ftl2ver

FARADAY DESIGN KIT FTL2VER

User Guide
Rev.: 202001.0.0v1.0
Issue Date: July 2020

REVISION HISTORY

Faraday Design Kit ftl2ver User Guide

Date	Rev.	From	To
Apr. 2003	1.0	-	Original
Sep. 2003	1.1	-	Changed the Vt format
Apr. 2004	2.0	-	Added the message list
Sept. 2006	200601.3.2v.2.1	-	Updated the design kit version to “200601.3.2.v2.1”
Jan. 2007	200701.1.1v1.0	-	Updated the design kit version to “200701.1.1”
Jan. 2008	200801.1.1v1.0	-	Updated the design kit version to “200801.1.1”
Feb. 2009	200802.1.2v1.0	-	• Added the “-noppa” option • Updated the design kit version to “200802.1.2”
Feb. 2009	200901.1.1v1.0	-	200901.1.1 major release
Mar. 2010	201001.1.1v1.0	-	201001.1.1 major release
Apr. 2011	201101.0.0v1.0	-	201101.0.0 major release
May 2012	201201.0.0v1.0	-	201201.0.0 major release
Mar. 2013	201301.0.0v1.0	-	201301.0.0 major release
Nov. 2013	201302.0.0v1.0	-	201302.0.0 major release
Mar. 2014	201401.0.0v1.0	-	2014 major release
Mar. 2015	201501.0.0v1.0	-	2015 major release
Mar. 2016	201601.0.0v1.0	-	2016 major release
Mar. 2017	201701.0.0v1.0	-	2017 major release
Dec. 2018	201806.0.0v1.0	-	• 2018 major release • Support -pin_postfix • Support -use_ftl_timescale
Jul. 2020	202001.0.0v1.0	-	202001 major release

Printed in Taiwan 2020

Faraday Technology Corporation
No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Faraday's home page can be found at:
http://www.faraday-tech.com

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Overview

ftl2ver is a program that converts *ftl* (Faraday Tester interface Language) into the Verilog HDL test bench. It provides an easy way to convert the cycle-based test data to the HDL stimulus for the validation of the test bench.

Figure 1-1 shows the input and output files of the **ftl2ver** process flow.

