Annex F (Datagram Communication)

General

By contrast to TCP/IP data transmission, the datagram communication serves for higher performance in making trace data available. With a view to this, data are made available in a self-descriptive form. This should enable the evaluation of data by host applications without SCPI data interface. The required protocol is described in this section. The various data types are identified by means of tags and are provided with length information. Therefore, host applications can filter out and process specific data and need not implement the whole protocol. Also see "Program example UDP" in Annex D.

Addressing

The "**UdpPath**" concept is intended for the distribution of data. A UdpPath contains an IP address, a specific port number and configuration data. UdpPath is not the equivalent of a host since several hosts can be addressed simultaneously via the IP address (using broadcast / multicast addressing) and a host can also serve several UdpPaths (different port numbers with different configurations).

Configuration

The configuration of the various UdpPath entries contains first of all the type of the data. The different trace types can be configured by means of **tags. Flags** can specify the trace data more precisely.

Protocol

Each datagram (Udp packet) contains a **header** that clearly identifies a data item of the protocol. What follows is one or several data units denoted **GenericAttribute** which can be distinguished with the use of tags. A C-type notation is used for protocol specification. All data are represented in the network byte order, in other words in the big endian order. This is especially important for host applications running on PC (Intel) base since in this case data first have to converted to the little endian format.

EB200Datagram {

EB200Header

GenericAttribute_1

GenericAttribute_2

...

GenericAttribute_n

}

Header

The header marks the beginning of each EB200 datagram.

EB200Header {

```
    unsigned long magic_number;              /* constant: 0x000EB200 */
    unsigned short minor_version_number;     /* 0x26 for this version */
    unsigned short major_version_number;     /* 0x02 for this version */
    unsigned short sequence_number;          /* increments by one */
    char[6] reserved;
```

}

Description of EB200 header:

- **magic_number**
  This is a constant value which never changes.
- **minor_version_number** and **major_version_number**
  The minor_version_number is incremented by appending tags or changing the tags in an upward-compatible form. The major_version_number is incremented by the non-compatible change of the tags or general structural changes (the latter will be extremely rare).
- **sequence_number**
  This starts at a certain value and is incremented by one for every new packet per UdpPath. When the highest value is attained, a wraparound takes place.
- **reserved**
  This element is left free for future extensions.

From minor_version_number 0x24, in the DSCAN datagram the additional flag newStepScheme is transmitted with the OptionalHeader. This flag shows whether the old one or the new channel spacing was chosen.

With *minor_version_number 0x26* and higher, a demodulation string is sent with AUDIO.

GenericAttribute

This describes the general structure of every subsequent data element (as GenericAttribute). All data types (ie all trace data are marked by tags) are of the same structure.

GenericAttribute {

unsigned short tag;

unsigned short length;

char data[length];

}

- **tag**
  Determines the content of the GenericAttribute.
- **length**
  Specifies the length of the GenericAttributes in bytes but not of the elements "tag" and "length".
- **data**
  Contains useful data. The length of this data area is determined by the preceding data element "length".

The various tags are now described.

A common structure exists for the tags defined at present (FSCAN, MSCAN, DSCAN, AUDIO, IFPAN, FASTLEVCW, LIST and CW) and will be described below. This structure is contained in the element "data" of the GenericAttribute.

Symbolic tag name	*Numeric tag value (decimal)*
FSCAN	101
MSCAN	201
DSCAN	301
AUDIO	401
IFPAN	501
FASTLEVCW	601
LIST	701
CW	801

Table 1: Description of tags

TraceAttribute

The common structure of all trace data defined up to now is described here.

TraceAttribute {

```
    short number_of_trace_items;          /* number of values per data type in PeriodicTraceData
                                             */
    char reserved;
    unsigned char optional_header_length; /* size of the optional header in bytes */
    unsigned long selectorFlags;          /* exact specification of data */

    OptionalHeader;                       /* this is described accurately for the corresponding
                                             trace*/

    PeriodicTraceData;                    /* actual trace data */
                                          /* depending on SelectorFlags or tag and
                                             OptionalHeader */
```

}

The value in "optional_header_length" should always be referenced to access the actual trace data in the PeriodicTraceData. Upward compatibility is thus ensured even for "minor_version_number" modifications.

