Contents - Chapter 7 "Remote Control - Programming Examples"

7 Remote Control - Programming Examples

The following programming examples have a hierarchical structure, ie subsequent examples are based on previous ones. It is thus possible to compile very easily an operational program from the modules of the given examples.

Basic Steps of IEC/IEEE-Bus Programming

The examples explain the programming of the instrument and can serve as a basis to solve more complex programming tasks.

VISUAL BASIC has been used as programming language. However, the programs can be translated into other languages.

Including IEC-Bus Library for VisualBasic

Programming hints:

- **Output of texts using the "Print" function**

  The following programming examples are based on the assumption that all subroutines are part of a form (file extension: .FRM). In this case the syntax

  ```
  Print "Text"
  ```

  is allowed.

  If however the subroutines are stored as a so-called module (file extension: .BAS), the print instruction should be preceded by the name of a form which has the required print method. If, for example, there is a form with the name "Main", the associated print instruction is as follows:

  ```
  Main.Print "Text".
  ```

- **Access to funtions of GPIB.DLL**

  To create Visual Basic control applications the file GPIB.BAS (as from VB 6.0 VBIB-32.BAS) is added to a project so that the functions of the RSIB.DLL can be called. In addition, the file NIGLOBAL.BAS is added to the project. This file contains constants and definitions for the processing of errors, timeout values, etc.

- **Declaration of DLL functions as procedures**

  Since the functions all return an integer value, the functions in the file GPIB.BAS are all declared as follows:

  ```
  Declare Function xxx Lib "gpib.dll" ( ... ) As Integer
  ```

  The function value with the status variables `ibsta` should be allocated a variable when it is called up. Since this value is also returned via a reference parameter of the functions, the functions can be declared as procedures as follows:

  ```
  Declare Sub xxx Lib "rsib.dll" ( ... )
  ```

- **Generating a response buffer**

  Since the DLL returns zero-terminated strings in case of responses, a string of sufficient length should be generated prior to calling the functions `ibrd()` and `ilrd()`, since Visual Basic prefixes a length value to the strings which is not updated by the DLL.
  The following two possibilities are available to generate a length value for a string:

  - **`Dim Rd as String * 100`**
  - **`Dim Rd as String`**
    `Rd = Space$(100)`

Initialization and Default Status

Variables used by all subroutines should be stored at the beginning of every program.
Then the IEC/IEEE bus as well as the settings of the instrument are brought into a defined default status at the beginning of every program. Subroutines `"InitController"` and `"InitDevice"` are used to this effect.

Creating global variables

Global variables are placed in so-called "modules" (file extension: .BAS) in Visual Basic. Therefore, at least one module (z.B. "GLOBALS.BAS) should be created which contains the variables used by all subroutines, such as the device addresses used by the IEC/IEEE-bus driver.
The file should contain the following instructions for the programming examples below:

```
Global analyzer As Integer
Global boardId As Integer
```

Initiate Controller

```
REM ------------ Initiate controller -------------------------------------
Public SUB InitController()


iecaddress% = 20                          'IEC/IEEE-bus address of the
                                          'instrument
CALL IBFIND("GPIB0", boardId%)            'Open port to the controller
CALL IBFIND("DEV1", analyzer%)            'Open port to the instrument
CALL IBPAD(analyzer%, iecaddress%)        'Inform controller on instrument
                                          'address
CALL IBTMO(analyzer%, 11)                 'Response time to 1 sec
END SUB
REM ************************************************************************
```

Initiate Instrument

The IEC-bus status registers and instrument settings of the instrument are brought into the default status.

```
REM ------------ Initiate instrument -------------------------------------
Public SUB InitDevice()

CALL IBWRT(analyzer%, "*CLS")           'Reset status registers
CALL IBWRT(analyzer%, "*RST")           'Reset instrument
END SUB
REM*************************************************************************
```

Configuring Power Save Function of Display

The results on the screen are often not required during IEC/IEEE-bus operation. Although the command "SYSTem:DISPlay:UPDate OFF" switches off the display of results which brings considerable advantages in terms of speed in the remote control mode, the display itself and in particular the backlighting remain switched on.

If required, the display should be switched off by means of the power-save function, the response time having to be set in minutes prior to activation.

Note: *The display is switched on as soon as a key is pressed on the instrument front panel.*

```
REM --------- Configure power save function -----------------------------
Public SUB PowerSave()

CALL IBWRT(analyzer%, "SYSTem:PSAVe:HOLDoff 1") 'Set holdoff to 1 minute
CALL IBWRT(analyzer%, "SYSTem:PSAVe ON")        'Power save function on
END SUB
REM*************************************************************************
```

Transmission of Simple Instrument Setting Commands

Center frequency, span, and reference level of the instrument are set in this example.

```
REM -------- Instrument setting commands --------------------------------
PUBLIC SUB SimpleSettings()

CALL IBWRT(analyzer%, "FREQUENCY:CENTER 128MHz")  'Center frequency 128 MHz
CALL IBWRT(analyzer%, "FREQUENCY:SPAN 10MHZ")     'Span 10 MHz
CALL IBWRT(analyzer%, "DISPLAY:TRACE:Y:RLEVEL -10dBm")
                                                  'Reference level -10dBm
END SUB
REM ***************************************************************************
```

Return to manual control

```
REM -------- Switch instrument over to manual control -------------------
CALL IBLOC(analyzer%)                'Set instrument to Local state
REM ***************************************************************************
```

Reading out Instrument Settings

The settings made in the above example are read out here. The abbreviated commands are used.

```
REM --------- Reading out instrument settings ---------------------------
PUBLIC SUB ReadSettings()

CFfrequency$ = SPACE$(20)           'Provide text variables (20 characters)
CALL IBWRT(analyzer%, "FREQ:CENT?") 'Request center frequency
CALL IBRD(analyzer%, CFfrequency$)  'Read value

CFspan$ = SPACE$(20)                'Provide text variables (20 characters)
CALL IBWRT(analyzer%, "FREQ:SPAN?") 'Query span
CALL IBRD(analyzer%, CFspan$)       'Read value

RLevel$ = SPACE$(20)                'Provide text variables (20 characters)
CALL IBWRT(analyzer%, "DISP:TRAC:Y:RLEV?")
                                    'Query reference level
CALL IBRD(analyzer%, RLevel$)       'Read value
```

```
REM ---------------- Display values on the screen ------------------------
PRINT "Center frequency: "; CFfrequency$,
PRINT "Span:             "; CFspan$,
PRINT "Reference level:  "; RLevel$,
REM*************************************************************************
```

Positioning a Marker and Displaying Values

```
REM -------------- Examples of marker functions --------------------------
PUBLIC SUB ReadMarker()

CALL IBWRT(analyzer%, "CALC:MARKER ON;MARKER:MAX")
            'Activate marker1 and start peak search
MKmark$ = SPACE$(30)              'Provide text variables (30 characters)
CALL IBWRT(analyzer%, "CALC:MARK:X?;Y?")    'Query frequency and level
CALL IBRD(analyzer%, MKmark$)               'Read value

REM -------------- Display values on the screen -------------------------
PRINT "Center frequency / level "; MKmark$,
REM ************************************************************************
```

