Monostatic Radar Application Programming Interface (API) Specification

Version 1.2.2

TDSR UWB Radios

TDSR Headquarters
810 Tight Bark Hollow Road
Petersburg, TN 37144 USA
www.tdsr-uwb.com
Tel: +1 256.990.4217

320-0298H
December 2024

Copyright

Trademarks

Rights

Table of Contents

1 Introduction

The P400, P410, P440, and P452 Monostatic Radar Modules--P400 MRM, P410 MRM, P440 MRM, and P452 MRM (P452 MRM shown in **Figure 1**)--are single-board Ultra Wideband (UWB) radio components intended to be integrated into users' electronic devices for enabling high precision distance measurement to non-cooperating targets in high clutter environments. Since the P400, P410, P440, and P452 interfaces are functionally equivalent, this manual refers to the devices interchangeably as an "MRM." Any platform-related differences are specifically identified.

This manual specifies the programming interface between the user's Host processor and the MRM. This document provides a reference of the message structures and bit patterns in an Ethernet UDP/IP programming interface. A separate application note entitled *Using the USB and Serial Interfaces* describes the extended header bytes and protocol required to support both the USB and 3.3V TTL Serial UART interfaces.

Fig. 1: P452 MRM with attached BroadSpecs showing 3 Antenna Configurations

We recommend the software developer become familiar with the API through use of the MRM Reconfiguration and Evaluation Tool (MRM RET) application delivered with the TDSR Radar Kit or Lab. This MS Windows PC application provides a graphical representation of the interface data structures and allows the user to quickly become familiar with host behaviors.

The *MRM Quick Start Guide* provides instructions for getting up and running quickly with MRM RET. The user should reference and build upon the sample applications delivered with the Radar Kit or Lab.

Usage Notes

This section provides a short overview of key facts relative to MRM behavior and interfaces. Much of this information is covered in the *MRM Quick Start Guide*. Critical points for interfacing via Ethernet are repeated here for convenience.

1. Upon power-up, the MRM boots with default configuration parameters previously stored in its FLASH memory. The Host, by setting or querying these configuration parameters, also provides the IP address and port which the MRM will respond with data.
2. Upon successful power-up, the edge-mounted amber Power LED indicates the board is on. The neighboring green LED is off until the MRM has booted and is running. Once running, the green LED will turn on solid. Afterwards, the green LED will toggle with each scan measurement indicating activity.

The user connects to the MRM either through an Ethernet or USB interface. The process is different for the two cases.

Items 3-4 describe the USB connection process.

3. Connect the MRM to the Host using a USB 2.0 A to Micro-B cable (supplied in the TDSR Radar Kit).
4. As described in the *MRM Quick Start Guide*, the user can access the MRM using MRM RET. If MRM RET is not used, then review the document entitled *Using the USB and Serial Interfaces*.

Items 5-9 describe the process using Ethernet (this is only applicable to P400 and P440 MRMs).

5. Connect the MRM to the Host PC using either a crossover Ethernet cable or through an Ethernet switch (some laptops have auto-sensing).
6. As covered in the *MRM Quick Start Guide*, the user should configure his Host PC’s TCP/IPv4 properties to a static IP address such as 192.168.1.1 with Subnet mask 255.255.255.0. This address should not conflict with the attached MRM (typically assigned the IP address 192.168.1.100). The Windows Firewall must be disabled, at least for the MRM addresses of interest.
7. Determine the IP address of the MRM connected to the Host. This number is written on a label attached to the Ethernet connector on the MRM. The default UWB Node IDs will correlate with the default IP addresses. For instance, the MRM delivered in the Kit/Lab will have UWB Node ID 100. This MRM will have a default IP address of 192.168.1.100. The MRM Node ID can be changed through this API.
8. If connecting with the MRM through MRM RET, enter the IP address of the MRM in the field entitled “Network IP Address” and click on the Connect button. If connecting with MRM RET, the user should “ping” the MRM’s Ethernet address using a command window (or terminal).
9. The user’s code should create a UDP socket targeting port 21210 on the MRM. The MRM will respond to the port that sent the message.

Miscellaneous items:

10. Data transferred to/from the MRM is big-endian (network byte order). Code developed on Intel processors must swap bytes (see example code). The Host Service mimics this behavior.
11. The MRM requires two antennas. One is used for transmission, the other for reception. The Host can control which port (A or B) is used for transmission and which is used for reception. Single-antenna operation is not currently supported.

