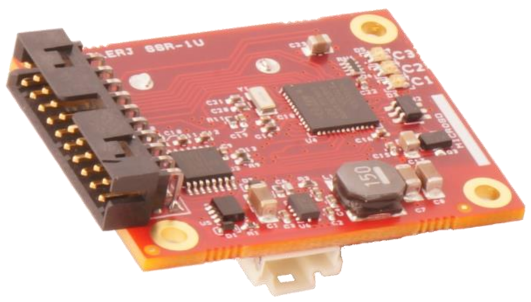
SLERJ



SSR-1U User’s Manual

Revision -

27 December 2019

CONTENTS

Warranty

The SSR-1U Serial Data Recorder is warranted against defects in materials and manufacturing for a period of one year from the date of purchase. In the event of a product failure due to materials or workmanship, Slerj will, at its discretion, repair or replace the product. For warranty service, return the defective produce to Slerj, shipping prepaid, for prompt repair or replacement. Slerj, its suppliers, and its licensors shall in no event be liable for any damages arising from the use of or inability to use this product. This includes business interruption, loss of business information, or other loss which may arise from the use of this product.

SLERJ PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF SLERJ PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE UNDERTAKEN SOLELY AT THE CUSTOMER'S OWN RISK. Should a customer purchase or use Slerj products for any such unauthorized application, the customer shall indemnify and hold Slerj and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs damages and attorney fees which could arise.

1 Introduction

1.1 Description

The SSR-1U is a flexible and robust serial data recording device that takes care of the details of storing data so that you can focus on your application. Up to four streams of serial data can be recorded simultaneously, and each channel is configurable with a variety of serial and storage options. Streams can be recorded automatically at power up, on command through a digital or PWM input, or using software commands. The SSR-1U extends the capabilities of the SSR-1 with an additional serial channel and USB access to the recorded data.

1.2 Features

- Four asynchronous serial channels recorded simultaneously
- Two RS-232 channels and two 5V TTL compatible channels
  (Version available with 3.3V CMOS channels instead of RS-232.)
- Up to 230.4k baud recording on all channels
  (*A high-speed version is available supporting 921.6k baud recording on all channels*)
- Wide supply voltage (4.5 to 32VDC)
- Small size: 1.65 x 1.23 x 0.45 inches (42 x 32 x 12 mm)
- Supports microSD, microSDHC, and microSDXC cards
- exFAT, FAT32, FAT16, and FAT12 file system support
- Long File Name support
- Support for raw and time-tagged recording
- Battery backed real time clock powered by an onboard button cell battery
- Flexible record control: digital input, PWM input, software controlled, or automatic
- User shell for configuration and file system operations
- Binary control protocol for machine automation
- Flexible recording modes (overwrite/append, user defined path and file names, etc.)

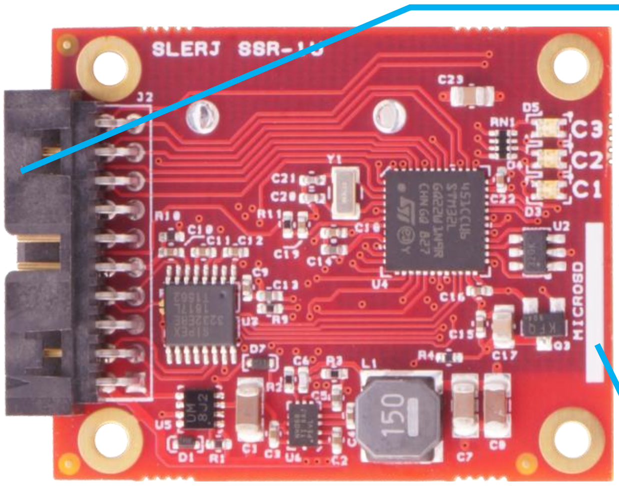
2 Getting Started

2.1 Package Contents

The Serial Recorder is packaged with:

- The SSR-1U Serial Recorder
- Mating connector (Molex 87568-2093) with 9 inch ribbon cable
- Mating cable for the auxiliary USB/channel-4 connector
- Lithium CR1220 button cell battery (U.S. shipments only)
- A microSD card with Users Manual, puttytel terminal emulator, STTP time tagged archive parser with source code, and an example Windows® control protocol utility with source code.

2.2 The SSR-1U Hardware



Main Connector

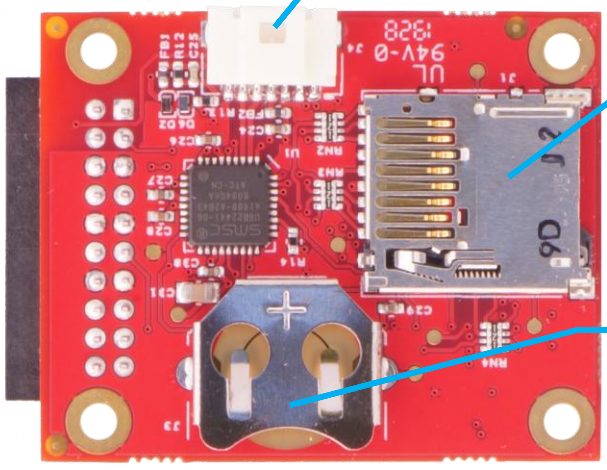
Channel Status LEDs

Insert card as shown in slot beneath board.

Figure 1. Top View

Aux connector (USB and Channel 4)

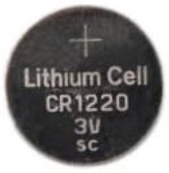
microSD Card Slot (push-push type connector)

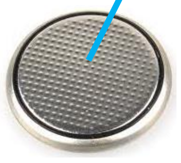


Coin Battery Holder

Figure 2. Bottom View

Insert battery as shown.

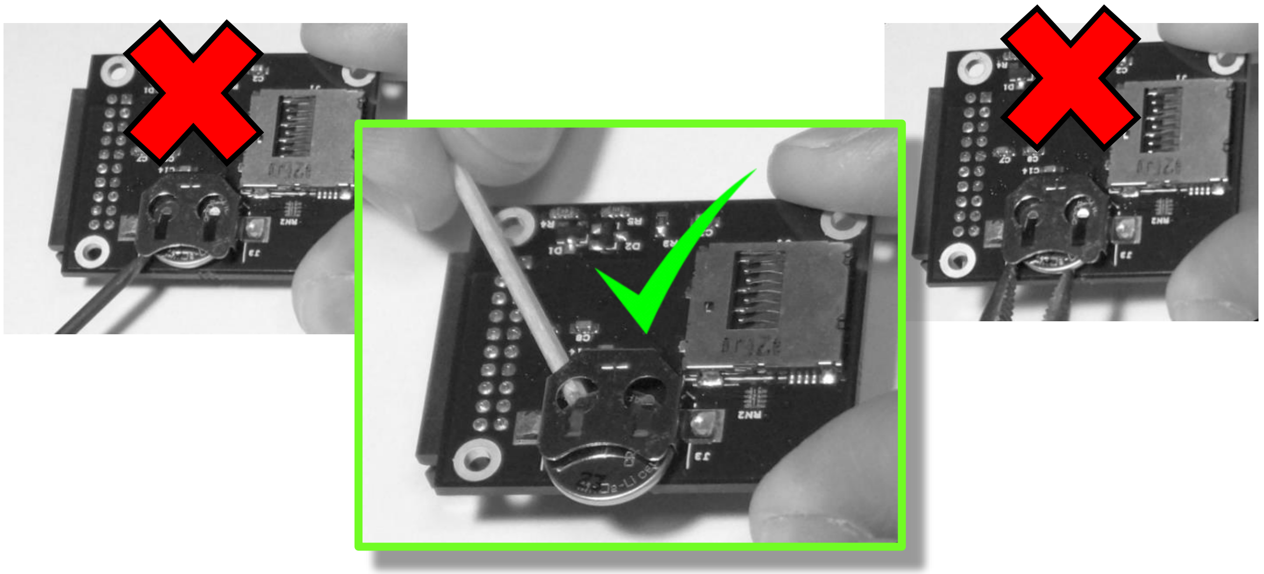




This is the negative side of the battery. It must make contact with the printed circuit board.

WARNING: Improper removal of the backup battery can damage the SSR-1U. To remove the battery, use a blunt stick to push the battery out of the holder.

DO NOT PRY OR PULL THE BATTERY.



CAUTION: Like most electronic components, the SSR-1U can be damaged by electrostatic discharge. Observe typical precautions for handling electrostatic discharge sensitive devices.

2.3 Part Numbers

SSR-1U part numbers follow the following form:

SR1U__T_

H – high-speed variant support 921.6k baud recording

22 – channels 1 and 2 are RS-232
CC – channels 1 and 2 are 3.3V CMOS (not 5V tolerant)
(Unless otherwise noted, all specifications and notes in this document apply to the 22 version. Contact us for questions on the CC version.)

The standard configuration is SR1U22T.

2.4 Connecting the SSR-1U

The pins of the main connector are shown in Figure 3 and described in Table 1. The pins of the auxiliary connector are shown in Figure 4 and described in Table 2. The default configuration of the SSR-1U provides access to the user shell (see Section 4 Interactive Shell, for details) on channel 4 (aux connector) and causes channels 1, 2 and 3 to record when the digital input command pin (DI, pin 15) is pulled to ground. All channels default to 115200 baud, 8 data bits, no parity, and 1 stop bit.

