PARSING A SERIAL STRING WITH A SCRIPT

1. Introduction

Serial communication using RS232 and RS485 is commonly used as a communications technique when interfacing to sensors and machines. There are common protocols such as NMEA which establish how data is sent over serial and are often based on ASCII text separated by spaces, commas, or other delimiters.

Other serial strings may be binary or hex, or the position of an element in the string may dictate its meaning. These strings can also be parsed with simple text and text to value manipulation commands. An example of a serial string where the location of the data dictates its meaning would be a CAN bus message which conforms to a standard such as J1939 commonly found in industrial vehicles.

This application note discusses the parsing of serial strings that are separated by known delimiters or are at known positions. Parsing a serial string allows individual pieces of information to be extracted so that they can be turned into measurements, mathematically manipulated, and used to create exceptions. Parsing of CAN messages will be detailed in a separate application note.

2. Receiving the Serial Data

The Senquip ORB has a serial interface that can be configured as RS232 or RS485, and that can implement protocols like MODBUS. Where MODBUS is used over RS485 or RS232, parsing is not required as the MODBUS protocol returns individual measurements.

The Senquip serial port can be configured to start capturing data based:

- A start string, this is typically a known set of characters such as "$GPRMC" in the case of the NMEA-0183 position, velocity, and time message.
- After a period of no serial activity. This can be useful where there is no known start to the serial data being received.

Capturing can be ended after:

- A known end character or set of characters is received, carriage return, and line feed are common examples.
- After a number of characters is received
- After a specified period of time.

For further details on capturing serial data, please refer to the ORB User Guide: http://docs.senquip.com/orbug/

This application note will assume that the following example serial string has been received and will focus on the script to parse it.

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A<CR><LF>

The RMC message always starts with $GPRMC, always ends with a carriage return and line feed, has the data separated by commas and includes a checksum as the last element of the message. It would be prudent, when writing a script to parse serial data, to use the elements of the serial string to perform checks on the validity of the data.

Field	Meaning
0	Message ID
1	UTC of position fix
2	Status A=active or V=void
3	Latitude
4	Longitude
5	Speed over the ground in knots
6	Track angle in degrees (True)
7	Date
8	Magnetic variation in degrees
9	The checksum data, always begins with *

3. Parsing the ASCII String

We are assuming that the serial message is being correctly received by the Senquip ORB. Figure 2 shows the RMC string arriving at the Senquip Portal. Note that the serial message is configured to be shown as Escaped Text. It could also be shown in hex or Base64. Base64 is a common method for encoding serial data when sending it as text. All ORB data is sent in JSON format and so is text based. Encoding in Base64 ensures that data is correctly received whether it be text or binary data. There are free Bas64 encoders and decoders available online.

To see the received string in Base64, view the raw JSON data in the *Raw Data* window. Figure 1 shows the Base64 encoded version of the RMC message. Note also that the key in the JSON data below is capture1. The key will be used to retrieve the data from the ORB.

"capture1":
"JEdQUk1DLDEyMzUxOSxBLDQ4MDcuMDM4LE4sMDExMzEuMDAwLEUsMDIyLjQsMDg0LjQsMjMwMzk0LDAwMy4xLFcqNkENCg=="

Figure 1 - Base64 encoded string

Let us assume in our example, that we want to extract speed over the ground from the RMC message and that we want to display it in km/h and not knots as is native in the message.

We can see from the description of the RMC message that we need to extract field number 5 “Speed over the ground in knots”, this field can be found after the 7th comma and is 22.4 knots. To scale from knots to km/h, we will have to multiply by 1.852.

$$kilometers\ per\ hour = 1.852\ \times knots$$

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd (“the Company”) is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

Once parsed and converted to km/h, the speed will be dispatched to the Senquip Portal for display and storage.

Figure 2 - RMC data being received by an ORB

4. Writing the Script

In this application note, we will assume that you have access to the scripting feature, that the ORB is subscribed to a Premium plan, and that you are familiar with the scripting environment on the Senquip Portal. For further details on scripting on the Senquip ORB, please see the scripting guide: https://docs.senquip.com/scripting_guide.

