Smart Technology. Delivered.™

RM1xx Debug Features

RM1xx Series

Application Note *v1.0*

Introduction

The Laird RM1xx series of LoRaMAC modules are a combination of a Nordic Semiconductor based Laird BL620 BLE Central device and the Semtech SX1272 860MHz to 1020 MHz low power, long range transceiver.

The RM1xx is designed to send data to a remote network application, such as a website or database via a gateway device which receives RF data packets from the RM1xx and then passes this data onto the network application via a TCP/IP connection.

The RM1xx module is very much a black box. There is little feedback to the user as to what is happening. During the development process, this can be very frustrating and could lead to slower development.

To help overcome this, in firmware versions 17.4.1.0 and 18.4.1.0 and later, a new debug *smart*BASIC API and three new events have been created to aid debugging.

Events

As previously stated, three new smartBASIC events have been created to aid debugging.

EVLORAMACTXDONE

This is output when the TxDone signal is received by the Nordic chip. This signal is sent from the SX1272 and indicates that a packet of data has been transmitted from the radio. This includes JoinRequests,data packets or LinkChecks. It is a very important signal as the timings of the two RxWindows are calculated from the time this signal is received.

EVLORAMACNOSYNC

This event is output when an RxWindow fails to receive a sync pulse at the expected time and so closes. As there are two possible RxWindows, there can be two EVLORAMACNOSYNC events for each transmitted packet.

It’s possible to infer from this event which RxWindow, if any, a downlink packet was received in. If the event is never called, then the downlink packet was received in the first RxWindow. If one event is received, then the packet was received in the second RxWindow. If two events are received, then no packet was received from the Gateway.

The length of time that a receive window is open looking for this pulse is dependent on the number of symbols it is looking for and the data rate. This is discussed in greater detail later in this document.

EVLORAMACADR

This event indicates that an ADR command was received from the gateway and that some configuration parameters, specifically the Datarate, ChannelsMask or TxPower, MAY have changed. On receipt of this signal, it is worth checking these parameters. The ADR command is sent frequently from the Gateway so it is possible that the parameters are already set as required and no change will take place.

Debug Command

The new *smart*BASIC debug API outputs various radio transmit and receive parameters as a text string. More importantly, it also outputs waveforms on two configurable SIO pins that indicate when the LoRa radio is in Tx or Rx mode.

Command Syntax	LoramacSetDebug(nEnable, nTxSio, nRxSio)
Parameters:	
nEnable	1 – Enable debug feature 0 – Disable debug feature (default)
nTxSio	SIO pin on which the transmit waveform is output. The waveform defaults high. When the module goes into transmit mode, the waveform goes low. It returns to high when it receives the TxDone signal (see EVLORAMACTXDONE) or the transmit timeout is triggered.
nRxSio	SIO pin on which the receive waveform is output. The waveform defaults high. When the module enters receive mode, the waveform goes low. It returns to high when it receives the RxDone signal or the receive window times out (see EVLORAMACNOSYNC).

There are several spare SIO pins that are available for use on the RM1xx with the LoramacSetDebug command, specifically SIOs 4, 5, and 6. SIOs 0, 3, and 17 are also good possibilities if not already being used. If an SIO is already used, you must close it before it can be used with this command.

When enabling debug mode, nTxSio and nRxSio cannot be the same SIO pin. An error message is output if this is attempted.

When disabling debug mode, nTxSio and nRxSio can be the same value. We recommend that you use LoramacSetDebug(0, 0 , 0) in this instance.

Debug Text Output

When the module goes into transmit mode, the frequency on which the packet is to be output and the spreading factor of that packet are output as a text string.

When the module goes into receive mode, the data rate and the symbols timeout value are output as a text string.

For an example, refer to the Example Waveforms section.

Example Waveforms

You can get a very good idea of what is happening with your LoRa module by looking at the output transmit and receive waveforms. For example, in Figure 1 you can see an example of a failed JoinRequest to a gateway. Figure 2 shows the resulting debug. This example was carried out on an RM186.

Transmit is the blue and receive the red waveforms respectively.

