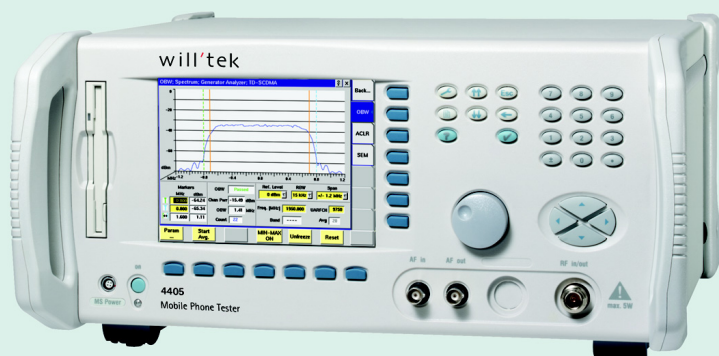


will'tek

Willtek 4400

Mobile Phone Tester



4450 TD-SCDMA System Option
Non-Call Mode
user's guide
version 6.10

Notice Every effort was made to ensure that the information in this document was accurate at the time of printing. However, information is subject to change without notice, and Willtek reserves the right to provide an addendum to this document with information not available at the time this document was created.

Copyright © Copyright 2005 Willtek Communications GmbH. All rights reserved. Willtek and its logo are trademarks of Willtek Communications. All other trademarks and registered trademarks are the property of their respective owners. No part of this guide may be reproduced or transmitted electronically or otherwise without written permission of the publisher.

Trademarks Willtek is a trademark of Willtek Communications GmbH in Germany and other countries.

Specifications, terms and conditions are subject to change without notice. All trademarks and registered trademarks are the property of their respective companies.

Ordering information This guide is issued as part of the **Willtek 4450 TD-SCDMA System Option Non-Call Mode**. The ordering number for a published guide is M 294 251. The ordering number for the product depends on the model of the 4400 Series as follows:

Table 1 Order numbers for the 4400 Series

| Description | Order number |
|---|---------------------|
| Willtek 4403 Mobile Phone Tester | M 101 105 |
| Willtek 4405 Mobile Phone Tester | M 101 104 |
| 4479 WCDMA/TD-SCDMA Hardware Option | M 248 690 |
| 4450 TD-SCDMA System Option Non-Call Mode SW-4400 | M 897 255 |
| 4479 WCDMA/TD-SCDMA Hardware Upgrade Package for 440x with serial number ≤ 0511000 | M 248 683 |
| 4479 WCDMA/TD-SCDMA Hardware Upgrade Package for 440x with serial number = 0611xxx/0711xxx | M 248 684 |
| 4479 WCDMA/TD-SCDMA Hardware Upgrade Package for 440x with serial number ≥ 0811000 | M 248 685 |

Compliance statements This manual refers to the TD-SCDMA Non-Call Mode System Option software. For hardware compliance with any national regulations, please refer to the getting started manual that was delivered with your 4400. The specifications for the product can be found in the data sheet.

Table of Contents

| | | |
|-------------------------|--|-----------|
| About This Guide | | xi |
| | Index of the 4450 TD-SCDMA System Option Non-Call Mode | xii |
| | Purpose and scope..... | xii |
| | Assumptions | xii |
| | Related information | xiii |
| | Technical assistance..... | xiii |
| | Conventions | xiii |

| | | |
|---------------------|------------------------------------|-----------|
| Safety Notes | | xv |
| | Safety class..... | xvi |
| | Before startup..... | xvi |
| | During test | xvi |
| | During maintenance and repair..... | xvi |
| | Shutdown when defective..... | xvii |

| | | |
|------------------|--------------------------------------|----------|
| Chapter 1 | Overview | 1 |
| | About the 4400 | 2 |
| | What's new in version 6.10..... | 3 |
| | Features and capabilities..... | 3 |
| | Connecting test leads..... | 4 |
| | Crash course | 4 |
| | Cabling | 4 |
| | Cable connection | 5 |
| | Air connection..... | 5 |
| | Using the front and rear panels..... | 6 |
| | Front panel | 7 |
| | Rear panel | 8 |
| | Connectors..... | 9 |
| | Connectors on the front panel | 9 |
| | Connectors on the rear panel | 10 |

| | |
|---|----|
| The keys on the front panel | 14 |
| The softkeys | 15 |
| The marker tabs | 16 |
| The function keys | 16 |
| The numeric keys | 18 |
| The cursor keys | 18 |
| The selection key | 18 |
| Keyboard mapping table | 19 |
| Menus and levels | 20 |
| The areas of a test menu | 21 |
| Menu fields | 22 |
| Entry fields | 22 |
| Display fields | 25 |
| Help on Help | 26 |
| Navigating help | 27 |
| Using the online help on an external PC | 28 |
| Installing the online help on a PC | 28 |
| Navigating through the help pages | 28 |
| Accessories and options | 29 |
| Expanding the 4400's measurement capabilities | 29 |
| Little aids for speeding up tests | 31 |
| Rack installation, cases | 35 |

Chapter 2

| | |
|---|-----------|
| TD-SCDMA Non-Call Mode | 37 |
| Overview | 38 |
| RF Generator | 39 |
| What it does | 40 |
| Signal Generator parameters | 40 |
| RF Analyzer for modulation quality | 41 |
| Modulation quality parameters | 42 |
| Reading the results | 42 |
| Constellation Display | 44 |
| RF Analyzer for power | 45 |
| Power parameters | 45 |
| Reading the results | 46 |
| RF Analyzer for spectrum | 47 |
| OBW (Occupied Bandwidth) | 47 |
| Parameters | 47 |
| Reading the results | 48 |
| ACLR (Adjacent Channel Leakage Power Ratio) | 49 |
| Parameters | 49 |
| Reading the results | 50 |
| SEM (Spectrum Emission Mask) | 51 |
| Parameters | 51 |
| Reading the results | 52 |
| Softkeys of the RF Analyzer | 53 |
| System parameters | 54 |
| RF Gen. | 54 |
| Slot 1 | 55 |
| DwPTS | 55 |
| Slot 2 - 6 | 55 |
| Scrambling Code | 56 |

| | | |
|------------------|--|-----------|
| | Measurement limits | 56 |
| | Coupling Loss | 58 |
| | How to activate a previously stored coupling loss definition | 59 |
| | How to store a coupling loss definition on the 4400 | 59 |
| | How to edit a coupling loss definition already stored on the 4400 .. | 60 |
| | Entering coupling loss values from the Manual Input menu..... | 61 |
| | TD-SCDMA basics | 62 |
| | TD-SCDMA frequency bands..... | 62 |
| | Channel arrangement..... | 62 |
| | UARFCN | 63 |
| | UE power classes | 63 |
| Chapter 3 | Tools | 65 |
| | Overview | 66 |
| | Configuration | 66 |
| | Access to the configuration menus..... | 66 |
| | Setup..... | 67 |
| | Settings of the Setup menu..... | 67 |
| | I/O configuration | 68 |
| | IEEE 488.2 port settings | 68 |
| | TCP/IP port settings..... | 69 |
| | TCP/IP Troubleshooting..... | 71 |
| | Parallel port settings..... | 72 |
| | Options..... | 73 |
| | Basic info area | 73 |
| | Options..... | 74 |
| | Installing additional options | 75 |
| | Service and software update..... | 76 |
| | Changing the version of the system software..... | 76 |
| | The softkeys of this menu | 77 |
| | Initiating a software update from a remote PC | 78 |
| | Utilities..... | 79 |
| | Access to the Utilities menus | 79 |
| | I/O trace for GPIB communications..... | 79 |
| | The softkeys of this menu | 80 |
| | Info trace | 80 |
| | The softkeys of this menu | 81 |
| | MS power supply | 81 |
| | MS Power Supply Option | 82 |
| | Current Measurement Option..... | 83 |
| | Spectrum measurements..... | 85 |
| Chapter 4 | RAPID! | 87 |
| | Overview | 88 |
| | Using RAPID!..... | 88 |
| | Introduction | 89 |
| | RAPID! = BASIC + SCPI..... | 89 |
| | Entering and exiting the RAPID! environment | 89 |
| | Marker tabs of the RAPID! environment..... | 90 |

| | |
|--|-----|
| File menu | 91 |
| File name – directory – selection area | 91 |
| Browser area | 92 |
| RAPID! basic file area | 93 |
| Softkeys of the file menu..... | 93 |
| Edit menu | 95 |
| Edit area | 96 |
| Status area | 96 |
| Softkeys of the Edit menu..... | 97 |
| Search/Replace menu..... | 97 |
| Typing tabs | 98 |
| Run menu | 99 |
| Output area..... | 100 |
| Softkeys of the Run menu..... | 101 |
| Debug menu | 101 |
| Source area | 102 |
| Variable area | 102 |
| Softkeys of the Debug menu..... | 104 |
| RAPID! syntax..... | 105 |
| General syntax | 105 |
| BASIC form | 105 |
| Program and line formats | 105 |
| Basic rules | 106 |
| Syntax check..... | 106 |
| Notation..... | 106 |
| Program lines | 107 |
| Variables | 108 |
| Constants..... | 109 |
| Operators..... | 110 |
| Expressions | 111 |
| Numeric expressions..... | 112 |
| String expressions | 112 |
| Boolean expressions | 112 |
| Commands | 113 |
| Overview | 113 |
| General commands | 114 |
| CHAIN..... | 114 |
| END | 115 |
| LET | 115 |
| REM or ' (Comments) | 115 |
| STOP | 116 |
| WAIT | 116 |
| Screen commands | 116 |
| CLS..... | 117 |
| INPUT | 118 |
| LOCATE..... | 118 |
| PRINT (OUTPUT) | 119 |
| SOFTKEYS..... | 120 |
| TEXTATTR | 120 |

| | |
|---|-----|
| Commands related to variables, procedures and functions | 121 |
| DIM | 121 |
| ERASE | 121 |
| FUNCTION – EXIT FUNCTION – END FUNCTION | 122 |
| GLOBAL | 123 |
| SUB – EXIT SUB – END SUB – CALL | 123 |
| VARIABLE | 124 |
| Control commands | 125 |
| DO ... LOOP, WHILE, UNTIL | 125 |
| FOR ... NEXT | 126 |
| IF ... THEN, ELSE, ELSEIF, END IF | 127 |
| SELECT CASE, CASE ELSE, END SELECT | 128 |
| Branch commands | 129 |
| ERROR | 129 |
| GOSUB ... RETURN | 129 |
| GOTO | 130 |
| ON ERROR, RESUME | 130 |
| Commands for input/output handling | 132 |
| CLOSE | 132 |
| INPUT | 132 |
| OPEN (on files) | 133 |
| OPEN (on communication ports) | 134 |
| PRINT or OUTPUT | 137 |
| Functions | 138 |
| Numeric functions | 139 |
| BIN | 139 |
| BIN\$ | 139 |
| CINT | 140 |
| HEX | 140 |
| HEX\$ | 140 |
| OCT | 141 |
| OCT\$ | 141 |
| VAL | 141 |
| VAL\$ | 142 |
| String functions | 142 |
| ASC | 142 |
| CHR\$ | 143 |
| INSTR | 143 |
| LEN | 143 |
| LEFT\$ | 144 |
| MID\$ | 144 |
| RIGHT\$ | 144 |
| SPACE\$ | 145 |

| | |
|------------------------------------|-----|
| I/O functions..... | 145 |
| CHDIR..... | 146 |
| CLOCK..... | 146 |
| CURDIR\$..... | 146 |
| DATE\$..... | 146 |
| DIR\$..... | 147 |
| EOF..... | 147 |
| EVENTWAIT, EVENTSTATUS..... | 148 |
| FREEFILE..... | 149 |
| INKEY..... | 149 |
| INKEYWAIT..... | 150 |
| KILL..... | 150 |
| MKDIR..... | 150 |
| NAME..... | 151 |
| RMDIR..... | 151 |
| SHELL..... | 151 |
| TIME\$..... | 152 |
| Functions for error handling..... | 152 |
| ERR..... | 152 |
| ERL..... | 153 |
| ERF\$..... | 153 |
| Mathematical functions..... | 153 |
| ABS..... | 154 |
| LOG, LGT..... | 154 |
| RND, RANDOMIZE..... | 154 |
| SIN, COS, TAN, ATAN..... | 155 |
| SQRT..... | 155 |
| SGN..... | 155 |
| Tables..... | 156 |
| RAPID! commands and functions..... | 156 |
| Syntax errors..... | 159 |
| Runtime errors..... | 163 |

Chapter 5

| | |
|---------------------------------|------------|
| SCPI | 167 |
| Overview..... | 168 |
| What SCPI is..... | 168 |
| Structure..... | 169 |
| Syntax and notation..... | 169 |
| Compound commands..... | 170 |
| Parameters..... | 171 |
| Queries..... | 171 |
| Common commands..... | 171 |
| SCPI notation..... | 172 |
| SCPI and RAPID!..... | 173 |
| Executing SCPI commands..... | 173 |
| Example 1..... | 173 |
| Example 2..... | 173 |
| Example 3..... | 174 |
| Reading SCPI data..... | 174 |
| Using queries..... | 175 |
| Building queries..... | 175 |
| Event handling – registers..... | 176 |

| | |
|---|-----|
| Programming examples | 176 |
| Standard TX measurements..... | 177 |
| Message exchange..... | 178 |
| Example of a GPIB protocol..... | 179 |
| Command subsystem overview | 180 |
| Using the SCPI commands..... | 180 |
| Schematic view of the subsystems of the 4400..... | 180 |
| Common commands..... | 181 |
| The communication-related subsystems..... | 182 |
| The SYSTem subsystem..... | 182 |
| The STATus subsystem..... | 182 |
| Understanding the STATus subsystem..... | 183 |
| Table of registers | 186 |
| The PROGRAM subsystem - overview..... | 189 |
| The FORMat subsystem - overview | 189 |
| The BS and MS parameter subsystems..... | 190 |
| The CONFigure subsystem | 190 |
| The CALL subsystem | 191 |
| The Measurement subsystems | 191 |
| The MEASure subsystem | 191 |
| :MEASure[:CONTInuous]..... | 192 |
| MEASure:ARRay | 194 |
| :MEAS[:CONT]:BLOCkdata | 196 |
| :MEAS:BLOC:MSP[:CURRent] | 197 |
| :MEASure:::GROup | 198 |
| The FETCh subsystem | 199 |
| FETCh:LAST..... | 199 |
| FETCh:BLOCkdata:::?. | 200 |
| FETCh<{:measProp}>..... | 200 |
| The CALCulate Subsystem | 201 |
| Reading the basic scheme | 202 |
| CALCulate:LIMit | 203 |
| CALCulate:{Statistics} | 205 |
| Using limits | 205 |
| Working with complex limits | 207 |
| Measurement device configuration subsystems | 208 |
| RF measurement devices | 208 |
| AF measurement devices | 208 |
| The RFGenerator subsystem | 208 |
| The RFANalyser subsystem | 209 |
| The RFSpectrum subsystem..... | 209 |
| The AFGenerator subsystem | 209 |
| The AFANalyser subsystem | 209 |
| The MS Power Supply subsystem | 209 |
| SCPI command errors..... | 209 |

Appendix A

| | |
|---|------------|
| SCPI Command Reference | 213 |
| Signaling operation register group | 214 |
| Common commands..... | 214 |
| General common commands..... | 214 |
| Commands affecting the event status register..... | 215 |
| Commands affecting the service register..... | 216 |

| | |
|---------------------------------|-----|
| SYSTem subsystem | 219 |
| STATus subsystem | 227 |
| PROGram subsystem..... | 239 |
| FORMat subsystem | 242 |
| CONFigure subsystem..... | 244 |
| MEASure subsystem | 250 |
| FETCh Subsystem..... | 274 |
| CALCulate Subsystem..... | 286 |
| RFGenerator subsystem | 327 |
| RFANalyser subsystem | 337 |
| RFSPectrum subsystem | 338 |
| AFGenerator subsystem | 340 |
| AFANalyser subsystem | 344 |
| MS Power Supply subsystem | 349 |

| | | |
|-------------------|------------------------------------|------------|
| Appendix B | Warranty and Repair | 351 |
| | Warranty information..... | 352 |
| | Equipment return instructions..... | 353 |

| | |
|----------------------------|------------|
| Publication History | 355 |
|----------------------------|------------|

About This Guide

- ["Index of the 4450 TD-SCDMA System Option Non-Call Mode" on page xii](#)
- ["Purpose and scope" on page xii](#)
- ["Assumptions" on page xii](#)
- ["Related information" on page xiii](#)
- ["Technical assistance" on page xiii](#)
- ["Conventions" on page xiii](#)

Index of the 4450 TD-SCDMA System Option Non-Call Mode

["Safety Notes"](#) – Read before starting!

["Overview"](#) – Crash course, cabling, menus and levels, fields, parts, help on help

["TD-SCDMA Non-Call Mode"](#) – Testing a mobile phone in service mode

Tools – Description of the available tools

["RAPID!"](#) – Introduction, syntax, language elements, error messages

["SCPI"](#) – General introduction and structure

SCPI Command Reference – The commands for the 4450 TD-SCDMA System Option Non-Call Mode

["Warranty and Repair"](#) – Just in case that your unit should need fixing

NOTE

This manual covers all presently known variants of the Willtek 4400 Mobile Phone Tester Series: 4400M, 4400S, 4403, 4405, 4407. The manual refers to all members of the Willtek 4400 series simply as the 4400.

Purpose and scope

The purpose of this guide is to help you successfully use the 4450 TD-SCDMA System Option Non-Call Mode features and capabilities. This guide includes task-based instructions that describe how to install, configure, use, and troubleshoot the 4450 TD-SCDMA System Option Non-Call Mode. Additionally, this guide provides a description of Willtek's warranty, services, and repair information.

Assumptions

This guide is intended for novice, intermediate, and experienced users who want to use the 4450 TD-SCDMA System Option Non-Call Mode effectively and efficiently. We are assuming that you have basic computer and mouse experience and are familiar with basic telecommunication concepts and terminology.

Related information

Use this guide in conjunction with the following information:

4400 getting started manual, order number M 295 011

Technical assistance

If you need assistance or have questions related to the use of this product or call one of Willtek's technical assistance centers. You can also contact Willtek by e-mail at customer.support@willtek.com.

Table 1 Technical assistance centers

| Region | Phone number | Fax number, e-mail address |
|-----------------------------------|--|----------------------------|
| UK | +44 (0) 161 486 3353 | +44 (0) 161 486 3354 |
| Europe, Middle East, Asia, Africa | +49 (0) 89 996 41 386 +49 (0) 89 996 41 227 | +49 (0) 89 996 41 440 |
| Americas | +1 317 595 2021 +1 866 WILLTEK | +1 317 595 2023 |

Conventions

This guide uses naming conventions and symbols, as described in the following tables.

Table 2 Typographical conventions

| Description | Example |
|---|--|
| User interface actions appear in this typeface . | On the Status bar, click Start . |
| Buttons or switches that you press on a unit appear in this TYPEFACE . | Press the ON switch. |
| Code and output messages appear in this typeface. | All results okay |
| Text you must type exactly as shown appears in this typeface . | Type: a : \set.exe in the dialog box. |
| Variables appear in this <typeface>. | Type the new <hostname>. |
| Book references appear in this typeface . | Refer to Newton's Telecom Dictionary |
| A vertical bar means "or": only one option can appear in a single command. | platform [a b e] |

Table 2 Typographical conventions (Continued)

| Description | Example |
|--|-----------------------|
| Square brackets [] indicate an optional argument. | login [platform name] |
| Slanted brackets < > group required arguments. | <password> |

Table 3 Keyboard and menu conventions

| Description | Example |
|---|---|
| A plus sign + indicates simultaneous keystrokes. | Press Ctrl+s |
| A comma indicates consecutive keystrokes. | Press Alt+f,s |
| A slanted bracket indicates choosing a submenu from menu. | On the menu bar, click Start > Program Files. |

Table 4 Symbol conventions






| | |
|---|---|
|  | This symbol represents a general hazard. |
|  | This symbol represents a risk of electrical shock. |
|  | NOTE This symbol represents a note indicating related information or tip. |

Table 5 Safety definitions

| | |
|---|---|
|  | WARNING Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. |
|  | CAUTION Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury. |

Safety Notes

This chapter provides the safety notes for the 4400 Series Mobile Phone Tester. Topics discussed in this chapter include the following:

- “Safety class” on page xvi
- “Before startup” on page xvi
- “During test” on page xvi
- “During maintenance and repair” on page xvi
- “Shutdown when defective” on page xvii

Safety class

The 4400 is designed and tested in line with DIN 57411 part 1 (protective measures for electronic test equipment). The instrument complies with safety class I. It left the works in a perfectly safe condition for operation. To make sure it stays this way and can be operated without any risk, please observe carefully the following notes, which are based on section 17 of DIN 57411 part 1 a.

Before startup



Ensure proper supply voltage

Before powering on, ensure that the operating voltage which is permitted for the 4400 (115 V \pm 20% or 240 V \pm 20%) is identical with your line voltage. You do not need to set the voltage range; the 4400 automatically adjusts to the applied (permissible) line voltage.

The power plug of the 4400 may only be inserted in outlets with a ground contact.

Never use extension cables without grounding conductor.



Apply proper grounding

The grounding conductor must under no circumstances ever be interrupted, neither inside nor outside the test set. If there is no grounding through the grounding conductor, the cabinet of the 4400 could become live as the result of a defect. This can make the test set a potential risk.

During test



Apply proper grounding

Always plug in the power socket of the 4400 before connecting a test circuit in order to use the protective effect of the 4400 grounding.

During maintenance and repair

Live parts can be exposed when you open covers or remove components. Connecting parts can also be live.



Live parts with and without supply voltage connected

Before any adjustment, maintenance repair or replacement of parts, the test set must be separated from all voltage sources if it will be necessary to open it. If any jobs have to be performed on the 4400 while voltage is applied, they should only be undertaken by a specialist who is aware of the dangers that are involved. Capacitors in the power supply can still be charged, even though the instrument has been separated from all voltage sources.



High voltages after power disconnect

High and dangerous voltages can still be present in the instrument even when it is being powered from a battery.



Use proper fuses

Defective fuses must be replaced with fuses of identical specifications. Never patch fuses or short the fuse holder.

Shutdown when defective

In the following cases, safe operation is very likely to be no longer possible:

- if the test set exhibits visible damage,
- if the test set will no longer work,
- after a longish period of storage under the wrong conditions (see data sheet),
- following transport under adverse conditions.

If you suspect that it is unsafe to continue operating the 4400, shut it down immediately and pull out all power connectors.

Secure the test set in such way that nobody else can start it up again, in order to protect any third party.

Get in touch with your nearest Willtek service center.

Overview

A green square containing the white number 1, indicating the chapter number.

This chapter provides a general description of the 4400 Series Mobile Phone Tester. Topics discussed in this chapter include the following:

- "About the 4400" on page 2
- "What's new in version 6.10" on page 3
- "Features and capabilities" on page 3
- "Connecting test leads" on page 4
- "Using the front and rear panels" on page 6
- "Menus and levels" on page 20
- "Menu fields" on page 22
- "Help on Help" on page 26
- "Accessories and options" on page 29

About the 4400

The Willtek 4400 Mobile Phone Tester Series has been designed to meet the needs and requirements of service, manufacturing, quality assurance and engineering facilities. For all these areas, accuracy is the key and the 4400 provides exceptional value (superb VSWR and accuracy). For manufacturing, speed is an additional key requirement.

The 4400 features two different operating modes: The **synchronous mode (call mode)** provides call processing with multiband handovers for fast measurements in different frequency bands. The **asynchronous mode (non-call mode)** is dedicated to the board-level alignment, providing generator and analyzer capabilities.

The 4400 series of products provides solutions for:

- CDMA, AMPS (CDMA2000 1xRTT System Option),
- GSM (GSM System Option),
- HSCSD (with Multislot HSCSD Option),
- GPRS (GSM/GPRS System Option),
- EGPRS (EDGE System Option),
- WCDMA/UMTS (WCDMA Non-Call Mode Option, WCDMA Call Mode Option),
- TD-SCDMA (TD-SCDMA Non-Call Mode Option).

The built-in programming tool **RAPID!** allows for easy test automation. Default standard test routines are already stored on the machine, and more enhanced sequences are available (as options).

RAPID! even allows for data input from bar code readers or the control of other external devices. For example, the ability to send AT commands to a cellular phone enables completely automatic testing. Results can be easily documented through screen dumps to an attached printer or file storage. The numerical data of graphs can be read out via GPIB or written to a floppy disk.

Since requirements for the different application areas are different, the 4400 provides a wide range of various options:

The **Audio Option** allows for acoustical measurements and supports dedicated vocoders (speech codecs).

The **Power Supply Option** works as a battery replacement for terminals.

The **Current Measurement Option** allows for easy calculation of standby or talk times.

What's new in version 6.10

WCDMA System Option:

- Additional frequency bands
- BER/BLER measurement (call mode)
- Inter-frequency handover (call mode)
- Speed enhancements in Modulation Quality and Constellation Display

TD-SCDMA System Option (non-call mode):

- Constellation Display
- Multislot measurements
- Improved EVM measurement accuracy

GSM System Option:

Several improvements for the audio bidirectional mode

GSM/GPRS System Option:

On/off switch for ID requests during GPRS attach procedure

EDGE System Option:

Enhanced coupling loss handling

Features and capabilities

- Based on the multiformat, future-proof platform 4400
- Adaptable to individual needs with options including CDMA2000, GSM/GPRS, EDGE, WCDMA, TD-SCDMA, Audio, Codec, MS Power Supply, MS Current and TCP/IP
- Easy-to-use graphical user interface with external mouse, keyboard and monitor
- Standard SCPI commands for remote control
- Built-in RAPID! for stand-alone automated procedures, allowing the user to easily develop and run programs on the 4400
- DSP platform supports parallel testing of TX, RX and Audio
- Driver software available for Test Stand and Lab Windows CVI

Connecting test leads

[Crash course](#) – Starting a test is easy!

[Cabling](#) – Connecting a mobile to the 4400

Crash course To perform a test on the 4400, keep to the following procedure:

- 1 Switch on the 4400 using the standby button on the front panel.
- 2 Select the required communication system, e.g. GSM or CDMA.
- 3 Connect the mobile to the 4400.
- 4 Switch on the mobile and set up a call.
- 5 Navigate to the required test menu.



Don't exceed the maximum RF power level

Before connecting any RF equipment to the 4400, please check that the maximum input power level is not exceeded. The maximum input power level is indicated on the front panel of the 4400, just besides the RF in/out socket.

Cabling

For precise measurements, we strongly recommend to use an RF adapter cable between the mobile under test and the Willtek 4400. Mobile-specific RF adapter cables are available from the mobile's manufacturer or from Willtek. Please contact your local Willtek representative.

However, should the adapter cable for your mobile not be at hand, the air connection offers a simple-to-use alternative for quick tests. With the optional RF Shield, precise measurements can be achieved without interference.



Don't exceed the maximum RF power level

Before connecting any RF equipment to the 4400, please check that the maximum input power level of the 4400 is not exceeded in any case. Otherwise, the highly sensitive input stage may be destroyed immediately. The maximum input power level is indicated on the front panel of the 4400, just besides the RF in/out socket.

Cable connection



The cable connection between a mobile under test and the 4400 requires a mobile-specific RF adapter cable. This RF adapter provides the mobile-specific connector on one end and the Willtek RF-click connector on the other one. Please note that you will need the optional adapter cable 'RF click to N-type' in order to use the RF adapters on the 4400's N-type RF in/out socket.

Air connection

Using the air connection is a simple method to connect a mobile under test to the 4400 when there is no special RF adapter available. Basically, there are three possibilities to use the air connection.



The simplest possibility is to put an optional antenna on the 4400's RF in/out socket. However, the drawbacks of this solution are the unknown coupling loss and the potential interference.

The coupling loss depends on factors such as the orientation of the mobile towards the 4400, the distance and the frequency.

The possible interference depends on the type of RF radiating equipment being used in the same room or area and on the distance to e.g. a 'real' base station. Consequently, this solution only works for basic functional tests.

Please also note that you may need different antennas for different frequency bands (GSM 900/1800/1900 or cellular/PCS bands).



Using the optional Antenna Coupler, you already have two antennas built into a special casing. Because of the fixed position of the mobile, the coupling loss is much better reproducible and can be fairly compensated using a coupling loss

table. However, there is still no protection against potential interference. The operating range of the Antenna Coupler covers the complete GSM 900, 1800 and 1900 frequency ranges.

As a standard, the Antenna Coupler comes with an RF-click connector plus an adapter cable 'RF click to TNC'.

Please note that you will need the optional adapter 'TNC to N-type' in order to use the Antenna Coupler on the 4400's N-type RF in/out socket.



If you need to perform reproducible measurements without having the possibility of using a cable, we recommend to work with the optional RF Shield. This is a metal box that covers both the mobile under test and the Antenna Coupler. The RF Shield keeps radio interference away from the air connection. The achievable measurement quality can be as good as when using a cable if the attenuation is compensated with the help of the coupling loss table.

The RF Shield is available with either a TNC-type or N-type connector.

NOTE

The RF Shield Package contains both the RF Shield and the Antenna Coupler.

Using the front and rear panels

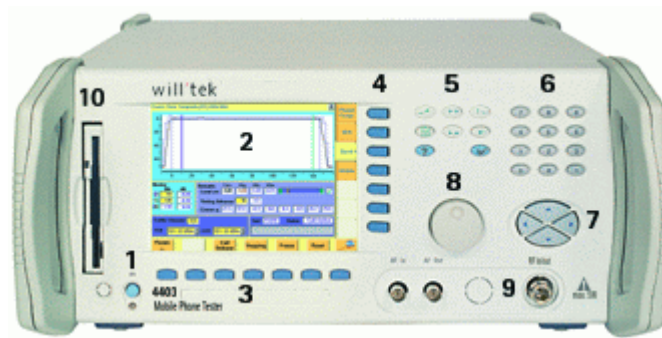
[Front panel](#) – Get to know the ten sections of the front panel.

[Rear panel](#) – Here you will find all about the rear panel of your 4400.

[Connectors](#) – The three connectors on the front panel provide the interfaces to the mobile under test. The back panel features all the interfaces you need to integrate the 4400 into your test environment.

[The keys on the front panel](#) – Don't be confused about softkeys, marker tabs, function keys, numeric keys, cursor keys, the selection key or mapping the 4400's keys to an external keyboard. Here you will find all you need to know about any key.

Front panel The front panel of the 4400 is divided into ten sections.



Section 1 – Standby button and MS Power Supply connector

When you turn on the power switch on the back of the 4400, the unit goes to standby mode and the indicator just underneath the standby button is backlit in red.

Push the standby button to switch the 4400 into operating mode. The standby indicator will then turn green.

When you push the standby button again, the unit will go back into standby mode.

NOTE

The power switch is located on the rear panel of the 4400. A dark standby button indicates that the power switch is off.

The MS Power Supply connector is used in conjunction with the MS Power Supply and Current Measurement Options. It provides supply voltage and current for the mobile under test.

The cable for this connector is delivered with the MS Power Supply Option.

Section 2 – Display

The display is the graphical user interface of the 4400. It will show you menus with entry fields and display fields. The display also labels the software-driven softkeys (see section 3) and marker tabs (see section 4).

Section 3 – Softkeys

Softkeys call specific functions that vary with the menu displayed. For instance, when you switch on the 4400, the softkeys will allow you to select the kind of system you want to work with (e.g. GSM or CDMA).

Section 4 – Marker tabs

Like the softkeys, the marker tabs are software-labelled keys used for selecting a test (e.g. Phase/Frequency test), for navigating through this online help, or for loading different tools (e.g. Analyzer).

Section 5 – Function keys

Function keys are used for general operation. They have a fixed function. The **ENTER** key, for instance, confirms changes on entry fields.

Section 6 – Numeric keys

The numeric keys are mainly used to enter values on entry fields.

Section 7 – Cursor keys

The **LEFT** and **RIGHT** cursor keys are used to select the required entry field of a menu. A selected entry field is inverted and therefore easily visible on-screen. With the help of the **UP** and **DOWN** cursor keys, values on entry fields can be changed.

Section 8 – Selection key

The selection key is actually more a wheel. It is used to quickly change values on entry fields.

Section 9 – Connectors

The connectors on the front panel provide the interfaces for the mobile under test.

The connection of a mobile is outlined in section [Cabling](#).

The detailed description of all connectors on both the front and back panel of the 4400 can be found in section [Connectors](#).



HAZARD

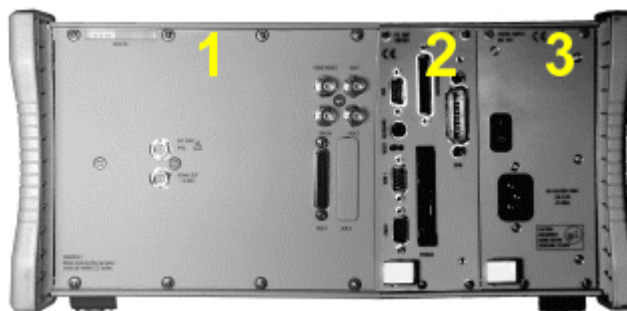
Before connecting any equipment, please make sure that the maximum input levels of the connectors are not exceeded. Otherwise, the highly sensitive stages of the 4400 might be destroyed immediately.

Section 10 – Floppy drive

As with all floppy drives, the floppy disk will only fit in one direction. Please insert the floppy disk carefully. Do not use any force as this could damage both, the floppy disk and the floppy drive.

Rear panel

The rear panel of the 4400 is subdivided into three main sections.



Section 1 – The RF/IF/AF section

The RF/IF/AF section provides several interfaces that may support your tests. Most of these interfaces are related to synchronization and triggering.

Section [Connectors](#) gives a detailed overview of all the connectors, their functionality and technical specifications.

Section 2 – The PC unit

The 4400 is powered by an industrial PC. Therefore, you will find all the standard PC interfaces on the back of the 4400.

If you need to know anything regarding the PC interfaces, please refer to [Connectors](#).

Section 3 – The power supply

In this section, the power switch and the power connector are located.

NOTE

The power switch has to be **on** before you will be able to start the 4400 using the **STANDBY** button.



HAZARD

Before connecting the 4400 to the power line, please check that the power available on your supply line is within the operating range of the 4400's power supply as indicated on the back of the unit and in the getting started manual.

Connectors

Connectors on the front panel

There are three connectors on the front panel:



AF in — BNC socket. Input for AF (audio) signal. This input is only used when the Audio option is installed.

This input can be used either as a balanced or as an unbalanced input (software switch).

The input impedance is approx. 250 k Ω /20 pF.

The usable AC frequency range stretches from 30 Hz to 20 kHz.



HAZARD

The absolute maximum input voltage is ± 40 V DC or 30 V (rms) AC. An AF input voltage of more than ± 40 V DC or 30 V (rms) AC may result in an immediate destruction of the highly sensitive AF input stage of the 4400. Willtek will not accept liability for any damage of the input stage due to overload.

AF out — BNC socket. Unbalanced AF (audio) signal output of the 4400. Used only when the Audio option is installed.

Best performance is achieved on a 600 Ω load.

The maximum output voltages are 4 V (rms) for sinusoidal signals or 11 V peak-to-peak for all other signals.

RF in/out — RF signal input for transmitter measurements and output for receiver measurements. It depends on the current system test whether the 4400 measures an input signal, generates an output signal or both.



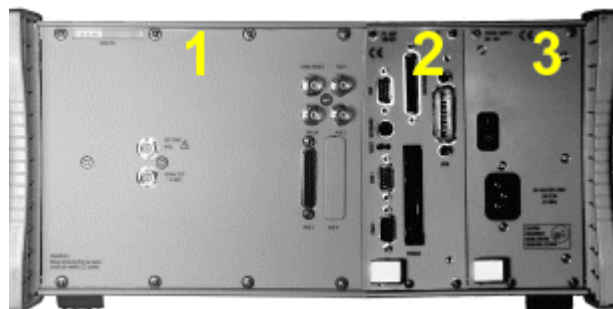
HAZARD

The absolute maximum input power is 5 W (37 dBm). An RF input power of more than 5 W may result in an immediate destruction of the highly sensitive RF input stage of the 4400. Willtek will not accept liability for any damage of the input stage due to overload.

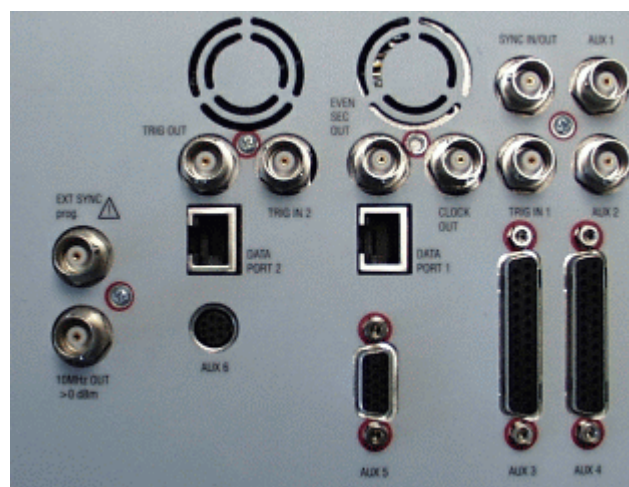
Note: There is no time limitation for applying the maximum input power.

Connectors on the rear panel

The rear panel is divided into three main sections. These are from left to right: the RF section (1), the PC unit (2) and the Power supply (3). Each section provides one or more connectors for specific functions. The connectors are described from left to right and from top to bottom.



Connectors on the RF section



Not all of the connectors may be fitted in your 4400. The connectors actually available depend on your system configuration, in particular the installed system options.

EXT SYNC — BNC socket. Input for external reference signal. If a high-precision reference signal of 5 MHz, 10 MHz or 13 MHz is connected here, the internal 10 MHz reference signal of the 4400 will not be used.

Input level: approx. 0 dBm

Frequency tuning range: approx. 1 ppm (10E-6) (equals 10 Hz at 10 MHz)

Input resistance: approx. 50 Ω

10 MHz OUT — BNC socket. Output of the 4400's internal 10 MHz reference signal. With the help of this output, the internal time base of the 4400 can be used to synchronize external equipment.

This output is very useful if the optional oven-controlled oscillator (OCXO) is installed. More information on the OCXO can be found in section [Accessories and options](#).

Output power: approx. +5 dBm

Output resistance: approx. 50 Ω

SYNC IN/OUT — BNC socket, providing a CMOS-TTL signal.

The signal on this connector allows the GSM-specific synchronization with external units. The connector is currently configured as **Sync out**. The 4400 outputs its internally generated frame synchronization signal. This signal is TTL-low for (downlink) time slots 0 to 2 and TTL-high for time slots 3 to 7.

NOTE

In GSM, the timing of up- and downlink is shifted by three time slots to avoid that a mobile has to receive and transmit during the same time slot. The time difference between the falling and the rising edge of the Sync out signal reflects exactly that time shift.

TRIG IN — BNC socket, expecting a CMOS-TTL signal.

A trigger signal applied on this connector triggers all GSM-related RF measurements. The 4400 allows you to select between triggering on the rising or falling edge of the signal connected here.

The main application for this connector will be measurements in non-call mode at very low power levels, e.g. asynchronous measurements.

TRIG IN 2 — BNC connector for future use.

TRIG OUT — BNC connector for future use.

NOTE

For GSM the 4400 can internally trigger its asynchronous measurements as long as the RF power level transmitted by the mobile is above -30 dBm. Only in case you are working with lower RF power levels, it is necessary to supply an external trigger signal on this input.

AUX 1 — Reserved for later use. Please do not connect anything here.

AUX 2 — Reserved for later use. Please do not connect anything here.

AUX 3 — Reserved for later use. Please do not connect anything here.

AUX 4 (AUX out) — Using the Audio Option of the 4400, audio signals can be made available on this output, for example to connect the 4400 to an external loudspeaker. The output is available on **pins 8 and 9** of this connector. The output impedance is $8\ \Omega$; the maximum output power level is 1 W.



HAZARD

AUX out is a balanced output. This means that the voltages provided are **floating**.

Never short-circuit these two pins and never connect them to ground as this could result in an **immediate destruction** of the AUX out output stage of the 4400.

AUX 4 (AUX in) — This input can be used to pick up audio signals with a microphone and to use those signals for audio measurements.

The auxiliary input is available on **pins 12 and 13** of this connector.

This unbalanced input is only used when the Audio option is installed.

The input impedance is $< 1\ \text{k}\Omega$.

The usable AC frequency range stretches from 30 Hz to 20 kHz.



HAZARD

The absolute maximum input voltage is **1 V (rms) AC**. An AF input voltage of more than 1 V (rms) AC may result in an **immediate destruction** of the highly sensitive AUX in input stage of the 4400. Willtek will not accept liability for any damage of the input stage due to overload.

AUX 5 — This is a diagnostic port without any user application.

AUX 6 — For future use.

EVEN SEC OUT — This is a CDMA-related periodic signal. The signal on this BNC connector is TTL, with a period of 2 seconds. A negative pulse occurs every two seconds, with a pulse width of approximately 51 nanoseconds.

CLOCK OUT — This BNC connector transmits a periodic TTL CDMA clock signal. Please refer to sections CDMA Call Mode and CDMA Non-call Mode in the CDMA user's guide for more information.

DATA PORT 1 — This RJ45 connector is a test port for CDMA without any user application.

DATA PORT 2 — RJ45 connector for future use.

Connectors on the PC unit



VGA — Standard connector for a VGA monitor.

NOTE

The resolution of the 4400's video output is set to 640 x 480 pixels and cannot be altered.

Keyboard — Standard PS/2 connector for a PC keyboard.

NOTE

An external keyboard is highly recommended for **RAPID!** programming.

COM 1 — Standard 9-pin D-sub connector for serial interface 1. Usually used to connect a standard PC mouse.

NOTE

A mouse is highly recommended for **RAPID!** programming.

COM 2 — Standard 9-pin D-sub connector for serial interface 2.

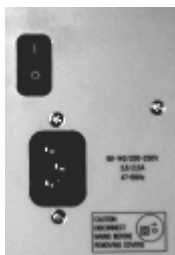
CENTRONICS — Standard parallel port to connect a PC printer (LPT 1).

PCMCIA — These two PCMCIA slots are reserved for LAN interfacing and for future applications.

The slot takes either two Type 1 cards or one Type 2 card.

GPIB — General Purpose Interface Bus according to specification IEEE-488.2.

Connector on the power supply



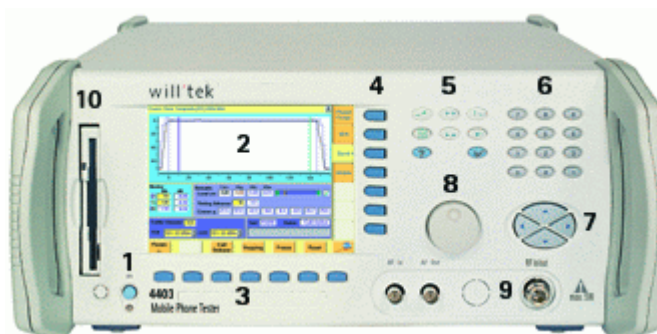
Power in — Standard IEC 320 power connector (3 pins). Please make sure that the power cable is properly grounded.



HAZARD

Before connecting the 4400 to the power line, please check that the power available on your supply line is within the operating range of the 4400's power supply as indicated on the back of the unit and in the getting started manual.

The keys on the front panel



There are six different groups of keys. In the photo of the front panel above, they are marked with numbers 3 through 8.

- The softkeys (3)
- The marker tabs (4)
- The function keys (5)
- The numeric keys (6)
- The cursor keys (7)
- The selection key (8)

Notation of keys in this manual and online help



Hard keys, such as the **ENTER** key, appear in capital letters.

Param

...

Either a graphical image of the softkey is used, or its label is in bold print, e.g. **Param....**

External keyboard and mouse

There is the possibility to connect a standard PC keyboard and/or a standard serial PC mouse to the 4400. When an external keyboard is connected, the keys on the front panel of the 4400 and the keys on the external keyboard may be used in parallel.

In order to give you access to the 4400-specific keys like the **TOOLS** function key, those keys are mapped to the external keyboard. The mapping table can be found in section [Keyboard mapping table](#).

The connectors for both the mouse (COM 1, Sub-D 9) and the external keyboard (Keyboard, PS/2) are located on the back panel of the 4400 in the PC section. Section [Connectors](#) provides you with all the information about those connectors and their locations.

The softkeys

Softkeys call specific functions that vary with the menu displayed. The specific function is written in the last two lines of the display, just above the physical key. This is what we call the label of the softkey.

Example: In the Welcome menu of the 4400, the softkeys will allow you to select between measurements in call mode and those in non-call mode.



Notes on softkeys

- When the label of the softkey ends with an ellipsis (...), this softkey will take you to a new menu (of a lower level), where you can enter e.g. additional parameters.
To return from a lower menu level, push the **ESCAPE** function key.
Example: Softkey **Param...** will take you to a new menu, where system parameters can be set.
- When there is a test in progress, some softkeys will be **blocked**. They turn to gray to indicate that the previously started test has to be either completed or aborted, before a new selection with the softkeys is possible.

Example:



While softkey **STOP** could be pushed any time, softkey **Paging** is not accessible as long as the current test or procedure is in progress.

- While the Online help is on display, the softkeys change to navigating aids and help you to find your way through the Online manual.

The marker tabs



When there are several menus of the same level, e.g. different system tests, the seven marker tabs make it easy to switch between them.

When the Online help is on display, the marker tabs provide the links to related pages.

Selected, deselected and blocked marker tabs

Marker tabs can be selected, deselected or blocked.

A **selected marker tab** is highlighted in yellow and indicates that the current menu has been selected.

Deselected marker tabs are orange to indicate that they could be selected any time.

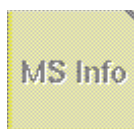
When there is a test in progress, some or all marker tabs will be **blocked**. They turn to gray to indicate that the previously started test has to be either completed or aborted, before a new selection with the marker tabs is possible.



Marker tab [MS Info], not selected



Marker tab [MS Info], selected



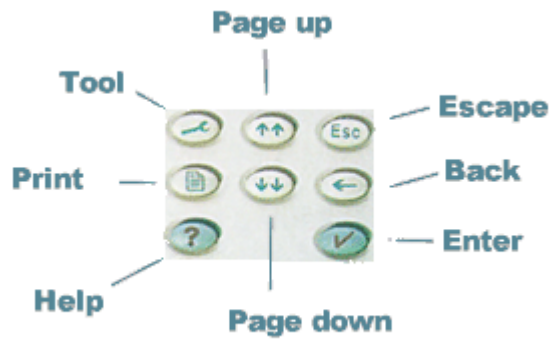
Marker tab [MS Info], blocked

Notation of marker tabs in this documentation









Wherever this Online help manual refers to a marker tab, either the marker tab is shown graphically, or it is indicated by setting its name between square brackets, e.g. [MS Info].

The function keys

With the help of the function keys, you have instant access to important or useful functions of the 4400. Some of the function keys help you to navigate through the current menu.



Meaning of the function keys

| Symbol | Name | Description |
|---|-------------------|---|
|  | TOOL | Calls the tools menu. From there, you start the special tools like the Audio measurements or the MS Power Supply. |
|   | PGUP, PGDN | A push selects the first entry field of the previous or next area. When the Online help is on display, these keys call the previous or next page of the Online help. |
|  | ESCAPE | A push on this key takes you one menu level up. Should this cause a change of a parameter, you will be prompted to either confirm or discard your change in menu levels. |
|  | BACK | Deletes the character left to the cursor on an entry field. |
|  | ENTER | Confirms a value entered on an entry field. |
|  | HELP | Calls the Online help on-screen. Once that you are done with the help screens, press [Escape] to return to the previous screen. For more information about the online help system, see section Help on Help . |
|  | PRINT | Prints the current screen to the attached printer or generates a screenshot file on the floppy disk. For more detailed information, see section Parallel Port Settings. |

The numeric keys



With the help of the Numeric keys, you enter parameter values on entry fields.

Important function keys mainly used in combination with the numeric keys



deletes the character left to the cursor on an entry field.





confirms all the changes or entries.

The cursor keys



With the help of the  (**LEFT**) or  (**RIGHT**) cursor keys, you navigate the cursor from one entry field to the previous or next one.

The  (**UP**) or  (**DOWN**) cursor keys allow you to increase or decrease the parameter value on a numeric entry field or to step through the predefined settings of a selection field.

The selection key



This wheel is used to quickly change values on entry fields.

When the cursor is positioned on an Entry field (the Entry field is then inverted), a turn on the selection key directly changes the value. Any change made with the selection key is instantly confirmed. It is not necessary to push the **ENTER** key

Altering numeric values: navigate to the required entry field.

Clockwise rotation (i.e. to the right) of the selection key increases the value, counterclockwise rotation (i.e. to the left) decreases it.

Changing the setting of selection fields: navigate to the selection field. The available predefined settings are highlighted one by one when you slowly turn the selection key clockwise or counterclockwise.

Keyboard mapping table

Every key of the 4400 is mapped to a specific key or key combination of an external PC keyboard.

Softkeys

From left to right, the seven softkeys are mapped to the key combinations **SHIFT+F1** to **SHIFT+F7**.

Marker tabs

From up to down, the seven marker tabs are mapped to the key combinations **CTRL+F1** to **CTRL+F7**.

Function keys

| 4400 front panel | Keyboard |
|------------------|------------------|
| HELP | F1 |
| TOOL | F2 |
| PRINT | F3 |
| PGUP | PAGE UP |
| PGDN | PAGE DOWN |
| ESCAPE | ESC |
| BACK | BACKSPACE |
| ENTER | ENTER |

Numeric keys

Mapped to the numeric keys on the standard PC keyboard.

NOTE

Before using the numeric keypad of the external keyboard, please check that the **NUM LOCK** is switched on.

The following specific keys are mapped to:

±: - i.e. the minus key

.: i.e. the dot key

Cursor keys

| 4400 front panel | Keyboard |
|------------------|-------------------|
| LEFT | SHIFT+TAB |
| RIGHT | TAB |
| UP | ARROW UP |
| DOWN | ARROW DOWN |

Selection key

F11 for counterclockwise turn, i.e. to the left,
F12 for clockwise turn, i.e. to the right

Menus and levels

Menus

The entire area of the display is what we call a **menu**. Any time you switch on your 4400, it first displays the start or Welcome menu. The Welcome menu is the highest menu level of the hierarchically structured user interface.



With the help of the marker tabs (on the right-hand side of the display), you select between the different system menus (e.g. [GSM] or [CDMA]).

The softkeys (just below the display) give you instant access to the various test menus (e.g. **Calls & Meas.** or **Gen/Ana**).

Levels

The user interface of the 4400 is organized in a hierarchical order. This means that there are several levels.

The Welcome menu is at the top level. As you go lower in menu levels, you will reach more specialized or detailed functions.

- When there are several menus of the same level, you can select between them using the marker tabs on the right-hand side of the display.
- To navigate to a menu of the next lower level, push any one of the softkeys.
- When you would like to return to the next level above, push the **ESCAPE** function key.

Example: You would like to go to the GSM Burst Zoom menu.

You can use either the Calls & Meas. or the Gen/Ana menus. For tests in call mode, i.e. involving a call setup, simply push the **Calls & Meas.** softkey from the Welcome menu. This will take you to the test menu where you can select between the various tests (this is the so-called test level).

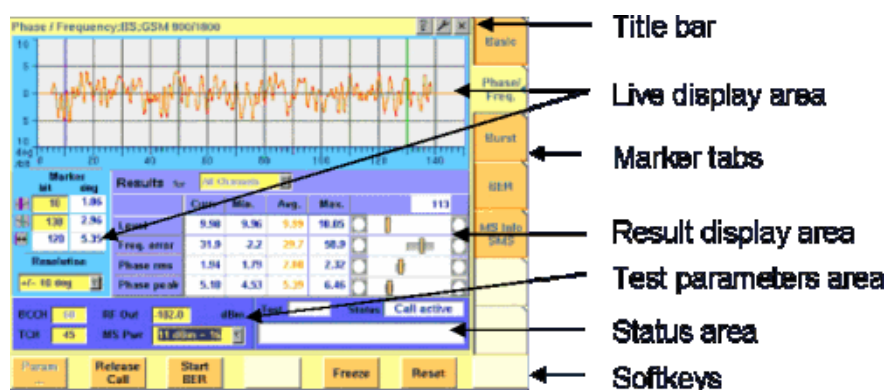
Now you push the third marker tab (from the top) and the 4400 will display the Burst test menu.

The **Zoom** softkey of the Burst test menu leads you to the Zoom menu.

The areas of a test menu

Each test menu is divided into different color-coded areas, each providing different functions. These functions are described in detail in the test sections of the related system.

The figure below gives you a first overview. The example we have chosen is the Phase/Frequency test menu of the GSM system, but the principle is the same for all other test menus.



- The **title bar** shows the name of the menu you are currently working in. In the example above, this is the menu testing the phase/frequency characteristics of a mobile in a GSM 900/1800 system.
- The **live display area** shows the measurement results in real time, whether numerical or graphical. To make the live display area easy to identify on the display, it is placed on a light blue background.
- The area with the purple background is the **result display area**. It shows the statistical analysis of the measurement results and the bar display.

- The two lower areas of the menu are both on dark blue background. The left one is the **test parameters area**. On this area, you will find entry fields that allow you to instantly alter relevant test parameters without being forced to navigate to some parameters menu first. The area on the right-hand side is called the **status area**. Here, the 4400 tells you if the current test has passed or failed. Furthermore, the signaling status will be shown in this area.
- The **softkey area** forms the bottom of a menu. With the help of the seven softkeys you can directly call a system parameters menu, start special test functions or zoom the test display.
- The **marker tabs area** on the right-hand side of the menu carries the labels of the seven marker tabs. They provide access e.g. to different tests.


Menu fields

In the menus of the 4400, we use two different types of fields:

- An **Entry field** is what we call a field that is accessible by the cursor and accepts an entry. Entry fields are mainly used to enter test parameters. There are entry fields that only accept numeric inputs (like e.g. a channel number) and there are entry fields that accept alphanumeric data (like e.g. an SMS text).
There are also two special forms of entry fields: the so-called **selection fields** where you may select one of several predefined possibilities (e.g. what bit class(es) you want to use for a bit error rate measurement) and the **check boxes**.
Check boxes are like an on/off switch. You can't enter values, you can just select or deselect them.
All entry fields have a yellow background to make them easy to allocate on-screen. The only exception to this rule are the check boxes.
- **Display fields** show the results of the performed measurements. Again, there are different types of display fields as measurement results can be purely numerical (e.g. RF power) or they can be alphanumeric (e.g. the serial number of a mobile). The **bar display** also is a display field.
To make display fields easy to distinguish from entry fields, they always have a white background.

Entry fields

The Willtek 4400 offers you different types of entry fields:

- **Alphanumeric, hexadecimal or numeric** entry fields accept text or numbers respectively.
 Example: TCH – Here you enter the number of the traffic channel the measurement shall be performed on.
- **Selection fields** are entry fields that offer you a choice from a list of predefined values. Click with the mouse on the triangle to see the available options, or press the **UP** or **DOWN** cursor key.



Example: **TCH Type** – This selection field enables you to choose a specific TCH type from a list.

- Check boxes only need a simple yes or no. The 'yes' is indicated by a green check box. To activate or deactivate the function, click on it with the mouse or go the field and press **ENTER**.



Example: The **Call on Loc. update** check box is unchecked. This means that the function will not be performed.

- Radio buttons allow you to select between different options, similar to selection fields.



Example: Of the audio generator radio buttons in the picture, the sine function has been selected and the other functions are inactive. By selecting the rectangle function, sine would automatically be deselected. Select a function by clicking on the appropriate radio button, or move to the radio button with the cursor keys and press **ENTER**.

Navigating to an entry field

Jump to the required area using the function keys **PgUP** or **PgDn**.

Select the entry field you want to make changes on with the help of the **LEFT** or **RIGHT** cursor keys.

The selected entry field is inverted to make it easily visible on-screen.

Entering values on numeric entry fields

There are two ways how to work with numeric entry fields: either you enter the value using the 4400's numeric keys or you use the selection key. The achievable results are basically the same. However, there is no entry discard function when you use the selection key.

- Enter the required value using the numeric keys of the 4400. While the entry field is open, entered values can also be edited using the **BACK** function key. To **confirm** the changes, push the **ENTER** function key. If your entry should be out of the possible data range, the 4400 will prompt you with *Data out of range* in the status area of the current menu. To **discard** the changes, just leave the entry field using the **LEFT** or **RIGHT** cursor keys.
- Instead of typing in a new value, you may also turn the selection key to increase or decrease the current value of the entry field. The **UP** or **DOWN** cursor keys also increase or decrease the value. **Note:** The cursor keys have a built-in 'autorepeat function'. This means that the value on the entry field will be incremented respectively decremented by one step on a short push or by a couple of steps if you push the key for a longer period of time.

NOTE

When you use the selection key or the cursor keys to change a parameter value on an entry field, the changes are confirmed automatically. It is not required to push the **ENTER** key.

Entering values on hexadecimal entry fields

Where an entry in hexadecimal notation (0 to 9, A to F) is expected, the marker tabs change to the letters A to F. You can enter data with the numeric keys (0 to 9) and with the softkeys (A to F).



Entering values on alphanumeric entry fields

On alphanumeric entry fields as e.g. **SMS to mobile phone**, texts are best entered using an external keyboard. Of course, it is also possible to enter alphanumeric values directly on the 4400:

As soon as the entry field is opened, the marker tabs change to groups of letters and symbols. The lowest marker tab incorporates the "shift" function and allows you to toggle between lowercase and uppercase letters.



As soon as you select a group of letters or symbols by pushing the corresponding Marker tab, the Marker tabs will change again to single letters or symbols. Push the Marker tab of the desired letter or symbol. That one will appear on the entry field and the Marker tabs will change back to groups of letters and symbols. While the entry field is open, entered values can also be edited using the **RIGHT** function key.

Entering values on selection fields

Again, there are two ways how to work with selection fields: either you change the selection using the 4400's cursor keys or you simply turn the selection key. The achievable results are basically the same. However, there is no entry discard function when you use the selection key.

- Select the required value using the **UP** or **DOWN** cursor keys. To confirm the changes, push the **ENTER** function key. To discard the changes, just leave the entry field using the **LEFT** or **RIGHT** cursor keys.
- Instead of selecting a new value with the cursor keys, you may also turn the selection key.

NOTE

When you use the selection key to change a parameter value on a selection field, the changes are confirmed automatically. It is not required to push the **ENTER** key.

Switching check boxes

Any push on the **ENTER** function key toggles the currently selected check box. This means that an unchecked check box will be switched to 'on' or 'yes' with the first push and back to unchecked with the second push. Unchecked boxes are indicated in gray; checked ones are highlighted in green.

Display fields

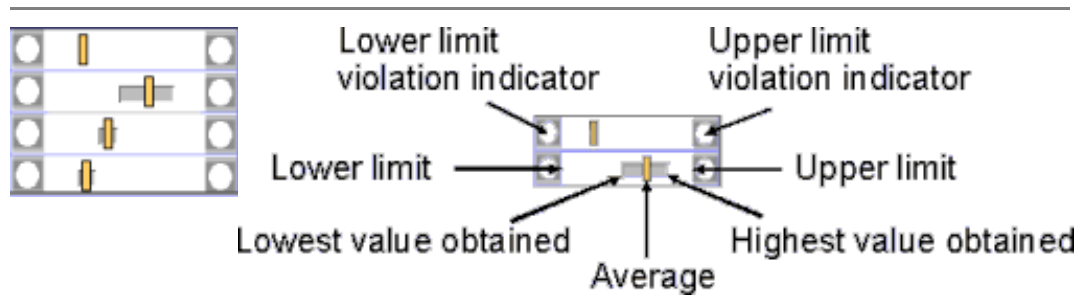
The 4400 offers you two different types of display fields:

(Alpha)numeric display fields

| MS Report | | |
|-----------|----------------|---------|
| | RX Lev | RX Qual |
| Full | 63 = -48.0 dBm | 0 |
| Sub | 63 = -48.0 dBm | 0 |

These kind of display fields are used to show test results as numeric values, as strings (text) or as graphical display of numeric values (e.g. the shape of a received burst within a template).

The bar display



The bar display is used to display measurement results that are obtained by running a specific test a certain number of times. It shows a couple of important values on a relatively small area:

- The large **white area** indicates the range between the lower and the upper limit of the specific test.
- The **dark gray** section within the white area shows the range of all measurement values obtained from the minimum to the maximum.
- The small **orange rectangle** marks the average value.
- The two **limit violation indicators** are white as long as no violations of the test limits occur.
- As soon as a measurement value obtained is smaller than the lower limit, the lower limit violation indicator will turn to red.
- The same will happen with the upper limit violation indicator when a measurement value exceeds the acceptable range.

NOTE

To reinitialize the statistics, push the **Reset** softkey.

Help on Help

This section describes how to work with the Online help. Here you will find all about

- [Navigating help](#),
- [Using the online help on an external PC](#).

Notation rules

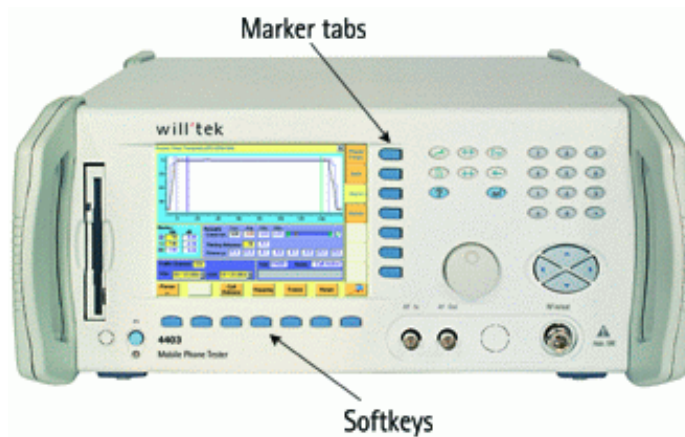
While working with the Online help, you will have to push softkeys, marker tabs, or other keys. To keep things as clear as possible, the online manual either uses the graphical image of the item as you see it on-screen or it uses the following notation rules:

Calls & Meas. The label of a softkey is indicated like this.

[RF Gen] The label of a marker tab is indicated in square brackets.

| | |
|---------------|--|
| ESCAPE | The name of function keys, numeric keys or cursor keys is indicated in this type of letters. |
| Channel | Entry fields are indicated in normal letters. |

Navigating help



With the help of both the marker tabs and the softkeys, it is easy to navigate through this Online help.




Marker tabs – the links

This Online manual is a so-called hypertext document. It contains **links**. Those links allow you to jump to a different page where more detailed information about a specific topic is provided. On a PC, you would simply click the link. On the 4400, you can use the mouse or the marker tabs for 'clicking' the links.

Example: This page provides links to the documents explaining the softkeys and marker tabs in detail. Use the marker tabs to directly jump to the related pages.

Softkeys – the quick navigators

Using the softkeys you quickly navigate through a chapter or call its index. There are five softkeys available:

| | |
|---|---|
|  | This softkey calls the first page (i.e. the page directly following the index page) of the chapter. If that first page is already on display, nothing will happen. |
|  | A push on this softkey takes you to the previous page of a chapter. When you push this softkey on any index page, you will get the last page of the previous chapter on-screen. |
|  | Calls the subsequent page of a chapter. From the last page of a chapter, this softkey will take you to the index page of the subsequent chapter. |

Index

A push on this softkey takes you to the index page of the current chapter. This index page shows the contents of the chapter along with links to the different topics. When you push **Index** on an index page, then the index page of the next level up will be shown.

Back

Takes you back to the help page shown last before the current page.

Using the online help on an external PC

Installing the online help on a PC

Standard PCs offer a much larger screen diameter compared to the 4400. Therefore, it might be more convenient for you to have the Online help running on a PC. Furthermore, your PC would usually be connected to a printer, so printing pages or chapters of the Online manual is easier when working on a PC. In order to install the Online help on a PC, you need to have the Online manual on disk. This can be obtained directly from Willtek.

NOTE

The Online manual contains a number of subdirectories. Please make sure that those subdirectories are copied to your PC as well. Otherwise, you may experience difficulties getting access to specific images or descriptions.

To start the Online help on your PC, load the file called `frames.htm`. You will find this file in the top level directory of the Online help.

You will be provided with the same navigating aids as on the 4400: the lower frame will show five buttons, having the same functionality as the navigating softkeys on the 4400.

NOTE

If you should explore any problems to navigate with the help of the buttons in the lower frame, load the file `index_e.htm` instead. It is located in the same directory. Loading this file, there will be no specific navigating aids – but you still can use the previous/back functionality of your browser.

Navigating through the help pages

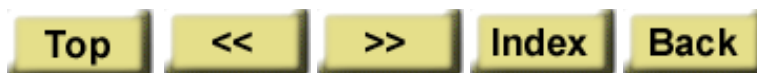
Links instead of marker tabs

On the PC, you may simply click the links on a page like on any Internet web page. For easy identification, these links appear in blue and are underlined. The mouse pointer will change as soon as you position it on any link.

Buttons instead of softkeys

Working with Internet Explorer or Netscape Navigator, you have the same functionality at your disposal as on your 4400.

On the PC screen, you will see two frames. The upper and larger one shows the information, the lower one provides the buttons like on the 4400.



Each section of the manual can be printed with the printing function of the web browser.

Accessories and options

Your 4400 is a highly integrated test set that allows you to perform most of the tests and measurements required in a standard production environment already with its basic configuration. However, there are much more areas of application for your 4400. It can even be used in R&D, for high-level service or for training your staff on digital radio systems.

Willtek has developed a broad range of accessories that help you perform specific tests or measurements, to easily connect mobiles to the 4400 and much more. Please find below a short overview of the available accessories and options, and the applications they have been designed for. Should you miss something in the list below, please check with your local Willtek representative. Due to ongoing product development and improvement, the turnkey solution to your requirement may already be available.

All the latest developments for your 4400 can also be found on the web at www.willtek.com.

NOTE

Specifications are used in this manual for illustration purposes only. Therefore, the specifications of the product delivered may be different from the specifications shown here. However, all relevant specifications for the 4400 can be found in the test set's data sheet. Should that data sheet not be at your disposal, please ask your local Willtek contact for a copy or feel free to download it from Willtek's web site.

Expanding the 4400's measurement capabilities

Audio measurements

In its basic configuration, the 4400 does not support audio measurements. In order to provide audio functions, you need to install the 4470 Audio Option. It provides your 4400 with both an audio generator and audio analyzer.

Additional optional features for audio testing are available for the GSM and GSM/GPRS System Options: The 4471 Basic Codec Option and the 4472 Codec Extension Option allow you to convert the digital data exchanged between the mobile and the 4400 into audible sound.

As you know, GSM currently uses three different codecs.

Full Rate is what GSM started with and what is still the most common codec type. To convert Full Rate into audible sound, the Codec Basic Option is all you need.

Half Rate is supported by many mobiles, although the network operators are hesitant to use this feature because they are afraid of losing subscribers due to poor speech quality.

Enhanced Full Rate is a newer codec algorithm that provides better speech quality than Full Rate. To convert Half Rate or Enhanced Full Rate into audible sound, you will need the Codec Extension Option in addition to the Basic Codec Option.

| Option | Willtek order number |
|---|-----------------------------|
| 4470 Audio Option | M 248 360 |
| 4470 Audio Option for 4400 with CDMA2000 only | M 248 653 |
| 4471 Basic Codec Option | M 248 364 |
| 4472 Codec Extension Option | M 897 156 |



Audio measurements on the 4400

High-precision time base

In its basic configuration, the 4400 is equipped with a TCXO 10 MHz time base. The specified accuracy is approx. 1E-6 or 1 ppm (10 Hz). The OCXO option provides an increased accuracy of approx. 5E-8 or 0.05 ppm (0.5 Hz).

| Option | Willtek order number |
|------------------|-----------------------------|
| 4471 OCXO option | M 214 028 |

Multislot measurement capabilities (HSCSD) (for GSM, GSM/GPRS and EDGE only)

HSCSD (= High Speed Circuit Switched Data) enhances standard GSM with the bundled use of several time slots within a GSM frame for data transmission. Theoretically, the maximum data transmission rate in HSCSD is $n * 14.400$ bps, where n is the number of the bundled time slots.

In case of the 4400, the maximum for n equals 4; this means that the 4400 can transmit and/or receive on four time slots per frame.

The tool to perform tests on HSCSD mobiles is called Multislot/HSCSD Option.

| Option | Willtek order number |
|-----------------------------|-----------------------------|
| 4461 Multislot/HSCSD Option | M 897 158 |

ACPM or Output RF Spectrum Option (for GSM and GSM/GPRS only)

The Adjacent Channel Power Measurement (ACPM) Option provides the ETSI-specified measurements on Output RF Spectrum, both due to modulation and due to switching transients. Please refer to the 4400 data sheet for specifications. The measurements are explained in the sections about ACPM Modulation Spectrum Measurements and ACPM Transient Spectrum Measurement, respectively.

| Option | Willtek order number |
|------------------|----------------------|
| 4475 ACPM Option | M 897 163 |

NOTE

This option is already included with the 4400M.

NOTE

Adjacent channel power measurements (ACPM) are already included with the CDMA2000 1xRTT System Option.

AM Signal Generator (for GSM and GSM/GPRS only)

The Amplitude Modulation (AM) Signal Generator is required by some manufacturers for testing and alignment of mobile phones. The functionality is explained in the section about the non-call mode.

| Option | Willtek order number |
|---------------------------------|----------------------|
| 4481 AM Signal Generator Option | M 897 165 |

Little aids for speeding up tests**Operating the 4400 over a LAN with the TCP/IP Option**

The 4478 TCP/IP Option allows you to connect the 4400 to a LAN. This method has multiple benefits:

- It allows you to control the 4400 from an external PC (resolving the need for a GPIB card in the PC).
- You can store result files on a remote PC from within your RAPID! application or from the file manager (see sections ["File menu" on page 91](#) and ["Commands for input/output handling" on page 132](#)).
- You can access files on the 4400 from a remote PC (requires additional NFS client software on the PC).
- You can speed up software updates and avoid bothering with floppy disks. See section ["Initiating a software update from a remote PC" on page 150](#).

The 4478 TCP/IP Option adds remote access capabilities to your 4400 from the software side. The option includes an ethernet card to physically connect the 4400 to your LAN. This card can be plugged into the PCMCIA card slot of the 4400 and is fitted with a plug for an Ethernet connection (10 Base T).

NOTE

The 4400 supports only a limited number of different Ethernet card types. It is strongly recommended that you use Willtek's ethernet card.

| Option | Willtek order number |
|--------------------|----------------------|
| 4478 TCP/IP Option | M 248 632 |

Turbo Option (for GSM and GSM/GPRS only)

High throughput is required in mobile production lines and many service factories. Transmitter test speed can be doubled at least with the Turbo Option.

NOTE

This option is not available for the 4400S and the 4403.

| Option | Willtek order number |
|-------------------|----------------------|
| 4476 Turbo Option | M 248 359 |

Test SIMs (for GSM, GSM/GPRS, EDGE and WCDMA)

A Test SIM is a Subscriber Identification Module programmed with a specific mobile phone number, the so-called Test-IMSI (International Mobile Subscriber Identity). A Test SIM is available from Willtek, with the IMSI set to 001-01-0123456789. The 4400 knows this IMSI and uses it to page the mobile under test.

The Test SIM can also be used with other mobile phone testers from Willtek.

NOTE

When a mobile is tested without a SIM card, only emergency calls are possible.

| Option | Willtek order number |
|---|----------------------|
| 1100 Test SIM | M 860 188 |
| 1102 USIM test SIM card (for WCDMA only) | M 860 173 |

Mobile-specific RF adapters

The cable connection between a mobile under test and the 4400 requires a mobile-specific connection. Some manufacturers provide a mobile-specific cable or a test adapter. For some mobile phones, Willtek also provides some RF adapters. These RF adapters have a mobile-specific connector on one end and the Willtek RF-click connector on the other one. Please consult with your local dealer concerning available RF adapters.

By using the mobile-specific RF adapters, it is very easy to establish a cable connection between the mobile under test and the 4400. Section [Cabling](#) provides you with all the technical background information.

NOTE

To connect any RF adapter to the 4400's N-type RF in/out socket, you will need the optional RF-click to N-type adapter cable.

Antennas for use on the 4400

When there is a need to perform quick basic tests on a mobile, an antenna might be good to have. However, there are a couple of drawbacks using an antenna for real measurements. Section [Cabling](#) provides you with all the technical background information.

If you need to perform reproducible measurements without having the possibility of using a RF adapter cable, we recommend to work with the optional RF Shield. This is a metal box that covers the mobile under test and contains the Antenna Coupler. This Antenna Coupler carries the mobile under test and provides antennas for all mobile phone frequency ranges.

The RF Shield keeps radio interference away from the air connection. Together with a coupling loss table, the achievable measurement quality is as good as using a cable.

The RF Shield is available with either a TNC-type or N-type connector.

NOTE

The RF shield package contains both the RF Shield and the Antenna Coupler.

Antennas on the
4400



Antenna Coupler



RF Shield



| Accessory | Willtek order number |
|---|-----------------------------|
| Antenna for GSM 900 with TNC connector | M 860 261 |
| Antenna for GSM 1800/GSM 1900 with TNC connector | M 860 262 |
| 4916 Antenna Coupler | M 248 641 |
| 4921 RF Shield (N) (including RF cable N – N) | M 248 346 |
| Rear panel for RF Shield customisation | M 300 850 |
| RF Shield Package (4921 RF Shield, 4916 Antenna Coupler) | M 248 349 |

MS Power Supply Option and Current Measurement Option

The MS Power Supply is an option that simulates the power supply (battery) of the mobile under test. Some of the benefits are:

- It can be used to replace an extra power supply, which may be expensive and consumes space on the desktop or in a rack.
- As with any simulated battery, you can set operating, under or overvoltages in a wide range to check the mobile's reaction to that on the RF side.

In conjunction with the Current Measurement option, there are even more applications feasible to test the mobile phone:

- You can measure the average and peak current the mobile draws from the battery (e.g. when it transmits a burst).
- Even a current vs. time measurement with graphical display can be obtained.
- Last but not least, the average power requirement is calculated from the measurement results.

| Option | Willtek order number |
|--|-----------------------------|
| MS Power Supply Option | M 248 355 |
| MS Current Measurement Option (requires MS Power Supply Option) | M 248 356 |

Rack installation, cases

Rack Mount Set

Your 4400 can easily be mounted in a 19 inch rack. The optional Rack Mount Set comprises all the required material including a 19 inch front panel (5 HU) with all necessary cutouts for cables etc.

| Option | Willtek order number |
|---------------------|-----------------------------|
| 4490 Rack Mount Set | M 378 260 |

Hard cases

The "Carrying Case" for the 4400 is a hard case that is an ideal protection whenever the 4400 needs to be shipped or transported. It also is a safe place to store the test set.

Wheels and handle of the carrying case ease transportation. The telescopic handle can be retracted when not in use.

Color: White, with black wheels and handle.

