Handbuch • Manual

Geschäftsbereich • Division
Überwachungs- und Ortungstechnik • Radiomonitoring and Radiolocation

HF DIGITAL WIDEBAND RECEIVER

R&S®EM510

4065.7728.02

ROHDE&SCHWARZ

4065.7763.34-01.00

Printed in the Federal
Republic of Germany

Rohde & Schwarz GmbH & Co. KG
Mühldorfstraße 15
D-81671 München
www.rohde-schwarz.com

4065.7763.34-01.00

Dear Customer,

EM510 is the abbreviation used throughout the manual for R&S EM510.

R&S® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.

AMMOS® is a registered trademark of Rohde & Schwarz GmbH & Co. KG.

References

[1] Serial Front Panel Data Port Specification VITA 17-199x Draft 0.5

Contents

Further Appendices

2 Characteristics

2.1 Use

The EM510 is designed to cover the frequency range of 9 kHz to 32 MHz. It is made up of a compact case, power supply modules and the HF receiver module.

The EM510 is controlled via the LAN interface (TCP/IP) based on the SCPI (Standard Commands for Programmable Instruments) syntax.

It operates in the following modes:

- Fixed Frequency Mode (FFM)
- Memory Scan
- Frequency Scan
- Panorama Scan (option)
- Test

Data output is possible in the following formats:

- Baseband signal (I and Q) in digital form at
  – LAN (B_{max} = 1 MHz)
  – FPDP (B_{max} = 10 MHz)
- IF analog GC, variable center frequency 0 to 21.4 MHz, 2 channels or Video analog, DC to BW/2, 2 channels (AM/FM or I/Q)
- Video digital via LAN 2 channels
  AM/FM (BW_{max} = 250 kHz) or I/Q (BW_{max} = 500 kHz)
- AES3 for recording digital AF data streams
- AF digital via LAN
- AF analog (600 Ω line and headphones)

The EM510 is designed for bench top use. It is also well-suited for mounting in a rack environment.

2.2 Operating Modes and Control

The **Fixed Frequency Mode** is the standard mode of the receiver. In this mode a fixed frequency channel is set at which the signal is received, filtered and demodulated.

The following parameters can be set:

- Frequency
  Setting range from 9 kHz to 32 MHz in 1 Hz steps
- Demodulation mode
  The following demodulation modes can be selected:
  – FM
  – AM
  – PULSE (AM PULSE)
  – PM
  – USB
  – LSB
  – ISB
  – CW
  – I/Q

- Bandwidth
  The IF bandwidths can be selected in 30 steps between 100 Hz and 10 MHz.
- Measurement time (automatic or settable between 0.5 ms to 900 s)
- Detector modes "continues" or "periodic"
- Squelch
  The level squelch can be set in 1 dB steps in the range from -30 to +130 dBµV.
- AFC (Automatic Frequency Control)
  With AFC switched on, the receiver frequency is tuned within the IF bandwidth.
- Level detector
  For level measurement the detector can be switched to AVG, PEAK, RMS or FAST.
- Frequency offset detector
- Attenuator
  The attenuator operates in manual (0 to 25 dB) or automatic mode.
- Gain control (GC)
  Automatic (AGC) and manual (MGC) gain control is selectable.
  The MGC setting covers the input signal range from -30 to 130 dBµV.
- Video Panorama
  Spectrum of the demodulated signal with analysis features (squared AM, FM)
- IF Panorama (Option)
  Independent from IFBW, settable from 10 kHz to 9.6 MHz
- ITU Measurement (Option)
  AM modulation index (AM+, AM-, AM)
  FM deviation (FM+, FM-, FM)
  PM deviation (0 to 4π)
  Bandwidth (0 to 9.6 MHz)

In the **Memory Scan** mode, the receiver settings can be programmed for the monitoring of up to 10000 channels. These channels can be scanned with the "Memory Scan" command. A single channel can be called with the "Recall" command.

The squelch level serves as a criterion for dwelling at the same frequency or switching to the next channel. If the level criterion is met, the receiver allows for the selectable dwell time to elapse and then switches to the next channel.

Parameters selectable for each channel:

- Memory location status
- Frequency
- Demodulation mode
- Bandwidth
- Attenuator
- AFC settings
- Squelch parameters
- Antenna number

The "Continue" command can be used to switch to the next channel before the dwell time has elapsed.

In the **Frequency Scan** mode "Start Frequency", "Stop Frequency", and "Frequency Step" are defined for monitoring a specific frequency range. This frequency range can be scanned with the "Frequency Scan" command.

The squelch level serves as a criterion for dwelling at the same frequency or switching to the next frequency. If the level criterion is met, the EM510 allows for the selectable dwell time to elapse and then switches to the next frequency. The demodulator settings are fixed for the defined search range. The "Continue" command can be used in this case to switch to the next frequency before the dwell time has elapsed.

In **Memory Scan** or **Frequency Scan** mode a selectable number of different measurements (e.g. level offset, AM modulation index, FM deviation, bandwidth) are performed in parallel.

In the **Panorama Scan** mode the receiver is tuned from start to stop frequency in nearly 10 MHz steps, performing a high resolution FFT at each step. The resolution bandwidth covers a range from 125 Hz up to 100 kHz, resulting in an outstanding scan speed of up to 600000 Ch/s or 16 GHz/s.

In the "Fixed Frequency Mode" a comprehensive **selftest** of the receiver can be performed. The test can be carried out in full or in a shorter version with only "GO" or "NOGO" being issued.

2.3 Design

(see Fig. 2-1: Front View, Fig. 2-2: Rear View and Fig. 2-3: Modules)

The EM510 basically consists of the following modules:

- Compact case
- AC/DC converter and power supply
- HF receiver module
- Fans

The compact case corresponds to the 19” standard and can be fitted into any 19” rack. Two brackets are provided at the sides for rack mounting.

At the rear on the right the interface and RF connectors are located. Three LEDs located above the 10/100 Base-T Ethernet LAN connector indicate link status and activity. On the left there are the mains input socket with integral fuse holder and the DC source connector for feeding in the mains and DC supply voltage. Fuse F3 is for the DC supply voltage. The four feet of the compact case protect the connectors when the case is set down.

At the front the POWER switch, POWER LED and HEADPHONE socket are located.

The HF receiver module is located in the left-hand part of the chassis. It is arranged in such a way that the RF connections between the receiver and rear connectors are as short as possible. The interface connectors are directly accessible through an opening in the rear panel. The HF receiver module is designed as a slide-in module with its own front panel indicators and connectors.

The AC/DC converter and the power supply are located in the right-hand part of the compact case.

The two fans incorporated in the partition panel ensure sufficient cooling of the power supply and the HF receiver module. There is no temperature-dependent control for adjusting the fan speed.

The fans expel the hot air to the sides. Both side panels are equipped with ventilation slots for drawing and expelling cooling air. On installing the EM510, sufficient room should be left at the sides of the unit to ensure an unhindered flow of hot air.

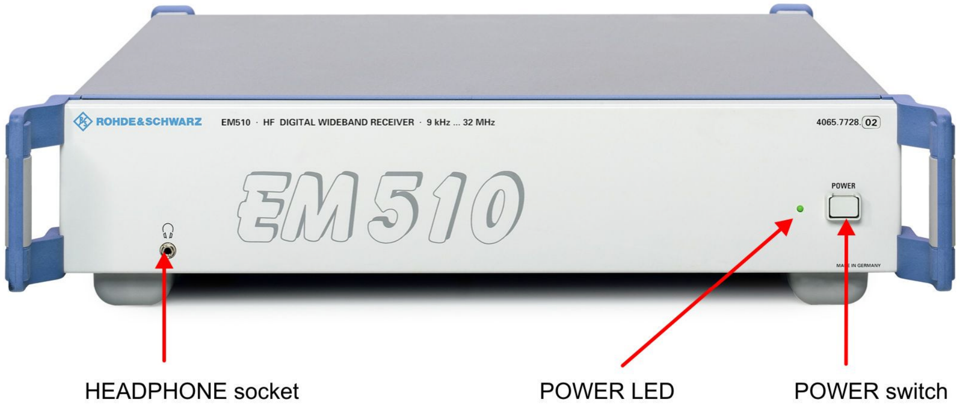


Fig. 2-1: Front View

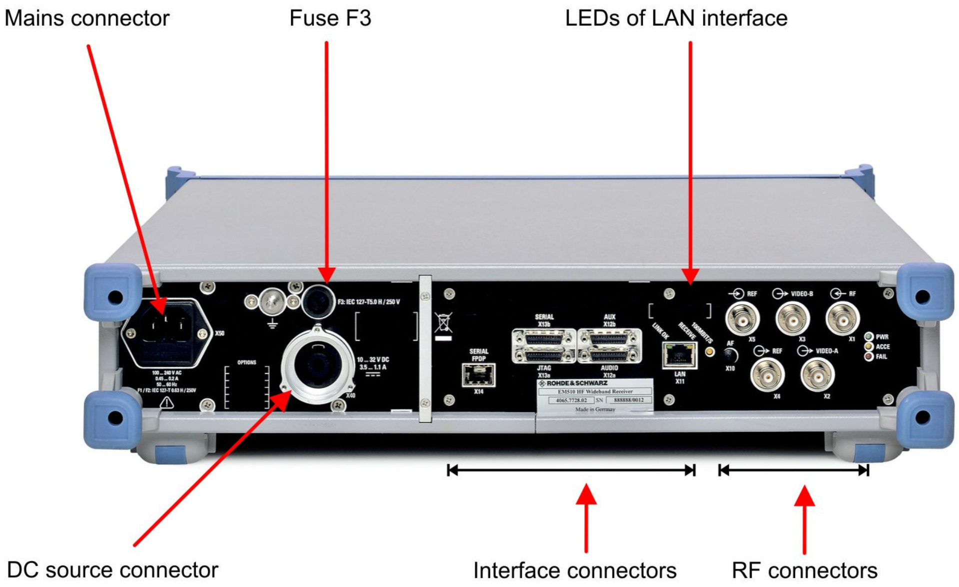


Fig. 2-2: Rear View

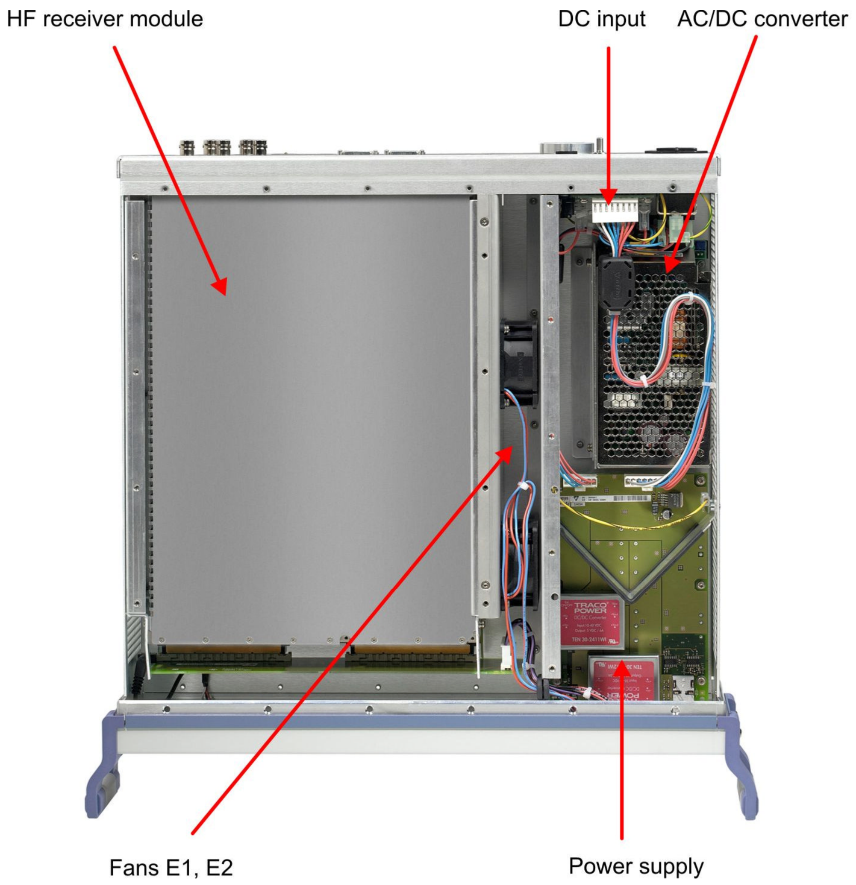


Fig. 2-3: Modules

2.4 Detailed Description

(see Fig. 2-4: EM510 Detailed Block Diagram)

This section deals with the functions of the major functional blocks.

2.4.1 Power Supply Modules

The EM510 operates either on AC or on DC.

- DC voltage: 12 to 32 VDC

- AC voltage: 100 to 240 VAC (115, 230 VAC nominal) with a frequency of 50 to 60 Hz

If both voltage sources are connected, AC supply operation is favored. If the AC supply fails, switch-over to battery operation is effected automatically.

The mains connector is provided with two mains fuses (F1, F2) and an EMC filter. The mains fuses protect the electronic circuits against overload and short circuits. The EMC filter prevents radiated interference from leaving the equipment and also prevents picked-up and conducted interference from entering the equipment.

The external AC voltage is fed from the mains connector to the AC/DC converter. This converts the AC voltage into a DC voltage of 24 V, which is fed to the power supply.

The 12 to 32-VDC voltage is fed directly from the DC input to the power supply.

If the POWER switch on the front panel is on, a switch-over circuit in the power supply connects one of the DC voltages.

The POWER LED indicates whether or not power is being supplied to the power supply.

The power supply contains five DC/DC converters which provide the HF receiver module and the fans with the required operating voltages. The following stabilized voltages are generated:

For the HF receiver module

- +5.15 ± 0.15 VDC / 5.5 A

For the fans

- +8 VDC / 0.5 A

All outputs are protected against short circuit and overvoltage. Green LEDs on the power supply indicate the presence of the DC input and supply voltages.

2.4.2 HF Receiver Module

The HF receiver module is controlled via the LAN interface X11.

The necessary signal inputs and outputs of the HF receiver module are routed via RF cables to the connectors on the rear and front panel or are available through the opening in the rear panel.

The RF antenna signal is fed in at connector X1 RF and then taken via RF cable to the RF input of the HF receiver module. In the digital receiver the received signals are passed through the filters in the preselection on to the A/D converter, and are subsequently digitally filtered and demodulated to produce the baseband, video and AF signals. These signals can be transmitted in digital form via the LAN interface. Together with further output signals for detailed signal analysis and monitoring they are also made available via the following connectors:

Rear panel:

- X6 **VIDEO A**: Analog video output A for AM and I components
- X7 **VIDEO B**: Analog video output B for FM and Q components
- X12a **AUDIO:** Audio output analog and digital
- X14 **FPDP**: Wideband digital interface. FPDP (Front Panel Data Port) is a platform-independent 32-bit synchronous data flow path that allows data to be transferred at high speeds (up to 1 Gbit/s), e.g. AF or baseband signal (I and Q) in digital form.

Front panel:

- X10 **AF**: Phone jack (stereo socket AF_L; AF_R), AF analog headphone output (600 Ω line and headphones)

For the purposes of synchronization the following inputs and outputs are available on the rear panel:

- X8 **REF OUT**: Output for internal reference signal, f = 10 MHz
- X9 **REF IN**: Input for external reference signal, f = 10 MHz

The HF receiver module can use an internal frequency reference or can be set to synchronize to the external frequency at X9 REF IN to provide even better frequency accuracy and stability.

2.4.3 Fans

The two 12-VDC fans for cooling the power supply and the HF receiver module are supplied with a reduced voltage of approximately 8 VDC to operate the fans below maximum speed. This increases the life of the fans and additionally reduces acoustic noise.

The rotation speed of the fans is constant and not controlled as a function of the temperature.

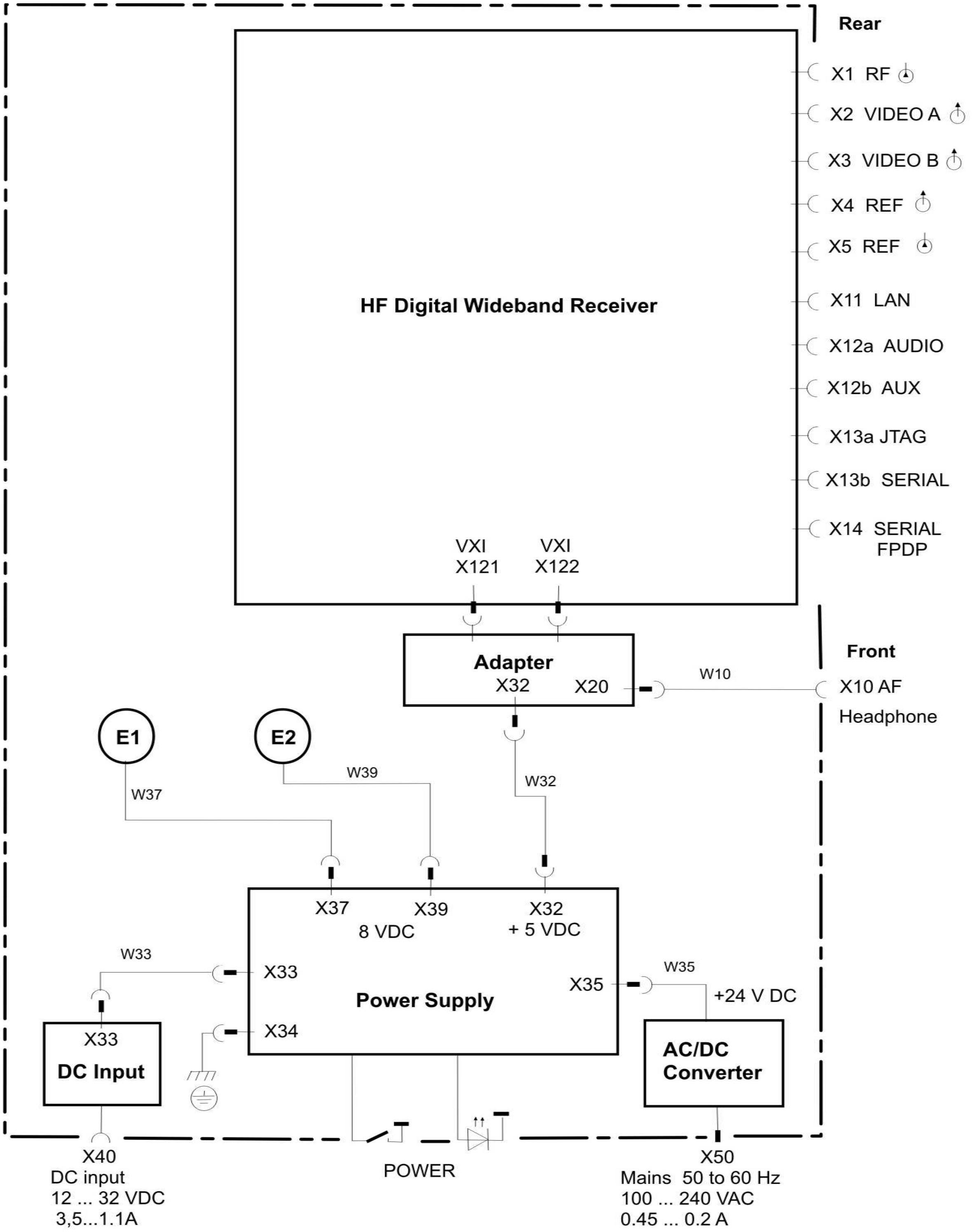


Fig. 2-4: EM510 Detailed Block Diagram

2.5 Accessories

Standard delivery of the EM510 comprises

- 1 mains connection cable
- 1 CD set including documentation and firmware & utilities

The delivery note is always included.

A screened battery cable is delivered by Rohde & Schwarz on request.

3 Installation and Cabling

The EM510 is supplied completely assembled except for the handles and mounting brackets (use in rack), which come separately and must be attached by the user.

3.1 Unpacking and Checking

Unpack the EM510.

⇒ Remove the packing material.
⇒ Remove the transport shells.
⇒ Check all delivered items according to the delivery note.
⇒ Screw on the side handles.
⇒ Check the EM510 for visible damage that may have been caused in transit.
⇒ Contact the transport agents immediately if damage is found.

Note: Keep the packing material for re-use!

3.2 Safety Precautions

Caution: *Switching the unit off by means of the POWER switch on the front panel does not separate the unit from mains power. The mains voltage of 100 to 240 VAC is highly dangerous. Therefore proceed with extreme care when handling such voltages.*

When fitting operating rooms and installing and operating electrical equipment, the relevant national and international safety provisions and regulations have to be adhered to.

The following safety instructions apply in particular:

⇒ IEC 364
⇒ VDE 0100
⇒ DIN 57100

These safety regulations deal with the following subjects:

⇒ Protection:
- Prevention of accidents
- Protection against overvoltage
- Insulation of equipment
- Grounding

⇒ Characteristics and laying of lines and cables
⇒ Provisions for operational facilities and rooms and systems of a special nature.

3.3 Bench Operation

⚠ ***Attention:*** *Do not expose the EM510 to humidity.*
Leave at least 50 mm of empty space along both side panels with ventilation slots in order to ensure proper cooling function.

There are no special requirements for bench operation.

To facilitate access to the front panel elements, we suggest that you raise the front of the EM510 by deploying the unit's feet.

3.4 Rack Mounting

⚠ ***Attention:*** *The EM510 should be used in an area where the ambient temperature does not exceed –10 to +55 °C.*
The EM510 is fan cooled and must be installed with sufficient space on the sides to permit a free flow of air. Make sure that hot air can escape freely. To ensure sufficient cooling, do not attach telescopic rails to the sides of the unit.

If the rack is exposed to high ambient temperatures, sufficient ventilation must be ensured for the rack.

⚠ ***Note:*** *Do not remove the enclosure when rack mounting the EM510.*

⇒ Place the EM510 onto the guide rails of the rack and insert it into the rack. Use the four screws of the front mounting bracket to fasten the unit to the rack.

3.5 Cabling

Cabling always depends on the particular application involved.

Connect the EM510 observing the following sections and instructions for use.

For further connections not mentioned in the following sections refer to the Interfaces Description.

3.5.1 Connection of Power Supply

The EM510 is suitable for AC mains operation as well as DC operation.

Connection of the EM510 both to the mains and the DC source permits AC/DC switch-over, i.e. if there is a power failure the DC source (battery) will take over automatically.

3.5.1.1 Mains Connection

Attention:
Make sure that the available mains voltage is between 110 and 240 VAC.

The EM510 is connected to the mains voltage by the plug with mains filter on the rear panel.

The power cable is supplied as an accessory.

3.5.1.2 DC Source Connection

Attention:
Make sure that the available supply voltage is between 12 and 32 V.
Observe correct voltage polarity when connecting.
Incorrect polarity may blow the rear panel fuse or damage the EM510.

Note: If MIL-Spec regarding EMC must be fulfilled, a screened battery cable not exceeding 3 m in length is necessary. It is delivered by Rohde & Schwarz on request.

The EM510 is connected to an external 12 to 32-VDC source (e.g. battery) by the connector X40 12 TO 32 VDC on the rear panel.

Recommended connector: Neutrik® Speakon® NL4FX

Installing the connector:

1. Insert the Speakon® NL4FX connector into the socket X40 on the rear panel.
2. Turn the connector clockwise until it is locked into place and is secured by the safety-latch.
3. Screw on the fitting for the hose (Fig. 3-2).

Removing the connector:

1. Unscrew the fitting for the hose (Fig. 3-2).
2. Press and chuck back the safety-latch of the Speakon® NL4FX connector (Fig. 3-1).
3. Turn the connector counter-clockwise and withdraw it.

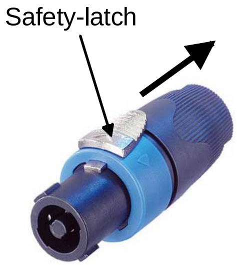


Fig. 3-1: Connector: Neutrik® Speakon® NL4FX

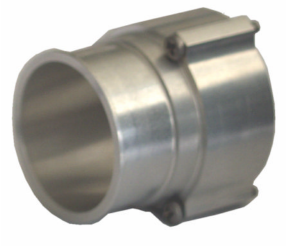


Fig. 3-2: Hose Fitting

3.5.2 Antenna Connection

The antenna is connected to socket X1 on the rear panel.

- Frequency range: 9 kHz to 32 MHz

- RF level: -137 to +10 dBm, P_{max} = 50 mW (+15 dBm)

- Z = 50 Ω

3.5.3 Connection of External Reference (optional)

The EM510 can be set to synchronize to an external frequency. To do so, apply the reference signal for the synchronization to socket X9 REF IN on the rear panel.

- Frequency range: 1 to 20 MHz

- RF level: 0 to +10 dBm

- Z = 50 Ω

Note: If MIL-Spec regarding EMC must be fulfilled, a double screened connecting cable is necessary.

