AN0901: Interfacing a SPLat SL100 Controller to a Maple Systems HMI520T Touch Screen

Rev 1.0, 2009 01 28

Disclaimer.

The information in this application note is provided as is. We do not provide technical support for 3rd party products. Support plans are available for SPLat products.

Introduction

The purpose of this application note is to present a fully worked example of setting up a SPLat controller to work with a 3rd party touch screen HMI. You should be able to use the information and code provides as the basis for your own implementation.

The Maple Systems Silver series appears to be the best value for money example of a touch screen interface that we have found on the web. This application note describes how to connect a Maple Systems HMI520T touch screen controller to a SPLat control board. The SPLat control board used for this demonstration was the SL100, but the application can be modified to work with any SPLat controller with ModBus master enabled firmware.

In this example, the touch screen is used in ModBus slave mode with the SPLat controller being the ModBus master. We were not able to make the HMI520T work as a ModBus master.

What You Will Need

Apart from a touch screen, Maple Systems model HMI520T, and a SPLat Controls SL100 controller, you are going to need a regulated 24 VDC power supply capable of at least 0.3 Amps. You will also need to buy/ make up an RS232 serial communications connection cable to connect the HMI to the SL100. This cable is made up of a male DB9 connector to male DB9 connector with RX and TX "crossed over". No hand shaking is used and so only 3 wires need to be connected:

HMI520T DB9 male connector pin	SPLat SL100 DB9 male connector pin
2 (TX)	3 (RX)
3 (RX)	2 (TX)
5 (Common)	5 (Common)

A "null modem" cable should also work. Note that RS232 is only suitable for relatively short cable runs of less than 50 feet (15 metres), depending on the electrical environment. If you make your own cable, test it with a multimeter. The cable is plugged into the "PLC[RS232]" connector on the back of the HMI520T and connects to the SL100 on board DB9 connector.

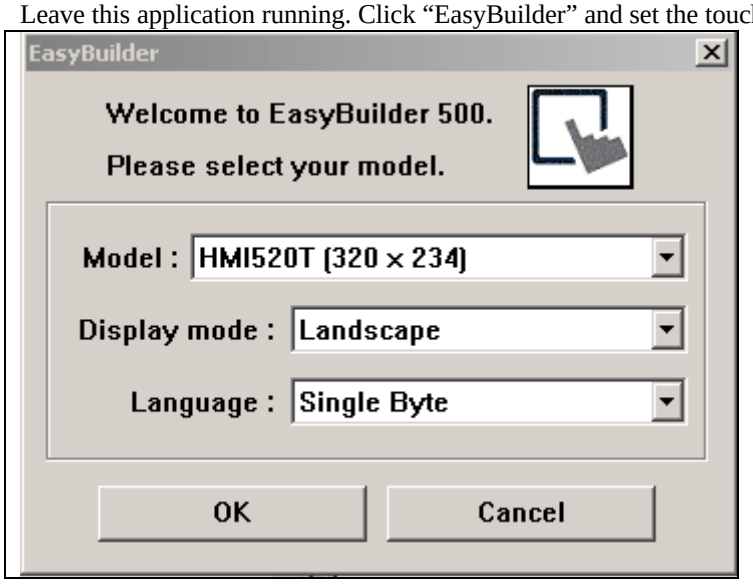
You will also need the HMI520T programming cable. It has three DB9 connectors. The connector labelled "PC" should be connected to the a RS232 comm port on your PC. The connector labelled "HMI" should be connected to the HMI520T "PLC[RS485] PC [RS232] port. The connector marked "PLC" is not used in this example.

Touch Screen Setup

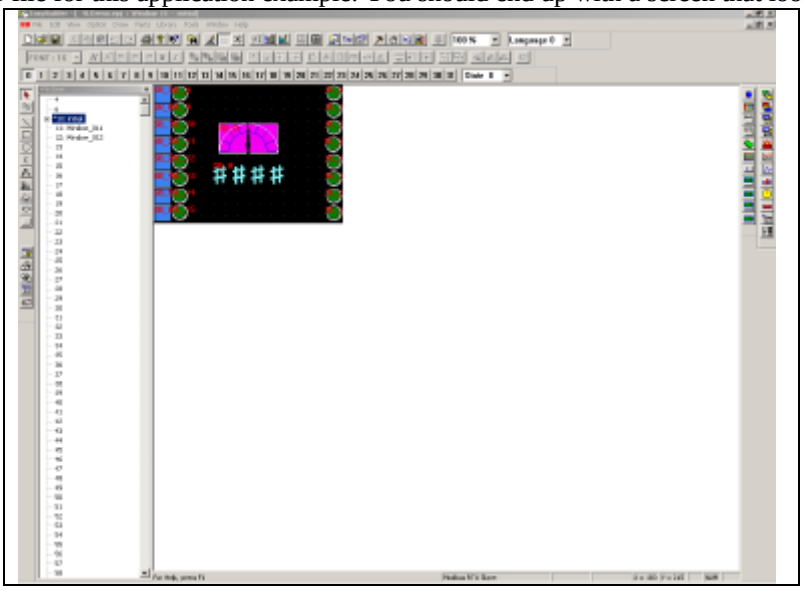
You will need to install the Maple Systems Silver Series Ezware-500 software on your computer. Follow the Maple Systems documentation to achieve this. It is beyond the scope of this document to give a tutorial on how to use the software or how to design your own screen layouts. Please refer to the Maple Systems documentation.