Dimensions: 700 x 400 x 360 mm (1.040 x 400 x 360 mm with handle fully pulled out).

| Option | Willtek order number |
|---------------|-----------------------------|
| Carrying Case | M 300 808 |

TD-SCDMA Non-Call Mode

2

This chapter provides task-based instructions for using the nonsignaling mode (Generator/Analyzer) within the 4450 TD-SCDMA System Option Non-Call Mode. Topics discussed in this chapter are as follows:

- "Overview" on page 38
- "RF Generator" on page 39
- "RF Analyzer for modulation quality" on page 41
- "RF Analyzer for power" on page 45
- "RF Analyzer for spectrum" on page 47
- "Softkeys of the RF Analyzer" on page 53
- "System parameters" on page 54
- "TD-SCDMA basics" on page 62

Overview

The Generator/Analyzer mode of the 4400 provides test and measurement tools for asynchronous measurements. These are measurements without requiring a call setup. This test mode offers features required to tune a TD-SCDMA phone, or user equipment (UE), in a production or high-level service environment. The RF Signal Generator generates CW (Continuous Wave), TD-SCDMA and Burst signals which can be used for various testing purposes. The RF Analyzer allows basic TD-SCDMA measurements, e.g. output power, frequency error, modulation spectrum, Occupied Bandwidth (OBW), Adjacent Channel Leakage Power Ratio (ACLR) and Spectrum Emission Mask (SEM).

The description of the Generator/Analyzer mode in this chapter is divided into the following sections:

- [RF Generator](#) – This section describes the menu of the RF Signal Generator.
- [RF Analyzer for modulation quality](#) – This section describes the menu providing measurement results for assessing the modulation quality.
- [RF Analyzer for power](#) – This section describes the menu providing power measurements.
- [RF Analyzer for spectrum](#) – This section describes the menu providing different spectrum measurements.
- [Softkeys of the RF Analyzer](#) – This section describes the softkeys available on the RF Analyzer menus as well as parameter fields which the RF Analyzer menus have in common.
- [System parameters](#) – The most frequently used parameters can be adjusted to your needs in the respective generator or analyzer menu. Some additional parameters are available that affect the generated signal and the measurement views. These parameters can be modified in the parameter menus.

To gain access to the menus of the Generator/Analyzer tool proceed as follows:

- 1 Go to the Welcome menu (see following picture).
- 2 Push the marker tab for the TD-SCDMA system option.
- 3 Push the **Gen/Ana** softkey.

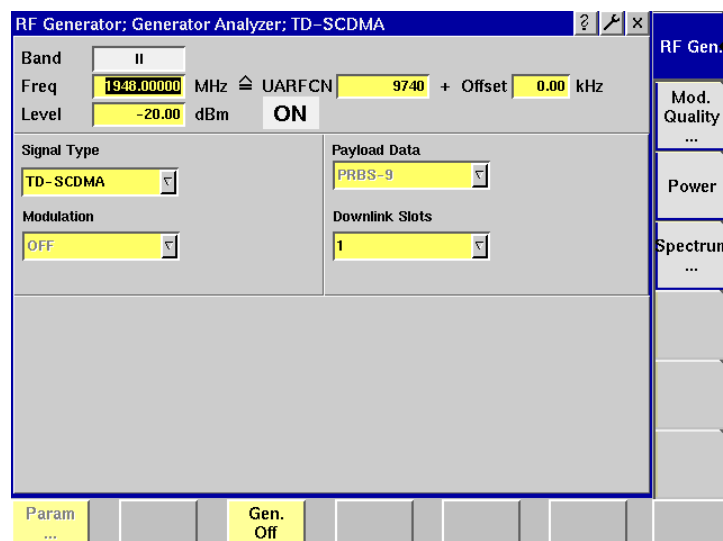


RF Generator

To gain access to this menu, push the [RF Gen.] marker tab while working with the Generator/Analyzer. Also, after pushing the **Gen/Ana** softkey on the Welcome screen as described in [Overview](#) the RF Generator menu will be selected by default.

This menu allows you to set the parameters of the RF Generator.

The RF Generator can be set to generate radio signals at a defined carrier frequency. Supported signal types are Continuous Wave, Burst and TD-SCDMA Modulation can be QPSK or none.



What it does

The 4400 can be set up to generate a typical base station signal but without supporting any exchange of signaling messages. The generator can be used to test the receiver of the mobile phone in a special test (engineering) mode. It can also be used to provide a frequency and timing reference which the phone may require before transmitting any signal in test mode.

The channels sent by the 4400 generator are Primary Common Control Physical Channel (P-CCPCH), Secondary Common Control Physical Channel (S-CCPCH) and Dedicated Physical Channel (DPCH). The respective level of each channel can be set up in the Parameter menus. For further details on setting and modifying system parameters please refer to ["System parameters" on page 62](#).

Signal Generator parameters

Band — Display field for the frequency band. This field is related to the Frequency field and the UARFCN (UTRA Absolute Radio Frequency Channel Number) field. After entering the frequency or the UARFCN the associated operating band will be displayed here.

NOTE

For details regarding TD-SCDMA frequency ranges, check with section ["TD-SCDMA basics" on page 62](#).

Freq — Entry field to set the output frequency of the RF generator.
Entry ranges: 430 to 500 MHz, 800 to 100 MHz and 1700 to 2300 MHz. The default is 1950 MHz.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section ["UARFCN" on page 63](#). If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Offset — Entry field for the offset from the channel frequency specified. The value entered here sets off the signal relative to the centre frequency.
Entry range: -100 to 100 kHz in increments of 10 Hz.

Level — Entry field for the RF output power level of the RF generator. The output signal will be available on the RF in/out connector.
Entry range: -120.0 to -10.0 dBm in increments of 0.1 dB. The default is -60.0 dBm.

NOTE

The level set in this entry field will only be correct when the load impedance is 50 Ω .

ON/OFF — Display field for the current output status of the RF Generator. When ON is displayed, RF signals are output; when OFF is displayed, the RF Generator is silent.
To toggle between the two states, use the **Gen. Off** or **Gen. On** softkey respectively.

Signal Type – Selection field for the type of signal. Selections possible: CW, Burst and TD-SCDMA. If you select TD-SCDMA here, QPSK modulation will be activated and the Modulation field will be greyed out.

Modulation – Selection field for the modulation type. Selections possible: OFF and QPSK.

NOTE

Turning the modulation off means that the RF output signal generated by the 4400 will be unmodulated, i.e. a pure carrier sine wave.

Payload Data – Selection field for the physical data pattern for CW and burst signals. Selections possible: PRBS-9 (PRBS = Pseudo Random Bit Sequence), PRBS-15, PRBS-23, 00000000, 11111111, 10101010, 11001100.

Downlink Slots – Selection field for the number of downlink slots to be used for TD-SCDMA modulation. Selections possible: 1 to 6.

RF Analyzer for modulation quality

With this option the RF Analyzer offers a method for verifying a UE's modulation quality, i.e. its modulation accuracy, in non-call mode.

To gain access to the Modulation Quality menu, push the [Mod. Quality] marker tab while working with the Generator/Analyzer.

This menu displays the results of the modulation quality measurements of the RF signal received at the **RF IN/OUT** connector. The display includes all parameters essential for assessing the UE's modulation accuracy, e.g. EVM (Error Vector Magnitude), Magnitude Error, Phase Error, Frequency Error, Rho, I/Q Offset, and I/Q Imbalance. For details on these quality assessment parameters refer to "Reading the results" on page 42 in this section.

| Modulation Quality; Generator Analyzer; TD-SCDMA | | | | | | Count | 42 | Back... |
|--|---------|----------------|---------|----------|--------------------------|--------------------------|-------|---------|
| | Curr. | Min. | Avg. | Max. | | | | |
| Power RMS [dBm] | -15.527 | -15.529 | -15.523 | -15.519 | | Status | sync. | |
| Power Peak [dBm] | -10.130 | -10.136 | -10.130 | -10.123 | | | | |
| EVM RMS [%] | 1.839 | 1.800 | 1.908 | 2.092 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| EVM Peak [%] | 8.074 | 7.863 | 8.191 | 9.495 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Mag. Error RMS [%] | 1.196 | 1.189 | 1.194 | 1.202 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Mag. Error Peak [%] | 8.027 | 7.826 | 7.947 | 8.111 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Phase Error RMS [deg] | 0.800 | 0.773 | 0.852 | 0.984 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Phase Error Peak [deg] | 3.273 | 2.312 | 2.873 | 3.819 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Frequency Error [Hz] | 2000.6 | 1998.6 | 2000.2 | 2002.1 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Rho | 0.99966 | 0.99956 | 0.99964 | 0.99968 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| I/Q Offset [dB] | -56.940 | -74.968 | -58.931 | -54.398 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| I/Q Amp. Imbalance [dB] | -71.039 | -100.07 | -66.689 | -61.030 | <input type="checkbox"/> | <input type="checkbox"/> | | |
| Frequency | | 1952.00000 MHz | | UARFCN | | 9760 | | |
| Scrambling Code | | 0 | | UL slot | | 1 | | |
| Param ... | | Gen. Off | | Unfreeze | | Reset | | |

As mentioned above, the RF Analyzer takes measurements in non-call mode, without any call processing functions.

Any [Coupling loss](#) table defined in the appropriate menu will be used for non-call measurements as well.

Before starting a measurement, the radio frequency to be measured on has to be entered. There are entry fields available for measurement parameters in the lower area of the menu.

Modulation quality parameters

Freq — Entry field to set the UE's output frequency. This field is linked to the UARFCN field. If you enter a frequency value here, the corresponding UTRA Absolute Radio Frequency Channel Number will be calculated and displayed in the following field.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section ["UARFCN" on page 78](#). If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Status — Display field for the synchronisation status: no sync, lev. trigger (waiting for the first burst) or sync.

Scr. Code — Entry field for the scrambling code used for scrambling the TD-SCDMA signal.

UL Slot — Selection field for the uplink slots to be measured. The 4400 checks how many uplink slots the mobile under test sends and displays the relevant numbers in this field. You can then select number of the uplink slot to be tested.

Reading the results

The modulation quality parameters are displayed in a table showing the current, minimum, average and maximum values of the measurements taken. The Count field shows the number of iterative tests over which average values are calculated. To the right-hand side of the table a bar display allows for a quick and easy identification of limit violation by providing violation indicators. For a detailed description of the bar display refer to section ["Display fields" on page 25](#).

Power RMS [dBm] — Shows the RMS value (i.e. mean value) for the output power of the UE's signal in dBm. For a detailed description of power measurement result and limit values refer to ["RF Analyzer for power" on page 45](#).

Power Peak [dBm] — Shows the peak output power of the UE's signal in dBm. For a detailed description of power measurement result and limit values refer to ["RF Analyzer for power" on page 45](#).

EVM RMS [%] — Shows the RMS value of the Error Vector Magnitude, i.e. the RMS value of the error vector measured, divided by the RMS value of the reference signal, expressed as a percentage value. The EVM measures the modulation quality of the RF signal transmitted by the UE. It shows the deviation of the measured signal vector from an ideal, calculated signal vector, i.e. the error vector (magnitude and phase).

As the error vector value changes continuously and besides the peak value a mean value is needed for facilitating the assessment of modulation quality, the root mean squared Error Vector Magnitude is determined as a measure of modulation accuracy at the decision points. The RMS value is an average of all the decision points (symbols) across a whole measurement interval.

The EVM value shall not exceed 17.5% at an output power of equal to or greater than -20 dBm and a power step size of 1 dB under normal operating conditions.

EVM Peak [%] — Shows the peak value determined for the EVM (Error Vector Magnitude). The EVM is expressed as a percentage value.

Note: The error vector can be broken down into a magnitude and a phase component; see following result fields.

Mag. Error RMS [%] — Shows the RMS value determined for the difference between the measured vector's and the ideal vector's magnitude, expressed as a percentage value. The root mean squared magnitude error is a measurement of the error in the mobile's transmit signal size (magnitude) at the decision points.

Mag. Error Peak [%] — Shows the peak value determined for the magnitude error. The value is normalized to the magnitude of the ideal modulated signal vector and expressed as a percentage value.

The magnitude error value shall not exceed 50%.

Phase Error RMS [deg] — Shows the RMS value determined for the phase difference, i.e. for the angle difference, between the signal vector measured and the ideal signal vector. The root mean squared phase error is a measurement of the phase component of the vector error of the UE's transmit signal at the decision points. It is expressed in degrees.

The phase error RMS value shall not exceed $\pm 10^\circ$.

Phase Error Peak [deg] — Shows the peak value determined for the phase error, expressed in degrees.

The phase error peak value shall be in the range of $\pm 45^\circ$

Frequency Error [Hz] — Shows the value determined for any difference between the UE's RF-modulated carrier frequency and the assigned carrier frequency in Hertz.

According to TD-SCDMA specifications the UE's carrier frequency shall not deviate from the assigned frequency by more than ± 0.1 ppm. Thus the value displayed in this field shall be in the range of ± 200 Hz.

Rho — Displays the waveform quality factor, a measure of modulation accuracy. A value of 1 indicates perfect waveform quality.

I/Q Offset [dB] — Shows the I/Q origin offset. The value displayed here is the determined ratio between the I/Q offset vector and the average signal vector corrected by offset and is expressed in dB.

The I/Q offset value shall not exceed -25 dB.

I/Q Amp. Imbalance [dB] — Shows the I/Q amplitude imbalance value, which is a factor for assessing the accuracy of the I/Q modulator's modulating signal amplitude balance. It is the ratio of the power in the desired sideband carrier

produced and the undesired sideband carrier produced due to an amplitude difference between the input signals to the I/Q modulator, expressed in dB. The I/Q imbalance value shall not exceed -15 dB.

The limit values for all modulation quality assessment factors are set in the Measurement Limits Parameter menu. For details on setting or modifying systems parameters please refer to section "System parameters" on page 54.

The softkey functions are explained in section "Softkeys of the RF Analyzer" on page 53.

Constellation Display

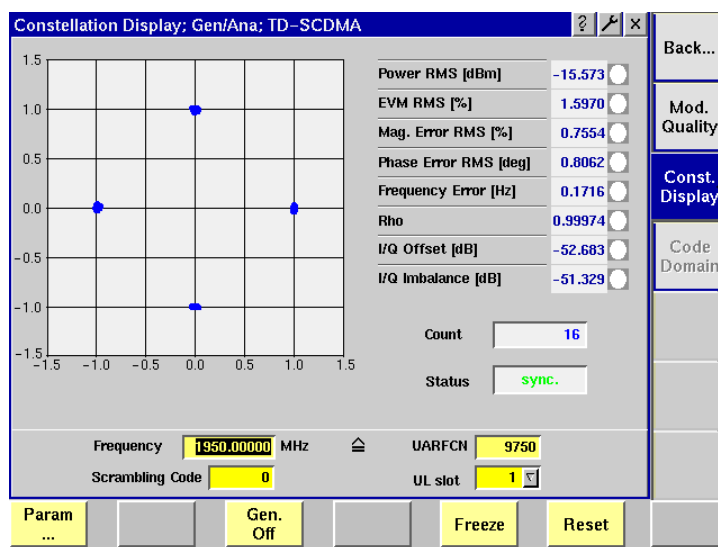
The Constellation Display screen provides a graphical analysis of the modulation quality measurement's result. It allows technicians to pinpoint faults of the terminal for a more detailed failure analysis.

In order to access the constellation display select the [Const. Display] marker tab in the Modulation Quality menu.

The parameters used for measuring and displaying the modulation quality factors on this screen are identical to the parameters used on the Modulation Quality screen. For a parameter description refer to "Modulation quality parameters" on page 42.

On the right-hand side of the constellation display there are several display fields showing the channel power value in dBm as well as the measurement results for the following modulation quality factors: EVM RMS as a percentage, magnitude error RMS as a percentage, phase error RMS in degrees, Rho, I/Q offset in dB and I/Q imbalance in dB. For details on the meaning of these factors and their associated valid limits see "Reading the results" on page 42.

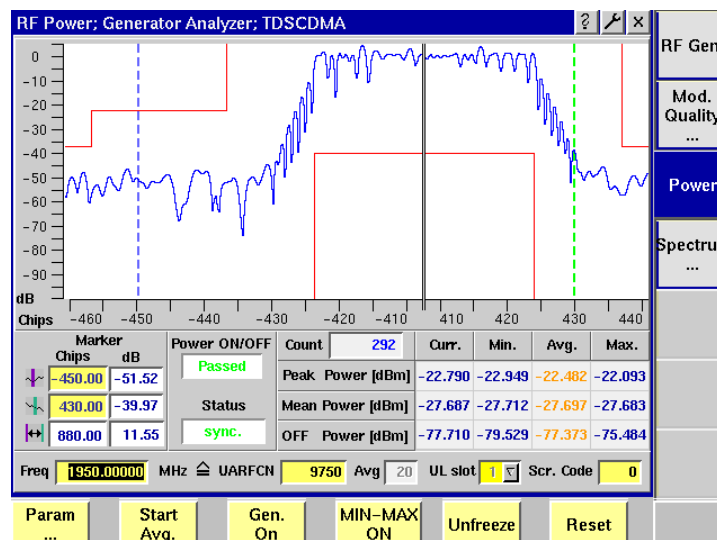
On the left-hand side there is the constellation display which shows the signal in a graphical constellation diagram. Here, the signal's magnitude and phase is shown in relation to the carrier.



RF Analyzer for power

With the Power option the RF Analyzer offers a measurement application which can be used for calibrating a UE's output power, which is an essential factor for stabilizing transmission and ensuring efficient radio resource management in TD-SCDMA systems. The purpose of the power measurements performed is for example to verify that the UE's maximum output power corresponds to the nominal maximum output power and tolerance required. An excessive maximum output power can cause interference with other channels or systems. On the other hand, if the UE's maximum output power is too small, the coverage area decreases. Both situations have an adverse effect on system capacity.

To gain access to this menu, push the [Power] marker tab while working in Generator/Analyzer mode. This menu displays the results of the power measurements of the RF signal received at the **RF IN/OUT** connector while in nonsignaling mode.



The RF Analyzer for power takes measurements in non-call mode, without any call processing functions. Any coupling loss table defined in the appropriate menu will be used for asynchronous measurements as well. For further information on the definition of coupling loss values refer to "[Coupling Loss](#)" on page 71. Before starting a measurement, the radio frequency to be measured on has to be entered.

There are entry fields available for these parameters in the lower area of the menu, just above the softkeys.

Power parameters

Freq — Entry field to set the UE's output frequency. This field is linked to the UARFCN field. If you enter a frequency value here, the corresponding UTRA Absolute Radio Frequency Channel Number will be calculated and displayed in the following field.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section "**UARFCN**" on page 63. If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Avg — Push the Softkey **Start Avg.** and use the arrow keys to go to the Avg field to define the number of last measurements results over which to average. The default value is 1, you can increase the value using the upward arrow key.

UL Slot — Selection field for the uplink slots to be measured. The 4400 checks how many uplink slots the mobile under test sends and displays the relevant numbers in this field. You can then select the number of the uplink slot to be tested.

Scr. Code — Entry field for the scrambling code used for scrambling the TD-SCDMA signal.

Reading the results

Markers

The Power Menu offers markers for easy reading of the graphical display. On the input field in the first column you set the markers to the required horizontal position. The results display in the next columns shows the measurement values obtained for the respective position. The blue and the green markers indicate the power level at the time indicated by the marker. The third marker, the delta marker, shows the difference in power level and time between the green and the blue marker.

Power ON/Off — This field indicates whether the limit check of the power vs. time measurement passes or fails. Here, a Pass or Fail verdict is displayed.

Status — Display field for the synchronisation status: no sync, lev. trigger (waiting for the first burst) or sync.

Peak Power [dBm] — Shows the peak output power of the UE's signal in dBm.

Mean Power [dBm] — Shows the mean output power of the UE' signal in dBm.

OFF Power [dBm] — Shows the off power of the UE's signal in dBm. According to TD-SCDMA specifications the transmit off power shall be less than -65 dBm.

The allowed value for the maximum output power of the UE and the associated tolerance values are defined according to its power class. Thus the measurement results have to be in accordance with the following specifications:

Table 1 Maximum output power specifications

| Power class | Maximum output power | Tolerance |
|-------------|----------------------|-----------|
| 1 | 30 dBm | 1/-3 dB |
| 2 | 24 dBm | 1/-3 dB |
| 3 | 21 dBm | 1/-2 dB |

Table 1 Maximum output power specifications

| Power class | Maximum output power | Tolerance |
|-------------|----------------------|-----------|
| 4 | 10 dBm | 4/-4 dB |

For all power measurements (peak, mean and off) a results table shows the current, minimum, average and maximum value measured over the number of measurements taken. With each measurement taken the fields Curr., Min., Avg., and Max. are updated. The Count field continually shows the number of measurements taken.

The softkey functions are explained in section ["Softkeys of the RF Analyzer" on page 53.](#)

RF Analyzer for spectrum

The group of spectrum measurements available in the Spectrum menu offers functions for measuring the RF output spectrum emission. These functions are used to measure the parameters for assessing the quantity of power that leaks outside the assigned radio channel, i.e. the off-carrier power. If this power value exceeds certain limits, interference with neighbouring channels may increase. Furthermore, system capacity may be affected. Spectrum measurements for TD-SCDMA include Occupied Bandwidth (OBW), Adjacent Channel Leakage Power Ratio (ACLR) and Spectrum Emission Mask (SEM).

To gain access to this menu, select the [Spectrum ...] marker tab while working in Generator/Analyzer mode. You can also access this menu via the **TOOLS** function key.

In the Spectrum menu three marker tabs for the different spectrum analyzer measurements are available: OBW, ACLR and SEM. On selecting [Spectrum ...] from the RF Gen or Power the OBW marker tab will be selected by default.

OBW (Occupied Bandwidth)

This measurement shows the bandwidth which the UE's signal occupies. The occupied bandwidth identifies the frequency range into which a given percentage of the signal power falls. 99% of the entire power should be spread within a range of no more than 1.6 MHz around the carrier frequency. The purpose of this measurement is to verify that the UE's occupied bandwidth is lower than 1.6 MHz based on a chip rate of 1.82 Mcps. An occupied bandwidth exceeding this value may increase interference with other channels or other systems.

Parameters

Ref. Level — Selection field for the level at which the measurement is to be performed. The field's drop-down list offers a value range from -30 dBm to 30 dBm.

RBW — Selection field for the resolution bandwidth, i.e. the filter resolution, to be used for measuring. In the field's drop-down list you can choose between 15 and 30 kHz.

Span — Selection field for the span to be used for measuring. In the field's drop-down list you can choose between ± 1.2 and ± 2.4 MHz.

Freq — Entry field to set the UE's output frequency. This field is linked to the UARFCN field. If you enter a frequency value here, the corresponding UTRA Absolute Radio Frequency Channel Number will be calculated and displayed in the following field.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section "[UARFCN](#)" on page 63. If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Band — Display field for the frequency band. This field is related to the Frequency field and the UARFCN (UTRA Absolute Radio Frequency Channel Number) field. After entering the frequency or the UARFCN the associated operating band will be displayed here.

NOTE

For details regarding TD-SCDMA frequency ranges, check with section "[TD-SCDMA frequency bands](#)" on page 62.

Avg — Push the softkey **Start Avg.** and use the arrow keys to go to the Avg field to define the number of last measurements results over which to average. The default value is 1, you can increase the value using the upward arrow key.

Reading the results

OBW — Display field for the Pass or Fail measurement verdict.

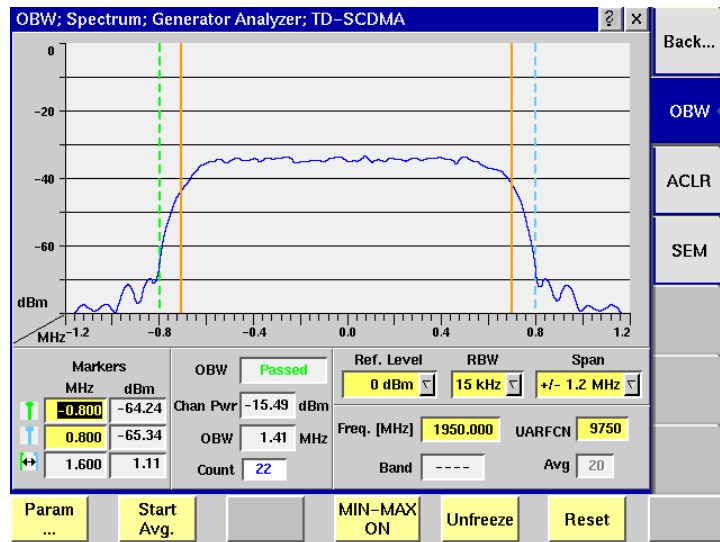
Chan Pwr — Display field for the channel power measured in dBm.

OBW — Display field for the occupied bandwidth determined in MHz. The value displayed in this field must not exceed 1.6 MHz in order to achieve a "Passed" test verdict displayed in the OBW test verdict field above.

Count — Counter field displaying the number of measurements being taken.

Markers

The 4400 offers two markers for easy reading of the graphical displays. On the input fields in the first column you set the markers to the required horizontal positions. The corresponding result fields in the next column shows the measurement values obtained for the respective positions. In the display field below the two marker fields the distance between the two markers set is displayed.



If the occupied bandwidth measured does not exceed 1.6 MHz, the test verdict in the OBW field reads "Passed". Should the values measured for the UE not comply with the specified requirements, the test verdict will read "Failed". In the display you can show the average, minimum and maximum results graph in different colours. and display and hide minimum and maximum results using the **MIN-MAX ON/OFF** softkey. For details on the usage of softkeys see "[Softkeys of the RF Analyzer](#)" on page 53.

ACLR (Adjacent Channel Leakage Power Ratio)

This measurement determines the ratio of the spectral power in the neighbouring channels to the power in the allocated channel. The purpose of this measurement is to verify that the ACLR value does not exceed TD-SCDMA limits and to ensure thereby that the mobile's modulator does not create sideband emissions that would then disturb transmission on adjacent traffic channels. According to TD-SCDMA specifications the measurement should be taken with the UE transmitting at maximum transmit power, which in terms depends on which of the four power classes the phone belongs to (see also [Table 1](#)). The 4400, however, allows this measurement at any transmit power level.

The ACLR limits according to TD-SCDMA specifications are as follows:

Table 2 UE ACLR limits

| Power class | Adjacent channel | ACLR limit |
|-------------|--------------------------|------------|
| 2, 3 | UE channel ± 1.6 MHz | 33 dB |
| 2, 3 | UE channel ± 3.2 MHz | 43 dB |

Parameters

Freq — Entry field to set the UE's output frequency. This field is linked to the UARFCN field. If you enter a frequency value here, the corresponding UTRA Absolute Radio Frequency Channel Number will be calculated and displayed in the following field.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section "UARFCN" on page 63. If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Band — Display field for the frequency band. This field is related to the Frequency field and the UARFCN (UTRA Absolute Radio Frequency Channel Number) field. After entering the frequency or the UARFCN the associated operating band will be displayed here.

NOTE

For details regarding TD-SCDMA frequency ranges, check with section "TD-SCDMA frequency bands" on page 62.

Avg — Push the Softkey **Start Avg.** and use the arrow keys to go to the Avg field to define the number of last measurements results over which to average. The default value is 1, you can increase the value using the upward arrow key.

Reading the results

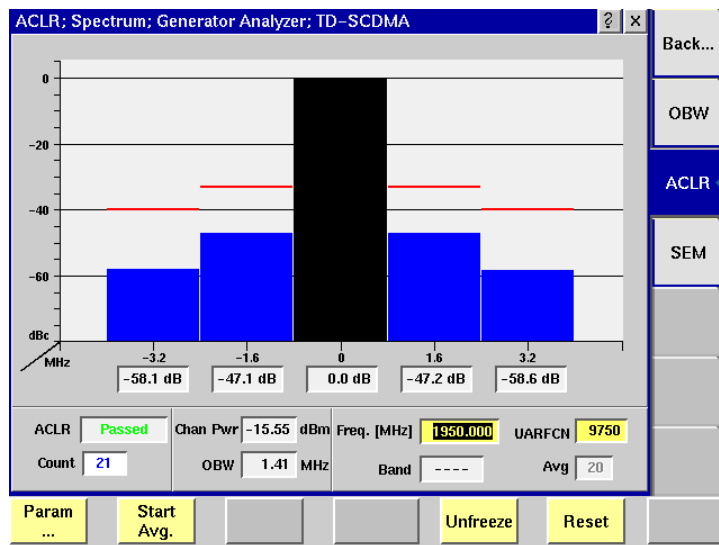
ACLR — Display field for the test verdict, Passed or Failed.

Count — Counter field displaying the number of measurements being performed.

Chan Pwr — Display field for the channel power measured in dBm.

OBW — Display field for the occupied bandwidth measured in MHz.

In the ACLR limit display fields below the measurement graph the measured ACLR results are displayed continuously as the measurements proceed. In order to achieve a "Passed" test verdict the values displayed must be below the limits specified in Table 2 on page 49. In the graphical display the columns for the adjacent channels (in blue) must not surpass the red limit lines.



In this example the ACLR values are below the upper limit shown by the horizontal line in the display. Accordingly, the test verdict reads "Passed".

The softkey functions are explained in section ["Softkeys of the RF Analyzer" on page 53](#).

SEM (Spectrum Emission Mask)

In the spectrum emission mask the signal spectrum outside the allocated channel is measured. The resulting display is split into two parts. The measurement is performed in a spacing between 0.8 and 4.0 MHz from the carrier frequency using a resolution bandwidth of 30 kHz. The TD-SCDMA specification defines upper limits depending on the frequency. The limit values used for measurements are preprogrammed in full accordance with TD-SCDMA specifications and marked red on the 4400 display.

The following table shows the TD-SCDMA spectrum emission mask requirements. The aim of the spectrum emission measurement is to verify that the power of any UE emission does not surpass the levels specified in this table.

Table 3 Spectrum emission mask requirement

| Δf in MHz | Minimum requirement | Measurement bandwidth |
|-------------------|--|-----------------------|
| 0.8 | -35 dBc | 30 kHz |
| 0.8 to 1.8 | See TS 125.102 v6.0.0 Release 6, Chapter 6.6.2 | 30 kHz |
| 1.8 to 2.4 | | 30 kHz |
| 2.4 to 4.0 | -49 dBc | 1 MHz |

NOTE

Δf is the separation between the carrier frequency and the center of the measuring filter.

Parameters

Freq — Entry field to set the UE's output frequency. This field is linked to the UARFCN field. If you enter a frequency value here, the corresponding UTRA Absolute Radio Frequency Channel Number will be calculated and displayed in the following field.

UARFCN — Entry field for the UTRA Absolute Radio Frequency Channel Number. The UARFCN designates the carrier frequency. For information on the UARFCN range supported please refer to section ["UARFCN" on page 63](#). If you enter a new value in this field, the Frequency field will be recalculated and the Band display field will be updated.

Band — Display field for the frequency band. This field is related to the Frequency field and the UARFCN (UTRA Absolute Radio Frequency Channel Number) field. After entering the frequency or the UARFCN the associated operating band will be displayed here.

NOTE

For details regarding TD-SCDMA frequency ranges, check with section ["TD-SCDMA frequency bands" on page 62](#).

Avg — Push the Softkey **Start Avg.** and use the arrow keys to go to the Avg field to define the number of last measurements results over which to average. The default value is 1, you can increase the value using the upward arrow key.

Reading the results

SEM — Display field showing the test verdict, Passed or Failed.

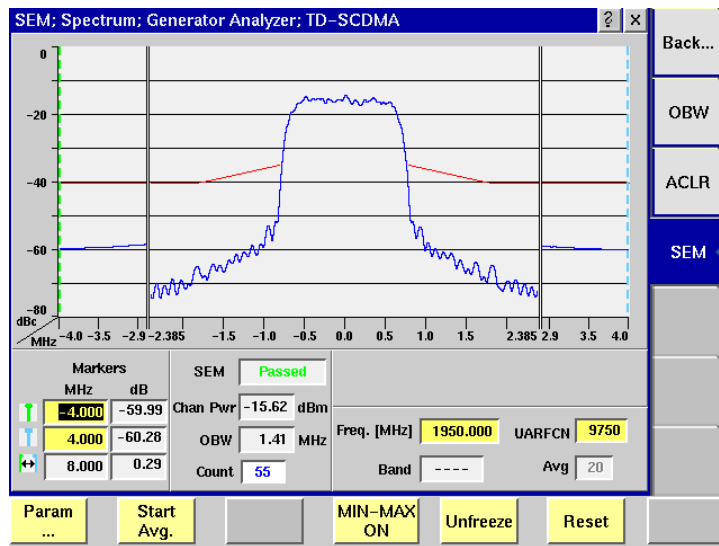
Chan Pwr — Display field for the channel power measured in dBm.

OBW — Display field for the occupied bandwidth measured in MHz.

Count — Counter field displaying the number of measurements being performed.

Markers

The 4400 offers two markers for easy reading of the graphical displays. On the input fields in the first column you set the markers to the required horizontal positions. The corresponding result fields in the next column shows the measurement values obtained for the respective positions. See following figure.



If the signal spectrum measured outside the allocated channel does not surpass the limits shown by the red limit lines onscreen, the test verdict displayed in the SEM field reads "Passed". You can display and hide minimum and maximum results using the MIN-MAX ON/OFF softkey. For details on the usage of softkeys see ["Softkeys of the RF Analyzer" on page 53.](#)

Softkeys of the RF Analyzer

The following softkeys are common RF Analyzer elements.

Param
...

This softkey leads you to the parameter screens where you can set up coupling loss factors and parameters for RF Generator operation.

**Gen.
Off**

A push on this softkey switches the [RF Generator](#) either on or off.

NOTE: This softkey is not available in the Spectrum measurements menus.

Freeze

To freeze the current display for further or detailed analysis (or for printing), push this softkey.

The softkey will change to **Unfreeze**.

NOTE: After the **Freeze** softkey has been pushed, all measurements will continue in the background, but the display will not be updated.

Reset

A push on this softkey resets the statistic evaluation of the measurement results. The measurement counter will be reset as well.

The following softkeys are specific to Power and Spectrum measurement menus (OBW, ACLR and SEM).

**Start
Avg.**

Using this softkey you can start the process of averaging the measurement results over a number of last measurements.

The number of measurements over which to average is defined in the Avg field. The Default is 1. When you push this softkey, it will change to **Stop Avg**.

**MIN-MAX
ON**

A push on this softkey either activates the display of the minimum and maximum measurement results graphs or deactivates it. When you push this softkey, it will change to

MIN-MAX OFF. If activated, the minimum and maximum results graphs are displayed in different colours (for example minimum in green, maximum in red) für visual distinction. To hide the two graphs push the **MIN-MAX OFF** softkey again.

NOTE: This softkey is not available in the ACLR menu.

System parameters

The Parameter menus allow you to access additional parameters at a more detailed level, which need not be changed in everyday use. In these menus you can set a number of system parameters or define a coupling loss curve.

To gain access to these menus, simply push the **Param...** softkey, which is available on all measurement screens, while the UE is in idle state.

Using the marker tabs, you can select between the following menus:

- **RF Gen.** – in this menu you can set the parameters for the RF generator.
- **Measurement limits** – this menu allows you to set measurement limits for modulation quality measurements.
- **Coupling Loss** – this menu allows you to compensate the losses of cables, splitters and other RF equipment. This is especially important when working with the antenna coupler.

NOTE

Each of the parameter menus includes a **Default** softkey, setting the parameters on the respective screen to their default (factory) settings.

RF Gen. In this parameter menu you can set the parameters for the RF Generator operation, e.g. the different channel parameters for slot 1 and 2 to 6.

| RF Generator; Parameter: TD-SCDMA | | | | | RF Gen. |
|-----------------------------------|-------------------------------------|---------|--------|-----|---------------|
| Slot 1 | on/off | power | data | SF | |
| P-CCPCH | <input checked="" type="checkbox"/> | 0.00 dB | PRBS-9 | fix | Meas. Limits |
| S-CCPCH | <input type="checkbox"/> | 0.00 dB | PRBS-9 | fix | |
| DPCH | <input type="checkbox"/> | 0.00 dB | PRBS-9 | 1 | |
| DwPTS | <input type="checkbox"/> | 0.00 dB | | | Coupling Loss |
| Slot 2 - 6 | | | | | |
| S-CCPCH | <input type="checkbox"/> | 0.00 dB | PRBS-9 | fix | |
| DPCH | <input type="checkbox"/> | 0.00 dB | PRBS-9 | 1 | |
| Scrambling Code | | | | | 0 |

Slot 1 In this area you can specify the signal parameters for slot 1.

P-CCPCH The Primary Common Control Channel transports synchronization and broadcast information for users. Within a cell there is only one P-CCPCH. It's frame structure differs from the downlink DPCH (Dedicated Physical Channel) in that no TPC (Transmit Power Control) commands, no TFCI (Transport Format Combination Indicator) and no pilot bits are transmitted. In order to activate the P-CCPCH for signal generation select the **on/off** button and press the **ENTER** key. If you want to deactivate the channel again, simply press the **Enter** key again. In the Power field you can specify the power level for this channel in dB. In the data field you can select the physical data pattern for CW and burst signals. Selections possible: PRBS-9 (PRBS = Pseudo Random Bit Sequence), PRBS-15, PRBS-23, 00000000, 11111111, 10101010, 11001100. In the SF column the setting for the spreading factor is displayed which in the case of this channel is fix and not selectable.

S-CCPCH The Secondary Common Control Physical Channel carries the FACH (Forward Access Channel) and the PCH (Paging Channel). The parameter fields available are identical to those of the Primary Common Control Channel.

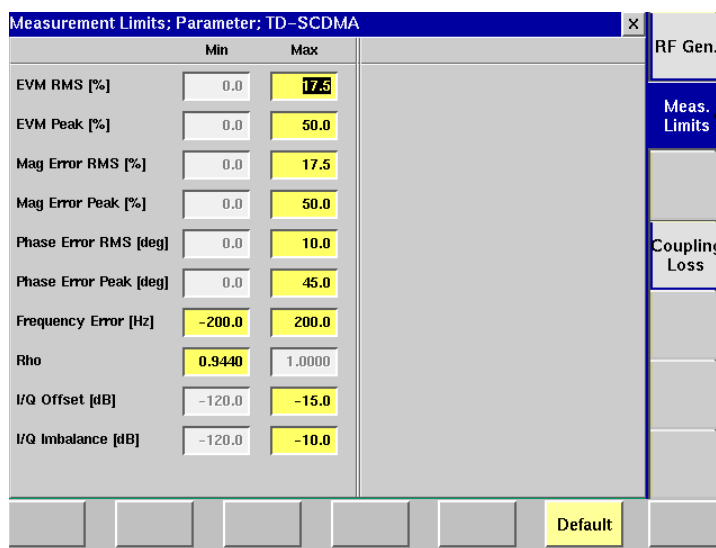
DPCH The Dedicated Physical Channel is used in UL and DL to carry the DPDCH (Dedicated Physical Data Channel) and the DPCCH (Dedicated Physical Control Channel). The DPDCH transmits the user data whereas the DPCCH transports the pilot symbols, the control commands for fast power control as well as rate information (TFCI, Transport Format Combination Indicator). The parameter fields available are identical to those of the other two channels in this area. However, you can select the spreading factor for this channel from the drop-down list of the SF field. Selections possible: 1 to 16.

DwPTS In this area you can activate the Downlink Pilot Time Slot and specify its power level. Handling of the activation button and the power entry field is identical to the handling already described.

Slot 2 - 6 In this area you can specify the signal parameters for slot 2 to 6. Here, parameter fields for S-CCPCH and DPCH are available. For a description of these fields refer to [Slot 1](#).

Scrambling Code In this field you can enter the scrambling code to be used for scrambling the signal.

Measurement limits In this parameter menu you can define limits for several measurements returning values for assessing the UE's modulation accuracy, e.g. phase error, EVM and I/Q imbalance. The defaults displayed on this screen correspond to the limits set by TD-SCDMA specifications. However, you can change the limit values in order to refine testing requirements and verify adherence to even more stringent requirements. The results of these measurements are displayed on the Modulation Quality menu and, in graphical form, on the Constellation Display.



EVM RMS [%] The EVM RMS value is an average of all the decision points (symbols) across a whole measurement interval.
Limit value according to TD-SCDMA specifications: 17.5%

EVM Peak [%] The EVM peak value indicates the difference between the measured vectors and the ideal vectors, expressed as a percentage value.
Limit value according to TD-SCDMA specifications: 50%

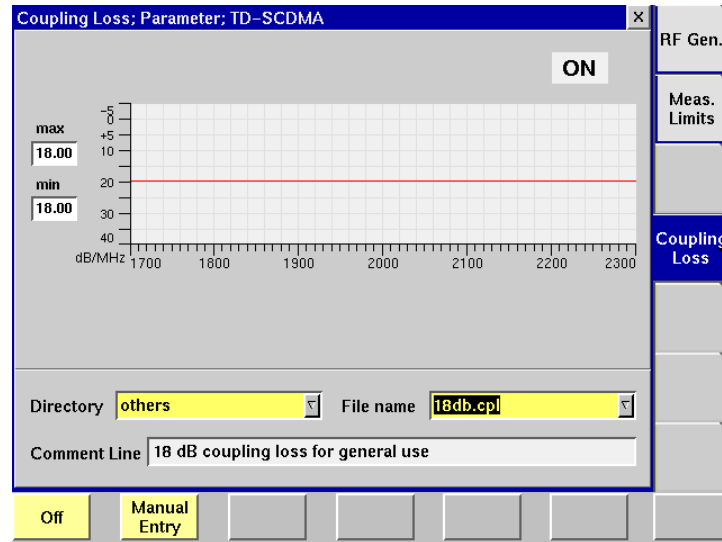
Mag Error [%] RMS The root mean squared magnitude error is a measurement of the error in the mobile's transmit signal size (magnitude) at the decision points.
Limit value according to TD-SCDMA specifications: 17.5%

Mag Error Peak [%] The magnitude error peak value indicates the difference between the measured vector's and the ideal vector's magnitude, also expressed as a percentage value.
Limit value range according to TD-SCDMA specifications: $\pm 50\%$

| | |
|-------------------------------|---|
| Phase Error RMS [deg] | <p>The root mean squared phase error is a measurement of the phase component of the vector error of the UE's transmit signal at the decision points.</p> <p>Limit value range according to TD-SCDMA specifications: $\pm 10^\circ$</p> |
| Phase Error Peak [deg] | <p>The peak phase error indicates the phase difference, i.e. the angle difference, between the signal vector measured and the ideal signal vector.</p> <p>Limit value range according to TD-SCDMA specifications: $\pm 45^\circ$</p> |
| Frequency Error [Hz] | <p>Here the limit for the frequency error measurement is set. The measurement results (with a pass/fail verdict according to these limits) are shown on the Basic menu.</p> <p>Limit value range according to TD-SCDMA specifications: ± 200 Hz</p> |
| Rho | <p>Rho is the waveform quality factor, a measure of modulation accuracy. A value of 1 indicates perfect waveform quality.</p> <p>Limit value according to TD-SCDMA specifications: 0.9440</p> |
| I/Q Offset [dB] | <p>The I/Q imbalance value is the determined ratio between the I/Q offset vector and the average signal vector corrected by offset. It is expressed in dB.</p> <p>Limit value according to TD-SCDMA specifications: -15 dB</p> |
| I/Q Imbalance [dB] | <p>The I/Q imbalance value indicates the ratio of the power in the desired sideband carrier produced and the undesired sideband carrier produced due to an amplitude difference between the input signals to the I/Q modulator, expressed in dB.</p> <p>Limit value according to TD-SCDMA specifications: -10 dB</p> |

Coupling Loss

When inserting cables, splitters, antennas or other RF equipment between the UE under test and the 4400, there will always be some attenuation or coupling loss. The attenuation associated with this loss typically varies with both the type of UE being tested and the frequency. An example of a coupler being used is the Willtek 4916 Antenna Coupler.



This menu allows compensation of that attenuation. To do so, a *.cpl file is required, containing the information about losses on specific frequencies. Those *.cpl files can be created either with the file editor within the RAPID! environment, on an external PC (and loading the file to the 4400 using a floppy disk), or from the User-defined Attenuation menu.

On the 4400, the contents of the currently selected *.cpl file will be displayed graphically. The graph shows the coupling loss for the frequency range 1700 to 2300 MHz.

On the left-hand side of the coupling loss graph, the 4400 provides two display fields giving the minimum and maximum attenuation for each frequency range.

The coupling loss parameters are stored in files; the file format is explained below.

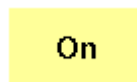
The file and directory structure allows you to store a set of parameters for each mobile and to locate the files for the phones from one vendor in one directory. The 4400 comes with a number of preinstalled directories for some of the larger manufacturers and a directory with parameter files for various "others". You can add more directories using the RAPID! file manager. The location of the coupling loss files and directories is /rapid/cpl.

How to activate a previously stored coupling loss definition

| | | | |
|--------------|-------------------------------------|-----------|----------|
| Directory | others | File name | 15db.cpl |
| Comment Line | 15 dB coupling loss for general use | | |

Directory Select the directory for the brand of the mobile. This field allows you to browse the list of coupling loss directories that are installed on the 4400. If you do not find the manufacturer of the phone under test in the list of directories, check the files in the "others" directory. The 4400 will allow you to browse through the files in the selected directory using the File name scroll field.

File name Select the corresponding file name in this field. The 4400 will load the *.cpl file and display the data stored graphically. Any comments or special instructions saved in the *.cpl file will be displayed on the Comment Line field. The files in the list are in alphabetical order; the uppercase letters appear before the lowercase letters. Coupling loss files created by the Manual Input feature (see ["Entering coupling loss values from the Manual Input menu" on page 61](#)) have a dot at the beginning so that they can be easily found at the beginning of the file list.



Push this softkey to activate the coupling loss compensation. The display field on the right top of this menu will change to ON. The headline in all test masks will now carry a clear indication that the coupling loss compensation has been activated: Coupling Loss will be shown in red letters.

How to store a coupling loss definition on the 4400

- 1 Save the *.cpl file on a floppy disk.

NOTE

The coupling loss compensation parameters for leading mobile phone types can be downloaded from the web (see above).

- 2 Insert the floppy disk into the 4400's disk drive.
- 3 Call the Welcome menu on-screen using the **ESCAPE** key.
- 4 Start RAPID! by first pushing the **TOOLS** key and then the [RAPID!] marker tab. This will call the file manager on-screen.
- 5 Using the file manager, navigate to the /io/floppy directory.
- 6 Select the file to be stored on the 4400.
- 7 Push the **ETC** softkey and then the **Copy** softkey. This will call the File Copy to menu on-screen.
- 8 Enter the path and the file name to store the file, e.g. /rapid/cpl/nokia/filename.cpl.

NOTE

The maximum length for `filename` is 8 characters.

**How to edit a coupling loss
definition already stored on the
4400**

- 9 To confirm the entry and to finally copy the file, push the **OK** softkey.
 - 10 A push on the **Exit** softkey will leave RAPID! and take you back to the Welcome menu of the 4400.
- 1 Call the Welcome menu on-screen using the **ESCAPE** key.
 - 2 Start RAPID! by first pushing the **TOOLS** key and then the [RAPID!] marker tab.
This will call the file manager on-screen.
 - 3 Using the file manager, navigate to the `/rapid/cpl` directory.
 - 4 Select the file to be edited.
 - 5 Push the **Open** softkey and then the [Edit] marker tab to start the editing process.

The following issues should be considered when using coupling loss files:

- The *.cpl files are organized as tables.
- An optional comment may be placed in the first line. A comment is indicated by two slashes: `//comment`
- Within the *.cpl file there is a section for the frequency range from 1700 to 2300 MHz.
- The section must contain at least one line. The maximum for the section is 10 lines.
- Every line consists of two values: the frequency in MHz and the corresponding coupling loss in dB. Those two values are separated by a comma (,).

Example:

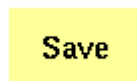
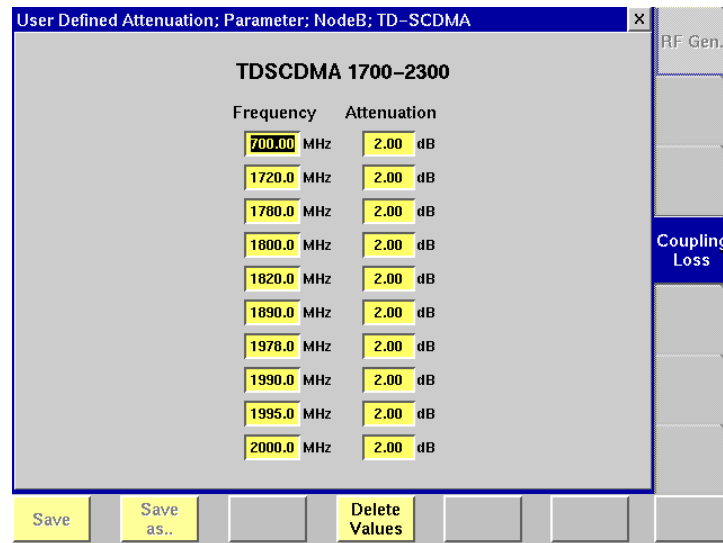
```
//Motorola P7389  
825.0,15.0  
1750.0,19.0
```

- Push the [File] marker tab to return to the file manager.
- Push the **Close** softkey to terminate any operation on the file previously opened.
- After the editing has been completed, push the **Save** softkey to store the file.
- A push on the **Exit** softkey will leave RAPID! and take you back to the Welcome menu of the 4400.

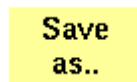
Entering coupling loss values from the Manual Input menu

This procedure allows you to enter coupling loss values and frequencies in a menu rather than invoking a text editor:

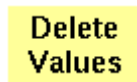
- On the Coupling Loss menu, select a coupling loss file that you want to change. Alternatively, select one on which you want to base your own file.
- Push the **Manual Entry** softkey within the Coupling Loss menu. As a result, the Manual Input menu appears. This menu allows you to enter or change pairs of frequencies (in MHz) and coupling loss values (in dB) for downlink and uplink.
- Change values by overwriting them.



When pushing the **Save** softkey, the entered values will be stored in the previously selected file.



You can also give the file a different name by pushing this softkey. The 4400 then prompts you for the file name. The file name extension is added automatically. The file name also gets an additional "." at the beginning so that the manually created files appear at the beginning of the file name list on the Coupling Loss screen. This list is ordered alphabetically. **Note:** All coupling loss files are stored in the `/rapid/cpl` directory.



You can delete a row of entered values by placing the cursor in one of the three fields of that row and then pushing this softkey.

After pushing the **ESCAPE** key, the Coupling Loss screen returns. You can now select the saved file from the list of coupling loss files.

NOTE

If you manually enter values and save them without selecting a file name, a previously existing file named `.manual.cpl` will be overwritten.

TD-SCDMA basics

In this section, you will find a short introduction to TD-SCDMA system parameters.

We will look at the following aspects of a TD-SCDMA system:

- [TD-SCDMA frequency bands](#)
- [Channel arrangement](#)
- [UARFCN](#)
- [UE power classes](#)

TD-SCDMA frequency bands

The table below shows the specifications for the various TD-SCDMA bands.

Table 4 Frequency bands

| | |
|----------|--|
| Band I | 1900 to 1920 MHz 2010 to 2025 MHz UL and DL transmission |
| Band II | 1850 to 1910 MHz 1930 to 1990 MHz UL and DL transmission |
| Band III | 1910 to 1930 MHz UL and DL transmission |

Channel arrangement

The following sections provide information concerning channel spacing, channel raster, channel number and UARFCN.

Channel spacing

The nominal channel spacing is 1.6 MHz. However, to optimise performance in a specific deployment scenario this value can be adjusted.

Channel raster

The channel raster is 200 kHz.

Channel number

The UARFCN (UTRA Absolute Radio Frequency Channel Number) designates the carrier frequency. The following table shows the definitions of the UARFCN values.

UARFCN For each band the following UARFCN range is supported.

Table 5 UTRA Absolute Radio Frequency Channel Number

| Frequency band | Frequency range | UARFCN UL and DL transmission |
|----------------|--------------------------------------|-------------------------------|
| I | 1900 to 1920 MHz 2010 to 2025 MHz | 9504 to 9596 10054 to 1021 |
| II | 1850 to 1910 MHz 1930 to 1990 MHz | 9254 to 9546 9654 to 9946 |
| III | 1910 to 1930 MHz | 9554 to 9646 |

UE power classes

Table 6 TD-SCDMA UE power classes overview

| Power class | Nominal maximum output power | Tolerance |
|-------------|------------------------------|------------|
| 1 | 30 dBm | 1 dB/-3 dB |
| 2 | 24 dBm | 1 dB/-3 dB |
| 3 | 21 dBm | 2 dB/-2 dB |
| 4 | 10 dBm | 4 dB/-4 dB |


Tools

3

This chapter provides task-based instructions for additional tools within the 4450 TD-SCDMA System Option Non-Call Mode. Topics discussed in this chapter are as follows:

- ["Overview" on page 66](#)
- ["Configuration" on page 66](#)
- ["Utilities" on page 79](#)
- ["Spectrum measurements" on page 85](#)

Overview

This chapter shows you the functionality in the Tools section of the user interface. The Tools menu can be found with a push on the Tools  function key.

The Tools menu can be divided into the following sections:

- Section ["Configuration" on page 66](#) lets you view and change the instrument configuration. This includes software updates and upgrades and interface settings.
- Section ["Utilities" on page 79](#) includes general functions such as a configurable power supply and current measurements.
- Section ["Spectrum measurements" on page 85](#) gives a short overview on the spectrum measurements available. For a detailed description refer to ["RF Analyzer for spectrum" on page 47](#).

Configuration

With the help of these menus, you may configure the interfaces of your 4400, check whether a specific option has been installed or install options and new software releases.

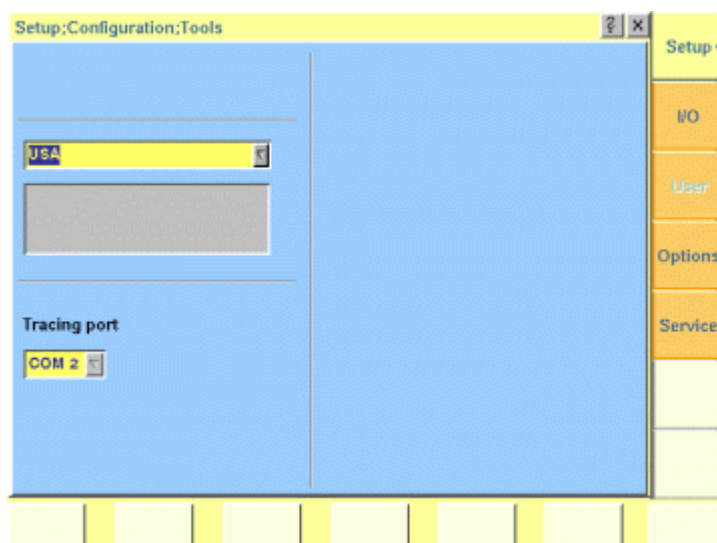
This section consists of the following parts:

- ["Setup"](#) – In this menu, you may select the language of the external keyboard as well as some other settings.
- ["I/O configuration"](#) – This menu allows to set the GPIB address and the external printer connected.
- ["Options"](#) – This menu provides an overview of the options installed on this 4400 and also allows to install further options.
- ["Service and software update"](#) – In this menu, you will find all details about the software revisions installed in the various subsystems of your test set. These details are very useful when discussing problems with the Willtek product specialists. Furthermore, this menu allows to install new software releases or to return to software releases installed previously.

Access to the configuration menus

Push the **TOOLS** function key. The tool bar provides access to the configuration menu with a push on the [Config] marker tab.

Setup Push the [Setup] marker tab to gain access to this menu.



Settings of the Setup menu

External Keyboard — This selection field allows to set the language of the external keyboard connected.

The settings possible are:

- USA (default)
- BelgiumFrench
- BelgiumDutch
- CanadaFrench
- CanadaEnglish
- Denmark
- France
- Germany
- Italy
- Japan
- LatinAmericaSpanish
- LatinAmericaPortugal
- Netherlands
- Norway
- Portugal
- Spain
- Sweden
- SwissFrench
- SwissGerman
- UnitedKingdom

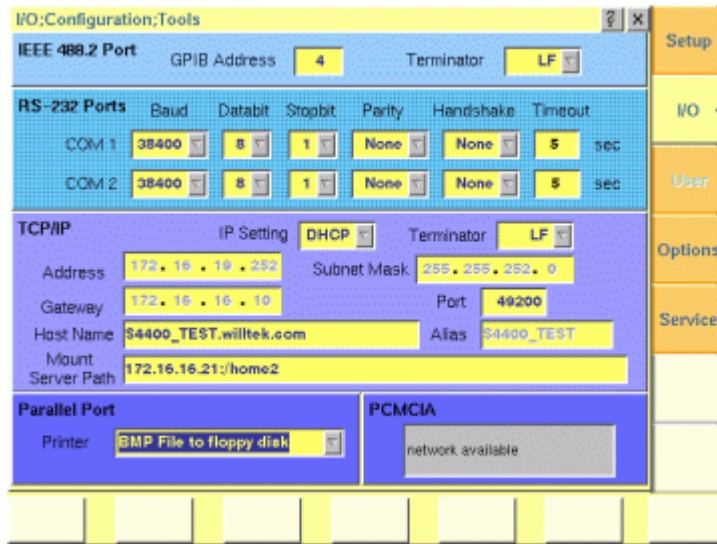
NOTE

The language set for the external keyboard affects the mapping of special keys and characters. This may in some cases affect the ["Keyboard mapping table"](#).

NOTE

Any change on this selection field will require a reboot of the test set. You will be reminded by the `Please reboot` message appearing on the display field just below this selection field.

I/O configuration Push the [I/O] marker tab to gain access to this menu.



This menu consists of five areas:

- **IEEE 488.2 Port** is the area where you set the GPIB address and terminator to be used by the 4400.
- **RS-232 Ports**. These entry fields do not have a function in this software release.
- **TCP/IP**. In this area, you can configure the TCP/IP port of the 4400 and define an address on the network that you want to access from the 4400.
- **Parallel Port** provides a selection of printer drivers (or alternatively a file format) to print out the screens of the 4400.
- **PCMCIA** informs you about the PCMCIA connections currently available.

IEEE 488.2 port settings This area of the menu gives access to the following parameters:



GPIB Address — Entry field to set the GPIB address of the 4400 in case the test set is controlled externally (and not used as the GPIB controller). Entry range: 0 to 31. Default is 4.

Terminator — Selection field to set the terminator for GPIB commands. Settings possible are: LF (default), CR and CR/LF.

TCP/IP port settings

The 4478 TCP/IP Option provides a range of additional applications to work with the 4400 over a local area network (LAN):

- Software updates (see section "Service and software update" on page 76)
- Remote control of the 4400 using "SCPI" commands
- Access to files on the 4400 using an NFS connection
- Access to files on the network from the 4400 using an NFS connection (see sections "File menu" and "Commands for input/output handling" in the RAPID! chapter)

The following conditions must be met in order to use these features and to set up the TCP/IP parameters of the 4400:

- The 4478 TCP/IP Option must be installed. This option is available through Willtek and can be installed in the field.
- The Ethernet PC Card must be installed. This is a PCMCIA card available through Willtek. The card must be inserted upside down into the left PCMCIA card slot. Please refer to the manual of the Ethernet PC Card for more information.

If both are installed, the 4400 runs TCP/IP software and also an NFS (network file server) client/server application to allow access both from and to the 4400 over a local area network (LAN).

The following TCP/IP and NFS parameters can be configured in this area if the TCP/IP option is available and a network card has been installed:

| TCP/IP | | IP Setting | Terminator |
|-------------------|-------------------------|-------------|---------------|
| Address | 172.16.19.252 | Subnet Mask | 255.255.252.0 |
| Gateway | 172.16.16.10 | Port | 49200 |
| Host Name | \$4400_TEST.willtek.com | Alias | \$4400_TEST |
| Mount Server Path | 172.16.16.21:/home2 | | |

NOTE

The LAN interface card (Willtek's Ethernet PC Card) must be plugged into the PCMCIA slot BEFORE turning on the 4400 because the driver software is loaded only during the boot-up phase.

NOTE

If you want to access directories and files that reside on a computer on the network, that computer must run an NFS server software. Computers running a UNIX operating system often have NFS server software already installed. NFS server software for Windows- and for MacOS-based computers is available as a third-party product.

NOTE

If you want to access directories and files on the 4400 from a remote computer, that computer must run an NFS client software. Most computers running a UNIX operating system have NFS client software already installed. NFS client software for Windows- and for MacOS-based computers is available as a third-party product.

Terminator — This selection field is used to set the terminator for SCPI commands when controlling the 4400 via LAN. The allowable settings are: **LF** (default), **CR** and **CR/LF**.

Address — If IP addressing is set to fixed, enter a valid IP address for the 4400 in this field, in the format **a.b.c.d**; **a** through **d** are numbers in the range 0 to 255. The address must not already be in use by another device. Refer to the administrator of your local area network for the assignment of a valid and accessible IP address within your local area network.

If IP addressing is set to DHCP, the IP address of the 4400 is automatically derived from a DHCP server.

NOTE

If you enter a new address you need to reboot the 4400 for the changes to take effect.

Port — Port number in the 4400 for access to the 4400 when controlling the instrument from a remote computer via LAN using the SCPI command set. The default is 49200. For successful remote control via LAN, you will have to use the same port number in the application controlling the 4400 as defined in this input field.

Mount Server Path — You can access a server disk on the network from certain applications on the 4400 if you enter a valid network address for that disk here. A valid address consists of a server IP address followed by a colon and a path (directory or folder) that is accessible from the 4400 over the network. A typical example for a server path is **172.16.16.21:/disk3/4400_sw**. Typical applications include software updates and loading files from and to the network using the RAPID! environment. In the File manager, the mounted disk will appear as **/io/server**.

NOTE

The 4400 is not aware of the address for your local domain name server, therefore you have to provide the IP address (in the a.b.c.d format) rather than a symbolic computer name.

NOTE

To access a remote computer, that computer must grant access rights for the appropriate service, e.g. if you want to write data to the disk on the network the operating system on that computer must be set up to grant write access to all users for the respective directory on that disk.

NOTE

This field allows you to mount only one disk. If you need to access server disks from a RAPID! program or from a remote control application (via GPIB or TCP/IP), you can mount more than one disk from your application program.

TCP/IP Troubleshooting

- 1 The network connection does not appear to work at all.
 - Ensure that the network card is installed when you switch on the 4400. You can remove and reenter the same PCMCIA card once the 4400 has booted.
 - Check if the 4400 sounds a double-beep while booting. If it does, it found the network card; if not, the PCMCIA card may be defective.
 - Check if the cables are tightly connected to the PCMCIA card and to the Ethernet wall outlet.
 - Ensure that the PC supports the TCP/IP protocol stack. When a network card is installed, some versions of the Microsoft Windows operating system default to another protocol such as NETBEUI. Windows can run several protocol stacks simultaneously, so installing the TCP/IP stack usually does not affect other network connections and applications.
 - Check if the IP address is valid and not already in use by another device on the network. You can check this with your computer if it is on the same network as the 4400: On a PC running the Microsoft Windows operating system, open a command shell box (usually under Start-Programs-Accessories) and enter the following command line: **ping <address>** where <address> is the IP address of your 4400. The ping should time out if the 4400 is switched off, but there should be responses from your 4400 if it is up and running. If there are pings while the 4400 is switched off, there is another device (such as a PC or another 4400) on the network; there must only be one device per address on the network. If there is no ping even when the 4400 is switched on, the address is not a valid IP address in this network or network segment, or the 4400 is not properly connected to the network (check cabling).
- 2 I can enter a new IP address, but the 4400 does not respond to messages over the Ethernet connection. After making changes to the IP parameters, they are only used after rebooting the 4400.
- 3 Although I have entered an IP address for the 4400 and a server path, I do not see the mounted device in the list of directories (e.g. when installing a new software version).
 - Check if the IP address is valid and not already in use by another device on the network. You can check this with your computer if it is on the same network as the 4400: On a PC running the Microsoft Windows operating system, open a command shell box (usually under Start-Programs-Accessories) and enter the following command line: **ping <address>** where <address> is the IP address of your 4400. The ping should time out if the 4400 is switched off, but there should be responses from your 4400 if it is up and running. If there are pings while the 4400 is switched off, there is another device (such as a PC or another 4400) on the network. There must only be one device per address on the

network.

If there is no ping even when the 4400 is switched on, the address is not a valid IP address in this network or network segment, or the 4400 is not properly connected to the network (check cabling).

- Check if the remote PC runs an NFS server software. UNIX-like operating systems typically include an NFS server; applications for Windows are commercially available.

Parallel port settings



Printer — Selection field. Selects the type of the external printer, connected to the 4400's parallel port ("CENTRONICS").

The possible settings are:

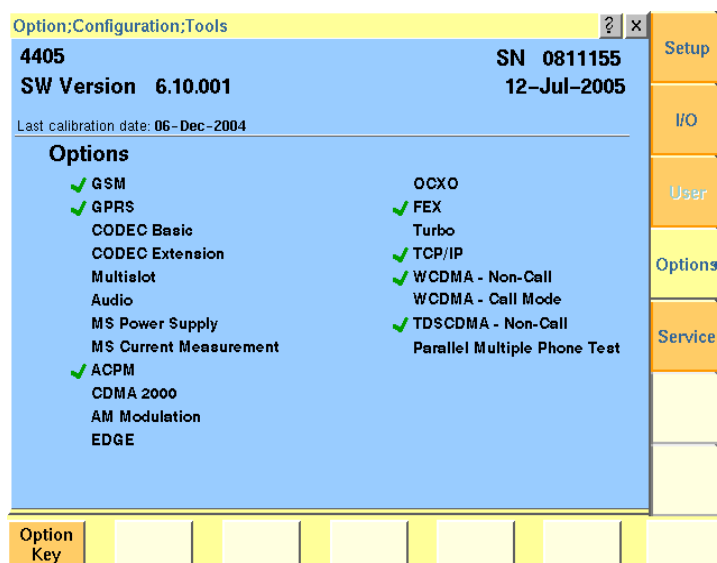
- HP DeskJet 400 (default)
- HP DeskJet 680
- HP Laserjet
- Canon
- Epson Stylus Color
- Epson Stylus Color II
- Epson Stylus Color IIs
- Epson Stylus Color 200
- Epson Stylus Color 400
- Epson Stylus Color 500
- Epson Stylus Color 600
- Epson Stylus Color 800
- Epson Stylus Color 1500
- Epson Stylus Color 1520
- Epson Stylus Color 3000
- Epson Stylus Pro
- Epson Stylus Pro XL
- Epson Stylus Pro XL+
- BMP File to floppy disk

Note: Setting BMP File to floppy disk will output the current screen as a bit-map file (*.BMP) on a disk inserted into the 4400's disk drive. A prompt will appear on-screen when the 'printing to disk' process starts. The printing can be aborted by pushing the **ESCAPE** function key or the corresponding key of the external keyboard as long as the prompt is visible.

The bit-map file may then be used with standard word processing or DTP software.

The size of a single bit-map file is approx. 305 kB; therefore four files will fit on a disk.

Options Push the [Options] marker tab to gain access to this menu.



This menu consists of two areas:

- The upper area of the menu is called the basic info area. Here you will find the basic information about your test set, like the software release currently installed.
- **Options** is the larger area of the menu that informs you about all options available and the ones installed. See "[Accessories and options](#)" on page 29 for more details on available options.

The **Option Key** softkey is used to install additional options (see below).

Basic info area This area of the menu gives the following basic information:



4400 — In the upper left corner of this area, the 4400 model is indicated (4400, 4403, 4405 or 4407).

SN 0811155 — In the upper right corner of this area, the serial number of this test set is displayed.

NOTE

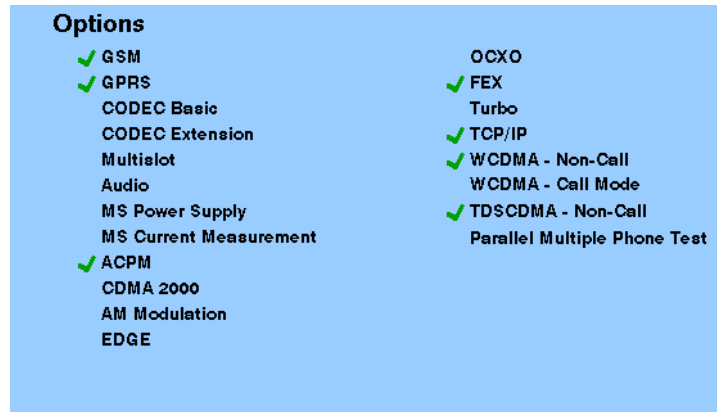
The availability of an option may be dependent on the serial number.

SW Version 6.10.001 — Here, the software version currently in use on this test set is displayed.

12-Jul-2005 — This is the release date of the installed software version.

Last calibration date: 06-Dec-2004 — Here, the date of the last calibration of the test set will be displayed.
If a calibration date of the whole unit is not found, the date of the last calibration of the RF section is indicated.

Options The options area provides a quick overview of all options installed.



GSM — When a tick is shown in front of this list entry, the GSM System Option software is installed.

GPRS — When a tick is shown in front of this list entry, the GPRS System Option software is installed.

CODEC Basic — When a tick is shown in front of this list entry, the Basic CODEC Option is installed.

CODEC Extension — When a tick is shown in front of this list entry, the CODEC Extension Option is installed.

Multislot — When a tick is shown in front of this list entry, the HSCSD-Multislot Option is installed.

Audio — When a tick is shown in front of this list entry, the Audio Option is installed.

MS Power Supply — When a tick is shown in front of this list entry, the MS Power Supply Option is installed.

MS Current Measurement — When a tick is shown in front of this list entry, the MS Current Measurement Option is installed.

ACPM — When a tick is shown in front of this list entry, the ACPM Option for special spectrum measurements is installed. This is always the case with the 4400M where ACPM is a standard feature. Note that ACPM is not an option in the CDMA2000 1xRTT Option.

CDMA 2000 — When a tick is shown in front of this list entry, the CDMA2000 1xRTT Option is installed.

AM Modulation — When a tick is shown in front of this list entry, the AM Signal Generator Option is installed.

EDGE — When a tick is shown in front of this list entry, the EDGE System Option software is installed.

OCXO — When a tick is shown in front of this list entry, the OCXO Option is installed to provide a high-precision time base.

FEX — When a tick is shown in front of this list entry, the Frequency range Extension Option is installed.

Turbo — When a tick is shown in front of this list entry, the Turbo Option is installed.

TCP/IP — When a tick is shown in front of this list entry, the TCP/IP Option is installed.

WCDMA Non-Call — When a tick is shown in front of this list entry, the WCDMA Non-Call Mode Option is installed.

WCDMA Call Mode — When a tick is shown in front of this list entry, the WCDMA Call Mode Option is installed.

TD-SCDMA Non-Call — When a tick is shown in front of this list entry, the TD-SCDMA Non-Call Mode Option is installed.

Parallel Multiphone Test — When a tick is shown in front of this list entry, the Parallel Multiphone Test Option is installed.

Installing additional options

Some of the options of the 4400 require both software and hardware (like e.g. the high-precision time base option or OCXO). These options can only be installed by the Willtek product specialists.

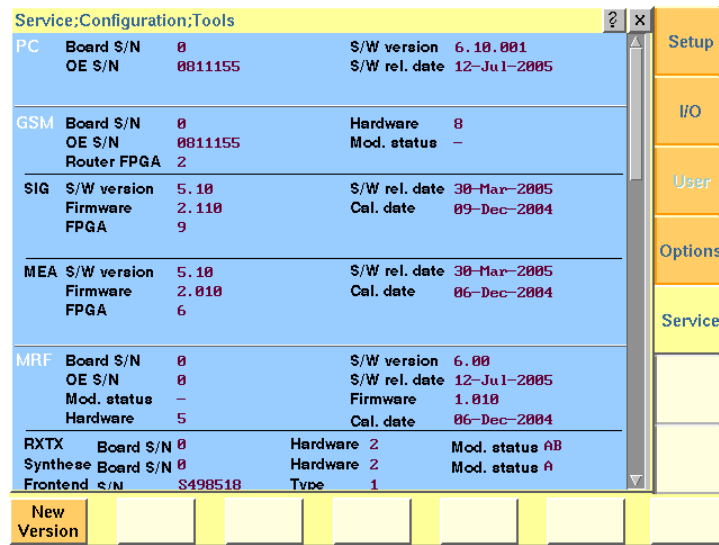
Options that are software-based only (like ACPM for instance) may be installed on-site. The only thing required then is a special code available from Willtek. When you have the code at hands, follow the instructions below:

Option Key

- 1 Push this softkey.
- 2 The **Input Option Key** prompt will appear on the screen.
- 3 Enter the option key received from Willtek.
- 4 Confirm the option key entered by pushing the **ENTER** function key or the corresponding key on the external keyboard.
- 5 If the code is recognized by the 4400, it will display an **ACCEPT** message. After leaving the entry box with the **ESCape** key, the tick in front of the corresponding option will appear and the full functionality of the option will be at your disposal.
- 6 In case the option key entered was not recognized by the 4400, you will see the **NO VALID KEY!** message. Try to enter the correct option key again.

Service and software update

Push the [Service] marker tab to gain access to this menu.



This menu provides all relevant information on the hardware and software revisions of the various subsystems of your 4400. The information on display here is very important when tracing problems or discussing upgrade possibilities with the Willtek product specialists.

Changing the version of the system software

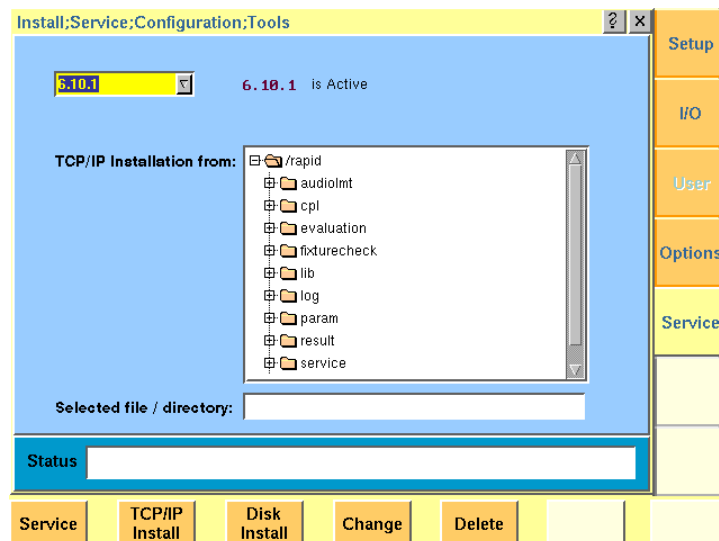
Willtek seeks to permanently improve its products and makes new software versions available on the Internet at www.willtek.com. Several different methods exist to upgrade the software of your 4400: From floppy disks, from a remote PC over the LAN (TCP/IP required) and from a remote PC over GPIB.



HAZARD

Be very careful using the functions provided in this menu. If used the wrong way, functions described below may leave your test set in a condition where it is not operable anymore. Never 'play' with this menu!

