Summary of the RPLIDAR A2 Interface Protocol and Datasheet

Umaar Abdullah Morshed

RPLIDAR A2 Pin Definition and Specification

RPLIDAR A2 is using XH2.54-5P specification plug. Please use it with socket that meet the specification of XH2.54-5P. The detailed pin definition is shown as below:

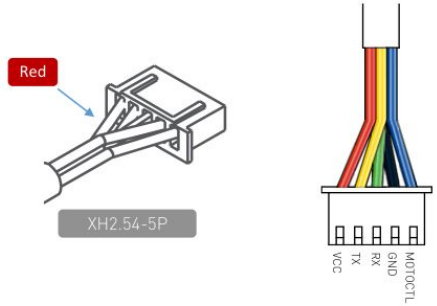


Figure 3-1 RPLIDAR A2 Pins

Color	Signal name	Type	Description	Minimum	Typical	Maximum
Red	VCC	Power	Power supply for the whole RPLIDAR	4.9V	5V	5.5V
Yellow	TX	Output	Serial output for RPLIDAR scan core	0V	3.3V	3.5V
Green	RX	Input	Serial input for RPLIDAR scan core	0V	3.3V	3.5V
Black	GND	Power	GND	0V	0V	0V
Blue	MOTOCTL	Input (pull down)	Enable pin for RPLIDAR scan motor/PWM control signal (active high)	0V	3.3V	5V

Figure 3-2 RPLIDAR Pin Definition and Specification

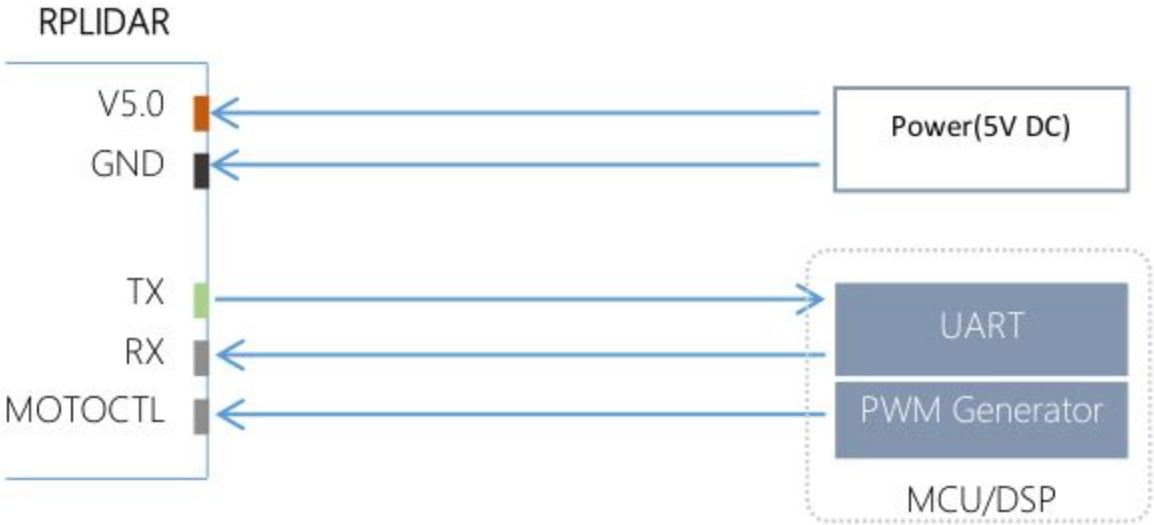


Figure 3-3 RPLIDAR A2 Pins Reference Design

Protocol Basics

Basic Communication Mode

The RPLIDAR uses a non-textual binary data packet based protocol to communicate with host systems. And all the packets transmitted on the interface channel share uniform packet formats.

A communication session is always initialized by a host system, i.e. a MCU, a PC, etc. RPLIDAR itself won't send any data out automatically after powering up.

If a data packet is sent from host systems to RPLIDARs, such a packet is called a **Request.** Once an RPLIAR receives a request, it will reply the host system with a data packet called a **Response**.

RPLIDAR will only start performing related operations required by a host system once after it receives a request. If RPLIDAR should reply to the host system, it will send one or more required response packets.

In order to let an RPLIDAR start scanning operation and send out data, a host system is required to send a pre-defined **Start Scan** request packet to RPLIDAR. RPLIDAR will start scanning operation once after it receives the request and the scan result data is sent out to the host system continuously.

