TETware *professional* RT

Embedded Realtime System Testing

Embedded Realtime Systems run in a variety of hardware and operating environments. POSIX 1003.13 defines four different profiles for realtime applications. Some of these need not support a file-system or multiple processes.

A testing strategy for systems conforming to the .13 Profiles needs to reflect real products. TETware *professional* cannot be used in the more restrictive profiles, since it makes certain assumptions about its operating environment that are not valid in these profiles. Modifications have been made to TETware *professional* in order to enable it to control the execution of tests on embedded Realtime Systems. This enhanced product is called TETware *professional* RT.

TETware *professional* Realtime (TETware *professional* RT) is an extension to TETware *professional*, which enables TETware *professional* to control the execution of tests on Embedded POSIX Realtime Systems that cannot support TETware *professional* directly.

TETware *professional* is designed to operate on systems that support at least the functionality described in POSIX 1003.1 (1990). The POSIX standard for Embedded Realtime Systems (POSIX 1003.13) defines four profiles for realtime systems, three of which do not include all the functionality described in P1003.1. Therefore, TETware *professional* cannot be used to execute test cases directly on these systems.

POSIX 1003.13 Profiles

The POSIX 1003.13 Profiles 51 to 54 provide four levels of functionality to which realtime environments can conform. The profiles are based on a study of existing commercial practice, though most vendors have products that fall in a continuum covering the range of functionality that the profiles describe in snapshots.

All Profiles include some or all of POSIX .1, .1b and .1c, and some parts of POSIX .2 and .2a for the development platform (which is likely to be on a host system for Profiles 51-53).

The only one of these profiles that TETware *professional* can be used to support tests on is Profile 54.

To support testing under Profiles 51 to 53 The Open Group has developed TETware *professional* RT.

		Process	
		Single	Multi
File System	No	51	53
	Yes	52	54

System Architecture

In a ‘‘normal’’ non-distributed testing setup, both TETware *professional*, and the test cases that it processes, run on the same system. The whole process is driven by a list of test cases contained in the scenario file. A Test Case Manager (TCM) module is linked into each test case executable. The TCM calls each Test Purpose (TP) function in turn. Each TP completes whatever processing is necessary to perform the test, and then calls an API function to record a result. When all the TP functions have been called, the Test Case Manager exits. Finally, TETware *professional* gathers the results of each TP, writes them to the journal and moves on to the next test case (Figure 1).

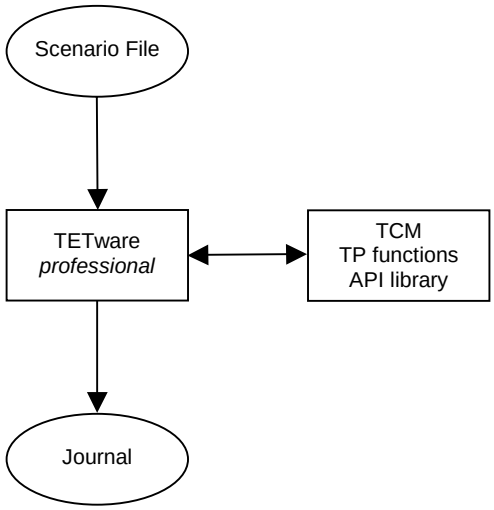

Figure 1: TETware *professional* non-distributed testing

When testing embedded realtime systems, this model needs to be modified. This is mainly for the following reasons:

- TETware *professional* cannot run on the realtime system. All the control operations must be performed on a host system.
- Operating system facilities on the realtime system may be limited. If a test case malfunctions on the realtime system, it may be necessary to reset the system in order to regain control.

The required modification is accomplished by using TETware *professional*’s **exec tool** facility to run the TETware *professional* RT Test Manager on the host system (that is: the system on which TETware *professional* runs). The Test Manager acts as an agent for the test case that is running on the realtime system (Figure 2).

The RT Test Manager

TETware *professional* executes a new instance of the Test Manager each time it executes a test case. The Test Manager performs the following operations:

1. Read information from the test manifest, including information about the arrangement of ICs and TP functions in the test case.
2. Use the dynamic test case interface to adapt itself to the IC/TP arrangement described in the test manifest.
3. Load the test case onto to realtime system and execute it.
4. Open a communication channel to the test case on the realtime system.
5. Instruct the TCM on the realtime system to invoke the test case's startup function, TP functions and cleanup function.
6. For each of these functions, enter a service loop, responding to requests from the TCM/API on the realtime system. The loop is terminated when the function returns to the realtime system's TCM, or when a timeout expires.
7. Deliver a TP function's result to the journal.
8. If the TP timed out: Reset the realtime system.