The "selectorFlags" are dealt with here in general but are only partly useful when applied to the individual traces.

SelectorFlag	*Hexadecimal value*	*Data type*	*Corresponding flags*	*Note*
LEVEL	0x01	short	"VOLTage:AC"	
OFFSET	0x02	long	"FREQuency:OFFSet"	
FSTRENGTH	0x04	short	"FSTRength"	with EB200FS
AM (only ESMB)	0x04	short	"AM"	only ESMB
AM_POS (only ESMB)	0x08	short	"AM:POSitive"	only ESMB
AM_NEG (only ESMB)	0x10	short	"AM:NEGative"	only ESMB
FM (only ESMB)	0x20	long	"FM"	only ESMB
FM_POS (only ESMB)	0x40	long	"FM:POSitive"	only ESMB
FM_NEG (only ESMB)	0x80	long	"FM:NEGative"	only ESMB
PM (only ESMB)	0x100	short	"PM"	only ESMB
BANDWIDTH (only ESMB)	0x200	long	"BANDwidth"	only ESMB
CHANNEL	0x00010000	unsigned short	"CHANnel",	
FREQUENCY	0x00020000	unsigned long	"FREQuency:RX“	
SWAP	0x20000000		"SWAP"	
SIGNAL_GREATER_SQUELCH	0x40000000		„SQUelch“	
OPTIONAL_HEADER	0x80000000		"OPTional"	

Table 2: Description of SelectorFlags

The SelectorFlags describe the data to be found in the PeriodicTraceData, whether an OptionalHeader has been transmitted and whether the trace data are of "SIGNAL_GREATER_SQUELCH“ data. The sequence of the data in the PeriodicTraceData corresponds to the sequence of the above table, ie depends on the set SelectorFlags.

The flags are determined by the corresponding configuration command "TRAC:UDP:.....", the set sensor functions and whether this setting is at all allowed by the trace type. If all these settings allow a specific data item, the latter is sent in the datagram trace and the corresponding SelectorFlag set.

FScanTrace

All data specified in the SelectorFlags are relevant for this trace type.

Description of the OptionalHeader.

```
OptionalHeader {
    short cycleCount;
    short holdTime;
    short dwellTime;
    short directionUp;
    short stopSignal;
    unsigned long startFrequency;
    unsigned long stopFrequency;
    unsigned long stepFrequency;
}
```

Example of a complete attribute:

```
FScanAttribute {
    short number_of_trace_values;          /* corresponds to the number_of_trace_items */
    char reserved;
    unsigned char optional_header_length;  /* 0 or 22 */
    unsigned long selectorFlags;           /* see Table 2: Description of SelectorFlags */
    OptionalHeader;                        /* as described above of optional_header_length = 22 */

    short level-1;
    short level-2;
    short level-3;
    ....
    short level-number_of_trace_values;

    long offset-1;
    long offset-2;
    long offset-3;
    ....
```

```
    long offset-number_of_trace_values;

    short am-1;
    short am-2;
    short am-3;
    ....
    short am-number_of_trace_values;

    /* etc */
    /* see SelectorFlags table for sequence*/

    ....
    unsigned long frequency-1;
    unsigned long frequency-2;
    unsigned long frequency-3;
    ....
    unsigned long frequency- number_of_trace_values;
}
```

It may well be that in the future elements will be added to the OptionalHeader, more precisely appended. If an application with optional_header_length exceeds the optional header to get to the trace data, there will be no problems for the existing programs (upward compatibility).

MScanTrace

This trace has the same structure as FScanTrace with the exception of the OptionalHeader which do not contain some elements of the FScanTrace-OptionalHeader.

Description of the OptionalHeader.

```
OptionalHeader {
    short cycleCount;
    short holdTime;
    short dwellTime;
    short directionUp;
    short stopSignal;
}
```

The optional_header_length is thus either 0 or 10.

DScanTrace

SelectorFlags for DScanTrace can contain:

LEVEL

FSTRENGTH

SIGNAL_GREATER_SQUELCH

OPTIONAL_HEADER

Description of the OptionalHeader.

```
OptionalHeader {
    unsigned long startFrequency;
    unsigned long stopFrequency;
    unsigned long stepFrequency;
    unsigned long markFrequency;
    short bwZoom;
    short referenceLevel;
    short newStepScheme;        /* from UDP version 0x0224
}
```

From minor_version_number 0x24, in the DSCAN datagram the additional flag newStepScheme is transmitted with the OptionalHeader. This flag shows whether the old one or the new channel spacing was chosen (see also annex J).

