Annexe F (Communication par datagrammes)

Généralités

Le but de la communication par datagrammes et de mettre à disposition les données de trace de façon plus performante qu'en transmission TCP/IP. Les données sont mises à disposition sous forme auto-descriptive. Même des applications hôtes sans interface de données SCPI doivent ainsi pouvoir exploiter les données. Le protocole nécessaire à cet effet sera décrit dans ce chapitre. Les différents types de données sont identifiés par des balises ou "tags" et dotés d'une information de longueur. Les applications hôtes peuvent ainsi filtrer et traiter les données qui les intéressent et n'ont pas nécessairement besoin de mettre en œuvre la totalité du protocole. Voir aussi l'annexe C "Exemple de programme UDP".

Adressage

La distribution des données a conduit à l'élaboration du concept "**UdpPath**". Un tel UdpPath contient une adresse IP, un certain numéro de port et des données de configuration. UdpPath ne saurait donc être assimilé à un hôte ou "Host", car, d'une part, il est possible de s'adresser à plusieurs hôtes en même temps via l'adresse IP (par adressage du type Broadcast ou Multicast) et, d'autre part, un hôte peut desservir plusieurs UdpPaths (différents numéros de port avec des configurations différentes).

Configuration

La configuration des différents UdpPaths comprend premièrement le type des données. Les différents types de trace peuvent être configurés par différentes balises ou **tags**. Différents indicateurs ou **flags** permettent en outre de spécifier plus précisément les données de trace désirées.

Protocole

Chaque datagramme (paquet Udp) possède un en-tête ou **header**, qui indique qu'il s'agit bien de données de ce protocole. Viennent ensuite une ou plusieurs unités de données, désignées par **GenericAttribute**, qui peuvent se distinguer par différents tags. La spécification du protocole fait appel à une notation analogue à celle de C. Normalement, les données sont transmises dans le Network-Byte-Order, c'est-à-dire dans le Big-Endian-Order. Ceci est particulièrement important pour les applications hôtes tournant sur PC (Intel), car, dans ce cas, il faut qu'une conversion ait lieu au format Little-Endian. Les données utiles proprement dites peuvent toutefois être aussi transmises dans le Little-Endian-Order (voir plus loin la description des instructions de télécommande).

EB200Datagram {

    EB200Header

    GenericAttribute_1

    GenericAttribute_2

    ...

    GenericAttribute_n

}

Header

Le header est l'en-tête introduisant tout datagramme EB200.

EB200Header {

```
    unsigned long magic_number;              /* constant: 0x000EB200 (pour ESMB aussi) */
    unsigned short minor_version_number;     /* 0x24 pour cette version */
    unsigned short major_version_number;     /* 0x02 pour cette version */
    unsigned short sequence_number;          /* incrémenté de un */
    char[6] reserved;
}
```

Description de l'EB200Header :

- **magic_number**
  C'est une constante qui ne changera jamais.
- **minor_version_number** et **major_version_number**
  L'addition de tags ou la modification de tags sous une forme à compatibilité ascendante incrémentera le minor_version_number. En cas de modifications non compatibles de tags ou de modifications structurelles générales, c'est le major_version_number qui sera incrémenté (ce dernier cas, s'il se présente, sera très rare).
- **sequence_number**
  Ce nombre part d'une certaine valeur et est incrémenté de un à chaque nouveau paquet d'un UdpPath. Arrivé à la valeur maximale, il y a un bouclage ou wrap-around.
- **reserved**
  Cet élément est réservé à de futures extensions éventuelles.

GenericAttribute

Cet attribut générique décrit la structure générale de chacun des éléments de données qui suivent. Tous les types de données (c'est-à-dire toutes les données de trace caractérisées par les différents tags) possèdent la même structure.

GenericAttribute {

unsigned short tag;

unsigned short length;

char data[length];

}

- **tag**
  Détermine le contenu proprement dit de ce GenericAttribute.
- **length**
  Indique la longueur de ce GenericAttribute en octets, à l'exclusion des éléments "tag" et "length".
- **data**
  C'est ici que viennent se ranger les données utiles proprement dites. La longueur de cette zone de données est déterminée par l'élément précédent length".