To be able to monitor the output, two custom parameters, ground speed in knots and km/h, will be generated and displayed on the Senquip Portal. Figure 3 shows the creation of the custom parameters cp1 (knots) and cp2 (km/h).

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

Custom Data Parameters

Unsaved Changes

✓	[cp1]	Ground Speed	knots
✓	[cp2]	Ground Speed	km/h
✗	[cp3]	Name	Units
✗	[cp4]	Name	Units
✗	[cp5]	Name	Units
✗	[cp6]	Name	Units

+ Add Parameter

Save Changes

Figure 3 - Two custom parameters are created

Figure 4 shows the script to parse the RMC message and convert it to km/h. A brief explanation for each line is given below.

1) Load the Senquip Library that includes all the functions prefixed with SQ
3) Setup the function that will be executed at each base interval, after the measurements have been taken
5) We parse the JSON data to create an object that can be indexed by the JSON key
6) Retrieve the Base64 encoded serial data from capture 1
7) Decode the Base64 serial data so we now have an ASCII string that starts with $GPRMC….
8) Split the RMC string into its components as separated by commas
9) Retrieve item 7, the speed over ground in knots, from the split RMC string
10) Convert the ASCII string into a number in 2 parts, in this line the units, from position 0, 3 characters (022) in decimal
11) Convert the ASCII fractions into a number
12) Add the units and fractions to create speed over ground in knots
13) Convert knots to kilometers per hour
15) Dispatch the speed over ground in knots to the Senquip Portal to be shown as custom parameter 1.
16) Dispatch the speed over ground in km/h to the Senquip Portal to be shown as custom parameter 1

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

19) Close the function

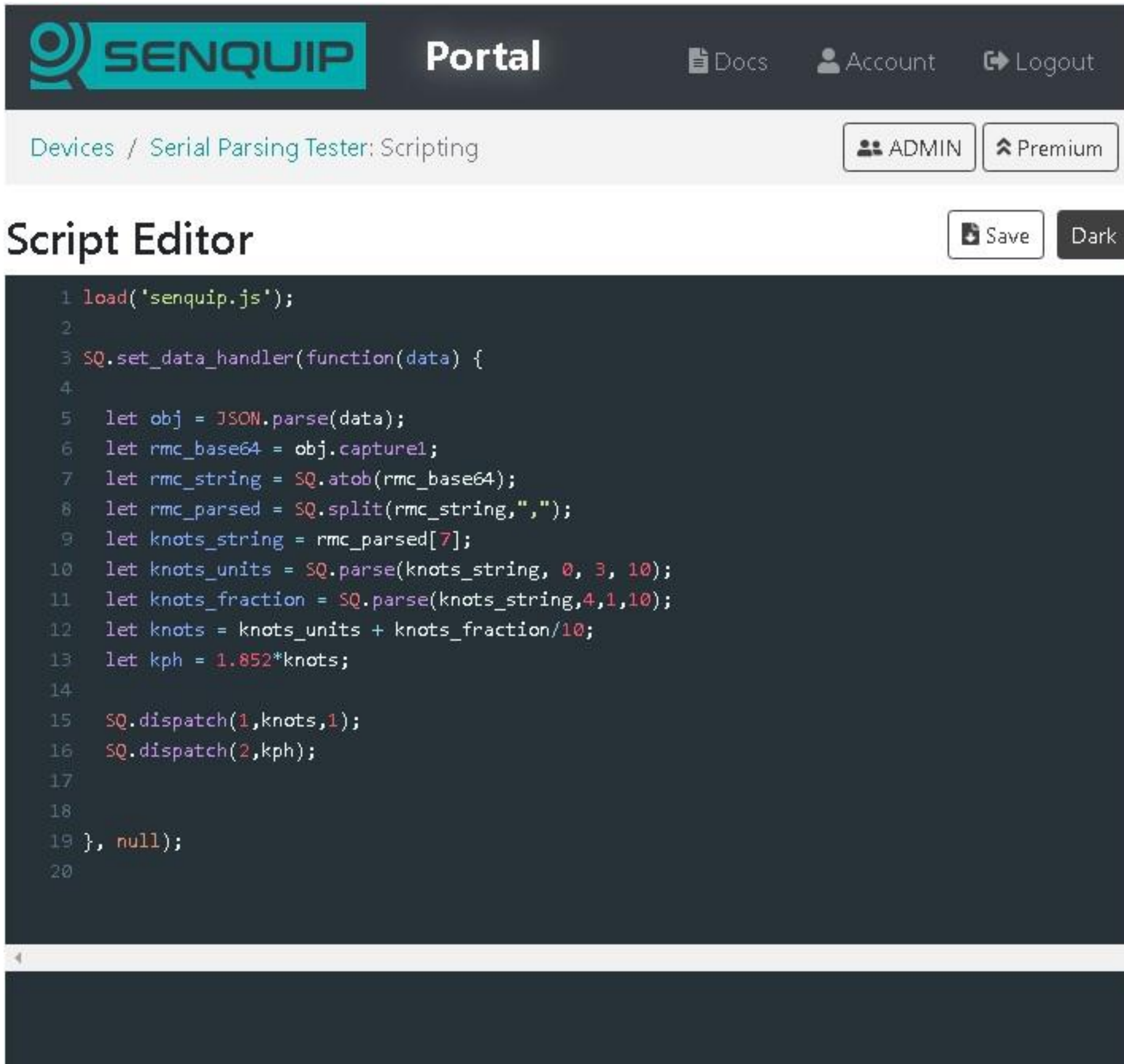
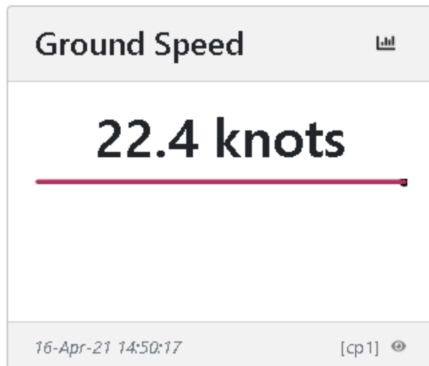
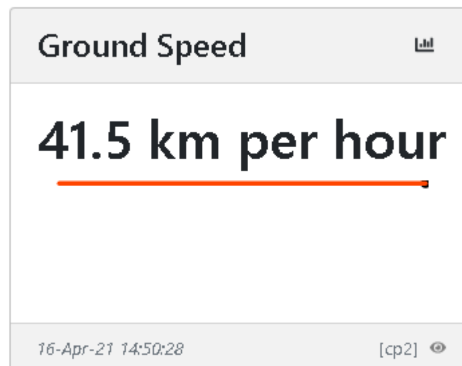


