Agilent E6651A Mobile WiMAX™ Test Set

User's Guide

Agilent Technologies

Notices

WiMAX™ is a certification mark of the WiMAX Forum.

Windows® and MS Windows XP® are U.S. registered trademarks of Microsoft Corporation.

Edition

First Edition, July 2007

Agilent Technologies UK, Ltd
Station Road, South Queensferry
West Lothian EH30 9TG, UK

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Contents

1 Introduction

This document describes the operation of the E6651A Test Set, a Mobile WiMAX 802.16e Subscriber Station Tester. This product is used for protocol verification and performance measurement of Mobile WiMAX Subscriber Stations (SS). The E6651A Test Set was designed to be operated as a standard Mobile WiMAX base station, and is also capable of analyzing and generating Mobile WiMAX signals.

Product Overview

The E6651A Test Set is designed to test and analyze the performance and signaling of Mobile WiMAX subscriber stations based on the IEEE 802.16e standard. The Test Set consists of three major operating modes:

Figure 1 The Agilent E6651A Mobile WiMAX Test Set

Base Station Emulator (BSE)

In Base Station Emulator (BSE) mode, the Test Set simulates the operation of a Mobile WiMAX base station.

Signal Analyzer (SA)

In Signal Analyzer (SA) mode, the Test Set can be used to analyze Mobile WiMAX uplink signals using modulation, spectrum and flatness analysis. Modulation Analysis mode displays the OFDM signal in both frequency and time domain. The Spectrum Analysis functionality, implemented using a Fast Fourier Transform (FFT) algorithm, displays the measured WiMAX signal in the frequency domain. Flatness Analysis enables comparison of power levels of the spectral components.

Signal Generator (SG)

In Signal Generator (SG) mode, the Test Set generates standard Mobile WiMAX uplink and downlink signals.

Functions and Features

E6651A Features

- IEEE 802.16e OFDMA Mobile WiMAX subscriber station tester
- Real-time Mobile WiMAX downlink signal modulation
- Real-time Mobile WiMAX uplink demodulation
- Base station emulation with MAC, protocol stack
- TDD synchronization (auto-switching) of signal generator and signal analyzer
- Ranging code detection
- Various preamble, FCH, DL-MAP, UL-MAP, burst configuration settings
- DL PUSC, DL FUSC, UL PUSC

Mobile WiMAX radio (PHY) measurements

- OFDM constellation and EVM (BPSK, QPSK, 16QAM, 64QAM)
- UL data EVM
- Error vector spectrum, error vector versus time
- CCDF (complimentary cumulative distribution function)
- Channel power, occupied bandwidth

Transmitter

- Modulation and coding
- Ranging support
- Power level control
- Synchronization
- Maximum output signal
- Frequency accuracy
- Channel bandwidth measurement
- Spectral flatness
- Relative constellation error (EVM)

Receiver

- Maximum tolerable signal
- Maximum input signal
- Sensitivity
- Cyclic prefix
- Preambles
- Frame duration timing
- RSSI
- BER versus CINR

Mobile WiMAX MAC test items

- Response time for MAC management request message
- Initial ranging procedure verification under exceptional RF conditions
- Dynamic service flow status for given conditions
- SS MAC layer verification and performance test

General Specifications

Environmental

Operating Temperature	0°C to +55°C
Storage Temperature:	-20°C to +70°C
Humidity:	15% to 95% Relative Humidity at +40°C
Altitude:	3000m (9,840 ft.)
EMC:	Meets EN55011: 1991 (Group 1, Class A), and EN50082-1:1992.

Physical Specifications

Weight (Net):	25.8 Kg
Dimensions:	222 H x 444 W x 647 D mm nominal

Power Requirements

This equipment has an IEC 60320-1 C14 inlet for connecting a detachable mains cord set.

CAUTION This instrument has an autoranging line voltage input, ensure the supply voltage is within the specified range.

Table 1 Agilent E6651A Mobile WiMAX Test Set Power Requirements

	**Line Power:**	**Input Voltage Range:** 100 to 240 Vac, automatic selection
		Input Frequency Range: 50 to 60 Hz
		Power Requirement: 150 VA (max)

WARNING **This is a Safety Class 1 Product (provided with a protective earthing ground, incorporated in the power cord). The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. Any interruption of the protective conductor inside or outside of the instrument is likely to make the instrument dangerous. Intentional interruption is prohibited.**

CAUTION

Ventilation Requirements: When installing the instrument in a cabinet, the convection into and out of the instrument must not be restricted. The ambient temperature (outside the cabinet) must be lest than the maximum operating temperature of the instrument by 4×C for every 100 Watts dissipated in the cabinet. If the total power dissipated in the cabinet is greater then 800 Watts, then forced convection must be used.

Agilent Sales and Service Offices

In any correspondence or telephone conversations, refer to the Test Set by its model number and full serial number. With this information, the Agilent representative can quickly determine whether your unit is still within its warranty period.

UNITED STATES	Agilent Technologies (tel) 1 800 829 4444
CANADA	Agilent Technologies Canada Inc. Test & Measurement (tel) 1 877 894 4414
EUROPE	Agilent Technologies Test & Measurement European Marketing Organization (tel) (31 20) 547 2000
JAPAN	Agilent Technologies Japan Ltd. (tel) (81) 426 56 7832 (fax) (81) 426 56 7840
LATIN AMERICA	Agilent Technologies Latin America Region Headquarters, USA (tel) (305) 267 4245 (fax) (305) 267 4286
AUSTRALIA and NEW ZEALAND	Agilent Technologies Australia Pty Ltd. (tel) 1-800 629 4852 (Australia) (fax) (61 3) 9272 0749 (Australia) (tel) 0-800 738 378 (New Zealand) (fax) (64 4) 802 6881 (New Zealand)
ASIA PACIFIC	Agilent Technologies, Hong Kong (tel) (852) 3197 7777 (fax) (852) 2506 9284

You can visit our website: - http://www.agilent.com/find/assist.

1 Introduction

2 Getting Started

This chapter describes the primary hardware and software interfaces used in the operation of the E6651A Test Set, as well as basic operating procedures for the system.

Basic Configuration

Front Panel

To begin using the E6651A Test Set, you should become familiar with the layout of the Front Panel and the displayed menu systems. The items described in this section are the Front and Rear Panel of the Test Set, the Measurement Screen, the Setting Window, and the Menu Tree configuration.

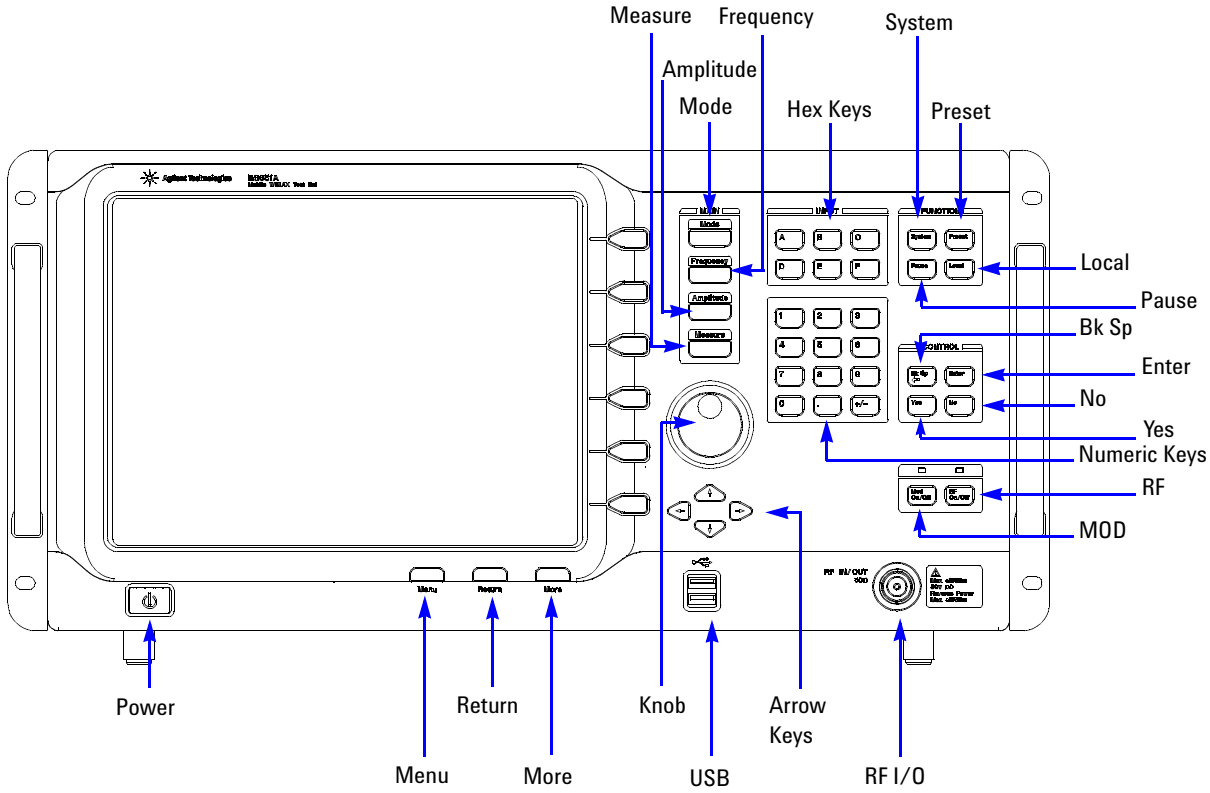


Figure 2 Front Panel Configuration

Power This is the On/Off button for AC power. Pressing this button when the Test Set is powered off turns it on. Pressing this button after the OS has completed booting turns the power off.

The **Power** button does *not* power the Test Set off while the Graphical User Interface (GUI) is running. To power off from this state, press the **No** button on the Front Panel to close the GUI, followed by the **Power** button. If pressing the **Power**

button does not turn the power off, hold the button down for 5 seconds to bypass the operating system termination, and power off.

LCD Screen Measurement results are displayed on the LCD Screen. Different screen layouts are used for each mode of operation. In each layout, the screen is divided into three areas: the Setting Window, the Working Window, and the Selection Menu.

Menu Press the **Menu** button to display the top level menu for the current Measurement Window.

Return Press the **Return** button to display the previous menu for the current Measurement Window.

More Use this button to select additional options when more than 6 menu options are available.

Knob Increments and decrements the value of the currently selected parameter.

USB Port The Test Set software runs on an embedded operating system. Devices using a USB interface may be connected to this port.

Arrow Key Move the on-screen cursor using the Left and Right Arrow Keys.

Numeric and Hex Keys Parameters like frequency can be input using these keys. Hexadecimal values can be input using the Hexadecimal Keys.

RF I/O An antenna or cable is connected to this port for communication with the subscriber station. This port can act as an RF input port, an RF output port or a duplex port based on the mode of operation. In BSE mode, this port automatically switches between input and output based on the downlink and uplink frame duration. In SA mode, this port may operate as an RF input or duplex port.

Menu Selection Keys Six buttons are available on the right hand side of the LCD Screen for menu selection. The current menu is displayed at the right side of the screen. When more than 6 menu options are available, select **More** to see the additional options.

Amplitude Adjust values related to input power using this button, including Amplitude, Attenuation, Reference Level, Scale and Amplitude Offset.

Frequency Use this button to set frequency-related values including Center Frequency, Frequency Offset, Resolution Bandwidth (RBW), Channel Bandwidth (CBW) and SPAN. The Center Frequency is used as the starting point for frequency parameter adjustment.

Mode Use this button to select the BSE (Base Station Emulator), SA (Signal Analyzer) or SG (Signal Generator) mode of operation.

Measure Use this button to display the Measurement Menu in SA mode. When the unit is in BSE or SG mode, pressing this button switches the mode to SA. SA measurement selections available in this mode include Spectrum Analysis, WiMAX Modulation Analysis, I/Q input Analysis, Error Vector Spectrum Analysis, and CCDF.

System This button displays the Mode Setup Menu including options for Input Source Selection, Reference Clock Selection, Trigger On/Off Selection, I/Q Output Level Adjustment, Connect E6655A On/Off Selection, Machine ID Adjustment, UL Permbase Adjustment and I/Q Reverse On/Off Selection.

Preset Use this button to revert most parameters to their default values and some are unaffected.

Pause Use the **Pause** button to stop Test Set operation in BSE or SA mode. When Pause status is active, PAU in Setting Window is lit in red.

Local Use this to return the Test Set to front panel control after remote interface operation.

Bk Sp Press the **Bk Sp** (back space) key to delete the selected digit.

ENT Press the **Enter** key to apply inputs and terminate input selection.

Yes This button is used to confirm the action or choice presented in the Yes/No window

No Press this button to terminate the Test Set GUI.

RF Press this button to generate an RF modulation signal. Both the **RF** and **MOD** functions must be enabled to generate a WiMAX output signal.

MOD Press this button to generate a modulation signal. Both the **RF** and **MOD** functions must be enabled to generate a WiMAX output signal.

Rear Panel

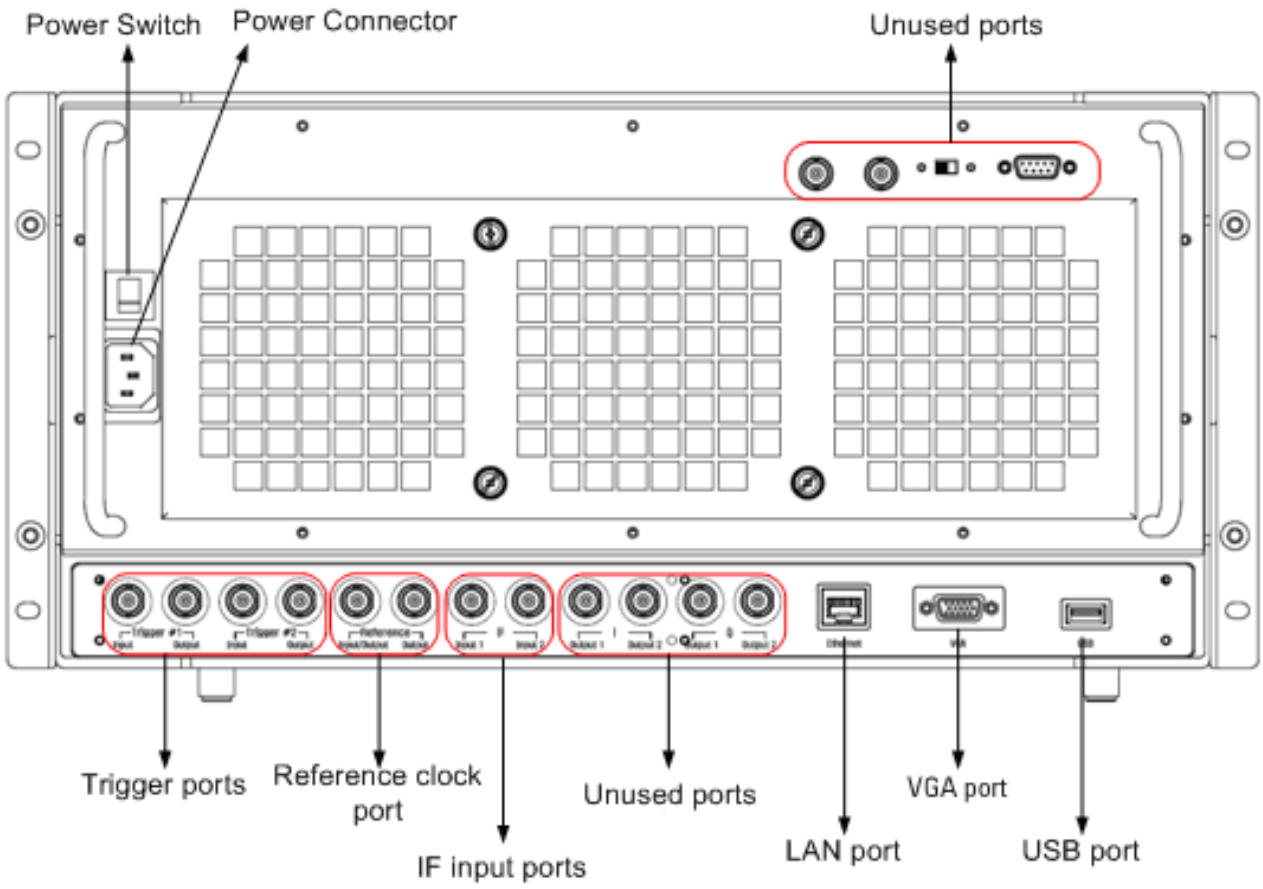


Figure 3 The E6651A Test Set Rear Panel

Power Switch This switch must be 'ON' to enable switching the Test Set on and off using the Front Panel On/Off switch.

Power Connector: Connect the AC power cable here.

Reference clock Port Use the 10 MHz Clock Port to synchronize all system clocks of the Test Set with the Device Under Test (DUT). Use the Output Port if you want to supply the DUT with the Test Set's clock. Use the Input Port if you want to provide the DUT's clock to the Test Set.

Trigger Ports Use the Trigger Port to synchronize the WiMAX TDD frame of the Test Set with the DUT. Use the Output Port if you want to apply the synchronization signal from the Test Set to the DUT. Use the Input Port if you want to apply the signal from the DUT to the Test Set. Trigger #2 ports are used to provide downlink and uplink transition information.

IF input Ports The Test Set gets IF input signal from external RF device using IF input ports. This port is not used in current product version.

LAN Port This port is used to interface with an external internet network or controller. When both are required simultaneously, an Ethernet hub may be connected here. A hub is required when connecting to the Agilent E6655A WiMAX Lab Application.

VGA Port Standard VGA output port.

USB Port Devices using a USB interface such as a keyboard or mouse may be connected to this port.

Measurement Screen Configuration

This section describes the main areas of the Measurement Screen including the Setting Window, Working Window, Selection Menu, Program Title, Menu Title and Program Info.

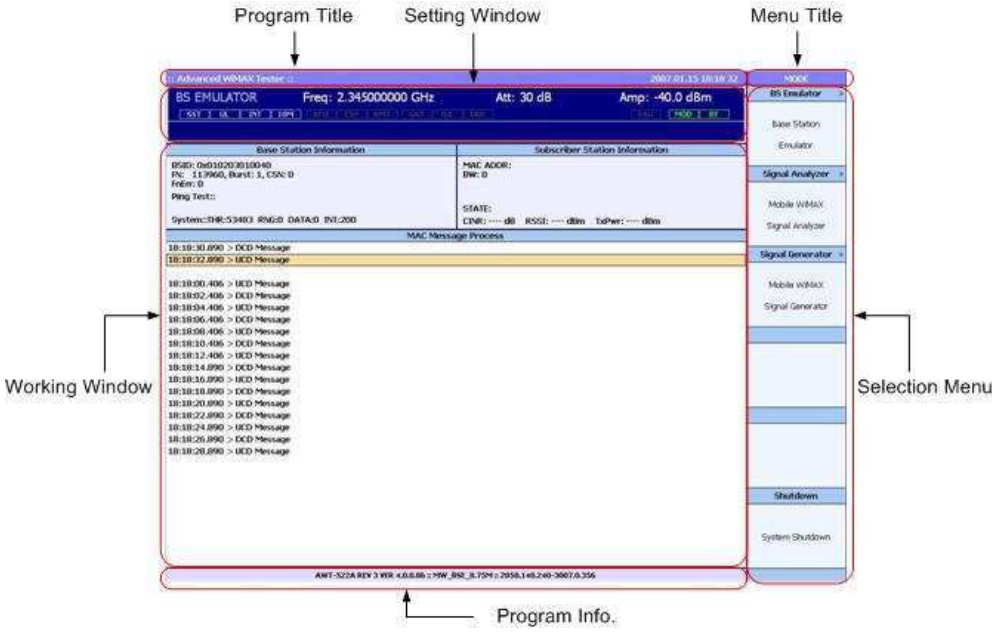


Figure 4 E6651A Test Set Measurement Screen Configuration

Setting Window This window displays settings including the current operation mode, the frequency, the attenuator setting, and the input value. Refer to "Setting Window" on page 22 for more information.

Working Window This window displays measurement values in graphical or numeric format. The items displayed here are applicable to the current mode of operation.

Selection Menu Selectable menu items are displayed here, aligned with the menu selection buttons. Select individual items using the selection buttons or Mouse. Menu items shown in Figure 4 include the BS Emulator, Signal Analyzer, Signal Generator.

Program Title The Test Set program title, current date and current time are displayed in this area. The date and time displayed are based on the system's PC clock time.

Menu Title The title of the current Selection Menu is displayed here.

Program Info The program information, including version information, is displayed in this area.

Setting Window

This window contains detailed information about the mode of operation and parameter settings.

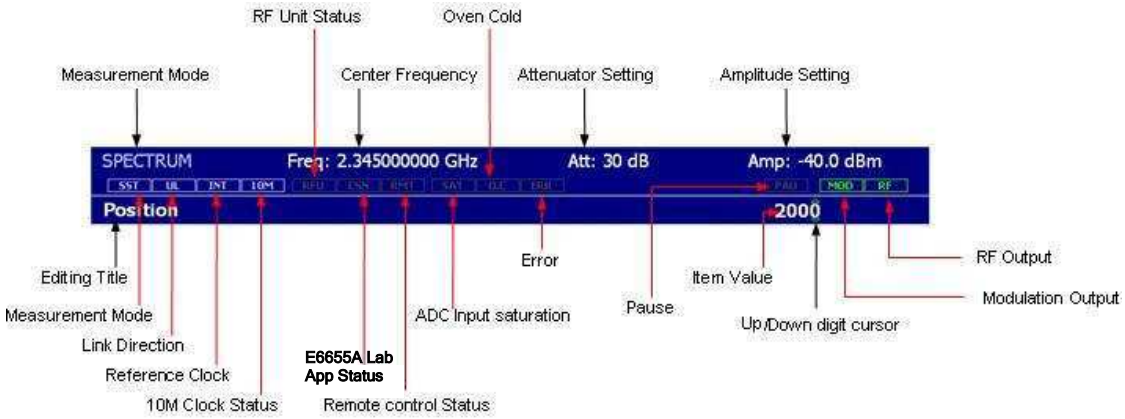


Figure 5 Setting Window Configuration

Measurement Mode The current mode of operation (**BSE**, **SA** or **SG**) is displayed in this window.

Center Frequency The current center frequency is displayed here.

Link Direction The Link Direction display always indicates "**UL**" in SA and BSE mode.

ADC Input Saturation Saturation may cause the signals to become distorted. When this condition occurs, the **SAT** Indicator is illuminated in one of four colors.

Attenuator Setting The current Attenuator Setting is displayed here.

Error When an error occurs during operation, the **ERR** Indicator is illuminated in red.

Amplitude Setting This displays the Test Set's current transmitter power.

RF Output This area displays the status of the RF output. When the RF output is on, this area is highlighted in green.

Editing Title The Editing Title shows the parameter currently selected to be modified.

Measurement mode This area indicates the selected mode.

Reference Clock The Reference Clock selection is displayed in this area as either INT (Internal) or EXT (External).

Oven Cold When the Oven Controlled Crystal Oscillator (OCXO), used to generate the internal reference clock, has not sufficiently 'warmed up' for reliable operation, the **O.C.** Indicator is illuminated in red.

Pause This indicator lights up when the **Pause** button is pressed during Test Set operation.

Item Value: The Item Value shows the current value of the parameter selected to be modified.

Up/Down Digit Cursor This indicator marks the position of the digit selected for modification. Use the Arrow Keys to select the digit you want to modify. Increase or decrease the value of the selected digit using the Arrow Keys or the Knob.

Modulation Output This area displays the status of the Modulation. When the Modulation is on, this area is highlighted in green.

Lab Application Status When the Test Set interoperates with the E6655A WiMAX Lab Application to connect to an external IP network this area is highlighted in green.

Remote control Status When Test Set is remote controlled, this area is highlighted in yellow.

10M Clock Status This area displays status of 10MHz clock.

Menu Tree Configuration

The following diagrams show how the menu structure is organised under the Mode, Frequency, Amplitude, Measure and System hardkeys.

