CEM User
Release Notes
Release 20020530
Version 3.9.24

Purpose

The purpose of this document is to provide information pertaining to the release of the CEM Kernel and user interface. This document is divided into three sections:

Enhancements	Describes changes that enhance the capability of the CEM Kernel.
Problem Reports	Describes changes that correct errors reported in problem reports.
CEM Help	Describes changes made to the user interface that will eventually be added to the CEM On-Line Help.

Enhancements

RTS-to-CEM Information Message - Device Debug Level

With this release, a new RTS-to-CEM Information Message has been added to allow the RTS to send a Device Debug Level to the CEM Module. The syntax of this Message is as follows:

DBG:*ddd=nnn*

where: *ddd* is the Device Name.
nnn is the Level (0 <= *nnn* <= 255).

Examples: **MI DBG:acps=1**
MI MTA:acps=1 MLA:acps=1 MSA:acps=22 CTL:acps=3 DBG:acps=5

RTS-to-CEM Information Message - Device Simulation Level

With this release, a new RTS-to-CEM Information Message has been added to allow the RTS to send a Device Simulation Level to the CEM Module. The syntax of this Message is as follows:

SIM:*ddd=nnn*

where: *ddd* is the Device Name.
nnn is the Level (0 <= *nnn* <= 255).

Examples: **MI SIM:acps=2**
MI MTA:acps=1 MLA:acps=1 MSA:acps=22 CTL:acps=3 SIM:acps=7

CEM Kernel - Device Driver Debug and Simulation Levels

With this release, the CEM Kernel provides support for Device Driver debugging and simulation as follows:

- Upon receipt of an RTS-to-CEM Information Message containing a Device Debug or Simulation Level (as described above), the CEM Kernel extracts the Level and saves it in each member of the **DevDat[]** array of **DEVDAT** structures for which the Device Name in the Message matches the Device Name in a member of **DevDat[]**.

Note The **DEVDAT** Structure Definition has been expanded to support this capability. **DEVDAT** is now four bytes longer: one byte for each Level and two spare bytes. The Levels are initialized to zero.

- CEM Device Drivers may acquire Debug and Simulation Levels by invoking the new CEM Macros *GetCurDevDbgLvl()* and *GetCurDevSimLvl()* as defined below.

WCEM Kernel - Increased Support for Device Bus Controllers

Prior to this release, the WCEM Kernel supported a maximum of 10 Device Bus Controllers whose numbers were required to be between 1 and 10.

Note This did not show up as a problem for Studio RTS Users because the Studio RTS had not yet been updated to send RTS-to-CEM Information Messages containing Device Controller Numbers.

With this release, the WCEM Kernel supports a maximum of 100 Device Bus Controllers whose numbers are required to be between 1 and 9999.

Note Studio RTS Users whose Bus Configuration File(s) specify either more that 10 Bus Channels or Bus Channel Numbers greater that 10 must rebuild their WCEM Modules once the Studio RTS has been modified to send Device Controller Numbers to the WCEM Kernel.

WCEM Kernel - Setup Function

Prior to this release, there was no backwards-compatible way for the Studio RTS to pass initialization-type data to the WCEM Kernel so that the WCEM Kernel would have access to such data during its initialization phase. A problem occurred during debugging and testing whenever the Studio RTS simultaneously loaded multiple WCEM Modules because each WCEM Module would attempt to access its CEM Log File by the File Name "CEM.log", effectively destroying the contents of the File. Although the Studio RTS requires that simultaneously-loaded WCEM Module have unique File Name, subsequent investigation revealed that there was no way for the WCEM Kernel to determine (in a backwards-compatible way) its own File Name.

With this release, the WCEM Kernel provides a Setup Function *CHsetup()* that is available only to the Studio RTS and the WRTS for the purpose of transferring initialization-type data to the WCEM Module. The Function Prototype is:

int CHsetup(char *pcIdentifier, char *pcValue)

where: **pcIdentifier** is a Pointer to an Identifier String.
pcValue is a Pointer to a Value String.

Valid Identifiers are:

Identifier String	Value String
CemModule	CEM Module Name or File Name.
CemLogFile	File Name of CEM Log File.

This Function will return one (1) if no errors detected or two (2) if an error was detected.

The following describes the sequence of events pertaining to the use of this Function:

- The Studio RTS loads a WCEM Module.
- The Studio RTS calls *CHsetup()* specifying the **CemModule** Identifier String and an associated Value String.
- *CHsetup()* validates the **CemModule** Identifier String and saves the associated Value String for later processing.
- The Studio RTS calls *CHsetup()* specifying the **CemLogFile** Identifier String and an associated Value String.
- *CHsetup()* validates the **CemLogFile** Identifier String and saves the associated Value String for later processing.
- The Studio RTS calls *CHconn()* to start the CONNECT Non-ATLAS Action processing.
- *CHconn()* uses the saved Value Strings during its initialization processing prior to calling the User's CONNECT Non-ATLAS Action Function.

