DRAFT 0.2

MARS-ALE MMI-RADIO Developers Guide

For v3.00 Build 1.0.0.0

POC for this document is:

NNN0WWL/NNN0ASL-14 NC
nnn0wwl@navymars.org
nnn0wwl@marsale.org

MARS-ALE support forum:

http://groups.yahoo.com/group/MARS-ALE/

MMI-RADIO
Developers Guide

OVERVIEW

The MMI-RADIO capability empowers the end user to take charge of their own radio control and the commands sent to their HF SSB transceiver for MARS-ALE operation. MMI-RADIO provides the user access to the Man Machine Interface (MMI) command interpreter to a means to create a custom radio control library of MACRO files. This enables the user to support any make/model of HF SSB radio that supports remote control that have not yet or may never be added to MARS-ALE via the linked at build time Radio Control Library (.LIB) or to experiment with radio features that are not provided by the standard Radio Control support.

The MMI-RADIO concept is similar to the DLL-RADIO concept in that the user develops their own radio control capability to a set of interface rules. However there is no need to know how to program in any particular computer programming language that will support creating a Windows 32 bit Dynamic Link Library (DLL) to the specification detailed for the DLL-RADIO interface. Instead the MMI-RADIO concept makes use of the ASCII MMI commands RDCMD and HEXRDCMD to create MACRO’s in a library of MMI-RADIO class files. MARS-ALE executes this MACRO (.MAC) files at specific points during its execution in support of radio control.

However, the MMI-RADIO interface does not provide the flexibility of the DLL-RADIO interface which itself does not provide the flexibility of the standard Radio Control Library. As such MMI-RADIO cannot support radios that require any type of Polling, ACK/NAK, Keep-alive-pings, RS232 Breaks, on-the-fly calculated command packet checksum requirements, process status from the radio or perform any other two-way radio remote control communications. The MARS-ALE Radio Control Library can support any type of radio remote control communications requirements and the DLL-RADIO interface can support many but not all. Thus depending on the make/model radio in question, the MMI-RADIO interface may not be the suitable interface solution.

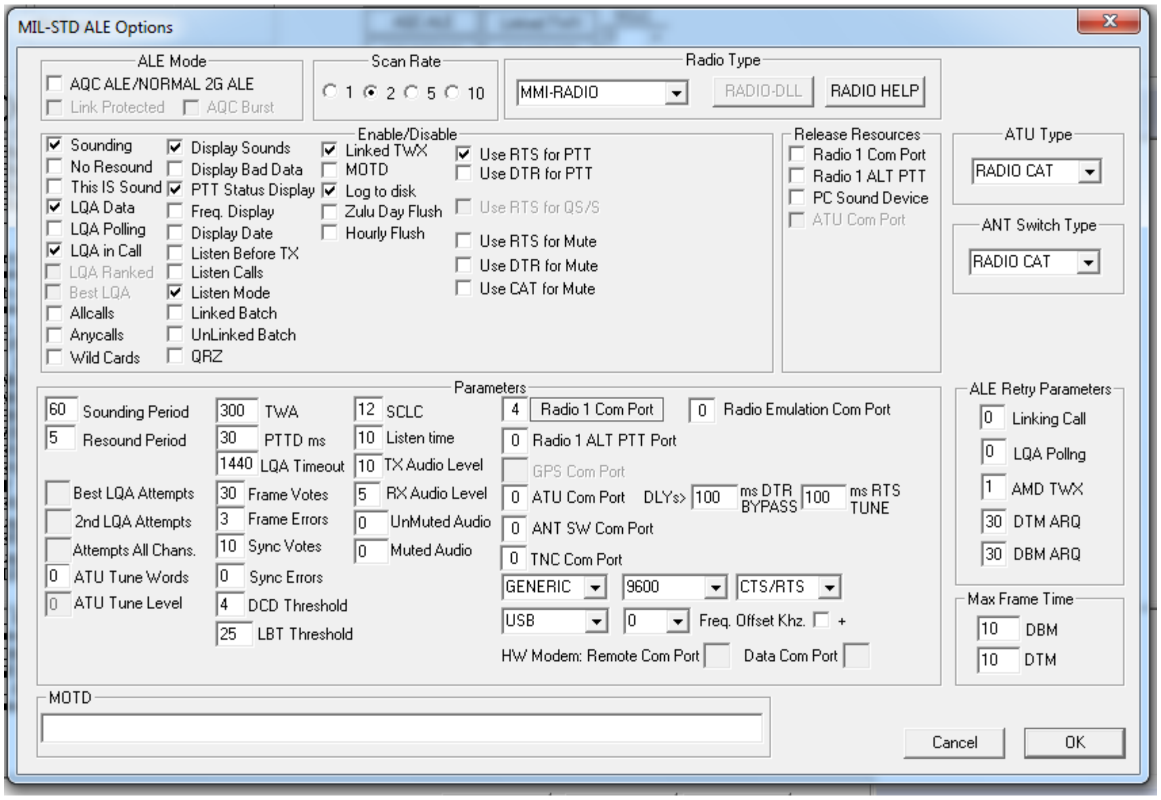
In addition MMI-RADIO has the inherent limitation of using the MMI interpreter which is a slow to process commands compared to compiled code and MMI-RADIO is dependent on file I/O speed of the host computer as we are loading the MACRO’s from a library of MACRO files. However today’s hard drives are fast and Solid State Drive (SSD) technology that is replacing the hard drive is even faster.

