Document Number
xxxx.xxx.00.1

Topic
Due to the further development of existing and planned test tools the header used by the test interface has to be renewed.

Table of Contents

0 Document control

0.1 References

[TCGEN]	tcgen_prj_sheet.doc
[FRAME]	8434.100.02.001, Januar 04, 2002, Frame Users Guide (frame_users_guide.doc)
[TAP]	8415.028.99.301, Januar 15, 2002, TCC - Test Case Control
[PCO]	8415.090.00.002, May 15, 2001, PCO2 – Tracing Environment (pco_userguide.doc)

0.2 Abbreviations

ACI	Application Control Interface (AT Commands)
G23 Stack	The Condat implementation of Layers 2 and 3 of the GSM Protocol
G23 Target System	Hardware which executes G23
LCD	Liquid Crystal Display
MM	Mobility Management
MOC	Mobile Originated Call
MTC	Mobile Terminated Call
PC	Personal Computer
PCO	Point of Control and Observation
PIN	Personal Identification Number
RS232	Serial Communication Standard
TAP	Test Application Process
Target System	Shortened form of 'G23 Target System'
TCGEN	Test Case Generator

0.3 Terms

Entity	Program which executes the functions of a layer
Message	A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and

	infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.
Primitive	A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters.
Service Access Point	A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

1 Brief Problem Description

The test interface is the denomination of the connection between a PS and test tools (like [PCO], [TAP],...). On both sides a dedicated FRAME entity (TST) is responsible to send/receive all kind of primitives (protocol primitives, system primitives, traces – the single word *primitive* is further on used as a pseudonym of all these). It uses a hierarchy of drivers from which the TIF-driver handles the proprietary format of the primitives – that is a header and corresponding data. There are three possible cases as illustrated in the following pictures:

Bytes	1	4	4	4	4	<Size>-8
Content	T	<Time>	<Size> (number of bytes after this field)	<Sender name>	<Receiver name>	<Trace data>

-> traces (17 bytes header)

Bytes	1	4	4	4	4	<Size>-8
Content	S	<Time>	<Size> (number of bytes after this field)	<Sender name>	<Receiver name>	<system command>

-> system primitives (17 bytes header)

Bytes	1	4	4	4	4	4	<Size>-12
Content	P	<Time>	<Size> (number of bytes after this field)	<Sender name>	<Receiver name>	<OPC>	<primitive data>

-> protocol primitives (21 bytes header)

<Time>, <Size> and <OPC> are represented in ASCII format. See [FRAME] for more details.

Due to the further development of existing and planned test tools (like [TCGEN]) this header has to be renewed. Essentials are the introduction of an <Original receiver>-field, the enlargement of the <Time>-field and new fields to support segmentation of primitives larger than 255 byte (since the TI multiplexer currently doesn't support bigger ones). The <Original receiver>-field and bigger time values are already supported by the other headers used inside the FRAME.

2 Proposed Solution

One main precondition of an introduction of a new TST-header format was backwards compatibility. Since it will be possible (see details below) to distinguish between an old and a new header by looking at the first byte this will be fulfilled.

Together with the introduction of the requested enhancements some major changes will be performed:

- no more ASCII representation of numeric values will be used (at least to save space and transmission time) -> the byte order of the binary values will be in Intel-style (little endian)
- no more pre-interpretation of the format inside the TR driver (to make TIF and TR really independend)
- the size will be the first element after the <Info>-byte to support the TR driver implementation

The following pictures show the proposed format in detail:

Bytes	1	2	4	4	4	<Size>-12
Content	<Info> (has to be none of {TISIP})	<Size> (number of bytes after this field)	<Time>	<Sender name>	<Receiver name>	<trace data I system command>

-> traces or system primitives (15 bytes header)

Bytes	1	2	4	4	4	4	4	<Size>-20
Content	<Info> (has to be none of {TISIP})	<Size> (number of bytes after this field)	<Time>	<Sender name>	<Receiver name>	<Original receiver name>	<OPC>	<primitive data>

-> protocol primitives (23 bytes header)

The <Info>-byte is interpreted bit wise:

```
|   |   |   |   |   |   |   |   |
|___|___|___|___|___|___|___|___|
| version |  type   |time type|reserved |
| 10- now | 01 - P  | 01- ms  | 00- now |
| 00-later| 10 - T  | 10- TDMA|         |
| 11-later| 11 - S  |         |         |
```

That means primitives are still sent in one part – at least from the view of TR. If the TI-Multiplexer capabilities are not increased bigger primitives have to be segmented and reassembled by the drivers below – TI_TRC and SOCKET (in TI_MODE).

Concerning the traces another enhancement shall be considered. The first bytes should give informations about the used trace class, the compression state a.s.o. The detailed format has to be defined.

The main advantages of the new format are summarised:

- support of new features requested by test tools
- general support of 32bit OPC-s (the formerly introduced type Q can be discarded)
- support of very large protocol primitives (up to approximately 64kb)
- support of version information which will ease the handling of later changes (of course a version-switch should only be implemented on tool side for performance reasons)
- backward compatibility. The new header is the default now, all new builds using GPF Release 13 or higher are using the new format. Pre-release 12 stacks in conjunction with new tools can be used with an optional "newheader" argument of tst.exe. The argument can also be send to an already running instance of tst.exe. The new "oldheader" argument is the reverse. If a wrong format is used, stack or tool side change format automatically, skipping a received, malformated primitive. Of course, the automatic adaptation can't be performed by an old stack receiving a primitive with NEW header (or an old tst.exe, used by accident in conjunction with a new tst.exe).

Furthermore it can be stated that the header overhead was not magnificently increased (even reduced for traces and system primitives).

3 Estimated Efforts

The following table contains the currently identified tasks and estimated minimal efforts:

Task	**Who**	**Hours**	**Hours finished**
Concept and specification	MP,FR, RME,HSC, RK et al.	3	√
Getting used to TST driver implementation	RME	10	
TR adaptation	RME		
TIF adaptation	RME		

Evtl. TI_TRC/SOCKET adaptation (segmentation)	RME		
Testing	MP,RME,RK et al.		
Sum		-	