Microcontrollers employ multiple approaches to communicate synchronously with peripheral devices and other microcontrollers. The synchronous serial interface (SSI) system can operate as a master or as a slave. The channel can have one master and one slave, or it can have one master and multiple slaves. With multiple slaves, the configuration can be a star (centralized master connected to each slave), or a ring (each node has one receiver and one transmitter, where the nodes are connected in a circle.) The master initiates all data communication. The Nokia5110 is an optional LCD display as shown in Figure 11.13. The interface uses one of the synchronous serial ports on the TM4C123.

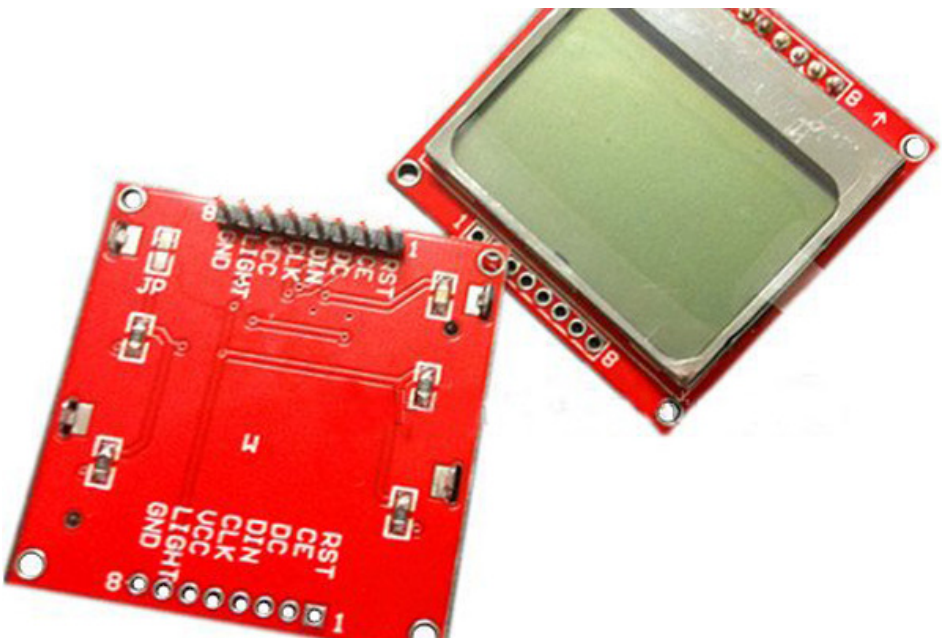

Figure 11.13. Optional Nokia 5110 LCD. Notice the PCB gives the signal names. Use the signal names not the pin numbers when connecting.

The TM4C123 microcontrollers has 4 **Synchronous Serial Interface** or **SSI** modules. Another name for this protocol is Serial Peripheral Interface or SPI. The fundamental difference between a UART, which implements an asynchronous protocol, and a SSI, which implements a synchronous protocol, is the manner in which the clock is implemented. Two devices communicating with asynchronous serial interfaces (UART) operate at the same frequency (baud rate) but have two separate clocks. With a UART protocol, the clock signal is not included in the interface cable between devices. Two UART devices can communicate with each other as long as the two clocks have frequencies within ±5% of each other. Two devices communicating with synchronous serial interfaces (SSI) operate from the same clock (synchronized). With a SSI protocol, the clock signal is included in the interface cable between devices. Typically, the master device creates the clock, and the slave device(s) uses the clock to latch the data (in or out.) The SSI protocol includes four I/O lines. The slave select **SSI0Fss** is a negative logic control signal from master to slave signal signifying the channel is active. The second line, **SCK**, is a 50% duty cycle clock generated by the master. The **SSI0Tx** (master out slave in, MOSI) is a data line driven by the master and received by the slave. The **SSI0Rx** (master in slave out, MISO) is a data line driven by the slave and received by the master. In order to work properly, the transmitting device uses one edge of the clock to change its output, and the receiving device uses the other edge to accept the data. SSI allows data to flow both directions, but the Nokia5110 interface only transmits data from the TM4C123. Notice the pin **SSI0Rx** (PA4) is not used, which would have allowed for receiving data from the device. The Nokia5110 interface does not use **SSI0Rx** (PA4).

The Nokia5110 can be connected to pins PA7, PA6, PA5, PA3, and PA2 on the TM4C123.