Request/Response Modes

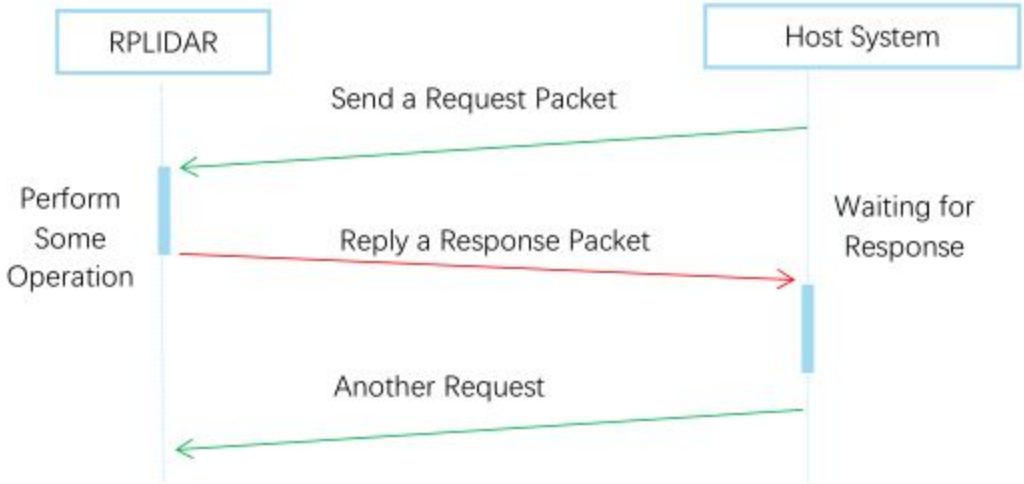


Figure 2-1 RPLIDAR Request/Response Modes

Request Packets' Format

All request packets sent by a host system share the following common format. Little endian byte order is used.

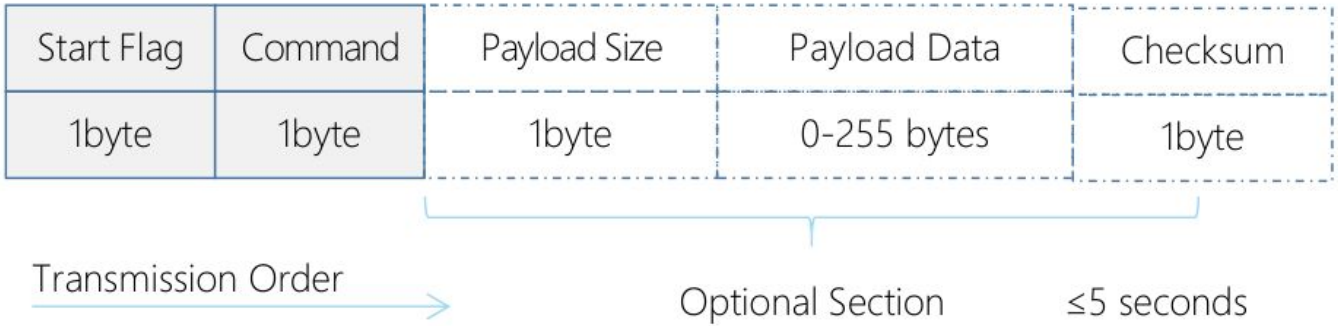


Figure 2-4 RPLIDAR Request Packets' Format

A fixed 0xA5 byte is used for each request packet, RPLIDAR uses this byte as the identification of a new request packet. An 8bit (1byte) command field must follow the start flag byte.

Major Working States and Transition Conditions

RPLIDAR has the following 4 major states: Idle, Scanning, Request Processing and the Protection Stop state.

The translation conditions are depicted in the following figure:

