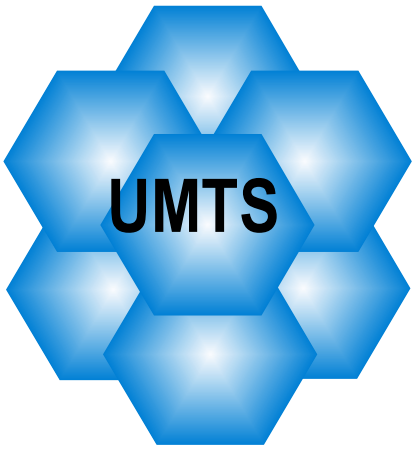

G23-UMTS Protocol Stack

tds to tdc converter

User Guide

Author: Condat AG
Alt Moabit 90a
10559 Berlin
Germany

Date: 4 March, 2002

ID: 8434_513_02.002

Status: Being Processed

Table of Contents

0 Document Control

Condat AG
Alt Moabit 90a
10559 Berlin
Germany

Telephone: +49.30.39 49 0
Fax: +49.30.39 49 1300
Internet: www.condat.de

0.1 Document History

ID	Author	Date	Status
8434_513_02.001	CSH	1 February, 2002	Being Processed
8434_513_02.002	CSH	4 March, 2002	Being Processed

0.2 References, Abbreviations, Terms

[C_7010.801] 7010.801, References and Vocabulary, Condat AG

1 Introduction

This document explains how you can convert TDS test cases to TDC test cases using the TDS_TO_TDC tool. In the first section we shortly explain what have been changed in the tool chain.

The TDC front end has resulted in some changes in the UMTS test tool chain. Before you will understand this we n eed to take a look at the UMTS test tool chain as it was originally (TDS test tool chain).

1.1 Test tool chain

Now lets have a look at the changes in the tool chain. Figure 1 shows the TDS Test Tool chain.

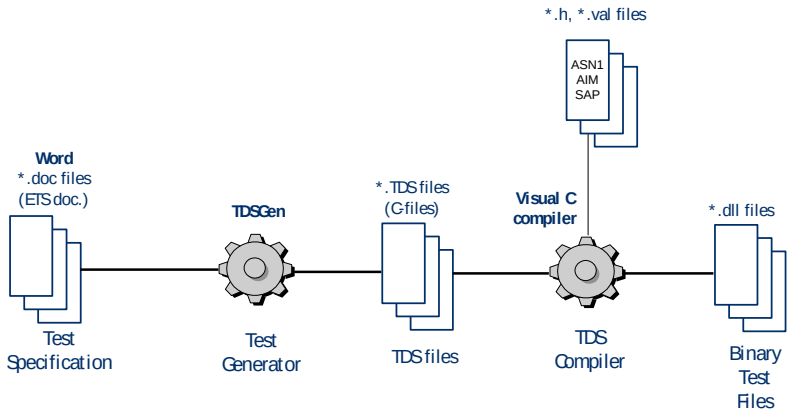

Figure 1 TDS test tool-chain

Using the TDS test tool chain all ETS documents were specified in MS-word. The word documents were saved as text documents and processed by TDSgen. TDSgen parsed the text files into TDS files. During this parsing the ASCII MSC's were converted to SEND or AWAIT macros (test events).

The TDS files are real C files with a lot of macros. The Visual Studio compiler generated binary test files (dll-file) for each test case (tds-file) with the use of some macros (macros.h). These dll's are then processed by the TAP2.exe application.

Figure 2 shows the new test tool chain – the TDC tool chain.