EVLORAMACTXDONE and EVLORAMACNOSYNC are enabled and output “Tx Done” and “No Sync pulse” if the event is triggered.

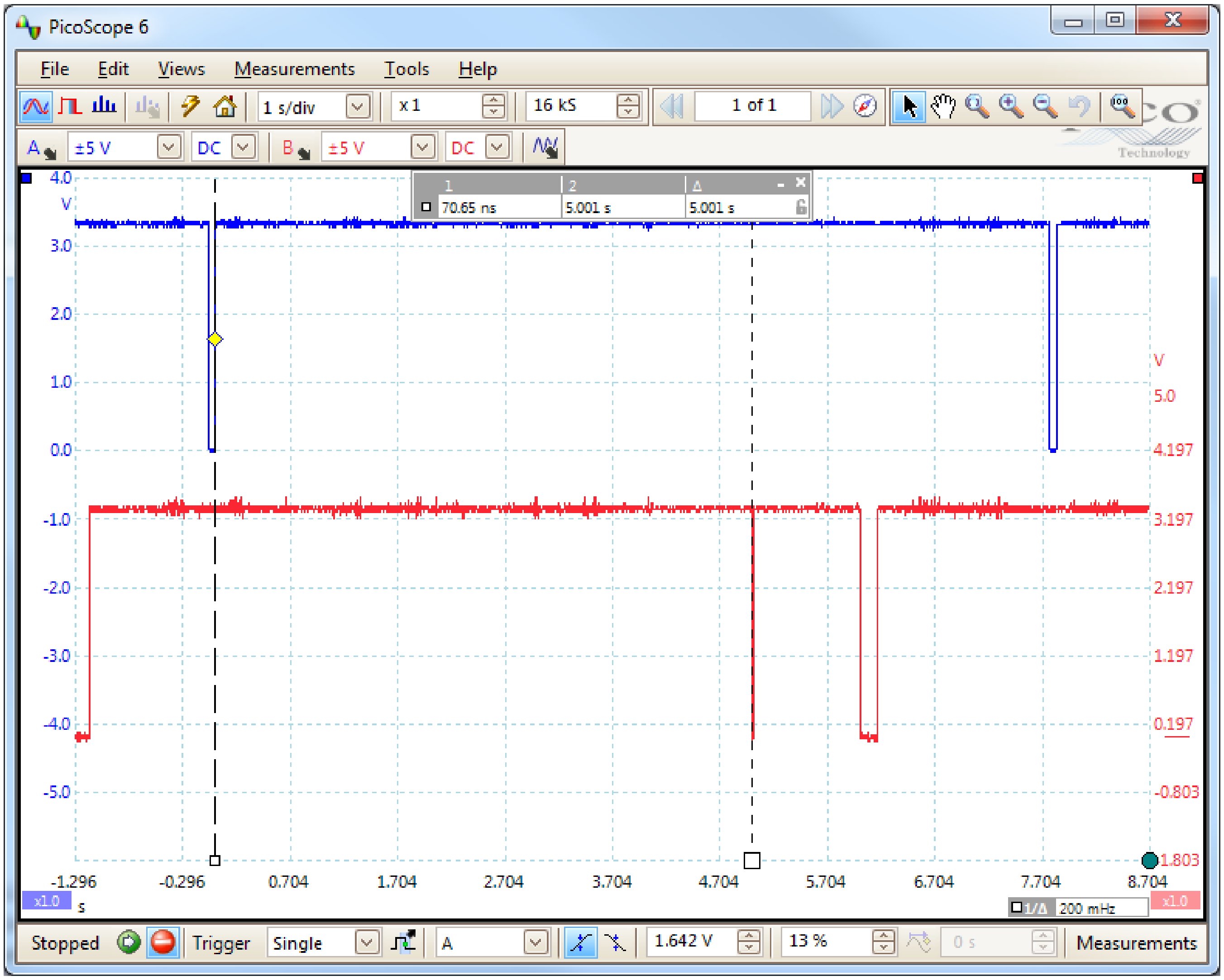
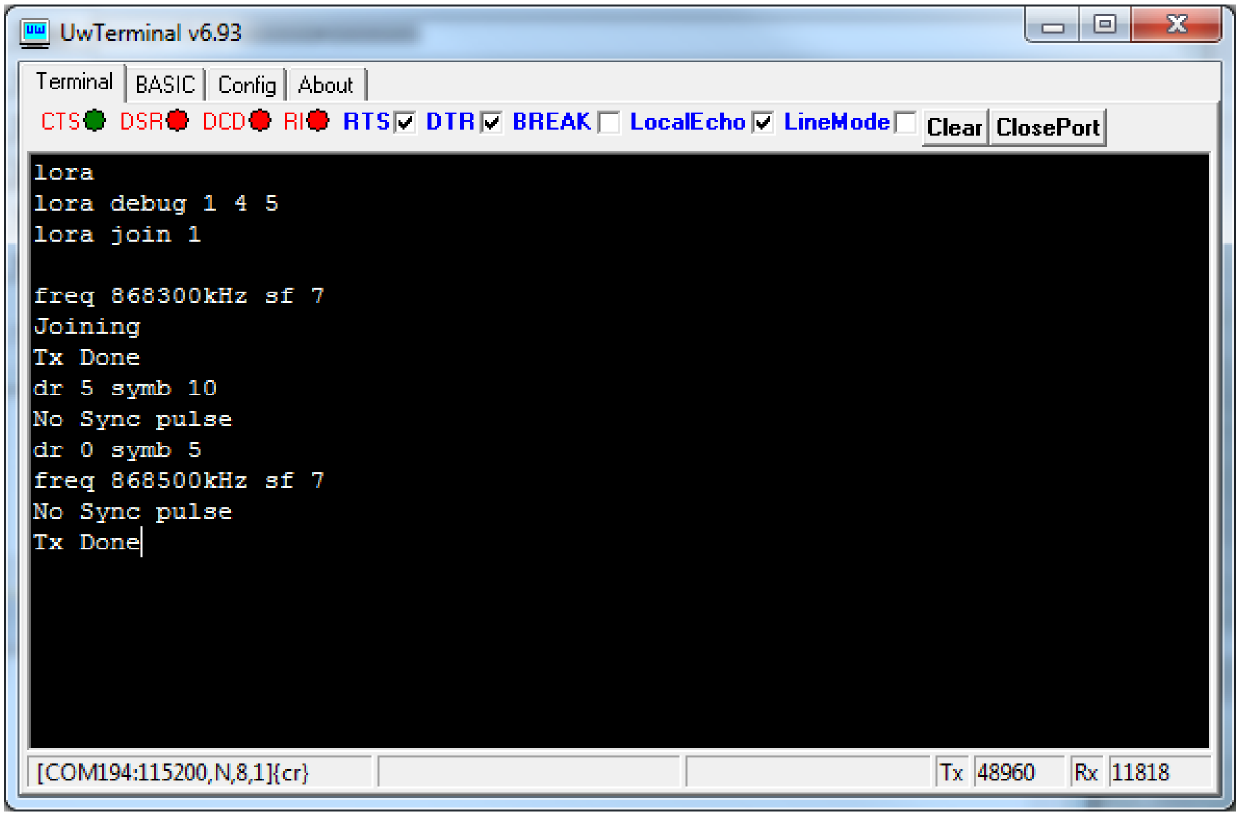