Figure 4 - Script to parse a serial string

Once the script is downloaded and the ORB is rebooted, the function runs, and the portal now shows speed over ground in knots and kilometers per hour.

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.





The code as presented has been written to be simple. It is unnecesarily verbose and an effort should be made to compress the code to conserve memory on the ORB.

The code should be further developed to perform checks on the data to ensure integrity and to protect the JavaScript interpreter from invalid data.

5. Conclusions

Users can write their own JavaScript for the Senquip ORB. Scripting allows serial messages to be parsed to extract components. Those components can then be converted to numbers, which allows them to be manipulated mathematically, and to create complex exceptions. The calculated values can be dispatched to the Senquip Portal or another server for display and storage.

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.

APPENDIX 1: SOURCE CODE

```
load('senquip.js');

SQ.set_data_handler(function(data) {

  let obj = JSON.parse(data);
  let rmc_base64 = obj.capture1;
  let rmc_string = SQ.atob(rmc_base64);
  let rmc_parsed = SQ.split(rmc_string,",");
  let knots_string = rmc_parsed[7];
  let knots_units = SQ.parse(knots_string, 0, 3, 10);
  let knots_fraction = SQ.parse(knots_string,4,1,10);
  let knots = knots_units + knots_fraction/10;
  let kph = 1.852*knots;

  SQ.dispatch(1,knots,1);
  SQ.dispatch(2,kph);

}, null);
```

Copyright © 2020 Senquip Pty Ltd. Senquip Pty Ltd ("the Company") is the owner of the copyright and all confidential information in this document. The document and its text, images, diagrams, data and information it contains must not be copied or reproduced in whole or in part, in any form or by any means, without the prior written consent of the Company.