Once installed, you will need to launch the "EasyManager" and choose a suitable comm port on your computer.

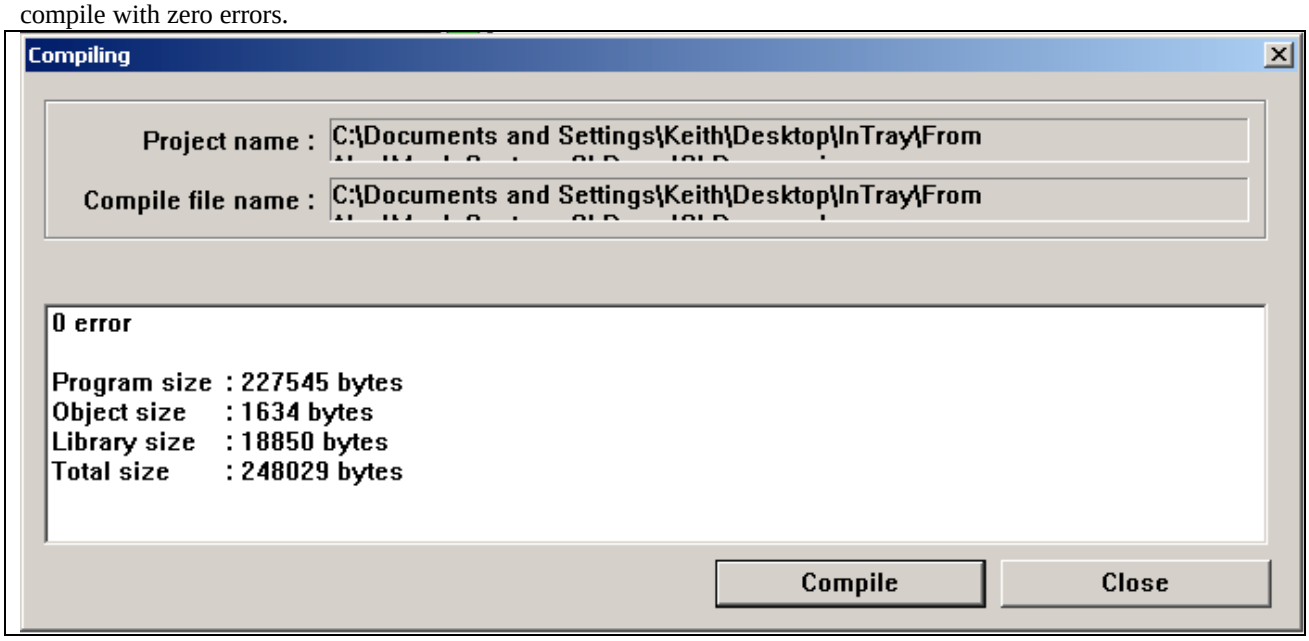
Leave this application running. Click "EasyBuilder" and set the touch screen model, aspect and byte mode:



Click OK to open the environment. Now using the file menu, open the file "SLDemo.epj", which is the screen layout and behaviour file for this application example. You should end up with a screen that looks like this:

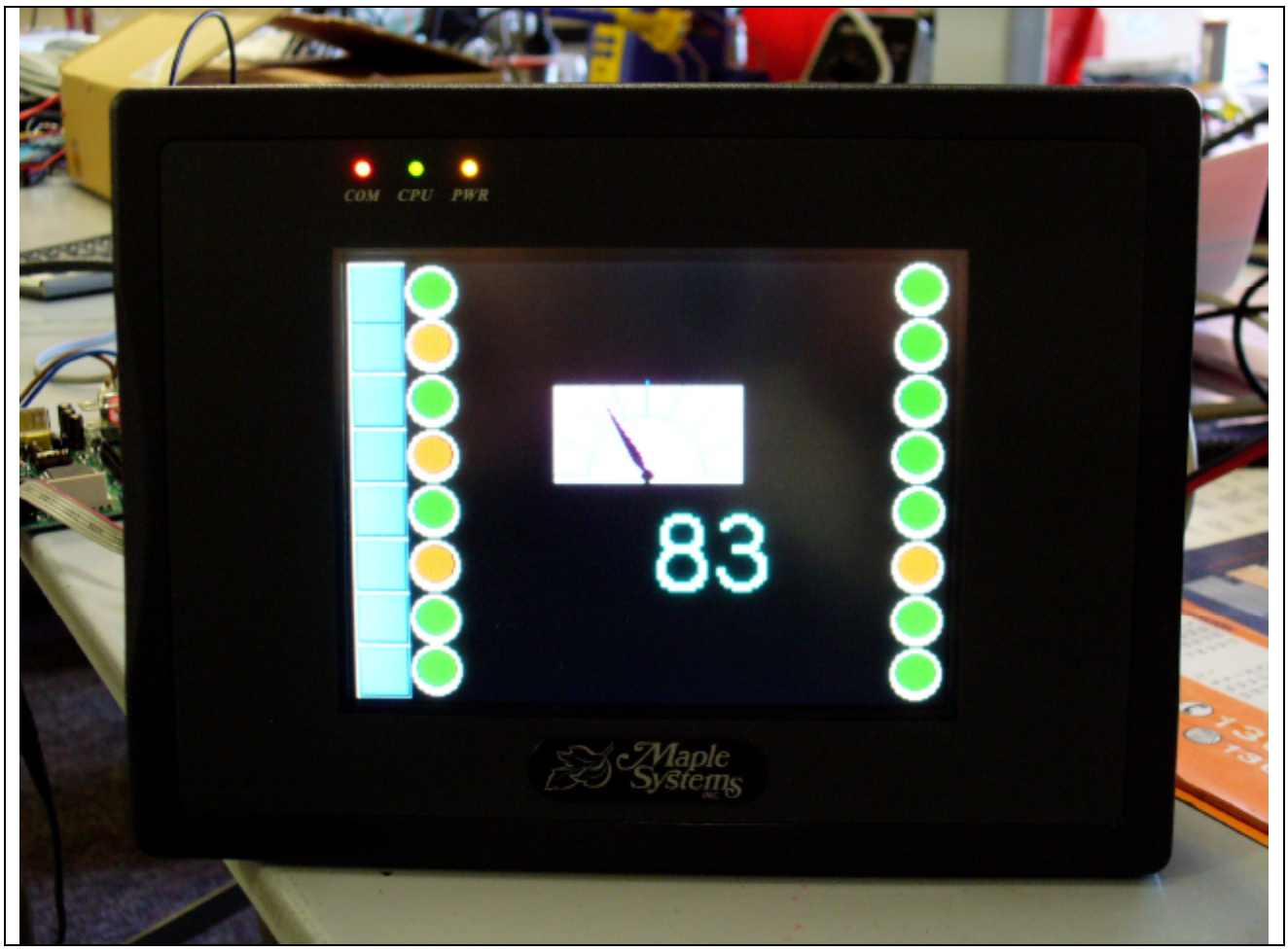


Now select “Compile” from the Tools menu and click the COMPILE in the new window that appears. The file should compile with zero errors.



Now click the CLOSE button. Again from the tools menu, select “DOWNLOAD”. This will down load the compiled file to the HM520T. This will take about 40 seconds in total and while it is happening, the HMI will have green writing scrolling up the screen as the progress bar progresses on the PC. AT the end of the down load click “OK” on the “EasyDownload” “Mission Complete” window.

To set the program running in the HMI, click the “Jump To Application” button in the “EasyManager” window. Alternately, cycle the 24 VDC power to the HMI. When either of these are done, the HMI screen will look like this:



Note that the numeric value and position of the panel meter display depends on the analog value being sent from the SL100. With no SL100 connected or running it will read zero.

Touch Screen Example Layout

The eight virtual push buttons on the left-hand side of the screen have virtual LEDs next to them which indicate whether the push buttons are in the on or off state. That is to say, their state toggles with each press. These are mapped to the eight outputs on the SL100, so when you press one of the buttons on the HMI screen, the corresponding SL100output should respond..

On the right hand side of the screen are eight LEDs that indicate the state of the eight inputs on the SL100 controller.

In the centre is a numeric value coupled to an analog meter movement which reads the 8 bit analog value applied to the SL100's analog input.