The MMI-RADIO specification herein details the specific files that must exist, regardless of their actual need to control a particular radio, as well as the files that must exist to control any radio. In addition the files required based on the user’s configuration selections such as use of CAT PTT or CAT ATU are detailed. The required library of radio specific MMI based MACRO (.MAC) files or collectively the MMI-RADIO Library of files for the particular make/model radio to be controlled must be stored in the \MMI-RADIO\ subdirectory.

To the end user there is little to no difference in the use of MMI-RADIO verses there other radio control interface methods if properly written for the specific radio. On the ALE Options dialog the “Radio Type” selection of MMI-RADIO or MMI-RADIO_HS must be made.

RADIO TYPE MMI-RADIO

To use MMI-RADIO it must be the radio type selection made on the ALE Options dialog as seen below. MMI-RADIO supports both CAT ATU and CAT ANT SW if the radio being implemented via MMI-RADIO is so equipped.


MIL-STD ALE Options

ALE Mode: ☐ AQC ALE/NORMAL 2G ALE; ☐ Link Protected; ☐ AQC Burst

Scan Rate: 1 [✓ 2] 5 10

Radio Type: MMI-RADIO — RADIO-DLL — RADIO HELP

Enable/Disable:
☑ Sounding, ☐ No Resound, ☐ This IS Sound, ☑ LQA Data, ☐ LQA Polling, ☑ LQA in Call, ☐ LQA Ranked, ☐ Best LQA, ☐ Allcalls, ☐ Anycalls, ☐ Wild Cards
☑ Display Sounds, ☐ Display Bad Data, ☑ PTT Status Display, ☐ Freq. Display, ☐ Display Date, ☐ Listen Before TX, ☐ Listen Calls, ☑ Listen Mode, ☐ Linked Batch, ☐ UnLinked Batch, ☐ QRZ
☑ Linked TWX, ☐ MOTD, ☑ Log to disk, ☐ Zulu Day Flush, ☐ Hourly Flush
☑ Use RTS for PTT, ☐ Use DTR for PTT, ☐ Use RTS for QS/S, ☐ Use RTS for Mute, ☐ Use DTR for Mute, ☐ Use CAT for Mute

Release Resources: ☐ Radio 1 Com Port, ☐ Radio 1 ALT PTT, ☐ PC Sound Device, ☐ ATU Com Port

ATU Type: RADIO CAT

ANT Switch Type: RADIO CAT

Parameters:

60	Sounding Period	300	TWA	12	SCLC	4	Radio 1 Com Port
5	Resound Period	30	PTTD ms	10	Listen time	0	Radio 1 ALT PTT Port
		1440	LQA Timeout	10	TX Audio Level		GPS Com Port
	Best LQA Attempts	30	Frame Votes	5	RX Audio Level	0	ATU Com Port
	2nd LQA Attempts	3	Frame Errors	0	UnMuted Audio	0	ANT SW Com Port
	Attempts All Chans.	10	Sync Votes	0	Muted Audio	0	TNC Com Port
0	ATU Tune Words	0	Sync Errors				
0	ATU Tune Level	4	DCD Threshold				
		25	LBT Threshold				

0 Radio Emulation Com Port
DLYs> 100 ms DTR BYPASS; 100 ms RTS TUNE
GENERIC — 9600 — CTS/RTS
USB — 0 — Freq. Offset Khz. ☐ +
HW Modem: Remote Com Port — Data Com Port

ALE Retry Parameters:

0	Linking Call
0	LQA Pollng
1	AMD TWX
30	DTM ARQ
30	DBM ARQ

Max Frame Time:

10	DBM
10	DTM

MOTD

Cancel — OK

However MMI-RADIO does not fully support the Radio Emulation Port (REM) CAT Server mode in that there is not direct on-the-fly frequency control of the radio implemented. In all other aspects the current REM port implementation is supported by the use of MMI-RADIO.

RADIO PORT PARAMETERS

In MARS-ALE regardless of the actual radio type in implemented with MMI-RADIO or MMI-RADIO_HS as the selected “Radio Type”, the default radio port parameters of 9600 N81 will be applied. For other serial radio port control parameters you will need to use the custom port parameters setup by clicking the “Radio 1 Com Port” button.

