Dialogic® SS7 Protocols
Call Test Utility (CTU) User Guide

Document Reference  U25SSS
## Revision History

<table>
<thead>
<tr>
<th>ISSUE</th>
<th>DATE</th>
<th>CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>05-Oct-01</td>
<td>Initial release</td>
</tr>
<tr>
<td>2</td>
<td>16-Jun-03</td>
<td>Branding changed: Septel PCI now SPCI4/SPCI2S and SPCI2S, SPCI4S and CPM8 now CPM8. References to NUP, QNX and SCO removed.</td>
</tr>
<tr>
<td>3</td>
<td>01-Oct-07</td>
<td>Remove references to PCCS6 and add support for SIGTRAN M2PA links</td>
</tr>
</tbody>
</table>
## Contents

1 **Introduction** ................................................................................................................................. 5
   1.1 Software requirements ............................................................................................................... 5

2 **CTU Application** .......................................................................................................................... 6
   2.1 Message Sequence Chart ........................................................................................................... 7
   2.2 Customizing the CTU application ............................................................................................ 9
   2.3 CTU source code ....................................................................................................................... 10

3 **Building the CTU application** .................................................................................................... 11
   3.1 Host software directory structure ........................................................................................... 12
   3.2 Building CTU ........................................................................................................................... 12

4 **Configuration** ............................................................................................................................. 14
   4.1 System Configuration .............................................................................................................. 14
   4.1.1 SPCI2S, SPCI4 and SS7HD ........................................................................................... 14
   4.1.2 SIGTRAN M2PA ............................................................................................................. 14
   4.1.3 SS7G2x SIU .................................................................................................................... 14
   4.2 Protocol Configuration ............................................................................................................. 15
   4.2.1 ISUP .............................................................................................................................. 15
   4.2.2 TUP ............................................................................................................................... 15

5 **Running the CTU application** ..................................................................................................... 16
   5.1 CTU Command Line Arguments ............................................................................................ 16
   5.1.1 CTU Options .................................................................................................................. 17

6 **References** ................................................................................................................................... 18

7 **Abbreviations** ............................................................................................................................... 19
1 Introduction

The Dialogic® Call Test Utility (CTU) is an example application designed to demonstrate the use of the Dialogic® ISUP and TUP modules. This user guide describes the design, build and usage of this application for developers.

The purpose of CTU is to demonstrate the behavior of the telephony modules by offering simple functionality while showing real examples of the interface to the modules. The CTU application can also be used to provide simple verification of system configuration and set-up.

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 ISUP and TUP modules.

1.1 Software requirements

The CTU 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, ISUP/TUP host binaries, as required
4. For SIGTRAN-based configurations:
   - Dialogic® M2PA, MTP3, ISUP/TUP host binaries, as required

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

The Dialogic® CTU application is used to receive incoming calls. When an incoming call is received, CTU performs simple checks on the parameters in the received message for example checking the parameter name and its length, and answers the call before making an outgoing call on the next circuit.

CTU takes a number of command line options (refer to Section 5.1 CTU Command Line Arguments), which if set shall allow full tracing of sent and received messages.

**Figure 1.** CTU Network Architecture

**Figure 2.** Typical configuration
2.1 Message Sequence Chart

The following pages contain message sequence charts showing typical message flows for phone calls made when using CTU. Each chart shows the message flow between the application and the telephony module e.g. ISUP.

As a basic behavior, CTU waits for a Setup Indication to be received and then responds with an Alerting Request followed by a Setup Response on the same circuit. (Refer to the appropriate programmer’s manual [1] and [2] for further information). After the incoming call is connected, CTU initiates a new outgoing call on the next circuit (with a Setup Request using the same parameters as those in the received Setup Indication) and waits for the remote application to complete the connection (Figure 4).

![Message Sequence Chart]

*Figure 3. Typical basic incoming call*
Figure 4. Typical basic outgoing call
2.2 Customizing the CTU application

CTU is example program for development using the ISUP and TUP protocols and as such, a number of simplifications and limitations have been implemented.

It is possible to use the CTU application for the development of protocols, such as ISUP, irrespective of its variant e.g. ITU-T, ANSI.

The main simplification and limitation is that the outgoing call is always made on the next circuit. In a real application, whenever an incoming call is received some kind of processing would be required to determine information such as circuit selection and routing. However, as an example application, CTU will always use the same data that was received from the incoming call to setup the outgoing call. CTU provides support for a limited number of messages and parameters. Therefore, only mandatory and few optional parameters are supported. CTU has no provision for handling segmented messages nor the ability to handle proprietary messages and parameters.

