MOTOROLA

Software Commands

Overview

This section provides definitions for the Man-Machine-Interface (MMI) commands. MMI commands are used to test and configure the EBTS equipment via a service computer. Two command sets are used to accomplish this; the Generation 3 Site Controller (Gen 3 SC) or Integrated Site Controller (iSC) command set and Base Radio (BR) command set.

The Gen 3 SC or iSC command set is described in detail in the Gen 3 SC or iSC Supplement to this manual. Base radio commands are described herein.

The following table lists the chapter topics.

Section	Page	Description
MMI Command Overview	2	Describes the MMI command access levels, and conventions
Legacy Base Radio Commands	4	Defines Legacy Base Radio individual commands used to configure and test the Base Radios
Generation 2 Base Radio Software Command Overview	58	Describes the MMI command access levels and conventions.
QUAD Channel BR Software Commands	85	Describes the MMI and Base Radio software commands for the QUAD Channel BR.

MMI Command Overview

This section describes all MMI commands pertaining to Base Radios. All valid commands are described, along with the syntax, definitions, and examples.

MMI commands are input from a service computer to the system RS-232 serial port (19200 bps, 8 data bits, 1 stop bit, no parity). The RS-232 is accessed from the iSC, or from the front of each BRC in the RF Cabinet.

The test procedure for the Base Radio uses these commands to test and configure the system. Refer to the Base Radio section of this manual for the Base Radio test procedures.

This section covers Base Radio MMI commands only; refer to the iSC Supplement to this manual for a complete description of all MMI commands pertaining to the iSC.

Access Level

The Base Radio commands are available through the use of the field password. This password allows the service technician access to a subset of the MMI command set. This subset is used for field service and does not allow permanent configuration of the Base Radio.

NOTE

The **motorola** password is a default field password that is programmed during manufacturing. The password will be changed by the Operations and Maintenance Center (OMC) as soon as the controller takes a download. The OMC default field password is **Motorola**. The field password is customer defined and can be changed at the OMC. Please check with the OMC operator on duty to obtain your unique customer field password.

Most of the commands are valid only while the Base Radio is in the test mode. The configuration data is temporarily stored in RAM until the Base Radio is taken out of the test mode. Most MMI commands do not allow the configuration data to be permanently stored in EEPROM, although a limited number do. Commands that allow configuration data to be changed are noted.

Conventions

All Base Radio MMI commands are presented in alphabetical order. The command syntax is case sensitive. The syntax for each command is presented as follows:

- plain text shows the actual text to be typed to invoke a command or action
- *italic* text shows where a parameter or value is to be substituted
- text enclosed in brackets [ ] indicates an optional value that may be entered.
- Where items are separated by vertical bars | , the items are the applicable choices that may be entered
- text enclosed in braces { } indicates a corresponding selection or parameter that **must** be entered for the command to execute
- A series of dots ... indicates one or more occurrences of a preceding parameter

The syntax for the BR commands is case sensitive. Each example is shown in the format that should be entered by the operator.

Some commands require the use of parameters. If input parameters are not entered, a response is returned identifying the proper syntax for the command.

A definition describes in detail each command's purpose and function. Where helpful, the definition is followed by an example of the commands response. Typical values have been used whenever possible.

Some commands return varying responses (such as available, not available, unknown, o.k., and alarm). Only one of the possible responses is listed in each example.

Legacy Base Radio Commands

DEKEY

Syntax:

dekey

The **dekey** command stops all RF transmission.

After the command is entered, an indication of a successful transmission stop is returned.

Example:

```
BRC> dekey
XMIT OFF INITIATED
```

GET ALARMS

Syntax:

get alarms

The **get alarms** command returns Base Radio alarm conditions.

Example:

If alarm conditions exist, all active alarms are returned.

```
BRC> get alarms
[brc fru warning]
[external reference failure]
[gps failure]
```

If no alarm conditions exist, a message is returned indicating alarms have not been detected.

```
BRC> get alarms
NO ALARM CONDITIONS DETECTED.
```

GET ALARM_MASK

Syntax:

get alarm_mask

The **get alarm_mask** command returns twelve, 1-byte hexadecimal fields. These bytes represent which alarms are enabled or disabled.

ff indicates that all alarms covered by that byte are enabled.

Example:

```
BRC> get alarm_mask
 ALARM MASK is |ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|ff|
```

GET ALARM_REPORTS

Syntax:

get alarm_reports

The **get alarm_reports** command returns the enabled/disabled status of the alarm reports.

Example:

```
BRC> get alarm_reports
 ALARM REPORTS: TRACE is ENABLED
```

GET BRC_KIT_NO

Syntax:

get brc_kit_no

The **get brc_kit_no** command returns the kit number of the Base Radio Controller (BRC) module.

Example:

```
BRC> get brc_kit_no
BRC KIT NUMBER is TRN7515A
```

GET BRC_REV_NO

Syntax:

get brc_rev_no

The **get brc_rev_no** command returns the hardware revision number of the BRC module.

Example:

```
BRC> get brc_rev_no
BRC REVISION NUMBER is RXX.XX.XX
```

GET BRC_SCRATCH

Syntax:

get brc_scratch

The **get brc_scratch** command reads the allocated EEPROM field reserved for a scratch pad on the BRC module.

Example:

```
BRC> get brc_scratch
BRC SCRATCH is Motorola, Inc.
```

GET CABINET

Syntax:

get cabinet

The **get cabinet** command returns the cabinet in which the current Base Radio resides.

Example:

```
BRC> get cabinet
CABINET is 1
```

GET DEFAULT_TX_POWER

Syntax:

get default_tx_power

The **get default_tx_power** command returns the default transmit operating power level. The value is returned in Watts and dBm.

800 MHz Base Radio Example:

```
BRC> get default_tx_power
DEFAULT TRANSMITTER POWER is 50.00 watts (46.99 dBm)
```

GET ENET_ID

Syntax:

get enet_id

The **get enet_id** command returns the Ethernet address for the current BRC.

Example:

```
BRC> get enet_id
BRC ETHERNET ADDRESS is 08 00 3E C0 02 C8
```

GET EXCITER_SCALING_FACTOR

Syntax:

get exciter_scaling_factor {*port: 0->11*}

The **get exciter_scaling_factor** command returns the scaling factor for a specified Exciter module A/D port.

Example:

```
BRC> get exciter_scaling_factor 1
EXCITER SCALING FACTOR 1 is 1.000000
```

GET EXT_REF

Syntax:

get ext_ref

The **get ext_ref** command returns the current enabled/disabled state of phase locking circuit on the BRC.

Example:

```
BRC> get ext_ref
EXTERNAL REFERENCE is ENABLED
```

GET EX_AD

Syntax:

get ex_ad [*port: 0 -> 11*]

The **get ex_ad** command returns the current hexadecimal value of all A/D ports on the Exciter module with their interpreted voltages.

If the variable for the port number is not entered, the current value of all ports are returned.

Example:

```
BRC> get ex_ad
EXCITER A->D PORT[0] = 0x6c  [14.24v].
EXCITER A->D PORT[1] = 0x0   [0.00v].
EXCITER A->D PORT[2] = 0xa7  [10.21v].
EXCITER A->D PORT[3] = 0xff  [4.98v].
EXCITER A->D PORT[4] = 0x7c  [4.84v].
EXCITER A->D PORT[5] = 0x39  [1.04v].
EXCITER A->D PORT[6] = 0x52  [1.58v].
EXCITER A->D PORT[7] = 0x78  [6.42v].
EXCITER A->D PORT[8] = 0x81  [5.04v].
EXCITER A->D PORT[9] = 0x0   [0.00v].
EXCITER A->D PORT[10] = 0x0  [0.00v].
```

GET EX_KIT_NO

Syntax:

get ex_kit_no

The **get ex_kit_no** command returns the kit number of the Exciter module.

800 MHz Base Radio Example:

```
BRC> get ex_kit_no
EXCITER KIT NUMBER is TLF7000A
```

GET EX_REV_NO

Syntax:

get ex_rev_no

The **get ex_rev_no** command returns the hardware revision number of the Exciter module.

Example:

```
BRC> get ex_rev_no
EXCITER REVISION NUMBER is Rxx.xx.xx
```

GET EX_SCRATCH

Syntax:

get ex_scratch

The **get ex_scratch** command reads the allocated EEPROM field reserved for the scratch pad on the Exciter module.

Example:

```
BRC> get ex_scratch
EXCITER SCRATCH is Motorola, Inc.
```

GET FWD_PWR

Syntax:

get fwd_pwr

The **get fwd_pwr** command returns the current value of forward power. This reading is taken from the built-in power meter of the RF Power Amplifier module. The results are returned in Watts and dBm.

This command should be used only when the transmitter is keyed to obtain accurate results.

800 MHz Base Radio Example:

```
BRC> get fwd_pwr
FORWARD POWER is 66.32 watts [48.22 dbm]
```

GET FWD_WATTMETER_SCALING_FACTOR

Syntax:

get fwd_wattmeter_scaling_factor

The **get fwd_wattmeter_scaling_factor** command returns the linear multiplier used to derive the forward power level from the external wattmeter located in the RFDS, if applicable.

Example:

```
BRC> get fwd_wattmeter_scaling_factor
FORWARD POWER WATTMETER SCALING FACTOR is 52.00
```

GET K_FACTOR

Syntax:

get k_factor

The **get k_factor** command returns the current operational k_factor value.

Example:

```
BRC> get k_factor
K FACTOR is 0.85000000
```

GET MAX_VSWR

Syntax:

get max_vswr

The **get max_vswr** command returns the maximum Voltage Standing Wave Ratio (VSWR) before an alarm is triggered, as measured by the external wattmeter located in the RFDS, if applicable.

Example:

```
BRC>get max_vswr
MAXIMUM VSWR is 4.00:1
```

GET MAX_WATTMETER_VSWR

Syntax:

get max_wattmeter_vswr

The **get max_wattmeter_vswr** command returns the maximum VSWR before an alarm is triggered, as measured by the built-in power meters of the RF Power Amplifier module.