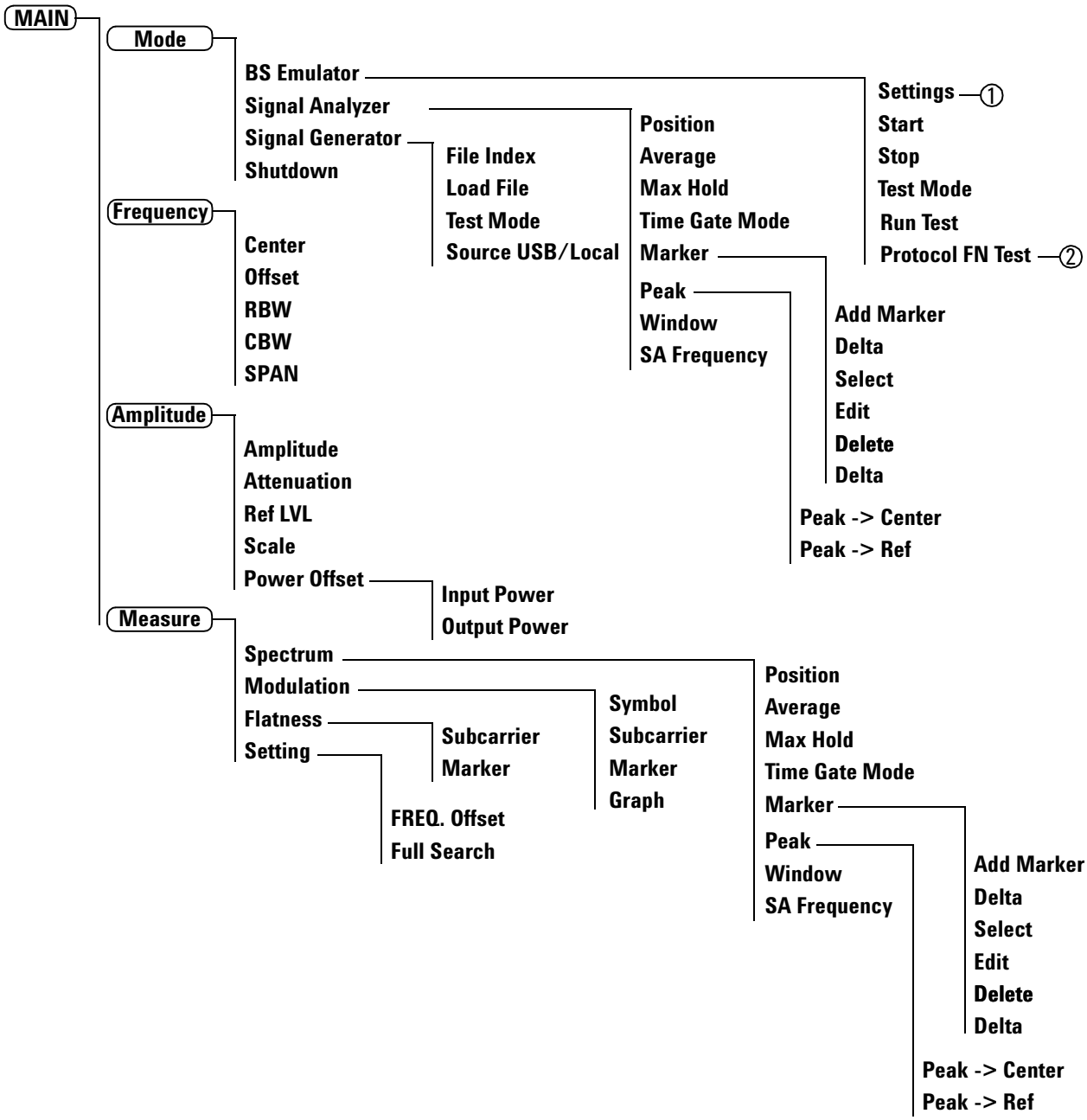


Figure 6 Menu Tree Configuration 1 of 2

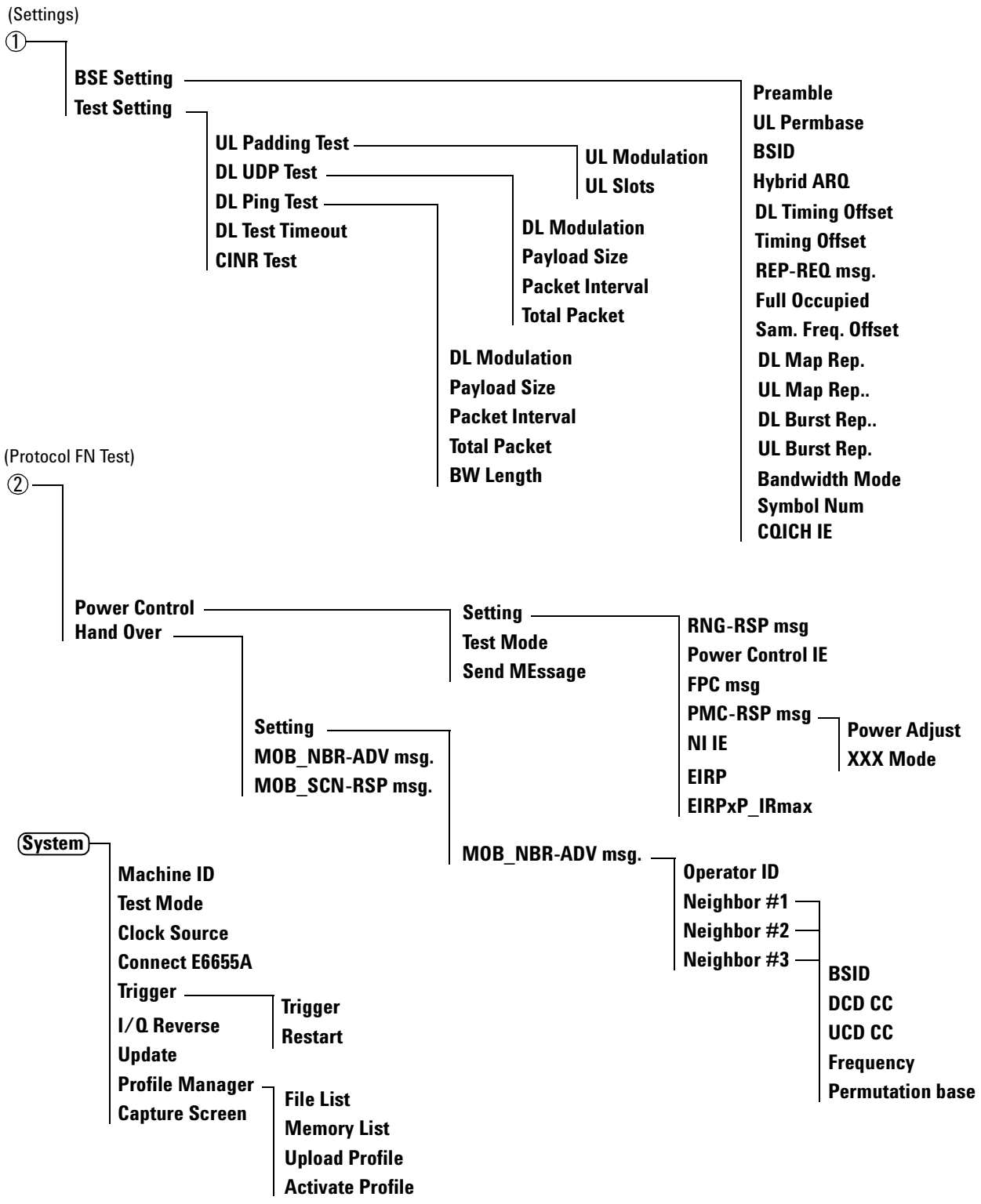


Figure 7 Menu Tree Configuration 2 of 2

Basic Operation

Turning the Power ON

1. **Connect the AC power cable:** Insert the AC power cable into the power inlet on the rear panel of the Test Set.
2. **Turn on the Power Switch:** Turn on the Power Switch above the AC power inlet.
3. **Power the Unit On:** With the system in stand-by, press the Power button on the Front Panel. The Power LED turns green, indicating that the Test Set is in Power On State. The E6651A Test Set program starts automatically.
4. **Program Start-Up:** Figure 3-6 depicts the start-up screen displayed while the Test Set measurement application loads. Please be aware that this process may take several minutes.

Figure 8 E6651A Test Set Program Start-Up Screen

Operational Mode Selection

When the application has loaded completely, the Operational Mode Selection Window, shown in Figure 9, is displayed.

Figure 9 E6651A Test Set Operational Mode Selection Window

Press the **Mode** button on the Front Panel to display the Operational Mode Selection Menu on the right side of the screen. The three operational modes available are:

BS Emulator (Base Station Emulator) Press this button to select BSE mode. In this mode, the Test Set simulates a standard Mobile WiMAX base station for air interface and subscriber station communication testing.

Signal Analyzer (Mobile WiMAX Signal Analyzer) Press this button to select SA mode. Alternatively, press the Measure button on the Front Panel for SA mode operation. In this mode, the Test Set performs spectrum analysis and modulation analysis for standard Mobile WiMAX signals.

Signal Generator (Mobile WiMAX Signal Generator) Press this button to select SG mode. In this mode, the Test Set transmits a standard Mobile WiMAX signal stored in memory. You can choose from among several signals and adjust the output power of the signal.

Refer to Figure 6 on page 24 for details of the menu structure for each operational mode.

Turning the Power OFF (Normal Termination)

To power the Test Set off:

1 **Terminate the program:** Select **Mode > shutdown**. The dialog box shown in Figure 10 is displayed.

2 Press the **Yes** button on the front panel.

CAUTION Turning the power OFF by pressing power button may cause damage to the Test Set.

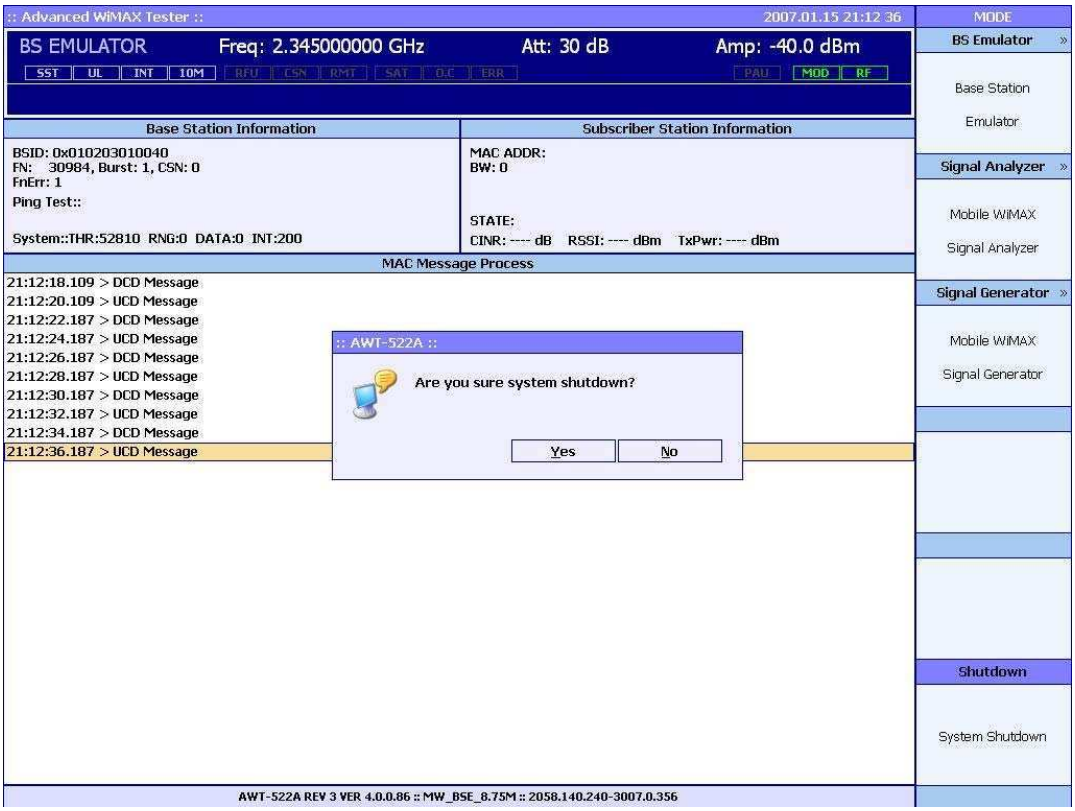


Figure 10 Terminate the E6651A Test Set program

3 When the measurement applications and Windows have closed, the message "**It is now safe to turn off your computer**" is displayed. Press the front panel Power button to power down the Test Set.

Turning the Power OFF (Abnormal Termination)

When the program cannot be terminated normally, terminate manually by holding down the **Power** button for more than 5 seconds.

CAUTION Do not turn the power off while the equipment is reading to or writing from the internal disk (this includes the OS boot process and program initiation). Doing so may damage the disk's internal file system.

3
Base Station Emulator Mode

In Base Station Emulator (BSE) mode, the E6651A Test Set simulates a Mobile WiMAX base station's operation for use in the development and test of Mobile WiMAX subscriber stations. This section describes the procedure for interfacing with a subscriber station and for running uplink and downlink tests in BSE mode.

Interfacing With the Subscriber Station

To begin interfacing with the Subscriber Station (SS) in BSE mode:

1 **Connect the E6651A and the SS:** After applying AC power to the E6651A and initializing the program, connect the SS to the unit using an RF cable or antenna.

2 **Select BSE mode:** In the initial Measurement Window, select the **Test BS** menu item.

3 **Adjust Output Power:** The default output power of the E6651A is set to the minimum level (-124 dBm) to protect the subscriber station from damage.
It is necessary to adjust the output power to an appropriate level for the test subscriber station's specifications and the method of connection between the SS and the unit. To adjust the output power, press **Amplitude** on the Front Panel and enter the desired level using the **Knob** or **Numeric Keys**. When using the **Numeric Keys**, press **ENT** to confirm the input. When an antenna is used to connect the SS to the Test Set, the recommended output level is about 0 dBm. When a cable connection is used, the recommended range is between -50 and -40 dBm. Please contact the SS vendor for more detailed guidelines on a specific device.

4 **Adjust Input Attenuation Value:** The Test Set can attenuate the input signal to protect it's internal circuitry. It is necessary to adjust the input attenuation value to an appropriate level for the SS specifications and connection method. Please refer to Table 2 for recommended input attenuation value settings.

Table 2 Recommended Attenuation Range

Reference Level (dBm)	-30	-20	-10	0	10	20	30
Attenuation Setting (dB)	0	10	20	30	40	50	60
	5	15	25	35	45	55	65
	10	20	30	40	50	60	70
	15	25	35	45	55	65	75
	20	30	40	50	60	70	80
	25	35	45	55	65	75	85
	30	40	50	60	70	80	

5 **Set the Modulation:** The Test Set can modulate the output signal as described in the Mobile WiMAX specification. Press **MOD** on the Front Panel to modulate the output signal.

6 **Set the RF Modulation:** The Test Set can apply OFDMA. Press **RF** on the Front Panel to apply.

NOTE Both **MOD** and **RF** must be enabled to generate a WiMAX output signal.

7 **Begin Signal Transmission:** To begin transmitting an output signal, press **START** on the **Selection** Menu. Alternatively, press the **Pause** button on the Front Panel.

After completing these steps, the Test Set transmits Downlink Channel Descriptor (DCD) and Uplink Channel Descriptor (UCD) messages continuously to the subscriber station. The BSE Mode Setting Window, shown in Figure 11, is displayed.

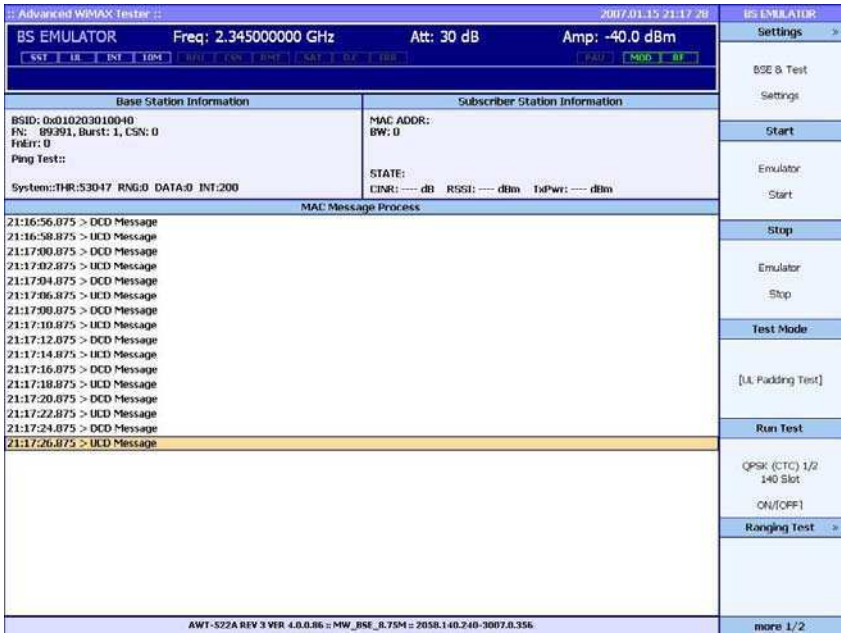


Figure 11 BSE Mode Setting Window

8 **Establishing Two Way Communications With the SS:** The SS must execute a Network Entry procedure in order to begin two way communications with the E6651A. Details of this procedure are provided in "Appendix A - Network Entry Procedure" on page 171. The Network

Entry procedure must be invoked from the SS using a Network Entry Application. Obtain this application from the SS vendor. A Network Entry Application is typically applied from a PC, PDA, or from the SS itself.

After successfully establishing two way communications, various control messages between the E6651A and the SS are displayed in the Action Window, as shown in Figure 12. If only DCD and UCD messages are present in the window, two way communications have not been established.

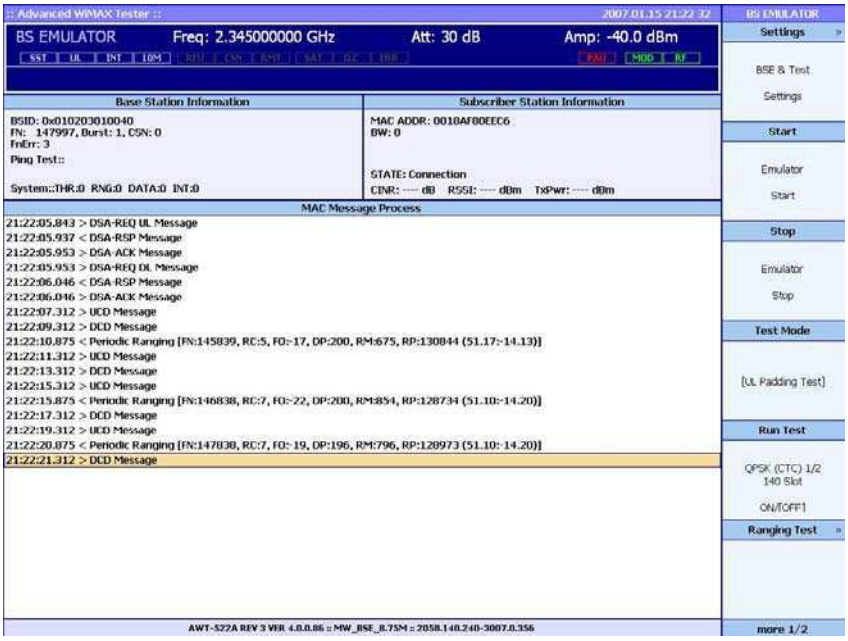


Figure 12 Establishing Two Way Communications Between the E6651A and the SS

Uplink Padding Test

Upon establishing two way communications, control and broadcast messages are transmitted between the Test Set and the SS, though no user traffic is present. The E6651A can test the transmitter performance of the SS by invoking the transmission of uplink data from the SS.

To perform the Uplink Padding Test:

1 **Select the Uplink Padding Test Mode:** After establishing two way communications, press **Test Mode** in the **BSE Mode Selection** Menu to select UL Padding Test. **Test Mode** is a toggle function.

2 **Run the Uplink Padding Test:** Press **Run Test** in the **Selection** Menu. The measurement screen is frozen and the product continuously demodulates the uplink signal.

3 **Perform Modulation Analysis:** While the Uplink Test is in progress, the Test Set can analyze the uplink signal's quality and performance. Press **Measure** on the Front Panel, followed by **Modulation** in the **Selection** Menu. The "**Uplink SYNC Detection**" message is displayed, indicating that uplink synchronization processing is in progress. When this process is complete, the Measurement Window in Figure 13 is displayed. Refer to "Signal Analyzer Mode" on page 57 for more details of analyzer functions.

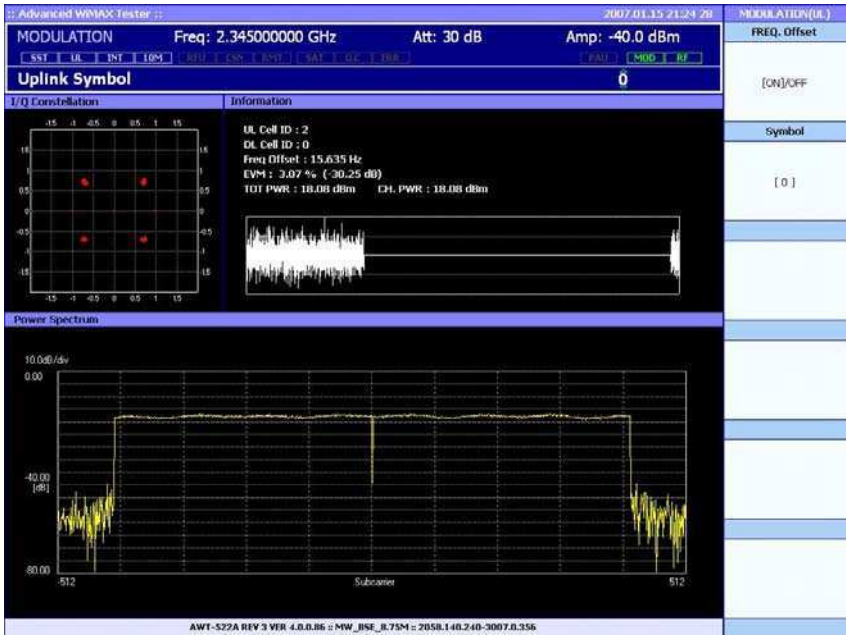


Figure 13 Uplink Test Modulation Analysis Window

4 **Apply a Frequency Offset:** The E6651A can measure the frequency offset of an uplink signal, and to analyze signal performance after a frequency offset has been applied.

To apply the frequency offset in the **Frequency** Menu (see also "Frequency Menu" on page 61), set **FREQ. Offset** to **ON** using the **Selection** Menu. To analyze a signal without frequency offset, ensure that **FREQ. Offset** is set to **OFF**.

5 **Select the Symbol Position:** The E6651A can measure the signal quality for each symbol within a frame. Press **Symbol** button in the **Selection** Menu and use the **Knob** or **Numeric Keys** to select the desired symbol position.

6 **Perform Spectrum Measurement:** During the Uplink Test, the E6651A can provide spectrum measurement for the uplink signal. To activate spectrum measurement, press **Measure** on the Front Panel, followed by **Spectrum** in the **Selection** Menu. The Measurement Window shown in Figure 14 is displayed. Refer to Chapter 4, "Signal Analyzer Mode" for more information about analysis operations. Note that spectrum analysis must be performed after modulation analysis.

Figure 14 Uplink Spectrum Measurement

Downlink UDP Test

Upon establishing two way communications, control and broadcast messages are transmitted between the E6651A and the SS, though no user traffic is present. The E6651A can test the receiver performance of the SS by transmitting user data in the downlink direction. To perform the Downlink UDP Test:

1 **Set the Downlink UDP Test Mode:** After establishing two way communications, press **Test Mode** in the **BSE Mode Selection** Menu to select **DL UDP Test**. **Test Mode** is a toggle type.

2 **Run the Downlink UDP Test:** Press **Run Test** in the **Selection** Menu. While Downlink UDP test is in progress, the counts of transmitted packets, received packets, and lost packets in Base Station Information window are being updated continuously with error rate value.

3 **Measure the Signal Quality:** In Downlink Test Mode, the E6651A transmits a downlink signal that is only available for measurement at the subscriber station. Contact the SS vendor for a testing tool that may be connected to the SS for downlink signal performance measurement.

Downlink Ping Test

Upon establishing two way communications, control and broadcast messages are transmitted between the E6651A and the SS, though no user traffic is present. The E6651A can test the receiver performance of the SS by transmitting user data in the downlink direction. To perform the Downlink Ping Test:

1 **Set the Downlink Ping Test Mode:** After establishing two way communications, press Test Mode in the **BSE Mode Selection** Menu to select DL Ping Test. Test Mode is toggle type.

2 **Run the Downlink Ping Test:** Press Run Test in the **Selection** Menu. While DL Ping Test is in progress, the counts of transmitted packets, received packets and lost packets in Base Station Information window are being updated continuously with error rate value.

3 **Measure the Signal Quality:** In Downlink Test Mode, the E6651A transmits a downlink signal that is only available for measurement at the subscriber station. Contact the SS vendor for a testing tool that may be connected to the SS for downlink signal performance measurement.

NOTE

The difference between DL UDP Test and DL Ping Test:

The purpose of DL UDP Test and DL Ping Test is to measure downlink performance of Mobile WiMAX subscriber station.

- In DL UDP Test, E6651A continuously transmits test packets to subscriber station. And subscriber station sends only acknowledgements to E6651A for the received packets.
- In DL Ping Test, E6651A continuously transmits test packets to subscriber station. And subscriber station sends back the received packets.

Therefore, in DL Ping Test, identical load is applied in downlink path and uplink path. But in DL UDP test, load is applied in downlink direction only.

CINR Test

Upon establishing two way communications, the E6651A can adjust the CINR value of the transmitted signal.

1 **Set the Downlink CINR Test Mode:** After establishing two way communications, press **Settings** in the **BSE Mode Selection** Menu to select **Parameter Settings**.

2 **Run the Downlink CINR Test:** 4 test modes are available: OFF, Preamble, Pilot and Hybrid.

- **OFF:** signal with no interference is transmitted.
- **Preamble:** interference is added in signal in Preamble portion to produce signal with designated CINR value.
- **Pilot:** interference is added in signal in Pilot portion to produce signal with designated CINR value.
- **Hybrid:** interference is added in signal in Preamble and pilot portions together to produce signal with designated CINR value.

3 **Adjust CINR value:** use the **Knob** or **Numeric Keys** to select the desired CINR value of transmitting signal.

4 **Measure the Signal Quality:** In Downlink Test Mode, the E6651A transmits a downlink signal that is only available for measurement at the subscriber station. Contact the SS vendor for a testing tool that may be connected to the SS for downlink signal performance measurement.

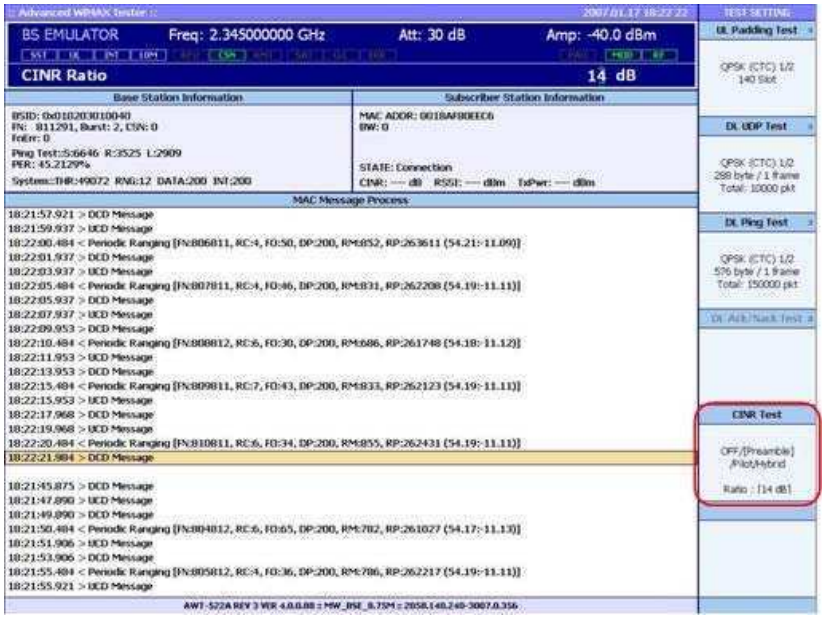


Figure 15 CINR test

Ranging Test

Upon establishing two way communications, the E6651A can adjust transmit power of the SS.