Command synchronization

The possibilities for synchronization implemented in the following example are described in Chapter 5, Section "Command Order and Command Synchronization".

```
REM -------- Examples of command synchronization -------------------------
PUBLIC SUB SweepSync()


REM The command INITiate[:IMMediate] starts a single sweep if the command
REM INIT:CONT OFF was previously sent. It should be ensured that the next
REM command is only then executed when the entire sweep is complete.
CALL IBWRT(analyzer%, "INIT:CONT OFF")

REM -------- First possibility: Use of *WAI ------------------------------
CALL IBWRT(analyzer%, "ABOR;INIT:IMM; *WAI")


REM -------- Second possibility: Use of *OPC? ----------------------------
OpcOk$ = SPACE$(2)                  'Space for *OPC? - Provide response
CALL IBWRT(analyzer%, "ABOR;INIT:IMM; *OPC?")
REM -------- here the controller can service other instrument-------------
CALL IBRD(analyzer%, OpcOk$)                  'Wait for "1" from *OPC?


REM -------- Third possibility: Use of *OPC ------------------------------
REM In order to be able touse the service request function in conjugation
REM with a National Instruments GPIB driver, the setting "Disable
REM Auto Serial Poll" must be changed to "yes" by means of IBCONF!
CALL IBWRT(analyzer%, "*SRE 32")      'Permit service request for ESR
CALL IBWRT(analyzer%, "*ESE 1")       'Set event-enable bit for
                                      'operation-complete bit
CALL IBWRT(analyzer%, "ABOR;INIT:IMM; *OPC") 'Start sweep and
                                             'synchronize with OPC
CALL WaitSRQ(boardID%,result%)        'Wait for service request
REM Continue main program here.
END SUB
REM ************************************************************************
```

Service Request

The service request routine requires an extended initialization of the instrument in which the respective bits of the transition and enable registers are set.
In order to use the service request function in conjugation with National Instruments GPIB driver, the setting "Disable Auto Serial Poll" must be changed to "yes" by means of IBCONF.

Initiate Service Request

```
REM ---- Example of initialization of the SRQ in the case of errors --------
PUBLIC SUB SetupSRQ()

CALL IBWRT(analyzer%, "*CLS")                 'Reset status reporting system
CALL IBWRT(analyzer%,"*SRE 168")              'Permit service request for
                                              'STAT:OPER,STAT:QUES and ESR
                                              'register
CALL IBWRT(analyzer%,"*ESE 60")               'Set event-enable bit for
                                              'command, execution, device-
                                              'dependent and query error
CALL IBWRT(analyzer%,"STAT:OPER:ENAB 32767")  'Set OPERation enable bit for
                                              'all events
CALL IBWRT(analyzer%,"STAT:OPER:PTR 32767")   'Set appropriate OPERation
                                              'Ptransition bits
CALL IBWRT(analyzer%,"STAT:QUES:ENAB 32767")  'Set questionable enable bits
                                              'for all events
CALL IBWRT(analyzer%,"STAT:QUES:PTR 32767")   'Set appropriate questionable
                                              'Ptransition bits
CALL WaitSRQ(boardID%,result%)                'Wait for Service Request
IF (result% = 1) THEN CALL Srq                'If SRQ is recognized =>
                                              'subroutine for evaluation
END SUB
REM ***************************************************************************
```

Service request routine

A service request is then processed in the service request routine.
Note: the variables userN% and userM% must be pre-assigned usefully!

```
REM ------------ Service request routine ---------------------------------
Public SUB Srq()

ON ERROR GOTO noDevice                          'No user existing
CALL IBRSP(analyzer%, STB%)                     'Serial poll, read status byte
IF STB% > 0 THEN                                'This instrument has bits set
                                                'in the STB
     SRQFOUND% = 1
     IF (STB% AND 16)  > 0 THEN CALL Outputqueue
     IF (STB% AND 4)   > 0 THEN CALL ErrorQueueHandler
     IF (STB% AND 8)   > 0 THEN CALL Questionablestatus
     IF (STB% AND 128) > 0 THEN CALL Operationstatus
     IF (STB% AND 32)  > 0 THEN CALL Esrread
  END IF
noDevice:
END SUB                                           'End of SRQ routine
REM ************************************************************************
```

Reading out the status event registers, the output buffer and the error/event queue is effected in subroutines.

Read Out of Output Buffer

```
REM -------- Subroutine for the individual STB bits -----------------------
Public SUB Outputqueue()                        'Reading the output buffer

result$ = SPACE$(100)                           'Make space for response
CALL IBRD(analyzer%, result$)
PRINT "Contents of Output Queue : "; result$
END SUB
REM ************************************************************************
```

Read Out of Error Messages

```
REM -------- Subroutine for reading the error queue ----------------------
Public SUB ErrorQueueHandler()
ERROR$ = SPACE$(100)                    'Make space for error variable
CALL IBWRT(analyzer%, "SYSTEM:ERROR?")
CALL IBRD(analyzer%, ERROR$)
PRINT "Error Description : "; ERROR$
END SUB
REM ************************************************************************
```

Evaluation of SCPI Status Registers

```
REM ------ Subroutine for evaluating Questionable Status-Register ---------
Public SUB Questionablestatus()
Ques$ = SPACE$(20)                  'Preallocate blanks to text variable
CALL IBWRT(analyzer%, "STATus:QUEStionable:EVENt?")
CALL IBRD(analyzer%, Ques$)
PRINT "Questionable Status: "; Ques$
END SUB
REM ************************************************************************

REM ------ subroutine for evaluating Operation Status-Register ------------
Public SUB Operationstatus()
Oper$ = SPACE$(20)                  'Preallocate blanks to text variable
CALL IBWRT(analyzer%, "STATus:OPERation:EVENt?")
CALL IBRD(analyzer%, Oper$)
PRINT "Operation Status: "; Oper$
END SUB
REM ************************************************************************
```

Evaluation of Event Status Register

```
REM ------ Subroutine for evaluating the Event Status Register ------------
Public SUB Esrread()
Esr$ = SPACE$(20)              ’Preallocate blanks to text variable
CALL IBWRT(analyzer%, "*ESR?")                              ’Read ESR
CALL IBRD(analyzer%, Esr$)
IF (VAL(Esr$) AND 1) > 0 THEN PRINT "Operation complete"
IF (VAL(Esr$) AND 2) > 0 THEN PRINT "Request Control"
IF (VAL(Esr$) AND 4) > 0 THEN PRINT "Query Error"
IF (VAL(Esr$) AND 8) > 0 THEN PRINT "Device dependent error"
IF (VAL(Esr$) AND 16) > 0 THEN
    PRINT "Execution Error; Program aborted"’  Output error message
    STOP                                       ’Stop software
    END IF
IF (VAL(Esr$) AND 32) > 0 THEN
    PRINT "Command Error; Program aborted"’    Output error message
    STOP                                       ’Stop software
    END IF
IF (VAL(Esr$) AND 64) > 0 THEN PRINT "User request"
IF (VAL(Esr$) AND 128) > 0 THEN PRINT "Power on"
END SUB
REM ************************************************************************
```

More Complex Programming Examples

Basic settings of the Analyzer

The following settings are an example of how to modify the default setting of FSP.
It should be noted that only some settings are necessary depending on the example of application. In particular, the settings for resolution bandwidth, video bandwidth and sweep time are often not needed since these parameters are automatically calculated in the default setting on modifying the frequency range (span). The insertion loss is also automatically calculated depending on the reference level. The level detectors are coupled to the selected trace mode in the default setting.
The settings which are automatically calculated in the default setting are marked by (*) in the following programming example.

