

**Release Notes
RTS Version 3.9.18
12 November 2000**

1.0 Overview

This document describes changes included within:

<u>Module Name</u>	<u>Module Description</u>
SysConf	System Configuration
TPS	TYX Programming Support
PORF	PAWS Output Report Formatter
Targetter	ATLAS Program Targetter
XUtil	TYX X-Window Utilities
WinUtil	TYX MS Windows Utilities
PLI	PAWS LAPS Interpreter
PTE	PAWS Test Executive
CEM	PAWS CIIL Emulation Module

Note: SysConf is used by TPS, PORF, Targetter, XUtil, WinUtil, PLI, PTE and CEM
TPS is used by PORF, Targetter, XUtil, WinUtil, PLI, PTE and CEM.
XUtil and WinUtil are used by PLI and PTE.

<u>Module</u>	<u>Changes</u>	<u>Rebuilt</u>	<u>Current Version</u>
SysConf	None	N/A	20000501
TPS	None	No	20000501
PORF	None	No	20000508
Targetter	None	No	20000508 3.8.3
Xutil	None	No	20000508
WinUtil	None	No	20000508
PLI	None	No	20000508 3.9.13
PTE	Minor	Yes	20001112 3.9.18
CEM	Minor	Yes	20001112 3.9.18

PTE means both the Run-Time System (RTS) and the Simulator (SIM).

PTE, RTS, CEM and SIM mean all Platforms.

UPTE, URTS, UCEM and USIM mean UNIX Platforms only.

WPTE, WRTS, WCEM and WSIM mean MS-Windows Platforms only.

XPTE, XRTS and XSIM mean X-Window Platforms only.

NOTICE TO CEM USERS

This version of the RTS is compatible with CEM Modules built with CEM Files distributed with RTS Version 19980714 3.9.7 and later. You may install this version of the RTS without rebuilding your CEM Module. However, in order to make use of the new CEM capabilities, you **MUST** install this version of the RTS, and then you **MUST** recompile and relink your CEM Module(s) with the TYX-supplied CEM Files distributed with this version of the RTS.

As time goes on, RTS/CEM testing to provide error-free backwards-compatibility becomes increasingly difficult. Therefore, TYX Corporation very strongly recommends that CEM Users

recompile and relink their CEM Modules after installing a new version of the RTS.

1.1 Enhancements

PTE Non-Removal Of Temporary Data File During Crash

When PTE loads an ATLAS Program, PTE makes a copy (File Extension .TMP) of the Data File (File Extension .DAT) for use during execution of the ATLAS Program. When PTE unloads an ATLAS Program, the Temporary Data File is removed.

Prior to this release, the Temporary Data File was always removed, regardless of the reason for the unloading of the ATLAS Program.

With this release, the Temporary Data File is removed unless the RTS is unloading the ATLAS Program as a result of a PTE “crash” (i.e., a catastrophic failure). This allows post-crash analysis of the Temporary Data File.

CEM Kernel Detection Of Invalid Device/Channel Information

The RTS and CEM communicate using a Client/Server protocol based on the IEEE-488 Protocol. When the RTS desires to get data from the CEM (e.g., a Fetch or Status Request), the sequence of Messages between the RTS and CEM are:

<u>From</u>	<u>To</u>	<u>Message</u>	<u>Purpose</u>
RTS	CEM	COMMAND	Command CEM Device/Channel to “listen”
RTS	CEM	WRITE	Specify operation (FETCH, STATUS, etc.)
RTS	CEM	COMMAND	Command CEM Device/Channel to “unlisten”
RTS	CEM	COMMAND	Command CEM Device/Channel to “talk”
CEM	RTS	READ	Acquire Data
RTS	CEM	COMMAND	Command CEM Device/Channel to “untalk”

The “listen” Command contains the Device/Channel information used by the CEM Kernel to call the appropriate CEM Device Driver ATLAS Action User Functions. If the Device/Channel information is invalid (usually as a result of a programming error), the CEM Kernel generates an Error Message.

