Dialogic® SS7 Protocols User Part Example (UPE) User Guide

Document Reference U26SSS

Publication Date: October 2007

Document Number: U26SSS, Issue 3

Revision History

Issue	Date	Description
1	04-Oct-01	Initial release
2	16-Jun-03	Branding changed: Septel PCI now SPCI4/SPCI2S and SPCI2S, SPCI4S and CPM8 now CPM8.
3	01-Oct-07	Changed copyright to Dialogic Corporation. Remove reference to PCCS6, include non-circuit protocol configuration commands in config.txt and add support for SIGTRAN M2PA links

Contents

1 Introduction

The Dialogic® User Part Example (UPE) is part of the Dialogic® SS7 Development Package. UPE is an example application program designed to demonstrate interfacing to the Dialogic® MTP3 module.

This user guide is intended for users who choose to develop their own applications that will interface with and use the functionality provided by the MTP3 module.

1.1 Software requirements

The UPE application requires the following software:

1. Dialogic® SS7 Development Package
2. Dialogic® User Part Development Package
3. For TDM-based configurations:
    - ss7.dc3 or ss7.dc4 codefile
    - Dialogic® MTP3 host binary, as required
4. For SIGTRAN-based configurations:
    - Dialogic® M2PA and MTP3, host binaries, as required

Software can be downloaded from
http://www.dialogic.com/support/helpweb/signaling/software3.htm

2 UPE Application

UPE receives an MTP-TRANSFER Indication message on a signaling link and sends it back as an MTP-TRANSFER Request message to the originating point code.

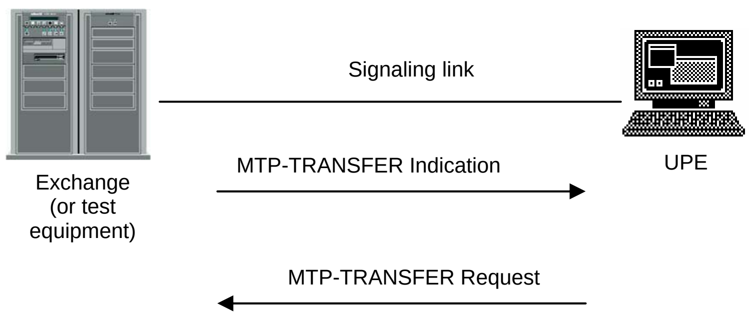

Figure 1. UPE Network Architecture

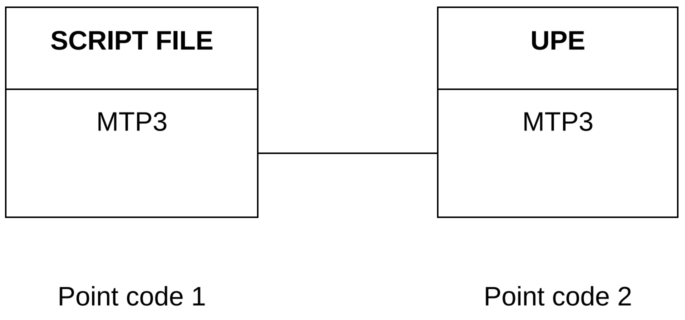

Figure 2. Typical configuration

Dialogic® UPE application is the MTP3 test utility that supports messages from the MTP i.e. MTP-PAUSE, MTP-RESUME MTP-STATUS and MTP-TRANSFER.

Using command line options and depending on configuration, UPE can act in a “data echo” mode and respond to any valid MTP-TRANSFER indications that it receives with MTP-TRANSFER requests towards the originating point. Refer to Section 5 for details.

2.1 Message Sequence Chart

The following diagram shows a message sequence chart of a typical message flow between UPE and the network.