Setting the IEC/IEEE Bus Status Register

```
REM ************************************************************************
Public Sub SetupStatusReg()

'--------- IEEE 488.2 status register -------------------------------------
CALL IBWRT(analyzer%,"*CLS")                ’Reset Status Registers
CALL IBWRT(analyzer%,"*SRE 168")            ’Enable service request
                                            ’for STAT:OPER-,STAT:QUES- and
                                            ’ESR registers
CALL IBWRT(analyzer%,"*ESE 61")             ’Set Event-Enable bit for:
                                            ’Operation Complete
                                            ’Command-, Execution-,Device
                                            ’Dependent- and Query Error
'--------- SCPI status register -----------------------------------------
CALL IBWRT(analyzer%,"STAT:OPER:ENAB 0")    ’Disable OPERation Status Reg
CALL IBWRT(analyzer%,"STAT:QUES:ENAB 0")    ’Disable Questionable Status
                                            ’Register
End Sub
REM ************************************************************************
```

Default Setting for Measurements

```
REM ************************************************************************
Public Sub SetupInstrument()

'-------------------- Basic settings -----------------------------------
CALL SetupStatusReg                          'Set status registers
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")     'ON: display indication on
                                             'OFF: off(improved
                                             '         performance)
CALL IBWRT(analyzer%,"DISP:FORM SINGle")     'Full screen
CALL IBWRT(analyzer%,"DISP:WIND1:SEL")       'Active screen A
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
'--------- Set frequency ------------------------------------------------
CALL IBWRT(analyzer%,"FREQUENCY:CENTER 100MHz")       'Center frequency
CALL IBWRT(analyzer%,"FREQ:SPAN 1 MHz")               'Span
'--------- Set level ----------------------------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV -20dBm")  'Reference level
CALL IBWRT(analyzer%,"INP:ATT 10dB")                  'Input attenuation (*)
'--------- Scale y-axis -------------------------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:SPAC LOG")     'Log level axis
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:SCAL 100dB")   'Level range
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:SCAL:MODE ABS")'Absolute scaling
CALL IBWRT(analyzer%,"CALC:UNIT:POW DBM")             'Unit of y-axis
'--------- Trace and detector settings ----------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC1:MODE AVER")     'Trace1 average
CALL IBWRT(analyzer%,"AVER:TYPE VID")                 'Average mode video;
                                                      '"LIN" for linear
CALL IBWRT(analyzer%,"SWE:COUN 10")                   'Sweep count
CALL IBWRT(analyzer%,"DISP:WIND:TRAC2:STAT OFF")      'Trace2 blank
CALL IBWRT(analyzer%,"DISP:WIND:TRAC3:STAT OFF")      'Trace3 blank
CALL IBWRT(analyzer%,"CALC:MATH:STAT OFF")            'Trace mathematics off
CALL IBWRT(analyzer%,"DETECTOR1 RMS")                 'Detector Trace1   (*)
CALL IBWRT(analyzer%,"DET2:AUTO ON")                  'Detector Trace2   (*)
CALL IBWRT(analyzer%,"DET3:AUTO ON")                  'Detector Trace3   (*)
'--------- Band width and sweep time ------------------------------------
CALL IBWRT(analyzer%,"BAND:RES 100KHz")               'Resolution BW (*)
CALL IBWRT(analyzer%,"BAND:VID 1MHz")                 'Video bandwidth   (*)
CALL IBWRT(analyzer%,"SWE:TIM 100ms")                 'Sweep time        (*)
END SUB
REM ************************************************************************
```

Using Marker and Delta Marker

Marker Search Functions, Limitation of Search Range

The example below is based on an AM-modulated signal at 100 MHz with the following characteristics:

- Carrier signal level: –30 dBm
- AF frequency: 100 kHz
- Modulation depth: 50 %

Marker 1 and delta marker 2 are set one after the other to the highest maxima of the measurement curve and then the frequency and level are read out. The default setting of the analyzer can be used for the following measurements (SetupInstrument).

```
REM ************************************************************************
Public Sub MarkerSearch()


result$ = Space$(100)
CALL SetupInstrument                            'Basic setting
'--------- Peak search without search limit -------------------------------
CALL IBWRT(analyzer%,"INIT:CONT OFF")           'Switch to single sweep
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")      'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")       'Switch on Marker 1
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")        'Assign marker1 to Trace 1
CALL IBWRT(analyzer%,"INIT;*WAI")               'Sweep with sync
CALL IBWRT(analyzer%,"CALC:MARK:MAX;X?;Y?")     'Marker to peak; read out
CALL IBRD(analyzer%, result$)                   'frequency and level
Print "Marker 1: ";result$
CALL IBWRT(analyzer%,"CALC:DELT2:STAT ON;MAX;MAX:LEFT")
                                                'Switch on delta marker 2
                                                'Peak and then Next Peak Left
CALL IBWRT(analyzer%,"CALC:DELT:MODE ABS")      'Delta marker 2 frequeny output
                                                'absolute
CALL IBWRT(analyzer%,"CALC:DELT2:X?;Y?")        'Delta marker 2 - Read out
                                                'frequency and level
CALL IBRD(analyzer%, result$)
Print "Delta 2: ";result$
```

```
'--------- Peak search with search limit in x-direction --------------------
CALL IBWRT(analyzer%,"CALC:DELT:MODE REL")   'Delta marker frequency output
                                             'relative
CALL IBWRT(analyzer%,"CALC:MARK:X:SLIM:STAT ON;LEFT 0Hz;RIGHt 100.05MHz")
                                             'Search limit on and set below
                                             'LF on the right side
CALL IBWRT(analyzer%,"CALC:DELT3:STAT ON;MAX;MAX:RIGHt")
                                             'Delta marker 3      on
                                             'Peak and then Next Peak Right
CALL IBWRT(analyzer%,"CALC:DELT3:X?;Y?")     'Delta marker 3; Read out
                                             'frequency und level, both must

CALL IBRD(analyzer%, result$)                'have the value 0
Print "Delta 3: ";result$

'--------- Peak search with search limit in y-direction --------------------
CALL IBWRT(analyzer%,"CALC:DELT:MODE REL")   'Deltamarker frequency output
                                             'relative
CALL IBWRT(analyzer%,"CALC:THR –35DBM;THR:STAT ON")
                                             'Threshold on and set above LF
CALL IBWRT(analyzer%,"CALC:DELT3:STAT ON;MAX;MAX:NEXT")
                                             'Delta marker 3 on
                                             'Peak and then Next Peak
                                             ' => is not found
CALL IBWRT(analyzer%,"CALC:DELT3:X:REL?;:CALC:DELT:Y?")
CALL IBRD(analyzer%, result$)                'Delta marker 3; Read out
                                             'Frequency und level, both must
CALL IBRD(analyzer%, result$)                'have the value 0
Print "Delta 3: ";result$

'---- Set center frequency and reference level by means of markers ---------
CALL IBWRT(analyzer%,"CALC:MARK2:FUNC:CENT") 'Delta marker 2 -> Marker and
                                             'center frequency = Marker 2
CALL IBWRT(analyzer%,"CALC:MARK2:FUNC:REF")  'Ref level = Marker 2
Call ibwrt(analyzer%,"INIT;*WAI")            'Sweep with sync
END SUB
REM ************************************************************************
```