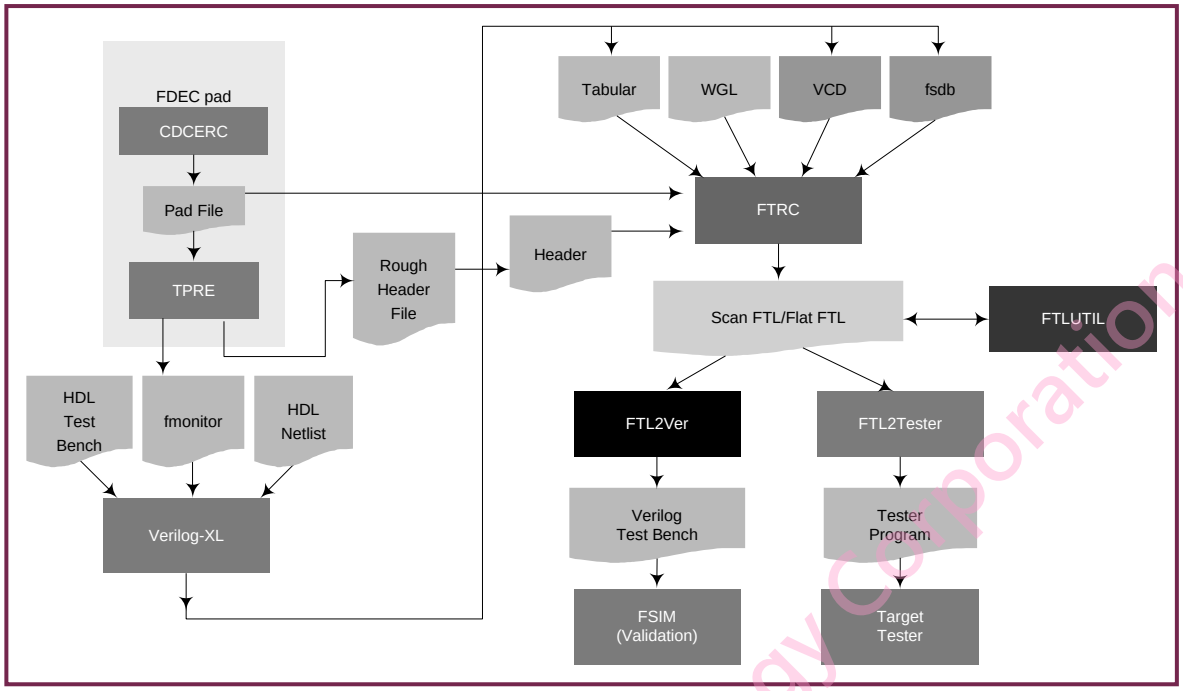

Figure 1-1. Operation Flow

Chapter 2

Command Syntax

The syntax of **ftl2ver** is:

ftl2ver <id> [-noppa]

where:

<id> is the identifier of the *ftl* test vectors. According to the limitation of the tester, <id> can only be supported up to 32 characters.

[-noppa] is an optional command argument. It indicates that **ftl2ver** generates the Verilog test bench without the package information of the ppa file. For the ppa file definition, please refer to the user guide of the **fppa** design kit. This command argument should only be used when the design is an IP or a hard block.

Chapter 3

Input Files

setup.ftc is the FTC setup file. There are eight arguments referred by **ftl2ver**, please refer to the following arguments for details:

```
TOGGLE          = [ON|OFF]
CONTENT         = [ON|OFF]
FLOAT           = [ON|OFF]
MONITOR         = [ON|OFF]
RULECHECK       = [ON|OFF]
CASE_SENSITIVITY = [ON|OFF]
CASE_TO_UPPER   = [ON|OFF]
TRC_TESTER      = [DEFAULT| <tester type>]
DESIGN          = [PROJECT ID ]
```

<id>. *ftl*	This is the *ftl* test vector file. <id> can be any combination of the alphanumeric characters. For example: pat1.*ftl*, ac.*ftl*, dc1.*ftl*, mem.*ftl*, and so on. According to the limitation of the tester, <id> can only be supported up to 32 characters.
<DESIGN>.pad	This is the report of the pad usage generated by **fpad**. It is required to prepare the Verilog stimulus file <id>.vt for **ftl2ver**.
<DESIGN>.ppa	This is the report of the package usage generated by **fppa**.

Because some signals in the design top module port may not have a corresponding output pin in the package and some signals may down-bond to ground in the package. The information in the ppa file, such as the package pins and down-bond pins, should be taken care during the simulation. **ftl2ver** will generate the Verilog test bench according to the pin information and down-bond information of the package. It means that only the package pins have to be simulated and the down-bond condition should be considered during the simulation. If the design is an IP or a hardcore, there might be no package information. Under this condition, users should apply the [-noppa] command argument while executing **ftl2ver**. If users apply the [-noppa] command, **ftl2ver** will generate the Verilog test bench without the package information.

Chapter 4

Output Files

<id>.vt	This is the Verilog stimulus file written in the HDL format. This *.vt* file handles the waveform formats of all the signals and performs the comparison between the simulation results and the expected values of the output signals. A ROM table file (*.rom*) will be required when running the simulations.
<id>.rom	This is the ROM table file. It describes the input values and the expected output values that come from the *ftl* file in the cycle-based representations.
ftl2ver.cmd	This is the intermediate command file.
<id>.flg	This is the execution log file.

