CEM User
Release Notes
Release 19990806
06 August 1999

N O T I C E TO C E M U S E R S

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.

Overview

1.1 Enhancements
1.1.1 Multiple CEM Controllers.
1.1.1.1 DEVDAT Structure.
1.1.1.2 RTS-to-CEM Information Message - Device Information.
1.1.1.3 RTS-to-CEM Information Message - Controller Information.
1.1.1.4 CEM Driver Access to Bus Configuration Information.
1.1.1.5 Multiple IFC/DCL Sequences.
1.1.1.6 Abnormal Termination of CEM User Functions.
1.1.1.7 CEM User Stub Functions.
1.1.1.8 userStubRESET() (during IFC/DCL Sequence).
1.1.1.9 userStubDEVICECLEAR().

1.2 Problem Reports.
PR 98-128 CEM Errors during IFC/DCL Sequence lost.
PR 99-024 CEM Multiple Errors lost. *(Not applicable to PAWS Studio)*
PR 99-029 Directory Name with ASCII Spaces not handled.
PR 99-053 CEM Error 1030 (Fatal Unload) does not cause IFC/DCL Sequence. *(Not applicable to PAWS Studio)*

1.3 CEM Help
Update to Miscellaneous CEM Macro Group
`AbortUserAction()`
Update to Bus Configuration CEM Macro Group
`GetCurCtlMLA()`
`GetCurCtlMTA()`
`GetCurCtlNam()`
`GetCurDevCtl()`
`GetCurDevMLA`
`GetCurDevMSA`
`GetCurDevMTA()`

1.4 Error Codes and Associated RTS Actions (Table)

Detailed Description

2.1 Enhancements

2.1.1 Multiple CEM Controllers.

Prior to this release, the CEM Model partially supported multiple CEM Controllers.

With this release, multiple CEM Controllers are fully supported. The RTS supports up to a maximum of ten (10) Bus Controllers of different types (including "Channel"). Those "slots" not typed as "Channel" limit the number of multiple CEM Controllers accordingly.

2.1.1.1 DEVDAT Structure.

A new Member (short a_ctl) has been added to the DEVDAT Structure to hold the number of the Controller associated with the Device/Channel Entry.

2.1.1.2 RTS-to-CEM Information Message - Device Information.

Prior to this release, the RTS and CEM Module were enhanced so that the RTS would provide the CEM Kernel with Device Bus Addresses (MTA, MLA and MSA) prior to the first INTERFACE CLEAR / RESET / DEVICE CLEAR Sequence (the "IFC/DCL Sequence").

With this release, the syntax of this Message has been expanded to include the Device's Controller Number (as defined in the Bus Configuration File after the Keyword "BUS"). The new syntax of the RTS-to-CEM Device Information Message is:

MI MTA:*Dev***=***MTA* **MLA:***Dev***=***MLA* **MSA:***Dev***=***MSA* **CTL:***Dev***=***Ctl*

where *Dev* is the Device Name;
MTA, *MLA* and *MSA* are the Device's Talk, Listen and Secondary Bus Addresses, respectively; and
Ctl is the Controller Number.

For example:

```
MI  MTA:DMM=3  MLA:DMM=3  MSA:=12  CTL=1
```

Note The original design of RTS-to-CEM Information Messages specified that Fields (i.e., KeyWord:Label=Data) have no order dependency. Unfortunately, a programming error in the CEM Kernel Information Message processing causes Message processing to terminate if an unrecognizable KeyWord is detected. (Unrecognizable KeyWords may occur when a new RTS runs with an old CEM

Module.) Although this error has been corrected in this release, the RTS should append any new Fields to existing Messages (just to be safe).

2.1.1.3 RTS-to-CEM Information Message - Controller Information.

The syntax of this new Information Message is:

MI CTLNAM:*Ctl***=***Name* **CTLMTA:***Ctl***=***MTA* **CTLMLA:***Ctl***=***MLA*

where *Ctl* is the Controller Number;
Name is the Controller File Name; and
MTA and *MLA* are the Controller's Talk and Listen Bus Addresses, respectively.

For example:

```
MI CTLNAM:1=ChanVXI CTLMTA:1=30 CTLMLA:1=30
```

Note The Controller File Name (as defined in the Bus Configuration File) is used by the RTS and the CEM Kernel for information purposes only. CEM User's may specify whatever is desired - a real Vendor-specified File Name, or, as in the example, a String which indicates the type of Bus the CEM Driver(s) will be accessing.

2.1.1.4 CEM Driver Access to Bus Configuration Information.

Four new CEM Macros have been added to the Bus Configuration CEM Macro Group to allow CEM Drivers to acquire Current Controller Bus Configuration information. Also, three new CEM Macros have been added to the Bus Configuration CEM Macro Group to allow CEM Drivers to acquire Current Device Bus Configuration information in a form identical to that in the Bus Configuration File. Refer to Section 3 in this Document for a full description of these new CEM Macros.

2.1.1.5 Multiple IFC/DCL Sequences.

Prior to this release, the RTS would initiate an IFC/DCL Sequence for each defined CEM Controller.

With this release, the RTS initiates an IFC/DCL Sequence only for the first CEM Controller. (It does not matter which CEM Controller is used because there is no Controller Information associated with the IFC/DCL Sequence.)