Frequency Counting

The following example is based on a signal with a level of –30 dBm at 100 MHz. The default setting of the analyzer can also be used for this measurement (SetupInstrument). The objective of frequency counting is to determine the exact frequency of the signal at 100 MHz.

```
REM ************************************************************************
Public Sub MarkerCount()

result$ = Space$(100)
CALL SetupInstrument                              'Basic settings
'--------- Measure signal frequency with frequency counter ----------------
CALL IBWRT(analyzer%,"INIT:CONT OFF")             'Single sweep on
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")        'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")         'Marker 1 on
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")          'Assign marker1 to trace 1
CALL IBWRT(analyzer%,"CALC:MARK:X 100MHz")        'Set Marker1 to 100MHz
CALL IBWRT(analyzer%,"CALC:MARK:COUNT:RES 1HZ")'Frequency counter 1Hz
CALL IBWRT(analyzer%,"CALC:MARK:COUNT ON")        'frequency counter on
CALL IBWRT(analyzer%,"INIT;*WAI")                 'Sweep with sync
CALL IBWRT(analyzer%,"CALC:MARK:COUNT:FREQ?")     'Read out frequency
Print "Marker Count Freq: ";result$
END SUB
REM ************************************************************************
```

Operation mit Fixed Reference Point (Reference Fixed)

The following example is based on a signal with a level of –20 dBm at 100 MHz. The harmonics of the signal lie at 200 MHz, 300 MHz, etc. In the presence of high-quality signal sources these harmonics may lie out of the dynamic range of FSP. In order to measure harmonic suppression, however, the level should be set to higher sensitivity for measuring the harmonics; the carrier has to be suppressed by a notch filter to avoid overloading the analyzer RF input.

In the following example two measurements are therefore performed with different level settings, first with a high reference level at the carrier frequency and then with a low reference level at the frequency of the 3rd harmonic.

The default setting of the analyzer for measurements (SetupInstrument) is used as starting point and adaptations are then made for the measurement.

```
REM ************************************************************************
Public Sub RefFixed()

result$ = Space$(100)
CALL SetupInstrument                            'Basic settings
'--------------- Measure the reference point ------------------------------
CALL IBWRT(analyzer%,"INIT:CONT OFF")           'Single sweep
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")      'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")       'Marker1 on
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")        'Assign marker1 to Trace 1
CALL IBWRT(analyzer%,"INIT;*WAI")               'Sweep with sync
CALL IBWRT(analyzer%,"CALC:MARK:MAX")           'Set Marker1 to 100MHz
CALL IBWRT(analyzer%,"CALC:DELT:FUNC:FIX ON")   'Reference fixed
'-----Setting freq., level and bandw. for meas. of harmonic distortion ----
CALL IBWRT(analyzer%,"FREQ:CENT 400MHz;Span 1MHz" )'Set freq. of 3rd harmonic
CALL IBWRT(analyzer%,"BAND:RES 1kHz")             'and appropriate RBW
CALL IBWRT(analyzer%,"SWEEP:TIME:AUTO ON")        'Couple sweep time
CALL IBWRT(analyzer%,"INP:ATT:AUTO ON")           'Optimize level
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV –50dBm")
CALL IBWRT(analyzer%,"INIT;*WAI")               'Sweep with sync
CALL IBWRT(analyzer%,"CALC:DELT:MODE REL")      'Delta marker frequency
                                                'relative
CALL IBWRT(analyzer%,"CALC:DELT:MAX;X?;Y?")     'Read out delta marker
Call ibrd(analyzer%, result$)                   'Read out frequency and level
Print "Deltamarker 1: "; result$
END SUB
REM ************************************************************************
```

Phase and Phase Noise Measurement

During phase noise measurement the noise power referred to 1 Hz is brought into proportion to the power of an adjacent carrier signal. The spacing often used between the measured frequency and the carrier frequency is 10 kHz.

For the noise measurement the measured absolute level is referred to a bandwith of 1 Hz.

The following example is again based on a signal with a level of –30 dBM at 100 MHz. Two markers are used to determine the noise and the phase noise at an offset of 10 kHz from the carrier signal.

```
REM ************************************************************************
Public Sub Noise()

result$ = Space$(100)
'--------- Basic settings ------------------------------------------------
CALL SetupStatusReg                          'Configure status register
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
'--------- Set frequency -------------------------------------------------
CALL IBWRT(analyzer%,"FREQUENCY:CENTER 100MHz")       'Center frequency
CALL IBWRT(analyzer%,"FREQ:SPAN 100 kHz")             'Span
'--------- Set level -----------------------------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV –20dBm")  'Reference level
CALL IBWRT(analyzer%,"INIT;*WAI")              'Sweep with sync
'--------- Set reference point -------------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")     'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")      'Marker1 on
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")       'Assign marker1 to trace1
CALL IBWRT(analyzer%,"CALC:MARK:MAX")          'Set marker1 to 100MHz
CALL IBWRT(analyzer%,"CALC:DELT:FUNC:PNO ON")  'Phase Noise on
'--------- Measure phase noise -------------------------------------------
CALL IBWRT(analyzer%,"CALC:DELT:X 10kHz")      'Set delta marker
CALL IBWRT(analyzer%,"CALC:DELT:FUNC:PNO:RES?")'Read out result of
Call ibrd(analyzer%, result$)                  'Phase Noise meas.
Print "Phase Noise [dBc/Hz]: "; result$
'--------- Measure noise -------------------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:X 99.96MHz")   'Set Marker 1
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:NOIS:RES?")'Read out result
Call ibrd(analyzer%, result$)
Print "Noise [dBm/Hz]: "; result$
END SUB
REM ************************************************************************
```

Shape Factor Measurement (using n-dB down)

The n-dB-down function of the analyzer is used twice to determine the shape factor of a filter (ratio of bandwidths at 60 dB and 3 dB below the filter maximum).

The following example is again based on a signal with a level of –30 dBm at 100 MHz. The shape factor is determined for the 30 kHz resolution bandwidth. The default setting of the analyzer is used for measurements (SetupInstrument).

```
REM ************************************************************************
Public Sub ShapeFactor()

result$ = Space$(100)
'--------- Basic settings analyzer ----------------------------------------
CALL SetupInstrument                          'Basic settings
CALL IBWRT(analyzer%,"INIT:CONT OFF")         'Single sweep
'--------- Set frequency --------------------------------------------------
CALL IBWRT(analyzer%,"FREQ:SPAN 1MHz")        'Span
CALL IBWRT(analyzer%,"BAND:RES 30kHz")        'Resolution bandwidth
CALL IBWRT(analyzer%,"INIT;*WAI")             'Sweep with sync
'--------- Measure 60 dB value --------------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")    'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")     'Marker1 on
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")      'Assign marker1 to Trace 1
CALL IBWRT(analyzer%,"CALC:MARK:MAX")         'Set marker1 to 100MHz
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:NDBD 60dB")'Bandbreite bei 60dB messen
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:NDBD:RES?")'und auslesen
CALL IBRD(analyzer%,result$)
result60 = Val(result$)
'--------- Measure 3 dB Down value  ---------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:NDBD 3dB")'Read out bandwidth at 60dB
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:NDBD:RES?")
CALL IBRD(analyzer%,result$)
result3 = Val(result$)
'--------- Read out shape factor-------------------------------------------
Print "Shapefaktor 60dB/3dB: ";result60/result3
END SUB
REM ************************************************************************
```

Measuring the Third Order Intercept Point

The intercept point of 3rd order is the (virtual) level of two adjacent useful signals at which the intermodulation products of 3rd order have the same level as the useful signals.