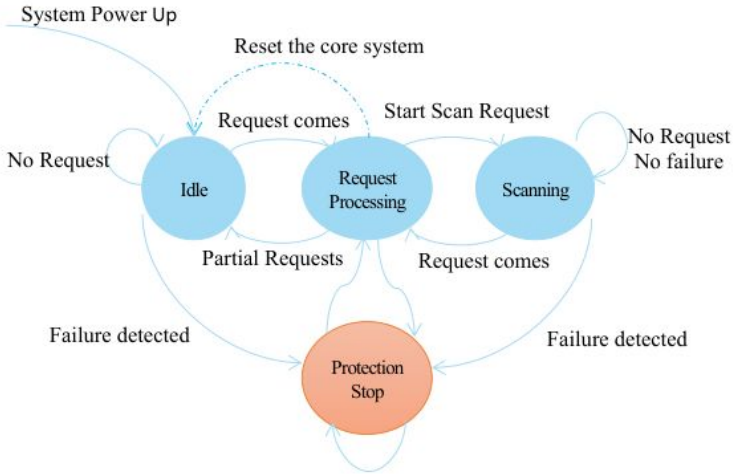


Figure 3-1 RPLIDAR's Major Status Translation

Requests Overview

Request Name	Value	Payload	Response Mode	RPLIDAR Operation
STOP	0x25	N/A	No response	Exit the current state and enter the idle state
RESET	0x40	N/A		Reset(reboot) the RPLIDAR core
SCAN	0x20	N/A	Multiple response	Enter the scanning state
EXPRESS_SCAN	0x82	YES		Enter the scanning state and working at the highest speed
GET_INFO	0x50	N/A	Single response	Send out the device info (e.g. serial number)
GET_HEALTH	0x52	N/A		Send out the device health info
GET_SAMPLERATE	0x59	N/A		Send out single sampling time
GET_LIDAR_CONF	0x84	YES		Get LIDAR configuration

Figure 4-1 The Available Requests of RPLIDAR

Response Packets' Format

All the response packets are divided into two classes: **response descriptors** and **data responses**. If the current request received by RPLIDAR requires a response, RPLDAR will always send a response descriptor packet first and then send one or more data response packets based on the type of requests. Only one response descriptor packet will be sent out during a request/response session. The response descriptors carry the information of the incoming data responses. All the response descriptors share a same format.

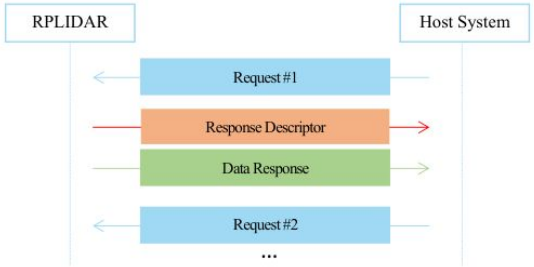


Figure 2-5 Response Packets Sent during a Single Request-Single Response Mode

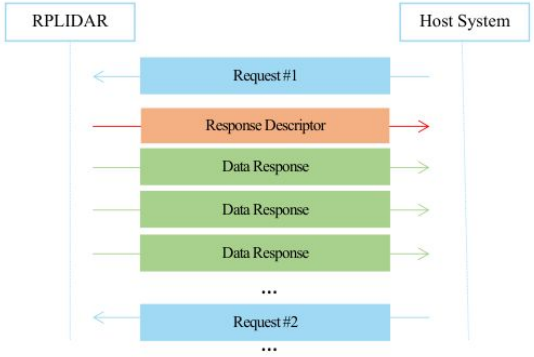


Figure 2-6 Response Packets Sent during a Single Request-Multiple Response Mode

The format of response descriptors is depicted in the following figure.

Start Flag1	Start Flag2	Data Response Length	Send Mode	Data Type
1byte (0xA5)	1byte (0x5A)	30bits	2bits	1byte

Transmission Order

Figure 2-7 RPLIDAR Response Descriptors' Format

The format of response descriptors is depicted in the following figure.

Start Flag1	Start Flag2	Data Response Length	Send Mode	Data Type
1byte (0xA5)	1byte (0x5A)	30bits	2bits	1byte

Transmission Order

Figure 2-7 RPLIDAR Response Descriptors' Format

A response descriptor uses fixed two bytes' pattern 0xA5 0x5A for the host system to identify the start of a response descriptor. The 30bit Data Response Length field records the size of a **single** incoming data response packet in bytes. (All the incoming data response packets within a request/response session should have the same format and length). The 2bits Send Mode field describes the request/response mode of the current session. Its values are listed below:

Send Mode	Description
0x0	Single Request – Single Response mode, RPLIDAR will send only one data response packet in the current session.
0x1	Single Request – Multiple Response mode, RPLIDAR will continuously send out data response packets with the same format in the current session.
0x2	Reserved for future use
0x3	Reserved for future use

Figure 2-8 RPLIDAR Data Response Packets Value

The 1byte Data Type describes the type of the incoming data response packets. It is related to the type of the request RPLIDAR just received. Host systems can choose different data receiving and handling policy based on this field.

