

UI4x20

Product Overview

The UI4x20 has been designed to allow SPLat controller with or without an LCD to have a remote user interface. Once set up, the LCD on the UI4x20 is seamlessly controlled by using any of the standard LCD instructions. The UI4x20 has the following features:

- Fully programmable SPLat controller with all the latest, relevant SPLat features. An entire application can run in the UI4x20 so it can be part of an intelligent distributed system
- Connects to other SPLat controllers via Xwire and allows any SPLat controller to have a remote user interface (LCD, buttons and LEDs) regardless of whether the controller has an LCD or not
- Parts or all of the LCD can be controlled by the controller connected to the UI4x20
- 8 --> 24 VDC operation, 100mA maximum at 24 VDC
- 4 x 20 alpha numeric LCD with dimmable back light
- 24 pulse/rev rotary quadrature encoder (clockwise increments, anti-clockwise decrements, 4 pulses per detent)
- 4 push buttons
- 4 bi color LEDs (red/green)
- Beeper
- RS485 Xwire interface (allows for very long cable runs of up to 1.2km, 3900 feet)
- TTL Xwire interface for short cable runs of less than 2 metres
- TTL communications interface (for program download, MODBUS or user defined comms)
- Connects to any SPLat controller with using Xwire (older boards may need to be reFlashed to the latest version of firmware)

LCD Magic

SPLat has a bunch of LCD instructions, like:

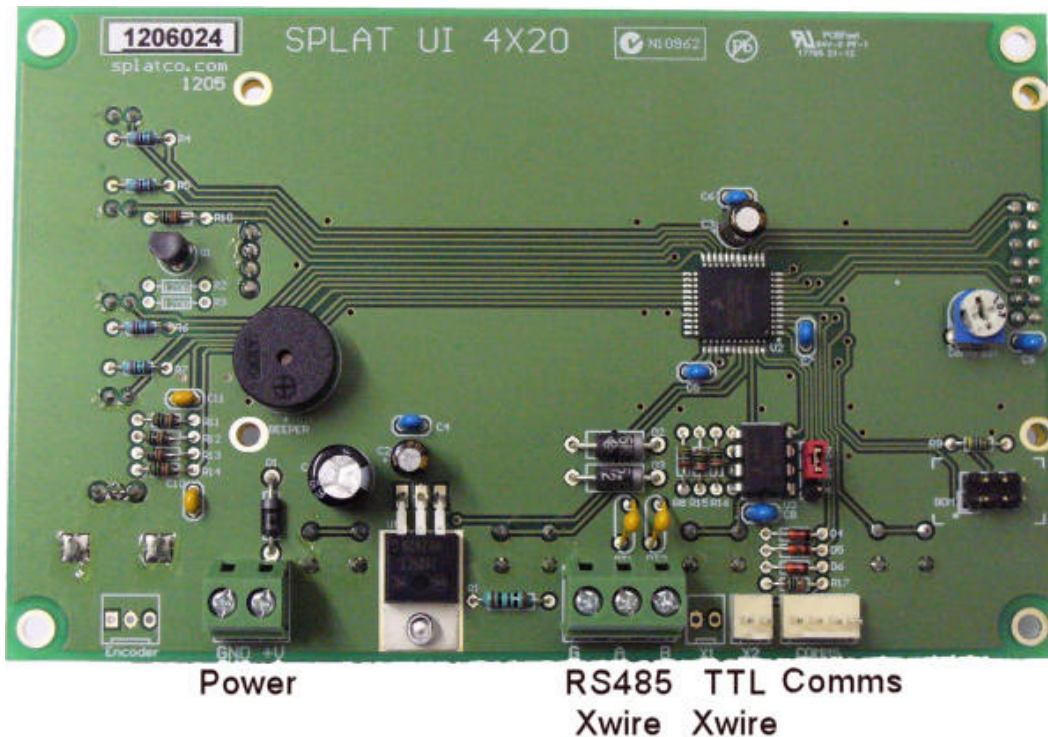
- `OBLCD_Text "Hello World."`
displays a text message on the screen
- `OBLCD_fDispW 3,1`
displays W as a real number in the format 'nnn.n'
- `OBLCD_SetCur 1,0`
moves the cursor to the start of the second line, this is where the next characters will be displayed.

Normally these instructions are only used in controllers with a LCD, like the MMI202 and MS120, however they are actually usable on all current SPLat controllers, with or without an LCD. So for example, you can add a user interface to our inexpensive CC18 controller.

The SPLat Xwire bus allows the OBLCD instructions to work remotely. Once setup, no special handling is required. Whence saying `OBLCD_Text "Hello World."` will make the message appear on the UI4x20. The Xwire chapter shows how to enable the remote LCD feature.