A push on the **New Version** softkey provides access to the following menu:



The display field in the upper middle of the menu (**6.10.1 is Active** in this example) indicates the software release that is currently in use.

The selection field to the left of the display field allows you to select a different software release that was previously installed on this test set. This will usually be an older software version.

NOTE

The 4400 allows a maximum of five software releases to be stored internally. In case you try to update a test set where five software releases have been stored already, you will be prompted to delete a software release first.

CAUTION

After installing version 5.10 version 5.00 must be deleted as switching back to the old version can in this case cause problems.

The file selection box **TCP/IP Installation from:** allows you to select a file on the network from which to install new software. The file with the software upgrade usually has the file name extension **.tar**. The box displays the directories and files available on the mounted server disk. The server disk can be mounted in the **Tools > Config > I/O** menu.

NOTE

The network computer with the server disk must be an NFS server.

The **Status** area of the menu is where the 4400 will display additional information, comments and messages.

The softkeys of this menu

Service

Closes the current menu and displays the service menu (toggle).
The softkey will change to **New Version**.

TCP/IP Install

A push on this softkey starts the **update routine via LAN (TCP/IP)**. The 4400 will load and install the file selected in the file selector box. Please follow the instructions shown on the screen of the 4400.

Note: NEVER close the window or switch off the 4400 while the 4400 is updating the system (i.e. after copying software from the network onto the internal hard disk and before the EPLD update is completed). Otherwise the instrument will not be usable anymore and needs to go back to the factory or to a Willtek service center.

**Disk
Install**

A push on this softkey starts the **update routine from floppy disk**. Please follow the instructions shown on the screen of the 4400.

Notes:

- To return from the update routine without updating, close the window on-screen with the external mouse and confirm the prompt with **Yes**.
- NEVER close the window or switch off the 4400 while the 4400 is updating the system (i.e. after copying software from floppy disk onto the internal hard disk and before the EPLD update is completed). Otherwise the instrument will not be usable anymore and needs to go back to the factory or to a Willtek service center.

Change

This softkey can be used to change from the current software version to a previously installed version. First, select the software release to change to on the selection field of this menu.

Changing to another softkey takes less than one minute on most systems, but the software must be rebooted. Before pushing this softkey, consider the consequences: Software releases lower than 2.86, for instance, will allow no return to the current release state.

Note: After installing version 5.10 version 5.00 must be deleted as switching back to the old version can in this case cause problems.

Delete

If five software versions are already installed on the 4400's hard disk, it will prompt you to delete an older version first when you try to install yet another version.

First, select the software release to be deleted on the selection field of this menu. Before pushing this softkey, consider the consequences.

After a software release has been deleted, it must be installed anew if you want to work with it again.

A currently used software version cannot be deleted. If you want to delete the current version, change to another version first.

**Initiating a software update
from a remote PC**

Willtek also offers a tool to update the 4400 software from a PC over the GPIB. This requires a GPIB interface in the PC. The file to download and a Windows-based program to perform the software download are available in the download section on www.willtek.com.

NOTE

The download program requires a National Instruments GPIB card.

Utilities

The menus of this section provide useful tools for debugging when the 4400 is used in a system environment.

This section consists of the following parts:

- **"I/O trace for GPIB communications"** – allows to monitor communication on the GPIB in great detail.
- **"Info trace"** – stores internal messages of the system and is helpful when discussing problems with the Willtek product specialists.
- **"MS power supply"** – the MS Power Supply option provides the mobile phone with supply voltage; the Current Measurement option helps identify problems that are caused by shortcuts on the printed circuit board of the phone.

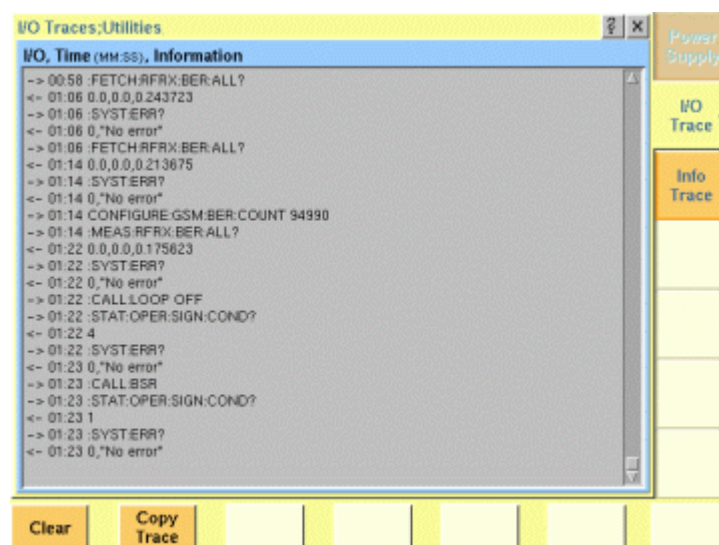
Access to the Utilities menus

Push the **TOOLS** function key from the Welcome menu. The tool bar provides access to the utilities menus with a push on the [Utilities] marker tab.

I/O trace for GPIB communications

This menu allows for monitoring the communication on the GPIB, e.g. to debug a new PC program for remote control of the 4400.

Push the [I/O Trace] marker tab to gain access to this menu.



This menu consists of an area to show all the commands and responses going back and forth between a controlling device (such as a PC) and the 4400. This communications display is called "trace" and is written to both the screen and a trace file.

NOTE

The trace is generated only as long as the screen is active, otherwise no trace is generated (or the existing trace will not be updated with new information).

NOTE

Generating the trace consumes more processing power from the 4400 i.e. the 4400 may work slower than without tracing.

NOTE

If you don't need the trace for debugging purposes, it is strongly recommended to go back to the Welcome menu before starting remote control operation.

The softkeys of this menu

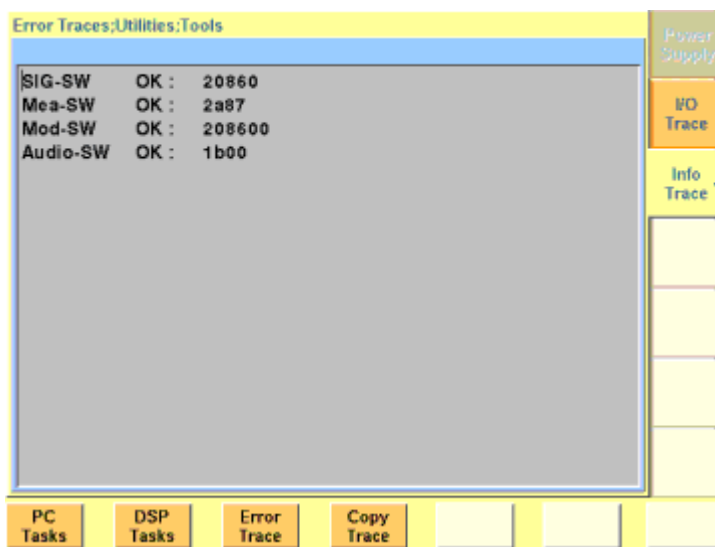


Resets the entire I/O Trace menu (all entries in all areas of the menu will be cleared).



Before pushing this softkey, insert a disk in the 4400's disk drive as a push on this softkey copies the entire I/O Trace to the disk. A text file named `GPIBPROT.TXT` will be generated, containing the I/O Trace in ASCII format. If the I/O Trace is empty, the file will be empty as well.

Info trace Push the [Info Trace] marker tab to gain access to this menu.



This menu provides detailed information about internal procedures and messages of the 4400.

It does not contain any user-serviceable settings or entry fields.

Its main application is internal debugging at Willtek. It may, however, also provide useful information for the Willtek product specialists when tracing a problem.

The softkeys of this menu

**PC
Tasks**

Shows the trace of the procedures, running on the internal PC.

**DSP
Tasks**

Shows the trace of the procedures, running on the internal DSP.

**Error
Trace**

Shows internal messages and error trace.

**Copy
Trace**

This softkey can be used to copy the Error Trace onto a floppy disk.

Before pushing this softkey, insert a disk in the 4400's disk drive. A push on this softkey copies the internal message and error trace onto the floppy disk. A text file named `INFO-DAT.TXT` will be created, containing the internal message and error trace in ASCII format. If the trace is empty, the file will be empty as well.

The Error Trace file may be requested by Willtek Product Specialists when reporting a problem.

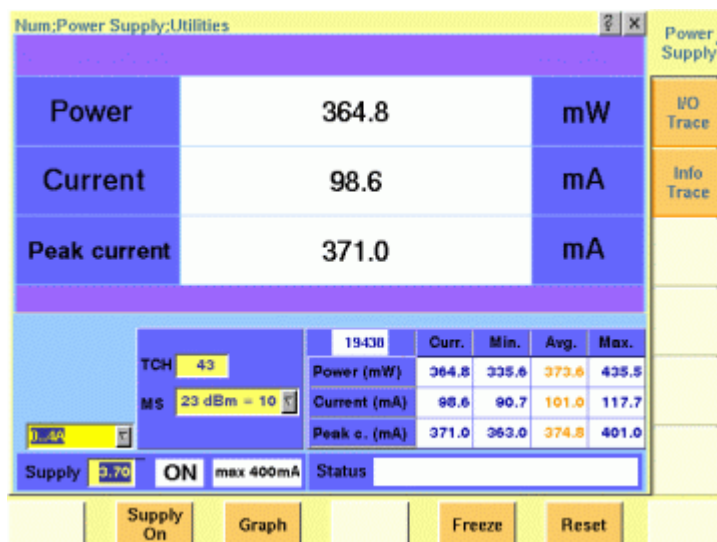
MS power supply

Push the [Power Supply] marker tab of the Utilities section to gain access to this menu. It offers you the controls and measurements for two options: the MS Power Supply Option and the Current Measurement Option.

The MS Power Supply is an option to the 4400, therefore this screen is accessible only if the MS Power Supply Option is installed.

The MS Power Supply Option is a prerequisite for the Current Measurement Option. The Current Measurement Option shows you the current that the mobile phone draws from the battery simulated by the 4400.

The measurement results from the Current Measurement Option are visible only if this option is installed.



MS Power Supply Option

This menu allows you to select and switch on/off the required supply voltage for the mobile phone, for use with the MS Power Supply option of the 4400. The MS Power Supply options is meant to replace a battery e.g. in environments where the original battery is not yet connected to the mobile (such as in manufacturing lines) or where faults must be traced back to either the phone or the battery (such as in service). Moreover, the MS Power Supply can be used to test the effect of over- or under-voltage on RF transmitter, RF receiver and audio parts.

Connection: The MS Power Supply connector can be found at the left-hand side of the 4400 front panel, below the floppy disk drive. The MS Power Supply option comes with a cable of one meter length. One end of the cable can be plugged into the 4400, while the other end is open-ended. The open ends can be used to connect the power supply to a fixture holding the mobile phone. Willtek delivers the cable with open ends because these fixtures are manufacturer-specific and hence open ends are most universal.



The Supply entry field can be used to select a supply voltage in the range 0 to approximately 11 V DC, in steps of 0.05 V (i.e. 50 mV). The option is specified from 0 to 10 V; higher voltages may or may not be achievable. A status field next to the supply voltage entry field indicates the current status of the power supply, i.e. either ON or OFF. A softkey is available to turn the 'supply on' or 'supply off', respectively.

NOTE

For safety reasons, the supply voltage is always off at power-on. So the power supply has to be switched on when required.

NOTE

If the power supply is short-circuited, the supply voltage and current goes down. When the short circuit is removed again, the voltage will come back after a few seconds (PTC resistor).

Current Measurement Option

If installed, this option displays the current drawn from the simulated battery.

The menu can show the measurement results in two different modes: a numerical-only display and a combined graphical and numerical display. In both cases, the lower results section is the same, indicating a statistical representation of the results for measured power, average and peak current.

The upper results section varies depending on the selected display mode; the mode can be selected with a softkey.

- In numerical-only mode, the upper results section shows the numerical results for power and current from the latest measurement.
- In graphical mode, the upper results section displays the shape of the current over time for one TDMA frame.

Selecting the display mode and display properties

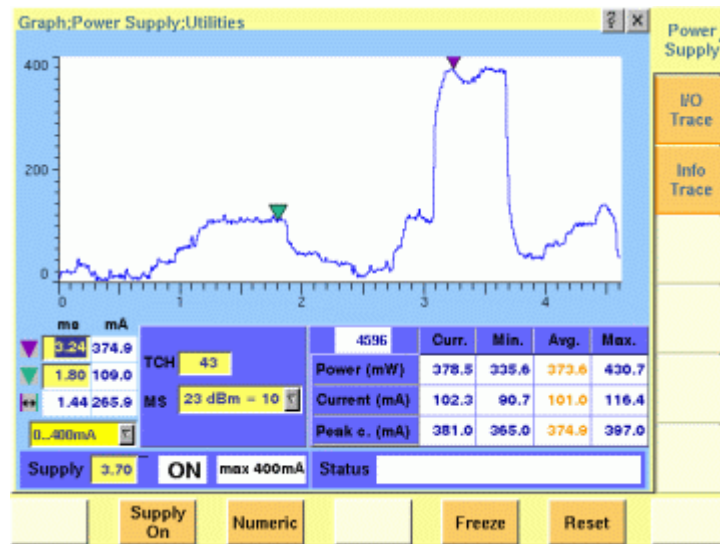
The numerical display is the default mode. To switch to the combined graphical/numerical mode, push the `Graph` softkey. As a result, the current versus time display appears and the softkey description changes to "Numeric". Push this softkey again to return to numerical-only value.

In combined graphical/numerical mode, the menu shows a graph with the current versus time. The scale of the vertical axis is mA and the horizontal axis scaling is in ms (milliseconds). Two cursors and a cursor readout area allow you to read the measurement values. Two cursors and a cursor readout area allow you to read the measurement values.

The graphical display resolution can be changed as well: Use the scroll field in the numerical results area to change the vertical resolution.

NOTE

The Duration field is not available in this software version.



The measurement and display range covers a complete TDMA frame from the start of a downlink frame. With a traffic channel active on time slot 3 and an offset of three time slots between downlink and uplink, the highest current level can be expected at $(3+3) * 0.54738 \text{ ms} = 3.28 \text{ ms}$. Other current peaks may result from an active receive/demodulation process.

In the measurement example above, a period of low current can be identified in the beginning. After 1 ms, the mobile phone sets up its receiver for reception of the downlink slot. From 1.8 ms to 3 ms, the demodulated data are decoded, and the transmitter prepares for the uplink slot. Transmission of the uplink slot follows, requiring a lot of current from the mobile phone; the absolute value depends on the selected uplink power level. The final phase of the frame is characterized by preparations for the next uplink slot transmission, for example interleaving in the channel coder.

Understanding result fields

The difference between current, minimum, maximum and average values is explained in section ["Statistical data evaluation" on page 1](#).

Power — This field shows the average power drawn from the simulated battery, measured in mW. It is calculated from the selected supply voltage and the measured current.

Current — The Current Measurement Option measures the current that the mobile draws from the MS Power Supply outlet. The result is averaged over one TDMA frame and displayed in mA.

Peak c. — This field indicates the maximum current that has been detected during the last TDMA frame. The peak current result is shown in mA.

Changing the RF parameters

While measuring the current consumption, you can also switch the RF channel or the transmitted RF power to evaluate the current consumption under different parameter settings.

Use the input fields TCH to change to a different channel.

Use the MS scroll field to change the transmit power level.

Spectrum measurements

The group of spectrum measurements available in the Spectrum menu offers functions for measuring the RF output spectrum emission. These functions are used to measure the parameters for assessing the quantity of power that leaks outside the assigned radio channel, i.e. the off-carrier power. If this power value exceeds certain limits, interference with neighbouring channels may increase. Furthermore, system capacity may be affected. Spectrum measurements for TD-SCDMA include Occupied Bandwidth (OBW), Adjacent Channel Leakage Power Ratio (ACLR) and Spectrum Emission Mask (SEM).

In order to get access to the Spectrum menu push the **TOOLS** function key. You can also select the [Spectrum ...] marker tab while working in Generator/Analyzer mode. For a detailed description of the spectrum measurements available refer to ["RF Analyzer for spectrum" on page 47](#)

RAPID!

4

This chapter describes how to use and how to program RAPID! applications. Main topics described in this chapter are as follows:

- "Overview" on page 88
- "Using RAPID!" on page 88
- "RAPID! syntax" on page 105
- "Commands" on page 113
- "Functions" on page 138
- "Tables" on page 156

Overview

RAPID! is a combination of the simple-to-use programming language BASIC and the powerful SCPI command language, developed for the 4400. RAPID! turns your 4400 into a fully automated radio test set that takes measurements, prompts the operators for inputs and generates a test result report that may be printed or sent to the manufacturing line control computer by LAN.

This chapter gives you all the information you need for successful editing of programs, for running and debugging them. It is divided into four subsections:

- [Using RAPID!](#) – Gives an introduction and explains everything about loading and saving files as well as editing, running and debugging programs.
- [RAPID! syntax](#) – This chapter is the 'Syntax manual' of RAPID!. It outlines the general syntax, the program line syntax, the use of variables, constants, operators and expressions.
- [Commands](#) – This part of the RAPID! reference contains a precise explanation of available commands together with application examples.
- [Functions](#) – A range of functions is available to work with numerical and string variables, as well as for input and output handling.
- [Tables](#) – Should any error occur during editing a program or during runtime, this subsection explains the reason for that in detail and gives information on how to trace back an error and to debug it.

NOTE

For all work within the RAPID! environment, we strongly recommend the use of an external keyboard (PS/2 connector) and a standard PC mouse (Sub-D 9-pin connector). For more details, please check with section "[Connectors on the rear panel](#)" on page 10.

Using RAPID!

This subsection gives an introduction and explains everything about loading and saving files as well as editing, running and debugging programs.

- [Introduction](#) – Here you will find a general explanation of what RAPID! is, and how to [enter the RAPID! working environment](#).
- [File menu](#) – In this menu, you load, save, copy, rename or erase program files.
- [Edit menu](#) – The RAPID! editor provides powerful tools for entering code or for searching and replacing contents.
- [Run menu](#) – Here you select the file to run and start a program.
- [Debug menu](#) – This powerful menu provides you with all the features you need for successful program development: running a program in single steps, watching and altering variables and more.

Introduction

RAPID! is a programming language that turns the 4400 into a fully automated radio test set. Controlled by RAPID!, the 4400 will for instance execute a series of tests with minimal human interaction, and at the same time produce a test report with a clear evaluation of the results measured.

Of course, you may halt any program and show e.g. adjustment instructions on the display of the 4400. Once the operator responds to those instructions, the RAPID! program will continue with new measurements and/or instructions.

RAPID! is a great time saver for extensive tests that are performed frequently. Typical examples are automated acceptance tests after manufacturing or regular routine checks as part of maintenance or quality assurance.

RAPID! = BASIC + SCPI

RAPID! programs are a powerful combination of the simple-to-use BASIC programming language and the 4400 specific SCPI commands. The BASIC instructions implemented are outlined in detail in subsection [RAPID! syntax](#) while [Chapter 5](#) is fully dedicated to SCPI.

BASIC commands

BASIC (Beginner's All-purpose Symbolic Instruction Code) commands provide the required program sequence. They also enable the further processing of measured results, the entry of numeric values and texts (character strings) as well as the formatted output of reports to e.g. a printer.

SCPI commands

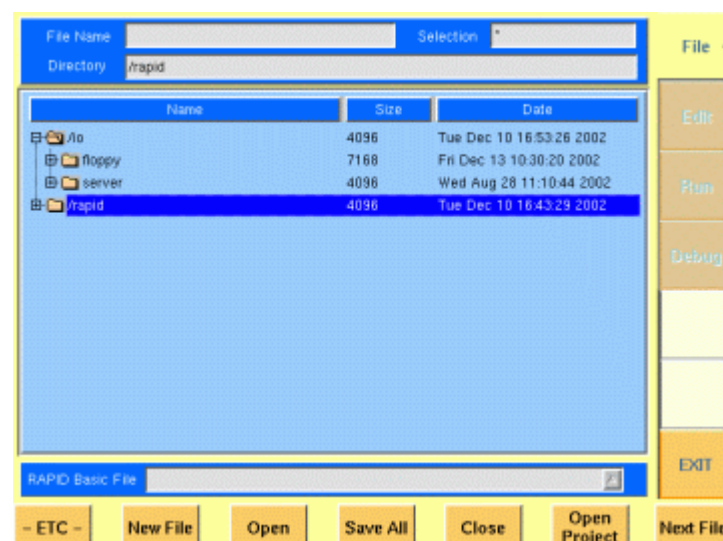
SCPI commands are used for configuring the 4400 and for executing and evaluating measurements. You'll find more about the use of SCPI commands in subsection "[SCPI and RAPID!](#)" on page 173.

Entering and exiting the RAPID! environment

How to enter

To enter the RAPID! environment, navigate to the [Welcome menu](#). There, simply push **TOOLS** > [RAPID!].

If an AUTOSTART.bas programm has been loaded and is active, the 4400 will now start that program. Otherwise the 4400 will now display the RAPID! File menu.



With the help of the File menu, you may select the file you want to edit or create a new file with the help of the related softkeys and marker tabs.

How to exit

The [Exit] marker tab quits the RAPID! environment and takes you back to the Welcome menu.

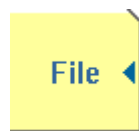
In case you did any program editing without saving the changes, the 4400 will prompt you to either save or ignore the previous editing:



You may then save the changes by pushing the **Yes** softkey, while a push on softkey **No** will discard any changes and leave the file as it was. **Cancel** takes you back to the RAPID! environment.

Marker tabs of the RAPID! environment

The marker tabs offer direct access to the menus of the RAPID! environment:



File menu – Loading, saving, copying, renaming and erasing files in general.



Edit menu – Selecting a file to edit, entering code, saving a single program file, searching for and replacing text.



Run menu – Selecting a file to run and starting a program.



Debug menu – Running a program, single-stepping a program, watching and altering variables.

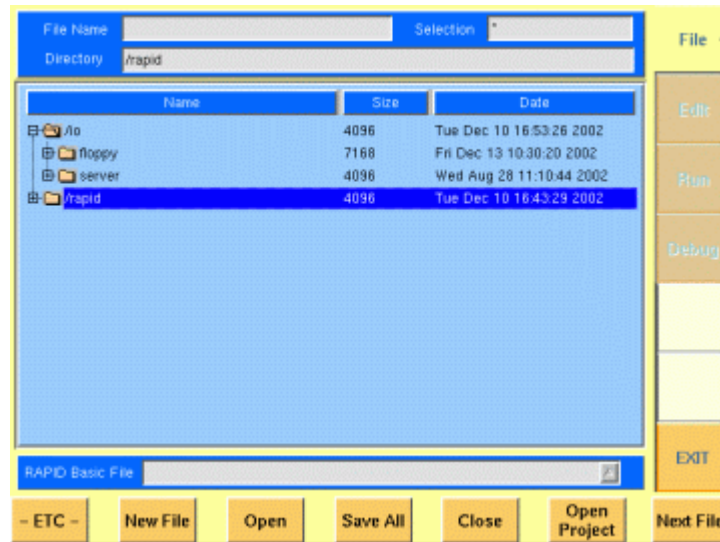


Quitting the RAPID! environment.

File menu In a similar way as e.g. Windows Explorer, the File menu gives access to the directory structure of the internal hard disk drive and allows to search for and select files.

NOTE

Some 'typical Windows' functions, e.g. double-clicking on a file name to open it, are not included.



To keep work with the File menu as simple as possible, it is subdivided into three main sections:

- **File name – directory – selection area**, where you may select directories, file names and filters for the display of file names in the browser area.
- **Browser area** – this is where the 4400 shows its directory structure and allows you to browse through all directories and files.
- **RAPID! basic file area** – this is a selection field where you may select one of the previously opened RAPID! Basic Files.

The softkeys are used to call specific file functions (as e.g. creating a new file, opening a file or deleting a file) while the marker tabs give access to the various menus of the RAPID! environment.

NOTE

As long as no file has been opened or newly created, all marker tabs (except [File]) are grayed out and cannot be selected.

File name – directory – selection area

In this area, you may manually select a directory, a file name and/or the filter for the files displayed in the browser area.

To gain access to these three entry fields, simply click on the related entry field with the mouse or push the |New File| softkey in the file menu.



To navigate from one entry field to the subsequent one, simply use the **PGUP** and **PGDN** keys.

File Name — This is where you enter the name of the file (including extension). In case you enter an invalid file name, the 4400 will display a warning message on-screen.

To confirm the file name and its directory, simply push the **ENTER** key. When creating a new file, it will be added to the RAPID Basic File list automatically.

NOTE

The easiest way to enter a file name is of course with the help of the external keyboard. If it is currently not available, all characters are accessible through the [Typing tabs](#).

Directory — Here, the 4400 displays the currently selected directory. In order to save a newly created file in a different directory or to change the currently selected directory, just enter the name of the directory you want to work within.

NOTE

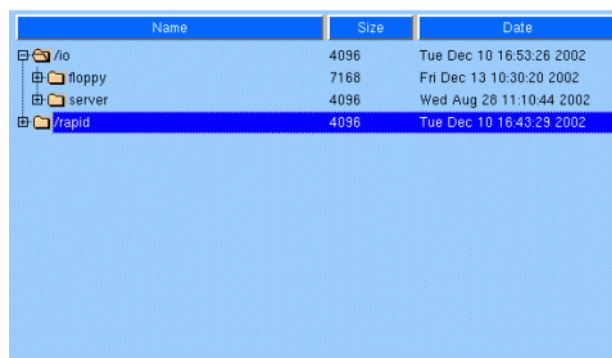
The directory path displayed here is the same as selected in the browser area, as long as you do not change it.

Selection — This entry field sets a filter for the file display within the browser area. The default setting is * and will make the browser display all files (with any extension).

Browser area

In this area, you may browse through directories just as with any other standard browser.

As soon as you enter the RAPID! environment, the file menu is automatically displayed and the browser area is active.



| Name | Size | Date |
|--------|------|--------------------------|
| /io | 4096 | Tue Dec 10 16:53:26 2002 |
| floppy | 7168 | Fri Dec 13 10:30:20 2002 |
| server | 4096 | Wed Aug 28 11:10:44 2002 |
| /rapid | 4096 | Tue Dec 10 16:43:29 2002 |

The most convenient way to navigate through the Browser area is with the help of the mouse. Alternatively, use the **PGUP** and **PGDN** keys to scroll by pages and the **UP** and **DOWN** keys to scroll by lines.

A directory is opened or closed by a simple push on the **ENTER** key after it has been selected.

To open a file, mark it with the cursor (it is then highlighted) and push the **Open** softkey. As soon as you open a RAPID! file, it is automatically added to the RAPID Basic File list.

The top level directories are:

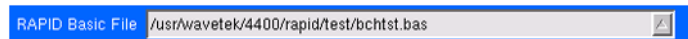
- **/io**. This directory includes the **floppy** subdirectory which in turn carries the contents of the floppy disk (if any is inserted in the floppy drive). An additional subdirectory, **server** may be available if the 4400 is equipped with an Ethernet PC card and the TCP/IP Option and if a server disk has been mounted. This disk is shown under **/io/server**.
- **/rapid**. This directory carries various subdirectories with example RAPID! programs, result files, coupling loss files and many more. You can use this area to locally save your files on the 4400.

NOTE

Access to a remote PC on the network requires NFS server software to be running on the PC; the server must grant at least read access for the directories and files. Please read section I/O Configuration for more details.

RAPID! basic file area

This area is made up by a selection field. You may select any file that has been opened or newly created before. The RAPID! Basic File selection field is available in all RAPID! menus. A simple push on the **Next File** softkey or a click on the selection field navigates the cursor to it.



To select a file, simply click on the file name. Alternatively, push the **UP** and/or **DOWN** keys until the requested file is highlighted. Then confirm your selection by pushing the **ENTER** key.

Softkeys of the file menu

The file menu contains two sets of softkeys. The **ETC** softkey toggles between them.



New File — Creates a new RAPID! program file. You specify the name (and directory) in the **File name – directory – selection area**, which is activated automatically after you push this softkey.

As soon as the file has been created, it is added to the RAPID! Basic File list by the 4400 and you may start editing any time.

NOTE

As this softkey gives access to those three entry fields, you may also use this softkey if you want to e.g. set a new filter for the files to be displayed within the browser area (see section [File name – directory – selection area](#) for details).

Open – A push on this softkey opens the file currently highlighted in the [Browser area](#). It also adds it to the RAPID! Basic File list. After a file has been opened, you may edit, run or debug the file.

Save All – Saves all currently open files. If there have been no changes made to a file since it was saved last, the file will remain unchanged.

Close – Closes the file currently selected on the RAPID! Basic File selection field list and removes it from the list. If there have been any changes made to that file since it was saved last, you will be prompted to save the file.

Open Project – This softkey works similar to the **Open** softkey. Unlike the **Open** softkey, this softkey does not only open the file selected (highlighted) but also all basic files chained to this one. All files will be added to the RAPID! Basic File list.

Next file – Pushing this softkey will activate the RAPID! basic file selection field and allow you to select one of the currently open RAPID! basic files.

New Dir – Creates a new subdirectory in the directory currently shown in the File Name – Directory – Selection display field. The 4400 will prompt you with an entry field to enter the name of the subdirectory to be created. To enter the name, use the external keyboard or the typing tabs.

Copy – A push on this softkey allows you to copy the currently selected (= highlighted) file. Additional softkey labels will be displayed that allow the following choices:

OK – Copies the currently selected file. The 4400 will prompt you with a dialogue box that allows you to enter the destination file name (including directory). To enter the file and directory name, use the external keyboard or the typing tabs.

to IO – Copies the currently selected file to the floppy inserted in the 4400's floppy drive. The copy of the file will have the same name as the original file.

to RAPID – Copies the currently selected file to the 4400's internal hard disk drive. The test set will prompt you with a dialogue box that allows you to enter the destination file name (including directory). To enter the file and directory name, use the external keyboard or the typing tabs.

Cancel – Quits the copy function and takes you back to the second set of softkeys.

Move — With the help of this function, you may rename the file currently selected in the browser area.

The 4400 will prompt you with a dialogue box that allows you to enter the destination file name (including directory). To enter the file and directory name, use the external keyboard or the typing tabs.

Erase — Deletes the file currently selected on the RAPID Basic File selection field. Before the file is deleted, you will be prompted to confirm or discard the erasure of the file.

Print — After pushing this softkey, the 4400 will prompt you with a print output message:

Printing file

Is a Printer connected to LPT, switched on and online?
If you confirm by pushing **YES**, the 4400 will print the file highlighted (without any formatting).

Softkey **NO** aborts the printing process.

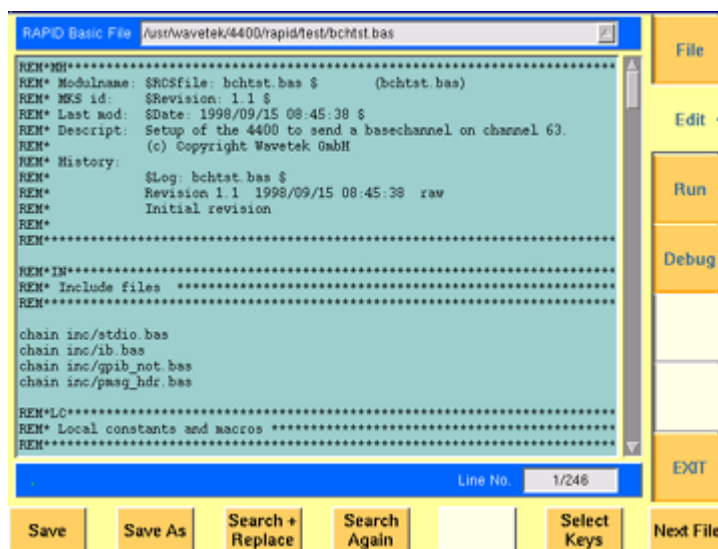
NOTE

The settings for the printer are performed in the I/O menu of the configuration section. To gain access, return to the [Welcome menu](#), push the **TOOLS** key and then the [I/O] marker tab. Settings for the parallel printer port are performed in the section marked with Parallel Port.

Edit menu

For program developers, this is probably the most important menu of the RAPID! environment.

The Edit menu is the menu you actually write programs in, before they can be run or debugged.



The Edit menu is subdivided into three main sections:

- [RAPID! basic file area](#). Here you select the file you want to edit.
- [Edit area](#), where the editing is carried out.

- **Status area.** This is where the 4400 displays error messages or gives the current cursor position in relation to the entire file.

The softkeys provide access to important functions of the Edit menu, like the search for text.

The marker tabs give instant access to the various menus of the RAPID! environment.

Edit area The Edit area is where you create and edit program code, using the built-in BASIC-type language of RAPID!.

```

REM*MH*****
REM* Modulname: $RCSfile: bchtst.bas $      (bchtst.bas)
REM* MKS id:   $Revision: 1.1 $
REM* Last mod: $Date: 1998/09/15 08:45:38 $
REM* Descript: Setup of the 4400 to send a basechannel on channel 63.
REM*          (c) Copyright Wavetek GmbH
REM* History:
REM*          $Log: bchtst.bas $
REM*              Revision 1.1  1998/09/15 08:45:38  raw
REM*              Initial revision
REM*
REM*****
REM*IN*****
REM* Include files *****
REM*****
chain inc/stdio.bas
chain inc/ib.bas
chain inc/gpib_not.bas
chain inc/pmsg_hdr.bas

REM*LC*****
REM* Local constants and macros *****
REM*****

```

To work on program code, we strongly recommend the use of an external keyboard.

In principle, it is possible to do without, using the typing tabs, but this is a rather time-consuming way.

An external keyboard not only provides mapping of the 4400 keys. It also allows you to use the standard 'control codes' as with any usual text editor. Some of the standard control codes are listed in the table below.

| | |
|---------------|------------|
| Ctrl+a | Select all |
| Ctrl+c | Copy |
| Ctrl+x | Cut |
| Ctrl+v | Paste |

Status area



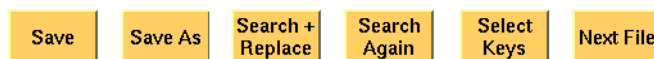
In the status area, the 4400 shows the following information:

- **Syntax errors** – these are mainly typing errors with commands or missing command elements and prevent a RAPID! program from being executable.
- **Runtime errors.** These errors occur during runtime of a RAPID! program and usually lead to an unexpected end of the program run.

- The display field `Line No.` gives both the number of the line, the cursor is currently located in and the total number of lines of the file.

Softkeys of the Edit menu

The Edit menu provides a number of file-handling functions that can easily be accessed through the softkeys.



Save — A push on this softkey saves the currently open file without changing its name.

Save As — Saves the currently open file under a new name. After a push on this softkey, the 4400 will display a dialogue box that allows you to enter the destination file name (including directory). To enter the file and directory name, use the external keyboard or the typing tabs.

Also, the labels of the softkeys change and the 4400 will offer three new softkeys:

- **Save** When you push this softkey, the 4400 will save the current file under the file name entered in the dialogue box.
- **Protect** When you push this softkey, the 4400 will save the current file under the file name entered in the dialogue box in protected mode. This means that the file will be completely inaccessible to any editing attempts.



WARNING

There is no way to edit a protected file again. Before selecting this function, please be sure that there is an unprotected copy of this file available.

- **Cancel** Quits the function and takes you back to the Edit menu with its basic softkeys.

Search+Replace — Enables you to find text passages and optionally replace them with other ones.

The text the 4400 shall search for is entered in the [Search/Replace menu](#).

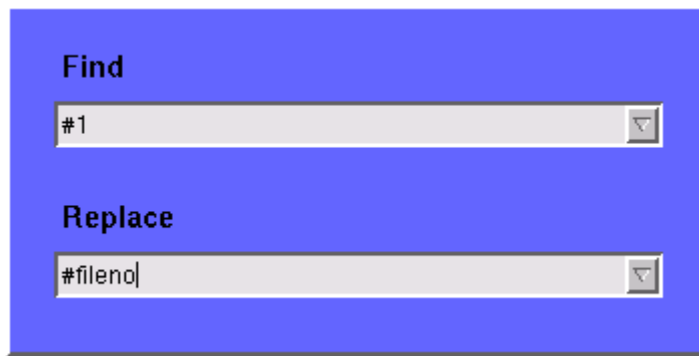
Search Again — Continues the search of the text passage as entered in the [Search/Replace menu](#) before. The direction of the search is "downwards", i.e. from the current cursor position towards the end of the file.

Select Keys — Changes the marker tabs to typing tabs. While the typing tabs are active, the label of this softkey changes to **Deselect Keys**. A push on that softkey brings back the basic marker tabs of the RAPID! environment and the initial softkey label.

Next File — Activates the RAPID Basic File selection field and allows you to select a different file. After you confirm your choice, the newly selected file is then displayed in the Edit area.

Search/Replace menu

In this menu, you enter the text to find. Optionally, it can be replaced by new text, entered in the same menu.



The text to be found is entered in the upper line (Find).

If the text found shall be replaced by another text, enter the new text in the lower line of this menu (Replace).

To move the cursor from the upper to the lower selection field and vice versa, use the **PGUP** and **PGDN** keys.

To get back to any text entered previously on these selection fields, push the **Select Prev.** softkey and the usual drop-down menu of a selection field will appear.

Softkeys of the Search/Replace menu



Find — Searches for the text entered on the Find selection field. The direction of the search is "downwards", i.e. from the current cursor position towards the end of the file.

Replace — Searches for the text entered on the Find selection field and replaces it with the text entered on the Replace selection field. The direction of the search is "downwards", i.e. from the current cursor position towards the end of the file.

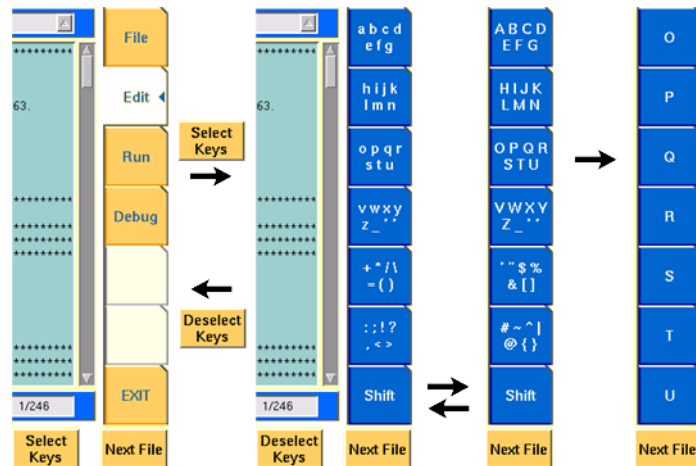
Replace All — Searches for all occurrences of the text entered on the Find selection field and replaces every single one with the text entered on the Replace selection field. The direction of the search is "downwards", i.e. from the current cursor position towards the end of the file.

Select Prev. — Calls the usual drop-down menu of the currently active selection field and allows you to select any text entered previously.

Cancel — Quits the Search/Replace menu and takes you back to editing.

Typing tabs

In case an external keyboard is not accessible, there is still the built-in 'type-writer' of the 4400 available. The marker tabs change to typing tabs and allow to enter and/or edit text in a way similar to an SMS on a mobile phone.



As soon as you push the **Select Keys** softkey, the marker tabs change to groups of letters and symbols.

The lowest marker tab changes to [Shift] and allows you to toggle between lower and uppercase letters.

As soon as you select a group of letters or symbols by pushing the corresponding marker tab, the marker tabs will change again to single letters or symbols. Push the marker tab of the desired letter or symbol. That one will appear on the entry field or edit menu and the marker tabs will change back to groups of letters and symbols automatically.

NOTE

Entry fields can also be edited using the **BACK** function key.

NOTE

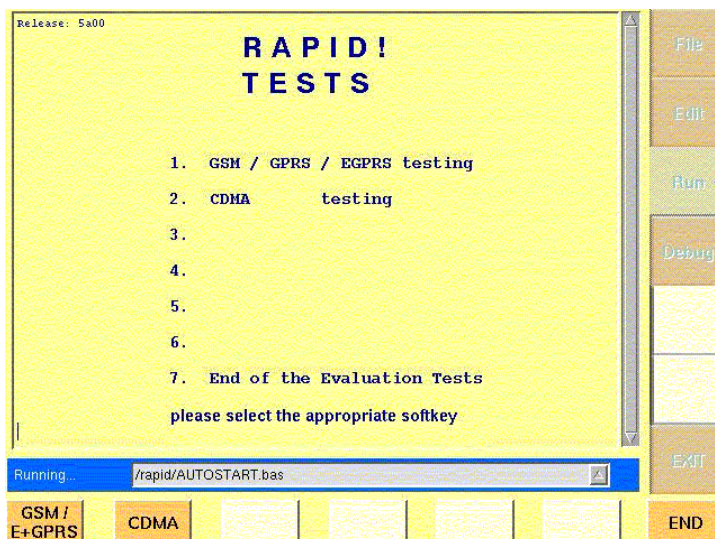
Softkey **Deselect Keys** takes you back to the usual marker tabs of the RAPID! environment.

NOTE

Use the numeric keypad to enter numerals, the minus sign (-) or a period (.).

Run menu

The Run menu is where programs are started. If there are no syntax or runtime errors encountered, the program will smoothly run until it reaches its end and will display its status messages and comments in this menu.



The Run menu is subdivided into two main sections:

- **RAPID! basic file area.** Here you select the file you want to run.
- **Output area** where the output of the running RAPID! program is displayed.

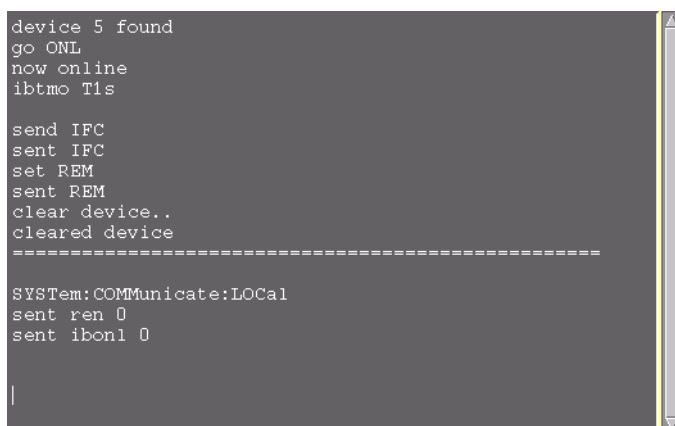
The softkeys provide access to important functions of the Run menu, like starting a RAPID! program.

The marker tabs give instant access to the various menus of the RAPID! environment.

NOTE

As long as no file has been opened or newly created, the marker tabs are grayed out and cannot be selected.

Output area In the output area, the 4400 displays the output and status messages created by a running RAPID! program.



Softkeys of the Run menu

The Run menu offers two softkeys:

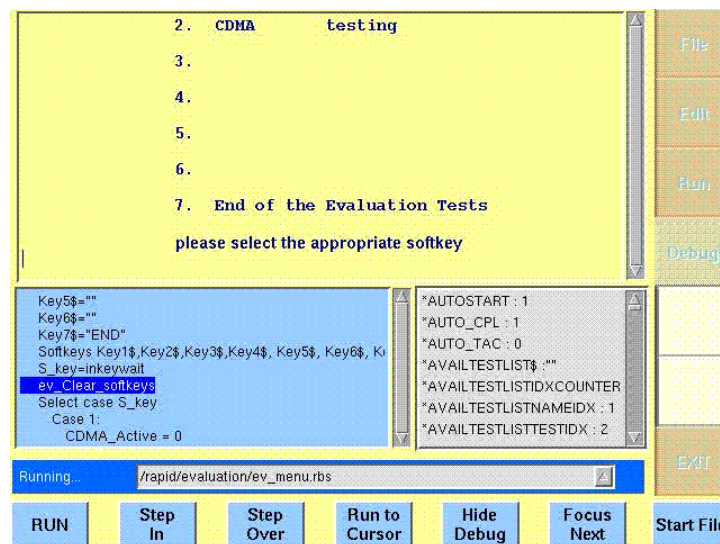


RUN — The RAPID! file, currently selected on the RAPID Basic File selection field, will be loaded and the program execution started.

Start File — Takes the cursor to the RAPID Basic File selection field and thus allows you to select the required file from the list. Next time you push the |RUN| softkey, this newly selected file will be executed.

Debug menu

The Debug menu provides you with a number of powerful tools in order to support your program development. The Debug menu consists of four areas that provide you with full information on different aspects of the program run. In order to work with this menu in an efficient way, we strongly recommend the use of an external keyboard and a mouse.



In this Debug menu only, the softkeys of the 4400 appear in blue. Thus, they can be easily distinguished from program-generated softkeys, as those appear in standard orange.

The areas are:

- **RAPID! basic file area.** Here you select the file you want to run.
- **Output area** where the output of the running RAPID! program is displayed.
- **Source area.** Here, the 4400 shows the program code being currently executed.
- The **Variable area** displays all program variables as well as their current value.

The softkeys provide access to important functions of the Debug menu, like starting a RAPID! program.

The softkeys of the Debug menu appear on blue background. Thus, they can be

easily distinguished from the program-generated softkeys.
The marker tabs give instant access to the various menus of the RAPID! environment.

NOTE

As long as no file has been opened or newly created, the marker tabs are grayed out and cannot be selected.

Source area In the Source area, the 4400 displays the program code being currently executed. The program line highlighted in blue is the line that either has just been executed (in RUN mode) or the line that will be executed next (in STEP mode).

```
param(1) = Modulationsart
param(2) = BT
param(3) = NormalHub

STOP
OUT_sendMsg SIG_DUPO, MOD_TARGET, MOD_PARAM

if DEBUG = TRUE then
  print " Anzahl parameter: "; AnzParam
```

To browse through the program, simply use the external keyboard or the cursor keys and function keys.

It is not possible to edit the program in this area. In order to do so, please use the Edit menu.

To run a program or to execute it step by step, please use the appropriate softkeys of the Debug menu.

In the Debug menu, only the first screen line of continued program lines will be highlighted.

Variable area In the variable area, the 4400 displays all variables used in the RAPID! program run, including their current values.

While a program is e.g. waiting for an input, you may access the Variable area with the mouse and have a look at all current variables.

```
BT : 0.3
CALL_NETW_STAT : 132
CAL_HF_REC_RANG : 97
CAL_HF_REC_RANG_CLE : 96
CMD$ : "Start : 11:08:42"
DEBUG : 0
DEBUG_MESSAGE : 21
```

Altering the value of variables directly

Within the variable area, the values of the variables may be altered directly:

- 1 Select the variable with the mouse. The 4400 indicates the selection of the variable by highlighting it in dark blue.



- 2 Enter the new value for the selected variable in the `Edit Watch:` dialogue box, using the 4400's numeric keys or the external keyboard.
- 3 Confirm the value entered with a push on the **ENTER** function key on the 4400 or a push on the external keyboard's **Enter** key.
- 4 The new value will now be used when you run or step the program using the corresponding softkeys of the Debug menu.

Alternatively

- 1 Open the Variable area by pushing the **Focus Next** softkey until the Variable area gets highlighted.
- 2 Select the variable you would like to alter with the cursor (simply use the cursor keys to do so). The currently selected variable is indicated by a thin black frame.
- 3 Activate the variable editing mode by confirming your selection with a push on the **Enter** function key on the 4400 or a push on the external keyboard's **Enter** key.
The 4400 indicates the selection of the variable by highlighting it in dark blue.

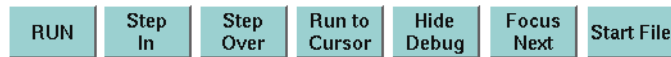


- 4 Enter the new value for the selected variable in the `Edit Watch:` dialogue box, using the 4400's numeric keys or the external keyboard.
- 5 Confirm the value entered by pushing the **Enter** function key on the 4400 or by pushing the external keyboard's **Enter** key.
- 6 The new value will now be used when you run or step the program using the corresponding softkeys of the Debug menu.

Note: The variable selected last will remain highlighted until you select a new variable or run a new program.

Softkeys of the Debug menu

The Debug menu provides a number of specific debugging functions that can easily be called with the help of the softkeys.



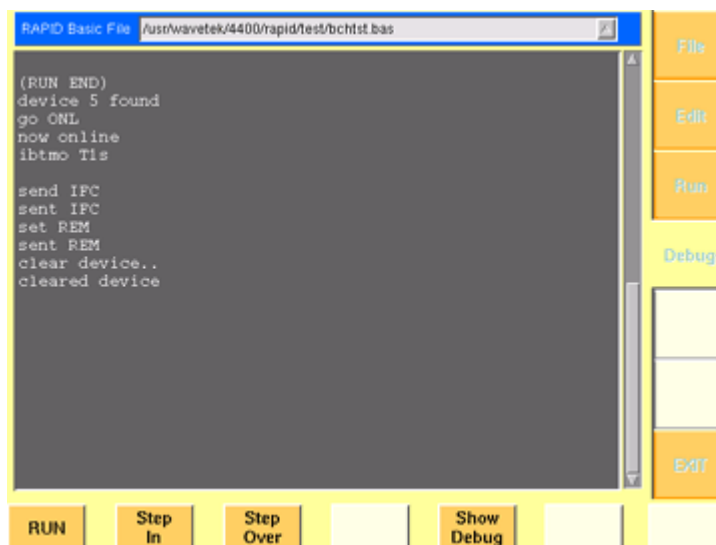
RUN — Loads the program currently selected on the RAPID Basic File selection field and starts execution of the program.
In this mode, the 4400 will highlight the last command performed in the Source area.

Step In — Executes the program line highlighted.
In case this line contains a GOSUB, CHAIN, SUB or FUNCTION call, the debugger steps in, i.e. it will follow the program branching.
In case the line contains an INKEY command, the program will be halted and wait for the push of a softkey. After the softkey was pushed, the program-defined softkey label will be displayed.
In this mode, the 4400 will highlight the next command to be performed in the Source area.

Step Over — Executes the program line highlighted.
In case this line contains a GOSUB, CHAIN, SUB or FUNCTION call, the debugger steps over it to the following line, i.e. the program branching will be ignored.
In case the line contains an INKEY command, the program will not be halted and will assume that no softkey was pushed.
In this mode, the 4400 will highlight the next command to be performed in the Source area.

Run to Cursor — Executes the program from the line currently highlighted to the line where the cursor is located.

Hide Debug — Hides the source and variable area to show the complete output area.
The label of this softkey will change to **Show Debug**. A push on **Show Debug** will display the hidden areas again.



Focus Next — Toggles the cursor location between the output, the source and the variable area .

Start File — Activates the RAPID Basic File selection field and allows you to select a different file. After you confirm your choice, the newly selected file is ready for debugging.

RAPID! syntax

This chapter gives an overview of the RAPID! syntax and is subdivided into the following sections:

- [General syntax](#) – Here you will find all about the general aspects of the syntax like program line formats, basic rules, syntax check or notation.
- [Program lines](#) – This section outlines all details on the program line format and its elements.
- [Variables](#) – Are you familiar with numeric variables, string and array variables or dimensions? Here you will find it all.
- [Constants](#) – This section gives all information on numeric and string constants.
- [Operators](#) – RAPID! provides you with a number of operators like sign, arithmetic operators, comparisons or boolean operators.
- [Expressions](#) – These are combinations of variables, constants, function results and/or operators.

General syntax

BASIC form The RAPID! programming syntax is compatible with current BASIC derivatives, such as e.g. Quick BASIC or Visual BASIC. However, RAPID! in its current form does not allow for object-oriented extensions. On the other hand, RAPID! supports a number of functions that are vital for the use of the programming language on the 4400.

Program and line formats Line labels (i.e. numbers) are optional. This means, they are not necessary.

A program line may include several BASIC commands. In that case, they have to be separated by colons (:).

The maximum length of a program line is 254 characters. However it is suggested to keep the program lines short. This increases serviceability and eases debugging of a program.

The length of a program is (in theory) only limited by the disk space available.

Basic rules The RAPID! program interpreter is not case-sensitive. This means that commands as well as constants and variables may be written in both uppercase and lowercase notation.

RAPID! commands and reserved words like `GET` or `GOTO` are not permitted as line labels or as names for constants and variables.

RAPID! commands, names of constants and variables as well as numeric values **must not** include blanks (spaces).

Syntax check The syntax check is performed by the built-in interpreter. In other words, after the start of a program, every single line is interpreted one by one and the corresponding action is initiated. If the built-in interpreter comes across a command in a program line that it can't understand, then this is regarded a syntax error.

NOTE

Most of the syntax errors are simple misspellings of commands or typing errors.

The interpreter always checks a program after it was started. A detected syntax error causes the program to be aborted and leads to an error message indicating the erroneous program line.

Notation To explain the language syntax, the following notation is used:

`[item]`

Square brackets indicate an optional item, which can also be omitted.

Example: `[sign] number` is a number with an optional + or – sign.

`item | item`

Vertical bars separate entries of a list and indicate that precisely one element must be used.

Example: The following list contains the four seasons `spring | summer | fall | winter`. One of these entries gives the current season.

`{item}`

Brackets enclose an item or a list of items, which can be repeated several times.

Example: `{ A...Z }` is the notation for text.

`item1 := item2`

The colon followed by the equality symbol gives the definition of an item.

Example: `binary digit := 0 | 1` defines the digits of a dual number system in a way that a 'binary digit' can either be '0' or '1'.

NOTE

Notations can also be nested. This makes them suitable for expressing even complicated syntax.

Example: `boolExp := NOT boolExp | boolExp AND | OR | XOR boolExp`

This means by words: A boolean expression is either the negation of another boolean expression, or two boolean expressions linked with one of the three operators: AND, OR, and XOR.

Please note that this is only an example. The complete definition of [Boolean expressions](#) is a bit more complex.

Program lines

Each RAPID! program consists of a number of program lines. Those lines contain the BASIC statements and have the following structure:

```
[label:] statement [: statement]
```

```
label:
```

A program line starts with an optional label. In contrary to AUTORUN programs, line numbers are not required.

With RAPID!, labels are mainly used as branch targets.

A label must always end with a colon.

```
statement
```

A statement is a BASIC command.

Important notes

- A program line always needs to be terminated by a carriage return (CR). This symbol is equivalent to pushing the 'Enter' key at the end of a program line.
- It is not possible to show the maximum of 254 characters in one line on-screen. Therefore, a program line may stretch out over several lines on-screen. To indicate that the program line is continued, put a blank followed by an underscore ' _ ' and then push **Enter** to start a new line.
Example: `print "RAPID!" _
;" Program"`
- There must not be empty lines between continued program lines.
- An underscore at the end of a 'REM' command will cause a syntax error.
- The debugger will only highlight the first screen line of continued program lines.
- As mentioned before, a program line may contain up to 254 characters and several statements. However, we strongly recommend to use only one statement per program line wherever possible. Several statements in a program line lower the readability of the program code and usually make servicing a program quite tricky.

Variables

Variables are used to save data for later use. There are two types of variables: numeric variables and string variables.

- **Numeric variables** contain numbers only (0...9) and are saved in the so-called double (floating point) format on the 4400. This is an eight-byte value with a range from $-1.79769313486232E308$ to $-4.94065645841247E-324$ for negative values and from $4.94065645841247E-324$ to $1.79769313486232E308$ for positive values.
- **String variables** may contain any combination of numbers (0...9) and letters (A...Z). Their name always must end with the '\$' symbol. The length of a string variable (i.e. the number of characters) may range from 0 to 65,535 characters.

Variables in the RAPID! environment are always local. This means that a variable is only valid within its own RAPID! program.

When you chain programs, a local variable is not available to the program chained. If a variable shall be global (i.e. it is available to all chained programs), simply specify it as global.

The main commands dealing with variables are DIM, GLOBAL, VARIABLE and ERASE.

Arrays

- All RAPID! variables can make up arrays of up to two dimensions. The index value per dimension may range from $-32,765$ to $+32,765$.

NOTE

An array is similar to a table or a spreadsheet.

Syntax

A variable name is not limited in length, but only the first 20 characters are significant. A variable name has the following syntax:

- `A...Z | _{0...9 | A...Z | _} [(index1 [, index2])]` for numeric variables and
- `A...Z | _{ 0...9 | A...Z | _}$ [(index1 [, index2])]` for string variables.

Notes

- The first character of a variable name has to be a letter.
- Uppercase and lowercase letters have the same meaning with RAPID!.
- A...Z are uppercase or lowercase letters.
- 0...9 are numbers between 0 and 9.
- `index1` and `index2` are integer values, making up an array. The range of an integer value is from $-32,765$ to $+32,765$.

Examples

- `1_stValue` is not a valid variable name as it starts with a number.
- `myvar_1`, `MyVar_1` or `MYVAR_1` will be understood by RAPID! as being the same variable.

- `StringArray_of_Dimension_2$(1,1)` defines a valid string variable array. However, due to the length of its name, RAPID! will not distinguish it from `StringArray_of_Dimension_1$(1,1)`.

Physical units

Numeric variables and expressions can have physical units. Depending on the group they belong to, the units of those variables and expressions may be:

| Group Name | Physical units |
|------------|-------------------------|
| Frequency | Hz, kHz, MHz, GHz |
| Power | dB, dBm, μ W, mW, W |
| Angles | DEG or $^{\circ}$, RAD |
| Percent | % |

Notes

- Percent values are treated as unit-less when arithmetical operations are performed.
- Additions, subtractions and comparisons may only be performed within a group as specified above.
- Divisions through a value containing a physical unit of the same group will lead to a result without physical unit.
- When multiplying values, only one factor may have a physical unit.

Constants

Constants are used to store numeric or string values, which need to be kept unchanged during runtime of a RAPID! program.

Typical examples are common mathematical constants like 'pi', the power loss factor of a cable used for testing or the type code of a mobile phone.

Numeric constants

A numeric constant is a decimal, binary, octal or hexadecimal value with an optional physical unit:

`decimal | binary | octal | hexadecimal [physical unit]`

| Type of constant | Syntax | Example | Limits |
|------------------|--------------------------------|-------------------------------|--|
| Decimal | <code>{0...9} [.{0..9}]</code> | 123.456 | -1.79769313486232E308 to -4.94065645841247E-324 for negative values and from 4.94065645841247E-324 to 1.79769313486232E308 for positive values |
| Binary | <code>&tB {0...1}</code> | <code>&tB100111011</code> | 2 to the power of 32 (2^{32}) |

| | | | |
|-------------|---------------------|----------|---|
| Octal | &tO {0...7} | &tO37721 | 2 to the power of 32 (2 ³²) |
| Hexadecimal | &tH {0...9 A...F} | &tH4AF3 | 2 to the power of 32 (2 ³²) |

Names of constants without any identifier (&tB, &tO, &tH) are always interpreted as double (floating point) values.

These are eight-byte values with a range from -1.79769313486232E308 to -4.94065645841247E-324 for negative values and from 4.94065645841247E-324 to 1.79769313486232E308 for positive values.

String constant

The syntax of a string constant is as follows:

```
" {any character} "
```

A string constant may contain any character accessible via the (external) keyboard and is always framed by quotation marks.

NOTE

If you want a quotation mark to appear within a string, simply enter two consecutive quotation marks.

Example: "Willtek 4400 ""RAPID!"" Program" equals the following contents of the string variable: 'Willtek 4400 "RAPID!" Program'

NOTE

The maximum length (i.e. number of characters) for a string constant is 65,535.

Operators

The following table lists the numeric, string and boolean operators, along with their priority. The smaller the number in the priority column, the higher the priority.

During calculations, first all operators with priority 1 are performed, then all of priority 2 and so on. Priorities assure that the usual sequence of operations (e.g. multiplication and division first, then addition or subtraction) are performed without the need of brackets.

| Type | Operator | Priority | Meaning |
|----------------------|----------|----------|----------------------|
| Sign operators | + | 1 | Positive sign |
| | - | 1 | Negative sign |
| | NOT | 1 | Negation (Inversion) |
| Arithmetic operators | * | 2 | Multiplication |
| | / | 2 | Division |
| | \ | 2 | Integer division |
| | MOD | 2 | Modulo division |

| | | | |
|----------------------|-----|---|------------------|
| | + | 3 | Addition |
| | - | 3 | Subtraction |
| Comparison operators | = | 4 | Equality |
| | <> | 4 | nonequality |
| | < | 4 | Less |
| | > | 4 | Greater |
| | <= | 4 | Less or equal |
| | >= | 4 | Equal or greater |
| Boolean operators | AND | 5 | Logical AND |
| | OR | 5 | Logical OR |
| | XOR | 5 | Logical XOR |

NOTE

In case of an integer division or a modulo division, the two operands are first converted into integer values, before the operation is performed. This means, that the operands will be rounded before any operation takes place. The rounding procedure will follow usual 'commercial' rules.

Example: The result of (5 MOD 2.3) will be 1 as it is interpreted as (5 MOD 2) by RAPID!.

NOTE

Regardless of the format of the operands with comparison operators, the result will always be a boolean value, i.e. number 0|1.

How to use operators

- Numeric expressions may use any operator.
- Boolean expressions, which are in fact numbers (0 or 1), may use any operator.
- String expressions may use all comparison operators and the arithmetic addition.

Example: "Willtek " + "4400" will result in the string expression "Willtek 4400".

Expressions

An expression is a valid combination of variables, constants, function calls and operators.

Basically, there are three types of expressions:

- numeric expressions
- string expressions
- boolean expressions

Numeric expressions `numExp := [signOp] numVar | numConst | numFct | boolExp | numExp [operator numExp]`

Where

- `signOp` is the sign operator (+ or -),
- `numVar` is a numeric variable,
- `numConst` is a numeric constant,
- `numFct` is a BASIC function that returns a numeric result,
- `boolExp` is a boolean expression, as explained below

NOTE

A numeric expression always delivers a numeric result.

Example: `OneVar + (LEN (AString$) - 2) * 5`
This is a numeric expression that contains

- a numeric variable (`OneVar`),
- a function (`LEN`) of a string variable (`AString$`),
- the operators +, - and *
- and two numeric constants (2 and 5).

The result of this expression will be a numeric value.

String expressions `stringExp := stringVar | stringConst | stringFct { + stringVar | stringConst | stringFct }`

Where

- `stringVar` is a string variable,
- `stringConst` is a string constant,
- `stringFct` is a BASIC function that returns a string as the result.

Note: A string expression always delivers a string.

Example: `FirstName$ + " " + LastName$` is a string expression. It delivers back a string that contains the two string variables `FirstName$` and `LastName$`, separated by a blank.

Boolean expressions Boolean expressions are more complex than the preceding ones:

`boolExp := boolOperation | comparison | boolFct`

Where

- `boolOperation := NOT boolExp | boolExp AND | OR | XOR boolExp`
- `comparison := stringExp compOperator stringExp | numExp compOperator numExp`

- `compOperator` := < | <= | = | >= | >
- `boolFct` is the return value of a boolean function.

NOTE

A boolean expression always delivers either number 0 (for false) or 1 (for true) as the result. You may perform arithmetic operations with these results.

Example: `(Country$="UK") AND (NetworkPrefix>=262)`

In case that string variable `Country$` equals "UK" and at the same time the numeric variable `NetworkPrefix` is equal to or greater than 262, then this boolean expression will deliver 1 as the result. In any other case, the result delivered will be 0.

Commands

The language elements of RAPID! are BASIC commands and BASIC functions. In order to access the internal registers of the 4400, the SCPI commands are available which are described in detail in section [SCPI](#).

- Commands make up a RAPID! program. They are used to tell the 4400 what it is to do.
- Functions perform specific predefined tasks and provide values, that can be used within any RAPID! program.
- In BASIC, you may place a single command in a program line. However, placing just a single function in a program line will lead to an error.

Overview

Dependent on their general functions, the BASIC commands of RAPID! are subdivided into six groups:

- [General commands](#) are used to control the overall aspects of a program. Typical general commands are e.g. CHAIN, REM, END, WAIT or STOP.
- [Screen commands](#) control the screen output of the 4400. They comprise CLS, TEXTATTR, LOCATE, INPUT, PRINT/OUTPUT and SOFTKEYS.
- [Commands related to variables, procedures and functions](#). These are DIM, GLOBAL, VARIABLE, ERASE, SUB, FUNCTION.
- [Control commands](#) determine the flow of the program. There are two sorts of control commands: loops (FOR-NEXT, DO LOOP, WHILE, UNTIL) and conditions (IF, CASE).
- [Branch commands](#), such as GOTO, GOSUB and ON ERROR allow to change the continuous sequential flux of a program and to jump to specific areas of a program.
- [Commands for input/output handling](#) allow special input/output tasks. They are OPEN, CLOSE, INPUT, PRINT and OUTPUT.

General commands

General commands are used to control the overall aspects of a program. The following general commands are available in RAPID!:

| Command | Short description |
|----------|--|
| CHAIN | Connects additional RAPID! programs to the main program. |
| END | Ends the program. |
| LET | Assigns an expression to a variable. |
| REM or ` | Comment. |
| STOP | Debugger breakpoint. |
| WAIT | Halts the program for a specified amount of time. |

CHAIN

| | |
|--------------------|---|
| Syntax | CHAIN stringExp |
| Parameters | stringExp is a string expression. It must be identical with an existing RAPID! program file name. |
| Description | <p>Loads the specified program and executes any command lines which come before the first subroutine or function. When this is complete, control is returned to the statement following the chain command.</p> <p>Global variables of the initially calling program can be used by all chained programs.</p> <p>Subs, functions and labels are global by definition and can be used by all programs (i.e. the initially calling program and all chained programs). Consequently, all names of subs and labels have to be unambiguous.</p> <p>Note: Subs and functions of the chained program will be ignored and not run if not explicitly called (see example below).</p> <p>CHAIN is a perfect tool to split large programs into logical units or to build a library with modules for specific tasks and functions. If you do so, please keep the following hints in mind:</p> <ul style="list-style-type: none"> – In the chained program, first declare the variables required as global, – then write the required program code as a function or sub. – Use the chain statement at the beginning of the main program. This will result in the 4400 running the program code of the chained programs (with the exception of the functions themselves). Thus, the 4400 will initialize all required global variables for later use. <p>A chained program may also chain other programs down to a depth of four nested CHAINS.</p> |

Example Contents of main program:

```
PRINT "Now calling chained program"
CHAIN "TESTPROG.BAS"
PRINT "Back again to main program"
testsub
```

Contents of TESTPROG.BAS:

```
PRINT "Hello from chained program
TESTPROG"
SUB testsub()
PRINT "Subroutine called"
END SUB
```

Execution of the main program will result in the following screen output:

```
Now calling chained program
Hello from chained program TESTPROG
Back again to main program
Subroutine called
```

END

Syntax END

Parameters END does not have any parameters.

Description Ends the program.

Example IF pwr > 2.2 then END
 In case the value of variable pwr exceeds 2.2, the program will be ended immediately.

LET

Syntax [LET] numVAR = numExp | stringVar = stringExp

Parameters numVAR is a numeric variable,
 numExp is a numeric expression,
 stringVar is a string variable,
 stringExp is a string expression.

Description Assigns the value of a (numeric or string) expression to a (numeric or string) variable.
Note: LET is optional and can be omitted.

Example TheString\$ = "This is a "String": +
 Inside\$ + ", in case anybody doubts
 it."

REM or ' (Comments)

Syntax REM anyText or ' anyText

| | |
|--------------------|---|
| Parameters | <code>anyText</code> can be any text not exceeding the current line. Note: The program line extension symbol can't be used on REM lines. |
| Description | Enables you to include comments in the programs. Comments not only help other people to maintain your code; it might well be that you need some hints to understand your own code when you look at it from a long distance in time. |
| Example | <pre>REM The next line will define a variable LET i = 42 ` Who does not call his integers "i"</pre> |

STOP

| | |
|--------------------|--|
| Syntax | STOP |
| Parameters | STOP does not have any parameters. |
| Description | Stops the program execution, but does not end the program (important e.g. to keep the values of variables when debugging a program). |
| Example | If n = 10 then STOP |

WAIT

| | |
|--------------------|---|
| Syntax | WAIT msec |
| Parameters | <code>msec</code> specifies the time the program will wait in milliseconds. <code>msec</code> can only be entered as an integer value. The maximum time is 32,767 milliseconds. |
| Description | Stops the program execution for the duration of the time specified in parameter <code>msec</code> . |
| Example | <pre>PRINT "Please connect mobile now" WAIT 2000</pre> Prints user information and then halts the program for two seconds. |

Screen commands

These commands control the screen output of the 4400. The following screen commands are available in RAPID!:

| Command | Short description |
|---------|--|
| CLS | Clears the screen. |
| INPUT | Waits for a user entry and stores the value. |

| | |
|-----------------|---|
| LOCATE | Sets the cursor position on-screen for the subsequent output command. |
| PRINT or OUTPUT | Shows an expression on-screen. |
| SOFTKEYS | Labels the 4400's softkeys. |
| TEXTATTR | Sets background color and text size, font and color. |

CLS

| | |
|--------------------|--|
| Syntax | CLS [textCol] [, [bgCol] [, fontSize[b] [i]]] |
| Parameters | textCol is the new text color, bgCol is the new background color, fontSize is the new font setting, b stands for bold text, i stands for italic text. |
| Description | <p>Clears the screen and positions the cursor in the upper left screen corner.</p> <p>Optionally, new background and text settings for the subsequent output operations may be specified:</p> <p>Text and background color may be assigned one of the predefined constants BLACK, WHITE, GREY, MGREY, DGREY, RED, GREEN, BLUE, YELLOW, MAGENTA, CYAN, DGREEN, DCYAN, DBLUE, BROWN, PURPLE.</p> <p>Alternatively, you may assign a code between 0 . . &HFFFFFF. This code represents an RGB model of the screen colors, where the first two hexadecimal figures represent the red color, the middle two figures the green and the last two figures the blue color.</p> <p>The font setting is specified by the font code followed by the font size.</p> <p>The following fonts are available (corresponding codes in parentheses): Charter (char), Courier (cour), Courier 10 Pitch (courier), Dutch 801 (dutch), Helvetica (helv), Lucida (lu), Lucida Bright (lub), Lucida Terminal (lut) New Century Schoolbook (ncen) PC Sansserif (pcss), PC Serif (pcs), PC Terminal (pcterm), Swiss 721 (swiss), Symbol (symb) Technical (tech), Times (time) Useless (uselss), Utopia (utop) .</p> <p>The selectable font sizes are 8, 10, 12, 14, 18, 20, 24, 32, 48 pt.</p> <p>Optionally, the text can be set to bold (b) and/or italic (i).</p> <p>Note: The font codes are string values, they have to be put in quotation marks (e.g. "cour14"). Otherwise, a syntax error will occur.</p> |

Examples CLS (Clears the screen and keeps the current text and background settings.)
CLS , , "cour14" (Clears the screen, keeps the colors and selects Courier 14 pt as new font.)
CLS white, blue, "lu10bi" (Clears the screen and continues with white text on blue background, font Lucida 10 pt, bold and italic.)

INPUT

Syntax INPUT [[inputPrompt\$] ,variable]

Parameters inputPrompt\$ is a string expression, variable is a numerical or string variable.

Description Causes the program to wait for user input, reads data entered on the keyboard into a variable and then continues the program run.

INPUT will first output the inputPrompt\$ on the 4400 screen. Then it waits for the user's input. INPUT can read data entered either into a numeric or a string variable. The 'signal' for input finished is the CR symbol (achieved by the user pushing the **ENTER** key). The symbols entered by the user will be stored in variable and the program will continue.

Notes:

- External keyboard and the keys of the 4400 may be used in parallel. This means that an entry on the external keyboard may be confirmed by pushing the 4400's **ENTER** key.
- In case variable has been defined as a numerical variable and the user enters any other symbol than numbers and the decimal point, this will lead to an error. A simple trick to avoid that is to always read the value entered as a string and later convert that string into a numerical value:
INPUT "Please enter channel number:" _
 , channelno\$
channelno = VAL(channelno\$)
- With the help of the INPUT command, you may also read data from a file or the SCPI system.

Examples INPUT "What's your name? ", name\$
INPUT (The program waits until the user pushes the Enter key.)

LOCATE

Syntax LOCATE x, y

Parameters x stands for the cursor's new horizontal position, y for the cursor's new vertical position.

| | |
|--------------------|--|
| Description | Sets the cursor to a new position on-screen. Any subsequent text output will start from that position. Parameter <i>x</i> is counted in character positions rather than in pixels; parameter <i>y</i> in screen lines. The values for <i>x</i> and <i>y</i> are dependent on the currently set font size. However, the 4400 enables the user to scroll the screen output. For practical reasons, <i>x</i> should be kept below 50 and <i>y</i> below 25. |
| Examples | <pre>IF (channelno > 2099) then LOCATE 24, 0 Print "Input out of range. " _ ;"Please reenter." (In case the channel number entered by the user is out of range, a prompt will appear in the 'status line' of this RAPID! program.)</pre> |

PRINT (OUTPUT)

| | |
|--------------------|---|
| Syntax | <pre>PRINT [{ [USING usestr\$,] exp [, ; TAB(number);] }]or OUTPUT [{ [USING usestr\$,] exp [, ; TAB(number);] }]</pre> |
| Parameters | <p><i>exp</i> is a numeric or string expression, <i>usestr\$</i> defines an output format for numeric data, <i>number</i> stands for an integer figure, forcing the next <i>exp</i> of the same PRINT or OUTPUT command to appear on a specific position on-screen.</p> |
| Description | <p>Displays the specified expression(s) on-screen. In case the separator is a comma (,), the expression fol- lowing the current one will be displayed at the next tab- ulator position. In case the separator is a semicolon (;), the expression following the current one will be displayed directly after the previous one. OUTPUT is a 100% synonym for PRINT. PRINT USING "a.b" allows to format numeric output. While <i>a</i> gives the total amount of digits to be used for the output (including decimal places, the decimal point and the sign), <i>b</i> specifies the number of decimal places (see example below). Note: The PRINT command is also available to output data to a file or the SCPI system.</p> |

Examples PRINT var1, var2, var3 (Three numeric variables are displayed, separated by tabs).
 OUTPUT "Here "; "you "; "go!" (The phrase will be displayed as 'Here you go!').
 PRINT USING "10.2"; 21, 5 (The value will be displayed using a total of 10 digits (including the decimal point and the sign). 2 digits will be used for the decimal places. The value will be printed as '21.50' with five leading blanks.)
 OUTPUT TAB(10); 9 (The number 9 will be printed on the tenth cursor position following the current one: '9')

SOFTKEYS

Syntax SOFTKEYS [str1\$, [str2\$, [str3\$, [str4\$, [str5\$, [str6\$, [str7\$]]]]]]]

Parameters str1\$...str7\$ are string expressions.

Description Labels the 4400's softkeys. The labels specified are assigned to the softkeys from left to right. In case less than seven string expressions are used, one or more softkeys remain without label.

Examples SOFTKEYS "Key1", "", "Key3"
 (The first and third softkey from the left are labelled and may be used for INKEY or INKEYWAIT commands.)
 SOFTKEYS
 (All softkeys remain unlabelled. There are no values that could be handed over to an INKEY or INKEYWAIT command, whatever softkey may be pushed. In case of an INKEYWAIT command, this will result in an endless loop.)