Different from response descriptors, there is no common format used among response data packets. Each type of response data has its own data format and packet length based on its type.

Single Response Command

Get Device Health Status (GET_HEALTH) Request and Response

Request Packet:

A5	52

Response Descriptor:

A5	5A	3	00	00	00	06

Response Mode: **Single** Data Response Length: **3 bytes**

A host system can send the GET_HEALTH request to query RPLIDAR's health state. If the RPLIDAR has entered the Protection Stop state caused by hardware failure, the related error code of the failure will be sent out.

Format of the Data Response Packets

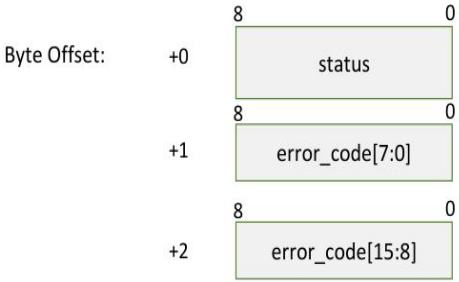


Figure 4-20 Format of a Device Health Data Response Packet

The following table describes the meanings of each field in the preceding packet:

Field Name	Description	Examples / Notes
status	RPLIDAR Health status	Value definition: 0: Good 1: Warning 2: Error When the core system detects some potential risk that may cause hardware failure in the future, the status value will be set to Warning(1). But RPLIDAR can still work as normal. When RPLIDAR is in the Protection Stop state, the status value is set to Error(2).
error_code	Specific warning / Error code	When a warning or error status occurs, the specific error code is recorded in this field.

Figure 4-21 Field Definition of Device Health Status Data Response Packet

When a host system detects RPLIDAR has entered the Protection Stop state, it can set a RESET request to let RPLIDAR core system reboot to escape the Protection

Multiple Data

response

Start Scan (SCAN) Request and Response

Note: RPLIDAR device **models support** 4khz or higher sampling rate will lower the sampling rate when processing this request. Please use EXPRESS_SCAN for the best performance.

This command supports Legacy scan mode only.

Copyright (c) 2009-2013 RoboPeak Team
Copyright (c) 2013-2023 Shanghai Slamtec Co., Ltd.

Request Packet:	A5	20

Response Descriptor:	A5	5A	05	00	00	40	81

Response Mode: **Multiple** Data Response Length: **5 bytes**

RPLIDAR, except for the RPLIDAR that is in the Protection Stop State, will enter the scanning state once it receives this request from a host system. Each measurement sample result will be sent out using an individual data response packet. If the RPLIDAR has been in scanning state already, it will stop the current measurement sampling and start a new round of scanning. This request will be ignored when RPLIDAR is in the Protection Stop state.

Format of the Data Response Packets:

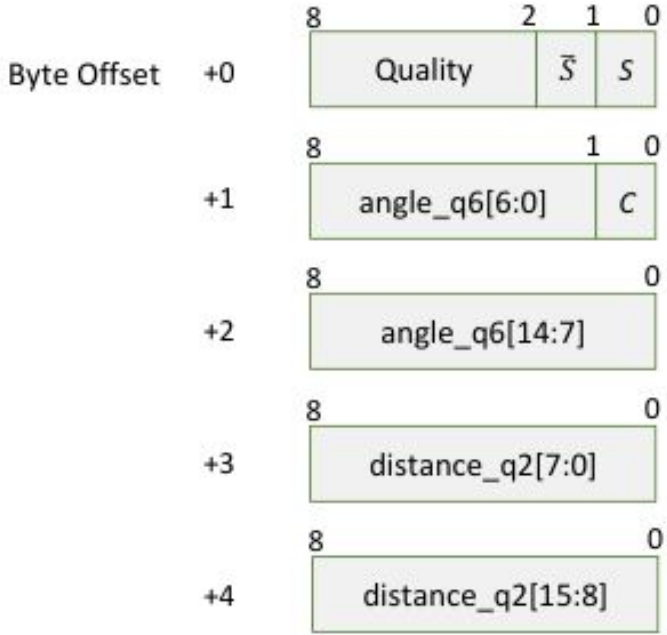


Figure 4-4 Format of a RPLIDAR Measurement Result Data Response Packet

Typical work flow of retrieving scanning data from an RPLIDAR

It is recommended that a host system always follows the below sequence to enable RPLIDA's scanning operation and retrieve the scanning data. Before sending a SCAN request, the host system should send a GET_HEALTH request in advance to query the RPLIDAR's health status. In case RPLIDAR is in the Protection Stop state, the host system can send a RESET request to try to escape the Protection Stop state. Please refer to the SDK code for implementation details.