Example:

```
BRC>get max_wattmeter_vswr
MAXIMUM VSWR AT WATTMETER: 4.00:1
```

GET PA_AD

Syntax:

get pa_ad [*port: 0 -> 11*]

The **get pa_ad** command returns the current hexadecimal value of all A/D ports on the Power Amplifier module with their interpreted voltages.

If the variable for the port number is not entered, the current value of all ports are returned.

Example:

```
BRC> get pa_ad
PA A->D PORT[0] = 0x0          [0.00v].
PA A->D PORT[1] = 0x0          [0.00v].
PA A->D PORT[2] = 0x2          [0.04v].
PA A->D PORT[3] = 0x7b         [2.40v].
PA A->D PORT[4] = 0xb          [0.21v].
PA A->D PORT[5] = 0xb          [0.21v].
PA A->D PORT[6] = 0x6          [0.06v].
PA A->D PORT[7] = 0x8          [0.06v].
PA A->D PORT[8] = 0xb          [0.21v].
PA A->D PORT[9] = 0x80         [2.50v].
PA A->D PORT[10] = 0x8         [0.06v].
```

GET PA_COEF

Syntax:

get pa_coef

The **get pa_coef** command returns the Power Amplifier coefficients. These values are determined and programmed during manufacturing.

800 MHz Base Radio Example:

```
BRC> get pa_coef
***AT AND BELOW 858.500 MHz***
PA COEFFICIENT FACTOR A: 0.04900
PA COEFFICIENT FACTOR B: 3.04000
PA COEFFICIENT FACTOR C: 3.66000

***ABOVE 858.500 MHz***
PA COEFFICIENT FACTOR D: 0.00300
PA COEFFICIENT FACTOR E: 3.37000
PA COEFFICIENT FACTOR F: 3.73000
```

GET PA_KIT_NO

Syntax:

get pa_kit_no

The **get pa_kit_no** command returns the kit number of the Power Amplifier module.

800 MHz Base Radio Example:

```
BRC> get pa_kit_no
POWER AMPLIFIER KIT NUMBER is TRN7713A
```

GET PA_REV_NO

Syntax:

get pa_rev_no

The **get pa_rev_no** command returns the hardware revision number of the Power Amplifier module.

Example:

```
BRC> get pa_rev_no
POWER AMPLIFIER REVISION NUMBER is RXX.XX.XX
```

GET PA_SCALING_FACTOR

Syntax:

get pa_scaling_factor *{port: 0->11}*

The **get pa_scaling_factor** command returns the scaling factor for a specified Power Amplifier module A/D port.

Example:

```
BRC> get pa_scaling_factor 1
POWER AMPLIFIER SCALING FACTOR 1 is 1.000000
```

GET PA_SCRATCH

Syntax:

get pa_scratch

The **get pa_scratch** command reads the allocated EEPROM field reserved for the scratch pad on the Power Amplifier module.

Example:

```
BRC> get pa_scratch
POWER AMPLIFIER SCRATCH PAD is Motorola, Inc.
```

GET PCTRL

Syntax:

get pctrl

The **get pctrl** command returns the current enabled/disabled state of the power leveling functionality of the Base Radio.

Example:

```
BRC> get pctrl
POWER CONTROL is ENABLED
```

GET PEND

Syntax:

get pend

The **get pend** command returns the current warp value setting and the internal temperature of the pendulum IC.

Example:

```
BRC> get pend
PENDULUM WARP is 0x94
PENDULUM TEMPERATURE is +33 C
```

GET PEND_LOCK

Syntax:

get pend_lock

The **get pend_lock** command returns the current locked/unlocked status of the pendulum lock bit.

Example:

```
BRC> get pend_lock
PENDULUM is LOCKED
```

GET POSITION

Syntax:

get position

The **get position** command returns the position number of where the current Base Radio is mounted within a selected cabinet. This *does not* represent the cabinet in which the Base Radio resides.

Example:

```
BRC> get position
POSITION is 2
```

GET PS_AD

Syntax:

get ps_ad [*port: 0 -> 11*]

The **get ps_ad** command returns the current hexadecimal value of all A/D ports on the Power Supply module with their interpreted voltages.

If the variable for the port number is not entered, the current value of all ports are returned.

Example:

```
BRC> get ps_ad
PWR SUPPLY A->D PORT[0] = 0xed          [28.28v].
PWR SUPPLY A->D PORT[1] = 0xe3          [14.23v].
PWR SUPPLY A->D PORT[2] = 0xd6          [5.08v].
PWR SUPPLY A->D PORT[3] = 0xea          [4.57v].
PWR SUPPLY A->D PORT[4] = 0x5           [0.10v].
PWR SUPPLY A->D PORT[5] = 0xde          [4.34v].
PWR SUPPLY A->D PORT[6] = 0x92          [2.83v].
PWR SUPPLY A->D PORT[7] = 0xff          [4.98v].
PWR SUPPLY A->D PORT[8] = 0xfe          [4.90v].
PWR SUPPLY A->D PORT[9] = 0xff          [4.98v].
PWR SUPPLY A->D PORT[10] = 0x0          [0.00v].
```

GET REF_PWR

Syntax:

get ref_pwr

The **get ref_pwr** command returns the current value of reflected power. This reading is taken from the built-in power meter of the RF Power Amplifier module. The results are returned in Watts and dBm.

This command should only be used when the transmitter is keyed to obtain accurate results.

Example:

```
BRC> get ref_pwr
REFLECTED POWER is 1.50 watts [31.75 dbm]
```

GET REF_WATTMETER_SCALING_FACTOR

Syntax:

get ref_wattmeter_scaling_factor

The **get ref_wattmeter_scaling_factor** command returns the linear multiplier used to derive the reflected power level from the external wattmeter located in the RFDS, if applicable.

Example:

```
BRC> get ref_wattmeter_scaling_factor
REFLECTED POWER WATTMETER SCALING FACTOR is 52.00
```

GET ROM_VER

Syntax:

get rom_ver

The **get rom_ver** command returns the current software version stored in firmware on the BRC module.

Example:

```
BRC> get rom_ver
BRC ROM VERSION is RXX.XX.XX
```

GET RPTR_STATUS

Syntax:

get rptr_status

The **get rptr_status** command returns the overall status of the repeater.

800 MHz Base Radio Example:

```
BRC> get rptr_status
BRC HOST CODE VERSION is Rxx.xx.xx
BRC FIRMWARE VERSION is Rxx.xx.xx

BRC REVISION is Rxx.xx.xx
EXCITER REVISION is Rxx.xx.xx
POWER AMPLIFIER REVISION is Rxx.xx.xx
RECEIVER 1 REVISION is Rxx.xx.xx
RECEIVER 2 REVISION is Rxx.xx.xx
RECEIVER 3 REVISION is Rxx.xx.xx

RECEIVER 1 is PRESENT
RECEIVER 2 is PRESENT
RECEIVER 3 is PRESENT

PENDULUM WARP is 0x94
PENDULUM TEMPERATURE is +33 C
PENDULUM is LOCKED

RECEIVE FREQUENCY is 815.00000 MHz
TRANSMIT FREQUENCY is 859.00000 MHz
TRANSMIT INTERMEDIATE FREQUENCY is 118.50000 MHz.

WINDOW CLIPPING LEVEL is 5.5 db
WINDOW CLIPPING SATURATION LEVEL is 15 db
WINDOW CLIPPING MODE is ENABLED

SOFTWARE GAIN CONTROL is ENABLED
SOFTWARE GAIN CONTROL DELAY is 246 units (2.050000 msec)
EXTERNAL REFERENCE is ENABLED

PERIODIC TRAINING is ENABLED
PERIODIC TRAINING INTERVAL is 30000 units (5 sec)

POWER CONTROL is ENABLED
POWER CONTROL INTERVAL is 90000 units (15 sec)

POWER WATCHDOG is ENABLED
```

GET RSSI

Syntax:

get rssi {*no. of reports*} {*no. of samples*}

The **get rssi** command allows examination of the received RF signal quality of the Base Radio. A performance report is returned including Bit Error Rate (BER), Received Signal Strength Indication (RSSI), frequency offset, and the sync miss rate.

RSSI data is calculated for the specified number of samples. Each sample is averaged over the specified number of reports specified. A report is generated once every 90 msec.

Example:

BRC> **get rssi 2 100**

Starting RSSI monitor for 2 repetitions averaged each 100 reports.

Line	RSSI1 dBm	RSSI2 dBm	RSSI3 dBm	SGC dB	DIVBER dBm%	SyncMiss %
1	-109.1	-127.0	-127.0	0.0	-109.02.942e+000.000e+00	
2	-108.7	-127.0	-127.0	0.0	-109.02.874e+000.000e+00	

GET RX*(n)*_AD

Syntax:

get rx1_ad [*port: 0 -> 11*]

get rx2_ad [*port: 0 -> 11*]

get rx3_ad [*port: 0 -> 11*]

The **get rx*(n)*_ad** command returns the current hexadecimal value of all A/D ports on the Receiver module with their interpreted voltages.

If the variable for the port number is not entered, the current value of all ports are returned.

Example:

```
BRC> get rx1_ad
RX1 A->D PORT[0] = 0xe0      [9.71v].
RX1 A->D PORT[1] = 0x87      [5.27v].
RX1 A->D PORT[2] = 0xe2      [9.80v].
RX1 A->D PORT[3] = 0xff      [4.98v].
RX1 A->D PORT[4] = 0x7d      [4.88v].
RX1 A->D PORT[5] = 0xe6      [4.49v].
RX1 A->D PORT[6] = 0x57      [1.70v].
RX1 A->D PORT[7] = 0x67      [2.01v].
RX1 A->D PORT[8] = 0x7d      [4.88v].
RX1 A->D PORT[9] = 0xd0      [8.13v].
```

GET RX*(n)*_DELTA

Syntax:

get rx1_delta

get rx2_delta

get rx3_delta

The **get rx(n)_delta** command returns the contents of the RSSI offset value in dBm for a selected receiver. This is a calibrated value that is set during manufacturing.

Example:

```
BRC> get rx1_delta
RECEIVER 1 RECEIVE SIGNAL STRENGTH DELTA is 0.0
```

GET RX*(n)*_KIT_NO

Syntax:

get rx1_kit_no

get rx2_kit_no

get rx3_kit_no

The **get rx*(n)*_kit_no** command returns the kit number of a selected Receiver module.