1 **Set the Ranging Test Mode:** Upon establishing two way communication path between E6651A and subscriber station, select **BS Emulator**, **Ranging Test**.

2 **Run the Ranging Test:** There are 3 sub menus in Ranging test. They are **Power UP(1dB)**, **Power Down(1dB)** and **RNG-RSP Power Offset**.

- **Power UP(1dB):** When selected, a control command to boost the output power of SS by 1dB is transmitted to subscriber station.
- **Power Down(1dB):** When selected, a control command to decrease the output power of SS by 1dB is transmitted to subscriber station.
- **RNG-RSP Power Offset:** When **ON** is selected, a control command to boost the output power of SS by 1dB is transmitted to subscriber station after receiving ranging request message.

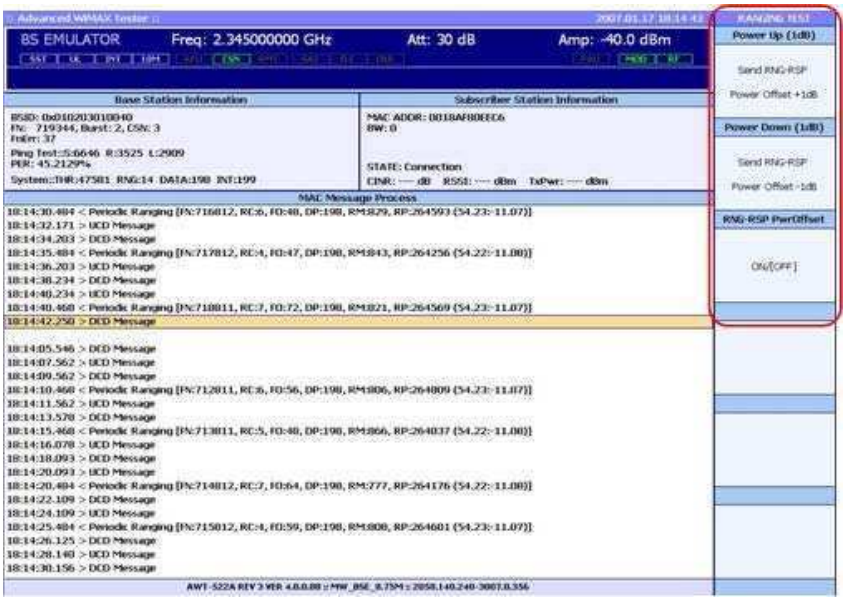


Figure 16 Ranging Test

PMC-RSP Test

Upon establishing two way communications, control and broadcast messages are transmitted between the E6651A and the SS. The E6651A can test the power control performance of the SS.

1 **Set the PMC-RSP Test Mode:** After establishing two way communications, press **PMC-RSP Test** in the **BSE Mode Selection** Menu.

2 **Set parameters:** There are 3 sub menus in PMC-RSP test. They are Power control mode, Start Frame and Power Adjust.

- **Power Control Mode:** Closed Loop power control and Open Loop power control options can be selected. This is a toggle function.
- **Start Frame:** This is to set the start frame for power control. Power control function is activated after the number of frames defined by the button.
- **Power Adjust:** This sets the amount of power adjustment.

3 **Run the PMC-RSP Test:** Pressing **Send PMC-RSP** sends a power control command to the subscriber station after Start Frame.

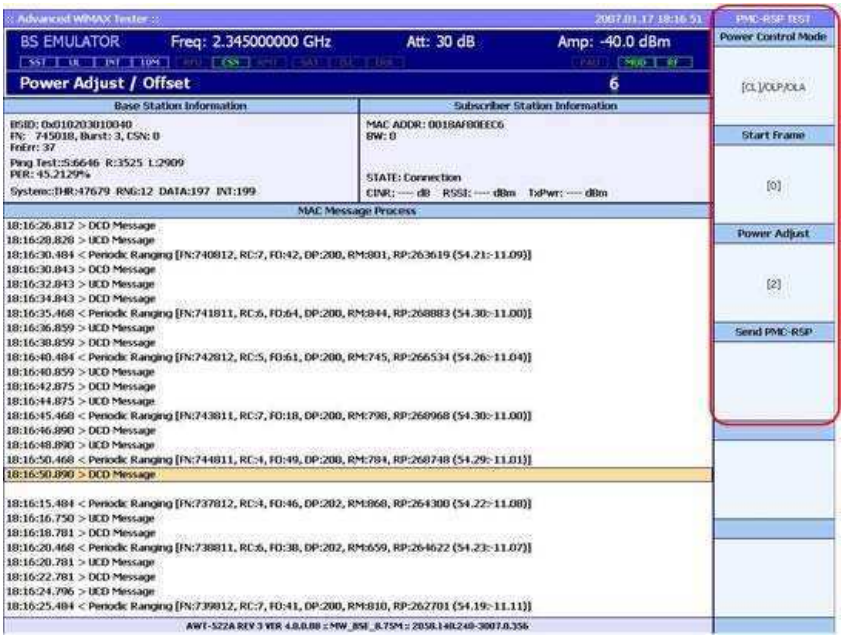


Figure 17 PMC-RSP test

Protocol Function Test

The Protocol Function Test enables the E6651A to test the MAC layer protocol of mobile WiMAX. Click **Protocol Function Test** at the bottom of **BS Emulator** menu as shown in Figure 18.

Figure 18 BS Emulator menu

You can use the Protocol Function Test to simulate Power Control and Handover functions.

Press **Protocol Function Test** to display the screen as shown in Figure 19 on page 44.

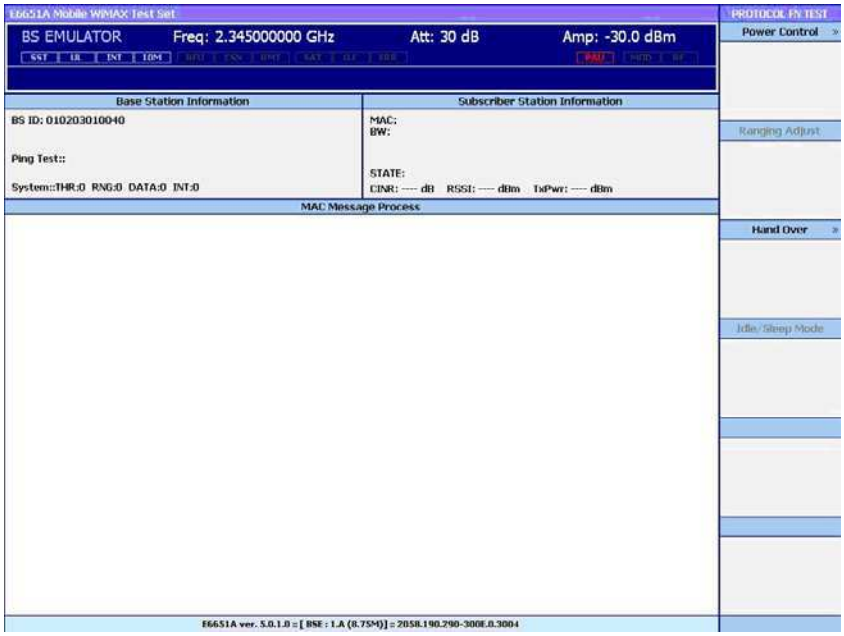


Figure 19 Protocol FN Test window

The following functions are available:

- **Power Control** - Test the SS Tx Power Control function.
- **Handover** - Test the scanning function to get Neighboring BS CINR information.

Power Control Test

The Power Control Test function implemented in the E6651A Test Set operates through MAC messages between the BS and SS. All the power control algorithms specified in IEEE802.16 are supported in the E6651A.

Power Control modes supported in E6651A

E6651A supports close loop and open loop mode which are the SS power control mode.

Power control parameters supported in E6651A

- Closed Loop - Adjust the Power Adjust value.
- Open Loop - Adjust the OffsetBSperSS and NI (UL noise and interference level IE) value.

The E6651A Power Control menu has three sub menus:

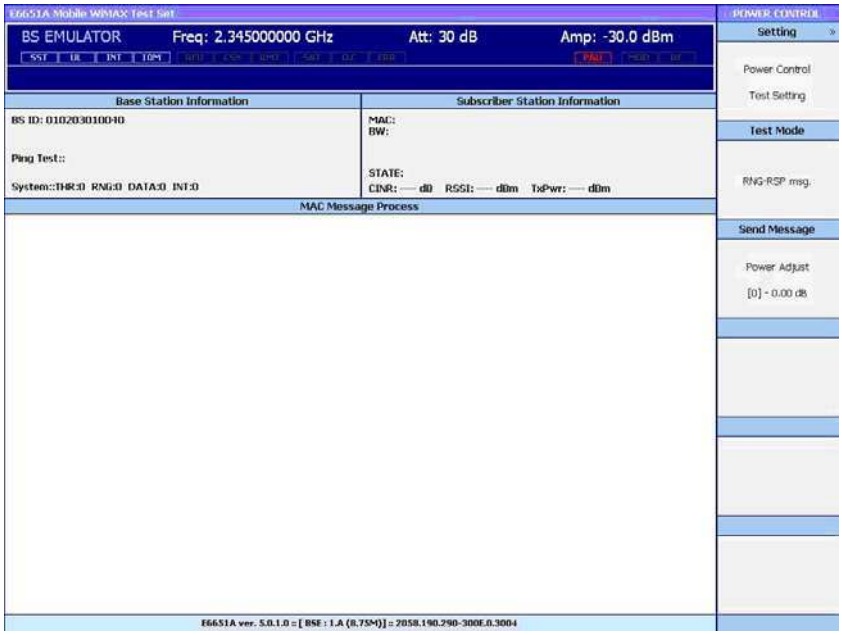


Figure 20 Power Control menu screen

- **Setting** - Use **Setting** to configure the power control parameters.
- **Test Mode** - Use **Test Mode** to configure the power control mode (algorithm).
- **Send Message** - Use **Send Message** to send the MAC message for the selected test mode to the SS.

To test the power control function of the SS, you can configure power control related parameters for the DUT using **Setting**. The power control mode is selected using **Test Mode**. Finally, select **Send Message** to send the selected message to SS after establishing network connection between the BS and SS.

Figure 21 Power Control Setting window 1/2

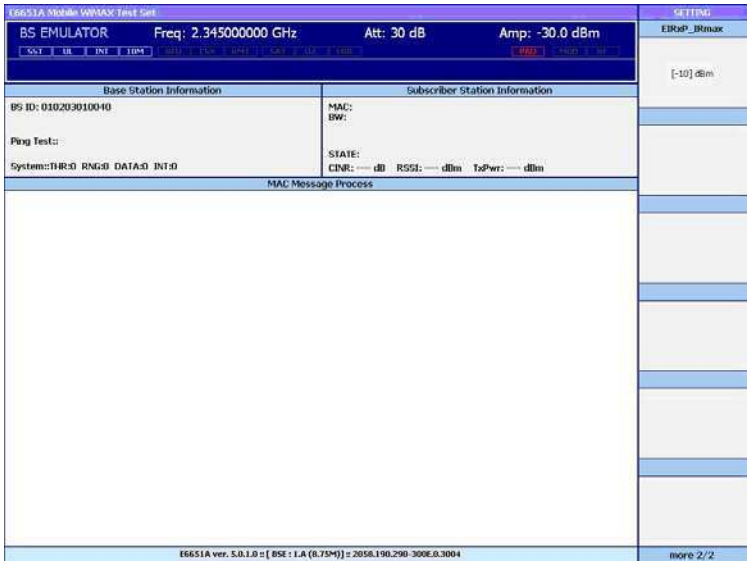


Figure 22 Power Control Setting window 2/2

Power Control Setting parameter

The available menu functions are as follows:

- **RNG-RSP msg**. - Adjust Power Adjust value of RNG-RSP message, one of Power Control methods of the E6651A.
- **Power Control IE** - Adjust Power Adjust value of Power Control IE message, one of Power Control methods of the E6651A.
- **FPC msg.** - Adjust Power Adjust value of FPC message, one of Power Control methods of the E6651A.
- **PMC-RSP msg.** - Adjust Power Adjust value of PMC-RSP message, one of Power Control methods of the E6651A. Or select power control mode.
- **NI IE** - Select “NI IE” value. “NI IE” is “UL noise and interference level IE” which is used in Open Loop Power Control and broadcast from the BS.
- **EIRP** - Select EIRP value. EIRP is BS's “Equivalent isotropic radiated power” and is basic information to determine the SS Tx Power.
- **EIRxP_IRmax** - Select EIRxP_IRmax value. EIRxP_IRmax is the BS maximum received power in Equivalent isotropic and is basic information to determine the SS Tx power.

Power Control Test Mode menu

E6651A provides four power control methods:

- **RNG-RSP MAC Message** - Controls the SS Tx power in 0.25 dB steps.
- **Power Control IE** - Controls the SS Tx power in 0.25 dB steps.
- **FPC MAC Message** - Controls the SS Tx power in 0.25 dB steps.
- **PMC_RSP MAC Message** - Controls the SS Tx power in 0.25 dB steps.

Hand-over Test function

The Hand-over Test function implemented in the E6651A operates through MAC messages between the BS and SS. All the power control algorithms specified in IEEE802.16 are supported in E6651A.

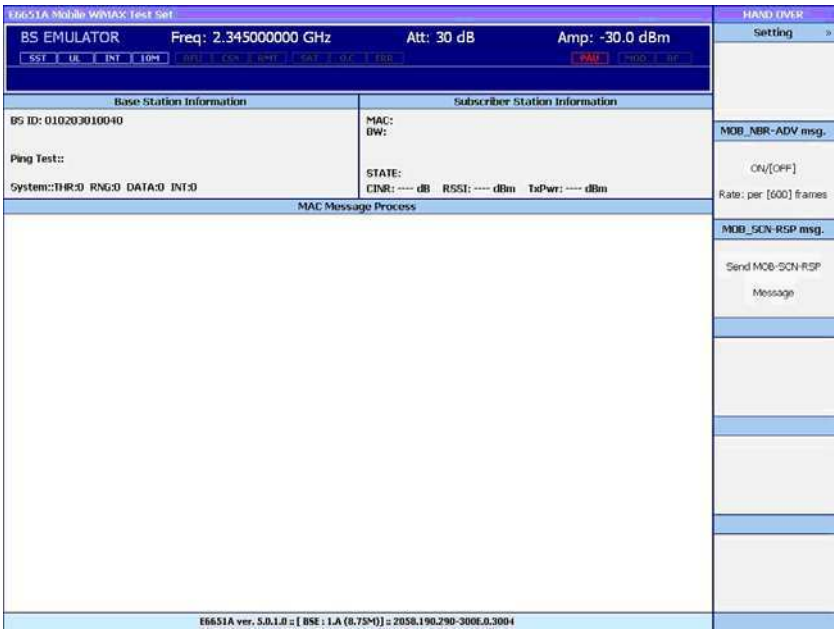


Figure 23 E6651A Handover Menu window

Major Hand-over functions supported in E6651A

The E6651A controls MOB_NBR-ADV and MOB_SCN-RSP which determine hand-over mode.

E6651A's Hand-over related major Parameters

- **MOB_NBR-ADV** -UCD CC (Configuration Change Count), DCD CC (Configuration Change Count), frequency and Permutation Base parameters can be adjusted.
- **MOB_SCN-RSP** - a response message specified in mobile WIMAX specification.

  Scan Duration = 0
  Report Mode = periodic report
  Report period = 50 frame
  Report metric: CINR

The E6651A Handover menu has three sub menus:

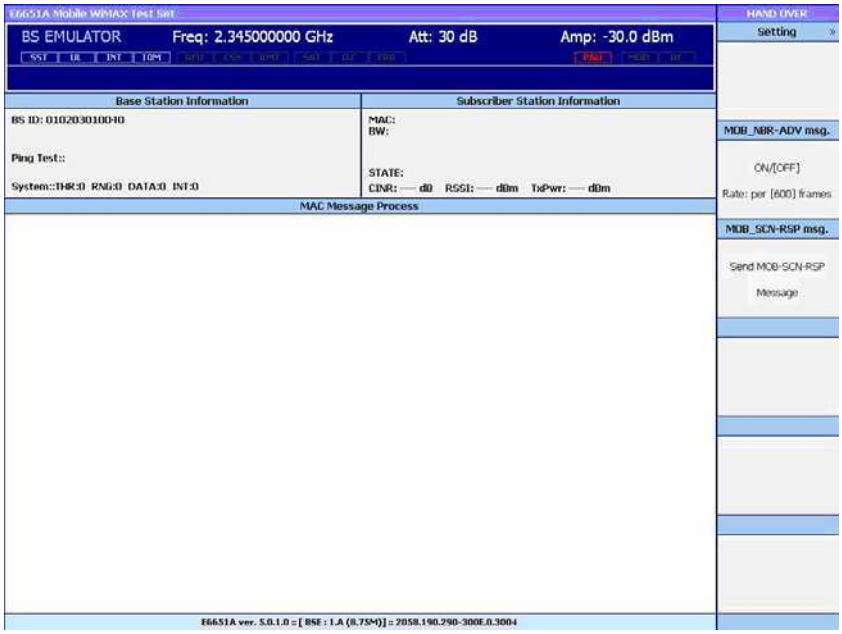


Figure 24 E6651A Handover Menu window

- **Setting:** Use **Setting** to configure the MOB_NBR-ADV message parameters.
- MOB_NBR-ADV - Use **MOB_NBR_ADV** to send the MOB_NBR-ADV message periodically.
- MOB_SCN-RSP - Use **MOB_SCN-RSP** to send the MOB_SCN-RSP message.

To test the Handover function of the SS configure hand-over related parameters for the DUT with the **Setting** function. Select **MOB_NBR_ADV** or **MOB_SCN-RSP** to send the selected message to the SS after establishing a network connection between the BS and SS.

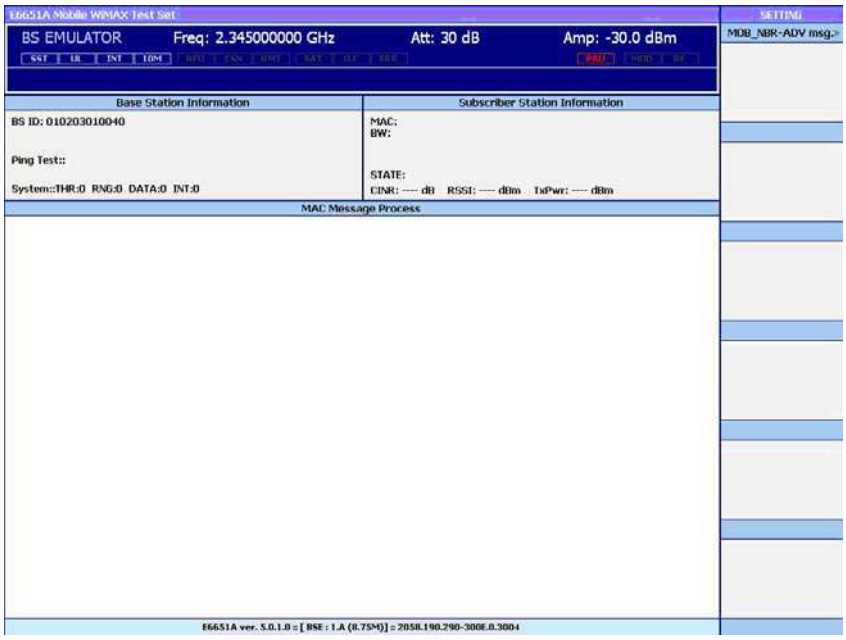


Figure 25 Handover Setting window 1/3

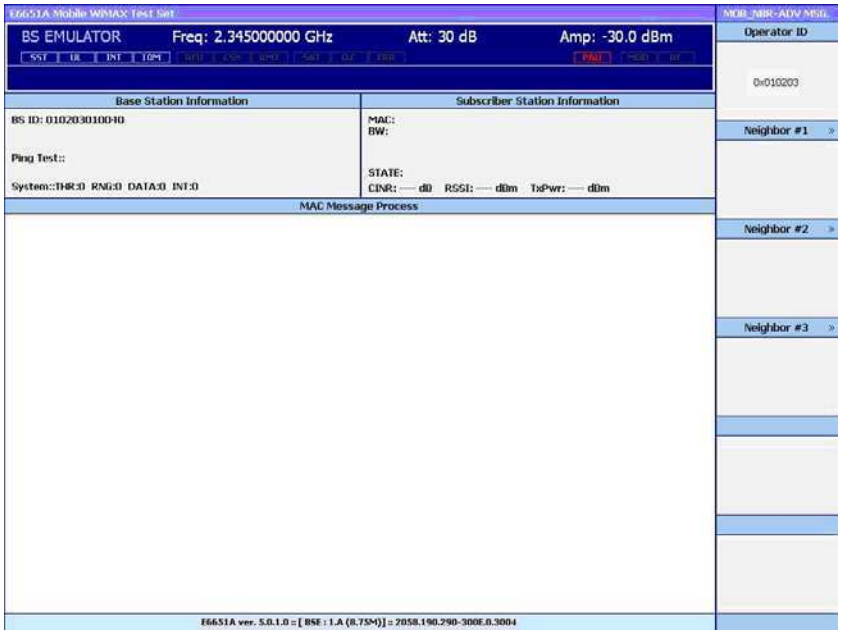


Figure 26 Setting window after selecting MOB_NBR-ADV msg. 2/3

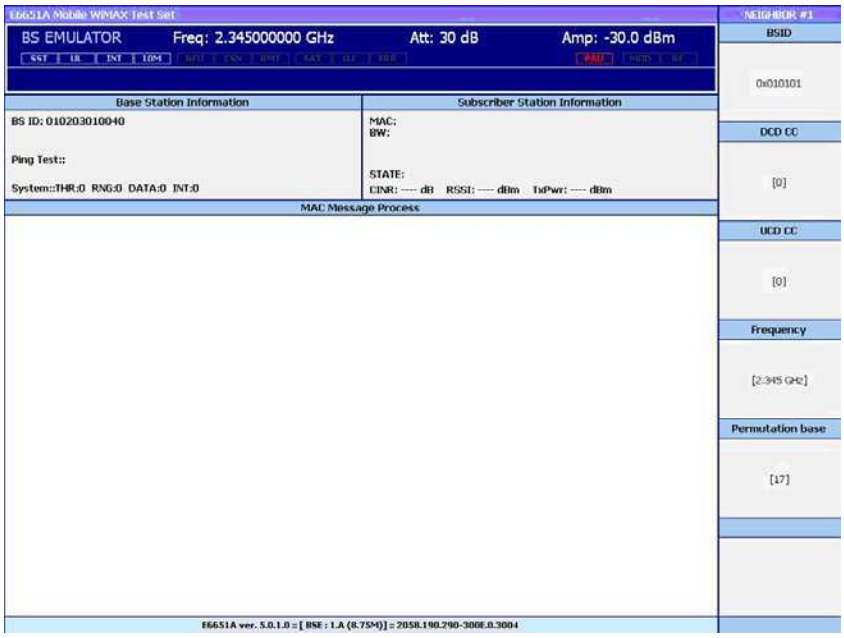


Figure 27 Setting window after selecting Neighbor. 3/3

Handover Setting Parameter

The available menu functions are as follows:

MOB_NBR-ADV msg - Configure parameters in MOB_NBR-ADV message, one of hand over test functions.

- **Operator ID** - Select Operator ID parameter. Shared by several E6651As as Common ID.
- **Neighbor #1** - Configuring Neighbor #1 information. (The same parameters can be configured for Neighbor #2 and #3.)
  - **BS ID** - Select Neighbor BS ID. Minimum 24 bits base station Id parameter information in DL-MAP message.
  - **DCD CC** - Select DCD CC parameter. DCD CC is “DCD Configuration Change Count” and SS can get information on whether neighbor BS's DCD parameters are changed or not. Upon DCD parameters change, the count number will be changed accordingly.
  - **UCD CC** - Select UCD CC parameter. UCD CC is “UCD Configuration Change Count” and SS can get information on whether neighbor BS's UCD parameters are changed or not. Upon UCD parameters change, the count number will be changed accordingly.

- **Frequency** - Select Frequency value. Provides center frequency information.
- **Permutation Base** - Select UL permutation base value. ID information of UL data region.

Setting Parameters

Setting General BSE configuration parameter

It is possible to adjust the settings for various parameters used in the Test mode operations. Press **Settings** in the **BSE Mode Selection** menu to display available parameters as shown in Figure 28. Select the parameters to be modified and adjust the parameter value.