WARNING: Channels 1 and 2 are RS-232 voltage level compatible, and channels 3 and 4 are 5 Volt tolerant, TTL compatible, 3.3V CMOS. Do not connect an RS-232 device to channel 3 or 4 of the SSR-1U.

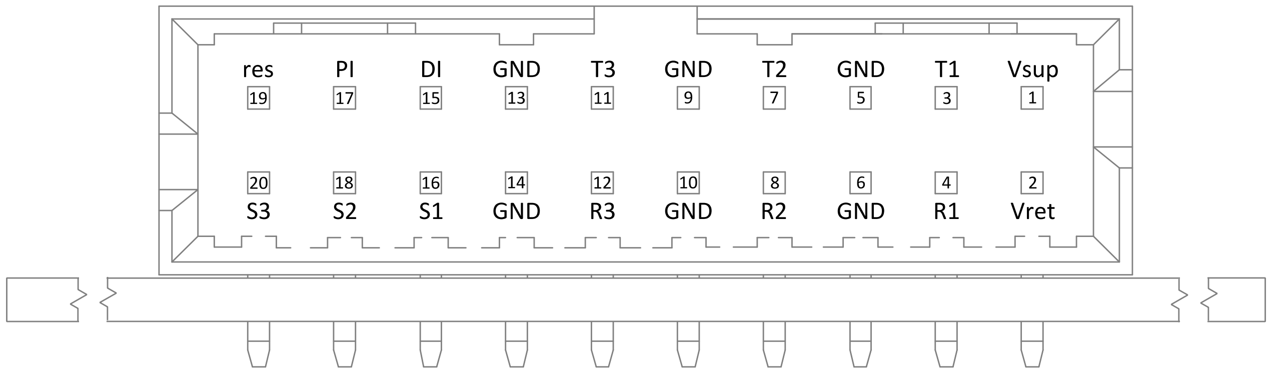


Figure 3. Main Connector (View looking into the face of the connector)

Table 1. Main Connector Pins

Pin [1]	ID	Description
1	Vsup	Supply voltage (4.5-32 VDC).
2	Vret	Supply return (tied to GND onboard the SSR-1U).
3,7 [2]	T1,T2	Asynchronous serial transmitter output for channels 1 and 2.
4,8 [2]	R1,R2	Asynchronous serial receiver input for channels 1 and 2.
11 [3]	T3	Asynchronous serial transmitter output for channel 3.
12 [4]	R3	Asynchronous serial receiver input for channel 3.
16,18,20 [3]	S*x*	Status indication for channel *x*. High level indicates that the channel is recording.
15 [4]	DI	Digital input record command.
17 [4]	PI	PWM input record command.
19 [4]	res	Reserved.

[1] See Section 7.1 Electrical for detailed electrical specifications.
[2] RS-232 compatible (SR122T). For the SR1CCT, these are 3.3V CMOS (not 5V tolerant).
[3] 3.3V CMOS output
[4] 5V tolerant, TTL compatible, 3.3V CMOS input. Internally pulled up to 3.3 VDC.

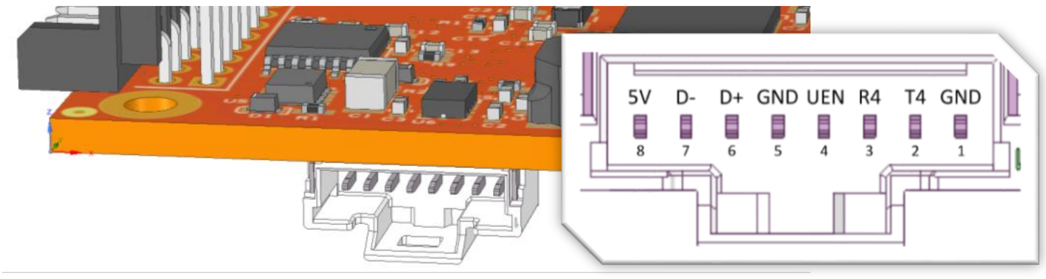


Figure 4. Auxiliary Connector (View looking into the face of the connector)

Table 2. Auxiliary Connector Pins

Pin [1]	ID	Description
1	GND	Ground connection for use with channel 4
2 [2]	T4	Asynchronous serial transmitter output for channel 4
3 [3]	R4	Asynchronous serial receiver input for channel 4
4 [4]	UEN	Enables USB access to the microSD card (recording disabled)
5	GND	USB ground connection
6	D+	USB D+ connection
7	D-	USB D- connection
8 [5]	5V	USB 5V supply

[1] See Section 7.1 Electrical for detailed electrical specifications.
[2] 3.3V CMOS output
[3] 5V tolerant, TTL compatible, 3.3V CMOS input. Internally pulled up to 3.3 VDC.
[4] 5V tolerant, TTL compatible, 3.3V CMOS input. Active polarity is configurable using the Interactive Shell.
[5] The SSR-1U has onboard diodes to allow power from either the main or aux connector, and both can be applied simultaneously without damage to the device. It is not necessary to supply 5V on pin 8 to access USB if power is supplied at the main connector.

Wiring for a typical application is shown in Figure 5. The default SSR-1U configuration is assumed. In the figure, channel 4 is connected to a PC serial port for shell access, allowing configuration and file system operations via a terminal application on the PC. Typically, this connection is made using a USB to TTL serial adapter cable. When the *Close to Record* switch is closed, channels 1, 2, and 3 on the main connector record data from connected serial devices. While the default for channel 4 is shell access, it is a general purpose channel and can also be configured for recording.

In the default configuration, USB mass media access to the microSD card is enabled when the UEN pin is high. In the diagram, UEN is tied to the 5V USB supply so that USB access is enabled when the USB cable is connected to the device.

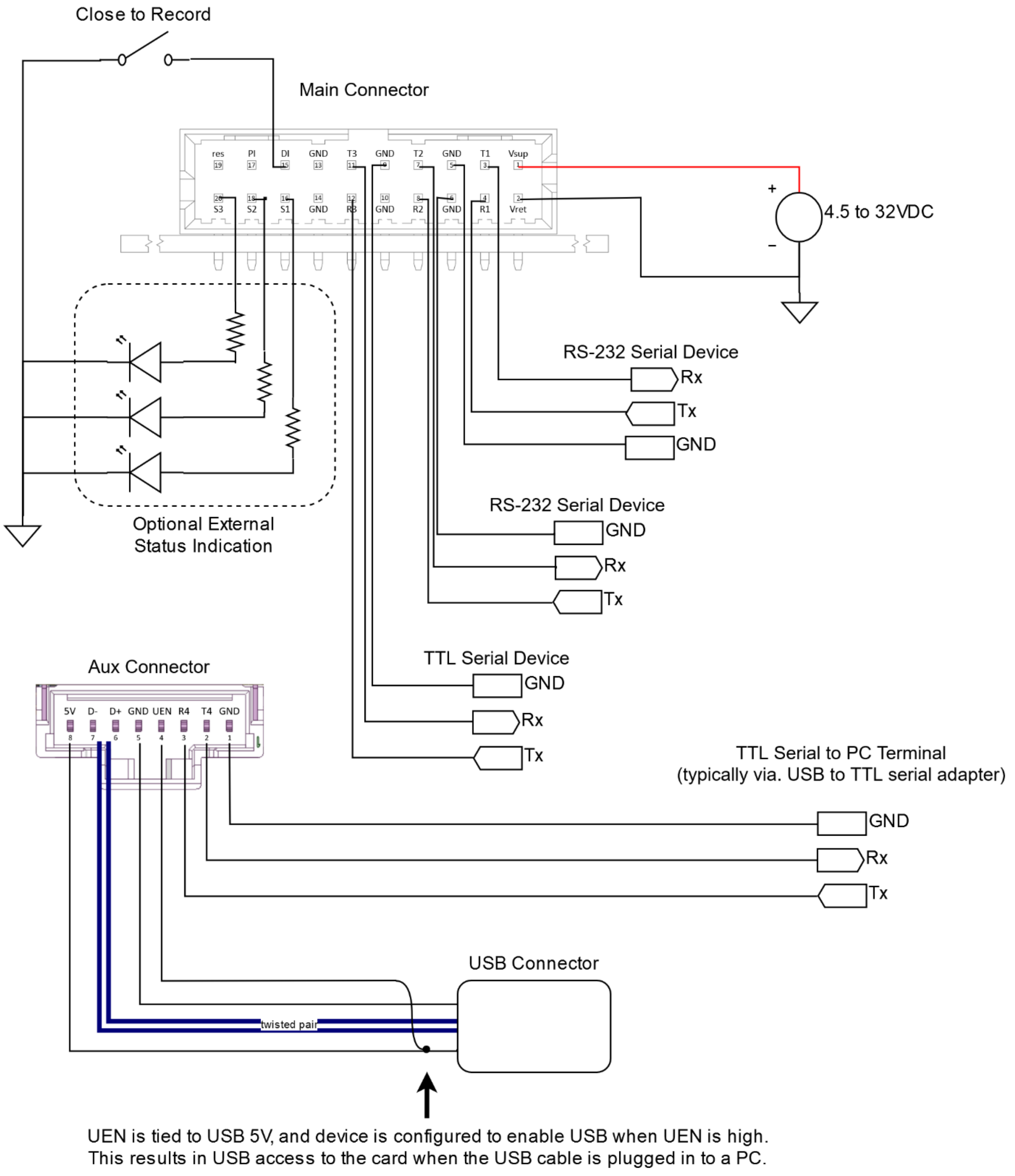

Close to Record
Main Connector
res
PI
DI
GND
T3
GND
T2
GND
T1
Vsup
S3
S2
S1
GND
R3
GND
R2
GND
R1
Vret
4.5 to 32VDC
RS-232 Serial Device
Rx
Tx
GND
Optional External Status Indication
RS-232 Serial Device
GND
Rx
Tx
TTL Serial Device
GND
Rx
Tx
Aux Connector
5V
D-
D+
GND
UEN
R4
T4
GND
TTL Serial to PC Terminal (typically via. USB to TTL serial adapter)
GND
Rx
Tx
USB Connector
twisted pair
UEN is tied to USB 5V, and device is configured to enable USB when UEN is high.
This results in USB access to the card when the USB cable is plugged in to a PC.