Connections

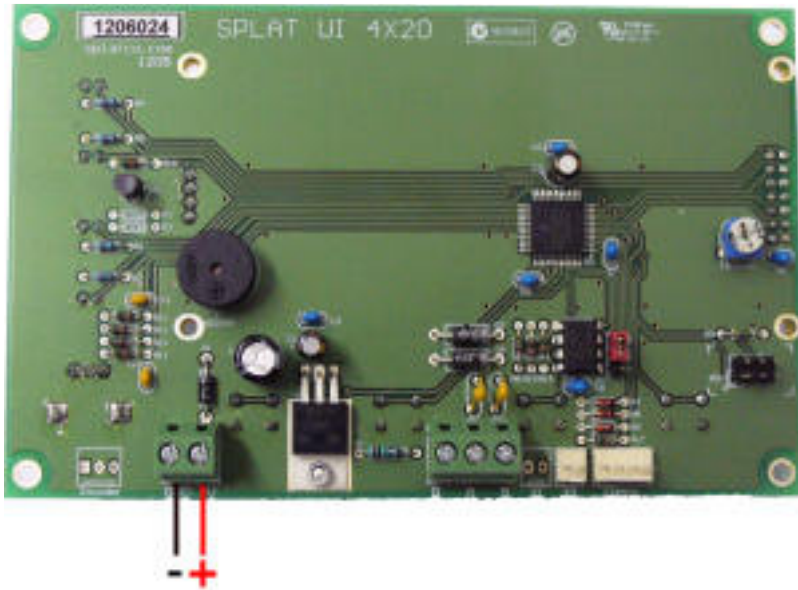
The connectors on the UI4x20 are shown and labeled in the picture below.



- **Power**
This is a 2 way screw terminal block for supplying power to the UI4x20. Refer to the Power Supply section for more details.
- **RS485 Xwire**
This is the connection for the Xwire bus using RS485 signaling. This interface should be used in preference to the TTL Xwire interface if the UI4x20 is located more than two metres from the master controller, or if the wiring is run in an electrically noisy cabinet. The RS485 interface will work over a distance of 1.2km.
- **TTL Xwire**
This is the low voltage Xwire bus connection. Both X1 and X2 are the same, having two terminals simplifies multi-drop wiring. These terminals should be used only when the distance to other Xwire devices is within a couple of metres or so.
- **Comms**
This connector is used for programming the UI4x20. The COMMS connector works at 5V CMOS/TTL levels so a PC232 or PCUSB programming cable must be used to connect to a PC. The comms connector may also be used for communicating to other serial devices, but note a level convertor may be required. This port is also capable of MODBUS RTU slave or master.

Power Supply

The power supply is connected to the terminals as shown below. Take care to ensure the polarity is correct.



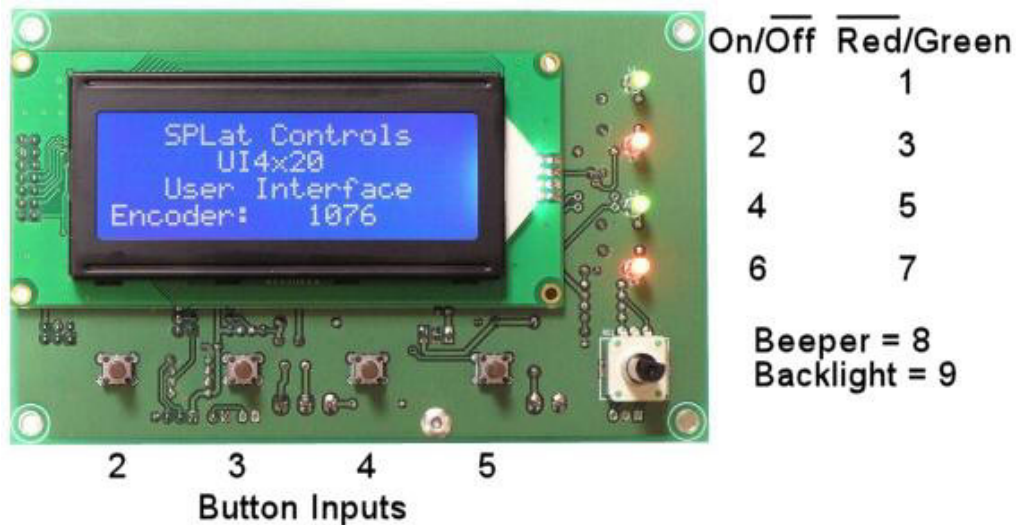
In general, the UI4x20 needs a supply voltage between 8VDC and 24VDC and will draw a maximum of 100mA, less if the LEDs, beeper or backlight are off. The voltage is must not exceed 32V.