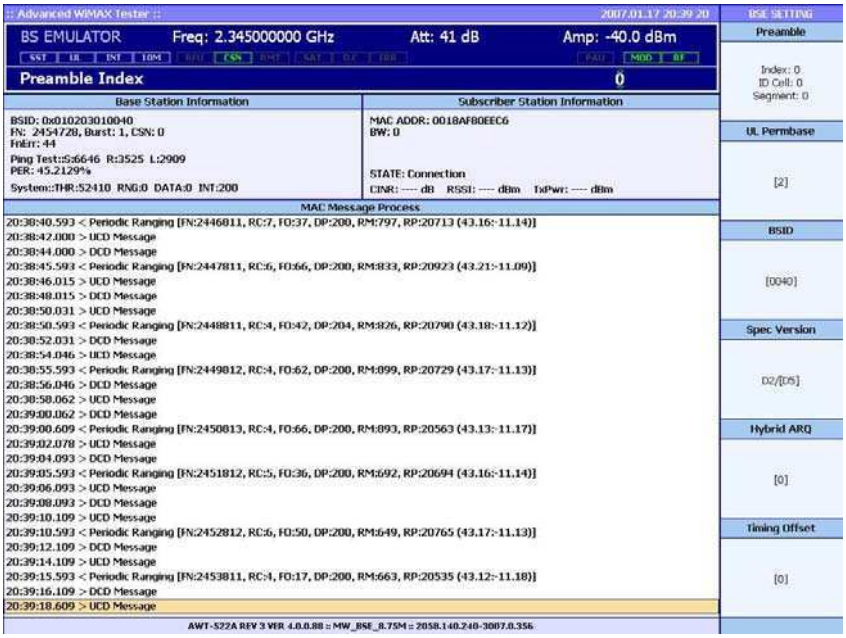


Figure 28 Setting General BSE configuration parameters

Preamble The **Cell ID** and **Segment ID** can be adjusted to simulate different base stations in the Downlink Test. Use the **Knob** to adjust the parameter values. The value of **Segment ID** and **Cell ID** is varied together. The range of **Segment ID** is 0 to 2. The range of **Cell ID** is 0 to 31. The range of **Preamble Index** is 0 to 113.

UL Permbase Select to change the Uplink Permutation base value.

BSID Use the **Knob** or **Numeric Keys** to adjust the Base Station ID to a unique value.

Hybrid ARQ the Hybrid ARQ function of subscriber station can be tested.

Frame Offset Use the **Knob** or **Numeric Keys** to adjust the frame offset of the downlink signal.

Timing Offset Use the **Knob** or **Numeric Keys** to adjust the timing offset of the uplink signal.

UL MAP Offset MCS scheme in uplink direction is applied after the number of frames specified in UL MAP Offset.

Setting Test Parameters

It is possible to adjust the settings for various parameters used in the Uplink and Downlink Test modes. Press the Parameter Settings button in the BSE Mode Selection menu to display available parameters as shown in Figure 29 and Figure 30. Select the parameters to be modified and adjust the parameter value.

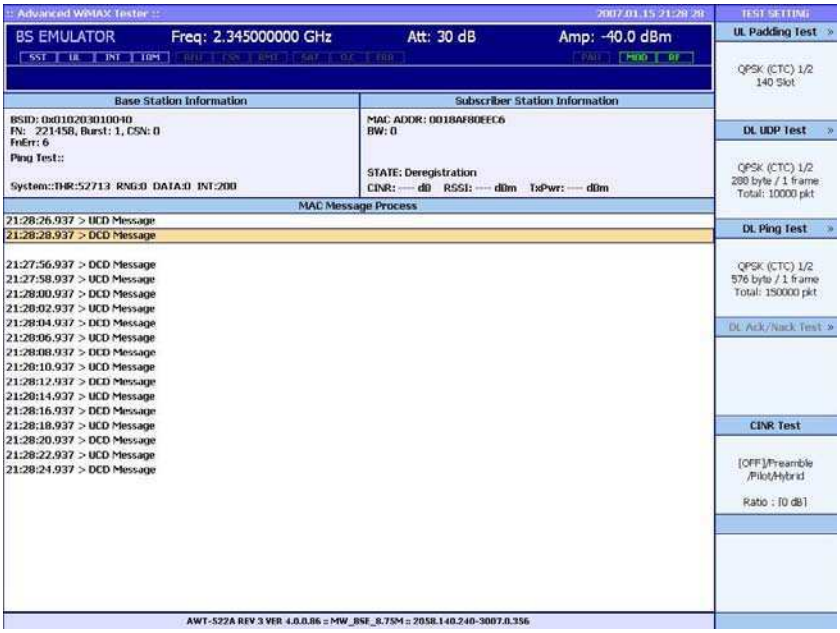


Figure 29 Adjustable Parameters (Screen 1 of 2)

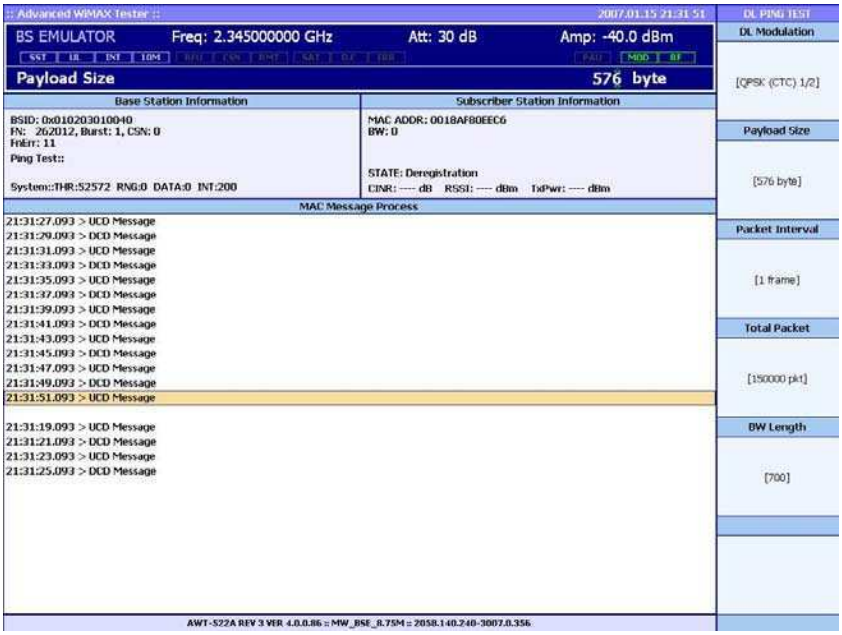


Figure 30 Adjustable Parameters (Screen 2 of 2)

UL Modulation Select to toggle between available uplink modulation schemes. Selected mode will be applied to UL Padding Test.

DL Modulation Select to toggle between available downlink modulation schemes. Selected mode will be applied to DL UDP Test and DL Ping Test.

Payload size use this to determine the payload size of packets in DL UDP Test and DL Ping Test.

Packet Interval use this to designate the number of frames to convey 1 data packet in downlink direction.

Total Packet use this to designate total number of test packets for DL Ping Test and DL UDP Test.

BW Length use this to designate the bandwidth value in uplink direction for DL Ping Test. The unit is in bytes

UL Slot use this to designate the number of data slots in one frame for UL Padding Test.

Terminating BSE Mode Operation

To terminate BSE Mode and stop signal transmission, press **STOP** in the **BSE Mode Selection** Menu. The Test Set stops transmitting signals in downlink and measurement window is frozen.

4 Signal Analyzer Mode

In Signal Analyzer (SA) mode, the E6651A Test Set may be used to analyze Mobile WiMAX uplink signals using modulation, spectrum and flatness analysis. Modulation Analysis mode displays the OFDM signal in both frequency and time domain form. The Spectrum Analysis functionality, implemented using a Fast Fourier Transform (FFT) algorithm, displays the measured WiMAX signal in frequency domain graph form. This mode is used to analyze the center frequency, bandwidth and amplitude of the broadband signal. Flatness analysis allows for the comparison of power intensity among spectral components.

Measurement Preparation

SA Mode Setup Procedure

1 **Connect the Test Set With the Device Under Test (DUT):** After applying AC power to the E6651A and initializing the program, connect the DUT to the Test Set using either an RF cable or antenna.

2 **Select SA Mode:** In the initial **Measurement Window**, select the **Signal Analyzer** menu item. The DUT is now transmitting a signal. Figure 31 shows the Initial Measurement Screen for SA mode.

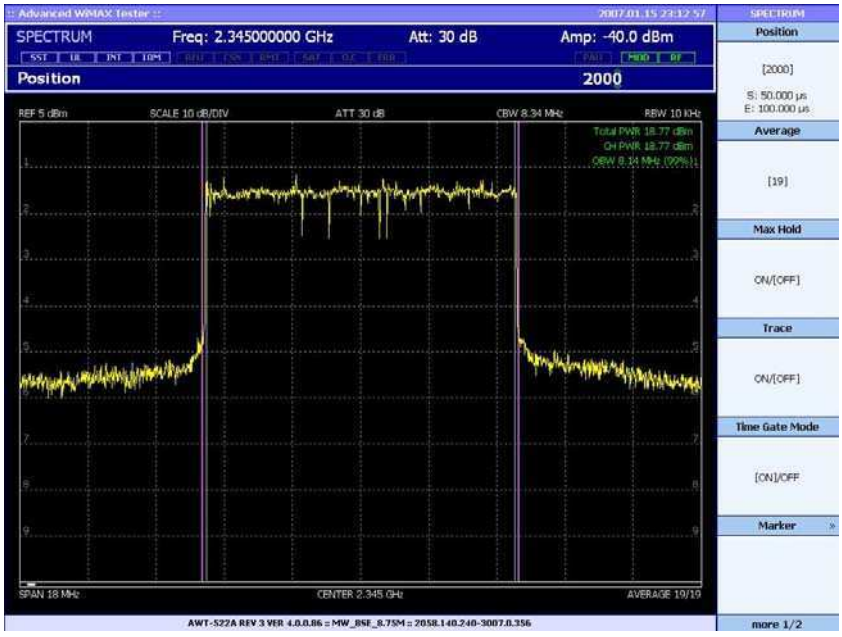


Figure 31 Initial Measurement Screen for SA Mode

System Settings

Press **System** to display the Mode Setup (System) Menu. This menu includes options for Mobile WiMAX, Machine ID, Test Mode, Clock Source, Connect E6655A, Trigger, Update.

Figure 32 System Settings (Screen 1 of 2)

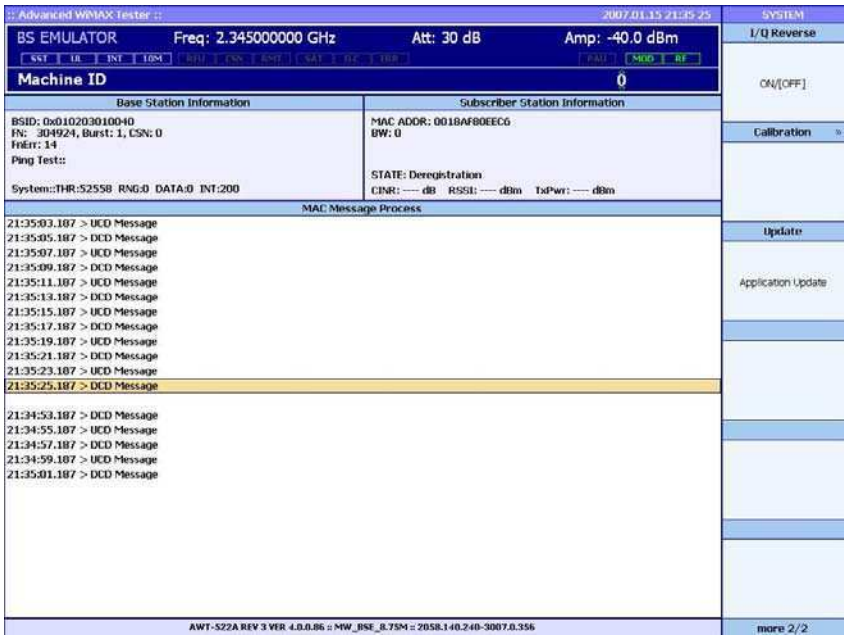


Figure 33 System Settings (Screen 2 of 2)

Machine ID Use this option to adjust the unique Test Set ID. This is used when a PC running the Agilent E6655A Lab Application is connected to Test Set for external IP connection.

Test Mode Selection Use this option to select the mode as **SA**, **SG** or **SA + SG** (BSE) Mode. When **BSE** mode is selected from the **Mode Menu**, **SA + SG** mode is automatically set.

Clock Source Use this option to select either the Internal Clock (**INT**) or the External Clock (**EXT**) as the reference clock.

Connect E6655A Use this option to connect or disconnect the E6655A Lab Application. The E6655A is an optional PC Application used for quality and performance measurement of application services.

Trigger Display the Trigger Menu to select either **Single Mode** or **Continuous Mode**. In Single Mode, a single set of input data are captured and analyzed. In this mode, use **Pause** and **Restart** options on the **Trigger** Menu or press **Pause** on the Front Panel to control data capture and analysis. In Continuous Mode, input data is continuously captured and analyzed until the program is manually paused.

I/Q Reverse Use this option to swap the I and Q output.

Update The Test Set provides ease of use software upgrade. Connect the memory device which contains upgrade software to USB port. Press **Update** and the Test Set downloads the software from the memory device and the upgrade is executed automatically.

Frequency Menu

Press **Frequency** on the Front Panel to access the **Frequency** Menu. The **Frequency** Menu is used to adjust frequency-related values including Center **Frequency**, Frequency **Offset**, Resolution Bandwidth (**RBW**) and Channel Bandwidth (**CBW**), **SPAN** using the **Knob** or **Numeric Keys**.

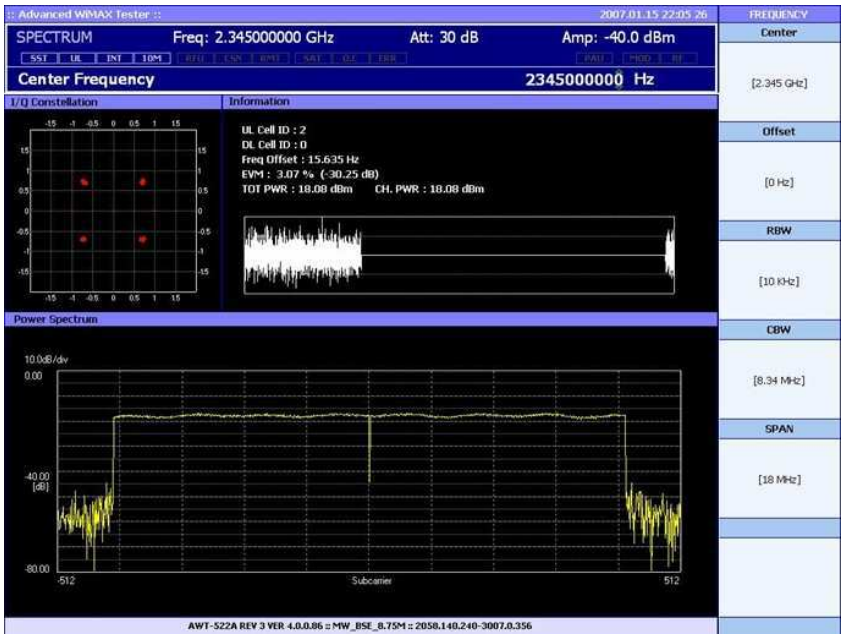


Figure 34 Frequency Menu

Center Frequency Select to adjust the Center Frequency using the **Numeric Keys** or the **Knob**.

Frequency Offset Select to adjust the Frequency Offset. Frequency Offset can be enabled in the Modulation Accuracy Analysis Window, as described in "Modulation Analysis" on page 64.

RBW Select to adjust the Resolution Bandwidth. Resolution Bandwidth is used to control the sampling frequency within the displayed range.

CBW Select to adjust the Channel Bandwidth. Channel Bandwidth refers to the bandwidth containing 99% of the total input power for the signal.

SPAN Use to adjust the frequency range shown on the display screen.

Amplitude Menu

The Amplitude Menu is used to adjust values related to input power including the Amplitude, Attenuation, Reference Level, Scale and Amplitude Offset. Access the Amplitude Menu by pressing **Amplitude** on the Front Panel.

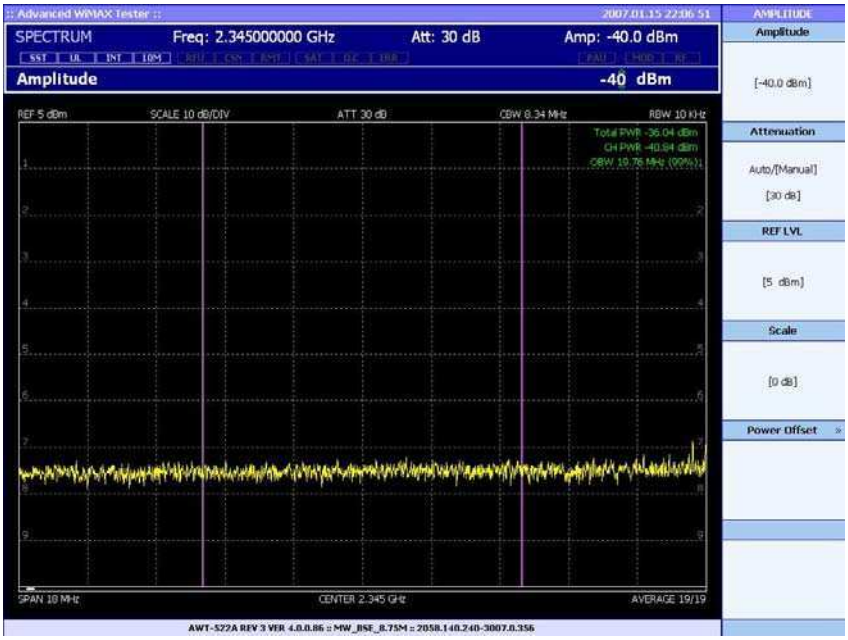


Figure 35 Amplitude Menu

Amplitude Use to adjust the output signal power.

Attenuation The input attenuation is automatically adjusted based on the reference level to protect the first stage mixer and ensure linearity. Select **Attenuation** to manually adjust the input attenuation.

Reference Level Use to set the power level displayed at the top of the Spectrum Analysis Screen.

Scale Use to adjust the power level scale on the Spectrum Analysis Screen.

InPower Offset and OutPower Offset Use to adjust the offset power for the measured power level. This function is used to compensate for cable loss and other discrepancies between the Test Set and the Device Under Test.

Measurement Menu

Following completion of the subscriber station's network entrance procedure, the Uplink Padding test can be performed. The Uplink signal analysis function is provided when the Uplink Padding Test is in progress.

Uplink Measurement Menu

In SA mode, press **Measure** on the Front Panel to display the menu shown in Figure 36. This mode is used to measure and analyze the quality of uplink signals transmitted by a Mobile WiMAX subscriber station or repeater. This section lists the options available in the Uplink Measurement Menu.

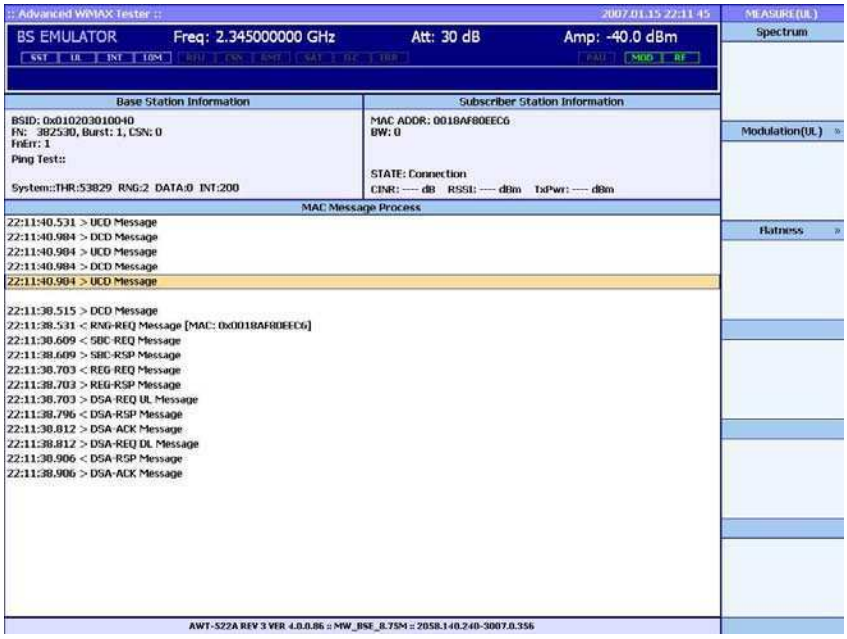


Figure 36 SA Mode Uplink Measurement Window

Spectrum Use to display the Spectrum Measurement Window. "Spectrum Measurement Window" on page 66 describes this window in detail.

Modulation Use to display the Mobile WiMAX Modulation Accuracy Analysis Window. "Mobile WiMAX Modulation Accuracy Analysis Window" on page 64 describes this window in detail.

Flatness Use this button to display the Mobile WiMAX Flatness Analysis Window. "Flatness Analysis" on page 71 describes this window in detail.

Modulation Analysis

Modulation Analysis mode of the E6651A Test Set displays the Mobile WiMAX OFDM signal in both frequency and time domain form. This is typically useful in the development of Mobile WiMAX subscriber stations and repeaters. A detailed description of these functions are provided here.

Mobile WiMAX Modulation Accuracy Analysis Window

The Modulation Accuracy Analysis Window displays the Mobile WiMAX OFDM signal in both time and frequency domain. The frequency domain graph provides information on subcarriers including the number of subcarriers, subcarrier usage, and the relative powers of subcarriers. The time domain graph shows the length of an OFDM signal and relative power of different parts of the signal. The I/Q constellation graph shows the constellation for the currently selected symbol. Access this window from the **Uplink Measurement** Menu.

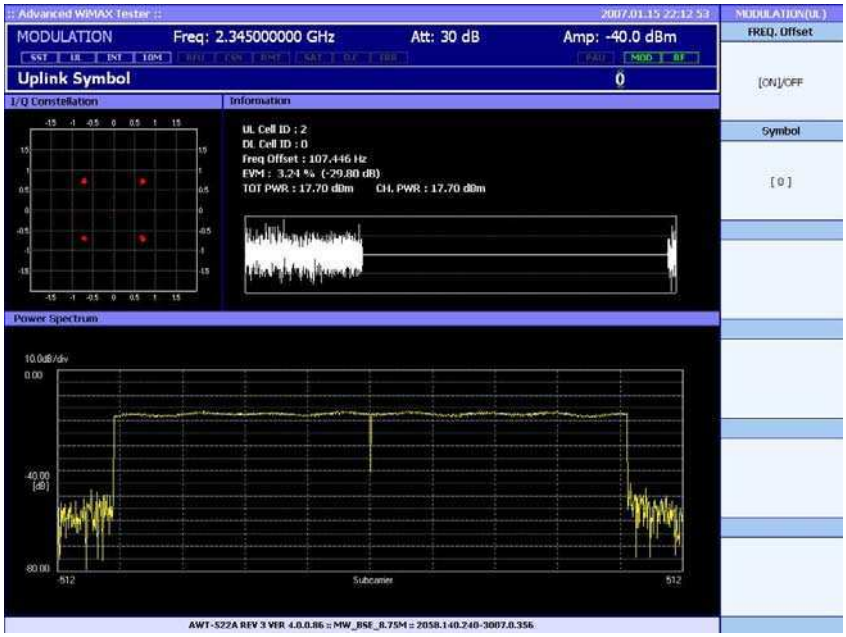


Figure 37 Modulation Accuracy Analysis Window

Three sub-windows are provided in the Modulation Accuracy Analysis Window:

I/Q Constellation This displays the constellation of the currently selected symbol.

Information The Information Window displays signal information and a time-domain waveform of the signal. The signal information includes the detected preamble's cell ID, segment ID, measured frequency offset, and the preamble's EVM. Additionally, for a downlink symbol with Symbol Index = 0, the pilot EVM average value is given. Otherwise, the symbol data EVM average is provided. The waveform graph in the Information Window provides a time domain view of the input data used for analysis. This view displays data for a single frame in the time domain providing the length of the frame and the relative power of different components of the frame.

Power Spectrum This window displays a frequency domain subcarrier spectrum graph of the currently selected symbol. This view can be used to display the number of subcarriers, subcarrier usage, and the relative powers of the subcarriers.

Two selections are available from the **Modulation** Menu:

FREQ. Offset Use to apply the offset specified in the **Frequency** menu to the measured carrier's frequency.

Symbol Use to select the specific symbol index for detailed analysis. For downlink analysis, the symbol index range is 0-26 with 0 being the preamble. For uplink analysis, the symbol index range is 0-11.

NOTE Note that Position 0 in the Processing Symbol Index corresponds to Position 3 in the Original Symbol Index.

Spectrum Analysis

Spectrum Analysis mode displays the measured WiMAX signal in the frequency domain. This mode is used to analyze the center frequency, bandwidth and amplitude of the broadband signal.

Spectrum Measurement Window

This section describes the menu options available from the Spectrum Measurement Window, the primary screen used for Spectrum Analysis. Access this window through the **Uplink Measurement** Menu.