The intermodulation product at f_{S2} is obtained by mixing the first harmonic of useful signal P_{N2} with signal P_{N1}, the intermodulation product at f_{S1} by mixing the first harmonic of useful signal P_{N1} with signal P_{N2}.

$f_{s1} = 2 \times f_{n1} - f_{n2}$ (1)

$f_{s2} = 2 \times f_{n2} - f_{n1}$ (2)

The following example is based on two adjacent signals with a level of –30 dBm at 100 MHz and 110 MHz. The intermodulation products lie at 90 MHz and 120 MHz according to the above formula. The frequency set is in a way that the examined mixture products are displayed on the diagram. Otherwise the default setting of the analyzer is used for measurements (SetupInstrument).

```
REM ************************************************************************
Public Sub TOI()

result$ = Space$(100)
'--------- Basic settings ------------------------------------------------
CALL SetupStatusReg                          'Set status registers
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")     'ON: display indication on
                                             'OFF: off
'--------- Set frequency -------------------------------------------------
CALL IBWRT(analyzer%,"FREQ:STARt 85MHz;STOP 125 MHz") 'Span
'--------- Set level -----------------------------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV –20dBm")  'Reference level
CALL IBWRT(analyzer%,"INIT;*WAI")              'Sweep with sync
'--------- TOI measurement -----------------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")     'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:TOI ON")  'TOI on
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:TOI:RES?")'and read out results
CALL IBRD(analyzer%,result$)
'--------- Read out result -----------------------------------------------
Print "TOI [dBm]: ";result$
END SUB
REM ************************************************************************
```

Measuring the AM Modulation Depth

The example below is based on an AM-modulated signal at 100 MHz with the following characteristics:

- Carrier signal level: –30 dBm
- AF frequency: 100 kHz
- Modulation depth: 50 %

The default setting of the analyzer for measurements can be used for the measurements described below (SetupInstrument).

```
REM ************************************************************************
Public Sub AMMod()

result$ = Space$(100)
CALL SetupInstrument                          'Basic settings
'--------- Peak search ----------------------------------------------------
CALL IBWRT(analyzer%,"INIT:CONT OFF")         'Single sweep
CALL IBWRT(analyzer%,"INIT;*WAI")             'Sweep with sync
CALL IBWRT(analyzer%,"CALC:MARK:PEXC 6DB")    'Peak Excursion
CALL IBWRT(analyzer%,"CALC:MARK:STAT ON")     'Marker 1 on
CALL IBWRT(analyzer%,"CALC:MARK:TRAC 1")      'Assign marker1 to trace 1
'--------- Measure modulation depth----------------------------------------
CALL IBWRT(analyzer%,"CALC:MARK:MAX;FUNC:MDEP ON") 'Marker to Peak;
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:MDEP:RES?")   'Mod. Depth on
CALL IBRD(analyzer%, result$)                      'Read out result
'--------- Read out result ------------------------------------------------
Print "AM Mod Depth [%]: ";result$
END SUB
REM ************************************************************************
```

Limit Lines and Limit Test

The example below shows the definition and use of a new limit line 5 for trace 1 on screen A and trace 2 on screen B with the following characteristics:

- Upper limit line
- Absolute x axis in the frequency range
- 5 reference values: 120 MHz / -70 DB, 126 MHz/-40 dB, 127 MHz/-40 dB, 128 MHz/-10 dB, 129 MHz/-40 dB, 130 MHz/-40 dB, 136 MHz / - 70 dB
- Relative y axis with unit dB
- Absolute threshold at -75 dBm
- No margin

The signal of the integrated calibration source (128 MHz, -30 dBm) is used to check the limit test.

```
REM ************************************************************************
Public Sub LimitLine()

result$ = Space$(100)
'--------- Basic settings -----------------------------------------------
CALL SetupInstrument                                  'Basic settings
CALL IBWRT(analyzer%,"FREQUENCY:CENTER 128MHz;Span 10MHz")'Span
Call ibwrt(analyzer%,"Diag:Serv:Inp Cal;CSO -30dBm")  'Cal signal on
'--------- Definition of limit lines ------------------------------------
CALL IBWRT(analyzer%,"CALC:LIM5:NAME 'TEST1'")        'Define name
CALL IBWRT(analyzer%,"CALC:LIM5:COMM 'Upper limit'")  'Define comment
CALL IBWRT(analyzer%,"CALC1:LIM5:TRAC 1")      'Assign trace in screen A
CALL IBWRT(analyzer%,"CALC2:LIM5:TRAC 2")      'Assign trace in screen B
CALL IBWRT(analyzer%,"CALC:LIM5:CONT:DOM FREQ")'Define x-axis range
CALL IBWRT(analyzer%,"CALC:LIM5:CONT:MODE ABS")'Define x-axis scaling
CALL IBWRT(analyzer%,"CALC:LIM5:UNIT DB")      'Define y-axis unit
CALL IBWRT(analyzer%,"CALC:LIM5:UPP:MODE REL") 'Define y-axis scaling
'--------- Definition of data points and threshold ------------------------
xlimit$ = "CALC:LIM5:CONT 120MHZ,126MHZ,127MHZ,128MHZ,129MHZ,130MHz,136MHz"
CALL IBWRT(analyzer%, xlimit$)                 'Set values for x-axis
CALL IBWRT(analyzer%,"CALC:LIM5:UPP –70,-40,-40,-20,-40,-40,-70")
                                               'Set values for y-axis


CALL IBWRT(analyzer%,"CALC:LIM5:UPP:THR -75DBM") 'Set y-threshold (only
                                                  'possible for relative
                                                  'y-axis)
'------------------------------------------------------------------------
'A margin or an x-/y offset can be defined here.
'----------- Activate and evaluate the limit line in screen A ------------
CALL IBWRT(analyzer%,"CALC1:LIM5:UPP:STAT ON") 'Activate line 5 in screen A
CALL IBWRT(analyzer%,"CALC1:LIM5:STAT ON")     'Activate limit check in
                                               'screen A
CALL IBWRT(analyzer%,"INIT;*WAI")              'Sweep with sync
```

```
CALL IBWRT(analyzer%,"CALC1:LIM5:FAIL?")       'Query result of limit
                                               'check
CALL IBRD(analyzer%, result$)                  'Result: 1 (= FAIL)
'--------- Read out result ------------------------------------------------
Print "Limit Result Line 5: ";result$

'------ Evaluate limit line in screen A by means of status registers -------
CALL IBWRT(analyzer%,"*CLS")                  'Reset status register
'--------- Measure -------------------------------------------------------
CALL IBWRT(analyzer%,"INIT;*OPC")             'Sweep with sync
CALL WaitSRQ(boardID%,status%)                'Wait for service request
'--------- Read out result ------------------------------------------------
IF (status% = 1) THEN
   CALL IBWRT(analyzer%,"STAT:QUES:LIM1:COND?") 'Read out STAT:QUES:LIMit
   CALL IBRD(analyzer%, result$)                'register
   IF ((Val(result$) And 16) <> 0) THEN
      Print "Limit5 failed"
   ELSE
      Print "Limit5 passed"
   END IF
END IF
END SUB
REM ***************************************************************************
```

Measuring the Channel and Adjacent Channel Power

In the following example the channel and adjacent channel power is first measured on a signal with a level of 0 dBm at 800 MHz to IS95. Then the channel and adjacent channel power is measured on a GSM signal at 935.2 MHz with fast ACP measurement (FAST ACP).

