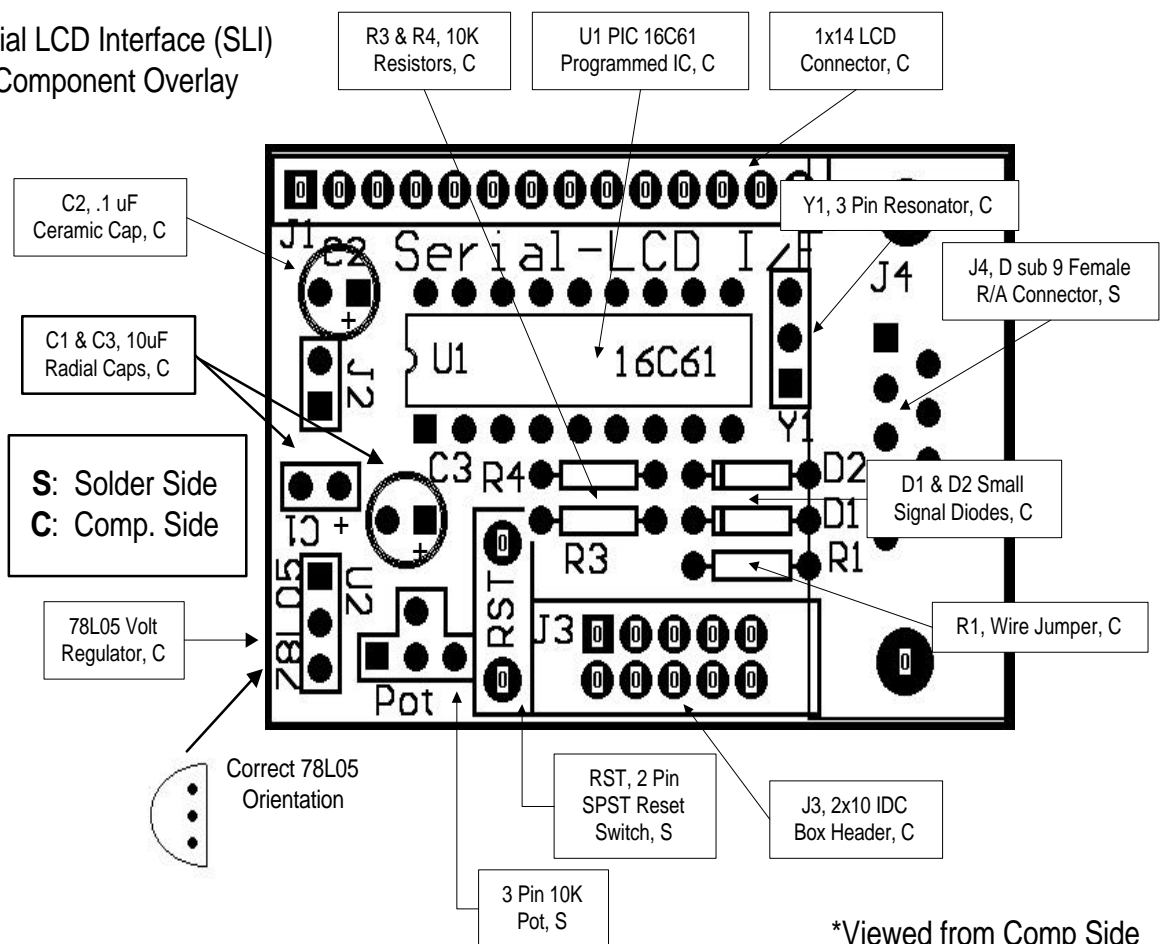
6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

3.0 - Parts List and Component Placement

Silkscreen Name	Description	Installation Side
R1	Wire Jumper	Comp
R2, R3	10 K, 1/4 Watt Resistor	Comp
D1, D2	Small Signal Diode	Comp
J1	1x14 LCD Socket	Comp
J2	1x2 Power Input	Optional
J3	2x5 IDC Data & Power Input	Comp
J4	D sub 9 RS-232 Serial Input	Solder
U1	Custom Programed PIC 16C61	Comp
U2	20 Mhz Resonator	Comp
RST	SPST NO Reset Switch	Solder
POT	10 K Thumbwheel Pot	Solder
78L05	78L05 Voltage Regulator, TO-92	Comp

Serial LCD Interface (SLI)
Component Overlay



Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

4.0 - Assembly Instructions

Suggestions and Precautions:

- A basic knowledge of electronics and appropriate soldering skills are necessary to assemble the kit.
- A soldering iron, rosin flux core solder, and basic hand tools will be required for assembly. A multi-meter will also be helpful for testing connections and trouble shooting.
- Read all instructions before beginning assembly.
- Double check all connections before applying power to the board.
- Always disconnect power from the board before altering it.
- These instructions are designed to allow the user to build the SLI™ and avoid getting into any tight places because of part placement conflicts. Any deviation from the recommended mounting side should be considered carefully. The mounting side of connectors J1, J3, J4, and IC U1 should not be switched.
- A P.C. with a free RS232 serial port and a terminal program or the SeeThis SLI™ test program is helpful to verify proper operation. The latest SeeThis program can be obtained via the Internet at <http://www.wirz.com/> or from any of the SLI™ Distributors.
- Assembly time is approximately 15 to 30 minutes in duration.