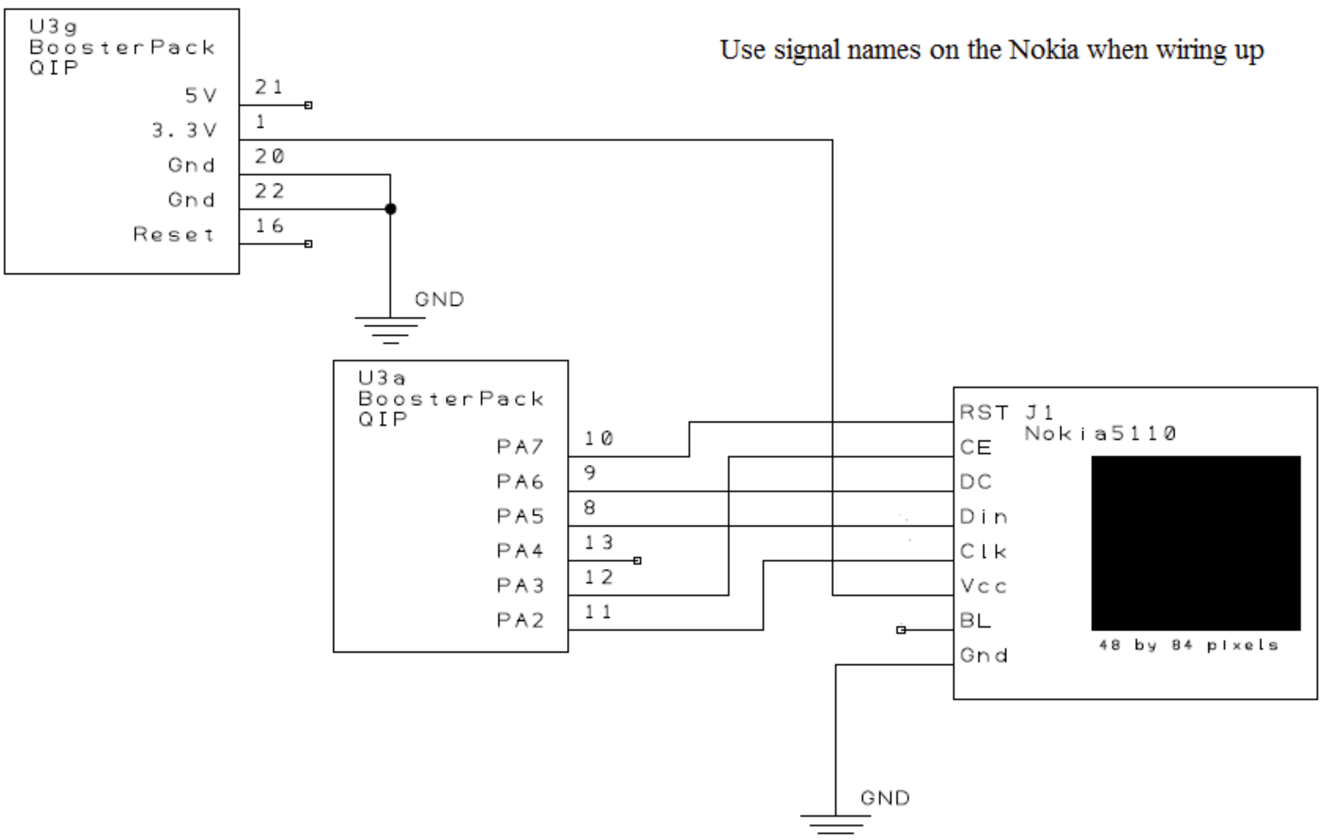

Figure 11.14. Connection between the Nokia 5110 LCD and the TM4C123. Use the signal names not the pin numbers when connecting.

Program 11.9 shows the I/O port connections and the Nokia display. Be careful, there are multiple displays for sale on the market with the same LCD but different pin locations for the signals. Please look on your actual display for the pin name and not the pin number. Program 11.9 also lists some of the prototypes for public functions available in the software starter project. If you have ordered and received the Nokia5110 display, open the C11_Nokia5110 starter project, connect the display to PortA. Be careful when connecting the backlight, at 3.3V, the back light draws 80 mA. If you want a dimmer back light connect 3.3V to a 100 ohm resistor, and the other end of the resistor to the **BL** pin.