In addition TCP/IP support for Radio Control is on the “To Do” list where MMI-RADIO_TCP will be provided in the future.

RADIO ADDRESSING

Should a radio make/mode require Radio Addressing, the MMI Commands used in the various MMI-RADIO[command class].MAC file shall provide the radio addressing.

MMI RADIO COMMANDS SYNTAX

Command Syntax:

DELAY nn..nnnn

For use with both RADCMD or HEXRADCMD.

Where nn..nnnn supports 10ms to 1000ms (1 second) of timing delay.

Each line with DELAY nn..nnnn command statement will provide for a delay between commands sent to the radio. Thus if the implemented radio make/model requires 100ms pacing between commands the use of DELAY 100 would be made.

CR1

For use with RADCMD only. Adds a single Carriage Return to the message sent to the radio when used as an argument to RADCMD.

CR2

For use with RADCMD only. Adds a two Carriage Returns to the message sent to the radio when used as an argument to RADCMD.

LF1

For use with RADCMD only. Adds a single Line Feed to the message sent to the radio when used as an argument to RADCMD.

LF2

For use with RADCMD only. Adds two Line Feeds to the message sent to the radio when used as an argument to RADCMD.

CRLF

For use with RADCMD only. Adds a Carriage Return and Line Feed to the message sent to the radio when used as an argument to RADCMD.

RADCMD [Argument1..Argument5]

or

HEXRADCMD [Argument1..Argument30]

RADCMD supports radio make/models using an ASCII command set whereas HEXRADCMD supports any command set.

The commands RADCMD and HEXRADCMD can be used in lower or uppercase.

Arguments entered on the same line as RADCMD or HEXRADCMD must delimited by a SPACE character. When the radio protocol supports stacking commands with no spaces then the data string is looked at as one Argument by RADCMD. Due to the nature of HEXRADCMD each byte must be separated by a SPACE character.

Any data that can be sent using RADCMD can be translated to HEX for sending with HEXRADCMD should any special characters not supported by RADCMD be required to control the radio.

To make use of RADCMD or HEXRADCMD the user MUST know the remote control protocol for their particular make/model radio.

The number of RADCMD or HEXRADCMD statements used in a MACRO file must be kept to a minimum to keep processing time to a minimum. This is especially true when in Multi-Channel ALE operation where the radio is being updated during Scanning, Sounding, making Linking Calls or responding to Linking Calls. What this means is the stacking of commands when possible in the use of a single RADCMD or HEXRADCMD is used to send multiple commands to the radio at one time.

Below are some examples of the use of RADCMD and HEXRADCMD in the creation of MACRO's for radio control purposes.

Here is an example of sending the Frequency of 10Mhz. to a Kenwood radio on VFO A:

```
radcmd FA00010000000;
```

Here is an example of stacking commands in sending both Frequency and Mode as one argument to a Kenwood radio on VFO A as a single argument to RADCMD where the semi-colon termination is used for each command packet sent to the radio:

```
radcmd FA00010000000;MD2;
```

Here is an example of sending the same commands separated by a SPACE character and thus as two arguments to RADCMD:

```
radcmd FA00010000000; MD2;
```

It is obvious that as Kenwood allows stacking commands very efficiently, as such the limit of 5 arguments with RADCMD is rather moot. However, the particular Kenwood radio in question may not be able to process all the commands as fast as they can be sent depending on the sequence of commands and some pacing or time delay between commands may be required. This is true when the radio commands are coded in C++ and compiled. However the use of MMI processing itself adds built in delays. So it's a matter of working with the particular make/model radio to determine what is acceptable.

Here is an example of sending 3 arguments by RADCMD consisting of Frequency to a Ten Tec RX-340 receiver where a single carriage return termination is required:

```
radcmd $1F 10.000000 cr1
```

HEXRADCMD supports sending HEX bytes to radios such as ICOM and old TEN-TEC models, older YAESU models and others. The process of using HEXRADCMD is similar to the use of RADCMD except that each byte of the radio command must be entered as a HEXRADCMD Argument, which requires a space character be inserted between each data byte of the radio command.

HEXRADCMD also supports the sending of command to radios that used a mixture of printable and non-printable characters, a good example being Cubic receivers which require one or more start bytes that are non-printable, followed by the receiver address and command and then a command terminator type.