Instructions:

Except where noted otherwise, all components are installed on the component side and soldered on the solder side.

1. Install diodes D1 and D2 into the card and make sure the bands are facing away from the 9 Pin D-Shell connector.
2. Install a wire jumper next to the two diodes in the position marked R1.
3. Install the ceramic resonator at Y1. The resonator is a 3 pin epoxy dipped device with the part # ZTT 20.00 MX stamped on it.
4. Install and solder the 9 Pin D Shell Connector at J4 with the pins coming up on the comp. side (the Shell on the Solder Side) and solder the two mounting tabs.

To test to make sure the diodes and 9 pin connector is installed properly, plug the assembly into a 9 Pin Male D-Shell connector setup as DCE (Data Communications Equipment - Pin 3 is TX) from the host PC. Boot a terminal emulator and see if you get a positive voltage between the band side of the diodes and ground. If you don't, check the wiring.

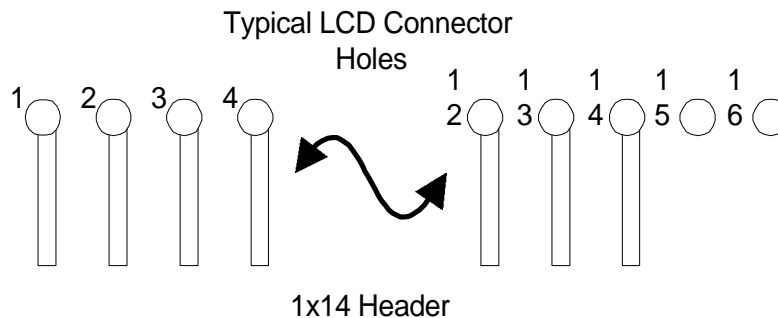
5. Install the two 10K Resistors at R3 and R4.
6. Install the reset switch. This is done from the solder side, like the 9 Pin D-Shell. The switch has a slight bend at the end of the lead, this should be straightened. It is important to push the switch in firmly so that the leads protrude far enough on the component side for proper soldering.

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

7. Install the 5x2 IDC connector at J3 on the comp side.
 8. Install the contrast potentiometer from the solder side at the position marked Pot.
 9. Install the 78L05 on the component side. The 78L05 is a three pin voltage regulator in a TO-92 package. The flat side of this three terminal device should be facing in toward the middle of the SLI™ as denoted by the component placement diagram above.
 10. Next, install the two 10 uF caps at C1 and C3. The capacitor polarity should be noted in the above placement diagram. It is important to install all capacitors with the correct polarity. You should leave 0.050" between the board and the bottom of all capacitors. This is to prevent the vents from being sealed, or flux getting into them by capillary action.
 11. Install the 0.1 uF cap at C2. The capacitor polarity should again be noted.
- The assembly should be connected to the PC, and the terminal emulator should be booted again. At this point, +5V should be measured between Pin 14 of IC U1 and ground.
12. Install the PIC™ IC at U1. Pin 1 (or the notch at the end of the chip body) should be facing the outside of the SLI™ Card.
 13. Install the 14x1 Socket for the LCD at J1. When doing this, you may want to use a piece of adhesive tape to keep the socket vertical.
 14. Install the 14 Pin Header on the LCD to pins 1 to 14. Some LCD's have 16 holes, SLI only uses the first 14.



15. Plug in the LCD, plug the assembly into the host PC, and boot the terminal program or SeeThis.

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

5.0 - SLI™ Operation

Because of the wide range of options for the SLI™, the operating instructions for the SLI™ have been broken up into a number of different subgroups.

5.1 - Electrical Connections

The SLI™ can be hooked up easily to an RS-232 DCE Host. This means that data output comes through Pin 3 of the 9 pin connector. Self-Power comes through Pin 4 and Pin 7 of the 9 pin connector. These voltages must be greater than +6 Volts from the host.

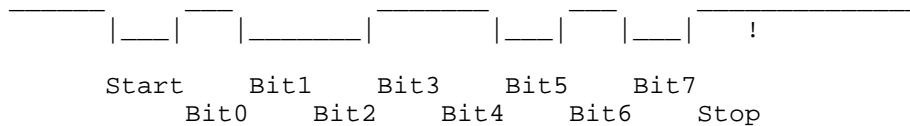
The SLI™ can also be hooked up through the IDC connector on the back of the PCB following the pinouts given in the specifications. The SLI™ can either be powered by the host circuit (driving the +5V Lines) or by an unregulated Voltage greater than 6 Volts for the on-board regulator.

If the on-board regulator is used, up to 50 mA at +5V can be drawn from the SLI™ for powering other circuits. This current margin may not be available with self-powered RS-232 Connections.