Figure 1 : Failed JoinRequest



```
lora
lora debug 1 4 5
lora join 1

freq 868300kHz sf 7
Joining
Tx Done
dr 5 symb 10
No Sync pulse
dr 0 symb 5
freq 868500kHz sf 7
No Sync pulse
Tx Done
```

Figure 2: Failed JoinRequest Debug

We can see from Figure 1 and Figure 2 that a JoinRequest was transmitted at 863 MHz and the TxDone is received. Five seconds later, the transmit line goes high (TxDone) and the first receive window opens. The fact that the second receive window opens a second later indicates that the JoinAccept wasn't received in the first window.

You can see that, according to the debug, a second packet appears to be transmitted before the 2nd Receive window is closed. This is due to the internal delays in dealing with the events and when debug is output. You can see from the waveform in Figure 1 that the resend is after the 2nd Receive window closes.

Another indication that no packet was received is the size of the windows. If you zoom in on the first receive window (Figure 3), you can see how long the window was open, which is approximately 11.5 milliseconds.

From the debug screen we can see that, for the first receive window, the data rate is 5 and the symbol timeout is 10. From the LoRaWan specification, data rate 5 equates to SF7 @125 kHz. So the minimum time the window needs to be open is

```
(Spreading factor/bandwidth) * Number of symbols
```

So in this case this is

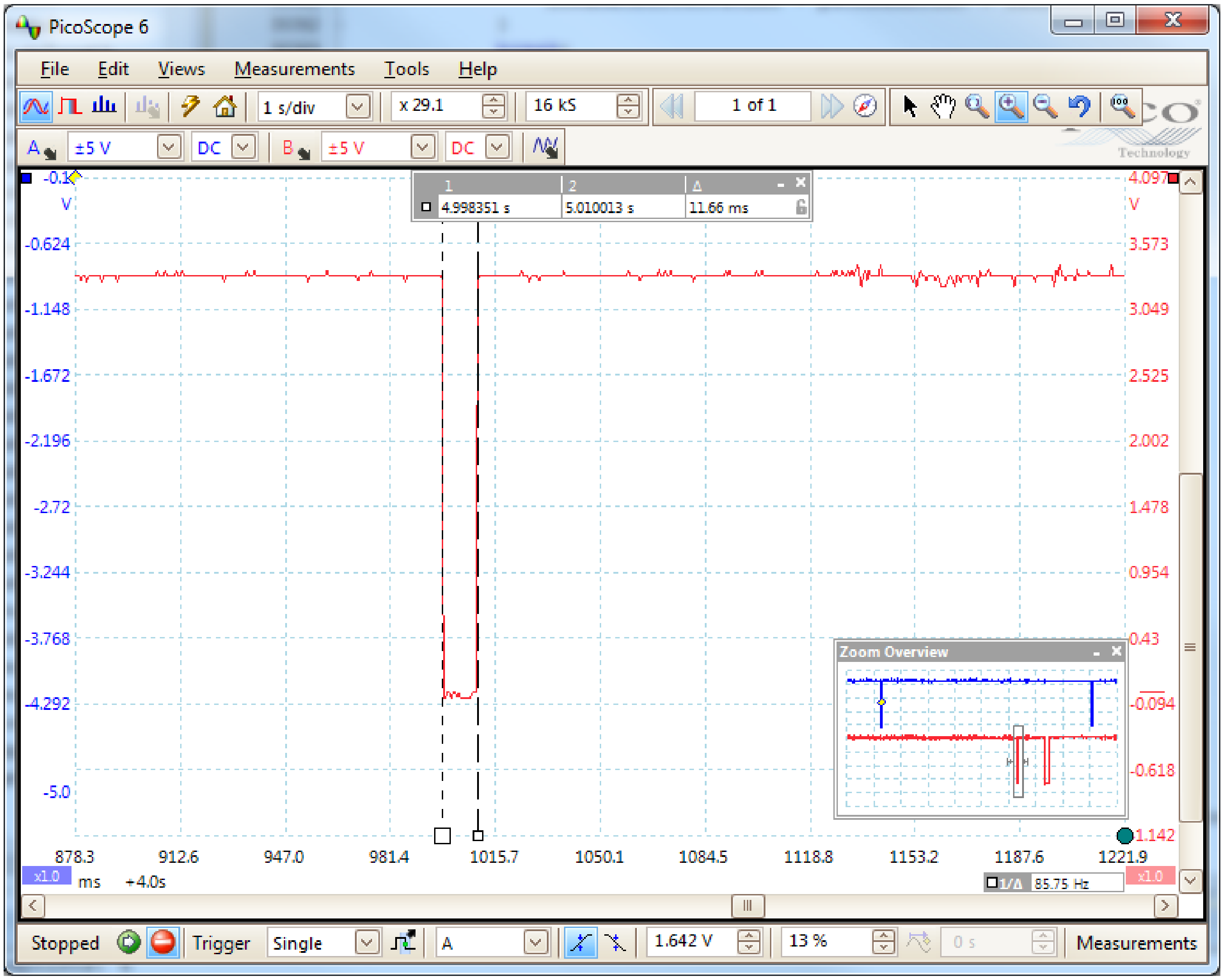
```
(128/125kHz) * 10 = 10.24ms
```

which, allowing for delays in the system, is approximately the length measured in the waveform.