Figure 5. Typical application wiring using the default SSR-1U configuration

2.5 Using the SSR-1U

The SSR-1U default configuration sets all channels to 115200 baud, 8 data bits, no parity, and 1 stop bit. Channels 1-3 are configured to record when DI is pulled low. Channel 4 is configured to present the user shell.

On power up, the SSR-1U displays a boot loader announcement and device details. If a channel is attached to the user shell, it will present the user shell prompt. A typical power-on sequence would produce output similar to:

```
Slerj Boot Loader v1.0.0
MK:Slerj
HW:SSR1U
MG:1608747
MD:SSR-1U
SN:1
MV:22T

SSR-1U Shell [Firmware 2.1.0]
>
```

Details of the shell interface are provided in Section 4 Interactive Shell. As an example of shell usage, consider changing the command source for channel 3 to ***+dig*** so that the device records when the DI pin is high. In the following sequence, <*enter*> means pressing the Enter/Return key to execute the command in the shell. With the SSR-1U shell connected to a terminal program, type

    config <enter>

to show the current SSR-1U configuration.

To change the command source for channel 3 to ***+dig***, type

    config 3 src +dig <enter>

Verify that the configuration has been changed using:

    config 3 <enter>

To save the modified configuration in on-board non-volatile memory so that it is preserved across power cycles, type:

    config save <enter>

To confirm that the configuration has been saved, reboot the SSR-1U and verify configuration using:

    reset <enter>
    config <enter>

Note that 'cfg' is an alias for 'config' and can be used as a shortcut.

3 Functional Overview

The SSR-1U consists of four asynchronous serial channels, a data recording subsystem, a user interface module, a real-time clock, digital input/output for status and control, and a USB interface for accessing microSD card data as a mass storage device.

3.1 Serial Channels

The behavior of each serial channel is independent and is defined by a number of configurable parameters, which can be changed using the interactive shell or the control protocol, documented in Sections 4 and 5, respectively. Those parameters are:

- Baud rate – 600 to 230400 baud, inclusive
- Bits – 8 or 7
- Parity – Even, odd, or none
- Stop – 1, 1.5 or 2 stop bits.
- Echo – (Boolean) Echoes received characters out through the transmitter.
- Function
    - *disabled* – The channel is not used.
    - *record* – The channel will record received data when commanded.
    - *shell* – The channel will be tied to the interactive user shell function of the user interface module.
    - *control* – The channel will be tied to the control protocol function of the user interface module.

      Note that the user interface module can be attached only to a single channel. If one of the channels is assigned to either the SHELL or CONTROL functions, no other channel may be assigned to SHELL or CONTROL.

3.2 Record Function

In addition to the basic serial channel parameters above, when a channel is assigned to the RECORD function, several other configurable parameters apply:

- Command Source
  Command Source determines how the channel is commanded to record. Options include a discrete digital signal, a PWM signal, or software controlled through the user interface module (shell or control protocol). *Command Source* for a channel can be set to one of the following:
    - **-soft** – The channel records when the *Soft Command* parameter (defined in the next major bullet) is *true*. With -soft, *Soft Command* is set to *false* at startup, and the DI and PI pins are ignored.

- **+soft** – The channel records when the *Soft Command* parameter is *true*. With +soft, *Soft Command* is set to *true* at startup, and the DI and PI pins are ignored. This selection for *Command Source* causes the channel to automatically record at startup.
- **-dig** – The channel records when the digital input pin (DI) is low.
- **+dig** – The channel records when the digital input pin (DI) is high.
- **-pwm** – Recording starts when the pulse width on the pulse input pin (PI) is 1ms ± 250µs. Recording stops when the pulse width on PI is 2ms ± 250µs.
- **+pwm** – Recording starts when the pulse width on the pulse input pin (PI) is 2ms ± 250µs. Recording stops when the pulse width on PI is 1ms ± 250µs.

The PWM input is designed to work with the type of signal used by hobby servos. The signal is considered valid when high going pulses are present with a pulse width between 750µs and 2250µs, and a period of 4ms to 65ms. The record state for a channel using the PWM input is changed only when a valid PWM signal is present that meets the requirements of the selected PWM *Command Source* (+pwm or –pwm, specified above).

Both the DI and PI pins are 5 volt tolerant and internally pulled to 3.3V. See Section 7.1 Electrical for specifications.

- Soft Command
  Soft Command is a Boolean software parameter that is used when the *Command Source* parameter is ±*soft*. When *Command Source* is ±*soft* and *Soft Command* is *true*, the channel records. The *Soft Command* parameter can be set through the user interface module (interactive shell or control protocol).

- File Type
  The SSR-1U supports three archive types: raw, tagged line, and time tagged archives.
  - When file type is *raw*, bytes are written to file just as they are received, and no timestamp information is attached to the data.
  - When file type is *tl* (tagged line), text timestamps are inserted into the stream at the first printable character following a newline or carriage return. This file type is convenient for line-oriented data, but has a number of limitations as compared to the time tagged archive. Since received data is modified with timestamp strings prior to recording, the original data stream is not preserved. The timestamp format is non-configurable, YYMMDDhhmmss.sss with a trailing space. Since this mode adds 17 characters to every line received, it can significantly inflate the volume of data written to the card. A series of 4 byte lines into the serial channel becomes a series of 21 byte writes to the card, and in extreme cases (short lines at high speed) could exceed the write bandwidth of the card. This mode is not appropriate for non-text data.

- When the file type is *tt* (time tagged), bytes are encapsulated into an archive file structure that associates a timestamp with each group of received bytes with a resolution of 2ms. The time tagged archive format overcomes all of the limitations of the tagged line format, but requires post processing to retrieve the data. The archive is not intended to be human readable. A utility to perform the post processing, STTP, is provided with source code. For details, see Section 6 Time Tagged Archives.

- File Mode
Supported file creation modes are *retry*, *overwrite*, and *append*. When file mode is *retry*, the SSR-1U will continue to retry the file creation operation until it succeeds. File creation can fail if a file with the same name already exists. This mode is a useful complement to user definable file paths (next bullet). *Overwrite* will cause an existing file to be replaced by a newly commanded recording. *Append* will cause new data to be appended to an existing file. For both *overwrite* and *append* modes, if the file specified by File Path does not exist, it is created.

- File Path
The File Path parameter holds a path template that specifies the name and location of the file to be created when recording is commanded. A path template is a normal path string that has replaceable fields defined in Table 3 below. A field is identified in the template by a backslash followed by one of the field identifiers, or several consecutive identifiers encapsulated in square brackets. For example, the path template /c[chms].dat would be translated to the path /c1083000.dat for channel 1 if the time is 08:30:00 when a file is created. Similarly, the path template /gps/nmea\4.txt would be replaced by /gps/nmeaXXXX.txt where XXXX is a number that is incremented on each attempt to open the file. Currently, path templates of up to 44 bytes are supported, and the resulting path (with fields replaced) must be no more than 80 bytes.

Table 3. Path Template Field Codes

Field Identifier	**Replaced With**
c*	channel number [1-3]
Y	year [00-99]
M	month [01-12]
D	day [01-31]
h	hour [00-23]
m	minute [00-59]
s*	second [00-59]
t	tenth of second [0-9]
y	year (4 digit) [2001-2099]

X	hex digit month [1-C]
d	day of year [001-366]
2	two digit sequence number [00-99]
3	three digit sequence number [000-999]
4	four digit sequence number [0000-9999]

[*] This field identifier is lower case.

- File Size
  The SSR-1U supports automatic file close and reopen when a size (or time) threshold is reached. Threshold values of 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, 32 MB, 64 MB, 128 MB, 256 MB, 512 MB, and 1024 MB are supported. Additionally, the File Size parameter can be set to *Hour*, *Day*, or *Week*, causing new files to be started based on time instead of size. By default, the File Size threshold is *off*, and no automatic close/reopen operations are performed.

3.3 User Interface Module

The user interface module provides user access to file system operations, device status, and configuration. The module can be assigned to only one serial channel and can be configured to present either an interactive shell interface or the control protocol on that channel. More information on the interactive shell and control protocol can be found in Sections 4 and 5, respectively.

3.4 Real-Time Clock

The Real-Time Clock (RTC) maintains calendar time for the SSR-1U. An on-board backup battery (CR1220 or CR1216, Lithium 3V) allows the RTC to keep time across power cycles. RTC time is used for file creation and data timestamps.