800 MHz Base Radio Example:

```
BRC> get rx1_kit_no
RECEIVER 1 KIT NUMBER is CRF6010A
```

GET RX*(n)*_REV_NO

Syntax:

get rx1_rev_no

get rx2_rev_no

get rx3_rev_no

The **get rx*(n)*_rev_no** command returns the hardware revision number of the specified Receiver module.

Example:

```
BRC> get rx1_rev_no
RECEIVER 1 REVISION NUMBER is RXX.XX.XX
```

GET RX*(n)*_SCALING_FACTOR

Syntax:

get rx1_scaling_factor {*port: 0 -> 11*}

get rx2_scaling_factor {*port: 0 -> 11*}

get rx3_scaling_factor {*port: 0 -> 11*}

The **get rx*(n)*_scaling_factor** command returns the scaling factor for a specified Receiver module A/D port.

Example:

```
BRC> get rx1_scaling_factor 1
RECEIVER 1 SCALING FACTOR 1 is 2.000000
```

GET RX*(n)*_SCRATCH

Syntax:

get rx1_scratch

get rx2_scratch

get rx3_scratch

The **get rx*(n)*_scratch** command reads the allocated EEPROM field reserved for the scratch pad on the specified Receiver module.

Example:

```
BRC> get rx1_scratch
RECEIVER 1 SCRATCH is Motorola, Inc.
```

GET RX_FREQ

Syntax:

get rx_freq

The **get rx_freq** command returns the programmed receiver frequency for the current Base Radio.

800 MHz Base Radio Example:

```
BRC>get rx_freq
The RX FREQUENCY is: 806.00000 MHz
```

GET RX_FRU_CONFIG

Syntax:

get rx_fru_config

The **get rx_fru_config** displays the current receiver diversity configuration of a Base Radio.

Example:

```
BRC> get rx_fru_config
RECEIVER CONFIGURATION {RX1 RX2 RX3}
```

GET RX_INJ

Syntax:

get rx_inj

The **get rx_inj** command returns the high/low side injection status of the second Local Oscillator (LO) for all receivers.

Example:

```
BRC> get rx_inj
RECEIVER INJECTION is LOW
```

GET RX_MODE

Syntax:

get rx_mode

The **get rx_mode** command returns the enabled/disabled status of the receiver.

Example:

```
BRC>get rx_mode
RECEIVER 1 is ENABLED
RECEIVER 2 is ENABLED
RECEIVER 3 is ENABLED
```

GET RX_QSIGN

Syntax:

get rx_qsign

The **get rx_qsign** command returns the current Q sign status of the receivers.

Example:

```
BRC> get rx_qsign
RECEIVER Q SIGN is NON-INVERTED
```

GET RX_SANITY

Syntax:

get rx_sanity

The **get rx_sanity** command returns the receive Digital Signal Processor (DSP) operational condition as either passed or failed.

Example:

```
BRC> get rx_sanity
RECEIVE DSP SANITY TEST passed
```

GET RX_STATUS

Syntax:

get rx_status

The **get rx_status** command returns status information of the receivers.

Example:

```
BRC> get rx_status
RECEIVER INJECTION is LOW
BER STATUS is LOCKED
RECEIVER Q SIGN is NON-INVERTED
RECEIVER 1 is ENABLED
RECEIVER 2 is ENABLED
RECEIVER 3 is ENABLED
```

GET RX_VERSION

Syntax:

get rx_version

The **get rx_version** command returns the current RX Digital Signal Processor (DSP) software version.

Example:

```
BRC> get rx_version
RECEIVE DSP VERSION is 251.235
```

GET SGC

Syntax:

get sgc

The **get sgc** command returns the enabled/disabled status of the Software Gain Control (SGC) routine.

Example:

```
BRC> get sgc
SOFTWARE GAIN CONTROL is ENABLED
```

GET SGC_ATTEN

Syntax:

get sgc_atten {*no. of repetitions: 1->10,000*}

The **get sgc_atten** command returns the attenuator values as reported from the Digital Signal Processor (DSP) to the screen for the number of repetitions specified.

Example:

```
BRC> get sgc_atten 10
Starting SGC monitor for 10 repetitions
displays hex number of 2-dB attenuation steps
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
      0     0     0     0     0     0
```

GET SGC_DELAY

Syntax:

get sgc_delay

The **get sgc_delay** command returns the current setting of the delay used by the software gain control routine.

Example:

```
BRC> get sgc_delay
SOFTWARE GAIN CONTROL is 246 UNITS (2.050000 msec)
```

GET SYS_GAIN

Syntax:

get sys_gain

The **get sys_gain** command returns the enabled/disabled status of the system gain factor.

Example:

```
BRC> get sys_gain
SYSTEM GAIN is ENABLED
```

GET TRAINING_INTERVAL

Syntax:

get training_interval

The **get training_interval** command returns the number of timer ticks between training operations.

Example:

```
BRC> get training_interval
TRAINING INTERVAL: is 30000 ticks (5 min)
```

GET TXLIN

Syntax:

get txlin [*register: 0x00 -> 0x1a*]

The **get txlin** command returns the corresponding byte of the tranlin register as mapped into memory.

Example:

```
BRC> get txlin
TXLIN[0x00]: 0x56     TXLIN[0x01]: 0x08TXLIN[0x02]: 0x16
TXLIN[0x03]: 0x29     TXLIN[0x04]: 0xF1TXLIN[0x05]: 0x1E
TXLIN[0x06]: 0x2C     TXLIN[0x07]: 0x00TXLIN[0x08]: 0x3A
TXLIN[0x09]: 0xBB     TXLIN[0x0A]: 0x53TXLIN[0x0B]: 0x80
TXLIN[0x0C]: 0xA3     TXLIN[0x0D]: 0x40TXLIN[0x0E]: 0x20
TXLIN[0x0F]: 0x80     TXLIN[0x10]: 0x38TXLIN[0x11]: 0x4D
TXLIN[0x12]: 0x00     TXLIN[0x13]: 0x1FTXLIN[0x14]: 0x7F
TXLIN[0x15]: 0x13     TXLIN[0x16]: 0xFFTXLIN[0x17]: 0x00
```

GET TXLIN_STAT

Syntax:

get txlin_stat

The **get txlin_stat** command returns the tranlin operational status. The unassigned internal registers with dummy data are polled.

Example:

```
BRC> get txlin_stat
Checksum: 1880
Test Register: 0x1e
Clip Detect Bit OFF
Local Osc. Locked
I - Channel Software Offset Bit set.
Q - Channel Software Offset Bit set.
Level Set  : 0xff
Sine Value : 0x0
Cosine Value: 0x7d
```

GET TX_FREQ

Syntax:

get tx_freq

The **get tx_freq** command returns the programmed transmitter frequency for the current Base Radio.

800 MHz Base Radio Example:

```
BRC> get tx_freq
TRANSMIT FREQUENCY is 851.00000MHz
```

GET TX_IF

Syntax:

get tx_if

The **get tx_if** command returns the current programmed transmit IF frequency.

800 MHz Base Radio Example:

```
BRC> get tx_if
TRANSMIT INTERMEDIATE FREQUENCY is 118.50000 MHz
```

GET TX_MODE

Syntax:

get tx_mode

The **get tx_mode** command returns the current transmit mode.

Example:

```
BRC> get tx_mode
TRANSMIT MODE is DC
```

GET TX_SANITY

Syntax:

get tx_sanity

The **get tx_sanity** command returns the Tx Digital Signal Processor (DSP) operational condition as either passed or failed.

Example:

```
BRC> get tx_sanity
TRANSMIT DSP SANITY TEST passed
```

GET TX_VERSION

Syntax:

get tx_version

The **get tx_version** command returns the current TX Digital Signal Processor (DSP) software version.

Example:

```
BRC> get tx_version
TRANSMIT DSP VERSION is 251.237
```

GET VSWR

Syntax:

get vswr

The **get vswr** command calculates the current Voltage Standing Wave Ratio (VSWR), as measured by the built-in power meters of the RF Power Amplifier module. This command should only be used when the transmitter is keyed to obtain accurate results.

Example:

```
BRC> get vswr
VSWR is 1.35:1
```

GET WATTMETER

Syntax:

get wattmeter

The **get wattmeter** command returns the forward and reverse power readings and calculates the VSWR from the external wattmeter which is connected to the antenna port. The output power readings are calibrated and returned in Watts.

This command should only be used when the transmitter is keyed to obtain accurate results.

Example:

```
BRC> get wattmeter
FORWARD POWER AT WATTMETER is 27.42 Watts(44.38 dBm)
REFLECTED POWER AT WATTMETER is 1.20 Watts (30.79 dBm)
WATTMETER VSWR is 1.53
```

GET WINDOW_CLIPPING_PARAMETERS

Syntax:

get window_clipping_parameters

The **get window_clipping_parameters** command returns the current variables in the window clipping algorithm.

Example:

```
BRC> get window_clipping_parameters
WINDOW CLIPPING THRESHOLD is 5.5000000
WINDOW SATURATION THRESHOLD is 15.000000
```

HELP

Syntax:

help

The **help** command returns all commands available for the Base Radio software. The display is dependent on the given access level. This command will return the subset of commands available for field personnel.