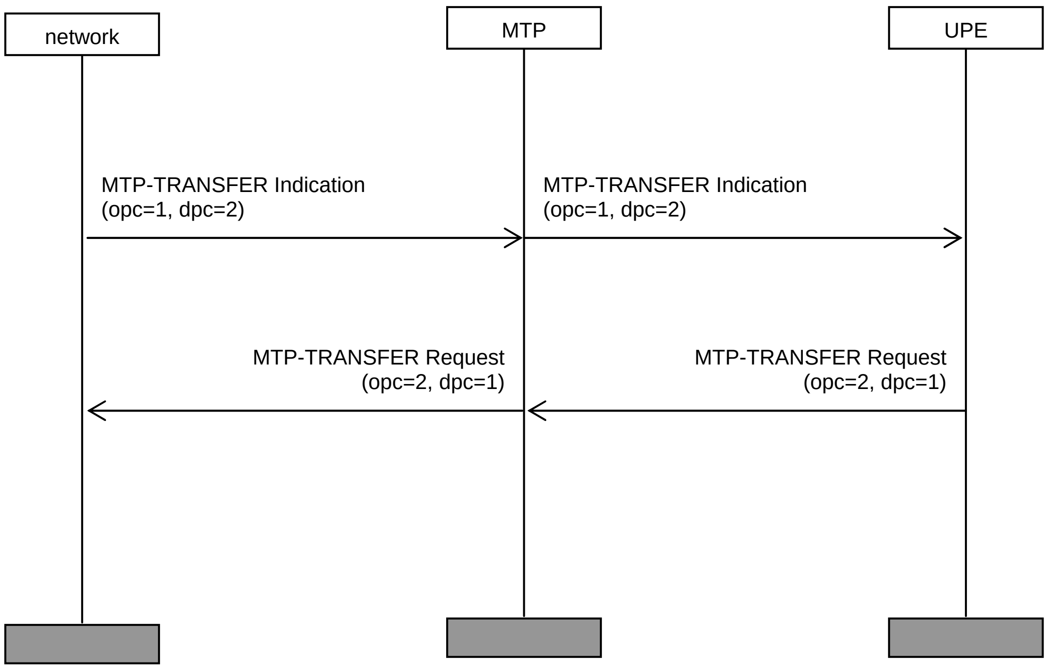

Figure 3. Typical message flow

2.2 Customizing the UPE application

UPE is an example program for development using the MTP3 module and as such, a number of simplifications and limitations have been imposed for this purpose.

UPE supports a shorter version of the MTP-STATUS primitive and may indicate “congestion” or “user part unavailable”. The unavailability cause parameter is not supported.

Depending on configuration, UPE will exchange the point codes for the MTP-TRANSFER indications and MTP-TRANSFER requests and send MTP-TRANSFER requests to the originating point.

UPE can be extended by the user to meet additional requirements. The developer should be aware of the limited nature of the example applications when making use of the source code for building their own application.

2.3 UPE source code

The UPE program can be found in the *Dialogic® User Part Development Package.* The following table describes the files required by the UPE application:

File	**Notes**
upe_main.c	This file contains the **main()** function. This parses the command line arguments and passes them to **upe_ent()**.
upe.c	The function **upe_ent()** contains the main control loop for the application. It waits for a message to be received from the MTP and calls the appropriate function e.g. **UPE_mtp_status()** for handling. For each of the received MTP indications, the parameters within the message are recovered and then the received data is handled according to the message type.

3 Building the UPE application

Example make-files for the following operating systems are provided and identified by a unique suffix:

Operating system	Suffix
Generic UNIX (Solaris, Linux)	.mak
Windows®	.mnt

A single definitions file (one for each operating system) which contains the definitions relating to the user's own development environment is supplied in the *Dialogic® User Part Development Package.* The definitions files are as follows, and the appropriate file should be used depending on the operating system:

makdefs.mak	(Linux)
makdefs_sol.mak	(Solaris)
makdefs.mnt	(Windows®)

For Windows®, a dynamically linked GCT library that allows the application to link to the GCT functions is supplied in the *Dialogic® SS7 Development Package* as follows:

gctlib.dll	(Visual C++® compiler)

For 'UNIX', a GCT shared object is supplied in the *Dialogic® SS7 Development Package*

e.g.	libgctlib.so.1.0.0	(Linux & Solaris)

The source code for the example program should be compiled and linked with the appropriate library for the operating system in use.

3.1 Host software directory structure

To build the MTU application, the user should first ensure that the required files are copied into the correct directories as follows:

1. Copy either the zip or tar file from the *Dialogic® User Part Development Package* to the *Dialogic® SS7 Development Package* directory and decompress using the appropriate tool. The choice of the zip or tar file is up to the user; both will create the UPD directory structure shown in the table below. The table below shows files required by the UPE program only.
2. The C header files in the INC directory shown in the table below The C header files in the INC directory shown in the table below list the header files required by the INTU program.

   The following table lists the directory structure and files required to build the INTU programs supplied on the *Dialogic® User Part Development Package*.

