
FaultTree+ V11 Dynamic Link Library

Technical Specification

Platform

Runs under Windows 95, 98, NT, 2000, Me and Xp.

Summary of capabilities :

The FaultTree+ dynamic link library (DLL) allows users to construct applications that interface to the FaultTree+ program.

Users may construct new FaultTree+ project files that may be opened using FaultTree+ V10.1 and later versions. In addition users may edit, or read data from, FaultTree+ project files that have been constructed by the FaultTree+ program itself.

The FaultTree+ DLL may also be used to send messages to the FaultTree+ program whilst it is running on the same machine. The DLL also provides a routine to start up the FaultTree+ program and open an existing project file. Messages may only be sent to FaultTree+ V10.1.2 and later versions if it is licensed to run in fully operational mode. Messages may not be sent to FaultTree+ if it is operating in demonstration mode. One of the primary functions of the message system is to allow external applications to construct and modify fault tree projects and then use FaultTree+ to perform the calculations. Results may then be saved within the FaultTree+ project file and read into the external application.

Users may therefore build up applications to

- Automatically construct fault trees from design models stored in other applications
- Automatically manipulate failure data and perform special global edit operations
- Provide a high level interface to manipulate fault tree data and assess the effects of changes (Living PSA)
- Develop operational risk monitors
- Provide special graphical data input and data display capabilities

Example External Application

The following example coding is written using the Microsoft Visual C++ development environment. Similar coding might be written in the Visual Basic and other development environments.

```
// =====
// Test 1 - Create a new fault tree using the dll only and then automatically
// start up FaultTree+ with the new project
// =====

// First clear any existing data (this isn't strictly necessary here as
// we haven't opened a project or created any data yet.

PSADLLNewProject ( ) ;

// Create the gates that will be in our fault tree

gindex[0] = PSADLLCreateRecord ( "Gates" , "TOP" ) ;
gindex[1] = PSADLLCreateRecord ( "Gates" , "G1" ) ;

// Create the events that will be in our fault tree

eindex[0] = PSADLLCreateRecord ( "Events" , "PUMP1" ) ;
eindex[1] = PSADLLCreateRecord ( "Events" , "PUMP2" ) ;
eindex[2] = PSADLLCreateRecord ( "Events" , "POWER" ) ;
eindex[3] = PSADLLCreateRecord ( "Events" , "HOUSE1" ) ;

// Create the generic failure models

minindex[0] = PSADLLCreateRecord ( "Generic Models" , "PUMP" ) ;

// Define the tree structure (note that we must do this after the
// gates and events have been created)

PSADLLSetFieldValue ( "Gates" , "Input Type 0" , gindex[0] , "" , PSA_GATE , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Index 0" , gindex[0] , "" , gindex[1] , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Type 1" , gindex[0] , "" , PSA_PEVENT , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Index 1" , gindex[0] , "" , eindex[2] , 0.0 ) ;

PSADLLSetFieldValue ( "Gates" , "Input Type 0" , gindex[1] , "" , PSA_PEVENT , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Index 0" , gindex[1] , "" , eindex[0] , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Type 1" , gindex[1] , "" , PSA_PEVENT , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Index 1" , gindex[1] , "" , eindex[1] , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Type 2" , gindex[1] , "" , PSA_PEVENT , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Input Index 2" , gindex[1] , "" , eindex[3] , 0.0 ) ;

// Set the gate types

PSADLLSetFieldValue ( "Gates" , "Type" , gindex[0] , "" , PSA_OR , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Type" , gindex[1] , "" , PSA_AND , 0.0 ) ;

// Set the retain results flag for the TOP gate

PSADLLSetFieldValue ( "Gates" , "Retain Results Flag" , gindex[0] , "" , 1 , 0.0 ) ;

// Set the gate descriptions

PSADLLSetFieldValue ( "Gates" , "Description" , gindex[0] , "Pump standby system
failure" , 0 , 0.0 ) ;
PSADLLSetFieldValue ( "Gates" , "Description" , gindex[1] , "Both pumps fail" , 0 ,
0.0 ) ;

// Set the event descriptions and types

PSADLLSetFieldValue ( "Events" , "Description" , eindex[0] , "Pump 1 primary failure"
, 0 , 0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Description" , eindex[1] , "Pump 2 primary failure"
, 0 , 0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Description" , eindex[2] , "Power supply failure" ,
0 , 0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Description" , eindex[3] , "House event" , 0 , 0.0 )
;
PSADLLSetFieldValue ( "Events" , "Type" , eindex[3] , "" , PSA_HOUSE , 0.0 ) ;
```

Isograph Reliability Software

www.isograph-software.com

```
PSADLLSetFieldValue ( "Events" , "Logic Mode" , eindex[3] , "" , PSA_FAILED_MODE , 0.0
) ;

// Set the model descriptions, type and failure rate

PSADLLSetFieldValue ( "Generic Models" , "Description" , mindex[0] , "Pump failure
model" , 0 , 0.0 ) ;
PSADLLSetFieldValue ( "Generic Models" , "Type" , mindex[0] , "" , PSA_RATE , 0.0 ) ;
PSADLLSetFieldValue ( "Generic Models" , "Parameter 0" , mindex[0] , "" , 0 , 2.15e-2
) ;

// Assign generic model to pump events

PSADLLSetFieldValue ( "Events" , "Generic Model Flag" , eindex[0] , "" , 1 , 0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Generic Model Index" , eindex[0] , "" , mindex[0] ,
0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Generic Model Flag" , eindex[1] , "" , 1 , 0.0 ) ;
PSADLLSetFieldValue ( "Events" , "Generic Model Index" , eindex[1] , "" , mindex[0] ,
0.0 ) ;

// Set local model data for power supply event

PSADLLSetFieldValue ( "Events" , "Local Model Type" , eindex[2] , "" , PSA_RATE , 0.0
) ;
PSADLLSetFieldValue ( "Events" , "Local Model Parameter 0" , eindex[2] , "" , 0 ,
4.67e-3 ) ;

// Modify project options

PSADLLGetProjectOptions ( &pdef ) ;
pdef.life_time = 2.0 ;
pdef.custom_mode = 1 ;
pdef.ft_expand_sets = 1 ;
PSADLLSetProjectOptions ( &pdef ) ;

// Save the database information to a FaultTree+ project file

PSADLLSaveProject ( "c:\\psa2002dlltest\\test1.psa" ) ;

// Now start up FaultTree+ and open the project

err = PSADLLStartFaultTreePlus ( "c:\\Program Files\\RAMS\\Ftp\\10.1\\Program" ,
"c:\\psa2002dlltest\\test1.psa" , 0 , 1000 ) ;

PSADLLSendMessageToFaultTreePlus ( PSA_DLL_PERFORM_ANALYSIS ) ;

PSADLLSendMessageToFaultTreePlus ( PSA_DLL_SAVE_PROJECT ) ;

PSADLLSendMessageToFaultTreePlus ( PSA_DLL_EXIT ) ;

PSADLLOpenProject ( "c:\\psa2002dlltest\\test1.psa" ) ;

PSADLLGetFieldValue ( "Gates" , "Unavailability" , gindex[0] , "" , &ival , &q ) ;

sprintf ( string , "Q=%g" , q ) ;
MessageBox ( h_window , string , "DLL Tester" , MB_OK ) ;
```