For parameters received by CTU, only supported parameters will be passed transparently from the incoming call to the outgoing call to the remote end. Unsupported or unrecognized parameters are ignored and shall consequently be discarded. Similarly, on receipt of an unsupported or unrecognized message, the message shall be ignored and discarded. The CTU example program may be further extended by the user to meet additional requirements.

Refer to Appendix E - Supported Messages and Parameters for a list of all messages and parameters supported.

Refer to Appendix F - Adding new messages and parameters for information on adding new messages and parameters.
2.3 CTU source code

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

<table>
<thead>
<tr>
<th>File</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctu_main.c</td>
<td>This file contains the main() function. This reads the command line arguments and passes them to ctu_ent().</td>
</tr>
<tr>
<td>ctu.c</td>
<td>The function ctu_ent() contains the main control loop for the application. It waits for a message to be received and calls CTU_cal_ind() to handle each one. CTU_cal_ind() first recovers the parameters into a C structure (CAL_msg_to_ind()), displays the message according to the command line options which were set (CTU_display_ind()), and then handles the data received according to the current call state.</td>
</tr>
<tr>
<td>call.c</td>
<td>Call control interface library functions. This file contains functions for recovering message parameters into a structured form (CAL_msg_to_ind()), and vice versa (CAL_req_to_msg()).</td>
</tr>
<tr>
<td>call.h</td>
<td>Contains #defines, structure definitions and function prototypes used by CTU. Includes &quot;cpp_inc.h&quot; (see Note).</td>
</tr>
<tr>
<td>snd_cgsc.c</td>
<td>This file is not part of the CTU example program but has been included to demonstrate how blocking and reset may be initiated by using a Circuit Group Supervision Control request.</td>
</tr>
</tbody>
</table>

Note: cpp_inc.h (included in the Dialogic® SS7 Development Package) contains #defines used by the Common Call Control interface. Although these definitions are largely based on ITU-T Q.763, due to the number of protocols and variants supported there may be some differences in the definitions used as it is not always possible to use the same definition as specified by the ITU-T recommendation.
3 Building the CTU application

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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic UNIX (Solaris, Linux)</td>
<td>.mak</td>
</tr>
<tr>
<td>Windows®</td>
<td>.mnt</td>
</tr>
</tbody>
</table>

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 the Windows® operating system, 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 CTU 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 CTU 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 CTU program.

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

<table>
<thead>
<tr>
<th>Root directory</th>
<th>Septel</th>
<th>UPD</th>
<th>SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>INC</td>
<td>ccpx_inc.h</td>
<td>msg.h</td>
<td>pack.h</td>
</tr>
<tr>
<td></td>
<td>ss7_inc.h</td>
<td>strtonum.h</td>
<td>sysgct.h</td>
</tr>
<tr>
<td></td>
<td>system.h</td>
<td>BIN</td>
<td>CTU</td>
</tr>
<tr>
<td></td>
<td>BACKUP_WIN</td>
<td>ctu.bnt</td>
<td>ctu.mnt</td>
</tr>
<tr>
<td></td>
<td>BACKUP_LNX</td>
<td>ctu.mak</td>
<td>ctu.c</td>
</tr>
<tr>
<td></td>
<td>BACKUP_SOL</td>
<td>ctu_iss.txt</td>
<td>call.c</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ctu_main.c</td>
<td>call.h</td>
</tr>
<tr>
<td></td>
<td></td>
<td>makeall.bat</td>
<td>makedefs.mnt</td>
</tr>
<tr>
<td></td>
<td></td>
<td>makeall</td>
<td>makedefs_sol.mak</td>
</tr>
<tr>
<td></td>
<td></td>
<td>makeall_sol</td>
<td>makedefs.mak</td>
</tr>
</tbody>
</table>

3.2 Building CTU
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.bat       (Windows®)
makeall           (Linux)
makeall_sol       (Solaris)
```

A pre-built copy of the CTU application, for each operating system, can be located within the backup subdirectories in the BIN directory.
To build the CTU program, change to the SRC\CTU directory and type one of the following commands depending on the operating system:

```
nmake /f ctu.mnt
make –f ctu.mak
make -f ctu_sol.mak
```
4 Configuration

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

<table>
<thead>
<tr>
<th>Root directory</th>
<th>RUN</th>
<th>CTU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>“CONFIG1”</td>
<td>“CONFIG2”</td>
</tr>
<tr>
<td>Config.txt</td>
<td></td>
<td>Config.txt</td>
</tr>
<tr>
<td>System.txt</td>
<td></td>
<td>System.txt</td>
</tr>
</tbody>
</table>

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 - Example CTU configuration files for further information.

4.1 System Configuration

The GCT environment is configured using the Dialogic® gctload program and the system.txt file. The basic board configuration along with the Dialogic® MTP, SCCP, TCAP and MAP modules 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, ISUP or TUP modules are configured using the config.txt file.

Example configuration files for CTU are contained in the Appendix. When running CTU on a Windows® host system using a SPCI4 with the MTP3 and ISUP modules running on the board, the provided example configuration files may be used without any modification. Configuration details for other board types are also provided for reference.

4.1.2 SIGTRAN M2PA

It is also possible to run the CTU applications from 2 hosts connected in back-to-back with SIGTRAN M2PA links.

4.1.3 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 [4].

*Note: These files are not contained in the User Part Development Package.*
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
- Signaling timeslot

4.2.1 ISUP

The local point code is contained in the main ISUP configuration command (refer to the programmer’s manual [1] for details). In addition, configuration commands are required for each circuit group.

4.2.2 TUP

The local point code is contained in the main TUP configuration command (refer to the programmer’s manual [2] for details). In addition, configuration commands are required for each circuit group.
5 Running the CTU application

Before running the Dialogic® CTU application, the GCT environment must first be initialized and the signaling links brought into service. This is achieved by running the Dialogic® gctload program, and activating the links using the Dialogic® mtpsl utility. Refer to manuals [4], or [5] for details as appropriate.

In the example configuration, calls must be initiated at the remote end (config2) either by the use of test equipment or script files may be used in conjunction with the Dialogic® s7_play utility. If using these example configuration files, CTU must be run at the end using config1. Example script files are provided in Appendix B - Example script files.

For further information on these commands, refer to the appropriate Programmer’s manual ([1] or [2]).

5.1 CTU Command Line Arguments
The module takes a number of command line arguments, which are summarized below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m</td>
<td>0x3d</td>
<td>CTU module Id</td>
</tr>
<tr>
<td>-c</td>
<td>0x23</td>
<td>CTU user part module (defaults to ISUP). If using TUP ensure that this is set to 0x4a.</td>
</tr>
<tr>
<td>-o</td>
<td>0x0010</td>
<td>Run-time options (defaults to display received indications) Add together required values for tracing options, if required (see section 5.1.1 CTU Options).</td>
</tr>
</tbody>
</table>

Example:

cwu –c0x23 –o0x0017

The above example will set the internal CTU tracing for ISUP with OPT_TR_PARAM, OPT_TR_PRIM, OPT_TR_CALL and OPT_TR_RX_IND run time options. These options are described in the following subsection.
### 5.1.1 CTU Options

<table>
<thead>
<tr>
<th>Tracing option</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPT_TR_PARAM</td>
<td>0x0001</td>
<td>Display received message parameter buffer</td>
</tr>
<tr>
<td>OPT_TR_PRIM</td>
<td>0x0002</td>
<td>Display primitive messages</td>
</tr>
<tr>
<td>OPT_TR_CALL</td>
<td>0x0004</td>
<td>Display call parameters</td>
</tr>
<tr>
<td>OPT_TR_TX_REQ</td>
<td>0x0008</td>
<td>Display transmit requests</td>
</tr>
<tr>
<td>OPT_TR_RX_IND</td>
<td>0x0010</td>
<td>Display received indications</td>
</tr>
<tr>
<td>OPT_ANSI</td>
<td>0x0100</td>
<td>Configures CTU to run in ANSI mode</td>
</tr>
</tbody>
</table>
6 References

[7] ITU-T Recommendation Q.763, Signaling System No.7 – ISDN user part formats and codes

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

The following lists acronyms alphabetically used in this user guide.

CTU   Call Test Utility
DPC   Destination point code
GID   Group ID
IC    Incoming
ISUP  ISDN User part
MTP   Message transfer part
OG    Outgoing
OPC   Originating point code
SIU   Signaling Interface Unit
SLC   Signaling link code
SSF   Sub-service field
TUP   Telephony user part
UP    User Part
Appendix A - Example CTU configuration files

This section provides example configuration files (system.txt and config.txt) for use with the Dialogic® CTU application on a Windows® host system for Dialogic® SPCI4 boards. The ISUP module is running on the board and CTU is running as module ID 0x3d.

Before configuring the protocol modules it is useful to determine information such as the local point code and remote point code relative to each network entity. For this example configuration, the local point code is 1 and the remote point code is 2 (see Fig. 5).

**Example configuration**

- **Operating system:** Windows®
- **Board type:** SPCI4
- **Local point code:** 1 (CTU)
- **Remote point code:** 2 (test equipment or script files)
- **CTU module ID:** 0x3d
- **Modules running on the board:** ISUP/MTP3
- **Modules running on the host:** None

![Figure 5. Example configuration](image)

**Figure 5.** Example configuration
A.1 system.txt

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

The following example system.txt file is valid for point code 1 and point code 2. All comments are denoted by ‘*’. For reference, the provided system.txt file also includes example configurations for TUP, although all commands specific to the TUP protocol modules have been commented out.

A.1.1 system.txt for point code 1(CTU) and point code 2(remote end)

********************************************************************
* Example system.txt.                                            *
* Edit this file to reflect your configuration.                   *
********************************************************************

* Essential modules running on host:
*
LOCAL 0x20  * ssds - Board interface task
LOCAL 0x00  * tim_nt - Timer task

* Optional modules running on the host:
*
LOCAL 0xcf  * s7_mgt - Management/config task
LOCAL 0x3d  * ctu - Example user part task
LOCAL 0xef  * s7_log - Logs status, trace and error messages

* Modules running on the board (all redirected via ssd):
*
REDIRECT 0x23 0x20 * ISUP module
* REDIRECT 0x4a 0x20 * TUP module
REDIRECT 0x22 0x20 * MTP3 module
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
FORK_PROCESS ..\..\..\..\..\..\tim_nt.exe
FORK_PROCESS ..\..\..\..\..\..\tick_nt.exe
FORK_PROCESS ..\..\..\..\..\s7_mgt.exe
FORK_PROCESS ..\..\..\..\..\s7_log.exe
A.1.2 Using different operating systems and configurations

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

A.1.2.1 Running CTU with SS7HD

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
```