TEXTATTR

Syntax TEXTATTR [textCol] [, [bgCol] [, fontSize [b] [i]]]

Parameters textCol is the new text color,
 bgCol is the new background color,
 fontSize is the new font setting,
 b stands for bold text,
 i stands for italic text.

Description Selects new color and text settings for the subsequent output operations. For detailed parameter description, please refer to command CLS in this subsection.

Examples TEXTATTR red (Text color set to red.)
 TEXTATTR , cyan, "cour18" (Courier size 18 on cyan background.)

Commands related to variables, procedures and functions

The following commands concern the declaration, output and deletion of variables and the usage of procedures and functions:

| Command | Short description |
|----------------|---|
| DIM | Declares local variables. |
| ERASE | Releases allocated memory of a variable. |
| FUNCTION | Declares a function. |
| GLOBAL | Declares global variables. |
| SUB | Declares a procedure (subroutine). |
| VARIABLE | Prints variable contents to screen or file. |

DIM

| | |
|--------------------|--|
| Syntax | DIM varName [{, varName}] |
| Parameters | varName is a valid variable name. |
| Description | <p>Declares a local variable. The variable is only accessible to the program element it has been declared in. This means, a local variable declared in a function is not accessible by the main program and vice versa. Variables that are accessible for all program elements are called 'global' (please see below for more details).</p> <p>Note: There is no need to declare local variables. Within the RAPID! environment, all variables are interpreted as local by definition.</p> <p>However, after a variable has been declared using the DIM command, all subsequent variables have to be declared with DIM as well.</p> <p>In order to keep programs easily maintainable and to avoid errors due to misspelled variable names, we strongly recommend to declare variables in larger programs.</p> |
| Example | DIM Days\$(7), Months\$(12) declares two string arrays. |

ERASE

| | |
|-------------------|-----------------------------------|
| Syntax | ERASE varName [{, varName}] |
| Parameters | varName is a valid variable name. |

Description Deletes a variable and releases the memory used. This command is useful to keep memory consumption of a program as low as possible. Especially large variable arrays or long string variables should be erased as soon as they are no longer required.
Note: Local variables are automatically erased at the termination of the related procedure or function.

Example ERASE Journal\$

FUNCTION – EXIT FUNCTION – END FUNCTION

Syntax

```
FUNCTION fnName [$] ( [ parameter  
[ {, parameter } ] ] )  
[ instructions ]  
fnName [$] = fnExp  
[ EXIT FUNCTION ]  
END FUNCTION
```

Parameters *fnName* is a valid name for the function (same syntax rules apply as for variable names),
parameter may be any valid numeric or string expression,
instructions are an (optional) number of program lines,
fnExp is an expression (of the same data type as the function itself).

Description Declares a function (FUNCTION ... END FUNCTION). Functions are commonly used to e.g. calculate the result of mathematical expressions.
After a function has been declared, it may be called by its name *fnName*.
A function call must contain all parameters in the same order as specified in the function's declaration.
The return value of a function will either be a numeric expression (in case of *fnName*) or a string expression (in case of *fnName*\$).
A function may call itself recursively.
All variables used in a function are local unless declared as GLOBAL (please see below for details).
All parameters are treated as local.
Using the EXIT FUNCTION command, you may leave any function at any place or condition (e.g. in case a value exceeds a certain limit, the function exits and passes on control to an error handling system).

Example The following example calculates the faculty ($n! := 1*2*3*...*n$) of a given integer number n . In order to do so, the function calls itself recursively:

```
FUNCTION Faculty(n)
  IF n<2 THEN
    Faculty = 1
  EXIT FUNCTION
  END IF
  Faculty = n * Faculty(n-1)
END FUNCTION
```

GLOBAL

Syntax GLOBAL varName [{, varName}]

Parameters varName is a valid variable name.

Description Declares a global variable. Global variables are accessible in the main program, in all programs included by [CHAIN](#) and in all related procedures and functions. Any global variable is valid and accessible from the program line, in which it has been declared.

Note: If a local variable has been declared or used before, that carries the same name as the global variable, then the global variable is not accessible.

Example GLOBAL strg\$, counter declares a global string and a global numeric variable.

SUB – EXIT SUB – END SUB – CALL

Syntax

```
SUB subName ( [ parameter [{, parameter } ] ] )
  [ instructions ]
  [ EXIT SUB ]
END SUB
...
[CALL] subName ( [ parameter [{, parameter } ] ] )
```

Parameters subName is a valid name for the procedure (subroutine) (same syntax rules apply as for variable names), parameter may be any valid numeric or string expression, instructions are an (optional) number of program lines.

Description SUB ... END SUB declare a procedure. Procedures are commonly used to e.g. perform certain operations that have to be carried out several times during the program run. After a procedure has been declared, it may be called by [CALL] subName. A call of a procedure must contain all parameters in the same order as specified in the procedure's declaration. A procedure may call itself recursively. All variables used in a procedure are local unless declared as GLOBAL (please see above for details). All parameters are treated as local. Using the EXIT SUB command, you may leave any procedure at any place or condition (e.g. in case a value exceeds a certain limit, the procedure exits and passes on control to an error handling system).

Example The following example defines a procedure that writes the current contents of all variables into a file with a given file name:

```
SUB VarsToFile(fileName$)
LET ff = FREEFILE
OPEN fileName$ FOR OUTPUT AS #ff
VARIABLE #ff
CLOSE #ff
END SUB
```

VARIABLE

Syntax VARIABLE [#fileNo]

Parameters fileNo is a handler of an open file.

Description Outputs the contents of all currently used variables on-screen. The currently used variables are all variables that were defined or used before the VARIABLE command appeared in the program run. In case you specify the optional file handler (#fileNo), the output will be sent to a file. **Note:** During program development, it is sometimes quite useful to know the contents of all variables. The VARIABLE command also is a good tool to unmask misspellings with variable names.

Example The following example program writes the contents of all variables to a file called 'VAR.TXT'.

```
OPEN "VAR.TXT" FOR OUTPUT AS #1
VARIABLE #1
CLOSE #1
```


Control commands

Control commands organize the process of a program. They e.g. allow to run a specific part of a program several times or to select between different sets of commands, depending on a condition.

In RAPID!, the following control commands have been implemented:

| Command | Short description |
|-----------------|---|
| DO ... LOOP | Loop with condition. |
| FOR ... NEXT | Loop with a specified number of cycles. |
| IF ... THEN | Checks a condition. |
| SELECT CASE | Selects one of several cases. |

DO ... LOOP, WHILE, UNTIL

| | |
|--------------------|--|
| Syntax | <p>Syntax 1</p> <pre>DO [{WHILE UNTIL} boolExp] [instructions] [EXIT DO] [instructions] LOOP</pre> <p>Syntax 2</p> <pre>DO [instructions] [EXIT DO] [instructions] LOOP [{WHILE UNTIL} boolExp]</pre> |
| Parameters | <p><code>boolExp</code> is a boolean expression, <code>instructions</code> are an (optional) number of program lines.</p> |
| Description | <p>Dependent on a condition (<code>boolExp</code>), this command structure repeatedly runs a program segment.</p> <p>Syntax 1 checks the condition before the loop is run. If the condition is true in the moment the program arrives at the loop, it will not be run at all.</p> <p>Syntax 2 checks the condition after the loop has been run.</p> <p><code>WHILE</code> means that the loop will be run as long as <code>boolExp</code> is true.</p> <p><code>UNTIL</code> will run the loop until <code>boolExp</code> becomes true.</p> <p><code>EXIT DO</code> causes the loop to be left instantly.</p> <p>If the optional part with the condition is left out, the loop will be run endlessly. In this case, the only way out is an <code>EXIT DO</code> command somewhere within the loop to prevent the program from being trapped.</p> |

Examples The three following examples perform the same task in different manners. They all print a text to the screen until a softkey is pushed:

```
DO
PRINT "Please push a softkey! ";
IF INKEY <> 0 THEN EXIT DO
LOOP
DO WHILE INKEY = 0
PRINT "Please push a softkey! ";
LOOP
DO
PRINT "Please push a softkey! ";
LOOP UNTIL INKEY <> 0
```

FOR ... NEXT

Syntax FOR countVar = numStart TO numEnd [
STEP numStep]
[instructions]
[EXIT FOR]
[instructions]
NEXT countVar

Parameters countVar is a numeric variable,
numStart, numEnd, numStep are numeric expressions,
instructions are an (optional) number of program lines.

Description Repeats a program segment a specified number of times. To do so, a counter (countVar) is used. The initial value of the counter is stated in the numStart expression, while the final value is stated in the numEnd expression. After each run, the counter is increased by numStep. If the optional STEP part is missing, the counter will be incremented by 1 per run. After the loop was run, the condition (countVar = numEnd) will be checked. If the condition is false, the loop will be run again. If the condition is true, the program will resume with the command following the NEXT countVar command. In order to immediately exit a FOR...NEXT loop, use the EXIT FOR command.

Examples The following example program prints the ASCII characters A..Z to the screen:

```
FOR i=65 to 90
PRINT CHR$(i)
NEXT i
```

The following example reverts the string `Matter$` into the string `AntiMatter$`:

```
AntiMatter$ = ""
FOR i=len(Matter$) TO 1 STEP -1
AntiMatter$ = AntiMatter$ + MID$(Mat-
ter$, i, 1)
NEXT i
```

IF ... THEN, ELSE, ELSEIF, END**IF****Syntax**

```
Syntax 1
IF boolExp1 THEN instructions1 [ ELSE
instructions2 ]
Syntax 2
IF boolExp2 THEN
[ instructions3 ]
[ ELSEIF boolExp3 THEN
[ instructions4 ] ]
[ ELSE
[ instructions5 ] ]
END IF
```

Parameters

`boolExp1/2/3` are boolean expressions,
`instructions1/2/3/4/5` are an (optional) number of
program lines.

Description

Depending on a boolean expression or condition (`bool-Exp`), the program will resume with different sets of commands.

There are two different forms of this command, indicated by 'Syntax 1' and 'Syntax 2' above.

Syntax 1 is the 'one-line' command. If `boolExp1` is true (or a numeric value is $\neq 0$), the command(s) following `THEN` will be executed. If `boolExp1` is false (or a numeric value = 0), the commands following the optional `ELSE` will be executed.

Syntax 2 is the 'multiline' version of the command. This syntax should be used in case there are several commands to be executed, depending on the conditions. In case `boolExp2` is true, `instructions3` will be executed.

If `boolExp2` is false, the subsequent `ELSEIF` command will be looked for. If this is true (i.e. `boolExp3` is true), `instructions4` will be executed.

There may be several `ELSEIF` conditions within one `IF`-clause.

Should all `boolExp2/3` be false, `instructions5` will be performed.

Examples

```
IF tired THEN PRINT "Go to bed!" ELSE
PRINT "Go shopping!"
```

```
IF (i <= 0) THEN
PRINT "Not a positive number"
ELSEIF (i MOD 2 = 0) THEN
PRINT "Even"
ELSE
PRINT "Odd"
END IF
```

SELECT CASE, CASE ELSE, END SELECT

Syntax

```
SELECT CASE varName
CASE varExp [ { , varEXP } ] instruc-
tion
[ { CASE varExp [ { , varEXP } ]
instruction } ]
CASE ELSE instruction
END SELECT
```

Parameters

`varName` is a valid variable name,
`varEXP` are expressions of the same type as `varName`,
`instruction` are valid RAPID! commands.

Description

Performs one of several blocks of commands, depending on the value of `varName`.

If `varName` is identical with any one of the `varExp` expressions, the commands following that `CASE` statement will be performed until the subsequent `CASE` or the `END SELECT` command is reached. The program will then resume with the command following the `END SELECT` statement.

If `varName` is identical with several `varExp`, only the first one found will be executed.

Several `varExp` may be used in the same `CASE` command; they need to be separated by commas.

Examples

```
SELECT CASE Day
CASE 1: PRINT "MONDAY"
CASE 2: PRINT "TUESDAY"
CASE 3: PRINT "WEDNESDAY"
CASE 4: PRINT "THURSDAY"
CASE 5: PRINT "FRIDAY"
CASE 6, 7: PRINT "WEEKEND"
CASE ELSE
PRINT "Data out of range"
END
END SELECT
```

```

SELECT CASE Day$
CASE "MONDAY", "TUESDAY", "WEDNESDAY",
"THURSDAY", "FRIDAY"
PRINT "Got to work!"
CASE "WEEKEND" : PRINT "I'm off for
the seaside!"
END SELECT

```

Branch commands With the help of these commands, it is easy to branch program execution, e.g. depending on conditions or user entries.

| Command | Short description |
|------------------------|---|
| ERROR | Creates an error with a specified error number. |
| GOSUB ... RETURN | Branches to the specified label and returns at the next RETURN command. |
| GOTO | Branches to the specified label. |
| ON ERROR | Branches to the specified label, if an error occurs during program execution. |

ERROR

| | |
|--------------------|--|
| Syntax | ERROR errNo |
| Parameters | errNo is an integer number. |
| Description | Creates a runtime error with the specified error number. This command enables you to create your own error codes. Note: For the built-in error codes, please refer to the table of Runtime errors . |
| Examples | The following example defines an error code based on a simple condition and gives a first idea of error handling. <pre> IF (frequ > 2 GHz) THEN ERROR 57 ... SELECT CASE err case 57 : Print "Input out of range!" END CASE </pre> |

GOSUB ... RETURN

| | |
|-------------------|---|
| Syntax | GOSUB targetLabel [instructions] RETURN |
| Parameters | targetLabel is a label. |

Description Branches the program to the symbolic label and continues program execution with the commands following that label.

As soon as the RETURN command is encountered, program execution will come back to the command following the initial GOSUB command.

Example

```
IF Day$="Friday" THEN GOSUB fishDay
ELSE GOSUB meatDay
...
fishDay
PRINT "Today we eat fish!"
RETURN
...
meatDay: PRINT "Today we will have a
steak!"
RETURN
```

GOTO

Syntax GOTO targetLabel

Parameters targetLabel is a label.

Description Branches to the symbolic label and continues program execution with the commands following that label.

Examples

```
...
IF i>10 THEN GOTO loopOver
...
loopOver: PRINT "Ten steps later..."
```

ON ERROR, RESUME

Syntax ON ERROR { GOTO {targetLabel | 0} | RESUME NEXT }

```
...
RESUME [ NEXT | targetLabel ]
```

Parameters targetLabel is a label.

| | |
|--------------------|--|
| Description | <p>If an error occurs during program execution, the program would normally be terminated.</p> <p>The <code>ON ERROR</code> command structure gives you the chance to trap the error, handle it, and then continue the program without a break.</p> <p><code>ON ERROR GOTO targetLabel</code> branches to the specified label in case an error occurs.</p> <p><code>ON ERROR GOTO 0</code> deactivates the error trapping. Any subsequent error will stop the program.</p> <p><code>RESUME</code> continues the program with the command that caused the error.</p> <p><code>RESUME targetLabel</code> continues the program at the specified label.</p> <p><code>RESUME NEXT</code> continues the program with the command following the one, that caused the error.</p> <p>Note: If an error occurs within the error handling, the program will be terminated.</p> <p>There are several built-in functions available for error handling.</p> |
| Examples | <p>In the first example, the 4400 prints an error message, giving the error code, the line in which the error occurred and the name of the related RAPID! program file.</p> <pre>ON ERROR GOTO errorHandler ... errorHandler PRINT "Error "; ERR; " in line "; ERL; " in file "; ERF\$; " Resuming..." RESUME NEXT</pre> <p>In the following example, a user-specific error code is generated first. Then a <code>CASE SELECT</code> command is used to print the correct error message.</p> <pre>ON ERROR GOTO errorHandler ... LET measQ = measNum / measRes IF (measRes<0) THEN ERROR 57 ... errorHandler: SELECT CASE ERR CASE 104: PRINT "Division by zero! Restart measurement!" CASE 57: PRINT "Measurement result negative! Restart measurement!" CASE ELSE PRINT "Unknown error. Program halts." END END CASE RESUME NEXT</pre> |

Commands for input/ output handling

These commands enable special input/output tasks related to the 4400's disk drive, hard disk and communication ports. Frequently, input/output operations are abbreviated with I/O.

| Command | Short description |
|--------------------|--|
| CLOSE | Closes a previously opened file. |
| INPUT | Reads data from an open file or from an open port. |
| OPEN | Opens a file or a port and allows for reading or writing data. |
| PRINT or OUTPUT | Writes data to an open file or port. |

CLOSE

| | |
|--------------------|--|
| Syntax | CLOSE [fileNo] |
| Parameters | fileNo is a numeric expression, representing a valid file number. fileNo must result in a positive integer. Software people tend to call fileNo the 'file handle'. |
| Description | Closes the file with the specified fileNo. After the file has been closed, fileNo is no longer related to the initial file and may be used again later to open a completely different file. If fileNo is omitted, then all currently open files will be closed. As long as a file is closed, it is not accessible for I/O operations unless it is opened again, using the OPEN command (please see below for details). |
| Example | The following example program reads a text file and stores the data read from the file in a string array. <pre>LET i=0 OPEN "STRINGS.TXT" FOR INPUT AS #1 DO WHILE NOT EOF(1) LET i=i+1 INPUT #1, theStrgs\$(i) LOOP CLOSE #1</pre> |

INPUT

| | |
|-------------------|--|
| Syntax | INPUT #fileNo , variable |
| Parameters | fileNo is a numeric expression resulting in a valid file handle (a positive integer). variable is a numeric or string variable. |

Description Reads data from a previously opened file or communication port with the specified file handle into a variable. `INPUT` will always read a complete line and then increment the line counter. This means that a second `INPUT` command with the same file handle will read the second line of the file.

Notes:

- Before `INPUT` may read from a file, it needs to be opened for input first (please see the description of the `OPEN` command and the `INPUT` mode in this subsection). If you try to read from a file that was not opened before, a runtime error will occur.
- It is recommended that you check if the end of the file was reached before you issue a subsequent `INPUT` command: if `INPUT` tries to read after the end of the file was reached, a runtime error will occur.
- With the help of the `INPUT` command, you may also prompt the user to enter data during runtime of the program. Please see subsection "[Screen commands](#)" on page 116 for reference.
- `INPUT` also reads data from the SCPI system. Please refer to section [SCPI and RAPID!](#) for details.

Example The following program reads a text file and stores the data read from the file in a string array.

```
LET i=0
OPEN "STRINGS.TXT" FOR INPUT AS #1
DO WHILE NOT EOF(1)
LET i=i+1
INPUT #1, theStrgs$(i)
LOOP
CLOSE #1
```

OPEN (on files)

Syntax `OPEN fileName$ FOR openMode AS #fileNo`

Parameters `fileName$` is a string expression containing a valid file name including an optional path or the specification for a communication port of the 4400 (please see below for details).

`openMode` := {APPEND | INPUT | OUTPUT}
`fileNo` is a numeric expression resulting in a positive integer (the so-called file handle).

Description Opens a file or a communication port for input or output operations. As long as a file has not been opened, I/O operations can not be performed and a runtime error will occur.

`fileName$` is a string, containing either a valid file name or a specification for a communication port of the 4400 (please see below for details).

`openMode` may either be `INPUT`, `OUTPUT` or `APPEND`. When `openMode` is `INPUT`, then the file can only be read from. If the file does not exist, a runtime error will occur.

In case, `openMode` is `OUTPUT`, then it is only possible to store data to that file. If the file does not exist, it will be created anew. Should the file already exist, then it will be **overwritten without warning**.

`openMode APPEND` is similar to `OUTPUT`. However, the file will not be overwritten with new data. Instead, new data will be added at the end of the existing file. Should there be no file, it will be created anew.

Notes:

- When a file is already open, a further `OPEN` command may open the same file as long as the mode is identical. If the mode is not identical (e.g. a file was opened for output previously and shall now be opened for input), a runtime error will occur.
- `OPEN` may also be used to open a path to the SCPI system. See section [SCPI and RAPID!](#) for details.

Examples This example writes a string to a text file.

```
LET File=FREEFILE
OPEN "STRINGS.TXT" FOR OUTPUT AS
#File
PRINT #File, "Save me"
CLOSE #File
```

OPEN (on communication ports)

Syntax `OPEN fileName$ AS #fileNo`

Parameters `fileName$` is a string expression containing a specification for a communication port on the 4400 or a valid file name (please see above for details).
`fileNo` is a numeric expression resulting in a positive integer (the so-called file handle).

Description

Opens a communication port for input or output operations. As long as a port has not been opened, I/O operations can not be performed and a runtime error will occur.

`filename$` is a string, containing a specification for a communication port of the 4400 or a valid file name (please see above for details).

Valid ports are GPIB0 (for the GPIB/IEEE-Bus), SCPI, COM1 and COM2 (the two serial interfaces of the 4400, please see ["Rear panel" on page 8](#) for reference).

A port name always needs to be followed by a colon (:). After the colon, you may specify parameters for the interface and the communication.

Please note that the values of parameters must be positive integers if not specified otherwise.

Parameters for GPIB:

`addr=` this parameter specifies the primary GPIB bus address. The valid range is 1...32.

`sec=` states the secondary GPIB bus address. The valid range is 0...64.

`time=` with the help of this parameter, you may set a timeout in milliseconds.

`eoI=` states, whether the hardware EOI will be used (parameter set to 1) or not (parameter set to 0).

`srq=` a '1' states that the hardware 'Service Request' will be used, while a '0' indicates that it will not be used.

`end=` specifies the end byte (usually ASCII 10 or ASCII 13). The valid range is 0...255.

Parameters for SCPI:

There are no parameters for SCPI.

Parameters for COM1/COM2:

`baud=` this parameter specifies the baud rate of the serial interface. It is highly recommended to always specify a baud rate when working with the serial ports to keep them in a defined state. The following values may be entered: 110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200.

`bits=` states the number of data bits used. The valid range is 5...8.

`stop=` specifies the number of stop bits to be used. The entry range is 1|2.

`parity=` states, whether the parity check bit will be used. Value '0' switches the use of a parity bit off. '1' indicates odd parity and '2' even parity.

`xonoff=` defines whether the software handshake will be used. A '0' switches the software handshake off, while '1' turns it on. Software handshake means that the receiver (RAPID! program) may halt the transmission by sending ASCII 19 (Ctrl-S) to the transmitter. A Ctrl-Q (ASCII 17) sent to the transmitter will continue transmission again. Default value is '0' (no software handshake).
`rts=` defines whether the hardware handshake will be used. A '0' switches the hardware handshake off, while '1' turns it on. Hardware handshake means that the receiver (RAPID! program) will only allow the transmitter to send while it is expecting data in an `INPUT` command. Physically, pin RTS will be kept to 'low' and only turn to 'high' when the RAPID! program expects input data. Default value is '0' (no hardware handshake).

`cts=` states, whether the signal on pin CTS will be ignored (value = '1') or listened to (value = '0'). If `cts=0`, the transmitter is only allowed to send data as long as the signal on pin CTS is active (i.e. 'high'). In case CTS is to be ignored `cts=1`, the receiver (RAPID! program) will always wait until all characters have been transmitted (please see parameter `wait=`). Default value is '0' (CTS signal will be followed).

`time=` with the help of this parameter, you may set a timeout in seconds. When the timeout expires, an empty string will be delivered to the RAPID! program as input. Default is 600 seconds (5 minutes).

`end=` specifies the end byte (usually ASCII 10 or ASCII 13). The valid range is 0...255. If an end byte was specified, data will be read in until the specified end byte is recognized. If no end byte is specified (`end=0`), only the currently available bytes will be read in. Default is `end=0`.

`flush=` defines the use of the port input buffer. In case `flush=0`, all characters will be read in, even when they occurred at the port long before the port was opened with the `OPEN` command. `flush=1` however clears the input buffer in the moment, the `OPEN` command for that port occurs in the program run. Default is `flush=0` (all data stored in input buffer will be kept).

`wait=` specifies whether the program is to wait until all characters have been transmitted over the port, or not. `wait=0` will continue the program after the transmission of data has been initialized. `wait=1` will halt the program until all characters have been transmitted. Default is `wait=0`.

Notes:

The SCPI port may only be opened once. If you try to open the previously opened SCPI port, a runtime error will occur.

All other ports may be opened several times with different parameters as long as the file handles are different.

Examples

This example opens the GPIB0 port.

```
gpib=FREEFILE
OPEN "GPIB0:addr=25,sec=0,time=10,
eoi=1,srq=1,end=13" AS #gpib
PRINT #gpib, "Output to GPIB"
INPUT #gpib, ingpib$
CLOSE #gpib
```

The program segment below opens a serial interface.

```
com2=FREEFILE
open-
com$="COM2:baud=9600,bits=8,stop=1,
parity=1,xonoff=0"
opencom$=opencom$ +
"time=10,end=10,flush=1,wait=1"
open opencom$ as #com2
PRINT #com2, "Output to serial port
2"

INPUT #com2, incom2$
CLOSE #com2
```

PRINT or OUTPUT

Syntax

```
PRINT #fileNo [USING formatstring ,] exp
[ [TAB[n];] [BYTE(m)] ] [{ ,|; exp} ]
or
OUTPUT #fileNo [USING formatstring ,] exp
[ [TAB[n];] [BYTE(m)] ] [{ ,|; exp} ]
```

Parameters

exp is a numeric or string expression.
fileNo is a numeric expression resulting in a positive integer (the so-called file handle).
formatstring specifies the output format to be used, e.g. the number of decimal places.
n is an optional parameter giving the distance between the previous and this tabulator position.
m is the decimal representation of the ASCII character to be printed.

Description Writes the specified expression(s) to a previously opened file, using the specified file handle.
With the help of this command, several expressions may be printed in the same line. When the separator is a comma (,), the subsequent expression will be written at the next tabulator position.
When the separator is a semicolon (;), the subsequent expression will be written directly behind the previous one.

Notes:

- Before `PRINT` may output to a file, it needs to be opened for output or appending first (please see the description of the `OPEN` command and the `OUTPUT` and `APPEND` modes in this subsection). If you try to write to a file that was not opened before, a runtime error will occur.
- With the help of the `PRINT` command, you may also print data to the 4400's screen. Please refer to subsection "[Screen commands](#)" on page 116 for details.
- `PRINT` also reads data from the SCPI system. Please check with section "[SCPI and RAPID!](#)" on page 173 for details.

Examples This example writes a string to a text file.

```
LET File=FREEFILE
OPEN "STRINGS.TXT" FOR OUTPUT AS #File
PRINT #File, "Save me"
CLOSE #File
```

Please compare the use of the `PRINT` command in this context with the third example for the `PRINT` command in the [Screen commands](#) description, too.

Functions

The functions of the RAPID! environment are subdivided into five groups:

- [Numeric functions](#). They convert numeric values into different systems or return numerical values. Typical examples are `BIN`, `BIN$`, `CINT`, `OCT`, `OCT$`, `HEX`, `HEX$`, `VAL` or `VAL$`.
- [String functions](#) operate on strings or return string values. Typical examples are `ASC`, `CHR$`, `INSTR`, `LEN`, `LEFT$`, `MID$`, `RIGHT$` or `SPACE$`.
- [I/O functions](#) provide useful information concerning input and/or output operations. Typical examples are `DATE$`, `CLOCK`, `EOF` or `FREEFILE`.
- [Functions for error handling](#) return values or strings related to errors. `ERR`, `ERL` and `ERF$` belong to this group.
- [Mathematical functions](#) support program-internal calculations with the mathematical functions `SQRT`, `SIN`, `COS` or `TAN` just to name a few.

Numeric functions

This subsection describes functions that convert numbers or perform numeric operations.

Note: In case uninitialized variables are used as arguments, the returned values will usually be "0" or 0.

| Function | Short description |
|-----------------|--|
| BIN | Converts a binary representation of a numeric expression into its decimal equivalent. |
| BIN\$ | Transforms a numeric expression into its binary representation. |
| CINT | Rounds a numeric expression to an integer, using commercial rounding. |
| HEX | Converts a hexadecimal representation of a numeric expression into its decimal equivalent. |
| HEX\$ | Transforms a numeric expression into its hexadecimal representation. |
| OCT | Converts an octal representation of a numeric expression into its decimal equivalent. |
| OCT\$ | Transforms a numeric expression into its octal representation. |
| VAL | Converts a string expression into a numeric expression. |
| VAL\$ | Converts a numeric expression into a string expression. |

BIN

| | |
|--------------------|--|
| Syntax | <code>BIN (string\$)</code> |
| Parameters | <code>string\$</code> is a string expression, containing a valid binary representation of an integer number. |
| Description | Returns the decimal representation of a binary expression contained in <code>string\$</code> . The value returned will be an integer. |
| Examples | <code>LET i=BIN("101")</code> assigns '5' to variable <code>i</code> . |

BIN\$

| | |
|-------------------|---|
| Syntax | <code>BIN\$ (aNumber [, digits])</code> |
| Parameters | <code>aNumber</code> is a valid numeric expression, <code>digits</code> is an expression, resulting in a positive integer. |

Description Returns a string containing the binary representation of `aNumber`. In case, `aNumber` is not an integer, the decimals will be truncated (that means no rounding will take place).
When `digits` is stated, the string will be filled up with zeros (0) to the left.

Examples `LET b$=BIN$(126, 8)` assigns "01111110" to variable `b$`.

CINT

Syntax `CINT (aNumber)`

Parameters `aNumber` is a valid numeric expression.

Description Converts a numeric expression to the related integer figure, using commercial rounding.

Examples `i = CINT(1.49)` assigns '1' to variable `i`.
`j = CINT(1.50)` assigns '2' to variable `j`.

HEX

Syntax `HEX (string$)`

Parameters `string$` is a string expression, containing a valid hexadecimal representation of an integer number.

Description Returns the decimal representation of a hexadecimal expression contained in `string$`.
The value returned will be an integer number.

Examples `LET i=HEX("FF")` assigns '255' to variable `i`.

HEX\$

Syntax `HEX$(aNumber [, digits])`

Parameters `aNumber` is a valid numeric expression,
`digits` is a numeric expression, resulting in a positive integer.

Description Returns a string containing the hexadecimal representation of `aNumber`. In case, `aNumber` is not an integer, the decimals will be truncated (that means no rounding will take place).
When `digits` is stated, the string will be filled up with zeros (0) to the left.

Examples `LET a$=HEX$(254, 4)` assigns "00FE" to variable `a$`.

OCT

| | |
|--------------------|---|
| Syntax | <code>OCT(string\$)</code> |
| Parameters | <code>string\$</code> is a string expression, containing a valid octal representation of an integer number. |
| Description | Returns the decimal representation of an octal expression contained in <code>string\$</code> . The value returned will be an integer number. |
| Examples | <code>LET i=OCT("15")</code> assigns '13' to variable <code>i</code> . |

OCT\$

| | |
|--------------------|--|
| Syntax | <code>OCT\$(aNumber [, digits])</code> |
| Parameters | <code>aNumber</code> is a valid numeric expression, <code>digits</code> is a numeric expression, resulting in a positive integer. |
| Description | Returns a string containing the octal representation of <code>aNumber</code> . In case, <code>aNumber</code> is not an integer, the decimals will be truncated (that means no rounding will take place). When <code>digits</code> is stated, the string will be filled up with zeros (0) to the left. |
| Examples | <code>LET a\$=OCT\$(254, 6)</code> assigns "000376" to variable <code>a\$</code> . |

VAL

| | |
|--------------------|---|
| Syntax | <code>VAL(string\$)</code> |
| Parameters | <code>string\$</code> is a string expression, containing a valid numeric representation of a decimal figure like "-1.4" or "3E+5" |
| Description | Converts a decimal number contained in <code>string\$</code> to that decimal number. If the string does not contain a valid decimal number, '0' will be returned. |
| Examples | <code>i = VAL("1.234")</code> assigns '1.234' to variable <code>i</code> To avoid runtime errors due to typing errors with an <code>INPUT</code> command, first a string is read in that is later converted into the decimal representation of the figure entered. <code>INPUT "Please enter channel number:" _</code> <code>, channelno\$</code> <code>channelno = VAL(channelno\$)</code> |

VAL\$

| | |
|--------------------|--|
| Syntax | VAL\$(aNumber [, digits]) |
| Parameters | aNumber is a numeric expression. digits is a numeric expression, resulting in a positive integer. |
| Description | Returns a string containing the numeric representation of aNumber. The number of decimal places may be specified using the optional digits command element. In case aNumber uses more decimal places than specified with digits, the additional decimal places will be truncated (i.e. there will be no rounding). If digits is omitted, six decimal places will be used. |
| Examples | VAL\$(123) returns the string "123.000000". VAL\$(123.456789, 2) returns the string "123.45". VAL\$(123.456789, 0) returns the string "123". |

String functions

This subsection describes functions that operate on strings.

Note: In case uninitialized variables are used as arguments, the returned values will usually be empty strings or '0'.

| Function | Short description |
|----------|--|
| ASC | Returns the ASCII code of the first character in a string. |
| CHR\$ | Returns the character representing the ASCII value of an integer number as a string. |
| INSTR | Searches for the occurrence of a specific sequence of characters within a string. |
| LEN | Returns the number of characters included in a string. |
| LEFT\$ | Reads a specific part of a string, starting from its beginning. |
| MID\$ | Reads a specific part of a string. |
| RIGHT\$ | Reads a specific part of a string, starting from its end. |
| SPACE\$ | Returns a string, filled with a specific number of blanks. |

ASC

| | |
|--------------------|---|
| Syntax | ASC(aString\$) |
| Parameters | aString\$ is a valid string expression. |
| Description | Returns the ASCII code of aString\$'s first character. If the string is empty, '0' will be returned. |

Examples ASC ("A") will return '65'.

CHR\$

Syntax CHR\$(aNumber)

Parameters aNumber is a valid numeric expression.

Description Returns the character corresponding to the ASCII code of aNumber as a string.
If aNumber is outside the range 0...255, then its least significant byte will be used (i.e. a modulo 256 operation will be performed on aNumber).
Should aNumber have decimal places, then those will be truncated (this means that there will be no rounding).
If this command is used on a variable that has not been initialized before, '0' will be returned.

Examples a\$=CHR\$(65) assigns "A" to variable a\$.

INSTR

Syntax INSTR([startPos,] aString\$, searchString\$)

Parameters startPos is a numeric expression, interpreted as positive integer.
aString\$ and searchString\$ are string expressions.

Description Searches aString\$ for the first occurrence of searchString\$.
The search will start at the position given by startPos or, if omitted, at the beginning of aString\$.
Return value is the position, where searchString\$ begins.
If searchString\$ is not found within aString\$, '0' will be returned.
Should startPos have decimal places, then those will be truncated (this means that there will be no rounding).

Examples a\$ = "international or national"
i = INSTR(a\$, "national")
j = INSTR(8, a\$, "national")
In this example, variable i will be set to '6', while variable j will be set to '18'.

LEN

Syntax LEN(aString\$)

Parameters aString\$ is a string expression.

Description Returns the number of characters contained in aString\$.

Examples `LEN("A short one")` will return '11'.

LEFT\$

Syntax `LEFT$(aString$, chars)`

Parameters `aString$` is a string expression,
`chars` is a numeric expression, interpreted as positive integer.

Description Returns the first `chars` characters of `aString$`, or the whole string if `chars` is greater than the amount of characters contained in `aString$`.
Should `chars` have decimal places, then those will be truncated (this means that there will be no rounding).

Examples `a$="Good morning"`
`b$=LEFT$(a$, 4)`
Will assign the string "Good" to `b$`.

MID\$

Syntax `MID$(aString$, startPos, chars)`

Parameters `aString$` is a string expression,
`startPos` and `chars` are numeric expressions, interpreted as positive integers.

Description Returns `chars` characters of `aString$`, starting at position `startPos`.
If `startPos+chars` is greater than the total amount of characters contained in `aString$`, the entire string, starting from `startPos` will be returned.
If `chars` is omitted, the entire string, starting from `startPos` will be returned.
If `startPos` is greater than the total amount of characters contained in `aString$` (i.e. `startPos` points past the end of `aString$`), then an empty string will be returned.
Should `startPos` or `chars` have decimal places, then those will be truncated (this means that there will be no rounding).

Examples `MID$("This is a test", 6, 4)` will return the string "is a".

RIGHT\$

Syntax `RIGHT$(aString$, chars)`

Parameters `aString$` is a string expression,
`chars` is a numeric expression, interpreted as positive integer.

Description Returns the last `chars` characters of `aString$`, or the whole string if `chars` is greater than the amount of characters contained in `aString$`. Should `chars` have decimal places, then those will be truncated (this means that there will be no rounding).

Examples `a$="Good morning"`
`c$=RIGHT$(a$, 7)`
Will assign the string "morning" to `c$`.

SPACE\$

Syntax `SPACE$(chars)`

Parameters `chars` is a numeric expression, interpreted as positive integer.

Description Returns a string containing a total amount of `chars` blanks.

Examples `d$ = "Long"+SPACE$(1)+"distance"`
will assign the string "Long distance" to `d$`.

I/O functions The 4400's RAPID! environment provides a large number of I/O functions:

| Function | Short description |
|---------------------------|---|
| CHDIR | Changes the current directory. |
| CLOCK | Returns the time since the system was last started. |
| CURDIR\$ | Returns the currently selected directory. |
| DATE\$ | Returns a string, containing the system date. |
| DIR\$ | Function to read out the entries in a directory. |
| EOF | Checks whether the end of a file was reached. |
| EVENTWAIT, EVENTSTATUS | Waits for an event and returns the event code. |
| FREEFILE | Returns the subsequent free file handle. |
| INKEY | Checks whether a softkey was pushed and returns its code. |
| INKEYWAIT | Halts the program until a softkey is pushed. |
| KILL | Deletes a file. |
| MKDIR | Creates a new directory. |
| NAME | Renames an existing file. |
| RMDIR | Removes a directory. |

| | |
|---------------------|--|
| <code>SHELL</code> | Calls the operating system shell in order to perform operating system commands (e.g. so-called pipes). |
| <code>TIME\$</code> | Returns a string containing the system time. |

CHDIR

| | |
|--------------------|---|
| Syntax | <code>CHDIR path\$</code> |
| Parameters | <code>path\$</code> is a valid string expression. |
| Description | Changes the current directory to the directory specified in <code>path\$</code> . <code>path\$</code> may either contain the absolute or relative path. If <code>path\$</code> is not a valid path, the current directory will be kept and a runtime error will occur. |
| Example | <code>CHDIR = "/usr/new"</code> |

CLOCK

| | |
|--------------------|---|
| Syntax | <code>CLOCK</code> |
| Parameters | There are no parameters. |
| Description | Returns the time since the system was started last. The value returned is a floating-point double value, expressing the time expired since the last start of the system in seconds. |
| Example | <code>t = CLOCK</code> Assigns the time since the system was started last to variable <code>t</code> (<code>t</code> would then be set to e.g. '3724.23' (seconds)). |

CURDIR\$

| | |
|--------------------|--|
| Syntax | <code>CURDIR\$</code> |
| Parameters | There are no parameters. |
| Description | Returns the currently selected directory. |
| Example | <code>PRINT "Current directory:";CURDIR\$</code> |

DATE\$

| | |
|-------------------|--------------------------|
| Syntax | <code>DATE\$</code> |
| Parameters | There are no parameters. |

Description Returns a string containing the system date in the following format: "mmm-dd-yyyy", where "mmm" represent a three-letter abbreviation of the month, "dd" stand for the integer, giving the day of the month and "yyyy" represent the current system year in four-digit code.

Examples DATE\$ would have returned "Jan-03-2001" on the day this subsection was created.

DIR\$

Syntax DIR\$ [(path\$)]

Parameters path\$ is a valid string expression.

Description Returns an entry in the directory specified with the optional command element path\$. DIR\$ will read the first entry in the specified directory only. Repeated DIR\$ function calls (without the path command element) will read out the subsequent entries of the currently selected directory one by one. If there are no more entries, an empty string will be returned. path\$ may either contain the absolute or relative path. If path\$ is not a valid path, the returned DIR\$ will be empty and a runtime error will occur.

Example The following example will print a list of all entries in the directory specified.

```
PRINT "Directory entries: "; DIR$ ("/
home")
DO    d$ = DIR$
      IF d$ = "" THEN EXIT DO
      PRINT d$
LOOP
```

EOF

Syntax EOF(fileHandle)

Parameters fileHandle is a numeric expression, interpreted as positive integer.

Description Checks whether the end of the file, specified by fileHandle, has been reached. If so, EOF will return 1 (or true), otherwise 0 (or false). fileHandle needs to be a currently valid file handle, and the file needs to be opened first, before EOF may be used (otherwise, a runtime error will occur). **Note:** When you try to read from a file with the INPUT command after the end of the file was reached, a runtime error will occur. To avoid that, it is highly recommended to check EOF, before you issue an INPUT command.

Example

The following example program reads a text file and stores the data read from the file in a string array.

```
LET i=0
OPEN "STRINGS.TXT" FOR INPUT AS #1
DO WHILE NOT EOF(1)
    LET i=i+1
    INPUT #1, theStrgs$(i)
LOOP
CLOSE #1
```

EVENTWAIT, EVENTSTATUS

Syntax

```
EVENTWAIT (timeOut)
EVENTSTATUS
```

Parameters

`timeOut` is a numeric expression, interpreted as positive integer.

Description

EVENTWAIT

Waits for an event and returns the event code.

Events may be created by various devices or tasks (e.g. reception of a string on the GPIB bus).

`timeOut` specifies a wait time in milliseconds. During the wait time, the program will be halted and will wait for the event to occur. If the expected event does not occur, a timeout will be generated.

EVENTWAIT may be used to react on any of the following events:

KEY push of a softkey. In this case, the file handle returned will be '0'.

TIME-OUT means that there was no device or task creating an event; time simply expired without anything happening. In this case, the file handle returned will be '-1'.

GPIB an event on the GPIB bus (e.g. data have been applied and are ready to be read).

SCPI an event on the SCPI interface (e.g. a measurement value is ready to be handed over).

EVENTSTATUS

In case there is an event created by the GPIB bus, EVENTSTATUS may be used to read the status byte.

Example

The following example labels the softkeys, opens several communication ports and then waits 5 seconds for an event to occur.

Depending on the device or task creating the event, event handling will take place.

```
SOFTKEYS "OK",,,,,,"EXIT"
OPEN "GPIB0:..." AS #1
scpi = FREEFILE
OPEN "SCPI:" AS #scpi
DO
SELECT CASE EVENTWAIT(5000)
CASE -1: PRINT "Time out"
CASE 0
IF INKEY = 7 THEN END
PRINT "Softkey 'OK' was pushed."
CASE 1
INPUT #1, gpib$
PRINT "Received on GPIB:",
gpib$,EVENTSTATUS
CASE SCPI
INPUT #scpi, scpi$
PRINT "Received from SCPI:", scpi$
CASE ELSE
PRINT "Any other event occurred!"
END SELECT
LOOP
```

FREEFILE

Syntax

FREEFILE

Parameters

There are no parameters.

Description

Returns the subsequent available file handle as a positive integer.

The returned file handle may then be used by an [OPEN \(on files\)](#) command.

FREEFILE helps to avoid conflicts with file handles.

Example

This example writes a string to a text file.

```
LET File=FREEFILE
OPEN "STRINGS.TXT" FOR OUTPUT AS
#File
PRINT #File, "Save me"
CLOSE #File
```

INKEY

Syntax

INKEY

Parameters

There are no parameters.

Description Checks whether a softkey was pushed and returns the number of the softkey. The softkeys on the 4400 are numbered from '1' to '7', starting from the left. If no softkey was pushed, `INKEY` will return '0'. In contrary to the `INKEYWAIT` command, `INKEY` does not halt the program.

Note: Please be sure that there have been labels assigned to the softkeys with the [SOFTKEYS](#) command, before you use `INKEY` to read out the softkey pushed.

Example

```
SOFTKEYS "Accept", "", "", "", "", "", "",  
"Exit"  
DO  
    IF INKEY = 7 THEN GOTO endProg  
LOOP
```

INKEYWAIT

Syntax `INKEYWAIT`

Parameters `INKEYWAIT` does not have any parameters.

Description Halts program execution until the user pushes a softkey.

Notes:

- If you want to determine the softkey pushed without halting the program, choose the `INKEY` function instead.
- Please assign labels to the softkeys using the [SOFTKEYS](#) command, before using `INKEYWAIT`. Otherwise, the program may wait for eternity.

Example

```
SOFTKEYS "OK", "", "", "", "", "", "",  
"EXIT"  
IF INKEYWAIT = 7 THEN END
```

KILL

Syntax `KILL fileName$`

Parameters `fileName$` is a valid string expression.

Description Deletes the file specified in `fileName$` in the currently set directory without prompting. `fileName$` may also contain a path. If `fileName$` is not a valid file name or if `fileName$` is currently open, a runtime error will occur.

Examples

```
KILL "/usr/new/file.txt"
```

MKDIR

Syntax `MKDIR path$`

| | |
|--------------------|--|
| Parameters | <code>path\$</code> is a valid string expression. |
| Description | Creates a new directory with the <code>path\$</code> specified. <code>path\$</code> may either be a relative or absolute directory name. If <code>path\$</code> is not a valid directory name, a runtime error will occur. |
| Examples | <code>MKDIR "/usr/new"</code> |

NAME

| | |
|--------------------|---|
| Syntax | <code>NAME oldName\$ AS newName\$</code> |
| Parameters | <code>oldName\$</code> and <code>newName\$</code> are both valid string expressions. |
| Description | Changes the name of the file specified with <code>oldName\$</code> to <code>newName\$</code> . Both <code>oldName\$</code> and <code>newName\$</code> have to be valid file names. If <code>oldName\$</code> is currently open, a runtime error will occur. |
| Example | <code>NAME "/usr/new/file.txt" AS "/usr/new/file_2.txt"</code> |

RMDIR

| | |
|--------------------|--|
| Syntax | <code>RMDIR path\$</code> |
| Parameters | <code>path\$</code> is a valid string expression. |
| Description | Deletes a directory with the <code>path\$</code> specified without prompting. <code>path\$</code> may be either a relative or an absolute directory name. If <code>path\$</code> is not a valid directory name or if the directory is not empty, a runtime error will occur. |
| Example | <code>RMDIR "/usr/new"</code> |

SHELL

| | |
|-------------------|--|
| Syntax | <code>SHELL parameter\$</code> |
| Parameters | <code>parameter\$</code> is a valid string expression. |

| | |
|--------------------|---|
| Description | Calls a command of the operating system, specified in <code>parameter\$</code> . <code>parameter\$</code> may contain all legal commands of the operating system including the pipe ('>') symbol. Thus, output generated by a command of the operating system may be routed to a file and read in by a RAPID! program. If <code>parameter\$</code> does not contain a valid command of the operating system, a runtime error will occur. |
| Examples | The following one-line program reads the file list of all files contained in a specified directory and stores them in a text file. <pre>SHELL "ls /usr/new > /usr/new/ls.txt"</pre> |

TIME\$

| | |
|--------------------|---|
| Syntax | TIME\$ |
| Parameters | There are no parameters. |
| Description | Returns a string containing the system time in the following format: "hh:mm:ss", where "hh" is a positive integer, giving the current system hour (in 24 h format), "mm" give the current minutes and "ss" represent the current seconds. |
| Examples | TIME\$ would have returned "10:29:37" at the time this table was created. |

Functions for error handling

For program-internal error handling, the following functions have been implemented in RAPID!

| Function | Short description |
|-----------------|---|
| ERR | Returns the error code of the most recent error. |
| ERL | Gives the program line, in which the last error occurred. |
| ERF\$ | Returns the name of program file, in which the last error occurred. |

ERR

| | |
|--------------------|--|
| Syntax | ERR |
| Parameters | There are no parameters. |
| Description | Returns the error code of the latest error that occurred. For the meaning of the error codes, please see "Runtime errors" on page 163 for reference. |

Example In this example, the 4400 prints an error message, giving the error code, the line in which the error occurred and the name of the related RAPID! program file.

```
ON ERROR GOTO errorHandler
...
errorHandler
    PRINT "Error "; ERR; " in line ";
    ERL;" in file ";ERF$
RESUME NEXT
```

ERL

Syntax ERL

Parameters There are no parameters.

Description Returns the number of the program line, in which the last error occurred.

Example Please see example with ERR function above.

ERF\$

Syntax ERF\$

Parameters There are no parameters.

Description Returns the name of the RAPID! program file, in which the last error occurred.

Examples Please see example with ERR function above.

Mathematical functions

The RAPID! environment of the 4400 provides most of the important mathematical functions:

| Function | Short description |
|------------------------------|--|
| ABS | Returns the absolute value of a numeric expression. |
| LOG, LGT | Calculates the natural logarithm and the common logarithm, respectively. |
| RND, RANDOM- IZE | Functions for generating random numbers. |
| SIN, COS, TAN, ATAN | The most important trigonometric functions. |
| SQRT | Calculates the square root of a numeric expression. |

SGN Returns the sign of a numeric expression.

ABS

Syntax ABS (numExp)
Parameters numExp is a valid numeric expression.
Description Returns the absolute value of numExp.
If numExp is an unassigned variable, a '0' will be returned.
Examples t = ABS (-1)
Assigns '+1' to variable t.

LOG, LGT

Syntax LOG (numExp)
LGT (numExp)
Parameters numExp is a valid numeric expression.
Description LOG returns the natural logarithm of numExp while LGT returns the common logarithm of numExp.
Examples i = LOG (1000)
j = LGT (1000)
Variable i will be set to '6.907755' while j will be set to '3'.

RND, RANDOMIZE

Syntax RND
RANDOMIZE (numExp)
Parameters numExp is a valid numeric expression, interpreted as positive integer.
Description RND generates a random double floating-point number between '0' and '1'.
RANDOMIZE is used to reinitialize the random generator.
Note: RANDOMIZE (x) will always lead to the same sequence of random numbers.
Examples The following example program reinitializes the random generator and then prints ten random numbers.
RANDOMIZE 321
FOR i = 1 TO 10
PRINT RND
NEXT i

SIN, COS, TAN, ATAN

| | |
|--------------------|---|
| Syntax | SIN (numExp) COS (numExp) TAN (numExp) ATAN (numExp) |
| Parameters | numExp is a valid numeric expression. |
| Description | <p>SIN (numExp) Calculates the sine of numExp and delivers back a double floating-point number. numExp is interpreted as the radiant expression of the angle (SIN(3.14...) = 0).</p> <p>COS (numExp) Calculates the cosine of numExp and delivers back a double floating-point number. numExp is interpreted as the radiant expression of the angle (COS(3.14...) = -1).</p> <p>TAN (numExp) Calculates the tangent of numExp and delivers back a double floating-point number. numExp is interpreted as the radiant expression of the angle (TAN(3.14...) = 0).</p> <p>ATAN (numExp) Calculates the arcus tangent of numExp and delivers back a double floating-point number. numExp is interpreted as the radiant expression of the angle (ATAN(3.14...) = 1.262627).</p> |
| Examples | Please see examples nested in the description above. |

SQRT

| | |
|--------------------|--|
| Syntax | SQRT (numExp) |
| Parameters | numExp is a valid numeric expression. |
| Description | Returns the square root of numExp as a double floating-point number. |
| Examples | i = SQRT (25) Assigns '5' to variable i. |

SGN

| | |
|--------------------|--|
| Syntax | SGN(numExp) |
| Parameters | numExp is a valid numeric expression. |
| Description | Returns the sign of numExp. If numExp is equal to or greater than '0', SGN will return '+1'. If numExp is less than '0', SGN will return '-1'. |

Examples `i = SGN(234)`
 `j = SGN(-0.023)`

Variable `i` will be set to '+1', while variable `j` will be set to '-1'.

Tables

While working with RAPID!, it is quite helpful to keep the following tables at hand:

- List of all [RAPID! commands and functions](#) in alphabetic order.
- Table of all [Syntax errors](#), giving also their meaning and hints how to trace the problem.
- Table of all [Runtime errors](#), explaining their meaning and giving hints of how to solve the matter.

RAPID! commands and functions

In this table, you will find all RAPID! commands and functions in alphabetic order for a quick overview.

| Command or function | Short description |
|----------------------------|---|
| ABS | Returns the absolute value of a numeric expression. |
| ASC | Returns the ASCII code of the first character in a string. |
| ATAN | Calculates the arcus tangent of a numeric expression. |
| BIN | Converts a binary representation of a numeric expression into its decimal equivalent. |
| BIN\$ | Transforms a numeric expression into its binary representation. |
| CHAIN | Loads and executes a different RAPID! program. |
| CHDIR | Changes the current directory. |
| CHR\$ | Returns the character representing the ASCII value of an integer number as a string. |
| CINT | Rounds a numeric expression to an integer, using commercial rounding. |
| CLOCK | Returns the time since the last start of the system. |
| CLOSE | Closes a previously opened file. |
| CLS | Clears the screen. |

| | |
|---------------------------|---|
| COS | Returns the cosine of a numeric expression. |
| CURDIR\$ | Returns the currently selected directory. |
| DATE\$ | Returns a string, containing the system date. |
| DIM | Declares local variables. |
| DIR\$ | Function to read out the entries in a directory. |
| DO . . . LOOP | Loop with condition. |
| END | Ends the program. |
| EOF | Checks whether the end of a file was reached. |
| ERASE | Releases allocated memory of a variable. |
| ERF\$ | Returns the name of program file, in which the last error occurred. |
| ERL | Gives the program line, in which the last error occurred. |
| ERR | Returns the error code of the most recent error. |
| ERROR | Creates an error with a specified error number. |
| EVENTWAIT, EVENTSTATUS | Waits for an event and returns the event code. |
| FOR . . . NEXT | Loop with a specified number of cycles. |
| FREEFILE | Returns the subsequent free file handle. |
| FUNCTION | Declares a function. |
| GLOBAL | Declares global variables. |
| GOSUB . . . RETURN | Branches to the specified label and returns at the next RETURN command. |
| GOTO | Branches to the specified label. |
| HEX | Converts a hexadecimal representation of a numeric expression into its decimal equivalent. |
| HEX\$ | Transforms a numeric expression into its hexadecimal representation. |
| IF . . . THEN | Checks a condition. |
| INKEY | Checks whether a softkey was pushed and returns its code. |
| INPUT | Reads data from an open file or from an open port or waits for a user entry and stores the value entered. |
| INSTR | Searches for the occurrence of a specific sequence of characters within a string. |
| KILL | Deletes a file. |

| | |
|-----------------|--|
| LEFT\$ | Reads a specific part of a string, starting from its beginning. |
| LEN | Returns the number of characters included in a string. |
| LET | Assigns an expression to a variable. |
| LGT | Calculates the common logarithm. |
| LOCATE | Sets the cursor position on-screen for the subsequent output command. |
| LOG | Calculates the natural logarithm. |
| MID\$ | Reads a specific part of a string. |
| MKDIR | Creates a new directory. |
| NAME | Renames an existing file. |
| OCT | Converts an octal representation of a numeric expression into its decimal equivalent. |
| OCT\$ | Transforms a numeric expression into its octal representation. |
| ON ERROR | Branches to the specified label, if an error occurs during program execution. |
| OPEN | Opens a file or a port and allows for reading or writing data. |
| PRINT or OUTPUT | Writes data to an open file or port or displays an expression on-screen. |
| RANDOMIZE | Initializes the random generator. |
| REM or ` | Comment. |
| RIGHT\$ | Reads a specific part of a string, starting from its end. |
| RMDIR | Removes a directory. |
| RND | Generates a random number. |
| SELECT CASE | Selects one of several cases. |
| SGN | Returns the sign of a numeric expression. |
| SHELL | Calls the operating system shell in order to perform operating system commands (e.g. so-called pipes). |
| SIN | Calculates the sine of a numeric expression. |
| SOFTKEYS | Labels the 4400 softkeys. |
| SPACE\$ | Returns a string, filled with a specific number of blanks. |
| SQRT | Calculates the square root of a numeric expression. |

| | |
|----------|---|
| STOP | Debugger breakpoint. |
| SUB | Declares a procedure (subroutine). |
| TAN | Returns the tangent of a numeric expression. |
| TEXTATTR | Sets background color and text size, font and color. |
| TIME\$ | Returns a string containing the system time. |
| VAL | Converts a string expression into a numeric expression. |
| VAL\$ | Converts a numeric expression into a string expression. |
| VARIABLE | Prints variable contents to screen or file. |
| WAIT | Halts the program for a specified amount of time. |

Syntax errors

| Code | Message and Comment |
|------|---|
| 1 | No program file argument The argument needs to be a valid file name of an existing file. |
| 2 | Program file not found The program file was not found in the current or specified directory. |
| 3 | Expression expected A numeric expression was expected, but was not found. |
| 4 | Variable expected A valid variable name was expected, but was not found. |
| 5 | String expected In a context, where only a string is allowed, there was no string found. |
| 6 | Number expected In a context, where only a numeric parameter is allowed, there was none found. |
| 7 | String const exceeds line The string was not terminated with a quotation mark ("). |
| 8 | (B)in, (O)ct or (H)ex expected After a '&t', only a binary (B), octal (O) or hexadecimal (H) number is allowed. |
| 9 | Program line too long Either the program line is more than 254 characters long or the program line continue mark ('_') was used too often. |

- 10 EOL expected
The program line should end here. In many cases, the colon (:)
that separates two valid commands is missing.
- 11 `(` expected
An open round bracket was expected, but was not found.
- 12 `)` expected
A closed round bracket was expected, but was not found.
- 13 `,` expected
A comma was expected, but was not found.
- 14 `=` expected
A '=' sign was expected, but was not found.
- 15 `#` expected
A '#' symbol was expected, but was not found. '#'s are used
to indicate file handles.
- 20 CASE, CASE ELSE or END SELECT without SELECT
At the beginning of a SELECT block, there must be a
SELECT command.
- 21 CASE Command:
Number expected
When the SELECT block is used to decide on numbers, the
CASE statements need to be numeric as well.
- 22 CASE Command:
String expected
When the SELECT block is used to decide on strings, the
CASE statements need to be strings as well.
- 23 CASE Command:
Missing END SELECT
At the end of a SELECT block, there must be an END
SELECT statement.
- 24 CASE only at begin of line
The CASE statements are only allowed at the beginning of a
program line.
- 30 IF Command:
Missing END IF
When using a multiline IF...THEN command, an END IF
statement must be used in the last line of the block.
- 31 IF Command:
THEN expected
Every IF has to be followed by a THEN.
- 32 IF Command:
ELSEIF only at begin of line
The ELSEIF statements are only allowed at the beginning
of a program line.

- 37 DO Command:
Missing LOOP
A DO command needs always to be finished with a LOOP statement.
- 38 DO Command:
WHILE or UNTIL expected
Within a DO...LOOP command, only one of the conditions WHILE or UNTIL may be used – either with DO or with LOOP.
- 40 NEXT Variable and FOR Variable are not the same
The variable used with the FOR statement must be identical with the variable used with the NEXT statement of the same block.
- 41 FOR Command:
Missing NEXT
A FOR command block must be ended with a NEXT statement.
- 42 FOR Command:
TO expected
The FOR statement may include a range for the variable to be incremented. This range is specified with TO (e.g. FOR i = 1 TO 10).
- 46 Chain Stack overflow
The maximum depth of chained programs is 4. In case more than those 4 programs have been chained (e.g. by a chained program that itself chains other programs), this message will occur.
- 47 Chain file not found
The RAPID! program file that should be chained could not be found in the current or specified directory.
- 50 Expression: NOT, +, - are not allowed here
Signs may not be used in this context.
- 51 Expression: Command or Function is not allowed here
A command or a function is not allowed in this context.
- 52 Label expected
A symbolic label is expected (e.g. after an ON ERROR, GOTO, GOSUB or RESUME command).
- 53 Label not found
The symbolic label issued with a branch command could not be found in the program. Mostly, this is due to typing errors.
- 54 Label already exists
The symbolic label stated has already been used with a different branch command.

| | |
|--------|---|
| 58 | <p>Variable not declared In order to access an indexed variable, that one must be declared first, using the DIM command.</p> |
| 59 | <p>Variable already declared This variable was declared already, using a DIM command (happens frequently when chaining programs).</p> |
| 60 | <p>Indexvariables are not allowed here Indexed variables are not allowed in this context (probably a loop can help).</p> |
| 61 | <p>PRINT Command: ' , ' or ` ; ' expected Several expressions may be handled, using just one PRINT command. However, the single expressions need to be separated by commas (,) or semicolons (;).</p> |
| 62 | <p>Keyword not allowed here The keyword is not allowed in this context. Most of these errors are caused by using a reserved RAPID! name as a symbolic label.</p> |
| 63 | <p>Stack overflow Too many loops are interlocked. Try to finish one or more of those loops before the subsequent are started.</p> |
| 64, 65 | <p>Misplaced argument The command or argument is not allowed in this context.</p> |
| 66 | <p>Compare operand expected The program expects an operand, expressing comparison (=, <, >, <=, >=, <>)</p> |
| 70 | <p>OPEN command: FOR needs APPEND, BINARI, INPUT, OUTPUT or RANDOM When opening a file with the FOR command option, a file mode has to be specified. Currently, only the following three file modes are available: INPUT, OUTPUT or APPEND.</p> |
| 71 | <p>OPEN command: ACCESS needs READ, WRITE or READ WRITE This message should not occur as the related functionality has not been implemented yet.</p> |
| 72 | <p>OPEN command: AS expected Indicates a missing file handle.</p> |
| 73 | <p>OPEN command: SHARED, LOCK READ, LOCK WRITE or LOCK READ WRITE expected This message should not occur as the related functionality has not been implemented yet.</p> |
| 80 | <p>SUB or FUNCTION header error Please check the declaration of your procedure or function; looks like something is missing there.</p> |

| | |
|----|---|
| 81 | SUB or FUNCTION already exist There is already a procedure or function with this name. Many of those cases occur by chaining RAPID! program files. |
| 82 | SUB not declared The procedure you want to use currently has not been defined before. In order to avoid that, it is recommended to declare all procedures and functions at the beginning of the main program. |
| 83 | SUB or FUNCTION declaration misplaced A procedure or function declaration cannot be used at this place. |
| 84 | SUB END or FUNCTION END does not match to header The last program line of any procedure or function must be either an END SUB (for procedures) or an END FUNCTION (for functions). |
| 85 | SUB EXIT or FUNCTION EXIT does not match to header Any procedure must be exited with EXIT SUB. Any function must be exited with EXIT FUNCTION. |
| 86 | SUB END or FUNCTION END missing The last program line of any procedure or function must be either an END SUB (for procedures) or an END FUNCTION (for functions). |
| 89 | Too many arguments More arguments than expected were used. |
| 97 | Internal Error: Parameter A procedure or function was called, using the wrong type or the wrong number of arguments. |
| 98 | Internal Error: Command not implemented You are trying to use a command that has not been implemented yet. However, the command is a reserved RAPID! name and every attempt to use it will cause this message. |
| 99 | Internal Error: PASS2 This message indicates an internal error of the program loader. If you receive this message after a restart, please contact a Willtek support center. |

Runtime errors

| Code | Message and Comment |
|------|---------------------|
|------|---------------------|

| | |
|----------|--|
| 100 | <p>Out of memory There is no more memory available for variables. To avoid this situation, please check whether there are indexed variables that are not completely used during runtime or whether one or several indexed variables may be erased using the ERASE command.</p> |
| 101 | <p>Too many GOSUBs Overflow of the GOSUB . . . RETURN stack. To avoid this error message, reduce the number of recursive procedure calls.</p> |
| 102 | <p>RETURN without GOSUB A RETURN command was found, but the program is in no procedure currently.</p> |
| 103 | <p>VAL: Number expected The VAL function returns a numeric expression (contained in a string expression).</p> |
| 104 | <p>Division by zero This is a major problem for program-internal error-handling. To completely avoid this problem, it is recommended to check the operands of a division before the division is carried out.</p> |
| 105, 106 | <p>Index out of range The index of an indexed variable must be in the range of -32,766 to +32,766.</p> |
| 107 | <p>ON ERROR not allowed in an error handling routine Within program-internal error handling, an ONERROR command is not allowed.</p> |
| 109 | <p>Dimension Error The physical dimensions of the operands do not match (e.g. $i = 1 \text{ kHz} + 5 \text{ mW}$).</p> |
| 110 | <p>No such file or directory The file name or directory specified does not exist.</p> |
| 111 | <p>Too many open files Too many files have been opened and there is no further file handle available. To avoid this message, it is recommended to close all files as soon as they are no longer required to be open.</p> |
| 112 | <p>Permission denied There is no permission to access the specified file or directory.</p> |
| 113 | <p>Invalid access code The access code used is not valid and access is still denied.</p> |

| | |
|-----|---|
| 114 | <p>Unexpected File error Happens e.g. when you write data to a disk and remove that one during program run. However, it may also indicate problems with the disk drives or controllers. Should this error message occur repeatedly, please contact your nearest support centre.</p> |
| 115 | <p>File not open The file specified has not been opened before. Files may only be accessed after they have been opened with the OPEN command.</p> |
| 116 | <p>Bad file number The file handle used is invalid. This happens for instance, when you try to open a file with the file handle of a file that was opened before.</p> |
| 117 | <p>File already exists The file or directory specified does exist and cannot be created anew.</p> |
| 118 | <p>Filesystem is Read Only The current file system only allows to read data. This message will occur when you try to write data to a file that is marked 'Read Only'.</p> |
| 119 | <p>A component of Path is not a directory The path specified – or parts of it – do not fulfill the naming conventions for directories. Please remember that a directory name must not contain any symbols and is restricted to a maximum length of 8 characters.</p> |
| 120 | <p>Directory busy The directory specified is currently in use by another task and cannot be accessed.</p> |
| 121 | <p>Directory not empty You tried to remove a directory, but the directory is currently not empty. Before a directory may be deleted, all entries need to be deleted first.</p> |
| 122 | <p>Filename must not be empty The string specifying a file name is empty.</p> |
| 123 | <p>File number already in use The current file handle has already been used to open a file and this file is still open. Either close the open file or open the current file with a different file handle. This problem can easily be avoided if you use the FREEFILE function to assign file handles.</p> |
| 124 | <p>SUB/FUNCTION Stack overflow Overflow of the GOSUB...RETUN and FUNCTION stacks. To avoid this error message, it is recommended to keep the number of recursive calls of functions and procedures to a minimum.</p> |

| | |
|-----|--|
| 125 | <p>Open GPIB failed The GPIB port could not be opened. Please check whether all parameters used in the OPEN "GPIB0:..." command are correct.</p> |
| 126 | <p>SCPI already open The SCPI port may only be opened once.</p> |
| 127 | <p>Input Timeout on COM While waiting for an input on one of the serial ports of the 4400, a timeout occurred. This may be due to faulty configuration (either hardware or software) of the handshake signals. Please also check whether all parameters used in the OPEN "COMx:..." command are correct.</p> |
| 128 | <p>Output Timeout on COM It was not possible to transmit data over the serial ports before the time limit expired. This may be due to faulty configuration (either hardware or software) of the handshake signals. Please also check whether all parameters used in the OPEN "COMx:..." command are correct.</p> |
| 199 | <p>USER BREAK The program was terminated by the user.</p> |
| 201 | <p>Widget Error: Pos and Size must be defined All new widgets need to carry a definition of their position and size. However, widget programming has not been implemented yet.</p> |
| 202 | <p>Widget Error: Widget not found The program tried to use a widget that does not exist (anymore). The widget handle is no longer valid. However, widget programming has not been implemented yet.</p> |
| 203 | <p>Widget Error: No valid Widget Call The widget call used does not correspond to the widget types available. However, widget programming has not been implemented yet.</p> |

SCPI

5

This chapter describes the remote control capabilities of the 4400. Test results described in this chapter are as follows:

- ["Overview" on page 168,](#)
- ["What SCPI is" on page 168](#)
- ["Structure" on page 169](#)
- ["Syntax and notation" on page 169](#)
- ["SCPI and RAPID!" on page 173](#)
- ["Command subsystem overview" on page 180](#)
- ["The BS and MS parameter subsystems" on page 190](#)
- ["The communication-related subsystems" on page 182](#)
- ["SCPI command errors" on page 209](#)

Overview

SCPI (pronounced as 'skippy') is a world-wide standard. The basic idea of SCPI is to define a command language for measurement systems that is independent of the related manufacturers.

This chapter explains how SCPI works on the 4400. It is divided into six subsections:

What SCPI is – Here you will find a short definition of SCPI and some general information.

Structure – This subsection explains the common commands, the subsystems and the command tree.

Syntax and notation – The command syntax, the parameters and the syntax of queries are explained in this subsection.

SCPI and RAPID! – SCPI and RAPID! together form a powerful time-saver for extensive tests that are performed on a regular basis, like automated acceptance tests after manufacturing or regular routine checks as part of maintenance or quality assurance.

Command subsystem overview – This subsection gives a brief overview on the subsystems into which the SCPI commands are split.

The BS and MS parameter subsystems

The communication-related subsystems

SCPI command errors – Here, you will find a table with explanations of syntax errors, run-time errors, device errors or any other errors.

What SCPI is

SCPI (Standard Commands for Programmable Instruments) was introduced in 1990. It is a world-wide standard, independent of single manufacturers. The SCPI specifications define a command language for measurement systems and – in principle – are based on IEEE 488.2.

SCPI is independent of the physical transmission channel of the commands. Although this chapter refers to the GPIB as the physical interface, you can also control the 4400 via a local area network (TCP/IP Option required).

The idea behind SCPI is to shorten program development times for the automated control of test equipment and to make that program development as efficient as possible.

Of course, one of the main requirements for this goal is that the language must be understood by as many measurement devices as possible. Therefore, SCPI is pushed by the SCPI consortium. Quite a number of the main test and measurement equipment manufacturers are members of the SCPI consortium.

Implementing just standard commands on a complex communications test system like the 4400 would lead to a poor performance. Therefore, we were obliged to find a compromise between standard commands and performance. This is the reason why you will find many more SCPI commands on the 4400 than specified in the standard SCPI specification. However, all SCPI commands implemented on the 4400 follow the standard SCPI syntax and rules.

For additional details on the SCPI standard, the current version can be found on page www.scpiconsortium.org/scpistandard.htm. You can download the full SCPI specification from there free of charge (some 3.5 Mb in PDF format).

Structure

SCPI defines programming commands, program messages, return values and data formats, which are consistent for all measurement systems independent of their manufacturer and purpose.

SCPI uses a device-independent command set, the so-called **Common commands**, understood by all SCPI devices.

The 4400-specific SCPI commands are called **Compound commands** and will only be understood by the 4400 and its subsystems.

A subsystem in terms of SCPI is quite abstract: it is the set of commands implemented to perform specific tasks of the SCPI device (the 'measurement subsystem' of the 4400, for instance, is the set of commands implemented for taking all kind of measurements, while the 'configuration subsystem' is the set of all configuration commands for all areas of the 4400).

All these subsystems are using the same, SCPI-based messaging and data formats.

Any SCPI command is built in a hierarchical way – similar to how a path in a file system is built.

The single command elements are separated by colons (:).

The complete set of commands of a subsystem is called the 'command tree'.

The command trees for the SCPI command set implemented on the 4400 is available in this manual.

Syntax and notation

There are two types of SCPI commands:

- compound commands and
- common commands.

Both types of commands differ in syntax.

Compound commands

Compound commands are always referred to as **commands** throughout this chapter.

- Any compound command is built in a hierarchical way. The single command elements are separated by colons (:).
- A command usually starts with a colon (:). However, the colon **must be omitted** when the subsequent command continues on the same hierarchical level (see examples below).
- The first command element always is the name of the subsystem like `CONFigure` or `MEASure`.
- Then follows one of the commands available for that subsystem like `GSM` or `RFTX`.
- The subsequent command element(s) may now be one or more subcommand(s) and/or one or more parameter term(s) (e.g. `BS:ID:BCC 5`).
- A SCPI program line may contain more than one command. In this case, the single commands have to be separated by a semicolon (;).
- There is also a short form for every command. This is usually formed of the first four letters (`CONF` instead of `CONFigure`). When the fourth letter is a vowel (a,e,i,o,u), only the first three letters are used (`RFG` instead of `RFGenerator`).
- Throughout this manual, the short form is always written in capitals to make it easy to identify it. However, the SCPI system of the 4400 is not case-sensitive.

Examples

- The complete SCPI command to set the base station color code (i.e. the training sequence) to 5 would be: `:CONFigure:GSM:BS:ID:BCC 5`. The short form `:CONF:GSM:BS:ID:BCC 5` is completely identical.
- `:RFGenerator:STATE ON` is identical with `:RFG:STAT ON`. However, the long version of the commands is – especially in the beginning – easier to work with.
- `:RFGenerator:STATE ON; LEVEL 20` is a valid two-command statement. `STATE ON` and `LEVEL` are both commands of the `RFGenerator` subsystem. Both are working on the same hierarchical level. Therefore, the colon in front of `LEVEL 20` has to be omitted.
An alternative with the identical meaning would be: `:RFGenerator:STATE ON; :RFGenerator:LEVEL 20`. The form `:RFGenerator:STATE ON; :LEVEL 20` would be **invalid**, because `LEVEL` is no subsystem of the 4400.
- `:RFGenerator:MODulation:STATE ON` is **invalid**. `STATE ON` is no subcommand of the `:RFG:MOD` level.
- `:RFGenerator:MODulation ON; FREQUENCY 850.2` In this case, the first part up to the semicolon(;) is valid. The second part is invalid as `FREQ` is no subcommand of the `:RFG:MOD` level.
The correct command would be: `:RFGenerator:MODulation ON; :RFGenerator:FREQUENCY 850.2`.

Parameters Many commands require parameters to be specified. Those parameters are placed behind the command, separated by at least one blank (space). The following types of parameters do exist on the 4400:

- **Numeric parameters.** These are integers, floating point numbers (with a maximum of 6 decimals) or exponential numbers (see specifications IEEE 488.2 NRf format or ANSI X3.42-1975 for details).
Some numeric parameters may carry a physical unit (details can be found in subsection variables of the RAPID! chapter).
Example: `:RFG:FREQ 930.071965 MHz` makes the RF generator switch to the specified transmission frequency.

NOTE

The decimal point of floating point numbers **must be** the dot (.) within SCPI as a comma (,) will always be interpreted as the separator between two parameters (see Notes below for details).

- **Boolean parameters** are specified using the binary numbers 0 | 1, or ON | OFF instead.
- **Enumerated parameters** are strings that only may be selected from a predefined list. To distinguish them from string parameters, the enumerated parameters **must not** be put in quotation marks.
Example: `:CONFigure:GSM:TYPE GSM9001800` sets the 4400 to work as test set for GSM 900/1800.
- **String parameters** are user-defined strings.
Example: `PROGram:NAME "/rapid/examples/ms_test.rbm"` loads the specified RAPID! program file.
- **Block parameters** are a special case and will be described with the related commands.

Queries Many commands also have a query form. These queries enable you to read out the current value of parameters or the results of measurements. For a query, simply add a question mark to the command (without any spaces or other symbols in-between).
Example: `:RFG:FREQ?` returns the current transmission frequency of the RF generator.

NOTE

The result of a query is saved internally on the 4400. Details can be found in subsection ["Using queries" on page 175](#).

Common commands Common commands are defined in IEEE 488.2. They work on the device itself (and on any subsystem) and always start with an asterisk (*).
Example: `*RST` resets the 4400 and sets all system parameters to default values.

A list of all common commands can be found in subsection ["Common commands" on page 171](#).