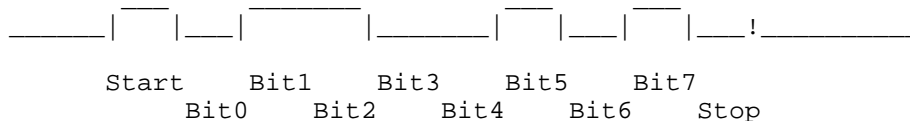
5.2 - Basic Operation

Upon Power-Up, the SLI™ requires a nominal delay of 40 ms to initialize the LCD and output the opening message.

After the initialization has taken place, the input line is polled for data polarity. If straight TTL is used, then the SLI™ will use positive logic, i.e. the data input will look like:



RS-232 will appear as negative logic to the SLI™, the previous data would look like:



These differences are compensated automatically in the SLI™. The important issue is that the polarity cannot change without the SLI™ being reset. If the polarity is changed after the initial power-up, garbage characters will be displayed on the LCD.

The SLI™ can be reset at any time by pressing the Reset button on the back of the device or grounding the reset line on the IDC connector. If, after 40 msec, the message

Send <CR> 8-N-1
Esc - Menu

does not come up, then check the contrast control on the back of the SLI™. If the message does still not come up, then check for +5V on the LCD.

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

While we have done everything possible to make sure the SLI™ works with every possible LCD, it is impossible to test each and every LCD for compatibility with the SLI™. If the LCD you are using does not display the message above after power up even after checking the contrast level, you will have to go back to the LCD manufacturer to understand what is the pinout and controller used on the LCD.

Once the Initial Message is up, an ASCII carriage return (0x0D) is sent to the SLI™ which causes the autobaud feature to determine the speed of the data coming in. The SLI™ keys off the pattern of the ASCII Carriage Return. While it may work in some circumstances with other characters, it is not guaranteed that the SLI™ autobauding will work at all data speeds.

After sending the ASCII Carriage Return and waiting 5 msec for the LCD to clear, the SLI™ is ready to run.

The SLI™ displays the ASCII data set as defined by Hitachi for use in the HD44780. This means that some ASCII characters may not come out as expected (i.e. "\" comes out as a "Y" with a bar through it).

Note that data received in packets in formats other than 8-N-1, will be displayed in a strange manner or not at all. This is because the SLI™ handles 8 characters and looks for the stop bit. Formats which may place a "0" at the Stop Bit will cause unpredictable results.

There is a reset switch and a contrast adjustment on the back of the SLI™. These controls are used as their names imply. The only important notes about these two controls is if they are controlled by an external host through the IDC connector. The Reset Line is active low and should only be driven by an open-collector output. If the contrast is to be controlled externally, then the pot must not be installed and the contrast line driven with 0 to 5 Volts DC.

5.3 - Data Speeds

The SLI is designed to accept any data rate from 100 to 125,000 bps. Writes and updates to the LCD screen take place at approximately 1 KHz, at data transfer rates higher than 9600 bps the SLI will buffer any display characters or LCD commands for later use. This buffer is a 16 byte long circular buffer which will overwrite itself. To prevent data from being lost at data transfer speeds above 9600 bps, it is important to put a delay in the sending software after sending 16 bytes to allow the LCD write operations to complete.

The following example illustrates this point. With SLI connected to a PC running a terminal program at 115,200 bps a user can type on the keyboard as fast as he can and the text will always be displayed correctly because the users typing rate is much slower then the screen update rate. But if data was sent at this speed continuously, the SLI's internal buffer would probably fill up before the LCD completed displaying the first character.

5.4 - Regular Characters

The SLI™ handles five ASCII control characters: Escape, Carriage Return, Line Feed, Backspace, and Form Feed. When any of these characters are returned, special action is taken on the LCD and the Hitachi ASCII equivalents and user-definable characters are not displayed. Along with these characters, 0x0FE or 254d is supported as a method of directly controlling the LCD Display, 0x0FE or 254d should never be sent to the SLI™ as data. In this section, the basic ASCII Control Characters are described. The special characters are described in later sections.

The Carriage Return/Line Feed combination is used to start a new line on the display. If a single line display is used (note specification in the Escape Menu below), the combination is used to clear the display. In two line displays the second line is scrolled up.

The SLI™ will accept either a Carriage Return or Line Feed to cause the Scroll Up of the display. If one character is received and the previous character was the other character (i.e. Line Feed followed by a Carriage Return) the character is ignored. This feature is to prevent a CR/LF combination from causing two line scrolls (many terminal

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

emulator programs send a CR/LF as a matter of course).

The Form Feed Character is simply a method for clearing the screen and putting the Cursor at the first location. The backspace key will delete the character before the cursor and move the cursor into its position.

5.4 - User-Definable Characters

The SLI™ provides for up to 8 user defined characters. These characters can be displayed on the screen using Hex Codes 0x00 through 0x007. These are actually infrequently used ASCII Control Characters.