Example:

```
BRC> help
                dekey
get             alarms
get/set         alarm_mask
get/set         alarm_reports
get             brc_kit_no
get             brc_rev_no
get/set         brc_scratch
get/set         cabinet
get             default_tx_power
get             enet_id
get/set         exciter_scaling_factor
get             ext_ref
get             ex_ad
get             ex_kit_no
get             ex_rev_no
get/set         ex_scratch
get             fwd_pwr
get/set         fwd_wattmeter_scaling_factor
                help
get/set         k_factor
get/set         max_vswr
get/set         max_wattmeter_vswr
get             pa_ad
get             pa_coef
get             pa_kit_no
get             pa_rev_no
get/set         pa_scaling_factor
get/set         pa_scratch
get/set         pctrl
get             pend
get             pend_lock
get/set         position
```

get/set	ref_wattmeter_scaling_factor
	reset
get	rom_ver
get	rptr_status
get	rssi
get	rx(n)_ad
get/set	rx(n)_delta
get	rx(n)_kit_no
get	rx(n)_rev_no
get/set	rx(n)_scaling_factor
get/set	rx(n)_scratch
get/set	rx_freq
get/set	rx_inj
get/set	rx_mode
get/set	rx_qsign
get	rx_sanity
get	rx_status
get	rx_version
get/set	sgc
get	sgc_atten
get/set	sgc_delay
get/set	sys_gain
set	tone
set/set	training_interval
get/set	txlin
get	txlin_stat
get/set	tx_freq
get/set	tx_if
get/set	tx_mode
set	tx_power
get	tx_sanity
get	tx_version
get	vswr
get	wattmeter
set	window_clipping

RESET

Syntax:

reset

The **reset** command performs a software reset of the Base Radio. All parameters entered from the service computer will be lost.

Example:

```
BRC> reset
Base Radio Controller
Firmware Version Rxx.xx.xx
Copyright © 1998
Motorola, Inc. All rights reserved.

DRAM TEST: passed
SRAM TEST: passed
ENET TEST: passed
```

SET ALARM_MASK

Syntax:

set alarm_mask {*byte: 0->11*} {*data: 0x00->0xff*}

The **set alarm_mask** command enables/disables alarms from being acknowledged by the Base Radio. The input parameters are the byte number and the data (or mask).

Example:

The following example enables all alarms in byte 1.

```
BRC> set alarm_mask 1ff
set ALARM MASK 1 to 0xFF in RAM
```

SET ALARM_REPORTS

Syntax:

set alarm_reports {on | off}

The **set alarm_reports** command enables/disables asynchronous alarm reporting. Alarms are not reported to the local terminal if they occur when the alarm reports are disabled.

Example:

```
BRC>set alarm_reports on
set ALARM REPORTS TRACE to ENABLED in RAM
```

SET BRC_SCRATCH

Syntax:

set brc_scratch [*scratch text; 40 char limit*]

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set brc_scratch** command writes to the allocated EEPROM field reserved for the scratch pad of the Base Radio Controller module. This space is overwritten whenever the set brc_scratch command is issued. A maximum of 40 characters may be entered into the scratch pad.

Example:

```
BRC> set brc_scratch abcdef
set BRC SCRATCH to abcdef in RAM and EEPROM
```

SET CABINET

Syntax:

set cabinet {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set cabinet** command sets the cabinet number of the Base Radio.

Example:

```
BRC> set cabinet 1
set CABINET to 1 in RAM and EEPROM
```

SET EXCITER_SCALING_FACTOR

Syntax:

set exciter_scaling_factor {*port: 0->11*} {*scaling factor*}

The **set exciter_scaling_factor** command changes the multiplier on the corresponding Exciter module A/D port. These values should not be trained, they are calibrated during manufacturing.

Example:

```
BRC> set exciter_scaling_factor 1 1
set EXCITER SCALING FACTOR 1 to 1 in RAM
```

SET EX_SCRATCH

Syntax:

set ex_scratch [*scratch text; 40 char limit*]

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set ex_scratch** command writes to the allocated EEPROM field reserved for the scratch pad of the Exciter module. This space is overwritten whenever the set ex_scratch command is issued. A maximum of 40 characters may be entered into the scratch pad.

Example:

```
BRC> set ex_scratch xyz123
set EXCITER SCRATCH to xyz123 in RAM and EEPROM
```

SET FWD_WATTMETER_SCALING_FACTOR

Syntax:

set fwd_wattmeter_scaling_factor {*1.0 -> 1000.0*}

The **set fwd_wattmeter_scaling_factor** command changes the linear multiplier used to derive the forward power level from the external wattmeter located in the RFDS, if applicable.

Example:

```
BRC> set fwd_wattmeter_scaling_factor 52.00
set FORWARD POWER WATTMETER SCALING FACTOR to 52.00 in RAM
```

SET K_FACTOR

Syntax:

set k_factor {*-.99< k_factor < .99*}

The **set k_factor** command alters the TX Digital Signal Processor (DSP) k-factor. The k-factor changes average power.

Example:

```
BRC> set k_factor 0.85
set K FACTOR to 0.85000000 in RAM
```

SET MAX_VSWR

Syntax:

set max_vswr {*1.1 -> 4.0*}

The **set max_vswr** command sets the maximum Voltage Standing Wave Ratio (VSWR) for the internal Base Radio power monitor. The power is reduced if this value is reached.

Example:

```
BRC> set max_vswr 4
set MAX VSWR to 4 in RAM
```

SET MAX_WATTMETER_VSWR

Syntax:

set max_wattmeter_vswr {*1.1 -> 4.0*}

The **set max_wattmeter_vswr** command sets the maximum Voltage Standing Wave Ratio (VSWR) for the external wattmeter located in the RFDS, if applicable. The power is rolled back if this value is reached.

Example:

```
BRC>set max_wattmeter_vswr
set MAX WATTMETER VSWR to 4 in RAM
```

SET PA_SCALING_FACTOR

Syntax:

set pa_scaling_factor {*port: 0->11*} {*scaling factor*}

The **set pa_scaling_factor** command changes the multiplier on the corresponding Power Amplifier module A/D port. These values should not be changed; they are calibrated during manufacturing.

Example:

```
BRC> set pa_scaling_factor 1 1
set POWER AMPLIFIER SCALING FACTOR 1 to 1.000000 in RAM
```

SET PA_SCRATCH

Syntax:

set pa_scratch [*scratch text; 40 char limit*]

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set pa_scratch** command writes to the allocated EEPROM field reserved for the scratch pad of the Power Amplifier module. This space is overwritten whenever the set pa_scratch command is issued. A maximum of 40 characters may be entered into the scratch pad.

Example:

```
BRC> set pa_scratch xyz123
set PA SCRATCH to xyz123 in RAM and EEPROM
```

SET PCTRL

Syntax:

set pctrl {on | off}

The **set pctrl** command enables/disables the power leveling functionality of the Base Radio. The output indicates and verifies the changes.

Example:

```
BRC> set pctrl on
set POWER CONTROL to ENABLED in RAM
```

SET POSITION

Syntax:

set position {1 | 2 | 3 | 4 | 5 | 6}

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set position** command programs the position number of where the current Base Radio is mounted within a selected cabinet. This *does not* represent the cabinet in which the Base Radio resides.

Example:

```
BRC> set position 2
set POSITION to 2 in RAM and EEPROM
```

SET REF_WATTMETER_SCALING_FACTOR

Syntax:

set ref_wattmeter_scaling_factor {*1.0 -> 1000.0*}

The **set ref_wattmeter_scaling_factor** command changes the linear multiplier used to derive the reflected power level from the external wattmeter located in the RFDS, if applicable.

Example:

```
BRC> set ref_wattmeter_scaling_factor 52
set REFLECTED POWER WATTMETER SCALING FACTOR to 52.00 in
RAM
```

SET RX*(n)*_DELTA

Syntax:

set rx1_delta {*>-100.0 -> +100.0 dBm*}

set rx2_delta {*>-100.0 -> +100.0 dBm*}

set rx3_delta {*>-100.0 -> +100.0 dBm*}

The **set rx*(n)*_delta** command defines the contents of the RSSI offset value for a selected receiver.

Example:

```
BRC> set rx1_delta 0.98
set RECEIVER 1 RECEIVE SIGNAL STRENGTH DELTA to 0.98 in RAM
```

SET RX*(n)*_SCALING_FACTOR

Syntax:

set rx1_scaling_factor {*port: 0-> 11*} {*scaling factor*}

set rx2_scaling_factor {*port: 0 -> 11*} {*scaling factor*}

set rx3_scaling_factor {*port: 0 -> 11*} {*scaling factor*}

The **set rx*(n)*_sclaing_factor** command changes the value of the multiplier on the specified A/D port for a selected receiver. These values should not be changed, they are calibrated during manufacturing.

Example:

```
BRC> set rx1_scaling_factor 1 2
set RECEIVER 1 SCALING FACTOR 1 to 2 in RAM
```

SET RX*(n)*_SCRATCH

Syntax:

set rx(n)_scratch [*scratch text; 40 char limit*]

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

The **set rx*(n)*_scratch** command writes to the allocated EEPROM field reserved for the scratch pad of a selected Receiver module. This space is overwritten whenever the rx(n)_scratch command is issued. A maximum of 40 characters may be entered into the scratch pad.

Example:

```
BRC> set rx1_scratch abc899
set RECEIVER 1 SCRATCH to abc899 in RAM and EEPROM
```

SET RX_FREQ (800 MHz Base Radio)

Syntax:

set rx_freq {*806.000 - 821.000*}

The **set rx_freq** command programs the receiver frequency in the 800 MHz band. The receive frequency for each receiver within a selected Base Radio are programmed at the same time with this command.

The programmed receiver frequency must be in the range of 806.000 MHz to 821.000 MHz in 6.25 kHz increments.

Example:

```
BRC>set rx_freq 806.00000
set RECEIVE FREQUENCY to 806.0000 MHz in RAM
```

SET RX_FRU_CONFIG

NOTE

This command permanently stores the data in EEPROM and is not lost when you exit test mode.

Syntax:

set rx_fru_config {1 | |12 | 123}

The **set rx_fru_config** command sets which receivers should be present in a Base Radio for the intended receive diversity. It is stored in the BRC EEPROM.

Example:

```
BRC>set rx_fru_config 123
RECEIVER CONFIGURATION {RX1 RX2 RX3}
```

SET RX_INJ

Syntax:

set rx_inj {high | low}

The **set rx_inj** command sets the current second Local Oscillator (LO) injection setting to achieve high/low side injection.

Example:

```
BRC> set rx_inj low
set RECEIVER INJECTION to LOW in RAM
```

SET RX_MODE

Syntax:

set rx_mode {1 | 2 | 3 | 12 | 13 | 23 | 123}

The **set rx_mode** command enables/disables any of the individual receivers of the current Base Radio. If a receiver is disabled using this command, it is not used in calculations for BER, RSSI, etc.