NOTE

The SCPI system is not case-sensitive. It does not matter for the syntax whether commands are written using capital letters, lowercase letters, or a mixture of both.

However, for easy maintenance of SCPI programs, it is recommended to type in the short form of a command in capital letters (CONF) and the rest of it in lowercase letters (CONFigure).

NOTE

Some commands allow more than one parameter. In those cases, the single parameters are separated by commas (.). There **must not be** any spaces between the commas and the parameters.

Example: :CONF:GSM:BS:TCH:NCELL 63,45,39,17,23,9

NOTE

The SCPI notation of commands differs from the RAPID! notation. Please, do not confuse them as this could lead to severe program errors.

SCPI notation

The notation for SCPI commands is partly different from the [RAPID! syntax](#). Please, do not confuse them as this could lead to severe program errors.

[*item*] (identical with RAPID!) – Square brackets indicate an optional item, which can also be omitted.

Example: :MEASure[:CONTinuous]:RFTX:PPEak. Regardless whether the :CONTinuous command element is used or not, the 4400 will start taking continuous measurements of the peak phase error.

item | *item* (identical with RAPID!) – Vertical bars separate entries of a list and indicate that precisely one element out of that list **must** be used.

Example: Some commands require boolean parameters to be specified. This means

that either on (1) or off (0) needs to follow the command. This is expressed by using vertical bars: ON | OFF.

< *item* > (does not exist with RAPID!) – Pointed brackets indicate that either a parameter or a subcommand must be used in order to build a valid command.

Example: :RFGenerator:STATE <ON | OFF>. The RF generator can be set either on or off.

{ *item* } (usage different to RAPID!) – Braces stand for a parameter or a subcommand that has to be selected from a predefined table.

Example: :MEASure<:{measProp}> indicates all subcommands of the MEASure subsystem, like RFTX, RFRX or AFAN.

SCPI and RAPID!

This section is an application-oriented guide on how to use the functionality of SCPI through RAPID!.

[Executing SCPI commands](#) through RAPID!

[Reading SCPI data](#) through RAPID!

[Using queries](#)

[Event handling – registers](#)

[Programming examples](#)

Executing SCPI commands

RAPID! treats the SCPI system like a file. This means that

- the SCPI system needs to be opened first, before any communication may be established.
- to communicate between RAPID! and SCPI, the standard file-related RAPID! commands (like PRINT or INPUT) are used.

Example 1 This example opens the SCPI system as a communication port:

```
LET scpi = FREEFILE
OPEN "scpi" as #scpi
...
...
...
CLOSE #scpi
```

Example 2 After the SCPI system has been opened as a communication port, commands may be sent, using the PRINT or OUTPUT command:

```
...
PRINT #scpi, "*RST"
OUTPUT #scpi, ":CONFigure:GSM:TYPE GSM9001800"
PRINT #scpi, ":MEASure:GSM:RFTX:ALL"
...
PRINT #scpi, ":FETCh:GSM:RFTX:ALL"
INPUT #scpi, rftx_result_all$ ...
```

The first command resets the 4400 to factory defaults using SCPI common command *RST.

The subsequent command configures the 4400 as a test set for GSM 900/1800. The :MEASure:RFTX:ALL command starts an internal procedure that will continuously take all RFTX measurements in a row.

The current set of measurement results can be read out, using the INPUT command.

Example 3 Converting the string read into numeric variables.

```
...
index = 0
rftx_result_all$ = rftx_result_all$ + ","
DO

    P = INSTR(rftx_result_all$, ",")
    result(index) = VAL(LEFT$(rftx_result_all$, P -
1))
    rftx_result_all$ = MID$(rftx_result_all$, P +1)
    index = index + 1

LOOP UNTIL rftx_result_all$=""
```

This example program looks for the commas, separating the single measurement result values. Then it reads the part of the string between the commas and converts it back into a numeric variable.

More details regarding this example program can be found in section "[Standard TX measurements](#)" on page 177.

Important notes:

- Please keep in mind that RAPID! has to check the GPIB actively as there will be no automated reaction to any GPIB control sequences. However, please note that measurement results, polled through GPIB commands will also be available for any RAPID! program.
- To make sure that the 4400 is not blocked by other tasks, only perform RAPID! or SCPI measurements from the [Welcome menu](#).
- In order to prevent the 4400 from 'waiting for eternity', use a standard timeout of 10 s (allow more time for complicated measurements like RX testing).
- A timeout does not speed up things (or slow them down). A call establishment will take its time.

Reading SCPI data

To read SCPI data (such as measurement results) into RAPID!, the INPUT command is used.

Again, the SCPI system has to be opened as a communication port first.

NOTE

All results handed back from the SCPI system have the format of a string. Should there be more than one result, the single results will be separated by commas.

Example:

This example reads out the version of the 4400.

```
LET scpi = FREEFILE
OPEN "scpi" as #scpi
PRINT #scpi, "*IDN?"
INPUT #scpi, version$
CLOSE #scpi
```

SCPI common command `*IDN?` is used to read out the identification of the test set. The value returned is then stored in RAPID! string variable `version$`. The content of `version$` will be something like "Willtek,4400,0511099".

Using queries Many commands have a query form. These queries enable you to read out data from the SCPI system like measurement results or the current value of parameters (like the current transmission frequency of the RF generator).

Building queries For a query, simply add a question mark to the command (without any spaces or other symbols in-between).

Example 1

The example below initiates the measurement of the burst length and returns the related value.

```
...
OUTPUT #scpi, ":MEASure:RFTX:LENGth?"
INPUT #scpi, burstlength$
```

Example 2

More than one query can be placed into a single command line. The individual queries must be separated by semicolons (;). The returned values will also be separated by semicolons.

The example below reads out the current system time and then queries the contents of the event status register.

```
...
OUTPUT #scpi, ":SYSTem:TIME?; :*ESR?"
INPUT #scpi, result$
```

The returned result string could be "17,40,55;4", for example. This would denote a time of 5:40 PM and an event status register value of 4.

Example 3

In case, a query needs a parameter to be specified, this parameter is then placed behind the question mark. As with normal commands, there must be at least one blank between the question mark and the parameter:

```
...
OUTPUT #scpi, ":MEASure:GSM:ARRay:RFRX:BER:CIA? 2"
...
```

This command will perform the BER measurement on the class 1a bits twice and return both measurement results in a string. The string returned in this example is "2.1,2.2".

Event handling – registers

Basically, measurements can be started and results can be read out using the MEASure and FETCh subsystems of the 4400. However, system events (like errors) could occur while those measurements are in progress and the measurement results read in could thus be invalid. Therefore, the 4400 is equipped with an event handling system. This system may be programmed by the user to raise an event on certain conditions. Using the programming features, the SCPI system of the 4400 will for instance generate an event in case an error occurs during the performance of the current command. Before the result string of a measurement is read out, it is a good idea to check the internal event registers of the 4400 first. Most frequently, this check is a look at the test set's service register. Bit 2 of this register will be set as soon as there is an error message in the error queue.

NOTE

To avoid confusion between the registers of the STATus subsystem and the much more general SCPI statusbyte, the latter is called the service register throughout this manual.

The programming of the event handling system is described in detail in subsection ["Understanding the STATus subsystem" on page 183](#). The use of the service register is outlined in subsection ["At the top: the service register" on page 185](#).

The scheme to use the registers for communication between a RAPID! program or the GPIB on one side and the 4400 on the other side always follows the following principles:

| Step | RAPID! | GPIB |
|---------------------------------------|-------------------------------------|-------------------------|
| Preparing service register | :PROGram:SRE <i1> | *SRE <i1> |
| Set up a loop that waits for an event | DO..."read service register"...LOOP | "SRQ" |
| Read service register | :PROGram:STB? | serial poll or *STB? |

Section ["Programming examples" on page 176](#) shows some practical examples for the basic use of the register communication between SCPI and RAPID!.

Programming examples

In this subsection, you will find some application-oriented examples for RAPID! programs using SCPI on the 4400. These examples concentrate on the following topics:

- ["Standard TX measurements" on page 177](#)
- ["Message exchange" on page 178](#)
- ["Example of a GPIB protocol" on page 179](#)

Standard TX measurements

The example below illustrates how to perform a standard TX test using RAPID! and SCPI.

```

`FILE: RFTXDEMO.RBS
`DESCRIPTION: RAPID! program that measures
`the RMS phase error, the frequency,
`peak power and also checks whether the shape
`of the burst is within the PTT.

`Definition of variables
DIM result(5)

`Opening SCPI as communication port
LET scpi = freefile
OPEN "scpi" as #scpi

`Configuring the measurements as above as a group and
`starting a group measurement
PRINT #scpi, ":CONfigure:MEASure:GROup:RFTX
PRMS,FREQuency,POWer,TEMPlate"
PRINT #scpi, ":MEASure:RFTX:GROup?"

`Read out the result string
INPUT #scpi, result$

`Read out the service register to check if there was
`some error
PRINT #scpi, ":PROG:STB?"
INPUT #scpi, A$
A$ = BIN$(VAL(A$),8)

`Select reaction on event occurred
IF (MID$(A$,2,1) = "1" THEN
`Some error occurred
PRINT #scpi, ":SYST:ERR?"
INPUT #scpi, Err$
GOTO FAIL_EXIT
ELSE

`Measurements have been completed without errors
GOTO PRINT_RESULTS
END IF

`Result procedure of the program
PRINT_RESULTS:
`Converting the result string back into
`four result values
index = 0
result$ = result$ + ","
DO

```

```

' Find the position within result$, where the first
' comma appears
P = INSTR(result$, ",")
' Read out the part of result$ in front of the first
' comma and convert it into a numeric value
result(index) = VAL(LEFT$(result$, P - 1))
' Cut off the value just read from result$
result$ = MID$(result$, P + 1)
index = index + 1
LOOP UNTIL result$=""

IF result(3) = 0 THEN
A$ = "PASS"
ELSE
A$ = "FAIL"
END IF

'Print results
PRINT "RMS phase deviation :"; result(0)
PRINT "Frequency :"; result(1)
PRINT "Peak power :"; result(2)
PRINT "Burst shape:"; A$
GOTO OKAY_EXIT

'Entry point of any error handling routine
FAIL_EXIT:
PRINT "An error did occur during measurements :", Err$

'Ending the program
OKAY_EXIT:
END

```

Message exchange

This example demonstrates a message exchange between a RAPID! program and an external control computer (connected on the GPIB). The message exchange is triggered by events via the `SYSTEM:MESSAge` and the `PROGram:MESSAge` queue. Although not complete, the example below illustrates the principle.

NOTE

Only the core program code has been set in `courier` to make it easy to identify it. All comments are in 'normal' letters.

```

'FILE: MSGDEMO.RBS
'DESCRIPTION: RAPID! program to demonstrate message exchange between a
' RAPID! program and the controlling computer.

...

'Setting the mask for the service register: rising SRQ on an entry of a message
' into the message queue.
'Bits to set: SRQ (bit 6 = 64) and system.message (bit 0 = 1)
PRINT #0, "*SRE 65"

```

'Setting the mask for the program's service register: rising an event on an entry
' of a message. Bits to set: event (bit 6 = 64) and program.message (bit 0 = 1)

```
PRINT #0,":PROG:SRE 65"

DO

  ` Read out programm message queue
PRINT #0,":PROG:MESS?"
INPUT #0,message$

  `Check message string for key letters
result$ = MID$(message$,1,2)
SELECT CASE result$
CASE "TX"
result$ = DoTXAlign(Message$)
CASE "RX"
result$ = DoRXAlign(Message$)
CASE "SP"
result$ = DoSPAlign(Message$)
CASE "AN"
result$ = DoAFANAlign(Message$)
CASE ELSE
result$=""
END SELECT

IF result$="" THEN PRINT #0,":SYST:MESS ";result$

LOOP
```

Example of a GPIB protocol

| to/ from 4400 | Data | Comment |
|---------------------|---|---|
| to | :PROG:NAME MSG- DEMO.RBS | Load program |
| to | :PROG:STAT RUN | Start program on 4400 |
| to | :PROG:MESS "TX, 1, 62, 124, -105" | Start (user-programmed) TX cali- bration process on 4400 |
| from | SRQ | 4400 raises service request after calibration |
| to | serial poll | Controller identifies SRQ device |
| from | SYST:MESS? | Controller queries system message |
| to | "TX Calibration OK" | 4400 confirms successful TX cali- bration |
| from | :PROG:MESS "RX, 1, 62, 124, 15" | Start (user-programmed) RX cali- bration process on 4400 |

| to/ from 4400 | Data | Comment |
|---------------------|-------------------------------------|---|
| | serial poll | Controller identifies SRQ device |
| to | SYST:MESS? | Controller queries system message |
| from | "RX Calibration OK" | 4400 confirms successful RX calibration |
| to | :MEASure:GSM:RFTX:ALL | Start relevant RFTX measurements |
| | serial poll | Controller identifies SRQ device |
| to | :CALC:GSM:RFTX:ALL:LIM:FAIL? | Controller queries the result of the limit check of the measurement result values |
| from | 0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 | 4400 delivers results |

Command subsystem overview

This subsection provides information about the SCPI command subsystems.

[The Measurement subsystems](#) – MEASure, FETCh, CALCulate

[Measurement device configuration subsystems](#) – RFGenerator, RFSPectrum, RFANalyser, AFGenerator, AFANalyser, MS Power Supply

[The BS and MS parameter subsystems](#) – CONFigure, CALL

[The communication-related subsystems](#) – PROGram, STATus, SYSTem, FORMat

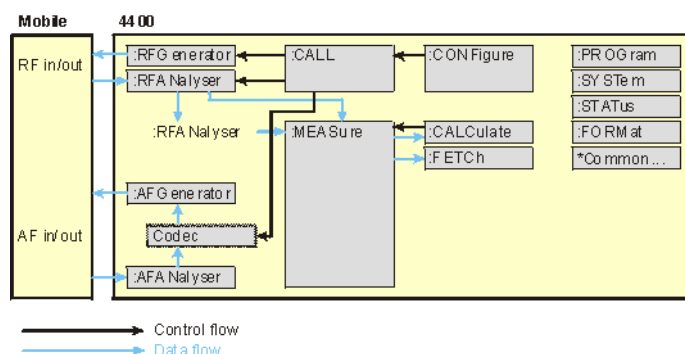
[SCPI command errors](#)

Using the SCPI commands

The various SCPI commands and their arguments/parameters are described below. Please note that any SCPI command specified with an invalid or without an argument required will be totally ignored by the system. This means that an incomplete SCPI command will not affect the current system status at all.

Schematic view of the subsystems of the 4400

This is a very simplified schematic view of the control and data flow inside the 4400.



Common commands

The common commands are understood by all SCPI and IEEE 488.2 instruments. Their purpose is to perform general tasks and to read or work with the registers common to all SCPI instruments. The following common commands have been implemented.

| Command | Short description | Command group |
|---------|--|-------------------------|
| : *CLS | Resets the entire status reporting system. | General common commands |
| : *ESE | Sets and queries the event status register mask. | Event status register |
| : *ESR? | Reads out the current contents of the event status register. | Event status register |
| : *IDN? | Returns device identity. | General common commands |
| : *OPC | Waits until previous command has been completed. | General common commands |
| : *RST | Resets the entire test set. All parameters, limits etc. will be set to internally predefined default values. | General common commands |
| : *SRE | Sets and queries service register mask. | Service register |
| : *STB? | Returns the current contents of the service register and clears the service register. | Service register |
| : *WAI | Waits until previous command has been completed. | General common commands |

The communication-related subsystems

These subsystems cover system relevant tasks, which are not primarily involved in the measurement process.

SYSTEM – System parameters, such as the number of unread error messages, the test set's GPIB address etc. may be read out or set using the commands of this subsystem.

PROGRAM – This subsystem deals with all activities related to RAPID! programs.

STATUS – The STATUS subsystem controls and provides information on the state of the 4400. There are two types of states: operational states describe what is currently going on within the test set while questionable states deliver questionable states of the 4400.

FORMAT – The commands of this subsystem enable settings of the data output format in remote mode.

The SYSTEM subsystem

System parameters, such as the number of unread error messages, the test set's GPIB address etc. may be read out or set using the commands of this subsystem.

The STATUS subsystem

The STATUS subsystem delivers detailed information about the internal status of the 4400, its error conditions and special events. These three areas are dealt with in three different sections of the status subsystem. These three sections are addressed, using different commands.

| Status Area | Related commands | Main functional aspects |
|------------------|------------------------|--|
| Operation Status | STATUS:OPERation... | These commands deal with the operation status of the 4400. They describe what is currently going on inside the test set, mainly in respect of signaling and measuring. |
| System Errors | STATUS:QUESTionable... | This area of the internal status report system mainly deals with errors and warnings regarding the hardware stages of the 4400 (like 'RF input overload' or 'frequency out of range'). |
| Execution Errors | *ESE, *ESR? | Mainly program or SCPI command execution errors are dealt with in this area of the internal status report system. |

The status subsystem provides in-depth information about the internal status of the test set. Furthermore, powerful event processing tools allow any form of flexible control over the 4400.

However, the use of the status subsystem is a bit tricky because of the many parameters involved. Therefore, we kindly suggest to carefully read subsection ["Understanding the STATus subsystem" on page 183](#) before using the status subsystem.

A table of all registers implemented can be found in subsection ["Table of registers" on page 186](#).

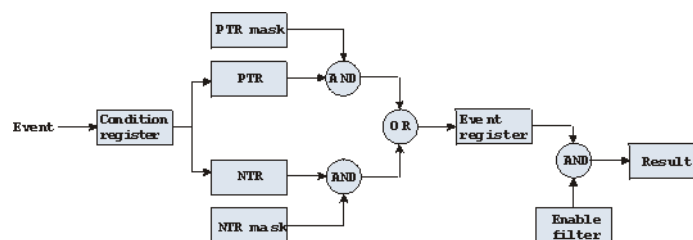
Understanding the STATus subsystem

The STATus subsystem is intended to deal with special events occurring inside the test set. It provides tools to enable any current condition to raise a system event. These system events may then be used to trigger service requests (SRQ) on GPIB, or to trigger RAPID! programs.

There are several **groups of registers**, structured in a hierarchical order. Lower-level registers work on specific conditions and single parameters while higher-level registers integrate the result of several lower-level registers and thus provide a more general view.

The highest level of these registers is the **service register**, sometimes also referred to as the **statusbyte register** or **status byte**.

The figure below gives an example of a group of registers.



A group of registers starts with a **Condition register**. An internal Event will set the corresponding bit of the Condition register (to 'set a bit' means a transition from binary 0 to a binary 1). Any Condition register will be updated continuously. This means that a bit will be reset as soon as the condition which rose that bit is no longer valid.

Example:

As soon as the 4400 starts to page a mobile under test, a certain bit of a certain condition register will be set. When the 4400 stops paging the mobile (because e.g. it responded to the paging requests), this bit will be reset. Now, there is no chance to find any evidence for a paging process in this Condition register.

Therefore, there is an **Event register** included in every group of registers. In the Event register, a bit will remain set even when the condition for it to be set is no longer valid. Any Event register, however, will be cleared after a query.

Example:

Continuing our example from above, in the related Event register, the corresponding bit would still be set. A query of this register would provide evidence that there has been some paging in progress. However, the Event register is not able to provide any information if the condition is still valid.

Summary 1: condition-type versus event-type registers

- The condition-type registers reflect the current status of the test set and is updated continuously. When you would like to know if a special condition is currently valid, then read out the related condition-type register with a query.
- The condition-type register and the event-type register have an identical structure. This means that they are of the same length and the single indicator bits are at the same positions.
- The event-type registers are the 'memory' of the status system. Once a bit has been set, it will remain set until the event-type register is read out with a query. When you want to trigger e.g. a RAPID! program with a certain event, always read out the related event-type register.
- Event-type registers are read-only and self-destructive. They will be cleared after any query.

Transition of a bit from the condition-type to the event-type register

How will a bit find its way to the Event register?

This depends on the transition filter and its settings. The transition filter works as follows:

First, there are two branches: the Positive TRansition and the Negative TRansition filter. Both only react on the corresponding transitions of bits and both contain as many bit positions as the condition register.

PTR will carry a binary 1 at a bit position only when the corresponding bit of the condition register is set, while NTR will carry a binary 1 at a bit position only when the corresponding bit of the condition register is reset.

Both the output of PTR and NTR will be combined with the corresponding mask, using a logical AND operation.

These masks are user-definable (using the **STAT:xxxx:xTR** commands) and again contain as many bit positions as the condition register.

Example:

The only chance for bit 4 (that has just been set in the condition register) to reach the Event register is that the PTR mask carries a binary 1 at bit position 4. The logical AND between the PTR filter and the PTR mask will then deliver a binary 1. This binary 1 will pass the logical OR and thus set bit 4 of the Event register.

Summary 2: from the condition-type to the event-type registers

- There are two detectors for every bit of a condition register: the **positive** transition and the **negative** transition branch. Positive transition means a change of a bit from a binary 0 to a binary 1 while a negative transition is a bit change from a binary 1 to a 0.
- To every branch, there is a filter mask (PTR and NTR mask). This filter mask is user-definable.
- The filter mask and the result of the transition filter are combined using a logical AND operation.
- The results of the AND operation in both branches will be combined, using a logical OR operation.
- The results of the OR operation are the contents of the related event-type register.

Moving up: an event reaches the condition register one hierarchical level up

The **Event register** contains summary bits, corresponding to the bits of the condition register. A summary bit will be set, when the initial event passes the transition filter.

The **Enable filter** is a mask to filter events that are allowed to move one level up. The Enable filter is user-definable (using the **STAT:xxxx:ENABLE** commands) and again contains as many bit positions as the condition-type register.

Again, the Enable filter mask is ANDed with the Event register and a nonzero result will finally set the **Result bit**. This Result bit may be a summary bit either in a higher-level register, or in the service register.

Summary 3: raising the Result bit

- The event-type register will be ANDed with the related Enable filter mask.
- The Enable filter mask is user-definable.
- The result of the logical AND operation between event-type register and related Enable filter will be the Result bit.
- The result bit will set the corresponding bit of the condition register one level up.

At the top: the service register

The **service register** contains eight summary bits: three for the status groups available on the 4400, two for internal queue handling, two for remote control and one bit for it's own status: the **summary status bit**.

When an event passes through and sets one of the seven corresponding summary bits of the service register, the contents of the service register will be ANDed with the service register mask. This mask can be set using the ***SRE** common command.

When the logical AND operation of the current contents of the service register and the service register mask leads to a binary 1, then the summary status bit will be set as well.

If both bit 6 of the service register mask and bit 6 (the summary status bit) of the service register are set, then a service request (SRQ) is executed.

Table of registers The STATUS subsystem uses and/or provides access the following registers:

Service register

This register represents the highest level within the report structure of the 4400. The service register contains eight bits. A detailed description of the service register can be found in the appendix (SCPI Command Reference).

Event status register group

This group of registers collects all general events of the 4400 (mostly command errors).

Depending of the setting of the event status register mask, bits set in the event status register may be transferred to bit 5 of the service register.

For further details regarding the event status register, please refer to the appendix (SCPI Command Reference).

General operation register group

This group of registers is 16 bits wide and reflects the general operation status of the 4400.

Some of the bits of this register (like bits 8 and 9) are summary bits. Those summary bits are result bits of subordinate groups of registers as described below.

NOTE

The commands related to the general operation register group and its subordinate groups of registers all start with **:STATUS:OPERation:**.

| Bit | Decimal | Meaning |
|-----|---------|--|
| 0 | 1 | Set during calibration. |
| 1 | 2 | Set while settling. |
| 2 | 4 | Set while ranging. |
| 3 | 8 | Set while sweeping. |
| 4 | 16 | Set as long as any measurement is being carried through. |
| 5 | 32 | Set while the 4400 waits for a trigger. |
| 6 | 64 | Set while the 4400 waits for an arm. |
| 7 | 128 | Set while a correction process is in progress. |
| 8 | 256 | This is the SIGNaling summary bit. This is the Result bit of the signaling operation register group (see below for details). |
| 9 | 512 | This is the MEASure summary bit. This is the Result bit of the measuring operation register group (see below for details). |

| | | |
|----|-------|-------------|
| 10 | 1024 | Not in use. |
| 11 | 2048 | Not in use. |
| 12 | 4096 | Not in use. |
| 13 | 8192 | Instrument. |
| 14 | 16384 | Not in use. |
| 15 | 32768 | Not in use. |

Signaling operation register group

This group of registers is 16 bits wide. Its main task is to deal with events related to the signaling status. The signaling operations depend on the system option and the contents of the signaling operation register group are detailed in the appendix.

The Result bit of this group of registers is forwarded to bit 8 of the general operation register group.

Measuring operation register group

This group of registers is 16 bits wide. Its main task is to deal with events related to the measurement status.

The Result bit of this group of registers is forwarded to bit 9 of the general operation register group.

| Bit | Decimal | Meaning |
|-----|---------|--|
| 0 | 1 | Set while RFTX measurements are in progress. |
| 1 | 2 | Set while RFRX (BER) measurements are in progress. |
| 2 | 4 | Set while RFSpectrum measurements are in progress. |
| 3 | 8 | Set while audio measurements are in progress. |
| 4 | 16 | Not in use. |
| 5 | 32 | Not in use. |
| 6 | 64 | Not in use. |
| 7 | 128 | Not in use. |
| 8 | 256 | Not in use. |
| 9 | 512 | Not in use. |
| 10 | 1024 | Not in use. |
| 11 | 2048 | Not in use. |
| 12 | 4096 | Not in use. |
| 13 | 8192 | Not in use. |
| 14 | 16384 | Not in use. |
| 15 | 32768 | Not in use. |

General questionable status register group

This group of registers is 16 bits wide and reflects the general questionable status of the 4400. The events taken care of this group of registers are mainly errors and warnings.

Some of the bits of this register (like bits 9,10 and 11) are summary bits. Those summary bits are result bits of subordinate groups of registers and described below.

NOTE

The commands related to the general questionable status register group and its subordinate groups of registers all start with `:STATus:QUESTionable:`.

| Bit | Decimal | Meaning |
|-----|---------|--|
| 0 | 1 | Voltage out of range. Not used on the 4400. |
| 1 | 2 | Current out of range. Not used on the 4400. |
| 2 | 4 | Time out of range. Not used on the 4400. |
| 3 | 8 | Power out of range. Not used on the 4400. |
| 4 | 16 | Temperature out of range. Not used on the 4400. |
| 5 | 32 | Frequency out of range. Not used on the 4400. |
| 6 | 64 | Phase out of range. Not used on the 4400. |
| 7 | 128 | Modulation out of range. Not used on the 4400. |
| 8 | 256 | Calibration out of range. |
| 9 | 512 | This is the RF summary bit. This is the Result bit of the RF questionable status register group (see below for details). |
| 10 | 1024 | This is the SYNChronization summary bit. This is the Result bit of the SNYChronization questionable status register group (see below for details). |
| 11 | 2048 | This is the AUDio summary bit. This is the Result bit of the AUDio questionable status register group (see below for details). |
| 12 | 4096 | Not in use. |
| 13 | 8192 | General warning, concerning the test set. |
| 14 | 16384 | Command not understood warning. |
| 15 | 32768 | Not in use. |

RF questionable status register group

This group of registers is 16 bits wide. Its main task is to deal with warnings and errors regarding the RF stages of the 4400.

The Result bit of this group of registers is forwarded to bit 9 of the general questionable status register group.

| Bit | Decimal | Meaning |
|--------|---------|---|
| 0 | 1 | Input overload. Reduce RF power immediately to avoid possible damage of the 4400's highly sensitive input stages! |
| 1 | 2 | Output level out of range. |
| 2 | 4 | Transmission frequency out of range. |
| 3 | 8 | Reception frequency out of range. |
| 4...15 | | Not in use. |

SYNChronization questionable status register group

This group of registers is 16 bits wide. Its main task is to deal with warnings and errors regarding the external synchronization of the 4400. The Result bit of this group of registers is forwarded to bit 10 of the general questionable status register group.

| Bit | Decimal | Meaning |
|--------|---------|--|
| 0 | 1 | Set when an external RF synchronization signal is recognized on the EXT SNYC prog. socket. |
| 1 | 2 | Set when an external frame synchronization signal is recognized on the SNYC IN/OUT socket. |
| 2...15 | | Not in use. |

AUDio questionable status register group

This group of registers is 16 bits wide. Its main task is to deal with warnings and errors regarding the audio stages of the 4400. The Result bit of this group of registers is forwarded to bit 11 of the general questionable status register group.

| Bit | Decimal | Meaning |
|--------|---------|---|
| 0 | 1 | Input overload. Reduce signal level immediately to avoid possible damage of the 4400's highly sensitive input stages! |
| 1 | 2 | Output level out of range. |
| 2...15 | | Not in use. |

The PROGRAM subsystem – overview

This subsystem contains commands related to loading and executing RAPID! program files.

The FORMat subsystem – overview

The FORMat subsystem sets and queries settings concerning the data output in remote mode.

The BS and MS parameter subsystems

These subsystems allow access to the base station parameters (i.e. the 4400 simulating a base station) and to the information received from the mobile under test like the measurement report. The commands of these subsystems are described here.

CONFigure – This subsystem incorporates all changeable BS parameters of all implemented communication systems.

NOTE

The settings made here directly affect all communication system-specific subsystems.

CALL – This subsystem handles call procedures and allows to read out the measurement report generated by the mobile under test. The information available is dependent on the current state of a call, i.e. some commands require an established radio communication link between the 4400 and the mobile under test.

The CONFigure subsystem

This subsystem incorporates all changeable BS parameters of all systems implemented. The key commands are the following:

| | |
|--|---|
| CONFigure:CSYSstem | Selects the communications system to work with. |
| CONFigure:<SystemOption>:... | These commands select parameters within a communications system. |
| CONFigure:<SystemOption>:BS:... | These are the commands to set specific system parameters like the base station's RF output power level or its identity. |
| CONFigure:<SystemOption>:MSTation:... | The mobile-specific information is handed over to the 4400 using these commands. One example is the power level. |
| CONFigure:<SystemOption>:BER:... | These commands set the BER parameters. |
| CONFigure:<SystemOption>:GROup:... | With the help of these commands, groups of measurements may be defined. |
| CONFigure:COUPloss:... | These commands provide access to the coupling loss compensation feature of the 4400. |

- The CALL subsystem** This subsystem contains commands
- for call setup and handling procedures and
 - to read out the measurement report, generated by the mobile.

The Measurement subsystems

The 4400 provides the following measurement subsystems:

The MEASure subsystem – This subsystem provides the commands for all kinds of measurements: Single-shot as well as series of measurements, measurements of single parameters as well as of groups of parameters.

The FETCh subsystem – To read out the latest measurement result of a specific parameter or a group of parameters.

NOTE

FETCh requires that a measurement is started first, using the MEASure commands.

NOTE

FETCh will neither start nor terminate continuous measurements.

The CALCulate Subsystem – All kinds of statistic evaluations and checks of measurement results against predefined limits.

The MEASure subsystem

The MEASure subsystem is probably the most important SCPI command subsystem of the 4400. There you will find all commands to acquire measurement results of the mobile under test. Measurements can be taken as one-shot measurements or as series of measurements:

| Type of measurement | Related command element | Short description |
|---------------------|-------------------------|--|
| One-shot | [:CONTinuous] | Actually :CONT starts a measurement, that will be performed continuously. Single measurement results can be read out using the related :FETCh commands. Thus several measurements can be started (where those measurements are started first that take the longest time, like e.g. BER). |

Series :ARRay

This command element offers the possibility to carry through a specific measurement a certain number of times. All the single measurement result values can be read out with just one command. Using this feature, measurement results returned can then be used e.g. for statistic data evaluation on an external computer.

Example:

First a BER measurement command is issued to start a BER measurement (because it takes some time). Then several RF TX measurements are performed and the measurement results are read out and used for statistic data evaluation. After that the result of the BER measurement is read out.

Important notes:

- A newly issued RF TX MEASure command will terminate any RF TX or RF spectrum measurements currently in progress.
- A newly issued RF SPectrum MEASure command will terminate any RF TX or RF spectrum measurements currently in progress.
- A newly issued RF RX MEASure command will terminate any RF RX measurements currently in progress.
- A newly issued AF MEASure command will terminate any RF TX, RF RX, RF spectrum or AF measurements currently in progress.
- Measurements are always started, using the current system parameters and the current state of the 4400.
- In case the 4400 is in a state that does not allow a specific measurement command to be performed or completed, an error message will be added to the 4400's internal error queue. The related flag in [The STATus subsystem](#) will be set, too.
In case this is the first error message to appear within the error queue, the error indicator bit of the service register (bit 2) will be set as well.

:MEASure[:CONTinuous] MEASure commands that are stated with or without the optional :CONTinuous command element will make the 4400 perform the related measurement for an unlimited number of times. The measurement will only be terminated if a new MEASure command of the same or related type is issued (see **Notes** above).

| | |
|-------------------|--|
| Syntax | :MEASure [:CONTinuous] < { :measProp } > [?] |
| Parameters | The one-shot measurements do not require any parameters. |

| | |
|--------------------|--|
| Description | <p>Starts the (continuous) measurement of the specified measurable property. The <code>CONTinuous</code> command element is optional. Unless a measurement result is read out (using the <code>FETCh</code> subsystem), no measurement result values will be given back. The measurement result(s) of any measurement will be stored internally. Any previously stored result will be overwritten as soon as a new measurement result has been achieved. The latest result measured may be read out using the <code>FETCh</code> subsystem. Any <code>FETCh</code> command will wait for a measurement result value(s) to be available. In case there is more than one measurement result value, the single measurement result values are separated by commas (like e.g. "50.5,3.46"). Should the <code>FETCh</code> command fail to obtain a measurement result value (because e.g. the current state of the 4400 does not allow the measurement to be performed or completed), a timeout will occur and an error message will be added to the 4400's internal error queue. The related flag in the <code>STATus</code> subsystem will be set, too. In case this is the first error message to appear within the error queue, the error indicator bit of the service register (bit 2) will be set as well. The main application of the combination of the <code>MEAS</code> and the <code>FETC</code> subsystems: Starting a measurement that takes some time to deliver a measurement result back (e.g. the BER measurement. After the measurement has been started with the <code>MEAS</code> command, the test set 'is free' to perform other tasks. If in this case the query form of the <code>MEAS</code> command is used, the test set is blocked until a measurement result is available. Sometimes, measurement results will need to 'sweep in'. In this case, the first measurement result might be totally misleading. Using the <code>MEAS</code> command will allow the measurement result value to stabilize on a meaningful result before the latest result value is read out using the <code>FETC</code> subsystem.</p> |
| Query | <p>The query form of any <code>MEASure</code> command will start the (continuous) measurement of the specified measurable property. The <code>CONTinuous</code> command element is optional. After the first measurement has been completed, the measurement result value(s) will be delivered back in a string as outlined above. Should the query fail to obtain a measurement result value, a timeout will occur and an error message will be added to the 4400's internal error queue as explained above. The main application of the query form of a <code>MEAS</code> command is speed. When combined with 'fast' measurements (like e.g. the fast power level measurement), the query form of a <code>MEAS</code> command delivers measurement results as fast as possible. Note: The measurement started with the query form of the <code>MEAS</code> command will be continued in the background. Any further measurement result values may be read out, using the appropriate <code>FETCh</code> command.</p> |

Examples

```
:MEASure:GSM:CONTinuous:RFTX:PPEAk and
:MEASure:GSM:RFTX:PPEAk
```

are identical. Either command will start the measurement of the peak phase error. The latest result of this measurement (like "5.84") will be stored internally. It will be overwritten as soon as a new measurement result has been achieved.

To read out the current measurement result, use the FETCh subsystem:

```
:FETCh:GSM:RFTX:PPEAk?
:MEASure:RFTX:ALL
```

This command will start the continuous measurement of all relevant RFTX parameters. After this command has been issued, you may continue with e.g. an RF RX MEASure command.

All 19 single measurement results will be stored internally. As soon as a new measurement result has been achieved, the previous value will be overwritten.

To read out the measurement results achieved by the :MEAS:RFTX:ALL command, use the :FETCh:GSM:RFTX:ALL? command.

```
:MEASure:GSM:RFTX:ALL?
```

As in the example above, the continuous measurement of all relevant RFTX parameters will be started. Unlike the example above, this command will wait until all 19 measurement results have been achieved and will return all of them in a string, separated by commas. The measurements will continue and later results may be read out, using the FETCh:GSM:RFTX:ALL? command.

How to convert a result string back into numeric variables

The example program below illustrates how the returned string can be converted back into numeric variables in a RAPID! program.

```
PRINT #scpi, ":MEAS:GSM:RFTX:ALL?"
INPUT #scpi, result$
index = 0
result$ = result$ + ","
```

```
DO
```

```
  ' Find the position within result$, where the first
  ' comma appears
```

```
  P = INSTR(result$, ",")
```

```
  ' Read out the part of result$ in front of the first
  ' comma and convert it into a numeric value
```

```
  result(index) = VAL(LEFT$(result$, P - 1))
```

```
  ' Cut off the value just read from result$
```

```
  result$ = MID$(result$, P + 1)
```

```
  index = index + 1
```

```
LOOP UNTIL result$=""
```

MEASure:ARRay

The :ARRay command element makes the 4400 perform any measurement property a user-definable number of times. All measurement result values obtained during the process will be stored in an internal array and can be read out using the related commands of the FETCh subsystem or will be returned in case the measurement process has been started using the query format of the command.

After the specified number of measurements have been performed, the measurement will be stopped and no further measurement result values will be stored

internally. Therefore, any attempt to read out data again (unless any measurement has been started before) will result in a timeout and thus in an error message.

This is one of the main differences between the `[:CONTinuous]` and the `:ARRay` command element.

Note: Any measurement will be terminated if a new `MEASure` command of the same or related type is issued (see **Notes** in section [“The MEASure subsystem” on page 191](#) for details).

| | |
|--------------------|---|
| Syntax | <code>MEASure:ARRay<{:measProp}>[?] <numMeas></code> |
| Parameters | <code>numMeas</code> is the number of measurements to be performed. |
| Description | <p>Takes a <code>numMeas</code> number of measurements of the specified type <code>measProp</code>. The results of the single measurements will be stored in an internal array. The measurement results array can be read out using the related command of the <code>FETCh</code> subsystem (see examples below for reference). After an array has been read out using the related <code>FETCh</code> command, the internal array will be cleared. Any subsequent <code>FETCh</code> command trying to read out the same measurement result array will not be able to read any measurement results and thus result in a timeout. In this case, an error message will be added to the 4400's internal error queue (for further details, refer to section “:MEASure[:CONTinuous]” on page 192).</p> |
| Query | <p>The query form of any <code>MEAS:ARR</code> command will start the related measurement the specified number of times. After the number of measurements specified have been performed, the measurement will be stopped. The query will then return a string containing all the measurement results. The single result values will be separated by commas. Any subsequent <code>FETCh</code> command trying to read out the same measurement result array will not be able to read any measurement results and thus result in a timeout. In this case, an error message will be added to the 4400's internal error queue (for further details, refer to section “:MEASure[:CONTinuous]” on page 192).</p> |
| Examples | <pre> :MEASure:GSM:ARRay:RFTX:PPEAK 10 This command will make the 4400 perform 10 independent measurements of the maximum phase error in GSM. After those 10 results have been achieved, the measurement will be stopped. The ten result values will be stored in an internal array. To read out the measurement result array, use the FETCh subsystem: :FETCh:GSM:RFTX:PPEAK? will return the 10 values in one string (like "5.42,5.44,5.80,...5.72,5.64") Any subsequent :FETCh:GSM:RFTX:PPEAK? command will result in a timeout. MEASure:GSM:ARRay:RFTX:ALL? 2 This command takes all relevant RFTX measurements twice. The measurements will be stopped as soon as the 2 x 19 result values are available. The 38 result values will be returned as a string; the single values will be separated by commas. Any subsequent :FETCh:GSM:RFTX:PPEAK? command will result in a timeout. </pre> |

How to convert a result string back into numeric variables

The example program below illustrates how the returned string can be converted back into numeric variables in a RAPID! program.

```

PRINT #scpi, ":MEAS:GSM:ARR:RFTX:ALL? 2"
INPUT #scpi, result$
index = 0
result$ = result$ + ","

DO

  ' Find the position within result$, where the first
  comma appears
  P = INSTR(result$, ",")
  ' Read out the part of result$ in front of the first
  comma and convert it into a numeric value
  result(index) = VAL(LEFT$(result$, P - 1))
  ' Cut off the value just read from result$
  result$ = MID$(result$, P + 1)
  index = index + 1

LOOP UNTIL result$=""

```

:MEAS[:CONT]:BLOCKdata

The `BLOCKdata` command element of this subsystem is used to read out all the single measurement results necessary to generate the following result graphics:

- shape of the TX burst (burst received by the 4400 from the mobile)
- phase error
- modulation spectrum of burst received by the 4400 from the mobile
- AF spectrum (the audio spectrum)

:MEASure:BLOCKdata:BURStshape

This command starts a continuous measurement of the mobile's burst. The corresponding result string (to be read out either with the query form of the command or with `:FETCh:BLOCKdata:BURStshape` will contain 711 floating point real numbers, representing the measurement result values. The numbers will only have one digit behind the dot. The single measurement result values are separated by commas.

The meaning of the single values within the result string is as follows:

- The first two values are necessary for the scaling of the data. All remaining 709 values represent the measurement results.
- The first value (index = 0) gives the offset of the middle of the burst (bit 73) in respect to the measurement result array.
Example: An offset of 353.0 means that the middle of the burst will be the 355th value within the array (or the value contained at index 354).
- The second value (index = 1) gives the peak power level of the burst in dBm.
- All following values (indices 2 to 710) give the RF power levels of the related positions.

The resolution on the time axis is approx. 0.875 microseconds (i.e. the distance in time between two consecutive measurement points).

:MEASure:BLOCKdata:PHASerror

First, a continuous measurement of the mobile's phase error will be started. The corresponding result string (to be read out either with the query form of the command or with `:FETCh:BLOCKdata:PHASerror` will contain 711 floating point numbers, representing the measurement result values. The single measurement result values are separated by commas. The meaning of the single values within the result string is as follows:

- The first two values are necessary for the scaling of the data. All remaining 709 values represent the measurement results.
- The first value (index = 0) gives the offset of the middle of the burst (bit 73) in respect to the measurement result array.
Example An offset of `353.0` means that the middle of the burst will be the 355th value within the array (or the value contained at index 354).
- The second value (index = 1) is always set to `0.0`.
- All following values (indices 2 to 710) give the phase error of the related positions.

The time resolution is approx. 0.25 bit period (i.e. four measurement results per bit period).

:MEAS:BLOC:MSP[:CURRent]

This command starts a continuous measurement of the modulation spectrum of the mobile's burst.

The corresponding result string (to be read out either with the query form of the command or with `FETCh:BLOCKdata:MSpectrum` will contain floating point real numbers, representing the measurement result values. The single measurement result values are separated by commas.

The number of the measurement result values returned is dependent on the span and resolution, set with the commands `:RFSPpectrum:MSpectrum:SPAN` and `:RFSPpectrum:MSpectrum:RESolution`.

:MEASure:BLOCKdata:MSpectrum:AVG

This command starts the measurement of the modulation spectrum of the mobile's burst for a user-definable number of times and averages the measurement results.

The corresponding result string (to be read out either with the query form of the command or with `FETCh:BLOCKdata:MSpectrum` will contain floating point real numbers, representing the measurement result values. The single measurement result values are separated by commas.

The number of the measurement result values returned is dependent on the span and resolution, set with the commands `:RFSPpectrum:MSpectrum:SPAN` and `:RFSPpectrum:MSpectrum:RESolution`.

:MEASure:BLOCKdata:AFSPpectrum

This command starts a continuous measurement of the audio spectrum. The corresponding result string (to be read out either with the query form of the command or with `FETCh:BLOCKdata:AFSPpectrum` will contain floating point real numbers, representing the measurement result values. The single measurement result values are separated by commas. The number of the measurement result values returned is dependent on the span and resolution.

NOTE

Audio measurements can only be performed on the 4400 when the Audio Option is installed.

:MEASure:....:GROup

As mentioned before (see Notes in section "The MEASure subsystem" on page 191), a new measurement will always terminate a preceding one of the same or related type.

Therefore, the GROup command element has been implemented in the SCPI command set of the 4400. This command element allows to specify a user-definable list of measurements that can then be started with one command. The measurement results can be read out using the query form of this command – or with the related command of the FETCh subsystem..

NOTE

In this respect, :MEAS:RFTX:ALL can be regarded as a predefined 'group', containing all important RFTX measurements.

NOTE

The AFANalyser subsystem also allows to define 'groups'. However, all commands of the AFANalyser subsystem will only obtain measurement results if the Audio Option has been installed.

| | |
|--------------------|--|
| Syntax | <pre>:CONFigure:MEASure:GROup[:RFTX] <{RFTXprop}> MEASure[:CONTInuous]:RFTX:GROup or MEASure:ARRay:RFTX:GROup or :CONFigure:MEASure:GROup:AFANalyser <{AFANprop}> MEASure[:CONTInuous]:AFANalyser:GROup or MEASure:ARRay:AFANalyser:GROup</pre> |
| Parameters | <p><{RFTXprop}> is one or more of the single RFTX measurements PPEAK, PRMS, FREQuency, LENGth, UTIme, POWer, TEMPlate, CORNer, FLATness</p> <p><{AFANprop}> is one or more of the single AF Analyser measurements SINad, DISTortion, FREQuency, ACV:PEAKp, ACV:RMS, DCV:RMS</p> |
| Description | <p>Starting a 'group' measurement will take all measurements specified just with one single command.</p> <p>After the group command has been completed, all 'group' measurement results are available at the same time and can be read out using either the query form of the command or the related commands of the FETCh subsystem.</p> |
| Example | <pre>:CONF:GSM:MEAS:GRO:RFTX PPEAK, FREQuency, POWer, LENGth :MEAS:RFTX:GRO</pre> <p>This sequence of commands first defines a group of RFTX measurements and then issues a group command. As soon as all the measurements specified in the group command have been completed, the measurement result values can be read out using the :FETCh:GSM:RFTX:GRO? command.</p> |

The FETCh subsystem

The FETCh subsystem enables you to read out the currently valid measurement result value(s) of a measurement.

Important notes

- Before a measurement value may be read out with commands of the FETCh subsystem, a MEASure command has to be issued first.
- After a continuous measurement has been started, the latest measurement result value can be obtained using the related **:FETCh** command.
- In case an array measurement has been started, the related **:FETCh** command will return the entire measurement result array.
- If there are no measurement results to be read out by a **FETCh** command for any reason, a timeout will occur. The wait time until a timeout occurs is dependent on the type of measurement to be performed (see below).
- If the preceding **MEASure** command and the **FETCh** command do not match, a timeout will occur.
- When you use the query form of any **MEASure** command, all measurement results obtained will be handed back and the internal result register will be cleared afterwards. Consequently, a subsequent **FETCh** command will lead to a timeout (as above).
- The following timeouts have been implemented:
 - 5 s for all RFTX measurements
 - 30 s for all RFRX measurements
 - 10 s for all RFSpectrum measurements
 - 10 s for all AF measurements

There are two versions of a FETCh command:

- The **:FETCh:LAST?** command will read out the latest result of the last MEASurement command issued – whatever command that was. Using this command, please keep in mind that your control program then has to take care of the number and format of the measurement result values returned.
- The **:FETCh:{measProp};** commands will read out the latest result of the measurement specified with **{measProp}**.

FETCh:LAST

The **:FETCh:LAST?** command will read out the latest result of the last MEASurement command issued – whatever command that was. Using this command, please keep in mind that your control program then has to take care of the number and format of the measurement result values returned. To convert a result string back into single measurement result values, please refer to section ["How to convert a result string back into numeric variables" on page 194](#).

| | |
|--------------------|--|
| Syntax | FETCh : LAST? |
| Description | Returns a string, containing the latest measurement result(s) for the MEASure command issued last. Format and number of the measurement result values contained in the string are depending on the preceding MEASure command. |
| Examples | <p>MEASure : GSM : RFTX : PPEAk FETCh : LAST? will return the latest result (e.g. 5 . 84).</p> <p>MEASure : GSM : ARRay : RFTX : PPEAk 5 FETCh : LAST? will return the entire array (e.g. 5 . 84 , 5 . 81 , 5 . 94 , 5 . 74 , 5 . 79).</p> <p>MEASure : GSM : ARRay : RFTX : PPEAk? 5 FETCh : LAST? In this case, a timeout will occur as the :MEAS query will return the entire array and there will be no measurement result values for the :FETCh command to collect.</p> |

FETCh:BLockdata:...? The commands with the **:BLockdata** command element are used to read out all the single measurement results necessary to generate the following result graphics:

- shape of the TX burst (burst received by the 4400 from the mobile)
- phase error
- modulation spectrum of burst received by the 4400 from the mobile)
- AF spectrum (the audio spectrum)

FETCh<{:measProp}> The **:FETCh<{:measProp}>** commands will read out the latest result of the measurement specified with **measProp**.

| | |
|--------------------|--|
| Syntax | FETCh<{:measProp}>? |
| Description | Returns a string, containing the last measurement result(s) obtained for measProp . |

Examples

MEASure : GSM : RFTX : ALL

The result of this command will be that the 4400 continuously measures all RFTX parameters.

To read out the 19 measurement result values, use the

FETCh : GSM : RFTX : ALL?

command. It will deliver back all measurement results contained in a string. This command may be used repeatedly (as long as the corresponding **MEASure** command was not terminated by another **MEASure** command).

MEASure : GSM : ARRay : RFTX : PPEAk 10

This command will start the measurement of the peak phase error. As soon as 10 measurement results have been obtained, the measurements will be stopped.

To read out the 10 measurement results, use the

FETCh : GSM : RFTX : PPEAk?

The string delivered back contains the 10 measurement results, starting from the first one (the oldest).

MEASure : GSM : ARRay : RFTX : PPEAk? 10

Unlike the example above, this command will start the measurement of the peak phase error. As soon as 10 measurement results have been obtained, those results will be given back in a string and the measurements will be stopped.

Any

FETCh : GSM : RFTX : PPEAk?

command will lead to a timeout as there are no new measurement results to be collected.

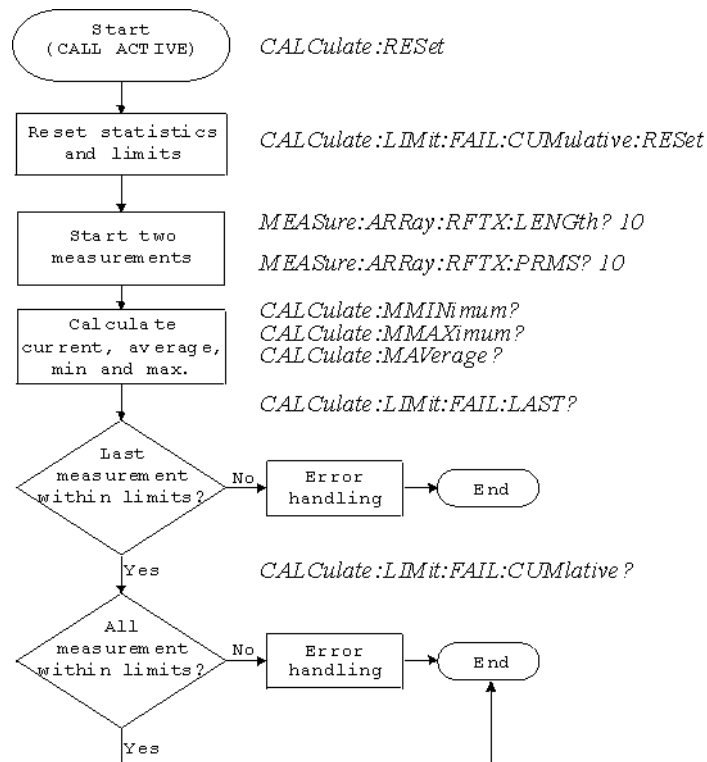
The CALCulate Subsystem

The CALCulate subsystem performs statistic evaluation of measurement results and also allows to check results against user-definable limits.

The basic scheme outlined below gives an idea of how to work with the CALCulate subsystem and to use the single queries as branching decisions within a program flow.

NOTE

Statistic evaluations on measurable properties like average, minimum or maximum will only be performed on the type of measurement started last (i.e. the **PRMS** measurement in the example below).



Reading the basic scheme

- First, the limits of the measurements to be performed are set (not shown in the basic scheme).
- After a call has been established, the CALCulate subsystem is reset using the **:CALCulate:RESet** command. Only measurements started after this command will be taken into account for the statistic evaluation. Any results from previously started (and still running) measurements will be ignored as well as all existing measurement results.
- The CALCulate subsystem of the 4400 allows to perform a cumulative check of the measurement results of a measurement. This means that all measurement results obtained (since the last reset of this part of the subsystem) will be checked against the corresponding limits. This cumulative check is reset as well in the example shown above with the **:CALCulate:LIMit:FAIL:CUMulative:RESet** command. For further details, check with section **"CALCulate:LIMit"** on page 203.
- Then, two measurements are started in this example (commands **:MEASure:ARRay:RFTX:LENGth? 10** and **:MEASure:ARRay:RFTX:PRMS? 10**).
- As soon as the measurements have been completed, the CALCulate subsystem is used to identify the minimum and maximum measurement result value and to calculate the average measurement result value. Please note that these commands will work only on the results of the measurement started last (the RMS-valued phase error in this example).
- The command **:CALCulate:LIMit:FAIL:LAST?** will return a boolean number, indicating whether the last measurement of the RMS-valued phase error was within the limits set (then a 0 will be returned) or whether it was off the limits (then a 1 will be returned).

- The last command (**:CALCulate:LIMit:FAIL:CUMulative?**) is similar to the one above, with the main difference that this query will tell whether **all** measurements of the RMS-valued phase error taken since the last reset of the cumulative check are within the limits set (then a **0** will be returned) or whether at least one measurement result value did violate at least one of the user-definable limits (in this case, the query will deliver back a **1**).

CALCulate:LIMit The **CALCulate** commands that incorporate the **LIMit** command element

- check whether one or more measurement result(s) did violate the user-definable limits
- reset the cumulative limit evaluation system
- switch the limit evaluation system for specific measurements on or off
- set the limits

FAIL[:LAST]? - did the latest measurement result value fail?

| | |
|--------------------|--|
| Syntax | :CALCulate:LIMit:FAIL[:LAST]? |
| Returns | 0 when the limits were not violated or 1 in case at least one limit was violated by the latest measurement result value. |
| Description | Checks whether the latest measurement result value of the measurement started last is within its limits. |
| Examples | <p>:MEAS:GSM:ARR:RFTX:LENG :MEAS:GSM:ARR:RFTX:PRMS :CALC:GSM:LIM:FAIL:LAST?</p> <p>This command of the CALCulate subsystem will check whether the latest measurement result value of the PRMS measurement is within its limits.</p> <p>:MEAS:GSM:ARR:RFTX:LENG 10 :CALC:GSM:LIM:FAIL:LAST?</p> <p>In this example, the :CALC command will check if all 10 measurement result values of the burst length measurement array are within their limits.</p> |

{measProp}:LIMit[:FAIL] - did the latest measurement result value of a specific measurement fail?

| | |
|--------------------|--|
| Syntax | :CALCulate:{measProp}:LIMit[:FAIL]? |
| Returns | 0 when the limits were not violated or 1 in case at least one limit was violated by the latest measurement result value. |
| Description | <p>Checks whether measurement result values of the measurement specified with the {measProp} command element are within their limits.</p> <p>In case, this command is used during a continuous measurement, only the latest measurement result will be checked.</p> <p>When this command is used subsequent to an array measurement, all measurement result values of the array will be checked against the limits. This means that a 1 will be returned if a single measurement result of an array is off the limits.</p> |

| | |
|----------------|---|
| Example | <pre> :MEAS:GSM:ARR:RFTX:LENG 10 :MEAS:GSM:ARR:RFTX:PRMS 10 :CALC:GSM:RFTX:LENG:LIM:FAIL? </pre> <p>In this example, the <code>:CALC</code> command will check whether all 10 measurement result values of the <code>LENG</code> measurement array are within their limits.</p> |
|----------------|---|

FAIL:CUMulative? – did any measurement result value fail?

| | |
|--------------------|---|
| Syntax | CALCulate:LIMit:FAIL:CUMulative? |
| Returns | 0 when the limits were not violated or 1 in case at least one limit was violated by at least one measurement result value. |
| Description | Checks whether all measurement result values of the measurement started last are within their limits. |
| Example | <pre> :MEAS:GSM:ARR:RFTX:LENG 10 :MEAS:GSM:ARR:RFTX:PRMS 10 :CALC:GSM:LIM:FAIL:CUM? </pre> <p>This command of the <code>CALCulate</code> subsystem will check whether all measurement result values of the <code>PRMS</code> measurement are within their limits.</p> |

FAIL:CUM:RESet – resets the cumulative limit evaluation

| | |
|--------------------|---|
| Syntax | CALCulate:LIMit:FAIL:CUMulative:RESet |
| Description | Resets (clears) the cumulative check of measurement result values against their limits. Only the measurement results from <code>:MEAS:...</code> commands issued subsequent to this reset command will be taken into account for any limit checks. |

{measProp}:LIMit:STATe – switches the limit check for a specific measurement on or off

| | |
|--------------------|--|
| Syntax | CALCulate:{measProp}:LIMit:STATe |
| Description | Switches the limit check for a measurement specified with the <code>{measProp}</code> command element either ON or OFF . |
| Example | <pre> :CALC:RFTX:ALL:LIM:STAT OFF </pre> <p>Switches the limit check for the 19 main RF TX measurements off.</p> |

{measProp}:LIMit:UPPer[:DATA] – sets the upper limit for a specific measurement

| | |
|--------------------|--|
| Syntax | CALCulate:{measProp}:LIMit:UPPer[:DATA] |
| Description | Sets the upper limit for the limit check of the measurement specified with the <code>{measProp}</code> ; command element. The upper limit is the maximum measurement result allowed. Any measurement result value exceeding the value set with this command will result in a violation of the upper limit. |

| | |
|----------------|--|
| Example | <p>:CALC:GSM:RFTX:PRMS:LIM:UPP 10.0 This command sets the maximum RMS-valued phase error allowed to 10.0. Any measurement result value exceeding this limit (e.g. 10.01) will result in a violation of the limits of the RMS-valued phase error measurement.</p> |
|----------------|--|

:{measProp}:LIMit:LOWer[:DATA] – sets the lower limit for a specific measurement

| | |
|--------------------|--|
| Syntax | CALCulate:{measProp}:LIMit:LOWer[:DATA] |
| Description | Sets the lower limit for the limit check of the measurement specified with the {measProp}; command element. The lower limit is the minimum measurement result allowed. Any measurement result value falling below the value set with this command will result in a violation of the lower limit. |
| Example | <p>:CALC:AFAN:SIN:LIM:LOW 25.5 This command sets the minimum SINAD required in order to pass the test to 25.5. Any measurement result value falling below this limit (e.g. 25.4) will result in a violation of the limits of the RMS-valued phase error measurement.</p> |

CALCulate:{Statistics} On the results of the last measurement performed, statistic test evaluation can be used. The related commands are described in this section.

NOTE

The commands described in this section will be reset with any new **MEASure** command. Therefore, the commands described below will only deliver the statistic evaluation of the measurement started last.

| | |
|--------------------|---|
| Syntax | CALCulate:{Statistics}? |
| Parameters | {Statistics} is one command element out of the following list: MAverage MMAximum MMINimum |
| Returns | A floating point real number representing the result of the related statistic evaluation, performed on all available results of the measurement started last. |
| Description | Calculates and returns the specified statistical property referring to the most recent measurement. |
| Examples | <p>MEASure:GSM:ARRay:RFTX:ALL 10 MEASure:GSM:ARRay:RFTX:PPEAK 10 CALCulate:GSM:MMINimum? The first command takes 10 measurements of all RFTX parameters (190 then all together); the second one takes ten measurements of the peak phase error. The CALC:MMIN? command will only deliver back the minimum result of the peak phase error measurement as this was the measurement started last. The value delivered back in the result string is "5.05" in this example.</p> |

Using limits The question 'do the performance characteristics of a mobile stay within the limits set by the appropriate specifications' is the core question of all GSM testing. The **LIMit** subsystem of the 4400 offers a broad range of:

- defining single or complex limits, using the **[:DATA]** command element of the **CALCulate** subsystem and
- reading out the results of the limit checks, using the **:LIMit [:FAIL] ?** query.

The limit evaluation system can be switched on or off for every single measurement parameter using the **:LIMit :STATe** command element.

[:DATA] – customize or query limits

| | |
|--------------------|--|
| Syntax | <code>CALCulate<{measProp}>:LIMit<{limType}>[:DATA] <Val></code> |
| Parameters | limType is a placeholder for :UPPer or :LOWer . Val is the numeric value (floating) for the limit of the specific measurement parameter. |
| Description | Sets or queries the limit value(s) for the limit identified with limType . The limit evaluation will check the measurement results obtained against the limits set, using this command. Note: Some measurement types have more complex limits (such as the power/time template or the modulation spectrum). Please find a detailed explanation on those complex limits in subsection “Working with complex limits” on page 207 . |
| Examples | :CALCulate:GSM:RFTX:PRMS:LIMit:UPPer:DATA 4.00 Sets the upper limit of the RMS phase error to 4.00 . :CALC:GSM:RFTX:PPEA:LIM:UPP? Queries the currently set upper result limit for the peak phase error. The value will be returned as floating in the result string like 6.35 . |

[:FAIL] ? – pass/fail result query

| | |
|--------------------|---|
| Syntax | <code>CALCulate<{measProp}>:LIMit[:FAIL]?</code> |
| Returns | A boolean value or array (dependent on the type of measurement as defined by measProp). When all measurements of a type are within the limits, a 0 (pass) will be returned. If at least one measurement result is beyond the limits, a 1 (fail) will be returned. |
| Description | Checks whether any of the current measurement(s) failed to meet the limits. The type of measurement is defined by measProp . |
| Examples | MEASure:GSM:ARRay:RFTX:POWer 10 Starts a series of 10 measurements of the RF peak power. CALCulate:GSM:RFTX:POWer:LIMit:FAIL? This command reads out the result of the 10 measurements checked against the limit. When all 10 measurement results are within the limits, a single 0 will be returned. If one or more measurement results are beyond the limits, a single 1 will be returned. |

:STATe – switch limit evaluation on/off

| | |
|-------------------|---|
| Syntax | <code>CALCulate<{measProp}>:LIMit:STATe <limState></code> |
| Parameters | limState is either ON or OFF . |

| | |
|--------------------|---|
| Description | <p>Activates or deactivates the check of the measurement results against the limit of the measurement type defined by measProp.</p> <p>Note: When the limit evaluation has been switched off, a :FAIL query will only return 0(s) (pass).</p> |
| Example | <p>CALCulate:GSM:RFTX:PRMS:LIMit:STATE OFF This command switches the limit evaluation of the RMS phase error off.</p> <p>MEASure:GSM:RFTX:PRMS Starts a series of RMS phase error measurements.</p> <p>CALCulate:GSM:RFTX:PRMS:LIMit:FAIL? This query of the pass/fail evaluation will always return 0 (pass) as the limit evaluation of this parameter has been switched off.</p> |

Working with complex limits

It is not possible to define all relevant limits by just one number. Some limits are quite complex, like the power/time template, the corner points or the modulation spectrum.

Limits for the power/time template

The limits for the power/time template are made up by a total of 16 vectors; 9 for the upper limits and 7 for the lower limits.

These vectors have the following format: x,y.

- where x is the time in microseconds in relation to the beginning of the burst (i.e. the first bit of the useful part)
- and y is the RF power level in dB(c) in relation to the nominal output power level of the mobile.

Examples:

```
CALC:RFTX:TEMP:LIM:UPP -42,-47,-28,-47,-18,-28,-10,
-4,0,4, 552.8,1,560.8,-4,570.8,-28,580,-47
```

```
CALC:RFTX:TEMP:LIM:LOW 0,-150,0,-150,0,-40,20,
-1,270,-1,543,-1,543,-150
```

Limits for the corner points

The corner points are a maximum of eight positions to check critical parts of the burst. For each position, a minimum and a maximum RF power level may be specified.

- Positions are specified in microseconds in relation to the beginning of the burst (i.e. the first bit of the useful part)
- minimum and maximum RF power levels are specified in dB(c) in relation to the nominal output power level of the mobile.

Examples:

```
CALC:RFTX:CORN:POS -28,-18,-10,0,542.8,
552.8,560.8,570.8
```

```
CALC:RFTX:CORN:LIM:LOW -150,-150,-150,-150,
-150,-150,-150,-150
```

```
CALC:RFTX:CORN:LIM:UPP 4,4,4,4,4,4,4,4
```

Limits for the modulation spectrum

In case of the modulation spectrum, 23 positions (i.e. frequencies) have been predefined (see subsection Generator/Analyser for details).
With the related commands, the upper and lower limits for those 23 positions may be set. Those limits are specified in dB.

Measurement device configuration subsystems

These subsystems provide commands for setting and reading out the states of the various measurement devices of the 4400.
The following subsystems are available and described in this subsection.

RF measurement devices

RFGenerator – The radio frequency generator provides both continuous signals and bursts according to the specifications of the system (e.g. GSM) currently set. The RFGenerator subsystem controls the accessible parameters of the RF generator, such as RF level, frequency etc.

RFAnalyser – This subsystem gives access to the setup parameters of the RF Analyser, such as frequency or trigger mode.

RFSPpectrum – The commands of this subsystem are used to set span and resolution of the RF spectrum analyser.

AF measurement devices

AFGenerator – The audio frequency generator provides signals for audio measurements.
The AFGenerator subsystem controls the accessible parameters of the AF generator, such as output level, frequency, multitone signals etc.

AFAnalyzer – The AF analyzer measures audio parameters. The related subsystem gives access to the settings.

MS Power Supply – The MS power supply option is an (optional) external device that simulates the power supply (battery) of a mobile under test. Together with the 4400, it provides a lot of interesting performance data of the mobile station. The related subsystem gives access to all relevant settings.

The RFGenerator subsystem

The RFG subsystem controls the accessible parameters of the RF generator.

Important notes:

- The RF generator can only be used if all communication systems have been switched off (and unloaded) before.

- The RF generator functionality of the 4400 will enable you to even provide a base channel to allow the mobile under test to synchronize to the base station. However, as long as the RF generator is active, there will be **no call setup and no reaction to signaling**.
Some of the data transmitted by the 4400 in the base channel can be set or altered using the SCPI commands described in section ["The CONFigure subsystem" on page 190](#).
- A good way to set up the 4400 as a RF generator for **circuit-switched standard GSM** signals is the use of the `:CONFigure:CSYSstem GCGenana` command.
While working in this mode, a base channel will **not** be provided.
- To set up the 4400 as a RF generator for **both packet-switched and circuit-switched GSM** signals, use the `:CONFigure:CSYSstem GPGenana` command instead.
In this mode, a base channel can be provided (see command `:RFG:GSM:MOD:CHAN` for details).

The RFANalyser subsystem The RFAN subsystem controls the accessible parameters of the RF analyzer.

The RFSPectrum subsystem The RFSP subsystem controls the accessible parameters of the RF modulation spectrum analyzer.

The AFGenerator subsystem The AFG subsystem controls the accessible parameters of the audio generator. Please note that all commands of this subsystem require the Audio Option to be installed on your 4400.

The AFANalyser subsystem The AFAN subsystem controls the accessible parameters of the AF analyzer. Please note that all commands of this subsystem require the Audio Option to be installed on your 4400.

The MS Power Supply subsystem The PSUPply subsystem controls the accessible parameters of the MS Power Supply Option.

SCPI command errors

This subsection contains a table of SCPI command errors.

NOTE

If a query for an error code returns `0` then no error did occur.

| Error Code | Meaning |
|-------------------|---|
| 61 | QNX semaphore error. The operating system of the 4400 encountered a flag communication error. |
| 62 | TMSG QNX send error. |
| 63 | TMSG QNX sync error. |
| 64 | Internal communication error. |
| 65 | Unknown message received. |
| 66 | GPIB cannot be initialized. |
| 67 | This command is invalid. |
| 68 | Internal error of the task state system. |
| 69 | Error of the GPIB system. |
| 70 | QNX proxy error. |
| 71 | Process coordination error. |
| 72 | Message sent to the GPIB system is not understood there. |
| 75 | Error within the error message system. |
| 78 | Language expression invalid. |
| 80 | Name attachment error. |
| 81 | Proxy attachment error. |
| 82 | Proxy detach error. |
| 83 | Timer attachment error. |
| 85 | Timer delete error. |
| 86 | Parameter can not be set. |
| 87 | INI file error. |
| 88 | The file selected was not found. |
| 89 | DSP setup error. |
| 100 | Command error. |
| 101 | Invalid character. |
| 102 | Syntax error. |
| 103 | Invalid separator. |
| 104 | Data type error. |
| 108 | Parameter not allowed. |
| 109 | Parameter missing. |
| 111 | Header separator error. |
| 112 | Program mnemonic too long. |
| 113 | Undefined header. |

| | |
|-----|--|
| 114 | Header suffix out of range. |
| 121 | Invalid character within a number. |
| 123 | Exponent too large. |
| 128 | Numeric data not allowed in this context. |
| 131 | Invalid suffix. |
| 134 | Suffix too long. |
| 138 | No suffix allowed in this context. |
| 141 | Invalid character data. |
| 144 | Character data too long. |
| 158 | No string allowed in this context. |
| 160 | Block data error. |
| 168 | No block data allowed in this context. |
| 200 | General execution error. |
| 201 | Multislot not active. |
| 202 | The external synchronization frequency is not within the ranges specified. |
| 203 | External synchronization changed during remote operation. |
| 204 | The operation is not possible in the current state of the 4400. |
| 221 | Settings of the 4400 lead to a conflict. |
| 222 | Data out of range. |
| 225 | No communication system running. |
| 226 | Timeout occurred while waiting for an uplink message to arrive from the mobile. |
| 227 | Layers 2/3 failed. Communication could not be established due to problems on the layer 2 and/or layer 3 level. |
| 228 | Mobile can not work in the enhanced frequency range. |
| 229 | No call release while an SMS is in progress. |
| 230 | Generator/Analyzer not running. |
| 231 | System running – no system expected. |
| 250 | Mass storage error. |
| 253 | Corrupt media. |
| 256 | File name not found. |
| 272 | Macro execution error. |
| 280 | Program error. |
| 310 | System error. |

| | |
|-----|--|
| 320 | Save/recall memory lost. |
| 330 | Function not supported. |
| 350 | Queue overflow. |
| 362 | There needs to be an active call in order to start the codec option. |
| 364 | No audio hardware. The audio option would be required to complete a command, but it is not installed. |
| 365 | No codec hardware. The codec option would be required to complete a command, but it is not installed. |
| 370 | No results available. |
| 371 | Fetch: timeout occurred. |
| 372 | Fetch: no BER synchronization. |
| 373 | Fetch: arb data. |
| 374 | Measurement task error. |
| 375 | Error in burst data encountered. |
| 376 | ACPM receive error. |
| 377 | Autotemplate error. |
| 378 | Setting value for modulation spectrum out of range. |
| 399 | Invalid error code. |
| 401 | General CDMA measurement error: Invalid return code for sample acquisition. |
| 402 | CDMA Measurement error: Signal level too high for current input attenuation. |
| 403 | CDMA Measurement error: Signal level dropped below valid level during acquisition. |
| 407 | CDMA Measurement error: Sample acquisition RAM failure. |
| 408 | CDMA Measurement error: Signal level below minimum accuracy specification. |
| 411 | CDMA Measurement error: DSP EEPROM error. Default correction data loaded, measurement accuracy not verified. |
| 413 | CDMA Measurement error: synchronization to mobile signal failed. |
| 415 | CDMA Measurement error: Cannot trigger on mobile signal. |

SCPI Command Reference

A

This appendix describes the commands for test automation using RAPID!, the General Purpose Interface Bus (GPIB) or a TCP/IP connection. The SCPI commands are divided into the following subsystems:

- ["Signaling operation register group" on page 214](#)
- ["Common commands" on page 214](#)
- ["SYSTEM subsystem" on page 219](#)
- ["STATus subsystem" on page 227](#)
- ["PROGram subsystem" on page 239](#)
- ["FORMat subsystem" on page 242](#)
- ["CONFigure subsystem" on page 244](#)
- ["MEASure subsystem" on page 250](#)
- ["FETCh Subsystem" on page 274](#)
- ["CALCulate Subsystem" on page 286](#)
- ["RFANalyser subsystem" on page 337](#)
- ["RFSpectrum subsystem" on page 338](#)
- ["AFGenerator subsystem" on page 340](#)
- ["AFANalyser subsystem" on page 344](#)
- ["MS Power Supply subsystem" on page 349](#)

Signaling operation register group

This group of registers is 16 bits wide. Its main task is to deal with events related to the signaling status. The signaling operations depend on the system option. The Result bit of this group of registers is forwarded to bit 8 of the general operation register group.

Common commands

The common commands are understood by all SCPI and IEEE 488.2 instruments. Their purpose is to perform general tasks and to read or work with the registers common to all SCPI instruments. The following common commands have been implemented.

General common commands *CLS

| | |
|--------------------|--|
| Syntax | : *CLS |
| Parameters | There are no parameters. |
| Description | Resets the entire status reporting system: <ul style="list-style-type: none">- The service register will be cleared (all bits will be set to 0).- The event status register will be cleared (all bits will be set to 0).- The error message queue will be emptied.- All event-type registers will be cleared. |
| Query | There is no query form of this command available. |

*IDN?

| | |
|--------------------|--|
| Syntax | : *IDN? |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | Returns a string, containing the following information: <ul style="list-style-type: none">- manufacturer's name- name of the device- serial number- software revision number All parameters are separated by commas. Note: In times of company mergers and acquisitions, it is a good idea to check the name of the device, not the manufacturer's name which may change between software updates. This does not preclude any name changes at Willtek but rather applies to instrumentation in general. |
| Example | "WILLTEK, 4400, 0511099, 3.10.0001" |

***OPC**

| | |
|--------------------|--|
| Syntax | :*OPC |
| Parameters | There are no parameters. |
| Description | Postpones the execution of a command until all commands issued previously have been completed. |
| Query | Returns the 'operation complete' flag in a string. A 1 indicates that all commands have been completed while a 0 means that there is at least one command still under execution. |

***RST?**

| | |
|--------------------|---|
| Syntax | :*RST? |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | Resets the entire test set. All parameters, limits etc. will be set to the internally pre-defined default values. |

***WAI**

| | |
|--------------------|--|
| Syntax | :*WAI |
| Parameters | There are no parameters. |
| Description | Postpones the execution of a command until all commands issued previously have been completed. |
| Query | There is no query form of this command available. |

Commands affecting the event status register

The event status register contains eight bits. The meaning of these bits is outlined in the table below.
The commands working on the event status register are described below the table.

| Bit | Decimal | Meaning |
|-----|---------|--|
| 7 | 128 | Power on – this bit is always set. |
| 6 | 64 | User Request – a 1 on this position indicates that the 4400 is no longer controlled by remote commands but by user interaction. |
| 5 | 32 | Command error – this bit indicates that one of the "SCPI command errors" occurred. |
| 4 | 16 | Execution error – is set after a SCPI execution error did occur. |

| | | |
|---|---|--|
| 3 | 8 | Device-dependent error – this bit indicates that a device-specific SCPI error did occur. |
| 2 | 4 | Query error – is set after a SCPI query error occurred. |
| 1 | 2 | Request control – this bit is reserved for future use. |
| 0 | 1 | Operation complete flag – is set as soon as the execution of a command has been completed. |

***ESE**

| | |
|--------------------|---|
| Syntax | :*ESE <int1> |
| Parameters | int1 is an integer. The valid range is from 0 to 255 (8 bits). |
| Description | Sets the enable filter (mask) of the event status register. int1 is the decimal representation of the binary mask. The mask and the current content of the event status register will be ANDed. If the result is not zero, then bit 5 of the "Service register" will be set. |
| Query | The query form reads out the enable filter (mask) currently set and returns its binary representation in a string. |
| Example | :*ESE 128 As soon as power has been switched on, bit 7 (Power on) will be set. ANDed with the mask 128 , a binary 1 will occur and thus bit 5 of the service register will be set. |

***ESR?**

| | |
|--------------------|---|
| Syntax | :*ESR? |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | Returns the decimal representation of the current contents of the event status register in a string. Note: This register is self-destructive, i.e. its contents will be cleared after reading. |
| Example | After power-on, an :*ESR? command will return " 128 ". This means that bit 7 is set and all the other bits of the event status register are 0 . The command will clear the event status register and a subsequent :*ESR? command will return " 0 ". |

Commands affecting the service register

The service register represents the highest level within the report structure of the 4400.
The service register contains eight bits.
If any one of the bits 0...5 or 7 is set, the summary status bit (bit 6) of the service register will be set as well.

