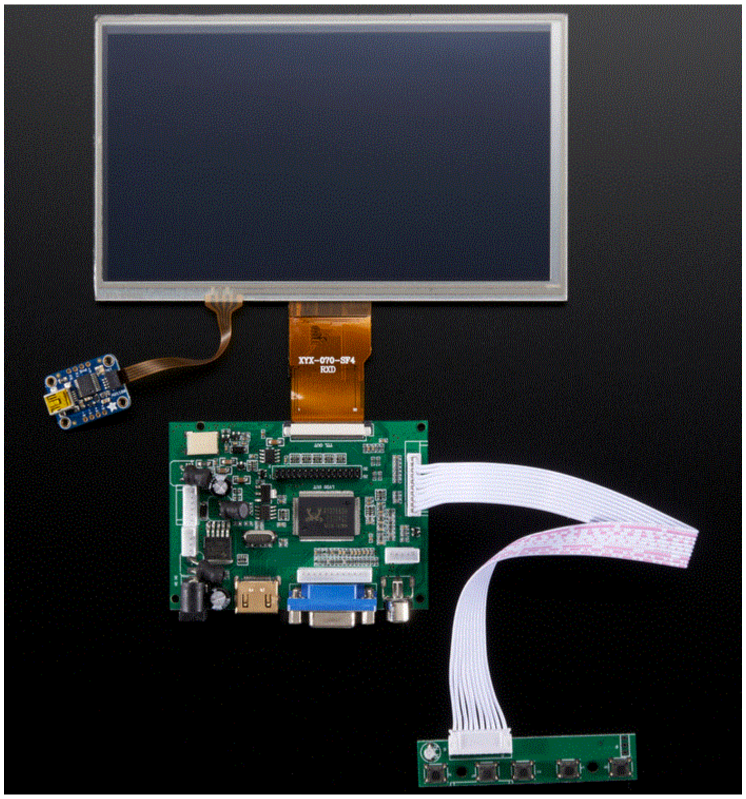
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF BRITISH COLUMBIA
CPEN 391 – Computer Systems Design Studio
Fall 2015/2016 Term 2

Exercise 1.4
Adding a Touch Screen Display

Introduction

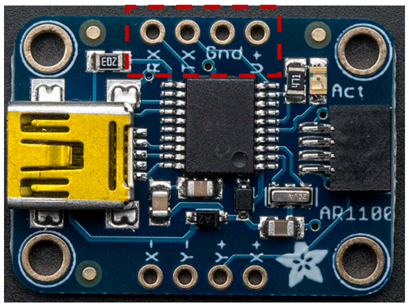
You will be given a small 7 inch TFT touch screen display by your Instructor see below. This has a VGA input which we will eventually connect to a graphics controller on the DE2, but for now we are going to focus on the resistive touchscreen interface. **Marking**: When you have completed this exercise, show it to the TA for marking.

NOTE: Under no circumstances should you use any sharp instrument such as pen/pencil, piece of wire etc. on the screen. Use only your finger. If you mark/break the display you will be asked to pay for it by the Dept. They are approx. $90 from Adafruit - part number 2395 (if you want to buy your own after the course).



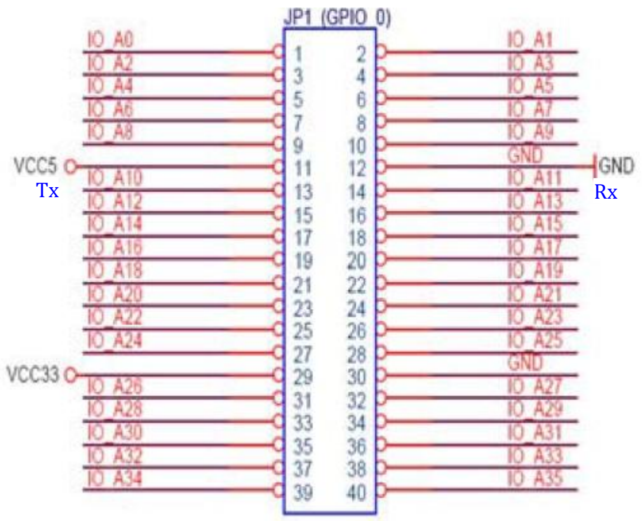
The resistive touchscreen controller circuit is shown above to the left of the main circuit board. You can see a 4 wire flexible resistive connector coming from the screen to the controller.

The touch screen controller is shown in close up below. It has a USB interface (*for easy connection to PCs etc.*), but it also has a simple serial connection along the top of the circuit board (*see the signals labeled Rx/Tx and also Gnd and V+ 5 volts*) for use with small embedded microcontrollers. The controller chip itself - the AR1100 is made by Microchip and the data sheet for it can be found on Connect.