Refer to [5] for further information.

A.1.2.2 Running CTU on the with SS7G2x

If using the SIU, additional commands required by the SIU will need to be included. Therefore, the example system.txt provided in this appendix should not be used. Refer to [4] for further information.

A.1.2.3 Running protocols on the host

If using a host binary, so that the protocol module is run on the host instead of the board, start up the appropriate host binary using the `FORK_PROCESS` command, 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 [4], [5] and [5] as appropriate.

A.1.2.4 Running CTU 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 [4], [5] and [5] as appropriate.

A.1.2.5 Running CTU with TUP

TUP has a different module ID, 0x4a, from ISUP (0x23) and if TUP is to be run, on the board or on the host ensure that the TUP module ID is specified in the system.txt file.

Refer to manuals [2] and [5] for further information as appropriate.

A.1.2.6 Running CTU 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 [8] for further information as appropriate.
A.2 config.txt

This section provides two example config.txt files for a SPCI4 board running under Windows® using the example configuration described earlier in this appendix showing the protocol modules can be configured for use.

The following example config.txt files are for point code 1 and point code 2. All comments are denoted by ‘*’. For reference, the provided config.txt file also includes example configurations for TUP, therefore all messages specific to the TUP protocol modules have been commented out.

Using the two example config.txt files (one at each end of the link) will allow a call using ISUP or TUP with 14-bit point codes to be demonstrated. If connecting to other equipment, all the various parameters in the file 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)
- variant of ISUP or TUP (e.g. ANSI ISUP, China TUP)
- signaling timeslot

The example files provided in this appendix should not be used for the SIU (refer to [4] for further information).
A.2.1 config.txt for point code 1 (CTU)