Root directory			
Septel			
INC	**UPD**		
msg.h	**BIN**	**SRC**	
pack.h	**BACKUP_WIN**	**UPE**	makdefs.mnt
ss7_inc.h	**BACKUP_LNX**	upe.mnt	makdefs_sol.mak
strtonum.h	**BACKUP_SOL**	upe.mak	makdefs.mak
sysgct.h		upe_sol.mak	makeall.bat
system.h		upe.c	makeall
		upe_iss.txt	makeall_sol
		upe_main.c	

3.2 Building UPE

It is assumed that the UPD is extracted in the *Dialogic® SS7 Development Package* directory i.e. for Windows® C:\Septel as shown above.

A script is provided in the SRC directory to build and copy all of the example programs into the UPD\BIN directory. To run this script, change to the SRC directory and type one of the following commands depending on the operating system:

makeall	(Linux)
makeall_sol	(Solaris)
makeall.bat	(Windows®)

A pre-built copy of the UPE application, for each operating system, can be located within the backup subdirectories in the BIN directory.

To build the UPE program, change to the SRC\UPE directory and type one of the following commands depending on the operating system:

```
make –f upe.mak
make -f upe_sol.mak
nmake /f upe.mnt
```

4 Configuration

The local and remote ends of the system need to be configured before the Dialogic® UPE application may be run. Example configuration files are provided in the *Dialogic® User Part Development Package*, and after installation will be stored in the directories as shown in the following table:

Root directory	
RUN	
UPE	
"CONFIG1" config.txt system.txt	**"CONFIG2"** config.txt system.txt

The configuration files in the CONFIG1 (for point code 1) and CONFIG2 (for point code 2) directories should be copied to the appropriate node. Refer to Appendix A - UPE configuration files - for further information.

4.1 System Configuration

The GCT environment is configured using the gctload program and the system.txt file. The basic board configuration along with the Dialogic® MTP3 module is achieved using the config.txt file.

4.1.1 SPCI2S, SPCI4 and SS7HD

The GCT environment is configured using the gctload program and the system.txt file. The basic board configuration along with the MTP layers, links, link sets and routes and the protocol configuration is achieved using the config.txt file. Example configuration files for UPE and the remote end are contained in Appendix A.

When running UPE on a Windows® host system using SPCI4 with MTP3 running on the host, the provided configuration files may be used without any modification.

4.1.2 SS7G2x SIU

System and protocol information is configured using the SIU management module and commands in the config.txt and system.txt files. Further information on this can be obtained from the SIU user manual [2].

Note: These files are not contained in the User Part Development Package.

4.1.3 SIGTRAN M2PA

It is also possible to run a similar configuration as described but with SIGTRAN M2PA links. Refer to the Appendix for details.

4.2 Protocol Configuration

All protocol modules are configured using commands in the config.txt file. The example configuration files given in the appendices will perform the appropriate protocol configuration shown below. If the user wishes to better understand or alter the configuration given, the following sections will be of interest.

Before configuring the protocol modules, it is useful to determine the following information relative to each network entity:

- Local point code
- Remote point code
- Point code format

• MTP3

The local point code is contained in the MTP_LINKSET configuration command message and this should be set to the appropriate value. In addition, configuration commands are required for the link and route.

5 Running the UPE application

Before running the Dialogic® UPE application, the environment must first be initialized and the signaling links brought into service. This may be achieved by using the Dialogic® gctload program and activating the links using the Dialogic® mtpsl utility. Refer to the manual for details as appropriate.

5.1 UPE Command Line Arguments

The UPE module takes a number of run time options, which are summarized below.

Option	Default	Notes
-m	0x2d	UPE module Id
-s	0xa	Service indicator (see 5.1.1)
-c	14	Point code format (see 5.1.2)

Example:

```
upe –m0x2d –c24
```

The above example will set the UPE module id to 0x2d and configure it to use 24-bit point codes.

5.1.1 Service indicator

On receipt of MTP-TRANSFER indications, if the -s (service indicator) option is set, UPE will check the value of this option and determine if it has the same value as the service indicator parameter in the MTP_USER_PART command (in the config.txt file). If both service indicator values match, UPE will exchange the originating and destination point codes received in the MTP-TRANSFER Indication, and send them in the MTP-TRANSFER request to the MTP. If the service indicator values are different, any received MTP-TRANSFER indications for that destination will be discarded.