You will have to solder "pin strip" connector (given to you in class) to the above board if it does not already one present. This will allow us to "insert" the controller into a breadboard via the long pins sticking out through the bottom of the module, while at the same time, attaching "jumper wires" to the top of the pins. We will then connect the other end of the wires to the GPIO_0 port on the DE2 (*check out the connections in the Quartus Pin Planner for the Touch Screen serial ports we added in exercise 1.3*). Both 5v and ground for this chip can be picked up directly from the GPIO_0 connector via Pins 11 and 12 (see below).

Unless you have changed anything, the touch screen serial Port signal TxData is wired in the pin planner to PIN_N18 and RxData is connected to PIN_P18 (verify this for yourself). These should be the pins immediately below the 5 volt and ground connections shown below.



You'll be given the special jumper wires to make it easy to connect the DE2 to the AR1100 controller board - make sure you use these.

IMPORTANT: The Tx output shown on the GPIO_0 connector above should be connected to the Rx input on the AR1100 controller. Likewise the Rx input on GPIO_0 above should connect to the Tx output on the AR1100. That is, Tx and Rx should be **crossed over**. This is common in serial ports, since Tx and Rx are always defined relative to the board where the connection occurs, i.e. both DE2 and AR1100 are both transmitters and receivers of data. By analogy the mouth piece of your telephone (the transmitter) is connected to the earpiece (the receiver) of the other persons phone and vice versa.

Using the Controller

Study the data sheet for the AR1100 in particular

- Section 4 - Communication, baud rate, parity, stop bits etc.
- Section 5 - Commands in particular touch enable/disable
- Section 7 - Operation

The touchscreen should have been calibrated (*at least roughly*) in the factory and it should "just work" out of the box. Microchip provide a Windows based utility to calibrate the screen via the USB port and your PC - Your instructor can help with this if this is needed. Top left of the touchscreen should be coordinate [0,0] unless it has been calibrated upside down. But depending upon up-down, left/right offset of the overlay on the screen you may need to "map" the coords to a pixel in software.

There is a little RED Led on the controller that flashes at 1 sec intervals when the touchscreen is *inactive* and blinks wildly when you touch the screen.

USB or Serial Port
The AR1100 chip will decide whether to use its USB or serial port based on which of those two ports sends data to it first. That means you should send it a **TouchScreen_Enable** command via the serial port as part of your initialization of the serial port to help it make the decision - **don't** connect a USB cable if you are using the serial port.

Each command will generate a respons. It's up to you to decide if this is worth processing or important. When you touch the screen, it will send and continue to send a PEN_DOWN data packet - see page 15 of the AR1100 data sheet. The first byte of this packet is the character hex 80. It will also send a 12 bit X and a 12 bit Y coordinate in the next 4 bytes i.e. a resolution of 4096 points/positions vertically and horizontally. You will have to map these coords to a pixel on your screen in software. **Note** this controller does not cope with multi-touch (e.g. two finger gestures), it's designed only to mimic a mouse with a button. If you finger moves without being released, new X,Y coords will be sent with each packet

A PEN_UP data packet is sent once when you take your finger off the screen. This data packet begins with hex 81 and also sends the X,Y coord of your finger at the point of release (useful for dragging etc). Complete the exercise as follows.

- Wire the controller to the DE2 GPIO_0 connector
- Using the C code functions we wrote for the Serial port in Exercise 1.3 as a template, write equivalent routines to drive the touchscreen (#define new pointers to the Touch screen 6850 serial port).
- Add the following functions and test them with some harness code.

```c
/*****************************************************************************
**  Initialise touch screen controller
*****************************************************************************/
void Init_Touch(void)
{
    // Program 6850 and baud rate generator to communicate with touchscreen

    // send touchscreen controller an "enable touch" command
}

/*****************************************************************************
**   test if screen touched
*****************************************************************************/
int ScreenTouched( void )
{
    // return TRUE if any data received from 6850 connected to touchscreen
    // or FALSE otherwise
}

/*****************************************************************************
**   wait for screen to be touched
*****************************************************************************/
void WaitForTouch()
{
      while(!ScreenTouched())
             ;
}

/* a data type to hold a point/coord */

typedef struct { int x, y; } Point ;

/*****************************************************************************
* This function waits for a touch screen press event and returns X,Y coord
*****************************************************************************/
Point GetPress(void)
{
    Point p1;

    // wait for a pen down command then return the X,Y coord of the point
    // calibrated correctly so that it maps to a pixel on screen

    return p1;
}

/*****************************************************************************
* This function waits for a touch screen release event and returns X,Y coord
*****************************************************************************/
Point GetRelease(void)
{
    Point p1;

    // wait for a pen up command then return the X,Y coord of the point
    // calibrated correctly so that it maps to a pixel on screen

    return p1;
}
```