UM Unified Microsystems

ATS-1 User's Manual

Revision 1.03 April 2012

Overview

The ATS-1 is an Arduino™ compatible shield that acts as a small terminal. Long time computer developers who used dumb terminals will be familiar with this concept. The ATS-1 features a 16X2 LCD, 6 navigation switches, user controllable buzzer and LED, and a master reset switch. It plugs into Arduino Uno, Duemilanove, and other CPU modules with the same foot print. The ATS-1 communicates with the host through only two serial lines, leaving the rest of the I/O lines for the use by your application. The ATS-1 normally uses the D0 and D1 lines, but can be configured to use any two of the other digital lines. Communications with the ATS-1 is done using the standard Arduino Serial.write(), Serial.print() and Serial.read() functions.

Specifications

Size: 4.0" X 2.35"

Power:

AST-1: 5V at 15ma typical, supplied by Arduino host (no LCD backlight option)

ATS-1-YBL: 5V at 150ma typical, supplied by Arduino host (Yellow LED backlight option)

Communications: Serial, 4800 Baud, No parity, 1 stop bit. 5V TTL signal levels (not RS-232)

Features:

LCD: 16 X 2 characters

Switches: 6 navigation, communicate with Arduino host, Reset (Resets both ATS-1 and host Arduino)

LED: red, controlled by software

Buzzer: controlled by software

Contrast adjust: user LCD contrast adjustment

Installation

Caution: The ATS-1 is a solid state electronic device. Use electronic static protection practices to prevent damage!

1. Decide if you want to use the normal serial lines D0 and D1 or two of the other I/O pins. D0 and D1 are shared with the USB port to the development PC. If you use the standard lines you must remove G2 when you upload a new sketch (program) into the Arduino. Put G2 back in after the upload is complete, press the Reset switch, and the Arduino will communicate properly with the ATS-1. If you wish to use other pins and leave D0 and D1 dedicated to the USB port, read the section on Using Software Serial UARTS.
2. Plug the ATS-1 into the Arduino compatible host.
3. Connect the USB cable and start programming!
4. Trimmer R1 (mounted on the bottom of the board) is used to adjust the display contrast. It is set at the factory, but can be adjusted by the user if needed.

Programming information

It is assumed that the user is familiar with the Arduino Serial functions. More information is available at the Arduino web site: www.arduino.cc

Before the Arduino can communicate with the ATS-1, the serial port must be set up. Be sure the **Serial.begin(4800);** statement is in the setup function at the start of your program.

```
void setup()
{
Serial.begin(4800);   // opens serial port, sets data rate to 4800 bps
}
```

Sending data to be displayed on the LCD display is very simple. You use the standard Serial.print() functions. If you are using a software UART, there will be a similar function. The first program written by programmers is the "Hello, world!" program. Printing this only requires the following line of code: **Serial.print("Hello, world!");**

When a character is sent to the ATS-1 to be displayed, it will show up at the position indicated by the cursor, just like when typing on a PC. After a reset, the ATS-1 will have the visible cursor turned off. You can optionally display a cursor in the form of a blinking box or an underline. See Appendix 1 for instructions on controlling the cursor display.

Serial.write() is used to issue commands to the LCD. Some of the commands allow the program to clear the display, position the cursor, select the cursor type and turn it on or off, etc. These are described in Appendix 1. Note that the commands use non-printable characters, requiring the Serial.write(n); statement. Note that the Serial.print(n,BYTE); statement, which was an alternative way to send non-printable characters is not supported starting with the Arduino 1.0 development environment.

The buzzer and LED are also controlled with Serial.write(n); commands. There is a command to turn the LED ON and another to turn it OFF. The LED will stay in its current state until it gets another LED command. Sending the Beep command turns the buzzer on for 1/3 of a second after which it will stop. If a second Beep command is sent before the first one is completed, an internal timer will be reset for an additional 1/3 second of the buzzer running from the time the second command is sent. Sending a series of Beep commands will not accumulate the amount of time the buzzer operates. For example sending the command 3 times in succession with no delays between them will only play a few milliseconds longer than the normal 1/3 second.

Pressing a switch sends a serial byte of data to the Arduino. The switches are numbered 1-6. Pressing switch 1 will send an ASCII '1' to the Arduino. Pressing switch 2 sends an ASCII '2', etc. Switches 1-4 are arranged in a cross pattern with arrow legends for UP, DOWN, LEFT, and RIGHT. Your software can use them for moving the LCD cursor around, selecting options in a menu, or whatever you want your program to use them for. Switches 5 and 6 are labeled F1 and F2 and can be used for any purpose you define them for.

Switch Label	Character Returned
^	1
>	2
<	3
V	4
F1	5
F2	6

Pressing a switch will send a single ASCII character. The switch must be released before it or another switch closure will be recognized by the ATS-1 microcontroller.
The RESET switch does not send commands, but rather does a hard reset on the ATS-1 and the Arduino host board.