3.5.4 Connection of Headphones (optional)

Connect the headphones to HEADPHONE socket X10 AF on the front panel.

- V_{pp} max = 5 V

- R_i = 100 Ω

3.5.5 Connection of PC or LAN for Remote Control

⇒ Use a LAN crossed-over cable (point-to-point connection) with RJ45 connector to connect the Ethernet port of the PC directly to the socket X11 LAN (ETHERNET 10/100 BASE-T, RJ45 8-pin connector) on the rear panel of the EM510.

or

⇒ Use a LAN patch cable with RJ45 connector to connect the hub or network to the socket X11 LAN on the rear panel of the EM510.

4 Preparation for Use

4.1 Power Up

⚠ ***Attention:***
Before switch-on:
- Check that the available mains voltage is between 100 and 240 VAC.
- Check that the available supply voltage is between 12 and 32 VDC.

The EM510 is switched on via the POWER switch.

The POWER LED lights up after switch-on.

4.2 Local Control

Except for the POWER LED and POWER switch the EM510 has no control elements.

4.3 Functions of Controls and Indicators

Control and Indicator	Function
Switch **POWER**	This switch is used to switch on the EM510.
Green LED **POWER**	The LED is illuminated when the mains voltage is applied and the POWER switch is set to ON.
Yellow LED **100MBIT/S** (rear panel, Fig. 4-1)	The LED indicates operation in 100 Mbit/s mode.
Green LED **LINK OK** (rear panel, Fig. 4-1)	The LED will be on if the physical connection to the network is intact.
Yellow LED **RECEIVE** (rear panel, Fig. 4-1)	The LED will flash yellow if the EM510 receives data from the network. The frequency of flashing is directly related to the network activity.

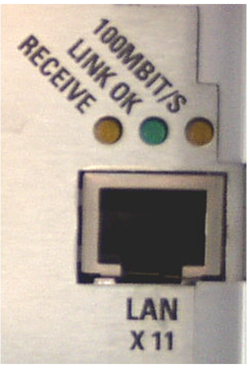


Fig. 4-1: LEDs of Ethernet LAN Interface

4.4 Firmware Update

The entire firmware (software for the PPC and DSP processors and for the FPGA) of the EM510 can be renewed by a firmware update. The CPLD on the mainboard is only programmed in production by JTAG.

The EM510 firmware update can be done via the RS232 interface or via the LAN interface.

The firmware is updated via the serial interface (X13b on the EM510 front panel) or the LAN interface (X11 on the EM510 front panel).

An update via the LAN interface requires the update tool UPD32.EXE, which runs under WinNT, Win2000 and WinXP.

An update via RS232 can be done using the update tool UPDATE.EXE, which runs under DOS, Win95 and Win98, or with the update tool UPD32.EXE, which runs under WinNT, Win2000 and WinXP.

4.4.1 System Requirements

4.4.1.1 System Requirements for Update via RS232

- IBM-compatible PC with RS232 interface (COM1 or COM2)
- Serial null-modem cable (RxD, TxD crossed), 9-pin female to 9-pin female
- Serial Adapter X13B to 9-pin male

4.4.1.2 System Requirements for Update via LAN

- IBM-compatible PC with WinNT, Win2000 or WinXP and LAN interface
- LAN crossed cable with RJ45 connector, for point-to-point connection

  or
- LAN patch cable with RJ45 connector, direct for connection via HUB or network

4.4.2 Preparations before Update

4.4.2.1 Preparations before Update via RS232

- Plug the serial adapter into X13B of the EM510.
- Use the null-modem cable to connect the COM port of your PC to the adapter of the EM510

4.4.2.2 Preparations before Update via LAN

- Use the crossed LAN cable to connect the Ethernet port of the PC directly to the X11 of the EM510

  or
- Use the patch LAN cable to connect the hub or network to the X11 of the EM510

4.4.3 Update Procedure

4.4.3.1 Prepare Update Data

- Create a new directory on your PC by command:
  `md EM510\V0160`
- Copy the self-extracting archive file to this directory.
- Unpack the self-extracting archive file in the directory by command:

  `EM510_V0160.exe`
- After unpacking you will find the following files in your directory:

`UPDATE.EXE`	update program for DOS, Win95, Win98
`UPDATE.HLP`	help file for DOS update program
`UPD32.EXE`	update program for WinNT, Win2000, WinXP
`UPD32.HLP`	help file for update program for WinNT, Win2000, WinXP
`EM510P1.cfg`	configuration file
`BOOTLOAD.ELF`	boot loader
`EM510P1.elf`	firmware update code

4.4.3.2 Installation of WinPcap ©

If you want to update your target via Ethernet you have to install the necessary WinPcap components. If WinPcap is already installed, you only have to reinstall WinPcap if your version of WinPcap is older than 3.0.

As of version 3.31, UPD32.EXE uses WinPcap and its API for its Ethernet capabilities. WinPcap is a free, public system for direct network access under Windows.

The version of WinPcap delivered with EM510 Update is not necessarily the most recent version. To obtain the most recent version of WinPcap as well as further information about the software kit, visit http://winpcap.polito.it/ .

Version 3.41 of UPD32.EXE has been developed and tested using WinPcap 3.1.

Install WinPcap simply by double-clicking the supplied WinPcap installation file. Read the installation instructions carefully.

Please read the following disclaimer very carefully and also see the About Box:

Important notes about WinPcap:

Neither the name of the "Politecnico di Torino" nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes software developed by the University of California, Lawrence Berkeley Laboratory and its contributors.

4.4.3.3 Update via RS232

- Switch off the EM510 before running the update program.
- Start update program UPDATE.EXE under DOS and Win3.1, Win95, Win98
  - Select configuration file EM510P1.CFG through:
    File - Open Config File
  - Configure the connected COM port through:
    File - COM port
  - Start update with:
    Actions - Update device
- Start update program UPD32.exe under WinNT, Win2000, WinXP
  - Select configuration file EM510P1.CFG through:
    File - Config File
  - Configure the connected COM port through:
    Config - COM Port
  - Start update with:
    Update - via COM
- Start update process
  - Switch on the EM510 within 30 seconds after the update has been started on the PC.

4.4.3.4 Update via Ethernet (LAN)

In this mode you can update one or more targets using the same update software.

- Switch off the target EM510 unit(s) before running the update program.
- Start update program UPD32.exe
- Select configuration file EM510P1.CFG through menu “File - Config File”
- If your computer contains more than one network adapter, UPD32 will take the first as the default adapter to be used for the update. To change the network adapter, select ”Network Adapter” in the ”Config” menu.
- Then select ”via Ethernet” within the ”Update” menu.
  Once the menu item has been selected, the system will start sending broadcast telegrams.
- Switch on or reset the target EM510 unit(s)
  The targets will receive the broadcast telegrams and respond with a hardware identification telegram. If this ID corresponds to the selected configuration file, the target will be listed in the next line.
- Now you can select the desired targets from the list.
- Start the update procedure with the “START” button. The target units will now be updated one by one.

4.4.4 Messages and Indication

4.4.4.1 Update Messages

UPD32 displays the following messages during the update procedure:

```
STARTING COMMUNICATION FOR TARGET (ADR:00 90 B8 10 01 14)
Load boot code....
Erasing...
Loading program code...
Calculating checksum...
UPDATE COMPLETE
```

4.4.4.2 Indication on the EM510 front panel

The lighted ACCE LED on the front panel of the EM510 shows that the update is in progress.

After the update the ACCE LED is switched off and the FAIL LED is switched on.

If no problem is detected, the EM510 application starts automatically and switches off the FAIL LED.

If a problem appears, the FAIL LED toggles the error codes.

Error Code	Definition	Note
0	ERROR_NO_ERROR	No error
1	ERROR_CPLD_NOT_PROGRAMMED	Internal
2	ERROR_EEPROM_NO_IDENT_BLOCK	Internal
3	ERROR_EEPROM_NO_SERIAL_NUMBER	Internal
4	ERROR_EEPROM_NO_ETH_ADDR	Internal
5	ERROR_WRONG_LOCATED	Internal
6	ERROR_NO_APPLICATION	There is no application in the Flash EEprom

5 Remote Control via LAN with SCPI Command Syntax

5.1 Introduction

The EM510 can be remote controlled via LAN by means of SCPI command syntax. The control commands and the status reporting system are described in this chapter. The socket RJ45 is on the back side of the EM510.

Within the TCP/IP protocols, the EM510 supports the SCPI command syntax version 1993.0 (Standard Commands for Programmable Devices). The SCPI standard is based on IEEE 488.2 and is aimed at standardizing device-specific commands, error handling and status registers (see "Notation" on page 5.13).

This section assumes a basic knowledge of programming and controller operation.

The requirements of the SCPI standard for command syntax, error handling and configuration of the status registers are explained in the relevant sections. Tables and figures provide a fast overview of the commands implemented in the device and the bit assignment in the status registers. The tables are supplemented by a comprehensive description of every command and the status registers. Detailed programming examples of the main functions are to be found in Annex E: LAN Programming Examples.

5.1.1 Remote Control via LAN interface

The default values of the interface parameters of the EM510 are configured

with the host name or IP-address 89.10.11.23 and port number 5555.

For more information see the Interfaces Description in the annex.

1. Connect the unit with controller via Ethernet cable by RJ45 plugs. In case of a direct connection to a computer network card a crossed cable must be used. If the EM510 is connected via hub or directly to a network, a 1-to-1 cable is required.
2. TCP/IP must be installed on the controller. The network card can be set to half duplex or full duplex. The EM510 determines the respective configuration during the switch-on and reacts correspondingly.
3. If the EM510 is operated in a network, it must be set to a network-compatible IP address. Consult your network administrator for further information. See also Annex D: LAN Configuration.
4. The ping command is a simple way to check whether the controller is able to connect to the EM510. Just enter the command “ping <IP address>“ (e.g..: “ping 89.10.11.23“) “ in the DOS box.
5. Commands can be sent to and messages received from the EM510 by means of a Telnet application which is started and configured with the interface parameters of the EM510.
6. To test the connection, enter *idn?, for example, to query the EM510's identification. The response string should then be displayed.

Note:
If the same IP address already exists for a different device, the corresponding entry must be erased from the ARP table before setting up a new connection. This can be done in the DOS box by entering the command "ARP -d <IP address>".

5.1.2 Setting the IP Address and Port Number

The configuration of the LAN interface is described in Annex D: LAN Configuration.

IP address and port number settings will take effect immediately.

The EM510 is capable of assuming an IP address dynamically assigned by RARP (see Annex D: LAN Configuration).

5.2 Structure and Syntax of the Device Messages

5.2.1 SCPI Introduction

SCPI (Standard Commands for Programmable Devices) describes a standard command set for programming devices, irrespective of the type of device or manufacturer. The goal of the SCPI consortium is to standardize the device-specific commands to a large extent. For this purpose, a model was developed which defines the same functions inside a device or for different devices. Command systems were generated which are assigned to these functions. Thus it is possible to address the same functions with identical commands. The command systems are of a hierarchical structure. Figure 5-1 illustrates this tree structure using a section of command system SENSe, which operates the sensor functions of the devices. The other examples regarding syntax and structure of the commands are derived from this command system.

SCPI is based on standard IEEE 488.2, i.e. it uses the same syntactic basic elements as well as the common commands defined in this standard. Part of the syntax of the device responses is defined with greater restrictions than in standard IEEE 488.2 (see 5.2.4 "Responses to Queries").

5.2.2 Structure of a Command

The commands consist of a so-called header and, in most cases, one or more parameters. Header and parameters are separated by a "white space" (ASCII code 0 to 9, 11 to 32 decimal, e.g. blank). The headers may consist of several keywords. Queries are formed by directly appending a question mark to the header.

Note:
The commands used in the following examples are not in every case implemented in the device.

Common commands

Common commands consist of a header preceded by an asterisk "*" and one or several parameters, if any.

Examples:	`*RST`	RESET, resets the device
	`*ESE 253`	EVENT STATUS ENABLE, sets the bits of the event status enable register
	`*ESR?`	EVENT STATUS QUERY, queries the contents of the event status register.

Device-specific commands

Hierarchy: Device-specific commands are of hierarchical structure (see Figure 5-1). The different levels are represented by combined headers. Headers of the highest level (root level) have only one keyword. This keyword denotes a complete command system.

Example: `SENSe` This keyword denotes the command system `SENSe`.

For commands of lower levels, the complete path has to be specified, starting on the left with the highest level, the individual keywords being separated by a colon ":".

Example: `SENSe:FREQuency:STARt 118 MHz`

This command lies in the third level of the SENSe system. It sets the starting frequency of a scan to 118 MHz.

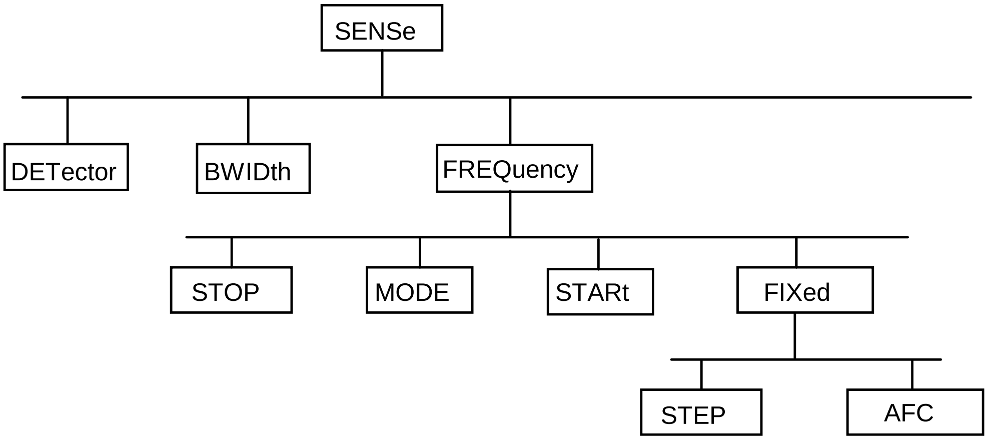


Figure 5-1: Tree structure of the SCPI command system using the SENSe system by way of an example

Some keywords occur at several levels within one command system. Their effect depends on the structure of the command, that is to say, at which position in the header of a command they are inserted.

Example: `OUTPut:OUTPut:SQUelch:STATe ON`

This command contains the keyword `STATe` at the third command level. It defines the state of the `SQUelch` function.

`OUTPut:FILTer:LPAS:STATe OFF`

This command contains the keyword STATe at the fourth command level. It defines the state of the AF filter.

Optional keywords: Some command systems permit certain keywords to be optionally inserted into the header or omitted. These keywords are marked by square brackets in the description. The full command length must be recognized by the device for reasons of compatibility with the SCPI standard. Some commands are considerably shortened by these optional keywords.

Example:

```
[SENSe]:FREQuency[:CW]: STEP [:INCRement]
25 kHz
```

This command sets the stepwidth for frequency UP-DOWN to 25 kHz. The following command has the same effect:

```
FREQuency:STEP 25 kHz
```

Note:
An optional keyword must not be omitted if its effect is specified in detail by a numeric suffix.

Long and short form: The keywords can be of a long form or a short form. Either the short form or the long form can be entered, other abbreviations are not permissible.

Example:		
	Long form:	STATus:QUEStionable:ENABle 1
	Short form:	STAT:QUES:ENAB 1

Note: *The short form is marked by upper-case letters, the long form corresponds to the complete word. Upper-case and lower-case notation only serve the above purpose, the device itself does not make any difference between upper- and lower-case letters.*

Parameter: The parameter must be separated from the header by a "white space". If several parameters are specified in a command, they are separated by a comma ",". A few queries permit the parameters MINimum, MAXimum and DEFault to be entered. For a description of the types of parameter, refer to 5.2.5"Parameters".

Example: `SENSe:FREQuency? MAXimum` Response: `3000000000`

This query requests the maximal value for the input frequency.

Numeric suffix: If a device features several functions or characteristics of the same kind, the desired function can be selected by a suffix added to the command. Entries without suffix are interpreted like entries with the suffix 1.

5.2.3 Structure of a Command Line

Several commands in a line are separated by a semicolon ";". If the next command belongs to a different command system, the semicolon is followed by a colon.

Example:

```
SENSe:FREQuency:STARt MINimum;:OUTPut:FILTer:LPAS:STATe ON
```

This command line contains two commands. The first command is part of the SENSe system and is used to specify the start frequency of a scan. The second command is part of the OUTPut system and sets the AF filter.

If the successive commands belong to the same system, having one or several levels in common, the command line can be abbreviated. To this end, the second command after the semicolon starts with the level that lies below the common levels (see also Figure 5-1). The colon following the semicolon must be omitted in this case.

Example:

```
SENSe:FREQuency:MODE CW;:SENSe:FREQuency:FIXed:AFC ON
```

This command line is represented in its full length and contains two commands separated from each other by the semicolon. Both commands are part of the SOURce command system, subsystem FREQuency, i.e. they have two common levels.

When abbreviating the command line, the second command begins with the level below SENSe:FREQuency. The colon after the semicolon is omitted.

The abbreviated form of the command line reads as follows:

```
SENSe:FREQuency:MODE CW;FIXed:AFC ON
```

However, a new command line always begins with the complete path.

Example: SENSe:FREQuency:MODE CW

SENSe:FREQuency:FIXed:AFC ON

5.2.4 Responses to Queries

A query is defined for each setting command unless explicitly specified otherwise. It is formed by adding a question mark to the associated setting command. According to SCPI, the responses to queries are partly subject to stricter rules than in standard IEEE 488.2.

1. The requested parameter is transmitted without header.

   Example: `SENSe:FREQuency:MODe` Response: `SWE`

2. Maximum values, minimum values and all further quantities which are requested via a special text parameter are returned as numerical values.

   Example: `FREQuency? MAX` Response: `3000000000`

3. Numerical values are output without a unit. Physical quantities are referred to the basic units.

   Example: `FREQuency?` Response: `100000000` for 100 MHz

4. Truth values <Boolean values> are returned as 0 (for OFF) and 1 (for ON).

   Example: `OUTPut:FILTer:STATe?` Response: `1`

5. Text (character data) is returned in a short form (see also 5.2.5"Parameters").

   Example: `SENSe:FREQuency:MODe?` Response: `SWE`

5.2.5 Parameters

Most commands require a parameter to be specified. The parameters must be separated from the header by a "white space". Permissible parameters are numerical values, Boolean parameters, text, character strings, block data and expressions. The type of parameter required for the respective command and the permissible range of values are specified in the command description (see 5.3"Description of Commands").

Numerical values

Numerical values can be entered in any form, i.e. with sign, decimal point and exponent. Values exceeding the resolution of the device are rounded up or down. The mantissa may comprise up to 41 characters, the exponent must lie inside the value range -999 to 999. The exponent is introduced by an "E" or "e". Entry of the exponent alone is not permissible. In the case of physical quantities, the unit can be entered. Permissible units are as follows:

for frequencies	GHz, MHz or MAHz, kHz and Hz, default unit is Hz
for times	s, ms, µs, ns; default unit is s
for levels	dBuV; default unit is dBuV
for percentage	PCT, default unit PCT

If the unit is missing, the default unit is used.

Example:

```
SENSe:FREQuency 123 MHz = SENSe:FREQuency 123E6
```

Special numerical

The texts MINimum, MAXimum, UP, DOWN and INFinity are interpreted as special numerical values.

In the case of a query, the numerical value is provided.

Example: Setting command: `SENSe:GCONtrol MAXimum`

Query: `SENSe:GCONtrol` Response: `100`

MIN/MAX

MINimum and MAXimum denote the minimum and maximum value.

UP/DOWN

UP, DOWN increases or reduces the numerical value by one step. The step width can be specified for some parameter which can be set via UP, DOWN via an allocated step command (see page 5.1). Some parameters can only be changed in fixed steps (e.g. `SENSe:BWIDth UP`).

INF `INFinity` represent + ∞. In case of queries the numeric value 9.9E37 is output. The INF value 9.9E37 is entered into the result buffers `MTRACE` and `ITRACE` for `MSCAN, FSCAN` or `PSCAN` to identify the range limit.

NINF `Negative INFinity (NINF)`represent -∞.. In case of queries the numeric value -9.9E37 is output. This value is output when a measured value is queried and measurement is not possible because of the unit settings.

NAN Not A Number (NAN) represents the value 9.91E37. NAN is only sent as device response. This value is not defined. Possible causes are the division of zero by zero, the subtraction of infinite from infinite and the representation of missing values (e.g. at TRACe[:DATA]?).

Boolean Parameters Boolean parameters represent two states. The ON state (logically true) is represented by `ON` or a numerical value unequal to 0. The OFF state (logically untrue) is represented by `OFF` or the numerical value 0. 0 or 1 is provided in a query.

Example: Setting command:

```
OUTPut:FILTer:STATe ON
```

Query: `OUTput:FILTER:STATe?`

Response: `1`

Text Text parameters (character data) observe the syntactic rules for keywords, i.e. they can be entered using the short or long form. Like any parameter, they have to be separated from the header by a "white space". In the case of a query, the short form of the text is provided.

Example: Setting command: `SENSe:FREQuency:MODE FIXed`

Query: `SENSe:FREQuency:MODE?` Response `FIX`

Strings Strings must always be entered in quotation marks (' or ").

Example: `SYSTem:SECurity: OPTion "123ABC"` or

`SYSTem:LANGuage 'English'`

Block data

Block data (Definite Length Block) are a transmission format which is suitable for the transmission of large amounts of data. A command using a block data parameter has the following structure:

Example: `HEADer:HEADer #45168xxxxxxxx`

ASCII character # introduces the data block. The next number indicates how many of the following digits describe the length of the data block. In the example the 4 following digits indicate the length to be 5168 bytes. The data bytes follow. During the transmission of these data bytes all End or other control signs are ignored until all bytes are transmitted. Data elements comprising more than one byte are transmitted with the byte being the first which was specified by SCPI command "FORMat:BORDer".

Expressions

Expressions must always be in brackets. The device requires this data format for the indication of channel lists. A channel list always starts with @ followed by a path name or channel numbers or ranges of channel numbers.

Example: `ROUTe:CLOSe (@23)`

5.2.6 Overview of Syntax Elements

The following survey offers an overview of the syntax elements.

:	The colon separates the key words of a command. In a command line the colon after the separating semicolon marks the uppermost command level.
;	The semicolon separates two commands of a command line. It does not alter the path.
,	The comma separates several parameters of a command.
?	The question mark forms a query.
*	The asterisk marks a common command.
"	Quotation marks introduce a string and terminate it.
#	ASCII character # introduces block data.
	A "white space" (ASCII-Code 0 to 9, 11 to 32 decimal, e g blank) separates header and parameter.
()	Brackets enclose an expression (channel lists).

5.3 Description of Commands

Note:
For an overview of commands see the tables in Annex K.

5.3.1 Notation

In the following sections, all commands implemented in the device are described in detail. The notation corresponds to a large extent to that of the SCPI standards. The SCPI conformity information can be taken from the list of commands in Annex K.

Indentations The different levels of the SCPI command hierarchy are represented in the description by means of indentations to the right. The lower the level, the further the indentation to the right. Please observe that the complete notation of the command always includes the higher levels as well.

Example: `SENSe:FREQuency:MODE` is indicated in the description as follows:

SENSe	first level
. :FREQuency	second level
. . :MODE	third level

Upper-/lower-case notation

Upper/lower-case letters serve to mark the long or short form of the keywords of a command in the description (see next section). The device itself does not distinguish between upper- and lower-case letters.

Special characters |

A selection of keywords with an identical effect exists for some commands. These keywords are given in the same line and are separated by a vertical stroke. Only one of these keywords has to be indicated in the header of the command. The effect of the command is independent of the keywords being indicated.

```
Example: SENSe
                :FREQuency
                     :CW|:FIXed
```

The two following commands of identical meaning can be formed. They set the frequency of the device to 123 MHz:

```
SENSe:FREQuency:CW 123E6 = SENSe:FREQuency:FIXed 123E6
```

A vertical stroke in indicating the parameters marks alternative possibilities in the sense of "or". The effect of the command is different, depending on which parameter is entered.

Example: Selection of parameter for command

```
SENSe:GCONTrol:MODE    FIXed|MGC AUTO|AGC
```

If the parameter FIXed is selected, the gain is determined by the MGC voltage. In case of AUTO the gain depends on the signal.

The two parameters MGC and AGC are synonymous for FIXed and AUTO.

[ ]

Keywords in square brackets can be omitted in the header (see chapter 5.3.2"Common Commands"). The device has to accept the full command length due to reason of compatibility to SCPI standard.

Parameters in square brackets can also be optionally inserted into the command or can be omitted.

{ }

Parameters in braces can be inserted in the command either with the options not at all, once or several times.