In addition, the limit test is activated.

```
REM ************************************************************************
Public Sub ACP()

result$ = Space$(100)
'--------- Basic settings ------------------------------------------------
CALL SetupStatusReg                          'Set status register
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")     'ON: display indication on
                                             'OFF: off
'--------- Set frequency -------------------------------------------------
CALL IBWRT(analyzer%,"FREQ:CENT 800MHz")       'Set frequency
'--------- Set level -----------------------------------------------------
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV 10dBm")   'Reference level

'--------- Example 1: Configure CP/ACP for CDMA  -------------------------
CALL IBWRT(analyzer%,"CALC2:MARK:FUNC:POW:SEL ACP")   'ACP measurement on
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:POW:PRES F8CDMA")'Select CDMA800 FWD
CALL IBWRT(analyzer%,"SENS:POW:ACH:ACP 2")     'Select 2 adjacent channels
CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES ACP")  'Optimize settings
CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES:RLEV") 'Optimize reference level
CALL IBWRT(analyzer%,"SENS:POW:ACH:MODE ABS")  'Absolute measurement
CALL IBWRT(analyzer%,"SENS:POW:HSP ON")        'Fast ACP measurement
'--------- Measure and query result --------------------------------------
CALL IBWRT(analyzer%,"INIT;*WAI")              'Sweep with sync
CALL IBWRT(analyzer%,"CALC2:MARK:FUNC:POW:RES? ACP")  'Query result
CALL IBRD(analyzer%, result$)
'--------- Read out result -----------------------------------------------
Print "Result (CP, ACP low, ACP up, Alt low, Alt up): "
Print result$
```

```
'--------- Example 2: Configure CP/ACP manually für GSM  ------------------

result$ = Space$(100)

CALL IBWRT(analyzer%,"FREQ:CENT 935.2MHz")     ’Set frequency

CALL IBWRT(analyzer%,"CALC:MARK:FUNC:POW:SEL ACP")  ’ACP measurement on

CALL IBWRT(analyzer%,"SENS:POW:ACH:ACP 1")          ’1 adjacent channel

CALL IBWRT(analyzer%,"SENS:POW:ACH:BAND 200KHZ")    ’Channel band width
CALL IBWRT(analyzer%,"SENS:POW:ACH:BAND:ACH 200KHZ")’Adjacent channel bandw.
CALL IBWRT(analyzer%,"SENS:POW:ACH:SPAC 200KHZ")    ’Channel spacing

CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES ACP")  ’Optimize settings
CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES:RLEV") ’Optimize reference level

CALL IBWRT(analyzer%,"SENS:POW:ACH:MODE ABS")  ’Absolute measurement
’--------- Start measurement and query result -----------------------------

CALL IBWRT(analyzer%,"INIT;*WAI")              ’Sweep with sync

CALL IBWRT(analyzer%,"CALC:MARK:FUNC:POW:RES? ACP")   ’Query result
CALL IBRD(analyzer%, result$)

'--------- Read out result ------------------------------------------------

Print "Result (CP, ACP low, ACP up): "
Print result$

'--------- Active limit check ---------------------------------------------

result$ = Space$(100)

CALL IBWRT(analyzer%,"CALC:LIM:ACP:ACH 30DB, 30DB")   ’Set relative limit

CALL IBWRT(analyzer%,"CALC:LIM:ACP:ACH:ABS –35DBM,-35DBM")
                                                      'Set absolute limit

CALL IBWRT(analyzer%,"CALC:LIM:ACP:ACH:STAT ON")      'Rel. limit check on
CALL IBWRT(analyzer%,"CALC:LIM:ACP:ACH:ABS:STAT ON")  'Abs. limit check on
CALL IBWRT(analyzer%,"CALC:LIM:ACP ON")               'Limit check on

’--------- Start measurement and query result -----------------------------

CALL IBWRT(analyzer%,"INIT;*WAI")                     'Sweep with sync

CALL IBWRT(analyzer%,"CALC:LIM:ACP:ACH:RES?")         'Query result of
CALL IBRD(analyzer%, result$)                         'limit check

’--------- Read out result ------------------------------------------------

Print "Result Limit Check: ";result$

END SUB

REM ************************************************************************
```

Occupied Bandwidth Measurement

In the following example, the bandwidth is to be found in which 95% of the power of a GSM signal is contained. Signal frequency is 935,2 MHz; channel bandwidth is 200 kHz.

```
REM ************************************************************************
Public Sub OBW()

result$ = Space$(100)
'--------- Basic settings ------------------------------------------------
CALL SetupStatusReg                          'Set status register
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")     'ON: display indication on
                                             'OFF: off
'--------- Configure analyzer for OBW for GSM  ---------------------------
CALL IBWRT(analyzer%,"FREQ:CENT 935.2MHz")          'Set frequency
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:POW:SEL OBW")  'OBW measurement on
CALL IBWRT(analyzer%,"SENS:POW:ACH:BAND 200KHZ")    'Channel bandwidth
CALL IBWRT(analyzer%,"SENS:POW:BWID 95PCT")    'Percentage of power
CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES OBW")  'Set frequency and
CALL IBWRT(analyzer%,"SENS:POW:ACH:PRES:RLEV") 'optimize level settings
CALL IBWRT(analyzer%,"SENS:POW:NCOR OFF")      'Noise correction
'--------- Measure and query result --------------------------------------
CALL IBWRT(analyzer%,"INIT;*WAI")              'Sweep with sync
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:POW:RES? OBW")   'Query result
CALL IBRD(analyzer%, result$)
Print result$
END SUB
REM ************************************************************************
```

Time Domain Power Measurement

In the following example the mean carrier power of a signal with 300 kHz bandwith at 100 MHz is to be determined. In addition, the peak power, the rms value and the standard deviation are measured. To do this, the time-domain-power measurement functions are used.

```
REM ************************************************************************
Public Sub TimeDomainPower()

result$ = Space$(100)
'--------- Basic settings ------------------------------------------------
CALL SetupStatusReg                          'Set status register
CALL IBWRT(analyzer%,"*RST")                 'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")     'ON: display indication on
                                             'OFF: off
'--------- Configure analyzer for time domain power measurement ------------
CALL IBWRT(analyzer%,"FREQ:CENT 100MHz;SPAN 0Hz")   'Set frequency
CALL IBWRT(analyzer%,"BAND:RES 300kHz")             'Resolution bandwidth
CALL IBWRT(analyzer%,"SWE:TIME 200US")              'Sweep time
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:SUMM:PPE ON")  'Pos. Peak measurement
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:SUMM:MEAN ON") 'Mean measurement
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:SUMM:RMS ON")  'RMS measurement
CALL IBWRT(analyzer%,"CALC:MARK:FUNC:SUMM:SDEV ON") 'Standard deviation
'------------------- Measure and query results ----------------------------
CALL IBWRT(analyzer%,"INIT;*WAI")                   'Sweep with sync
                                                    'Query results:
query$ =          " CALC:MARK:FUNC:SUMM:PPE:RES?;"  'Pos. Peak measurement
query$ = query$ + ":CALC:MARK:FUNC:SUMM:MEAN:RES?;" 'Mean measurement
query$ = query$ + ":CALC:MARK:FUNC:SUMM:RMS:RES?;"  'RMS measurement
query$ = query$ + ":CALC:MARK:FUNC:SUMM:SDEV:RES?"  'Standard deviation
Call IBWRT(analyzer%, query$)
CALL IBRD(analyzer%, result$)
Print result$
END SUB
REM ************************************************************************
```

Reading Trace Data

In the following example the trace data recorded together at the default setting are read out of the instrument and displayed on the screen in the form of a list. Reading is performed consecutively in the binary format and in the ASCII format, at span > 0 and also at span = 0.