To set the frequency on a Cubic R3030 receiver that requires three start bytes mixed with other characters would like the following when sent to the radio where the pound (#) sign represents the two characters that follows to be a HEX byte in a serial spy monitor tool:

```
#02#02#0201F01200000#0D
```

Broken down there are three HEX start bytes: #02#02#02

The R3030 set to ASCII address: 01

The ASCII frequency information is: F01200000

The radio command string termination is a carriage return as HEX byte: #0D

The same radio command packet represented completely in HEX would be:

```
#02#02#02#30#31#46#30#31#32#30#30#30#30#30#0D
```

To send this command to the R3030 using HEXRADCMD you would enter:

```
hexradcmd 02 02 02 30 31 46 30 31 32 30 30 30 30 30 0D
```

Here are some examples for ICOM and YAESU models that take into account the actual ICOM radio model HEX address, when using HEXRADCMD you can substitute 00 for the actual radio address you are directly sending these commands to the radio. Thus for the ICOM 746PRO examples below you would change the third byte which is the radio address from 66 to 00 for the address, 00 is the ICOM universal listener address to send the same to command to all devices on the bus if you were using more than one radio at a time.

Set ICOM 746PRO using factory address 66h to USB at 14.109Mhz:

```
hexradcmd FE FE 66 E0 05 00 90 10 14 00 FD FE FE 66 E0 06 01 FD
```

Set ICOM 746PRO using factory address 66h to USB at 14.109Mhz:

```
hexradcmd FE FE 66 E0 05 00 90 10 14 00 FD
```

Set ICOM 746PRO using factory address 66h to USB:

```
hexradcmd FE FE 66 E0 06 01 FD
```

Set ICOM 746PRO using factory address 66h to LSB:

hexradcmd FE FE 66 E0 06 00 FD

Set ICOM 746PRO using factory address 66h to USB-D:

hexradcmd FE FE 66 E0 06 01 FD FE FE 66 E0 1A 06 01 FD

Set ICOM 746PRO using factory address 66h to LSB-D:

hexradcmd FE FE 66 E0 06 00 FD FE FE 66 E0 1A 06 01 FD

Set ICOM 7200 using factory address 76h to USB-D: (NOTE: USB-D/LSB-D selection differs with 7200 vs. all previous ICOM models)

hexradcmd FE FE 76 E0 06 01 FD FE FE 76 E0 1A 04 01 01 FD

Set ICOM 7200 using factory address 76h to LSB-D:

hexradcmd FE FE 76 E0 06 00 FD FE FE 76 E0 1A 04 01 01 FD

Set Yaesu FT847 to USB at 14.109Mhz:

hexradcmd 01 00 00 00 07 01 41 09 00 01

Set Yaesu FT847 to 18.106Mhz:

hexradcmd 01 81 06 00 01

Set Yaesu FT847 to USB:

hexradcmd 01 00 00 00 07

Set Yaesu FT847 to LSB:

hexradcmd 01 00 00 01 07

When creating a MACRO to control the radio of interest, the DataBar when MMI is checked can be used to test the radio response to the MACRO. A Trace display of the command packets sent to the radio while doing so or when actually executing a MACRO file can also be enabled.

MMI CHANGE: SENT RADIO COMMAND

The normal use of RADCMD and HEXRADCMD command via all command line interfaces generates the display of a message to the engineering window such as:

```
[07:37:05][FRQ 02046500][MMI CHANGE: Sent radio command MD2;]
```

However when the radio type is "MMI-RADIO" this automatic confirmation or RADCMD and HEXRADCMD use is disabled.

To make use of this facility in troubleshooting the development of MMI-RADIO class files the user needs to enable the feature using the MMI command **ENABLE TRACEMMIRADIO**. To again disable the use of **DISABLE TRACEMMIRADIO** is required. Then each MMI-RADIO MACRO file processed will display its radio command packets as seen below when manually stepping from channel 1 to channel 2. It is recommended that this trace facility NOT be enabled during scanning operation as it is a heavy load to process.

```
[07:49:25][CHN 02][MMI CHANGE: Sent radio command MD2;]
[07:49:25][CHN 02][MMI CHANGE: Sent radio command FA00003349000;]
[07:49:22][CHN 01][MMI CHANGE: MMI-RADIO TRACE ENABLED]
```

MMI-RADIO MACRO CLASS FILES

Each of the MMI-RADIO class files will reside in the \MMI-RADIO\ sub directory under the directory containing the MARS-ALE .EXE file and will use the master naming convention of:

MMI-RADIO_[command class].DAT

MMI-RADIO FILE CLASSES:

QS/S -

QS/S MACRO class file naming convention:

MMI_RADIO_ENABLE_QSS.MAC
MMI_RADIO_DISABLE_QSS.MAC

If the radio supports setup per the QS/S requirements then the code required to enable and disable QS/S is required else the files can be left blank

PRECONFIUGRATION -

PRECONFIUGRATION MACRO file naming convention:

MMI_RADIO_PRECONFIUGRATION.MAC

This file is executed just after opening the radio serial port at the point where "MMI-RADIO being initialized" is displayed prior to "MMI-RADIO has been initialized" as seen in the Engineering Windows as depicted below.

```
[05:14:02][CHN 00][Radio Setup: MMI-RADIO has been initialized]
[05:14:02][CHN 00][Radio Setup: MMI-RADIO being initialized]
```

It should be noted that all of this happens prior to any radio control being executed which can be seen by either CHN or FREQ as selected being all 0's.