Chapter 5

Examples of Input/Output Files

This chapter contains the following sections:

5.1 "*setup.ftc*" Arguments for ftl2ver

There are eight arguments in the *setup.ftc* file referred by **ftl2ver** as described below:

TOGGLE	To evaluate and report the toggle rate during the simulation The default is "ON".
CONTENT	To check the signal contention problem The default is "ON".
FLOAT	To check the signal floating problem The default is "ON".
MONITOR	If this argument is set to "ON", the Verilog output test bench of **ftl2ver** will monitor the primary input and output signals in the simulation, and dump the tabular file for the usage. The default is "OFF".
RULECHECK	To validate the *ftl* with the tester rule This tester rule is the same as the one referred by **ftrc**. The default is "ON".
TRC_TESTER	This argument specifies the family of the desired tester rule in a tester rule file. It can be "DEFAULT" or any available tester. The setting is only valid when "RULECHECK" is set to "ON". About the detailed check items of tester rule, please refer to the user guide of ftrc.
CASE_SENSITIVITY	To perform the comparison of all the signal names listed in the input files, it will be performed in a case-sensitive manner. The default is "ON".
CASE_TO_UPPER	To convert all the signal names of the input files into the upper case The default is "OFF".

The three functions, "TOGGLE", "CONTENT", and "FLOAT", are the Verilog PLI routines developed by Faraday. The re-compiled Verilog-XL module is bundled by the design kit program, **fsim**. Please refer to the Faraday PLI routines for Verilog-XL Simulator document for the detailed information.

5.2 Test Data of *ftl* Format

This section contains the detailed information of the *ftl* format.

The examples of the input/output files for **ftl2ver**, such as the *ftl* file, Verilog stimulus, and the ROM file, are illustrated in Figure 5-1, Figure 5-2, and Figure 5-3. Users can easily understand the relationships among these files through the provided examples.

The following is the original *ftl* file.

```
TESTTYPE FUNC;

INPUT(1) A0,A1;
INPUT(3) DISTRN,DOSTRN;
INPUT(4) RCLK;
OUTPUT(2)    BAUDOT,DDIS ;
INOUT(1,2)   D0,D1,D2,D3,D4,D5,D6,D7 ;

TIMEUNIT     1 NS;
CYCLE        100;
TIMEGEN(1)   DNRZ,0;
TIMEGEN(3)   DNRZ,30;
TIMEGEN(4)   RZ,25,50;
TIMEGEN(2)   STROBE,50,10;
SEQUENCE     A0,A1,DISTRN,DOSTRN,,RCLK,,
             BAUDOT,DDIS,,D0,D1,D2,D3,D4,D5,D6,D7;
BEGIN
0000_1_XX_XXXXXXXX;      // 1 : 0
0000_0_HH_11110000;      // 2 : 1000
1100_0_HL_XXXXXXXX;      // 3 : 2000
1100_1_HL_LLLLHHHH;      // 4 : 3000
0011_0_HH_XXXXXXXX;      // 5 : 4000
0011_0_LL_00001111;      // 6 : 5000
0000_0_LL_XXXXXXXX;      // 7 : 6000
0000_0_LL_HHHHLLLL;      // 8 : 7000
END
```