NOTE

The service register is self-destructive. This means that its contents will be cleared after reading.

| Bit | Decimal | Meaning |
|-----|---------|--|
| 7 | 128 | OPERational status summary. When this bit is set, an event within the "General operation register group" (e.g. the 4400 is waiting for a trigger) passed all filters. |
| 6 | 64 | Summary status bit. This bit will always be set as soon as any other bit of the service register has been set. Note: The summary status bit may be ANDed with the service request enable filter in order to generate a Service ReQuest on GPIB. The related command is *SRE (see below for details). |
| 5 | 32 | Event status summary. When this bit is set, an event within the "Event status register group" (e.g. an error occurred) passed all filters. |
| 4 | 16 | Message available. This bit will be set to 1 as soon as a query has been completed and measurement result(s) are available. |
| 3 | 8 | QUESTionable status summary. If this bit is set, an event within the "General questionable status register group" (e.g. 'value out of range') passed all filters. |
| 2 | 4 | Error queue status. When this bit is set, the error queue contains error messages. Up to 10 error messages can be logged in the error queue. The error queue can be read out, using the :SYSTem:ERRor? command. |
| 1 | 2 | Remote command completed. This bit will be set to 1 after a remote (SCPI) command has been completed. Note: However, when the 4400 receives a SCPI command, it will block the GPIB until the command has been completed. |
| 0 | 1 | Message queue status. This bit will be set to 1 as soon as a message is available in the 4400's internal message queue. Up to 10 messages can be logged in the message queue. To write to or to read from the message queue, use the :SYSTem:MESSAge command. |

***SRE**

| | |
|--------------------|---|
| Syntax | :*SRE <int1> |
| Parameters | int1 is an integer. The valid range is from 0 to 255 (8 bits). |
| Description | Sets the enable filter (mask) for the service register. int1 is the decimal representation of this binary mask. The mask and the current content of the service register will be ANDed. If the result is not zero, a service request (SRQ) will occur on the GPIB. |
| Query | The query form reads out the mask currently set and returns its binary representation in a string. |
| Example | :*SRE 68 As soon as an error occurs, bits 2 and 6 of the service register will be set. ANDed with the mask (68), a binary 1 will be the result and a SRQ will occur on the GPIB. |

***STB?**

| | |
|--------------------|---|
| Syntax | :*STB? |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | Returns the decimal representation of the current contents of the service register in a string. Note: This register is self-destructive, i.e. its contents will be cleared after reading. |
| Example | A :*STB? command returns "68". The return value of 68 (= 64 + 4) means that an error occurred (4). |

SYSTEM subsystem

System parameters, such as the number of unread error messages, the test set's GPIB address etc. may be read out or set using the commands of this subsystem.

:SYSTEM:ERROR[:NEXT]?

| | |
|--------------------|--|
| Syntax | <code>:SYSTEM:ERROR[:NEXT]?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the oldest unread error message from the internal error queue of the Willtek 4400. The queue entry returned will be a string containing the error no. and additional text. The maximum length of the string is 255 characters. Note: An overview of all SCPI error messages can be found in section " SCPI command errors ". |
| Example | <code>:SYSTEM:ERROR:NEXT?</code> String returned: <code>"66 GPIB cannot be initialised."</code> |

:SYSTEM:ERROR:COUNT?

| | |
|--------------------|--|
| Syntax | <code>:SYSTEM:ERROR:COUNT?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the number of unread error messages in the internal error queue of the Willtek 4400. The string returned will contain one integer. The maximum number of errors stored internally is 10. |
| Example | <code>:SYSTEM:ERROR:COUNT?</code> String returned: <code>"0"</code> This means that there are no unread error messages in the error queue. |

:SYSTEM:ERROR:CODE[:NEXT]?

| | |
|--------------------|--|
| Syntax | <code>:SYSTEM:ERROR:CODE[:NEXT]?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the code of the oldest unread error message in the internal error queue of the Willtek 4400. The string returned will contain one integer (and no text). Note: An overview of all SCPI error messages can be found in section " SCPI command errors ". |
| Example | <code>:SYSTEM:ERROR:CODE?</code> String returned: <code>"66"</code> This means that the GPIB could not be initialized. |

:SYSTem:ERRor:CODE:ALL?

| | |
|--------------------|---|
| Syntax | :SYSTem:ERRor:CODE:ALL? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the error codes of all unread error messages in the internal error queue of the Willtek 4400. The string returned will contain a maximum of 10 integers, separated by commas. Note: An overview of all SCPI error messages can be found in section " SCPI command errors ". |
| Example | :SYSTem:ERRor:CODE:ALL? String returned: "371,66" This means that there were two unread error messages in the error queue (the first one indicating that there was a timeout on a FETCh command and the second one meaning that the GPIB could not be initialized). |

:SYSTem:ERRor:REMOte:DISPlay

| | |
|--------------------|---|
| Syntax | :SYSTem:ERRor:REMOte:DISPlay <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: OFF ON. Default is OFF. |
| Description | Switches the error display on the start screen on and off. If it is switched to ON, the error log will be displayed after the first error occurred. |
| Query | Reads and returns the current setting. |
| Example | :SYST:ERR:REM:DISP ON In remote case the error log window will be displayed. |

:SYSTem:MESSage

| | |
|--------------------|--|
| Syntax | :SYSTem:MESSage <string1> |
| Parameters | string1 is a string (text) parameter. The maximum length of string1 is 255 characters. |
| Description | Writes the message specified with the string1 parameter to the Willtek 4400's internal system message queue. |
| Query | Reads and returns the oldest unread message in the Willtek 4400's internal message queue. The string returned will contain a maximum of 255 characters. |
| Example | :SYSTem:MESSage "23.17,Procedure A5" :SYST:MESS? String returned: "23.17,Procedure A5" In this example, a RAPID! program performs some internal calculations and then writes the result to the system message queue. This result is then read by the external controller. |

:SYSTem:COMMunicate:LOCal

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:LOCal |
| Parameters | There are no parameters. |
| Description | Sets up the Willtek 4400 to allow manual operation on the front panel during SCPI operation. Note: This command may be used e.g. to allow interactive alignment procedures in a production or quality assurance flow. |
| Query | There is no query form of this command available. |
| Example | :SYSTem:COMM:LOC |

:SYSTem:COMMunicate:GPIB:ADDRESS

| | |
|--------------------|--|
| Syntax | :SYSTem:COMMunicate:GPIB:ADDRESS <int1>[,<int2>] |
| Parameters | intx are two integers. The minimum value for int1 is 1 , the maximum is 32 . The default value is 4 . The minimum value for int2 is 0 , the maximum is 30 . The default value is 1 . int1 must be specified while int2 is an optional parameter. |
| Description | Sets the GPIB address of the Willtek 4400. For details regarding the GPIB address, refer to section Setting the GPIB address. |
| Query | Reads and returns the current setting of the GPIB address as explained above. |
| Example | :SYST:COMM:GPIB:ADDR 14 Sets the GPIB address to 14. |

:SYSTem:COMMunicate:GPIB:TERMinator

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:GPIB:TERMinator <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: LF CR CRLF . Default is LF . |
| Description | Sets the terminator used on the GPIB. For details regarding the terminator, refer to section Setup. |
| Query | Reads and returns the current setting of the terminator used on the GPIB as explained above. |
| Example | :SYST:COMM:GPIB:TERM CRLF Sets the GPIB terminator to 'carriage return plus line feed'. |

:SYSTem:COMMunicate:TCPIP:ADDRESS

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:TCPIP:ADDRESS <string> |
| Parameters | string is a string only containing the IP address for the 4400. |
| Description | This command sets the IP address of the 4400. See section I/O Configuration for more details. |

| | |
|----------------|---|
| Query | Reads and returns the current setting of the IP address as explained above. |
| Example | SYST:COMM:TCP:ADDR "192.16.16.114" sets the IP address to a defined value. |

:SYSTem:COMMunicate:TCPIP:NETMask

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:TCPIP:NETMask <string> |
| Parameters | string is a string only containing the net mask for the 4400. |
| Description | This command sets the net mask of the 4400. See section I/O Configuration for more details. |
| Query | Reads and returns the current setting of the net mask as explained above. |
| Example | SYST:COMM:TCP:NETM "255.255.255.0" sets the net mask to a defined value. |

:SYSTem:COMMunicate:TCPIP:GATeway

| | |
|--------------------|--|
| Syntax | :SYSTem:COMMunicate:TCPIP:GATeway <string> |
| Parameters | string is a string only containing the default gateway address for the 4400. |
| Description | This command sets the default gateway address of the 4400. See section I/O Configuration for more details. |
| Query | Reads and returns the current setting of the gateway address as explained above. |
| Example | SYST:COMM:TCP:GAT "192.16.16.1" sets the gateway address to a defined value. |

:SYSTem:COMMunicate:TCPIP:PORT

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:TCPIP:PORT <int> |
| Parameters | int defines the TCP/IP port address of the 4400. The address must be in the range from 49152 to 65535. |
| Description | This command sets the port address on which the 4400 can be controlled via LAN. See section I/O Configuration for more details. |
| Query | Reads and returns the current setting of the port used by TCP/IP as explained above. |
| Example | SYST:COMM:TCP:PORT 49200 sets the TCP/IP port address to its default. |

:SYSTem:COMMunicate:TCPIP:TERMinator

| | |
|--------------------|--|
| Syntax | :SYSTem:COMMunicate:TCPIP:TERMinator <PredefExp> |
| Parameters | PredefExp can take on one of the following values: LF or CR or CRLF . The default is LF . |
| Description | The command defines the delimiter for SCPI control strings. See section I/O Configuration for more details. |

| | |
|----------------|---|
| Query | Reads and returns the current setting of the terminator used by TCP/IP as explained above. |
| Example | SYST:COMM:TCP:TERM CRLF sets the line terminator for SCPI commands via LAN to CR (Carriage Return) followed by LF (Line Feed). |

:SYSTem:COMMunicate:TCPIP:MOUNT

| | |
|--------------------|---|
| Syntax | :SYSTem:COMMunicate:TCPIP:MOUNT <string1> <string2> |
| Parameters | string1 defines the network address which shall be mounted as a device for remote control. The maximum allowable length of the string is 255 characters. string2 is the symbolic device address used in SCPI programming. The maximum allowable length of string2 is 25 characters. The default is "server". |
| Description | This command mounts a server disk as a 4400 device which can be used to load or save data to/from. See section I/O Configuration for more details. |
| Query | Reads and returns the last settings for the mount path and the corresponding local name as explained above. |
| Example | :SYST:COMM:TCP:MOUNT "unixpc/disk2/results", "resdir" |

:SYSTem:COMMunicate:TCPIP:DHCP

| | |
|--------------------|--|
| Syntax | :SYSTem:COMMunicate:TCPIP:DHCP <PredefExp> |
| Parameters | PredefExp can take on one of the following values: ON or OFF. The default is OFF. |
| Description | The command turns DHCP on or off. See section I/O Configuration for more details. |
| Query | Reads and returns the current setting of DHCP operation. |
| Example | :SYST:COMM:TCP:DHCP ON sets the software to use DHCP. |

:SYSTem:COMMunicate:TCPIP:DHCP

| | |
|--------------------|--|
| Syntax | :SYSTem:COMMunicate:TCPIP:DHCP <PredefExp> |
| Parameters | PredefExp can take on one of the following values: ON or OFF. The default is OFF. |
| Description | The command turns DHCP on or off. See section I/O Configuration for more details. |
| Query | Reads and returns the current setting of DHCP operation. |
| Example | :SYST:COMM:TCP:DHCP ON sets the software to use DHCP. |

:SYSTEM:COMMunicate:SERA:PARAMeter

| | |
|--------------------|--|
| Syntax | :SYSTEM:COMMunicate:SERA:PARAMeter <int1>,<int2>,<int3>,<PredefExpr4> |
| Parameters | There are four parameters. int1 is the bit rate on the serial interface. Valid values are 110,300,600,1200,2400,4800,9600,19200,38400,57600,115200 . The default value is 38400 . int2 is the number of bits per character. The minimum value is 5 , the maximum is 8 . The default value is 8 . int3 is the number of stop bits. It can take on the values 1 or 2 . The default value is 1 . <PredefExpr4> specifies the parity bit. The value is one of the following pre-defined expressions: NO ODD EVEN . Default is NO . |
| Description | Sets the parameters for serial port COM1. This command uses the following format: baud,bits,stop,parity where baud stands for the bit rate (int1), bits stands for the number of bits per character (int2), stop stands for the number of stop bits (int3) and parity represents the parity (No, Odd or Even)(PredefExpr4). |
| Query | Reads and returns the current settings of COM1 as explained above. |
| Example | :SYST:COMM:SERA:PAR 9600,8,1,ODD :SYST:COMM:SERA:PAR? String returned: "9600,8,1,ODD". |

:SYSTEM:COMMunicate:SERA:BAUD

| | |
|--------------------|--|
| Syntax | :SYSTEM:COMMunicate:SERA:BAUD <int1> |
| Parameters | int1 is the bit rate on the serial interface. Valid values are 9600,19200,38400,57600,115200 . The default value is 57600 . |
| Description | Sets the baud rate for serial port COM1. The other serial parameter settings are fixed at 8 bits per char, 1 stop bit, no parity. |
| Query | Reads and returns the current settings of COM1 as explained above. |
| Example | :SYST:COMM:SERA:BAUD 115200 :SYST:COMM:SERA:BAUD? Returns the following string: "9600". |

:SYSTEM:KEYBoard

| | |
|-------------------|--|
| Syntax | :SYSTEM:KEYBoard <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: USA BELgium_fr BELgium_nl CANFr CANEng DEN FR GER ITA JAP LATAm_spa LATAm_port NL NOR PORTugal SPA SWE SWISS_fr SWISS_ger UK . Default is USA . |

| | |
|--------------------|--|
| Description | Selects the language for the external keyboard. Note: The language setting defines the position and type of special characters. |
| Query | Reads and returns the current setting of the language of the external keyboard. The string returned will contain one of the predefined expressions as explained above. |
| Example | :SYST:KEYB SWE Sets the language for the external keyboard to 'Swedish'. |

:SYSTem:DATE

| | |
|--------------------|---|
| Syntax | :SYSTem:DATE <int1>,<int2>,<int3> |
| Parameters | intx are three integers. The minimum value for int1 is 1998 , the maximum is 2100 . The default value is 1998 . The minimum value for int2 is 1 , the maximum is 12 . The default value is 1 . The minimum value for int3 is 1 , the maximum is 31 . The default value is 1 . |
| Description | Sets the system date. This command uses the following format: jjjjmmdd where jjjj stands for the four digits of the year (int1), mm gives the two digits of the current month (int2) and, dd represents the day of the current month (int3). |
| Query | Reads and returns the current system date in a string, using the format explained above. |
| Example | :SYST:DATE 2001,7,6 Sets the system date to the 6th of July, 2001. |

:SYSTem:TIME

| | |
|--------------------|---|
| Syntax | :SYSTem:TIME <int1>,<int2>,<int3> |
| Parameters | intx are three integers. The minimum value for int1 is 0 , the maximum is 23 . The default value is 0 . The minimum value for int2 is 0 , the maximum is 59 . The default value is 0 . The minimum value for int3 is 0 , the maximum is 59 . The default value is 0 . |
| Description | Sets the system time. This command uses the following format: hhmmss where hh stands for the two digits of the current hour, using a 24 hour time format (int1), mm gives the two digits of the current minute (int2) and, ss represents the seconds of the system time (int3). |
| Query | Reads and returns the current system time in a string, using the format explained above. |
| Example | :SYST:TIME? String returned: " 12,56,05 ". |

:SYSTem:VERSion?

| | |
|--------------------|---|
| Syntax | :SYSTem:VERSion? |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |

| | |
|----------------|---|
| Query | Reads and returns the version number of the SCPI command system used on your Willtek 4400. The command will return a string, containing one floating point real number. |
| Example | :SYST:VERS? String returned: "2001.7". |

:SYSTem:PRINTer

| | |
|--------------------|--|
| Syntax | :SYSTem:PRINTer <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: HP400 HP680 EPST HPLJ EPSP EPSX EPSI EPCI EPS1 EPPX EPC2 EPC4 EPC5 EPC6 EPC8 EPC1 EPC3 BMPF CANO. Default is HP400 . |
| Description | Selects the printer for screen dumps. The printers selectable are: HP400 means the Hewlett-Packard (HP) deskjet 400 series, HP680 stands for the DP deskjet 680 series, EPST means the Epson (EP) stylus series, HPLJ stands for the HP laserjet series, EPSP, EPSX, EPSI, EPCI, EPS1, EPPX, EPC1, EPC2, EPC3, EPC4, EPC5, EPC6, and EPC8 stand for the corresponding Epson printer series, BMPF means that the screen dump will be output as a bit map file, while CANO will generate an output signal suitable for Canon printers. |
| Query | Reads and returns the current setting for the printer. The string returned will contain one of the predefined expressions as explained above. |
| Example | :SYST:PRIN? String returned: " HPLJ ". |

STATus subsystem

The STATus subsystem offers commands to read out and deal with

- the general operation register group and its subordinate groups of registers and
- the general questionable status register group and its subordinate groups of registers.

NOTE

The commands to deal with the event status register group and the service register are part of the ["SCPI command errors"](#).

NOTE

For further details on the STATus subsystem, please refer to section ["Understanding the STATus subsystem"](#) on page 183.

:STATus:PRESet

| | |
|--------------------|--|
| Syntax | :STATus:PRESet |
| Parameters | There are no parameters. |
| Description | This command sets all user-definable settings of the status subsystem to their factory default values. The default values for the single commands are explained below. |
| Query | There is no query form of this command available. |
| Example | :STATus:PRESet Will reset all parameters of the status subsystem to their default values as listed below. |

:STATus:OPERation[:EVENT]?

| | |
|--------------------|--|
| Syntax | :STATus:OPERation[:EVENT]? |
| Parameters | There are no parameters. |
| Description | Reads out the current contents of the General Operation Event Register. Note: Event-type registers are read-only and self-destructive. They will be cleared after any query. |

| | |
|----------------|--|
| Query | <p>There is only a query form of this command available. The query will return a string, containing one integer.</p> <p>The value returned represents all general operation events that have occurred since the last query of this register. As with any event-type register, the single bits will remain set even when the reason for the bits to be set is no longer valid. Please note that this is the main difference between event-type and condition-type registers. Condition-type registers reflect the current state of the Willtek 4400. Consequently, condition-type registers will be updated continuously.</p> <p>Notes</p> <ul style="list-style-type: none"> – For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. – In case, a certain event shall be trapped in a loop within a program, always query the event-type register. |
| Example | <p>:STATus:OPERation:EVENT?</p> <p>Value returned: "32".</p> <p>This means that bit 5 (the 'waiting for a trigger' bit) has been set. This indicates that a 'waiting for a trigger' event did occur. If you want to know whether the Willtek 4400 is still waiting for the trigger to occur, read out the related condition-type register (see command explained below).</p> |

:STATus:OPERation:CONDition?

| | |
|--------------------|---|
| Syntax | :STATus:OPERation:CONDition? |
| Parameters | There are no parameters. |
| Description | <p>This command reads out the current contents of the General Operation Condition Register. This register reflects the current operational state of the Willtek 4400 and will be updated continuously.</p> <p>Note: This register is nondestructive. This means that it will keep its contents after any query.</p> |
| Query | <p>There is only a query form of this command available. The query will return a string, containing one integer. The integer will express all bits currently set in the 16 bit general operation condition register.</p> <p>Notes</p> <ul style="list-style-type: none"> – For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. – Bit 15 (the MSB) of this register is not used. Therefore, the maximum value returned will be "32767". |
| Example | <p>:STATus:OPERation:CONDition?</p> <p>Value returned: "512".</p> <p>This means that bit 9 (the MEASure summary bit) has been set, indicating that some measurement is currently in progress. Bit 9 will be reset as soon as the measurement has been completed.</p> |

:STATus:OPERation:ENABLE

| | |
|-------------------|--|
| Syntax | :STATus:OPERation:ENABLE <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767 . The default value is 0. |

| | |
|--------------------|---|
| Description | This command sets the Enable filter of the " General operation register group ". This mask will be ANDed with the general operation event register and thus decide what kind of events will be forwarded to bit 7 of the service register. Note: The service register is often also referred to as the statusbyte register or status byte. |
| Query | There is no query form of this command available. |
| Example | :STATus:OPERation:ENABLE 129 This means that any setting of bits 1 (calibrating) or 7 (correcting) of the general operation event register will rise bit 7 of the service register. |

:STATus:OPERation:NTRansition

| | |
|--------------------|--|
| Syntax | :STATus:OPERation:NTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 0. |
| Description | Sets the NTR mask of the " General operation register group ". This mask will be ANDed with the up to 15 bits of the NTR transition filter of the general operation condition register to allow a reset (negative transition) of any bit (i.e. a transition from logic '1' to '0') to reach the general operation event register. Notes <ul style="list-style-type: none"> - The default of this mask is 0 – that means that the mask will not allow any negative transition of the lower 15 bits of the condition-type register to reach the event-type register. - Bit 16 of the general operation condition register is not used. |
| Query | There is no query form of this command available. |
| Example | :STATus:OPERation:NTRansition 32767 This means that all of the negative transitions of the condition-type register will be forwarded to the event-type register. |

:STATus:OPERation:PTRansition

| | |
|--------------------|---|
| Syntax | :STATus:OPERation:PTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 32767. |
| Description | Sets the PTR mask of the " General operation register group ". This mask will be ANDed with the up to 15 bits of the PTR transition filter of the general operation condition register to allow a positive transition of any bit (i.e. a transition from logic '0' to '1') to reach the general operation event register. Notes <ul style="list-style-type: none"> - The default of this mask is 32767 – that means that the mask will allow any positive transition of the lower 15 bits of the condition-type register to reach the event-type register. - Bit 16 of the general operation condition register is not used. |
| Query | There is no query form of this command available. |
| Example | :STATus:OPERation:PTRansition 0 This means that none of the positive transitions of the condition-type register will be forwarded to the event-type register. |

:STATus:OPERation:MEASuring[:EVENT]?

| | |
|--------------------|---|
| Syntax | :STATus:OPERation:MEASuring[:EVENT]? |
| Parameters | There are no parameters. |
| Description | <p>Reads out the current contents of the measuring operation event register.</p> <p>Note: Event-type registers are read-only and self-destructive. They will be cleared after any query.</p> |
| Query | <p>There is only a query form of this command available. The query will return a string, containing one integer. The value returned represents all measuring operation events that have occurred since the last query of this register. As with any event-type register, the single bits will remain set even when the reason for the bits to be set is no longer valid. Please note that this is the main difference between event-type and condition-type registers. Condition-type registers reflect the current state of the Willtek 4400. Consequently, the condition type registers will be updated continuously.</p> <p>Notes</p> <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section Understanding the STATus Subsystem. - In case, a certain event shall be trapped in a loop within a program, always query the event-type register. |
| Example | <p>:STATus:OPERation:MEASuring:EVENT? Value returned: "8". This means that bit 3 (stands for AF measurement) has been set.</p> |

:STAT:OPERation:MEASuring:CONDition?

| | |
|--------------------|---|
| Syntax | :STAT:OPERation:MEASuring:CONDition? |
| Parameters | There are no parameters. |
| Description | <p>This command reads out the current contents of the measuring operation condition register. This register reflects the current state of the Willtek 4400 in terms of measurements and will be updated continuously.</p> <p>Note: This register is nondestructive. This means that it will keep its contents after any query.</p> |
| Query | <p>There is only a query form of this command available. The query will return a string, containing one integer. The integer will express all bits set in the 16 bit measuring operation condition register.</p> <p>Notes</p> <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. - Bits 4 to 15 (the MSB) of this register are not used. Therefore, the maximum value returned will be "7". |
| Example | <p>:STATus:OPERation:MEASuring:CONDition? Value returned: "1". This means that bit 0 has been set and that there is an RFTX measurement currently in progress. This bit will be reset as soon as the RFTX measurements have been completed.</p> |

:STATus:OPERation:MEASuring:ENABLE

| | |
|--------------------|--|
| Syntax | :STATus:OPERation:MEASuring:ENABLE <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0 , the maximum value is 32767 . The default value is 0 . |
| Description | This command sets the enable filter of the "Measuring operation register group". This mask will be ANDed with the measuring operation event register and thus decide what kind of events will be forwarded to bit 9 of the "General operation register group". |
| Query | There is no query form of this command available. |
| Example | :STATus:OPERation:MEASuring:ENABLE 4 This means that any setting of bit 2 (RF spectrum) of the measuring operation event register will rise bit 9 of the general operation condition register. |

:STAT:OPERation:MEASuring:PTRansition

| | |
|--------------------|--|
| Syntax | :STAT:OPERation:MEASuring:PTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0 , the maximum value is 32767 . The default value is 32767 . |
| Description | Sets the PTR mask of the "Measuring operation register group". This mask will be ANDed with the up to 16 bits of the PTR filter of the measuring operation condition register to allow a positive transition of any bit (i.e. a transition from logic '0' to '1') to reach the measuring operation event register. Note: The default of this mask is 32767 – that means that the mask will allow any positive transition of the lower 15 bits of the condition-type register to reach the event-type register. |
| Query | There is no query form of this command available. |
| Example | :STATus:OPERation:MEASuring:PTRansition 0 This means that none of the positive transitions of the condition-type register will be forwarded to the event-type register. |

:STAT:OPERation:MEASuring:NTRansition

| | |
|--------------------|--|
| Syntax | :STAT:OPERation:MEASuring:NTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0 , the maximum value is 32767 . The default value is 0 . |
| Description | Sets the NTR mask of the "Measuring operation register group". This mask will be ANDed with the up to 16 bits of the NTR filter of the measuring operation condition register to allow a reset (negative transition) of any bit (i.e. a transition from logic '1' to '0') to reach the measuring operation event register. Note: The default of this mask is 0 – that means that the mask will not allow any negative transition of the 16 bits of the condition-type register to reach the event-type register of this group. |
| Query | There is no query form of this command available. |

| | |
|----------------|---|
| Example | :STATus:OPERation:MEASuring:NTRansition 0 This means that none of the negative transitions of the condition-type register will be forwarded to the event-type register. |
|----------------|---|

:STATus:QUESTionable[:EVENT]?

| | |
|--------------------|---|
| Syntax | :STATus:QUESTionable[:EVENT]? |
| Parameters | There are no parameters. |
| Description | Reads out the current contents of the general questionable status event register. Note: Event-type registers are read-only and self-destructive. They will be cleared after any query. |
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The value returned represents all general questionable status events that have occurred since the last query of this register. As with any event-type register, the single bits will remain set even when the reason for the bits to be set is no longer valid. Please note that this is the main difference between event-type and condition-type registers. Condition-type registers reflect the current state of the Willtek 4400. Consequently, condition-type registers will be updated continuously. Notes <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. - In case, a certain event shall be trapped in a loop within a program, always query the event-type register. |
| Example | :STATus:QUESTionable:EVENT? Value returned: "256". This means that bit 8 has been set, indicating that the calibration of the Willtek 4400 is out of range. If you want to know whether the Willtek 4400 calibration is still out of range, read out the related condition-type register (see command explained below). |

:STATus:QUESTionable:CONDition?

| | |
|--------------------|---|
| Syntax | :STATus:QUESTionable:CONDition? |
| Parameters | There are no parameters. |
| Description | This command reads out the current contents of the general questionable status condition register. This register reflects the current questionable state of the Willtek 4400 and will be updated continuously. Note: This register is nondestructive. This means that it will keep its contents after any query. |
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The integer will express all bits currently set in the 16 bit questionable status condition register. Notes <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. - Bit 15 (the MSB) of this register is not used. Therefore, the maximum value returned will be "32767". |

Example **:STATus:QUEStionable:CONDition?**
 Value returned: "512".
 This means that bit 9 (the RF summary bit) has been set, indicating a current problem on the RF side of the Willtek 4400.

:STATus:QUEStionable:ENABLe

Syntax **:STATus:QUEStionable:ENABLe <int1>**

Parameters **int1** is an integer. The minimum value for **int1** is **0**, the maximum value is **32767**. The default value is **0**.

Description This command sets the Enable filter of the "[General questionable status register group](#)". This mask will be **ANDed** with the general questionable status event register and thus decide what kind of events will be forwarded to bit 3 of the service register. **Note:** The service register is often also referred to as the statusbyte register or status byte.

Query There is no query form of this command available.

Example **:STATus:QUEStionable:ENABLe 512**
 This means that any setting of bit 9 (RF summary bit) of the general questionable status event register will rise bit 3 of the service register.

:STATus:QUEStionable:PTRansition

Syntax **:STATus:QUEStionable:PTRansition <int1>**

Parameters **int1** is an integer. The minimum value for **int1** is **0**, the maximum value is **32767**. The default value is **32767**.

Description Sets the PTR mask of the "[General questionable status register group](#)". This mask will be **ANDed** with the up to 16 bits of the PTR filter of the general questionable status condition register to allow a positive transition of any bit (i.e. a transition from logic '0' to '1') to reach the questionable status event register. **Note:** The default of this mask is 32767 – that means that the mask will allow any positive transition of the lower 15 bits of the condition-type register to reach the event-type register.

Query There is no query form of this command available.

Example **:STATus:QUEStionable:PTRansition 0**
 This means that none of the positive transitions of the condition-type register will be forwarded to the event-type register of this group.

:STATus:QUEStionable:NTRansition

Syntax **:STATus:QUEStionable:NTRansition <int1>**

Parameters **int1** is an integer. The minimum value for **int1** is **0**, the maximum value is **32767**. The default value is **0**.

| | |
|--------------------|--|
| Description | Sets the NTR mask of the " General questionable status register group ". This mask will be ANDed with the up to 16 bits of the NTR filter of the general questionable status condition register to allow a reset (negative transition) of any bit (i.e. a transition from logic '1' to '0') to reach the general questionable status event register. Note: The default of this mask is 0 – that means that the mask will not allow any negative transition of the 16 bits of the condition-type register to reach the event-type register. |
| Query | There is no query form of this command available. |
| Example | :STATus:QUESTionable:NTRansition 0 This means that none of the negative transitions of the condition-type register will be forwarded to the event-type register of this group. |

:STATus:QUESTionable:RF[:EVENT]?

| | |
|--------------------|--|
| Syntax | :STATus:QUESTionable:RF[:EVENT]? |
| Parameters | There are no parameters. |
| Description | Reads out the current contents of the RF questionable status event register. Note: Event-type registers are read-only and self-destructive. They will be cleared after any query. |
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The value returned represents all RF questionable status events that have occurred since the last query of this register. As with any event-type register, the single bits will remain set even when the reason for the bits to be set is no longer valid. Please note that this is the main difference between event-type and condition-type registers. Condition-type registers reflect the current state of the Willtek 4400. Consequently, condition-type registers will be updated continuously. Notes <ul style="list-style-type: none"> – For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. – In case, a certain event shall be trapped in a loop within a program, always query the event-type register. |
| Example | :STATus:QUESTionable:RF:EVENT? Value returned: "1". This means that bit 1 has been set, indicating an RF input overload. If you want to know whether this RF input overload still persists, read out the related condition-type register (see command explained below). |

:STATus:QUESTionable:RF:CONDition?

| | |
|--------------------|--|
| Syntax | :STATus:QUESTionable:RF:CONDition? |
| Parameters | There are no parameters. |
| Description | This command reads out the current contents of the RF questionable status condition register. This register reflects the current questionable state of the Willtek 4400 regarding RF and will be updated continuously. Note: This register is nondestructive. This means that it will keep its contents after any query. |

| | |
|----------------|--|
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The integer will express all bits currently set in the 16 bit RF questionable status condition register. |
| | Notes |
| | <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. - Bits 4 to 15 (the MSB) of this register are not used. Therefore, the maximum value returned will be "15" |
| Example | <p>:STATus:QUESTionable:RF:CONDition? Value returned: "8". This means that bit 3 has been set, indicating that the frequency currently received is out of range of the Willtek 4400 receiver.</p> |

:STATus:QUESTionable:RF:ENABLe

| | |
|--------------------|--|
| Syntax | :STATus:QUESTionable:RF:ENABLe <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 0. |
| Description | This command sets the enable filter of the " RF questionable status register group ". This mask will be ANDed with the RF questionable status event register and thus decide what kind of events will be forwarded to bit 9 of the " General questionable status register group ". |
| Query | There is no query form of this command available. |
| Example | <p>:STATus:QUESTionable:RF:ENABLe 15 This means that any setting of one of the lower four bits of the RF questionable status event register will rise bit 9 of the general questionable status condition register.</p> |

:STATus:QUESTionable:RF:PTRansition

| | |
|--------------------|--|
| Syntax | :STATus:QUESTionable:RF:PTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 32767. |
| Description | <p>Sets the PTR mask of the "RF questionable status register group". This mask will be ANDed with the up to 16 bits of the PTR filter of the RF questionable status condition register to allow a positive transition of any bit (i.e. a transition from logic '0' to '1') to reach the RF questionable status event register.</p> <p>Note: The default of this mask is 32767 – that means that the mask will allow any positive transition of the lower 15 bits of the condition-type register to reach the event-type register of this group.</p> |
| Query | There is no query form of this command available. |
| Example | <p>:STATus:QUESTionable:RF:PTRansition 0 This means that none of the positive transitions of the condition-type register will be forwarded to the event-type register.</p> |

:STATus:QUESTionable:RF:NTRansition

| | |
|--------------------|---|
| Syntax | :STATus:QUESTionable:RF:NTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 0. |
| Description | Sets the NTR mask of the "RF questionable status register group". This mask will be ANDed with the up to 16 bits of the NTR filter of the RF questionable status condition register to allow a negative transition of any bit (i.e. a transition from logic '1' to '0') to reach the RF questionable status event register. Note: The default of this mask is 0 – that means that the mask will not allow any negative transition of the 16 bits of the condition-type register to reach the event-type register of this group. |
| Query | There is no query form of this command available. |
| Example | :STATus:QUESTionable:RF:NTRansition 0 This means that none of the negative transitions of the condition-type register will be forwarded to the event-type register. |

:STATus:QUESTionable:SYNChron[:EVENT]?

| | |
|--------------------|---|
| Syntax | :STATus:QUESTionable:SYNChron[:EVENT]? |
| Parameters | There are no parameters. |
| Description | Reads out the current contents of the synchronization questionable status event register. Note: Event-type registers are read-only and self-destructive. They will be cleared after any query. |
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The value returned represents all synchronization questionable status events that have occurred since the last query of this register. As with any event-type register, the single bits will remain set even when the reason for the bits to be set is no longer valid. Please note that this is the main difference between event-type and condition-type registers. Condition-type registers reflect the current state of the Willtek 4400. Consequently, condition-type registers will be updated continuously. Notes <ul style="list-style-type: none"> – For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. – In case, a certain event shall be trapped in a loop within a program, always query the event-type register. |
| Example | :STATus:QUESTionable:SYNChron:EVENT? Value returned: "2". This means that bit 2 has been set, indicating that an external frame synchronization signal has been recognized on the SYNC IN/OUT socket on the rear panel of the Willtek 4400. |

:STAT:QUESTionable:SYNChron:CONDition?

| | |
|-------------------|--|
| Syntax | :STAT:QUESTionable:SYNChron:CONDition? |
| Parameters | There are no parameters. |

| | |
|--------------------|--|
| Description | This command reads out the current contents of the synchronization questionable status condition register. This register reflects the current questionable state of the Willtek 4400 regarding synchronization and will be updated continuously. Note: This register is nondestructive. This means that it will keep its contents after any query. |
| Query | There is only a query form of this command available. The query will return a string, containing one integer. The integer will express all bits currently set in the 16 bit synchronization questionable status condition register. Notes <ul style="list-style-type: none"> - For further details regarding the basic functions of the STATus subsystem, please refer to section "Understanding the STATus subsystem" on page 183. - Bits 2 to 15 (the MSB) of this register are not used. Therefore, the maximum value returned will be "3". |
| Example | :STATus:QUESTionable:SYNChron:CONDition? Value returned: "1". This means that bit 1 has been set, indicating that an external RF synchronization signal is being received on the EXT SYNC prog. socket on the real panel of the Willtek 4400. |

:STATus:QUESTionable:SYNChron:ENABLE

| | |
|--------------------|--|
| Syntax | :STATus:QUESTionable:SYNChron:ENABLE <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 0. |
| Description | This command sets the Enable filter of the " SYNChronization questionable status register group ". This mask will be ANDed with the synchronization questionable status event register and thus decide what kind of events will be forwarded to bit 10 of the " General questionable status register group ". |
| Query | There is no query form of this command available. |
| Example | :STATus:QUESTionable:SYNChron:ENABLE 3 This means that any setting of one of the lower two bits of the synchronization questionable status event register will rise bit 10 of the general questionable status condition register. |

:STAT:QUES:SYNChron:PTRansition

| | |
|--------------------|---|
| Syntax | :STAT:QUES:SYNChron:PTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 32767. The default value is 32767. |
| Description | Sets the PTR mask of the " SYNChronization questionable status register group ". This mask will be ANDed with the up to 16 bits of the PTR filter of the synchronization questionable status condition register to allow a positive transition of any bit (i.e. a transition from logic '0' to '1') to reach the synchronization questionable status event register. Note: The default of this mask is 32767 – that means that the mask will allow any positive transition of the lower 15 bits of the condition-type register to reach the event-type register of this group. |
| Query | There is no query form of this command available. |

| | |
|----------------|--|
| Example | :STAT:QUES:SYNChron:PTRansition 0 This means that none of the positive transitions of the synchronization questionable status condition register will be forwarded to the event-type register. |
|----------------|--|

:STAT:QUES:SYNChron:NTRansition

| | |
|--------------------|--|
| Syntax | :STAT:QUES:SYNChron:NTRansition <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0 , the maximum value is 32767 . The default value is 0 . |
| Description | Sets the NTR mask of the "SYNChronization questionable status register group". This mask will be ANDed with the up to 16 bits of the NTR filter of the synchronization questionable status condition register to allow a reset (negative transition) of any bit (i.e. a transition from logic '1' to '0') to reach the corresponding event-type register. Note: The default of this mask is 0 – that means that the mask will not allow any negative transition of the 16 bits of the condition-type register to reach the corresponding event-type register. |
| Query | There is no query form of this command available. |
| Example | :STATus:QUESTionable:SYNChron:NTRansition 0 This means that none of the negative transitions of the condition-type register will be forwarded to the event-type register of this group. |

PROG

ram subsystem

This subsystem contains commands related to loading and executing RAPID! program files.

:PROGram[:SELeCted]:NAME

| | |
|--------------------|--|
| Syntax | :PROG |
| Parameters | string is an existing RAPID! program file name. The maximum length of string is 50 characters. |
| Description | This command loads a RAPID! basic program file. To start or stop the program, please use the :PROG ram[:SELeCted]:STATus command described below. The name of the RAPID! basic program file may contain a path. For more details regarding RAPID!, please refer to chapter "RAPID!" on page 87 . |
| Query | The query returns the name of the currently loaded RAPID! basic program file as a string. |
| Example | :PROG ram:SELeCted:NAME "test7389.bas" :PROG :NAME? Value returned: "test7389.bas". |

:PROGram[:SELeCted]:NUMBer

| | |
|--------------------|---|
| Syntax | :PROG |
| Parameters | string is the name of an existing numeric variable in the currently loaded RAPID! program file. The maximum length of string is 100 characters. int1 and int2 are two integers. The minimum value for each intx is -32765, the maximum value is 32765. The default value for each intx is 0. |
| Description | This command exists in a query form only. It is used to read out the current value of a numeric variable of the currently loaded RAPID! basic program. The value will be returned as a floating point real number. For more details regarding floating point real numbers, please refer to section "Variables" on page 108 . |
| Query | The query returns the current value of a numerical variable as one floating point real number contained in a string. |
| Example | :PROG ram:SELeCted:NUMBer? "BER_Samples",1,0 This example will read out the current value of the variable BER_Samples at array position 1, 0. The result could be 20000.0. |

:PROGram[:SELeCted]:STRing

| | |
|--------------------|---|
| Syntax | :PROG |
| Parameters | string is the name of an existing string variable in the currently loaded RAPID! program file. The maximum length of string is 100 characters. int1 and int2 are two integers. The minimum value for each intx is -32765 , the maximum value is 32765 . The default value for each intx is 0 . |
| Description | This command exists in a query form only. It is used to read out the current value of a string variable of the currently loaded RAPID! basic program. The value will be returned as a string. For more details regarding strings, please refer to section "Variables" on page 108 . |
| Query | The query returns the current value of a string variable as a string. |
| Example | :PROG |

:PROGram[:SELeCted]:CRC

| | |
|--------------------|--|
| Syntax | :PROG |
| Parameters | There are no parameters. |
| Description | This command exists in a query form only. It allows to read out the CRC Checksum of the RAPID! basic program currently loaded. The value will be returned as a long integer number. Note: A long integer number allows a maximum value of 2,147,483,647. |
| Query | The query returns the CRC checksum of the currently loaded RAPID! basic program as a string, containing one long integer number. |
| Example | :PROG |

:PROGram[:SELeCted]:STATus

| | |
|--------------------|--|
| Syntax | :PROG |
| Parameters | PreDefExp is one of the following predefined expressions: RUN STOP . Default is RUN . |
| Description | Starts (RUN) or stops (STOP) the execution of the currently loaded RAPID! program file. For details regarding RAPID!, please refer to chapter "RAPID!" on page 87 . |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :PROG |

:PROGram[:SElected]:MESSAge

| | |
|--------------------|---|
| Syntax | :PROGram[:SElected]:MESSAge <string> |
| Parameters | string is string with a maximum length of 255 characters. |
| Description | This command either reads (in its query form) or writes a string from/to the RAPID! program message queue. |
| Query | The query returns the latest entry of the RAPID! program message queue. |
| Example | :PROGram:SElected:MESSAge "Put this in the queue" :PROG:MESS? The result read back will be "Put this in the queue". |

:PROGram[:SElected]:SRE

| | |
|--------------------|--|
| Syntax | :PROGram[:SElected]:SRE <int1> |
| Parameters | int1 is an integer. Minimum value for int1 is 0, maximum value is 255. Default value for int1 is 0. |
| Description | This command sets the mask for the service register of the RAPID! program. This is similar to the *SRE common command. For more details regarding the status subsystem of the Willtek 4400, please refer to section " Understanding the STATUS subsystem " on page 183. |
| Query | There is no query form of this command available. |
| Example | :PROGram:SElected:SRE 16 This will set bit 4 of the mask of the service register. The contents of the service register will be ANDed with this mask. As soon as there is a valid measurement result available for the measurement started last, bit 4 of the service register will be set to '1'. ANDed with this mask (bit 4 set to 1), a binary 1 will be the result and a service request will be issued. |

:PROGram[:SElected]:STB

| | |
|--------------------|--|
| Syntax | :PROGram[:SElected]:STB |
| Parameters | There are no parameters. |
| Description | This command exists in a query form only. It reads out the contents of the service register of the RAPID! program. This is similar to the *STB common command. For more details regarding the status subsystem of the Willtek 4400, please refer to section " Understanding the STATUS subsystem " on page 183. |
| Query | The command returns a string containing one integer. This integer represents the contents of the service register of the RAPID! program. |
| Example | :PROGram:SElected:STB? If a 4 is returned, this indicates that there is a valid measurement result available for the measurement started last. |

FORMat subsystem

The FORMat subsystem sets and queries settings concerning the data output in remote mode.

:FORMat:MRESult:STYPe

| | |
|--------------------|--|
| Syntax | :FORMat:MRESult:STYPe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: STB SIGNaling MEASuring OPERation QUESTionable ALL . Default is that no additional information will be provided with the measurement result values. |
| Description | Specifies the type of additional information to be returned with any measurement result obtained by a FETCh command. The meaning of the settings is as follows: STB will deliver the current contents of the "Service register". SIGNaling stands for the current contents of the signaling operation condition register. MEASuring means that the current contents of the measuring operation condition register will be delivered back. OPERation will deliver the current contents of the general operation condition register. QUESTionable stands for the current contents of the general questionable status condition register. ALL will deliver the current contents of the eight most important registers. The order is as follows: 1. "Service register", 2. event status register, 3. general operation condition register, 4. signaling operation condition register, 5. measuring operation condition register, 6. general questionable status condition register, 7. RF questionable status condition register, and 8. synchronization questionable status condition register. The contents of every single register will be returned as an integer; the single values will be separated by commas. |
| Query | There is no query form of this command available. |
| Example | :FORMat:MRESult:HEADer ON :FORMat:MRESult:STYPe ALL :MEASure:RFTX:PRMS :FETCh:RFTX:PRMS The first command switches the transmission of the additional information on, the second command specifies that all current contents of the eight most important registers shall be returned. The third command starts a continuous measurement. The last command finally delivers the latest measurement result value plus the current contents of the main registers. The string delivered back: "0,128,256,8,1,0,0,0,4.63". |

:FORMat:MRESult:HEADer

| | |
|---------------|------------------------------------|
| Syntax | :FORMat:MRESult:HEADer <PredefExp> |
|---------------|------------------------------------|

| | |
|--------------------|---|
| Parameters | PredefExp is one of the following predefined expressions: OFF ON . Default is OFF . |
| Description | Switches the transmission of the additional information (see explanation of the command above) either on or off. ON means that the current contents of the corresponding register(s) will be added at the beginning of every string returned by a FETCh command. |
| Query | There is no query form of this command available. |
| Example | :FORM:MRES:HEAD This command will switch the transmission of the additional information off. |

:FORMat:ADELimiter

| | |
|--------------------|--|
| Syntax | :FORMat:ADELimiter <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: COMMa COLOn SEMIcolon . Default is COMMa . |
| Description | Selects the delimiter to be used to separate parameters on commands and single measurement result values. COMMa stands for commas (default), COLOn sets the delimiter to be a colon (:), while SEMIcolon will use and expect a semicolon (;) to be used. |
| Query | There is no query form of this command available. |
| Example | :FORM:ADEL Defines the comma to be used as delimiter for both commands and measurement results. |

:FORMat:RESolution

| | |
|--------------------|--|
| Syntax | :FORMat:RESolution <int1> |
| Parameters | int1 is an integer. The minimum value for <int1> is 0, the maximum is 20. The default value is 6. |
| Description | Defines the number of digits after the decimal point to be used for floating point real numbers. |
| Query | There is no query form of this command available. |
| Example | :FORM:RES 0 Defines that there will be no digits after the decimal point. |

CONFigure subsystem

This subsystem incorporates all changeable NB parameters of all systems implemented. The key commands are the following:

| | |
|---|---|
| CONFigure:CSYSstem | Selects the communications system to work with. |
| CONFigure:<SystemOption>:... | These commands select parameters within a communications system. |
| CONFigure:<SystemOption>:NB:... | These are the commands to set specific system parameters like the base station's RF output power level or its identity. |
| CONFigure:<SystemOption>:UE:... | The mobile-specific information is handed over to the 4400 using these commands. One example is the power level. |
| CONFigure:<SystemOption>:BER:... | These commands set the BER parameters. |
| CONFigure:<SystemOption>:GROup:... | With the help of these commands, groups of measurements may be defined. |
| CONFigure:COUPloss:... | These commands provide access to the coupling loss compensation feature of the 4400. |

NOTE

Always select the communication system via the `:Configure:CSYSstem` command first before you change any parameters using different SCPI commands.

:CONFigure:CSYSstem

| | |
|-------------------|--|
| Syntax | <code>:CONFigure:CSYSstem <PredefExp></code> |
| Parameters | <p>PredefExp is one of the following predefined expressions: NONE GCGenana GPGenana EGPGenana CDGenana WCGenana AMGenana GSM GPRS EGPRs CDMA WCDMA AMPS TDSCdma . Default is NONE</p> |

| | |
|--------------------|---|
| Description | <p>Selects the type of communication system to be used.</p> <p>NONE means that there is no system loaded and that there will be no basic generator or analyzer functionality available. This parameter may be used to cancel all RF radiation from the Willtek 4400.</p> <p>GENana will make the Willtek 4400 work as a generator and analyzer for circuit-switched GSM signals. This setting is identical to GCGenana. Because of the implementation of packet-data channels, we recommend not to use the GENana parameter in new programs but the GCGenana parameter as it clearly marks that the Willtek 4400 will generate and analyze circuit-switched GSM signals only with this setting.</p> <p>GCGenana will set up the Willtek 4400 as a generator and analyzer for circuit-switched GSM signals (including multislots/HSCSD). Generator/analyzer means that there will be no call setup and therefore, all measurements will be asynchronous.</p> <p>GPGenana will make the Willtek 4400 work as an asynchronous generator and analyzer for all kinds of GSM signals (circuit-switched signals in single or multislots mode as well as packet-data channels (PDTCH in GPRS)).</p> <p>GSM will set up the Willtek 4400 as a test set for circuit-switched GSM systems (including multislots/HSCSD). All tests performed with this setting require a call setup. Therefore, this test mode is called the 'call mode'.</p> <p>GPRS will bring the Willtek 4400 into call mode for standard GSM and GPRS systems. This means that this parameter will enable testing of all kinds of GSM signals (circuit-switched signals in single or multislots mode as well as packet-data channels (GPRS)).</p> <p>Note: Please keep in mind that you have to select the communication system first when working with SCPI, as the default of this command is NONE.</p> |
| Query | The query form of this command will return the current setting. The string delivered back will contain the short-form version of one of the predefined expressions explained above. |
| Example | <pre>:CONFigure:CSYStem GSM :CONF:CSYS? Value returned: "GSM".</pre> |

:CONFigure:COUPloss:STATe

| | |
|--------------------|---|
| Syntax | :CONFigure:COUPloss:STATe <PredefExp1> |
| Parameters | PredefExp1 is one of the following predefined expressions: ON OFF . Default is OFF . |
| Description | <p>This command switches the use of a coupling loss table either on or off.</p> <p>Note: Coupling loss tables are used to compensate e.g. losses in cables. For more details, please refer to section Coupling Loss.</p> |
| Query | The query form of this command will return the current setting. The string delivered back will contain one predefined expression as explained above. |
| Example | <pre>:CONFigure:COUPloss:STATe ON :CONF:COUP:STAT? Value returned: "ON"</pre> |

:CONFigure:COUPloss:NAME

| | |
|--------------------|---|
| Syntax | :CONFigure:COUPloss:NAME <string1> |
| Parameters | string1 is as string, giving the complete file name (and directory) of the coupling loss file to be loaded. The maximum length of string1 is 50 characters. The default for string1 is " example.cpl ". |
| Description | This command loads the coupling loss description file, specified with the command's parameter. Please note that the data contained in the file need to be activated (using the CONF:COUP:STAT ON command described above) before the data contained in the file specified will have any effect on the measurement results. For more details, please refer to section "Coupling Loss". |
| Query | The query form of this command will return the name of the currently loaded coupling loss description file. The string delivered back will contain the full file name. |
| Example | :CONFigure:COUPloss:NAME "m7389.cpl" :CONF:COUP:NAME? Value returned: "m7389.cpl" |

:CONFigure:COUPloss:INFormation?

| | |
|--------------------|---|
| Syntax | :CONFigure:COUPloss:INFormation? |
| Parameters | There are no parameters. |
| Description | This command is used to read out the comments saved with the coupling loss description file currently loaded. Note: There is only a query form of this command available. |
| Query | The query form of this command will return the comments saved with the coupling loss description. The string delivered back will contain a maximum of 255 characters. |
| Example | :CONFigure:COUPloss:INFormation? Value returned: " Motorola P7389 with Antenna Coupler " |

:CONFigure:COUPloss:DATA

| | |
|---------------|---|
| Syntax | :CONFigure:COUPloss:DATA <string1> ,<realf1>,<reala1>,<realf2>,<reala2> [, realf3] [, reala3] [, realf4] [, reala4] [, realf5] [, reala5] [, realf6] [, reala6] [, realf7] [, reala7] [, realf8] [, reala8] [, realf9] [, reala9] [, realf10] [, reala10] [, realf11] [, reala11] [, realf12] [, reala12] [, realf13] [, reala13] [, realf14] [, reala14] [, realf15] [, reala15] [, realf16] [, reala16] [, realf17] [, reala17] [, realf18] [, reala18] [, realf19] [, reala19] [, realf20] [, reala20] |
|---------------|---|

| | |
|--------------------|---|
| Parameters | <p>string1 is a comment line related to the coupling loss data. The maximum length of string1 is 255 characters.</p> <p>realfx are floating point real numbers, giving frequencies in MHz while realax are floating point real numbers, giving the corresponding coupling loss in dB. All realfx have to be within two frequency ranges. The lower frequency range is from 800.0 MHz to 1000.0 MHz; the higher frequency range is from 1700.0 MHz to 2000.0 MHz.</p> <p>The minimum resolution for all realfx values is 10 Hz (0.00001 MHz). The default value for all realfx is 800.0 MHz. The minimum value for all realax is -5.0 dB. The maximum value for all realax is 40.0 dB. The minimum resolution for all realax is 0.01 dB. The default value for all realax is 0.0.</p> <p>Notes</p> <ul style="list-style-type: none"> - Please keep in mind that at least one pair of values for a frequency and the related attenuation must be specified per frequency range, while all other 18 pairs are optional. - All realax values are interpreted as an attenuation level in dB. As a consequence, negative values mean an amplification of the input signal. |
| Description | With the help of this command, you may create a coupling loss description table, similar to how you would do it on the graphical user interface of the Willtek 4400. |
| Query | There is no query form of this command available. |
| Example | <code>:CONFigure:COUPloss:DATA "Motorola 7389 with cable #23",825.0,15.0,1750.0,19.0</code> |

:CONFigure:ESYNc

| | |
|--------------------|--|
| Syntax | <code>:CONFigure:ESYNc?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>The query form will return the status of the current external synchronization. The string delivered back will contain one of the following expressions: NONE MHZ5 MHZ10 MHZ13.</p> |
| Example | <p><code>:CONFigure:ESYNc?</code> Value returned for example: "MHZ10". In this example, the 4400 frequency reference is synchronized to an external 10 MHz clock signal fed into the EXT SYNC connector on the rear panel.</p> |

:CONFigure:TDSCdma:COUPloss:STATE

| | |
|--------------------|---|
| Syntax | <code>:CONFigure:TDSCdma:COUPloss:STATE <PredefExp1></code> |
| Parameters | <p>PredefExp1 is one of the following predefined expressions: ON OFF. Default is OFF.</p> |
| Description | <p>This command switches the use of a coupling loss table either ON or OFF. Note Coupling loss tables are used to compensate e.g. losses in cables. For more details, please refer to section Coupling Loss.</p> |

| | |
|----------------|--|
| Query | The query form of this command will return the current setting. The string delivered back will contain one predefined expression as explained above. |
| Example | :CONFigure:TDSCdma:COUPloss:STATe ON :CONF:COUP:STAT? Value returned in this example: "ON" |

:CONFigure:TDSCdma:COUPloss:NAME

| | |
|--------------------|--|
| Syntax | :CONFigure:TDSCdma:COUPloss:NAME <string1> |
| Parameters | string1 is a string, specifying the complete file name (and directory) of the coupling loss file to be loaded. The maximum length of string1 is 50 characters. The default for string1 is "example.cpl". |
| Description | This command loads the coupling loss description file, specified with the command's parameter. Please note that the data contained in the file need to be activated (using the CONF:COUP:STAT ON command described above) before the data contained in the file specified will have any effect on the measurement results. For more details, please refer to section Coupling Loss. |
| Query | The query form of this command will return the name of the currently loaded coupling loss description file. The string delivered back will contain the full file name. |
| Example | :CONFigure:TDSCdma:COUPloss:NAME "m7389.cpl" :CONF:COUP:NAME? Value returned in this example: "m7389.cpl" |

:CONFigure:TDSCdma:COUPloss:INFormation?

| | |
|--------------------|---|
| Syntax | :CONFigure:TDSCdma:COUPloss:INFormation? |
| Parameters | There are no parameters. |
| Description | This command is used to read out the comments saved with the coupling loss description file currently loaded. For more details, please refer to section Coupling Loss. Note: There is only a query form of this command available. |
| Query | The query form of this command will return the comments saved with the coupling loss description. The string delivered back will contain a maximum of 255 characters. |
| Example | :CONFigure:TDSCdma:COUPloss:INFormation? Value returned in this example: "Motorola P7389 with Antenna Coupler" |

:CONFigure:TDSCdma:COUPloss:DATA

| | |
|--------------------|--|
| Syntax | <pre>:CONFigure:TDSCdma:COUPloss:DATA <string1> ,<realf1>,<reala1>,<realf2>,<reala2> [,realf3] [,reala3] [,realf4] [,reala4] [,realf5] [,reala5] [,realf6] [,reala6] [,realf7] [,reala7] [,realf8] [,reala8] [,realf9] [,reala9] [,realf10] [,reala10] [,realf11] [,reala11] [,realf12] [,reala12] [,realf13] [,reala13] [,realf14] [,reala14] [,realf15] [,reala15] [,realf16] [,reala16] [,realf17] [,reala17] [,realf18] [,reala18] [,realf19] [,reala19] [,realf20] [,reala20]</pre> |
| Parameters | <p>string1 is a comment line related to the coupling loss data. The maximum length of string1 is 255 characters.</p> <p>realfx are floating point real numbers specifying frequencies in MHz while realax are floating point real numbers specifying the corresponding coupling loss in dB. All realfx have to be within two frequency ranges. The lower frequency range is from 800.0 MHz to 1000.0 MHz; the higher frequency range is from 1700.0 MHz to 2000.0 MHz.</p> <p>The minimum resolution for all realfx values is 10 Hz (0.00001 MHz). The default value for all realfx is 800.0 MHz. The minimum value for all realax is -5.0 dB. The maximum value for all realax is 40.0 dB. The minimum resolution for all realax is 0.01 dB. The default value for all realax is 0.0.</p> <p>Notes:</p> <ul style="list-style-type: none"> - Please keep in mind that at least one pair of values for a frequency and the related attenuation must be specified per frequency range, while all other 18 pairs are optional. - All realax values are interpreted as an attenuation level in dB. As a consequence, negative values mean an amplification of the input signal. |
| Description | <p>With the help of this command, you may create a coupling loss description table, similar to how you would do it on the graphical user interface of the Willtek 4400. For further details, please refer to section Coupling Loss.</p> |
| Query | <p>There is no query form of this command available.</p> |
| Example | <pre>:CONF:TDSC:COUP:DATA "Motorola 7389 with cable #23",825.0,15.0,1750.0,19.0</pre> |

MEASure subsystem

The MEASure subsystem is probably the most important SCPI command subsystem of the 4400. There, you will find all commands required to acquire measurement results of the mobile under test.

:MEASure [:CONT] :AFANalyser :GROup

| | |
|--------------------|--|
| Syntax | <code>:MEASure [:CONT] :AFANalyser :GROup</code> |
| Parameters | There are no parameters. |
| Description | <p>Starts a continuous measurement of the audio tests, specified with the <code>:CONF:MEAS:GRO:AFAN</code> command. To read out the latest measurement results, use the <code>:FETCh:AFAN:GROup</code> command.</p> <p>Notes:</p> <ul style="list-style-type: none">– Please keep in mind that the start of a new AF test will always terminate all other measurements (see section "The MEASure subsystem" on page 191 for details).– To perform any audio measurements on your Willtek 4400, the Audio Option must be installed.– For further details regarding group measurements, please refer to section ":MEASure:::GROup" on page 198. |
| Query | <p>The query form of this command will start the measurements and deliver back a string, containing the latest set of measurement result values. All measurement result values returned will be floating point real numbers. The order of the measurement result values returned is as described below (see command <code>:MEAS:AFAN:ALL</code>).</p> <p>The single measurement result values are separated by commas.</p> |
| Example | <p><code>:CONF:MEAS:GRO:AFAN SIN,FREQ</code> <code>:MEASure:CONTinuous:AFANalyser:GROup?</code></p> <p>In this example, the group of measurements is defined by a SINAD measurement, combined with an AF frequency measurement. The measurement result string returned is: <code>"1000.0,50.5"</code>.</p> <p>Because of the internal order, the first measurement result value delivered back is the audio frequency, the second one the SINAD.</p> |

:MEASure [:CONT] :AFANalyser :ALL

| | |
|-------------------|--|
| Syntax | <code>:MEASure [:CONT] :AFANalyser :ALL</code> |
| Parameters | There are no parameters. |

| | |
|--------------------|---|
| Description | <p>Starts a continuous measurement of the most important audio tests. To read out the latest measurement results, use the <code>:FETCh:AFAN:ALL</code> command.</p> <p>The audio tests performed by this command are:</p> <p>ACVPeakp, the peak-to-peak measurement of an AC voltage, ACVRms, the RMS-valued measurement of an AC voltage, DCV is used to measure AC ripple on DC lines (this measurement gives the root-mean square voltage of the AC component of the applied DC signal). FREQuency is the measurement of the audio frequency in Hertz, DISTortion is the distortion measurement on the third harmonic of a sine wave and expressed in %, while SINad is the measurement of the signal to noise ratio, expressed in dB.</p> <p>Notes:</p> <ul style="list-style-type: none"> - Please keep in mind that the start of a new AF test will always terminate all other measurements (see section "The MEASure subsystem" on page 191 for details). - Any AF measurement will need the Audio Option to be installed on your Willtek 4400. - For further details regarding group measurements, please refer to section "<code>:MEASure:....:GROup</code>" on page 198. |
| Query | <p>The query form of this command starts the measurements and – after all measurements have been completed and all measurement results obtained – delivers a string, containing six floating point real numbers, representing the six measurement result values. The order of these measurement result values delivered back is as follows:</p> <ol style="list-style-type: none"> 1. ACVPeakp, representing the AC peak-to-peak voltage of the AF signal, 2. ACVRms, representing the RMS-valued AC voltage of the AF signal, 3. DCV, representing the RMS-valued AC voltage on an applied DC signal, 4. FREQuency, representing the audio frequency, 5. DISTortion, representing the third-harmonic distortion of the applied sine-wave AF signal and, 6. SINad, representing the signal to noise ratio of the applied AF signal. |
| Example | <p><code>:MEASure:CONTinuous:AFANalyser:ALL?</code></p> <p>In this case, all audio measurements will be performed in a sequence. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing six measurement result values:</p> <p><code>"0.7,0.25,0.0,1000.0,0.3,50.5"</code>.</p> |

`:MEAS[:CONT]:AFAN:ACVoltage:PEAKp`

| | |
|--------------------|--|
| Syntax | <code>:MEAS[:CONT]:AFAN:ACVoltage:PEAKp</code> |
| Parameters | There are no parameters. |
| Description | <p>Starts a continuous measurement of the AC peak-to-peak voltage of the AF signal applied to the audio analyzer. To read out the latest measurement result, use the <code>:FETCh:AFAN:ACV:PEAK</code> command.</p> <p>Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400.</p> |
| Query | <p>The query form of this command starts the measurement and delivers a string, containing a floating point real number, representing the first measurement result value. The physical dimension is volt, measured peak-to-peak (V_{pp}).</p> |
| Example | <p><code>:MEAS:CONT:AFAN:ACV:PEAKp?</code></p> <p>String delivered back: <code>"0.7"</code>.</p> |

:MEAS[:CONT]:AFAN:ACVoltage:RMS

| | |
|--------------------|---|
| Syntax | :MEAS[:CONT]:AFAN:ACVoltage:RMS |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the RMS-valued AC voltage of the AF signal applied to the audio analyzer. To read out the latest measurement result, use the :FETCh:AFAN:ACV:RMS command. Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and delivers a string, containing a floating point real number, representing the first measurement result value. The physical dimension is volt, RMS-valued (V_{rms}). |
| Example | :MEAS:CONT:AFAN:ACVoltage:RMS? String delivered back: "0.25". |

:MEAS[:CONT]:AFAN:DCVoltage

| | |
|--------------------|--|
| Syntax | :MEAS[:CONT]:AFAN:DCVoltage |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the RMS-valued AC component of an applied DC signal. To read out the latest measurement result, use the :FETCh:AFAN:DCV command. Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and delivers a string, containing a floating point real number, representing the first measurement result value. The physical dimension is volt, RMS-valued (V_{rms}). |
| Example | :MEAS:CONT:AFAN:DCVoltage? String delivered back: "0.025". |

:MEAS[:CONT]:AFAN:FREquency

| | |
|--------------------|---|
| Syntax | :MEAS[:CONT]:AFAN:FREquency |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the frequency of the audio signal applied to the audio analyzer. To read out the latest measurement result, use the :FETCh:AFAN:FREQ command. Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and delivers a string, containing a floating point real number, representing the first measurement result value. The physical dimension is Hertz (Hz). |
| Example | :MEAS:CONT:AFAN:FREquency? String delivered back: "1000.0". |

:MEAS[:CONT]:AFAN:DIStortion

| | |
|--------------------|--|
| Syntax | :MEAS[:CONT]:AFAN:DIStortion |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the third-harmonic distortion of a sine wave applied to the audio analyzer. To read out the latest measurement result, use the :FETCh:AFAN:DISt command. Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and delivers a string, containing a floating point real number, representing the first measurement result value. The physical dimension is percent (%). |
| Example | :MEAS:CONT:AFAN:DIStortion? String delivered back: "0.3". |

:MEASure[:CONT]:AFANalyser:SIAd

| | |
|--------------------|--|
| Syntax | :MEASure[:CONT]:AFANalyser:SIAd |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the SINAD (i.e. signal to noise ratio). To read out the latest measurement result, use the :FETCh:AFAN:SIAd command. Note: Any AF test command needs the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and delivers a string, containing one floating point real number, representing the first measurement result value. The physical dimension is dB. |
| Example | :MEAS:CONT:AFANalyser:SIAd? Value delivered back: "50.5". |

:MEASure[:CONTInuous]:PSUPply:GRoup

| | |
|--------------------|--|
| Syntax | :MEASure[:CONTInuous]:PSUPply:GRoup |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the power supply tests specified with the :CONF:MEAS:GRO:PSUP command. To read out the latest measurement results, use the :FETCh:PSUP:GRoup command. Notes <ul style="list-style-type: none"> - To perform any power and current consumption measurements on your Willtek 4400, the MS Power Supply and Current Measurements Option must be installed. - For further details regarding group measurements, please refer to section :MEASure:...:GRoup. |

| | |
|----------------|---|
| Query | The query form of this command will start the measurements and deliver back a string containing the latest set of measurement result values. All measurement result values returned will be floating point real numbers. The order of the measurement result values returned is as described below (see command :MEAS:PSUP:ALL). The single measurement result values are separated by commas. |
| Example | :CONF:MEAS:GRO:PSUP ACUR,APOW :MEAS:PSUP:GRO? In this example, the group of measurements is defined by a power consumption measurement combined with a current consumption measurement. The measurement result string returned in this example is: "863.6,304.2". Because of the internal order, the first measurement result value delivered back is the power consumption, the second one the average current consumption measurement. |

:MEASure[:CONTInuous]:PSUPply:ALL

| | |
|--------------------|---|
| Syntax | :MEASure[:CONTInuous]:PSUPply:ALL |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the power and current consumption tests. To read out the latest measurement results, use the :FETCh:PSUP:ALL? query. The power supply/current measurement tests performed by this command are: APOW for the average power consumption measurement, measured in mW; ACUR for the average current consumption measurement in mA, PCUR for the peak current consumption in mA. Notes <ul style="list-style-type: none"> - Any power and current consumption measurement will need the MS Power Supply and the Current Measurement Options to be installed on your Willtek 4400. - For further details regarding group measurements, please refer to section :MEASure:...:GROup. |
| Query | The query form of this command starts the measurements and - after all measurements have been completed and all measurement results obtained - delivers back a string containing three floating point real numbers, representing the three measurement result values. The order of these measurement result values delivered back is as follows: <ul style="list-style-type: none"> - APOW, representing the average power consumption in mW, - ACUR, representing the average current consumption in mA, - PCUR, representing the peak current consumption in mA. |
| Example | :MEAS:PSUP:ALL? In this case, all power/current consumption measurements are performed in one go. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing three measurement result values, for example "863.6,304.2,1352.9". |

:MEASure[:CONTInuous]:PSUPply:APOWer

| | |
|-------------------|--------------------------------------|
| Syntax | :MEASure[:CONTInuous]:PSUPply:APOWer |
| Parameters | There are no parameters. |

| | |
|--------------------|---|
| Description | Starts a continuous measurement of the power consumption test. To read out the latest measurement result, use the <code>:FETCh:PSUP:APOW?</code> query. Note: Any power and current consumption measurement will need the MS Power Supply and the Current Measurement Options to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurements and - after all measurements have been completed and all measurement results obtained - delivers back a string containing the measurement result value. The value represents the average power consumption in mW. |
| Example | <code>:MEAS:PSUP:APOW?</code> In this case, all the average power consumption measurements is performed. As soon as the measurement has been completed, a string will be delivered back containing the measurement result value, for example "863.6". |

:MEASure[:CONTInuous]:PSUPply:ACURrent

| | |
|--------------------|---|
| Syntax | <code>:MEASure[:CONTInuous]:PSUPply:ACURrent</code> |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the average current consumption test. To read out the latest measurement result, use the <code>:FETCh:PSUP:ACUR</code> command. Any power and current consumption measurement will need the MS Power Supply and the Current Measurement Options to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and - after the measurement has been completed and the measurement result obtained - delivers back a string containing a floating point real number representing the average current consumption in mA. |
| Example | <code>:MEAS:PSUP:ACUR?</code> In this case, the average current consumption measurement is performed. As soon as the measurement has been completed, a string will be delivered back containing the measurement result value, for example "304.2,". |

:MEASure[:CONTInuous]:PSUPply:PCURrent

| | |
|--------------------|--|
| Syntax | <code>:MEASure[:CONTInuous]:PSUPply:PCURrent</code> |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the peak current consumption test. To read out the latest measurement result, use the <code>:FETCh:PSUP:PCUR</code> command. Any power and current consumption measurement will need the MS Power Supply and the Current Measurement Options to be installed on your Willtek 4400. |
| Query | The query form of this command starts the measurement and - after the measurement has been completed and the measurement result obtained - delivers back a string containing a floating point real number representing the peak current consumption in mA. |
| Example | <code>:MEAS:PSUP:PCUR?</code> In this case, all the peak current consumption measurement is performed. As soon as the measurement has been completed, a string will be delivered back containing the measurement result value, for example "1352.9". |

:MEASure:ARRay:AFANalyser:GRoup

| | |
|--------------------|--|
| Syntax | :MEASure:ARRay:AFANalyser:GRoup <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | Performs the audio tests, specified with the :CONF:MEAS:GRO:AFAN command for a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:GRoup command. Notes: <ul style="list-style-type: none"> - Please keep in mind that the start of a new AF test will always terminate all other measurements (see section "The MEASure subsystem" on page 191 for details). - To perform any audio measurements on your Willtek 4400, the Audio Option must be installed. - For further details regarding group measurements, please refer to section ":MEASure:...:GRoup" on page 198. - More information regarding ARRay measurements can be found in section "MEASure:ARRay" on page 194. |
| Query | The query form of this command will start the sequence of audio measurements as specified with the :CONF:MEAS:GRO:AFAN command for a specific number of times (set with the int1 parameter). As soon as all measurements have been completed, a string will be delivered back. It will contain floating point real numbers, representing the measurement result values. The order of the measurement result values returned is as described below (see command :MEAS[:GSM]:ARR:AFAN:ALL). The single measurement result values are separated by commas. Note: The number of measurement result values delivered back is the number of measurement result values of a single test run multiplied with the int1 parameter. |
| Example | :CONF:MEAS:GRO:AFAN SIN,FREQ :MEASure:ARRay:AFAN:GRoup? 3 In this example, the group of measurements is defined by a SINAD measurement, combined with an AF frequency measurement. This group of measurements will be carried out three times. After all measurements have been completed, a result string will be delivered back, containing six measurement result values. The measurement result values returned are: "1000.0,50.5,1000.1,50.1,999.9,50.6". Because of the internal order, the first (third and fifth) measurement result value delivered back is the audio frequency, the second (fourth and sixth) one the SINAD. |

:MEASure:ARRay:AFANalyser:ALL

| | |
|-------------------|---|
| Syntax | :MEASure:ARRay:AFANalyser:ALL <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |

| | |
|--------------------|---|
| Description | <p>Performs a standard sequence of the most important audio tests a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:ALL command. The tests performed by this command are:</p> <p>ACVPeakp, the peak-to-peak measurement of an AC voltage, ACVRms, the RMS-valued measurement of an AC voltage, DCVRms is used to measure AC ripple on DC lines (this measurement gives the root-mean square voltage of the AC component of the applied DC signal). FREQuency is the measurement of the audio frequency in Hertz, DISTortion is the distortion measurement on the third harmonic of a sine wave and expressed in %, while SINad is the measurement of the signal to noise ratio, expressed in dB.</p> <p>Notes:</p> <ul style="list-style-type: none"> - Please keep in mind that the start of a new AF test will always terminate all other measurements (see section "The MEASure subsystem" on page 191 for details). - Any AF measurement will need the Audio Option to be installed on your Willtek 4400. - For further details regarding group measurements, please refer to section ":MEASure:....:GROup" on page 198. |
| Query | <p>The query form of this command performs the measurements and - after all measurements have been completed and all measurement results obtained - delivers a string, containing (6* int1) measurement result values. The single measurement result values are separated by commas. All measurement result values are floating point real numbers. The order of these measurement result values delivered back is as follows:</p> <ol style="list-style-type: none"> 1. ACVPeakp, representing the AC peak-to-peak voltage of the AF signal, 2. ACVRms, representing the RMS-valued AC voltage of the AF signal, 3. DCVRms, representing the RMS-valued AC voltage on an applied DC signal, 4. FREQuency, representing the audio frequency, 5. DISTortion, representing the third-harmonic distortion of the applied sine-wave AF signal and, 6. SINad, representing the signal to noise ratio of the applied AF signal. |
| Example | <p>:MEASure:ARRay:AFANalyser:ALL? 2</p> <p>In this case, the sequence of standard AF measurements will be performed twice. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing twelve measurement result values:</p> <p>"0.7,0.25,0.0,1000.0,0.3,50.5, 0.75,0.3,0.0,1000.0,0.25,50.9"</p> |

:MEAS:ARRay:AFAN:ACVoltage:PEAKp

| | |
|--------------------|--|
| Syntax | :MEAS:ARRay:AFAN:ACVoltage:PEAKp <int1> |
| Parameters | <p>int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0.</p> |
| Description | <p>Performs the measurement of the AC peak-to-peak voltage of the AF signal applied to the audio analyzer a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:ACV:PEAK command.</p> <p>Note: Any AF test command will need the Audio Option to be installed on your Willtek 4400.</p> |

| | |
|----------------|---|
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain int1 floating point real numbers. The physical dimension of the measurement result values is volt, measured peak-to-peak (V_{pp}). The single measurement results are separated by commas. |
| Example | :MEAS:ARR:AFANalyser:ACVoltage:PEAKp? 5 The string returned is: "0.7,0.8,0.75,0.71,0.79". |

:MEAS:ARR:AFAN:DCVoltage

| | |
|--------------------|---|
| Syntax | :MEAS:ARR:AFAN:DCVoltage <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | Performs the measurement of the RMS-valued AC component of a DC signal applied to the audio analyser a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:DCV command. Note: Any AF test command will need the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain int1 floating point real numbers. The physical dimension of the measurement result values is volt, RMS-valued (V_{rms}). The single measurement results are separated by commas. |
| Example | :MEAS:ARR:AFANalyser:DCVoltage? 5 The string returned is: "0.025,0.029,0.019,0.030,0.025". |

:MEASure:ARRay:AFAN:FREQuency

| | |
|--------------------|--|
| Syntax | :MEASure:ARRay:AFAN:FREQuency <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | Performs the measurement of the frequency of the audio signal applied to the audio analyzer a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:FREQ command. Note: Any AF test command will need the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain int1 floating point real numbers. The physical dimension of the measurement result values is Hertz. The single measurement results are separated by commas. |
| Example | :MEASure:ARRay:AFANalyser:FREQ? 5 The string returned is: "1000.0,1000.5,1000.9,999.9,1000.0". |

:MEASure:ARRay:AFAN:DISToRTion

| | |
|--------------------|--|
| Syntax | :MEASure:ARRay:AFAN:DISToRTion <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | Performs the measurement of the distortion of a sine wave applied to the audio analyzer a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:DIS command. Note: Any AF test command will need the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain int1 floating point real numbers. The physical dimension of the measurement result values is percentage (%). The single measurement results are separated by commas. |
| Example | :MEASure:ARRay:AFANalyser:DIS? 5 The string returned is: "0.5,0.7,1.1,1.2,0.9". |

:MEASure:ARRay:AFANalyser:SINad

| | |
|--------------------|--|
| Syntax | :MEASure:ARRay:AFANalyser:SINad <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | Performs the measurement of the SINAD (i.e. signal to noise ratio) a specific number of times (set with the int1 parameter). To read out the entire measurement results array, use the :FETCh:AFAN:SIN command. Note: Any AF test command will need the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain int1 floating point real numbers. The physical dimension of the measurement result values is dB. The single measurement results are separated by commas. |
| Example | :MEASure:ARRay:AFANalyser:SINad? 5 The string returned is: "50.5,50.0,49.1,51.2,50.9". |

:MEASure:ARRay:PSUPply:GRoup

| | |
|-------------------|---|
| Syntax | :MEASure:ARRay:PSUPply:GRoup <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |

| | |
|--------------------|---|
| Description | <p>Performs the power supply measurements specified with the <code>:CONF:MEAS:GRO:PSUP</code> command for a specific number of times (set with the <code>int1</code> parameter). To read out the entire measurement results array, use the <code>:FETCh:PSUP:GROup</code> command.</p> <p>Notes</p> <ul style="list-style-type: none"> - To perform any power and current consumption measurements on your Willtek 4400, the MS Power Supply and Current Measurements Option must be installed. - For further details regarding group measurements, please refer to section ":MEASure:....:GROup" on page 198. |
| Query | <p>The query form of this command will start the sequence of power/current consumption measurements as specified with the <code>:CONF:MEAS:GRO:PSUP</code> command for a specific number of times (set with the <code>int1</code> parameter). The order of the measurement result values returned is as described below (see command <code>:MEAS:PSUP:ALL</code>).</p> |
| Example | <p><code>:CONF:MEAS:GRO:PSUP ACUR,APOW</code> <code>:MEASure:ARRay:PSUP:GROup?</code></p> <p>In this example, the group of measurements is defined by a power consumption measurement combined with a current consumption measurement. The measurement result string returned is: "863.6,304.2".</p> <p>Because of the internal order, the first measurement result value delivered back is the power consumption, the second one the average current consumption measurement.</p> |

:MEASure:ARRay:PSUPply:ALL

| | |
|--------------------|---|
| Syntax | <code>:MEASure:ARRay:PSUPply:ALL <int1></code> |
| Parameters | <p><code>int1</code> is an integer.</p> <p>The minimum value for <code>int1</code> is 0, the maximum value is 100. The default value for <code>int1</code> is 0.</p> |
| Description | <p>Performs all measurements of the power and current consumption tests for a specific number of times (set with the <code>int1</code> parameter). To read out the latest measurement results, use the <code>:FETCh:PSUP:ALL</code> command.</p> <p>The power supply/current measurement tests performed by this command are:</p> <ul style="list-style-type: none"> APOW for the average power consumption measurement, measured in mW; ACUR for the average current consumption measurement in mA, PCUR for the peak current consumption in mA. <p>Notes</p> <ul style="list-style-type: none"> - Any power and current consumption measurement will need the MS Power Supply and the Current Measurement Options to be installed on your Willtek 4400. - For further details regarding group measurements, please refer to section ":MEASure:....:GROup" on page 198. |
| Query | <p>The query form of this command starts the measurements for a specific number of times (set with the <code>int1</code> parameter). The order of these measurement result values delivered back is as follows:</p> <ul style="list-style-type: none"> - APOW, representing the average power consumption in mW, - ACUR, representing the average current consumption in mA, - PCUR, representing the peak current consumption in mA. |

Example :**MEASure:ARRay:PSUP:ALL? 10**
 In this case, all power/current consumption measurements are performed ten times. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing ten times three measurement result values.