Prior to this release, the CEM Kernel would also generate an Error Message if the “talk” Command contained invalid Device/Channel information. This resulted in continual Error Message being generated when the RTS was attempting (via Status Requests) to acquired queued Error Messages, thereby causing an infinite loop that could only be stopped by the Operator aborting the RTS manually (i.e., outside of the RTS).

With this release, the CEM Kernel no longer generates an Error Message if the “talk” Command contains invalid Device/Channel information. This scheme works because Device/Channel information is not used during the processing of either the “talk” Command or the Read processing (when the CEM Kernel is simply sending data (previously provided to the CEM Kernel by CEM Device Drivers) to the RTS).

CEM Kernel Conversion Of Datum Decimal Values

The CEM Kernel is responsible for converting Datum Decimal Values (specified by CEM Device Drivers) to ASCII Strings for transmission to the RTS.

Prior to this release, the CEM Kernel converted these Values to “Decimal Strings” in the form:

```
si...idf...f
```

where:

“s” is the sign of the Value;
“i...i” is the Integer Portion of the Value;
“d” is the Decimal Point; and
“f...f” is the Fraction Portion of the Value.

Given that the various Floating-Point Formats supported by PTE and CEM contain only enough bits to provide accuracy to about 15 or 16 digits, this format caused no problems for Values in the range: $10^{*-15} \leq \text{Absolute Value} \leq 10^{*+15}$ because the Character Array used during the conversion contained over 100 Characters. Additionally, reasonable Values that fell outside of this range caused no real problems either (even though the Decimal String became longer and longer). However, extremely large or small Values (possibly caused by a software error that results in an invalid Floating-Point Number) would cause the Decimal String to overflow the Character Array.

With this release, the CEM Kernel converts these Values to “Exponential Strings” in the form:

```
sdf...fExee
```

where:

“s” is the sign of the Value;
“d” is the Decimal Point;
“f...f” (limited to 15 Digits) is the Fraction Portion (or Mantissa) of the Value;
“E” is the ASCII Character “E”;
“x” is the sign of the Exponent; and
“ee” is the Exponent.

WCEM Kernel Programming Error Message Number 30 Changed to 19

The CEM Kernel contains (as does most TYX software) various safety checks for the detection of software errors. When one of these safety checks fails, the CEM Kernel sends a “Programming Error” Error Message to the RTS.

Prior to this release, WCEM Kernel Programming Error Messages contained the Error Number 30.

With this release, WCEM Kernel Programming Error Messages contain the Error Number 19. This change has been made in order to prevent the WCEM Kernel from using any Error Numbers within the 20 to 49 range to which the Studio RTS has assigned certain actions.

WCEM Kernel Suppression Of Multiple CONNECT and DISCONNECT Non-ATLAS Actions

The purpose of the CONNECT Non-ATLAS Action is to inform the CEM Module that the RTS is “connecting” to the CEM Module and thereby allow the CEM Module to perform any first-time processing. Conversely, the purpose of the DISCONNECT Non-ATLAS Action is to inform the CEM Module that the RTS is “disconnecting” from the CEM Module and thereby allow the CEM Module to perform any last-time processing. URTS and WRTS cause these Actions to occur only once per loading/unloading of the CEM Module, regardless of the number of “Bus Channels” defined in the Bus Configuration File. However, the Studio RTS attempts to cause these Actions for each defined “Bus Channel” because the Studio RTS supports multiple DLL’s. There is no problem as long as these multiple DLL’s are different DLL’s, but there is a problem if any of these DLL’s are identical.

Prior to this release, the WCEM Kernel did not suppress multiple attempts to cause these Actions.

With this release, the WCEM Kernel allows the first CONNECT Non-ATLAS Action to occur, but suppresses any additional attempts. Conversely, the WCEM Kernel suppresses all but the last DISCONNECT Non-ATLAS Action.