3.5 Digital I/O

Digital input and output lines are provided to control recording and provide status. On the main connector, PI and DI provide record control as discussed in Section 3.2. Additionally, a status line is provided for channels 1-3 (S1, S2, and S3) to indicate when the channel is recording. A bi-color (red and green) LED indicates status for channels 1-3 (Figure 1). The green segment flashes to indicate reception of serial data on the channel. The red segment illuminates solid red when the corresponding channel is recording. When the channel is not recording, the following flash codes are presented using the red LEDs.

Table 4. Red LED Flash Patterns When Not Recording

Status	Flash Pattern	Description
READY[1]	▪ Short flash every 4 seconds	Indicates that a record channel is ready to record when commanded.
CARD USB[2]	▬ ▪ ▪ ▪ ▬ ▪ ▪ ▪ Long flash followed by three short flashes every 2 seconds	Indicates that the card is being accessed by USB and unavailable for recording.
CARD ERROR[2]	▬ ▪ ▪ ▬ ▪ ▪ Long flash followed by two short flashes every 2 seconds	Indicates that the card is missing or an unrecoverable error has occurred.
CARD FULL[2]	▬ ▬ ▬ ▬ Two long flashes every 2 seconds	Indicates that the card is full.

[1] Presented only on channels configured to record.
[2] Presented on all channel LEDs simultaneously.

3.6 USB Interface

USB 2.0 High-Speed access to the microSD card is provided through the auxiliary connector. When USB access is enabled, complete control of the card is given to the USB host, and serial recording is suspended. In addition to the standard USB connections (two data signals, 5V power, and ground), a USB Enable pin (UEN) is provided for flexibility. The SSR-1U can be powered from the USB interface without enabling USB access to the card. Options for UEN control are detailed with the *config usben* command in Section 4.3 Device Configuration.

4 Interactive Shell

The interactive shell is designed to provide easy access to the SD card file system, device status, and configuration options. Entering '`?`' or '`help`' at the command prompt provides information about using the shell. Each command can be followed by '`?`' to retrieve help information. Multiple commands can be separated by a semi-colon. All commands are case sensitive. For example:

```
>cls ?
Usage: cls
  Clears the screen.
  Aliases: clear

>date;time
20130327
102840
>
```

The shell supports line editing and keeps a history of recently used commands. The ANSI escape sequences shown in Table 5 are supported.

Table 5. Shell Line Editing Sequences

Keyboard Key	**Alternate Sequence**[1]	**Function**
Up-arrow	^p	Recall the previous command to the command line.
Down-arrow	^n	Recall the next command to the command line. This is only available when up-arrow has been used to recall a previous command.
Home	^a	Move the cursor to the start of the command line.
End	^z	Move the cursor to the end of the command line.
Left-arrow	^k	Move the cursor left one character.
Right-arrow	^l	Move the cursor right one character.
Ctrl + Left-arrow	^b	Move the cursor left (backward) one word.
Ctrl + Right-arrow	^f	Move the cursor right (forward) one word.

[1] The caret (^) indicates use of the Ctrl key with the letter.

In the description of individual commands below, the following conventions are used:

[ ] indicates optional parameters
{ } identifies a set of choices separated by | (choose one)
< > indicates a variable defined in the help text

4.1 System Commands

System commands provide access to general system functions including the real time clock and operational status.

Table 6. System Commands

Command	Aliases	Description
cls	clear	Clears the screen.
date [*yyyymmdd*]		Sets the current date to the year, month, and day specified. If no date is specified, this command returns the current date.
help	?	Provides help for using the shell.
reset		Performs a device reset.
status	stat	Displays device status (date/time, inputs, record channels).
time [*hhmmss*][*ap*]		Sets the current time using the hour, minute, and second specified. The hour is assumed to be in 24 hour format. However, the time may be appended with an 'a' or 'p' to explicitly identify AM or PM if a 12 hour format is used.

4.2 File Commands

File commands provide access to the SD card file system. FAT12, FAT16 and FAT32 volumes are supported, and long filenames are supported on FAT32. Many file system commands require a *path*. Both relative and absolute paths are supported in the shell. Directories are separated by a forward slash (/).

Table 7. File Commands

Command	Aliases	Description
chdir <*path*>	cd	Changes the current working directory.
del <*path*>	rm	Removes a file or an empty directory.
df		Prints the volume size and free space -.
dir [*path*]	ls	Lists the contents of a directory. If no path is provided, this command lists the contents of the current directory.
mkdir <*path*>	md	Creates a directory.
pwd		Prints the current working directory.
ren <*path1*> <*path2*>	mv,rn	Moves or renames a file or directory from *path1* to *path2*. [NOTE: Do not move open files]
touch <*path*>		Updates the timestamp on a file or directory.
sz <*path*>		Send a file to the connected terminal[1] using the zmodem protocol.

[1] The zmodem file transfer has been tested successfully with several freely available terminal emulators, including ExtraPutty, Teraterm, SyncTERM, and qodem.

4.3 Device Configuration

Device configuration is manipulated through the user interface module. The current working configuration is held in system memory (RAM) and can be saved to non-volatile memory for preservation across resets. On startup, if the contents of the non-volatile configuration memory are valid, the stored configuration is loaded and used by the SSR-1U. The shell provides access to device configuration through the following commands:

Table 8. Global Configuration Commands

Command	**Description**
config	Prints the current configuration (including all channels).
config save	Saves the working configuration to non-volatile memory.
config load	Retrieves the stored configuration from non-volatile memory.
config erase	Erases the non-volatile configuration memory, but does not change the current working configuration. When non-volatile configuration memory has been erased, configuration defaults are loaded at startup.
config <*ch#*> [*args*]	Provides access to channel configuration. The parameter *ch#* is the channel number (1 to 4). If no additional arguments are specified, this command prints the configuration for the specified channel. Specific channel configuration commands are in Table 9.
config usben {hi | lo | yes | no}	Determines when USB is enabled. It can be enabled based on the UEN pin of the auxiliary connector (high or low), always enabled, or never enabled. The last two are useful for controlling USB access via the shell interface. When UEN is configured *lo* or *hi*, an internal pull-up/down resister is activated to pull the pin into the disabled state.
config leds {on | off | onx}	Allows the on-board LEDs to be turned off for power savings. The *onx* option causes the LEDs to be active for 10 seconds after power is applied. Reserved pin 19 on the main connector can be programmed to enable the LEDs using the p19 configuration option.
config p19 {none | leds}	Reserved pin 19 can be configured in support of various features. The default configuration is *none*. The *leds* option causes pin 19 (when pulled low) to enable LEDs that are inactive as a result of using the leds *off* or *onx* options.

In addition to the global configuration commands presented in Table 8, there are several channel specific configuration commands. The commands in the following table are entered as part of a command line 'config <ch#> *command*'.

Table 9. Channel Configuration Commands

Command	Alias	Description
baud <*rate*>		Sets baud to *rate* (600 to 115200).
bits {8 | 7}		Sets data bits to 8 or 7.
parity { E | O | N | e | o | n }		Sets parity to even, odd, or none.
stop { 1 | 1.5 | 2 }		Sets the number of stop bits.
echo <*bool*[1]>		Enables echoing of received characters on the channel.
function { record | disabled | shell | control }	func	Sets the configured function for the channel. Note that the active function for a channel may be different from the configured function (see Section 4.4 Capturing the Shell).
source [{ + | - }][2]{ soft | dig | pwm }	src	Sets the channel command source.
soft <*bool*>		Sets the channel soft command.
file type { raw | tl | tt }		Selects between raw, tagged line, and time-tagged archives for the channel.
file mode { retry | append | overwrite }		Sets the channel file mode.
file path <*path*>		Sets the channel file path template to *path*. See Section 3.2 Record Function for more information on path templates.
file size { off | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | hour | day | week }		Sets the file size threshold. See Section 3.2 Record Function for more information on file size thresholds.

[1] *bool* denotes a Boolean expression, and may be specified using { y | Y | t | T | true | yes | on } for affirmative and { n | N | f | F | false | no | off } for negative.
[2] The { + | - } prefix is optional. If not specified, + is assumed.

Note that multiple channel configuration commands may be specified together. For example, to set the baud, parity and stop parameters of channel 2 with a single command, type

```
config 2 baud 38400 parity N stop 1
```

Also note that the file option commands are two-word commands (don't omit the word 'file'). For example, to set the file type to *raw* on channel 2, type

```
config 2 file type raw
```

4.4 Capturing the Shell

When all four channels are configured to record, then the shell interface is unavailable. To overcome this limitation, a special procedure is provided to allow access to the shell on startup through any channel. When the shell is 'captured' in this way, the configured function for the channel is temporarily suspended. The capture mechanism operates prior to loading stored configuration data, so all channels operate at 115200 baud, 8 bits, no parity, and 1 stop bit for capture.

The process for capture involves presentation of the boot loader message at startup, the user quickly typing the string *config*, the device replying with an upper case, four character challenge string, and the user echoing the challenge string back to the device in lower case. Details are as follows:

- On startup, the SSR-1U displays the boot loader message and device information.
- An 800ms window begins in which a valid character of the capture sequence must be received from the user. If an invalid character is received, or 800ms elapses, the capture sequence is aborted and the device boots normally according to its stored configuration. Each valid character received resets the capture window to 800ms. Since it is difficult to judge the time from power on until the device is ready to begin receiving the capture sequence, the valid capture sequence includes up to 5 lower case *z* characters prior to the string *config*. A typical capture will involve hitting *z* while powering on the device until the *z* character is echoed from the device, then typing *config*.
- When the previous step has been completed, the SSR-1U will send a random challenge string consisting of 4 upper case characters, and a new capture window of 5 seconds is established. The user must type those same characters in lower case to complete the capture process. If the challenge string is not answered in 5 seconds, the capture process is aborted and the SSR-1U starts normally.