:MEASure:ARRay:PSUPply:APower

Syntax :**MEASure:ARRay:PSUPply:APower** <int1>
Parameters **int1** is an integer.
 The minimum value for **int1** is 0, the maximum value is 100. The default value for **int1** is 0.
Description Performs the power consumption measurement for a specific number of times (set with the **int1** parameter). To read out the latest measurement results, use the **:FETCh:PSUP:APow** command.
Query The query form of this command starts the measurements for a specific number of times (set with the **int1** parameter). The returned string contains **int1** floating point values representing the average power consumption in mW.
Example :**MEASure:ARRay:PSUP:APow? 3**
 In this case, the power consumption measurement is performed three times. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing the three measurement result values.

:MEASure:ARRay:PSUPply:ACurrent

Syntax :**MEASure:ARRay:PSUPply:ACurrent** <int1>
Parameters **int1** is an integer.
 The minimum value for **int1** is 0, the maximum value is 100. The default value for **int1** is 0.
Description Performs the average current consumption measurement for a specific number of times (set with the **int1** parameter). To read out the latest measurement results, use the **:FETCh:PSUP:ACUR** command.
Query The query form of this command starts the measurements for a specific number of times (set with the **int1** parameter). The returned string contains **int1** floating point values representing the average current consumption in mA.
Example :**MEASure:ARRay:PSUP:ACUR? 3**
 In this case, the current consumption measurement is performed three times. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing the three measurement result values.

:MEASure:ARRay:PSUPply:PCurrent

Syntax :**MEASure:ARRay:PSUPply:PCurrent** <int1>
Parameters **int1** is an integer.
 The minimum value for **int1** is 0, the maximum value is 100. The default value for **int1** is 0.

| | |
|--------------------|--|
| Description | Performs the peak current consumption measurement for a specific number of times (set with the int1 parameter). To read out the latest measurement results, use the :FETCh:PSUP:PCUR command. |
| Query | The query form of this command starts the measurements for a specific number of times (set with the int1 parameter). The returned string contains int1 floating point values representing the peak current consumption in mA. |
| Example | :MEASure:ARRay:PSUP:PCUR? 3 In this case, the peak current consumption measurement is performed three times. As soon as all measurements have been completed and all measurement results obtained, a string will be delivered back containing the three measurement result values. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:FREQuency

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:FREQuency |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the frequency error. |
| Query | The query form of this command will start the sequence of frequency error measurements. |
| Example | :MEAS:TDSC:RFTX:FREQ? Returns the measured frequency error, e.g. -7.834 . |

:MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:MEAN

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:MEAN |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the mobile's RF power. |
| Query | The query form of this command will return the float value for power. |
| Example | :MEASure:TDSCdma:RFTX:POWer:MEAN? Starts an RF power measurement and returns the result (in dBm), e.g. -45.3 . |

:MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:PEAK

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:PEAK |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the Peak RF power. |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFTX:POWer:PEAK? Starts the Peak RF power measurement and returns the result in a string. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:OFF

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:OFF |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the OFF RF power. |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFTX:POWer:OFF? Starts the OFF RF power measurement and returns the result in a string. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:ONOFFpower[:DATA]

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:ONOFFpower[:DATA] |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the PTT RF power. |
| Query | The query form of this command will deliver a string, containing the measurement result values separated by commas. |
| Example | :MEASure:TDSCdma:RFTX:POWer:ONOFFpower[:DATA]? Starts the RF PTT measurement and returns the result in a string. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:ONOFFpower:AVG

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:POWer:ONOFFpower:AVG <Int1> |
| Parameters | Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1. |
| Description | Number of applied averages. |
| Query | The query form of this command will delivers a string, containing the measurement result values separated by commas. |
| Example | :MEASure:TDSCdma:RFTX:POWer:ONOFFpower:AVG? Starts the RF power measurement and returns an average of the result in a string. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:ALL

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:ALL |
| Parameters | There are no parameters. |
| Description | Starts a continuous measurement of the most important RF TX Modulation Quality tests. |

| | |
|----------------|--|
| Query | <p>The query form of this command starts the measurements and – after all 10 measurements have been completed and all measurement results obtained – delivers a string, containing 10 measurement result values, separated by commas. The order and type of these measurement result values delivered is as follows:</p> <ol style="list-style-type: none"> 1. <i>EVM RMS</i>, floating point real number representing the RMS vector error measurement in percent, 2. <i>EVM Peak</i>, floating point real number representing the peak vector error measurement in percent, 3. <i>Magnitude error RMS</i>, floating point real number representing the RMS magnitude vector error measurement in percent, 4. <i>Magnitude error Peak</i>, floating point real number representing the PEAK magnitude vector error measurement in percent, 5. <i>Phase error RMS</i>, floating point real number representing the RMS phase vector error measurement in degree, 6. <i>Phase error Peak</i>, floating point real number representing the PEAK phase vector error measurement in degree 7. <i>Frequency error</i>, floating point real number representing the mobile's frequency error, 8. <i>RHO</i>, floating point real number representing the mobile's modulation quality, 9. <i>I/Q Offset</i>, a floating point value with the result in dBc, representing the result of the origin offset vector error measurement, 10. <i>I/Q Imbalance</i>, floating point real number representing the result of the IQ imbalance vector error measurement with the result in dB. Note For a further description of the single measurements, see description of the related commands below. |
| Example | <pre>:MEAS:TDSC:RFTX:MODQ:ALL?</pre> <p>In this case, all relevant RF TX measurements will be performed in a sequence, e.g. 21.624, 72.8, 15.335, 72.736, 6.738, 27.512, -6.378, 0.961, -28.251, -43.743.</p> |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:ERMS

| | |
|--------------------|--|
| Syntax | <code>:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:ERMS</code> |
| Parameters | There are no parameters. |
| Description | Starts a continuous RMS vector error magnitude measurement. |
| Query | The query form of this command will return a floating point value; the unit is percent. |
| Example | <pre>:MEAS:TDSC:RFTX:MODQ:ERMS?</pre> <p>Returns the measured EVM RMS vector error, e.g. 21.624.</p> |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:EPEAK

| | |
|--------------------|---|
| Syntax | <code>:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:EPEAK</code> |
| Parameters | There are no parameters. |
| Description | Starts a continuous peak vector error magnitude measurement. |
| Query | The query form of this command will return a floating point value in percent. |
| Example | <pre>:MEAS:TDSC:RFTX:MODQ:EPEA?</pre> <p>Returns the measured EVM peak vector error, e.g. 72.8.</p> |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:MRMS

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:MRMS |
| Parameters | There are no parameters. |
| Description | Starts a continuous RMS magnitude measurement. |
| Query | The query form of this command will return a floating point value; the unit is percent. |
| Example | :MEAS:TDSC:RFTX:MODQ:MRMS? Returns the measured magnitude RMS vector error, e.g. 15.335. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:MPEAK

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:MPEAK |
| Parameters | There are no parameters. |
| Description | Starts a continuous peak magnitude measurement. |
| Query | The query form of this command will return a floating point value in percent. |
| Example | :MEAS:TDSC:RFTX:MODQ:MPEA? Returns the measured magnitude peak vector error, e.g. 72.736. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:PRMS

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:PRMS |
| Parameters | There are no parameters. |
| Description | Starts a continuous peak magnitude measurement. |
| Query | The query form of this command will return a floating point value in percent. |
| Example | :MEAS:TDSC:RFTX:MODQ:PRMS? Returns the measured RMS phase vector error, e.g. 6.738. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:PPEAK

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:PPEAK |
| Parameters | There are no parameters. |
| Description | Starts a continuous peak phase error measurement. |
| Query | The query form of this command will return a floating point value in percent. |
| Example | :MEAS:TDSC:RFTX:MODQ:PPEA? Returns the measured phase peak vector error, e.g. 27.512. |

:MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:RHO

| | |
|---------------|---|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFTX:MODQuality:RHO |
|---------------|---|

| | |
|--------------------|---|
| Parameters | There are no parameters. |
| Description | Starts a continuous closed loop measurement. |
| Query | The query form of this command will return a floating point value. |
| Example | :MEAS:TDSC:RFTX:MODQ:RHO? Returns the modulation quality, e.g. 0.9989. |

**:MEASure:TDSCdma[:CONTinuous]:RFTX:MODQuality
:IQOffset**

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTinuous]:RFTX:MODQuality:IQOffset |
| Parameters | There are no parameters. |
| Description | Starts a continuous IQ origin offset vector error measurement. |
| Query | The query form of this command will return a floating point value with the result in dBc. |
| Example | :MEAS:TDSC:RFTX:MODQ:IQOF? Returns the origin offset in dBc, e.g. -28.251. |

**:MEASure:TDSCdma[:CONTinuous]:RFTX:MODQuality
:IQImbalance**

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTinuous]:RFTX:MODQuality :IQImbalance |
| Parameters | There are no parameters. |
| Description | Starts a continuous IQ imbalance vector error measurement. |
| Query | The query form of this command will return a floating point value with the result in dB. |
| Example | :MEAS:TDSC:RFTX:MODQ:IQIM? Returns the IQ imbalance in dBc, e.g. -43.743. |

:MEASure:TDSCdma[:CONTinuous]:RFSpectrum:OBW

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma[:CONTinuous]:RFSpectrum:OBW? |
| Parameters | There are no parameters. |
| Description | Starts a continuous OBW measurement |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFSpectrum:OBW? Starts the OBW measurement and returns the value. |

:MEASure:TDSCdma[:CONTInuous]:RFSpectrum:OBW:AVG

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCDMA[:CONTInuous]:RFSpectrum:OBW:AVG <Int1> |
| Parameters | Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1. |
| Description | Starts an OBW measurement. |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFSpectrum:OBW:AVG? 10 Starts the OBW measurement and returns the average value from 10 OBW measurements. |

:MEASure:TDSCdma:RFSpectrum:CPOwer

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:RFSpectrum:CPOwer? |
| Parameters | There are no parameters. |
| Description | Starts a continuous CPower measurement. |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFSpectrum:CPOwer? Starts the CHPower measurement and returns the value. |

:MEASure:TDSCdma:RFSpectrum:CPOwer:AVG

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:RFSpectrum:CPOwer:AVG <Int1> |
| Parameters | Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1. |
| Description | Starts a CHPower measurement. |
| Query | The query form of this command will return the float value. |
| Example | :MEASure:TDSCdma:RFSpectrum:CPOwer:AVG? 10 Starts the OBW measurement and returns the average value from 10 CHPower measurements. |

:MEASure:TDSCdma[:CONTInuous]:RFSpectrum:MSpectrum[:DATA]

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma[:CONTInuous]:RFSpectrum:MSpectrum[:DATA]? |
| Parameters | There are no parameters. |
| Description | Starts a continuous spectrum measurement. |
| Query | The query form of this command will delivers a string, containing the measurement result values separated by commas. |

Example :MEASure:TDSCdma:RFSpectrum:MSpectrum[:DATA]?
 Starts the spectrum measurement and returns the values.

:MEASure:TDSCDMA:RFSpectrum:MSpectrum:AVG

Syntax :MEASure:TDSCdma:RFSpectrum:MSpectrum:AVG <Int1>
Parameters Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1.
Description Starts a spectrum measurement.
Query The query form of this command delivers a string containing the measurement average result values separated by commas.
Example :MEASure:TDSCdma:RFSpectrum:MSpectrum:AVG? 10
 Starts the spectrum measurement and returns the average values from 10 spectrum measurements.

:MEAS:TDSCdma[:CONTInuous]:RFSpectrum:ACLR

Syntax :MEAS:TDSCdma[:CONTInuous]:RFSpectrum:ACLR
Parameters There are no parameters.
Description Starts a continuous measurement of the ACLR modulation spectrum.
Query The query form of this command starts the measurement and delivers a string containing 5 measurement result values separated by commas.
Example :MEAS:TDSC:RFSP:ACLR

:MEASure:TDSCdma:RFSpectrum:ACLR:AVG

Syntax :MEASure:TDSCdma:RFSpectrum:ACLR:AVG <Int1>
Parameters Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1.
Description Starts an ACLR measurement.
Query The query form of this command delivers a string containing the measurement average result values separated by commas.
Example :MEASure:TDSCdma:RFSpectrum:ACLR:AVG? 10
 Starts the ACLR measurement and returns the average values from 10 ACLR measurements.

:MEAS:TDSCdma[:CONTInuous]:RFSpectrum:SEM

Syntax :MEAS:TDSCdma[:CONTInuous]:RFSpectrum:SEM
Parameters There are no parameters.
Description Starts a continuous measurement of the SEM modulation spectrum.

| | |
|----------------|--|
| Query | The query form of this command starts the measurement and delivers a string, containing the measurement result values separated by commas. |
| Example | :MEAS:TDSC:RFSP:SEM? |

:MEASure:TDSCdma:RFSpectrum:SEM:AVG

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:RFSpectrum:SEM:AVG <Int1> |
| Parameters | Int1 is an integer. The minimum value is 1, the maximum value is 100. The default value is 1. |
| Description | Starts a SEM measurement. |
| Query | The query form of this command delivers a string, containing the measurement average result values separated by commas. |
| Example | :MEASure:TDSCdma:RFSpectrum:SEM:AVG? 10 Starts the SEM measurement and returns the average values from 10 SEM measurements. |

:MEASure:TDSCdma:ARRay:RFTX:FREQuency

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:FREQuency <Int1> |
| Parameters | Int1 is an integer. The minimum value for Int1 is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the frequency error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:FREQ? 5 Starts five measurements, the results of which are returned. Example: 47.1, 91.5, 13.6, -4.8, 15.3 |

:MEASure:TDSCdma:ARRay:RFTX:POWer:MEAN

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCDMA:ARRay:RFTX:POWer:MEAN <Int1> |
| Parameters | Int1 is an integer. The minimum value for Int1 is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the measurement of the mean power as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:POW:MEAN? 10 |

:MEASure:TDSCdma:ARRay:RFTX:POWer:PEAK

| | |
|---------------|---|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:POWer:PEAK <Int1> |
|---------------|---|

| | |
|--------------------|--|
| Parameters | Int1 is an integer. The minimum value for Int1 is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the measurement of the mobile's peak RF power as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the measurement results. |
| Example | :MEAS:TDSC:ARR:RFTX:POW:PEAK? 10 |

:MEASure:TDSCdma:ARRay:RFTX:POWer:OFF

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:POWer:OFF <Int1> |
| Parameters | Int1 is an integer. The minimum value for Int1 is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the measurement of the mobile's Off RF power as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the measurement results. |
| Example | :MEAS:TDSC:ARR:RFTX:POW:OFF? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:ALL

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:ALL <Int1> |
| Parameters | There are no parameters. |
| Description | Starts an array measurement of the most important RF TX Modulation Quality tests as many times as specified. |

| | |
|----------------|---|
| Query | <p>The query form of this command starts the measurements and - after all 10 measurements have been completed and all <Int1> measurement results obtained - delivers a string, containing 10 times <Int1> measurement result values, separated by commas. The order and type of these measurement result values delivered is as follows:</p> <ol style="list-style-type: none"> 1. <i>EVM RMS</i>, floating point real number representing the RMS vector error measurement in percent, 2. <i>EVM Peak</i>, floating point real number representing the peak vector error measurement in percent, 3. <i>Magnitude error RMS</i>, floating point real number representing the RMS magnitude vector error measurement in percent, 4. <i>Magnitude error Peak</i>, floating point real number representing the PEAK magnitude vector error measurement in percent, 5. <i>Phase error RMS</i>, floating point real number representing the RMS phase vector error measurement in degree, 6. <i>Phase error Peak</i>, floating point real number representing the PEAK phase vector error measurement in degree, 7. <i>Frequency error</i>, floating point real number representing the mobile's frequency error, 8. <i>RHO</i>, floating point real number representing the mobile's modulation quality, 9. <i>I/Q Offset</i>, a floating point value with the result in dBc, representing the result of the origin offset vector error measurement, 10. <i>I/Q Imbalance</i>, floating point real number representing the result of the IQ imbalance vector error measurement with the result in dB. Note: For a further description of the single measurements, see description of the related commands below. |
| Example | <pre>:MEAS:TDSC:ARR:RFTX:MODQ:ALL? 5</pre> <p>In this case, all relevant RF TX measurements will be performed in a sequence.</p> |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:ERMS

| | |
|--------------------|--|
| Syntax | <code>:MEASure:TDSCdma:ARRay:RFTX:MODQuality:ERMS <Int1></code> |
| Parameters | <i>Int1</i> is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs an RMS vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | <pre>:MEAS:TDSC:ARR:RFTX:MODQ:ERMS? 19</pre> |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:EPEAK

| | |
|--------------------|--|
| Syntax | <code>:MEASure:TDSCdma:ARRay:RFTX:MODQuality:EPEAK <Int1></code> |
| Parameters | <i>Int1</i> is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs a peak vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | <pre>:MEAS:TDSC:ARR:RFTX:MODQ:EPEA? 10</pre> |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:MRMS

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:MRMS <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs an RMS magnitude vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:MRMS? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:MPEAK

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:MPEAK <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs a peak magnitude vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:MPEA? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:PHASe:PRMS

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:PHASe:PRMS <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs an RMS phase vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:PRMS? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:PPEAK

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:PPEAK <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs a peak phase vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:PPEA? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:RHO

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:RHO <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Starts a continuous closed loop measurement. |
| Query | The query form of this command will return a floating point value. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:RHO? 5 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:IQOffset

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:IQOffset <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs an origin offset vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:IQOF? 10 |

:MEASure:TDSCdma:ARRay:RFTX:MODQuality:IQImbalance

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFTX:MODQuality:IQImbalance <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs an IQ imbalance vector error measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the results. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:IQIM? 10 |

:MEASure:TDSCdma:ARRay:RFSPectrum:OBW

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFSPectrum:OBW <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the OBW measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the measurement results. |
| Example | :MEAS:TDSC:ARR:RFSP:OBW? 10 |

:MEASure:TDSCdma:ARRay:RFSpectrum:CPOWer

| | |
|--------------------|--|
| Syntax | :MEASure:TDSCdma:ARRay:RFSpectrum:CPOWer <Int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the channel power measurement as many times as specified. |
| Query | The query form of this command will perform the measurement the specified number of times, and return the measurement results. |
| Example | :MEAS:TDSC:ARR:RFSP:CPOW? 10 |

:MEASure:TDSCdma:ARRay:RFSpectrum:ACLR

| | |
|--------------------|---|
| Syntax | :MEASure:TDSCdma:ARRay:RFSpectrum:ACLR <int1> |
| Parameters | Int1 is an integer. The minimum value is 0, the maximum value is 100. The default value is 0. |
| Description | Performs the measurement of the ACLR modulation spectrum a specific number of times (set with the int1 parameter). |
| Query | The query form of this command will perform the measurement the specified number of times (int1 parameter). As soon as all measurements have been completed, all measurement result values will be returned in a string. The string delivered back will contain (25 * int1) floating point real numbers. The physical dimension is dBc. The single measurement results are separated by commas. |
| Example | :MEAS:TDSC:ARR:RFSP:ACLR? 5 :FETCh:TDSCDMA:RFSP:ACLR:MOD The string returned in this example would contain 125 floating point real numbers, representing the 25 measurement result values of five measurement runs. |

FETCh Subsystem

The commands of this subsystem are used to read out measurement result values both from continuous and array measurements.

:FETCh:LAST?

| | |
|--------------------|--|
| Syntax | :FETCh:LAST? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns in a string the latest measurement result value(s) of the measurement started last. The number and format of the measurement result values returned is dependent on the type of measurement. |

Example :**MEAS : GSM : RFTX : PRMS**
 :**MEAS : GSM : RFRX : RBER : ALL**
 :**FETCh : LAST?**
 In this example, the **FETC : LAST** command will return the three residual bit error ratios (three floating point real numbers) as this is the measurement started last.

: FETCh : BLOCkdata : AFSPectrum?

Syntax :**FETCh : BLOCkdata : AFSPectrum?**

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query The query form of this command returns the latest data block describing the graph of the audio spectrum measurement. For further details regarding this measurement, please refer to the description of the **:MEAS : BLOC : AFSP** command and the explanations given in section **:MEASure : BLOCkdata**. There, you will also find all information about the number, type and order of the measurement result values returned in the result string.

Example :**MEAS : BLOC : AFSP**
 :**FETC : BLOC : AFSP?**
 The **FETC** command will return a data array. The number of measurement result values delivered back is dependent on the span and resolution settings of the audio spectrum analyzer.

: FETCh : AFANalyser : GROUp?

Syntax :**FETCh : AFANalyser : GROUp?**

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query The query form of this command will return the latest measurement result values of the sequence of AF measurements as specified with the **:CONF : MEAS : GRO : AFAN** command and measured with the **:MEAS [ARRay] : AFAN : GRO** command. The string returned contains the related measurement result values, separated by commas. All measurement result values are floating point real numbers. The number of measurement result values handed back depends on the number and type of measurements defined with the **:CONF : MEAS : GRO : AFAN** command. The order of the measurement result values within the result string is as described below for the **:FETCh : AFANalyser : ALL** command.

Example :**CONF : MEAS : GRO : AFAN SIN , FREQ**
 :**MEAS : AFAN : GRO**
 :**FETCh : AFAN : GRO?**
 In this example, the group of measurements is defined by a SINAD measurement combined with an audio frequency measurement. The values returned are: **"1000.0 , 53.5"**. Because of the internal order (see description of the command below), the first measurement result value delivered back is the audio frequency, the second one the SINAD.

:FETCh:AFANalyser:ALL?

| | |
|--------------------|---|
| Syntax | :FETCh:AFANalyser:ALL? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>The query form of this command will return the latest measurement result values of the :MEAS[:ARRay]:AFAN:ALL measurement in a string. All measurement result values of the AF measurements will be floating point real numbers, separated by commas.</p> <p>The order of the measurement result values within the result string is as follows:</p> <ol style="list-style-type: none"> 1. ACVPeakp, representing the AC peak-to-peak voltage of the AF signal, expressed in V(pp) 2. ACVRms, representing the rms-valued AC voltage of the AF signal, expressed in V(rms), 3. DCVRms, representing the rms-valued AC voltage on an applied DC signal, expressed in V(rms), 4. FREQuency, representing the audio frequency, expressed in Hertz, 5. DISTortion, representing the third-harmonic distortion of the applied sine-wave AF signal (expressed in percent), 6. SINad, representing the signal to noise ratio of the applied AF signal, expressed in dB. <p>Note: Any AF test will need the audio option to be installed on your Willtek 4400.</p> |
| Example | <p>:MEAS:ARR:AFAN:ALL 2 :FETCh:AFAN:ALL?</p> <p>In this example, the six most important AF measurements are performed twice in a sequence. The measurement result string in this example is: "3.2,2.7,0.1,1000.0,1.1,53.5, 2.7,1.7,0.08,1000.0,0.9,55.2".</p> |

:FETCh:AFAN:ACVoltage:PEAKp?

| | |
|--------------------|--|
| Syntax | :FETCh:AFAN:ACVoltage:PEAKp? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>The query form of this command will return the latest measurement result value(s) of the :MEAS[:ARRay]:AFAN:ACV:PEAKp measurement. The string delivered back will contain at least one floating point real number.</p> <p>Note: Any AF test will need the audio option to be installed on your Willtek 4400.</p> |
| Example | <p>:MEAS:ARR:AFAN:ACV:PEAK 5 :FETCh:AFAN:ACV:PEAK?</p> <p>The string delivered back is: "3.2,3.1,3.3,3.2,2.9".</p> |

:FETCh:AFAN:ACVoltage:RMS?

| | |
|--------------------|---|
| Syntax | :FETCh:AFAN:ACVoltage:RMS? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |

| | |
|----------------|---|
| Query | The query form of this command will return the latest measurement result value(s) of the :MEAS [:ARRay] :AFAN:ACV:RMS measurement. The string delivered back will contain at least one floating point real number. Note: Any AF test will need the audio option to be installed on your Willtek 4400. |
| Example | :MEAS:AFAN:ACV:RMS :FETC:AFAN:ACV:RMS? The string delivered back is: "1.7". |

:FETCh:AFAN:DCVoltage?

| | |
|--------------------|---|
| Syntax | :FETCh:AFAN:DCVoltage? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | The query form of this command will return the latest measurement result value(s) of the :MEAS [:ARRay] :AFAN:DCV measurement. The string delivered back will contain at least one floating point real number. Note: Any AF test will need the audio option to be installed on your Willtek 4400. |
| Example | :MEAS:AFAN:DCV :FETC:AFAN:DCV? The string delivered back is: "1.7". |

:FETCh:AFANalyser:FREQuency?

| | |
|--------------------|--|
| Syntax | :FETCh:AFANalyser:FREQuency? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | The query form of this command will return the latest measurement result value(s) of the :MEAS [:ARRay] :AFAN:FREQ measurement. The string delivered back will contain at least one floating point real number. Note: Any AF test will need the audio option to be installed on your Willtek 4400. |
| Example | :MEAS:ARR:AFAN:FREQ 5 :FETC:AFAN:FREQ? The string delivered back is: "1000.5,1497.0,2004.7,2491.2,3005.0". |

:FETCh:AFANalyser:DISToRTion?

| | |
|--------------------|--|
| Syntax | :FETCh:AFANalyser:DISToRTion? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | The query form of this command will return the latest measurement result value(s) of the :MEAS [:ARRay] :AFAN:DIST measurement. The string delivered back will contain at least one floating point real number. Note: Any AF test will need the audio option to be installed on your Willtek 4400. |

| | |
|----------------|---|
| Example | :MEAS:AFAN:DIST :FETC:AFAN:DIST? The string delivered back is: "1.5". |
|----------------|---|

:FETCh:AFANalyser:SINad?

| | |
|--------------------|--|
| Syntax | :FETCh:AFANalyser:SINad? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | The query form of this command will return the latest measurement result value(s) of the :MEAS[:ARRay]:AFAN:SINad measurement. The string delivered back will contain at least one floating point real number. Note: Any AF test will need the audio option to be installed on your Willtek 4400. |
| Example | :MEAS:ARR:AFAN:SIN 3 :FETC:AFAN:SIN? The string delivered back is: "53.5,55.2,54.2". |

:FETCh:PSUPply:GROup

| | |
|--------------------|--|
| Syntax | :FETCh:PSUPply:GROup? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This query returns the latest measurement result values of the sequence of power/ current consumption measurements as specified with the :CONF:MEAS:GRO:PSUP command and measured with the :MEAS[:ARRay]:PSUP:GRO command. The string returned contains the related measurement result values, separated by commas. All measurement result values are floating point real numbers. The number of measurement result values handed back depends on the number and type of measurements defined with the :CONF:MEAS:GRO:PSUP command. The order of the measurement result values within the result string is as described below for the :FETCh:PSUP:ALL command. |
| Example | :CONF:MEAS:GRO:PSUP ACUR,APOW :MEAS:PSUP:GRO :FETCh:PSUP:GRO? In this example, the group of measurements consists of the average current consumption and the average power consumption measurements. The values returned in this example are: "859.2,293.7" . Because of the internal order (see description of the command below), the first measurement result value delivered back is the power consumption (in mW), followed by the average current consumption (in mA). |

:FETCh:PSUPply:ALL

| | |
|-------------------|----------------------------|
| Syntax | :FETCh:PSUPply:ALL? |
| Parameters | There are no parameters. |

| | |
|--------------------|--|
| Description | There is solely a query form of this command available. |
| Query | This query returns the latest measurement result values of the :MEAS[:ARRAY]:PSUP:ALL measurement in a string. All measurement result values of the power and current consumption measurements are floating point real numbers, separated by commas. The order of the measurement result values within the result string is as follows: <ol style="list-style-type: none"> 1. APOW, representing the average power consumption (in mW), 2. ACUR, representing the average current consumption, expressed in mA, 3. PCUR, representing the peak current consumption, expressed in mA. Note: Any current measurement will need the MS Power Supply and Current Measurement options to be installed on your Willtek 4400. |
| Example | :MEAS:PSUP:ALL :FETCh:PSUP:ALL? Returns a string like "893.5,395.4,1256.2". |

:FETCh:PSUPply:APOWer

| | |
|--------------------|--|
| Syntax | :FETCh:PSUPply:APOWer? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This query returns the latest measurement result values of the :MEAS[:ARRAY]:PSUP:APOW measurement in a string. The measurement result value is a floating point real numbers representing the average current consumption, expressed in mA. Note: Any current measurement will need the MS Power Supply and Current Measurement options to be installed on your Willtek 4400. |
| Example | :MEAS:PSUP:APOW :FETCh:PSUP:APOW? Returns a string like "893.5" giving the average power consumption in mW. |

:FETCh:PSUPply:ACURrent

| | |
|--------------------|--|
| Syntax | :FETCh:PSUPply:ACURrent? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This query returns the latest measurement result values of the :MEAS[:ARRAY]:PSUP:ACUR measurement in a string. The measurement result value is a floating point real numbers representing the average current consumption, expressed in mA. Note: Any current measurement will need the MS Power Supply and Current Measurement options to be installed on your Willtek 4400. |
| Example | :MEAS:PSUP:ACUR :FETCh:PSUP:ACUR? Returns a string like "395.4" giving the average current consumption in mA. |

: FETCh : PSUPply : PCURrent

| | |
|--------------------|---|
| Syntax | : FETCh : PSUPply : PCURrent? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>This query returns the latest measurement result values of the :MEAS [:ARRay] :PSUP :ALL measurement in a string. All measurement result values of the power and current consumption measurements are floating point real numbers, separated by commas.</p> <p>The order of the measurement result values within the result string is as follows:</p> <ol style="list-style-type: none"> 1. APOW, representing the average power consumption (in mW), 2. ACUR, representing the average current consumption, expressed in mA, 3. PCUR, representing the peak current consumption, expressed in mA. <p>Note: Any current measurement will need the MS Power Supply and Current Measurement options to be installed on your Willtek 4400.</p> |
| Example | <p>:MEAS : PSUP : PCUR : FETCh : PSUP : PCUR? Returns a string like "1256.2" giving the peak current consumption in mA.</p> |

: FETCh : TDSCdma : RFTX : FREQ

| | |
|--------------------|---|
| Syntax | : FETCh : TDSCdma : RFTX : FREQ? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | <p>The command returns the result of a previous :MEASure : TDSCdma : CONT ARRy : RFTX : FREQ measurement command.</p> |
| Example | <p>: FETC : TDSC : RFTX : FREQ? Returns the measured frequency error, e.g. -7.834.</p> |

: FETCh : TDSCdma : RFTX : POWer : MEAN

| | |
|--------------------|---|
| Syntax | : FETCh : TDSCdma : RFTX : POWer : MEAN? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | <p>The command returns the results from the :MEASure : TDSCdma : CONTinuous ARRy : RFTX : POWer : MEAN measurement command. The measurement must be started before send this command.</p> |
| Example | : FETC : TDSC : RFTX : POW : MEAN? |

: FETCh : TDSCdma : RFTX : POWer : PEAK

| | |
|-------------------|--|
| Syntax | : FETCh : TDSCdma : RFTX : POWer : PEAK? |
| Parameters | There are no parameters. |

| | |
|--------------------|--|
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma :CONTinuous ARRay:RFTX:POWer:PEAK measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFTX:POWer:PEAK? |

:FETCh:TDSCdma:RFTX:POWer:OFF

| | |
|--------------------|---|
| Syntax | :FETCh:TDSCdma:RFTX:POWer:OFF? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma :CONTinuous ARRay:RFTX:POWer:OFF measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFTX:POWer:OFF? |

:FETCh:TDSCdma:RFTX:POWer:ONOFFpower

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFTX:POWer:ONOFFpower? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma :CONTinuous:RFTX:POWer:ONOFFpower measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFTX:POWer:ONOFFpower? |

:FETCh:TDSCdma:RFTX:MODQuality:ALL?

| | |
|--------------------|-------------------------------------|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:ALL? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |

| | |
|----------------|--|
| Query | <p>The command returns the result of a previous <code>:MEASure:TDSCdma:CONT ARR:RFTX:ALL</code> measurement command. It delivers a string containing 10 measurement result values, separated by commas. The order and type of these measurement result values delivered is as follows:</p> <ol style="list-style-type: none"> 1. <code>EVM RMS</code>, floating point real number representing the RMS vector error measurement in percent, 2. <code>EVM Peak</code>, floating point real number representing the peak vector error measurement in percent, 3. <code>Magnitude error RMS</code>, floating point real number representing the RMS magnitude vector error measurement in percent, 4. <code>Magnitude error Peak</code>, floating point real number representing the PEAK magnitude vector error measurement in percent, 5. <code>Phase error RMS</code>, floating point real number representing the RMS phase vector error measurement in degree, 6. <code>Phase error Peak</code>, floating point real number representing the PEAK phase vector error measurement in degree, 7. <code>Frequency error</code>, floating point real number representing the mobile's frequency error, 8. <code>RHO</code>, floating point real number representing the mobile's modulation quality, 9. <code>I/Q Offset</code>, a floating point value with the result in dBc, representing the result of the origin offset vector error measurement, 10. <code>I/Q Imbalance</code>, floating point real number representing the result of the IQ imbalance vector error measurement with the result in dB. <p>Note: For a further description of the single measurements, see description of the related commands below.</p> |
| Example | <pre>:MEAS:TDSC:RFTX:MODQ:ALL :FETC:TDSC:RFTX:MODQ:ALL?</pre> <p>In this case, all relevant RF TX measurements will be performed in a sequence, e.g. <code>21.624,72.8,15.335,72.736,6.738,27.512,-6.378,0.961,-28.251,-43.743</code>.</p> |

:FETCh:TDSCdma:RFTX:MODQuality:ERMS?

| | |
|--------------------|--|
| Syntax | <code>:FETCh:TDSCdma:RFTX:MODQuality:ERMS?</code> |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | <p>The command returns the result of a previous <code>:MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:ERMS</code> measurement command.</p> |
| Example | <pre>:FETC:TDSC:RFTX:MODQ:ERMS?</pre> <p>Returns the measured EVM RMS vector error, e.g. <code>21.624</code>.</p> |

:FETCh:TDSCdma:RFTX:MODQuality:EPeak?

| | |
|--------------------|--|
| Syntax | <code>:FETCh:TDSCdma:RFTX:MODQuality:EPeak?</code> |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |

| | |
|----------------|---|
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:EPEAk measure- ment command. |
| Example | :FETC:TDSC:RFTX:MODQ:EPEAk? Returns the measured EVM peak vector error, e.g. 72.8. |

:FETCh:TDSCdma:RFTX:MODQuality:MRMS?

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:MRMS? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEA- Sure:TDSCdma:CONT ARR:RFTX:MODQuality:MRMS measurement com- mand. |
| Example | :FETC:TDSC:RFTX:MODQuality:MRMS? Returns the measured magnitude RMS vector error, e.g. 15.335. |

:FETCh:TDSCdma:RFTX:VERRor:MODQuality:MPEAk?

| | |
|--------------------|---|
| Syntax | :FETCh:TDSCdma:RFTX:VERRor:MODQuality:MPEAk? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:MPEAk measure- ment command. |
| Example | :FETC:TDSC:RFTX:MODQ:MPEA? Returns the measured magnitude peak vector error, e.g. 72.736. |

:FETCh:TDSCdma:RFTX:MODQuality:PRMS?

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:PRMS? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:PRMS measure- ment command. |
| Example | :FETC:TDSC:RFTX:MODQ:PRMS? Returns the measured RMS phase vector error, e.g. 6.738. |

:FETCh:TDSCdma:RFTX:MODQuality:PPEAk?

| | |
|---------------|---------------------------------------|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:PPEAk? |
|---------------|---------------------------------------|

| | |
|--------------------|--|
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:PPEAK measurement command. |
| Example | :FETC:TDSC:RFTX:MODQ:PPEAK? Returns the measured phase peak vector error, e.g. 27.512. |

:FETCh:TDSCdma:RFTX:MODQuality:RHO?

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:RHO? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:RHO measurement command. |
| Example | :FETC:TDSC:RFTX:MODQ:RHO? Returns the modulation quality, e.g. 0.9989. |

:FETCh:TDSCdma:RFTX:MODQuality:IQOffset?

| | |
|--------------------|---|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:IQOffset? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:IQOffset measurement command. |
| Example | :FETC:TDSC:RFTX:MODQ:IQOF? Returns the origin offset in dBc, e.g. -28.251. |

:FETCh:TDSCdma:RFTX:MODQuality:IQImbalance?

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFTX:MODQuality:IQImbalance? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the result of a previous :MEASure:TDSCdma:CONT ARR:RFTX:MODQuality:IQImbalance measurement command. |
| Example | :FETC:TDSC:RFTX:MODQ:IQIM? Returns the IQ imbalance in dBc, e.g. -43.743. |

:FETCh:TDSCdma:RFSpectrum:OBW

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFSpectrum:OBW? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma:CONTInuous ARRay:RFSpectrum:OBW measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFSpectrum:OBW? |

:FETCh:TDSCdma:RFSpectrum:CHPower

| | |
|--------------------|--|
| Syntax | :FETCh:TDSCdma:RFSpectrum:CHPower? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma:CONTInuous ARRay:RFSpectrum:CHPower measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFSpectrum:CHPower? |

:FETCh:TDSCdma:RFSpectrum:MSpectrum[:DATA]

| | |
|--------------------|---|
| Syntax | :FETCh:TDSCdma:RFSpectrum:MSpectrum[:DATA]? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma:CONTInuous:RFSpectrum:MSpectrum[:DATA] measurement command. The measurement must be started before send this command. |
| Example | :FETCh:TDSCdma:RFSpectrum:MSpectrum? |

:FETCh:TDSCdma:RFSpectrum:ACLR

| | |
|--------------------|---|
| Syntax | :FETCh:TDSCdma:RFSpectrum:ACLR? |
| Parameters | There are no parameters. |
| Description | Only the query form is supported. |
| Query | The command returns the results from the :MEASure:TDSCdma:CONTInuous ARRay:RFSpectrum:ACLR measurement command. The measurement must be started before send this command. |

Example :FETCh:TDSCDma:RFSpectrum:ACLR?

:FETCh:TDSCDma:RFSpectrum:SEM[:DATA]

Syntax :FETCh:TDSCDma:RFSpectrum:SEM[:DATA]?

Parameters There are no parameters.

Description Only the query form is supported.

Query The command returns the results from the :MEASure:TDSCDma:CONTinuous:RFSpectrum:SEM[:DATA] measurement command. The measurement must be started before send this command.

Example :FETCh:TDSCDma:RFSpectrum:SEM?

CALCulate Subsystem

The CALCulate subsystem provides a large number of commands in order to set limits, check measurement result values against those limits and to perform statistic evaluation of measurement result values.

:CALCulate:RESet

Syntax :CALCulate:RESet

Parameters There are no parameters.

Description Resets the **CALC** subsystem and brings it into a defined operating state. We kindly recommend to use this command to initialize the **CALC** subsystem when starting a new test run. For further details, refer to the example shown in section "[The CALCulate Subsystem](#)" on page 201.

Query There is no query form of this command available.

Example :CALCulate:RESet
:MEAS:GSM:ARR:RFTX:LENG 10
:CALC:MAverage?

This example first resets the **CALC** subsystem and then starts the measurement of the frequency error for 10 measurement runs. The average measurement result value of those ten measurement runs will then be calculated and returned by the :CALC:MAV? query (see below for details regarding this command).

:CALCulate:LIMit:FAIL[:LAST]?

Syntax :CALCulate:LIMit:FAIL[:LAST]?

Parameters There are no parameters.

Description There is solely a query form of this command available.

| | |
|----------------|---|
| Query | <p>Checks whether the result(s) of the last measurement performed did violate the user-defined measurement result limits or not. As long as all single measurement results are within the limits, a 0 will be returned. A 1 indicates that at least one measurement result did violate at least one limit.</p> <p>The limits for the single measurements can be set using the appropriate commands of the CALC subsystem (as explained in this section).</p> |
| Example | <pre>:CALCulate:RESet :MEAS:GSM:RFTX:ALL :CALC:LIM:FAIL?</pre> <p>This example first resets the CALC subsystem and then starts the measurement of all relevant RF TX parameters. If a measurement result of the current sequence violates the corresponding measurement result limits, the query will return a 1. When all measurement results are within their limits, a 0 will be returned.</p> |

:CALCulate:LIMit:FAIL:CUMulative?

| | |
|--------------------|--|
| Syntax | :CALCulate:LIMit:FAIL:CUMulative? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>Checks whether any result(s) of the measurement started last did violate their corresponding, user-defined measurement result limits. As long as all single measurement results are within their limits, a 0 will be returned. A 1 indicates that at least one measurement result did violate at least one limit.</p> <p>The limits for the single measurements can be set using the appropriate commands of the CALC subsystem (as explained in this section).</p> <p>To reset the cumulative check of the measurement results, use the :CALC:LIM:FAIL:CUM:RES command as explained below.</p> |
| Example | <pre>:CALC:LIM:FAIL:CUM:RES :MEAS:GSM:RFTX:TEMP ... (other SCPI commands) ... :CALC:LIM:FAIL:CUM?</pre> <p>This example first resets the cumulative process then starts the continuous check of the UL burst against the power/time template (PTT). After that, other SCPI commands are performed. After a while, the :CALC:LIM:FAIL:CUM command is used to check whether there has been any violation of the PTT since the start of the :MEAS:GSM:RFTX:TEMP measurement.</p> |

:CALCulate:LIMit:FAIL:CUMulative:RESet

| | |
|--------------------|---|
| Syntax | :CALCulate:LIMit:FAIL:CUMulative:RESet |
| Parameters | There are no parameters. |
| Description | This command resets the cumulative process as explained above (see description of command :CALC:LIM:FAIL:CUM for details). |
| Query | There is no query form of this command available. |

Example **:CALC:LIM:FAIL:CUM:RES**
 :MEAS:GSM:RFTX:TEMP
 ... (other SCPI commands) ... **:CALC:LIM:FAIL:CUM?**
 This example first resets the cumulative process then starts the continuous check of the UL burst against the power/time template (PTT). After that, other SCPI commands are performed. After a while, the **:CALC:LIM:FAIL:CUM** command is used to check whether there has been any violation of the PTT since the start of the **:MEAS:GSM:RFTX:TEMP** measurement.

:CALCulate:MAverage?

Syntax **:CALCulate:MAverage?**

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query Calculates and returns the **average** measurement result value of the measurement started last. The string delivered back will contain as many average values as measurement types performed. The single average values will always have the format of floating point real numbers and will be separated by commas.

Example **:CONF:MEAS:GRO:RFTX POW,PRMS**
 :CALC:RES
 :MEAS:GSM:ARR:RFTX:GROup 20
 :CALCulate:MAverage?
 In this example, first the group of measurements is defined by a power measurement combined with a RMS phase error measurement. This group of measurements is performed 20 times. After that, the single measurement result values will be averaged and returned. The values returned in this example are:
 "4.53,9.98".

:CALCulate:MMINimum?

Syntax **:CALCulate:MMINimum?**

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query Returns the **minimum** measurement result value of the measurement started last. The string delivered back will contain as many minimum values as measurement types performed. The single minimum values will always have the format of floating point real numbers and will be separated by commas.

Example **:MEAS:AFAN:SIN**
 :CALC:RES
 ... (other SCPI commands) ...
 :CALCulate:MMIN?
 In this example, first a continuous SINAD measurement is started. After some time, the **:CALC:MMIN** command is used to read out the minimum SINAD measured. The string returned in this example is "42.6".

:CALCulate:MMAXimum?

| | |
|--------------------|---|
| Syntax | :CALCulate:MMAXimum? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the maximum measurement result value of the measurement started last. The string delivered back will contain as many maximum values as measurement types performed. The single maximum values will always have the format of floating point real numbers and will be separated by commas. |
| Example | <pre> :MEAS:GSM:RFTX:FREQ :CALC:RES ... (other SCPI commands) ... :CALCulate:MMAX? </pre> <p>In this example, first a continuous measurement of the mobile's frequency error is started. After some time, the :CALC:MMAX command is used to read out the maximum frequency error of the mobile. The string returned in this example is "22.1".</p> |

:CALCulate:MSIGma?

| | |
|--------------------|---|
| Syntax | :CALCulate:MSIGma? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the mean value and the standard deviation of the measurement results since the last measurement start. The string delivered back contains as many pairs of mean and standard deviation values as measurement types performed. The individual values all have the format of floating point real numbers and will be separated by commas. |
| Example | <pre> :MEAS:AFAN:SIN :CALC:RES ...(other SCPI commands)... :CALCulate:MMIN? </pre> <p>In this example, first a continuous SINAD measurement is started. After some time, the :CALC:MMIN command is used to read out the minimum SINAD measured. The string returned in this example is "42.6".</p> |

:CALCulate:AFANalyser:MAVerage?

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser:MAVerage? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Calculates and returns the average measurement result value of the Audio measurement started last. The string delivered back will contain as many average values as measurement types performed. The single average values will always have the format of floating point real numbers and will be separated by commas. |

| | |
|----------------|---|
| Example | :CALC:RES :MEAS:ARR:AFAN:ALL :CALC:AFAN:MAV? |
|----------------|---|

:CALCulate:AFANalyser:MMINimum?

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser:MMINimum? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the minimum measurement result value of the Audio measurement started last. The string delivered back will contain as many minimum values as measurement types performed. The single minimum values will always have the format of floating point real numbers and will be separated by commas. |
| Example | :CALC:RES :MEAS:ARR:AFAN:ALL :CALC:AFAN:MMIN? |

:CALCulate:AFANalyser:MMAXimum?

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser:MMAXimum? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the maximum measurement result value of the Audio measurement started last. The string delivered back will contain as many maximum values as measurement types performed. The single maximum values will always have the format of floating point real numbers and will be separated by commas. |
| Example | :CALC:RES :MEAS:ARR:AFAN:ALL :CALC:AFAN:MMAX? |

:CALCulate:AFANalyser:MSIGma?

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser:MSIGma? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the mean value and the standard deviation of the audio measurement results since the last measurement start. The string delivered back contains as many pairs of mean and standard deviation values as measurement types performed. The individual values all have the format of floating point real numbers and will be separated by commas. |

| | |
|----------------|--|
| Example | <pre> :CALC:RES :MEAS:AFAN:ACV ... (other SCPI commands) ... :CALC:AFAN:MSIG? </pre> <p>In this example, first a continuous measurement of the audio voltage is started. After some time, the :CALC:AFAN:MSIGma command is used to read out the mean voltage and its standard deviation. The string returned in this example is "1.1, 0.2".</p> |
|----------------|--|

:CALC:AFANalyser:ALL:LIMit[:FAIL]?

| | |
|--------------------|---|
| Syntax | :CALC:AFANalyser:ALL:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>This command delivers six boolean numbers in a string, separated by commas. These numbers indicate whether the six single measurement result values of the MEAS:AFAN:ALL measurement did violate their individual limits (set with the commands explained below) or not.</p> <p>The position of the boolean number within the string returned indicates the result of the check of following measurements:</p> <ol style="list-style-type: none"> 1. the peak-to-peak measurement of the AC voltage, 2. the root-mean square value of the AC voltage, 3. the RMS-valued AC ripple on a DC voltage, 4. the audio frequency measurement, 5. the measurement of the third-harmonic distortion (THD) of a sine wave and 6. the SINAD measurement. <p>While a 0 on any position indicates that the related measurement result value is within the limits specified, a 1 indicates that the measurement result value did violate at least one of the limits set for the related test.</p> |
| Example | <pre> :MEAS:AFAN:ALL :CALC:AFAN:ALL:LIM? </pre> <p>String returned: "0,0,0,0,1,0"</p> <p>The 1 on position 5 indicates that the distortion measurement is off its limits.</p> |

:CALCulate:AFANalyser:ALL:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser:ALL:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:ALL measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | <pre> :CALC:AFAN:ALL:LIM:STAT ON </pre> <p>Switches the limit check of the main audio measurements on.</p> |

:CALCulate:AFANalyser:ALL:LIMit:UPPer[:DATA]

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFANalyser:ALL:LIMit:UPPer[:DATA] <real1>,<real2>,<real3>,<real4>,<real5>,<real6></code> |
| Parameters | <p>realx are six floating point real numbers.</p> <p>real1: The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 5.0.</p> <p>real2: The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 5.0.</p> <p>real3: The minimum value is -40.0, the maximum is 40.0. The minimum resolution possible is 0.0001, the default value is 5.0.</p> <p>real4: The minimum value is 0.0, the maximum is 20,000.0. The minimum resolution possible is 1.0, the default value is 5,000.0.</p> <p>real5: The minimum value is 0.0, the maximum is 100.0. The minimum resolution possible is 0.1, the default value is 5.0.</p> <p>real6: The minimum value is 0.0, the maximum is 100.0. The minimum resolution possible is 0.1, the default value is 20.0.</p> |
| Description | <p>Sets the upper limits for the six main audio frequency measurements with one command. The order of the floating point real numbers defines their meaning:</p> <p>real1 represents the upper limit of the peak-to-peak AC voltage measurement; the physical dimension of the number stated is V(pp).</p> <p>real2 represents the upper limit of the RMS-valued AC voltage measurement; the physical dimension of the number stated is V(rms).</p> <p>real3 represents the upper limit of the RMS-valued measurement of an AC voltage ripple on a DC voltage; the physical dimension of the number stated is V(rms).</p> <p>real4 represents the upper limit of the audio frequency measurement; the physical dimension of the number stated is Hertz.</p> <p>real5 represents the upper limit of the distortion measurement, carried out on the third harmonic of a sine wave. The physical dimension of the number stated is percentage.</p> <p>real6 represents the upper limit of the SINAD signal-to-noise ratio measurement; the physical dimension of the number stated is dB.</p> |
| Query | The query form of this command is not supported. |
| Example | <p>:CALC:AFAN:ALL:LIM:UPP 3,3,1,4000,5,100</p> <p>Sets the upper limits for the audio measurements: 3 V for the peak-to-peak AC voltage, 3 V for the rms-valued AC voltage, 1 V for the rms-valued ripple voltage, 4 kHz for the audio frequency, 5% for THD, and 100 dB for SINAD.</p> |

:CALCulate:AFANalyser:ALL:LIMit:LOWer[:DATA]

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFANalyser:ALL:LIMit:LOWer[:DATA] <real1>,<real2>,<real3>,<real4>,<real5>,<real6></code> |
| Parameters | <p>realx are six floating point real numbers.</p> <p>real1: The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 1.0.</p> <p>real2: The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 1.0.</p> <p>real3: The minimum value is -40.0, the maximum is 40.0. The minimum resolution possible is 0.0001, the default value is -5.0.</p> <p>real4: The minimum value is 0.0, the maximum is 20,000.0. The minimum resolution possible is 1.0, the default value is 1,000.0.</p> <p>real5: The minimum value is 0.0, the maximum is 100.0. The minimum resolution possible is 0.1, the default value is 0.0.</p> <p>real6: The minimum value is 0.0, the maximum is 100.0. The minimum resolution possible is 0.1, the default value is 0.0.</p> |
| Description | <p>Sets the lower limits for the six main audio frequency measurements with one command. The order of the floating point real numbers defines their meaning:</p> <p>real1 represents the lower limit of the peak-to-peak AC voltage measurement; the physical dimension of the number stated is V(pp).</p> <p>real2 represents the lower limit of the RMS-valued AC voltage measurement; the physical dimension of the number stated is V(rms).</p> <p>real3 represents the lower limit of the RMS-valued measurement of an AC voltage ripple on a DC voltage; the physical dimension of the number stated is V(rms).</p> <p>real4 represents the lower limit of the audio frequency measurement; the physical dimension of the number stated is Hertz.</p> <p>real5 represents the lower limit of the distortion measurement, carried out on the third harmonic of a sine wave. The physical dimension of the number stated is percentage.</p> <p>real6 represents the lower limit of the SINAD signal-to-noise ratio measurement; the physical dimension of the number stated is dB.</p> |
| Query | The query form of this command is not supported. |
| Example | <code>:CALC:AFAN:ALL:LIM:LOW 1,1,0,400,0,25</code> Sets the lower limits for the audio measurements. |

**:CALCulate:AFANalyser
:ACVoltage:PPEAk:LIMit[:FAIL]?**

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFANalyser :ACVoltage:PPEAk:LIMit[:FAIL]?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>This command delivers one boolean number, indicating whether the user-definable limits of the peak-to-peak measurement of the AC voltage applied to the audio analyzer were violated by a measurement result value or not.</p> <p>A 0 means that the measurement result value is within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below.</p> |

| | |
|----------------|--|
| Example | <pre>:MEAS:ARRAY:AFAN:ACV:PPEA 10 :CALC:AFAN:ACV:PPEA:LIM? String returned: "1" This string delivered back indicates that at least one measurement result value did violate the limits of the peak-to-peak AC measurement.</pre> |
|----------------|--|

:CALCulate:AFANalyser:ACVoltage:PEAKp:LIMit:STATE

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser:ACVoltage:PEAKp:LIMit:STATE <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:ACV:PEAK measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | <pre>:CALC:AFAN:ACV:PEAK:LIM:STAT ON Switches the limit check of the peak-to-peak AC voltage measurement on.</pre> |

**:CALCulate:AFANalyser
:ACVoltage:PEAKp:LIMit:UPPer[:DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :ACVoltage:PEAKp:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 5.0. |
| Description | Sets the upper limit for the peak-to-peak measurement of the AC voltage applied to the audio analyzer. The physical dimension of the number stated is V(pp). |
| Query | The query form of this command is not available. |
| Example | <pre>:CALC:AFAN:ACV:PEAK:LIM:UPP 3 Sets the upper limit of the peak-to-peak AC voltage measurement to 3 V.</pre> |

**:CALCulate:AFANalyser
:ACVoltage:PEAKp:LIMit:LOWer[:DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :ACVoltage:PEAKp:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 30.0. The minimum resolution possible is 0.0001, the default value is 1.0. |
| Description | Sets the lower limit for the peak-to-peak measurement of the AC voltage applied to the audio analyzer. The physical dimension of the number stated is V(pp). |
| Query | The query form of this command is not available. |
| Example | <pre>:CALC:AFAN:ACV:PEAK:LIM:LOW 0 Sets the lower limit of the peak-to-peak AC voltage measurement to 0 V.</pre> |

:CALCulate:AFANalyser
:ACVoltage:RMS:LIMit[:FAIL]?

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser :ACVoltage:RMS:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the RMS-valued measurement of the AC voltage applied to the audio analyzer were violated by a measurement result value or not. A 0 means that the measurement result value is within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. |
| Example | :MEAS:ARRAY:AFAN:ACV:RMS 5 :CALC:AFAN:ACV:RMS:LIM? String returned: "0" This string delivered back indicates that there was no violation of the limits set. |

:CALCulate:AFANalyser
:ACVoltage:RMS:LIMit:STATe

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :ACVoltage:RMS:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:ACV:RMS measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | :CALC:AFAN:ACV:RMS:LIM:STAT ON Switches the limit check of the RMS-valued AC voltage measurement on. |

:CALCulate:AFANalyser
:ACVoltage:RMS:LIMit:UPPer[:DATA]

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :ACVoltage:RMS:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0 , the maximum is 30.0 . The minimum resolution possible is 0.0001 , the default value is 5.0 . |
| Description | Sets the upper limit for the RMS-valued measurement of the AC voltage applied to the audio analyzer. The physical dimension of the number stated is V(rms). |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:ACV:RMS:LIM:UPP 3 Sets the upper limit of the rms-valued AC voltage measurement to 3 V. |

:CALCulate:AFANalyser
:ACVoltage:RMS:LIMit:LOWer[:DATA]

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:ACVoltage:RMS:LIMit:LOWer[:DATA] <real1></code> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0 , the maximum is 30.0 . The minimum resolution possible is 0.0001 , the default value is 1.0 . |
| Description | Sets the lower limit for the RMS-valued measurement of the AC voltage applied to the audio analyzer. The physical dimension of the number stated is V(rms). |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:ACV:RMS:LIM:LOW 0 Sets the lower limit of the rms-valued AC voltage measurement to 0 V. |

:CALCulate:AFANalyser
:DCVoltage:LIMit[:FAIL]?

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:DCVoltage:LIMit[:FAIL]?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the RMS-valued measurement of the AC ripple on a DC voltage were violated by a measurement result value or not. A 0 means that the measurement result value is within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. |
| Example | :MEAS:ARRay:AFAN:DCV 10 :CALC:AFAN:DCV:LIM? String returned: "0" This string delivered back indicates that there has been no violation of the limits set. |

:CALCulate:AFANalyser
:DCVoltage:LIMit:STATe

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:DCVoltage:LIMit:STATe <PredefExp></code> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the <code>:MEAS:AFAN:DCV</code> measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | :CALC:AFAN:DCV:LIM:STAT ON Switches the limit check of the RMS-valued AC voltage ripple measurement of a DC voltage on. |

:CALCulate:AFANalyser
:DCVoltage:LIMit:UPPer[:DATA]

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:DCVoltage:LIMit:UPPer[:DATA] <real1></code> |
| Parameters | real1 is a floating point real number. The minimum value is -40.0 , the maximum is 40.0 . The minimum resolution possible is 0.0001 , the default value is 5.0 . |
| Description | Sets the upper limit for the RMS-valued measurement of the AC voltage component on a DC voltage, applied to the audio analyzer. The physical dimension of the number stated is V(rms). |
| Query | The query form of this command is not supported. |
| Example | <code>:CALC:AFAN:DCV:LIM:UPP 6</code> Sets the upper limit of the rms-valued ripple voltage measurement to 6 V. |

:CALCulate:AFANalyser
:DCVoltage:LIMit:LOWer[:DATA]

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:DCVoltage:LIMit:LOWer[:DATA] <real1></code> |
| Parameters | real1 is a floating point real number. The minimum value is -40.0 , the maximum is 40.0 . The minimum resolution possible is 0.0001 , the default value is -5.0 . |
| Description | Sets the lower limit for the RMS-valued measurement of the AC voltage component on a DC voltage, applied to the audio analyzer. The physical dimension of the number stated is V(rms). |
| Query | The query form of this command is not supported. |
| Example | <code>:CALC:AFAN:DCV:LIM:LOW 1.75</code> Sets the upper limit of the rms-valued ripple voltage measurement to 1.75 V. |

:CALCulate:AFANalyser
:FREQuency:LIMit[:FAIL]?

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFANalyser</code> <code>:FREQuency:LIMit[:FAIL]?</code> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the AF frequency measurement were violated by a measurement result value or not. A 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. |
| Example | <code>:MEAS:AFAN:FREQ</code> <code>:CALC:AFAN:FREQ:LIM?</code> String returned: "1" This string delivered back indicates that there has been a violation of the limits set. |

:CALCulate:AFANalyser:FREQUENCY:LIMit:STate?

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser:FREQUENCY:LIMit:STate? <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:FREQ measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | :CALC:AFAN:FREQ:LIM:STAT ON Switches the limit check of the AF frequency measurement on. |

**:CALCulate:AFANalyser
:FREQUENCY:LIMit:UPPer[:DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser :FREQUENCY:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 20,000.0. The minimum resolution possible is 1.0, the default value is 5,000.0. |
| Description | Sets the upper limit for the AF frequency measurement. The physical dimension of the number stated is Hertz. |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:FREQ:LIM:UPP 10000 Sets the upper limit of the audio frequency measurement to 10 kHz. |

**:CALCulate:AFANalyser
:FREQUENCY:LIMit:LOWer[:DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser :FREQUENCY:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 20,000.0. The minimum resolution possible is 1.0, the default value is 1,000.0. |
| Description | Sets the lower limit for the AF frequency measurement. The physical dimension of the number stated is Hertz. |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:FREQ:LIM:LOW 950 Sets the lower limit of the audio frequency measurement to 950 Hz. |

**:CALCulate:AFANalyser
:DISTortion:LIMit[:FAIL]?**

| | |
|---------------|--|
| Syntax | :CALCulate:AFANalyser :DISTortion:LIMit[:FAIL]? |
|---------------|--|

| | |
|--------------------|---|
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the distortion measurement were violated by a measurement result value or not. A 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. Notes <ul style="list-style-type: none"> - The distortion measurement is carried through on the third harmonic of a 1 kHz sine wave. - All audio measurements require the Audio Option to be installed on your Willtek 4400. |
| Example | :MEAS:ARRay:AFAN:DIST 10 :CALC:AFAN:DIST:LIM? String returned: "1" This string delivered back indicates that there was a violation of the limits set. |

:CALCulate:AFANalyser
:DISTortion:AFAN:DIST:LIMit:STATe

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :DISTortion:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:DIST measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | :CALC:AFAN:DIST:LIM:STAT ON Switches the limit check of the distortion measurement on. |

:CALCulate:AFANalyser
:DISTortion:LIMit:UPPer[:DATA]

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :DISTortion:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0 , the maximum is 100.0 . The minimum resolution possible is 0.1 , the default value is 5.0 . |
| Description | Sets the upper limit for the distortion measurement. The physical dimension of the number stated is percent (the relation of the AC voltage measured for the third harmonic of a 1 kHz sine wave in relation to the AC voltage measured for the nominal frequency). |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:DIST:LIM:UPP 4.5 Sets the upper limit of the distortion measurement to 4.5%. |

:CALCulate:AFANalyser
:DISTortion:LIMit:LOWer[:DATA]

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser :DISTortion:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0 , the maximum is 100.0 . The minimum resolution possible is 0.1 , the default value is 0.0 . |
| Description | Sets the lower limit for the distortion measurement. The physical dimension of the number stated is percent (the relation of the AC voltage measured for the third harmonic of a 1 kHz sine wave in relation to the AC voltage measured for the nominal frequency). |
| Query | The query form of this command is not available. |
| Example | :CALC:AFAN:DIST:LIM:LOW 0.0 Sets the lower limit of the distortion measurement to 0%. |

:CALCulate:AFANalyser
:SINad:LIMit[:FAIL]?

| | |
|--------------------|---|
| Syntax | :CALCulate:AFANalyser :SINad:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the SINAD measurement were violated by a measurement result value or not. A 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. Notes <ul style="list-style-type: none"> - The SINAD measurement is carried through with a 1 kHz sine wave. - All audio measurements require the Audio Option to be installed on your Willtek 4400. |
| Example | :MEAS:ARRAY:AFAN:SIN 5 :CALC:AFAN:SIN:LIM? String returned: "1" This string delivered back indicates that there was a violation of the limits set. |

:CALCulate:AFANalyser
:SINad:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:AFANalyser :SINad:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:AFAN:SIN measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |

Example :CALC:AFAN:SIN:LIM:STAT ON
Switches the limit check of the SINAD measurement on.

:CALCulate:AFANalyser
:SINad:LIMit:UPPer[:DATA]

Syntax :CALCulate:AFANalyser
 :SINad:LIMit:UPPer[:DATA] <real1>

Parameters **real1** is a floating point real number. The minimum value is **0.0**, the maximum is **100.0**. The minimum resolution possible is **0.1**, the default value is **20.0**.

Description Sets the **upper** limit for the SINAD measurement. The physical dimension of the number stated is dB.

Query The query form of this command is not available.

Example :CALC:AFAN:SIN:LIM:UPP 100
Sets the upper limit of the SINAD measurement to 100%
nbsp;dB.

:CALCulate:AFANalyser
:SINad:LIMit:LOWer[:DATA]

Syntax :CALCulate:AFANalyser
 :SINad:LIMit:LOWer[:DATA] <real1>

Parameters **real1** is a floating point real number. The minimum value is **0.0**, the maximum is **100.0**. The minimum resolution possible is **0.1**, the default value is **0.0**.

Description Sets the **lower** limit for the SINAD measurement. The physical dimension of the number stated is dB.

Query The query form of this command is not available.

Example :CALC:AFAN:SIN:LIM:LOW 30
Sets the lower limit of the SINAD measurement to 30 dB.

:CALCulate:AFSPectrum:TEMPlate:LIMit[:FAIL]

Syntax :CALCulate:AFSPectrum:TEMPlate:LIMit[:FAIL]?

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query This command delivers ten boolean numbers, indicating whether the user-definable limits of the audio spectrum measurement were violated by a measurement result value or not.
A **0** means that the measurement result values for the respective section of the template were within the limits set, while a **1** indicates that at least one measurement result value did violate at least one of the limits.
The limits can be set using the commands described below.

| | |
|----------------|--|
| Example | <pre>:MEAS:ARRay:AFSP 10 :CALC:AFSP:TEMP:LIM:FAIL? String returned: "0,1,0,0,0,0,0,0,0,0" This string delivered back indicates that there was a violation of the limits set for the second area.</pre> |
|----------------|--|

:CALCulate:AFSPectrum:TEMPlate:LIMit:STATe

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFSPectrum:TEMPlate:LIMit:STATe <PredefExp></code> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the <code>MEAS:AFSP:TEMPlate</code> and <code>MEAS:ARR:AFSP:TEMPlate</code> measurements against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | <pre>:CALC:GSM:RFTX:TEMP:LIM:STAT ON Switches on the limit check of the audio spectrum measurement against the user-defined template.</pre> |

:CALCulate:AFSPectrum:TEMPlate:LIMit:LOAD

| | |
|--------------------|---|
| Syntax | <code>:CALCulate:AFSPectrum:TEMPlate:LIMit:LOAD <string1></code> |
| Parameters | string1 is a string giving the name of the file containing the audio limits. The maximum length of string1 is 50 characters. The default for string1 is "example.lmt" . |
| Description | This command loads the audio template description file. Please note that the data contained in the file need to be activated (using the <code>CALC:AFSP:TEMP:LIM:STAT ON</code> command described above) before the template will have any effect on the measurement results. |
| Query | There is no query form of this command available. |
| Example | <pre>:CALC:AFSP:LIM:LOAD "example.lmt" Loads the audio spectrum template file example.lmt which resides in the /rapid/audiolmt directory.</pre> |

:CALCulate:AFSPectrum:TEMPlate:LIMit:STORE

| | |
|--------------------|--|
| Syntax | <code>:CALCulate:AFSPectrum:TEMPlate:LIMit:STORE <string1></code> |
| Parameters | string1 is a string giving the name of the file in which the current audio limits are to be saved. The maximum length of string1 is 50 characters. |
| Description | This command saves the current audio template in a file. The template can be defined by the FREQ , UPPer , LOWer and COMMeNt comments. |
| Query | There is no query form of this command available. |
| Example | <pre>:CALC:AFSP:LIM:STORE "example.lmt" Stores the currently defined audio spectrum template in the file example.lmt in the /rapid/audiolmt directory. If a file of this name already exists, it will be overwritten.</pre> |

:CALCulate:AFSPectrum:TEMPlate:LIMit:COMment

| | |
|--------------------|--|
| Syntax | :CALCulate:AFSPectrum:TEMPlate:LIMit:COMment <string1> |
| Parameters | string1 is a comment line related to the coupling loss data. The maximum length of string1 is 80 characters. |
| Description | Defines a comment line for storage in the audio template file and for display in the Comment line field on the Limits screen. Note: The comment can be saved with the STORE command, together with the other template parameters. |
| Query | There is no query form of this command available. |
| Example | :CALC:AFSP:LIM:COMM "Audio template for Siemens S35" |

:CALCulate:AFSPectrum:TEMPlate:LIMit:FREQuency

| | |
|--------------------|--|
| Syntax | :CALCulate:AFSPectrum:TEMPlate:LIMit:FREQuency <real1>,<real2>,<real3>,<real4>, <real5>,<real6>,<real7>,<real8>,<real9>,<real10> |
| Parameters | real1 through realxp are floating point real numbers. The minimum value for all numbers is 0.0, the maximum value 20000.0. The minimum resolution possible for all realxt is 0.1. The default values are: for real0t : -41.0, for real1t : -28.0, for real2t : -18.0, for real3t : -10.0 for real4t : 0.0, for real5t : 553.0 for real6t : 561.0, for real7t : 571.0 for real8t : 580.0. The minimum value for all realxp is -150.0, the maximum value 5.0. The minimum resolution possible for all realx is 0.1. |
| Description | Sets the frequency values for the user-definable audio spectrum template. The values are in Hz. They are used in conjunction with the definitions for the upper and lower limits. |
| Query | The query form of this command is not available. |
| Example | :CALC:AFSP:TEMP:LIM:FREQ 25, 50, 100, 200, 500, 1000, 2000, 4000, 8000, 16000 This command sets the frequency values for the upper and lower limits. Note: These are the default values after an instrument reset. |

:CALCulate:AFSPectrum:TEMPlate:LIMit:UPPer

| | |
|--------------------|---|
| Syntax | :CALCulate:AFSPectrum:TEMPlate:LIMit:UPPer <real1>,<real2>,<real3>,<real4>, <real5>,<real6>,<real7>,<real8>,<real9>,<real10> |
| Parameters | The realx values are floating point real numbers. The minimum value is -120 , the maximum value is +10 . The resolution for all values is 0.1 . The values are the upper limits for the audio template in dB, relative to the reference level. |
| Description | Sets the upper limits for the user-definable audio spectrum template. These upper limits refer to the frequencies defined in the FREQuency command above. |
| Query | The query form of this command is not available. |
| Example | :CALC:AFSP:TEMP:LIM:UPP 5, 5, 5, 5, 5, 5, 5, 5, 5, 5 This command sets all upper limits to +5 dB at the frequencies defined with the :CALC:AFSP:TEMP:LIM:FREQ command. |

:CALCulate:AFSPectrum:TEMPlate:LIMit:LOWer

| | |
|--------------------|---|
| Syntax | :CALCulate:AFSPectrum:TEMPlate:LIMit:LOWer <real1>,<real2>,<real3>,<real4>, <real5>,<real6>,<real7>,<real8>,<real9>,<real10> |
| Parameters | The realx values are floating point real numbers. The minimum value is -120 , the maximum value is +10 . The resolution for all values is 0.1 . The values are the lower limits for the audio template in dB, relative to the reference level. |
| Description | Sets the lower limits for the user-definable audio spectrum template. These upper limits refer to the frequencies defined in the FREQuency command above. |
| Query | The query form of this command is not available. |
| Example | :CALC:AFSP:TEMP:LIM:LOW -25, -25, -25, -25, -25, -25, -25, -25, -25, -25 This command sets all lower limits to -25 dB at the frequencies defined with the :CALC:AFSP:TEMP:LIM:FREQ command. |

:CALCulate:AFSPectrum:VALue?

| | |
|-------------------|---|
| Syntax | :CALCulate:AFSPectrum:VALue? <real1> |
| Parameters | real1 is a floating point real number. It provides the position on the frequency axis for which the audio spectrum value is being requested. |
| Query | Provides the measurement value (audio spectrum level) for the given frequency value. If there is no measurement value at exactly this frequency, the value is interpolated. |
| Example | :MEAS:BLOC:AFSP:AVG :CALC:AFSP:VAL? 1200 Delivers the audio spectrum value at the frequency of 1.2 kHz. |

:CALCulate:AFSPectrum:MAXPeak

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:MAVerage? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | The query form of this command delivers back the frequency (in Hz) and the maximum audio spectrum level of the measured audio spectrum curve. The string returned will contain two floating point real numbers. The physical dimension of the measurement result values returned is for the first value Hz and for the second value dB. Note: Please note that you need to have a audio measurement taken before this command will return any meaningful result. We recommend to use the :MEAS:BLOC:AFSP:AVG command to do so. |
| Example | :MEAS:BLOC:AFSP:AVG 10 :CALC:AFSP:MAXP? This command will read at the audio spectrum measurement maximum the frequency and the measurement result value. The values returned in this example is: "1230.0,-5.2". |

:CALCulate:PSUPply:MAVerage?