Ci-dessous la description des différents tags.

Les tags actuellement définis (FSCAN, MSCAN, DSCAN, AUDIO, IFPAN, FASTLEVCW, LIST et CW) ont une structure de base commune, qui sera d'abord décrite. Il s'agit de la structure qui vient se ranger dans l'élément "data" du GenericAttribute.

Nom symbolique du TAG	*Valeur numérique du TAG (en décimal)*
FSCAN	101
MSCAN	201
DSCAN	301
AUDIO	401
IFPAN	501
FASTLEVCW	601
LIST	701
CW	801

Tableau 1 : Description des TAGs

TraceAttribute

La structure générale de toutes les données de trace définies jusqu'ici sera décrite ici.

TraceAttribute {

```
  short number_of_trace_items;          /* Nombre de valeurs par type de données dans
                                           PeriodicTraceData */
  char reserved;
  unsigned char optional_header_length;/* Taille de l'Optional Header en octets */
  unsigned long selectorFlags;          /* Spécification plus précise des données */

  OptionalHeader;                       /* Celui-ci sera décrit plus précisément pour les traces
                                           adéquates */

  PeriodicTraceData;                    /* Les données de trace proprement dites */
                                        /* selon SelectorFlags ou Tag et OptionalHeader */
```

}

Il convient de toujours faire appel à la valeur figurant dans "optional_header_length" pour accéder aux données de trace proprement dites dans PeriodicTraceData, afin de garantir la compatibilité ascendante même en cas de modifications du "minor_version_number".

Les "SelectorFlags" seront ici décrits de manière générale, mais n'ont pas toujours de sens pour les différentes traces.

SelectorFlag	*Valeur en hexadécimal*	*Type de données*	*Flags correspondants*	*Remarque*
LEVEL	0x01	short	"VOLTage:AC"	
OFFSET	0x02	long	"FREQuency:OFFSet"	
FSTRENGTH	0x04	short	"FSTRength"	avec EB200FS
AM	0x08	short	"AM"	ESMB seul
AM_POS	0x10	short	"AM:POSitive"	ESMB seul
AM_NEG	0x20	short	"AM:NEGative"	ESMB seul
FM	0x40	long	"FM"	ESMB seul
FM_POS	0x80	long	"FM:POSitive"	ESMB seul
FM_NEG	0x100	long	"FM:NEGative"	ESMB seul
PM	0x200	short	"PM"	ESMB seul
BANDWIDTH	0x400	long	"BANDwidth"	ESMB seul
CHANNEL	0x00010000	unsigned short	"CHANnel",	
FREQUENCY	0x00020000	unsigned long	"FREQuency:RX“	
SWAP	0x20000000		"SWAP"	
SIGNAL_GREATER_SQUELCH	0x40000000	-	„SQUelch“	
OPTIONAL_HEADER	0x80000000	-	"OPTional"	

Tableau 2 : Description des SelectorFlags

Ces SelectorFlags décrivent quelles sont les données que l'on peut trouver dans PeriodicTraceData, si un OptionalHeader a été transmis et si les données de trace sont des données à "SIGNAL_GREATER_SQUELCH“. Si le SelectorFlag SWAP est positionné, les données utiles sont transmises dans le Little-Endian-Order. L'ordre des données dans les PeriodicTraceData correspond à l'ordre adopté dans le tableau ci-dessus, compte tenu des SelectorFlags positionnés.

Ces flags sont conditionnés, premièrement, par les instructions de configuration correspondantes "TRAC:UDP:.....", deuxièmement, par les fonctions de détection momentanément réglées et, troisièmement, par la question de savoir si le type de trace considéré autorise ce réglage. Si tous ces réglages permettent un certain type de données, celles-ci sont envoyées dans la trace du datagramme, avec positionnement du SelectorFlag correspondant.

FScanTrace

Dans ce type de trace, toutes les données spécifiées dans les SelectorFlags sont pertinentes.