Figure 5-1. *ftl* Test Vector: *tpl. ftl*

The following is the generated *vt* file.

```
// Date   : Mon Aug 18 14:33:48 2003
// Creator: FTC
// Top Module : TOP


`ifdef fsim_typical_corner
   `define  result_file  "n_tp1.out"
   `define  tog_file  "n_tp1.tog"
   `define  cfl_file  "n_tp1.cfl"
   `define  flt_file  "n_tp1.flt"
   `define  dmp_file  "n_tp1.dmp"
   `define  fsdb_file "n_tp1.dmp.fsdb"
`endif

`ifdef fsim_best_corner
   `define  result_file  "b_tp1.out"
   `define  tog_file  "b_tp1.tog"
   `define  cfl_file  "b_tp1.cfl"
   `define  flt_file  "b_tp1.flt"
   `define  dmp_file  "b_tp1.dmp"
   `define  fsdb_file "b_tp1.dmp.fsdb"
`endif

`ifdef fsim_worst_corner
   `define  result_file  "w_tp1.out"
   `define  tog_file  "w_tp1.tog"
   `define  cfl_file  "w_tp1.cfl"
   `define  flt_file  "w_tp1.flt"
   `define  dmp_file  "w_tp1.dmp"
   `define  fsdb_file "w_tp1.dmp.fsdb"
`endif
```

```
`timescale 10ps/1ps
module TOP_sim;
parameter
        cycle           = 10000,
        input_pattern_file  = "tp1.rom",
        pattern_width       = 23,
        expect_width        = 10,
        input_vectors       = 8;
integer index,m,n,outfile,i;
wire [0:pattern_width] ERR;
reg [pattern_width-1:0] rom[0:input_vectors-1];


// Simulation non-bus I/O port
wire    A0, A1, DISTRN, DOSTRN, RCLK
        , BAUDOT, DDIS, D0, D1, D2
  , D3, D4, D5, D6, D7;

// ROM non-bus I/O port
wire    A0_i, A1_i, DISTRN_i, DOSTRN_i, RCLK_i
        , BAUDOT_e_i, DDIS_e_i, D0_i, D0_e_i, D1_i, D1_e_i, D2_i, D2_e_i, D3_i, D3_e_i
        , D4_i, D4_e_i, D5_i, D5_e_i, D6_i, D6_e_i, D7_i, D7_e_i;

// Input Assignment
assign  {A0_i, A1_i, DISTRN_i, DOSTRN_i, RCLK_i
        , BAUDOT_e_i, DDIS_e_i, D0_i, D0_e_i, D1_i
        , D1_e_i, D2_i, D2_e_i, D3_i, D3_e_i
        , D4_i, D4_e_i, D5_i, D5_e_i, D6_i
        , D6_e_i, D7_i, D7_e_i } = rom[m];


// Add input Offset
DNRZ #(10000,0)         _I_A0(A0,A0_i);
DNRZ #(10000,0)         _I_A1(A1,A1_i);
DNRZ #(10000,3000)      _I_DISTRN(DISTRN,DISTRN_i);
DNRZ #(10000,3000)      _I_DOSTRN(DOSTRN,DOSTRN_i);
RZ   #(10000,2500,5000)  _I_RCLK(RCLK,RCLK_i,m);
DNRZ #(10000,0)         _I_D0(D0,D0_i);
DNRZ #(10000,0)         _I_D1(D1,D1_i);
DNRZ #(10000,0)         _I_D2(D2,D2_i);
```

```
DNRZ #(10000,0)        _I_D3(D3,D3_i);
DNRZ #(10000,0)        _I_D4(D4,D4_i);
DNRZ #(10000,0)        _I_D5(D5,D5_i);
DNRZ #(10000,0)        _I_D6(D6,D6_i);
DNRZ #(10000,0)        _I_D7(D7,D7_i);


// Add output Offset
STROBE  #(10000,4500,2000)   _O_BAUDOT(ERR[5],BAUDOT,BAUDOT_e_i);
STROBE  #(10000,4500,2000)   _O_DDIS(ERR[6],DDIS,DDIS_e_i);
STROBE  #(10000,4500,2000)   _O_D0(ERR[7],D0,D0_e_i);
STROBE  #(10000,4500,2000)   _O_D1(ERR[8],D1,D1_e_i);
STROBE  #(10000,4500,2000)   _O_D2(ERR[9],D2,D2_e_i);
STROBE  #(10000,4500,2000)   _O_D3(ERR[10],D3,D3_e_i);
STROBE  #(10000,4500,2000)   _O_D4(ERR[11],D4,D4_e_i);
STROBE  #(10000,4500,2000)   _O_D5(ERR[12],D5,D5_e_i);
STROBE  #(10000,4500,2000)   _O_D6(ERR[13],D6,D6_e_i);
STROBE  #(10000,4500,2000)   _O_D7(ERR[14],D7,D7_e_i);


// Main Body of stimulus file
// initial $dcalc_path(top,"","");
// initial $sdf_annotate("TOP.sdf",top,,,,,"FROM_MTM");

// Dump File
`ifdef fsim_dump
initial
  begin
    $dumpfile(`dmp_file);
    $dumpvars;
  end

`endif
// Debussy waveform display
`ifdef fsim_fsdb
initial
  begin