// Blue Nokia 5110

// ---------------

// Signal (Nokia 5110) LaunchPad pin

// Reset (RST, pin 1) connected to PA7

// SSI0Fss (CE, pin 2) connected to PA3

```
// Data/Command  (DC,  pin 3) connected to PA6
// SSI0Tx       (Din, pin 4) connected to PA5
// SSI0Clk      (Clk, pin 5) connected to PA2
// 3.3V         (Vcc, pin 6) power
// back light    (BL,  pin 7) not connected, 4 white LEDs which draw ~80mA total
// Ground       (Gnd, pin 8) ground

// Red SparkFun Nokia 5110 (LCD-10168)
// -----------------------------------
// Signal       (Nokia 5110) LaunchPad pin
// 3.3V         (VCC, pin 1) power
// Ground       (GND, pin 2) ground
// SSI0Fss      (SCE, pin 3) connected to PA3
// Reset        (RST, pin 4) connected to PA7
// Data/Command  (D/C, pin 5) connected to PA6
// SSI0Tx       (DN,  pin 6) connected to PA5
// SSI0Clk      (SCLK, pin 7) connected to PA2
// back light    (LED, pin 8) not connected, 4 white LEDs which draw ~80mA total

//********Nokia5110_Init*****************
// Initialize Nokia 5110 48x84 LCD by sending the proper
// commands to the PCD8544 driver.
// inputs: none
// outputs: none
// assumes: system clock rate of 50 MHz or less
void Nokia5110_Init(void);

//********Nokia5110_OutChar*****************
// Print a character to the Nokia 5110 48x84 LCD.  The
// character will be printed at the current cursor position,
// the cursor will automatically be updated, and it will
// wrap to the next row or back to the top if necessary.
// One blank column of pixels will be printed on either side
// of the character for readability.  Since characters are 8
// pixels tall and 5 pixels wide, 12 characters fit per row,
// and there are six rows.
// inputs: data  character to print
// outputs: none
// assumes: LCD is in default horizontal addressing mode (V = 0)
void Nokia5110_OutChar(unsigned char data);

//********Nokia5110_OutString*****************
// Print a string of characters to the Nokia 5110 48x84 LCD.
// The string will automatically wrap, so padding spaces may
// be needed to make the output look optimal.
// inputs: ptr  pointer to NULL-terminated ASCII string
// outputs: none
```

```
// assumes: LCD is in default horizontal addressing mode (V = 0)
void Nokia5110_OutString(char *ptr);

//********Nokia5110_OutUDec*****************
// Output a 16-bit number in unsigned decimal format with a
// fixed size of five right-justified digits of output.
// Inputs: n  16-bit unsigned number
// Outputs: none
// assumes: LCD is in default horizontal addressing mode (V = 0)
void Nokia5110_OutUDec(unsigned short n);

//********Nokia5110_SetCursor*****************
// Move the cursor to the desired X- and Y-position.  The
// next character will be printed here.  X=0 is the leftmost
// column.  Y=0 is the top row.
// inputs: newX  new X-position of the cursor (0<=newX<=11)
//         newY  new Y-position of the cursor (0<=newY<=5)
// outputs: none
void Nokia5110_SetCursor(unsigned char newX, unsigned char newY);

//********Nokia5110_Clear*****************
// Clear the LCD by writing zeros to the entire screen and
// reset the cursor to (0,0) (top left corner of screen).
// inputs: none
// outputs: none
void Nokia5110_Clear(void);
```

Help

Program 11.9. Wiring connections to the Nokia5110 LCD display and high-level software functions prototypes (C11_Nokia5110).

For more information about SSI, see Section 8.3 in Embedded Systems: Introduction to ARM Cortex-M Microcontrollers, 2013, ISBN: 978-1477508992, **http://users.ece.utexas.edu/~valvano/arm/outline1.htm** (http://users.ece.utexas.edu/%7Evalvano/arm/outline1.htm) or from Section 7.5 in Embedded Systems: Real-Time Interfacing to Arm® Cortex™-M Microcontrollers, 2013, ISBN: 978-1463590154, http://users.ece.utexas.edu/~valvano/arm/outline.htm (http://users.ece.utexas.edu/%7Evalvano/arm/outline.htm)

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.

(http://www.facebook.com/EdxOnline)

(https://twitter.com/edXOnline)

(https://plus.google.com/108235383044095082735/posts)

(http://youtube.com/user/edxonline)

© 2014 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (https://www.edx.org/edx-privacy-policy)

Help