It must be noted that the more commands issued and more less command stacking used, the longer these two files will take to execute. As the MMI_RADIO_PRECONFIGURTION file is executed at program start when the radio port is opened, thus it will delay the start up process of the tool. It is also executed again should radio type selections be changed. Thus too many radio parameters being changed at program start may be noticed as the program GUI comes to life.

REMOTE -

REMOTE MACRO files naming convention:

MMI_RADIO_ENABLE_REMOTE.MAC
MMI_RADIO_DISABLE_REMOTE.MAC

These files support radios that must be commanded to enter and exit radio Remote Control operation, such as is required with many older Yaesu, JRC and other older make/model of radios.

DEINITIALIZE -

DEINITIALIZE MACRO file naming convention:

MMI_RADIO_DEINITIALIZE.MAC

The MMI_RADIO_DEINITIALIZE is executed at program termination, here the speed of processing is of little concern and will not be noticed.

MODE -

MODE MACRO files naming convention: MMI_RADIO_MODE_[xxx..xxxxx].MAC where xxx..xxxxx represents the MARS-ALE Scan Group Mode for RX where the supported modes for normal operation are:

USB and LSB as follows:

MMI_RADIO_MODE_USB.MAC
MMI_RADIO_MODE_LSB.MAC

In support of DATA ports the mode selections in the software are USB-D, LSB-D that correspond to:

MMI_RADIO_MODE_USB_D.MAC
MMI_RADIO_MODE_LSB_D.MAC

For use with commercial hardware TNC/Modem there is support of FSK, FSK-R and RTTY as well, where those files would be:

MMI_RADIO_MODE_FSK.MAC
MMI_RADIO_MODE_FSK_R.MAC
MMI_RADIO_MODE_RTTY.MAC

Most MMI-RADIO libraries will only require MMI_RADIO_MODE_USB.MAC where MMI_RADIO_MODE_USB-D.MAC will be the next most common need.

It is only the MODE MACRO files for the mode used in the Scan Group channels that are required. Where if USB-D or LSB-D is being used then the MIC port version is still required for Voice follow-on using the RED PTT button on the button bar to switch from the DATA port to the MIC port for TX and back again on RX.

CHANNEL -

CHANNEL MACRO files naming convention: MMI_RADIO_CHANNEL_[n..nnn].MAC where n..nnn represents the MARS-ALE Scan Group channel number where the overall range is 1..256 in one Scan Group or the total of all Scan Groups. At a minimum Channel files corresponding to the Scan Group in use must exist for the radio being implemented.

For example, if the Scan Group consists of channels 1 through 10 there would be 10 CHANNEL class files required in the \MMI-RADIO\ sub directory as follows:

MMI_RADIO_CHANNEL_1.MAC
MMI_RADIO_CHANNEL_2.MAC
MMI_RADIO_CHANNEL_3.MAC
MMI_RADIO_CHANNEL_4.MAC
MMI_RADIO_CHANNEL_5.MAC
MMI_RADIO_CHANNEL_6.MAC
MMI_RADIO_CHANNEL_7.MAC
MMI_RADIO_CHANNEL_8.MAC
MMI_RADIO_CHANNEL_9.MAC
MMI_RADIO_CHANNEL_10.MAC

If there are one or more other Scan Groups consisting of addition channels then MMI-RADIO files of the CHANNEL class must exist for those channels as well. However there need not be files for all 256 channels that can be supported unless actually in use.

The CHANNEL class of MMI-RADIO MARCRO could contain one MMI command for setting the discrete Frequency matching the Scan Group channel Frequency. It could also contain radio Frequency and Mode commands matching the Scan Group channel Mode. Then too it could contain the radio Memory channel command to select the corresponding programmed radio memory channel matching the Scan Group channel information.

At scanning start the correct MODE class file will be called once. The MODE class file which must exist to avoid an error message, it can be left empty if mode requirements are always addressed in the channel file.

However for use of DATA modes and the RED PTT support, it is best to not enter the mode into the CHANNEL class file.

During scanning if the mode changes from channel to channel then the correct MODE class file will be made executed. The MODE class file which must exist to avoid an error message, can be left empty if mode requirements are always addressed in the channel file.

In testing MMI-RADIO it has been found that the file I/O does not present any negative effects on the use of the higher 5ch/sec. and 10ch/sec. scan rates as tested with a Kenwood TS-480S which is capable of supporting those scan rates.