5.1.2 Point code format

UPE supports the following three point code formats:

-c parameter	Point code format
14	14-bit point code (Default)
16	16-bit point code
24	24-bit point code

6 References

[1] U10SSS, Dialogic® Software Environment Programmer’s Manual

[2] 05-2302, Dialogic® SS7G2x SIU Mode User Manual

[3] 05-2063, Dialogic® SS7HD Programmer’s Manual

[4] U03HSP, Dialogic®SS7 Programmer's Manual for SPCI2S, SPCI4S and CPM8

[5] U04STN, Dialogic® Programmer's Manual for Sigtran Host Software

Updates to the documentation are available on the Dialogic web site at
http://www.dialogic.com/support/helpweb/signaling/default.htm

7 Abbreviations

DPC	Destination point code
MTP	Message Transfer Part
OPC	Originating point code
SIO	Service indicator octet
SSF	Sub-service field
SLS	Signaling link selection

Appendix A - UPE configuration files

This section provides example configuration files (system.txt and config.txt) for use with the Dialogic® UPE application on a Windows® host systems with Dialogic® SPCI boards. Also, changes that may be required in the configuration files are indicated for different operating systems and board configurations.

Before configuring the MTP module it is useful to determine information such as the point code format, the local point code and remote point code relative to each network entity. For this example configuration, the local point code is 2 and the remote point code is 1. In the provided example configuration files, UPE is running on the local point code (see Fig. 5).

Example configuration

Operating system:	Windows®
Board type:	SPCI4
Local point code (UPE):	2
Remote point code:	1
UPE module ID:	0x2d
Module(s):	MTP3 (on host)

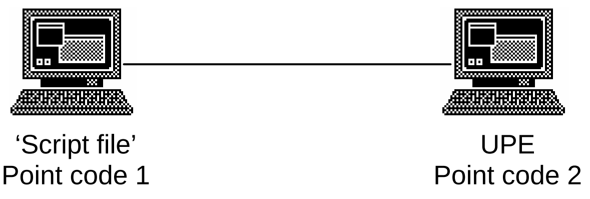

Figure 4. Example configuration

A.1 system.txt

This section provides an example system.txt file for a SPCI4 board running on a Windows® host using the example configuration described earlier in this appendix.

The following example system.txt file is for point code 1 and point code 2. All comments are denoted by '*'.

A.1.1 system.txt

```
****************************************************************************
* Example system.txt.
* Edit this file to reflect your configuration.
****************************************************************************
*
* Essential modules running on host:
*
LOCAL 0x20        * ssd/ssds - Board interface task
LOCAL 0x00        * tim_nt - Timer task
*
* Optional modules running on the host:
*
LOCAL 0xcf        * s7_mgt - Management/config task
LOCAL 0xef        * s7_log - logs messages
LOCAL 0x2d        * MTP3 module
LOCAL 0x2d        * upe - Example user part task
*
* Modules running on the board (all redirected via ssd):
*
REDIRECT    0x71  0x20  * MTP2 module
REDIRECT    0x10  0x20  * CT bus/Clocking control module
REDIRECT    0x8e  0x20  * On-board management module
*
* Redirection of status indications:
*
REDIRECT    0xdf  0xef  * LIU/MTP2 status messages

*
* Now start-up all local tasks:
FORK_PROCESS      ..\..\..\..\ssds.exe -d
FORK_PROCESS      ..\..\..\..\tim_nt.exe
FORK_PROCESS      ..\..\..\..\tick_nt.exe
FORK_PROCESS      ..\..\..\..\s7_mgt.exe -d
FORK_PROCESS      ..\..\..\..\s7_log.exe
FORK_PROCESS      ..\..\..\..\mtp_nt.exe -t
*
```

A.1.2 Using different board configurations and operating systems

The following subsections provide information regarding the system.txt file if using different operating systems or board based configurations.

A.1.2.1 Running UPE with different boards

If using boards types other than SPCI4, a number of commands may differ from the commands in the provided example system.txt file as follows:

If using SSHD boards, the following lines:

```
REDIRECT        0x71    0x20    * MTP2 module
FORK_PROCESS    SSDS.EXE -d
```

should be replaced by:

```
REDIRECT        0x81    0x20    * MTP2 module_id for SP0
REDIRECT        0x91    0x20    * MTP2 module_id for SP1
REDIRECT        0xe1    0x20    * MTP2 module_id for SP2
REDIRECT        0xf1    0x20    * MTP2 module_id for SP3

FORK_PROCESS    SSDH.EXE -d
```

A.1.2.2 Running UPE with host binary

If using a host binary so that the MTP3 module is run on the host instead of the board, a LOCAL declaration should be added to show that the module is running locally on the host and the corresponding REDIRECT command (which redirects messages for that module to the board) should be removed.