Thus the Test Manager provides the interface between TETware *professional* running on the host system, and the test case running on the realtime system. From TETware *professional's* point of view the Test Manager looks like an API-conforming test case.

The RT TCM and API

Each test case that is to run on the realtime system is linked with the TETware *professional* RT versions of the C TCM and API library. As in TETware *professional*, both single-threaded and thread-safe versions of these components are supplied. A substantial subset of the API functions implemented in TETware *professional*-Lite is available in the TETware *professional* RT version of the API library.

Although the supported API functions are the same, in many cases the implementations are quite different. For example, functions that write information to the execution results file in TETware *professional* instead send the information to the Test Manager in TETware *professional* RT. When the Test Manager receives this information, it writes the information to the execution results file on the host system.

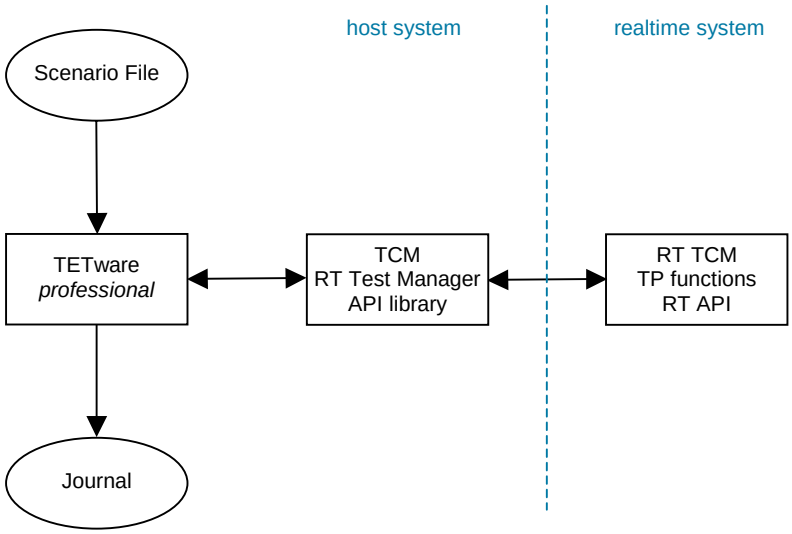

Figure 2: TETware *professional* RT

Manufacturer-specific Subsystems

An interface has been defined which enables TETware *professional* RT components to send requests to other hardware and software subsystems. The implementation of the underlying functionality is specific to the hardware and/or software involved, and is implemented by (or on behalf of) the suppliers of these components.

The subsystems are described in more detail below. For additional information please refer to the TETware *professional* Realtime and Embedded Systems Extension; Installation, User, Demonstration and Programmers Guide, available from the TETworks website.

Functions that are implemented on the host system are used by the Test Manager, and functions that are implemented on the realtime system are used by the TETware *professional* RT version of the Test Case Manager and API library.

Communication Subsystem

Provides two-way communication between the Test Manager on the host system and the TETware *professional* RT TCM/API on the realtime system.

This subsystem consists of two parts; one part on the host system and the other on the realtime system. Each part is responsible for establishing a communication channel to the other part, and for exchanging fixed length message packets over the channel. Typically this subsystem is implemented using TCP/IP (if the realtime system supports it) or a connection between serial ports on each system.

Exec Subsystem

Loads a test case executable on to the realtime system and executes it; and, provides a profile-independent mechanism for test case termination on the realtime system.

This subsystem consists of two parts; one part on the host system and the other on the realtime system.

The part on the host system is responsible for copying a program image file onto the realtime system and executing it. The part on the realtime system is used to terminate a running program, as if exit() has been called by the program.

Reset Subsystem

Resets the realtime system.

This subsystem consists of a single part on the host system. It is responsible for initializing the realtime system to a known state.

The following operations are defined: Soft reset; Hard reset.

Normally the Test Manager requests a soft reset if a test purpose times out, or if it is necessary to interrupt the currently running test purpose for some reason. If the soft reset operation fails, the Test Manager requests a hard reset. This process is analogous to sending a SIGTERM signal to a process running on a UNIX system, followed up by a SIGKILL signal if the process has not terminated within a reasonable time. If only one type of reset is possible for a particular realtime system, then it should be performed in response to both types of reset request.

For more information

Please contact the TETware team:

Tonya Henderson
Americas and Pacific Rim
+1 415 374 8285

t.henderson@opengroup.org

Alan Haffenden
Europe, Middle East & Africa
+44 (0) 118 902 3061

a.haffenden@opengroup.org