However this testing was performed using when two different Windows 7 laptops with conventional hard disks vs. Solid State Drives. Where one laptop was of rather recent manufacture and where the other was originally designed as a hardware compliant host for the MS-Vista OS and later upgraded.

Testing was also performed on a current BIS-6630 Atom N2800 @ 1.86Ghz CPU based embedded computer using an Intel 120GB 530 Series SDD and as would be expected with the faster SDD, there were no scan rate issues.

However the use of MMI-RADIO on older versions of the Windows OS and older hardware may not provide the same results due to both OS latency and file I/O latency times, especially at the 10 ch/sec. scan rate.

In MARS-ALE Single Channel operation the Scan Group channel Frequency and Mode information is sent to the radio at each channel step as separate MMI-RADIO commands. It is best to only enter the Frequency information into the CHANNEL file, however if it desired to also enter the mode, then the MODE class file which must exist to avoid an error message, can be left empty if mode requirements are always addressed in the channel file.

Here is an example of a Channel file in setting the channel frequency for a Kenwood TS-480S using the discrete VFO A frequency approach.

```
# MMI-RADIO MACRO Library for TS-480S
#
# TRI-SERVICE ALE NET CHANNEL 4
#
# Set radio frequency 5158.0kHz
#
radcmd FA00005158000;
```

It must be noted that if the radio Memory Channel approach is taken vs. the discrete frequency and mode approach, that for Voice vs. Data follow on, a like number of Voice and Data memories will be required for selection via used attended MACRO use as the RED PTT approach does not provide a channel number hook, thus discrete mode must also affect the mode change of the memory channel mode if using USB-D.

QS/S CHANNEL -

The QS/S CHANNEL MACRO class supports QS/S operation where the QS/S MACRO class files to ENABLE QS/S must be used when entering ALE scanning. The purpose of the QS/S CHANNEL class files are to update the radio B VFO for TX where the radio type being implemented uses the Split VFO approach to QS/S.

The QS/S CHANNEL MACRO files have a naming convention that adds “QSS” to the CHANNEL naming convention: MMI_RADIO_QSS_CHANNEL_[n..nnn].MAC

For each CHANNEL MACRO file there needs to be a QS/S CHANNEL file regardless or the radio type being implemented being able to make use of them, in which case they would be empty. Thus if there are 10 of the MMI_RADIO_CHANNEL_n.MAC class files there must also be 10 of the MMI_RADIO_QSS_CHANNEL_n.MAC class files in the \MMI-RADIO\ sub directory.

If the Split VFO method for the radio being implemented works, then each QS/S CHANNEL file must contain the remote control commands to setup the radio for the TX frequency and mode if required as well. For some make/model radios a single command equates VFO A with that of VFO B, ICOM for example is such a case where the command is there “EQUATE” sequence.

For other makes of radio there can be a lot of steps required, an example of this situation are Kenwood radio models. Here is an example of the steps required for setting up the B VFO for Kenwood to TX during QS/S scanning.

Below is the first approach that was taken in creating a QS/S MACRO file for the TS-480S. It followed C++ coded Kenwood QS/S flow which is time proven in the update of the B VFO.

```
# MMI-RADIO MACRO Library for TS-480S
#
# TRI-SERVICE ALE NET CHANNEL 4
```

```
#
# Setup B VFO for radio frequency 5158.0kHz
#
# Set radio RX to VFO B
#
radcmd FR1;
#
# Allow radio to settle
#
delay 25
#
# Set radio TX to VFO A
#
radcmd FT0;
#
# Allow radio to settle
#
delay 100
#
# Change TX mode
#
radcmd MD2;
#
# Allow Radio time to settle
#
delay 25
#
# Change TX frequency
#
radcmd FB00005158000;
#
# Allow Radio time to settle
#
delay 25
#
# Set radio RX to VFO A
#
radcmd FR0;
#
# Allow radio to settle
#
delay 25
#
# Set radio TX to VFO B
#
radcmd FT1;
#
# Allow radio to settle
#
delay 25
```

However it was determined in actually testing with the TS-480S that it took 6 seconds to process all the RADCMD statements, which is like 6 times what is permitted in ALE timing.