SL100 Set up

The SL100 used must have a firmware capable of ModBus master operation. Version 3.16 or higher firmware must be loaded into the SL100. Open, translate and transfer the SPLat program "Maple1.spt" to the SL100. Set the program running. Pressing buttons on the HMI520T should now toggle the outputs on the SL100. The input state of the SL100 should be mimicked on the HMI520T, while the analog reading on the SL100 analog input should be displayed both numerically and by the position of the HMI analog meter movement.

SL100 Program Notes

The SL100 is the ModBus master and the HMI520T is the ModBus slave. The SL100 program consists of two tasks. The first basically maps the SL100 input status to a memory location and outputs the contents of another memory location to the SL100 outputs. The analog value read form the SL100's analog port is also stored into memory. The ModBus master script basically maps these memory locations to registers within the HMI520T. The following explanation of the SL100 program assumes that the user is somewhat familiar with the ModBus protocol and has read the SPLat Knowledge Base with regards to ModBus master operation.

SL100 Inputs:
The I/O task reads the state of the eight SL100 inputs in one go using the "InputFM" instruction and the result is stored into "InputData" (memory location 0). The "InputFM" reads 8 bits starting at the supplied argument.

```
inputfm     0           ;get latest input data
store       InputData
```

To send the input data to the HMI, we must use the ModBus "Force Single Coil" (Fn 5) function as the HMI does not support the "Force Multiple Coils" (Fn 15) function. So in the ModBus master script, we must send each bit of the input status to the HMI separately. This is why there are eight entries into the ModBus master script to achieve this:

```
MMscr  nv0byte   5,SlaveAddr,#0,#0    ;Fn 5 - force single coil, SPLat semaphore 0 --> coil 1 in slave
       nv0byte   5,SlaveAddr,#1,#1    ;Fn 5 - force single coil, SPLat semaphore 1 --> coil 2 in slave
       nv0byte   5,SlaveAddr,#2,#2    ;Fn 5 - force single coil, SPLat semaphore 2 --> coil 3 in slave
       nv0byte   5,SlaveAddr,#3,#3    ;Fn 5 - force single coil, SPLat semaphore 3 --> coil 4 in slave
       nv0byte   5,SlaveAddr,#4,#4    ;Fn 5 - force single coil, SPLat semaphore 4 --> coil 5 in slave
       nv0byte   5,SlaveAddr,#5,#5    ;Fn 5 - force single coil, SPLat semaphore 5 --> coil 6 in slave
       nv0byte   5,SlaveAddr,#6,#6    ;Fn 5 - force single coil, SPLat semaphore 6 --> coil 7 in slave
       nv0byte   5,SlaveAddr,#7,#7    ;Fn 5 - force single coil, SPLat semaphore 7 --> coil 8 in slave
```

SL100 Output:
The I/O task take the value stored in memory location "OutputData" (memory location 10) and updates the SL100 eight outputs:

```
recall      OutputData  ;update SPLat outputs
loadx       255
outputm     0
```

The SL100 ModBus master script reads the data from the HMI using a read multiple coils (Fn 1) function:

```
nv0byte   1,SlaveAddr,#8,8,#80 ;Fn 1 - read multiple coils, slave coils 8 --> 15 maps to SPLat byte
                               ;at RAM location 10
```

Note that to get the data to appear into memory location 10, the SPaddr16 parameter must be 80 or 10x 8 semaphores/byte.

SL100 Analog
The I/O task reads the analog input in 8 bits and stores the data into "AnData" (memory location 20).

```
aninA                   ;get latest analogue data
store       AnData
```

This is sent to the HMI controller using the ModBus “Write Holding Register” (Fn 16) function.

```
nv0byte   16,SlaveAddr,#0,1,#20,0     ;Fn 16 - write holding register, SPLat RAM 20 --> holding register 0
                                      ;in slave
```

Using other Maple Systems HMI models

We have not tried other models of Maple Systems’ HMIs. The Ezware-500 code samples are based on the particular screen pixel dimensions of the HMI520T. Other models have different sized screens and would need a change to the screen layout design, in the very least. Even if you want to use a different model of Maple Systems Silver Series HMI, we strongly suggest you get the examples working on a HMI520T before moving over to a different model. It is always best to develop something incrementally, starting from a known good base, extending it in the direction you want to go one small step at a time.

Using other SPLat Controllers

You can use this technique with any SPLat current model (as of January 2009) controller that supports ModBus Master mode. For those models of SPLat that lack an RS232 compatible programming port (i.e. MS120 or CC18) you can use a programming cable as an RS232 adaptor.