12. The MRM provides a time-based scan of the reflectivity of the surrounding environment. A Windows Service can be enabled to optionally process these scans. This service provides three filters: a band-pass filter, a motion filter, and a detection list threshold filter. This API describes both the direct and MRM Service interfaces.
13. The MRM RET Host application and MRM Service is currently only available for Windows PCs. This API describes the Ethernet/UDP packet structure allowing any Ethernet-capable processor to gather and process UWB radar scans.

Sample host interfacing software in both C and MATLAB is provided on the delivery flash drive and on the TDSR website to help users begin developing their own UWB-enabled applications.

.

2 The MRM Interface

This is a high-level description of the data passed between a Host processor and the MRM.

MRM modules will power-up in an idle mode, waiting for a command from the Host.

Figure 2 provides a high-level overview of the essential MRM architecture. A Host PC running an application interfaces to the MRM to *configure* radar scan options, *control* the number and interval between scans, and (optionally) *configure* the *filter* in the MRM Service.

After reception of a *control* message, the MRM will begin streaming *raw scans* to the Host. If the MRM Service is installed and the application connects to the MRM Service, these scans will be *motion filtered* and converted to a *detection list* consisting of pulse reflection time and reflection amplitude measurements.

The detection time is a measure of the two-way reflection in picoseconds (ps). As RF travels at approximately 0.3 millimeters per picosecond the distance to target(s)(in mm) can be calculated by simply dividing by 2 and multiplying by 0.3.

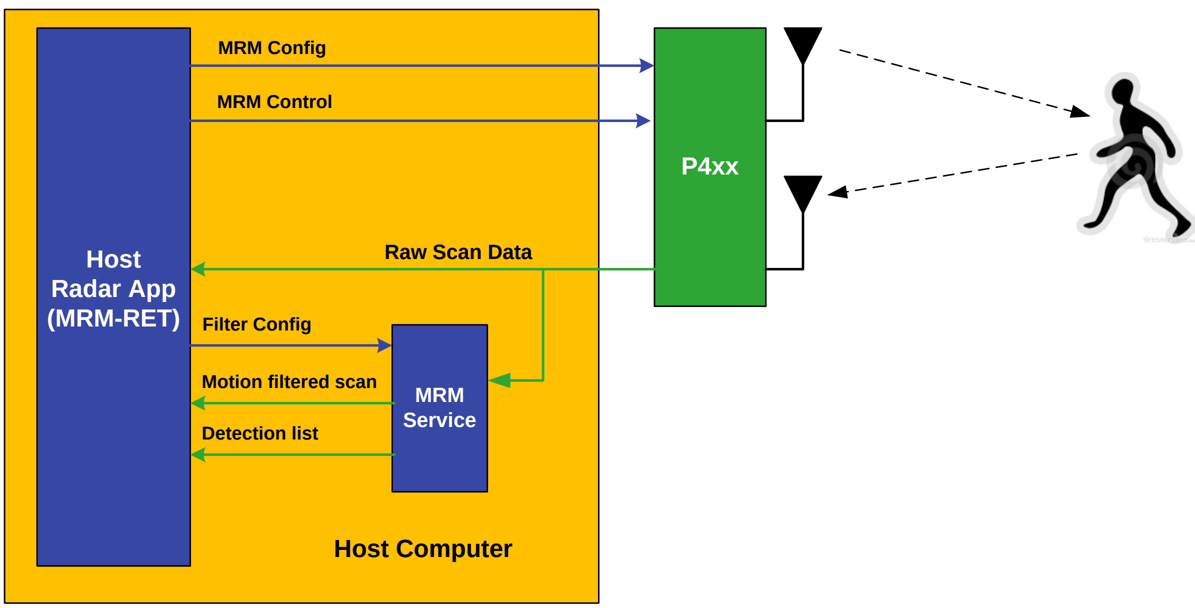

Fig. 2: MRM Host/Module message flow block diagram. A Host Radar Application (such as MRM RET) can connect directly to the MRM for raw scans or connect to the MRM Service for processed (radar filtered) scans.

The REQUEST and INFO messages between MRM and Host are described in the next subsection.