External Inputs and Outputs

The UI4x20 has no digital or analogue I/O. We made it this way to keep the cost low. If you need built in I/O with a operator interface, have a look at some of our other controllers such as the MMI202 or MS120, or use the UI4x20 with a cost effective controller such as the CC18.

Operator Interface

The digital I/O map for the buttons and LEDs is shown in the picture below.



LEDs

Each LED has two associated outputs. Outputs 1, 3, 5 and 7 select the colour of the LED where low (aka off) selects red and high (aka on) selects green. Outputs 0, 2, 4 and 6 turns the corresponding LED on and off. The colour may be selected either before or after the LED has been turned on. Here's an example that turns on top LED green, and alternates the second LED between red & green.

```
On          1      ;select top LED green
On          0      ;turn on the top LED
Blink       3      ;alternate red/green for the second LED
On          2      ;second LED on
```

Beeper

The beeper may be activated via output 8. In addition, KBeep can be used to provide a brief beep. InputK instructions will provide key press beep feedback.

Backlight

The backlight may be controlled via output 9. However the OBLCD_Dim instruction provides a better automatic way to control the backlight brightness. Be aware the backlight has a limited lifespan (10,000 hours) and dimming or turning it off completely when not in use will increase its operating life.

Buttons

The 4 push button input numbers are shown in the diagram.

Rotary Dial

The rotary dial is very useful for navigating menu options and changing values quickly. It may be read via the Onboard Quadrature Counter, refer to the knowledge base for more information.

Here's a simple application that counts up or down as the dial is turned:

```
OBLCD_Dim    1,3,1    ;configure LCD dimming
OBQC_Clr     0        ;clear the OBQC count
Loop
OBQC_fGet    0        ;get the current dial count
fWtoQ        ;put the count in Q
fLoadW       4.0      ;load the divisor
fDiv         ;now the count is 1 count per indent
OBLCD_SetCur 0,0     ;position text at the top left of the LCD
OBLCD_fDispW 5,0     ;display the dial count
Goto        Loop     ;do it all again
```

In the above example, each “click” of the dial is equal to 4 counts. The dial has 24 positions so one full turn of the dial is 96 counts. However the code above changes this so each “click” is a single count, so now a full revolution is 24 counts.

The encoder signals are also available on inputs 0 and 1.

Xwire

The Xwire bus is the primary means of exchanging information between the operator of the UI4x20 and other controllers.

Selecting the Xwire or RS485 interface

A 3-pin jumper on the board is labeled 'RS485 X'. Inserting a jumper in the two pins labeled RS485 will enable the RS485 interface. Inserting a jumper in the two pins labeled X will enable the TTL Xwire electrical interface. TTL Xwire should only be used if the distance no more than a couple metres while RS485 is good for distances of up to 1.2km. RS485 is also applicable in noisy electrical environments even if the cable run is short.

Connecting to the Xwire interface

Two connectors on one long side of the board are labeled X1 and X2. These are the TTL Xwire electrical connections. The two connectors are wired in parallel on the board to facilitate daisy chaining, and are completely interchangeable. The left-hand pin of each connector is ground (0V). If you are wiring to other boards with X1/X2 connectors, simply wire like pins together. If wiring to MMi202 or SL100, see the Xwire connection instructions in their documentation. See also wiring in the Xwire documentation.

Connecting to the RS485 interface

The 3-pin screw terminal block is for RS485 connection. The A and B terminals should go to A and B on all other devices on the RS485 system.

The G terminals on all boards in a system should all be connected together. For best results, especially over long distances and/or where the devices each end are powered off different power supplies, use shielded twisted pair cable, and use the shield for the G connection.

Programming

The UI4x20 may be configured as either a Xwire master or an Xwire slave, however configuring as an Xwire slave has an advantage of allowing another controller to simply display information on the UI4x20 LCD with the normal OBLCD instructions.

The Xwire master needs some specific entries in its Xwire table to drive the UI4x20 LCD. The format is as follows:

```
NV0BYTE 249, StartCharacterOffset, NumberOfCharacters, 0, 0
```

Where:

- **249**: This is the special address reserved for all remote LCDs. If there are one or more UI4x20 controllers on the Xwire bus, they will all display the message.
- **StartCharacterOffset**: This is the offset to the first character that should be displayed on the UI4x20. 0 is the very first character on the first line. 20 is the first character on the second line.
- **NumberOfCharacters**: The number of characters to display where the maximum is 32. As the LCD has 80 characters, multiple Xwire table entries will be required to send the whole screen worth of characters.
- **0, 0**: Nothing is sent back, ID 249 is only used for sending LCD characters.

