The MMDVM Specification (20150722)

Introduction

The MMDVM is intended to be an open-source Multi-Mode Digital Voice Modem, which utilises the power of an ARM processor and a simple analogue interface board. It is aimed at the Arduino Due and the Teensy 3.1 with other architectures potentially also supported, the only requirement being access to raw interrupts and an ADC and ADC ports as well as digital signalling ports. For this reason the Raspberry Pi and similar single board computers are not being targeted as access to the raw I/O is mediated by the operating system kernel and does not provide the performance, nor does the hardware provide the necessary direct access to the I/O pins on the ARM processor. However such boards are very useful for hosting the interface of the modem to external networks such as ircDDB in the case of D-Star.

Hardware Interfacing

Currently the only hardware device specified is the Arduino Due. However the requirements are common to both the Due and Teensy. The modem requires one analogue input, one analogue output, and one digital output. A second digital output, as well as a second USB serial port is advantageous.

The analogue input requires that the audio is low pass filtered with a cut-off of around 5 kHz and the output then level shifted so that a zero input signal provides an output of half the rail voltage (3.3V) for the ADC, it is important that the low frequency characteristics of this stage is very good with little or no low frequency roll-off due to any coupling capacitors in the signal path. The analogue output has very similar requirements to the input filtering with the caveat that the output signal from the DAC will be at half the rail voltage for a zero output. Both the input and output can be inverted in software,

The one required output line will be used to signal a transmit condition, the modem is capable of providing an inverted output if needed.

The extra output can be used to drive a LED which is used by the modem to indicate that an incoming signal is being decoded.

Usage	Arduino Due	Teensy 3.1
Analogue Input	?	
Analogue Output	?	
Transmit Output	8	
Decode Output	11	

Software Interfacing

The interface to the modem will be via a USB serial connection. On the Due this will be via the Programming Port, on this hardware the native USB port is used for logging/debugging messages if they are enabled at build time. The speed for both ports is 115200 baud.

The commands are split into generic commands and protocol specific commands. It is possible to build the modem firmware to include or exclude different modes, likewise it is possible to specify which modes are available at run-time also. It is impossible to enable a mode that has not been built into the modem when the firmware was compiled.

The general form of all commands and responses is:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	0x00 – 0xFF	Length. This value includes the Frame Start byte and all of the following data.
2	1	0x00 – 0xFF	Command or Response type.
3…	0..n		Data. The number of data bytes depends on the Command/Reponse byte.

Generic Commands and Responses

ACK

This is transmitted from the modem when a command is received correctly and does not require any specific response with data. It is currently only used when a valid Set Config (see below) command has been received by the modem.

It has a simple format:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	4	Length.
2	1	0x70	ACK
3	1	0x00 – 0xFF	The Command type of the command that caused this ACK to be generated.

NAK

The NAK is very similar to the ACK in format, but it is generated in many more cases. It signals that a command sent to the modem has some sort of problem associated with it. Both the command and the reason are included in the NAK for debugging purposes.

It too has a simple format:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	5	Length.

2	1	0x7F	NAK
3	1	0x00 – 0xFF	The Command type of the command that caused this NAK to be generated.
4	1	0 – 255	The reason for the NAK: 1 – Invalid command value 2 – Wrong mode 3 – Command too long 4 – Data incorrect 5 – Not enough buffer space

Get Version

The command sent from the host to the modem is:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	3	Length.
2	1	0x00	Get Version.

The response from the modem should be:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	N+4	Length.
2	1	0x00	Get Version.
3	1	0x01	Protocol Version.
4…	N		Textual description of the modem firmware in ASCII.

This command would typically be used to determine if there is an MMDVM connected to the host, and to provide a log entry of the version and maybe alter the serial commands depending on the version returned.

Get Status

This command is used to determine the current parameters of the modem. Some of these may be set with the Set Config command below, but many of the values reflect the internal state of the modem.

The command from the host to the modem is:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	3	Length.
2	1	0x01	Get Status.

The response from the modem should be:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	21	Length.
2	1	0x01	Get Status.
3	1	0x00 – 0x0F	Enabled modes: 0x01 – D-Star 0x02 – DMR 0x04 – System Fusion 0x08 – P25
4	1	0 – 4	Modem State: 0 – Idle 1 – D-Star 2 – DMR 3 – System Fusion 4 – P25
5	1	0x00 – 0x01	Internal Flags: 0x01 – TX On
6	1	0 – 255	D-Star Buffer Size. The number of D-Star data frames that can be sent to the modem, a D-Star header requires four of these buffers.
7	1	0 – 255	DMR Buffer Size for Slot 1. The number of DMR data frames that can be sent to the modem for slot 1.
8	1	0 – 255	DMR Buffer Size for Slot 2. The number of DMR data frames that can be sent to the modem for slot 2.
9	1	0 – 255	System Fusion Buffer Size. The number of System Fusion data frames that can be sent to the modem.
10	1	0 – 255	P25 Buffer Size. The number of P25 data frames that can be sent to the modem.
11..20	9	0x00	Spare.

When a mode is not built into the modem, or has not been enabled, then the buffer size value will be zero.