3 MRM API Messages

3.1 MRM_SET_CONFIG_REQUEST (0x1001)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SET_CONFIG_CONFIRM (Radio)

Purpose: This message configures the basic parameters in the MRM, thereby defining radar operation. Note the scan can (optionally) be broken into up to 4 segments, each with a different pulse integration, to allow increased pulse integration (increased SNR) on later (farther) scan points.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_CONFIG_REQUEST (0x1001)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages. The Host application can put any number into this field and it will be echoed in the MRM response. Typically an incrementing number is used. This field becomes important when a single Host is commanding multiple MRMs.
2	Node ID	UINT32	Specifies the node ID. Node IDs 0 and 2^32-1 have special meaning . The user should avoid using these two values.
3	Scan Start (ps)	INT32	Specifies the scan start time, in picoseconds, relative to the pulse transmission time. Valid values are between +/- 499,998 ps.
4	Scan End (ps)	INT32	Specifies the scan end time, in picoseconds, relative to the pulse transmission time.

5	Scan Resolution (bins)	UINT16	Specifies the resolution of the scan data. The standard value for scan resolution is 32 bins. One bin is approximately 1.907 ps thus the time between each scan point is approximately 61 ps. This scan resolution insures that the waveform is sampled faster than the Nyquist rate. It is possible to set scan resolution to any value between 1 and 511. If the user selects a scan resolution which is a 32 multiplied by a power of 2 then the waveform will be undersampled. If the user selects any other value then the rake receiver will be disabled and the time required to generate a scan will be increased by a factor of 12.
6	Base Integration Index	UINT16	Log2 of the number of integrated samples per scan point. Valid values are [6 to 15] implying a range of [64 to 32768].
7	Segment 1 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments. Non-zero overrides the Scan End specification.
8	Segment 2 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 1. Non-zero overrides the Scan End specification.
9	Segment 3 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 2. Non-zero overrides the Scan End specification.
10	Segment 4 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 3. Non-zero overrides the Scan End specification.
11	Segment 1 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 1. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment. For instance if the Base Integration Index is 6 and the Segment Integration Multiple is 5 then data points in this segment will be generated by integrating 2^(5+6) =2048 samples.
12	Segment 2 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 2. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.

13	Segment 3 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 3. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.
14	Segment 4 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 4. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.
15	Antenna Mode	UINT8	Valid values are: 3: Transmit on A, Receive on B
16	Transmit Gain	UINT8	Specifies the pulser transmit power from 0 (lowest) to 63 (highest). The relationship between transmit gain setting and transmit power (power to the base of the antenna) is provided in the relevant data sheets. A 63 setting will produce a signal equal to either the ETSI (EU) or FCC (US) regulatory limit. The exception is that for high power units a 63 will set the transmitter to a level approximately 16dB higher than the limit.
17	Code Channel	UINT8	Specifies the index of the active UWB pseudo-random coded channel. Radars on separate channels will exhibit minimal interference. Channels 0-10 are currently supported.
18	Persist Flag	UINT8	Specifies whether this configuration record will persist through power cycling (write to FLASH memory.) Possible values are 0 (will not persist) or 1 (will persist).

3.2 MRM_SET_CONFIG_CONFIRM (0x1101)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_SET_CONFIG_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in response to a MRM_SET_CONFIG_REQUEST message previously received by the MRM from the Host. Its purpose is to confirm successful operation of the MRM_SET_CONFIG_REQUEST.

Packet Definition:

#	Parameter	Type	Definition

0	MRM_SET_CONFIG_CONFIRM (0x1101)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.3 MRM_GET_CONFIG_REQUEST (0x1002)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_GET_CONFIG_CONFIRM (Radio)

Purpose: This is a request message sent by the Host to MRM for the current radio configuration.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_CONFIG_REQUEST (0x1002)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.4 MRM_GET_CONFIG_CONFIRM (0x1102)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_GET_CONFIG_REQUEST (Host)

Purpose: This message is sent by the MRM in response to a MRM_GET_CONFIG_REQUEST from the Host. It provides the current MRM configuration information.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_CONFIG_CONFIRM (0x1102)	UINT16	Message type