In order to support the capture feature, a distinction is made between the active function of a channel and its configured function. The active function will mirror the configured function when possible. Once the shell becomes the active function on a channel, the parameters that affect communication (baud, bits, parity, stop, and active function) are fixed until reset. Any channel configuration options changed for the captured channel (and saved) will not take effect until the next reset. If *shell* was the configured function on a channel other than the captured channel, the active function for that channel becomes *disabled*.

5 Control Protocol

The user interface module provides access to control, status, and configuration through the interactive shell or through the control protocol. This control protocol interface provides a structured message-response framework to support robust communication in machine automation environments. The notation used in this section is intended to be familiar to C programmers. Values presented in hexadecimal are prepended with *0x*. The *&* symbol represents the bit-wise AND logical operation.

5.1 Message Format

Control protocol messages are exchanged between the SSR-1U and the user in the form of byte-oriented packets. Each packet has a start sequence, an ID, a payload count, an optional payload, and a checksum.

Table 10. Control Protocol Message Format

Header				Payload	Checksum	
Start1	Start2	ID	Count	Payload	Cksum1	Cksum2
0x81	0xA1	*0xID*	*0xNN*	*0xAA 0xBB … 0xZZ*	*0xC1*	*0xC2*

The start sequence for every packet is the same (0x81 0xA1), and is followed by a single ID character. Count identifies the number of payload bytes that are included in the packet, and can be zero. The checksum bytes represent a Fletcher checksum as defined in internet RFC 1145. It is computed over the ID, Count, and payload bytes. The basic algorithm for computing the checksum is:

```
unsigned char Cksum1=0;
unsigned char Cksum2=0;
unsigned char *p = (address of message ID);
int i=0;
while(i<number_of_payload_bytes+2)
  {
  Cksum1 = Cksum1 + p[i];
  Cksum2 = Cksum2 + Cksum1;
  i = i + 1;
  }
```

Ordinarily, having a single byte to represent payload count would suggest that the maximum possible payload length is 255 bytes. But since many applications may require longer packets, the most significant bit (MSb) of the Count is given special significance in the control protocol. When the MSb is set, the lower 7 bits are interpreted as a count of 8 byte blocks. Also, since counts without the MSb set already provide coverage for payloads between 0 and 127 bytes, this special function starts at 128 bytes. When the MSb of Count is set, payload length is calculated as 128 + (Count&0x7F)*8.

For example, a Count of 0x80 would indicate a payload of 128 bytes. A Count of 0x81 would indicate a payload length of 136 bytes. The longest payload supported by this implementation is (Count = 0xFF) 1144 bytes. When the MSb of Count is set, the payload length will be modulo 8.

In the control protocol, all multi-byte values are sent most significant byte first (i.e., big endian). In this document, messages are divided into two groups: general messages and messages used to set and query configuration. In the message definitions, types will be identified by the following abbreviations shown in Table 11.

Table 11. Control Protocol Type Definitions

Type	Description
U1	Unsigned 8 bit integer
U2	Unsigned 16 bit integer
U4	Unsigned 32 bit integer
I1	Signed 8 bit integer
I2	Signed 16 bit integer
I4	Signed 32 bit integer
B*x*	String of *x* bytes
BN	Variable length string of bytes

5.2 General Messages

General messages (Table 12) provide access to command functions and status of the SSR-1U. All general messages received by the SSR-1U will be answered with a message to acknowledge (ACK), to negatively acknowledge (NACK), or to provide the requested data.

Table 12. General Control Protocol Messages

ID	Direction[1]	Description
0x10	in	Record
0x11	in	Stop
0x20	out	Command Status
0x21	out	Card Status
0x22	out	Disk Status
0x24	out	All Channel Status
0x30	in	Set Date
0x30	out	Date
0x31	in	Set Time
0x31	out	Time
0x50	in	Configuration Set
0x51	out	Configuration Query
0x99	in	Reset

[1] Direction is from the SSR-1U's point of view

ACK messages have ID 0x90 with a single payload byte that is the message ID being acknowledged. For example, an ACK for a Record message (ID=0x10) would comprise the following:

Header				Payload	Checksum	
Start1	Start2	ID	Count	Payload	Cksum1	Cksum2
0x81	0xA1	0x90	0x01	0x10	0xA1	0xC2

NACK responses to general messages have ID 0x91 and two payload bytes. The first payload byte is the message ID that is being negatively acknowledged. The second payload byte indicates the reason for the negative acknowledgement. For example, a NACK for a Record message that was rejected because an invalid channel number (error code = 0x02) was specified would be constructed as:

Header				Payload	Checksum	
Start1	Start2	ID	Count	Payload	Cksum1	Cksum2
0x81	0xA1	0x91	0x02	0x10 0x02	0xA5	0x6C

For a list of error codes, see Section 5.4 Error Codes.

All general output messages (direction = out) must be polled by the user. A message is polled by sending a message to the SSR-1U with the same message ID and zero payload. For example, to poll the All Channel Status message (ID = 0x24), construct the poll message as follows:

Header				Checksum	
Start1	Start2	ID	Count	Cksum1	Cksum2
0x81	0xA1	0x24	0x00	0x24	0x48

5.2.1 Record

Message	Record
Description	Requests recording to start immediately on the specified channel.

ID	**Payload Length**	**Direction**	**Message Rate**
0x10	1 to 45 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1	1	channel		Channel (1 – 4)
1	BN	2	path		Path Template to set prior to recording.

Notes:

1. The configured channel function must already be *record*. Otherwise, this command will have no effect. This command sets the channel command source to +*soft* and sets the soft command to *true*.
2. Path is optional and may be omitted. In that case, recording will begin using the currently defined path template.

Possible Replies:

ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_PATH_LEN
NACK_PATH_SYNTAX
NACK_PATH_INV_TOKEN
NACK_PATH_SEQ
NACK_PATH_XLEN

5.2.2 Stop

Message	Stop				
Description	Requests the specified channel to stop recording immediately.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x11		1 byte		in	
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1	1	channel		Channel (1 – 4)

Notes:
1. The configured channel function must already be *record*. Otherwise, this command will have no effect. This command sets the channel command source to +*soft* and sets the soft command to *false*.

Possible Replies:
- **ACK**
- **NACK_INV_LEN**
- **NACK_INV_CH**

5.2.3 Command Status

Message	Command Status				
Description	Provides the status of the record command sources.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x20		5 bytes		out	polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		status	bitfield	Status: bit 7: channel 4 soft command bit 6: channel 3 soft command bit 5: channel 2 soft command bit 4: channel 1 soft command bit 3: (reserved) bit 2: PWM input valid bit 1: PWM input 2ms (as opposed to 1ms) bit 0: digital input (DI pin) high
1	U2		width	μsec	PWM pulse width
3	U2		period	μsec	PWM period

Notes:
1. This message is polled by sending a message with ID=0x20 and no payload.

5.2.4 Card Status

Message	Card Status				
Description	Provides the status of the SD card as detected by the socket.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x21		1 byte		out	polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		status	bitfield	Status: bits 7-3: (reserved) bit 2: Card is write protected bit 1: Card is not inserted bit 0: Card is not initialized

Notes:
1. This message is polled by sending a message with ID=0x21 and no payload.

5.2.5 Disk Status

Message	Disk Status				
Description	Provides the status of the disk as detected by the file system.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x22		8 bytes		out	polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U4		size	kB	Size of the disk
4	U4		free	kB	Unused disk space

Notes:
1. This message is polled by sending a message with ID=0x22 and no payload.

5.2.6 All Channel Status

Message	All Channel Status				
Description	Provides the status of all channels.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x24		4 bytes		out	polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1	2	channel1	bitfield	Status: bit 7: record is commanded bit 6: (reserved) bits 5-4: channel function 0 = disabled 1 = record 2 = control 3 = shell bits 3-0: channel file state 0 = closed 1 = building path 2 = opening file 3 = recording 4 = path template translation error 5 = error building path 6 = error opening file 7 = disk error 8 = disk full
1	U1		channel2	bitfield	(see channel 1)
2	U1		channel3	bitfield	(see channel 1)
3	U1		channel4	bitfield	(see channel 1)

Notes:
1. This message is polled by sending a message with ID=0x24 and no payload.

5.2.7 Set Date

Message	Set Date				
Description	Sets the date.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x30		4 bytes		in	
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U2		year		Year (2001 – 2099)
2	U1		month		Month (1 – 12)
3	U1		day		Day of month (1 – 31)
Notes:					
Possible Replies: **ACK** **NACK_INV_LEN** **NACK_INV_DATE**					

5.2.8 Date

Message	Date				
Description	Provides the date.				
ID		**Payload Length**		**Direction**	**Message Rate**
0x30		6 bytes		out	polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U2		year		Year (2001 – 2099)
2	U1		month		Month (1 – 12)
3	U1		day		Day of month (1 – 31)
4	U1		doy		Day of year (1 – 366)
5	U1		wday		Weekday (0=Sunday, 1=Monday, … 6 = Saturday)
Notes: 1. This message is polled by sending a message with ID=0x30 and no payload.					