So instead of seeing the 6 second time difference in the following two lines from the Engineering Window display at the point of a Sounding transmission as seen here:

```
[02:59:30][FRQ 02046500][RADIO: PTT now ACTIVE]
[02:59:24][FRQ 02046500][SOUNDING: SENDING TIS SOUNDING]
```

We need to keep the QS/S MACRO file processing time down to less than 1 second were no difference in time or just a second later displayed as seen below.

```
[03:00:24][FRQ 03349000][RADIO: PTT now ACTIVE]
[03:00:24][FRQ 03349000][SOUNDING: SENDING TIS SOUNDING]
```

In the following version there is no use of the “DELAY” statement for command pacing and heavy using command stacking to minimize the use of the RADCMD statement came in under the time requirement to execute the MACRO.

```
# MMI-RADIO MACRO Library for TS-480S
#
# TRI-SERVICE ALE NET CHANNEL 4
#
# Setup B VFO for radio frequency 5158.0kHz
#
# Set radio RX to VFO B and TX to VFO A and update Mode and Freq.
#
radcmd FR1;FT0;MD2;FB00005158000;
# Set radio RX to VFO A and TX to VFO B
radcmd FR0;FT1;
```

The stacking of four Kenwood commands with no spaces represents one argument to the first RADCMD and then stacking the remaining two commands adds the needed pacing and reduces the entire MACRO file to just to MMI statements.

For a user that can assure themselves that the B VFO will always be in the proper Mode, the MD2; command can be removed and the first statement shortened to:

```
radcmd FR1;FT0;FB00005158000;
```

However doing so only aids in the radio speed of processing commands, it does not affect the MMI processing as we are already down to just one RADCMD statement for this group of radio commands.

If however the TS-480S could not deal with the stacking of these commands as such we would have had to made use of another RADCMD statement.

PTT -

PTT MACRO files naming convention support the Microphone (MIC) port and the DATA (DIG) port that a radio may have where the CAT PTT commands for the ports may differ. At a minimum, if CAT PTT is used, the MIC port PTT files must exist with the correct commands for the radio being implemented.

For the MIC port the PTT class files naming convention is:

MMI_RADIO_MIC_PTT_ON.MAC

MMI_RADIO_MIC_PTT_OFF.MAC

For the DIG port when USB-D or LSB-D are selected as the mode the PTT class files naming convention is:

MMI_RADIO_DIG_PTT_ON.MAC
MMI_RADIO_DIG_PTT_OFF.MAC

These files are only called if CAT PTT is active where the “DIG” are only called if USB-D or LSB-D is selected as the mode for the scan group channel in use.

The RED PTT button on the Data Bar when used, will only use the MIC port for PTT where if USB-D or LSB-D is the mode selected for the given scan group channel, the mode will be switched to USB or LSB respectfully for TX and then set back to USB-D or LSB-D respectfully upon RX.

If there is no specific PTT commands for the given radio being implemented for DIG port use, then the same PTT ON and PTT OFF commands would be entered in both types of PTT files.

MUTING -

The MUTING MACRO files naming convention is:

MMI_RADIO_UNMUTED_AUDIO.MAC
MMI_RADIO_MUTED_AUDIO.MAC

If “Use CAT for MUTE” is not checked on the ALE Options dialog this class of files does not need to exist.

The values for Muted and UnMuted audio on the ALE Options dialog are meaningless when “MMI-RADIO” is the selected radio type as the values will be user set in the MARCO files.

The “MMI_RADIO_UNMUTED_AUDIO.MAC” file will contain the command to set the desired normal RX volume level, however it can be set to any level the user desires.

The “MMI_RADIO_MUTED_AUDIO.MAC” file will contain the command to set the RX volume level to 0, however it can be set to any level the user desires.

ATU -

The ATU MACRO files naming convention:

MMI_RADIO_ATU_ON.MAC
MMI_RADIO_ATU_OFF.MAC
MMI_RADIO_ATU_START.MAC
MMI_RADIO_ATU_STOP.MAC

It CAT ATU is not selected for use these files will never be called and thus do not need to exist.

ANT PORT -

The ANT PORT MACRO files naming convention:

MMI_RADIO_ANT_PORT_1.MAC
MMI_RADIO_ANT_PORT_2.MAC
MMI_RADIO_ANT_PORT_3.MAC
MMI_RADIO_ANT_PORT_4.MAC

Most radios only support selectable 2 ANT ports if any, thus support for 4 should be enough for MMI-RADIO needs.

If ANT PORT is 0 for all scan group channels then none of these files will ever be called and thus do not need to exist.