Example:

```
BRC>set rx_mode 12
set RECEIVER 1 to ENABLED in RAM
set RECEIVER 2 to ENABLED in RAM
set RECEIVER 3 to DISABLED in RAM
```

SET RX_QSIGN

Syntax:

set rx_qsign {inverted | non-inverted}

The **set rx_qsign** command sets the Rx q_sign to inverted or non-inverted.

Example:

```
BRC> set rx_qsign non-inverted
set RECEIVER Q SIGN to NON-INVERTED in RAM
```

SET SGC

Syntax:

set sgc {on | off}

The **set sgc** command enables/disables the Software Gain Control (SGC).

Example:

```
BRC> set sgc on
set SOFTWARE GAIN CONTROL to ENABLED in RAM
```

SET SGC_DELAY

Syntax:

set sgc_delay {*0 - 1000*}

The **set sgc_delay** command sets the delay used by the software gain control routine.

Example:

```
BRC> set sgc_delay 246
set SOFTWARE GAIN CONTROL DELAY to 246 in RAM
```

SET SYS_GAIN

Syntax:

set sys_gain {on | off}

The **set sys_gain** command enables/disables the system gain factor from being used.

Example:

```
BRC> set sys_gain on
set SOFTWARE GAIN to ENABLED in RAM
```

SET TONE

Syntax:

set tone {*-18000 Hz -> 18000 Hz*}

ATTENTION

This command keys the transmitter. Verify transmission occurs on licensed frequencies or into a dummy load.

The **set tone** command initializes a continuous single tone transmission. The only way to discontinue this feature is via the **dekey** command.

Example:

```
BRC> set tone 1000
set TONE to 1000 in RAM
```

SET TRAINING_INTERVAL

Syntax:

set training_interval {*no. of ticks*}

The **set training_interval** command sets the period between tranlin training cycles.

Example:

```
BRC> set training_interval 30000
set TRAINING INTERVAL to 30000 in RAM
```

SET TXLIN

Syntax:

set txlin {*register: 0x00 -> 0x1a*} {*hex byte: 0x00 -> 0xff*}

The **set txlin** command writes one specific hexadecimal byte to the specified tranlin register and update the codeplug shadow registers.

Example:

```
BRC> set txlin 1 08
set TXLIN 1 to 0x08 in RAM
```

SET TX_FREQ (800 MHz Base Radio)

Syntax:

set tx_freq {*851.000 - 866.000*}

The **set tx_freq** command programs the transmit frequency in the 800 MHz band. When this command is entered, the transmitter frequency is programmed into the Base Radio Controller.

The transmit frequency is specified in 6.25 kHz increments. The programmed transmitter frequency must be in the range of 851.00000 MHz to 866.00000 MHz in 6.25 kHz increments.

Example:

```
BRC>set tx_freq 851.00000
The TRANSMIT FREQUENCY to 851.00000MHz in RAM
```

SET TX_IF

Syntax:

set tx_if {*frequency in MHz*}

The **set tx_if** command sets the transmitter IF frequency.

800 MHz Base Radio Example:

```
BRC> set tx_if 118.35
set TRANSMIT INTERMEDIATE FREQUENCY to 118.5000 MHz in RAM
```

SET TX_MODE

Syntax:

set txmode {outbound | dc | inbound | 6tone}

The **set tx_mode** command sets the transmit mode.

Example:

```
BRC> set tx_mode outbound
set TRANSMIT MODE to OUTBOUND in RAM
```

SET TX_POWER

Syntax:

set tx_power {*value in Watts*}

ATTENTION

This command keys the transmitter. verify transmission occurs on licensed frequencies or into a dummy load.

The **set tx_power** command keys the transmitter to a specified power without altering any programmed parameters. In test mode, the current default transmit mode setting (default_tx_mode) indicates the mode of the transmitter.

The range of allowable settings is dependent upon the Power Amplifier used (70W, 60W, or 40W). A message is returned indicating transmitter activity.

Example:

```
BRC> set tx_power 40

     WORKING...
     TRANSMITTER KEYED: 40.12 watts

BRC>
```

SET WINDOW_CLIPPING

Syntax:

set window_clipping {on | off}

The **set window_clipping** command enables/disables the window clipping algorithm.

Example:

```
BRC> set window_clipping on
set WINDOW CLIPPING to ENABLED in RAM
```

VER

Syntax:

ver

The **ver** command returns the current version of the BR software

Example:

```
BRC>ver
BRC SOFTWARE VERSION is Rxx.xx.xx
```

Generation 2 Base Radio Software Command Overview

Overview

This section provides definitions for Man-Machine-Interface (MMI) commands. MMI commands test and configure the EBTS equipment via a service computer. Two command sets accomplish these test and configuration tasks: The Generation 3 Site Controller (Gen3 SC) or integrated Site Controller (iSC) command set and Base Radio (BR) command set.

The Gen3 SC/iSC Supplement to this manual describes the controller command set in detail. This chapter section describes Generation 2 (Gen2) Base Radio commands.

The following table lists this section's topics.

Section	Page	Description
Generation 2 BR MMI Commands	59	Describes MMI commands, including access levels and conventions
Generation 2 Base Radio Commands	62	Defines the Base Radio commands used to configure and test Base Radios
Generation 2 Base Radio Application Examples	81	Demonstrates command use with sample settings.

Generation 2 BR MMI Commands

This section describes Test Application MMI commands for Generation 2 Base Radios. Command descriptions include syntax, definitions, and examples.

Call processing offers separate MMI commands that follow a different format. For information on the Call Processing MMI commands, refer to the System Release GR Binder.

An operator at a service computer inputs MMI commands to the system's RS-232 serial port. The transfer protocol is: 19,200 bps, 8-N-1. (That is, eight data bits, no parity, 1 stop bit.) Either the Gen3 SC/iSC or the front of the Enhanced Base Radio Controller (EBRC) in the RF Cabinet can access the RS-232 serial port.

The Base Radio test procedure uses MMI commands to test and configure the system. For Base Radio test procedures, refer to the Base Radio section of this manual.

NOTE

This section only covers Test Application MMI commands for Generation 2 Base Radios. For a complete description of MMI commands pertaining to the Gen3/iSC, refer to this manual's controller Supplement. For Single Carrier BR operation, refer to the Single Carrier BR MMI command section.

Access Level

The Test Application requires the user to log in, using a defined user ID and password. The login permits appropriate access to the MMI command set. The Test Application provides the following login IDs.

iDEN Generation 2 Login id's	Password	Configuration
field	motorola	factory default
field	Motorola	OMC default
field	<customer specified>	OMC customer set

NOTE

The 'Out of Box' default factory set field password is deleted and is replaced by the customer defined field password contained within the OMC. This occurs as soon as the controller module receives its initial OMC download.

NOTE

The OMC field password is customer configurable. Please contact the Operations and Maintenance Center (OMC) operator on duty to obtain your customer unique field password.

Example:

```
>login -ufield
password:<login password>

field>
```

Conventions

Generation 2 Base Radio MMI commands appear in alphabetical order. The command syntax is case sensitive. The syntax for each command appears as follows:

- Plain/**bold** text shows what to type to invoke a command or action.
- *Italic* text shows where to substitute a parameter or value.
- Text in braces { } indicates a selection or parameter that the operator **must** enter before the command can execute.
- A series of dots ... indicates one or more occurrences of a preceding parameter.

Generation 2 BR command syntax is case sensitive. Each example appears in the format that the operator should enter.

A definition details each command's purpose and function. Where helpful, an example of the command's response follows the definition. Whenever possible, this section uses typical values.

Some commands return varying responses (such as "available," "not available," "unknown," "o.k.," and "alarm"). Each example only lists one of the possible responses. Examples do not go through every possible scenario.

Generation 2 Base Radio Commands

Command Detail

The commands in bold text appear in long form, followed in parenthesis by the short form.

ALARMS

Syntax:

STRINGX -object = {fault_hndlr}

The ALARMS command returns the active alarms.

Example:

```
field> alarms -ofault_hndlr

ACTIVE_FAULT_ID=RX_LO1_LOCK

field>
```

CABINET_ID (CI)

Syntax:

STRINGX -object = {platform}

STRINGX -cabinet = {any positive integer}

The CI command controls the cabinet identification number.

Examples:

```
field> ci -oplatform
Cabinet ID=0

field> ci -oplatform -c1

field> ci -oplatform
Cabinet ID=1
```

DATA_PATTERN_MODE (DPM)

Syntax:

STRINGX -object = {txch1}

STRINGX -mode = {none, iden}

The DPM command controls the mode setting for the platform object's transmit path data pattern.

Examples:

```
field> dpm -otxch1

Mode=none

field> dpm -otxch1 -miden

field> dpm -otxch1

Mode=iden
```

DIVERSITY

Syntax:

STRINGX -**o***bject* = {rx_all}

STRINGX -**d***iversity* = {right justified binary bit map of the receive branch configuration, the right most bit field corresponds to receive branch 1}

NOTE: This setting does not change the platform in anyway. It is simple used as a configuration parameter.

Examples:

field> diversity -orx_all -d0 #Turn all branches off

field> diversity -orx_all -d1 #enable branch 1; 2 & 3 are off

field> diversity -orx_all -d10 #enable branch 2; 1 & 3 are off

field> diversity -orx_all -d11 #enable branch 1 & 2; 3 is off

field> diversity -orx_all -d100 #enable branch 3; 1 & 2 are off

field> diversity -orx_all -d101 #enable branch 3 & 1; 2 is off

field> diversity -orx_all -d110 #enable branch 3 & 2; 1 is off

field> diversity -orx_all -d111 # enable 1, 2 and 3

ENABLE

Syntax:

STRINGX -object = {txch1, rxch1}

for -object = rxch1

STRINGX -diversity = {br{1->3}}

STRINGX -state = {on, off, dumb}

The ENABLE command controls the enabled state of txch and rxch objects.