5.2.9 Set Time

Message	Set Time				
Description	Sets the time.				
ID		**Payload Length**	**Direction**		**Message Rate**
0x31		3 bytes	in		
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		hour		Hour (0 – 23)
1	U1		minute		Minute (1 – 59)
2	U1		second		Second (1 – 59)
Notes:					
Possible Replies: **ACK** **NACK_INV_LEN** **NACK_INV_TIME**					

5.2.10 Time

Message	Time				
Description	Provides the time.				
ID		**Payload Length**	**Direction**		**Message Rate**
0x31		5 bytes	out		polled[1]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		hour		Hour (0 – 23)
1	U1		minute		Minute (1 – 59)
2	U1		second		Second (1 – 59)
3	U2		msec		millisecond (0 – 999)
Notes: 1. This message is polled by sending a message with ID=0x31 and no payload.					

5.2.11 Configuration Set

Message	Configuration Set				
Description	Provides access to configuration functions.				
ID	**Payload Length**	**Direction**	**Message Rate**		
0x50	> 1 bytes	in			
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	BN	1	payload		Payload to pass to the configuration subsystem

Notes:

1. The payload is passed to a configuration subsystem that handles it as a configuration request. Generally the first byte of the payload specifies the configuration action or item to be affected. Details on the control protocol configuration subsystem are provided in Section 5.3 Configuration Messages.

Possible Replies:

Depends on the payload. See Section 5.3 Configuration Messages for details.

5.2.12 Configuration Query

Message	Configuration Query				
Description	Provides access to configuration data.				
ID	**Payload Length**	**Direction**	**Message Rate**		
0x51	1 byte	in			
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		item		Configuration item that is requested.

Notes:

Possible Replies:

NACK_INV_LEN
NACK_INV_CH
A Configuration Query Reply (Section 5.3.5 Query Channel Parameter)

5.2.13 Reset

Message	Reset		
Description	Resets the SSR-1U.		
ID	**Payload Length**	**Direction**	**Message Rate**
0x99	0 bytes	in	
Notes:			
Possible Replies: **ACK** **NACK_INV_LEN**			

5.3 Configuration Messages

The Control Protocol provides a packetized interface to the configuration of the SSR-1U. Operations are the same as those available through the shell. For an overview of configuring the SSR-1U, refer to Section 4.3 Device Configuration. The control protocol provides access to configuration through the Configuration Set and Configuration Query messages. The payload of these messages is passed to a configuration subsystem that handles the specific configuration request identified by the message payload. Configuration subsystem operations are available to load, save, and erase the non-volatile configuration memory, set channel parameters, and query current channel parameters.

As with general messages, the SSR-1U replies to all configuration messages with acknowledgement (ACK), negative acknowledgement (NACK), or the requested data. ACK messages are returned in response to successful Configuration Set (ID=0x50) requests.

Header				Payload	Checksum	
Start1	Start2	ID	Count	Payload	Cksum1	Cksum2
0x81	0xA1	0x90	0x01	0x50	0xE1	0x02

NACK messages can be provided in response to Configuration Set or Configuration Query messages and will have the form

Header				Payload	Checksum		
Start1	Start2	ID	Count	Payload	Cksum1	Cksum2	
0x81	0xA1	0x91	0x02	0x50 *0xEC*	*0xC1*	*0xC2*	<= Config Set NACK
0x81	0xA1	0x91	0x02	0x51 *0xEC*	*0xC1*	*0xC2*	<= Config Query NACK

where *0xEC* is an error code as defined in Section 5.4 Error Codes, and *0xC1* and *0xC2* are appropriately calculated checksums.

5.3.1 Load

Request	Load Configuration				
Description	Loads the configuration data stored in non-volatile configuration memory.				
Container	Configuration Set Message				
Configuration ID[1]	**Payload Length**[2]	**Direction**			
0x01	1 bytes	in			
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
Notes:					
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.					
2. Payload length is the length of the container message payload.					
Possible Replies:					
ACK					
NACK_INV_LEN					
NACK_INV_NV					

5.3.2 Save

Request	Save Configuration				
Description	Stores the working configuration data in non-volatile configuration memory.				
Container	Configuration Set Message				
Configuration ID[1]	**Payload Length**[2]	**Direction**			
0x02	1 bytes	in			
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
Notes:					
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.					
2. Payload length is the length of the container message payload.					
Possible Replies:					
ACK					
NACK_INV_LEN					

5.3.3 Erase

Request	Erase Configuration
Description	Erases the non-volatile configuration memory.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x03	1 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID

Notes:

1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:

ACK
NACK_INV_LEN

5.3.4 Set Channel Parameter

5.3.4.1 Line

Request	Set Line
Description	Sets all communication parameters of a channel.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x10	5 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		line		Line Parameters: bits 7-6: parity 0 = none 1 = odd 2 = even bits 5-4: stop bits 0 = 1 stop bit 1 = 1.5 stop bits 2 = 2 stop bits bit 3: data bits 0 = 8 data bits 1 = 7 data bits bits 2-0: (reserved)
3	U2		baud	baud/100	Baud rate (divided by 100).

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
- **ACK**
- **NACK_INV_LEN**
- **NACK_INV_CH**
- **NACK_INV_BAUD**
- **NACK_INV_PARITY**
- **NACK_INV_STOP**

5.3.4.2 Baud

Request	Set Baud
Description	Sets baud rate for a channel.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x11	4 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U2		baud	baud/100	Baud rate (divided by 100).

Notes:

1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:

ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_BAUD

5.3.4.3 Parity

Request	Set Parity
Description	Sets parity for a channel.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x12	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		parity		Parity: bits 7-2: (reserved) bits 1-0: parity 0 = none 1 = odd 2 = even

Notes:

1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:

ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_PARITY

5.3.4.4 Stop

Request	Set Stop
Description	Sets stop bits for a channel.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x13	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		stop		Stop: bits 7-2: (reserved) bits 1-0: stop bits 0 = 1 stop bit 1 = 1.5 stop bits 2 = 2 stop bits

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_STOP

5.3.4.5 *Data Bits*

Request	Set Data Bits
Description	Sets data bits for a channel.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x14	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		bits		Bits: bits 7-1: (reserved) bit 0: stop bits 0 = 8 data bits 1 = 7 data bits

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_PARITY

5.3.4.6 *Function*

Request	Set Function
Description	Sets a channel's function.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x20	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		function		Function: bits 7-2: (reserved) bits 1-0: function 0 = disabled 1 = record 2 = control 3 = shell

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH
NACK_SHCTRL_TAKEN

5.3.4.7 Source

Request	Set Source
Description	Sets a channel's record command source.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x21	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		source		Source: bits 7-3: (reserved) bits 2-0: source 0 = +soft (soft command, true on reset) 1 = -soft (soft command, false on reset) 2 = +dig (DI pin, record when high) 3 = -dig (DI pin, record when low) 4 = +pwm (PI pin, record when 2ms) 5 = -pwm (PI pin, record win 1ms)

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH
NACK_INV_SOURCE

5.3.4.8 *Soft Command*

Request	Set Soft Command
Description	Sets the value of a channel’s internal soft command.
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x22	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		soft		Soft Command: bits 7-1: (reserved) bit 0: soft command (0=false, 1 = true)

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH

5.3.4.9 File Type

Request	Set File Type
Description	Sets the type of archive that a channel will record (raw or time tagged).
Container	Configuration Set Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x30	3 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		type		File Type: bits 7-1: (reserved) bit 0: file type 0 = raw 1 = time tagged 2 = tagged line

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH

5.3.4.10 File Mode

Request	Set File Mode				
Description	Sets the file open mode used by a channel.				
Container	Configuration Set Message				
Configuration ID[1]		**Payload Length**[2]		**Direction**	
0x31		3 bytes		in	
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		mode		File Mode: bits 7-2: (reserved) bits 1-0: file mode 0 = retry 1 = append 2 = overwrite
Notes: 1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message. 2. Payload length is the length of the container message payload.					
Possible Replies: **ACK** **NACK_INV_LEN** **NACK_INV_CH** **NACK_INV_FM**					

5.3.4.11 File Path

Request	Set File Path		
Description	Sets the path template used by a channel when recording.		
Container	Configuration Set Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x33	3 to 46 bytes	in	

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	BN	3	path		File Path Template

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.
3. See Section 3.2 Record Function, for details on path templates.

Possible Replies:
- **ACK**
- **NACK_INV_LEN**
- **NACK_INV_CH**
- **NACK_PATH_LEN**
- **NACK_PATH_SYNTAX**
- **NACK_PATH_TOKEN**
- **NACK_PATH_SEQ**
- **NACK_PATH_XLEN**

5.3.4.12 File Size

Request	Set File Size		
Description	Sets the file size threshold used by a channel.		
Container	Configuration Set Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	
0x34	3 bytes	in	

Payload

Byte Offset	Type	Notes	Name	Units	Purpose / Comment
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		size		File Size: bits 7-4: (reserved) bits 3-0: file size 0 = off 1 = 1 MB 2 = 2 MB 3 = 4 MB 4 = 8 MB 5 = 16 MB 6 = 32 MB 7 = 64 MB 8 = 128 MB 9 = 256 MB 10 = 512 MB 11 = 1024 MB 12 = hour 13 = day 14 = week 15 = off

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Set or Query container message.
2. Payload length is the length of the container message payload.