     $fsdbDumpfile(`fsdb_file);
     $fsdbDumpvars;
  end
```

```
`endif

// Input assignment
initial
  begin
    // Optional PLI control
    `ifdef toggle
      $toggle_rate_begin(0,`tog_file,TOP_sim);
    `endif
    `ifdef content
      $bus_content_begin(0,`cfl_file,TOP_sim);
    `endif
    `ifdef float
      $bus_float_begin(10000,1,`flt_file,TOP_sim,"TOP.fbs");
    `endif
    $readmemb(input_pattern_file,rom);
    for(m=0;m<input_vectors;m=m+1)
    begin
       #cycle;
    end
    `ifdef fsim_dump
      $dumpflush;
    `endif
    // Optional PLI control
    `ifdef toggle
      $toggle_rate_end;
    `endif
    `ifdef content
      $bus_content_end;
    `endif
    `ifdef float
       $bus_float_end;
    `endif
    $fdisplay(outfile,"====================================");
    $fdisplay(outfile,"Normal Completion");
    $finish;
  end
```

```
initial
  begin
    outfile = $fopen(`result_file);
    $fdisplay(outfile,"O E SignalName      Cycle_no");
    $fdisplay(outfile,"====================================");
  end

always @(ERR[5])
      if($time!=0) $fdisplay(outfile,"%b %b BAUDOT    (%d)",BAUDOT,BAUDOT_e_i, $time/cycle+1);
always @(ERR[6])
    if($time!=0)    $fdisplay(outfile,"%b %b DDIS      (%d)",DDIS,DDIS_e_i,$time/cycle+1);
always @(ERR[7])
    if($time!=0)    $fdisplay(outfile,"%b %b D0        (%d)",D0,D0_e_i,$time/cycle+1);
always @(ERR[8])
    if($time!=0)    $fdisplay(outfile,"%b %b D1        (%d)",D1,D1_e_i,$time/cycle+1);
always @(ERR[9])
    if($time!=0)    $fdisplay(outfile,"%b %b D2        (%d)",D2,D2_e_i,$time/cycle+1);
always @(ERR[10])
    if($time!=0)    $fdisplay(outfile,"%b %b D3        (%d)",D3,D3_e_i,$time/cycle+1);
always @(ERR[11])
    if($time!=0)    $fdisplay(outfile,"%b %b D4        (%d)",D4,D4_e_i,$time/cycle+1);
always @(ERR[12])
    if($time!=0)    $fdisplay(outfile,"%b %b D5        (%d)",D5,D5_e_i,$time/cycle+1);
always @(ERR[13])
    if($time!=0)    $fdisplay(outfile,"%b %b D6        (%d)",D6,D6_e_i,$time/cycle+1);
always @(ERR[14])
    if($time!=0)    $fdisplay(outfile,"%b %b D7        (%d)",D7,D7_e_i,$time/cycle+1);
// Top Module Instance

TOP top(.A0(A0), .A1(A1), .DISTRN(DISTRN), .DOSTRN(DOSTRN)
            , .RCLK(RCLK), .BAUDOT(BAUDOT), .DDIS(DDIS), .D0(D0), .D1(D1)
            , .D2(D2), .D3(D3), .D4(D4), .D5(D5), .D6(D6)
            , .D7(D7));
endmodule
```

```
module DNRZ(out, in);
input in;
output out;
parameter cycle=10, Td = 10;
  assign  #Td out = in;
endmodule


module STROBE(error, sim, exp);
input sim,exp;
output error;
parameter cycle = 10, Tsd=80, Tsw=10;
reg enable, error;
initial
begin
  error = 0;
  enable = 0;
  while (1)
    begin
      #Tsd enable = ~enable;
      #Tsw enable = ~enable;
      #(cycle - Tsd - Tsw);
    end
 end
always @(sim or enable)
  if ((enable == 1'b1) & (exp !== 1'bx) & (sim !== exp))
     error = ~error;
endmodule

module RZ(out, in, cycle_begin);
input in, cycle_begin;
output out;
reg out;
parameter cycle = 10, Td = 10, Tp = 20;
always @(cycle_begin)
begin
      case (in)
            1'b0: out = 0;
            1'b1: begin
```

```
                out = 0;
                #Td out = ~out;
                #Tp out = ~out;
                end
            1'bz: out = 1'bz;
          default : out = 1'bx;
        endcase
    end
    endmodule
```