WCEM Kernel Suppression Of Multiple OPEN and CLOSE Non-ATLAS Actions

The purpose of the OPEN Non-ATLAS Action is to inform the CEM Module that the RTS is opening Instrument Busses and thereby allow the CEM Module to perform any Instrument Bus open processing. Conversely, the purpose of the CLOSE Non-ATLAS Action is to inform the CEM Module that the RTS is closing Instrument Busses and thereby allow the CEM Module to perform any Instrument Bus close processing. URTS and WRTS cause these Actions to occur only once per opening/closing of the Instrument Busses, regardless of the number of “Bus Channels” defined within the Bus Configuration File. However, the Studio RTS attempts to cause these Actions for each defined “Bus Channel” because the Studio RTS supports multiple DLL’s. There is no problem as long as these multiple DLL’s are different DLL’s, but there is a problem if any of these DLL’s are identical.

Prior to this release, the WCEM Kernel did not suppress multiple attempts to cause these Actions.

With this release, the WCEM Kernel allows the first OPEN Non-ATLAS Action to occur, but suppresses any additional attempts. Conversely, the WCEM Kernel suppresses all but the last CLOSE Non-ATLAS Action.

WCEM Kernel Access To Error Message Queue

Error Messages generated by either the CEM Kernel or CEM Device Drivers are not sent immediately to the RTS; they are added to the end of the CEM Kernel Error Message Queue. When the RTS requests data from the CEM Kernel, the CEM Kernel returns (and then deletes) the first Error Message in the queue. The RTS then loops, sending Status Requests to the CEM Kernel until all Error Messages have transmitted to the RTS. URTS and WRTS, upon receiving Error Messages, analyse each Error Message Number and determine what action to take after all Error Messages have been received. But due to Customer requirements, the Studio RTS requires more control over the timing of the receipt of Error Messages.

Prior to this release, the RTS had no control over the timing of the receipt of Error Messages.

With this release, the WRTS (which does not use this capability) and the Studio RTS (which does use this capability) have access to the WCEM Kernel Error Message Queue. This access is provided by five low-level Functions and one high-level Function.

Definition of Low-Level Functions

<code>int cem_err_init(void)</code>	Initializes for accessing the Error Message Queue. Returns zero if the Queue is empty; otherwise, returns number of Error Messages in the Queue.
<code>int cem_err_next(void)</code>	Prepares for accessing the Next Error Message in the Queue. Returns zero if the Queue is empty; otherwise, returns Error Number extracted from the Next Error Message.
<code>int cem_err_get_siz(void)</code>	Returns the size (in Bytes, not including the End-of-String Character) of the Next Error Message.
<code>int cem_err_get_msg(char *pcBuf)</code>	Copies the Next Error Message to the Character Array specified by <code>pcBuf</code> and returns the same value as <code>cem_err_get_siz()</code> .
<code>void cem_err_delete(void)</code>	Removes the Next Error Message from the Queue.

Usage of Low-Level Functions

```
char *pcBuf;  
int nErrMsgs, nErrNum, nMsgSiz;
```

```
nErrMsgs = cem_err_init();
if( nErrMsgs > 0 )
{
    for( ; ; )
    {
        nErrNum = cem_err_next();
        if( nErrNum == 0 ) break;
        nMsgSiz = cem_err_get_siz();
        pcBuf = AllocateMemory( nMsgSiz );
        cem_err_get_msg( pcBuf );
        Process Error Message.
        cem_err_delete();
    }
}
```

Definition of High-Level Function

```
int cem_error_dequeue( char *pcBuf, int nBufSiz )
```

Returns minus one (-1) if error detected.

Returns zero if Error Message Queue is empty.

Returns one (1) if first Error Message copied to Character Array specified by **pcBuf** and **nBufSiz**, and then removed from Error Message Queue.

1.2 Problem Reports.

None.