Possible Replies:
ACK
NACK_INV_LEN
NACK_INV_CH

5.3.5 Query Channel Parameter

Queries to the configuration subsystem are performed using Configuration Query messages. When a valid query is received, the SSR-1U replies with a Configuration Query Reply. Configuration Reply uses the same message ID (0x51) as the Configuration Query. The payload of the reply depends on the request, and available configuration items are listed below.

5.3.5.1 Line

Item	Line				
Description	Provides all communication parameters of a channel.				
Container	Configuration Query Reply Message				
Configuration ID[1]		**Payload Length**[2]		**Direction**	**Message Rate**
0x10		5 bytes		out	polled[3]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		line		Line Parameters: bits 7-6: parity 0 = none 1 = odd 2 = even bits 5-4: stop bits 0 = 1 stop bit 1 = 1.5 stop bits 2 = 2 stop bits bit 3: data bits 0 = 8 data bits 1 = 7 data bits bits 2-0: (reserved)
3	U2		baud	baud/100	Baud rate (divided by 100).

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.2 Baud

Item	Baud				
Description	Provides baud rate for a channel.				
Container	Configuration Query Reply Message				
Configuration ID[1]		**Payload Length**[2]		**Direction**	**Message Rate**
0x11		4 bytes		out	polled[3]
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U2		baud	baud/100	Baud rate (divided by 100).

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.3 Parity

Item	Parity
Description	Provides parity for a channel.
Container	Configuration Query Reply Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x12	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		parity		Parity: bits 7-2: (reserved) bits 1-0: parity 0 = none 1 = odd 2 = even

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.4 Stop

Item	Stop				
Description	Provides stop bits for a channel.				
Container	Configuration Query Reply Message				
Configuration ID[1]	**Payload Length**[2]		**Direction**	**Message Rate**	
0x13	3 bytes		out	polled[3]	
Payload					
Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		stop		Stop: bits 7-2: (reserved) bits 1-0: stop bits 0 = 1 stop bit 1 = 1.5 stop bits 2 = 2 stop bits

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.5 Data Bits

Item	Data Bits		
Description	Provides data bits for a channel.		
Container	Configuration Query Reply Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x14	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		bits		Bits: bits 7-1: (reserved) bit 0: stop bits 0 = 8 data bits 1 = 7 data bits

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.6 Function

Item	Function		
Description	Provides a channel’s function.		
Container	Configuration Query Reply Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x20	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		function		Function: bits 7-2: (reserved) bits 1-0: function 0 = disabled 1 = record 2 = control 3 = shell

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.7 Source

Item	Source		
Description	Provides a channel's record command source.		
Container	Configuration Query Reply Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x21	3 bytes	out	polled[3]

Payload

Byte Offset	Type	Notes	Name	Units	Purpose / Comment
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		source		Source: bits 7-3: (reserved) bits 2-0: source 0 = +soft (soft command, true on reset) 1 = -soft (soft command, false on reset) 2 = +dig (DI pin, record when high) 3 = -dig (DI pin, record when low) 4 = +pwm (PI pin, record when 2ms) 5 = -pwm (PI pin, record win 1ms)

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.8 *Soft Command*

Item	Soft Command
Description	Provides the value of a channel's internal soft command.
Container	Configuration Query Reply Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x22	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		soft		Soft Command: bits 7-1: (reserved) bit 0: soft command (0=false, 1 = true)

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.9 File Type

Item	File Type
Description	Provides the type of archive that a channel will record (raw or time tagged).
Container	Configuration Query Reply Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x30	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		type		File Type: bits 7-1: (reserved) bit 0: file type 0 = raw 1 = time tagged 2 = tagged line

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.10 File Mode

Item	File Mode		
Description	Provides the file open mode used by a channel.		
Container	Configuration Query Reply Message		
Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x31	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		mode		File Mode: bits 7-2: (reserved) bits 1-0: file mode 0 = retry 1 = append 2 = overwrite

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.3.5.11 File Path

Item	File Path
Description	Provides the path template used by a channel when recording.
Container	Configuration Query Reply Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x33	3 to 46 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	BN	4	path		File Path Template

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).
4. See Section 3.2 Record Function, for details on path templates.

5.3.5.12 File Size

Item	File Size
Description	Provides the file size threshold used by a channel.
Container	Configuration Query Reply Message

Configuration ID[1]	**Payload Length**[2]	**Direction**	**Message Rate**
0x34	3 bytes	out	polled[3]

Payload

Byte Offset	**Type**	**Notes**	**Name**	**Units**	**Purpose / Comment**
0	U1		CID[1]		Configuration ID
1	U1		channel		Channel (1 – 4)
2	U1		size		File Size: bits 7-4: (reserved) bits 3-0: file size 0 = off 1 = 1 MB 2 = 2 MB 3 = 4 MB 4 = 8 MB 5 = 16 MB 6 = 32 MB 7 = 64 MB 8 = 128 MB 9 = 256 MB 10 = 512 MB 11 = 1024 MB 12 = hour 13 = day 14 = week 15 *(not used)*

Notes:
1. Configuration ID is the first byte of the payload in the Configuration Query Reply container message.
2. Payload length is the length of the container message payload.
3. This message is polled using a Configuration Query message with a two byte payload (this Configuration ID and the channel number).

5.4 Error Codes

The control protocol provides a unified set of error codes to simplify interpreting and displaying errors to the user. The possible error codes are listed in Table 13.

Table 13. Control Protocol Error Codes

Value	Name	Description
1	NACK_INV_LEN	The message does not have the expected length.
2	NACK_INV_CH	An invalid channel number was provided. Must be 1 to 4.
3	NACK_INV_NV	The non-volatile configuration memory contents are invalid.
4	NACK_INV_DATE	The date is invalid.
5	NACK_INV_TIME	The time is invalid.
6	NACK_INV_BAUD	The baud rate is invalid (outside min / max allowed)
7	NACK_INV_PARITY	The parity is invalid.
8	NACK_INV_STOP	The stop bit value is invalid.
9	NACK_SHCTRL_TAKEN	The shell or control function is already assigned to another channel. It cannot be assigned to more than one channel.
10	NACK_INV_SOURCE	The record command source is invalid.
11	NACK_INV_FM	The file mode is invalid.
12	NACK_PATH_LEN	The path template is too long.
13	NACK_PATH_SYNTAX	The path has a syntax error.
14	NACK_PATH_INV_TOKEN	The path has an unrecognized template field code.
15	NACK_PATH_SEQ	A sequence field code was used in a directory name, and is only valid in a file name.
16	NACK_PATH_XLEN	The translated path is too long after replacing all field codes.
17	NACK_SD_DISK_ERR	File system disk error
18	NACK_SD_INT_ERR	File system internal error
19	NACK_SD_NOT_READY	The file system is not ready (card not present or not initialized).
20	NACK_SD_INV_DRIVE	(not currently used)
21	NACK_SD_NOT_ENABLED	(not currently used)
22	NACK_SD_NO_FS	No file system was found on the card.
23	NACK_SD_TIMEOUT	A timeout occurred when accessing the card.
24	NACK_SD_UNKNOWN	Unknown file system or card error.
25	NACK_UNKNOWN	The request was not recognized.

6 Time Tagged Archives

Often, it is important to know not only what serial data was transferred, but when it was transferred. This is useful in both the analysis of communication systems and in the ability to reconstruct streams as they originally occurred. The SSR-1U supports time tagged archives, in which received bytes are tagged with the time they were received. When a channel is configured to record time tagged archives, the received bytes are encapsulated in packets prior to being written to the file system. Two types of packets are currently defined: the data packet and a time correlation packet. All multi-byte words in the archive are big endian.

> Note that a software utility, including source code, is provided with the SSR-1U for parsing time tagged archives into a variety of useful formats. See section 6.3 The STTP Utility for details.

6.1 Data Packet

The data packet uses the system free running clock as the time stamp source. Bytes are grouped into 2ms windows for stamping. Each data packet begins with a base time stamp that identifies the whole second in which the data was collected. The base time stamp is followed by a series of frames that are composed of an incremental time stamp (fractional second within the base time stamp window) and a group of bytes that were received during the increment. The data packet is terminated by a Fletcher checksum as defined in section 5.1 Message Format.

Table 14. Time Tagged Data Packet

Element	Bytes	Description
Packet Header 0x82 0xA2	2	Packet start sequence.
Run Time	4	Current run time in seconds.
Frame repeated until an mSec_Count value of 0xFFFF is encountered.		
mSec_Count[2]	2	Fractional second and number of bytes for this frame. bits 15-7: milliseconds / 2 bits 6-0: number of bytes to follow (*n*)
Data	*n*	The *n* bytes that were received in the time window leading up to this packet frame.
End Sequence	2	0xFFFF (invalid mSec_Count)
Checksum	2	Fletcher checksum calculated between Run Time and End Sequence, inclusive.

[2] With the introduction of the high-speed unit, the 7 bit count value is no longer sufficient to represent all of the bytes that could be received in a 2ms frame when baud rate is greater than 635k. When more than 127 bytes are received in a frame, a second block for the frame will be generated with the same mSec value. The first block will have 127 bytes, and the second block will contain the remainder of the bytes received in the frame.

6.2 Time Correlation Packet

The time correlation packet associates the free running clock timer with the real time clock. A time correlation packet is written when the recording is started, every 10 minutes, and as the recording is stopped.