Each character is setup as an 8x5 box, requiring 8 bytes of data in the CG RAM Area. Each byte is a row in the box (the top most one being the lowest address). Accessing the CG RAM from DD RAM (which contains the Hex/ASCII Codes for each character) is accomplished by moving the cursor into the CG RAM area by the instruction 0b001xxxxxx (where "xxxxxx" is the address within the CG RAM). The address of the 8 Row Bytes in CG RAM starts at 8x the Character address. If the Character Graphic RAM is being written to and the Escape, CR, LF, FF, & BS commands are sent, SLI™ may behave unpredictably. A simple fix is to OR 0x020 to all the CG data writes.

For example, say you wanted to make character 0x02 a "\" (which isn't in the LCD Character Set). The sequence of instructions to do this would be:

```
Send( 254 )      ; Move the Cursor into the CGRAM Area
Send( 0x040 plus 8 plus 8 = 0x050 )      ; for Character 2
Send( 0x010 + 0x020 = 0x030 )      ; Slash:  X _ _ _ _
Send( 0x010 + 0x020 = 0x030 )      ;          X _ _ _ _
Send( 0x008 + 0x020 = 0x028 )      ;          _ X _ _ _
Send( 0x004 + 0x020 = 0x024 )      ;          _ _ X _ _
Send( 0x002 + 0x020 = 0x022 )      ;          _ _ _ X _
Send( 0x001 + 0x020 = 0x021 )      ;          _ _ _ _ X
Send( 0x001 + 0x020 = 0x021 )      ;          _ _ _ _ X
Send( 0x000 + 0x020 = 0x020 )      ;          _ _ _ _ _

Send( 254 )      ; Move Cursor Back to Start of DD RAM Area
Send( 0x080 )
```

* Note that + is a logical OR, I.E. 1+1=1.

5.5 - The Escape Menu

After sending an ASCII "Escape" character, you will get a menu and will be prompted to specify the number of columns and rows of the LCD you are using along with whether or not you want a shifting display, and if you want to display the incoming data as Hex codes rather than ASCII Characters.

The HD44780 can support LCDs from 8 columns and 1 row (8x1) to 40 columns and 2 rows (40x2). The default configuration can be changed in the Escape Menu by backspacing over the default values (16x2) and changing them. Columns can be from 8 to 40 and rows, 1 or 2.

Invalidly changing these parameters may result in data not being displayed, not being displayed properly, or the LCD being cleared at an unexpected time or not using up all the available columns. If any of these problems appear, the Escape Menu can be re-entered and the parameters changed until the LCD is behaving properly.

In our testing, we have found that some 16x1 displays are actually configured as 8x2 (with two columns of 8 on the same line). In this case, the configuration should be 8x2 and the display will appear to "Tab" when scrolling between the two rows of 8 columns.

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

After you are prompted to change the LCD, a prompt is received for shifting the display. In the HD44780 controller, there is display RAM available for up to 40 columns. In the SLI™, the full 40 columns can be used (even if the LCD doesn't support it) by enabling shifting. When the displayed characters reach the end of the LCD line, the characters are shifted over to the end of the 40 column line. At the end of the line, the LCD scrolls the data up and shifts back to the first column.

The final configuration item in the Escape Menu is whether or not the data is to be displayed as hex data. Each character is displayed as "-XX" with the "-" character used to differentiate characters and the "XX" being the data. The Hex Display Option is useful for debugging incoming data. The only way to exit Hex Display mode is to reset SLI and resend the <cr> for autobauding.

5.6 - Direct LCD Control

Commands can be sent directly to the HD44780 controller by first sending a 0x0FE or 254d character to the SLI™. The next command will be passed directly to the HD44780 as an instruction (not data to be displayed). 0x0FE or 254d should never be sent to the SLI™ as data.

This method of sending control data to the SLI™ must be used with caution. The commands are not parsed and the internal variables in the SLI™ controlling the behavior of the LCD (i.e. scrolling, backspacing, etc.) are not always updated. This may lead to unexpected results after sending data.

Before moving the cursor explicitly, it is recommended that either a Form Feed or Carriage return is sent to the SLI™ to ensure that any subsequent data writes do not cause an inadvertent scroll of the display.

A detailed table of the instructions can be found in the Hitachi HD44780 Data sheet, a few examples are listed in the next section.

5.61 - Example Direct LCD Control Operations

To Move the Cursor to the First Line:

Send 0x0FE
Send 0x080 ; Add 0 - 39 to put cursor at different columns of the line

To Move the Cursor to the Second Line:

Send 0x0FE
Send 0x0C0 ; Add 0 - 39 to put cursor at different columns of the Line

To Clear the Screen:

Send 0x0FE
Send 0x001

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

5.62 - Notes on Direct LCD Control with SLI™

<u>Instruction</u>	<u>Bits</u>	<u>Comments</u>
Clear Display	0000 0001	Sending a Form Feed (0x0C) will do exactly the same command. The Internal SLI™ Cursor Position will be updated.
Return Home	0000 001x	Sending this instruction will also update the Internal SLI™ Cursor Position.
Entry Mode Set	0000 01IS	Sets the cursor move direction and specifies whether or not to shift the display. This command should not be used because it may cause the internal SLI™ characters to be incorrect. Instead, the functions can be simulated using the shift option in the Escape Menu.
Display Control	0000 1DCB	Turns on/off the LCD display and cursor. The cursor display can be changed, but it is not recommended that the LCD Display ever be shut off.
Curs/Disp Shift	0001 SRxx	Allows moving the cursor and shifting the display without changing LCD display ram contents. While the cursor position is updated after using this command, this command is not recommended.
Function Set	001D Nfxx	Initializes the display and sets the interface length. This command is not recommended.
Set CG RAM Addr	01pp pppp	Allows the user to start writing at a specific location in CGRAM. After this command, a "Set DD Addr" command should be executed afterwards.
Set DD RAM Addr	1ppp pppp	Allows the user to move the cursor anywhere on the LCD. The internal SLI™ cursor position is updated after this command. Note that the top line starts at 0 and the bottom line starts at address 0x040. Up to 40 characters on each line can be written.

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

6.0 - Sample Serial Routines

All example serial routines are available for download on the Internet at <http://www.wirz.com/> or from any of the distributors.

Parallax Basic Stamp™ Sample Code:

```
'Program: SLI.BAS Printing at screen location with the Wirz
'Electronics Serial LCD Interface (SLI)™ using the Basic Stamp™ I.
,
'Only two lines have been added to allow the SLI™ to Auto-Baud
'detect on any data rate from 100 bps to 125 Kbps. This is a
'Carriage Return and delay.
,
Symbol I=254
Symbol ClrLCD=1
Symbol prn_at=136
low 7
pause 1000
serout 7,N2400,(13)          'This line added for SLI™ Auto-Baud.
pause 5                      'This line added for SLI™ Auto-Baud.
serout 7,N2400,(I,ClrLCD)
serout 7,N2400,("Hello World")
```

Custom Computer Services PCM C for the Microchip PIC™:

```
//      LCD.C, Ben Wirz, January 11, 1997
//      A sample program complied with the CCS PCM C Compiler for the
//      PIC16C84 running at 10 MHZ. It is important not to attempt serial
//      input as the Rx and Tx are defined as the same pin.  SLI only
//      requires one input data line.

#include <16c84.h>
#include <stdio.h>
#define Delay(Clock=10000000)          //Set Clock Speed
#define rs232(baud=9600,xmit=pin_b0,rcv=pin_b0)  //Setup Serial Port

main ()
{
    delay_ms(40);                //Wait 40 mS for SLI to initialize
    printf("r");                 //Send a Carriage Return for Autobauding
    delay_ms(5);                 //SLI is now ready to Receive Data
    while(1)                     //Infinite Loop
    {
        printf("Hello World\n");
        delay_ms(1000);          //1 Second Delay
    }
}
```

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

Microchip PIC™ Assembly Example :

Hello.asm

```
title "HELLO - Put 'Hello World' on the SLI Display."
;
; This is a simple PIC (16C84) Program to send the Message "Hello World"
; to the SLI from a PIC using RA0 as the Output.
;
; This is a Demonstration of serial.inc. Note that the Registers are defined the
; same way here.
;
; Hardware Notes:
; Reset is tied directly to Vcc and PWRT is Enabled.
; The Pic is Running with a 3.579545 MHZ (Colour Burst) Crystal for
; 16C84
; NOTE: The Frequency isn't all that important for proper Operation of the
; SLI.
; RA1 is Connected to the SLI
; _MCLR is pulled up using a 4.7K resistor and a
;
; Myke Predko
; 97.01.12
;
;
; LIST P=16C84, R=DEC ; 16C84 Runs at 3.579545 MHZ
errorlevel 0,-305
INCLUDE "c:\mplab\p16c84.inc"

; Register Usage
CBLOCK 0x0C ; Start at Top of File Registers
Count ; Count Values for Dlay "Loop"
Counthi
_VarEnd ; Mark Where the Serial Stuff Can Go
ENDC

PAGE
__CONFIG __CP_OFF & __XT_OSC & __PWRTE_ON & __WDT_OFF
; Note that the WatchDog Timer is OFF

; Code for "HELLO"

org 0

goto MainLine

org 4 ; Put in serial.inc starting at Int
Handler
INCLUDE "..\..\INCLUDES\SERIAL.INC"

PAGE
; Now, Put in Program Code

HelloWorld ; "Hello World" String
addwf PCL
dt " Hello World", 0

MainLine ; Now, Run the Program

movlw 2 ; Make Sure Nothing is Output
movwf PORTA
bsf STATUS, RP0 ; Make RA1 Output

bcf TRISA & 0x07F, 1
bcf STATUS, RP0

call Dlay ; Wait 60 msec for LCD to
become enabled
call Dlay

movlw 0x0D ; Send CR to Initialize the
Display
call SendCHAR

call Dlay ; Wait 30 msec for everything
to get Setup

clrf FSR ; Setup FSR to Output the
String

Loop ; Now, Output the Message

movf FSR, w ; Send the the Character at FSR
offset
call HelloWorld
iorlw 0 ; Are we at the End?
btfsc STATUS, Z ; Yes, Stop the Program
goto Stop

call SendCHAR ; No - Send the Character to the
SLI

incf FSR ; Point to the Next Character

goto Loop

Stop ; Finished with the Program
goto $ ; Yes, Loop Around forever

PAGE
; Subroutines Needed by "Hello"
Dlay ; Delay for 30 msec
movlw 16
movwf Count
movlw 39
movwf Counthi
decfsz Count
goto $ - 1
decfsz Counthi
goto $ - 3

return

end
```