1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Node ID	UINT32	Specifies the node ID.
3	Scan Start (ps)	INT32	Specifies the scan start time, in picoseconds, relative to the pulse transmission time.
4	Scan End (ps)	INT32	Specifies the scan end time, in picoseconds, relative to the pulse transmission time.
5	Scan Resolution (bins)	UINT16	Specifies the resolution of the scan data. The standard value for scan resolution is 32 bins. One bin is approximately 1.907 ps thus the time between each scan point is approximately 61 ps. This scan resolution insures that the waveform is sampled faster than the Nyquist rate. It is possible to set scan resolution to any value between 1 and 511. If the user selects a scan resolution which is a 32 multiplied by a power of 2 then the waveform will be under sampled. If the user selects any other value then the rake receiver will be disabled and the time required to generate a scan will be increased by a factor of 12.
6	Base Integration Index	UINT16	Log2 of the number of integrated samples per scan point. Valid values are [6 to 15] implying a range of [64 to 32768.]
7	Segment 1 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments. Non-zero overrides the Scan End specification.
8	Segment 2 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 1. Non-zero overrides the Scan End specification.
9	Segment 3 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 2. Non-zero overrides the Scan End specification.

10	Segment 4 Num Samples (NOT YET IMPLEMENTED)	UINT16	The number of points in this scan segment. A zero indicates no scan segments after segment 3. Non-zero overrides the Scan End specification.
11	Segment 1 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 1. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment. For instance, if the Base Integration Index is 6 and the Segment Integration Multiple is 5, then data points in this segment will be generated by integrating 2^(5+6) =2048 samples.
12	Segment 2 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 1. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.
13	Segment 3 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 1. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.
14	Segment 4 Integration Multiple (NOT YET IMPLEMENTED)	UINT8	Log2 of dwell multiple for segment 1. Valid values are 1 to 9. This value is combined with the Base Integration Index to determine the total number of samples combined to produce values in this segment.
15	Antenna Mode	UINT8	Valid values are: 3: Transmit on A, Receiver on B
16	Transmit Gain	UINT8	Specifies the pulser transmit power from 0 (lowest) to 63 (highest). The relationship between transmit gain setting and transmit power (power to the base of the antenna) is provided in the relevant data sheets. A 63 setting will produce a signal equal to either the ETSI (EU) or FCC (US) regulatory limit. The exception is that for high power units a 63 will set the transmitter to a level approximately 16dB higher than the limit.
17	Code Channel	UINT8	Specifies the index of the active UWB pseudo-random coded channel. Radars on separate channels will exhibit minimal interference. Channels 0-10 are currently supported.

18	Persist Flag	UINT8	Specifies whether this configuration record will persist through power cycling (write to FLASH memory.) Possible values are 0 (will not persist) or 1 (will persist.)
19	Timestamp	UINT32	Specifies the number of milliseconds elapsed since the P400 boot time.
20	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.5 MRM_CONTROL_REQUEST (0x1003)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_CONTROL_CONFIRM (Radio)

Purpose: This message configures the MRM to one of three operational/timing modes, and sets the automatic timing interval.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_CONTROL_REQUEST (0x1103)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Scan Count	UINT16	0 = Stop 1 = Single Shot 2 to 65534 = number of scans before stop 65535 = run forever (until Scan Count reset to zero) NOTE: when motion filter is enabled the first few scans will not be provided due to the scan depth/history required by the filter.
3	Reserved	UINT16	Reserved
4	Scan Interval Time (μs)	UINT32	Number of microseconds between the start of each radar scan. Specifying 0 or any value less than the actual amount of scan time required to implement a scan results in scanning as fast as possible.

3.6 MRM_CONTROL_CONFIRM (0x1103)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_CONTROL_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in response to a MRM_CONTROL_REQUEST command.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_CONTROL_CONFIRM (0x1103)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.7 MRM_SERVER_CONNECT_REQUEST (0x1004)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SERVER_CONNECT_CONFIRM (Radio)

Purpose: This message connects the User Application to the Host Server, specifying the MRM device under control. The User Application, through the MRM Service, can receive data from more than one MRM device, but configures only one at a time.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SERVER_CONNECT_REQUEST (0x1004)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	MRM IP Address	UINT32	The IP address of the MRM board.
3	MRM IP Port	UINT16	The IP port number of the MRM board.
4	Reserved	UINT16	Reserved

3.8 MRM_SERVER_CONNECT_CONFIRM (0x1104)

API: MRM API
Message type: CONFIRM (MRM)
Corresponding Message type: MRM_SERVER_CONNECT_REQUEST (HOST)