5.3.2 Common Commands

The common commands are taken from the IEEE 488.2 (IEC 625-2) standard. A particular command has the same effect on different devices. The headers of these commands consist of an asterisk "*" followed by three letters. Many common commands concern the "Status Reporting System" on page 5.128.

Command	Parameter	Unit	Remarks
***CLS**			no query
***ESE**	0 to 255		
***ESR?**			only query
***IDN?**			only query
***IST?**			only query
***OPC**			
***OPT?**			only query
***PRE**	0 to 255		
***RST**			no query
***SRE**	0 to 255		
***STB?**			only query
***TRG**			no query
***TST?**			only query
***WAI**			

***CLS**

CLEAR STATUS sets the status byte (STB), the standard event register (ESR) and the EVENt sections of the QUEStionable and the OPERation register to zero. The command does not alter the mask and transition parts of the registers. It clears the output buffer.

***ESE 0 to 255**

EVENT STATUS ENABLE sets the event status enable register to the value indicated. Query *ESE? returns the contents of the event status enable register in decimal form.

***ESR?**

STANDARD EVENT STATUS QUERY returns the contents of the event status register in decimal form (0 to 255) and subsequently sets the register to zero.

***IDN?**

IDENTIFICATION QUERY queries the unit about identification.

The output of the unit can be:

"ROHDE&SCHWARZ,EM510,100.017/002, 01. 62-4065.8376.00"

100.017/002	=	serial number of the unit
01.62	=	firmware version number
4065.8376.00	=	firmware identity number

***IST?**

INDIVIDUAL STATUS QUERY states the contents of the IST flags in decimal form (0 | 1).

***OPC**

OPERATION COMPLETE sets the bit in the event-status register to 0 if all previous commands were carried out. This bit can be used for triggering a service request (see "Status Reporting System" on page 5.128).

***OPC?**

OPERATION COMPLETE QUERY writes the message '1' into the output buffer as soon as all previous commands were carried out (see "Status Reporting System" on page 5.128).

***OPT?**

OPTION IDENTIFICATION QUERY queries about the options in the unit and outputs a list of installed options. The options are separated by a comma.

Explanation for the output characters:

SU = Spectrum Unit; IF Panorama (software option)

PS = panorama scan (software option)

IM = ITU Measurement Function (software option)

SL = Selcall; Sel Call Analysis (software option)

Example for reply from the unit: SU,0,IM,SL

***PRE 0 to 255**

PARALLEL POLL ENABLE sets the parallel poll enable register to the value indicated. Query *PRE? returns the contents of the parallel poll enable register in decimal form.

*RST

RESET sets the device to a defined default status. The default setting is indicated in the description of the commands.

*SRE 0 to 255

SERVICE REQUEST ENABLE sets the service request enable register to the value indicated. Bit 6 (MSS mask bit) remains 0. This command determines under which conditions a service request is triggered. Query *SRE? reads the contents of the service request enable register in decimal form. Bit 6 is always 0.

*STB?

READ STATUS BYTE QUERY reads out the contents of the status byte in decimal form.

*TRG

TRIGGER triggers the same actions as the INITiate:CONM[:IMMediate] command.

*TST?

SELFTEST QUERY triggers the module state test and yields a figure which is to be interpreted as the bit field:

Result = 0 -> all modules ok

Result ≠ 0 -> fault in one or several modules. The information about the possible error can be queried by means of the `SYSTem:ERRor?` command:

*WAI

WAIT-to-CONTINUE only permits the servicing of the subsequent commands after all preceding commands have been executed and all signals have settled (also see "Status Reporting System" on page 5.128 and "*OPC").

5.3.3 ABORt Subsystem

ABORt

Stop command for measurement. This command stops an active scan.
Parameters:
none

**RST state:*
none, as command is an event

Example:

```
ABORt
```

5.3.4 CALCulate Subsystem

CALCulate
. :IFPan
. . :AVERage
. . . :TYPE MINimum|MAXimum|SCALar|OFF

Setting of the averaging procedure for the IF-panorama data

Parameters:

`MINimum`	MIN hold function is on
`MAXimum`	MAX hold function is on
`SCALar`	AVG averaging function is on
`OFF`	Switching on the Clear Write function

Note:
For `the` averaging procedure the averaging time and measuring time can be set by commands `MEASure:TIME`. The IF-panorama is also used for bandwidth measurement. With measure mode periodic the detectors are discharged in the cycle of the measure time. Therefore the display of the IF panorama "pulses" in the cycle of the measure time.

**RST state:*
`OFF`

Example:
```
CALCulate:IFPan:AVERage:TYPE MAXimum
```

. . . :TYPE?

Query about the averaging process of the IF-panorama data

Result:

`MIN`	MIN hold process is running
`MAX`	MAX hold process is running
`SCAL`	AVG averaging process is running
`OFF`	Switching on the Clear Write function

Example:
```
CALCulate:IFPan:AVERage:TYPE? -> MAX
```

. . :CLEar

Restart of the MIN or MAX hold function for the IF-panorama data

Parameters:
none

Example:
```
CALCulate:IFPan:CLEar
```

. . :MARKer:MAXimum[:PEAK]

Centering of the IF-panorama spectrum to the absolute level maximum

Parameters:
none

Example:
```
CALCulate:IFPan:MARKer:MAXimum
```

. . :MARKer:MAXimum:LEFT

Centering of the IF-panorama spectrum to the next relative level maximum left of the marker when the squelch is off. When it is on the center frequency is set to the next level maximum to the left which is above the squelch line.

Parameters:
none

Example:
```
CALCulate:IFPan:MARKer:MAXimum:LEFT
```

. . :MARKer:MAXimum:RIGHt

Centering of the IF-panorama spectrum to the next relative level maximum right of the marker when the squelch is off. When it is on the center frequency is set to the next level maximum to the right which is above the squelch line.

Parameters:
none

Example:
```
CALCulate:IFPan:MARKer:MAXimum:RIGHt
```

5.3.5 CALibration Subsystem

. : ROSCillator[:DATA] <numeric_value> [MINimum|MAXimum|UP|DOWN]

Changing the calibration value (D/A-converter value) for setting the exact OCXO reference frequency.

Parameter:

`<numeric_value>`	0 ... 4095

**RST state:*
none

default state (not calibrated)
2048

Example:

```
CALibration:ROSCillator UP
```

. : ROSCillator[:DATA]? [MINimum|MAXimum]

Query about the calibration value (D/A-converter value) for setting the exact OCXO reference frequency.

Parameter:

none	query about the current calibration value
`MINimum|MAXimum`	query about the lowest/highest calibration value

Result:
Calibration value or D/A-converter value for setting the exact OCXO reference frequency.

Example:

```
CALibration:ROSCillator? MAX -> 4095
```

. : ROSCillator:DATE <year>,<month>,<day>

Setting the calibration date.

Result:

```
<year>,<month>,<day>
<year> = 1900..
<month> = 1..12
<day> =1..31
```

Example:

```
CALibration:ROSCillator:DATE 2003,07,24
```

. : ROSCillator:DATE?

Query about the calibration date.

Result:

```
<year>,<month>,<day>
<year> = 1900..
<month> = 1..12
<day> =1..31
```

Example:

```
CALibration:ROSCillator:DATE? -> 2003,07,24
```

. : ROSCillator:STORe

Storing the calibration value for setting the exact OCXO reference frequency and the calibration date.

Note:
When storing the calibration value and the calibration date the store command must be sent in one single string. If the store command is sent without date an error message is generated: `1,"Device dependent error;Calibration store only with date"`

Example:

```
CALibration:ROSCillator:DATE 2003,07,24;STORe
```

5.3.6 DIAGnostic Subsystem

DIAGnostic[:SERVice]
### .	:INFO
### .	.	:SDATe<numeric_suffix>?

Query about the software-generation date
If a module is not available, a zero string ("") is returned and the error message `HW MISSING` will be generated.

Parameters:
The processor is selected via the `<numeric suffix>`:

1 or no <numeric suffix>	SW_VERSION_MAIN	version date of the PPC
2	SW_VERSION_IF	version date of the DSP
3	SW_VERSION_FPGA	version date of the FPGA codes
4	SW_VERSION_MB_CPLD	version date of the CPLD codes
5	SW_VERSION_BOOTPROG	version date of the bootstrap loader

Result:
`<year>,<month>,<day>`
`<year> =` **1900** to
`<month> =` **1 to 12**
`<day> =`**1 to 31**

Example:
`DIAGnostic:INFO:SDATe1? -> 2003,07,24`

### .	.	:SVERsion<numeric_suffix>?

Query about the software version.
If a module is not available, a zero string ("") is returned and the error message `HW MISSING` will be generated.

Parameters:
The processor is selected via the `<numeric suffix>`:

1 or no <numeric suffix>	SW_VERSION_MAIN	version date of the PPC
2	SW_VERSION_IF	version date of the DSP
3	SW_VERSION_FPGA	version date of the FPGA codes
4	SW_VERSION_MB_CPLD	version date of the CPLD codes
5	SW_VERSION_BOOTPROG	version date of the bootstrap loader

Result:
Software version and identification number of software in format Vxx.yy-aaaa.bbbb.cc (also see `*IDN?`)

Example:
`DIAGnostic:INFO:SVERsion1? ->V01.62-` 4065.8376.00

. . :MODule? <module_name>

Read-out of a module information.

Parameters:

`<module_name>`	Abbreviation of the name of the module to be queried: V1 preselection Z1 HF tuner MB mainboard ALL all modules

Result:

`<module_name>`,	Abbreviation of the module name
`<PartNumber>`,	Module ID (e.g. 4065.7840.02)
`<HwCode>`,	Hardware code (e.g. 1)
`<ProductIndex>`,	change index (e.g. 01.00)
`<SN>`,	serial number (e.g. 100.002/002)
`<ProductDate>`,	production date (e.g. 2006,01,26)
`<ModuleName>`,	Module name (e.g. "EM510MAIN")

If "ALL" was entered for `<module_name>`, then the information about all possible modules is output separated by commas.

Example:

```
DIAGnostic:INFO:MODule? MB
          ->  MB,4065.7840.02,1,01.00,100.002/002,2006,01,26,"EM510MAIN "
```

or

```
DIAGnostic:INFO:MODule? ALL
       ->
           V1,4066.1317.02,1,01.00,100.006/002,2005,12,01,"PRESELECTION ",
           Z1,4065.7840.02,101,01.00,100.002/002,2006,02,13,"HF TUNER ",
           MB,4065.7840.02,1,01.00,100.002/002,2006,01,26,"EM510MAIN ",
```

. :MODule:STATe? <module_name>

Output of additional infomation of one or all modules.

Parameters:

`<module_name>`	information relating to module to be queried:	
	V1	preselection
	MB	mainboard
	ALL	all modules

Result:

`<module_name>`,0	The module is "UNDEFINED". The EEPROM data are corrupt.
`<module_name>`,1	The module is "OK".
`<module_name>`,2	The module is "FAIL". Error message of a test point of the module.
`<module_name>`,3	The module is " NOT_INSTALLED ".

If "ALL" was input for `<module_name>` the status information of all possible modules is output separated by commas.

Example:

```
DIAGnostic:MODule:STATe? V1 -> V1,1
```

or

```
DIAGnostic:MODule:STATe? ALL ->  V1,1,MB,1
```

. :MONitor? <module>

Output of test-point information of one or all recognized modules.

Parameters:

`<module_name>`	information relating to module to be queried:	
	V1	preselection
	MB	mainboard
	ALL	all modules

Result:

If the output format is set to ASCII, all information relating to the test points of a known module are output in a table.
The table comprises the following columns:
module identification, test point name, symbol for test point state, current voltage in mV, lower limit, upper limit

Meaning of symbols for the test point state:
“ “ = OK, test point voltage within limits
“^ “ = test point voltage is greater than the upper limit
“v“ = test point voltage is less than the upper limit

If the limits are irrelevant in the current operating mode, no limits will be output.

Example:

```
DIAGnostic:MONitor? V1->
V1     TPREAMP_P       1437   (    900 /   1800 )
V1     TPREAMP_N        805   (    500 /   1200 )
V1     TTEMP           1492   (    800 /   2200 )
```

If the output format is set to binary format, a binary data block will be output which is similar to the structure described under "4.4.5 Parameters" followed by the test point descriptions with:

2 bytes module identifiers
12 bytes test point name
2 bytes current value in mV
1 byte OK flag for test point voltage (0= OK, 1 = too low, 2 = too high)
1 byte validity flag for limit values (0 = invalid, 1 = valid)
2 bytes minimum value in mV
2 bytes maximum value in mV

Example:

```
DIAGnostic:MONitor? V1 -> #3180xxxxxx
```

. :TEMPerature[<numeric_suffix>]?

Output of the temperature at various measurement points of the device.

Parameters:

The measurement point (1 to 5) is selected through the `<numeric suffix>`.

1 or none `<numeric suffix>`	TEMP_LOCAL	temperature on the mainboard
2	TEMP_FPGA	temperature of the FPGA
3	TEMP_PPC	temperature of the Power PC
4	TEMP_ADSECT	temperature of AD converter section
5	TEMP_PRESEL	temperature in the preselection

Result:

Current temperature in °C at the selected test point.

Example:

```
DIAGnostic:TEMPerature3? -> 37
```

. :TPOint[<numeric_suffix>]? <module>

Output of test point voltage of a module. The test point (1 to n_{max}) on the particular module is selected via `<numeric suffix>`:

Parameters:

`<module>` information relating to module to be queried:

V1	preselection	$n_{max} = 3$
MB	mainboard	$n_{max} = 16$

Result:

Current test point voltage in mV

Example:

```
DIAGnostic:TPOint1? V1 -> 1448
```

5.3.7 DISPlay Subsystem

DISPlay
. :MENU[:NAME] <menu_name>

Selection of a specific data channel for the video panorama display from the pre-set list. In addition to the channel, this sets the pre-conditioning of the selected time signal. Please note that the demodulation mode should correspond to the channel selection and pre-conditioning. The table below identifies some plausible combinations. See also command `SYSTem:VIDeo:REMote:MODE`.

	Data channel selection					
Demodulation modes	AM	FM	AMSQ	FMSQ	IQ	IQSQ
AM, FM, PULSE, PM	X	X	X	X		
IQ, ISB, CW, LSB, USB	X	X	X	X	X	X

Parameters:

`<menu_name>`	OFF	no video panorama data
	DEFault	no video panorama data
	LEFT	left / AM channel / I channel of the video panorama data
	AM	left / AM channel / I channel of the video panorama data
	RIGHt	right / FM channel / Q channel of the video panorama data
	FM	right / FM channel / Q channel of the video panorama data
	IQ	video IQ panorama data
	AMSQuare	squared AM channel / I channel of the video panorama data
	FMSQuare	squared FM channel / Q channel of the video panorama data
	IQSQuare	squared video IQ panorama data

**RST state:*
`OFF`

Example:
```
DISPlay:MENU LEFT
```

. :MENU[:NAME]?

Query about a specific data channel that was selected for the video panorama display.

Parameters:
none

Result:

`<menu_name>`	OFF	no video panorama data
	LEFT	left / AM channel / I channel of the video panorama data
	RIGHt	right / FM channel / Q channel of the video panorama data
	IQ	video IQ panorama data
	AMSQ	squared AM channel / I channel of the video panorama data
	FMSQ	squared FM channel / Q channel of the video panorama data
	IQSQ	squared video IQ panorama data

Example:
```
DISPlay:MENU? -> LEFT
```

5.3.8 FORMat Subsystem

FORMat
. :BORDer NORMal|SWAPped

Specifies whether binary data is first to be transferred with low or high byte.

Note:
This command affects only the trace data. For UDP data there is a separate setting option.

Parameters:

`NORMal`	MSB -> ... -> LSB
`SWAPped`	LSB ->... -> MSB

**RST state:*
`NORMal`

Example:
`FORMat:BORDer SWAPped`

. :BORDer?

Query about the output order for binary data.

Parameters:
none

Result:
`NORM, SWAP`

Result:
`FORMat:BORDer? -> SWAP`

. [:DATA] ASCii|PACKed

Specifies the output format of the following queries:
`SENSe:DATA?`
`TRACe:DATA?`

Parameters:

`ASCii`	output with ASCII format
`PACKed`	output with internal binary data format

**RST state:*
`ASCii`

Example:
`FORMat PACKed`

. [:DATA]?

Query about the output format of the above-mentioned queries.

Parameters:
none

Result:
```
ASC, PACK
```

Example:
```
FORMat? -> PACK
```

. :DIAGnostic:MONitor ASCii|PACKed

Sets the output format for the query `DIAGnostic:MONitor?`.

Parameters:

`ASCii`	`output in` ASCII format
`PACKed`	output in internal binary data format

**RST state:*
```
ASCii
```

Example:
```
FORMat:DIAGnostic:MONitor PACKed
```

. :DIAGnostic:MONitor?

Query about the above stated output format

Parameters:
none

Result:
```
ASC, PACK
```

Example:
```
FORMat:DIAGnostic:MONitor? -> PACK
```

. :MEMory ASCii|PACKed

Specifies the output format of the queries `MEMory:CONTents?`

Parameters:

`ASCii`	output in ASCII format
`PACKed`	output in internal binary data format

***RST state:**

`ASCii`

Example:

`FORMat:MEMory PACKed`

. :MEMory?

Query about the output format of above-mentioned queries.

Parameters:

none

Result:

`ASC, PACK`

Example:

`FORMat:MEMory? -> PACK`

. :SREGister ASCII|BINary|HEXadecimal

Specify with which data format the queries of all `CONDition`, `EVENt`, `ENABle`, `PTRansition`, `NTRansition` registers and all IEEE-488.2 status registers are to be carried out.

Parameters:

`ASCii`	output as decimal figure in ASCII code (e.g. 128)
`BINary`	output as binary figure in ASCII code (e.g. #B1000000000000000)
`HEXadecimal`	output as hexadecimal figure in ASCII code (e.g. #H8000)

**RST state:*

`ASCii`

Example:

`FORMat:SREGister HEXadecimal`

.   :SREGister?

Query with which data format the above-mentioned queries are carried out.

Parameters:
none

Result:
```
ASC, BIN, HEX
```

Example:
```
FORMat:SREGister? -> HEX
```

5.3.9 INITiate Subsystem

INITiate
.   [:IMMediate]

Start command to initiate measurement. Is also used as a start command for different SCAN types.

If `SENSe:FREQuency:MODE` is set on `CW|FIXed` a measurement is carried out with every `INITiate` command and the measurement result might be stored in `MTRACE` or `ITRACE`.

If `SENSe:FREQuency:MODE` is set on `SWEep|MSCan|PSCan`, the corresponding scan is started and for each step a measurement is carried out.
If, for example, the path is set to the measurement value buffer `MTRACE` by means of the command `TRACE:FEED:CONTrol MTRACE, ALWays` then the measurement values are stored in `MTRACE`.

Parameters:
none

Example:
`INITiate`

.   :CONM
.   .   [:IMMediate]

Command to **CON**tinue a **M**easurement. Is also used as continue command for different SCAN types.
`MTRACE` and `ITRACE` data sets are <u>not</u> deleted and are filled with measurement results according to setting:
If `SENSe:FREQuency:MODE` is set to `CW|FIXed`, a measurement is carried out and possibly stored in `MTRACE` or `ITRACE`.
If `SENSe:FREQuency:MODE` is set to `SWEep|MSCan|PSCan`, a measurement is carried out for each step and stored in `MTRACE` or `ITRACE`. As an alternative, the command `*TRG` or the interface message Group Execute Trigger (`GET`) can be used. The response time is the shortest for a `GET,` which is why a `GET` should always be used for time-critical measurements.

Parameters:
none

Example:
`INITiate:CONM`

5.3.10 INPut Subsystem

INPut
. :ATTenuation<numeric_value>

Setting the attenuator.

Parameters:

`<numeric_value>`	0 to 25 dB

***RST state:**

0

Example:

```
INPut:ATTenuation 15
```

. :ATTenuation?

Query about the current attenuation.

Parameters:
none

Result:

`<numeric_value>`	0 to 25 dB

Example:

```
INPut:ATTenuation? -> 15
```

. :ATTenuation:AUTO <Boolean>

Setting attenuation so that the best dynamic range is obtained; explicit switch on/off of attenuator sets `AUTO` to `OFF`.

Parameters:

`ON`	attenuation is coupled to input-signal strength
`OFF`	attenuation is manually switched

**RST state:*
`ON`

Example:
```
INPut:ATTenuation:AUTO ON
```

. :ATTenuation:AUTO?

Query about the automatically attenuation setting.

Parameters:
none

Result:
Attenuation is automatically switched 1
Attenuation is manually switched 0

Example:
```
INPut:ATTenuation:AUTO? -> 1
```

. :ATTenuation:AUTO:HOLD:TIME <numeric_value> | MIN | MAX

Setting the hold time, which prevents the attenuation value from dropping too early when the input level decreases.

Parameters:

`<numeric_value>`	hold time in seconds (0 s to 10 s)
`MINimum|MAXimum`	minimum/maximum hold time

**RST state:*
`MIN`

Example:
```
INPut:ATTenuation:AUTO:HOLD:TIME 3
```

. :ATTenuation:AUTO:HOLD:TIME? MIN | MAX

Query about the selected hold time.

Parameters:

none	query about the current hold time
`MINimum|MAXimum`	query about the minimum/maximum hold time

Result:
Hold time in seconds.

Example:
```
INPut:ATTenuation:AUTO:HOLD:TIME? -> 3
```

. . :MODE LOWDistort| LOWNoise|NORMal

Selection of operating modes for different receiving conditions.

In module "Preselection" there is a selectable amplifier behind the range filters.

When "**Low Distortion**" is selected, the amplifier is cut off. This achieves the best linearity but the noise figure will be correspondingly higher. This setting is especially recommended for areas with a lot of strong signals.

When "**Normal**" and "**Low Noise**" is selected, the amplifier is cut in. This achieves a good noise figure and at the same time good high-level signal characteristics because the receiving bandwidth is limited in front of the amplifier. This setting is recommended for normal receiving conditions.

At this setting with ATT OFF, excessively strong signals (approx. >97 dBµV) at the antenna entry may cause an overload of the A/D converter.

Notes:
As opposed to the EM550, the EM510 modes "**Normal**“ and "**Low Noise**“ are identical in meaning. However, both modes are permitted for reasons of compatibility.

Parameters:

`LOWDistort`	low-distortion reception
`NORMal | LOWNoise`	normal reception, low-noise reception

**RST state:*
```
NORMal
```

Example:
```
INPut:ATTenuation:MODE LOWDistort
```

. . :MODE?

Query about the selection of operating modes for different receiving conditions.

Parameters:
none

Result:

`LOWN`	reception with low noise

`LOWD`	reception with low distortion
`NORM`	normal reception

Example:

```
INPut:ATTenuation:MODE? -> LOWD
```

5.3.11 MEASure Subsystem

MEASure

. :BANDwidth:MODE XDB|BETA

Selection of bandwidth measurement method.

Parameters:

`XDB`	bandwidth measurement method x dB
`BETA`	bandwidth measurement method Beta %

**RST state:*

`XDB`

Example:

```
MEASure:BANDwidth:MODE  XDB
```

. :BANDwidth:MODE?

Query about the selected bandwidth measurement method.

Result:

`XDB`	bandwidth measurement method x dB
`BETA`	bandwidth measurement method Beta %

Example:

```
MEASure:BANDwidth:MODE? ->  XDB
```

. :BANDwidth:XDB<numeric_value>

Configuration of bandwidth measurement method x dB.

Parameter:

`<numeric_value>`	0.0 to 100.0 dB

**RST state:*

`26.0 dB`

Example:

```
MEASure:BANDwidth:XDB 31
```

. :BANDwidth:XDB?

Query about the configuration of bandwidth measurement method x dB.

Result:

dB value of bandwidth measurement method x dB.

Example:

```
MEASure:BANDwidth:XDB? -> 31.0
```

. :BANDwidth:BETA<numeric_value>

Configuration of bandwidth measurement method Beta %.

Parameter:

`<numeric_value>`	0.1 to 99.9 %

**RST state:*

```
1.0 %
```

Example:

```
MEASure:BANDwidth:BETA 10
```

. :BANDwidth:BETA?

Query about the configuration of bandwidth measurement method Beta %.

Result:

% value of bandwidth measurement method Beta %.

Example:

```
MEASure:BANDwidth:BETA? -> 10.0
```

. :MODE CONTinuous|PERiodic

Setting the continuous or periodic measuring mode.

In the **PERiodic** measurement mode all detectors are discharged after the measuring time has elapsed, and the next measurement is started. Only the individual measured values per measuring period are displayed.
If the path to the result buffer `MTRACE` is enabled by command `TRACE:FEED:CONTrol MTRACE, ALWays`, a measured value is stored in `MTRACE` each time the measuring time has elapsed.