In the binary format the message header is evaluated with the length indication and used to calculate the x axis values.

In the ASCII format only the list of level values is output.

The binary data are read out in 3 steps:

1. Reading the number of digits of the length indication
2. Reading the length indication
3. Reading trace data

This procedure is necessary with programming languages that support only structures with similar data types (arrays) (such as Visual Basic) since the data types of header and data differ in binary data.

```
REM ************************************************************************
Public Sub ReadTrace()

'--------- Define variables -------------------------------------------
Dim traceData(1002) As Single               'Buffer for floating point
                                            'binary data
Dim digits As Byte                          'Number of digits of
                                            'length indication
Dim traceBytes As Integer                   'Length of trace data in bytes
Dim traceValues As Integer                  'Number of values in buffer

asciiResult$ = Space$(10000)                'Buffer for ASCII trace data

result$ = Space$(100)                       'Buffer for simple results

startFreq$ = Space$(100)                    'Buffer for start frequency

span$ = Space$(100)                         'Buffer for span
'--------- Basic settings ---------------------------------------------
CALL SetupInstrument                        'Basic settings

CALL IBWRT(analyzer%,"INIT:CONT OFF")       'Single sweep

CALL IBWRT(analyzer%,"INIT;*WAI")           'Sweep with sync
'--------- Define span for read out -----------------------------------
Call ibwrt(analyzer%,"FREQ:STARt?")         'Read out start frequency
Call ibrd(analyzer%,startFreq$)
startFreq = Val(startFreq$)

Call ibwrt(analyzer%,"FREQ:SPAN?")          'Read out span
Call ibrd(analyzer%,span$)
span = Val(span$)
```

```
'--------- Read out in binary format -------------------------------------
Call ibwrt(analyzer%, "FORMAT REAL,32")      'Select binary format
Call ibwrt(analyzer%, "TRAC1? TRACE1")       'Read out Trace 1
Call ilrd(analyzer%, result$, 2)             'Read out and store
digits = Val(Mid$(result$, 2, 1))            'number of digits of
                                             'length indication
result$ = Space$(100)                        'Initialize buffer again
Call ilrd(analyzer%, result$, digits)        'Read out
traceBytes = Val(Left$(result$, digits))     'and store indication of length
Call ibrd32(analyzer%, traceData(0), traceBytes)'Read trace data into buffer
Call ilrd(analyzer%, result$, 1)             'Read the terminator <NL>
'--------- Read out binary data as pairs of frequency/level values ---------
traceValues = traceBytes/4                   'Single precision = 4 bytes
stepsize = span/traceValues                  'Calculate frequeny step width
For i = 0 To traceValues – 1
  Print "Value["; i; "] = "; startFreq+stepsize*i; ", "; traceData(i)
Next i
'--------- Basic settings time domain -----------------------------------
Call ibwrt(analyzer%,"FREQ:SPAN 0Hz")        'Zero span on
CALL IBWRT(analyzer%,"INIT;*WAI")            'Sweep with sync
'--------- Read out in ASCII format -------------------------------------
Call ibwrt(analyzer%,"FORMAT ASCII")         'Select ASCII format
CALL ibwrt(analyzer%,"TRAC1? TRACE1")        'Read out Trace 1
CALL ibrd(analyzer%, asciiResult$)
Print "Contents of Trace1: ",asciiResult$
END SUB
REM ************************************************************************
```

Storing and Loading Device Settings

Storing Instrument Settings

In the following example the settings/measurement data to be stored are determined; only the hardware settings are stored. The selection commands for the other settings are indicated with the status OFF for the sake of completeness.

```
REM ************************************************************************
Public Sub StoreSettings()


'This subroutine selects the settings to be stored and creates
'the data set "TEST1" in directory D:\USER\DATA. It uses
'the default setting and resets the instrument after storage
'of the setting.
'--------- Basic settings ------------------------------------------------
Call SetupInstrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")        'Single sweep
CALL IBWRT(analyzer%,"INIT;*WAI")            'Sweep with sync
'--------- Select items to store -----------------------------------------
CALL IBWRT(analyzer%,"MMEM:SEL:HWS ON")      'Select hardware settings to
                                             'store
CALL IBWRT(analyzer%,"MMEM:SEL:TRAC OFF")    'Disable traces to store
CALL IBWRT(analyzer%,"MMEM:SEL:LIN:ALL OFF") 'Select active limit lines to
                                             'store
'--------- Define comment ------------------------------------------------
CALL IBWRT(analyzer%,"MMEM:COMM 'Test Setup'")
'--------- Store selected items ------------------------------------------
CALL IBWRT(analyzer%,"MMEM:STOR:STAT 1,'D:\USER\DATA\TEST1'")
'--------- Reset instrument ----------------------------------------------
CALL IBWRT(analyzer%,"*RST")
END SUB
REM ************************************************************************
```

Loading Device Settings

In the following example data set "TEST1" stored under D:\USER\DATA is reloaded into the isnstrument:

```
REM ************************************************************************
Public Sub LoadSettings()

'This subroutine loads data set "TEST1" in directory D:\USER\DATA.
'--------- Set status registers ----------------------------------------
Call SetupStatusReg                          'Set status register
'--------- Load data set -----------------------------------------------
CALL IBWRT(analyzer%,"MMEM:LOAD:STAT 1,'D:\USER\DATA\TEST1'")
'----Start measurement with the settings of the loaded data set ------------
CALL IBWRT(analyzer%,"DISP:TRAC1:MODE WRITE")  'Set trace to Clr/Write
CALL IBWRT(analyzer%,"INIT;*WAI")              'Start the sweep
END SUB
REM ************************************************************************
```

Setting the Data Set for Startup Recall

In the following example the analyzer is first reset. Then the data set TEST1 stored under D:\USER\DATA is selected for the function STARTUP RECALL, ie the data set is set for every *RST, PRESET and every device startup. For illustration, the command *RST is executed again.

```
REM ************************************************************************
Public Sub StartupRecallSettings()
'--------- Reset analyzer ----------------------------------------------
CALL IBWRT(analyzer%,"*RST")
'--------- Set status registers ----------------------------------------
Call SetupStatusReg                          'Set status register
'--------- Select startup recall data set-------------------------------
CALL IBWRT(analyzer%,"MMEM:LOAD:AUTO 1,'D:\USER\DATA\TEST1'")
'--------- Activate startup recall data set ----------------------------
CALL IBWRT(analyzer%,"*RST")
END SUB
REM ************************************************************************
```