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

Serial.Inc

This is an interrupt based serial include file for Microchip PIC™ 16C61, 16C71, and 16C84's.

```
subtitle "Serial Serial 9600 bps Interrupt Rx/Tx."
;
; These Routines are used to provide a Serial (Digital Logic)
; from a PIC. These routines do not use the Serial communications
; hardware in some PICs and does require interrupts. Therefore,
; The 16C61, 16C71, and 16C84 are the recommended target devices.
;
; Note that the input is DIGITAL LOGIC. For RS-232 "Cheats", the
; values Read/Output will have to be inverted.
;
; Currently they are setup for a PIC running at
; 3.579545 MHz (Colour Burst Frequency) and 9600 bps. This
; can be changed by changing the count value.
;
; Note that the Received Data is put into the 5 byte circular
; "Buffer".
;
; It is assumed that the host the PIC running this code can use
; a simple 3 wire interface.
;
; Hardware Notes:
; The Pic is Running with a 3.579545 MHz (Colour Burst) Crystal for
; 16C84
; RA1 is Connected to RS-232 "TX" Line
; RB0 is Connected to RS-232 "RX" Line
; NOTE: RA1 is used to Transmit data rather than RA0 because
; RA0 is usually used to look like the LSB of PORTB.
;
; Myke Predko
; 96.09.22

; Declarations
cblock _VarEnd
BitCount ; Count Value for the Bits
Char ; Temporary Storage of the
Character Read In
Temp ; Temporary Value for
Program
RRCount ; Loop Count for Reading the
Characters
SCCount
Next ; Pointer to the Next Value in
the Chain
ShadowFSR ; Saved the FSR Buffer Value
_w ; Context Save Registers
_status
BufferStart ; Character Buffer
Buffer1
Buffer2
Buffer3
BufferEnd
endc

; Constants
Bit EQU 27 ; Delay between the Bits, using the
BitStart EQU Bit / 3 ; Delay to check in character middle

; The Bit Count is Calculated by the Formula:
;
; Period = 1 / bps ; Get the Period required.
; Total Cycles = ( Period * ( PIC_Frequency / 4 )) - 15

; Number of Cycles to Execute
; Bit = ( Total_Cycles / 3 ) + 1
; The actual number of Delay
Loops
; For Example: Getting 2400 bps at 4 MHz:
; Period = 1 / 2400 = 0.0004167
; Total Cycles = ( 4.167(10^-4) * 4(10^6) / 4 ) - 15 = 402
; Bit = ( 402 / 3 ) + 1 = 135

; RX Interrupt Handler
; Read Serial Routine - When Character Read and Put in Buffer return to
Loop
RXInt

movwf _w ; Save the Context Registers
movf STATUS, w
movwf _status
bcf STATUS, RP0 ; Make Sure we're in Bank 1

ReadRS232 ; Start of Original Serial Read Routine

decfsz BitCount ; Wait for Halfway through the Character
goto $ - 1

btfss PORTB, 0 ; Do we *Still* have the Start Bit?
goto ShortRRLoop ; Point of Inversion depending on Logic

movlw 0x010 ; Reset Interrupts Before Returning
movwf INTCON
goto RREnd ; Wait for the Next Transition

ShortRRLoop ; Now, Loop Around Here for Each Character

movlw Bit + 2 ; Load and wait for the Bit Count
movwf BitCount ; NOTE: 3 goto $ + 1 removed
decfsz BitCount ; Now Delay
goto $ - 1

rrf PORTB, w ; Get the Bit Value Coming in
rrf Char ; Update the Character

decfsz RRCount ; Do 8x
goto ShortRRLoop

movlw Bit ; Now, Look at the Stop Bit
movwf BitCount
decfsz BitCount
goto $ - 1

movlw 0x010 ; Reset the Interrupt Enable (In Case
movwf INTCON ; one comes in during code below)

btfss PORTB, 0 ; Now, Check to see if the Stop
Bit is There
goto RREnd ; It's not, Ignore the Character

HaveRS232 ; We Have Correct RS232
Value - End and Return

incf Next ; Now, Store the Character in
the Buffer
```