Figure 5-2. Generated *vt* File

```
00001ZZZZZZZZ_XXXXXXXXXX
0000011110000_1111110000
11000ZZZZZZZZ_10XXXXXXXX
11001ZZZZZZZZ_1000001111
00110ZZZZZZZZ_11XXXXXXXX
0011000001111_0000001111
00000ZZZZZZZZ_00XXXXXXXX
00000ZZZZZZZZ_0011110000
```

Figure 5-3. Generated ROM File

Chapter 6

Supported Bus

The bus signal declaration in the *ftl* file is supported in this release. To support the bus signal declaration, the *ftl* file must follow the rules below:

1. The bus signals can be defined in the following three different formats and each signal can have the own timegen. The followings are the examples of the three acceptable bus notations:
   a. INPUT(2) mybus[5:0];
   b. INPUT(2) mybus[5:3];
      INPUT(2) mybus[2:0];
   c. INPUT(2) mybus[5];
      INPUT(2) mybus[4];
      INPUT(2) mybus[3];
      INPUT(2) mybus[2];
      INPUT(2) mybus[1];
      INPUT(2) mybus[0];

2. The orientation of the declared bus signal in the *ftl* header section must be identical, i.e., all from the low bit to the high bit or all from the high bit to the low bit.
3. The indices of a bus signal must be continuous.

4. The bus width of the bus signal in the *ftl* file must be the same as the one in the Verilog netlist.
5. The bus orientation of the bus signal in the *ftl* file must be the same as the one in the Verilog netlist. The following is an example of a Verilog module:

```
module test(a,b,c) ;
input [5:0] a;
input [0:5] b;
output [3:0] c;
...
endmodule
```

The following is the corresponding bus declaration in the *ftl* pattern.

```
TESTTYPE FUNC;
INPUT(1) a[5:0];
INPUT(2) b[0:5];
OUTPUT(3) c[3:0];
```

6. The bus notation is also supported in the SEQUENCE declaration of an ftl pattern. However, the orientation of the bus does not have any limitation. For example, the SEQUENCE of the above example can be declared as follow:

```
SEQUENCE a[3], a[2] ,a[5], a[4], a[0] , b[5:0], c[1:0], c[2:3];
```

Chapter 7

Parallel Simulation

To reduce the simulation time of the parallel simulation at different corners, the generated *.vt* file is enhanced from the major release of this design kit. The output files of the simulation, such as *.tog*, *.flt*, *.cfl*, *.out*, *.log*, *.dmp*, *.dmp*, and *.fsdb*, will have different names at different corners. The naming rule of these simulation output files is <case>_<pattern>.xxx.

Chapter 8

Message List

This chapter contains the following sections:

The followings are the message lists reported by the **flt2ver**.

8.1 System Related Messages

"err_sys001"	"can't allocate memory while processing line<%d>\n"
"err_sys002"	"environment variable<%s> not defined\n"
"err_sys003"	"can't open bus parameter file<%s>\n"
"err_sys004"	"section<%s> not existed in bus parameter file<%s>\n"
"err_sys005"	"no BUSRANGE or BUSINDEX in section<%s> of bus parameter file<%s>\n"
"err_sys006"	"fail to check out license feature<%s>\n"

8.2 Command File Related Messages

"err_cmd001"	"required argument<%s> for %s absent\n"
"err_cmd002"	"can't open %s file<%s>\n"
"err_cmd003"	"unknown target format<%s>\n"
"err_cmd004"	"unknown argument<%s>\n"

8.3 Pad File Related Messages

"err_pad001"	"pin<%s> declared in *ftl* file but has no match in *pad* file\n"
"err_pad002"	"pin<%s> type mismatch between *pad* file<%s> with *ftl* file<%s>\n"
"err_pad004"	"not an illegal *pad* file<%s>\n"
"err_pad005"	"error open *pad* file<%s>\n"
"err_pad006"	"unknown type of pin<%s> in p*ad* file\n"
"wng_pad001"	"pin<%s> declared in *pad* file but has no match in *ftl* file\n"
"wng_pad002"	"INOUT pin<%s> declared in *pad* file is assigned as %s pin in *ftl* file\n"