***************************************************************************
* This file needs to be modified to suit individual circumstances.        *
* Refer to the relevant Programmer's Manuals for further details.          *
***************************************************************************
* For SPCI2S, SPCI4S and CPM8 / PCI boards:                              *
* SEPTELCP_BOARD <board_id> <flags> <code_file> <run_mode>              *
  SEPTELPCI_BOARD  0 0x0043 ss7.dc3 ISUP                              *
* *                                                                      *
* 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 0x0000                                                  *
* * 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 16 16 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 0x2d                                             *
* * Configure ISUP module:                                             *
* ISUP_CONFIG <reserved> <reserved> <options> <num_grps>                *
  <num_ccts>                                                           *
  ISUP_CONFIG  0 0 0 0x0435 4 64                                       *
* * Configure ISUP circuit groups:                                     *
* ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> *
  <user_inst> <user_id> <opc> <ssf> <variant>                         *
  <options2>                                                          *
  ISUP_CFG_CCTGRP 0 2 0x01 0x01 0x7fff7fff 0x001c 0 0x3d 1 0x8 0 0x00  *
* * Configure TUP Parameters:                                         *
* TUP_CONFIG <reserved> <reserved> <options> <num_grps>                 *
  <num_Ccts>                                                           *
  TUP_CONFIG  0 0 0 0x8141 4 64                                       *
* * Define TUP circuit groups:                                        *
* TUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> *
  <user_inst> <user_id> <opc> <ssf>
*TUP_CFG_CCTGRP  0  1  0x01  0x01  0x7fff7fff  0x0030  0 0x3d  2 0x08
A.2.2 config.txt for point code 2 (remote end)

***************************************************************************
* This file needs to be modified to suit individual circumstances.       *
* Refer to the relevant Programmer's Manuals for further details.        *
***************************************************************************

* Configure individual boards:
* For SPCI2S, SPCI4S and CPM8 / PCI boards:
* SEPTELCP_BOARD <board_id> <flags> <code_file> <run_mode>
  SEPTELCPCI_BOARD 0 0x0043 ss7.dc3 ISUP
*
* 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 0x0000
*
* 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>
  (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 16 16 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
*

* Configure ISUP module:
* ISUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> 
  <num_ccts>
  ISUP_CONFIG 0 0 0 0x0435 4 64
*
* Configure ISUP circuit groups:
* ISUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options> 
  <user_inst> <user_id> <opc> <ssf> <variant> 
  ISUP_CFG_CCTGRP 0 1 0x01 0x01 0x7fff7fff 0x001c 0 0xef 2 0x8 0 0x00
*
* Configure TUP Parameters:
* TUP_CONFIG <reserved> <reserved> <reserved> <options> <num_grps> 
  <num_ccts>
  TUP_CONFIG 0 0 0 8141 4 64
Appendix A - Example CTU configuration files

* Define TUP circuit groups:
* TUP_CFG_CCTGRP <gid> <dpc> <base_cic> <base_cid> <cic_mask> <options>
*                      <user_inst> <user_id> <opc> <ssf>
*TUP_CFG_CCTGRP  0  1  0x01  0x01  0x7ffe7ffe  0x0030  0 0xef  2 0x08
*  
* Message tracing:
ISUP_TRACE 0xffffffff 0xffffffff 0xffffffff
*TUP_TRACE 0xffffffff 0xffffffff 0xffffffff
A.2.3 Using different operating systems and configurations

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

A.2.3.1 Running CTU with SS7HDP

If using SS7HDP boards, the SEPTELPCI_BOARD command should be replaced with the following:

```
    SS7_BOARD  0 SS7HDP 0x0003  ss7.dc4  MTP2
```

Refer to [5] for further information.

A.2.3.2 Running CTU with host binary

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

- For SPCI2S, SPCI4S and CPM8:
  the `<run_mode>` field in the SEPTELCP_BOARD command should be set to an appropriate
  'runmode' e.g. MTP2

Refer to [2] and [5] as appropriate.

A.2.3.3 Running CTU on the with SS7G2x

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 [4] for further information.

A.2.3.4 Running CTU with other operating systems

There are no additional commands specific to various operating systems.

Refer to [2] and [5] as appropriate.

A.2.3.5 Running CTU with TUP

The following is applicable only if the telephony module is to run on the board.

If TUP is to be used instead of ISUP, the following changes will be required:

- The appropriate run mode should be indicated in the `XXXX_BOARD <runmode>` field.
- The appropriate codefile should be indicated in the `XXXX_BOARD <code file>` field.
- In the 'Define a route for each remote signaling point' section, the `<user_part_mask>` field in
  the 'MTP_ROUTE' command should be set to 0x0010 (for TUP).
- Configure the protocol module using the appropriate commands e.g. TUP_CONFIG instead of
  ISUP_CONFIG, etc. These commands for TUP are provided in the example config.txt files (but
  have been commented out).