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

```

movlw   BufferEnd + 1           ; Are we at the End of the
Buffer?
subwf   Next, w
btfss   STATUS, C               ; If Carry Set, No
movlw   BufferStart - ( BufferEnd + 1 ) ; Else, have to Roll to Start
addlw   BufferEnd + 1           ; Reset the Start of Next

xorwf   FSR, w                  ; Swap Next with
                                ;FSR (Shadow Contains Current FSR)

xorwf   FSR
xorwf   FSR, w
movwf   Next                    ; Save the Old FSR Value

movf    Char, w                 ; Store the Character in the Buffer
movwf   INDF

movf    Next, w                 ; Restore the FSR
xorwf   FSR, w
xorwf   FSR
xorwf   FSR, w
movwf   Next                    ; Store the Current Next Value

RREnd                      ; Return to where the Routine was Called

movlw   8                       ; Reload the Bit Counter for the Loop
movwf   RRCCount

movlw   BitStart                ; Reload the Counters for the Next Serial Character
movwf   BitCount

IntEnd                      ; Finished with the Interrupt

movf    _status, w              ; Restore the Context Registers and Return
movwf   STATUS
swapf   _w
swapf   _w, w

retfie

; Interrupt Setup Routine - Init the Variables for the RXInt Handler
; NOTE: It is assumed that PORTA.1 is set for Output
RXSetup

movlw   BitStart                ; Setup the Values for the Interrupt
movwf   BitCount
movlw   8                       ; Want to Read 8 Characters
movwf   RRCCount

movlw   BufferStart              ; Initialize the Indexes to the Buffer
movwf   ShadowFSR
movwf   Next

movlw   0x090                   ; Turn on the Receive Interrupts
movwf   INTCON

return

; Character Read Routine. Wait for Something to be Available
GetCHAR

movf    ShadowFSR, w             ; Get the Last Character Read
subwf   Next, w                  ; in the Buffer and If it
btfss   STATUS, Z                ; hasn't been read - return it
goto    GetCHAR                  ; Else, Wait for the Next

incf    ShadowFSR, w             ; Point to the Next Value in
addlw   0 - ( BufferEnd + 1 ) ; the Table
btfsc   STATUS, C

```

```

addlw   BufferStart - ( BufferEnd + 1 )
addlw   BufferEnd + 1
movwf   ShadowFSR               ; Save it for the Next Read

movwf   FSR                      ; Get the newly entered Character
movf    INDF, w

return

; Character Send Routine - Send the Character in "w"
SendCHAR                          ; Serial Send the Character to the Host

movwf   Temp                     ; Save the Character to Send

movlw   8                       ; Setup the 8 Bits to Send
movwf   SCCCount

bcf     PORTA, 1                 ; Output the Start Bit

SCLoop                          ; Loop Here for Each Character

movlw   Bit                      ; Delay the Length of the Bit
movwf   Count
decfsz  Count
goto    $ - 1

rrf     Temp                     ; Rotate the Bit Over
btfsc   STATUS, C                ; Do we have to Send a '1'?
goto    SCSOne

nop                                     ; Send a Zero
bcf     PORTA, 1                 ; NOTE: it is Inverted!

goto    SCLoopEnd

SCSOne                          ; Send a One

bsf     PORTA, 1                 ; NOTE: it is Inverted!

goto    $ + 1                    ; Delay 2 cycles to Even Up

SCLoopEnd                      ; Do this for 8 Bits

nop                               ; To Time Everything Out

decfsz  SCCCount
goto    SCLoop

nop                               ; Keep everything on track

movlw   Bit                      ; Delay for the Bit
movwf   Count
decfsz  Count
goto    $ - 1

goto    $ + 1                    ; Delay the Correct Number of Cycles
goto    $ + 1

bsf     PORTA, 1                 ; Send a Zero Stop Bit

nop
movlw   Bit                      ; Wait the Correct Number of Cycles before Returning
movwf   Count
decfsz  Count
goto    $ - 1

return
PAGE

```

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

6.0 - Distributors

Australia

DonTronics

PO Box 595, Tullamarine 3043

Tel: +613 9338-6286 Fax: +613 9338-2935

<http://www.labyrinth.net.au/~donmck/>

South Africa

Interface Products (Pty) Ltd.

PO Box 15775, Doornfontein 2028

Tel: +27 (11) 402-7750 Fax: +27 (11) 402-7751

<http://www.ip.co.za/ip/>

United States

Wirz Electronics

6100 Pershing 2-A, St. Louis MO 63112

Tel: +1 (314) 862-3370 Fax: +1 (314) 862-3371

<http://www.wirz.com/>

7.0 - Feedback and Suggestions

Wirz Electronics is always working to improve our products. If you have product suggestions or improvements for existing products, please feel free to contact us by phone, fax, or visit our web page. You may also email your suggestions to sales@wirz.com.