8.4 *ppa* File Related Messages

"err_ppa001"	"error open *ppa* file<%s>\n"
"err_ppa002"	"no PPA_BEGIN found in *ppa* file<%s>\n"
"wng_ppa001"	"pin<%s> defined in *ppa* file but absent in FTL\n"

8.5 Rule File Related Messages

"err_rul001"	"can't open rule file<%s>\n"
"err_rul002"	"unknown field name<%s> in rule file at line<%d>\n"

8.6 *ftl* File Related Messages

"wng_ftl001"	"Stable region reaches %s cycle boundary and will be trimed.\n"
"err_ftl001"	"size of CYCLE<%.f%s> violates rule (min:%.f%s max:%.f%s)\n"
"err_ftl002"	"total vector length<%d> (cycle number) exceeds the limit of MAX_VECTOR<%.f>"
"err_ftl003"	"number of declared input *timegen*<%d> exceeds rule<%.f>\n"
"err_ftl004"	"number of declared output *timegen*<%d> exceeds rule<%.f>\n"
"err_ftl005"	"pin<%s> with *timegen*<%d> violates the rule %s (%.f%s %s %.f%s)\n"
"err_ftl006"	"pin<%s> associated with an undeclared *timegen* number<%d>\n"
"err_ftl007"	"decalred *timegen* index<%d> exceeds the limit<%d> at line<%d>\n"
"err_ftl008"	"pin<%s> declared but absent in SEQUENCE statement\n"
"err_ftl009"	"no FTL pattern found in *ftl* file\n"
"err_ftl010"	"number of declared pins<%d> mismatches SEQUENCE pins<%d>\n"
"err_ftl011"	"CYCLE not defined in *ftl* file\n"
"err_ftl012"	"TIMEUNIT not defined in *ftl* file\n"
"err_ftl013"	"SEQUENCE not defined in *ftl* file\n"
"err_ftl014"	"bit stream length<%d> not equal to pin count<%d> at line<%d>\n"
"err_ftl015"	"unknown pattern bit<%c> assigned to input signal<%s> at line<%d>\n"
"err_ftl016"	"unknown pattern bit<%c> assigned to output signal<%s> at line<%d>\n"
"err_ftl017"	"unknown pattern bit<%c> assigned to inout signal<%s> at line<%d>\n"
"err_ftl018"	"declared input timegen numbers exceeds limitation<%d> of STS tester\n"
"err_ftl019"	"declared output timegen numbers exceeds limitation<%d> of STS tester\n"

"err_ftl020"	"duplicate pin<%s> in signal declaration\n"
"err_ftl021"	"bus subset order not consistent in pin<%s>\n"
"err_ftl022"	"bus subset type not consistent in pin<%s>\n"
"err_ftl023"	"duplicate bus index<%d> in pin<%s>\n"
"err_ftl024"	"duplicate SEQUENCE section in FTL\n"
"err_ftl025"	"pin<%s> used in SEQUENCE but not declared before at line<%d>\n"
"err_ftl026"	"pin<%s> defined as a bus pin in SEQUENCE but declared as non-bus pin in signal section at line<%d>\n"
"err_ftl027"	"duplicate pin<%s> in SEQUENCE\n"
"err_ftl028"	"bus bits of pin<%s> defined in SEQUENCE exceeds its illegal range\n"
"err_ftl029"	"pin<%s> used in MASK but not declared before at line<%d>\n"
"err_ftl030"	"input pin<%s> can not be masked at line<%d>\n"
"err_ftl031"	"No support SBC waveform\n"
"err_ftl032"	"bus pin<%s> is not continuous or has duplicate bits at pad file\n"
"err_ftl033"	"input timegen of pin<%s> binds to invalid timegen category\n"
"err_ftl034"	"output timegen of pin<%s> binds to invalid timegen category\n"
"err_ftl035"	"scan signal<%s> not declared before at line<%d>\n"
"err_ftl036"	"null scan pattern for signal<%s> at cycle<%d>\n"
"err_ftl037"	"Strobe time %d ps of timegen number %d equals to clock period, this may cause simulation error."
"err_ftl038"	"File name %s is not supported by tester (too many dot), please change it."