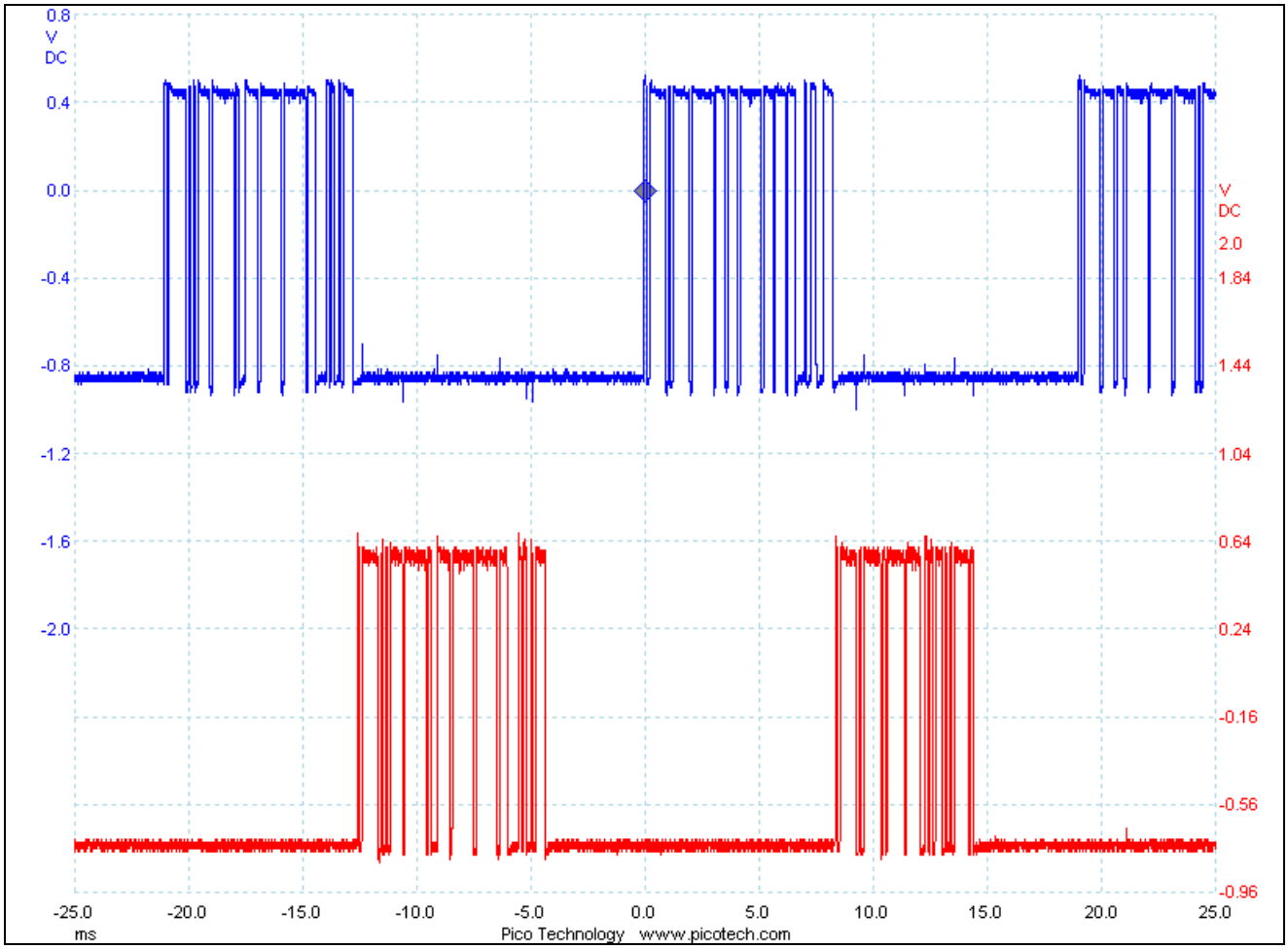
The controller must have firmware that supports ModBus Master mode. For details refer to your controller’s firmware release information in the SPLat Knowledge Base.

Issues

The ModBus communications is done at 9600 baud. The Maple Systems HMI520T replies to the ModBus master poll almost immediately (typically ~170us). This causes two problems:

1. Faster baud rates are not possible because the SL100 does not begin receiving immediately after sending a poll and
2. RS485 interfaces do not have enough time to turn the line around (for long cable runs).

Consequently, 9600 baud is as fast as we have been able to run. The SL100 always honours the 3.5 character delay times as set out in the ModBus specification.



The top trace is the SL100 ModBus master poll. The bottom trace is the HMI520T response. As you can see, the HMI520T does not honour the 3.5 character time gap before replying.

We were not able to make the HMI520T run as a ModBus master either.