Purpose: This message confirms reception of the MRM_SERVER_CONNECT_REQUEST command from the Host.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SERVER_CONNECT_CONFIRM (0x1104)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Connection Status	UINT32	0 = successful, 1 = general error, 2 = MRM already in use

3.9 MRM_SERVER_DISCONNECT_REQUEST (0x1005)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SERVER_DISCONNECT_CONFIRM (Radio)

Purpose: This message disconnects the User Application from the Server.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SERVER_DISCONNECT_REQUEST (0x1005)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.10 MRM_SERVER_DISCONNECT_CONFIRM (0x1105)

API: MRM API
Message type: CONFIRM (MRM)
Corresponding Message type: MRM_SERVER_DISCONNECT_REQUEST (HOST)

Purpose: This message confirms reception and operation of the MRM_SERVER_DISCONNECT_REQUEST command from the Host.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SERVER_DISCONNECT_CONFIRM (0x1105)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.11 MRM_SET_FILTER_CONFIG_REQUEST (0x1006)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SET_FILTER_CONFIG_CONFIRM (Radio)

Purpose: This message configures the radar filters in the MRM Service.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_FILTER_CONFIG_REQUEST (0x1006)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Filter Mask	UINT16	Specifies the filter operation and reporting. Multiple flags can be set simultaneously to provide multiple levels of processed radar scans. 1=raw 2=bandpass filter 4=motion filter 8=detection list

3	Motion Filter Index	UINT8	0: FIR2, a subtraction of the previous scan from the most recent raw scan. 1: FIR3, a finite impulse response combining the most recent and past 2 scans. 2: FIR4, a finite impulse response combining the most recent and past 3 scans. 3: IIR3, an infinite impulse response combining the latest scan as well as the past 2 output scans. Note: see the MRM RET User Guide for specific difference equations.
4	Reserved	UINT8	Reserved

3.12 MRM_SET_FILTER_CONFIG_CONFIRM (0x1106)

API: MRM API
Message type: CONFIRM (MRM)
Corresponding Message type: MRM_SET_FILTER_CONFIG_REQUEST (HOST)

Purpose: This message confirms reception of the MRM_SET_FILTER_CONFIG_REQUEST command from the Host.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_FILTER_CONFIG_CONNECT_CONFIRM (0x1106)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.13 MRM_GET_FILTER_CONFIG_REQUEST (0x1007)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_GET_FILTER_CONFIG_CONFIRM (Radio)

Purpose: This message requests the MRM Service to respond with its filter configuration.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_FILTER_CONFIG_REQUEST (0x1007)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.14 MRM_GET_FILTER_CONFIG_CONFIRM (0x1107)

API: MRM API
Message type: CONFIRM (MRM)
Corresponding Message type: MRM_GET_FILTER_CONFIG_REQUEST (HOST)

Purpose: This message confirms reception of the MRM_GET_FILTER_CONFIG_REQUEST command from the Host.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_FILTER_CONFIG_CONNECT_CONFIRM (0x1107)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Filter Mask	UINT16	Specifies the filter operation and reporting. Multiple flags can be set simultaneously to provide multiple levels of processed radar scans. 1 = raw 2 = bandpass filter 4 = motion filter 8 = detection list
3	Motion Filter Index	UINT8	0: FIR2, a subtraction of the previous scan from the most recent raw scan. 1: FIR3, a finite impulse response combining the most recent and and past 2 scans. 2: FIR4, a finite impulse response combining the most recent and and past 3 scans. 3: IIR3, an infinite impulse response combining the latest with the past 2 as well as the past 2 output scans. Note: see the MRM RET User Guide for specific difference equations.

4	Reserved	UINT8	Reserved
5	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.15 MRM_GET_STATUSINFO_REQUEST (0xF001)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_GET_STATUSINFO_CONFIRM (Radio)

Purpose: This message prompts the MRM to send the Host a data structure describing the hardware and software version numbers as well as other MRM status information.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_STATUSINFO_REQUEST (0xF001)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.16 MRM_GET_ STATUSINFO_CONFIRM (0xF101)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_GET_STATUSINFO_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in immediate response to a MRM_GET_VERSION_REQUEST command. This response provides a list of the hardware and software version numbers as well as other MRM status information.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_STATUSINFO_CONFIRM (0xF101)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	MRM Version Major	UINT8	MRM embedded major version number