Description de l'OptionalHeader :

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

Exemple d'un attribut complet :

```
FScanAttribute {
    short number_of_trace_values;          /* correspond à number_of_trace_items */
    char reserved;
    unsigned char optional_header_length;  /* 0 ou 22 */
    unsigned long selectorFlags;           /* voir Tableau 2 : Description des SelectorFlags */
    OptionalHeader;                        /* comme décrit ci-dessus si 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. etc. */
    /* Pour l'ordre, voir tableau des SelectorFlags */

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

Il est possible qu'à l'avenir, des éléments soient ajoutés à l'OptionalHeader, plus précisément à l'arrière. Dans la mesure où une application utilise optional_header_length pour sauter précisément ce dernier afin d'accéder aux données de trace, il n'en résultera aucun problème pour les programmes existants (compatibilité ascendante).

MScanTrace

Cette trace a la même structure que la FscanTrace, à l'exception de l'OptionalHeader, qui ne contient pas certains éléments de l'OptionalHeader de la FScanTrace.

Description de l'OptionalHeader :

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

Par analogie, optional_header_length est soit 0, soit 10.

DScanTrace

Les SelectorFlags pour DScanTrace peuvent contenir :

LEVEL

FSTRENGTH

SIGNAL_GREATER_SQELCH

OPTIONAL_HEADER

Description de l'OptionalHeader :

```
OptionalHeader {
    unsigned long startFrequency;
    unsigned long stopFrequency;
    unsigned long stepFrequency;
    unsigned long markFrequency;
    short bwZoom;
    short referenceLevel;
}
```

AUDio

Les SelectorFlags für pour données BF ne peuvent contenir que OPTIONAL_HEADER.

Description de l'OptionalHeader :

```
OptionalHeader {
    short audio_mode;                   /* voir instruction de télécommande
                                           SYSTem:AUDio:REMote:MODe */
    short frame_len;                    /* décrit le nombre d'octets par trame */
    unsigned long frequency;            /* fréquence de réception momentanée */
    unsigned long bandwidth;            /* largeur de bande FI momentanée */
    unsigned short demodulation;        /* type de démodulation momentané */
}
```

Exemple d'un attribut complet :

```
AudioAttribute {
    short number_of_frames;             /* correspond à number_of_trace_items */
    char reserved;
    unsigned char optional_header_length;
    unsigned long selectorFlags;

    OptionalHeader;                     /* comme décrit ci-dessus */

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

Les paquets de données BF sont transmis en interne avec un cycle de 30 ms. La taille des paquets UDP Pakete est comprise, suivant audio_mode, entre 325 octets et 4 koctets.

Chaque paquet UDP contient plusieurs trames complètes. La définition de l'audio_mode est décrite à l'instruction de télécommande SYSTem:AUDio:REMote:MODe.

Dans les modes PCM (audio_mode 1 à 12), une trame contient une ou deux voies, et chaque voie à une largeur de 8 ou 16 bits. Un trame contient ainsi, suivant la configuration, 1, 2 ou 4 octets.

Octet :

0	1						32						63	64
Première sous-trame ETSI / GSM 6.10							Deuxième sous-trame ETSI / GSM 6.10							

Disposition des bits et octets dans une trame :

		Désignation des bits selon l'ETSI	**Bits dans la trame**	**Octets dans la trame**
1ère sous-trame	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
2ème sous-trame	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

Tableau 3 : Description du format de données GSM

Le SelectorFlag „SWAP“ n'a pas d'effet sur la disposition des octets dans la trame GSM 6.10

.

IFPan

Les SelectorFlags pour analyse panoramique FI doivent contenir :

LEVEL

OPTIONAL_HEADER

Description de l'OptionalHeader :

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

FASTLEVCW

Les SelectorFlags pour ce mode doivent contenir LEVEL.

Il n'existe pas d'OPTIONAL_HEADER.

LIST

Les SelectorFlags pour ce mode doivent contenir LEVEL.

Il n'existe pas d'OPTIONAL_HEADER.

CW

Dans ce type de trace, toutes les données spécifiées dans les SelectorFlags sont pertinentes, comme pour FScanTrace.