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:MAVerage? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Calculates and returns the average measurement result value of the current measurement started last. The string delivered back will contain as many average values as measurement types performed. The single average values will always have the format of floating point real numbers and will be separated by commas. |
| Example | :CALC:RES :MEAS:ARR:PSUP:ALL :CALC:PSUP:MAV? 603.1,215.8,908.2 Returns the average values for average power consumption (603.1 mW), average current consumption (215.8 mA) and peak current consumption (908.2 mA). |

:CALCulate:PSUPply:MMINimum?

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:MMINimum? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the minimum measurement result value of the current measurement started last. The string delivered back will contain as many minimum values as measurement types performed. The single minimum values will always have the format of floating point real numbers and will be separated by commas. |

| | |
|----------------|--|
| Example | :CALC:RES :MEAS:ARR:PSUP:ALL :CALC:PSUP:MMIN? Returns the minimum values for average power consumption (in mW), average current consumption (in mA), and peak current consumption (in mA). |
|----------------|--|

:CALCulate:PSUPply:MMAximum?

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:MMAximum? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the maximum measurement result value of the current measurement started last. The string delivered back will contain as many maximum values as measurement types performed. The single maximum values will always have the format of floating point real numbers and will be separated by commas. |
| Example | :CALC:RES :MEAS:ARR:PSUP:ALL :CALC:PSUP:MMAx? Returns the maximum values for average power consumption (in mW), average current consumption (in mA), and peak current consumption (in mA). |

:CALCulate:PSUPply:MSIGma?

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:MSIGma? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | Returns the mean value and the standard deviation of the current measurement started last. The string delivered back will contain as many pairs of mean and standard deviation values as measurement types performed. All values have the format of floating point real numbers and are separated by commas. |
| Example | :CALC:RES :MEAS:ARR:PSUP:ALL :CALC:PSUP:MSIG? Returns three pairs of mean value and standard deviation for average power consumption (in mW), average current consumption (in mA), and peak current consumption (in mA). |

:CALCulate:PSUPply:ALL:LIMit[:FAIL]?

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:ALL:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |

| | |
|----------------|--|
| Query | <p>This command delivers three boolean values, indicating whether the user-definable limits of the current measurements were violated by a measurement result value or not.</p> <p>For each of the three boolean values, a 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value violated at least one of the limits.</p> <p>The limits can be set using the commands described below. The results are in the sequence power consumption, average current consumption, peak current consumption.</p> |
| Example | <pre>:MEAS:ARR:PSUP:CPEA 5 :CALC:PSUP:ALL:LIM? String returned: "1,0,0" This string delivered back indicates that there was a violation of the limits for the power consumption while the other two types of measurements passed the limits.</pre> |

:CALCulate:PSUPply:ALL:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:ALL:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | This command switches the check of the measurement result values of the MEAS:PSUP:ALL measurement against their user-definable limits either on or off. |
| Query | There is no query form of this command available. |
| Example | <pre>:CALC:PSUP:ALL:LIM:STAT ON Switches the limit check for the current measurements on.</pre> |

:CALCulate:PSUPply:ALL:LIMit:UPPer[:DATA]

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:ALL:LIMit:UPPer[:DATA] <real1>,<real2>,<real3> |
| Parameters | <p>real1, real2 and real3 are floating point real numbers.</p> <ul style="list-style-type: none"> - real1 is the upper limit for the average power consumption. Its minimum value is 0.0, the maximum is 2000.0, the default value is 2000.0. - real2 is the upper limit for average current consumption. Its minimum value is 0.0, the maximum is 1000.0, the default value is 1000.0 - real3 is the upper limit for peak current consumption. Its minimum value is 0.0, the maximum is 4000.0, the default value is 4000.0 |
| Description | Sets the upper limits for the current measurements. |
| Query | The query form of this command is not available. |
| Example | <pre>:CALC:PSUP:ALL:LIM:UPP 2.3, 200.0, 1400.0 Sets the upper limit of the current measurements to 2.3 W for the average power, 200 mA for the average current and 1400 mW for the peak current.</pre> |

:CALCulate:PSUPply:ALL:LIMit:LOWer[:DATA]

| | |
|---------------|---|
| Syntax | :CALCulate:PSUPply:ALL:LIMit:LOWer[:DATA] <real1>,<real2>,<real3> |
|---------------|---|

| | |
|--------------------|--|
| Parameters | <p>real1, real2, real3 are floating point real numbers.</p> <ul style="list-style-type: none"> - real1 is the lower limit for power consumption. Its minimum value is 0.0, the maximum is 2000.0, the default value is 2000.0. - real2 is the lower limit for average current consumption. Its minimum value is 0.0, the maximum is 1000.0, the default value is 1000.0 - real3 is the lower limit for peak current consumption. Its minimum value is 0.0, the maximum is 4000.0, the default value is 4000.0 |
| Description | Sets the lower limits for the current measurements. |
| Query | The query form of this command is not available. |
| Example | <p>:CALC:PSUP:ALL:LIM:LOW 0,0,0 Sets the lower limit of the current measurements to all 0.0.</p> |

:CALCulate:PSUPply:APOWer:LIMit[:FAIL]?

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:APOWer:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | <p>This command delivers one boolean number, indicating whether the user-definable limits of the average power consumption measurement were violated by a measurement result value or not.</p> <p>A 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits.</p> <p>The limits can be set using the commands described below.</p> |
| Example | <p>:CALC:PSUP:APOW:LIM:UPP 2.3 :CALC:PSUP:APOWer:LIM:LOW 0 :MEAS:ARRay:PSUP:APOWer 5 :CALC:PSUP:APOWer:LIM? String returned in this example:"1" This string delivered back indicates that there was a violation of the limits set.</p> |

:CALCulate:PSUPply:APOWer:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:APOWer:LIMit:STATe <PreDefExp> |
| Parameters | PreDefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | <p>This command switches the check of the power consumption measurement result values of the MEAS:PSUP:APOWer measurement against their user-definable limits either ON or OFF.</p> <p>Note: All current measurements require the MS Power Supply and Current Measurement options to be installed on your Willtek 4400.</p> |
| Query | There is no query form of this command available. |
| Example | <p>:CALC:PSUP:APOW:LIM:STAT ON Switches the limit check of the current measurement on.</p> |

:CALCulate:PSUPply:APOWer:LIMit:UPPer[:DATA]

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:APOWer:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 2000.0. The resolution is 1, the default value is 2000.0. |
| Description | Sets the upper limit for the power consumption measurement. The physical dimension of the number stated is mW. |
| Query | The query form of this command is not available. |
| Example | :CALC:PSUP:APOW:LIM:UPP 100 Sets the upper limit of the power consumption measurement to 100 mW. |

:CALCulate:PSUPply:APOWer:LIMit:LOWer[:DATA]

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:APOWer:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 2000.0. The resolution is 1, the default value is 2000.0. |
| Description | Sets the lower limit for the power consumption measurement. The physical dimension of the number stated is mW. |
| Query | The query form of this command is not available. |
| Example | :CALC:PSUP:APOW:LIM:LOW 10 Sets the lower limit of the power consumption measurement to 10 mW. |

:CALCulate:PSUPply:ACURrent:LIMit:STATE

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:ACURrent:LIMit:STATE <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the current measurement result values of the MEAS:PSUP:ACURrent measurement against their user-definable limits either ON or OFF. Note All current measurements require the MS Power Supply and Current Measurement options to be installed on your Willtek 4400. |
| Query | The query form of this command is not available. |
| Example | :CALC:PSUP:ACUR:LIM:STAT ON Switches the limit check of the average current measurement on. |

:CALCulate:PSUPply:ACURrent:LIMit:UPPer[:DATA]

| | |
|-------------------|---|
| Syntax | :CALCulate:PSUPply:ACURrent:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 1000.0. The resolution is 1, the default value is 1000.0. |

| | |
|--------------------|--|
| Description | Sets the upper limit for the average current measurement. The physical dimension of the number stated is mA. |
| Query | The query form of this command is not available. |
| Example | :CALC:PSUP:ACUR:LIM:UPP 400 Sets the upper limit of the average current measurement to 400 mA. |

:CALCulate:PSUPply:ACURrent:LIMit:LOWer[:DATA]

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:ACURrent:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 1000.0. The resolution is 1, the default value is 0.0. |
| Description | Sets the lower limit for the peak current consumption measurement. The physical dimension of the number stated is mA. |
| Query | The query form of this command is not available. |
| Example | :CALC:PSUP:ACUR:LIM:LOW 10 Sets the lower limit of the current consumption measurement to 10 mA. |

:CALCulate:PSUPply:PCURrent:LIMit[:FAIL]?

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:PCURrent:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers one boolean number, indicating whether the user-definable limits of the peak current measurement were violated by a measurement result value or not. A 0 means that all measurement result values were within the limits set, while a 1 indicates that at least one measurement result value did violate at least one of the limits. The limits can be set using the commands described below. |
| Example | :CALC:PSUP:PCUR:LIM:UPP 1000.0 :CALC:PSUP:PCURrent:LIM:LOW 0 :MEAS:ARRay:PSUP:PCURrent 5 :CALC:PSUP:PCURrent:LIM? String returned in this example: "1" This string delivered back indicates that there was a violation of the defined limits. |

:CALCulate:PSUPply:PCURrent:LIMit:STATe

| | |
|-------------------|---|
| Syntax | :CALCulate:PSUPply:PCURrent:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |

| | |
|--------------------|--|
| Description | This command switches the check of the peak current measurement result values of the MEAS:PSUP:PCURRENT measurement against their user-definable limits either ON or OFF. Note All current consumption measurements require the MS Power Supply and Current Measurement options to be installed on your Willtek 4400. |
| Query | There is no query form of this command available. |
| Example | :CALC:PSUP:PCUR:LIM:STAT ON Switches the limit check of the peak current measurement on. |

:CALCulate:PSUPply:PCURrent:LIMit:UPPer[:DATA]

| | |
|--------------------|---|
| Syntax | :CALCulate:PSUPply:PCURrent:LIMit:UPPer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 4000.0. The resolution is 1, the default value is 4000.0. |
| Description | Sets the upper limit for the peak current consumption measurement. The physical dimension of the number stated is mA. |
| Query | There is no query form of this command available. |
| Example | :CALC:PSUP:PCUR:LIM:UPP 1000.0 Sets the upper limit of the peak current consumption measurement to 1000 mA. |

:CALCulate:PSUPply:PCURrent:LIMit:LOWer[:DATA]

| | |
|--------------------|--|
| Syntax | :CALCulate:PSUPply:PCURrent:LIMit:LOWer[:DATA] <real1> |
| Parameters | real1 is a floating point real number. The minimum value is 0.0, the maximum is 4000.0. The resolution is 1, the default value is 0.0. |
| Description | Sets the lower limit for the peak current consumption measurement. The physical dimension of the number stated is mA. |
| Query | There is no query form of this command available. |
| Example | :CALC:PSUP:PCUR:LIM:LOW 10 Sets the lower limit of the peak current consumption measurement to 10 mA. |

:CALCulate:TDSCdma:RFSPectrum:ACLR:LIMit:FAIL

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFSPectrum:ACLR:LIMit:FAIL |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | The query form of this command will return the result of the check. 0 means pass. |
| Example | :CALCulate:TDSCdma:RFSPectrum:ACLR:LIMit:FAIL? |

:CALCulate:TDSCdma:RFSpectrum:ACLR:LIMit:UPPer

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:ACLR:LIMit:UPPer <Int1>,<Int2> |
| Parameters | Int1, Int2: Limit value for 3.2 MHz, 1.6 MHz. The minimum value is -80, the maximum is 0. The default value is -43. Int2: Limit value for 1.6 MHz. The minimum value is -80, the maximum is 0. The default value is -43, -33. |
| Description | Set the limit values for ACLR. |
| Query | The query form of this command will return the limit value for ACLR. |
| Example | :CALCulate:TDSCdma:RFSpectrum:ACLR:LIMit:UPPer -33,-43 Set the limit values to -33 and -43 dBc. |

:CALCulate:TDSCdma:RFSpectrum:SEM:VALue?

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:SEM:VALue? <real1> |
| Parameters | real1: frequency of the value. |
| Description | There is only a query form of this command available. |
| Query | Delivers a value to the frequency. You have to send the SCPI_Command :MEAS:TDSC:RFSP:SEM:AVG before. |
| Example | :CALCulate:TDSCdma:RFSpectrum:SEM:VALue? 3800 |

:CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:FAIL

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:FAIL |
| Parameters | There are no parameters. |
| Description | There is only a query form of this command available. |
| Query | The query form of this command will return the result of the check. 0 means pass. |
| Example | :CALCulate:TDSCdma:RFTX:SEM:LIMit:FAIL? |

:CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:FREQ

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:FREQ <real1>,<real2>,<real3>,<real4>,<real5>,<real6>,<real7>,<real8>,<real9>,<real10> |
| Parameters | realX: frequency edge point for the limit line. |
| Description | Set the frequency of the limit line. |
| Query | The query form of this command will return the limit value (frequency) for SEM. |
| Example | :CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:FREQ -4000,-2900,-2400.5,-1800,-800,800,1800,2400,2900,4000 |

:CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:MAXimum

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:MAXimum <int1>,<int2>,<int3>,<int4>,<int5>,<int6>,<int7>,<int8>,<int9>,<int10> |
| Parameters | intX: max value for the limit line. |
| Description | Set the value of the limit line. |
| Query | The query form of this command will return the limit value for SEM. |
| Example | :CALCulate:TDSCdma:RFSpectrum:SEM:LIMit:MAX -50,-50,-65,-50,-30,-30,-60,-60,-60,-60 |

:CALCulate:TDSCdma:RFSpectrum:MSpectrum:VALue?

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:MSpectrum:VALue? <real1> |
| Parameters | real1: frequency of the value. |
| Description | There is only a query form of this command available. |
| Query | Delivers a value to the frequency. You have to send the SCPI_Command :MEAS:TDSC:RFSP:MSP:AVG before. |
| Example | :CALCulate:TDSCdma:RFSpectrum:MSpectrum:VALue? -2350.0 |

:CALCulate:TDSCdma:RFSpectrum:OBW:LIMit:FAIL

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFSpectrum:OBW:LIMit:FAIL |
| Parameters | There are no parameters. |
| Description | There is only the query form available. |
| Query | The query form of this command will return the result of the check. 0 means pass. |
| Example | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:FAIL? |

:CALCulate:TDSCdma:RFTX:ONOFFpower:VALue?

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:ONOFFpower:VALue? <real1> |
| Parameters | real1: chip of the value. |
| Description | There is only the query form available. |
| Query | Delivers a value to the chip. You have to send the SCPI_Command :MEAS:TDSC:RFTX:ONOFF:AVG first. |
| Example | :CALCulate:TDSCdma:RFTX:ONOFFpower:VALue? -460.0 |

:CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:FAIL

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:FAIL |
| Parameters | There are no parameters. |
| Description | There is only the query form available. |
| Query | The query form of this command will return the result of the check. 0 is pass. |
| Example | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:FAIL? |

:CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:LOWer

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:LOWer <real1>,<real2>,<real3>,<real4> |
| Parameters | realX: value for the lower PTT limit line. |
| Description | Limits of the lower Power Time Template in pairs (x/y)=(chips/dBm). |
| Query | The query form of this command will return the lower limit value for PTT. |
| Example | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:LOWer -424, -40, 424, -40 |

:CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:UPPer

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:UPPer <real1>,<real2>,<real3>,<real4>,<real5>,<real6>, <real7>,<real8>,<real9>,<real10>,<real11>,<real12> |
| Parameters | realX: value for the upper PTT limit line |
| Description | Limits of the upper Power Time Template in pairs (x/y)=(chips/dBm). |
| Query | The query form of this command will return the upper limit value for PTT. |
| Example | :CALCulate:TDSCdma:RFTX:ONOFFpower:LIMit:UPPer -461, -65, -457, -65, -457, -50, -437, -50, 437, -65, 441, -65 |

:CALCulate:TDSCdma:RFTX:FREQ:LIMit[:FAIL]

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:FREQ:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:FREQ measurement violated its limits. |
| Example | :MEAS:TDSCDMA:ARR:RFTX:FREQ 20 :CALC:TDSC:RFTX:FREQ:LIM? The MEASurement command starts 20 frequency measurements. The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:FREQuency:LIMit:STATe

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:FREQuency:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSCD:RFTX:FREQ:LIM:STAT ON Switches the limit check for the frequency error measurement on. |

**:CALCulate:TDSCdma:RFTX:FREQuency:LIMit:UPPer
: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:FREQuency:LIMit:UPPer: [DATA] <real1> |
| Parameters | int1 is a real number. The minimum value for real1 is 0, the maximum is 1000, the default is 200. |
| Description | This command sets the upper limit of the frequency error measurement by band. The physical dimension of the number stated is Hz. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:FREQ:LIM:UPP 310 :CALC:TDSC:RFTX:FREQ:LIM:UPP? The query returns the previously set limit of 310 (hertz). |

**:CALCulate:TDSCdma:RFTX:FREQuency:LIMit:LOWer
: [DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:FREQuency:LIMit:LOWer: [DATA] <real1> |
| Parameters | int1 is a real number. The minimum value for real1 is -1000, the maximum is 0, the default is -200. |
| Description | This command sets the lower limit of the frequency error measurement by band. The physical dimension of the number stated is Hz. |
| Query | The query form returns the stored settings. |
| Example | :CALC:TDSC:RFTX:FREQ:LIM:LOW -310 :CALC:TDSC:RFTX:FREQ:LIM:LOW? The query returns the previously set limit of -310 (hertz). |

:CALCulate:TDSCdma:RFTX:MODQuality:ALL:LIMit[:FAIL]

| | |
|-------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:ALL:LIMit[:FAIL]? |
| Parameters | There are no parameters. |

| | |
|--------------------|---|
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:ALL measurement violated its limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:ALL 20 :CALC:TDSC:RFTX:MODQ:ALL:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:MODQuality:ALL:LIMit:STATE

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:ALL:LIMit:STATE <PredefExp> |
| Parameters | PredefExp may take on one of the following predefined expressions: ON OFF. The default is ON. |
| Description | This command switches the check of the measurement result values of all modulation quality measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:ALL:LIM:STAT ON Switches the limit check for all modulation quality measurement on. |

:CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit[:FAIL]

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:ERMS measurement violated its limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:ERMS 20 :CALC:TDSC:RFTX:MODQ:ERMS:LIM? Returns 1 if any of the EVM RMS vector error measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:STATE

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:STATE <PredefExp> |
| Parameters | PredefExp may take on one of the following predefined expressions: ON OFF. The default is ON. |
| Description | This command switches the check of the measurement result values of the EVM RMS vector error measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |

Example :CALC:TDSC:RFTX:MODQ:ERMS:LIM:STAT ON
Switches the limit check for the EVM RMS-averaged vector error measurement on.

:CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:UPPer
: [DATA]

Syntax :CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:UPPer
:[DATA] <Real1>

Parameters Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 17.5.

Description Sets the upper limit for the EVM RMS average vector error. The physical dimension of the number stated is percentage.

Query There is a query form which returns the stored settings.

Example :CALC:TDSC:RFTX:MODQ:ERMS:LIM:UPP 200
:CALC:TDSC:RFTX:MODQ:ERMS:LIM:UPP?
The query returns the previously set value 200.0.

:CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:LOWer
: [DATA]

Syntax :CALCulate:TDSCdma:RFTX:MODQuality:ERMS:LIMit:LOWer
:[DATA] <Real1>

Parameters Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0.

Description Sets the lower limit for the EVM RMS average vector error. The physical dimension of the number stated is percentage.

Query There is a query form which returns the stored settings.

Example :CALC:TDSC:RFTX:MODQ:ERMS:LIM:LOW 200
:CALC:TDSC:RFTX:MODQ:ERMS:LIM:LOW?
The query returns the previously set value 200.0.

:CALCulate:TDSCdma:RFTX:MODQuality:EPEAk:LIMit
[: FAIL]

Syntax :CALCulate:TDSCdma:RFTX:MODQuality:EPEAk:LIMit [: FAIL] ?

Parameters There are no parameters.

Description There is solely a query form of this command available.

Query This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:EPEA measurement violated its limits.

Example :MEAS:TDSC:ARR:RFTX:MODQ:EPEA 20
:CALC:TDSC:RFTX:MODQ:EPEA:LIM?
Returns 0 if all 20 results are within limits, or 1 if any result is outside the limits.

**:CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit
:STATe**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:EPEA:LIM:STAT ON Switches the limit check for the EVM peak vector error measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit
:UPPer: [DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit:UPPer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, and the default is value is 50.0. |
| Description | Sets the upper limit for the EVM peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:EPEA:LIM:UPP 200 :CALC:TDSC:RFTX:MODQ:EPEA:LIM:UPP? Returns 200. |

**:CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit
:LOWer: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:EPEAK:LIMit:LOWer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the lower limit for the EVM peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:EPEA:LIM:LOW 0 :CALC:TDSC:RFTX:MODQ:EPEA:LIM:LOW? Returns 0 in this example. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit
[:FAIL]**

| | |
|---------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit[:FAIL]? |
|---------------|---|

| | |
|--------------------|---|
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:MRMS measurement violated its limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:MRMS 20 :CALC:TDSC:RFTX:MODQ:MRMS:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:STATe <PredefExp> |
| Parameters | PredefExp may take on one of the following predefined expressions: ON OFF. The default is ON. |
| Description | This command switches the check of the measurement result values of the magnitude RMS vector error measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:MRMS:LIM:STAT ON Switches the limit check for the magnitude RMS-averaged vector error measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:UPPer
: [DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:UPPer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 17.5. |
| Description | Sets the upper limit for the magnitude RMS average vector error. The physical dimension of the number stated is percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:MRMS:LIM:UPP 200 :CALC:TDSC:RFTX:MODQ:MRMS:LIM:UPP? The query returns the previously set value 200.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:LOWer
: [DATA]**

| | |
|-------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MRMS:LIMit:LOWer :[DATA] |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |

| | |
|--------------------|--|
| Description | Sets the lower limit for the magnitude RMS average vector error. The physical dimension of the number stated is percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:MRMS:LIM:LOW 200 :CALC:TDSC:RFTX:MODQ:MRMS:LIM:LOW? The query returns the previously set value 200.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit
[:FAIL]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:MPEA measurement violated its limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:MPEA 20 :CALC:TDSC:RFTX:MODQ:MPEA:LIM? Returns 0 if all 20 results are within limits, or 1 if any result is outside the limits. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit
:STATe**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:MPEA:LIM:STAT ON Switches the limit check for the magnitude peak vector error measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit
:UPPer:[DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit:UPPer :[DATA] <Reall> |
| Parameters | Reall is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, and the default is value is 50.0. |
| Description | Sets the upper limit for the magnitude peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |

| | |
|----------------|--|
| Example | <pre>:CALC:TDSC:RFTX:MODQ:MPEA:LIM:UPP 200 :CALC:TDSC:RFTX:MODQ:MPEA:LIM:UPP? Returns 200.</pre> |
|----------------|--|

:CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit:LOWer:[DATA]

| | |
|--------------------|--|
| Syntax | <pre>:CALCulate:TDSCdma:RFTX:MODQuality:MPEAk:LIMit:LOWer :[DATA] <Real1></pre> |
| Parameters | Real1 is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the lower limit for the magnitude peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | <pre>:CALC:TDSC:RFTX:MODQ:MPEA:LIM:LOW 0 :CALC:TDSC:RFTX:MODQ:MPEA:LIM:LOW? Returns 0 in this example.</pre> |

:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit[:FAIL]

| | |
|--------------------|--|
| Syntax | <pre>:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit[:FAIL]?</pre> |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the :MEAS:TDSC:ARR:RFTX:MODQ:PRMS measurement violated its limits. |
| Example | <pre>:MEAS:TDSC:ARR:RFTX:MODQ:PRMS 20 :CALC:TDSC:RFTX:MODQ:PRMS:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits.</pre> |

:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:STATe

| | |
|--------------------|--|
| Syntax | <pre>:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:STATe <PredefExp></pre> |
| Parameters | PredefExp may take on one of the following predefined expressions: ON OFF. The default is ON. |
| Description | This command switches the check of the measurement result values of the Phase RMS vector error measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | <pre>:CALC:TDSC:RFTX:MODQ:PRMS:LIM:STAT ON Switches the limit check for the Phase RMS-averaged vector error measurement on.</pre> |

**:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:UPPer
: [DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:UPPer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 10.0. |
| Description | Sets the upper limit for the Phase RMS average vector error. The physical dimension of the number stated is percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PRMS:LIM:UPP 200 :CALC:TDSC:RFTX:MODQ:PRMS:LIM:UPP? The query returns the previously set value 200.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:LOWer
: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PRMS:LIMit:LOWer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the lower limit for the Phase RMS average vector error. The physical dimension of the number stated is percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PRMS:LIM:LOW 200 :CALC:TDSC:RFTX:MODQ:PRMS:LIM:LOW? The query returns the previously set value 200.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit
[: FAIL]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit[: FAIL]? |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the lower limit for the Phase RMS average vector error. The physical dimension of the number stated is percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PRMS:LIM:LOW 200 :CALC:TDSC:RFTX:MODQ:PRMS:LIM:LOW? The query returns the previously set value 200.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit
:STATe**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the measurement against their user-definable limits either ON or OFF. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PPEA:LIM:STAT ON Switches the limit check for the Phase peak vector error measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit
:UPPer: [DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit:UPPer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, and the default is value is 45.0. |
| Description | Sets the upper limit for the phase peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PPEA:LIM:UPP 200 :CALC:TDSC:RFTX:MODQ:PPEA:LIM:UPP? Returns 200. |

**:CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit
:LOWer: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:PPEAk:LIMit:LOWer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. Its minimum value is 0.0, the maximum is 200.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the lower limit for the phase peak vector error measurement. The physical dimension of the number is stated as a percentage. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:PPEA:LIM:LOW 0 :CALC:TDSC:RFTX:MODQ:PPEA:LIM:LOW? Returns 0 in this example. |

:CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit[:FAIL]

| | |
|---------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit[:FAIL]? |
|---------------|--|

| | |
|--------------------|---|
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the waveform quality measurement exceeds the limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:RHO? 1 :CALC:TDSC:RFTX:MODQ:RHO:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the waveform quality measurement against their user-definable limits either ON or OFF. |
| Query | There is no query form of this command available. |
| Example | :CALC:TDSC:RFTX:MODQ:RHO:LIM:STAT ON Switches the limit check for the waveform quality measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:UPPer
:[DATA]**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:UPPer :[DATA] <PredefExp> |
| Parameters | PredefExp is a floating point real number. The minimum value for this parameter is 0.90, the maximum is 1.0, the resolution is 0.0001, the default is value is 1.0. |
| Description | Sets the upper limit for the waveform quality measurement. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:RHO:LIM:UPP 1.0 :CALC:TDSC:RFTX:MODQ:RHO:LIM:UPP? Returns 1.0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:LOWer
:[DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:RHO:LIMit:LOWer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for this parameter is 0.0, the maximum is 1.0, the resolution is 0.0001, the default is value is 0.944. |
| Description | Sets the lower limit for the waveform quality measurement. |
| Query | There is a query form which returns the stored settings. |

| | |
|----------------|---|
| Example | :CALC:TDSC:RFTX:MODQ:RHO:LIM:LOW 0.944 :CALC:TDSC:RFTX:MODQ:RHO:LIM:LOW? Returns 0.944. |
|----------------|---|

**:CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit
[:FAIL]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit [:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the origin offset measurement exceeds the limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:IQOF? 20 :CALC:TDSC:RFTX:MODQ:IQOF:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

**:CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit
:STATe**

| | |
|--------------------|---|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit :STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. The default is ON. |
| Description | This command switches the check of the measurement result values of the origin offset measurement against their user-definable limits either ON or OFF. |
| Query | There is no query form of this command available. |
| Example | :CALC:TDSC:RFTX:MODQ:IQOF:LIM:STAT ON Switches the limit check for the origin offset measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit
:UPPer: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit:UPPer :[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the minimum is 0.0, the resolution is 0.1, the default is value is 0.0. |
| Description | Sets the upper limit for the origin offset measurement. The physical dimension of the number stated is dBc. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:IQOF:LIM:UPP 0 :CALC:TDSC:RFTX:MODQ:IQOF:LIM:UPP? Returns 0 in this case. |

:CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit:LOWer:[DATA]

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQOffset:LIMit:LOWer:[DATA] <Real1> |
| Parameters | Real1 is a floating point real number. Its minimum value is -99.0, the maximum is 0.0, the resolution is 0.1, the default is value is -99.0. |
| Description | Sets the lower limit for the origin offset measurement. The physical dimension of the number stated is dBc. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:IQOF:LIM:LOW -60.0 :CALC:TDSC:RFTX:MODQ:IQOF:LIM:LOW? Returns -60.0. |

:CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit[:FAIL]

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit[:FAIL]? |
| Parameters | There are no parameters. |
| Description | There is solely a query form of this command available. |
| Query | This command delivers a boolean number in a string which indicates if the IQ imbalance measurement exceeds the limits. |
| Example | :MEAS:TDSC:ARR:RFTX:MODQ:IQIM? 20 :CALC:TDSC:RFTX:MODQ:IQIM:LIM? The query returns 1 if any of the measurement results was out of limits, or 0 if none of the results was out of limits. |

:CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit:STATe

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is ON. |
| Description | This command switches the check of the measurement result values of the IQ imbalance measurement against their user-definable limits either ON or OFF. |
| Query | There is no query form of this command available. |
| Example | :CALC:TDSC:RFTX:MODQ:IQIM:LIM:STAT ON Switches the limit check for the IQ imbalance measurement on. |

**:CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:
 LIMit:UPPer: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit :UPPer: [DATA] <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for parameter is 0.0, the maximum is 0.0, the resolution is 0.1, the default is value is 0.0. |
| Description | This command sets the upper limit for the IQ imbalance measurement. The physical dimension of the number stated is dB. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:IQIM:LIM:UPP 0 :CALC:TDSC:RFTX:MODQ:IQIM:LIM:UPP? Returns 0. |

**:CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:
 LIMit:LOWer: [DATA]**

| | |
|--------------------|--|
| Syntax | :CALCulate:TDSCdma:RFTX:MODQuality:IQIMbalance:LIMit :LOWer: [DATA] <PredefExp> |
| Parameters | PredefExp is a floating point real number. The minimum value is -99.0, the maximum is 0.0, the resolution is 0.1, the default is value is -99.0. |
| Description | This command sets the lower limit for the IQ imbalance measurement. The physical dimension of the number stated is dB. |
| Query | There is a query form which returns the stored settings. |
| Example | :CALC:TDSC:RFTX:MODQ:IQIM:LIM:LOW -99.0 :CALC:TDSC:RFTX:MODQ:IQIM:LIM:LOW? Returns -99.0. |

RFGenerator subsystem

The RFG subsystem controls the accessible parameters of the RF generator.

Important notes:

- The RF generator can only be used if all communication systems have been switched off (and unloaded) before.
- The RF generator functionality of the 4400 will enable you to provide a base channel to allow the mobile under test to synchronize to the base station. However, as long as the RF generator is active, there will be **no call setup** and **no reaction to signaling**.
Some of the data transmitted by the 4400 in the base channel can be set or altered using the SCPI commands described in section ["CONFigure subsystem" on page 244](#).
- The RF generator and analyzer are enabled using the appropriate :CONFigure:CSYSstem command.

:RFGenerator:TDSCdma:STATE

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:STATE <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the RF generator ON or OFF. While the Willtek 4400 is in RF generator mode, there is no signalling active. This means that the Willtek 4400 may be used as RF signal generator for all kinds of RF and TDSCDMA signals. While working as a RF generator the Willtek 4400 does not respond to any messages sent by the mobile nor does the test set expect the mobile under test to react in any way. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFGenerator:TDSCdma:STATE ON :RFG:STAT? Value returned in this example "ON". |

:RFGenerator:TDSCdma:LEVel

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:LEVel |
| Parameters | real1 is a floating point real number. The minimum value for real1 is -120.0, the maximum value is 0.0. The minimum resolution for real1 is 0.1. The default value for real1 is -70.0. |
| Description | This command sets the RF output power level of Willtek 4400. The value specified for real1 is the power output level in dBm. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |
| Example | :RFGenerator:TDSCdma:LEVel -50.5 :RFG:TDSCdma:LEV? Value returned in this example: "-50.5". |

:RFGenerator:TDSCdma:FREQuency

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:FREQuency <real1> |
| Parameters | real1 is a floating point real number. The minimum value for real1 is 430,000,000, the maximum value is 2,300,000,000. The minimum resolution for real1 is 10. The default value for real1 is 1,910,800,000. |
| Description | This command sets the RF generator's frequency. The value specified for real1 is the output frequency in Hz. Note: For more details on the TDSCdma frequency ranges, please refer to section Basic Specifications of TDSCdma Bands. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |

| | |
|----------------|--|
| Example | <code>:RFGenerator:TDSCdma:FREQuency 880200010</code> <code>:RFG:TDSCdma:FREQ?</code> Value returned in this example: "880200010". |
|----------------|--|

:RFGenerator:TDSCdma:UARFcn

| | |
|--------------------|---|
| Syntax | <code>:RFGenerator:TDSCdma:UARFcn <int1></code> |
| Parameters | <code>int1</code> is an integer number. The minimum value is 9254 the maximum value 10121. The initial default is 9554. |
| Description | This command will set the downlink channel number of the 4400. |
| Query | The query form of this command will return the current setting. The string delivered will contain one integer. |
| Example | <code>:RFG:TDSC:UARF 9555</code> <code>:RFG:TDSC:UARF?</code> Value returned in this example: 9555. |

:RFGenerator:TDSCdma:FOFFset

| | |
|--------------------|--|
| Syntax | <code>:RFGenerator:TDSCdma:FOFFset <int1></code> |
| Parameters | <code>int1</code> is an integer number. The minimum value is -100000, the maximum value +100000. The initial default is 0. |
| Description | This command will set the downlink frequency offset of the 4400. |
| Query | The query form of this command will return the current setting. The string delivered will contain one integer. |
| Example | <code>:RFG:TDSC:FOFF 30000</code> <code>:RFG:TDSC:FOFF?</code> Value returned in this example: 30000. |

:RFGenerator:TDSCdma:BAND?

| | |
|--------------------|--|
| Syntax | <code>:RFGenerator:TDSCdma:BAND?</code> |
| Parameters | There are no parameters. |
| Description | The query form is supported only. |
| Query | This query determines the current signal band (based on the current UARFCN). |
| Example | <code>:RFG:TDSC:BAND?</code> Value returned in this example: 2. |

:RFGenerator:TDSCdma:TYPE

| | |
|-------------------|---|
| Syntax | <code>:RFGenerator:TDSCdma:TYPE <PredefExp></code> |
| Parameters | <code>PredefExp</code> is one of the following predefined expressions: <code>CW BURSt TDSCdma</code> . Default is <code>CW</code> . |

| | |
|--------------------|--|
| Description | Sets the type of generator signal. |
| Query | The query form of this command returns the current setting. |
| Example | :RFG:TDSC:TYPE TDSC :RFG:TDSC:TYPE? Value returned in this example "TDSC". |

:RFGenerator:TDSCdma:MODulation

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:MODulation <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: NONE QPSK. Default is NONE. |
| Description | Selects the modulation. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:MOD QPSK :RFG:TDSC:MOD? Value returned in this example "QPSK". |

:RFGenerator:TDSCdma:BITPattern

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:BITPattern <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBleonezer ONEZero FOURonezero EIGHTonezero. Default is PRBS9. |
| Description | Sets up the physical data pattern for burst/CW. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:BITP ALLO :RFG:TDSC:BITP? Value returned in this example "ALLO". |

:RFGenerator:TDSCdma:DLSLots

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:DLSLots <int1> |
| Parameters | int1 is an integer number. The minimum value is 1, the maximum value 6. The initial default is 1. |
| Description | Sets the number of downlink slots. |
| Query | The query form of this command returns the current setting. A query will return a int. |
| Example | :RFG:TDSC:DLSL 1 :RFG:TDSC:DLSL? Value returned in this example 1. |

:RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:ONOFF

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:ONOFF <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the primary common control physical channel (P-CCPCH) in the first slot on or off. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:PCCP:ONOF ON :RFG:TDSC:TDSC:FSL:PCCP:ONOF? Value returned in this example: "ON". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:POWer

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |
| Description | Sets up the relative power of the P-CCPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a real value. |
| Example | :RFG:TDSC:TDSC:FSL:PCCP:POW -20.0 :RFG:TDSC:TDSC:FSL:PCCP:POW? Value returned in this example: -20.0. |

:RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:DATA

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:DATA <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBLeonezer ONEZero FOURonezero. Default is PRBS9. |
| Description | Sets up the physical data pattern for the P-CCPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:PCCP:DAT PRBS15 :RFG:TDSC:TDSC:FSL:PCCP:DAT? Value returned in this example: "PRBS15". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:SCCPch:ONOFF

| | |
|---------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:SCCPch:ONOFF <PredefExp> |
|---------------|--|

| | |
|--------------------|---|
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the secondary common control physical channel (S-SSPCH) in the first slot on or off. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:SCCP:ONOF ON :RFG:TDSC:TDSC:FSL:SCCP:ONOF? Value returned in this example "ON". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:SCCPch:POWer

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:PCCPch:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |
| Description | Sets up the relative power of the S-CCPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a real. |
| Example | :RFG:TDSC:TDSC:FSL:SCCP:POW -20.0 :RFG:TDSC:TDSC:FSL:SCCP:POW? Value returned in this example -20.0. |

:RFGenerator:TDSCdma:TDSCdma:FSLot:SCCPch:DATA

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:SCCPch:DATA <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBLeonezer ONEZero FOURonezero. Default is PRBS9. |
| Description | Sets up the physical data pattern for the S-CCPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:SCCP:DAT ONEZ :RFG:TDSC:TDSC:FSL:SCCP:DAT? Value returned in this example "ONEZ". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:ONOFF

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:ONOFF <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the dedicated physical channel in the first slot on and off. |

| | |
|----------------|---|
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:DPCH:ONOF ON :RFG:TDSC:TDSC:FSL:DPCH:ONOF? Value returned in this example "ON". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:POWer

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |
| Description | Sets up the relative power of the DPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a real. |
| Example | :RFG:TDSC:TDSC:FSL:DPCH:POW 10 :RFG:TDSC:TDSC:FSL:DPCH:POW? Value returned in this example 10. |

:RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:DATA

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:DATA <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBleonezer ONEZero FOURonezero. Default is PRBS9. |
| Description | Set up the physical data pattern for the DPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:FSL:SCCP:DAT DOUB :RFG:TDSC:TDSC:FSL:SCCP:DAT? Value returned in this example "DOUB". |

:RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:SFACTOR

| | |
|--------------------|--|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:FSLot:DPCH:SFACTOR <int1> |
| Parameters | int1 is an integer number. The minimum value is 1, the maximum value 16. The initial default is 1. |
| Description | Sets up the spreading factor of the DPCH in the first slot. |
| Query | The query form of this command returns the current setting. A query will return a int. |
| Example | :RFG:TDSC:TDSC:FSL:DPCH:SFAC 1 :RFG:TDSC:TDSC:FSL:DPCH:SFAC? Value returned in this example 1. |

:RFGenerator:TDSCdma:TDSCdma:DWPTs:ONOFF

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:DWPTs:ONOFF <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the downlink pilot time slot (DwPTS) on or off. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:DWPT:ONOF ON :RFG:TDSC:TDSC:DWPT:ONOF? Value returned in this example "ON". |

:RFGenerator:TDSCdma:TDSCdma:DWPTs:POWer

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:DWPTs:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |
| Description | Sets up the relative power of the DwPTS. |
| Query | The query form of this command returns the current setting. A query will return a real value. |
| Example | :RFG:TDSC:TDSC:DWPT:POW -20.0 :RFG:TDSC:TDSC:DWPT:POW? Value returned in this example -20.0. |

:RFGenerator:TDSCdma:TDSCdma:ASLots:SCCPch:ONOFF

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:ASLots:SCCPch:ONOFF <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the secondary common control physical channel (S-CCPCH) in the additional slots on or off. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:ASL:SCCP:ONOF ON :RFG:TDSC:TDSC:ASL:SCCP:ONOF? Value returned in this example "ON". |

:RFGenerator:TDSCdma:TDSCdma:ASLot:SCCPch:POWer

| | |
|-------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:ASLot:SCCPch:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |

| | |
|--------------------|--|
| Description | Set up the relative power of the S-CCPCH in the additional slots. |
| Query | The query form of this command returns the current setting. A query will return a real. |
| Example | :RFG:TDSC:TDSC:ASL:SCCP:POW -40 :RFG:TDSC:TDSC:ASL:SCCP:POW? Value returned in this example -40.0. |

:RFGenerator:TDSCdma:TDSCdma:ASLot:SCCPch:DATA

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:ASLot:SCCPch:DATA <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBLeon- ezer ONEZero FOURonezero. Default is PRBS9. |
| Description | Set up the physical data pattern for the S-CCPCH in the additional slots. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:ASL:SCCP:DAT PRBS9 :RFG:TDSC:TDSC:ASL:SCCP:DAT? Value returned in this example "PRBS9". |

:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:ONOFF

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:ONOFF <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF. Default is OFF. |
| Description | Switches the dedicated physical channel (DPCH) in the additional slots on or off. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFG:TDSC:TDSC:ASL:DPCH:ONOF ON :RFG:TDSC:TDSC:ASL:DPCH:ONOF? Value returned in this example "ON". |

:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:POWer

| | |
|--------------------|---|
| Syntax | :RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:POWer <real1> |
| Parameters | real1 is a real value. The minimum value is -100.0, the maximum value 100.0. The initial default is 0. |
| Description | Set up the relative power of the DPCH in the additional slots. |
| Query | The query form of this command returns the current setting. A query will return a real. |

| | |
|----------------|---|
| Example | <pre>:RFG:TDSC:TDSC:ASL:DPCH:POW -50.0 :RFG:TDSC:TDSC:ASL:DPCH:POW? Value returned in this example -50.0.</pre> |
|----------------|---|

:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:DATA

| | |
|--------------------|--|
| Syntax | <pre>:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:DATA <PredefExp></pre> |
| Parameters | <p>PredefExp is one of the following predefined expressions: PRBS9 PRBS15 PRBS23 ALLZero ALLOne DOUBLeonezer ONEZero FOURonezero. Default is PRBS9.</p> |
| Description | Set up the physical data pattern for the DPCH in the additional slots. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | <pre>:RFG:TDSC:TDSC:ASL:DPCH:DAT FOUR :RFG:TDSC:TDSC:ASL:DPCH:DAT? Value returned in this example "FOUR".</pre> |

:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:SFACTOR

| | |
|--------------------|--|
| Syntax | <pre>:RFGenerator:TDSCdma:TDSCdma:ASLot:DPCH:SFACTOR <int1></pre> |
| Parameters | <p>int1 is an integer number. The minimum value is 1, the maximum value 16. The initial default is 1.</p> |
| Description | Sets up the spreading factor of the DPCH in the additional slots. |
| Query | The query form of this command returns the current setting. A query will return a int. |
| Example | <pre>:RFG:TDSC:TDSC:ASL:DPCH:SFAC 2 :RFG:TDSC:TDSC:ASL:DPCH:SFAC? Value returned in this example: 2.</pre> |

:RFGenerator:TDSCdma:TDSCdma:SCODE

| | |
|--------------------|--|
| Syntax | <pre>:RFGenerator:TDSCdma:TDSCdma:SCODE <int1></pre> |
| Parameters | <p>int1 is an integer number. The minimum value is 0, the maximum value 127. The initial default is 0.</p> |
| Description | Selects the scrambling code. |
| Query | The query form of this command returns the current setting. A query will return a int. |
| Example | <pre>:RFG:TDSC:TDSC:SCOD 0 :RFG:TDSC:TDSC:SCOD? Value returned in this example: 0.</pre> |

RFANalyser subsystem

The RFAN subsystem controls the accessible parameters of the RF analyzer.

:RFANalyser:TDSCdma:FREQuency

| | |
|--------------------|---|
| Syntax | :RFANalyser:TDSCdma:FREQuency <Real1> |
| Parameters | Real1 is a floating point real number. The minimum value for Real1 is 430,000,000, the maximum value is 2,300,000,000. The minimum resolution for real1 is 10. The default value is 1,9010. |
| Description | This command sets the RF analyzer's center frequency. The value specified is the center frequency in Hz. The frequency of the internal synthesizer can be changed in steps of 10 Hz. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |
| Example | :RFANalyser:TDSCdma:FREQuency 1950000000 :RFAN:TDSCdma:FREQ? Value returned in this example: 1950000000. |

:RFANalyser:TDSCdma:UARFCn

| | |
|--------------------|---|
| Syntax | :RFANalyser:TDSCdma:UARFCn <Int1> |
| Parameters | Int1 is a integer number. The minimum value for Int1 is 9254, the maximum value is 10121. The default value is 9554. |
| Description | This command sets the UARFCN. It changes the frequency. |
| Query | The query form of this command will return the UARFCN calculated from the frequency. If no valid UARFCN is attached to the frequency, it returns 0. |
| Example | :RFANalyser:TDSCdma:UARFCn 9554 :RFAN:TDSCdma:UARF? Value returned in this example: 9554. |

:RFAnalyzer:TDSCdma:ULSLot

| | |
|--------------------|--|
| Syntax | :RFAnalyzer:TDSCdma:ULSLot <Int1> |
| Parameters | Int1: The uplink slot. The minimum value is 1, the maximum is 6. |
| Description | The uplink slot which should be measured. |
| Query | The query form of this command returns the current setting. A query will return a string, containing the number of the slot. |
| Example | :RFAN:TDSC:ULSL 2 :RFAN:TDSC:BITP? Value returned in this example "2". |

:RFAnalyser:TDSCdma:SCODE

| | |
|--------------------|--|
| Syntax | :RFAnalyser:TDSCdma:SCODE <Int1> |
| Parameters | Int1 is a integer number. The minimum value for Int1 is 0, the maximum value is 127. The default value is 0. |
| Description | This command sets the scrambling code. |
| Query | The query form of this command will return the scrambling code. |
| Example | :RFAnalyser:TDSCdma:SCODE 1 :RFAN:TDSCdma:SCOD? Value returned in this example: 1. |

:RFAnalyser:TDSCdma:SYNC

| | |
|--------------------|--|
| Syntax | :RFAnalyser:TDSCdma:SYNC |
| Parameters | There are no parameters. |
| Description | Only the query form is available. |
| Query | The query form of this command will return the sync status of the last measure. If no measure was started, it returns NOSync |
| Example | :RFAnalyser:TDSCdma:SYNC? Value returned in this example: SYNC. |

RFSpectrum subsystem

The RFSP subsystem controls the accessible parameters of the RF modulation spectrum analyzer.

:RFSpectrum:TDSCdma:MSpectrum:RBW

| | |
|--------------------|---|
| Syntax | :RFSpectrum:TDSCdma:MSpectrum:RBW <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: RES15 RES30 RES75. Default is RES15. |
| Description | Sets the spectral resolution of the modulation spectrum analyzer. RES30 will set a resolution of 30 kHz. RES15 will set a resolution of 15 kHz. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFSpectrum:TDSCdma:MSpectrum:RBW RES15 :RFSP:TDSCdma:MSP:RES? Value returned in this example: "RES15". |

:RFSPpectrum:TDSCdma:MSpectrum:SPAN

| | |
|--------------------|--|
| Syntax | :RFSPpectrum:TDSCdma:MSpectrum:SPAN <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: SP24 SP48. Default is SP24. |
| Description | Sets the span (i.e. the spectral bandwidth) of the modulation spectrum analyzer. SP24 will set a span of 2.4 MHz (+/- 1.2 MHz), SP48 will set a span of 4.8 MHz (+/- 2.4 MHz). |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :RFSPpectrum:TDSCdma:MSpectrum:SPAN SP24 :RFSP:TDSCdma:MSP:SPAN? Value returned in this example "SP24". |

AFGenerator subsystem

The AFG subsystem controls the accessible parameters of the audio generator. Please note that all commands of this subsystem require the Audio Option to be installed on your 4400.

:AFGenerator:STATe

| | |
|--------------------|---|
| Syntax | <code>:AFGenerator:STATe <PredefExp></code> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is OFF . |
| Description | Switches the AF (audio frequency) generator on or off. Note: This command requires the Audio Option to be installed on your Willtek 4400. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | <code>:AFGenerator:STATe ON</code> <code>:AFG:STAT?</code> Value returned: "ON". |

:AFGenerator:MODE

| | |
|--------------------|--|
| Syntax | <code>:AFGenerator:MODE <PredefExp></code> |
| Parameters | PredefExp is one of the following predefined expressions: SINGLE REPLay . Default is SINGLE . |
| Description | This command sets the AF generator either to SINGLE tone generator mode or to wave file REPLay generator. |
| Query | The query form of this command returns the current setting. A query will return a string containing the short-form version of one of the predefined expressions explained above. |
| Example | <code>:AFGenerator:MODE REPLay</code> <code>:AFG:MODE?</code> Value returned: "REPL". |

:AFGenerator:OUTPut

| | |
|-------------------|--|
| Syntax | <code>:AFGenerator:OUTPut <PredefExp></code> |
| Parameters | PredefExp is one of the following predefined expressions: AF CODeC . Default is AF . |

| | |
|--------------------|---|
| Description | <p>This command routes the signal of the AF generator. AF means that the generator's signal will be available at the AF out connector, located on the front panel of the Willtek 4400. CODec means that the signal created by the AF generator will be output to the (internal) codec of the Willtek 4400.</p> <p>Notes</p> <ul style="list-style-type: none"> - RF cannot be signal source and generator destination the same time. Please check the current setting of the <code>AFG:OUTP</code> command before issuing the <code>AFAN:INP</code> command and vice versa to avoid measurement conflicts. - The codec is an option to the Willtek 4400. Please refer to section "Accessories and options" on page 29 for details. |
| Query | <p>The query form of this command returns the current setting. A query will return a string, containing the short-form version of one of the predefined expressions explained above.</p> |
| Example | <pre>:AFGenerator:OUTPut CODec :AFG:OUTP? Value returned: "COD".</pre> |

:AFGenerator:LEVel[:RELative]:AF

| | |
|--------------------|--|
| Syntax | <code>:AFGenerator:LEVel[:RELative]:AF <real1></code> |
| Parameters | <p>real1 is a floating point real number. The minimum value for real1 is 0.000, the maximum value is 4.000. The resolution for real1 is 0.001. The default value for real1 is 0.000.</p> |
| Description | <p>This command sets the output voltage of the signals, available at the "AF out" connector, located on the front panel of the Willtek 4400.</p> <p>The value specified for real1 is the peak voltage of the AF signal.</p> <p>Notes</p> <ul style="list-style-type: none"> - This command will only have an effect when AF has been selected as the output with the <code>AFG:OUTP</code> command explained above. - The maximum output voltage for sinusoidal signals is 4 V (rms) (that is equivalent to 5.6 V peak). |
| Query | <p>The query form of this command will return the current setting. The string delivered back will contain one floating point real number.</p> |
| Example | <pre>:AFGenerator:LEVel[:RELative]:AF 2.5 :AFG:LEV:AF? Value returned in this example: "2.5".</pre> |

:AFGenerator:LEVel[:RELative]:RF

| | |
|-------------------|---|
| Syntax | <code>:AFGenerator:LEVel[:RELative]:RF <real1></code> |
| Parameters | <p>real1 is a floating point real number. The minimum value for real1 is -55.0, the maximum value is 0.0. The minimum resolution for real1 is 0.1. The default value for real1 is -6.0.</p> |

| | |
|--------------------|--|
| Description | This command sets the 'modulation depth' of the audio signal in case it is output through the codec and thus used to modulate the RF output of the Willtek 4400. The physical dimension of real1 is dBFS (dB full scale). Notes <ul style="list-style-type: none"> - A setting of 0 dB means that the audio signal will use the full input range of the codec, while -6 dB for instance mean that only half the input range will be used. - This command will only have an effect when CODEc has been selected as the output with the AFG:OUTP command explained above. - The codec is an option to the Willtek 4400. Please refer to section "Accessories and options" on page 29 for details. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |
| Example | :AFGenerator:LEVel[:RELative]:RF -12.0 :AFG:LEV:RF? Value returned: "-12.0". |

:AFGenerator:SINGLE:FREQUENCY

| | |
|--------------------|---|
| Syntax | :AFGenerator:SINGLE:FREQUENCY <real1> |
| Parameters | real1 is a floating point real number. The minimum value for real1 is 20.0 , the maximum value is 20000.0 . The minimum resolution for real1 is 0.1 . The default value for real1 is 1000.0 . |
| Description | This command sets the frequency of the AF generator as long as it is run in single-tone generation mode. The physical dimension of real1 is Hertz. Note: Please refer to the description of the :AFG:MODE command for further details regarding the single-tone generation mode. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |
| Example | :AFGenerator:SINGLE:FREQUENCY 2500 :AFG:SING:FREQ? Value returned: "2500.0". |

:AFGenerator:SINGLE:SHAPE

| | |
|--------------------|---|
| Syntax | :AFGenerator:SINGLE:SHAPE <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: SINE RECTangle TRIangle POSitivep NEGativep . Default is SINE . |
| Description | Sets the waveform of the AF signal to be generated. SINE selects a sine wave, while RECTangle will generate a rectangular waveform. TRIangle selects a triangular waveform. POSitivep will generate a sawtooth with a rising (positive) slope, while NEGativep switches to a sawtooth with a falling (negative) slope. Note: This command will only work in single-tone generation mode. Please refer to the description of the :AFG:MODE command for further details regarding the single-tone generation mode. |
| Query | The query form of this command returns the current setting. A query will return a string, containing the short-form version of one of the predefined expressions explained above. |

| | |
|----------------|--|
| Example | :AFGenerator:SINGLE:SHAPE TRIangular :AFG:SING:SHAP? Value returned: "TRI". |
|----------------|--|

:AFGenerator:REPLay:DOWNload:FILE

| | |
|--------------------|--|
| Syntax | :AFGenerator:REPLay:DOWNload:FILE <string> |
| Parameters | string is the name of the audio file to be loaded from the /rapid/wav directory. |
| Description | Loads an audio file (*.wav) to be replayed. Note: Wave files to be replayed over the codec and the RF must be sampled at 8 kHz. Wave files for replay over the audio interface (AF out) must be available with a sampling rate of either 16 or 48 kHz. |
| Query | The query form of this command returns the current setting. A query will return a string containing the loaded file name. |
| Example | :AFG:REPL:DOWN:FILE efr1.wav Loads file ef1.wav from the /rapid/wav directory to the wave file replay function. |

:AFGenerator:REPLay:REPeat

| | |
|--------------------|--|
| Syntax | :AFGenerator:REPLay:REPeat <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is ON . |
| Description | Selects whether the samples shall play once or repetitively. |
| Query | The query form of this command returns the current setting. A query will return a string containing one of the predefined expressions explained above. |
| Example | :AFG:REPL:REP ON :AFG:REPL:REP? Value returned: is "ON". |

:AFGenerator:AUXout:STATe

| | |
|--------------------|--|
| Syntax | :AFGenerator:AUXout:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is OFF . |
| Description | Switches the routing of AF signals to the "AUX 4 (AUX out)" of the Willtek 4400 either on or off. The source of the signal to be output to the auxiliary output connector can be selected using the :AFG:SPEA:SOUR command as described below. |
| Query | The query form of this command returns the current setting. A query will return a string containing one of the predefined expressions explained above. |
| Example | :AFGenerator:AUXout:STATe ON :AFG:AUXout:STAT? Value returned: "ON". |

:AFGenerator:AUXout:SOURce

| | |
|--------------------|--|
| Syntax | :AFGenerator:AUXout:SOURce <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: AFIN AFOUt . Default is AFIN . |
| Description | This command routes signals coming from a selectable source to the "AUX 4 (AUX out)" of the Willtek 4400. AFIN means that the signal applied to the Willtek 4400 AF in connector will be made available at the auxiliary output connector, while AFOUt will route the signals from the AF out connector to AUX out. This can be signals of the internal AF generator, but also signals coming from the codec of the Willtek 4400. Notes <ul style="list-style-type: none"> - The audio generator and the codec are options to the Willtek 4400. For further details, please refer to section "Accessories and options" on page 29. - For further details regarding the connectors of the Willtek 4400, please refer to section "Connectors" on page 9. - Please note that the auxiliary output needs to be switched on first (see command AFG:SPE:STAT above for details). |
| Query | The query form of this command returns the current setting. A query will return a string, containing the short-form version of one of the predefined expressions explained above. |
| Example | :AFGenerator:AUXout:SOURce AFOUt :AFG:SPE:SOUR? Value returned: "AFOU". |

:AFGenerator:AUXout:VOLume

| | |
|--------------------|--|
| Syntax | :AFGenerator:AUXout:VOLume <int1> |
| Parameters | int1 is an integer. The minimum value for int1 is 0, the maximum value is 100. The default value for int1 is 0. |
| Description | This command sets the output volume of the external speaker connected to the Willtek 4400 using the "AUX 4 (AUX out)". int1 represents a relative volume in percentages of the maximum volume. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one integer. |
| Example | :AFGenerator:AUXout:VOLume 20 :AFG:SPE:VOL? Value returned: "20". This means that the volume at the auxiliary output is at 20% of its maximum. |

AFANalyser subsystem

The AFAN subsystem controls the accessible parameters of the AF analyzer. Please note that all commands of this subsystem require the Audio Option to be installed on your 4400.

:AFANalyser:INPut

| | |
|--------------------|---|
| Syntax | :AFANalyser:INPut <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: AF AUXin CODec . Default is AF . |
| Description | <p>Selects the input signal for the AF analyzer of the Willtek 4400.</p> <p>AF means that the signals applied to the test set's "AF in" connector will be forwarded to the AF analyzer, while</p> <p>AUXin means that the signals applied to the test set's AUX in input will be forwarded to the AF analyzer. The AUX in input is available on the AUX 4 connector on the back panel of the Willtek 4400 (RF section).</p> <p>CODec finally means that the output of the codec (i.e. the AF data transmitted by the mobile under test to the Willtek 4400) will be analyzed.</p> <p>Notes</p> <ul style="list-style-type: none"> - This command requires the Audio Option to be installed on your Willtek 4400. - The codec is an option to the Willtek 4400. For details, please refer to section "Accessories and options" on page 29. - Input voltage limitations apply on the AF in and the AUX in connectors of the Willtek 4400. Please refer to section "Connectors" on page 9 for details. - RF cannot be signal source and generator destination at the same time. Please check the current setting of the AFG:OUTP command before issuing the AFAN:INP COD command and vice versa to avoid measurement conflicts. |
| Query | The query form of this command returns the current setting. A query will return a string, containing the short-form version of one of the predefined expressions explained above. |
| Example | <p>:AFANalyser:INPut CODec</p> <p>:AFAN:INP? Value returned: "COD".</p> |

:AFANalyser:SAMPlerate

| | |
|--------------------|--|
| Syntax | :AFANalyser:SAMPlerate <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: KHZ8 KHZ11 KHZ22 KHZ33 KHZ44 KHZ48 . Default is KHZ48 . |
| Description | <p>Selects the fixed sampling rate for the AF analyzer's input signals.</p> <p>KHZ8 means a sampling rate of 8 kHz,</p> <p>KHZ11 a sampling rate of 11 kHz,</p> <p>KHZ22 a sampling rate of 22 kHz,</p> <p>KHZ33 a sampling rate of 33 kHz,</p> <p>KHZ44 a sampling rate of 44.1 kHz, while</p> <p>KHZ48 stands for a sampling rate of 48 kHz.</p> |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | <p>:AFANalyser:SAMPlerate KHZ22</p> <p>:AFAN:SAMP? Value returned: "KHZ22".</p> <p>This means a fixed sampling rate of 22 kHz.</p> |

:AFANalyser:COUPling

| | |
|---------------|----------------------------------|
| Syntax | :AFANalyser:COUPling <PredefExp> |
|---------------|----------------------------------|

| | |
|--------------------|--|
| Parameters | PredefExp is one of the following predefined expressions: AC DC DCLP . Default is AC . |
| Description | This command allows to select between AC and DC measurements of the audio signal applied to the "AF in" connector of the Willtek 4400. If AC is selected, the audio analyzer measures the AC signal only, i.e. any DC component in the signal is filtered out before the measurement. With the DC setting, the signal is measured directly, including any AC and DC components. If DCLP is chosen, the AC components will be filtered out and the Willtek 4400 only measures the DC component of the signal. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :AFANalyser:COUPLING DC :AFAN:COUP? Value returned: "DC". |

:AFANalyser:BALanced

| | |
|--------------------|---|
| Syntax | :AFANalyser:BALanced <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is OFF . |
| Description | Selects whether "AF in" shall be used as an unbalanced input (setting OFF) or as a balanced input (setting ON). Note: While a balanced signal requires a differential amplifier as an input stage, an unbalanced input doesn't. Balanced signal transmission usually is substantially less sensitive to electromagnetic interference. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :AFANalyser:BALanced OFF :AFAN:BAL? Value returned: "OFF". This means that the AF in socket will be used as a standard input for 'grounded' signals. |

:AFANalyser:VRANge

| | |
|-------------------|---|
| Syntax | :AFANalyser:VRANge <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: AUTO V30 V3 MV300 MV30 . Default is AUTO . |

| | |
|--------------------|---|
| Description | <p>This command sets the range of the expected input voltage on the Willtek 4400 "AF in" connector.</p> <p>AUTO means that the Willtek 4400 will take sample measurements first and then – based on these measurement results – will decide automatically which setting to use.</p> <p>V30 means that an input voltage of up to 30 volts (rms) is expected and that the resolution of the signal measured will be 100 mV.</p> <p>V3 means that an input voltage of up to 3 volts (rms) is expected and that the resolution of the signal measured will be 10 mV.</p> <p>MV300 expects a maximum input voltage of 300 mV (rms) and delivers a resolution of 1 mV, while</p> <p>MV30 means that the maximum input voltage will be 30 mV (rms) and the resolution will be 100 µV.</p> <p>Notes</p> <ul style="list-style-type: none"> – The better the current input voltage corresponds to the range setting performed with this command, the more precise the measurement results will be. – Whenever fast availability of measurement results is an issue, AUTO ranging should be avoided as it is time-consuming. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions as explained above. |
| Example | <pre>:AFANalyser:VRANge V30 :AFAN:VRAN? Value returned: "V30".</pre> |

:AFANalyser:MRANge

| | |
|--------------------|---|
| Syntax | :AFANalyser:MRANge <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: AUTO V1 MV100 . Default is AUTO . |
| Description | <p>This command sets the range of the expected input voltage on the Willtek 4400 "AUX 4 (AUX in)" connector.</p> <p>AUTO means that the Willtek 4400 will take sample measurements first and then – based on these measurement results – will decide automatically which setting to use.</p> <p>V1 means that an input voltage of up to 1 Volt (rms) is expected and that the resolution will be 1 mV.</p> <p>MV100 expects a maximum input voltage of 100 mV (rms) and delivers a resolution of 100 µV.</p> <p>Notes</p> <ul style="list-style-type: none"> – The better the current input voltage corresponds to the range setting performed with this command, the more precise the measurement results will be. – Whenever fast availability of measurement results is an issue, AUTO ranging should be avoided as it is time-consuming. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions as explained above. |
| Example | <pre>:AFANalyser:MRANge MV100 :AFAN:MRAN? Value returned: "MV100".</pre> |

:AFANalyser:FILTer

| | |
|-------------------|--|
| Syntax | :AFANalyser:FILTer <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: NONE CCITt CMESsage . Default is NONE . |

| | |
|--------------------|--|
| Description | Selects the type of filter to be inserted between the signal input and the AF analyzer. NONE means that there will be no filtering. CCITT selects a speech filter as defined by CCITT regulations. CMESsage is a speech filter according to US standards. |
| Query | The query form of this command returns the current setting. A query will return a string, containing the short-form version of one of the predefined expressions explained above. |
| Example | :AFANalyser:FILTer NONE :AFAN:FILT? Value returned: " NONE ". |

:AFANalyser:SPECTrum:RLEVel

| | |
|--------------------|--|
| Syntax | :AFANalyser:SPECTrum:RLEVel <real1> |
| Parameters | real1 is a floating point real number. The minimum value for real1 is 0.001 , the maximum value is 30.0 . The minimum resolution for real1 is 0.001 . The default value for real1 is 1.0 . |
| Description | This command sets the level of the reference voltage (i.e. the 0 dB line of the AF spectrum display). The physical unit of real1 is volt if the signal source is AF in . Notes <ul style="list-style-type: none"> - When the signal source is RF in, the 0 dB line is interpreted as full scale. - A typical audio standard is 0.775 volts for 0 dB. |
| Query | The query form of this command will return the current setting. The string delivered back will contain one floating point real number. |
| Example | :AFANalyser:SPECTrum:RLEVel 13.0 :AFAN:SPEC:RLEV? Value returned: " 13.0 ". |

MS Power Supply subsystem

The PSUPply subsystem controls the accessible parameters of the MS Power Supply Option.

:PSUPply:STATe

| | |
|--------------------|--|
| Syntax | :PSUPply:STATe <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: ON OFF . Default is OFF . |
| Description | Switches the output of the power supply option of the Willtek 4400 either on or off. Please note that this command needs the power supply option to be installed on your Willtek 4400. Details regarding this option can be found in section " Accessories and options " on page 29. |
| Query | The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above. |
| Example | :PSUPply:STATe ON :PSUP:STAT? Value returned: " ON ". |

:PSUPply:LEVel

| | |
|--------------------|--|
| Syntax | :PSUPply:LEVel <real1> |
| Parameters | real1 is a floating point real number. The minimum value for real1 is 0.0 , the maximum value is 10.0 . The minimum resolution for real1 is 0.05 . The default value for real1 is 0.0 . |
| Description | This command sets the output voltage of the optional power supply. The value specified for real1 is the output voltage in volt. Please note that this command needs the power supply option to be installed on your Willtek 4400. Details regarding this option can be found in section " Accessories and options " on page 29. |
| Query | The query form of this command returns the current setting. The query will return a string, containing one floating point real number. |
| Example | :PSUPply:LEVel 3.65 :PSUP:LEV? Value returned: " 3.65 ". |

:PSUPply:MEASure:CMAX

| | |
|-------------------|--|
| Syntax | :PSUPply:MEASure:CMAX <PredefExp> |
| Parameters | PredefExp is one of the following predefined expressions: A4 A2 A1 MA400 . Default is A4 . |

| | |
|--------------------|--|
| Description | <p>Selects the range of the maximum current the mobile under test is expected to draw. For a maximum current of up to 400 mA, the MA400 setting can be used. If the mobile will not draw more than 400 mA, the MA400 setting is appropriate. In case the maximum power requirement of the mobile is unknown, setting A4 (the default setting) is recommended as it will allow the mobile to draw a maximum current of 4 A.</p> <p>Please note that this command needs both the power supply option and the current measurement option to be installed on your Willtek 4400. Details regarding these options can be found in section "Accessories and options" on page 29.</p> |
| Query | <p>The query form of this command returns the current setting. A query will return a string, containing one of the predefined expressions explained above.</p> |
| Example | <pre>:PSUPply:MEASure:CMAX MA400 :PSUP:MEAS:CMAX? Value returned: "MA400".</pre> |

Warranty and Repair

A green square containing a white capital letter 'B'.

This chapter describes the customer services available through Willtek. Topics discussed in this chapter include the following:

- ["Warranty information" on page 352](#)
- ["Equipment return instructions" on page 353](#)

Warranty information

Willtek warrants that all of its products conform to Willtek's published specifications and are free from defects in materials and workmanship for a period of one year from the date of delivery to the original buyer, when used under normal operating conditions and within the service conditions for which they were designed. This warranty is not transferable and does not apply to used or demonstration products.

In case of a warranty claim, Willtek's obligation shall be limited to repairing, or at its option, replacing without charge, any assembly or component (except batteries) which in Willtek's sole opinion proves to be defective within the scope of the warranty. In the event Willtek is not able to modify, repair or replace nonconforming defective parts or components to a condition as warranted within a reasonable time after receipt thereof, the buyer shall receive credit in the amount of the original invoiced price of the product.

It is the buyer's responsibility to notify Willtek in writing of the defect or nonconformity within the warranty period and to return the affected product to Willtek's factory, designated service provider, or authorized service center within thirty (30) days after discovery of such defect or nonconformity. The buyer shall prepay shipping charges and insurance for products returned to Willtek or its designated service provider for warranty service. Willtek or its designated service provider shall pay costs for return of products to the buyer.

Willtek's obligation and the customer's sole remedy under this hardware warranty is limited to the repair or replacement, at Willtek's option, of the defective product. Willtek shall have no obligation to remedy any such defect if it can be shown: (a) that the product was altered, repaired, or reworked by any party other than Willtek without Willtek's written consent; (b) that such defects were the result of customer's improper storage, mishandling, abuse, or misuse of the product; (c) that such defects were the result of customer's use of the product in conjunction with equipment electronically or mechanically incompatible or of an inferior quality; or (d) that the defect was the result of damage by fire, explosion, power failure, or any act of nature.

The warranty described above is the buyer's sole and exclusive remedy and no other warranty, whether written or oral, expressed or implied by statute or course of dealing shall apply. Willtek specifically disclaims the implied warranties of merchantability and fitness for a particular purpose. No statement, representation, agreement, or understanding, oral or written, made by an agent, distributor, or employee of Willtek, which is not contained in the foregoing warranty will be binding upon Willtek, unless made in writing and executed by an authorized representative of Willtek. Under no circumstances shall Willtek be liable for any direct, indirect, special, incidental, or consequential damages, expenses, or losses, including loss of profits, based on contract, tort, or any other legal theory.

Equipment return instructions

Please contact your local service center for Willtek products via telephone or web site for return or reference authorization to accompany your equipment. For each piece of equipment returned for repair, attach a tag that includes the following information:

- Owner's name, address, and telephone number.
- Serial number, product type, and model.
- Warranty status. (If you are unsure of the warranty status of your instrument, include a copy of the invoice or delivery note.)
- Detailed description of the problem or service requested.
- Name and telephone number of the person to contact regarding questions about the repair.
- Return authorization (RA) number or reference number.

If possible, return the equipment using the original shipping container and material. Additional Willtek shipping containers are available from Willtek on request. If the original container is not available, the unit should be carefully packed so that it will not be damaged in transit. Willtek is not liable for any damage that may occur during shipping. The customer should clearly mark the Willtek-issued RA or reference number on the outside of the package and ship it prepaid and insured to Willtek.

Publication History

| Revision | Comment |
|------------|---------------|
| 0508-610-A | First version |

Willtek and its logo are trademarks of Willtek Communications GmbH. All other trademarks and registered trademarks are the property of their respective owners.

Specifications, terms and conditions are subject to change without notice.

© Copyright 2005 Willtek Communications GmbH. All rights reserved.

No part of this manual may be reproduced or transmitted in any form or by any means (printing, photocopying or any other method) without the express written permission of Willtek Communications GmbH.

Willtek Worldwide Offices

West Europe/Middle East/Africa

Willtek Communications GmbH
Gutenbergstr. 2-4
85737 Ismaning
Germany

info@willtek.com

Willtek Communications SARL
Aéroport – Bâtiment Aéronef
Rue de Copenhague – BP 9001
95728 Roissy CDG Cédex
France

willtek.fr@willtek.com

Willtek Communications Ltd.
Landmark House, Station Road
Cheadle Hulme
Cheshire SK8 7BS
United Kingdom

willtek.uk@willtek.com

North America/Latin America

Willtek Communications Inc.
7369 Shadeland Station Way, Suite 200
Indianapolis, IN 46256
USA

willtek.us@willtek.com

Asia Pacific

Willtek Communications
22, Malacca Street
#09-00, Royal Brothers Building
Raffles Place
Singapore 048980

willtek.ap@willtek.com

Willtek Communications (China) Ltd.
Room 1403, 14/F Cimic Tower,
1090 Century Avenue, Pudon,
Shanghai, 200120
China

willtek.cn@willtek.com