Refer to manuals [2] and [5] for further information as appropriate.
A.2.3.6 Running CTU with SIGTRAN M2PA

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

Refer to [8] as appropriate.
Appendix B - Example script files

This section provides example script files which may be used with the Dialogic® s7_play utility at the remote end (point code 2) to make a basic call. The scripts provided below, for ITU based operations, should be run in the listed order to initiate a new call, answer a second call and release both calls.

Setup request (IAM):
M-tc700-i0001-fef-d23-r0000-p0109010a02010204078310214365870f0a02031000

Alerting request (ACM):
M-tc700-i0002-fef-d23-r0000-p0600

Setup response (ANM):
M-tc700-i0002-fef-d23-r0000-p0900

Release response (RLC):
M-tc700-i0002-fef-d23-r0000-p1000

Release request (REL):
M-tc700-i0001-fef-d23-r0000-p0c00

Release response (RLC):
M-tc700-i0001-fef-d23-r0000-p1000
Appendix C - Sample output

The following is a sample output from running the Dialogic® CTU application using the provided example configuration and script files at the remote end.

CTU was started using the following run time options:

```
-c0x23 -0x001f -m0x3d
```

CTU example output:

CTU: Example application and CAL interface  (C) Dialogic Corporation 1995-2006. All Rights Reserved.

==========================================================================

CTU mod ID - 0x3d; User Part module Id 0x23

CTU receives an incoming call on cid = 1:

CTU Rx: inst = 0x00 cid = 0x0001 01 06 01 00 07 02 40 00 09 01 0a 02 01 02 04 07 83 10 21 43 65 87 0f 0a 02 03 10 00
ptype = (01) Setup indication
called_num = 83 10 21 43 65 87 0f
calling_num = 03 10
cpc = 0a

CTU IC: (cid = 0x0001) [cald->12345678.] cpc = 0x0a

CTU Tx: Inst = 0x00 cid = 0x0001 09 00
ptype = (09) Setup response

CTU makes an outgoing call on cid =2:

CTU Tx: Inst = 0x00 cid = 0x8002 01 06 01 00 09 01 0a 04 07 83 10 21 43 65 87 0f 00
ptype = (01) Setup request
called_num = 83 10 21 43 65 87 0f
cpc = 0a

CTU OG: (cid = 0x0002) [cald->12345678.] cpc = 0x0a

CTU Rx: inst = 0x00 cid = 0x8002 06 11 02 16 14 00
ptype = (06) Alerting indication
bci = 16 14

CTU Tx: Inst = 0x00 cid = 0x8002 09 00
ptype = (09) Setup confirmation

CTU Rx: inst = 0x00 cid = 0x8002 0c 12 02 80 90 00
ptype = (0c) Release request
causei = 80 90
Appendix D - SDL Diagrams

The following are a list of SDL diagrams for the Dialogic® CTU application:

1. Idle
2. Wait Idle
3. IC Circuit seized
4. IC Setup
5. IC Active
6. IC Wait COT check
7. OG Setup
Appendix D - SDL Diagrams

SDL 1. CTU Idle State

- Release Indication
  - Release Response
    - Wait Idle
  - CCT seized indication
    - COT required?
      - Y
        - Save parameters
        - Make OG call
          - Idle
          - 1
      - N
        - can connect transponder here
        - ANSI option?
          - Y
            - LPA request
          - N
            - IC circuit seized
            - 2
        - IC circuit seized
    - Y
      - can connect transponder here
      - COT required?
        - Y
          - Use the called address, calling address, calling party category (cpc), address presentation restriction (apr), screening indication (si) and instance values from the incoming call.
        - N
          - based on value of nature of address indicator (noci) in IAM.