In the **CONTinuous** measuring mode the measuring detector is read out every 200 msec, irrespective of the measuring time. These current measured values are displayed.
The measuring time has an effect on the level detectors. With AVG the measuring time determines the averaging time. With PEAK the measuring time determines the fall time. With FAST the measuring time does not have any impact since it is only the current value which is measured.
If the path to the result buffer `MTRACE` is enabled by command `TRACE:FEED:CONTrol MTRACE, ALWays`, a current measured value can be stored in `MTRACE`.

Parameters:

`CONTinuous`	continuous measurement
`PERiodic`	periodic measurement

**RST state:*

```
CONTinuous
```

Example:

```
MEASure:MODE PERiodic
```

. :MODE?

Query about the set measuring mode.

Parameters:
none

Result:

`CONT`	continuous measurement
`PER`	periodic measurement

Example:
`MEASure:MODE? -> PER`

. :TIME <numeric_value>|MINimum|MAXimum|DEFault

Setting the duration for all measurement functions.

Note:
The user is responsible for setting a useful measuring time. Time spans which are too short lead to faulty measurement results.

Parameters:

`<numeric_value>`	time span in seconds
`MINimum|MAXimum`	min/max time span
`DEFault`	default time span

**RST state:*
`DEFault`

Example:
`MEASure:TIME 200 ms`

. :TIME? [MINimum|MAXimum]

Query about the set measuring time.

Parameters:

none	queries the current time span
`MINimum|MAXimum`	queries the min/max time span

Result:
Time span in seconds; the default time span is indicated by `DEF`

Example:
`MEASure:TIME? -> 0.2`

5.3.12 MEMory Subsystem

This subsystem contains all the functions necessary to operate the EM510 memory locations. The memory locations are addressed with the text (see "Parameters" on page 5.9) `MEM0` to `MEM9999` (memory location 0 to memory location 9999). Some commands allow the receiver (data set of receiver settings) to be addressed by `Character Data RX,` the currently set memory location by `CURRENT` and the next free memory location by `NEXT` .

The number of the currently active memory location can also be queried by the `MSCAN:CHAnel`? command.

MEMory
. :CLEar <name> [,<count>|MAXimum]

Clearing the contents of a memory location. A certain number of memory locations to be cleared may also be specified.

Parameters

`<name>`	`MEM0` to `MEM9999 | CURRENT`
`<count>`	number of memory locations to be cleared from memory location `<name>`; as a default value `<count>` = 1 is accepted
`MAXimum`	clearing all memory locations from `<name>` to the last memory location

Example:
```
MEMory:CLEar MEM123
```

. :COPY <src_name>, <dest_name>

Copy the memory contents from src to dest.

Parameters:

`<src_name>|RX`	`MEM0` to `MEM9999 |RX | CURRENT`
`<dest_name>|RX`	`MEM0` to `MEM9999 |RX | CURRENT|NEXT`

Example:
```
MEMory:COPY MEM123, MEM10
```

. :CONTents <name>,<mem_paras>|<packed_struct>

Loading a memory location.
As an alternative to the parameter field (`<mem_paras>`) a `<Definite Length Block>` can be transferred with binary data.

Parameters:

`<name>`	`MEM0` to `MEM9999` | `RX` | `CURRENT` | `NEXT`
`<mem_paras>`	`<F>, <THR>, <DEM>, <BW>, <ANT>, <ATT>, <ATTA>, <SQUC>, <AFC>, <ACT>`
`<F>`	`frequency (see SENS:FREQ:CW)`
`<THR>`	`squelch threshold (see OUTP:SQU:THR)`
`<DEM>`	`type of demodulation (see SENS:DEM)`
`<BW>`	`bandwidth (see SENS:BWID)`
`<ANT>`	`antenna number (see ROUT:SEL)`
`<ATT>`	`attenuator (see INP:ATT)`
`<ATTA>`	`attenuator auto (see INP:ATT:AUTO)`
`<SQUC>`	`squelch function (see OUTP:SQU:STAT)`
`<AFC>`	`AFC function (see (SENS:FREQ:CW:AFC)`
`<ACT>`	`setting/resetting the memory to scan (ON/OFF or 1/0)`

`<packed_struct>`	binary data set as `<Definite Length Block>` with the following structure:
Frequency in Hz	4 bytes = unsigned long integer
Squelch threshold in 1/10 dBuV	2 bytes = signed integer
Demodulation type	2 bytes = meaning: 0 = `FM`, 1 = `AM`, 2 = `PULSe`, 3 = `PM`, 4 = `IQ`, 5 = `ISB`, 6 = `CW`, 7 = `USB`, 8 = `LSB`
Bandwidth	2 Byte = enumeration: 0 = 100 Hz, 1 = 150 Hz, 2 = 300 Hz, 3 = 600 Hz, 4 = 1 kHz, 5 = 1.5 kHz, 6 = 2.1 kHz, 7 = 2.4 kHz, 8 = 2.7 kHz, 9 = 3.1 kHz, 10 = 4 kHz, 11 = 4.8 kHz, 12 = 6 kHz, 13 = 9 kHz, 14 = 12 kHz, 15 = 15 kHz, 16 = 30 kHz, 17 = 50 kHz, 18 = 120 kHz, 19 = 150 kHz, 20 = 250 kHz, 21 = 300 kHz, 22 = 500 kHz 23 = 800 kHz 24 = 1 MHz 25 = 1.25 MHz 26 = 1.5 MHz 27 = 2.0 MHz 28 = 5.0 MHz 29 = 10.0 MHz
Antenna number	1 byte = unsigned character 0 to 99
Attenuator	1 byte = unsigned character (0 to 25)
Attenuator auto	1 byte = unsigned character (1 = on / 0 = off)
Squelch function	1 byte = unsigned character (1 = on / 0 = off)
AFC function	1 byte = unsigned character (1 = on / 0 = off)
Set/Reset memory	1 byte = unsigned character (1 = set / 0 = reset)

Total number of bytes = 16

Notes:

- *When loading the receiver data set (`RX`) the parameter `<ACT>` is ignored. It must however be specified.*
- *When loading with `<packed_struct>` the byte order within the 2- and 4-byte elements is determined by the setting command `FORMat:BORDer`.*

**RST state:*

The contents of the memory locations are kept after `*RST`.

Example:

```
MEMory:CONTents    MEM1,801 kHz,34, AM ,9 kHz,(@1),1,OFF,ON,OFF,ON
```

. :CONTents? <name>|RX

Query about contents of memory location.

Parameters:

`<name>`	`MEM0 to MEM9999	RX	CURRENT`

Result:

Depending on the setting by the command `FORMat:MEMory` either an ASCII data set or a binary data set is output:

The ASCII data set has the following structure:

`<F>,<THR>,<DEM>,<BW>,<ANT>,<ATT>,<ATTA>,<SQUC>,<AFC>,<ACT>`

`<F>`	frequency (see SENS:FREQ:CW?)
`<THR>`	squelch threshold (see OUTP:SQU:THR?)
`<DEM>`	demodulation type (see SENS:DEM?)
`<BW>`	bandwidth (see SENS:BWID?)
`<ANT>`	antenna number (see ROUT:CLOS:STAT?)
`<ATT>`	attenuator (see INP:ATT?)
`<ATTA>`	attenuator auto (see INP:ATT:AUTO?)
`<SQUC>`	squelch function (see OUTP:SQU:STAT?)
`<AFC>`	AFC function (see (SENS:FREQ:CW:AFC?)
`<ACT>`	set/reset for scan (1/0)

The binary data set is transferred as a `<Definite Length Block>` and has to be interpreted according to the above-mentioned format.

Notes:

- *During a query about the receiver data set (`RX`) the parameter `<ACT>` is not defined and has to be ignored.*
- *When trying to read out an empty memory location the error message "MEMORY EMPTY" is generated.*

Example:

```
MEMory:CONTents? MEM1 ->    801000,34,AM,9000,#14(@1),1,0,1,0,1
```

. . :MPAR <name>,<ACT>

Setting the memory location parameter (`MPAR = MemoryPARameter`) `<ACT>` .

Parameters:

`<name>`	`MEM0` to `MEM9999` | `CURRENT`
`<ACT>`	setting/resetting the memory to the scan (`ON/OFF` or `1/0`)

Example:

```
MEMory:CONTents:MPAR MEM1, OFF
```

. . :MPAR? <name>

Query about memory-location parameter `<ACT>.`

Parameters:

`<name>`	`MEM0` to `MEM9999` | `CURRENT`

Result:

`<ACT>`	set/reset for scan (1/0)

Example:

```
MEMory:CONTents:MPAR? MEM1 -> 0
```

. :EXCHange <name1>, <name2>

Exchange of contents of two memory locations.

Parameters:

`<name1>`	`MEM0` to `MEM9999` | `RX` | `CURRENT`
`<name2>`	`MEM0` to `MEM9999` | `RX` | `CURRENT`

Example:

```
MEMory:EXCHange MEM123, RX
```

5.3.13 OUTPut Subsystem

OUTPut
. :AUXMode FREQuency|ANTCtrl

The switch "AUXMode" determines whether the frequency in BCD or the antenna number in BCD and the CTRL byte are output via X12b at the front panel.

Parameters:

`FREQuency`	frequency output at "AUX" 4-digit BCD (10, 100 kHz, 1 MHz, 10 MHz)
`ANTCtrl`	output of antenna number in 2-digit BCD (ANTA1 to ANTA80) output of CTRL byte, binary (CTRL1 to CTRL8)

**RST state:*
`ANTCtrl`

Example:
`OUTPut:AUXMode FREQuency`

. :AUXMode?

Query about the "AUXMode" setting.

Parameters:
`none`

Result:
`FREQuency` (frequency output) or `ANTCtrl` (output of antenna number and CTRL byte)

Example:
`OUTPut:AUXMode?` → `FREQuency`

:BITAux [<numeric_suffix>]
.   .   [:STATe] <Boolean>

Sets the AUX bits at the rear panel.

<numeric_suffix>

1	byte 1 corresponds to CTRL1 at X12B.14'AUX'
2	byte 2 corresponds to CTRL2 at X12B.15'AUX'
3	byte 3 corresponds to CTRL3 at X12B.16'AUX'
4	byte 4 corresponds to CTRL4 at X12B.17'AUX'
5	byte 5 corresponds to CTRL5 at X12B.18'AUX'
6	byte 6 corresponds to CTRL6 at X12B.19'AUX'
7	byte 7 corresponds to CTRL7 at X12B.20'AUX'
8	byte 8 corresponds to CTRL8 at X12B.21'AUX'

**Parameters:*

`ON`	Bit set to high level
`OFF`	Bit set to low level

RST state
`OFF`

Example:
```
OUTPut3 : BITAux2 ON
```

.   .   [:STATe] ?

Query about AUX bits at the rear panel.

Parameters:
`none`

Result:

`1`	"high" level bit set
`0`	"low" level bit set

Example:
```
OUTPut : BITAux2? → 1
```

. :BYTAux
. . [:STATe] <numeric_value>

Sets the 8 AUX bits by a single byte command.

Parameters:

`<numeric_value>`	value of the AUX bytes (`0 to 255, #H00 to #HFF or #B0 to #B11111111`)

**RST state:*
`0`

Example:
`OUTPut : BYTAux 7`

. :BYTAux
. . [:STATe]?

Query about the 8 AUX bits by a single byte command.

Parameters:
none

Result:
Depending on the settings by the FORMat : SREGister command, the contents of the register are transferred decimally, binary or hexadecimally in ASCII code.

Example:
`OUTPut : BYTAux? → 7`

. :FILTer: MODE OFF | NOTCh | NR | BP

This switch sets the audio filter mode.

Parameters:

`OFF`	no filter function
`NOTCh`	automatic elimination of interference signals
`NR`	noise reduction filter
`BP`	bandpass filter 300 Hz to 3.3 kHz

**RST state:*
`OFF`

Example:
`OUTPut:FILTer:MODE NOTCH`

. :FILTer:MODE?

Query about the activated audio filter mode.

Parameters:
none

Result:

OFF	no filter function
NOTC	automatic elimination of interference signals
NR	noise reduction filter
BP	bandpass filter 300 Hz to 3.3 kHz

Example:
```
OUTPut:FILTer:MODE? ->NOTC
```

. :FPDP:MODE IF|DEModulator|PANorama

With this switch the kind of data is determined that is output via the FPDP interface.

Parameters:

IF	IQ data, digital IF data (before the demodulation) (see also: SYSTem:FPDP:REMote:MODE)
DEModulator	video streaming data (after the demodulation), 2-channelled
PANorama	IQ data from the panorama path

Depending on the selected demodulation mode, DEModulator results in the following output:

Demodulation mode	**FPDP A**	**FPDP B**
AM, FM, PULSE	AM	FM
PM	AM	PM
USB	I	Q
LSB	I	Q
CW	I	Q
IQ	I	Q
ISB	I	Q

The format of the IQ data (IF and PANorama) is set with command SYSTem:FPDP:REMote:MODE.

**RST-state:*
IF

Example:
```
OUTPut:FPDP:MODE PANorama
```

. :FPDP:MODE?

Query about the kind of data that is output via the FPDP interface.

Parameters:
`none`

Result:

`IF`	IQ data, digital IF data (before the demodulation) (see also: `SYSTem:FPDP:REMote:MODE`)
`DEM`	video streaming data (after the demodulation), 2-channelled
`PAN`	IQ data from the panorama path

Example:
```
OUTPut:FPDP:MODE? ->PAN
```

:SQUelch
. . [:STATe] <Boolean>

Switch on/off of squelch.

Parameters:

`ON`	squelch on
`OFF`	squelch off

**RST state:*
`OFF`

Example:
```
OUTPut:SQUelch ON
```

. . [:STATe]?

Query about squelch setting.

Parameters:
none

Result:

1	squelch on
0	squelch off

State:
```
OUTPut:SQUelch? -> 1
```

. . :CONTrol MEMory|NONE

Selection of the source for the operating state after switching the unit on, when reading the memory locations by the MEMory:COPY command, when using the RCL key or when running memory scan.

Parameters:

`MEMory`	squelch state and squelch value are read out of the memory locations
`NONE`	squelch state and squelch value are **not** read out of the memory locations

**RST state*
`MEMory`

Example:
```
OUTPut:SQUelch:CONTrol  NONE
```

. . :CONTrol?

Query about the source of squelch setting when reading memory locations.

Parameters:
none

Result:
`MEM, NONE`

Example:
```
OUTPut : SQUelch : CONTrol? → MEM
```

.   .   :THReshold
.   .   .   [:UPPer] <numeric_value>|UP|DOWN|MINimum|MAXimum

Setting the squelch threshold.

Parameters:

`<numeric_value>`	squelch threshold in dBuV
`UP|DOWN`	increase|decrease of squelch threshold by the value set with the command `OUTPut:SQUelch:THReshold[:UPPer]:STEP[:INCRement]`.
`MINimum|MAXimum`	sets the lowest/highest squelch threshold

**RST state:*
10 dBuV

Example:
`OUTPut:SQUelch:THReshold 35 dBuV`

.   .   .   ? [MINimum|MAXimum]

Query about the squelch threshold.

Parameters:

none	query about current squelch threshold
`MINimum|MAXimum`	query about lowest/highest squelch threshold

Result:
Level tone reference value in dBuV

Example:
`OUTPut:SQUelch:THReshold? -> 35`

.   .   .   .   :STEP
.   .   .   .   .   [:INCRement] <numeric_value>|MINimum|MAXimum

Setting the stepwidth for the command `OUTP:SQU:THR[:UPP] UP|DOWN.`

Parameters:

`<numeric_value>`	stepwidth of squelch threshold in dBuV
`MINimum|MAXimum`	sets the smallest|largest stepwidth

**RST state:*
`1 dBuV`

Example:
`OUTP:SQU:THR:STEP 10 dBµV`

. . . . . [:INCRement]? [MINimum|MAXimum]

Query about the stepwidth.

Parameters:

none	query about currently set stepwidth
`MINimum|MAXimum`	query about smallest|largest stepwidth

Result:
Stepwidth of squelch threshold in dBμV

Example:

```
OUTP:SQU:THR:STEP? -> 10
```

. :VIDeo:FREQuency <numeric_value>|MINimum|MAXimum

Setting the center frequency of the analog IF output. This command is only effective if the video mode was set on IF.

Parameters:

`<numeric_value>`	frequency value
`MINimum|MAXimum`	sets the lowest/highest center frequency

**RST state:*
`10700000`

Example:
`OUTPut:VIDeo:FREQuency 15 MHz`

. : VIDeo:FREQuency? [MINimum|MAXimum]

Query about the video mode settings.

Parameters:

`none`	query about the currently set center frequency
`MINimum|MAXimum`	query about the lowest/highest center frequency

Result:
center frequency `in Hz`

Example:
`OUTPut:VIDeo:FREQuency? → 15000000`

. :VIDeo:MODE IF|DEModulator

The video-mode switch determines whether the analog IF or the analog demodulated video signal is output at the front panel through one of the connectors "Video A X6" or "Video B X7". In the case of an analog IF, the center frequency can be set by the `OUTPut:VIDeo:FREQuency` command.

Parameters:

`IF`	output of the analog IF
`DEModulator`	output of the analog demodulated video signal

**RST state:*
`DEM`

Example:
`OUTPut:VIDeo:MODE IF`

. : :VIDeo:MODE?

Query about the current video mode.

Parameters:
`none`

Result:

`IF`	output of the analog IF
`DEM`	output of the analog demodulated video signal

Example:
`OUTPut:VIDeo:MODE? → IF`

5.3.14 ROUTe Subsystem

Two signal path switches (signal routing) are equipped in the EM510.
The command ROUTe without numeric suffix or with numeric suffix 1 is used for the antenna selector 1 to 99.
The command ROUTe with numeric suffix 2 is used for the crosspoint switch for Gigacast, FPDP and the regenerating paths.

ROUTe
. :CLOSe <channel_list>

Selection of an antenna; the previous antenna has to be switched off with `ROUTe:OPEN:ALL` (also see ROUTe:SELect)
Error message:
If more than one antenna is to be selected, an execution error `-221,"Settings conflict"` will be generated.

Parameters:

`<channel_list>`	may contain max. one number (0 to 99)

**RST state:*
`@1`

Example:
`ROUTe:CLOSe (@23)`

. :CLOSe? <channel_list>

Query about whether the corresponding antenna has been selected.

Parameters:

`<channel_list>`	contains a value for each antenna number to be queried

Result:

0	for each non-selected antenna number
1	for each selected antenna number

Example:
`ROUTe:CLOSe? (@2, 10:12, 23) -> 0,0,0,0,1`

. . :STATe? [MINimum|MAXimum]

Query about the antenna that has been selected.

Parameters:

none	query about the currently selected antenna
`MINimum|MAXimum`	query about the lowest|highest antenna number

Result:
Antenna number as a `<Definite Length Block>`

Example:
`ROUTe:CLOSe:STATe? -> #15(@23)`

. OPEN
. . :ALL

Do not select antenna (antenna number 0 is set).

Parameters:
none

**RST state:*
none, as command is an event

Example:
```
ROUTe:OPEN:ALL
```

. SELect <channel_list>|UP|DOWN|MINimum|MAXimum

Corresponds to the following combination:
```
ROUTe:OPEN:ALL
ROUTe:CLOSe <channel_list>
```

Parameters:

`<channel_list>`	must contain one number max. (0 to 99)
`UP|DOWN`	goes in the list of antennas one position up or down
`MINimum|MAXimum`	selects antenna with the smallest or the biggest number

**RST state:*
see `ROUTe:CLOSe`

Example:
```
ROUTe:SELect (@23)
```

5.3.15 SENSe Subsystem

[SENSe]
.   :BANDwidth|BWIDth
.   .   [:RESolution] <numeric_value>|UP|DOWN|MINimum|MAXimum

Selection of demodulation bandwidth.
Since in the EM510 the signal path is also used for the computation of the IF panorama, both the selection of the IF panorama and the selected demodulation bandwidth affect the selection of the preselection filter ranges. The broader frequency range determines the selection of the preselection filter range.

Parameters:

`<numeric_value>`	value of bandwidth
`UP|DOWN`	to next|previous bandwidth
`MINimum|MAXimum`	sets the narrowest|widest bandwidth

**RST state:*
9 kHz

Example:
```
BANDwidth 2.4 kHz
```

.   .   [:RESolution]? [MINimum|MAXimum]

Query about the current demodulation bandwidth.

Parameters:

none	query about the current bandwidth
`MINimum|MAXimum`	query about the narrowest|widest bandwidth

Result:
IF bandwidth in Hz without unit specified

Example:
```
BANDwidth? -> 2400
```

. : CORRection:ATTenuation?

Read out of the current total attenuation in the signal path from antenna input to IQ output. The attenuation depends on the current receiver frequency and the current temperature.

The absolute signal power can be calculated by adding this value to the IQ data level dB full scale. The automatic attenuation selection and the automatic gain control has to be switched off in this case.

Result:
Attenuation in dB

Example:

```
 CORRection:ATTenuation? -> 105
```

. : DECoder:SELCall[:STATe] ON|OFF|1|0

Switching on and off the Sel Call Analysis.

The following selective call methods can be detected and decoded:

CCIR7(2), CCIR1, CCITT, EEA, EIA, EURO, NATEL, VDEW, ZVEI1, ZVEI2, DTMF, CTCSS, DCS

The decoder automatically detects the most probable code and only this code is output. If several codes of equal probability are detected then all codes are output.

The codes are output exclusively via UDP (see Annex F: Datagram Communication).

Required receiver settings

Selective call methods are generally frequency and phase modulated (FM/PM). Therefore the receiver must be set for this type of demodulation. The bandwidth for the demodulation must conform to the signal that is generally between 15 and 30 kHz..

For the testing of the selective-calling functionality there is a web page within the device. This page can be called up via <IP address>/selcall.html.

Note:

Sel Call Analysis is only accessible with the installed software option EM510SL (Selcall).

Parameters:

ON|1	Decoder switched on
OFF|0	Decoder switched off

**RST state:*
`0`

Example:
`DEC:SELC ON`

. : DECoder:SELCall[:STATe]?

Query about the state of Sel Call Analysis.

Parameters:
none

Result:

1	Decoder switched on
0	Decoder switched off

Example:
`DEC:SELC? → 1`

. :DATA? [<data_handle>]

Query about the current measured values of active sensor functions.
When only the command `SENSe:DATA?` is used to query measured values, the measured values reported back may be as old as 200 ms. For display on the unit measured values are captured every 200 ms and put into a buffer.

In the event that current data are needed the command combination `INIT;:SENSe:DATA?` is to be used. As response to this query the current measured value is reported back.

When a complete measurement is to be started, possibly by using a predefined measuring time, for instance the command combination `INIT;:SENSe:DATA?` should be used. As a result, the measurement history is reset, i.e. the detectors are discharged, a measurement is started and the result is reported back when the measuring time has elapsed.

In order not to block remote-control communication during longer measuring times, the measured value should only be queried when the measurement has been completed.
The measured value thus obtained is stored in `MTRACe`, provided that the path to the result buffer `MTRACE` was enabled with command `TRACE:FEED:CONTrol MTRACE, ALWays`.
The unit may actively report the end of measurement (MEASUring bit in operation status register becomes inactive) via SRQ if the status register has been configured accordingly (see also "Status Reporting System" on page 5.128).

Note:
For this command the keyword `SENSe` must not be omitted as `DATA?` can be mixed up with the subsystem `TRACe/DATA`.

Parameters:

none	output of the measured values of all active sensor functions; if no function is switched on, an error `-221, "Settings Conflict"` will be generated
`"[SENSe:]FREQuency:OFFSet"`	output of offset value or error
`"[SENSe:]VOLTage:AC"`	output of level value or error `-221, "Settings Conflict"`

Note:
With SW option EM510IM (ITU Measurement) installed, apart from level and offset the measurement functions AM modulation index , FM frequency deviation, PM phase deviation and bandwidth measurement are also available.

`"AM"`	output of AM modulation index measurement value
`"AM:POSitive"`	output of AM positive modulation index measurement value
`"AM:NEGative"`	output of AM negative modulation index measurement value
`"FM"`	output of FM frequency deviation measurement value
`"FM:POSitive"`	output of FM positive frequency deviation measurement value
`"FM:NEGative"`	output of FM negative frequency deviation measurement value
`"PM"`	output of PM phase deviation measurement value
`"BANDwidth"`	output of band width measurement value

Result:

Level in dBuV, offset in Hz

The output format will be generated with the command `FORMat:DATA` according to the setting:

`ASCii`	normal ASCII output
`PACKed`	`<Definite Length Block>`: level in 1/10 dBuV (2 Byte) offset in Hz (4 Byte) AM-modulation index in 1/10 % (2 bytes), AM-positive modulation index in 1/10 % (2 bytes), AM-negative modulation index in 1/10 % (2 bytes), FM frequency deviation in Hz (4 bytes), positive frequency deviation in Hz (4 bytes), negative frequency deviation in Hz (4 bytes), phase deviation in 1/100 rad (2 bytes), bandwidth in Hz (4 bytes)

Examples:

```
SENSe:DATA? -> 23.4, -2500
SENSe:DATA? "VOLT:AC" -> 23.4
SENSe:DATA? "FREQuency:OFFSet" -> -2500
```

. :DEModulation AM|FM|PULSe|PM|A0|IQ|ISB|A1|CW|LSB|USB

Switchover of type of demodulation.