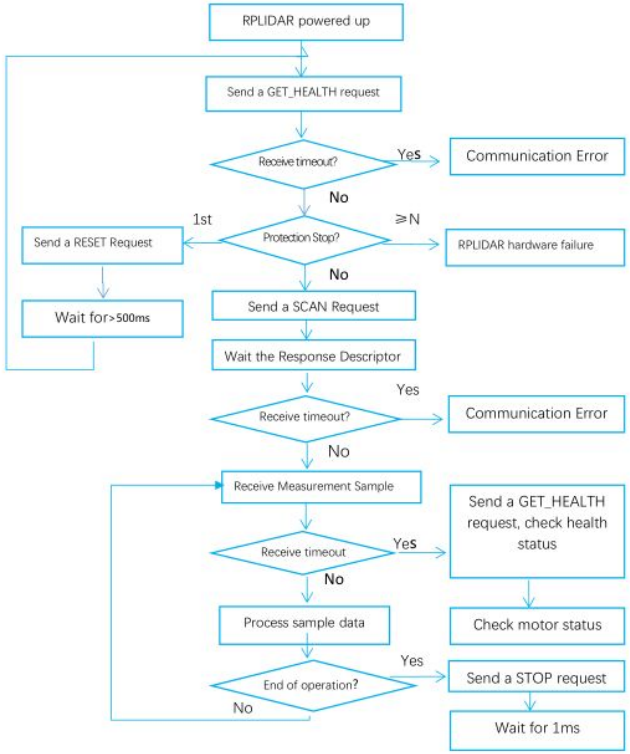


Figure 5-1 Recommendation for Starting RPLIDAR Scanning and Data Retrieving

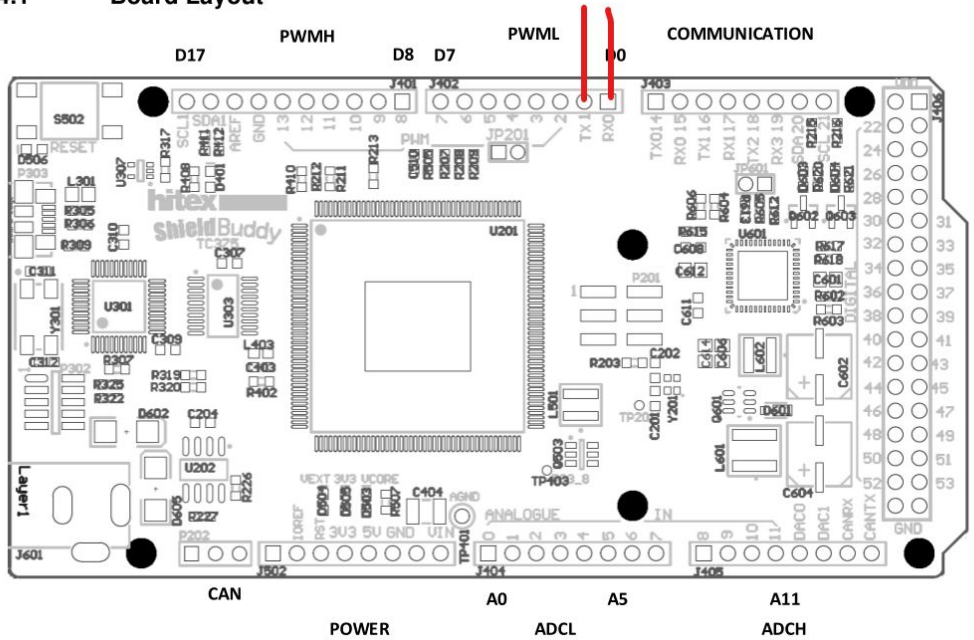
Aurix TC375 RPLIDAR Library Documentation

Umaar Abdullah Morshed

- UART Pins (UART port ***Asclin0***)

```
/* TX RX Pin Assignment */
#define UART_PIN_RX             &IfxAsclin0_RXB_P15_3_IN                /* UART receive port pin
#define UART_PIN_TX             &IfxAsclin0_TX_P15_2_OUT                /* UART transmit port pin
```