Set Config

This command is used to inform the modem about parameters relevant to its operation.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start.
1	1	10	Length.
2	1	0x02	Set Config.
3	1	0x00 – 0x07	Inversion Flags: 0x01 – Invert RX audio 0x02 – Invert TX audio 0x04 – Invert transmit output
4	1	0x00 – 0x0F	Mode Enable:

			0x01 – D-Star 0x02 – DMR 0x04 – System Fusion 0x08 – P25
5	1	0 – 100	TX Delay in milliseconds.
6	1	5 – 60	Idle Timeout in seconds.
7	1	0-255	RX Input Level adjust
8	1	0-255	TX Output Level adjust
9	1	0-15	DMR Color Code

The Idle Timeout is the amount of time from the last transmission in a particular mode before the modem goes back to Idle mode, and is listening in all enabled modes. This ensures that the modem doesn't flip from one mode to another too quickly and also to cut down on potential abuse.

If the command is accepted then the modem will reply with an ACK (see above) in response.

D-Star Specific Commands and Responses

Transmit/Receive a D-Star Header

This frame is used bi-directionally between the modem and the host when either transmitting or receiving a D-Star Header. If a header to be transmitted is malformed then a NAK (see above) will be returned, however no ACK will be returned if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	44	Length.
2	1	0x10	D-Star Header.
3	41		D-Star Header to be transmitted.

The D-Star Header is in same format as transmitted on-air. The data starts with the three one-byte flag fields and ends with the CCITT checksum.

Transmit/Receive D-Star Data

This frame is used bi-directionally between the modem and the host when either transmitting or receiving D-Star Data. If data to be transmitted is malformed then a NAK (see above) will be returned, however no ACK will be returned if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	15	Length.
2	1	0x11	D-Star Data.
3	12		D-Star Data to be transmitted.

The D-Star Data is in same format as transmitted on-air. The data starts with the AMBE data and ends with the three slow-data/sync bytes.

D-Star Transmission Lost

This frame is used between the modem and the host when a received D-Star transmission disappears/ends without receiving a valid end-of-transmission sequence.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	3	Length.
2	1	0x12	D-Star Lost.

Transmit/Receive D-Star End-Of-Transmission (EOT)

This frame is used bi-directionally between the modem and the host when either transmitting or receiving D-Star Data. If data to be transmitted is malformed then a NAK (see above) will be returned, however no ACK will be returned if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	3	Length.
2	1	0x13	D-Star EOT.

DMR Specific Commands and Responses

Transmit/Receive DMR Data

This frame is used bi-directionally between the modem and the host when either transmitting or receiving DMR Data. If data to be transmitted is malformed then a NAK (see above) will be returned, however no ACK will be returned if the data is correct.

Unlike the other modes, the format of the data sent to the modem is slightly different to that returned by the modem.

The format of the data returned from the modem is:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	37	Length.
2	1	0x18	DMR Data.
3	1	0x00 – 0xFF	DMR Control information: 0x80 – Slot number (unset = 1, set = 2) 0x40 – Data sync pattern detected 0x20 – Voice sync pattern detected
4	33		DMR Data to be transmitted.

The format of the data sent to the modem is:

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	37	Length.

2	1	0x18	DMR Data.
3	1	0x00 – 0xFF	DMR Control information: 0x80 – Slot number (unset = 1, set = 2)
4	33		DMR Data to be transmitted.

Set End of Transmission

In DMR mode the modem is not too clever. It can determine when data has started due to receiving a valid sync, but it cannot determine if the incoming transmission has ended. The host software has the intelligence to decode terminating data and/or determine if the transmission has been lost. When this occurs, the host software sends this frame to the modem to tell it to stop transmitting data to the host, and to only start again once a valid sync has been received.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	4	Length.
2	1	0x19	Set End of Transmission
4	1	0x00 – 0x80	DMR Control Information: 0x80 – Slot number (unset = 1, set = 2)

Set CACH Short LC Data

The Short LC Data is sent with the CACH repeatedly until changes, and indicates what information is being transmitted in each slot. It needs to be changed every time there is a change of data transmitted. The data is already encoded in variable-length BPTC by the time it is sent to the modem. If the data is malformed then a NAK will be returned from the modem, however an ACK will not be sent if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	12	Length.
2	1	0x1A	Set CACH Short LC Data.
4	9		Short LC data.

The Short LC Data is 68-bits in length, so the final four bits of the last byte are not used.

Set Idle Message

This frame is sent to the modem for transmission when the transmitter is active and there is no traffic in a particular slot. If the data is malformed then a NAK will be returned from the modem, however an ACK will not be sent if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	36	Length.
2	1	0x1B	Set idle message
3	33		Idle DMR Data to be transmitted.

DMR Transmit Start

This frame is used between the host and modem to indicate that the modem is to be put into DMR mode and to start transmitting. If data to be transmitted is malformed then a NAK will be returned, however no ACK will be returned if the data is correct.

Byte Number	Length (Bytes)	Value	Description
0	1	0xE0	Frame Start
1	1	3	Length.
2	1	0x1C	DMR Transmit Start.