Refer to [2], [3] and [4] as appropriate.

A.1.2.3 Running UPE on the SIU host

If using the SIU, additional commands required by the SIU will need to be included. The example system.txt provided in this appendix should not be used.

Refer to [2] for further information.

A.1.2.4 Running UPE with other operating systems

If using operating systems other than Windows®, the names of some of the executable files used in the FORK_PROCESS commands need to be changed.

Refer to [3] and [4] as appropriate.

A.1.2.5 Running UPE with SIGTRAN M2PA

Add the following to ‘Modules running on the host’:

```
LOCAL           0xc2            * MBM - Management task
LOCAL           0xd0            * SCTPD module
LOCAL           0xd1            * SCTP module
LOCAL           0xc1            * M2PA module
```

Make sure these modules are started using the FORK_PROCESS commands as follows::

```
FORK_PROCESS    ..\..\..\..\sctpd.exe
FORK_PROCESS    ..\..\..\..\sctp.exe
FORK_PROCESS    ..\..\..\..\m2pa_nt.exe -t
FORK_PROCESS    ..\..\..\..\mbm.exe -d
```

Refer to manuals [5] for further information as appropriate.

A.2 config.txt

This section provides two example config.txt files for a SPCI4 board running on a Windows® host using the example configuration described earlier in this appendix.

The following example config.txt files are for point code 1 and point code 2. All comments are denoted by '*'.

Using the two example config.txt files (one at each end of the link) will allow messages using MTP3 with 14-bit point codes to be demonstrated. If connecting to other equipment or other configurations, all the parameters in the config.txt file will need to be examined to determine if they are compatible with the configuration at the other end of the link, for example:

- point codes (OPC, DPC)
- point code format
- signaling timeslot
- route

The option field in MTP_CONFIG command will need to be set differently depending on the point code format used. The options available are as follows:

Point code format	**MTP_CONFIG command <option> field value**
14	0x00000000
16	0x00100000
24	0x00000200

The example files provided in this appendix should not be used for the SIU (refer to [2] for further information).

A.2.1 config.txt for point code 1 (remote end)

```
***************************************************************************
* Example config.txt.
* Edit this file to reflect your configuration.
***************************************************************************
*
* SEPTELPCI_BOARD <board_id> <flags> <code_file> <run_mode>
SEPTELPCI_BOARD  0  0x0043  ss7.dc3 MTP2
*
* Configure individual E1/T1 interfaces:
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format>
*            <crc_mode>
*LIU_CONFIG  0  0  5  1  1  1
*
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG  0 0  0x00000000
*
* Define linksets:
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc>
*             <ssf>
MTP_LINKSET  0  2  2  0x0000 1 0x08
*
* Define signaling links:
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink>
*          <stream> <timeslot> <flags>
* (Note: For PCCS6 boards the first LIU port is stream=16
* whilst for SPCI2S, SPCI4S and CPM8 / PCI boards the first LIU port is
stream=0)
MTP_LINK  0  0  0  0  0  0  0x10  0x10  0x0006
MTP_LINK  1  0  1  1  0  1  0x10  0x01  0x0006
*
* Define a route for each remote signaling point:
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
MTP_ROUTE  2  0  0x0020
*
* Define any user provided Layer 4 protocol:
* MTP_USER_PART <service_ind> <module_id>
MTP_USER_PART 0x0a 0xef
*
MTP_TRACE 1 1 0
*
```

A.2.2 Point code 2 (UPE)

```
***************************************************************************
* Example config.txt.
* Edit this file to reflect your configuration.
***************************************************************************
*
* SEPTELPCI_BOARD <board_id> <flags> <code_file> <run_mode>
SEPTELPCI_BOARD  0  0x0043  ss7.dc3 MTP2
*
* Configure individual E1/T1 interfaces:
* LIU_CONFIG <board_id> <liu_id> <liu_type> <line_code> <frame_format>
*            <crc_mode>
*LIU_CONFIG  0  0  5  1  1  1
*
* MTP Parameters:
* MTP_CONFIG <reserved> <reserved> <options>
MTP_CONFIG  0  0  0x000000000
*
* Define linksets:
* MTP_LINKSET <linkset_id> <adjacent_spc> <num_links> <flags> <local_spc>
*             <ssf>
MTP_LINKSET  0  1  2  0x0000  2  0x08
*
* Define signaling links:
* MTP_LINK <link_id> <linkset_id> <link_ref> <slc> <board_id> <blink>
*          <stream> <timeslot> <flags>
stream=0)
MTP_LINK  0  0  0  0  0  0  0x10  0x10  0x0006
MTP_LINK  1  0  1  1  0  1  0x10  0x01  0x0006
*
* Define a route for each remote signaling point:
* MTP_ROUTE <dpc> <linkset_id> <user_part_mask>
MTP_ROUTE  1  0  0x0020
*
* Define any user provided Layer 4 protocol:
* MTP_USER_PART <service_ind> <module_id>
MTP_USER_PART  0x0a  0x2d
*
MTP_TRACE 1  1  0
*
```