AUDio

SelectorFlags for audio data can only contain OPTIONAL_HEADER.

Description of the OptionalHeader.

```
OptionalHeader {
    short audio_mode;                   /* see remote command SYSTem:AUDio:REMote:MODe */
    short frame_len;                    /* specifies the number of bytes per frame */
    unsigned long frequency;            /* current receive frequency */
    unsigned long bandwidth;            /* current IF bandwidth */
    unsigned short demodulation;        /* current demodulation type */
    char    sDemodulation[8];           /* demodulation type string */
}
```

The different kinds of demodulation are coded as Enum in the EB200 as follows :

FM	0
AM	1
PULS	2
CW	3
USB	4
LSB	5
IQ	6

With *minor_version_number 0x26* and higher, a demodulation string is sent with AUDIO.

Example of a complete attribute:

```
AudioAttribute {
    short number_of_frames;             /* corresponds to the number_of_trace_items */
    char reserved;
    unsigned char optional_header_length;
    unsigned long selectorFlags;

    OptionalHeader; /                   * as described above */

    unsigned char data[frame_len * number_of_frames];
```

The AF data packets are transmitted internally in a cycle of 30 ms. The size of the UDP packets is between 325 bytes and 4 kbytes depending on the audio_mode.

Each UDP packet contains several complete frames. The definition of the audio_mode is given in the remote command SYSTem:AUDio:REMote:MODe.

In the PCM modes (audio_mode 1 to 12) a frame contains one or two channels and each channel is 16-bit wide. Thus each frame contains 1, 2 or 4 bytes, depending on the configuration.

In the GSM 6.10 mode (audio_mode 13) a frame contains 65 bytes. This 520-bit long frame contains two 260-bit long GSM subframes. The meaning of the bits can be looked in the ETSI standard (www.etsi.org).

Byte:

0	1							32						63	64
First ETSI / GSM 6.10 subframe								Second ETSI / GSM 6.10 subframe							

Allocation of bits and bytes in a frame:

		Bit designation corresponding to ETSI	**Bits in frame**	**Bytes in frame**
1st Subframe	LSB	1	0	0
		2	1	0
		3	2	0
		4	3	0
		5	4	0
		6	5	0
		7	6	0
		8	7	0
		9	0	1
		10	1	1
		11	2	1
		.	.	.
		.	.	.
		.	.	.
		254	5	31
		255	6	31
		256	7	31
		257	0	32
		258	1	32
		259	2	32
	MSB	260	3	32
2nd subframe	LSB	1	4	32
		2	5	32
		3	6	32
		4	7	32
		5	0.	33.
		.	.	.
		.	.	.
		257	4	64
		258	5	64
		259	6	64
	MSB	260	7	64

Table 3: Description of GSM data format

The SelectorFlag "SWAP" does not have any effect on the configuration of the bytes in the GSM 6.10 frame.

IFPan

SelectorFlags for IF Panorama can contain:

LEVEL

OPTIONAL_HEADER

Description of the OptionalHeader.

```
OptionalHeader {
    unsigned long frequency;
    unsigned long spanFrequency;
    short averageTime;
    short averageType;
}
```

FASTLEVCW

SelectorFlags for this mode must contain LEVEL:

There is no OPTIONAL_HEADER.

LIST

SelectorFlags for this mode must contain LEVEL:

There is no OPTIONAL_HEADER.

CW

With this trace type, as with FscanTrace, all data specified in the SelectorFlags are relevant.

Description of OptionalHeader:

```
OptionalHeader {
    unsigned long Frequency;
}
```

Remote commands

There is a fixed number of maximum configurable UdpPath entries. The first entry is always the default entry. This entry is saved in a CMOS RAM and retained on powering off and on again. **This means that scans initiated as a result of power-off (and subsequent power-on) can produce datagram packets without the user 'noticing' it.**

A UdpPath is always made up of the IP address (as string) and port number (as integer):

eg "192.168.1.1", 18457

Byte order of data to be transmitted:

The default byte order is Big Endian (native byte order of the device) which also corresponds to the network byte order used eg in the TCP/IP protocol. Using the "SWAP" flag described below this behaviour can be altered. If this flag is set the wanted data are transmitted in Little-Endian order. This applies to the various OptionalHeaders as well as to the PeriodicTraceData, but not to the data "above" them which are used as protocol. The format of the GSM 6.10 audio data also remains unaffected by the SWAP flag.