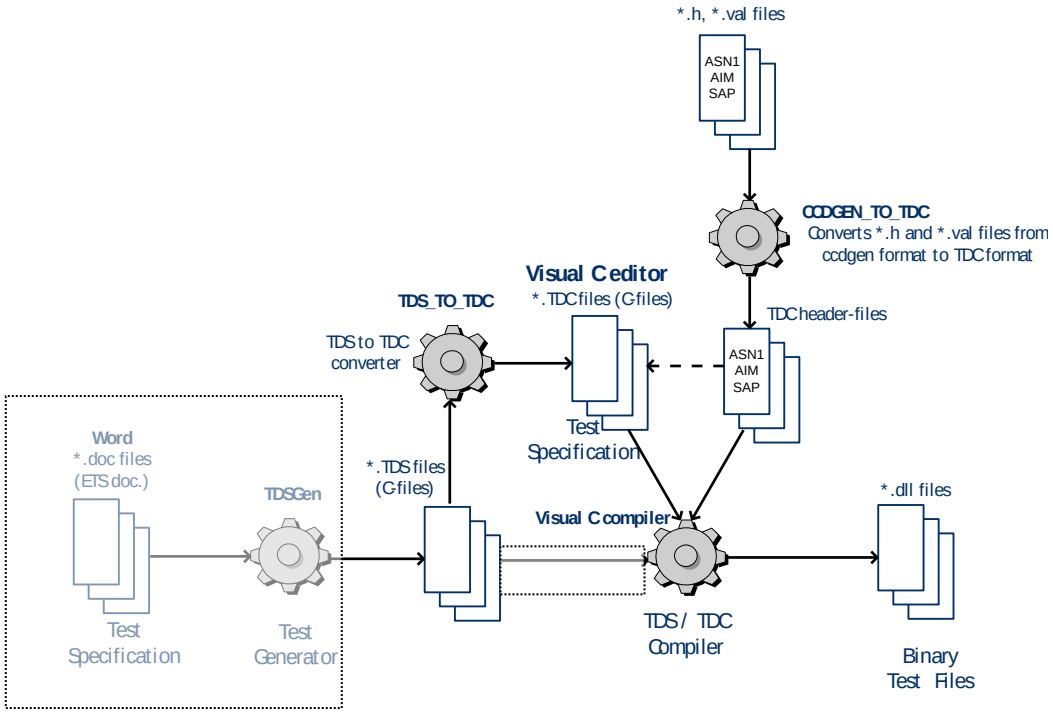

Figure 2 TDC test tool-chain

In the new tool chain the parts in the faded boxes are removed. Word is removed as editor and the output from TDSgen is replaced with another format – the new TDC syntax language. This format is now the test specification and can be edited in a normal C editor. The TDC file is a normal c-file (source code file). The figure shows that one tool is removed (TDSgen) and two new are introduced. The new ones are:

- TDS_TO_TDC.exe – which converts existing test case (tds files) to the new test specification files (tdc files)[1]
- CCDGEN_TO_TDC.exe - which converts generated include files from CCDgen to special TDC include files.

TDS_TO_TDC is only a one time tool, which means that after all ETS word documents (Test Specifications) have been converted this tool will not be used any more. In other words, the TDS_TO_TDC not will be a "real" part of the new tool chain and therefore the chain simplifies to the one in Figure 3.

[1] Please note that hand editing of unions might be needed. This will be covered in section 2 "How to use tds_to_tdc.exe "