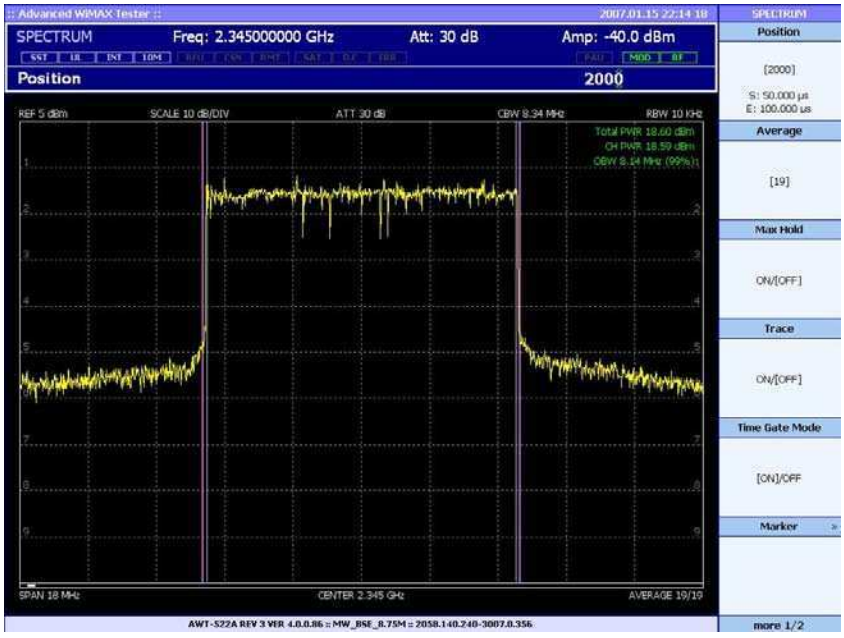


Figure 38 Spectrum Measurement Menu (Screen 1 of 2)

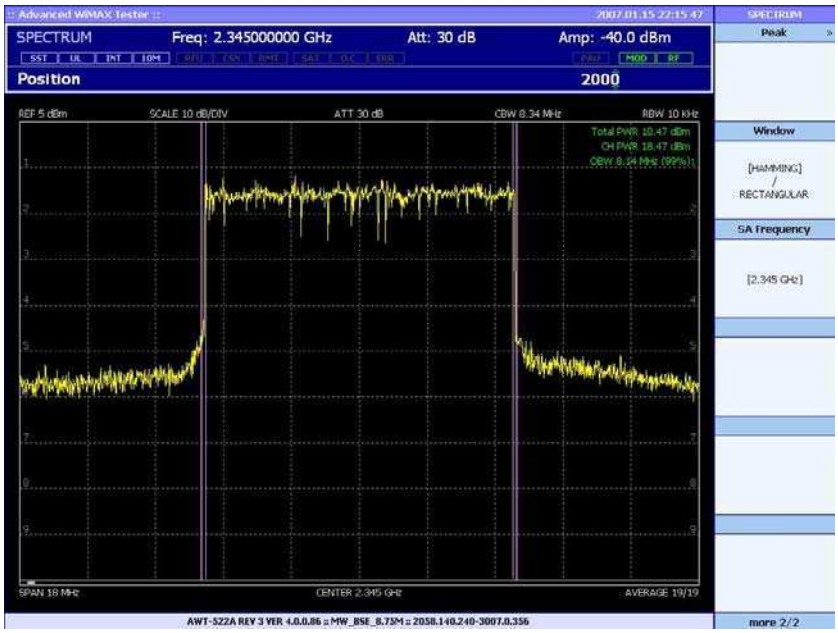


Figure 39 Spectrum Measurement Menu (Screen 2 of 2)

FFT Start Position The FFT Start Position represents the time index at which the FFT input begins. The index value is a sample of the mobile WiMAX 5 ms frame. With the Test Set sampling frequency of 40 MHz, the index values can range from 1-200,000. The starting time and ending time of the sample are calculated and displayed as “S:” and “E:” respectively.

Average This sets the number of consecutive measurement results averaged to produce the spectrum display. The available range is 0-100 frames.

Max Hold Set **Max Hold** to **ON** to display the frequency component with the maximum signal amplitude.

Trace Set **Trace** to **ON** to trace up to five signal frequency components.

Time Gate Mode Disable **Time Gate Mode** to determine the average spectrum information of a single Mobile WiMAX frame without the need for frame synchronization or knowledge of the exact starting point of the frame.

Marker Use to display the **Marker** Menu. The Marker functions can only be used for an RF input source. A detailed description of the **Marker** Menu is provided in “Marker Menu" on page 69.

Peak Use to display the **Peak** Menu. A detailed description of the **Peak** Menu is provided in “Peak Menu" on page 70.

Window Use to select either **Hamming** or **Rectangular** Windowing for frequency domain analysis.

SA Frequency Use to test the center frequency of the subscriber station for RCT testing.

Marker Menu

The **Marker** Menu is used to display the absolute value of the spectrum power at a particular frequency as well as the difference in power between two frequencies. Access the **Marker** Menu from the **Spectrum** Menu.

Figure 40 Marker Menu

Add MKR Select to display the absolute value of the spectrum power at a selected frequency.

Delta Select to add a second marker. Use this to determine the relative value between the two selected frequencies.

Select Use to select a marker to be edited or deleted.

Edit Use to change the frequency of the selected marker.

Delete Use to delete the selected marker.

Reset Use to delete all of the markers.

Peak Menu

The **Peak** Menu is used to adjust the display based on the strongest power spectrum measurements. Access the **Peak** Menu from the **Spectrum** Menu.

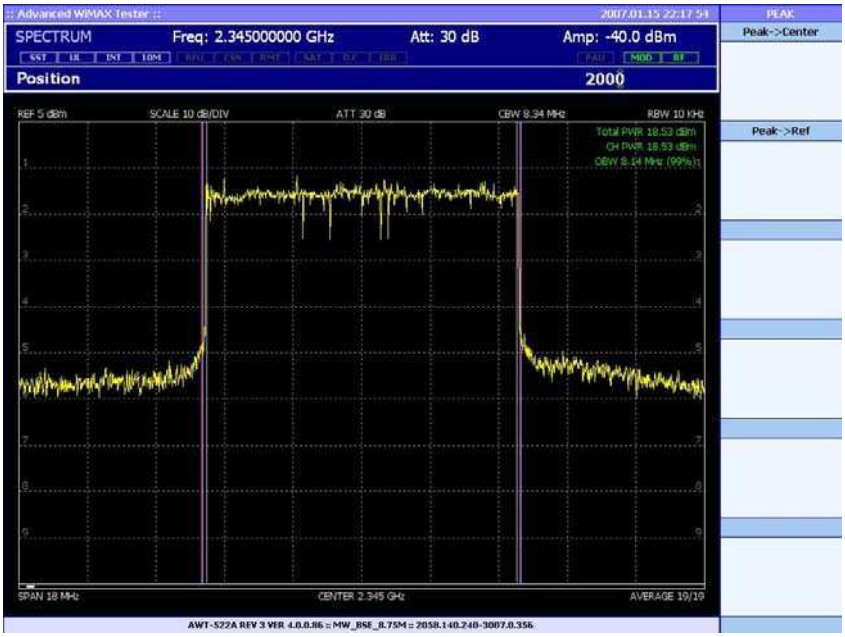


Figure 41 Peak Menu

Peak → Center Select to change the center frequency to the strongest frequency component.

Peak→ Reference Select to change the reference value to the highest measured amplitude level.

Flatness Analysis

Flatness Analysis mode of the Test Set allows for the measurement of the difference in power intensity among spectral components. Access this window from the **Uplink Measurement** Menu.

Figure 42 Flatness Analysis Window

Three sub-windows are provided in the **Flatness** Analysis Window:

I/Q Constellation This window displays a constellation graph of the currently selected subcarrier. Select the subcarrier using the **SubCarrier** menu option.

Subcarrier Power Table This window displays the power value for each subcarrier. Use the **SubCarrier** menu option to select the value of interest for display.

Subcarrier Group Average Power This window displays the difference between the average power of a group of subcarriers and the total average power.

One selection is available from the **Flatness** Menu:

SubCarrier Use to select the subcarrier value of interest to be highlighted in the Measurement Window.

5 Signal Generator Mode

In Signal Generator (SG) mode, the E6651A Test Set generates standard Mobile WiMAX downlink signals.

Signal Generator Mode

SG Mode Setup Procedure

To provide a standard Mobile WiMAX signal to a Device Under Test (DUT):

Connect the E6651A with the Device Under Test (DUT)

After applying AC power to the Test Set and initializing the program, connect the DUT to the Test Set using either an RF cable or antenna.

Select SG Mode

In the initial **Measurement** Window, select the **Signal Generator** menu item. The Test Set is not transmitting a signal at this point. Figure 43 shows the initial display window for SG mode.

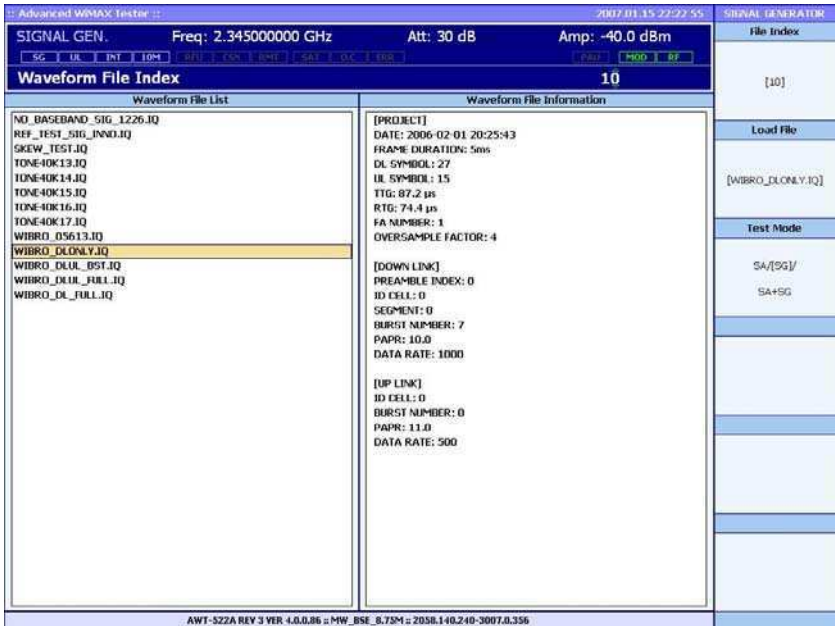


Figure 43 Signal Generator Window

The menu items available in SG mode are:

File Index Use to select one of the signal files in the **Waveform File List**. Detailed information for the selected signal is displayed in the Waveform File Information area. Adjust the **File Index** value using the **Knob** or **Numeric Keys**.

Load File Select this to continuously transmit the selected signal. To cease signal transmission, press the **MOD** and **RF** Front Panel buttons to turn off Modulation and RF output.

Test Mode Select **SG** when using Signal Generator mode.

6 Remote Interface (API) Programming Command Reference

The Remote Programming interface for the E6651A WiMAX Test Set takes the form of a Dynamic Link Library (DLL) or Application Programming Interface (API). This chapter shows you the basic steps required to use this DLL and lists the E6651A Command Set.

Creating a Test Program

This section shows you the 6 basic steps involved in making a test program using the Agilent E6651A Test Set API (or DLL - Dynamic Link Library).

1. Copy and Reference the Dynamic Link Library (DLL)
2. Install the IPX LAN protocol
3. Initialize and Start the E6651A remote interface
4. Configure the Measurement Parameters
5. Perform the Test
6. Get the Results
7. Close the Session

Step 1 “Reference”, Step 2 “Create” and Step 6 “Close” must be performed if the PC or the E6651A Test Set is power cycled. Steps 3 to 5 can be repeated as many times as required for a full test sequence of measurements to cover the chosen test plan for the WiMAX subscriber station.

It is advisable during software test development / debugging to catch any potential exceptions due to potential errors in the programming or setup. For example the use of Try / Catch Blocks in Microsoft Visual Studio .NET, allows the program to catch any errors without the program aborting.

Step 1 - Copy and Reference the Dynamic Link Library (DLL)

First copy the DLL and header files from the CD-ROM to a suitable location on your development computer. For development programming environments such as the Microsoft Visual Studio Integrated Development Environment (IDE), the programming language needs to first reference the E6651A DLL. How this is done varies between development environments.

Required Files:

- E6651_API.dll
- E6651_API.h

Reference the file: **E6651_API.DLL**

Step 2 - Install the IPX LAN protocol

Whilst TCP/IP is the most commonly used internet protocol, the IPX protocol is used for communication between the E6651A Test Set and your computer. Typically a computer does not have this protocol installed by default, but it is part of the Windows operating system and can be added using Windows XP system tools.

Addition of the IPX protocol does not interfere with normal communications using the default TCP/IP protocol.

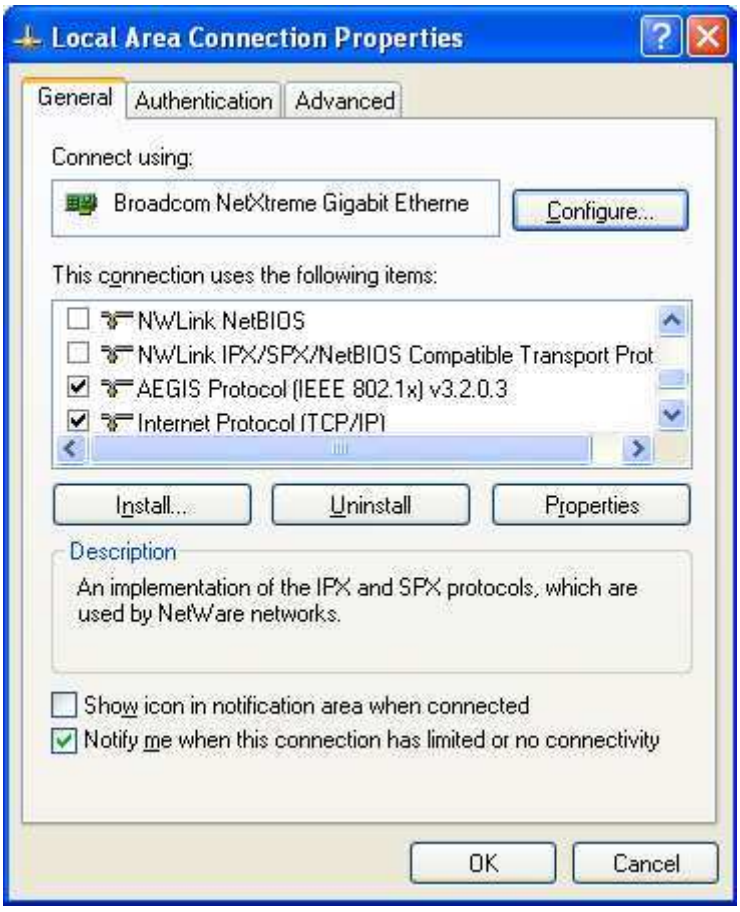
Proceed as follows:

1 Using the Windows XP desktop, click **start** > **Control Panel**.

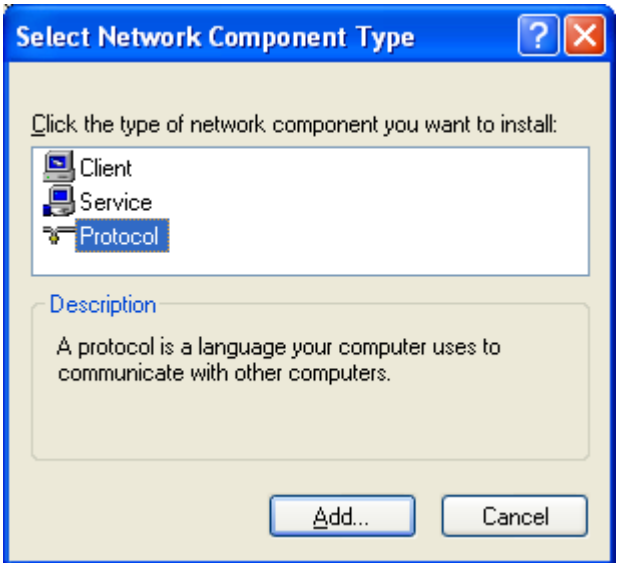
2 Select **Network Connections**.

3 In the Network Connections Window, right click on the **Local Area Connection** icon and select **Properties**. The 'LAN Connection Properties' Window is displayed.

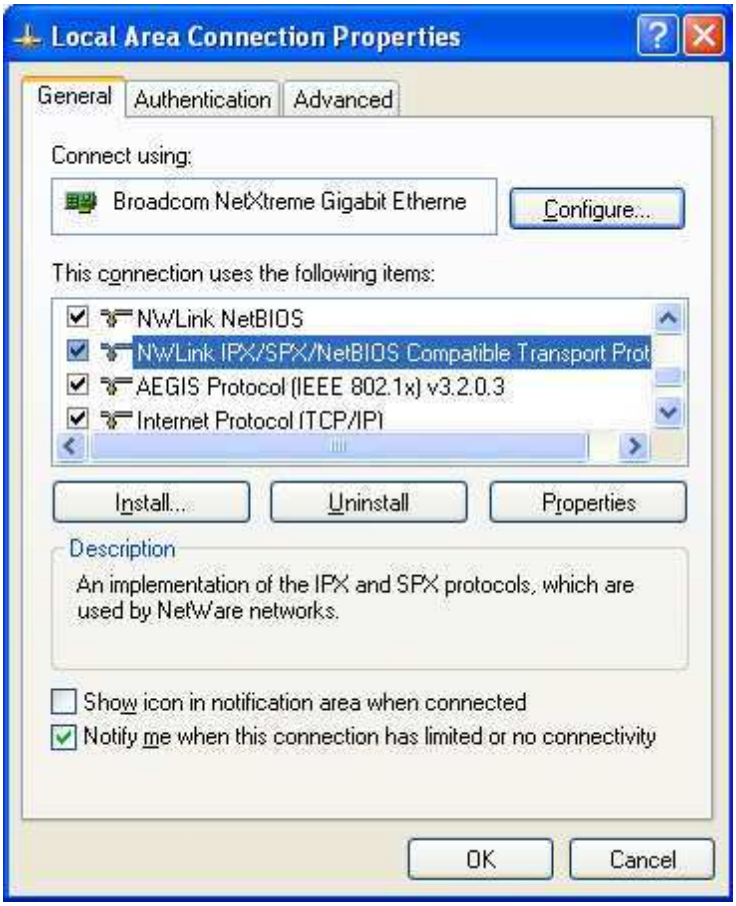
4 Click the **General** tab.



5 Click **Install...** and choose **Protocol.** Select **Add...**.



6 Scroll to find and select the **NWLink IPX/SPX/NetBIOS** checkbox.



7 Click **OK** to close the 'Local Area Connection Properties' window.

Installation of the IPX protocol is now complete.

Step 3 - Initialize the E6651A Interface

Each program begins with the initialization of the E6651 interface. You must also ensure the IPX LAN protocol is installed on your development computer. The following steps are required:

- Confirm the IPX LAN Protocol is installed on any computer being connected to the E6651A TestSet.
- Initialize the Interface using the `E6651_Init()` command.

- Start the Remote Interface operation using the `E6651_RemoteStart` command.
- Check the return values for success.

Example code: Initialize the Interface(C# syntax)

```
// Initialize and start the E6651 interface

BYTE mid = 0;

E6651_Init( );

E6651_remoteStart(mid);
```

Step 4 - Configure the Measurement Parameters

§

Step 5 - Perform the Test

§

Step 6 - Get the Measurement Results

§

Step 7 - Close the Session

End the program and release the test set using the E6651_close() function. As well as releasing memory space used by the E6651 API, this also releases the handle on the VISA resource allowing communication with the test set through other means.

Example code: Closing the session (C# syntax)

```
// Close the session

E6651_Close();
```

Command Reference

E6651_Init

Declaration `int E6651_Init(void);`

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description API command to initialize Remote API Library.

E6651_Close

Declaration `int E6651_Close(void);`

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description API command to close Remote API Library.

E6651_RemoteStart

Declaration `int E6651_RemoteStart(BYTE MID);`

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description API command to start the remote interface.

E6651_GetDLLVersion

Declaration `int E6651_GetDllVersion(char*);`

Parameter Version [out]: The version information of DLL

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Acquire version information of Remote API DLL currently being used.

E6651_SetTurnOffOption

Declaration `Int E6651_SetTurnOffOption(BYTE MID, Int Value);`

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Turn off option value to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set E6651(MID)'s power control (shut down) option.
0: Rebooting
1: Shut down
2: Exit Application

System information functions

E6651_GetSystemModel

Declaration `int E6651_GetSystemModel(BYTE MID, char* Model);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Model [out]:** E6651' s model information.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve model name (String value) of E6651 which is being controlled remotely.

E6651_GetSystemVersion

Declaration `int E6651_GetSystemVersion(BYTE MID, char* Version);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Version [out]:** The Current version information of E6651.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve version information (String value) of E6651 which is being controlled remotely.

E6651_GetSerialNumber

Declaration
```
int E6651_GetSerialNumber(BYTE MID, Char *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Serial Number is written in flash memory.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Serial Number is written in flash memory.

General functions

E6651_GetWorkMode

Declaration `int E6651_GetWorkMode(BYTE MID, int* Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** The current Work Mode value of E6651.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Work Mode value of E6651 which is being controlled remotely.
0: E6651
1: SA
2: SG

E6651_SetWorkMode

Declaration `int E6651_SetWorkMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Work Mode value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Modify Work Mode value of E6651(MID).
0: E6651
1: SA
2: SG

E6651_GetDLULMode

Declaration `int E6651_GetDLULMode(BYTE MID, int* Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** The current measure mode information. (DL: Downlink, UL: Uplink)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the measurement mode information of E6651, downlink or uplink.
0: DL
1: UL

E6651_SetDLULMode

Declaration `int E6651_SetDLULMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651' Measure Mode value to be set. (DL or UL)
  DL: 0
  UL: 1

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Measure Mode value of E6651(MID).

E6651_GetPathMode

Declaration `int E6651_GetPathMode(BYTE MID, int* Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Test Mode value of E6651
  SA: 0
  SG: 1
  SA+SG: 2

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Test Mode information of E6651(MID).

E6651_SetPathMode

Declaration `int E6651_SetPathMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651' Test Mode value to be set.
  SA: 0
  SG: 1
  SA+SG: 2

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Test Mode of E6651(MID).

E6651_GetDispMode

Declaration `int E6651_GetDispMode(BYTE MID, int* Value);`

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current Display Mode of SA. The E6651 is operating in SA mode.
Spectrum: 0
Modulation: 1
I/Q: 2
MAP: 3
EVS: 4
EVT: 5
CCDF: 6

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description Retrieve SA Display Mode information when E6651(MID) is operating in SA mode.

E6651_SetDispMode

Declaration `int E6651_SetDispMode(BYTE MID, int Value);`

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [in]: SA's Display Mode value to be set.
Spectrum: 0
Modulation: 1
I/Q: 2
MAP: 3
EVS: 4
EVT: 5
CCDF: 6

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description Set SA Display Mode of E6651(MID).

E6651_GetRFSwitch

Declaration `int E6651_GetRFSwitch(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current E6651's RF Switch operational information
  0: OFF
  1: ON

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve RF Switch's ON/OFF information.

E6651_SetRFSwitch

Declaration `int E6651_SetRFSwitch(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's RF Switch value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off RF Switch of E6651(MID).

E6651_GetMODSwitch

Declaration `int E6651_GetMODSwitch(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Modulation Switch On/Off information of E6651
  0:OFF
  1:ON

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Modulation Switch's operational information.

E6651_SetMODSwitch

Declaration `int E6651_SetMODSwitch(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Modulation Switch value to be set
  0: OFF
  1: ON

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off Modulation Switch of E6651(MID).

E6651_GetClockSource

Declaration `int E6651_GetClockSource(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current E6651's Clock Source information
  Internal: 0
  External: 1

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve E6651's Clock Source information.

Set_ClockSource

Declaration `int E6651_SetClockSource(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Clock Source value to be set.
  Internal: 0
  External: 1

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set E6651's Clock Source.

E6651_Preset

Declaration `int E6651_Preset(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Single Meas value for setting Measurement triggering mode of E6651(MID).

Return Value
- **Success :** STATE_SUCCESS(1)
- **Failure :** STATE_ERROR(-1)

Description Execute current preset of E6651(MID)

If (SingleMeas = TRUE) then Measurement triggering mode:= 'Single' (Default)

If (SingleMeas = FALSE) then Measurement triggering mode:= 'Cont'

E6651_GetFrequency

Declaration
```
int E6651_GetFrequency(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current E6651's Center Frequency information (Hz).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Center Frequency value of E6651(MID).

E6651_SetFrequency

Declaration `int E6651_SetFrequency(BYTE MID, double Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Center Frequency value to be set (Hz).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set E6651's Center Frequency.

E6651_GetAttenuate

Declaration `int E6651_GetAttenuate(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Attenuation value of E6651's attenuator (0 ~ 62 dB)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Attenuation value of E6651(MID).

E6651_SetAttenuate

Declaration `int E6651_SetAttenuate(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Attenuation value to be set(0 ~ 62 dB)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Attenuation value of E6651(MID).

E6651_GetAmplitude

Declaration
```
int E6651_GetAmplitude(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Amplitude value of E6651 (dBm).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve current Amplitude value of E6651(MID).

E6651_SetAmplitude

Declaration `int E6651_SetAmplitude(BYTE MID, double value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Amplitude value to be set (dBm).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Amplitude value of E6651(MID).

E6651_GetInPowerOffset

Declaration
```
int E6651_GetInPowerOffset(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Input Power Offset value of E6651.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Input Power Offset value of E6651(MID). Input Power Offset is introduced to compensate cable loss between E6651 and DUT in input power perspective.

When Input Power Offset value is set, E6651 calculates input power as measured input power plus Input Power Offset value.

E6651_SetInPowerOffset

Declaration
```
int E6651_SetInPowerOffset(BYTE MID, double
  value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651's Input Power offset value to be set. (-100dB ~ 100dB)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set the Input power offset value of E6651(MID).

E6651_GetOutPowerOffset

Declaration
```
int E6651_GetOutPowerOffset(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current E6651's Output power offset value (dBm).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Output Power Offset value of E6651(MID). Output Power Offset is introduced to compensate cable loss between E6651 and DUT in output power perspective. When Output Power Offset value is set, E6651 transmits output power more than designated amplitude of E6651 so that measured input power in the DUT front end shall match current amplitude of E6651.

E6651_SetOutPowerOffset

Declaration
```
int E6651_SetOutPowerOffset(BYTE MID, double
  value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	E6651's Output Power Offset value to be set (dBm).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Output Power Offset value of E6651(MID).

E6651_SetSAFrequency

Declaration
```
int E6651_SetSAFrequency(BYTE MID, double
  value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Frequency value of SA mode to be set when E6651 is operating in SA mode.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Center Frequency of SA mode when E6651(MID) is operating in SA mode.

E6651_SetSGFrequency

Declaration
```
int E6651_SetSGFrequency(BYTE MID, double
  value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Frequency value of SG mode to be set when E6651 is operating in SG mode.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Center Frequency of SG mode when E6651(MID) is operating in SG mode.

E6651_GetCAPTimeOffset

Declaration
```
int E6651_GetCAPTimeOffset(BYTE MID, DWORD*
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Current SA Capture Time offset value of E6651.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Capture Time Offset value which is calculated in E6651's Measure Mode.