Table 15. Time Tagged Time Correlation Packet

Element	Bytes	Description
Packet Header 0x82 0xA3	2	Packet start sequence.
Run Time	4	Current run time in milliseconds.
RTC Time	6	Real Time Clock word 0: bits 15-4: year (2001 – 2099) bits 3-0: month (1 – 12) word 1: bits15-11: day (1 – 31) bits 10-6: hour (0 – 23) bits 5-0: minute (0 – 59) word 2: bits15-10: second (0 – 59) bits 9-0: milliseconds (0 – 999)
Checksum	2	Fletcher checksum calculated between Run Time and RTC Time, inclusive.

6.3 The STTP Utility

The SLERJ Time Tagged Parser is a Windows command line utility (sttp.exe) provided with the SSR-1U to parse time tagged archives into various output types. Source code is provided under a non-restrictive (MIT) license so that it can be freely modified and incorporated into user applications. The utility has a number of functions including the ability to extract the raw data (without timestamps), extract line-oriented data prepended with configurable timestamps, extract only portions of the data in intervals or windows, and extract text representations of the packets stored in the archive. For more details on line-oriented extraction, see Application Note *AN002 – STTP Timestamped Lines*.

Usage of the sttp utility is summarized by its help output:

```
usage: sttp.exe [options] <infile>
    Version 1.5, Feb 21 2018 20:30:23
    options:
      -h                  Include headers in tcp and dat files.
      -r <raw_file>       Write raw stream data to raw_file
      -t <tcp_file>       Write Time Correlation Packets to tcp_file
      -d <dat_file>       Write Tagged data to dat_file
      -m <mxd_file>       Write both TCPs and tagged data to mxd_file
      -n <txt_file>       Write timestamped line text file
      -N "string"         Date format string for tagged line (strftime)
      -S                  Supress milliseconds in tagged line output
    Interval extraction options for timestamped line output:
         For the arguments below, 'N' is assumed to be in seconds
         unless suffixed with 'L', which denotes lines.  For example
         '-i 30' denotes an interval of 30 seconds, where '-i 30L'
         Denotes an interval of 30 timestamped lines of data.
      -k,--skip N         Skip N seconds/lines of data before output
      -i,--interval N     Extract excerpts at intervals of N seconds/lines
      -w,--window N       Extract N seconds/lines at each interval
      -v,--nwins M        Process M windows (default=0, to end of file)
```

The tagged line option can extract line-oriented data with user defined timestamps (see the readme.txt file that comes with the STTP utility for details). For example, consider an archive that contains data from an instrument that produces output:

```
S D  0.0000122 kg
S D  0.0000122 kg
S D  0.0000122 kg
S D  0.0000123 kg
```

Extracting this data with options `-n outputfile.txt -N "%m/%d/%Y %H:%M:%S."` would produce an output file:

```
02/03/2014 21:47:38.915 S D  0.0000122 kg
02/03/2014 21:47:39.013 S D  0.0000122 kg
02/03/2014 21:47:39.111 S D  0.0000122 kg
02/03/2014 21:47:39.207 S D  0.0000123 kg
```

For the options that extract the archive packets (-t, -d, and –m), the outputs are space delimited text files. Data bytes are represented as a series of hexadecimal text characters. An example of each of the textual output files is below.

Time Correlation Packet output example:

```
RunTime(ms) Year Month Day Hour Minute Second
4196 2013 3 25 9 52 4.625
604196 2013 3 25 10 2 3.628
1204196 2013 3 25 10 12 2.486
```

Tagged Data output example:

```
RunTime(ms) count HexBytes
4196 20 322E323530333630652B303520322E3339343433
4198 23 30652D3034202D312E343530303639652D303420322E37
4200 23 3637343235652D303420312E373134373036652D303120
```

Mixed output example:

```
A3 4196 2013 3 25 9 52 4.625
A2 4196 20 322E323530333630652B303520322E3339343433
A2 4198 23 30652D3034202D312E343530303639652D303420322E37
…
A2 604194 23 3032202D352E353633313634652D303120312E32323636
A3 604196 2013 3 25 10 2 3.628
A2 604196 23 3330652D303220332E313334343333652B303020302037
```

7 Specifications

7.1 Electrical

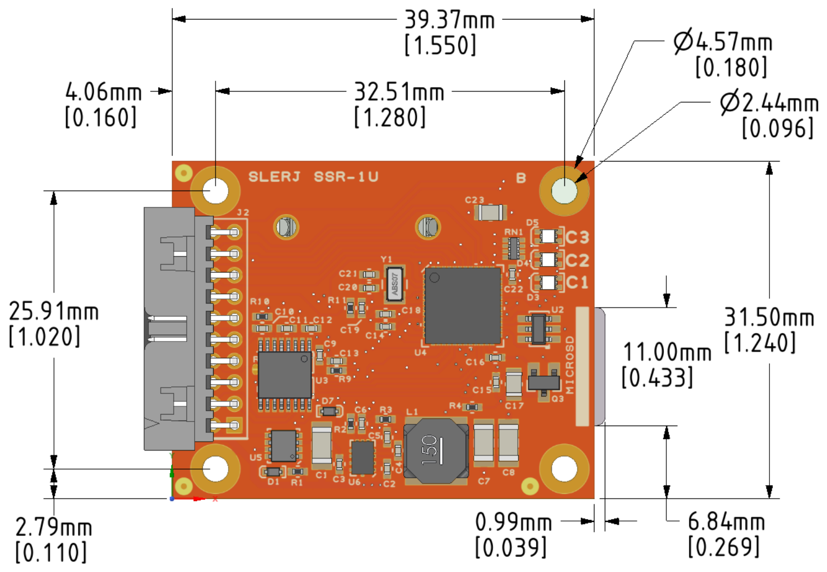
	Min	Typical	Max	Unit
Main Supply Voltage	4.5		32	VDC
Aux (USB) Supply Voltage	4.5	5.0	16	VDC
Supply Current	5 VDC Supply			
Idle[1]		8		mA
Recording[2]		39		mA
Recording[2] – LEDs Off		11		mA
	12 VDC Supply			
Idle[1]		7		mA
Recording[2]		22		mA
Recording[2] – LEDs Off		8		mA
	24 VDC Supply			
Idle[1]		12		mA
Recording[2]		22		mA
Recording[2] – LEDs Off		13		mA
Digital Input Characteristics (Channel 3/4 Rx, DI, PI, Res, UEN, USB)				
Low level input voltage			1.23	V
High level input voltage	1.88			V
Schmitt trigger hysteresis		200		mV
Weak pull-up equivalent resistor	25	40	55	kΩ
USB D+ and D- voltage range	-0.5		6	V
Digital Output Characteristics (Channel 3/4 Tx, S1-S3)				
Low level output voltage (±8mA)			0.4	V
High level output voltage (±8mA)	2.9			V
Low level output voltage (±20mA)			1.3	V
High level output voltage (±20mA)	2.0			V
RS-232 Transmitter Characteristics (Channels 1 and 2)				
Transmitter Output Voltage Range	-13.2		13.2	V
Transmitter Output Voltage into 3kΩ Load		±5		V
RS-232 Receiver Characteristics (Channels 1 and 2)				
Receiver Input Voltage Range	-25		25	V
Positive going input threshold voltage		1.5	2.4	V
Negative going input threshold voltage	0.6	1.2		V

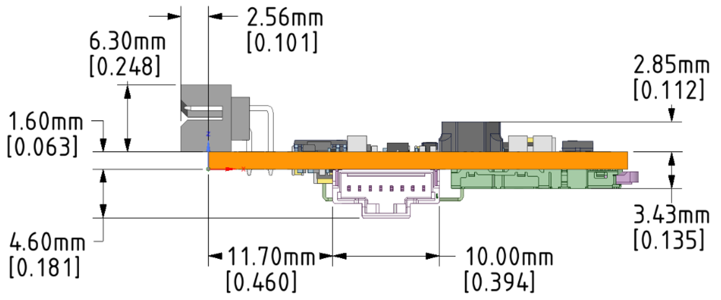
[1] SanDisk Industrial 8GB Class 10 microSDHC card inserted, but no data being received.
[2] SanDisk Industrial 8GB Class 10 microSDHC card inserted, recording three full streams at 115200 baud.

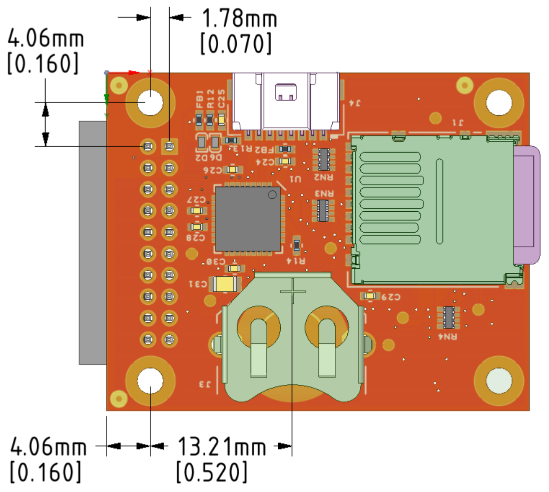
7.2 Environmental

The SSR-1U supports extended temperature operation (-40 to 85C).

7.3 Mechanical







Mounting holes designed for #2 hardware
Main Connector: Molex 87833-2020
Main Con Mate: Molex 87568-2093
Aux Connector: Molex 501568-0807
Aux Con Mate: Molex 15133-0803
Battery: CR1220 or CR1216, 3V button cell

8 Revision History

Date	Rev.	Changes
27 Dec 2019	-	Initial release