Note As of the date of this release, the Studio RTS had not yet been updated to use this Function. However, it is expected that the Studio RTS will be updated in the near future.

New User Stub Debug Function *userStubPrintDeviceInfo()*

Prior to this release, CEM Users were only able to see Device Information by calling the CEM User Stub RESET ATLAS Action Function *userStubRESET()*.

With this release, CEM Users can see Device Information for the Current Device by calling the new User Stub Debug Function *userStubPrintDeviceInfo()*. The Function Prototype is:

void userStubPrintDeviceInfo(char *pcCallerId)

where: **pcCallerId** is either a Pointer to a Caller Identification String (that may be used for identification purposes) or a Null Character Pointer.

The following are examples of usage:

```
userStubPrintDeviceInfo( "Some Characters" );
userStubPrintDeviceInfo( pcNULL );
userStubPrintDeviceInfo( (char *)0 );
```

New User Stub Debug Function *userStubFindDatum()*

Prior to this release, CEM Users were only able to see Datum Information by calling one of the CEM User Stub Functions, usually the CEM User Stub SETUP ATLAS Action Function *userStubSETUP()*. This did not work very well due to certain processing within this Function.

With this release, CEM Users can check to see if a particular Datum exists either for a specific Device/Channel or for the Current Device/Channel by calling the new User Stub Debug Function *userStubFindDatum()*. The Function Prototype is:

DATUM *userStubFindDatum(char *pcCallerId, char *pcDevChan, char *pcModifier, int fQualifier)

where: **pcCallerId** is either a Pointer to a Caller Identification String (that may be used for identification purposes) or a Null Character Pointer.
pcDevChan is a Pointer to a Device/Channel String or a Null Character Pointer (to specify the Current Device/Channel).

pcModifier is a Pointer to a Modifier String.
fQualifier is a Qualifier Token (**K_***xxx*).

If an error is detected or if the specified Datum cannot be found, a Null Datum Pointer (**DNULL**) is returned; otherwise, a Pointer to the specified Datum is returned. In addition, a Line will be printed showing Datum Information.

The following are examples of usage:

DATUM *psDatum;

psDatum = userStubFindDatum("xyz", "acps:CH0", "VOLT", K_SET);

(Search Channel 0 of the A/C Power Supply for a Voltage Modifier with the "Set" Qualifier.)

psDatum = userStubFindDatum(pcNULL, pcNULL, "FREQ", K_FTH);

(Search the Current Device/Channel for a Frequency Modifier with the "Fetch Qualifier".)

New User Stub Debug Function *userStubPrintValue()*

Prior to this release, CEM Users were only able to see Datum Values by calling one of the CEM User Stub Functions, usually the CEM User Stub SETUP ATLAS Action Function *userStubSETUP()*. This did not work very well due to certain processing within this Function.

With this release, CEM Users can see a particular Datum Value by calling the new User Stub Debug Function *userStubPrintValue()*. The Function Prototype is:

void userStubPrintValue(char *pcCallerId, DATUM *psDatum, long iValue)

where: **pcCallerId** is either a Pointer to a Caller Identification String (that may be used for identification purposes) or a Null Character Pointer.
psDatum is a Datum Pointer.
iValue is the Index Number (0 <= iValue) of the Datum Value to be printed.

The following are examples of usage:

DATUM *psDatum;

```
int  iValue, iValueMax;

psDatum = userStubFindDatum( pcNULL, pcNULL, “FREQ”,
K_FTH );
if( psDatum != DNULL )
        userStubPrintValue( “xyz”, psDatum, 0L );

psDatum = userStubFindDatum( pcNULL, “acps:CH0”, “VOLT”,
K_SET );
if( psDatum != DNULL )
{       iValueMax = DatCnt( psDatum );
        for( iValue = 0 ; iValue < iValueMax ; iValue++ )
                userStubPrintValue( pcNULL, psDatum, (long)iValue
);
}
```

Problem Reports

None

CEM Help

The purpose of this Section is to describe changes made to the CEM User Interface that will eventually be added to the CEM On-Line Help.

Update to Current State CEM Macro Group

GetCurDevDbgLvl()
GetCurDevSimLvl()

Name: GetCurDevDbgLvl

Type: Current State

Usage:
```
int    nLevel;
nLevel = GetCurDevDbgLvl();
```

Description: This Function returns the Debug Level for the Current Device/Channel.

Returns: Integer **>= 0** = Debug Level.
< 0 = Error detected.

See Also: CEM Macro *GetCurDevSimLvl().*

Name: GetCurDevSimLvl

Type: Current State

Usage: **int nLevel;**
nLevel = GetCurDevSimLvl();

Description: This Function returns the Simulation Level for the Current Device/Channel.

Returns: Integer **>= 0** = Simulation Level.
< 0 = Error detected.

See Also: CEM Macro *GetCurDevDbgLvl()*.