Registration of tags for a specific UdpPath:

TRACe:UDP:TAG[:ON] IP-ADDR, PORT-NUM, tag [, tag ..]

TRACe:UDP:DEFault:TAG[:ON] IP-ADDR, PORT-NUM, tag [, tag ..]

The first command registers a specific UdpPath, the second one the default UdpPath (index 0).

Possible tags:

FSCan, MSCan, DSCan, AUDio, IFPan, FASTlevcw, LIST, CW

Example:

TRAC:UDP:DEF:TAG "89.10.20.30", 17222, FSC, MSC

Registration of flags for a specific UdpPath:

TRACe:UDP:FLAG[:ON] IP-ADDR, PORT-NUM, flag [, flag ..]

TRACe:UDP:DEFault:FLAG[:ON] IP-ADDR, PORT-NUM, flag [, flag ..]

The first command registers a specific UdpPath, the second one the default UdpPath (index 0).

Possible flags	*Data output*	*Remark*
"VOLTage:AC"	Level	
"FREQuency:OFFSet"	Offset	
"FSTRength"	Field strength	with EB200FS
"AM"	AM modulation depth	only ESMB
"AM:POSitive"	AM – positive modulation depth	only ESMB
"AM:NEGative"	AM - negative modulation depth	only ESMB
"FM"	Frequency deviation	only ESMB
"FM:POSitive"	Positive frequency deviation	only ESMB
"FM:NEGative"	Negative frequency deviation	only ESMB
"PM"	Phase deviation	only ESMB
"BANDwidth"	Bandwidth	only ESMB
"CHANnel",	Channel number	
"FREQuency:RX“	Frequency	
"SWAP"	in Little-Endian order	
„SQUelch“	Only level values above squelch threshold	
"OPTional"	Additional OptionalHeader	

Table 4: Description of flags

Note:
The sensor function “FSTRength“ only provides results if the SW option EB200FS is fitted. Also see Annex H.

Note:
The flags are registered independent of the tags.

Example:

TRAC:UDP:FLAG "89.255.255.255", 18457, "AM", "AM:POSitive", "AM:NEGative", "OPT"

De-registration of tags for a specific path UdpPath:

TRACe:UDP:TAG:OFF IP-ADDR, PORT-NUM, tag [, tag ..]
TRACe:UDP:DEFault:TAG:OFF IP-ADDR, PORT-NUM, tag [, tag ..]

Tags, same as for registration.

Example:
TRAC:UDP:DEF:TAG:OFF "89.10.20.30", 17222, FSC

De-registration of flags for a specific UdpPath:

TRACe:UDP:FLAG:OFF IP-ADDR, PORT-NUM, flag [, flag ..]
TRACe:UDP:DEFault:FLAG:OFF IP-ADDR, PORT-NUM, flag [, flag ..]
Flags, same as for registration.

Example:
TRAC:UDP:FLAG:OFF "89.255.255.255", 18457, "OPT"

Delete of a UdpPath:

TRACe:UDP:DELete IP-ADDR, PORT-NUM

This command deletes a UdpPath from the list provided it can be found. The default UdpPath can be deleted in this way too.

Example:

TRACE:UDP:DELETE "89.255.255.255", 18457

Delete of all UdpPath:

TRACe:UDP:DELete ALL

This command deletes all UdpPath.

Example:

TRACE:UDP:DELETE ALL

Query of UdpPath:

TRACe:UDP? MINimum | MAXimum | DEFault

Query of the index of the first UdpPath, the highest UdpPath and the default UdpPath.

TRACe:UDP? <numeric_value>

Query of the UdpPath with the index <numeric_value>

Example:

TRAC:UDP? MAX -> 3
TRAC:UDP? 0 -> DEF "89.10.20.30", 18457, FSC, MSC, "VOLT:AC", "OPT"
TRAC:UDP? 3 -> 003 "255.255.255.255", 17222, DSC, "VOLT:AC", "OPT"