Doing a similar calculation for the second receive window, where data rate 0 equates to SF12@125 kHz, we get

```
(4096/125kHz) * 5 = 163.84ms
```

which, if you zoom in on the pulse, you can see is the approximate duration of the receive window.



1	2	Δ
4.998351 s	5.010013 s	11.66 ms

1/Δ 85.75 Hz

Figure 3: Zoomed in Rx Window1

An example of a successful JoinRequest is shown in Figure 4 and Figure 5.

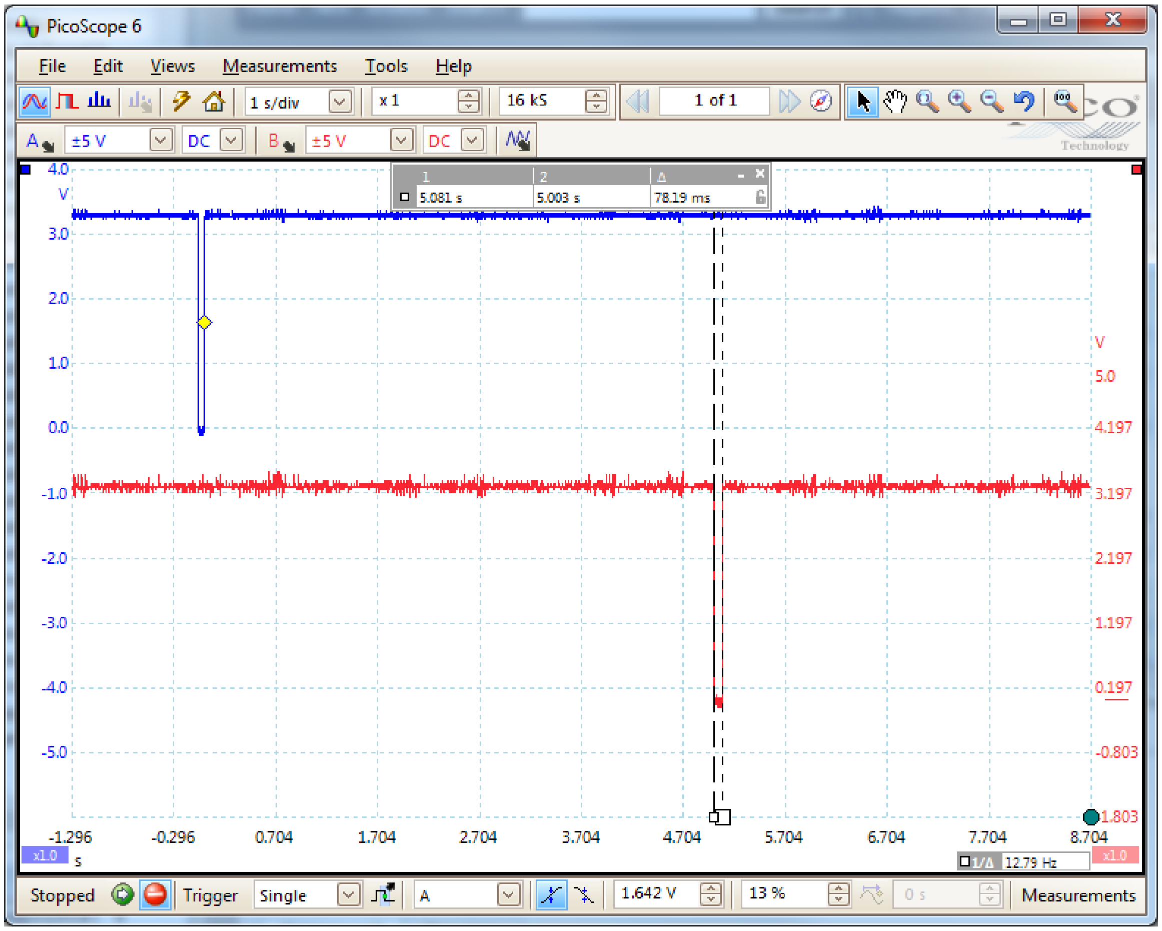
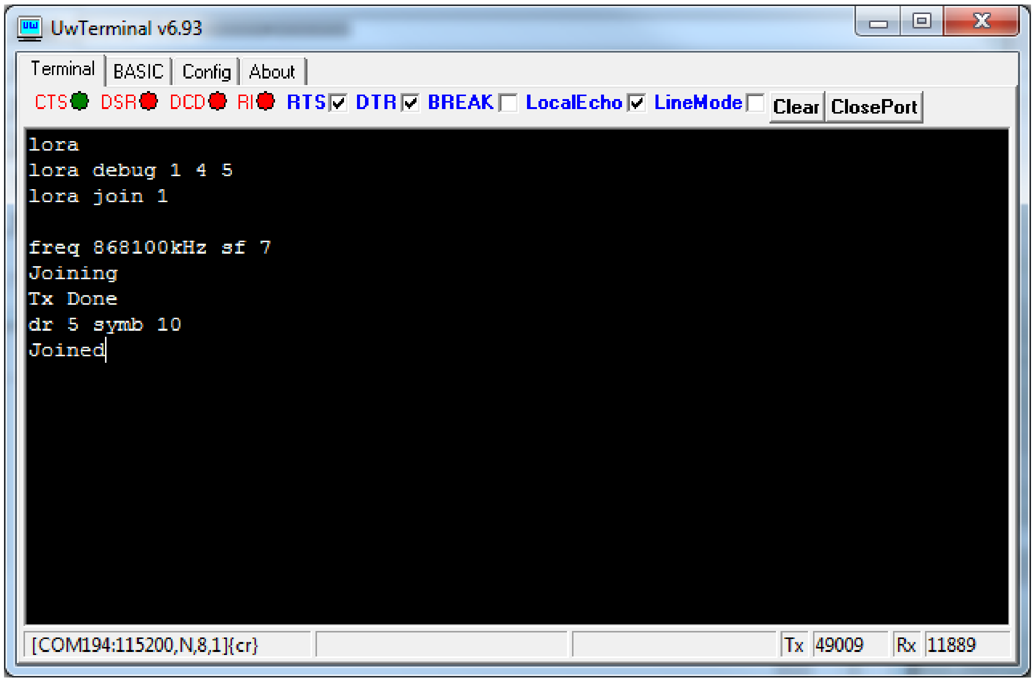