ATTN -

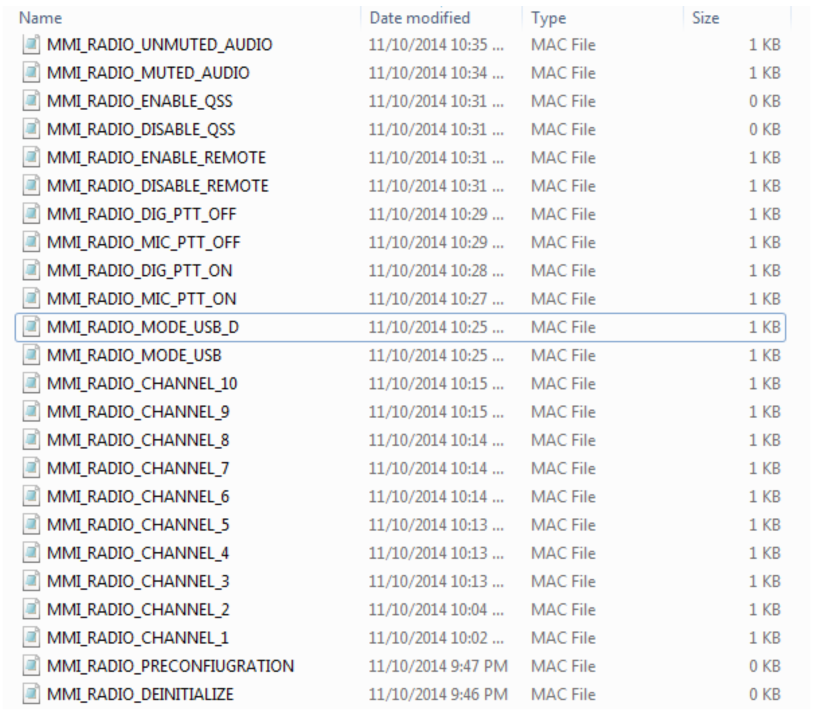
The ATTN MACRO files naming convention:

MMI_RADIO_ATTN_ON.MAC
MMI_RADIO_ATTN_OFF.MAC

If ATTN is not selected for any Scan Group channels the files will never be called and thus do not need to exist.

MINIMUM REQUIRED FILES

The following screen cap depicts an example of the minimum number of MMI-RADIO files required for a 10 channel Scan Group beginning with channel 1. In this example there is no QS/S support and USB and USB-D are depicted as the radio operating modes for Voice/DATA. If just using USB then the highlighted file for USB-D support can be ignored. If QS/S is required, the QS/S files which are empty in this image, need to be configured and the QS/S Channel files need to be added.



Name	Date modified	Type	Size
MMI_RADIO_UNMUTED_AUDIO	11/10/2014 10:35 ...	MAC File	1 KB
MMI_RADIO_MUTED_AUDIO	11/10/2014 10:34 ...	MAC File	1 KB
MMI_RADIO_ENABLE_QSS	11/10/2014 10:31 ...	MAC File	0 KB
MMI_RADIO_DISABLE_QSS	11/10/2014 10:31 ...	MAC File	0 KB
MMI_RADIO_ENABLE_REMOTE	11/10/2014 10:31 ...	MAC File	1 KB
MMI_RADIO_DISABLE_REMOTE	11/10/2014 10:31 ...	MAC File	1 KB
MMI_RADIO_DIG_PTT_OFF	11/10/2014 10:29 ...	MAC File	1 KB
MMI_RADIO_MIC_PTT_OFF	11/10/2014 10:29 ...	MAC File	1 KB
MMI_RADIO_DIG_PTT_ON	11/10/2014 10:28 ...	MAC File	1 KB
MMI_RADIO_MIC_PTT_ON	11/10/2014 10:27 ...	MAC File	1 KB
MMI_RADIO_MODE_USB_D	11/10/2014 10:25 ...	MAC File	1 KB
MMI_RADIO_MODE_USB	11/10/2014 10:25 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_10	11/10/2014 10:15 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_9	11/10/2014 10:15 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_8	11/10/2014 10:14 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_7	11/10/2014 10:14 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_6	11/10/2014 10:14 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_5	11/10/2014 10:13 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_4	11/10/2014 10:13 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_3	11/10/2014 10:13 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_2	11/10/2014 10:04 ...	MAC File	1 KB
MMI_RADIO_CHANNEL_1	11/10/2014 10:02 ...	MAC File	1 KB
MMI_RADIO_PRECONFIUGRATION	11/10/2014 9:47 PM	MAC File	0 KB
MMI_RADIO_DEINITIALIZE	11/10/2014 9:46 PM	MAC File	0 KB

SYSTEM .DAT FILES

The following system .DAT files may also be of interest to the user of MMI-RADIO where the use of RADCMD or HEXRADCMD statement execution can be made independent of the MMI-RADIO class files. The exception being the use of CONFIG.DAT which CANNOT be used as it executes before serial ports are opened.

STARTUP.DAT **-** Executes MMI commands at program start after radio port is opened.

SCANSTART.DAT **-** Executes MMI commands at the start of scanning.

LINKED.DAT **-** Executes MMI commands after an ALE inlink state is established.

UNLINKED.DAT **-** Executes MMI commands when the ALE inlink state is cleared.

SCANSTOP.DAT **-** Executes MMI commands when Scanning is stopped.

SHUTDOWN.DAT **-** Executes MMI commands at the termination of the application.

Any use of the radio control MMI commands in these files will generate status radio command messages being displayed to the Engineering Window.