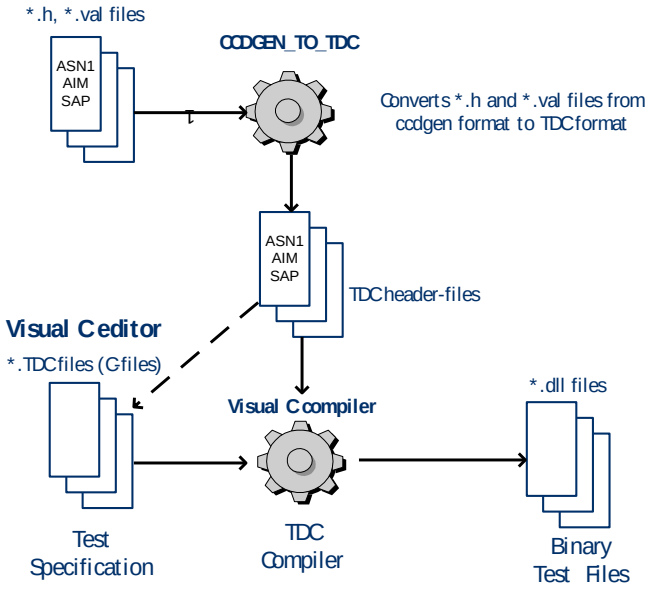

Figure 3 The final TDC test tool chain

Besides using the TDC-header files for generating the dll-file (executable test cases), they also provide type awareness so that dot completion will be available in Visual Studio 6, when specifying test cases. This dependency is indicated with the dotted arrow. CCDGEN_TO_TDC is a part of the makcdg.mak makejob, which means that the files you need for writing test specifications is automatically generated.

2 How to use tds_to_tdc.exe

This section will briefly describe the different parameters of the tds_to_tdc tool and the manual work needed will be explained. For an example of how to convert a test case see the next section.

Tds_to_tdc.exe is a tool developed in perl. The tds_to_tdc.exe file should be called from a 4NT-prompt, after initumts.bat (located under "view_letter:\GPF\initumts.bat") have been called. You call tds_to_tdc like this:

tds_to_tdc <ENTITY> <TESTCASE_DOC_NAME>

<ENTITY> is required.

<TESTCASE_DOC_NAME> is optional.

(If your test case document name is different from the entity name you should add TESTCASE_DOC_NAME – e.g. rrcmeas).

You can add following options:

-d if you want to generate the visual studio project file for you test environment

-v verbose mode

-e error_mode (all warnings are treated as errors – e.g. the conversion stops immediately if an error occurs)

-o specify the output directory where the generated files should be placed

-m add MSC's and descriptions as comments in the test source files.

Examples:

tds_to_tdc rrc rrcmeas

tds_to_tdc -d rlc

tds_to_tdc -d -o c:\temp rlc\

The tds_to_tdc.exe file is located in gpf\bin, but it can be called from any place in your environment.

Please note following:

- In all test step titles non-letter characters will be removed.
- Note that parking is now disabled – this might require that you manually switch this on, to get your test case to pass.
- If you try to convert you test case into a directory where you already have test case files with the same names – these files will be overwritten.
- It is not possible to convert suites.

2.1 Manual work

After using tds_to_tdc.exe, some manual work is required before you are able to compile your converted tdc test cases.

2.1.1 Conversion of union types

In tds the union is not used, instead the chosen union member is used. Therefore the tds_to_tdc.exe cannot know the union type, so you will have to convert these manually.

2.1.2 Pointer types

In the tool chain it is possible to specify pointer types in SAPs and in AIMs. When these types are processed by ccdgen.exe they get the "ptr" extension – see the table below:

Pointer type	No pointer type
Typedef struct { U8 msg_type; U16 bit_len; U8 c_bit_str; **U8 *ptr_bit_str;** U8 v_bit_str_fix; U8 bit_str_fix[2]; U32 integer; } T_XX_DYNARR_CODE_SEQ_A_PTR_REQ;	Typedef struct { U8 msg_type; U16 bit_len; U8 c_bit_str; **U8 bit_str[MAX_BITSTRING];** U8 v_bit_str_fix; U8 bit_str_fix[2]; U32 integer; } T_XX_DYNARR_CODE_SEQ_A_REQ;

Table 1 Types in h-files

When these types are used in the tds format, we can't see if it as pointer type or not. In our word documents we will write like this:

(1) XX_DYNARR_CODE_SEQ_A_REQ

msg_type	ASN1_DL_PTR_R_SEQUENCE_A_MSG_MSG
bit_len	6
bit_str	**BIT_STRING_ARRAY6B**
bit_str_fix	BIT_STRING_ARRAY16A
integer	10

(1) XX_DYNARR_CODE_SEQ_A_PTR_REQ

msg_type	ASN1_DL_PTR_R_SEQUENCE_A_MSG_MSG
bit_len	6
bit_str	**BIT_STRING_ARRAY6B**
bit_str_fix	BIT_STRING_ARRAY16A
integer	10

This means that when convert test cases we can't see if it as pointer or not, so you will have to changes this manually as well.