3	MRM Version Minor	UINT8	MRM embedded minor version number
4	MRM Version Build	UINT16	MRM embedded build version number
5	UWB Kernel Major	UINT8	Kernel code major version number
6	UWB Kernel Minor	UINT8	Kernel code minor version number
7	UWB Kernel Build	UINT16	Kernel code build version number
8	FPGA Firmware Version	UINT8	Firmware version number represented in Hexadecimal
9	FPGA Firmware Year	UINT8	Firmware year encoded. Use (year >> 4) * 10 + (year % 16) to get decimal value
10	FPGA Firmware Month	UINT8	Firmware month encoded. Use (month >> 4) * 10 + (month % 16) to get decimal value
11	FPGA Firmware Day	UINT8	Firmware day encoded. Use (day >> 4) * 10 + (day % 16) to get decimal value
12	Serial Number	UINT32	Device serial number represented in Hexadecimal
13	Board Revision	UINT8	PCB revision – a single ASCII character
14	Power-On BIT Test Result	UINT8	Built-in Test Results, non-zero indicates BIT failure.
15	Board Type	UINT8	Specifies the PCB type (or rather the radio type). 1=P400, 2=P410, 3=P412, 4=P440, 5=P452.
16	Transmitter Configuration	UINT8	Pulser type on the radio. 0=FCC, 1=High Power, 2=EU
17	Temperature	INT32	Board temp in 0.25oC (divide this number by 4 to produce floating point oC.).
18	Package Version	CHAR[32]	Human-readable string that identifies the embedded package release version. The actual package version string is typically less than 32 bytes; the rest of this field is zero-filled.
19	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.17 MRM_REBOOT_REQUEST (0xF002)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_REBOOT_CONFIRM (Host)

Purpose: This message causes the MRM to reboot.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_REBOOT_REQUEST (0xF002)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.18 MRM_ REBOOT_CONFIRM (0xF102)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_REBOOT_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in immediate response to a MRM_REBOOT_REQUEST command before reboot operation.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_REBOOT_CONFIRM (0xF102)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.19 MRM_SET_OPMODE_REQUEST (0xF003)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SET_OPMODE_CONFIRM (Radio)

Purpose: This message can be used to transition the MRM to MRM mode.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_OPMODE_REQUEST (0xF003)	UINT16	Message type
1	Message ID	UINT16	Associates request to confirm packets
2	Operational Mode	UINT32	1: MRM

3.20 MRM_ SET_OPMODE _CONFIRM (0xF103)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_SET_OPMODE_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in response to a MRM_SET_OPMODE_REQUEST command indicating the status of the request.

Packet Definition:

#	**Parameter**	**Type**	**Definition**
0	MRM_OPMODE_CONFIRM (0xF103)	UINT16	Message type
1	Message ID	UINT16	Associates request to confirm packets
2	Operational Mode	UINT32	New Operational Mode
2	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.21 MRM_SCAN_INFO (0xF201)

API: MRM API
Message type: INFO (Radio)
Corresponding Message type: none

Purpose: This message contains scan data sent by the MRM to the Host. This data can be raw or filtered depending on the scan mode included in the structure. The size of the entire scan is defined by the MRM_CONFIG structure. The number of scan points will most likely be larger than a single MRM_SCAN_INFO structure can support. The entire scan is sent using multiple MRM_SCAN_INFO messages, ordered through the scan_index parameter. single UDP packet.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SCAN_INFO (0xF201)	UINT16	Message type
1	MRM INFO Message ID	UINT16	Increments with each INFO message sent from the MRM.
2	Source ID	UINT32	Node ID of the transmitting radio
3	Timestamp	UINT32	Milliseconds from boot to time of data collection.
4	Reserved	UINT32	Reserved
5	Reserved	UINT32	Reserved
6	Reserved	UINT32	Reserved
7	Reserved	UINT32	Reserved
8	Scan Start (ps)	INT32	Start time of scan in integer picoseconds relative to the pulse transmission time.
9	Scan Stop (ps)	INT32	End time of scan in integer picoseconds relative to the pulse transmission time.
10	Scan Step (bins)	INT16	Specifies the resolution of the scan data. Currently only a resolution of 32 bins is supported. One bin is approximately 1.907 ps thus the time between each scan point is approximately 61 ps.
11	Scan Type	UINT8	Type of scan data (1 = RAW, 2 = fast time filtered, 3 = motion filtered)
11	Reserved	UINT8	Reserved