The Diversity option only applies to Rx

Examples:

```
field> enable -otxch1 -son

field> enable -otxch1
State=On

field> enable -orxch1 -son

field> enable -orxch1

br1=on
br2=on
br3=on

field> enable -orxch1 -dbr2 -soff

field> enable -orxch1

br1=on
br2=off
br3=on
```

EXTERNAL_SYNCHRONIZATION (ES)

Syntax:

STRINGX -object = {rx_all, tx_all}

for -object = rx_all

STRINGX -type = {trig_def, global, ext_trigger, pps, in_ptt}

for -object = tx_all

STRINGX -type = {trig_def, out_pp3s, out_frame, out_slot, out_ptt, out_mframe, out_fs_bs}

The ES command controls the object's external synchronization method.

Examples:

```
field> es -orx_all

Type=trig_def

field> es -orx_all -text_trigger

field> es -orx_all

Type=ext_trigger
```

FIRMWARE_VERSION (FV)

Syntax:

STRINGX -object = {control, rxch1, txch1, platform, PS}

The FV command returns the object's firmware version. Specifying the object 'platform' causes the software to return additional information.

Example:

```
field> fv -oplatform

Test Application Version =D01.01.07

Controller Rev No    = R07.00.00
Controller Kit No    = CLN 7428A
Controller Firmware  = 7.10.04
Power Supply Rev No  = R06.00.00
Power Supply Kit No  = CPN6080B
Core Software Ver    = 7.10.04
Platform ID          =
Tx1 Exciter Rev No   = R01.04.01
Tx1 Exciter Kit No   = TLF7000H
Tx1 Exciter Firmware = 01.00.02
Tx1 PA Rev No        = R40.01.00
Tx1 PA Kit No        = CLF1772B
Tx1 PA Firmware      =
Rx1 Rev No           = R01.03.00
Rx1 Kit No           = CRF6010D
Rx1 Firmware         = 01.00.00

Core Major Version=7
Core Minor Version=1

field>
```

FREQ

Syntax:

STRINGX -object = {txch1, rxch1}

FLOATX -frequency = {use status -orx_all or status -otx_all for applicable limits}

The FREQ command controls synthesizer frequency settings for the rxch and txch objects.

Examples:

```
field> freq -otxch1 -f860
field> freq -otxch1
freq=860.000000
field> freq -orxch1 -f806
field> freq -orxch1
freq=806.000000
field>
```

FRU_CONFIGURATION (FC)

Syntax:

STRINGX -object = {platform}

The FC command returns the fru configuration.

Example:

```
field> fc -oplatform

type=BRC
slot=Slot 04
revision number=R07.00.00
kit number=CLN7428AA
flags=Available,Supported, Known

type=Power Supply
slot=Slot 00
revision number=R06.00.00
kit number=CPN6080B
flags=Available,Supported, Known

type=Receiver
slot=Slot 03
revision number=R01.03.00
kit number=CRF6010D
flags=Available,Supported, Known
.
.
.
type=Exciter
slot=Slot 01
revision number=R01.00.00
kit number=CTF6321A
flags=Available,Supported, Known

type=Power Amplifier
slot=Slot 02
revision number=R45.01.00
kit number=CLF1772B
flags=Available,Supported, Known
```

HELP

Syntax:

STRINGX -command = {any string}

The HELP command displays help text for the test application. Help topics for the entered command string appear in alphabetical order. HELP operates similarly to a grep, where the string is part or all of the command.

Example:

```
field> help -cpower
power: This command controls the platform object power settings

    arguments:
        -o(object)=txch1 (for write)
        -o(object)=tx,all, ex (for read)
       -p(power)= 5.0-40.0 Watts with 40 Watt PA, 5.0-70.0 with 70 Watt
```

KIT_NUMBER (KN)

Syntax:

STRINGX -object = {ex1, pa1, rxch1, ps, control}

The KN command returns the object's kit number setting.

Example:

```
field> kn -opa1

kit_number=CLFCLF1772B

field>
```

LOGIN

Syntax:

STRINGX -userid = {dev, factory, field}

The LOGIN command allows the user to log in as a valid user. Note that each login requires a password and that the word 'login' be part of the command string.

Example:

```
> login -ufield
password: <field login password>

field>
```

iDEN Generation 2 Login id's	Password	Configuration
field	motorola	factory default
field	Motorola	OMC default
field	<customer specified>	OMC customer set

NOTE

The 'Out of Box' default factory set field password is deleted and is replaced by the customer defined field password contained within the OMC. This occurs as soon as the controller module receives its initial OMC download.

NOTE

The OMC field password is customer configurable. Please contact the Operations and Maintenance Center (OMC) operator on duty to obtain your customer unique field password.

LOGOUT

Syntax:

none

The LOGOUT command logs out the current user and displays the login prompt. No reset occurs.

Example:

```
field> logout

field>
```

PEER_PERFORMANCE_CONFIG (PPC)

Syntax:

STRINGX -object = {rxch1}

STRINGX -mode = {chn, path}

for -mode = path

INTX -path = {1,2,3, all}

for -mode = chn

INTX -slot = {1->6}

The PPC command controls the platform peer performance configuration. A -mode argument setting of “chn” provides the channel performance for all enabled branches. A -mode argument setting of “path” provides the individual receive path performance.

Examples:

```
field> enable -orxch1 -son

field> ppc -orxch1 -mpath -pall

field> ppc -orxch1

Mode=path
Path=all

field> ppc -orxch1 -mchn -s1

field> ppc -orxch1

Mode=chn
Slot=1
```

PEER_PERFORMANCE_REPORT (PPR)

Syntax:

STRINGX -object = {rxch1}, rx_all}

INTX -averages = {2- 65535 max}

INTX -reports = {1- 65535 max}

The PPR command returns the peer performance information. The operator must complete appropriate peer_performance_config settings before generating reports.

Example:

```
field> ppr -orxch1 -a10 -r1

SGC Atten.(dBm)=0.000000
Freq. Offset=-0.000031
Sync. Attempts=1.000000
Sync. Successes=0.000000
BER%=50.290699
RX Path1 RSSI=-129.727005
RX Path2 RSSI=-130.369965
RX Path3 RSSI=-132.486801
Chn sig. strength=-68.348839
Chn intf. strength=-68.348839
```

PEER_TEST_MODE (PTM)

Syntax:

STRINGX -object = {tx_all, rx_all}

for -object = rx_all

-mode = uplk_framed

for -object = tx_all

-mode = {dnlk_framed, tones, stop, suspend}

The PTM command controls the platform peer (DSP) test mode setting.

Examples:

```
field> ptm -otx_all

Mode=not_in_test

field> ptm -otx_all -mdnlk_framed

field> ptm -otx_all

Mode=dnlk_framed
```

POSITION_ID (PI)

Syntax:

STRINGX -object = {platform}

STRINGX -position = {any positive integer}

The PI command controls the platform position number.

Examples:

```
field> pi -oplatform

Position=0

field> pi -oplatform -p5

field> pi -oplatform

Position=5

field>
```

POWER

ATTENTION

The POWER command keys the transmitter. Make sure that transmission only occurs on licensed frequencies, or into an RF dummy load.

Syntax:

STRINGX -object = {tx_all, txch1}

FLOATX -power = {use status -otx_all for applicable limits}

The POWER command controls the platform object's transmit output power. A -power argument setting greater than 0 keys up the transmitter at the indicated power level. A -power argument setting of 0 de-keys the transmitter. Before key-up, the operator must configure the appropriate freq, enable, peer_test_mode and data_pattern_mode settings.

NOTE

The operator must set "peer_test_mode" to "stop." Otherwise, residual power may still be present at the transmit port.

Examples:

```
field> freq -otxch1 -f860

field> dpm -otxch1 -miden

field> ptm -otx_all -mdnlk_framed

field> power -otxch1 -p30

field> power -otxch1

Forward Power=30.931942
Reflected Power=1.362578
VSWR=1.531271

field>
```

RESET

Syntax:

STRINGX -object= platform

Executes the Core reset control for Gen2 and QUAD BRs.

Example:

```
field> reset -oplatform

field>
```

REVISION_NUMBER (RN)

Syntax:

STRINGX -object = {ex1, pa1, rxch1, ps, control}

The RN command returns the revision number setting of the specified object.

Example:

```
field> rn -oex1

revision_number=R01.04.11

field>
```

SCRATCH

Syntax:

STRINGX -object = {control, ex1, pa1, rxch1, ps}

The SCRATCH command returns the scratch pad memory contents for the specified object.

Example:

```
field> scratch -orxch1

scratch=Motorola,Inc

field>
```

STATUS

Syntax:

STRINGX -object = {tx_all, rx_all}

The STATUS command returns status information for the specified object.

Examples:

```
field> status -otx_all

TX Sanity=Sane
TX CPU Load=13.283999
Number of TX Logical Channels=1
Lowest Operational TX Frequency=851.000000
Highest Operational TX Frequency=870.000000
Minimum TX Output Power Capable=5.000000
Maximum TX Output Power Capable=43.895771
Training Data= 0x85, 0x4A, 0x02, 0x7F, 0xF10

field> status -orx_all

RX1 Sanity=Sane
RX2 Sanity=N/A
RX1 CPU Load=48.609333
RX2 CPU Load=N/A
RXCH1 Lock=Unlocked
RXCH2 Lock=N/A
RXCH3 Lock=N/A
RXCH4 Lock=N/A
Number of RX Logical Channels Supported=14
Lowest Operational RX Frequency=806.000000
Highest Operational RX Frequency=825.000000
Number of Physical Branches per RX Channel=3

field>
```

SYSTEM_GAIN_ENABLE (SGE)

Syntax:

STRINGX -object = {rx_all}

STRINGX -state = {on,off}

The SGE command controls the rx system gain enable state.

Examples:

```
field> sge -orx_all

State=on

field> sge -orx_all -soff

field> sge -orx_all

State=off
```

TIME

Syntax:

None

The TIME command displays the current platform time setting in seconds since the last reset. The system stamps events in the alarm_log with this time as reference.

Example:

```
field> time

Time=0000002703

field>
```

Generation 2 Base Radio Application Examples

Key the BR

1. Set channel 1's transmit frequency.

```
field> freq -otxch1 -f860

field>
```

2. Enable the channels by setting a data pattern to "iden."

```
field> dpm -otxch1 -miden

field>
```

ATTENTION

The following command keys the transmitter. Make sure that transmission only occurs on licensed frequencies, or into an RF dummy load.