E6651_SetCAPTimeOffset

Declaration
```
int E6651_SetCAPTimeOffset(BYTE MID, DWORD
  value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: SA Capture Time Offset value to be set.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Capture Time Offset value of E6651(MID).

E6651_GetSkewIGain

Declaration `int E6651_GetSkewIGain(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value of Skew I Gain.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve current Skew I Gain of E6651(MID).

E6651_SetSkewIGain

Declaration `int E6651_SetSkewIGain(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Skew I Gain value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Skew I Gain value of E6651(MID).

E6651_GetSkewQGain

Declaration `int E6651_GetSkewQGain(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value of Skew Q Gain.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve current Skew Q Gain value of E6651(MID).

E6651_SetSkewQGain

Declaration
```
int E6651_SetSkewQGain(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Skew I Gain value to be set.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Skew Q Gain value of E6651(MID).

E6651_GetSkewTheta

Declaration
```
int E6651_GetSkewTheta(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Current value of Skew Theta

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve current Skew Theta value of E6651(MID).

E6651_SetSkewTheta

Declaration
```
int E6651_SetSkewTheta(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Skew Theta value to be set.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Skew Theta value of E6651(MID).

E6651_GetSkewIOffset

Declaration `int E6651_GetSkewIOffset(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value of Skew I Offset

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Skew I Offset value of E6651(MID).

E6651_SetSkewIOffset

Declaration `int E6651_SetSkewIOffset(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Skew I Offset value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Skew I Offset value of E6651(MID).

E6651_GetSkewQOffset

Declaration `int E6651_GetSkewQOffset(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value of Skew Q Offset

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Skew Q Offset value of E6651(MID).

E6651_SetSkewQOffset

Declaration `int E6651_SetSkewQOffset(BYTE MID, int Value);`

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Skew I Offset value to be set

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Skew Q Offset value of E6651(MID).

Spectrum Mode

E6651_GetSPAverage

Declaration `int E6651_GetSPAverage(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** "Number of Average" information which is needed to determine measurement average value when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Number of Average" value of E6651(MID), when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPAverage

Declaration `int E6651_SetSPAverage(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "Number of Average" value when E6651 is operating in Spectrum Analyzer Mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Number of Average" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPCBW

Declaration `int E6651_GetSPCBW(BYTE MID, DWORD * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** "Channel Bandwidth" value when E6651 is operating is Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Channel Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPCBW

Declaration `int E6651_SetSPCBW(BYTE MID, DWORD Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "Channel Bandwidth" value to be set when E6651 is to be operated in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Channel Bandwidth" value of E6651(MID) when E6651 is to be operated in Spectrum Analyzer mode.

E6651_GetSPREF

Declaration `int E6651_GetSPREF(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Reference Level" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode (dB).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Reference Level" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPREF

Declaration `int E6651_SetSPREF(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "Reference Level" value of E6651 when E6651 is operating in Spectrum Analyzer mode. (-200dBm ~ 200dBm)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Reference Level" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPRBW

Declaration `int E6651_GetSPRBW(BYTE MID, DWORD * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Resolution Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Resolution Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPRBW

Declaration `int E6651_SetSPRBW(BYTE MID, DWORD Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** "Resolution Bandwidth" value of E6651 when E6651 is operating in Spectrum Analyzer mode. (10000 Hz (10KHz) ~ 100000Hz (100KHz)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Resolution Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPScale

Declaration `int E6651_GetSPScale(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Scale" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Scale" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPScale

Declaration `int E6651_SetSPScale(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "Scale" value of E6651 when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Scale" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPSpan

Declaration `int E6651_GetSPSpan(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Span" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Span" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetSPSpan

Declaration `int E6651_SetSPSpan(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "Span" value of E6651 when E6651 is operating in Spectrum Analyzer mode. (10000Hz(10KHz) ~ 16000000Hz(16MHz)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "Span" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetFFTPosition

Declaration `int E6651_GetFFTPosition(BYTE MID, DWORD * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "FFT Position" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode (PS - Physical Slot: 1 PS = 16 sample).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "FFT Position" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_SetFFTPosition

Declaration `int E6651_SetFFTPosition(BYTE MID, DWORD Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** "FFT Position" value of E6651 when E6651 is operating in Spectrum Analyzer mode (PS - Physical Slot: 1 PS = 16 sample).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set "FFT Position" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPCHPower

Declaration `int E6651_GetSPCHPower(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Channel power value of E6651(MID) within pre-defined channel bandwidth when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve measured channel power value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPTOTPower

Declaration `int E6651_GetSPTOTPower(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Total Power" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Total Power" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPOBW

Declaration `int E6651_GetSPOBW(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current "Occupied Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve "Occupied Bandwidth" value of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.

E6651_AddMarker

Declaration `int E6651_AddMarker(BYTE MID, double Freq);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Freq [in]:** Frequency value of Marker to be added.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Add Marker by setting Frequency value of the marker when E6651 is operating in Spectrum Analyzer mode.

E6651_EditMarker

Declaration `int E6651_EditMarker(BYTE MID, int Index, double Freq);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Index [in]:** Marker's index value when the marker's frequency value is to be changed (0 ~ 11)
- **Freq [in]:** New Frequency value of Marker.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Edit the Frequency value of a certain marker when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_DeleteMarker

Declaration `int E6651_DeleteMarker(BYTE MID, int Index);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Index [in]:** Marker's Index value when the Marker is to be deleted.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Delete a certain Marker when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_GetMarkerValue

Declaration `int E6651_GetMarkerValue(BYTE MID, int Index, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Index [in]:** Marker's Index value when user want to fetch the marker's power information.
- **Value [out]:** Measured power value of the Marker designated by Index.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve a certain marker's power information when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_GetUIQCHPower

Declaration `int E6651_GetUIQCHPower(BYTE Mouldable * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Channel power value when E6651 is operating in Spectrum Analyzer mode and Uplink measurement mode is in progress.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Channel power value of E6651(MID) when E6651 is operating in Uplink measurement mode.

E6651_GetUIQEVM

Declaration `int E6651_GetUIQEVM(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** EVM value when E6651 is operating in Spectrum Analyzer mode and Uplink measurement mode is in progress. (dB)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve EVM value of E6651(MID) when E6651 is operating in Uplink measurement mode.

E6651_GetUIQFreqOffset

Declaration
```
int E6651_GetUIQFreqOffset(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** designated Frequency Offset value of E6651 when SA is operating in Uplink Measure Mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve designated Frequency Offset value when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_SetSPMMode

Declaration `int E6651_SetSPMMode(BYTE MID, int Mode);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Mode [in]:** Time Gate Mode information of E6651(MID) when E6651 is operating is Spectrum Analyzer mode.
  ON: 1
  OFF: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off Time Gate mode when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_GetUIQMax

Declaration `int E6651_GetUIQMax(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** measured Maximum Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Maximum Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUIQMin

Declaration `int E6651_GetUIQMin(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** measured Minimum Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Minimum Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_SetSPMaxHold

Declaration `int E6651_SetSPMaxHold(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Max Hold information of E6651(MID) when E6651 is operating is Spectrum Analyzer mode.
  ON: 1
  OFF: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off Max Hold when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_GetUIQAvgPower

Declaration
```
int E6651_GetUIQAvgPower(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** measured Average Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Average Flatness value when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUIQNegHalfAvgPower

Declaration

```
int E6651_GetUIQNegHalfAvgPower(BYTE MID, double
  * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Average power of sub-carriers (sub-carrier interval:-N to -1) when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Average power within sub-carrier interval of –N to -1 when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUIQPosHalfAvgPower

Declaration

```
int E6651_GetUIQPosHalfAvgPower(BYTE MID, double
  * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Average power of sub-carriers (sub-carrier interval:+N to +1) when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Average power within sub-carrier interval of +N to +1 when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUIQNegQuaterAvgPower

Declaration

```
int E6651_GetUIQNegQuaterAvgPower(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Average power of sub-carriers (sub-carrier interval: -Nused/2 to -Nused/4) when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Average power within sub-carrier interval of -Nused/2 to -Nused/4 when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUIQPosQuaterAvgPower

Declaration

```
int E6651_GetUIQPosQuaterAvgPower(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Average power of sub-carriers (sub-carrier interval: +Nused/2 to +Nused /4) when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Average power within sub-carrier interval of +Nused/2 to +Nused/4 when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_IsUIQFlatness

Declaration `int E6651_IsUIQFlatness(MID: byte);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Get information of Flatness pass /fail result to see if DUT meet the pass criteria of flatness test.
Flatness: 1
non flatness: 0

E6651_SetRFAttenuate

Declaration `int E6651_SetRFAttenuate(BYTE MID, int Atten);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Eaten [in]: RF Attenuation value to be set (dB).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set RF Attenuation value of E6651(MID).

E6651_GetRFAttenuate

Declaration `int E6651_GetRFAttenuate(BYTE MID, int * Atten);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Atten [out]: Current RF Attenuation value of E6651(MID) (dB).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve RF Attenuation value of E6651(MID).

E6651_SetIFAttenuate

Declaration `int E6651_SetIFAttenuate(BYTE MID, int Atten);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Atten [in]:** IF Attenuation value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set IF Attenuation value of E6651(MID).

E6651_GetIFAttenuate

Declaration `int E6651_GetIFAttenuate(BYTE MID, int * Atten);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Atten [out]:** Current IF Attenuation value of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve IF Attenuation value of E6651(MID).

E6651_GetDLPreambleEVM

Declaration
```
int E6651_GetDLPreambleEVM(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** measured Preamble EVM value when E6651 is operating in SA DL Measure Mode. (%)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Get measured Preamble EVM value of E6651(MID) when E6651 is operating in SA DL Measure Mode.

E6651_GetDLPilotEVMPct

Declaration

```
int E6651_GetDLPilotEVMPct(BYTE MID, double *
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: measured Pilot EVM value when E6651 is operating in SA DL Measure Mode. (%)

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Get measured Pilot EVM value of E6651(MID) when E6651 is operating in SA DL Measure Mode.

E6651_SetSAWindowType

Declaration

```
int E6651_SetSAWindowType(BYTE MID, int
  WindowType);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

WindowType [in]: Windowing type of E6651(MID) when E6651 is operating in Spectrum Analyzer mode.
Hamming: 0
Rectangular: 1

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Designate E6651(MID)'s windowing type when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPMaskValue

Declaration

```
int E6651_GetSPMaskValue(BYTE MID, const double
  SFreq, const double EFreq, double* MaxFreq,
  double* MaxPwr);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

SFreq [in]: measurement Start Frequency of Spectrum Analyzer screen.

EFreq [in]: measurement Stop Frequency of Spectrum Analyzer screen.

MaxFreq [out]: Frequency component of peak power within start and stop frequency band.

MaxPwr [out]: peak power value within start and stop frequency band.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve power and frequency information of peak power point within start and stop frequency band when E6651(MID) is operating in Spectrum Analyzer mode.

E6651_GetSPAvgPower

Declaration

```
int E6651_GetSPAvgPower(BYTE MID, const double
  SFreq, const double EFreq, double * AvgPower);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

SFreq [in]: Measurement Start Frequency

EFreq [in]: Measurement Stop Frequency

AvgPower [out]: Averaged power value within frequency interval from SFreq to EFreq.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Averaged power value of E6651(MID) within frequency interval from SFreq to EFreq when E6651 is operating in Spectrum Analyzer mode.

E6651_GetSPUserCHPower

Declaration

```
int E6651_GetSPUserCHPower(BYTE MID, double
  Freq, double * UserCHPower);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Freq [in]: Center Frequency value when Channel Power is measured.

UserCHPower [out]: Channel power value for a signal with designated Center frequency.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve channel power value when a signal is defined using center frequency.

Flatness

E6651_GetUpFlatnessGroup0MIN

Declaration

```
int E6651_GetUpFlatnessGroup0MIN(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The minimum value of the first group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Average the minimum value of the first group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup0MAX

Declaration

```
int E6651_GetUpFlatnessGroup0MAX(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The maximum value of the first group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the maximum value of the first group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup1MIN

Declaration

```
int E6651_GetUpFlatnessGroup1MIN(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The minimum value of the second group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the minimum value of the second group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup1MAX

Declaration

```
int E6651_GetUpFlatnessGroup1MAX(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The maximum value of the second group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the maximum value of the second group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup2MIN Declaration

Declaration

```
int E6651_GetUpFlatnessGroup2MIN(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The minimum value of the third group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the minimum value of the third group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup2MAX

Declaration

```
int E6651_GetUpFlatnessGroup2MAX(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The maximum value of the third group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the maximum value of the third group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup3MIN

Declaration

```
int E6651_GetUpFlatnessGroup3MIN(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The minimum value of the last group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the minimum value of the last group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpFlatnessGroup3MAX

Declaration

```
int E6651_GetUpFlatnessGroup3MAX(BYTE MID,
  double * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: The maximum value of the last group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve the maximum value of the last group of 4 groups when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpGroup0MINSC

Declaration
```
int E6651_GetUpGroup0MINSC(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The sub carrier index of the minimum value in the first group.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve the sub carrier index of the minimum value in the first group.

E6651_GetUpGroup0MAXSC

Declaration
```
int E6651_GetUpGroup0MAXSC(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The sub carrier index of the maximum value in the first group.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve the sub carrier index of the maximum value in the first group.

E6651_GetUpGroup1MINSC

Declaration
```
int E6651_GetUpGroup1MINSC(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The sub carrier index of the minimum value in the second group.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve the sub carrier index of the minimum value in the first group.

E6651_GetUpGroup1MAXSC

Declaration
```
int E6651_GetUpGroup1MAXSC(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	The sub carrier index of the maximum value in the second group.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve the sub carrier index of the maximum value in the second group.

E6651_GetUpGroup2MINSC

Declaration
```
int E6651_GetUpGroup2MINSC(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	The sub carrier index of the minimum value in the third group.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve the sub carrier index of the minimum value in the third group.

E6651_GetUpGroup2MAXSC

Declaration
```
int E6651_GetUpGroup2MAXSC(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	The sub carrier index of the maximum value in the third group.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve the sub carrier index of the maximum value in the third group.

E6651_GetUpGroup3MINSC

Declaration
```
int E6651_GetUpGroup3MINSC(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The sub carrier index of the minimum value in the last group.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve the sub carrier index of the minimum value in the last group.

E6651_GetUpGroup3MAXSC

Declaration
```
int E6651_GetUpGroup3MAXSC(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The sub carrier index of the maximum value in the last group.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve the sub carrier index of the maximum value in the last group.

E6651_GetUpMAXABSDiff

Declaration
```
int E6651_GetUpMAXABSDiff(BYTE MID, double *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: The biggest difference range between two sub-carriers when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Average power within sub-carrier interval of +NUsed/2 to +Nused/4 when E6651(MID) is operating in Uplink Flatness Measure Mode.

E6651_GetUpMAXABSSC1

Declaration
```
int E6651_GetUpMAXABSSC1(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	One of sub-carrier indexes have the biggest difference between sub-carriers.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve one of sub-carrier indexes have the biggest difference between sub-carriers.

E6651_GetUpMAXABSSC2

Declaration
```
int E6651_GetUpMAXABSSC2(BYTE MID, double *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	The other of sub-carrier indexes have the biggest difference between sub-carriers. when E6651(MID) is operating in Uplink Flatness Measure Mode.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve the other of sub-carrier indexes have the biggest difference between sub-carriers. when E6651(MID) is operating in Uplink Flatness Measure Mode.

6651_GetFlatnessDCPWR

Declaration
```
int E6651_GetUIQPosQuaterAvgPower(BYTE MID,
  double * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	The relative power difference to total power.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve the relative power difference to total power when E6651(MID) is operating in Uplink Flatness Measure Mode.

BS Emulator Mode

E6651_GetBSStart

Declaration `int E6651_GetBSStart(BYTE MID, BYTE * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current status of E6651 Base Station Emulation mode.
  Start: 1
  Stop: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve E6651(MID)'s Base Station Emulation mode information.

E6651_SetBSStart

Declaration `int E6651_SetBSStart(BYTE MID, BYTE Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** E6651 Base Station Emulation mode information.
  Start: 1
  Stop: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Stop or Start Base Station Emulation mode of E6651(MID).

E6651_GetBSPreamble

Declaration `int E6651_GetBSPreamble(BYTE MID, BYTE * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Preamble Index value of E6651(MID). (0 ~ 113)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Preamble Index value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_SetBSPreamble

Declaration `int E6651_SetBSPreamble(BYTE MID, BYTE Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Preamble Index value to be set when E6651 is operating in Base Station Emulator mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Preamble Index value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_GetBSUlPermbase

Declaration
```
int E6651_GetBSUlPermbase(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Permutation base value of E6651(MID) when E6651 is operating in Base Station Emulator mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Current Permutation base value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_SetBSUlPermbase

Declaration `int E6651_SetBSUlPermbase(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** UL Permutation base value to be set when E6651 is operating in Base Station Emulator mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set UL Permutation base value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_GetBSID

Declaration `int E6651_GetBSID(BYTE MID, Char * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Current Base Station ID value of E6651(MID) when E6651 is operating in Base Station Emulator mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Base Station ID value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_SetBSID

Declaration `int E6651_SetBSID(BYTE MID, Char * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Base Station ID value to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Base Station ID value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_GetBSRngTimeOffset

Declaration
```
int E6651_GetBSRngTimeOffset(BYTE MID, int *
  Offset);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Offset [out]:** Current Timing Offset value of E6651(MID) when E6651 is operating in Base Station Emulator mode.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Timing Offset value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_SetBSRngTimeOffset

Declaration
```
int E6651_SetBSRngTimeOffset(BYTE MID, int
  Offset);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Offset [in]:	Timing Offset value to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Timing Offset value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_GetBSFrameOffset

Declaration
```
int E6651_GetBSFrameOffset(BYTE MID, int *
  Offset);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Offset [out]:	Current Frame Offset value of E6651(MID) when E6651 is operating in Base Station Emulator mode.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Frame Offset value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_SetBSFrameOffset

Declaration
```
int E6651_SetBSFrameOffset(BYTE MID, int
  Offset);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Offset [in]:	Frame Offset value to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Frame Offset value of E6651(MID) when E6651 is operating in Base Station Emulator Mode.

E6651_GetBSRepetition

Declaration `int E6651_GetBSRepetition(BYTE MID,int * value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Current Repetition value of E6651(MID).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Repetition value of E6651(MID).
0: none
1: 2 Repetition
2: 4 Repetition
3: 6 Repetition

E6651_SetBSRepetition

Declaration `int E6651_SetBSRepetition(BYTE MID,int * value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Repetition value of E6651(MID) to be set.

Return Value
Success: STATE_SUCCESS(1),
Failure: STATE_ERROR(-1)

Description Set Repetition value of E6651(MID)
0: none
1: 2 Repetition
2: 4 Repetition
3: 6 Repetition

E6651_GetSamplingFrequency

Declaration
```
int E6651_GetSamplingFrequency(BYTE MID, double
  * value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Current Sampling Frequency value of E6651(MID).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Sampling Frequency value of E6651(MID).

E6651_SetSamplingFrequency

Declaration `int E6651_SetSamplingFrequency(BYTE MID, double value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Sampling Frequency value of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Sampling Frequency value of E6651(MID).

E6651_GetBSRNGRSP_PowerOffset

Declaration `int E6651_GetBSRNGRSP_PowerOffset(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Power Offset of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Power Offset of E6651(MID).

E6651_SetBSRNGRSP_PowerOffset

Declaration `int E6651_SetBSRNGRSP_PowerOffset(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Power Offset of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Power Offset of E6651(MID) during Initial Ranging.

E6651_GetBSRNGRSP_FreqOffset

Declaration
```
int E6651_GetBSRNGRSP_FreqOffset(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Frequency Offset value of E6651(MID)

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Frequency Offset value of E6651(MID).

E6651_SetBSRNGRSP_FreqOffset

Declaration
```
int E6651_SetBSRNGRSP_FreqOffset(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Frequency Offset value of E6651(MID) to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Frequency Offset value of E6651(MID) is set during Initial Ranging.

E6651_GetBSRNGRSP_TimeOffset

Declaration
```
int E6651_GetBSRNGRSP_TimeOffset(BYTE MID,
  double * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Time Offset of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Time Offset of E6651(MID).

E6651_SetBSRNGRSP_TimeOffset

Declaration

```
int E6651_SetBSRNGRSP_TimeOffset(BYTE MID,
  double Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [in]: Time Offset of E6651(MID) to be set.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Set Time Offset of E6651(MID) is set during Initial Ranging. Units of set value: 0.25 * n ps.

E6651_GetBSRNGRSP_Status

Declaration

```
int E6651_GetBSRNGRSP_Status(BYTE MID, int *
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current State value of RNG-RSP msg. Transmission Mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Status value of Initial Ranging' Result Transmission Mode.

E6651_GetBSRNGRSP_Status retrieves:

0 if Transmission mode is changed 'Success' or 'Continue' by current rule when a mobile phone is trying Initial Ranging.

1 if Transmission mode is always 'Success' when a mobile phone is trying Initial Ranging.

2 if Transmission mode is always 'Continue' when a mobile phone is trying Initial Ranging.

3 if Transmission mode is always 'Abort' when a mobile phone is trying Initial Ranging.

E6651_SetBSRNGRSP_Status

Declaration

```
int E6651_SetBSRNGRSP_Status(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	State value of RNG-RSP msg. Transmission Mode to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Set State value of Initial Ranging' Result Transmission Mode.

E6651_SetBSRNGRSP_Status sets:

0 if Transmission mode is changed 'Success' or 'Continue' by current rule when a mobile phone is trying Initial Ranging.

1 if Transmission mode is always 'Success' when a mobile phone is trying Initial Ranging.

2 if Transmission mode is always 'Continue' when a mobile phone is trying Initial Ranging.

3 if Transmission mode is always 'Abort' when a mobile phone is trying Initial Ranging.

E6651_GetDLMAPRepetition

Declaration

```
int E6651_GetDLMAPRepetition(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current DL MAP Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve DL MAP Repetition value of E6651(MID).

E6651_SetDLMAPRepetition

Declaration
```
int E6651_SetDLMAPRepetition(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	DL MAP Repetition value of E6651(MID) to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set DL MAP Repetition value of E6651(MID).

E6651_GetULMAPRepetition

Declaration
```
int E6651_GetULMAPRepetition(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current UL MAP Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve UL MAP Repetition value of E6651(MID).

E6651_SetULMAPRepetition

Declaration
```
int E6651_SetULMAPRepetition(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	UL MAP Repetition value of E6651(MID) to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set UL MAP Repetition value of E6651(MID).

E6651_GetULBurstRepetition

Declaration
```
int E6651_GetULBurstRepetition(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current UL Burst Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve UL Burst Repetition value of E6651(MID).

E6651_SetULBurstRepetition

Declaration
```
int E6651_SetULBurstRepetition(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	UL Burst Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set UL Burst Repetition value of E6651(MID).

E6651_GetDLBurstRepetition

Declaration
```
int E6651_GetDLBurstRepetition(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current DL Burst Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve DL Burst Repetition value of E6651(MID).

E6651_SetDLBurstRepetition

Declaration

```
int E6651_SetDLBurstRepetition(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	DL Burst Repetition value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set DL Burst Repetition value of E6651(MID).

BS Emulator Functions

E6651_GetSSMacAddr

Declaration `int E6651_GetSSMacAddr(BYTE MID, Char * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** MAC Address information of SS (Subscriber Station) when the SS successfully finished network entrance procedure to E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the SS MAC address information after SS finish network entrance procedure to E6651(MID).

E6651_GetSE6651ate

Declaration `int E6651_GetSE6651ate(BYTE MID, BYTE * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current status of SS which is connected to E6651(Base Station Emulator mode)
  ssDREG: 0
  ssINIT: 1
  ssNEGO: 2
  ssPKM: 3
  ssREG: 4
  ssCONN: 5
  ssIDLE: 6
  ssHO: 7

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the SS operational status information when SS is inter-operating with E6651(MID).

E6651_GetSSCINR

Declaration `int E6651_GetSSCINR(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** SS CINR value when the SS is inter-operating with E6651(Base Station Emulator)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the SS CINR value when SS is inter-operating with E6651. SS periodically reports CINR value to E6651.

E6651_GetSSRSSI

Declaration `int E6651_GetSSRSSI(BYTE MID, double * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** The SS RSSI value when SS is inter-operating with E6651(Base Station Emulator)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the SS RSSI value when SS is inter-operating with E6651. SS periodically reports RSSI value to E6651.

E6651_GetE6651TxPower

Declaration
```
int E6651_GetE6651TxPower(BYTE MID, double *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** The SS Tx power value when SS is inter-operating with E6651(Base Station Emulator)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve the SS Tx power value when SS is inter-operating with E6651. SS periodically reports Tx power value to E6651.

E6651_GetPERInfo

Declaration

```
int E6651_GetPERInfo(BYTE MID, DWORD *
  TestState, DWORD * PktTotalCnt, DWORD *
  PktCnt, DWORD * Loss, double * PER);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

TestState [out]: Three test mode (UL Padding, DL UDP, DL Ping)'s status information when E6651 is operating in Base Station Emulator mode. (ON: 1, OFF: 0)

PktTotalCnt [out]: The number of Transmitted packets from E6651(Base Station Emulator) to SS in DL UDP or DL Ping test.

PktCnt [out]: The number of acknowledged packets for transmitted packets from E6651(Base Station Emulator) in DL UDP or DL Ping test.

Loss [out]: The number of unacknowledged packets for the transmitted packets form E6651(Base Station Emulator).

PER [out]: PER value measured at E6651(Base Station Emulator).

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve PER information when DL Ping or DL UDP test is in progress after SS finish network entrance procedure to E6651(Base Station Emulator).

BS Emulator Test Functions

E6651_GetBSTestMode

Declaration `int E6651_GetBSTestMode(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current activated test mode information.
  UL Padding: 0
  DL UDP: 1
  DL Ping: 2

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve current active test mode of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_SetBSTestMode

Declaration `int E6651_SetBSTestMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Test mode information to be set.
  UL Padding: 0
  DL UDP: 1
  DL Ping: 2

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set test mode of E6651(MID).