Error message:

If the set bandwidth exceeds 9 kHz at CW, LSB and USB, an error `-221,"Settings conflict"` will be generated if one of the SSB operating modes is to be switched on.

Parameters:

`FM`	switch on FM demodulator
`AM`	switch on AM demodulator
`PULSe`	switch on pulse demodulator
`PM`	switch on PM demodulator
`IQ or A0`	switch on IQ demodulator
`ISB`	switch on ISB demodulator
`CW or A1`	switch on SSB demodulator 1 kHz beat
`USB`	switch on SSB demodulator upper sideband
`LSB`	switch on SSB demodulator lower sideband

*RST state:
`AM`

Example:
`DEModulation FM`

. :DEModulation?

Query about the demodulation type.

Parameters:
none

Result:
`FM, AM, PULS, PM, IQ, ISB, CW, USB, LSB`

Example:
`DEModulation? -> FM`

. :DEModulation:BFO <numeric_value> | MIN | MAX

Setting the BFO frequency. The BFO is an auxiliary oscillator which, in CW mode, helps to recover carriers.

Parameters:

`<numeric_value>`	BFO frequency (-8 kHz to +8 kHz)
`MINimum|MAXimum`	setting the minimum/maximum frequency

***RST state:**

1000

Example:

```
DEModulation:BFO 1 kHz
```

. :DEModulation:BFO? MIN | MAX

Query about the BFO frequency.

Parameters:

`none`	query about the current frequency
`MINimum|MAXimum`	query about the minimum/maximum frequency

Example:

```
DEModulation:BFO? -> 1000  (result: current frequency)
```

. :DETector

. . [:FUNCtion] POSitive|PAVerage|FAST|RMS

Switching over the level-measuring process.

Parameters:

`POSitive`	measuring the peak value (PEAK)
`PAVerage`	measuring the average value of the voltage (AVG)
`FAST`	measuring the current value (FAST)
`RMS`	measuring the average value of the power (RMS)

**RST state:*
`PAVerage`

Example:
`DETector POSitive`

. . [FUNCtion]?

Query about the current set level-measuring process.

Parameters:
none

Result:
`POS,PAV,FAST FAST or RMS`

Example:
`DETector? → POS`

. :FM

. . :RDS[:STATe] 0|1|ON|OFF

Switching on or off the RDS decoder.

Note:
The RDS decoder is accessible only with installed software option EM510IM (ITU Measurement).

Parameters:

ON|1	Decoder switched on
OFF|0	Decoder switched off

***RST state:**

**RST state:*
`0`

Example:
```
FM:RDS ON
```

. . :RDS[:STATe]?

Query about the state of the RDS decoder

Parameters:
none

Result:

1	Decoder switched on
0	Decoder switched off

Example:
```
FM:RDS? → 1
```

. . :RDS:DATA?

Query about the RDS data.

Parameters:
none

Result:
Flags, PI-Code, TP, TA, MS, DI

Meaning:

Flags: Bit 0 -> Stereo pilot tone detected / not detected

Bit 1 -> ARI carrier detected / not detected

Bit 2 -> RDS synchronized / not synchronized

PI-Code: Program Identifier

TP: Traffic Program (1=traffic radio transmitter, 0=no traffic radio transmitter)

TA: Traffic Announcement (1=traffic announcement running)

MS: Music/Speech (transmitting 1=music, 0=speech)

DI: Decoder information

0	mono
1	stereo
2	not assigned
3	dummy head
4	mono compressed
5	stereo compressed
6	not assigned
7	dummy head compressed
8.. 15	not assigned

Example:

```
FM:RDS:DATA? → 6, 54035, 1, 0, 1, 1
```

. . :RDS:PS?

Query about RDS program string name.

Parameters:
none

Result:
Definite length binary block Identification of the radio station coded in binary data
'_' stands for characters not transmitted yet

Example:
`FM:RDS:PS?` → #18BAYERN 3

. . :RDS:RT?

Query about RDS radio text.

Parameters:
none

Result:
Definite length binary block Information text of the radio station; is exchanged cyclically.
'_' stands for characters not transmitted yet

Example:
`FM:RDS:RT?` → #264Bayern 3__- Klingt dreimal gut ________________________________

. . :RDS: GROups[:DATA]?

Query about RDS group code statistics.

Parameters:
none

Result:
Group0 Version A, Group1 Version A,....Group15 Version A,
Group0 Version B, Group1 Version B,....Group15 Version B

Meaning:

Indicates the number of separately transmitted group codes (0-15) for versions A and B ;

Example:

```
FM:RDS:GRO? → 1727, 0, 866, 147, 5, 0, 439, 0, 384, 0, 0, 400, 0, 0, 880,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

. . :RDS: GROups:CLEar

Reset of the RDS group codes.

Parameters:
none

Example:
`FM:RDS:GROups:CLEar`

. . :STEReo[:STATe] 1|0|ON|OFF

Switching on/off the stereo decoder.

Parameters:

1|ON	switching on the stereo decoder
0|OFF	switching off the stereo decoder

Example:

```
FM:STEReo  ON
```

. . :STEReo[:STATe]?

Query about the state of the stereo decoder.

Parameters:
none

Example:

```
FM:STEReo?  → 1
```

Notes:

The RDS decoder and the stereo decoder work only with FM demodulation. The bandwidth must be > = 120 kHz and <= 250 kHz.

The detector flags work only when the detector is working (see also the description of the SCPI commands `FM:RDS:DATA?` on page 5.67).

Stereo flag :	stereo decoder must be switched on
ARI/RDS flag:	RDS decoder must be switched on

If the stereo decoder detects the stereo pilot tone then it automatically switches the audio output over onto two-channel stereo. The stereo information in the RDS field DI is then independent of the stereo code in the flags field.

For the testing of the RDS-functionality there is a web page within the device. This page can be called up via <IP address>/rds.html.

. :FREQuency

. . :AFC <Boolean>

Switching on/off the AFC function.

Parameters:

`ON`	AFC function on
`OFF`	AFC function off

**RST state:*

`OFF`

Example:

```
SENSe:FREQuency:AFC ON
```

. . :AFC?

Query about the AFC function

Parameters:

none

Result:

AFC function on	`1`
AFC function off	`0`

Example:

```
SENSe:FREQuency:AFC? -> 1
```

. . [:CW|FIXed] <numeric_value>|UP|DOWN|MINimum|MAXimum

Setting the receiver frequency.

Parameters:

`<numeric_value>`	frequency value
`UP|DOWN`	increase|decrease of receiver frequency by the value set by command `SENS:FREQuency[:CW|FIX]:STEP[:INCRement]`
`MINimum|MAXimum`	sets the lowest|highest receiver frequency

**RST state:*

801 kHz

Example:

```
FREQuency 801 kHz
```

.   .   [:CW|FIXed]? [MINimum|MAXimum]

Query about the receiver frequency.

Parameters:

none	query about the current receiver frequency
`MINimum|MAXimum`	query about the lowest|highest receiver frequency

Result:
Frequency value in Hz

Example:
```
FREQuency? -> 801000
```

.   .   :MODE CW|FIXed|SWEep|MSCan|PSCan

Changing the operating mode of the receiver.

Parameters:

`CW	FIXed`	receiver monitors a frequency (CW and FIXed have equal meanings)
`SWEep`	receiver is in frequency-scan mode (see `SENSe:SWEep`)
`MSCan`	receiver is in memory-scan mode (see `SENSe:MSCan`)
`PSCan`	receiver is in panorama-scan mode (see `SENSe:PSCan`)

Note:
The receiver stays on the CW frequency until it starts scanning!

**RST state*
`CW`

Example:
```
FREQuency : MODE  SWEep
```

.   .   :MODE?

Query about the receiver operating mode.

Parameters:
none

Result:
```
CW, SWE, MSC, PSC
```

Example:
```
FREQuency:MODE? -> SWE
```

. . :PSCan

If the `STARt` and/or the `STOP` frequency is changed, `CENTer` or `SPAN` is matched.
If `CENTer` and/or `SPAN` is changed, `STARt-` and `STOP` frequency are matched.

In a command either CENTer and SPAN or STARt- and STOP frequency can be changed simultaneously. Other combinations of parameters are rejected.

. . . :CENTer <numeric_value>|MINimum|MAXimum

Setting of the center frequency of the panorama scan.
Corrects `STARt-` and `STOP` frequency.

Parameters:

`<numeric_value>`	frequency value
`MINimum|MAXimum`	setting of the lowest/highest center frequency

**RST state:*
`1.5 MHz`

Example:
`FREQuency:PSCan:CENTer 3 MHz`

. . . :CENTer? [MINimum|MAXimum]

Query about the center frequency of the panorama scan.

Parameters:

none	query about the current center frequency
`MINimum|MAXimum`	query about the lowest/highest center frequency

Result:
Frequency value in Hz

Example:
`FREQuency:PSCan:CENTer? → 3000000`

. . . :SPAN <numeric_value>|MINimum|MAXimum

Setting of the display range of the panorama scan.
Corrects `STARt` and `STOP` frequency.

Parameters:

`<numeric_value>`	frequency value
`MINimum|MAXimum`	setting of the smallest/widest display range

**RST state:*
`1 MHz`

Example:
`FREQuency:PSCan:SPAN 2 MHz`

.   .   .   :SPAN? [MINimum|MAXimum]

Query about the display range of the panorama scan.

Parameters:

none	query about the current display range
`MINimum|MAXimum`	query about the smallest/widest display range

Result:
Frequency value in Hz

Example:
```
FREQuency:PSCan:SPAN? -> 2000000
```

.   .   .   :STARt <numeric_value>|MINimum|MAXimum

Setting of the start frequency of the panorama scan.

Parameters:

`<numeric_value>`	frequency value
`MINimum|MAXimum`	setting of the lowest/highest start frequency

**RST state:*
```
1 MHz
```

Example:
```
FREQuency:PSCan:STARt 1.5 MHz
```

.   .   .   :STARt? [MINimum|MAXimum]

Query about the start frequency of the panorama scan.

Parameters:

none	query about the current start frequency
`MINimum|MAXimum`	query about the lowest/highest start frequency

Result:
Frequency value in Hz

Example:
```
FREQuency:PSCan:STARt? → 1500000
```

. . . :STOP <numeric_value>|MINimum|MAXimum

Setting of the stop frequency of the panorama scan.

Parameters:

`<numeric_value>`	frequency value
`MINimum|MAXimum`	Setting of the lowest/highest stop frequency

**RST state:*
`2 MHz`

Example:
```
FREQuency:PSCan:STOP 4 MHz
```

. . . :STOP? [MINimum|MAXimum]

Query about the stop frequency of the panorama scan.

Parameters:

none	query about the current stop frequency
`MINimum|MAXimum`	query about the lowest/highest stop frequency

Result:
Frequency value in Hz

Example:
```
FREQuency:PSCan:STOP? → 4000000
```

. . :SPAN<numeric_value>|UP|DOWN |MINimum|MAXimum

Selection of frequency range with option IF-panorama.
The following ranges are available: 10 kHz, 25 kHz, 50 kHz, 100 kHz, 150 kHz, 256 kHz, 300 kHz, 400 kHz, 600 kHz, 800 kHz, 1200 kHz , 2400 kHz, 4800 kHz und 9600 kHz. The entered frequency must exactly correspond to the upper values. It is not rounded.

Note:
The IF-panorama is available if the software option EM510IM (ITU Measurement) or the software option EM510SU (Spectrum Unit) is installed. Otherwise the error message *1,"Device dependent error;`Not installed`"* is output.

Parameters:

`<numeric_value>`	frequency range
`UP|DOWN`	taking the range after|before the current bandwidth
`MINimum|MAXimum`	sets the minimum|maximum frequency range

**RST state:*
`100 kHz`

Example:
`FREQuency:SPAN 25 kHz`

. . :SPAN? [MINimum|MAXimum]

Query about the frequency range with option IF-panorama.

Parameters:

none	query about the current frequency range
`MINimum|MAXimum`	query about the minimum|maximum frequency range

Result:
Frequency value Hz

Example:
`FREQuency:SPAN? → 25000`

. . :STARt <numeric_value>|MINimum|MAXimum

Setting the start frequency of a frequency scan.

Parameters:

`<numeric_value>`	frequency
`MINimum|MAXimum`	sets the lowest|highest start frequency

**RST state:*
`1 MHz`

Example:
`FREQuency:STARt 1.5 MHz`

. . :STARt? [MINimum|MAXimum]

Query about the start frequency of a frequency scan.

Parameters:

none	query about the current start frequency
`MINimum|MAXimum`	query about the lowest|highest start frequency

Result:
Frequency in Hz

Example:
`FREQuency:STARt? -> 1500000`

. . :STEP
. . . [:INCRement] <numeric_value>|MINimum|MAXimum

Setting the stepwidth for the command `SENSe:FREQuency[:CW|FIXed] UP|DOWN.`

Parameters:

`<numeric_value>`	frequency stepwidth
`MINimum|MAXimum`	sets the smallest|largest stepwidth

**RST state:*
`1 kHz`

Example:
`FREQuency:STEP 2 kHz`

. . . [:INCRement]? [MINimum|MAXimum]

Query about the stepwidth.

Parameters:

none	query about the currently set stepwidth
`MINimum|MAXimum`	query about the smallest|largest stepwidth

Result:
Frequency stepwidth in Hz
Example:
FREQuency:STEP? -> 2000

. . :STOP <numeric_value>|MINimum|MAXimum

Setting the stop frequency of a frequency scan.

Parameters:

`<numeric_value>`	frequency
`MINimum|MAXimum`	sets the lowest|highest stop frequency

**RST state:*
`2 MHz`

Example:
`FREQuency:STOP 4 MHz`

. . :STOP? [MINimum|MAXimum]

Query about a stop frequency of a frequency scan.

Parameters:

none	query about the current stop frequency
`MINimum|MAXimum`	query about the lowest|highest stop frequency

Result:
Frequency in Hz

Example:
`FREQuency:STOP? -> 4000000`

.   :FUNCtion

If the sensor function(s) is (are) changed, the trace data set `MTRACE` is always deleted.

.   .   :CONCurrent <Boolean>

Determines whether several sensor functions can at the same time be switched or not. If CONCurrent = OFF, the command `SENSe:FUNCtion[:ON]` has the effect of a 1-out-of-n selection (one is switched on, the previously activated is automatically switched off). If CONCurrent = ON, the command `SENSe:FUNCtion[:ON]` switches the corresponding function on, while all the other functions remain unchanged. If CONCurrent is switched from ON to OFF, the function `"VOLTage:AC"` is switched on and all other functions are switched off.

Parameters:

`ON`	CONCurrent on
`OFF`	CONCurrent off

**RST state:*
`ON`

Example:
```
FUNCtion:CONCurrent ON
```

.   .   :CONCurrent?

Query about several sensor functions that, at the same time, can be switched or not.

Parameters:
none

Result:

CONCurrent switched on	`1`
CONCurrent switched off	`0`

Example:
```
FUNCtion:CONCurrent? -> 1
```

. . :OFF <sensor_function> {,<sensor_function>}

Switching off one or several sensor functions.

Parameters:
see `SENSe:FUNCtion[:ON]`

**RST state:*
`"FREQ:OFFS"`

Example:
`FUNCtion:OFF "FREQ:OFFS"`

. . :OFF?

Query about the sensor functions being switched off.

Parameters:
none

Result:
List of the sensor functions being switched off. For strings see `SENSe:FUNCtion[:ON]`

Example:
`FUNCtion:OFF? -> "FREQ:OFFS"`

. . . :COUNt?

Query about the number of sensor functions being inactive.

Parameters:
none

Result:
Number of sensor functions being inactive

Example:
`FUNCtion:OFF:COUNt? -> 2`

. . [:ON] <sensor_function> {,<sensor_function>}

Switch on of one or several sensor functions.

Error message:
If CONCurrent = OFF, an error -108, "Parameter not allowed" will be generated for ten or more parameters.

Parameters:
<sensor_function> is one of the following strings:

"VOLTage:AC"	switch on level measurement
"FREQuency:OFFSet"	switch on offset measurement

Note:
The offset measurement works only with signals whose signal energy may vary but must not disappear completely. At QAM or 100 % AM-modulated signals, the offset measurement cannot provide correct values.

Note:
With SW option EM510IM (ITU Measurement) installed, apart from level and offset the measurement functions AM modulation index , FM frequency deviation, PM phase deviation and band width measurement are also available.

"AM"	switch on AM modulation index measurement
"AM:POSitive"	switch on AM positive modulation index measurement
"AM:NEGative"	switch on AM negative modulation index measurement
"FM"	switch on FM frequency deviation measurement
"FM:POSitive"	switch on FM positive frequency deviation measurement
"FM:NEGative"	switch on FM negative frequency deviation measurement
"PM"	switch on PM phase deviation measurement
"BANDwidth"	switch on bandwidth measurement

**RST state:*
"VOLTage:AC"

Example:

```
FUNCtion "VOLT:AC", "FREQ:OFFS"
```

. . [:ON]?

Query about sensor functions being switched on.

Parameters:
none

Result:
List of sensor functions switched on. If no function is active, a zero string ("") is output. The list has a specific order:

1. level measurement function
2. offset measurement function
3. reserved
4. AM modulation index measurement function
5. AM positive modulation index measurement function
6. AM negative modulation index measurement function
7. FM frequency deviation measurement function
8. FM positive frequency deviation measurement function
9. FM negative frequency deviation measurement function
10. PM phase deviation measurement function
11. bandwidth measurement function

The following strings are to be expected:

`"VOLT:AC"`	level measurement switched on
`"FREQ:OFFS"`	offset measurement switched on
`"FSTR"`	field strength measurement switched on
`"AM"`	AM modulation index measurement switched on
`"AM:POS"`	AM positive modulation index measurement switched on
`"AM:NEG"`	AM negative modulation index measurement switched on
`"FM"`	FM frequency deviation measurement switched on
`"FM:POS"`	FM positive frequency deviation measurement switched on
`"FM:NEG"`	FM negative frequency deviation measurement switched on
`"PM"`	PM phase deviation measurement switched on
`"BAND"`	bandwidth measurement switched on

Example:

```
FUNCtion? -> "VOLT:AC", "FREQ:OFFS"
```

. . . :COUNt?

Query about the number of sensor functions being active.

Parameters:
none

Result:
Number of sensor functions being active

Example:

```
FUNCtion:Count? -> 2
```

. :GCONtrol

. . [:FIXed|MGC] <numeric_value>|UP|DOWN|MINimum|MAXimum

Setting the MGC value.

Parameters:

`<numeric_value>`	gain control factor in dBµV -30 dBµV = no gain control -> maximum sensitivity 110 dBµV = maximum gain control -> minimum sensitivity
`UP|DOWN`	increase|decrease of the MGC value by the value set in the command `SENSe:GCONtrol[:FIXed|MGC]:STEP[:INCRement]`.
`MINimum|MAXimum`	sets the smallest|largest MGC value

**RST state:*
`50 dBuV`

Example:
`GCONtrol 50`

. . [:FIXed|MGC]? [MINimum|MAXimum]

Query about the MGC value.

Parameters:

none	query about the current MGC value
`MINimum|MAXimum`	query about the smallest|largest MGC value

Result:
Gain control

Example:
`GCONtrol? -> 50`

. . :AUTO:TIME SLOW | DEFault | FAST
. . :AGC:TIME SLOW | DEFault | FAST

Setting the gain control time.

Parameters:

`SLOW`	slow gain control time
`DEFault`	default gain control time
`FAST`	fast gain control time

**RST state:*
`DEF`

Example:
`GCONtrol:AUTO:TIME FAST`

. . :AUTO:TIME?
. . :AGC:TIME?

Query about the gain control time.

Result:

`SLOW`	slow gain control time
`DEF`	default gain control time
`FAST`	fast gain control time

Example:
```
GCONtrol:AUTO:TIME? -> FAST
```

. . . :STEP
. . . . [:INCRement] <numeric_value>|MINimum|MAXimum

Setting the stepwidth for the command `SENSe:GCONtrol[:FIXed|MGC] UP|DOWN.`

Parameters:

`<numeric_value>`	MGC stepwidth
`MINimum|MAXimum`	smallest|largest stepwidth

**RST state:*
1 dB

Example:
```
GCONtrol:STEP 10
```

.   .   .   .   [:INCRement]? [MINimum|MAXimum]

Query about the MGC stepwidth.

Parameters:

none	query about the currently set stepwidth
`MINimum|MAXimum`	query about the smallest|largest stepwidth

Result:
MGC stepwidth in dB

Example:
```
GCONtrol:STEP? -> 10
```

.   .   :MODE FIXed|MGC|AUTO|AGC

Type of gain control.

Parameters:

`FIXed|MGC`	control is determined by MGC value
`AUTO|AGC`	control is automatically generated (AGC)

**RST state:*
```
AUTO
```

Example:
```
GCONtrol:MODE AUTO
```

.   .   :MODE?

Query about the type of gain control.

Parameters:
none

Result:

`FIX, AUTO`	see above

Example:
```
GCONtrol:MODE? -> AUTO
```

. :MSCan

The `MSCan` system controls the frequency function of the device, provided the memory scan has been activated by `SENSe:FREQuency:MODE MSCan`. Each scan is started by `INITiate[:IMMediate]`. The memory locations are placed in the `MEMory` subsystem and are set for query during the scan.

. . :CHANnel <name>

Setting the current memory location.
The memory locations are addressed by the text (see "Parameters" on page 5.9) `MEM0` to `MEM9999` (memory location 0 to memory location 9999) and the next free memory location by `ACTUAL` .
During the memory scan, this command is not permitted.

Parameters:

`<name>`	`MEM0` to `MEM9999 | NEXT`

**RST state:*
`MEM0`

Example:
`MSCan:CHANnel MEM357`

. . :CHANnel?

Output of current memory location.

Parameters:
none

Result:
Number of current memory location.

Example:
`MSCan:CHANnel?` → `357`

. . :CONTrol
. . . :[ON]<control_function> {,<control_function>}

Command for switch-on of the `'STOP:SIGNal'` functions.

With `'STOP:SIGNal'` the disappearance of the signal during the dwell time causes the dwell time to be aborted. The hold time after the disappearance of the signal is set with `SENSe:MSCan:HOLD:TIME`.

Parameters:
`<control_function>` is the following string:

`'STOP:SIGNal'`	switches the signal-controlled dwell time on

**RST state*
After *RST there is no control function running

Example:
`MSCan:CONTrol 'STOP:SIGN'`

. . . [:ON]?

Query about the scan-control mechanism that is switched on.

Parameters:
none

Result:
A list of the scan-control mechanisms that are switched on is output. If there is nothing switched on then a zero string ("") is output.

Following strings can be expected:

`" "`	no mechanism switched on
`"STOP:SIGN"`	signal-controlled dwell time is switched on

Example:
```
MSCan:CONTrol?→"STOP:SIGN"
```

. . . :OFF <control_function> {,<control_function}

Switches off one or more scan-control mechanisms.

Parameters:
see `SENSe:MSCan:CONTrol [:ON]`

**RST state*
`after *RST there is no control function enabled`

Example:
```
MSCan:CONTrol:OFF "STOP:SIGN"
```

. . . :OFF?

Query about the scan-control mechanisms that are switched off.

Parameters:
none

Result:
A list of the scan-control mechanisms that are switched off is output. For strings see `SENSe:MSCan:CONTrol[:ON]`

Example:
```
MSCan:CONTrol:OFF?→"STOP:SIGN"
```

. . :COUNt <numeric_value>|MINimum|MAXimum|INFinity

Information about the number of MSCans.

Parameters:

`<numeric_value>`	number of scans
`MINimum|MAXimum`	minimum|maximum number
`INFinity`	infinite number

**RST state:*
INFinity

Example:

```
MSCan:COUNt 100
```

. . :COUNt? [MINimum|MAXimum]

Query about the number of MSCans.

Parameters:

none	query about the current number of scans
`MINimum|MAXimum`	query about the minimum|maximum number of scans

Result:
Number of scans; 9.9E37 is output for an infinite number

Example:

```
MSCan:COUNt? -> 100
```

. . :DIRection UP|DOWN

Sets scan direction

Parameters:

`UP`	scans in direction of descending memory numbers
`DOWN`	scans in direction of ascending memory numbers

**RST state*

```
UP
```

Example:

```
MSCan:DIRection DOWN
```

.   .   :DIRection?

Query about the scan direction.

Parameters:
none

Result:
```
UP, DOWN
```

Example:
```
MSCan:DIRection?→DOWN
```

.   .   :DWELl <numeric_value>|MINimum|MAXimum|INFinity

Setting the dwell time with the hold criterion fulfilled (T-DWELL in MSCAN-CONFIG menu).

Note:
According to the SCPI standard, this command is used to set the dwell time per scan step, i.e. the time required by a step. This definition is met in the EM510 if the squelch is switched off. The hold criterion is then fulfilled for each step.

Parameters:

`<numeric_value>`	dwell time in seconds
`MINimum|MAXimum`	lowest|highest dwell time
`INFinity`	infinite dwell time

**RST state:*
0.5 s

Example:
```
SWEep:DWELl 10 ms
```

.   .   :DWELl? [MINimum|MAXimum]

Query about the dwell time.

Parameters:

none	query about the current dwell time
`MINimum|MAXimum`	query about the lowest|highest dwell time

Result:
Dwell time in seconds; 9.9E37 is output for an infinite number

Example:
```
MSCan:DWELl? 0.010
```

. . :HOLD
. . . :TIME <numeric_value>|MINimum|MAXimum

Setting the hold time during signal-controlled scan continuation. If the signal disappears during the dwell time, the hold time is started. After completion of the hold time, the scan is continued with the next frequency even if the dwell time has not yet been completed. If the signal during the hold time exceeds the squelch threshold, the hold time is reset and the end of the dwell time or the renewed disappearance of the signal is awaited. The hold time is only important if the control function `"STOP:SIGNal"` (see `SENSe:MSCan:CONTrol`) is switched on.

Parameters:

`<numeric_value>`	hold time in seconds
`MINimum|MAXimum`	lowest|highest hold time

**RST state:*
`0 s`

Example:
`SWEep:HOLD:TIME 10 ms`

. . . :TIME? [MINimum|MAXimum]

Query about the hold time.

Parameters:

none	query about the current hold time
`MINimum|MAXimum`	query about the lowest|highest hold time

Result:
Hold time in seconds

Example:
`MSCan:HOLD:TIME? 0.010`

. :PSCan

The PSCan system checks the frequency function of the unit in case a panorama scan was initiated by SENSe:FREQuency:MODE. Each scan is started only by INITiate[:IMMediate].

. . :COUNt <numeric_value>|MINimum|MAXimum|INFinity

Output of a number of PSCan cycles.

Parameters:

<numeric_value>	number of cycles
MINimum|MAXimum	minimal/maximal number
INFinity	infinite number

**RST state:*
INFinity

Example:
PSCan:COUNt 100

. . :STEP<numeric_value>|MINimum|MAXimum| UP|DOWN

Setting of the channel raster and the resolution bandwidth correspondingly.

Parameters:

<numeric_value>	the following discrete values can be set: 125 Hz, 250 Hz, 500 Hz, 625 Hz, 1.25 kHz, 2.5 kHz, 3.125 kHz, 6.25 kHz, 12.5 kHz, 25kHz, 50 kHz, 100kHz
MINimum|MAXimum	setting of the smallest/widest resolution bandwidth
UP | DOWN	setting of the next smallest/widest resolution bandwidth

**RST state:*
100 kHz

Example:
PSCan:STEP 25 kHz

. . . ? [MINimum|MAXimum]

Query about the resolution bandwidth at the panorama scan.

Parameters:

none	query about the current resolution bandwidth
MINimum|MAXimum	query about the smallest/widest resolution bandwidth

Result:
Resolution bandwidth in Hz

Example:
PSCan:STEP? → 25000

. :ROSCillator

Control of reference oscillator.

. . :EXTernal
. . . :FREQuency?

Query about the given external reference frequency.

Parameters:
none

Result:
10000000

Example:
```
ROSCillator:EXTernal:FREQuency? -> 10000000
```

. . [:INTernal]
. . . :FREQuency?

Query about the internal reference frequency.

Parameters:
none

Result:
10000000

Example:
```
ROSCillator:FREQuency? -> 10000000
```

. . :SOURce INTernal|EXTernal

Setting whether external or internal reference frequency is to be used.

Parameters:

INTernal	internal reference oscillator
EXTernal	external reference oscillator

**RST state:*
INTernal

Example:
```
ROSCillator:SOURce EXTernal
```

.   .   :SOURce?

Query about the reference oscillator to be used.

Parameters:
none

Result:	
INT	internal reference oscillator
EXT	external reference oscillator

Example:
```
ROSCillator:SOURce? -> EXT
```

. :SWEep

The `SWEep` system controls the frequency function of the device if the frequency scan has been activated by the `SENSe:FREQuency:MODE SWEep` command. Each scan is initiated by `INITiate[:IMMediate]`.

. . :CONTrol
. . . :[ON] <control _function> {,<control_function>}

Command for switch-on of the `STOP:SIGNal`functions.
With "STOP:SIGNal" the disappearance of the signal during the dwell time causes the dwell time to be aborted. The hold time after the disappearance of the signal is set with `SENSe:SWEep:HOLD:TIME`.

Parameters:
`<control_function>` is one of the following strings:

`"STOP:SIGNal"`	switch-on signal-controlled scan continuation

**RST state:*
After *RST no control function is switched on.

Example:
```
SWEep:CONTrol "STOP:SIGN"
```

. . . [:ON]?

Query about the switched-on scan-control functions.

Parameters:
none

Result:
List of switched-off control functions. If no function is active, a zero string ("") is output.

The following strings are to be expected:

`"STOP:SIGN"`	signal-controlled scan continuation switched on
`""`	zero string : none of the two functions is active

Example:
```
SWEep:CONTrol? -> "STOP:SIGN"
```

. . . :OFF <control_function>{,<control_function>}

Switch-off of one or several scan-control functions.

Parameters:
see `SENSe:SWEEp:CONTrol  [:ON]`

**RST state:*
`after *RST there is no control function running`

Example:
`SWEep:CONTrol:OFF "STOP:SIGN"`

. . . :OFF ?

Query about the switched-off scan-control functions.

Parameters:
none

Result:
List of switched-off control functions. Strings see `SENSe:SWEep:CONTrol  [:ON]`

Example:
`SWEep:CONTrol:OFF? ->"STOP:SIGN"`

. . :COUNt <numeric_value>|MINimum|MAXimum|INFinity

Information about the number of sweeps.

Parameters:

`<numeric_value>`	number of sweeps
`MINimum|MAXimum`	minimum|maximum number
`INFinity`	infinite number

**RST state:*
INFinity

Example:
`SWEep:COUNt 100`

. . :COUNt? [MINimum|MAXimum]

Query about the number of sweeps.

Parameters:

none	query about the current number of sweeps
`MINimum|MAXimum`	query about the minimum|maximum sweeps

Result:
Number of sweeps; 9.9E37 is output for an infinite number

Example:
```
SWEep:COUNt? -> 100
```

. . :DIRection UP|DOWN

Setting the scan direction.

Parameters:

`UP`	scan with increasing frequency
`DOWN`	scan with decreasing frequency

**RST state:*
```
UP
```

Example:
```
SWEep:DIRection DOWN
```

. . :DIRection?

Query about the scan direction.

Parameters:
none

Result:
```
UP, DOWN
```

Example:
```
SWEep:DIRection? -> DOWN
```

. . :DWELl <numeric_value>|MINimum|MAXimum|INFinity

Setting the dwell time with the hold criterion fulfilled (T-DWELL in FSCAN-CONFIG menu).

Note:
According to the SCPI standard, this command is used to set the dwell time per scan step, i.e. the time required by a step. This definition is met in the EM510 if the squelch is switched off. The hold criterion is then fulfilled for each step.

Parameters:

`<numeric_value>`	dwell time in seconds
`MINimum|MAXimum`	lowest|highest dwell time
`INFinity`	infinite dwell time

**RST state:*
0.5 s

Example:
`SWEep:DWELl 10 ms`

. . :DWELl? [MINimum|MAXimum]

Query about the dwell time with hold criterion fulfilled.

Parameters:

none	query about the current dwell time
MINimum|MAXimum	query about the lowest|highest dwell time

Result:
Dwell time in seconds; ; 9.9E37 is output for an infinite number.

Example:
`SWEep:DWELl? 0.010`

. . :HOLD
. . . :TIME <numeric_value>|MINimum|MAXimum

Setting the hold time during signal-controlled scan continuation (T_NOSIG in FSCAN-CONFIG menu). If the signal disappears during the dwell time, the hold time is started. After completion of the hold time, the scan is continued with the next frequency even if the dwell time has not yet been completed. If the signal during the hold time exceeds the squelch threshold, the hold time is reset and the end of the dwell time or the renewed disappearance of the signal is awaited. The hold time is only important if the control function `"STOP:SIGNal"` (see `SENSe:SWEep:CONTrol`) is switched on.

Parameters:

`<numeric_value>`	hold time in seconds
`MINimum|MAXimum`	lowest|highest hold time

**RST state:*
0 s

Example:
`SWEep:HOLD:TIME 10 ms`

. . . :TIME? [MINimum|MAXimum]

Query about the hold time during signal-controlled scan continuation.

Parameters:

none	query about the current hold time
`MINimum|MAXimum`	query about the lowest|highest hold time

Result:
Hold time in seconds

Example:
```
SWEep:HOLD:TIME? 0.010
```

. . :STEP<numeric_value>|MINimum|MAXimum

Setting the frequency stepwidth for the frequency scan.

Parameters:

<numeric_value>	frequency value
`MINimum|MAXimum`	sets the smallest/biggest frequency stepwidth

**RST state:*
```
1 kHz
```

Example:
```
SWEep:STEP 25 kHz
```

. . :STEP? [MINimum|MAXimum]

Query about the frequency stepwidth of a frequency scan

Parameters:

none	query about the current frequency stepwidth
`MINimum|MAXimum`	query about the smallest|largest frequency stepwidth

Result:
Stepwidth in Hz

Example:
```
SWEep:STEP? -> 25000
```

. . :SUPPress

Insert current frequency into suppress list. The range is obtained from the bandwidth according to the following formulae:

```
    SSTARTn = SENSn: FREQ - SENSn:BAND/2
    SSTOPn  = SENSn: FREQ + SENSn:BAND/2
```

The frequency pair is inserted into an empty space of the trace. Free spaces (gaps) are characterized by a frequency pair with the values 0.0.

Error message:
If the corresponding suppress trace has no free space, an error `-223 "Too much data"` is generated.

Parameters:
none

**RST state:*
none, as command is an event

Example:
`SWEep:SUPPress`

. . . :SORT

Sort and condense suppress list. The frequency pairs are sorted to an ascending order of the start frequency. Overlapping is eliminated by extending the frequency pair. The other frequency pair is then deleted. Gaps within the suppress list are put to the end of the list.

Parameters:
`none`

**RST state:*
none, as command is an event

Example:
`SWEep:SUPPress:SORT`

5.3.16 STATus Subsystem

The following STATus register commands are possible according to SCPI standard:

```
STATus
.   :OPERation
.   .   :CONDition?
.   .   :ENABle <numeric_value>
.   .   :ENABle?
.   .   [:EVENt]?
.   .   :NTRansition <numeric_value>
.   .   :NTRansition?
.   .   :PTRansition <numeric_value>
.   .   :PTRansition?
.   .   :SWEeping
.   .   .   :CONDition?
.   .   .   :ENABle <numeric_value>
.   .   .   :ENABle?
.   .   .   [:EVENt]?
.   .   .   :NTRansition <numeric_value>
.   .   .   :NTRansition?
.   .   .   :PTRansition <numeric_value>
.   .   .   :PTRansition?

.   :QUEStionable
.   .   :CONDition?
.   .   :ENABle <numeric_value>
.   .   :ENABle?
.   .   [:EVENt]?
.   .   :NTRansition <numeric_value>
.   .   :NTRansition?
.   .   :PTRansition <numeric_value>
.   .   :PTRansition?
```

extended STATus register commands:

```
STATus
.   :EXTension
.   .   :CONDition?
.   .   :ENABle <numeric_value>
.   .   :ENABle?
.   .   [:EVENt]?
.   .   :NTRansition <numeric_value>
.   .   :NTRansition?
.   .   :PTRansition <numeric_value>
.   .   :PTRansition?

STATus
.   :TRACe
.   .   :CONDition?
.   .   :ENABle <numeric_value>
.   .   :ENABle?
.   .   [:EVENt]?
.   .   :NTRansition <numeric_value>
.   .   :NTRansition?
.   .   :PTRansition <numeric_value>
.   .   :PTRansition?
```

Exemplifying all STATus register commands, for example the STATus:OPERation:SWEeping register, the commands of the STATus:OPERation register are explained as follows.

STATus
. OPERation
. . :CONDition?

Query about the condition section of the OPERation status register.

Parameters:
none

Result:
Depending on the setting by the `FORMat:SREGister` command, the contents of the register are transferred as a decimal, binary or hexadecimal value in the ASCII code.

Example:
```
STATus:OPERation:CONDition? -> #H0008
```

. . :ENABle <numeric_value>

Setting the enable section of the OPERation status register.

Parameters:

`<numeric_value>`	value of the enable section (`0..65535 or #H0000..#HFFFF or #B0..#B1111111111111111`)

**RST state:*
will not be changed by *RST

Example:
```
STATus:OPERation:ENABle #H0008
```

. . :ENABle?

Query about the enable section of the OPERation status register.

Parameters:
none

Result:
Depending on the setting by the `FORMat:SREGister` command, the contents of the register are transferred decimally, binary or hexadecimally in the ASCII code.

Example:
```
STATus:OPERation:ENABle? -> #H0008
```

. . [:EVENt]?

Query about the event section of the OPERation status register.

Parameters:
none

Result:
Depending on the setting by the `FORMat:SREGister` command, the contents of the register are transferred as a decimal, binary or hexadecimal value in ASCII code.

Example:
```
STATus:OPERation? -> #H0008
```

. . :NTRansition <numeric_value>

Setting the negative transition filter of the OPERation status register.

Parameters:

`<numeric_value>`	value of the NTRansition section (`0..65535 or #H0000..#HFFFF or #B0..#B1111111111111111`)

**RST state:*
will not be changed by *RST

Example:
```
STATus:OPERation:NTRansition #H0000
```

. . :NTRansition?

Query about the negative transition filter of the OPERation status register.

Parameters:
none

Result:
Depending on the setting by the `FORMat:SREGister` command, the contents of the register are transferred as a decimal, binary or hexadecimal value in ASCII code.

Example:
```
STATus:OPERation:NTRansition? -> 0
```

. . :PTRansition <numeric_value>

Setting the positive transition filter of the OPERation status register.

Parameters:

`<numeric_value>`	value of the PTRansition section (`0..65535 or #H0000..#HFFFF or #B0..#B1111111111111111`)

RST state:
will not be changed by *RST

Example:

```
STATus:OPERation:PTRansition #B11111111
```

. . :PTRansition?

Query about the positive transition filter of the OPERation status register.

Parameters:
none

Result:
Depending on the setting by the `FORMat:SREGister` command, the contents of the register are transferred as a decimal, binary or hexadecimal value in ASCII code.

Example:

```
STATus:OPERation:PTRansition? -> 255
```

Other STATus commands described below do not directly influence the STATus registers.

. :PRESet

Setting the STATus registers with default values:

Register	ENABle/PTR/NTR	PRESet value
STATus:OPERational	ENABle	0
	PTR	65535
	NTR	0
STATus:QUEStionable	ENABle	0
	PTR	65535
	NTR	0
STATus:TRACe	ENABle	65535
	PTR	65535
	NTR	0
STATus:EXTension	ENABle	65535
	PTR	65535
	NTR	0
STATus:OPERation:SWEep	ENABle	65535
	PTR	65535
	NTR	0

Parameters:
none

**RST state:*
none, as command is an event

Example:
```
STATus:PRESet
```

. :QUEue?
. . [:NEXT]?

Reads the next entry from the Error Queue.

Parameters:
none

Result:
Next entry of Error Queue

Example:
```
STATus:QUEue? -> 0, "No Error"
```

5.3.17 SYSTem Subsystem

SYSTem:AUDIo:BALance <numeric_value> MINimum|MAXimum

Sets the balance of AF for the headphones.

Parameters:

`<numeric_value>`	balance of AF from -0.5 to +0.5	
	-0.5	= only left channel
	0	= mid position
	0.5	= only right channel
`MINimum|MAXimum`	only left AF channel | only right AF channel	

Note:
The parameter is rounded to the next internally settable discrete value.

**RST state:*
0

Example:
```
SYSTem:AUDio:BALAnce 0.5
```

### .	:AUDio:BALance? MINimum|MAXimum

Query about the AF balance.

Parameters:
none

`MINimum|MAXimum`	min. | max. value

Example:
```
SYSTem:AUDio:BALance? -> 0.5
```

. :AUDio:REMote:MODE <numeric_value>

Sets the mode of the digital AF that is transferred via the remote control interface per UDP. See also Annex D: LAN Configuration.

Parameter:
`<numeric_value>` Mode 0 to 12 of digital AF

Mode	Sampling rate [kHz]	Bits pro sample	Channels	Data rate [kbyte/s]	Length per frame [bytes]
0	-	-	-	0	
1	32	16	2	128	1
2	32	16	1	64	2
3	32	8	2	64	2
4	32	8	1	32	1
5	16	16	2	64	4
6	16	16	1	32	2
7	16	8	2	32	2
8	16	8	1	16	1
9	8	16	2	32	4
10	8	16	1	16	2
11	8	8	2	16	2
12	8	8	1	8	1

**RST status:*
`0`

Example:

```
SYSTem:AUDio:REMote:MODe 5
```

. :AUDio:REMote:MODe?

Query about the set digital AF mode.

Parameter:
None

Example:

```
SYSTem:AUDio:REMote:MODe? -> 5
```

. :FPDP:REMote:MODE OFF | SHORt | LONG | FLAGs | AMMos

Sets the format of the digital data (panorama, IF or demodulated data) that is output by FPDP. If command `OUTPut:FPDP:MODE` has been used to set the format to `DEModulator` (demodulated data), the demodulation modes AM, FM, PULS and PM will result in an output of AM instead of I, and of FM instead of Q.

Parameters:

`OFF`	digital data switched off
`SHORt`	digital data in the format 16 Bit I and 16 Bit Q
`LONG`	digital data in the format 32 Bit I and 32 Bit Q
`FLAGs`	digital data in the format 24 Bit I and 24 Bit Q with status flags
`AMMos`	digital data in the AMMOS format 16 Bit I and 16 Bit Q

In **`SHORt`** mode, raw data is output. The format within a 32-bit word is 16 bits I and 16 bits Q. The signed fractional imaginary part Q is contained in the upper 16-bit word, and the signed fractional real part I in the lower 16-bit word of the 32-bit word. Due to this combined format, there are no problems with synchronization.

In **`LONG`** mode, raw data is output. The format is 31 bits I and 31 bits Q.
The LSB (least significant bit) is used to identify the real part I (LSB = 1) or the imaginary part Q (LSB = 0).

In **`FLAGs`** mode, a 32-bit word contains the following additional information as flags besides the complex samples (24 bits I and 24 bits Q).

REAL PART:

<31:8>	real part 24 bits signed fractional				
<7>	always 1 (I-Strobe)				
<6>	SIGVALID				
<5>	blanking flag	<4>	reserved	<3:0>	running four-bit counter

IMAGINARY PART:

<31:8>	imaginary part 24 bits signed fractional
<7>	always 0 (Q-Strobe)
<6:0>	RxAtt (attenuation to antenna in dB – 50 dB)

Parameter RxAtt may be used to calculate the level at the antenna input.

level_antenna = level_IQ_RMS + RxAtt + 50 dB

In `AMMos` mode, further information such as the current receive frequency, recording bandwidth, parameter change flag, level correction value and timestamp is included in the data stream in addition to the IQ data.

The full data format is specified in "Annex B: Data Stream Format".

**RST state:*
`SHOR`

Example:
```
SYSTem:FPDP:REMote:MODe SHORT
```

. : FPDP:REMote:MODE?

Query about the set data format.

Parameters:
none

Result:

`OFF`	digital data switched off
`SHOR`	digital data in the format 16 Bit I and 16 Bit Q
`LONG`	digital data in the format 32 Bit I and 32 Bit Q
`FLAG`	digital data in the format 24 Bit I and 24 Bit Q with status flags
`AMM`	digital data in the AMMOS format 16 Bit I and 16 Bit Q

Example:

```
SYSTem:FPDP:REMote:MODE? -> SHOR
```

. :IF:REMote:MODE OFF | SHORt | LONG

Sets the mode of the digital IF that is transferred via the remote control interface per UDP.

Parameters:

`OFF`	digital IF switched off
`SHORt`	digital IF format 16 bit I and 16 bit Q
`LONG`	digital IF format 32 bit I and 32 bit Q

**RST state:*
`OFF`

Example:
```
SYSTem:IF:REMote:MODe SHORT
```

. :IF:REMote:MODE?

Query about the set digital IF mode.

Parameters:
none

Result:

`OFF`	digital IF switched off
`SHORt`	digital IF format 16 bit I and 16 bit Q
`LONG`	digital IF format 32 bit I and 32 bit Q

Example:
```
SYSTem:IF:REMote:MODE? -> SHORT
```

. :VIDeo:REMote:MODE OFF | SHORT | LONG

Sets the mode of the digital video output that is transferred via the remote control interface per UDP.
The output depends on which demodulator has been switched on.
See also command `DISPlay:MENU.`

Parameters:

`OFF`	digital video output switched off
`SHORT`	digital video output format 16 bit (I or AM) and 16 bit (Q or FM)
`LONG`	digital video output format 32 bit (I or AM) and 32 bit (Q or FM)

**RST state:*
`OFF`

Example:
```
SYSTem:VIDeo:REMote:MODe SHORT
```

. :VIDeo:REMote:MODE?

Query about the mode settings of the digital video output.

Parameters:
none

Result:

`OFF`	digital video output switched off
`SHORT`	digital video output format 16 bit (I or AM) and 16 bit (Q or FM)
`LONG`	digital video output format 32 bit (I or AM) and 32 bit (Q or FM)

Example:
```
SYSTem:IF:REMote:MODE? -> SHORT
```

. :AUDio:VOLume <numeric_value>|MINimum|MAXimum

Sets the volume of AF for loudspeakers and headphones.

Parameters:

`<numeric_value>`	volume of AF from 0 to 1	
	0	= no AF
	1	= full volume of AF
`MINimum|MAXimum`	no AF | full volume of AF	

Note:
The parameter is rounded to the next internally settable discrete value.

**RST state:*
0.2

Example:
```
SYSTem:AUDio:VOLume 0.5
```

. :AUDio:VOLume? [MINimum|MAXimum]

Query about the AF volume.

Parameters:

none	
`MINimum|MAXimum`	min. | max. volume

Example:
```
SYSTem:AUDio:VOLume? -> 0.5
```

. :SECurity
. . :OPTion <name>

A special optional firmware (e.g. PSCAN) can be activated by entering a certain option code. The unit must be switched on anew to activate this software option. When ordering a software option the serial number of the EM510 (for example 100217/002) must be indicated. This number is on the sticking-label at the front panel or in the ident string (command: *idn?).

Parameters:

`<name>`	8-digit number input

**RST state*
none

Example:
```
SYSTem:SECurity:OPTion "12345678"
```

. :VERSion?

Query about the SCPI standard used by the device.

Parameters:
none

Result:
Version in format YYYY.V, where YYYY stands for the corresponding version year and V for the corresponding revision number of this year.

Example:
```
SYSTem:VERSion? -> 1996.0
```

5.3.18 TEST Subsystem

The selftest can be run with two different test routines. The basic test runs continuously in the background and tests the test points inside the module. Based on this test, a "short test" or a "long test" can be triggered. In the short test, a line spectrum is fed in at the antenna input and the receiver is set to the line frequency nearest to the receive frequency. The complete receive path from the antenna input of the tuner to the level evaluation is then measured and evaluated. In the long test, each line frequency of the test spectrum is set and measured.

TEST? SHORt|LONG, REPort|QUIet

Triggering the "short test" or "long test".

Parameters:

`SHORt|LONG`	carry out short test | long test
`REPort|QUIet`	error messages in plain text are generated | not generated

Note:
If the test was initiated by `REPort` , plain text error reports are stored in the error queue. These reports can be queried by SYStem:ERRor?.

Result:

0	no error detected
≠ 0	error detected :

Example:

```
TEST? LONG, QUIET -> 1
```

5.3.19 TRACe Subsystem

Traces are used for summarizing data. The following traces are available:

Result trace:

For the results, two predefined traces (MTRACE = Measurement Trace and ITRACE = Information Trace)are available. They cannot be deleted.

Via the control instruction, a condition can be defined which can preselect the data to be written into the MTRACE or ITRACE. If the control conditions of the two traces are identical, each TRACE value has a corresponding information value in the ITRACE. When the maximum data set length is attained, MTRACE and ITRACE are closed down. Any subsequent data are thus lost.

MTRACE receives its data from the SENSe:FUNCtion block. All sensor functions switched on deliver their measured values to the MTRACE where they are stored.

ITRACE receives its data from the SENSe:FREQuency block. In addition to the current receiver frequency the corresponding channel number is also stored.

The start command to initiate measurement (INITiate[:IMMediate]) clears the MTRACE (or ITRACE) data set.

IF-panorama Trace IFPAN

If the software option EM510SU (IF panorama) is installed, spectrum data can be queried via Trace IFPAN.

The command