The Serial.read() function is used to catch switch press characters. Example:

```
char Key;  //define the variable to hold the byte send back
Key = Serial.read();              // wait until the user presses a switch, Key will get the value of the switch pressed (1-6)
if(Key == '1')                    //see if they pressed switch 1
        {
            // do the stuff for switch 1 press
        }
if(Key == '2')                    //check for switch 2
        {
            //do the switch 2 thing
```

```
}  // you may also want to use the switch-case program structure for selecting among multiple switch type presses
```

Using Software Serial UARTS

The Arduino uses the microcontroller's internal hardware UART peripheral for sending and receiving serial data. Using an internal hardware peripheral reduces the software overhead to conduct serial communications. The USB connection to the PC also uses this port for uploading software to the Arduino. The ATS-1 will interfere with data transfer between the PC and Arduino. Removing jumpers G1 and G2 disconnect the ATS-1 from the Arduino when uploading new software onto the Arduino. G2 must be removed when uploading new programs into the Arduino when using the D0 and D1 serial lines. If you wish, you can use a software UART which allows you to use different digital I/O pins on the Arduino to communicate with the AST-1.

Some applications require additional serial ports and developers have written software implementing software UARTS. At least two versions are available on the Arduino web site. Look for "Soft Serial" or "Software Serial" in the contributed libraries in the Arduino Language Reference pages. This software allows you to specify two pins for an additional serial port. You can pick two unused digital I/O pins and use these to communicate with the ATS-1. Software UARTs may require additional software considerations. Read the documentation with the software UART library, and account for any limitations in your applications.

To use I/O lines other than D0 and D1 perform the following steps:

a. Remove jumpers on G1 and G2.
b. Solder a wire from the pad labeled TX (next to G1) to a pad connected to J3 or J4.
c. Solder a wire from the pad labeled RX (next to G2) to another pad connected to J3 and J4.
d. Follow the software UART's instructions for specifying the pins used for TX and RS. TX is data (LCD, LED, Buzzer) sent from the Arduino. RX is data (switch) received by the Arduino.

Note: Soldering or other modifications to the ATS-1 may void the warranty. It is recommended you test the ATS-1 on the standard serial lines before attempting to use alternate ones.

Using the ATS-1 with Non-Arduino Systems

The ATS-1 can be used with other microcontroller system hosts. You will need to provide 5V DC for power plus connect to the serial lines of the host system. Power and ground can be supplied through pins 2 and 3 of connector J5. If you are powering the ATS-1 from a separate supply from the host, be sure the host and ATS-1 grounds are connected together. The host system must be a 5V system. Connect a wire from the pad labeled TX to the serial out port of the host system. Connect a wire from the RX pad to the serial input to the host. Be sure the host serial port connections are to 5V signal levels, and not RS-232 levels, or the ATS-1 will be damaged. Set the UART settings on the host for 4800 baud, 1 stop bit, no parity.

Checking the ATS-1 Firmware Version

You can check the version of the firmware by holding the '^' key down and then turning on the power. Press the reset button to return the ATS-1 to normal operation.

Warranty and Support

The ATS-1 is warranted for 90 days from purchase date. Unified Microsystems will, at its option, repair or replace defective units returned during the warranty period. Unified Microsystems reserves the right to change specifications of it products at any time without notice. Schematics, sample programs, FAQs, additional information can be found at **www.unifiedmicro.com/support**. Support is provided by email only: **w9xt@unifiedmicro.com** Include **ATS-1** in the subject line. **www.unifiedmicro.com**

APPENDIX 1 ASCII Commands Sent to ATS-1 Terminal Shield

Command	Hex	Decimal	Description	Example
CLR_DISP	0x01	1	Clear display & home cursor	Serial.write (1);
CUR_BACK_SPACE	0x08	8	Move cursor home and leave displayed characters	Serial.write(8);
LED_ON	0x11	17	Turn the LED on	Serial.write(17);
LED_OFF	0x12	18	Turn LED off	Serial.write(18);
CUR_RIGHT	0x15	21	Move cursor right 1 char (arrow key) – does not change displayed character	Serial.write(21);
CUR_LEFT	0x16	22	Move cursor left 1 char (arrow key) – does not change displayed char	Serial.write(22);
CUR_OFF	0x1C	28	Turn cursor off (do not display)	Serial.write(28);
CUR_BOX_ON	0x1D	29	Display blinking box cursor	Serial.write(29);
CUR_LINE_ON	0x1E	30	Display underline cursor on	Serial.write(30);
CUR_CR	0x0D	13	Carriage Return – moves cursor to first position on current line	Serial.write(13);
CUR_LF	0x0A	10	Line Feed – moves cursor to other line, same horizontal position	Serial.write(10);
Bell	0x07	7	Turns buzzer on for 1/3 second	Serial.write(7);
GOTOXY	0x14	20	Moves cursor to selected location	See Below

The LCD display holds two rows of 16 characters. The upper left character (home position) has X-Y coordinates (1,1). The X coordinate, or horizontal position, ranges from 1 (left most character) to 16, the right most character. The top row has a Y coordinate value of 1, and the bottom row has a coordinate value of 2.

The GOTOXY command is the only command with data bytes following the command. To move the cursor to a location, the GOTOXY byte is first sent, followed by the X coordinate (1-16), and finished with the Y coordinate (1-2). If the coordinate data is outside the valid range, the entire command is canceled.

Example: to move the cursor to the 8th position on line 2:

```
Serial.write(0x14);          //command to move the cursor
Serial.write(8);             //Specify the X location to position 8
Serial.write(2);             //Specify the line number (Y location) to the bottom line
```

To save programming time and space, create a function:

```
void Lcd_pos(char X, char Y)
{
Serial.write(0x14);      //command
Serial.print(X);         //horizontal position
Serial.print(Y);         // Line number
}
```

To use this function just call it with the position and row as parameters:

```
Lcd_pos(4,1);            //4th position in row 1
Lcd_pos(10,2);           //10th position in row 2
```