E6651_GetBSRunTest

Declaration `int E6651_GetBSRunTest(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** On/Off information of Test Mode
  ON: 1
  OFF: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve test mode On/Off information of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_SetBSRunTest

Declaration `int E6651_SetBSRunTest(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Activation information of Test Mode
  ON: 1
  OFF: 0

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off test mode of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_GetDLPayloadPattern

Declaration
```
int E6651_GetDLPayloadPattern(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Payload Pattern value of E6651(MID).
  1: Random Mode
  0: Pattern Mode

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Payload Pattern value of E6651(MID).

E6651_SetDLPayloadPattern

Declaration
```
int E6651_SetDLPayloadPattern(BYTE MID, int
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Payload Pattern value of E6651(MID) to be set.
  1: Random Mode
  0: Pattern Mode

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Payload Pattern value of E6651(MID).

UL Padding Test

E6651_SetBSULPadTest

Declaration

```
int E6651_SetBSULPadTest(BYTE MID, int Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [in]: Activation information of UL Padding Test
ON: 1
OFF: 0

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Turn On or Turn Off UL Padding test of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_GetBSULPadUlMod

Declaration

```
int E6651_GetBSULPadUlMod(BYTE MID, int *
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current UL Modulation Mode of E6651 (Base Station Emulator)
QPSK (CTC) 1/2: 1
QPSK (CTC) 3/4: 2
16-QAM (CTC) 1/2: 3
16-QAM (CTC) 3/4: 4

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve modulation mode information of E6651(MID) when E6651 is operating in UL Padding Test mode.

E6651_SetBSULPadUlMod

Declaration `int E6651_SetBSULPadUlMod(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** UL Modulation Mode value to be set.
  QPSK (CTC) 1/2: 1
  QPSK (CTC) 3/4: 2
  16-QAM (CTC) 1/2: 3
  16-QAM (CTC) 3/4: 4

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set modulation mode of E6651(MID) when E6651 is operating in UL Padding Test mode.

E6651_GetBSULPadUlSlot

Declaration
```
int E6651_GetBSULPadUlSlot(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** UL Slot number information for UL Padding Test.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve UL Slot value of E6651(MID) when E6651 is operating in UL Padding Test mode.

E6651_SetBSULPadUlSlot

Declaration
```
int E6651_SetBSULPadUlSlot(BYTE MID, Value:
  integer);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** UL Slot value to be set for UL Padding Test.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set UL Slot value of E6651(MID) when E6651 is operating in UL Padding Test mode.

DL UDP Test

E6651_SetBSDLUdpTest

Declaration `int E6651_SetBSDLUdpTest(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Activation information of DL UDP Test (ON: 1, OFF: 0)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Turn On or Turn Off DL UDP test of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_GetBSDLUdpDlMod

Declaration
```
int E6651_GetBSDLUdpDlMod(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Current DL Modulation Mode of E6651 (Base Station Emulator)
  QPSK (CTC) 1/2: 0
  QPSK (CTC) 3/4: 1
  16-QAM (CTC) 1/2: 2
  16-QAM (CTC) 3/4: 3
  64-QAM (CTC) 1/2: 4
  64-QAM (CTC) 2/3: 5
  64-QAM (CTC) 3/4: 6
  64-QAM (CTC) 5/6: 7

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve modulation mode information of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_SetBSDLUdpDlMod

Declaration `int E6651_SetBSDLUdpDlMod(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.

Value [in]: DL Modulation Mode value to be set in DL UDP test mode.
QPSK (CTC) 1/2: 0
QPSK (CTC) 3/4: 1
16-QAM (CTC) 1/2: 2
16-QAM (CTC) 3/4: 3
64-QAM (CTC) 1/2: 4
64-QAM (CTC) 2/3: 5
64-QAM (CTC) 3/4: 6
64-QAM (CTC) 5/6: 7

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set modulation mode of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_GetBSDLUdpLength

Declaration
```
int E6651_GetBSDLUdpLength(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Payload Length value in DL UDP test (1 ~ 3000).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Payload length value of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_SetBSDLUdpLength

Declaration
```
int E6651_SetBSDLUdpLength(BYTE MID, int Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Payload Length value to be set for DL UDP Test (1 ~ 3000).

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Payload length value of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_GetBSDLUdpRate

Declaration `int E6651_GetBSDLUdpRate(BYTE MID, int * Value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Frame Rate information for DL UDP Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Frame Rate information of E6651(MID) when E6651 is operating in DL UDP Test mode. Frame Rate determines how many frames are allocated for one data packet transmission. For example, single packet size is 300bytes and frame rate is three, then 300bytes data are transmitted over 3 frames period.

E6651_SetBSDLUdpRate

Declaration `int E6651_SetBSDLUdpRate(BYTE MID, int * Value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Frame Rate value to be set for DL UDP Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Frame Rate value of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_GetBSDLUdpTotal

Declaration
```
int E6651_GetBSDLUdpTotal(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Total Packet value information for DL UDP Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Total Packet value of E6651(MID) when E6651 is operating in DL UDP Test mode.

E6651_SetBSDLUdpTotal

Declaration
```
int E6651_SetBSDLUdpTotal(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Total Packet value to be set for DL UDP Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description
Set Total Packet value of E6651(MID) when E6651 is operating in DL UDP Test mode.

DL Ping Test

E6651_SetBSDLPingTest

Declaration
```
int E6651_SetBSDLPingTest(BYTE MID, int Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Activation information of DL Ping Test (ON: 1, OFF: 0)

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description
Turn On or Turn Off DL Ping test of E6651(MID) when E6651 is operating in Base Station Emulator mode.

E6651_GetBSDLPingDlMod

Declaration

```
int E6651_GetBSDLPingDlMod(BYTE MID, int *
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current DL Modulation Mode of E6651(Base Station Emulator)
QPSK (CTC) 1/2: 0
QPSK (CTC) 3/4: 1
16-QAM (CTC) 1/2: 2
16-QAM (CTC) 3/4: 3
64-QAM (CTC) 1/2: 4
64-QAM (CTC) 2/3: 5
64-QAM (CTC) 3/4: 6
64-QAM (CTC) 5/6: 7

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Retrieve modulation mode information of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_SetBSDLPingDlMod

Declaration

```
int E6651_SetBSDLPingDlMod(BYTE MID, int Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: DL Modulation Mode value to be set in DL Ping test mode.
QPSK (CTC) 1/2: 0
QPSK (CTC) 3/4: 1
16-QAM (CTC) 1/2: 2
16-QAM (CTC) 3/4: 3
64-QAM (CTC) 1/2: 4
64-QAM (CTC) 2/3: 5
64-QAM (CTC) 3/4: 6
64-QAM (CTC) 5/6: 7

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Set modulation mode of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_GetBSDLPingLength

Declaration
```
int E6651_GetBSDLPingLength(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Payload Length value in DL Ping test (1 ~ 3000)

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Payload length value of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_SetBSDLPingLength

Declaration
```
int E6651_SetBSDLPingLength(BYTE MID, int
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: Payload Length value to be set for DL Ping Test (1 ~ 3000)

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Payload length value of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_GetBSDLPingRate

Declaration
```
int E6651_GetBSDLPingRate(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Frame Rate information for DL Ping Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Frame Rate information of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_SetBSDLPingRate

Declaration
```
int E6651_SetBSDLPingRate(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Frame Rate value to be set for DL Ping test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Frame Rate value of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_GetBSDLPingTotal

Declaration
```
int E6651_GetBSDLPingTotal(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Total Packet value information for DL Ping Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Retrieve Total Packet value of E6651(MID) when E6651 is operating in DL Ping Test mode.

E6651_SetBSDLPingTotal

Declaration
```
int E6651_SetBSDLPingTotal(BYTE MID, int *
  Value);
```

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [out]: Total Packet value to be set for DL Ping Test.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set Total Packet value of E6651(MID) when E6651 is operating in DL Ping Test mode.

Power Control Test

E6651_GetPCEIRP

Declaration `int E6651_GetPCEIRP(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current DCD message's EIRP of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve DCD message's EIRP of E6651(MID).

E6651_SetPCEIRP

Declaration `int E6651_SetPCEIRP(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** EIRP value of E6651(MID) to be set in DCD message.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set EIPR value of E6651(MID) in DCD message.

E6651_GetPCEIRxP

Declaration `int E6651_GetPCEIRxP(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current DCD message's Earwax IR, max value of E6651(MID)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve DCD message's EIRxP IR, max value of E6651(MID).

E6651_SetPCEIRxP

Declaration `int E6651_SetPCEIRxP(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** EIRxP IR,max value of E6651(MID) to be set in DCD message.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set EIRxP IR,max value of E6651(MID) in DCD message.

E6651_GetPCNIEnable

Declaration `int E6651_GetPCNIEnable(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value is indicated that NI (Noise Interference) is used or not.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve NI (Noise Interference) using or not value.

E6651_GetPCNIEnable returns 0(Disable) if NI was not used, or returns 1 if NI was used.

E6651_SetPCNIEnable

Declaration `int E6651_SetPCNIEnable(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** The value decides to use NI or not to be set

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set the value decides to use NI or not of E6651(MID)

E6651_SetPCNIEnable sets 0(Disable) if NI will not be used, or returns 1 if NI will be used.

E6651_GetPCNIPUSC

Declaration `int E6651_GetPCNIPUSC(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Power Level of E6651(MID)'s NI PUSC

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Power Level of E6651(MID)'s NI PUSC.

Limits of Power Level: 0~255(-150dBm ~ -22.5 dBm)

E6651_SetPCNIPUSC

Declaration `int E6651_SetPCNIPUSC(BYTE MID, int value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Power Level of E6651(MID)'s NI PUSC to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Power Level of E6651(MID)'s NI PUSC.

Limits of Power Level: 0~255(-150dBm ~ -22.5 dBm).

E6651_GetBSRNGRSPOffset

Declaration
```
int E6651_GetBSRNGRSPOffset(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Power Offset is user can set additionally during Initial Ranging or Periodic Ranging.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Power Offset is user can set additionally during Initial Ranging or Periodic Ranging.

E6651_SetBSRNGRSPOffset

Declaration `int E6651_SetBSRNGRSPOffset(BYTE MID, int value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Power Offset of E6651(MID) is user can set additionally during Initial Ranging or Periodic Ranging to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Power Offset of E6651(MID) is user can set additionally during Initial Ranging or Periodic Ranging.

E6651_GetBSREPREQEnable

Declaration
```
int E6651_GetBSREPREQEnable(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current value is indicated that REP-REQ is used or not

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve REP-REQ using or not value.

E6651_ GetBSREPREQEnable returns 0(Disable) if REP-REQ was not used, or returns 1 if REP-REQ was used.

E6651_SetBSREPREQEnable

Declaration
```
int E6651_SetBSREPREQEnable(BYTE MID, int
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** The value decides to use REP-REQ or not to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set the value decides to use REP-REQ or not of E6651(MID). E6651_SetBSREPREQEnable sets 0(Disable) if REP-REQ will not be used, or returns 1 if REP-REQ will be used.

E6651_GetBSREPREQFrameRate

Declaration `int E6651_GetBSREPREQFrameRate(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Frame Rate value of REP-REQ.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Frame Rate value of REP-REQ.

E6651_SetBSREPREQFrameRate

Declaration `int E6651_SetBSREPREQFrameRate(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Frame Rate value of REP-REQ to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Frame Rate value of REP-REQ.

E6651_GetPCTestMode

Declaration `int E6651_GetPCTestMode(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current Power Control Test Mode of E6651(MID)

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve Power Control Test Mode of E6651(MID).
0: RNG-RSP
1: Power Control IE in UL-MAP
2: FPC
3: PMC-RSP

E6651_SetPCTestMode

Declaration `int E6651_SetPCTestMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** Power Control Test Mode of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set Power Control Test Mode of E6651(MID).
0: RNG-RSP
1: Power Control IE in UL-MAP
2: FPC
3: PMC-RSP

E6651_GetPCRNGRSPPowerAdjust

Declaration
```
int E6651_GetPCRNGRSPPowerAdjust(BYTE MID, int *
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current RNG-RSP Power Adjust value of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve RNG-RSP Power Adjust value of E6651(MID). Limits of Power Adjust: -128 ~ 127.

E6651_SetPCRNGRSPPowerAdjust

Declaration
```
int E6651_SetPCRNGRSPPowerAdjust(BYTE MID, int
  Value);
```

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** RNG-RSP Power Adjust value of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set RNG-RSP Power Adjust value of E6651(MID). Limits of Power Adjust: -128 ~ 127.

E6651_GetPCIEPowerAdjust

Declaration
```
int E6651_GetPCIEPowerAdjust(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Power Control IE Power Adjust value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description
Retrieve Power Control IE Power Adjust value of E6651(MID).
Limits of Power Adjust: -128 ~ 127.

E6651_SetPCIEPowerAdjust

Declaration
```
int E6651_SetPCIEPowerAdjust(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Power Control IE Power Adjust value of E6651(MID) to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description
Set Power Control IE Power Adjust value of E6651(MID).
Limits of Power Adjust: -128 ~ 127.

E6651_GetPCFPCPowerAdjust

Declaration
```
int E6651_GetPCFPCPowerAdjust(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current FPC Power Adjust value of E6651(MID).

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description
Retrieve FPC Power Adjust value of E6651(MID).
Limits of Power Adjust: -128 ~ 127.

E6651_SetPCFPCPowerAdjust

Declaration `int E6651_SetPCFPCPowerAdjust(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** FPC Power Adjust value of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set FPC Power Adjust value of E6651(MID). Limits of Power Adjust: -128 ~ 127.

E6651_GetPCPMCLoopMode

Declaration `int E6651_GetPCPMCLoopMode(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current PMC-RSP Loop Mode of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve PMC-RSP Loop Mode of E6651(MID).
0: Close Loop
2: Open Loop Passive
3: Open Loop Active

E6651_SetPCPMCLoopMode

Declaration `int E6651_SetPCPMCLoopMode(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** PMC-RSP Loop Mode of E6651(MID) to be set.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set PMC-RSP Loop Mode of E6651(MID).
0: Close Loop
2: Open Loop Passive
3: Open Loop Active

E6651_GetPCPMCPowerAdjust

Declaration `int E6651_GetPCPMCPowerAdjust(BYTE MID, int * Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [out]:** Current PMC-RSP Power Adjust value of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Retrieve PMC-RSP Power Adjust value of E6651(MID). Limits of Power Adjust: -128 ~ 127.

E6651_SetPCPMCPowerAdjust

Declaration `int E6651_SetPCPMCPowerAdjust(BYTE MID, int Value);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.
- **Value [in]:** PMC-RSP Power Adjust value of E6651(MID).

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Set PMC-RSP Power Adjust of E6651(MID). Limits of Power Adjust: -128 ~ 127.

E6651_PCSendMessage

Declaration `int E6651_PCSendMessage(BYTE MID);`

Parameter
- **MID [in]:** E6651's equipment number to be controlled remotely.

Return Value
- **Success:** STATE_SUCCESS(1)
- **Failure:** STATE_ERROR(-1)

Description Send Power Control Test Message to a mobile phone.

E6651_GetBSFullOccupied

Declaration

```
int E6651_GetBSFullOccupied(BYTE MID, int *
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current Full Occupied Function's On or Off information of E6651(MID) when E6651 is operating in Base Station Emulator mode.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Retrieve Full Occupied Function's On or Off information of E6651(MID) when E6651 is operating in Base Station Emulator mode.
0: OFF
1: ON

E6651_SetBSFullOccupied

Declaration

```
int E6651_SetBSFullOccupied(BYTE MID, int
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [in]: Full Occupied Function's ON or OFF value of E6651(MID) when E6651 is operating in Base Station Emulator mode to be set.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Set Full Occupied Function's ON or OFF of E6651(MID) when E6651 is operating in Base Station Emulator mode.
0: OFF
1: ON

E6651_GetBSSBCMaximumTransmittedPowerBPSK

Declaration

```
int E6651_GetBSSBCMaximumTransmittedPowerBPSK
  (BYTE MID, int * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current Maximum Transmitted Power for BPSK value is in SBC-REQ message from a mobile.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description Retrieve Maximum Transmitted Power for BPSK value is in SBC-REQ message from a mobile.

E6651_GetBSSBCMaximumTransmittedPowerQPSK

Declaration

```
int E6651_GetBSSBCMaximumtransmittedPowerQPSK
  (BYTE MID, int * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current Maximum Transmitted Power for QPSK value is in SBC-REQ message from a mobile.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description Retrieve Maximum Transmitted Power for QPSK value is in SBC-REQ message from a mobile.

E6651_GetBSSBCMaximumTransmittedPower16QAM

Declaration

```
int
  E6651_GetBSSBCMaximumTransmittedPower16QAM(BY
  TE MID, int * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [out]: Current Maximum Transmitted Power for 16QAM value is in SBC-REQ message from a mobile.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description Retrieve Maximum Transmitted Power for 16QAM value is in SBC-REQ message from a mobile.

E6651_GetBSSBCMaximumTransmittedPower64QAM

Declaration
```
int E6651_GetBSSBCMaximumTransmittedPower64QAM
  (BYTE MID, int * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Maximum Transmitted Power for 64QAM value is in SBC-REQ message from a mobile.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Maximum Transmitted Power for 64QAM value is in SBC-REQ message from a mobile.

E6651_GetBSHOOperatorID

Declaration
```
int E6651_GetBSHOOperatorID(BYTE MID, int *
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Operator ID of MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Retrieve Operator ID of MOB_NBR-ADV message.

E6651_SetBSHOOperatorID

Declaration
```
int E6651_SetBSHOOperatorID(BYTE MID, int
  Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Operator ID of MOB_NBR-ADV message to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description Set Operator ID of MOB_NBR-ADV message.

E6651_GetBSHONeighborBSID

Declaration

```
int E6651_GetBSHONeighborBSID(BYTE MID, int
  Index, int * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [out]:	Current Neighbor BSID of the index in MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve Neighbor BSID of the index is made from MOB_NBR-ADV message. There are 3 Neighbors and the index starts from 0.

E6651_SetBSHONeighborBSID

Declaration

```
int E6651_SetBSHONeighborBSID(BYTE MID, int
  Index, int Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [in]:	Neighbor BSID of the index in MOB_NBR-ADV message to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Set Neighbor BSID of the index is made from MOB_NBR-ADV message.

E6651_GetBSHONeighborDCDCC

Declaration

```
int E6651_GetBSHONeighborDCDCC(BYTE MID, int
  Index, int * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Index [in]: The value of index
0: Neighbor #1
1: Neighbor #2
2: Neighbor #3

Value [out]: Current Neighbor DCD Configuration Change Count value of the index in MOB_NBR-ADV message.

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Retrieve Neighbor DCD Configuration Change Count value of the index is made from MOB_NBR-ADV message (0 ~ 15).

E6651_SetBSHONeighborDCDCC

Declaration

```
int E6651_SetBSHONeighborDCDCC(BYTE MID, int
  Index, int Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Index [in]: The value of index
0: Neighbor #1
1: Neighbor #2
2: Neighbor #3

Value [in]: Neighbor DCD Configuration Change Count value of the index in MOB_NBR-ADV message to be set

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Set Neighbor DCD Configuration Change Count value of the index is made from MOB_NBR-ADV message (0 ~ 15).

E6651_GetBSHONeighborUCDCC

Declaration

```
int E6651_GetBSHONeighborUCDCC(BYTE MID, int
  Index, int * Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Index [in]: The value of index
0: Neighbor #1
1: Neighbor #2
2: Neighbor #3

Value [out]: Current Neighbor UCD Configuration Change Count value of the index in MOB_NBR-ADV message.

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Retrieve Neighbor UCD Configuration Change Count value of the index is made from MOB_NBR-ADV message (0 ~ 15).

E6651_SetBSHONeighborUCDCC

Declaration

```
int E6651_SetBSHONeighborUCDCC(BYTE MID, int
  Index, int Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Index [in]: The value of index
0: Neighbor #1
1: Neighbor #2
2: Neighbor #3

Value [in]: Neighbor UCD Configuration Change Count value of the in MOB_NBR-ADV message to be set.

Return Value

Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description

Set Neighbor UCD Configuration Change Count value of the index is made from MOB_NBR-ADV message (0 ~ 15).

E6651_GetBSHONeighborFrequency

Declaration

```
int E6651_GetBSHONeighborFrequency(BYTE MID, int
  Index, double * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [out]:	Current Neighbor Frequency value of the index in MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve Neighbor Frequency value of the index is made from MOB_NBR-ADV message.

E6651_SetBSHONeighborFrequency

Declaration

```
int E6651_SetBSHONeighborFrequency(BYTE MID, int
  Index, double Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [in]:	Neighbor Frequency value of the index in MOB_NBR-ADV message to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Set Neighbor Frequency value of the index is made from MOB_NBR-ADV message.

E6651_GetBSHONeighborPermutationBase

Declaration

```
int E6651_GetBSHONeighborPermutationBase(BYTE
  MID, int Index, int * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [out]:	Current Neighbor Permutation Base value of the index in MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve Neighbor Permutation Base value of the index is made from MOB_NBR-ADV message.

E6651_SetBSHONeighborPermutationBase

Declaration

```
int E6651_SetBSHONeighborPermutationBase(BYTE
  MID, int Index, int Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Index [in]:	The value of index 0: Neighbor #1 1: Neighbor #2 2: Neighbor #3
Value [in]:	Neighbor Permutation Base value of the index in MOB_NBR-ADV message to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Set Neighbor Permutation Base value of the index is made from MOB_NBR-ADV message.

E6651_GetBSHOMOB_NBR_ADVEnable

Declaration
```
int E6651_GetBSHOMOB_NBR_ADVEnable(BYTE MID, int * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Transmission mode of MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve Transmission mode of MOB_NBR-ADV message.
0: Stop
1: Start

E6651_SetBSHOMOB_NBR_ADVEnable

Declaration
```
int E6651_SetBSHOMOB_NBR_ADVEnable(BYTE MID, int Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [in]:	Transmission mode of MOB_NBR_ADV message to be set.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Set Transmission mode of MOB_NBR_ADV message.
0: Stop
1: Start

E6651_GetBSHOMOB_NBR_ADVRate

Declaration
```
int E6651_GetBSHOMOB_NBR_ADVRate(BYTE MID, int * Value);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
Value [out]:	Current Transmission Rate value of MOB_NBR-ADV message.

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Retrieve Transmission Rate of MOB_NBR-ADV message. Transmit once per a Value Frame.

E6651_SetBSHOMOB_NBR_ADVRate

Declaration

```
int E6651_SetBSHOMOB_NBR_ADVRate(BYTE MID, int
  Value);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Value [in]: Transmission Rate value of MOB_NBR_ADV message to be set.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Set Transmission Rate value of MOB_NBR-ADV message. Transmit once per a Value Frame.

E6651_SendBSHOMOB_SCN_RSP

Declaration

```
int E6651_SendBSHOMOB_SCN_RSP(BYTE MID);
```

Parameter

MID [in]: E6651's equipment number to be controlled remotely.

Return Value

Success: STATE_SUCCESS(1)

Failure: STATE_ERROR(-1)

Description

Transmit MOB_SCN-RSP Message.