12	Antenna ID	UINT8	Designator of receiving antenna (0=A, 1=B)
13	Operational mode	UINT8	Operational mode the P4xx was in when this scan was generated (1 = MRM).
14	Number of samples in message	UINT16	Defines the number of valid samples following in this message.
15	Number of samples total	UINT32	The number of (32bit) data points in the entire scan.
16	Message index	UINT16	The ordered index of this data in the entire scan.
17	Number of messages total	UINT16	The total number of MRM_SCAN_INFO messages used to provide the entire scan.
18	Scan Data	INT32	Scan values collected by the radio. This is a window of 1-350 valid data points each representing the signal amplitude.

3.22 MRM_DETECTION_LIST_INFO (0x1201)

API: MRM API
Message type: INFO (Radar)
Corresponding Message type: none

Purpose: This message contains scan index and magnitude data of each scan point that passed the Detection List algorithm's threshold. This combined sequence of tuples provides for multiple target time delays (distances) and associated delta-reflectivity (detection strength) at that range gate.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_DETECTION LIST_INFO (0x1201)	UINT16	Message type
1	MRM INFO Message ID	UINT16	Increments with each INFO message sent from the MRM.
2	Number of Detections	UINT16	The number of valid measurement pairs that follow. Varies from 1 to 350 (if zero are found this message will not be sent.)
3	Index[1]	UINT16	The number of the FIRST scan point crossing the Detection List algorithm's threshold.

4	Magnitude[1]	UINT16	The value of the FIRST scan point crossing the Detection List algorithm's threshold.
5	Index[2]	UINT16	The number of the SECOND scan point crossing the Detection List algorithm's threshold.
6	Magnitude[2]	UINT16	The value of the SECOND scan point crossing the Detection List algorithm's threshold.
…	…	…	…
…	Index[numDetections]	UINT16	The number of the FINAL scan point (up to 350) that crossed the Detection List algorithm's threshold.
…	Magnitude[numDetections]	UINT16	The value of the FINAL scan point (up to 350) that crossed the Detection List algorithm's threshold.
…	0	UINT16	First zero of pad.
…	…	...	…
703	0	UINT16	Padded with up to 698 zeros (this message will only be sent if one or more detections are found).

3.23 MRM_SET_SLEEPMODE_REQUEST (0xF005)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_SET_SLEEPMODE_CONFIRM (Radio)

Purpose: This message causes the MRM to transition to a low-power mode until woken by the host. This command structure is also used to "wake" from STANDBY modes (1, 2, and 3) to ACTIVE (mode 0.)

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_SLEEPMODE_REQUEST (0xF005)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to RCM CONFIRM messages.

2	Sleep Mode	UINT32	0: Transition to ACTIVE Mode. Available from IDLE, STANDBY_E, and STANDBY_S modes. 1: Transition to IDLE Mode 2: Transition to STANDBY_E mode; (Wake upon Ethernet command) 3: Transition to STANDBY_S mode; (Wake upon Serial command) 4: Transition to SLEEP_D; (Wake upon Discrete transition. See datasheet for wakeup pin location.)

3.24 MRM_SET_SLEEPMODE_CONFIRM (0xF105)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_SET_SLEEPMODE_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in response to a MRM_SET_SLEEPMODE_REQUEST message from the host. The host should inspect the status to assure the requested mode transition was successful.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_SET_SLEEPMODE_CONFIRM (0xF105)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Status	UINT32	0: Success 1: Failure (unsupported on hardware) 2: Failure (unsupported mode transition)

3.25 MRM_GET_SLEEPMODE_REQUEST (0xF006)

API: MRM API
Message type: REQUEST (Host)
Corresponding Message type: MRM_GET_SLEEPMODE_CONFIRM (Radio)

Purpose: This message causes the MRM to send the current power mode to the host. When in STANDBY_E mode this command is only supported over the Ethernet port. When in STANDBY_S mode this command is only supported over the serial port.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_SLEEPMODE_REQUEST (0xF006)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.

3.26 MRM_GET_SLEEPMODE_CONFIRM (0xF106)

API: MRM API
Message type: CONFIRM (Radio)
Corresponding Message type: MRM_GET_SLEEPMODE_REQUEST (Host)

Purpose: This message is sent by the MRM to the Host in response to a MRM_GET_SLEEPMODE_REQUEST message from the host.