As Xwire messages are limited to 32 bytes, at least 3 Xwire table entries will be required for the whole LCD to be addressed in the UI4x20, for example:

```
NV0Byte      249,0,32,0,0    ;LCD chars 0 thru 31
NV0Byte      249,32,32,0,0   ;LCD chars 32 thru 63
NV0Byte      249,64,20,0,0   ;LCD chars 64 thru 79 + 4 special bytes
```

The combination of StartCharacterOffset and NumberOfCharacters means the master controller can reserve part of the LCD screen. The other part of the LCD can be written directly by the UI4x20.

So, say you only want the master controller to write to the last line of the LCD and leaving the first 3 lines under control of the UI4x20, this is what the Xwire table would contain:

```
NV0Byte      249,60,20,0,0   ;Last line of the LCD chars 60 thru 79
```

There are special bytes at the end of the 80 byte LCD buffer which map LCD features such as cursor type and control etc:

- 80 - LCD type index, sets slave LCD type to masters LCD type
- 81 - LCD cursor on/off, blinking status
- 82 - LCD cursor row address
- 83 - LCD cursor column address

These are optional and only required if you want to display a blinking cursor etc.

Example

The LEDs, buttons and rotary dial require more effort to use. They may be acted upon by a program running in the UI4x20 or simply echoed to the master controller.

For the sake of brevity, this example shows how a CC18 running as an Xwire master may use the UI4x20 as a dumb operator interface:

- The rotary dial position is reported to the CC18 which converts it to 1 count per indent and displays it back on the UI4x20.
- Each button is reported to the CC18 that then lights a corresponding LED.

Common to both boards

This is the Xwire memory used for the Xwire buffers that are sent between both boards.

```
defBLOCK 1
bLEds          defBYTE      ;all the LEDs are controlled by a single bit
mapped byte
sLED0On        EQU 0        ;LED on bit in the bLEds byte
sLED0Green     EQU 1        ;LED colour bit in the bLEds byte
sLED1On        EQU 2        ;LED on bit in the bLEds byte
sLED1Green     EQU 3        ;LED colour bit in the bLEds byte
sLED2On        EQU 4        ;LED on bit in the bLEds byte
sLED2Green     EQU 5        ;LED colour bit in the bLEds byte
sLED3On        EQU 6        ;LED on bit in the bLEds byte
sLED3Green     EQU 7        ;LED colour bit in the bLEds byte

defBLOCK 5     ; 1 byte + 1 float = 5 bytes
bButtons       defBYTE      ;all the buttons are reported in a single bit
mapped byte
sButton0       EQU 0        ;button bit in the bButton byte
sButton1       EQU 1        ;button bit in the bButton byte
sButton2       EQU 2        ;button bit in the bButton byte
sButton3       EQU 3        ;button bit in the bButton byte
fCounter       defFLOAT     ;the counter is reported as a float
```

UI4x20 code

```
; Start of code, first some initialisation
OBLCD_Dim      1,3,1        ;configure LCD dimming
OBQC_Clr       0            ;clear the OBQC count
LoadX          0            ;we'll use Xwire address 0 for buttons and LEDs
XwireSetAddr   ;set our address
XwireSlave     pXwireSlave ;start running as an Xwire slave

Loop
Recall         bLEds        ;get the LED values
LoadX          %11111111    ;load the output mask (all 8 bits)
OutputM        0            ;drive the LEDs

InputFM        2            ;read all the buttons
Store          bButtons     ;place in the Xwire buffer

OBQC_fGet      0            ;get the current dial count
fStore         fCounter     ;place in the Xwire buffer
Goto          Loop         ;do it all again

; Start of nonvolatile memory
NVEM0
pXwireSlave    NV0Byte bButtons,5,bLEds,1
```

CC18 example code

```
; Start of code, first some initialisation
OBLCD_type      5                ;select 4 x 20 character display format
XwireMaster     pXwireMaster    ;start running as an Xwire slave

Loop
OBLCD_SetCur   0,0              ;position text at the top left of the LCD
fRecallQ        fCounter        ;get the dial position
fLoadW          4.0              ;divide by 4
fDiv            ;now the count is 1 count per indent
OBLCD_fDispW    5,0             ;display the dial count

;light each LED while the corresponding button is pressed
Recalls         sButton0,bButtons
Stores          sLED0On,bLEDs
Recalls         sButton1,bButtons
Stores          sLED1On,bLEDs
Recalls         sButton2,bButtons
Stores          sLED2On,bLEDs
Recalls         sButton3,bButtons
Stores          sLED3On,bLEDs

Goto            Loop            ;loop over and do it again

; Start of nonvolatile memory
NVEM0
pXwireMaster
NV0Byte         249,0,32,0,0      ;LCD chars 0 thru 31
NV0Byte         249,32,32,0,0     ;LCD chars 32 thru 63
NV0Byte         249,64,20,0,0     ;LCD chars 64 thru 79, control bytes
NV0Byte         0,bLEDs,1,bButtons,5 ;buttons and LEDs
NV0Byte         255              ;end of table
```