- Setup Indication
  - Other
    - Save parameters
    - can connect transponder here
      - COT required?
        - Y
          - Use the called address, calling address, calling party category (cpc), address presentation restriction (apr), screening indication (si) and instance values from the incoming call.
        - N
          - based on value of nature of address indicator (noci) in IAM.
      - N
        - can connect transponder here
        - IC circuit seized
        - 2

SDL 2. CTU Wait Idle
Appendix D - SDL Diagrams

SDL 3. IC Circuit Seized State

- **IC circuit seized**
  - COT report indication (failure)
    - disconnected loop back
      - IC circuit seized
  - Release Indication
    - COT required?
      - disconnected loop back
      - Release Response
        - Wait Idle
  - Setup Indication
  - Other
    - IC circuit seized
  - Save parameters
    - COT required?
      - Y
        - apply loop back
        - IC wait COT check
        - 2
      - N
    
- Based on value of Nature of address indicator (noci) in IAM
  - Use the called address, calling address, calling party category (cpc), address presentation restriction (apr), screening indication (si) and instance values from the incoming call.
IC Setup State

When ST digit or maximum number of digits received

Accept call? Based on value of CPC

Alerting request

Make OG call

Using next logical circuit ID (cid)

IC Active

Wait Idle

Release Response

Release Request

Release Indication

IC Setup

Other

Info Indication

Dave address digits

Address complete?

Y

N

Y

N

N
SDL 5. CTU IC Active State
SDL 6. CTU IC wait COT check State
Appendix D - SDL Diagrams

CTU OG Setup state

1

N
COT Requested previous cct?

Y

Setup Request

Continuity check indicator set to "continuity check not required on this circuit"

Setup Request

Continuity check indicator set to "continuity performed on previous circuit"

COT requested (success)

OG Setup

Alert Indication

OG Alerting

Other

OG Setup

Release Response

Wait Idle

Setup Confirmation

OG Setup

Alert Indication

OG Alerting

Other

OG Setup

Release Response

Wait Idle

As soon as the call is answered, CTU will release the call in the forward direction.

OG Setup state
## Appendix E - Supported Messages and Parameters

### E.1 Supported messages

This section tabulates messages supported by the Dialogic® CTU example application.

**Transmitting direction:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert request</td>
<td>CALPN_ALERT_REQ</td>
<td></td>
</tr>
<tr>
<td>Connect request</td>
<td>CALPN_CON_REQ</td>
<td></td>
</tr>
<tr>
<td>Continuity report request</td>
<td>CALPN_COT_REQ</td>
<td></td>
</tr>
<tr>
<td>Loop back acknowledgement request</td>
<td>CALPN_LPA_REQ</td>
<td></td>
</tr>
<tr>
<td>Release request</td>
<td>CALPN_RELEASE_REQ</td>
<td></td>
</tr>
<tr>
<td>Release response</td>
<td>CALPN_RELEASE_RESP</td>
<td></td>
</tr>
<tr>
<td>Setup request</td>
<td>CALPN_SETUP_REQ</td>
<td></td>
</tr>
<tr>
<td>Setup response</td>
<td>CALPN_SETUP_RESP_ANM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CALPN_SETUP_RESP_CON</td>
</tr>
</tbody>
</table>

**Receiving direction:**

<table>
<thead>
<tr>
<th>Message</th>
<th>Mnemonic</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alert indication</td>
<td>CALPN_ALERT_IND</td>
<td></td>
</tr>
<tr>
<td>Backward information indication</td>
<td>CALPN_INFO_IND</td>
<td></td>
</tr>
<tr>
<td>Call progress indication</td>
<td>CALPN_PROGRESS_IND</td>
<td></td>
</tr>
<tr>
<td>Circuit seized indication</td>
<td>CALPN_CCT_SZE_IND</td>
<td></td>
</tr>
<tr>
<td>Continuity report indication</td>
<td>CALPN_COT_IND</td>
<td></td>
</tr>
<tr>
<td>Information indication</td>
<td>CALPN_BINFO_IND</td>
<td></td>
</tr>
<tr>
<td>Release confirmation</td>
<td>CALPN_RELEASE_CONF</td>
<td></td>
</tr>
<tr>
<td>Release indication</td>
<td>CALPN_RELEASE_IND</td>
<td></td>
</tr>
<tr>
<td>Setup confirmation</td>
<td>CALPN_SETUP_CONF_ANM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CALPN_SETUP_CONF_CON</td>
</tr>
<tr>
<td>Setup indication</td>
<td>CALPN_SETUP_IND</td>
<td></td>
</tr>
</tbody>
</table>
### E.2 Supported parameters