```
TRACe:FEED:CONTrol IFPAN, ALWays
```

starts loading of the IFPAN Trace. The command

```
DISPlay:MENU IFPAN
```

starts the IF-panorama.

The data will be output in a raw form, i.e. like calculated by the DSP. The spectrum length is always 2047 points, regardless of the bandwidth chosen and the IF-panorama bandwidth. If data is available in the IFPAN trace the number of points can be queried by the command

```
TRACe:POINts? IFPAN
```

Suppress trace:

Remote sees the suppress lists as predefined traces. Each data set contains two traces with the names SSTART (= Suppress START) and SSTOP (=Suppress STOP).

The suppress list has 100 elements with each element consisting of two frequencies. The frequency pair specifies a frequency range which is suppressed during the scan. It is irrelevant that the 1st frequency is lower than the 2nd frequency. The sequence in the list is irrelevant, too. Gaps are specified by the frequency pair 0.0. If one frequency of the frequency pair is 0, the other frequency of the pair is seen as a single frequency.

Examples:

1st Frequency	2nd Frequency	Description
118000	136000	Suppression of range 118 to 136 kHz
98550	98450	Suppression of range 98.450 to 98.550 kHz
0	0	Empty frequency pair (irrelevant)
118375	0	Suppression of frequency 118.375 kHz
0	123400	Suppression of frequency 123.400 kHz
127675	127675	Suppression of frequency 127.675 kHz

In the status reporting system the states of the traces are coded in status bits (see "Status Reporting System" on page 5.128).

TRACe? SSTART|SSTOP

This query causes the data to be taken from the already corrected table.

1st frequency	2nd frequency
118000	136000
98450	98550
0	0
118375	118375
123400	123400
127675	127675

Clearing the suppress lists must always include both commands (TRAC SSTART, 0; TRAC SSTOP, 0).

TRACe|DATA

Note:
Instead of command word `TRACe, DATA` may be used as well.

. :CATalog?

Query about all defined trace names.

Note:
`IFPAN` trace is only available with the software option EM510IM (ITU Measurement) or software option EM510SU (Spectrum Unit) installed.

Parameters:
none

Result:
`"MTRACE", "ITRACE", "IFPAN", "SSTART", "SSTOP", "UDP"`

TRACe
. [:DATA] <trace_name>, <numeric_value> {, <numeric_value>} | <block>

Writing data to a trace.

Note:
Only the suppress traces can be written to.

Error messages:
If the trace name is unknown or not identical with a suppress trace, error -141 "Invalid character data" is generated.
If too many data are loaded in a suppress trace, error -223 "Too much data" is generated.

Parameter:

`<trace-name>`	name of the trace to be written to as `<Character Data>` SSTART
`<numeric_value>`	list of frequencies. If the list is not complete, the rest of the trace is filled with 0. ***Note:*** *In contrast to the SCPI standard a single value is not used for the complete trace!*
`<block>`	As an alternative to the frequency list a `<Definite Length Block>` can be transmitted with the following structure: Frequency list with frequencies in Hz, 4 bytes per frequency.

***RST state:*
No change of trace contents at *RST.

Example:
```
TRACe SSTART, 123.475 kHz, 118000, 98.550 kHz
```

. [:DATA] <trace_name>

Query about the trace data.
Error message:
If the trace name is unknown, an error `-141,"Invalid character data"` will be generated.

Parameters:

`<trace_name>`	name of desired trace as `<Character Data>` (`MTRACE, ITRACE, IFPAN` or `SSTART, SSTOP`)

Result:
The following is valid for `MTRACE`:
Output of measured values of all sensor functions switched on. If no function is switched on, `NaN` (Not a Number) is output.
If `FREQ:OFFS` is switched on, only the offset value is output. If `VOLT:AC` is switched on, only the level value is output. If `FREQ:OFFS` and `VOLT:AC` are switched on, first the level value and then the offset value are output.
The INF value 9.9E37 is entered into the result buffers MTRACE and ITRACE for MSCAN, FSCAN or PSCAN to identify the range limit.
If measurements can't be carried out due to current settings (e.g. offset measurement with I/Q, CW, ISB or SSB), it is indicated by the NINF value –9.9E37.

The following is valid for `ITRACE`:
Output of channel number and receive frequency.

Note:
With SW option EM510IM (ITU Measurement) installed, apart from level and offset the measurement functions AM modulation index , FM frequency deviation, PM phase deviation and band width measurement are also available.

Depending on the setting, the output format is defined by the `FORMat:DATA` command:
`ASCii` -> normal ASCII output:

- level in dBuV
- offset in Hz signed
- AM modulation index in %
- AM positive modulation index in %
- AM negative modulation index in %
- FM frequency deviation in Hz
- positive frequency deviation in Hz
- negative frequency deviation in Hz
- phase deviation in rad
- bandwidth in Hz
- channel number without unit
- frequency in Hz

`PACKed` -> `<Definite Length Block>`: (see "**Block data**" on page 5.11)

- level in 1/10 dBuV (2 bytes)
- offset in Hz (4 Byte)AM - modulation index in 1/10 % (2 bytes)
- AM - positive modulation index in 1/10 % (2 bytes)
- AM - negative modulation index in 1/10 % (2 bytes)
- FM frequency deviation in Hz (4 bytes)
- positive frequency deviation in Hz (4 bytes)
- negative frequency deviation in Hz (4 bytes)
- phase deviation in 1/100 rad (2 bytes)
- bandwidth in Hz (4 bytes)
- channel number (2 Byte)
- frequency in Hz (4 Byte)

Notes:

- *INF (end-of-range code) will be coded in the PACKed format as follows:*
  INF level = 2000
  INF offset = 10000000
  INF freq = 0
  INF channel = 0

- *NINF (measurement not possible) will be coded in the PACKed format as follows:*
  NINF Offset = 10000000-1
  The level can always be measured.

- *NaN is output as #110 in the `PACKed format`*

- *To ensure that for the two traces the same number of points is output, the two queries have to be one directly behind the other in the same command line (e.g. `TRACE? MTRACE;TRACE? ITRACE`).*

Example:

```
TRACe? MTRACE -> 23.4, -2500, 18.5, 1500
```

The following is valid for the IFPAN trace:

Output of the spectrum data. If there are no data available then a `NaN` (Not a Number) will be output.

The output format of the IFPAN trace depends on the settings made by command `FORMat:DATA`.

`ASCii ->` normal ASCII output:

- level in dBµV

`PACKed -> <Definite Length Block>:`

- level in 1/10 dBµV (2 bytes)

The following is valid for the suppress trace:

List of frequencies contained in the trace. The suppress-trace output format is defined according to the relevant setting through command `FORMat:DATA:`

`ASCii` ->Normal ASCII output:

- list of frequencies in Hz

`PACKed-><Definite Length Block>`

- list of frequencies in Hz, 4 bytes per frequency

Example:

```
TRACE? SSTART -> 123475, 118000, 98550
```

. :FEED? <trace_name>

Query about the data block connected with the trace.
Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`

Result:
Name of the block coupled to the trace.
The following is valid for `MTRACE`: `"SENS"`
The following is valid for `ITRACE`: `"FREQ"`
The following is valid for `IFPAN`: `"SENS"`

Example:
```
TRACe:FEED? MTRACE -> "SENS"
```

. . :CONTrol <trace_name>, ALWays|SQUelch|NEVer

Control of trace loading.
Error message:
If trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace-name>`	see `TRACe[:DATA]?`
`ALWays`	all data are stored
`SQUelch`	data are first stored if the signal has exceeded the squelch threshold defined in the `OUTPut:SQUelch` subsystem
`NEVer`	do not store any data in the trace.

**RST state:*
`NEVer`

Example:
```
TRACe:FEED:CONTrol MTRACE, ALWays
```

. . :CONTrol? <trace_name>

Query about the trace loading.
Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`

Result:
`ALW, SQU, NEV`

Example:
```
TRACe:FEED:CONTrol? MTRACE -> ALW
```

. :LIMit

. . [:UPPer] <trace_name>, <numeric_value>|MINimum|MAXimum

Setting the limit of a trace. If the limit is exceeded, the `Limit exceeded Flag` will be set in the `STATus:TRACe` register.
Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`
`<numeric_value>`	limit in percent of the maximum trace length
`MINimum|MAXimum`	sets the lowest|greatest limit

**RST state:*
50 PCT

Example:
```
TRACe:LIMit MTRACE, 50 PCT
```

. . [:UPPer]? <trace_name>[,MINimum|MAXimum]

Query about the trace limit
Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`
no further parameter	query about the current limit
`MINimum|MAXimum`	query about the smallest|greatest limit

Result:
Limit in percent of maximum trace length

Example:
```
TRACe:LIMit? MTRACE -> 50
```

. :POINts? <trace_name>[,MINimum|MAXimum]

Query about the number of values stored in a trace.
The number of values stored in the suppress traces is always 100. Thus, the MAXimum and MINimum value is also 100.
The number of IFPAN trace values is always 2047.

Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`
no further parameter	query about the current number
`MINimum|MAXimum`	query about the lowest|highest number

Result:
Number of values

Example:
```
TRACe:POINts? MTRACE, MAX -> 2048
```

. . :AUTO? <trace_name>

Query about the trace length (auto-adjust).
A 0 (no auto-adjust for trace length) is always output for a suppress trace.

Error message:
If the trace name is unknown, an error `-141, "Invalid character data"` will be generated.

Parameters:

`<trace_name>`	see `TRACe[:DATA]?`

Result:

0	no auto-adjust for trace length
1	auto-adjust for trace length

Example:
```
TRACe:POINts:AUTO? MTRACE;AUTO? ITRACE -> 1;1
```

. . :VALue? <trace_name>, <index>, <numeric_value>

Setting an element of a trace.

Note:
Only suppress traces can be set.

Error message:
If the trace name is unknown or not equal to a suppress trace name, an error `-141, "Invalid character data"` is generated.

Parameters:

`<trace_name>`	name of the trace to be set as `<Character Data>` `SSTART1,SSTOP`
`<index>`	Index of the element within the trace that is to be set. The first element of a trace has the index 1.
`<numeric _value>`	frequency value of the element

**RST state:*
see `TRACe:DATA`

Example:

```
TRACe:VALue SSTART, 13, 98.550 kHz
```

. . :VALue? <trace_name>, <index>

Query about an element of a trace.

Note:
Only elements of the suppress traces can be queried.

Error message:
If the trace name is unknown or not equal to a suppress trace name, an error `-141 "Invalid character data "` is generated.

Parameters:

`<trace_name>`	name of the trace to be read as `<Character Data>` `SSTART,SSTOP`
`<index>`	Index of the element within the trace that is to be set. The first element of a trace has the index 1.

Result:
Frequency value of the element of a trace in Hz

Example:

```
TRACe:Value? SSTART, 13 -> 98550
```

5.4 Device Model and Command Processing

The following figure shows the basic structure of the unit under firmware aspects. The actual receiver is isolated from the remote control units by a central data memory. This memory is at the core of the EM510 firmware and deals with the following tasks:

- Administration of connected modules (receiver, remote clients)
- Making data available to the receiver (e.g. receive frequency, scan parameters, etc.)
- Sequentialization of settings for simultaneous operation
- Sending messages on parameter changes to all modules.

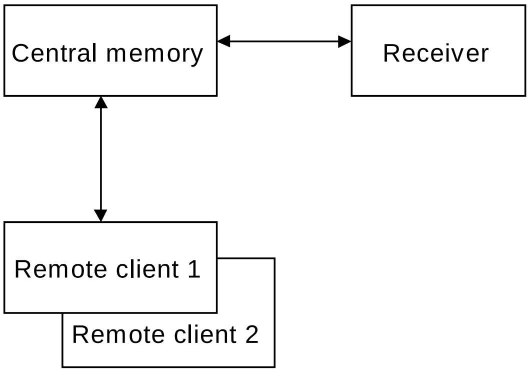


Figure 5-2: Device model with remote control

As mentioned under 5.3 "Description of Commands", the unit can be controlled from one or several remote control units – the remote clients – simultaneously (competitive control). Upon system start, the receiver is logged in to the data memory automatically. The remote clients are logged in if a host computer sets up a link to the EM510.

The receiver obtains the required data (receive frequency, bandwidth, etc.) from the memory. It has no data storage facility of its own and therefore has direct access to the central memory.

Due to the principle of competitive control, different clients can modify the same parameters. The central memory sequentializes the access procedures (last client wins) and sends messages to the other users that a parameter has been changed.

Example 1:

Remote client 1 modifies the frequency value. The central memory signals to the receiver that a new frequency is to be set. The remote client 2 (if connected) is supplied with the new frequency. Every remote client receives a modification report (see "STATus:EXTension Register" on page 5.139).

Example 2:

If the receive frequency is changed by the receiver due to a scanning procedure or an AFC correction, this is reported to remote clients 1 and 2.

5.4.1 Remote Client

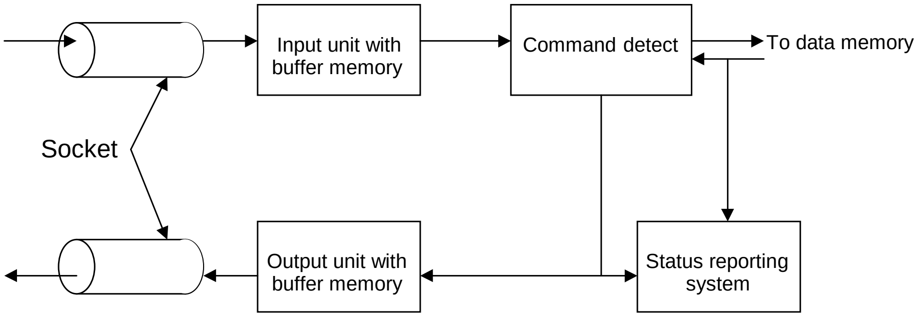
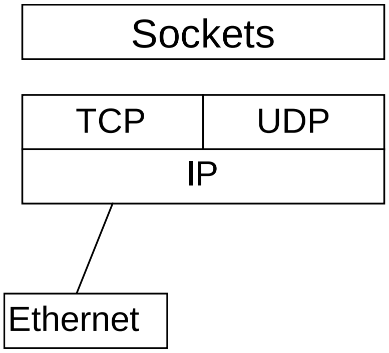


Figure 5-3: Structure of a remote client within the firmware

Sockets:

The remote clients are connected to the host computer by so-called sockets. These are logic point-to-point links that are independent of the transmission medium used. Sockets are based on the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP, not used in EM510). These two protocols are in turn based on the Internet Protocol (IP). The following figure shows the layer model of the sockets.



The transmission media are located beneath the IP layer.

The use of sockets has several advantages:

- the protocols used (PPP, IP, TCP, UDP) are standardized and implemented on all customary operating systems (WindowsNT, Windows95, Windows 3.1, UNIX, SunOS, and many more).
- TCP links are protected against transmission errors.
- Host software can be generated independent of the transmission medium used (LAN or RS232).
- Several logic links may use the same transmission medium.
- IP routing enables access also to remote units and over great distances (e.g. via the Internet).

When the unit is started, a so-called list socket is generated. It functions as the unit's "receptionist". Each host wishing to remote-control the EM510 has to log in with the list socket first. The list socket then generates a new remote client and allocates the link to a new socket so the list socket remains free to receive further hosts.

For login at the list socket, the host needs to have the address and port number of the unit. This can be set in the Setup Remote menu.

Input unit:

Data transmission via sockets is packet-oriented. Each packet received is handed over to command recognition.

Command detect:

Command detect analyzes the data received from the input unit. Data are processed in the sequence they have been received. The data received consist of strings that have to be in accordance with the SCPI standard. The SCPI standard is based on the IEEE 488 standard. Normally, this standard only applies to IEC/IEEE bus (also referred to as IEC625, HPIB or GPIB). Another IEEE standard, IEEE 1174, is a supplement to IEEE 488, making it applicable also to LAN and serial links (RS232). The EM510 uses this standard as a basis for SCPI commands via sockets.

Each identified setting command contained in an SCPI string is first stored in a buffer memory. Only a <Program Message Terminator> (line feed) or a query command will cause the setting commands to be sent to the data memory, where they are checked for consistency. If the commands are consistent, they will be executed at once, and the other modules will be informed. Query commands generate a request to the memory. The memory sends back the data, which will then be processed according to the SCPI standard by the command detect. Finally, the SCPI response strings are sent to the output unit.

Output unit:

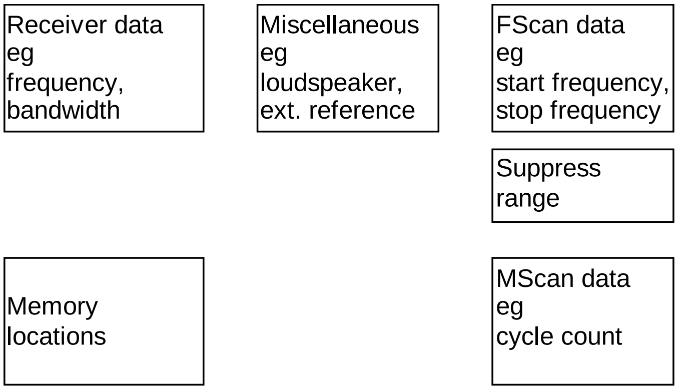
The output unit collects all data in the output buffer that were generated in response to query commands. If the command detect identifies the end of an SCPI command (by the <Program Message Terminators>), it causes the output unit to send the data in the output buffer to the host computer via the socket.

Status Reporting System:

The Status Reporting System gathers information on the device status and makes it available to the output unit on request. The Status Reporting System may be used for messaging asynchronous events (e.g. error statuses, availability of results, data modifications by other users, etc.) to the host computer.

5.4.2 Data Memory

This figure shows the classification of data into data groups. These groups are also reflected by the Status Reporting System of the remote clients in the extension register status.



5.5 Status Reporting System

The status reporting system (see Figure 5-5: Overview of status registers) stores all the information on the present operating state of the device, e.g. that the device presently carries out a SWEeping, and on errors which have occurred. This information is stored in the status registers and in the error queue. For each remote client there is a separate status reporting system and access to all registers of the error queue.

The information is of hierarchical structure. The register status byte (STB) defined in IEEE 488.2 and its associated mask register service request enable (SRE) form the uppermost level. The STB receives its information from the standard event status register (ESR) which is also defined in IEEE 488.2 with the associated mask register standard event status enable (ESE) and registers STATus:OPERation and STATus:QUEStionable which are defined by SCPI, the registers STATus:TRACe and STATus:EXTension not specified by SCPI as well as the two queues error queue and message queue.

The IST flag ("Individual STatus") and the parallel poll enable register (PPE) allocated to it are also part of the status reporting system. The IST flag, like the SRQ, combines the entire device status in a single bit. The PPE fulfills a function for the IST flag as the SRE does for the service request.

The message queue contains the messages the device sends back to the controller. It is not part of the status reporting system but determines the value of the MAV bit in STB and is thus shown in Figure 5-5.

5.5.1 Structure of an SCPI Status Register

Each SCPI register consists of 5 sections, each having a width of 16 bits and different functions (see Figure 5-4). The individual bits are independent of each other, i.e. a bit number being valid for all five sections is assigned to each hardware status. Bit 3 of the STATus:OPERation register, for example, is assigned to the hardware status "SWEeping" in all five sections. Bit 15 (the most significant bit) is set to zero for all sections. Thus the contents of the register sections can be processed by the controller as positive integers.

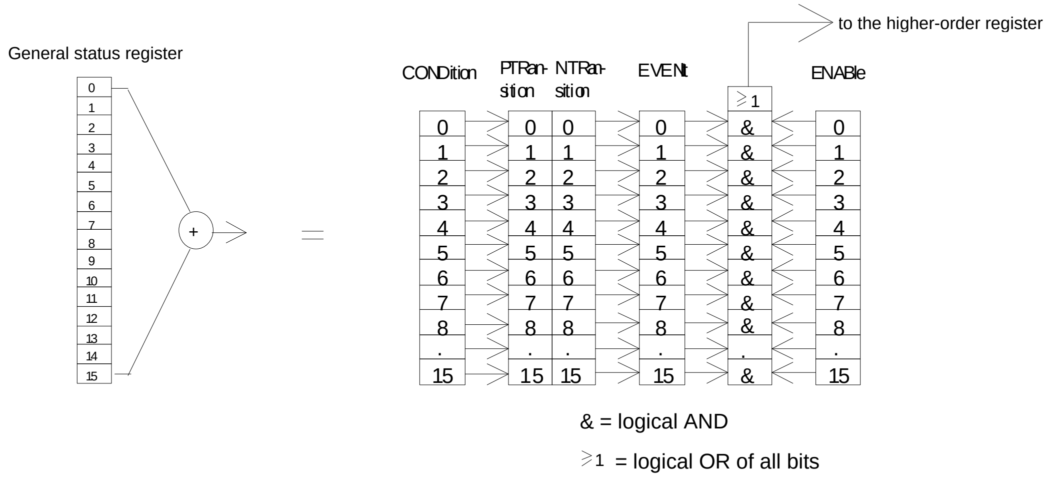


Figure 5-4: Status register model

CONDition section the CONDition section of a register directly reflects the state of the hardware. This register section can only be read. Its contents are not changed during reading.
As an alternative, a bit in a CONDition register can also contain the summary information of a further status register connected in front. In this case, the bit is cleared on reading out the status register.

PTRansition section the Positive-TRansition section acts as an edge detector. When a bit of the CONDition section is changed from 0 to 1, the associated PTR bit decides whether the EVENt bit is set to 1.

PTR bit =1: the EVENt bit is set.

PTR bit =0: the EVENt bit is not set.

This section can be written into and read in any way. Its contents are not changed during reading.

NTRansition section

the Negative-TRansition section also acts as an edge detector. When a bit of the CONDition section is changed from 1 to 0, the associated NTR bit decides whether the EVENt bit is set to 1.

NTR-bit = 1: the EVENt bit is set.

NTR-bit = 0: the EVENt bit is not set.

This section can be written into and read in any way. Its contents are not changed during reading.

With these two edge register sections the user can define which state transition of the condition section (none, 0 to 1, 1 to 0 or both) is stored in the EVENt section.

EVENt section

the EVENt section indicates whether an event has occurred since the last reading; it is the "memory" of the CONDition section. It only indicates events passed on by the edge filters. The EVENt section is permanently updated by the device. This part can only be read. During reading, its contents are set to zero. This section is often regarded as the entire register.

ENABle section

the ENABle section determines whether the associated EVENt bit contributes to the summary bit (see below). Each bit of the EVENt section is ANDed with the associated ENABle bit (symbol '&'). The results of all logical operations of this section are passed on to the summary bit via an OR function (symbol ' ≥ 1').

> ENABle bit = 0: the associated EVENt bit does not contribute to the summary bit.
>
> ENABle bit = 1: if the associated EVENT bit is "1", the summary bit is set to "1" as well.

This section can be written into and read by the user in any way. Its contents are not changed during reading.

Summary bit

as indicated above, the summary bit is obtained from the EVENt and ENABle section for each register. The result is then entered into a bit of the CONDition section of the higher-order register.

The device automatically generates the summary bit for each register. Thus an event, e.g. a PLL that has not locked, can lead to a service request through all the hierarchy levels.

Note:
The service request enable register SRE defined in IEEE 488.2 can be taken as ENABle section of the STB if the STB is structured according to SCPI. By analogy, the ESE can be taken as the ENABle section of the ESR.

5.5.2 Overview of the Status Registers

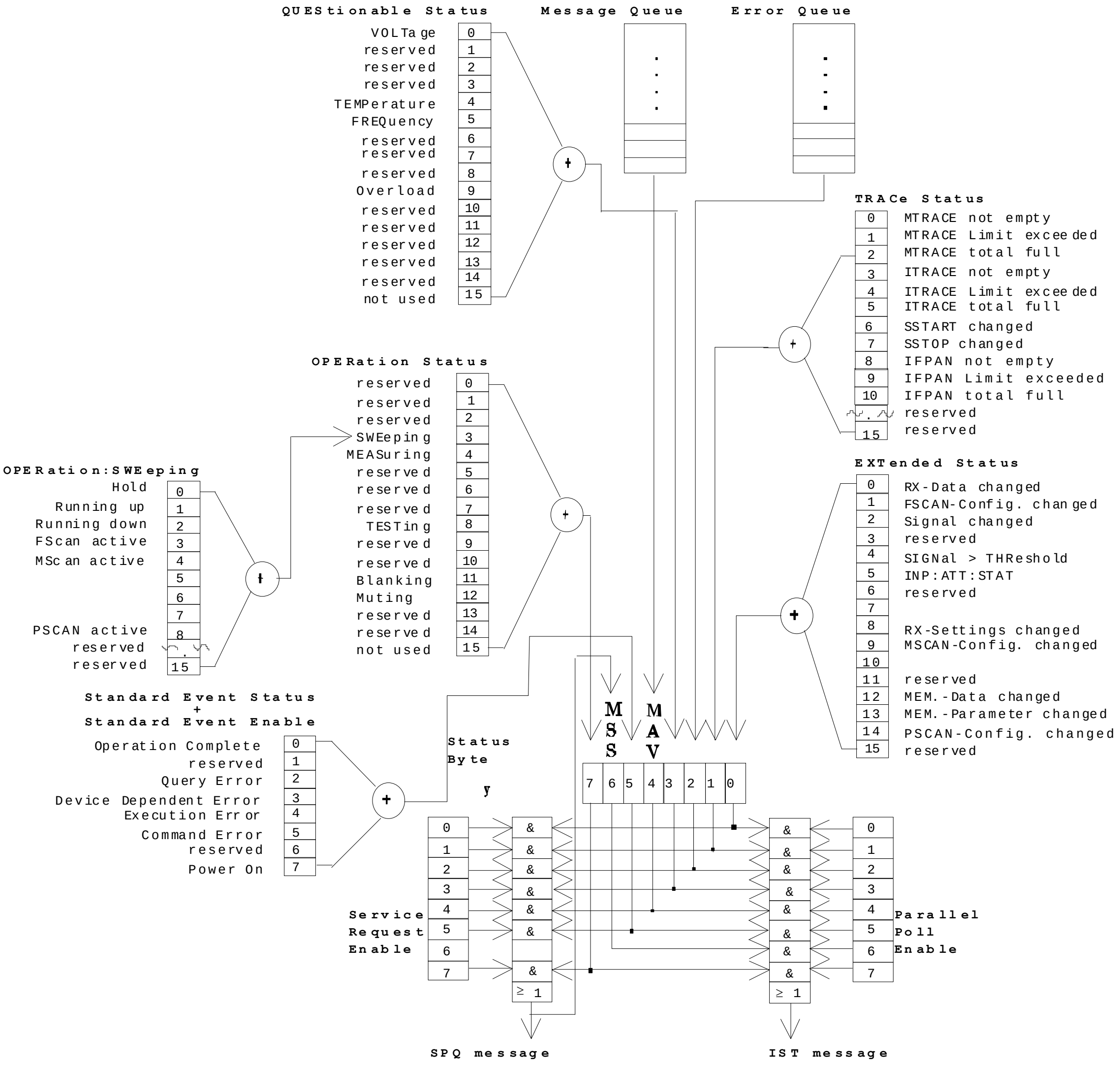


Figure 5-5: Overview of status registers

5.5.3 Description of the Status Registers

5.5.3.1 Status Byte (STB) and Service Request Enable Register (SRE)

The STB is already defined in IEEE 488.2. It provides an overview of the device status by collecting the pieces of information of the lower registers. It can thus be compared with the CONDition section of an SCPI register and assumes the highest level within the SCPI hierarchy. A special feature is that bit 6 acts as the summary bit of the remaining bits of the status byte.

The STATUS BYTE is read out using the command "*STB?" or a "serial poll".

The STB implies the SRE. As to its function, it corresponds to the ENABle section of the SCPI register. A bit in the SRE is assigned to each bit of the STB. Bit 6 of the SRE is ignored. If a bit is set in the SRE and the associated bit in the STB changes from 0 to 1, a Service Request (SRQ) is generated, which triggers an interrupt in the controller if this is appropriately configured.

The SRE can be set using command "*SRE" and read using "*SRE?".

Table 5-1: Bit allocation of status byte

Bit no.	Meaning
0	**EXTended status register summary bit** The bit is set if an EVENt bit is set in the EXTended status register and if the corresponding ENABle bit is set to 1. The states of the hardware functions and change bits are combined in the EXTended status register.
1	**TRACe status register summary bit** The bit is set if an EVENt bit is set in the TRACe status register and if the corresponding ENABle bit is set to 1. The states of the TRACes MTRACE, ITRACE, SSTART and SSTOP are represented in the TRACe status register.
2	**Error Queue not empty** The bit is set when the error queue contains an entry. If this bit is enabled by the SRE, an entry into the empty error queue generates a service request. Thus, an error can be recognized and specified in greater detail by polling the error queue. The poll provides an informative error message. This procedure is recommended since it considerably reduces the problems involved with the control.
3	**QUEStionable status register summary bit** The bit is set if an EVENt bit is set in the QUEStionable status register and the corresponding ENABle bit is set to 1. A set bit indicates a questionable device status which can be specified in greater detail by polling the QUEStionable status register.
4	**MAV bit (message available)** No meaning
5	**ESB bit** Summary bit of the EVENt status register. It is set if one of the bits in the EVENt status register is set and enabled in the EVENt status enable register. Setting this bit implies a serious error which can be specified in greater detail by polling the EVENt status register.

Bit no.	Meaning
6	**MSS bit** (master status summary bit) The bit is set if the device triggers a service request. This is the case if one of the other bits of this register is set together with its mask bit in the service request enable register SRE.
7	**OPERation status register summary bit** The bit is set if an EVENt bit is set in the OPERation status register and the corresponding ENABle bit is set to 1. A set bit indicates that the device is just performing an action. The type of action can be determined by polling the QUEStionable status register.

5.5.3.2 IST Flag and Parallel Poll Enable Register (PPE)

Analogous to the SRQ, the IST flag combines the entire status information in a single bit. It can be queried by using command "`*IST?`".

The parallel poll enable register (PPE) determines which bits of the STB contribute to the IST flag. The bits of the STB are ANDed with the corresponding bits of the PPE. In contrast to SRE bit 6 is also used here. The IST flag results from the ORing of all results. The PPE can be set using the "`*PRE`" commands and read using the "`*PRE?`" command.

5.5.3.3 Event Status Register (ESR) and Event Status Enable Register (ESE)

The ESR is already defined in IEEE 488.2. It can be compared with the EVENt section of an SCPI register. The EVENt status register can be read out using the "*ESR?" command .

The ESE is the associated ENABle section. It can be set using the "*ESE" command and read using the "*ESE?" command .

Table 5-2: Bit allocation of event status register

Bit No.	Meaning
0	**Operation Complete** This bit is set on receipt of the command *OPC exactly when all previous commands have been executed.
2	**Query Error** This bit is set if either the controller wants to read data from the device without having sent a query, or if it does not fetch requested data and sends new instructions to the device instead. The cause is often a query which is faulty and hence cannot be executed.
3	**Device-dependent error** This bit is set if a device-dependent error occurs. An error message with a number between -300 and -399 or a positive error number denoting the error in greater detail is entered into the error queue (see 6.2.1 , Error Messages).
4	**Execution Error** This bit is set if a received command is syntactically correct but cannot be performed for different reasons. An error message with a number between -200 and -299 denoting the error in greater detail is entered into the error queue (see 6.2.1 , Error Messages).
5	**Command Error** This bit is set if an undefined and syntactically incorrect command is received. An error message with a number between -100 and -199 denoting the error in greater detail is entered into the error queue (see 6.2.1 , Error Messages).
7	**Power On** (supply voltage on) This bit is set when the device is switched on.

5.5.3.4 STATus:OPERation Register

In the CONDition section, this register contains information about the type of actions currently being executed by the device. In the EVENt section, it also contains information about the type of actions having been executed since the last reading. It can be read using the commands

"`STATus:OPERation:CONDition?"` or

"`STATus:OPERation[:EVENt]?".`

Table 5-3: Bit allocation of STATus:OPERation register

Bit No.	Meaning
3	**SWEeping** This bit is set if the summary bit of the STATus:OPERation:SWEeping bits is set.
4	**MEASuring** This bit is set as long as a measurement is carried out.
8	**TESTing** This bit is set if the self-test has been triggered.
11	**BLANking** This bit is set if the blanking input has been activated
12	**Muting** This bit is set if the blanking input has been activated

5.5.3.5 STATus:OPERation:SWEeping Register

This register contains further information about the status of the device. The device is either in normal reception (fixed frequency) or in one of several scan modes (FSCAN, MSCAN, PSCAN).

The state is determined by the `SENSe:FREQuency:MODE` command with the `CW|FIXed` state being characterized by deleting bits 3 to 5 in the `STATus:OPERation:SWEeping` register.

Table 5-4: Bit allocation of STATus:OPERation:SWEeping register

Bit No.	Meaning
0	**Hold** This bit is set if a FSCAN or MSCAN was interrupted by a fulfilled hold condition.
1	**Running up** This bit is set if scanning is to be carried out to increasing frequency values or memory locations.
2	**Running down** This bit is set if scanning is to be carried out to decreasing frequency values or memory locations.
3	**FSCAN active** This bit is set if FREQ:MODE is set on SWEep.
4	**MSCAN active** This bit is set if FREQ:MODE is set on MSCan.
5 - 7	**reserved**
8	**PSCAN active** This bit is set if FREQ:MODE is set on PSCan (software option EM510PS).
9	**reserved**

5.5.3.6 STATus:QUEStionable Register

This register contains information on ambiguous device states. They can occur, for example if the device is operated outside its specification range. It can be queried using the commands `STATus:QUEStionable:CONDition?` or `STATus:QUEStionable[:EVENt]?.`

Table 5-5: Bit allocation of STATus:QUEStionable register

Bit No.	Meaning
0	**VOLTage** This bit is set if an ambiguous supply voltage occurs. Due to that all test points of the supply voltage are checked.
4	**TEMPerature** This bit is set if the internal temperature is too high. All temperature test points in the equipment are tested.
5	**FREQuency** This bit is set if an internal oscillator frequency is ambiguous. Due to that the test points of the oscillators are checked.
9	**OVERload** This bit is set if the IF section is overdriven by too high an input level. Then the result of a level measurement is questionable.

5.5.3.7 STATus:TRACe Register

This register contains information on ambiguous states of the traces MTRACE, ITRACE, IFPAN, SSTART and SSTOP.

It can be queried with the commands `STATus:TRACe:CONDition?` or `STATus:TRACe[:EVENt]?`.

Table 5-6: Bit allocation of STATus:TRACe register

Bit No.	Meaning
0	MTRACE not empty This bit is set if the MTRACE contains at least one measured value.
1	**MTRACE limit exceeded** This bit is set if the number of measured values contained in the MTRACE exceeds the threshold given by the command `TRACe:LIMit[:UPPer] MTRACE.`
2	**MTRACE total full** This bit is set if the MTRACE is loaded with the maximum number of measured values.
3	**ITRACE not empty** This bit is set if the ITRACE contains at least one information value.
4	**ITRACE limit exceeded** This bit is set if the number of measured values contained in the ITRACE exceeds the threshold given by the command `TRACe:LIMit[:UPPer] ITRACE`.
5	**ITRACE total full** This bit is set if the ITRACE is loaded with the maximum number of information values.
6	**SSTART changed** This bit is set if one or several start frequencies of the current suppress table have changed.
7	**SSTOP changed** This bit is set if one or several stop frequencies of the current suppress table have changed.
8	**IFPAN not empty** This bit is set if at least one measured value is stored under IFPAN.
9	**IFPAN Limit exceeded** This bit is set if the number of measured values stored under IFPAN exceeds the threshold set by `TRACe:LIMit[:UPPer] IFPAN`.
10	**IFPAN total full** This bit is set if the maximal number of measured values is stored under IFPAN.

5.5.3.8 STATus:EXTension Register

This register contains in the CONDition part information on different receiver states which cannot be assigned to the other registers. Information about the actions the unit has carried out since the last read out are stored in the EVENt part. The corresponding registers can be queried with the commands `STATus:EXTension:CONDition?` or `STATus:EXTension[:EVENt]?` .

Table 5-7: Bit allocation of STATus:EXTension register

Bit No.	Meaning
0	**RX-Data changed** This bit is set if the receiver data were changed by another remote client (see also "Data Memory" on page 5.127).
1	**FSCAN-Configuration changed** This bit is set if the FSCAN data were changed by another remote client (see also "Data Memory" on page 5.127).
2	**Signal changed** This bit is set if the received signal changes in level or offset.
3	**COR active** This bit is set if the COR relay becomes active.
4	**SIGNal > THReshold** This bit is set if the signal level is above the squelch threshold (precondition: squelch is switched on).
5	**INPut:ATTenuation:STATe** This bit is set if the input attenuator is switched on.
7	**Reserved**
8	**RX-Settings changed** This bit is set if a parameter was changed by another remote client in the data set "miscellaneous".
9	**MSCAN-Configuration changed** This bit is set if the MSCAN data set is changed by another remote client .
10	**Reserved**
12	**MEMory-Data changed** This bit is set if a memory data was changed by another remote client.
13	**MEMory-Parameter changed** This bit is set if the query bit was changed by another remote client.
14	**PSCAN-Configuration changed** This bit is set if the PSCAN data set is changed by another remote client (PSCAN is an option).

With bits 0 to 2 and 7 to 10 and 12 to 13, the host can be informed via an SRQ about parameter changes. Cyclical polling of the settings by the host is thus stopped during operation via several interfaces or if the signal parameters are to be indicated. In the CONDition section of the register, the change bits are set after a signal change and are reset by special query commands. Changes done by another remote client affect the change bits equally.

Table 5-8: Allocation of change bits in STATus:EXTension register

Bit No.	Set by change of	Reset by one of the commands
0	frequency, demodulation, bandwidth, threshold value, MGC value, control mode, antenna no., attenuation, detector mode, squelch actuation, squelch control, sensor function, AFC, Aux bit(s), Aux output mode, measure time, measure mode, preselection mode, video mode	FREQ?, DEM?, BAND?, OUTP:SQU:THR?, GCON?, GCON:MODE?, ROUTe:CLOSe:STATe?, INP:ATT?, DET?, OUTP:SQU?, OUTP:SQU:CONT?, FUNC?, FREQ:AFC?, MEM:CONT? RX, OUTP:BITAx?, OUTP:BYTAx?, OUTP:AUX?, MEAS:TIME?, MEAS:MODE?, INP:ATT:MODE?, OUTP:VID:MODE?
1	FSCAN: start frequency, stop frequency, step width, number of scans, synchronization time, listening time, scan mode	FREQ:STAR?, FREQ:STOP?, SWE:STEP?, SWE:COUN?, SWE:DWEL?, SWE:HOLD:TIME?, SWE:DIR?, SWE:CONT?
2	signal level, offset	SENS:DATA?
7	display mode, antenna name	DISP:MENU?, ROUT:PATH[:DEF]?
8	volume, balance, external reference	SYST:AUD:VOL?, SYST:AUD:BAL?, ROSC:SOUR?
9	MSCAN: number of scans, synchronization time, listening time, search mode	MSC:COUN?, MSC:DWEL?, MSC:HOLD:TIME?, MSC:DIR?; MSC:CONT?
10	**Reserved**	
12	frequency, demodulation, bandwidth, threshold value, antenna no., attenuation, squelch actuation, AFC	MEM:CONT? MEM0 ... MEM9999 MEM: CONT: MPAR? MEM0 ... MEM9999
13	query bit (set, set back)	MEM:CONT? MEM0 ... MEM9999 MEM: CONT: MPAR? MEM0 ... MEM9999
14	PSCAN: center frequency, span frequency, start frequency, stop frequency, number of scans, channel raster	FREQ:PSC:CENT?, FREQ:PSC:SPAN?, FREQ:PSC:STAR?, FREQ:PSC:STOP?, PSC:COUNT?, FREQ:PSC:STEP?

5.5.4 Use of the Status Reporting System

In order to be able to effectively use the status reporting system, the information contained there has to be transmitted to the host where it is further processed. There are several methods which are described in the following. Detailed programming examples are given in Annex E: LAN Programming Examples.

5.5.4.1 Service Request, Making Use of the Hierarchy Structure

Under certain circumstances, the device can send a "service request" (SRQ) to the host. As Figure 5-5 shows, an SRQ is always initiated if one or several bits of bits 0, 1, 2, 3, 4, 5 or 7 of the status byte are set and enabled in the SRE. Each of these bits combines the information of a further register, the error queue or the output buffer. By setting the ENABle sections of the status registers correspondingly, it is possible for any bits in any status register to initiate an SRQ. In order to make use of the possibilities of the service request, all bits should be set to "1" in enable registers SRE and ESE.

Examples (see also Figure 5-5):

Use command "*OPC" to generate an SRQ

- Set bit 0 in the ESE (Operation Complete)
- Set bit 5 in the SRE

After completion of the settings, the device generates an SRQ.

Indication of a signal during a sweep by means of an SRQ at the host

- Set bit 7 in the SRE (summary bit of the STATus:OPERation register)
- Set bit 3 (SWEeping) in the STATus:OPERation:ENABle.
- Set bit 3 in the STATus:OPERation:NTRansition so that the change of SWEeping bit 3 from 0 to 1 is also recorded in the EVENt section.
- Set bit 0 in STATus:OPERation:SWEeping:ENABle
- Set bit 0 in STATus:OPERation:SWEeping:PTRansition so that the change of hold bit 0 from 0 to 1 is also recorded in the EVENt section.

The device generates an SRQ after a signal has been found.

The SRQ is the only possibility for the device to become active on its own. Each host program should set the device so that a service request is initiated in case of malfunction. The program should react appropriately to the service request. A detailed example for a service request routine can be found in Annex E: LAN Programming Examples.

5.5.4.2 Query by Means of Commands

Each part of every status register can be read by means of queries. The individual commands are indicated in the detailed description of the registers in section "Description of the Status Registers" on page 5.132. Only one number is returned which represents the bit pattern of the register queried. The format of the number can be set by the `FORMat:SREGister` command.

Queries are usually used after an SRQ in order to obtain more detailed information on the cause of the SRQ.

5.5.4.3 Error-Queue Query

Each error state in the device results in an entry in the error queue. The entries of the error queue are detailed plain-text error messages which can be queried via the IEC/IEEE bus using the command `SYSTem:ERRor?`. Each call of `SYSTem:ERRor?` provides an entry from the error queue. If no error messages are stored there any more, the device responds with 0, "`No error`".

The error queue should be queried after every SRQ in the controller program as the entries describe the cause of an error more precisely than the status registers. Especially during the test phase of a controller program the error queue should be queried regularly since faulty commands from the controller to the device are recorded there as well.

5.5.5 Resetting Values of the Status Reporting System

Table 5-9 comprises the different commands and events causing the status reporting system to be reset. None of the commands, except for *RST, influences the functional device settings. In particular, DCL does not change the device settings.

Table 5-9: Resetting device functions

Event	Switching on supply voltage	DCL, SDC			
		(Device Clear, Selected Device Clear)	*RST	STATus:PRESet	*CLS
Effect					
Clear STB, ESR	yes	—	—	—	yes
Clear SRE, ESE	yes	—	—	—	—
Clear PPE	yes	—	—	—	—
Clear EVENTt sections of the registers	yes	—	—	—	yes
Clear ENABle section of all OPERation and QUEStionable registers, Fill ENABle sections of all other registers with "1".	yes	—	—	yes	—
Fill PTRansition sections with "1" , Clear NTRansition sections	yes	—	—	yes	—
Clear error queue	yes	—	—	—	yes
Clear output buffer	yes	yes	1)	1)	1)
Clear command processing and input buffer	yes	yes	—	—	yes

1) The first command in a line, i.e. immediately following a <PROGRAM MESSAGE TERMINATOR>, clears the output buffer.

6 Maintenance and Troubleshooting

6.1 Maintenance

6.1.1 Alignment of the 10-MHz reference oscillator crystal in the IF section

In order to ensure accurate receiver frequencies the internal 10-MHz reference oscillator crystal must be trimmed every year.

A 1.5-mm screwdriver for cheese head screws is required.

⇒ Connect a frequency counter with a tolerance $\leq 1\times10^{-8}$ to X4 REF EXT/INT at the rear panel.

⇒ Start the internal reference.

⇒ Trim the frequency to 10 MHz ± 0.5 Hz at room temperature.

The alignment is carried out via remote control and is described in chapter 5.3.5 CALibration Subsystem

6.1.2 Reset

A reset is carried out at the front panel of the device when line EXT_RST pin 23 of connector X12b is briefly grounded (for example, via pin 26 of connector X12b) while the unit is on.

The data for the configuration of the LAN interface are stored in an EEPROM safe of power failure and cannot be changed by a reset.

The data for the configuration of the LAN interface are stored fail-safe in an EEPROM and not changed by a reset.

6.1.3 Firmware Update

See also 4.4 Firmware Update.

If modules are exchanged, a firmware update might be required.

In this case a CD-ROM with the current firmware will be shipped together with the new module.

The CD-ROM also contains the release notes with instructions for the installation of the firmware.

The release notes also describe the changes that were made with the new firmware.

The firmware can also be obtained from the EM510 download area at the following internet address:

http://www.rohde-schwarz.com

6.2 Troubleshooting

6.2.1 Error messages

Error message	Cause	Module number
Mainboard defective	The mainboard (EM510MB) indicates an error.	4065.7840.02
DSP IF Section defective	The IF section (EM510Z1) indicates an error. The IF section is part of the mainboard.	4065.7840.02
Preselector defective	The preselection (EM510V1) indicates an error.	4065.7940.02
Check external reference	The EM510 is switched to external reference and there is no valid reference signal.	

6.2.2 Test Points

There is a constant voltage and temperature check at many test points of the modules of the EM510. Partly these test points depend on each other. The error messaging system considers these dependences and reports only the causal source of errors.

A query about the state of all modules can be done by the command `DIAGnostic:MODule:STATe?` .

A query about the information of all test points of all modules can be done by the command `DIAGnostic:MONitor? ALL`.

MB Mainboard

VXI +5V_PS → PPC (PowerPC 8260) → MB +3.3V →
Clock 76.8 MHz →
VXI +5V > 3.8 V
and Clock 76.8 MHz present
is precondition for measuring testpoints

	n	Testpoint	Description			
VXI +5V_PS →	0	+3.3V	Supply +3.3 V	MB +3.3V → VCC_IO_FPGA +3.3V →	MB	Q.S.V.
VXI +5V_PS →	1	+2.5V	Supply +2.5 V	MB +2.5V →	MB	Q.S.V.
VXI +5V_PS →	2	V_CORE_FPGA	Core Voltage FPGA	VCC_CORE_FPGA +2.5V →	MB	Q.S.V.
MB +3.3V →	3	V_CORE_PPC	Core Voltage PPC	VCC_CORE_PPC +2.0V →	MB	Q.S.V.
MB +3.3V →	4	V_CORE_DSP	Core Voltage DSP	VCC_CORE_DSP +1.6V →	MB	Q.S.V.
VXI +5V_PS →	5	+5.0V	Supply +5.0 V	MB +5.0V →	V1, MB(ADC)	Q.S.V.
VXI +5V_PS →	6	+9.0V	Supply + 9.0 V	MB +9.0V →	V1	Q.S.V.
VXI +5V_PS →	7	-5.0V	Supply -5.0 V	MB -5.0V →	V1	Q.S.V.
VXI +5V_PS →	8	-9.0V	Supply -9.0V	MB -9.0V →	V1	Q.S.V.
	9	REFLEV_10M	Referenz Level 10MHz	→ Q.S.F.		
MB +5.0V →	10	TUNE_76.8M	Tuning 76.8 MHz Clock	Clock 76.8 MHz →	Q.S.F.	
	11	TEMP_LOCAL	Temperature Mainboard -20 °C < T < 70 °C	→ Q.S.T.		
	12	TEMP_PPC	Temperature PPC -20 °C < T < 75 °C	→ Q.S.T.		
	13	TEMP_FPGA	Temperature FPGA -20 °C < T < 90 °C	→ Q.S.T.		
	14	TEMP_ADSECT	Temperature of IF section -20 °C < T < 70 °C	→ Q.S.T.		
MB +5.0V →	15	ADSV_3V3	A/D-Converter Supply 3.3V	→ Q.S.V.		

V1 Preselection

MB +3.3V → MB +3.3V is precondition for measuring testpoints
MB +9.0V → MB +9.0V is precondition for TREAMP_P
MB -9.0V → MB -9.0V is precondition for TREAMP_N
MB +5.0V → MB +5.0V supplies HF Relais

n	Testpoint	Description	
0	TPREAMP_P	Preamplifier Positive Supply Current	
1	TPREAMP_N	Preamplifier Negative Supply Current	
2	TTEMP	Temperature	→ Q.S.T.

Q.S.V.: Questionable Status Register Bit 0 Voltage
Q.S.T.: Questionable Status Register Bit 4 Temperature
Q.S.F.: Questionable Status Register Bit 5 Frequency