3. Set the test mode.

4. Set the transmit power to 40 watts. Key the BR.

```
field> power -otxch1 -p40

field>
```

Dekey Keyed BR

1. Set the transmit channel 1 power to 0.

```
field> power -otxch1 -p0

field>
```

2. Set the transmit DSP test mode to “stop.”

```
field> ptm -otx_all -mstop

field>
```

Query for BR Output Power

1. Query the current, channel 1 transmit power.

```
field> power -otxch1

field>
```

Query for RX Performance Data - Channel

1. Set the frequency of receive channel 1 to 806MHz.

```
field> freq -orxch1 -f806

field>
```

2. Enable receive channel 1.

```
field> enable -orxch1 -son

field>
```

3. Set the test mode of the receive channel 1 DSP to "uplk_framed."

```
field> ptm -orx_all -muplk_framed

field>
```

4. Configure the DSP performance characteristics of receive channel 1 for channel reports using slot 1 data.

```
field> ppc -orxch1 -mchn -s1

field>
```

5. Configure the receive channel 1 performance report. Begin reporting.

```
field> ppr -orxch1 -a1000 -r1

field>
```

Query for RX Performance Data - Path

1. Step1 - Set the frequency of receive channel 1 to 806MHz.

```
field> freq -orxch1 -f806

field>
```

2. Enable receive channel 1.

```
field> enable -orxch1 -son

field>
```

3. Set the test mode of the receive channel 1 DSP to “uplk_framed.”

```
field> ptm -orx_all -muplk_framed

field>
```

4. Configure the DSP performance characteristics of receive channel 1 for path 1 reports.

```
field> ppc -orxch1 -mpath -p1

field>
```

5. Configure the receive channel 1 performance report. Begin reporting.

```
field> ppr -orxch1 -a1000 -r1

field>
```

QUAD Channel BR Software Commands

Overview

This section provides definitions for Man-Machine-Interface (MMI) commands. MMI commands test and configure the EBTS equipment via a service computer. Two command sets accomplish these test and configuration tasks: The integrated Site Controller (iSC) command set and Base Radio (BR) command set.

The iSC Supplement to this manual describes the iSC command set in detail. This chapter section describes QUAD Channel base radio commands.

The following table lists this section's topics.

QUAD BR MMI Commands

This section describes Test Application MMI commands for QUAD Channel Base Radios. Command descriptions include syntax, definitions, and examples.

Call processing offers separate MMI commands that follow a different format. For information on the Call Processing MMI commands, refer to TBD.

An operator at a service computer inputs MMI commands to the system's RS-232 serial port. The transfer protocol is: 19,200 bps, 8-N-1. (That is, eight data bits, no parity, 1 stop bit.) Either the iSC or the front of each BRC in the RF Cabinet can access the RS-232 serial port.

The Base Radio test procedure uses MMI commands to test and configure the system. For Base Radio test procedures, refer to the Base Radio section of this manual.

NOTE

This section only covers Test Application MMI commands for QUAD Channel Base Radios. For a complete description of MMI commands pertaining to the iSC, refer to this manual's iSC Supplement. For Single Carrier BR operation, refer to the Single Carrier BR MMI command section.

Access Level

The Test Application requires the user to log in, using a defined user ID and password. Each login permits appropriate access to the MMI command set. The Test Application provides the following login IDs.

iDEN Generation 2 Login id's	Password	Configuration
field	motorola	factory default
field	Motorola	OMC default
field	<customer specified>	OMC customer set

NOTE

The 'Out of Box' default factory set field password is deleted and is replaced by the customer defined field password contained within the OMC. This occurs as soon as the controller module receives its initial OMC download.

NOTE

The OMC field password is customer configurable. Please contact the Operations and Maintenance Center (OMC) operator on duty to obtain your customer unique field password.

Example:

```
>login -ufield
password:<login password>

field>
```

Conventions

QUAD Base Radio MMI commands appear in alphabetical order. The command syntax is case sensitive. The syntax for each command appears as follows:

- Plain/**bold** text shows what to type to invoke a command or action.
- *Italic* text shows where to substitute a parameter or value.
- Text in braces { } indicates a selection or parameter that the operator **must** enter before the command can execute.
- A series of dots ... indicates one or more occurrences of a preceding parameter.

QUAD Channel BR command syntax is case sensitive. Each example appears in the format that the operator should enter.

A definition details each command's purpose and function. Where helpful, an example of the command's response follows the definition. Whenever possible, this section uses typical values.

Some commands return varying responses (such as "available," "not available," "unknown," "o.k.," and "alarm"). Each example only lists one of the possible responses. Examples do not go through every possible scenario.

QUAD Base Radio Commands

Command Detail

The commands in bold text appear in long form, followed in parenthesis by the short form.

ALARMS

Syntax:

STRINGX -object = {fault_hndlr}

The ALARMS command returns the active alarms.

Example:

```
field> alarms -ofault_hndlr

ACTIVE_FAULT_ID=RX_LO1_LOCK

field>
```

CABINET_ID (CI)

Syntax:

STRINGX -object = {platform}

STRINGX -cabinet = {any positive integer}

The CI command controls the cabinet identification number.

Examples:

```
field> ci -oplatform
Cabinet ID=0

field> ci -oplatform -c1

field> ci -oplatform
Cabinet ID=1
```

DATA_PATTERN_MODE (DPM)

Syntax:

STRINGX -object = {txch(1->4)}

STRINGX -mode = {none, iden}

The DPM command controls the mode setting for the platform object's transmit path data pattern.

Examples:

```
field> dpm -otxch1

Mode=none

field> dpm -otxch1 -miden

field> dpm -otxch1

Mode=iden
```

DIVERSITY

Syntax:

STRINGX -**o***bject* = {rx_all}

STRINGX -**d***iversity* = {right justified binary bit map of the receive branch configuration, the right most bit field corresponds to receive branch 1}

NOTE: This setting does not change the platform in anyway. It is simple used as a configuration parameter.

Examples:

field> diversity -orx_all -d0 #Turn all branches off

field> diversity -orx_all -d1 #enable branch 1; 2 & 3 are off

field> diversity -orx_all -d10 #enable branch 2; 1 & 3 are off

field> diversity -orx_all -d11 #enable branch 1 & 2; 3 is off

field> diversity -orx_all -d100 #enable branch 3; 1 & 2 are off

field> diversity -orx_all -d101 #enable branch 3 & 1; 2 is off

field> diversity -orx_all -d110 #enable branch 3 & 2; 1 is off

field> diversity -orx_all -d111 # enable 1, 2 and 3

ENABLE

Syntax:

STRINGX -object = {txch{1->4}, rxch{1->4}}

for -object = rxch{1->4}

STRINGX -diversity = {br{1->3}}

STRINGX -state = {on, off, dumb}

The ENABLE command controls the enabled state of txch and rxch objects.

Diversity option only applies to Rx.

Examples:

```
field> enable -otxch1 -son

field> enable -otxch1
State=On

field> enable -orxch1 -son

field> enable -orxch1

br1=on
br2=on
br3=on

field> enable -orxch1 -dbr2 -soff

field> enable -orxch1

br1=on
br2=off
br3=on
```

EXTERNAL_SYNCHRONIZATION (ES)

Syntax:

STRINGX -object = {rx_all, tx_all}

for -object = rx_all

STRINGX -type = {trig_def, global, ext_trigger, pps, in_ptt}

for -object = tx_all

STRINGX -type = {trig_def,

out_pp3s, out_frame, out_slot, out_ptt, out_mframe,

out_fs_bs}

The ES command controls the object's external synchronization method.

Examples:

```
field> es -orx_all

Type=trig_def

field> es -orx_all -text_trigger

field> es -orx_all

Type=ext_trigger
```

FIRMWARE_VERSION (FV)

Syntax:

STRINGX -object = {control, rxch1, txch1, platform. PS}

The FV command returns the object's firmware version. Specifying the object 'platform' causes the software to return additional information.

Example:

```
field> fv -oplatform

Test Application Version =D01.01.07

Controller Rev No    = R05.01.01
Controller Kit No    = CLF6290B
Controller Firmware  = 1.222
Power Supply Rev No  = R05.01.01
Power Supply Kit No  = CLN7275A
Core Software Ver    = 1.222
Platform ID          =
Tx1 Exciter Rev No   = R05.01.01
Tx1 Exciter Kit No   = CLF6290B
Tx1 Exciter Firmware = 00.00.18
Tx1 PA Rev No        = R05.00.00
Tx1 PA Kit No        = CLF1400A
Tx1 PA Firmware      =
Rx1 Rev No           = R05.01.00
Rx1 Kit No           = CRF6060A
Rx1 Firmware         = 00.00.15
Rx2 Rev No           = R05.01.00
Rx2 Kit No           = CRF6060A
Rx2 Firmware         = 00.00.15
Rx3 Rev No           = R05.01.00
Rx3 Kit No           = CRF6060A
Rx3 Firmware         = 00.00.15
Rx4 Rev No           = R05.01.00
Rx4 Kit No           = CRF6060A
Rx4 Firmware         = 00.00.15

Core Major Version=1
Core Minor Version=222
```

FREQ

Syntax:

STRINGX -object = {txch{1->4}, rxch{1->4}}

FLOATX -frequency = {use status -orx_all or status -otx_all for applicable limits}

The FREQ command controls synthesizer frequency settings for the rxch and txch objects.

Examples:

```
field> freq -otxch2

freq=860.025000

field> freq -otxch2 -f865

field> freq -otxch2

freq=865.000000

field> freq -orxch3

freq=820.000000

field> freq -orxch3 -f810

field> freq -orxch3

freq=810.000000

field>
```

FRU_CONFIGURATION (FC)

Syntax:

STRINGX -object = {platform}

The FC command returns the fru configuration.