4.1 Board Layout

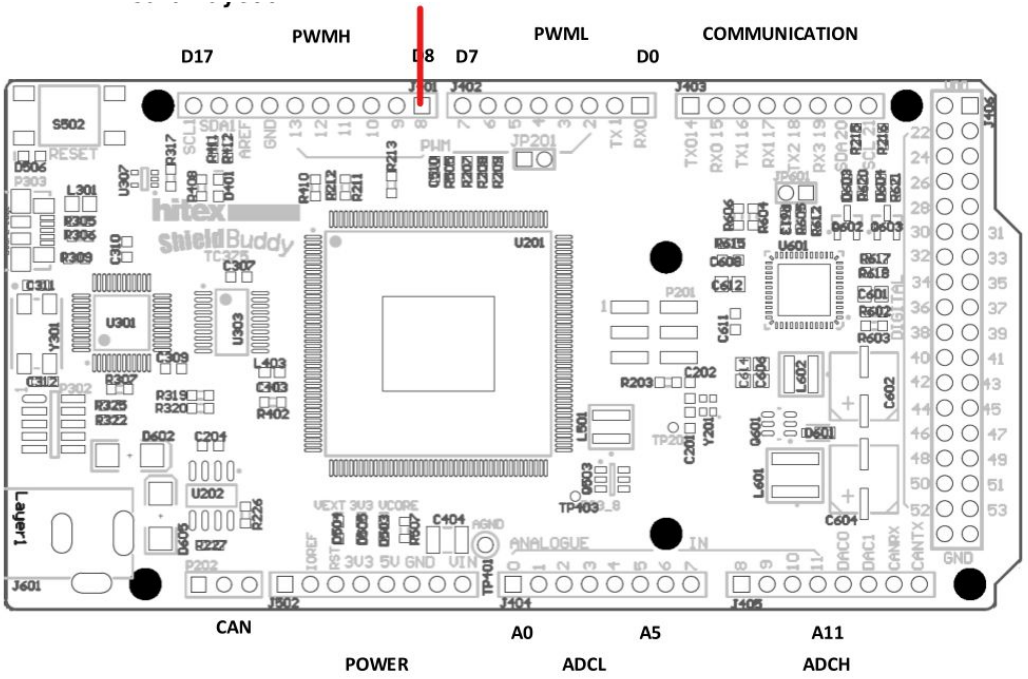


UART_SHELL Pins (UART port ***Asclin3***) (Fixed)

```
/* Pin configuration */
const IfxAsclin_Asc_Pins pins = {
        .cts        = NULL_PTR,                     /* CTS pin not used
        .ctsMode    = IfxPort_InputMode_pullUp,
        .rx         = &IfxAsclin3_RXD_P32_2_IN ,    /* Select the pin for RX connected to the USB port
        .rxMode     = IfxPort_InputMode_pullUp,     /* RX pin
        .rts        = NULL_PTR,                     /* RTS pin not used
        .rtsMode    = IfxPort_OutputMode_pushPull,
        .tx         = &IfxAsclin3_TX_P15_7_OUT,     /* Select the pin for TX connected to the USB port
        .txMode     = IfxPort_OutputMode_pushPull,  /* TX pin
        .pinDriver  = IfxPort_PadDriver_cmosAutomotiveSpeed1
};
ascConf.pins = &pins;
```

PWM Pin

```
#define EN1                IfxGtm_TOM0_3N_TOUT6_P02_6_OUT          /* PWM pin 8
```



UART Interrupt Priority

```
17 /* Communication parameters */
18 #define ISR_PRIORITY_ASCLIN_TX      8
19 #define ISR_PRIORITY_ASCLIN_RX      4
20 #define ISR_PRIORITY_ASCLIN_ER      12
21 #define ASC_TX_BUFFER_SIZE          256
22 #define ASC_RX_BUFFER_SIZE          256
23 #define ASC_BAUDRATE                115200
```

UART_SHELL Interrupt Priority

```
22 /* Communication parameters */
23 #define ISR_PRIORITY_ASCLIN_TX      9
24 #define ISR_PRIORITY_ASCLIN_RX      5
25 #define ISR_PRIORITY_ASCLIN_ER      13
26 #define ASC_TX_BUFFER_SIZE          256
27 #define ASC_RX_BUFFER_SIZE          256
28 #define ASC_BAUDRATE                115200
```

Debugging Notes

- Had to change priority of uart_shell interrupt. Wouldn’t have been able to do that if I had not been familiar with NVIC.