Figure 4 : Successful Join Request


UwTerminal v6.93
Terminal BASIC Config About
CTS DSR DCD RI RTS DTR BREAK LocalEcho LineMode Clear ClosePort
[COM194:115200,N,8,1]{cr}
Tx 49009 Rx 11889

```
lora
lora debug 1 4 5
lora join 1

freq 868100kHz sf 7
Joining
Tx Done
dr 5 symb 10
Joined
```

Figure 5: Successful JoinRequest Debug

You can see in this instance that there is no “No Sync pulse” debug message as it was received in the first receive window. Also, the receive window is much wider; approximately 78 milliseconds. From this, we can calculate that there were at most 76 bytes in the JoinAccept packet.

```
78ms = (128/125kHz) * number of bytes
```

The final example (Figure 6 and Figure 7) show a successful data packet.

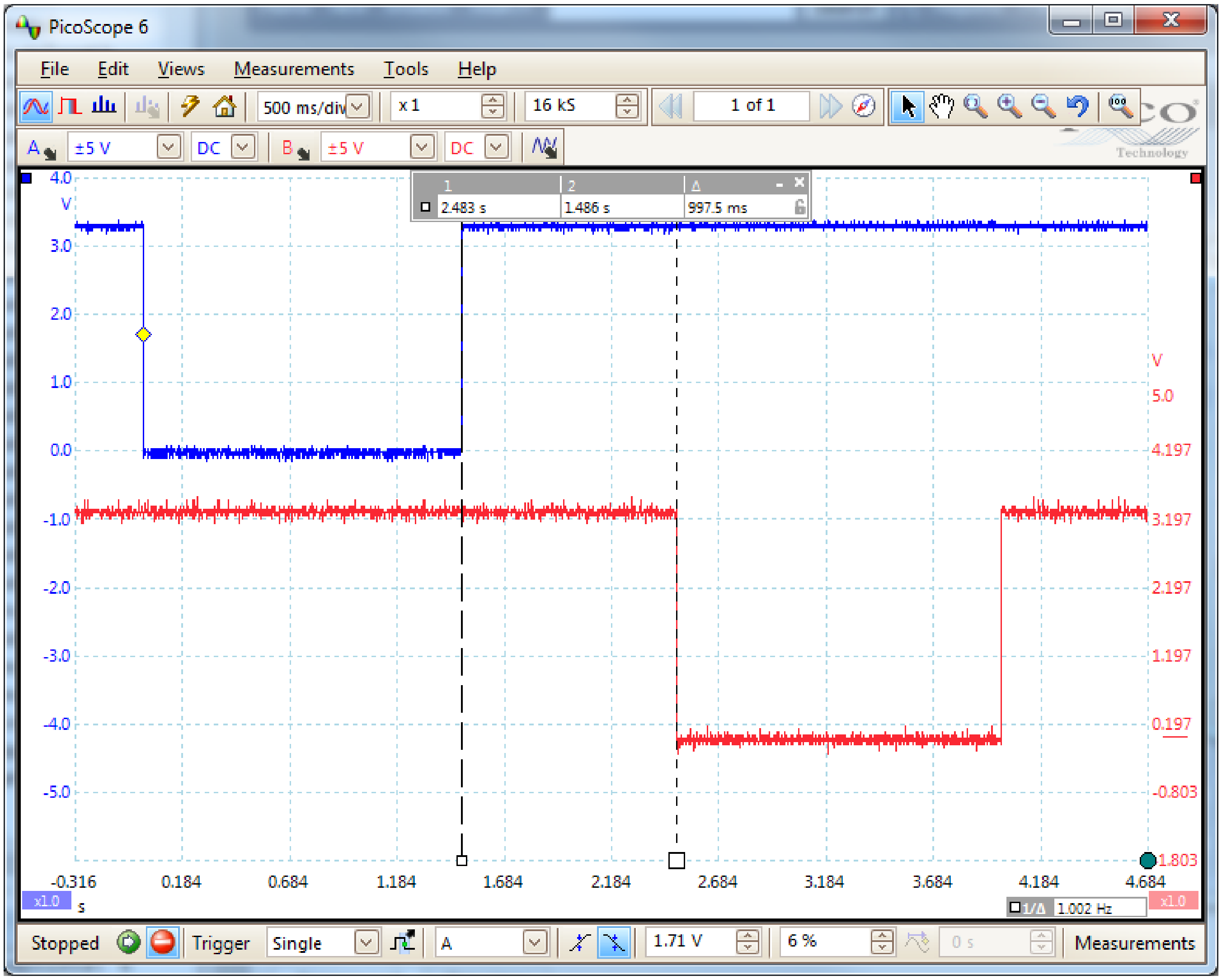
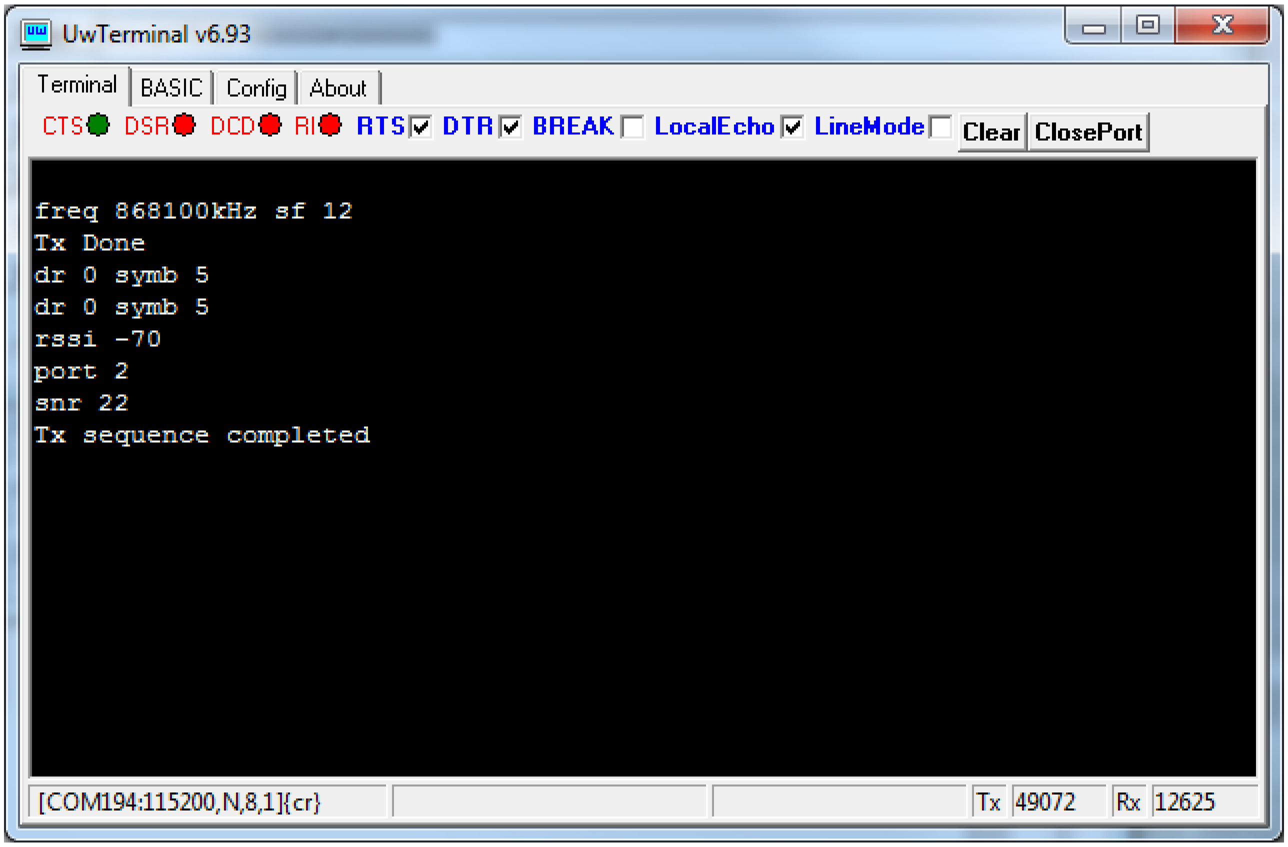


Figure 6 : Successful Data Packet



```
freq 868100kHz sf 12
Tx Done
dr 0 symb 5
dr 0 symb 5
rssi -70
port 2
snr 22
Tx sequence completed
```

[COM194:115200,N,8,1]{cr}

Figure 7 : Successful Data Packet Debug

You can see in this example that a packet was sent on 868.1 MHz. One second after the TxDone signal, the receive window opened and we can see the sync pulse was received as it stayed open to read in all the data. The size of the windows is large because the spreading factor SF12, 2^{12} or 4096, is the largest and so it takes longer to send the data.

Note that there are two receive debug strings, “dr 0 symb 5”, but no “No Sync pulse” debug message. The reason for this is the first receive window is still open and receiving data when the second receive window should open. The point in the code where the debug is printed is before the code realises it can’t open the second receive window, so the debug is still printed. However, the second receive window is not actually opened.

Revision History

Version	Date	Notes	Approver
1.0	27 Oct 2016	Initial Release	Colin Anderson