Example:

```
field> fc -oplatform

type=BRC
slot=Slot 01
revision number=R05.00.00
kit number=ABC1234A
flags=Available,Supported, Known

type=Power Supply
slot=Slot 00
revision number=R05.00.00
kit number=ABC1234A
flags=Available,Supported, Known

type=Receiver
slot=Slot 06
revision number=R05.01.00
kit number=CRF6060A
flags=Available,Supported, Known
.
.
.
type=Exciter
slot=Slot 01
revision number=R05.00.00
kit number=ABC1234A
flags=Available,Supported, Known

type=Power Amplifier
slot=Slot 02
revision number=R05.00.00
kit number=CLF1400A
flags=Available,Supported, Known
```

HELP

Syntax:

STRINGX -command = {any string}

The HELP command displays help text for the test application. Help topics for the entered command string appear in alphabetical order. HELP operates similarly to

a grep (search and replace text utility), where the string is part or all of the command.

Example:

```
field> help -cpower
power: This command controls the platform object power settings

    arguments:
        -o(object)=txch1,txch2,txch3,txch4 (for write)
        -o(object)=txch1,txch2,txch3,txch4,ta,wmt (for read)
        -p(power)=2.0-40.0 Watts on LSS, 5.0-56.5 Watts on Quad
        -a(auto power)=LSS Only: 2.0 - 40.0 W
```

KIT_NUMBER (KN)

Syntax:

STRINGX -object = {ex1, pa1, rxch{1->4}, ps, control}

The KN command returns the object's kit number setting.

Example:

```
field> kn -opa1

kit_number=CLF1400A

field>
```

LOGIN

Syntax:

STRINGX -userid = {dev, factory, field}

The LOGIN command allows the user to log in as a valid user. Note that each login requires a password and that the word 'login' be part of the command string.

Example:

```
> login -ufield
password: <field login password>

field>
```

iDEN Generation 2 Login id's	Password	Configuration
field	motorola	factory default
field	Motorola	OMC default
field	<customer specified>	OMC customer set

NOTE

The 'Out of Box' default factory set field password is deleted and is replaced by the customer defined field password contained within the OMC. This occurs as soon as the controller module receives its initial OMC download.

NOTE

The OMC field password is customer configurable. Please contact the Operations and Maintenance Center (OMC) operator on duty to obtain your customer unique field password.

LOGOUT

Syntax:

none

The LOGOUT command logs out the current user and displays the login prompt. No reset occurs.

Example:

```
field> logout

field>
```

PEER_PERFORMANCE_CONFIG (PPC)

Syntax:

STRINGX -object = {rxch(1->4)}

STRINGX -mode = {chn, path}

for -mode = path

INTX -path = {1,2,3, all}

for -mode = chn

INTX -slot = {1->6}

The PPC command controls the platform peer performance configuration. A -mode argument setting of “chn” provides the channel performance for all enabled branches. A -mode argument setting of “path” provides the individual receive path performance.

Examples:

```
field> enable -orxch1 -son

field> ppc -orxch1 -mpath -pall

field> ppc -orxch1

Mode=path
Path=all

field> ppc -orxch1 -mchn -s1

field> ppc -orxch1

Mode=chn
Slot=1
```

PEER_PERFORMANCE_REPORT (PPR)

Syntax:

STRINGX -object = {rxch{1->4}. rx_all}

INTX -averages = {2-65535 max}

INTX -reports = {1-65535 max}

The PPR command returns the peer performance information. The operator must complete appropriate peer_performance_config settings before generating reports.

Example:

```
field> ppr -orxch1 -a10 -r1

SGC Atten.(dBm)=0.000000
Freq. Offset=-0.000031
Sync. Attempts=1.000000
Sync. Successes=0.000000
BER%=50.290699
RX Path1 RSSI=-129.727005
RX Path2 RSSI=-130.369965
RX Path3 RSSI=-132.486801
Chn sig. strength=-68.348839
Chn intf. strength=-68.348839
```

PEER_TEST_MODE (PTM)

Syntax:

STRINGX -object = {tx_all, rx_all}

for -object = rx_all

-mode = {uplk_framed}

for -object = tx_all

-mode = {dnlk_framed, tones, stop, suspend}

The PTM command controls the platform peer (DSP) test mode setting.

Examples:

```
field> ptm -otx_all

Mode=not_in_test

field> ptm -otx_all -mdnlk_framed

field> ptm -otx_all

Mode=dnlk_framed
```

POSITION_ID (PI)

Syntax:

STRINGX -object = {platform}

STRINGX -position = {any positive integer}

The PI command controls the platform position number.

Examples:

```
field> pi -oplatform

Position=0

field> pi -oplatform -p5

field> pi -oplatform

Position=5

field>
```

POWER

ATTENTION

The POWER command keys the transmitter. Verify transmission only occurs on licensed frequencies or into an RF dummy load.

Syntax:

STRINGX -object = {tx_all, txch{1-4}}

FLOATX -power = {use status -otx_all for applicable limits}

The POWER command controls the platform object's transmit output power. A -power argument setting greater than 0 keys up the transmitter at the indicated power level. A -power argument setting of 0 de-keys the transmitter. Before key-up, the operator must configure the appropriate freq, enable, peer_test_mode and data_pattern_mode settings.

NOTE

The operator must set "peer_test_mode" to "stop." Otherwise, residual power may still be present at the transmit port.

Examples:

```
field> freq -otxch1 -f860

field> dpm -otxch1 -miden

field> ptm -otx_all -mdnlk_framed

field> power -otxch1 -p30

field> power -otxch1

Forward Power=30.931942
Reflected Power=1.362578
VSWR=1.531271
Auto Power=0.000000

field>
```

RESET

Syntax:

STRINGX -object= platform

Executes the Core reset control for Gen2 and QUAD BRs.

Example:

```
field> reset -oplatform

field>
```

REVISION_NUMBER (RN)

Syntax:

STRINGX -object = {ex1, pa1, rxch{1->4}, ps, control}

The RN command returns the revision number setting of the specified object.

Example:

```
field> rn -oex1

revision_number=R05.00.00

field>
```

SCRATCH

Syntax:

STRINGX -object = {control, ex1, pa1, rxch{1->4}, ps}

The SCRATCH command returns the scratch pad memory contents for the specified object.

Example:

```
field> scratch -orxch1

scratch=Motorola,Inc

field>
```

STATUS

Syntax:

STRINGX -object = {tx_all, rx_all}

The STATUS command returns status information for the specified object.

Examples:

```
field> status -otx_all

TX Sanity=Sane
TX CPU Load=13.283999
Number of TX Logical Channels=4
Lowest Operational TX Frequency=851.000000
Highest Operational TX Frequency=870.000000
Minimum TX Output Power Capable=5.000000
Maximum TX Output Power Capable=62.150002
Training Data=0x88, 0x51, 0x8e, 0xbf, 0xff, 0x7f, 0x13, 0x49, 0x7f, 0x7f,
0x55

field> status -orx_all

RX1 Sanity=Sane
RX2 Sanity=Sane
RX1 CPU Load=48.609333
RX2 CPU Load=48.609333
RXCH1 Lock=Unlocked
RXCH2 Lock=Unlocked
RXCH3 Lock=Unlocked
RXCH4 Lock=Unlocked
Number of RX Logical Channels Supported=4
Lowest Operational RX Frequency=806.000000
Highest Operational RX Frequency=825.000000
Number of Physical Branches per RX Channel=3
```

SYSTEM_GAIN_ENABLE (SGE)

Syntax:

STRINGX -object = {rx_all}

STRINGX -state = {on,off}

The SGE command controls the rx system gain enable state.

Examples:

```
field> sge -orx_all

State=on

field> sge -orx_all -soff

field> sge -orx_all

State=off
```

TIME

Syntax:

None

The TIME command displays the current platform time setting in seconds since the last reset. The system stamps events in the alarm_log with this time as reference.

Example:

```
field> time

Time=0000002703

field>
```

QUAD Application Examples

Key the BR (4 channels)

1. Set the frequency of transmit channel 1 through 4.

```
field> freq -otxch1 -f860
field> freq -otxch2 -f860.025
field> freq -otxch3 -f860.050
field> freq -otxch4 -f860.075

field>
```

2. Enable the channels by setting a data pattern to “iden.”

```
field> dpm -otxch1 -miden
field> dpm -otxch2 -miden
field> dpm -otxch3 -miden
field> dpm -otxch4 -miden

field>
```

ATTENTION

The following command keys the transmitter. Make sure that transmission only occurs on licensed frequencies, or into an RF dummy load.

3. Set the transmit DSP test mode to “dnlk_framed.”

```
field> ptm -otx_all -mdnlk_framed
field>
```

4. Set the transmit power to 40 watts. Key the BR

```
field> power -otxch1 -p40

field>
```

Dekey a Keyed BR

1. Set the transmit channel 1 power to 0.

```
field> power -otxch1 -p0

field>
```

2. Set the transmit DSP test mode to "stop."

```
field> ptm -otx_all -mstop

field>
```

Query the BR Output Power

1. Query the current, channel 1 transmit power.

```
field> power -otxch1

field>
```

Query the RX performance data - Channel

1. Set the frequency of receive channel 1 to 806MHz.

NOTE

Substitute **898** for **806** for 900 MHz QUAD BR

```
field> freq -orxch1 -f806

field>
```

2. Enable receive channel 1.

```
field> enable -orxch1 -son

field>
```

3. Set the test mode of the receive channel 1 DSP to “uplk_framed.”

```
field> ptm -orx_all -muplk_framed

field>
```

4. Configure the DSP performance characteristics of receive channel 1 for channel reports using slot 1 data.

```
field> ppc -orxch1 -mchn -s1

field>
```

5. Configure the receive channel 1 performance report. Begin reporting.

```
field> ppr -orxch1 -a1000 -r1

field>
```

Query the RX performance data - Path

1. Step1 - Set the frequency of receive channel 1 to 806MHz.

```
field> freq -orxch1 -f806

field>
```

2. Enable receive channel 1.

```
field> enable -orxch1 -son

field>
```

3. Set the test mode of the receive channel 1 DSP to “uplk_framed.”

```
field> ptm -orx_all -muplk_framed

field>
```

4. Configure the DSP performance characteristics of receive channel 1 for path 1 reports.

```
field> ppc -orxch1 -mpath -p1

field>
```

5. Configure the receive channel 1 performance report. Begin reporting.

```
field> ppr -orxch1 -a1000 -r1

field>
```

Query the Alarm log

1. Display the Alarm Log contents.

```
field> alarms -ofault_hndlr

field>
```