Tools Functions

E6651_BSAddBurst

Declaration

```
int E6651_BSAddBurst(BYTE MID, WORD CID, BYTE
  DIUC, Char * MSG);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
CID [in]:	CID to be added in Burst
DIUC [in]:	DIUC to be added in Burst
MSG [in]:	Message to be added in Burst

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Used when user want to generate an arbitrary Burst in E6651(MID)'s Downlink.

E6651_WaitMessage

Declaration

```
int E6651_WaitMessage(BYTE MID, BYTE MSG, DWORD
  TimeOut);
```

Parameter

MID [in]:	E6651's equipment number to be controlled remotely.
MSG [in]:	MAC Message Type number to wait until receive it.
TimeOut [in]:	Time duration to wait the message (ms)

Return Value

Success:	STATE_SUCCESS(1)
Failure:	STATE_ERROR(-1)

Description

Wait until E6651(MID) receive designated MAC Message Type.

Functions used in V1.5

E6651_SetBSPayloadLen

Declaration `int E6651_SetBSPayloadLen(BYTE MID, int Value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: E6651 Base Station Emulation Payload length value to be set. (0, 1, 2, etc.)

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set E6651(MID) Base Station Emulation Payload Length value.
If (Value = 0) then Value:= 288
if (Value = 1) then Value:= 576
if (Value = 2) then Value:= 1488 else Value:= 288;

E6651_SetBSDLTest

Declaration `int E6651_SetBSDLTest(BYTE MID, int Value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: DL Test Mode of E6651(MID) Base Station Emulator to be set.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set DL Test Mode of E6651(MID) Base Station Emulator.
0: Off
1: On

E6651_SetBSULTest

Declaration `int E6651_SetBSULTest(BYTE MID, int Value);`

Parameter
MID [in]: E6651's equipment number to be controlled remotely.
Value [in]: UL Test Mode of E6651(MID) Base Station Emulator to be set.

Return Value
Success: STATE_SUCCESS(1)
Failure: STATE_ERROR(-1)

Description Set DL Test Mode of E6651(MID) Base Station Emulator.
0: Off
1: On

Appendix A - Network Entry Procedure

A Mobile WiMAX subscriber station (SS) executes the Network Entry procedure in order to begin two way communications with a base station.

Network Entry Procedure Overview

A Mobile WiMAX subscriber station (SS) executes the Network Entry procedure in order to begin two way communications with any base station, including the E6651A Mobile WiMAX Test Set. The SS executes this procedure after acquiring relevant information from broadcast messages.

The initial registration procedure of the SS can be summarized as follows:

Scanning for Downlink Channel

The SS scans a Mobile WiMAX base station (BS) signal and determines whether the signal level is strong enough to provide adequate service quality.

Downlink Synchronization

The SS executes a synchronization procedure to obtain downlink channel information from the BS.

After acquiring synchronization, the SS obtains downlink and uplink information from the BS.

Obtaining Uplink Parameters

Using downlink information, the SS obtains uplink information for the Ranging and Registration procedures.

Execution of Ranging Procedure

Using the acquired uplink information, the SS executes the Ranging procedure which includes the adjustment of time, frequency and power parameters to meet the uplink burst allocation.

The Ranging algorithm used in Mobile WiMAX 1.A is the “CDMA Initial Ranging” algorithm. When this algorithm is used, the SS transmits an “Initial Ranging Code” in an Initial Ranging Region allocated by the BS.

Negotiation of Basic Capabilities

The SS negotiates physical layer properties and capabilities with the BS.

Registration With the BS

The SS provides information to the network and receives information from the network in order to register with the Mobile WiMAX service.

The Network Entry procedure is complete at the end of the Registration Procedure. From this point on, the SS executes a Periodic Ranging procedure to assist in maintaining a reliable connection with the network.

The flow diagram in Figure 44 depicts the general Network Entry procedure of the SS. “Network Entry Procedure Detail" on page 174 provides detail on each step in the Network Entry procedure.

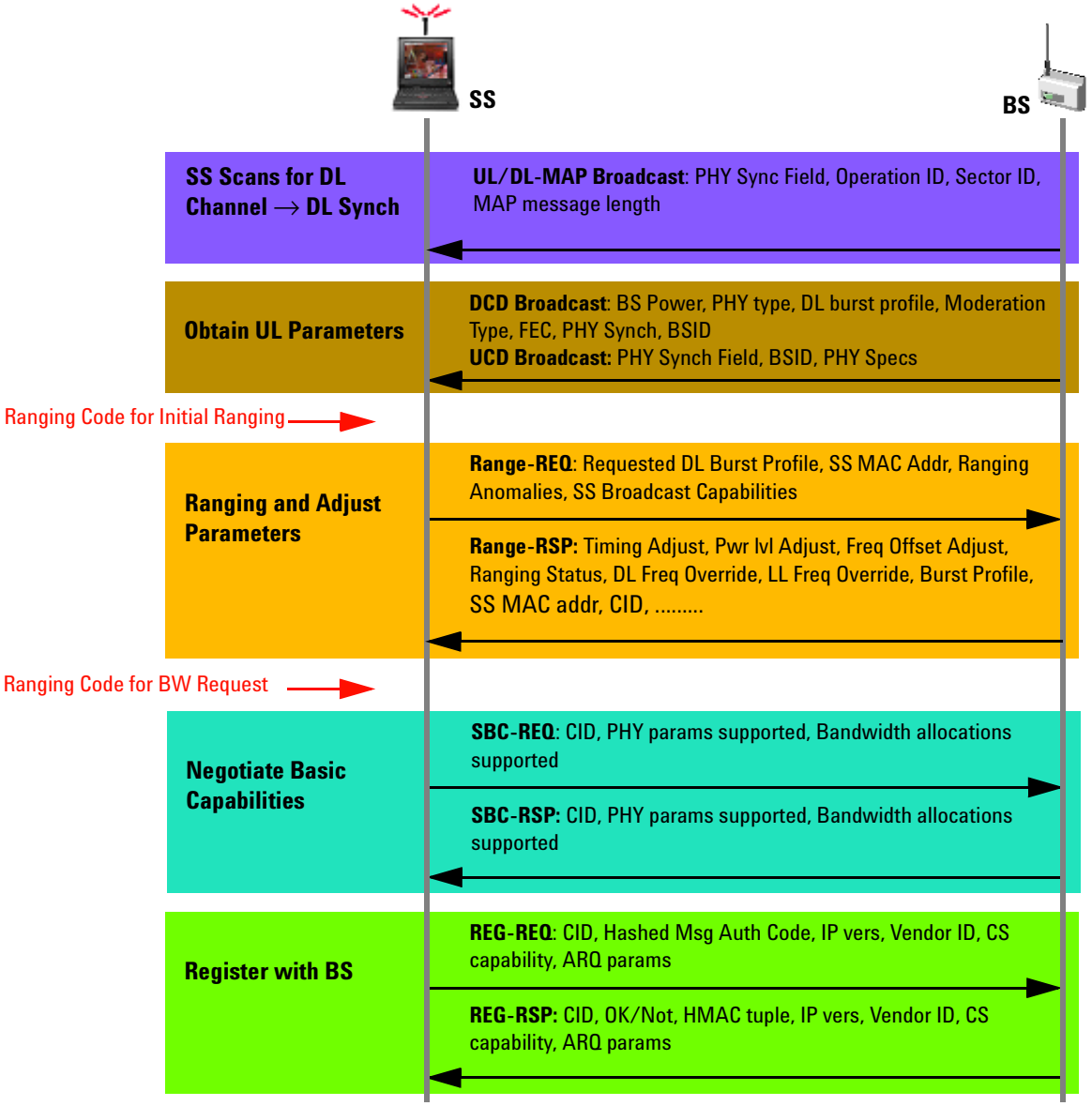


Figure 44 Initial Network Entry Procedure of the SS

Network Entry Procedure Detail

The Test Set engages in the Network Entry procedure when establishing two way communications with the SS. The messages transmitted during the each phase of the Network Entry procedure are displayed in the Measurement Window when the Test Set is operating in BSE mode. The following sections describe the detailed steps in the procedure, and show the operation of the Test Set for each step.

Downlink Synchronization and Uplink Parameter Information

In the Downlink Synchronization procedure, the BS transmits an OFDMA frame to the SS containing the DL-MAP information. The SS uses the DL-MAP information to obtain UL-MAP, DCD and UCD information. The OFDMA frame structure, including the mapping of each information type, is shown in Figure 45.

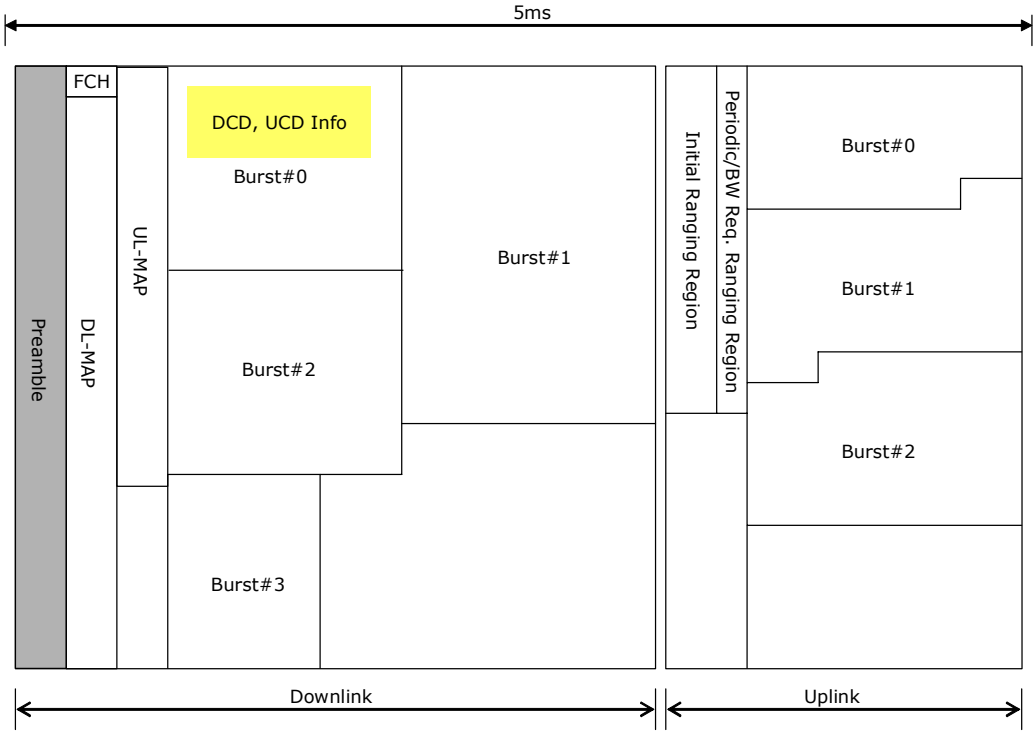


Figure 45 OFDMA Frame Structure

The Test Set periodically broadcasts DCD and UCD information to the SS to provide downlink synchronization and uplink parameter information at the designated

frequency. The broadcast operation of the product for downlink synchronization and uplink parameter information is shown in Figure 46.

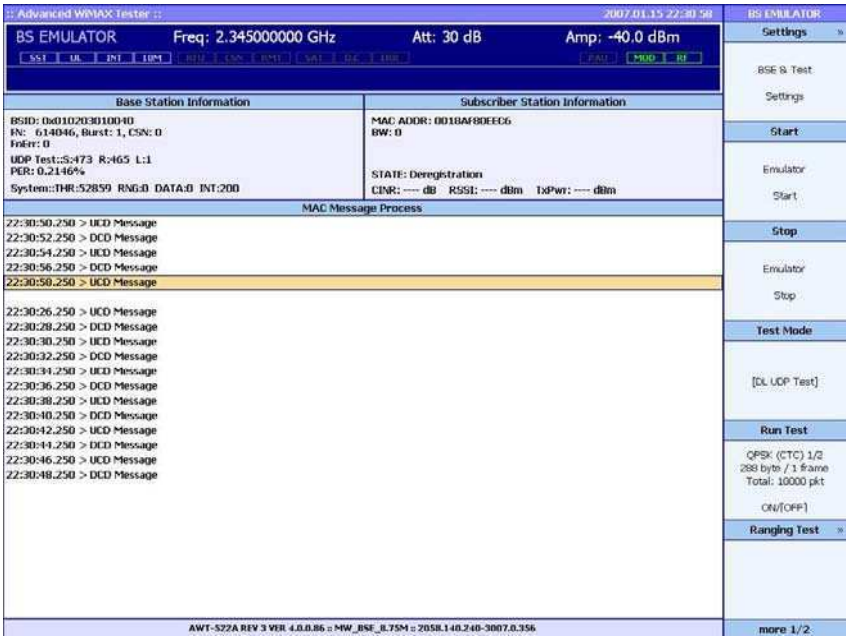


Figure 46 UCD and DCD Message Transmission for Downlink Synchronization and Uplink Parameter Information

The DCD and UCD messages shown in Figure 46 provide the following types of information:

DCD Message (Downlink Channel Descriptor)

- BS Power
- PHY Type
- DL Burst Profile
- Modulation Type
- FEC
- PHY Sync
- BSID Information

UCD Message (Uplink Channel Descriptor)

- PHY Sync field
- BSID
- PHY Specification

Initial Ranging Procedure Execution and Basic Capabilities Negotiation

Using the initial ranging region information and initial ranging code information obtained from the UL-MAP and UCD messages, the SS attempts the Contention Based Initial Ranging procedure.

The E6651A measures the quality of the initial ranging code. Provided that the code quality exceeds a specified threshold, the unit sends the ranging code, time offset, frequency offset and power offset through the RNG-RSP message and provides CDMA Allocation IE information through UL-MAP. CDMA Allocation IE information is used in the transmission of the RNG-REQ message.

When the RNG-RSP message and CDMA Allocation IE is received from the Test Set, the SS sends the RNG-REQ message in the uplink region specified in the CDMA Allocation IE.

Upon successful exchange of the RNG-REQ and RNG-RSP messages, the physical connection is established between the SS and the Test Set. At this point, the SS and the Test Set share information including the Basic CID and the Burst Profile.

In the next step of the Initial Ranging Procedure, known as “Basic Capabilities Negotiation”, the SS sends the BW Request message to the Test Set. Upon receipt of the BW Request message, the Test Set sends an ALLOC UL-MAP message, containing information about the bandwidth that the unit has allocated to the SS.

The SS provides its PHY and Bandwidth Allocation information by sending SBC-REQ messages using the uplink resources allocated by the Test Set. After receiving the SBC-REQ message from the SS, the Test Set responds with an SBC-RSP message to the SS. The SBC-REQ and SBC-RSP message pair constitutes the negotiation of basic physical layer properties between the SS and the BS.

The Initial Ranging Procedure and Negotiation of Basic Capabilities as seen in the Measurement Window of the E6651A are shown in Figure 47.

Figure 47 Measurement Window Showing the Initial Ranging Procedure and Negotiation of Basic Capabilities

The messages seen in the Initial Ranging Procedure and Basic Capabilities Negotiation of Figure 47 are:

Initial Ranging Region Information

The SS sends this message to provide ranging detection to the BS using information received in the Initial Ranging Region. This message serves as the initial request by the SS for uplink resources from the BS. Information contained in this message includes:

- **Frame Number (FN):** The frame number at which the BS detected Initial Ranging
- **Detected Position (DP):** The time at which the BS detected Initial Ranging
- **Ranging Code (RC):** The Ranging Code value sent by the BS and then used by the SS to gain access to allocated uplink resources
- **Frequency Offset (FO):** The Frequency Offset detected between the SS and the BS during Initial Ranging
- **Ranging Matching Rate (RM):** The ratio of valid Initial Ranging attempts to total ranging attempts

- **Ranging Power (RP):** The power information for the Initial Ranging Code

This is the response to the Ranging Code or RNG-REQ message sent from the BS to the SS. Information contained in this message includes:

- Timing Adjustment
- Power Level Adjustment
- Frequency Offset Adjustment
- Ranging Status
- DL Frequency Override
- UL Frequency Override
- Burst Profile
- SS MAC Address
- CID

RNG-REQ MAC Management Message

This is the ranging request message sent from the SS to the BS during the ranging procedure. Information contained in this message includes:

- Requested DL Burst Profiles
- SS MAC Address
- Ranging Anomalies
- SS Broadcast Capabilities

BW-REQ Message

This is the bandwidth request message sent from the SS to the BS.

ALLOC UL-MAP Message

The BS sends this message to the SS to provide information about the bandwidth allocated to the SS. Information contained in this message includes:

- **CID:** Requested uplink region's connection ID
- **BR:** Allocated bandwidth size in bytes
- **SL:** The number of slots corresponding to the allocated bandwidth size

SBC-REQ MAC Management Message

The SS sends this message to the BS to initiate the negotiation of basic physical layer properties. Information requested in this message includes:

- CID
- Supported PHY parameters
- Supported Bandwidth allocations

SBC-RSP MAC Management Message

The BS sends this message to the SS in response to the request for basic physical layer properties. Information provided in this message includes:

- CID
- Supported PHY parameters
- Supported Bandwidth allocations

Registration Procedure

After the negotiation of physical layer properties, the SS enters the registration procedure with the BS using the REG-REQ / REG-RSP message pair.

Using its primary CID, the SS sends a BW-REQ message to request a new uplink region for the transmission of user traffic. The BS responds with a ALLOC UL-MAP message containing updated bandwidth allocation and primary CID information.

Once the SS has received this information, it proceeds through the registration process consisting of the REG-REQ sent to the BS, followed by the receipt of a REG-RSP from the BS. The completion of the registration process marks the end of the Network Entry procedure.

The Registration Procedure as seen in the Measurement Window of the E6651A is shown in Figure 48.

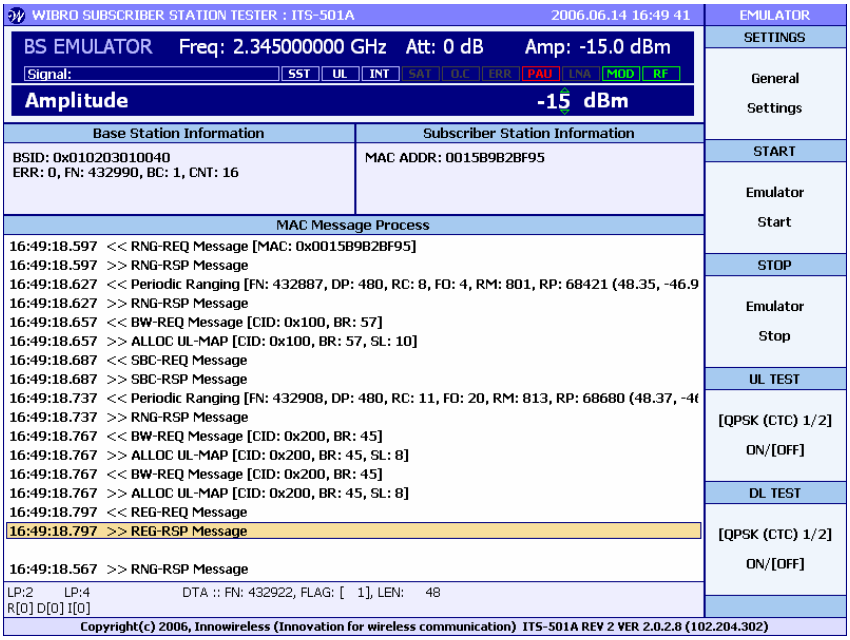


Figure 48 Uplink Region Allocation for Basic CID and Primary CID for Registration Procedure

The messages seen in the Registration Procedure of Figure 48 are:

REG-REQ MAC Management Message

The SS sends this message to request registration with the BS. Information provided in this message includes:

- Secondary CID

- Hashed Message Auth Code
- IP Version
- Vendor ID
- CS Capability
- ARQ Parameter

REG-RSP MAC Management Message

The BS sends this message to the SS in response to the registration request. Information provided in this message includes:

- Secondary CID
- OK/Not
- HMAC Tuple
- IP Version
- Vendor ID
- CS Capability
- ARQ Parameter

Periodic Ranging for Connection Maintenance

After the Network Entry procedure has been completed, the SS engages in the Periodic Ranging procedure to assist in maintaining a reliable connection with the network.

The Periodic Ranging Procedure as seen in the Measurement Window of the E6651A is shown in Figure 49.

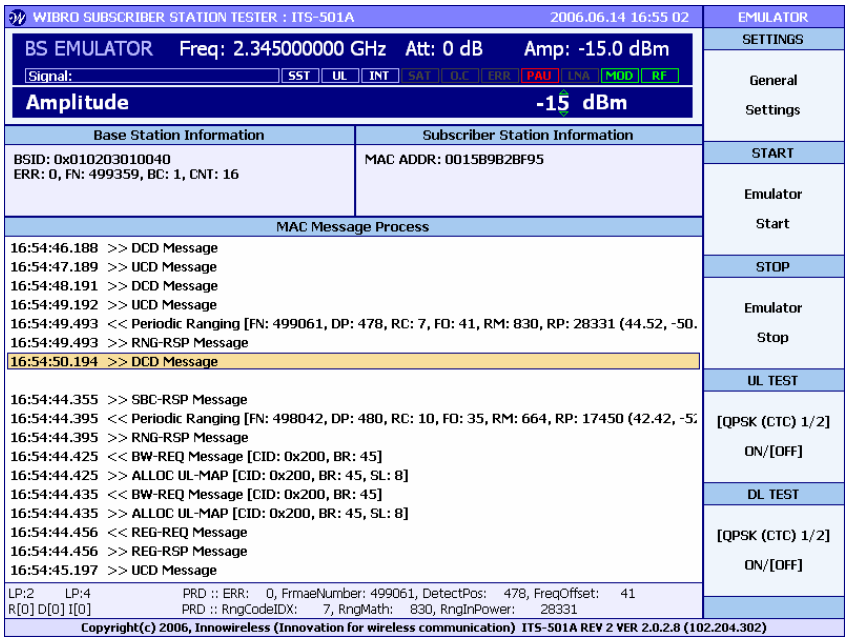


Figure 49 Periodic Ranging Procedure

The message used in the Periodic Ranging Procedure of Figure 49 is:

Periodic Ranging Region Information

The SS sends this message to provide ranging detection to the BS for connection maintenance using information received in the Periodic Ranging Region. This message serves as a request by the SS for continued uplink resources from the BS. Information contained in this message includes:

- **Frame Number (FN):** The frame number at which the BS detected Periodic Ranging
- **Detected Position (DP):** The time at which the BS detected Periodic Ranging
- **Ranging Code (RC):** The Ranging Code value sent by the BS and then used by the SS to gain access to allocated uplink resources
- **Frequency Offset (FO):** The Frequency Offset detected between the SS and the BS during Periodic Ranging

- **Ranging Matching Rate (RM):** The ratio of valid Periodic Ranging attempts to total ranging attempts
- **Ranging Power (RP):** The power information for the Periodic Ranging Code

REP-REQ message Setting

The E6651A Test Set can control the transmission of REP-REQ message. The message is sent from BS to SS and is used to get the SS's status information by the BS. Two control parameters are provided.

- REP-REQ msg. Enable/Disable Setting
- REP-REQ msg. Transmission rate control : determines time interval between subsequent messages.

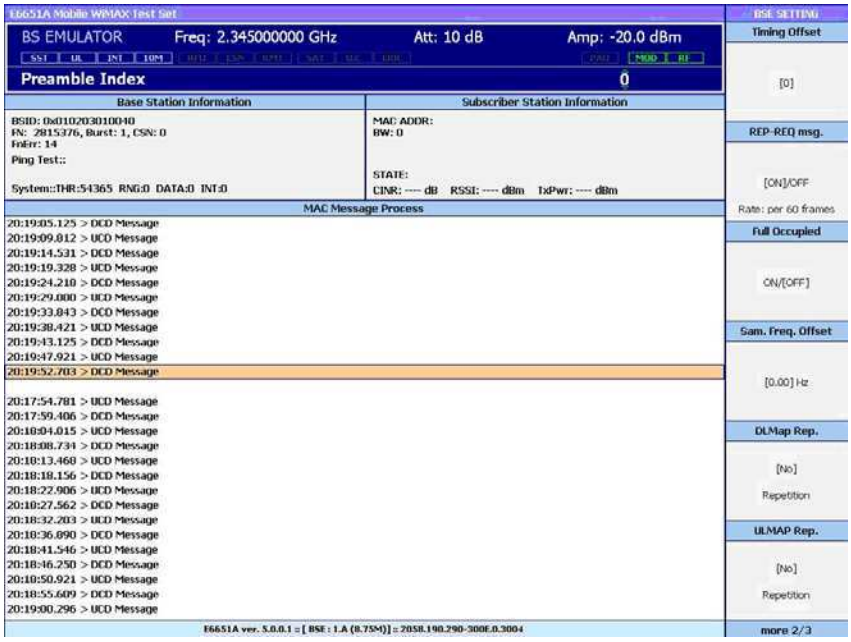


Figure 50 BSE Setting function - A

Full Occupied Setting

The E6651A provides the function to fill unallocated sub-channels and symbols of downlink frame with arbitrary data to support mobile WiMAX RCT test function. The "Full Occupied" setting is a toggle function.

- **On:** fill unallocated data region of downlink frame with arbitrary data bits.
- **Off:** don't fill unallocated data region of downlink frame.

Repetition Setting

The E6651A can repeat UL MAP and DL MAP information in downlink frames. The repetition control function is provided as shown in Figure 50 on page 183. And the DL/UL user data repetition function is provided also as shown in Figure 51.

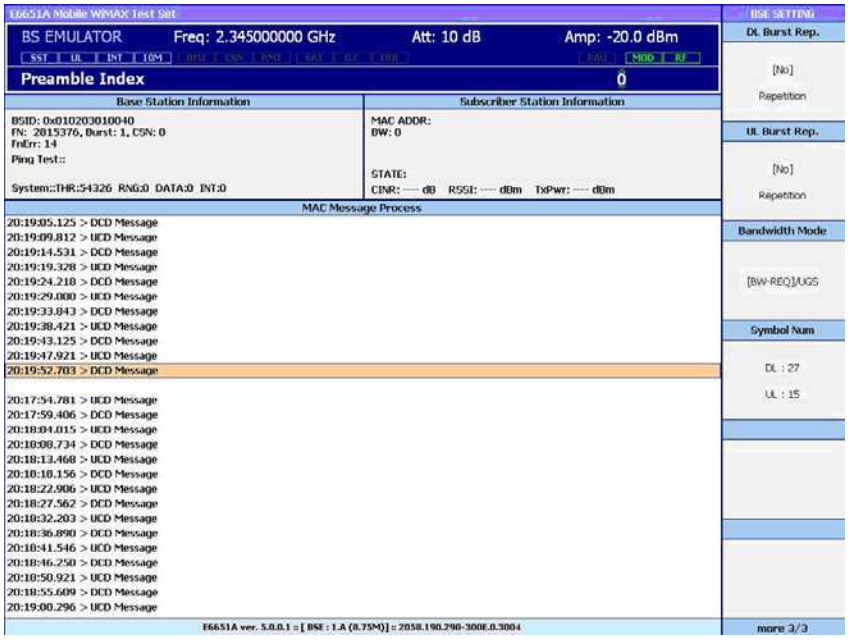


Figure 51 BSE Setting function - B

Bandwidth Request Mode Setting

The E6651A provides two bandwidth allocation algorithms to allocate uplink data region for SS's data transmission. They are BW-REQ and UGS. The BW-REQ algorithm allocates uplink data region based on bandwidth request from SS. The UGS algorithm allocates uplink data region amount to "UL BW Size" . You can select between the two algorithms as shown in Figure 51.

n

Symbol Number

Allocated number of symbols between uplink and downlink can be adjusted in the E6651A as shown in Figure 51. You can adjust uplink and downlink symbol ratio.

Profile Selection

The Profiles are displayed in 2 lists, the **File List** and the **Memory List**. The **Memory List** shows the profiles loaded in FPGA Memory. To be used, a profile must be loaded into FPGA memory.

The required profile can be selected for use as follows:

1 Press **System > More >** **Profile Manager** to display the Profile Manager screen as shown in Figure 52.

2 Press **Memory List** and use the knob, arrow or numeric entry keys to highlight the required profile in the Memory List.

3 Press **Activate Profile** to test the SS using the selected profile.

Figure 52 Certification Profile Selection window

4 If the required profile is not in the **Memory List** press **File LIst** and highlight a **Profile Type** in the **File List**.

5 Press **Memory List** and use the knob, arrow or numeric entry keys to highlight a free **Bank** in the **Memory List**.

6 Press **Upload Profile** to upload the selected profile into FPGA memory. The profile name is displayed in the Bank list.

7 Highlight and activate the profile for use as described in steps 2 and 3.

A Appendix A - Network Entry Procedure