Packet Definition:

#	Parameter	Type	Definition
0	MRM_GET_SLEEPMODE_CONFIRM (0xF106)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM CONFIRM messages.
2	Sleep Mode	UINT32	0: ACTIVE 1: IDLE Mode 2: STANDBY_E mode; (Wake upon Ethernet command) 3:Transition to STANDBY_S mode; (Wake upon Serial command)
3	Status	UINT32	0 = Successful. For error codes see Table 3-1 at the end of this section

3.27 MRM_READY_INFO (0xF202)

API: MRM API
Message type: INFO (Radio)
Corresponding Message type: none

Purpose: This message is sent by the MRM to the Host after the RCM transitions from a sleep mode of SLEEP_D to ACTIVE. Once the Host receives this INFO message, the MRM is in a state where it can receive and process additional API commands.

Packet Definition:

	Parameter	Type	Definition
0	MRM_READY_INFO (0xF202)	UINT16	Message type
1	Message ID	UINT16	A tracking number used to associate Host REQUEST messages to MRM INFO messages.

Table 3-1: CONFIRM Message Status Codes

0	Success	The REQUEST message was processed successfully
1	Generic Failure	Catch-all for uncategorized failures
2	Wrong Op Mode	The REQUEST message cannot be acted upon in the current op mode
3	Unsupported Value	The REQUEST message contained an unsupported value in one or more of its fields
4	Invalid During Sleep	The REQUEST message cannot be acted upon in the current sleep mode
5	Wrong Message Size	The number of bytes in the REQUEST message did not match the expected number of bytes for the message type
6	Not Enabled	The feature used by the REQUEST message is currently disabled
7	Wrong Buffer Size	The specified size of a buffer in the REQUEST message, or the size of the buffer itself, did not match the expected number of bytes for the message type
8	Unrecognized Message Type	The REQUEST Message Type was not recognized
0x80000000	Internal Error Code	An internal error code was generated. This status is or'ed with the internal error code itself and should be used in communication with TDSR.

Appendix A: MRM Low Power Consumption Modes

Overview

It is always useful to minimize power consumption of a system. To that end, TDSR has identified different operating states and has provided a number of API commands that enable operation in a variety of sleep states. In these sleep states, the unit will de-energize various circuits. The deeper the sleep state, the less power the MRM will consume. It will also take a few milliseconds for the MRM to transition to and from these sleep states. In general, the deeper the sleep state, the longer it will take to enter and emerge from the requested state. Finally, once the unit is in a sleep state, it will no longer be possible to generate scans or change radar parameters. Attempting to do so will generate an error message.

Description of Active and Sleep Modes

The power modes are described below.

INITIAL BOOT: When the MRM is initially powered, it will act as radio. It will idle with the RF receive circuitry on and search for incoming RF packets.

ACTIVE: Once a connection is made to the MRM, either through an MRM API command or by connecting with MRM RET, the MRM will cease operation as a radio and convert to radar operation. This is the normal mode of operation. In this state the user can change parameters, observe status, and start a scan.

ACTIVE (Scanning): Once the user starts a radar scan, it will engage all of the transmit and receive circuits. The MRM will experience its maximum power consumption in this state.

IDLE: In this state, the radar baseband logic and receive acquisition is halted. This reduces the dynamic power consumption of the baseband FPGA. Also, the low noise receive amplifiers and transmit amplifiers are disabled. The MRM software can respond to API commands over the Ethernet, USB, or Serial interfaces. The radar can transition from IDLE mode to ACTIVE mode very rapidly.

STANDBY_E: This state offers additional power savings over the IDLE mode by clock-gating the FPGA, thereby eliminating dynamic power consumption, and disabling power to the UWB RF Front End chips. As in the IDLE mode, the MRM software is able to respond to API commands over the Ethernet, USB, or Serial port interfaces.

STANDBY_S: This state is identical to STANDBY_E, except the Ethernet PHY chip is powered down for a slight additional power savings. As a result, when in this mode the MRM software is only able to respond to API commands over the Serial port.

Typical power consumption for each power mode and the time required to transition into and out of the modes are shown on the various P400/P410/P440/P452 Data Sheets.