The following table lists parameters supported by the CTU application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Mnemonic</th>
<th>Maximum length of parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward call indicators</td>
<td>CALPPN_BCI</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Called party number</td>
<td>CALPPN_CALLED_NUM</td>
<td>16</td>
<td>For CTU to make a subsequent outgoing call, the called party number in the IAM must contain a minimum of 8 digits.</td>
</tr>
<tr>
<td>Calling party's category</td>
<td>CALPPN_CPC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Calling party's number</td>
<td>CALPPN_CALLING_NUM</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Cause indicators</td>
<td>CALPPN_CAUSEI</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Continuity indicators</td>
<td>CALPPN_CONTI</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>End of parameters</td>
<td>CALPPN_TERMINATOR</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Forward call indicators</td>
<td>CALPPN_FCI</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Nature of connection indicators</td>
<td>CALPPN_NOCI</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NUP Forward call indicators</td>
<td>CALPPN_NUP_FCI</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Redirecting number</td>
<td>CALPPN_REDIR_NUM</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Redirection information</td>
<td>CALPPN_REDIR_INF</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Subsequent number</td>
<td>CALPPN_SUBSQ_NUM</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Transmission medium requirement</td>
<td>CALPPN_TMR</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Appendix F - Adding new messages and parameters

This section describes the steps required for adding new messages and parameters to the Dialogic® CTU application.

F.1 Adding new messages

Note that this section assumes that parameters to be added to the new message already exist. If new parameters are needed, see subsection F.2 Adding new parameters.

To add a new message for sending:

- create a new #define in call.h for the primitive type e.g. CALPN_ALERT_REQ
- create a structure definition for the primitive in call.h (e.g. calpt_alert_req) and add this new definition into the calpt_req structure (as part of the union)
- create a parameter table for the new primitive in call.c e.g. alert_req_tab
- add the primitive definition to req_ptable in call.c
- initialize the parameter table in CAL_init() in call.c
- create a new function to send the message e.g. CTU_alert_req() in ctu.c and call the function at appropriate place in CTU_cal_ind()
- add code to display the received message in CTU_display_req() in ctu.c

To add a new message for receiving:

- create a new #define in call.h for the primitive type e.g. CALPN_ALERT_IND
- create a structure definition for the primitive in call.h (e.g. calpt_alert_ind) and add this new definition into the calpt_ind structure (as part of the union)
- create a parameter table for the new primitive in call.c e.g. alert_ind_tab
- add the parameter table name to ind_ptable in call.c
- initialize the parameter table in CAL_init() in call.c
- handle the new primitive in CTU_cal_ind() in ctu.c in the appropriate state
- add code to display the received message in CTU_display_ind() in ctu.c
F.2 Adding new parameters

To add a new parameter for sending:

- create a new #define for the new parameter name token in call.h e.g. CALPN_CALLING_NUM
- create a new #define for the maximum parameter length e.g. CALPL_CALLING_NUM
- create a structure definition for the parameter in call.h e.g. calpt_calling_num
- add the new parameter to the structured form of the primitive e.g. CALPT_SETUP_REQ in call.h
- initialize the parameter in the parameter table for the primitive (e.g. setup_req_tab) in CAL_init() in call.c. Check that the number of parameters does not now exceed the capacity of the parameter table. If necessary, adjust the definition of CAL_MAX_PARMS in call.c.
- assign a value to the parameter in the structured form of the primitive in the function that builds and sends the primitive e.g. CTU_setup_req() in ctu.c
- initialize the new parameter in the parameter table for the required primitive (e.g. setup_ind_tab) in CAL_init() in call.c. Check that the number of parameters does not now exceed the capacity of the parameter table. If necessary, adjust the definition of CAL_MAX_PARMS in call.c.
- add code to display the value of the parameter e.g. CTU_setup_req() in ctu.c

To add a new parameter for receiving:

- create a new #define for the new parameter name token in call.h e.g. CALPN_CALLING_NUM
- create a new #define for the maximum parameter length e.g. CALPL_CALLING_NUM
- create a structure definition for the parameter in call.h e.g. calpt_calling_num
- add the new parameter to the appropriate primitive type in call.h e.g. calpt_setup_req
- initialize the new parameter in the parameter table for the required primitive (e.g. setup_ind_tab) in CAL_init() in call.c. Check that the number of parameters does not now exceed the capacity of the parameter table. If necessary, adjust the definition of CAL_MAX_PARMS in call.c.
- add code to display the value of the parameter e.g. CTU_setup_req() in ctu.c
- add code to handle the new parameter. For example, if the new parameter is to be received in the setup indication, it could be stored in the call data structure using the function CTU_save_setup_ind() in ctu.c.