3 Converting your test cases

This section will give a complete example of a test conversion and some of the common errors. When you want to convert a test case you must do like this (in the following it is assumed that we convert the rrcmeas test case):

1. Start a 4NT prompt – run initumts.bat.

   (Ensure that the path to the compiler is correct. If not, initialize it with the correct path manually – e.g. Call "C:\PROGRAM FILES\Microsoft Visual Studio\vc98\bin\VcVars32.bat")

2. Compile the tds test cases with **gpf\bin\make1intc.bat**:
   Be sure that you have compiled all SAP's in the right version. The SAP version should confirm with your test case version – e.g. if your test cases still are based on release Charlie, you will have to compile the SAPs used on release Charlie - otherwise you test case might not be compilable). Before you can convert you test cases, you must compile them with "makein1tc.bat". This is done in the same way as "maketc.bat" – e.g.
   makein1tc.bat rrcmeas

3. Create the test folder in the source directory:

   E.g. g23m\condat\ms\src\rrc\rrcmeas_test.

4. Now run tds_to_tdc. Go to the directory or add the directory as parameter for tds_to_tdc – e.g.

   tds_to_tdc –d –m –e –o <driveletter>:\g23m\condat\ms\src\rrc\rrcmeas_test rrc rrcmeas

   By using the –e parameter the compilation will stop immediately when encountering an error. However it might be advantageous to be aware of other errors as the same type probably occurs again and thus can be fixed at the same time. For a collection of common errors see the next section.

5. Open the XXX_dsp.file in *g23m\condat\ms\src\rrc\rrcmeas_test* –e.g.:

   rrrmeas_test.dsp

6. Try to compile the test cases. Fix unions, pointer types and bugs.

   Activate autocomplete in visual studio by touching the relevant .h files under "external dependencies" (open a .h file and insert a space and undo it. This way the Visual Studio autocomplete compiler recognizes the file. As this can be somewhat laborious with 20 .h files, a macro solution is on its way.

3.1 Common problems

This section will describe some of the most common problems encountered when converting the test cases. If you encounter a problem, which is not included in this document, please report it to the tool group.

3.1.1 ERROR: Unrecognized parameter line "SET_COMP("ul_gsm900_measurements", " in rrcmeas.def. SKIP_COMP("XX") or SET_COMP("XX",YY) expected. Current line position in input file: 1116

Go to the .def file (g23m\condat\tds\rrcmeas\rrcmeas.def) at line 1116. This problem usually occurs when encountering a line which has been broken into two lines as in the following:

SET_COMP("ul_gsm900_measurements",

CPHY_UL_COMP_MODE_NOT_NEEDED_FOR_GSM900)

As the conversion tool only reads one line at a time this causes an error. Fix the problem by making the two lines into one as in:

SET_COMP("ul_gsm900_measurements", CPHY_UL_COMP_MODE_NOT_NEEDED_FOR_GSM900)

3.1.2 ERROR: Could not find type of array for array_of_struct CUMAC_UL_TRCH_DESCR_A2_RC. Please check CUMAC_UL_TRCH_DESCR_A2_RC in the def file.

Maybe the array was only declared and not defined. Go to the .def file and see at the given line position. In some cases the array can be ignored and turned into a normal C comment by /*….. */ as the array size is val_0.

3.1.3 ERROR: Unrecognized line "xxxxxx" in rrcmeas.def, Current line position in input file: 872

This is normally due to some unwanted characters in a line, probably due to a typo. Remove the unwanted characters in the .def file. Save the file and try again.