Reading and Writing Files

Reading a File from the Instrument

In the following example file TEST1.SET stored under D:\USER\DATA is read from the instrument and stored in the controller.

```
REM ************************************************************************
Public Sub ReadFile()
'--------- Variables -----------------------------------------------------
Dim digits As Byte                          'Number of digits of
                                            'length indication
Dim fileBytes As Long                       'Length of file with trace data
                                            'in bytes
result$ = Space$(100)                       'Buffer for simple results
'--------- Basic settings status registers -------------------------------
Call SetupStatusReg                         'Set status register
'--------- Read out file -------------------------------------------------
Call ibwrt(analyzer%, "MMEM:DATA? 'D:\USER\DATA\TEST1.SET'")
                                            'Select file
Call ilrd(analyzer%, result$, 2)            'Read and store number of
                                            'digits in length
digits = Val(Mid$(result$, 2, 1))           'indication
Call ilrd(analyzer%, result$, digits)       'Read and store length
fileBytes = Val(Left$(result$, digits))     'indication
FileBuffer$ = Space$(fileBytes)             'Buffer for file
Call ilrd(analyzer%, FileBuffer, fileBytes) 'Read file into buffer
Call ilrd(analyzer%, result$, 1)            'Read terminator <NL>
'--------- Store file to controller --------------------------------------
Open "TEST1.SET" For Output As #1
Print #1, FileBuffer;                       ' ; to avoid linefeed at
                                            '   end of file
Close #1
END SUB
REM ************************************************************************
```

Creating a File in the Instrument

In the following example file TEST1.SET available on the controller is stored in the instrument under D:\USER\DATA\DUPLICAT.SET.

```
REM ************************************************************************

Public Sub WriteFile()

'--------- Variables -----------------------------------------------------

FileBuffer$ = Space$(100000)                ’Buffer for file
Dim digits As Long                          ’Number of digits of
                                            ’length indication
Dim fileBytes As Long                       ’Length of file in bytes
fileSize$ = Space$(100)                     ’Length of file in a string

result$ = Space$(100)                       ’Buffer for simple results

'--------- Basic settings status registers -------------------------------

Call SetupStatusReg                         ’Set status register

'--------- Prepare the definite length block data ------------------------

fileBytes = FileLen("H:\work\vb\test1.set")  ’Determine length of file
fileSize$ = Str$(fileBytes)

digits = Len(fileSize$) – 1                 'Determine number of digits of
fileSize$ = Right$(fileSize$, digits)       'length indication

FileBuffer$ = "#" + Right$(Str$(digits), 1) + fileSize$
                                            'Put length indication into
                                            'file buffer

'--------- Read file from controller -------------------------------------

Open "H:\work\vb\TEST1.SET" For Binary As #1

FileBuffer$ = FileBuffer$ + Left$(Input(fileBytes, #1), fileBytes)

Close #1

'--------- Write file ----------------------------------------------------
-

Call ibwrt(analyzer%, "SYST:COMM:GPIB:RTER EOI")      'Set receive
                                                      'terminator in the
                                                      'instrument

Call ibwrt(analyzer%, "MMEM:DATA 'D:\USER\DATA\DUPLICAT.SET'," +
           FileBuffer$)                               'Select file

END SUB

REM ************************************************************************
```

Configuring and Starting a Printout

The following example shows the configuration of the output format and output device for printing out a measurement mask.
The procedure is in the following order:

1. Set the measurement required for the printout
2. Query available output devices
3. Select an output device
4. Select the output interface
5. Configure the output format
6. Start printout with synchronization to the end

It is assumed that the setting required is a signal with a power of –20 dBm at 100 MHz and also that the printer required is the 6th of the available printers. The printout is first performed to the selected printer, then to a file.

```
REM ************************************************************************
Public Sub HCopy()


DIM Devices(100) as string                        'Buffer for printer name
FOR i = 0 TO 49
  Devices$(i) = Space$(50)                       'Preallocate buffer for
                                                 'printer name
NEXT i
'--------- Basic settings -------------------------------------------------
CALL SetupStatusReg                             'Set status register
CALL IBWRT(analyzer%,"*RST")                    'Reset instrument
CALL IBWRT(analyzer%,"INIT:CONT OFF")           'Single sweep
CALL IBWRT(analyzer%,"SYST:DISP:UPD ON")        'Display indication on
'--------- Configure measurement ------------------------------------------
CALL IBWRT(analyzer%,"FREQ:CENT 100MHz;SPAN 10MHz")   'Set frequency
CALL IBWRT(analyzer%,"DISP:WIND:TRAC:Y:RLEV -10dBm")  'Reference level
CALL IBWRT(analyzer%,"INIT;*WAI")                     'Start measurement
'--------- Query available output devices ---------------------------------
CALL IBWRT(analyzer%,"SYST:COMM:PRIN:ENUM:FIRSt?")    'Read out first
CALL IBRD(analyzer%,Devices$(0))                      'printer and indicate
PRINT "Drucker 0: "+Devices$(0)                       'printer name
For i = 1 to 99
  CALL IBWRT(analyzer%,"SYST:COMM:PRIN:ENUM:NEXT?")   'Read out the next
  CALL IBRD(analyzer%,Devices$(i))                    'printer name
  IF Left$(Devices$(i),2) = "''" THEN GOTO SelectDevice  'Abort at end of
                                                         'list
  PRINT "Drucker"+Str$(i)+": " Devices$(i)            'Indicate printer name
NEXT i
```

```
SelectDevice:
'------------ Select device, printer language and interface ----------------
CALL IBWRT(analyzer%,"SYST:COMM:PRIN:SEL "+ Devices(6))'Select printer #6
8 CALL IBWRT(analyzer%,"HCOP:DEST 'SYST:COMM:PRIN'")  'Configuration:
                                                      '"Print out to
                                                      'printer interface"
CALL IBWRT(analyzer%,"HCOP:DEV:LANG GDI")           'Output language 'GDI'
'----- Select orientation (portrait/landscape) and color/BW ---------------
CALL IBWRT(analyzer%,"HCOP:PAGE:ORI PORTrait")      'Portrait
CALL IBWRT(analyzer%,"HCOP:DEV:COL OFF")            'Black and white
'----- Configure and start print out --------------------------------------
 CALL IBWRT (analyzer%,"HCOP:ITEM:ALL")       'Select complete screen
'CALL IBWRT (analyzer%,"HCOP:ITEM:WIND1:TRAC:STAT ON")  'alternative: only
'CALL IBWRT (analyzer%,"HCOP:ITEM:WIND2:TRAC:STAT ON")  'traces in
                                                       'screen A/B
CALL IBWRT (analyzer%,"*CLS")                  'Reset status registers
CALL IBWRT (analyzer%,"HCOP:IMMediate;*OPC")   'Start print out
CALL WaitSRQ(boardID%,result%)                 'Wait for service request
IF (result% = 1) THEN CALL Srq                 'If SRQ is recognized =>
                                               'Subroutines for evaluation
'---- Print out into file in WMF format (BMP format) ----------------------
CALL IBWRT(analyzer%,"HCOP:DEST 'MMEM'")       'Configuration:
                                               '"Print to file"
 CALL IBWRT(analyzer%,"HCOP:DEV:LANG WMF")     'File format WMF
'CALL IBWRT(analyzer%,"HCOP:DEV:LANG BMP")     'File format BMP
CALL IBWRT(analyzer%,"MMEM:NAME 'D:\USER\DATA\PRINT1.WMF'") 'Determine
                                                            'file name
CALL IBWRT (analyzer%,"*CLS")                  'Reset status registers
CALL IBWRT (analyzer%,"HCOP:IMMediate;*OPC")   'Start print out
CALL WaitSRQ(boardID%,result%)                 'Wait for service request
IF (result% = 1) THEN CALL Srq                 'If SRQ is recognized  =>
                                               'Subroutines for evaluation
END SUB
REM ************************************************************************
```