Wirz Electronics

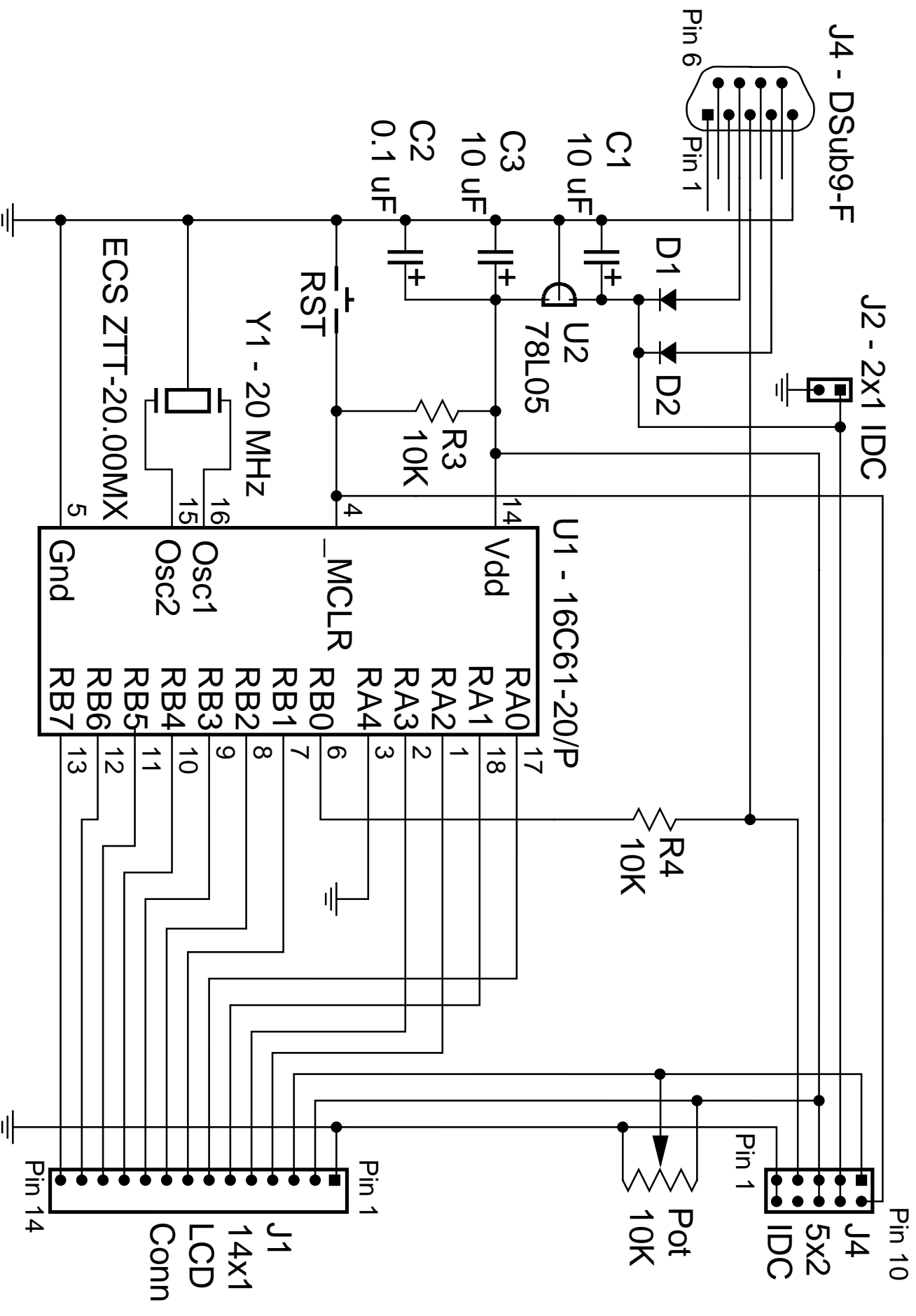
6100 Pershing, #2A St. Louis, MO 63112

Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>

Serial/LCD Interface



SLI



Wirz Electronics Copyright Information

Copyright © 1997 Wirz Electronics
6100 Pershing, #2A
St. Louis, MO 63112
All rights reserved.

Any person is hereby authorized to view, copy, print and distribute any portion of this document subject to the following conditions:

- 1 The document may be used for informational purposes only;
- 2 The document may only be used for non-commercial purposes except by Wirz Electronics' Distributors; and
- 3 Any copy of the document or portion thereof must include the above copyright notice.

Any product, process or technology described in the document may be subject to other intellectual property rights reserved by Wirz Electronics and are not licensed hereunder.

This document could include technical inaccuracies or typographical errors. Changes are regularly made to the information contained in this document. Changes may or may not be included in the future editions of the document. Wirz Electronics and its distributors may make improvements and/or changes in the Products, Processes, Technology, Descriptions, and/or Programs described in the document at any time.

This document is provided "As Is" without warranty of any kind, either express or implied, including but not limited to, any implied warranty or merchantability, fitness for a particular purpose, or non-infringement.

Life Support Policy

Wirz Electronics products are not authorized for use as critical components in life support devices or systems. As used herein:

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform properly can be reasonably expected to result in significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Trademark Notices

Serial LCD Interface™ and SLI™ are trademarks of Wirz Electronics

Basic Stamp™ is a registered trademark of Parallax Inc.

PIC™ is a registered trademark of Microchip Technology, Inc

Wirz Electronics

6100 Pershing, #2A St. Louis, MO 63112
Office: (314) 862-3370, Fax: (314) 862-3371, Internet: <http://www.wirz.com/>