A.2.3 Using different board configurations and operating systems

The following subsections provide information regarding the config.txt file if using different operating systems or board-based configurations.

A.2.3.1 Board configurations

For SS7HD PCI boards:

Replace the `SEPTELPCI_BOARD` command with the following:

```
      SS7_BOARD  0 SS7HDP 0x0003  ss7.dc4  MTP
```

Refer to [4] for further information.

A.2.3.2 Running UPE with host binary

When using a host binary so that the MTP3 module is run on the host instead of the board:

- For SPCI2S, SPCI4S and CPM8:
  the <run_mode> field in the SEPTELCP_BOARD command must be set to MTP2

Refer to [3] and [4] as appropriate.

A.2.3.3 Running UPE on the SIU host

If using the SIU, additional commands required the SIU will need to be included. Therefore, the example config.txt provided in this appendix should not be used.

Refer to [2] for further information.

A.2.3.4 Running UPE with other operating systems

There are no additional commands specific to various operating systems, however, if using operating systems other than Windows®, the names of some of the executable files used in the FORK_PROCESS commands need to be changed.

Refer to [3] and [4] as appropriate.

A.2.3.5 Running UPE with SIGTRAN M2PA

The board configuration commands `(SEPTELPCI_BOARD and LIU_CONFIG)`should be removed and replaced with the CNSYS and SNSLI commands.

Refer to [5] as appropriate.

Appendix B - Example script files

This section provides example scripts files which may be used with s7_play at the remote end (point code 1). For each of the provided MTP-TRANSFER Request messages, the header information is set as follows:

Service indicator octet(sio)	= 0x0a
Sub-service field(ssf)	= 0x8
Destination point code(dpc)	= 0x2
Origination point code(opc)	= 0x1
Signaling link selection(sls)	= 0x0

B.1 MTP-TRANSFER Request

14-bit point code format:

```
M-tcf00-i0000-fef-d22-s00-p8a02400000000000000000000000000000000000000
```

16-bit point code format:

```
M-tcf00-i0000-fef-d22-s00-p8a02000100000000000000000000000000000000
```

24-bit point code format:

```
M-tcf00-i0000-fef-d22-s00-p8a0200000100000000000000000000000000000000
```

Appendix C - Sample output

This section shows sample output for the UPE application.

C.1 14-bit point code format

UPE was started using :

```
upe –m0x2d -c14
```

UPE 14-bit point code example output :

```
UPE: User Part Example. Copyright (C) Dialogic Corporation 1991-2006. All Rights Reserved.
==========================================================================================
14 bit point code selected.

Requires the following entry in config.txt:
MTP_USER_PART 0x0a 0x2d

MTP-RESUME dpc=1
MTP-TRANSFER-IND: 8a 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

C.2 16-bit point code format

UPE was started using :

```
upe –m0x2d -c16
```

UPE 16-bit point code example output :

```
UPE: User Part Example. Copyright (C) Dialogic Corporation 1991-2006. All Rights Reserved.
==========================================================================================
16 bit point code selected.

Requires the following entry in config.txt:
MTP_USER_PART 0x0a 0x2d

MTP-RESUME dpc=1
MTP-TRANSFER-IND: 8a 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

C.3 24-bit point code format

UPE was started using :

```
upe –m0x2d –c24
```

UPE 24-bit point code example output :

```
UPE: User Part Example. Copyright (C) Dialogic Corporation 1991-2006. All Rights Reserved.
==========================================================================================
24 bit point code selected.

Requires the following entry in config.txt:
MTP_USER_PART 0x0a 0x2d

MTP-RESUME dpc=1
MTP-TRANSFER-IND: 8a 02 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MTP-TRANSFER-IND: 8a 02 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```