1. INTRODUCTION

1.1 Overview

This document is intended to serve the following major purposes:

- To serve as a reference source for the definition of the external interfaces to services that are provided by all Real Time Executive environments. This includes source-code interfaces and run-time behavior as seen by an application-program. It does not include the details of how the kernel implements these functions.

- To serve as a complete definition of Real Time Executive external interfaces, so that application source-code that conforms to these interfaces, will execute as defined in all Real Time Executive environments. It is assumed that source-code is recompiled for the proper target hardware. The basic objective is to facilitate the writing of applications-program source-code that is directly portable across all Real Time Executive implementations.

This document describes the basic set of functionality that makes up the Base System. This functionality has been structured to provide a minimal, stand alone run-time environment for application-programs originally written in a high-level language, such as C.

Other extensions to this Base System will be defined as a continuing effort to produce this standard Real Time Executive Run Time Environment.

It is anticipated that all conforming systems must support the source code interfaces and runtime behavior of the Base System. A system may conform to some, none, or all of the extensions.

1.2 Definitions

executive	That portion of software that constitutes the kernel or performs specific services on behalf of programs tasks.
Real Time Executive	Same as executive.
node	A processor within a multiprocessor system configuration.
local node	The processor within a multiprocessor system configuration on which the current operation is being executed.
remote node	A processor within a multiprocessor system configuration on which the current operation is *not* being executed.
target	The destination remote node in a multiprocessor system configuration.

1.3 Typedefs and Structures

For ease of documentation, the following typedefs are used in this document.

```
typedef    unsigned int    uint;       /* 32-bit unsigned integer */

typedef    void            (*ptf)();   /* pointer to a function that returns nothing */
```

LIST OF TABLES

2. Basic System Services

The Basic System Services is intended to support a minimal run-time environment for executable applications. The Basic System Services defines a set of Real Time Executive components needed by applications-programs. This basic set would be supported by any conforming system. It defines each component's source-code interface and run-time behavior, but does not specify its implementation. Source-code interfaces described are for the C language.

While only the run-time behavior of these components is supported by the Basic System Services, the source-code interfaces to these components are defined because an objective of the Real Time Executive Interface Definition is to facilitate application-program source-code portability across all Real Time Executive implementations. It is assumed that an application-program targeted to run on a system that provides only the Basic System Services (a run-time environment) would be compiled on a system supporting software development.

3. EXECUTIVE FACILITIES

The facilities of the executive have been grouped by function, and are discussed in the following paragraphs.

TABLE 1. Directives

Name	Input Parameters					Output Parameters
t_create	name	superstk	userstk	priority	flags	&tid
t_ident	name	node				&tid
t_start	tid	saddr	mode	argp		
t_restart	tid	argp				
t_delete	tid					
t_suspend	tid					
t_resume	tid					
t_setpri	tid	priority				&ppriority
t_mode	mode	mask				&pmode
t_getreg	tid	regnum				&regval
t_setreg	tid	regnum	regval			
q_create	name	count	flags			&qid
q_ident	name	node				&qid
q_delete	qid					
q_send	qid	buffer				
q_urgent	qid	buffer				
q_broadcast	qid	buffer				&count
q_receive	qid	buffer	flags	timeout		
ev_send	tid	event				
ev_receive	eventin	flags	timeout			&eventout
as_catch	asraddr	mode				
as_send	tid	signal				
as_return						
sm_create	name	count	flags			&smid
sm_ident	name	node				&smid
sm_delete	smid					
sm_p	smid	flags	timeout			
sm_v	smid					
tm_set	timebuf					
tm_get	timebuf					
tm_wkafter	ticks					
tm_wkwhen	timebuf					
tm_evafter	ticks	event				&tmid
tm_evwhen	timebuf	event				&tmid
tm_cancel	tmid					
tm_tick						
i_return						
k_fatal	errcode					