Description de l'OptionalHeader :

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

Instructions de télécommande

Le nombre maximal d'entrées UdpPath configurables est fixe. La première entrée représente toujours l'entrée par défaut. Cette entrée est sauvegardée dans la CMOS RAM et est conservée même après un cycle de commutation arrêt/marche. **Cela signifie que des balayages lancés même après un arrêt (suivi d'une remise en marche) peuvent produire des paquets de datagramme "sans qu'on le remarque".**

Un UdpPath se compose toujours d'une adresse IP (sous forme de chaîne de caractères) et d'un numéro de port (sous forme d'entier) :

Exemple : "192.168.1.1", 18457

Ordre des octets des données à transmettre :

Le Default Byte-Order est Big-Endian (ordre des octets natif de l'appareil), qui correspond également au Network-Byte-Order, utilisé, par exemple, dans le protocole TCP/IP. Le flag "SWAP" décrit plus loin permet de modifier ce comportement. Quand ce flag est positionné, les données utiles sont transmises dans le Little-Endian-Order. Ceci s'applique aussi bien aux divers OptionalHeaders qu'aux Periodic-TraceData, mais pas aux données "situées au-dessus", car celles-ci ont le caractère de protocole. Le format des données BF GSM 6.10 n'est pas non plus affecté par le flag SWAP.

Enregistrement de TAGs pour un UdpPath donné :

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

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

La première forme enregistre un UdpPath quelconque, la seconde le Default UdpPath (indice 0).

Tags possibles :

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

Exemple :

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

Enregistrement de FLAGs pour un UdpPath donné :

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

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

La première forme enregistre un Flag quelconque, la seconde le Default Flag (indice 0).

Flags possibles	*Données émises*	*Remarque*
"VOLTage:AC"	Niveau	
"FREQuency:OFFSet"	Décalage	
"FSTRength"	Champ	avec EB200FS
"AM"	Indice de modulation AM	ESMB seul
"AM:POSitive"	Indice de modulation positif AM	ESMB seul
"AM:NEGative"	Indice de modulation négatif AM	ESMB seul
"FM"	Excursion de fréquence	ESMB seul
"FM:POSitive"	Excursion de fréquence positive	ESMB seul
"FM:NEGative"	Excursion de fréquence négative	ESMB seul
"PM"	Excursion de phase	ESMB seul
"BANDwidth"	Largeur de bande	ESMB seul
"CHANnel",	Numéro de canal	
"FREQuency:RX“	Fréquence	
"SWAP"	Dans le Little-Endian-Order	
„SQUelch“	Uniquement valeurs de niveau supérieures au seuil du silencieux	
"OPTional"	Optional Header additionnel	

Tableau 4 : Description des flags

Nota :
La fonction de détection “FSTRength“ ne fournit de résultats que si l’appareil est équipé de l’option logicielle EB200FS. Voir aussi l’annexe H.

Remarque : Les flags sont enregistrés indépendamment des tags.

Exemple :

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

Désenregistrement de TAGs pour un UdpPath donné :

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

Tags comme pour l'enregistrement.

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

Désenregistrement de FLAGs pour un UdpPath donné :

TRACe:UDP:FLAG:OFF IP-ADDR, PORT-NUM, flag [, flag ..]
TRACe:UDP:DEFault:FLAG:OFF IP-ADDR, PORT-NUM, flag [, flag ..]
Flags comme pour l'enregistrement.

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

Effacement d'un UdpPath :

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

Efface un UdpPath de la liste, dans la mesure où il peut être trouvé. Le Default UdpPath peut, lui aussi, être effacé de cette manière.

Exemple :

TRACE:UDP:DELETE "89.255.255.255", 18457

Effacement de tous les UdpPaths :

TRACe:UDP:DELete ALL

Efface tous les UdpPaths.

Exemple :

TRACE:UDP:DELETE ALL

Interrogation des UdpPaths :

TRACe:UDP? MINimum | MAXimum | DEFault

Interrogation demandant l'indice du premier UdpPath, du dernier UdpPath et du Default UdpPath.

TRACe:UDP? <numeric_value>

Interrogation de l'UdpPath ayant l'indice <numeric_value>

Exemple :

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"