2.1.1.6 Abnormal Termination of CEM User Functions.

Prior to this release, CEM Drivers had no easy way of abnormally terminating their processing. If an error was detected at a low level of processing, the Subroutine detecting

the error would first record the error and then had to return control to its Caller, which had to return control to its Caller, etc., until eventually control was returned to the CEM User Function, which then returned control to the CEM Kernel.

With this release, CEM Drivers may abnormally terminate their processing (i.e., return control directly to the CEM Kernel) by calling a new CEM Macro *AbortUserAction()*. Refer to Section 3 in this document for a full description of this CEM Macro.

2.1.1.7 CEM User Stub Functions.

In the previous release, CEM User Stub Functions were implemented. The userStubRESET() and userStubDEVICECLEAR() Functions printed Action Lines containing Controller and Device Bus Addresses in a particular format.

With this release, the format of these Action Lines have been changed.

2.1.1.8 userStubRESET() (during IFC/DCL Sequence).

This Function now prints Device and Controller Bus Addresses in the following format:

C=*CN* **T=***DT* **L=***DL* **S=***DS* **/** *FN* **T=***CT* **L=***CL*

where: *CN* is the Device's Controller Number;
DT, *DL* and *DS* are the Device's Talk, Listen and Secondary Bus Addresses, respectively;
FN is the Device's Controller's File Name; and
CT and *CL* are the Device's Controller's Talk and Listen Bus Addresses, respectively.

Note This information is printed only when the Current Verb is zero. This situation occurs at least twice for an ATLAS Program: the first time is shortly after the ATLAS Program is loaded; and the second time is shortly before the ATLAS Program is unloaded.

2.1.1.9 userStubDEVICECLEAR() Function.

This Function no longer prints Controller Bus Addresses.

2.2 Problem Reports.

PR 98-128 CEM Errors during IFC/DCL Sequence lost.

Prior to this release, errors detected and reported by CEM Drivers during an IFC/DCL Sequence were lost as a result of CEM Module re-initialization.

With this release, CEM Driver calls to CEM Macro *ErrMsg()* result in the error being queued. The error will be reported back to the RTS the next time the RTS requests input from the CEM Module.

Notes Although errors are no longer lost, they are not reported back to the RTS in a timely manner. It is anticipated that this limitation will be corrected in a future release.

When the CEM User INTERFACE CLEAR and DEVICE CLEAR Functions are called, the Current Device is now set to "the Controller".

This Problem Report has been closed.

PR 99-024 CEM Multiple Errors lost. *(Not applicable to PAWS Studio)*

Prior to this release, when a CEM Driver called CEM Macro *ErrMsg()*, the CEM Module recorded only the first error for the Current Device. If additional calls were made to *ErrMsg()* before the first error was reported to the RTS, those errors were ignored.

With this release, the CEM Module queues errors. The CEM Module and RTS processing is as follows:

- A CEM Driver calls *ErrMsg()* to record an error. The CEM Kernel formats and queues a CEM Error Message consisting of the Current Device/Channel Name, the Current Module Name, the Current Line Number, the ATLAS Statement Number, and the CEM Driver Error Message.
- The next time the RTS requests input from the CEM Module (a FETCH, INITIATE or STATUS Request), the CEM Module responds with the first CEM Error Message in the queue. The RTS processes the CEM Error Message.
- The RTS fabricates and sends a STATUS Request to the CEM Module. The CEM Module returns the next queued CEM Error Message. The RTS processes the CEM Error Message. This series of actions continues until the CEM Module returns a Good Status.
- The RTS takes whatever action is necessary as a result of the most serious Error Code received. (Refer to Section 4 in this Document to see Error Codes and their associated RTS actions.)

Notes CEM Error Messages are no longer stored on a Device/Channel basis; they are queued in the CEM Error Message Buffer. This Buffer is currently 1,000 Bytes, which should hold at least ten (10) CEM Error Messages.

If the RTS receives twenty (20) consecutive Error Responses when in the loop described above, no further STATUS Requestes are sent to the CEM Module and an IFC/DCL Sequence is initiated immediately. (This is done for safety purposes. Given the size of the CEM Error Message Buffer, the CEM Module should never generate this many consecutive Error Responses.)

If the RTS receives an Error Code that requests the RTS to perform an IFC/DCL Sequence, no further STATUS Requests are sent to the CEM Module and an IFC/DCL Sequence is initiated immediately.

When the CEM Module receives an IFC Request, the CEM Module discards all queued CEM Error Messages.

When the CEM Module receives a fabricated STATUS Request, the CEM User STATUS Function is NOT called.

This Problem Report has been closed.

PR 99-029 Directory Name with ASCII Spaces not handled.

Prior to this release, the 32-Bit Windows *OSmain()* algorithm for extracting an Argument from an Argument File was:

- Skip leading Whitespace Characters (if any).
- Extract all Characters up to (but not including) the next Whitespace Character.

With this release, the 32-Bit Windows *OSmain()* algorithm for extracting an Argument from an Argument File is:

- Skip leading Whitespace Characters (if any).
- Examine the first Character of the Argument. If this Character is an ASCII Double Quote ("), define the Argument Delimiter to be this Character, and skip it. If not, define the Argument Delimiter to be an ASCII Space.
- Extract all Characters up to (but not including) the Argument Delimiter (or an ASCII Carriage Return or Line Feed).
- Skip the Argument Delimiter (or the ASCII Carriage Return or Line Feed).

This Problem Report has been closed.

PR 99-053 CEM Error 1030 (Fatal Unload) does not cause IFC/DCL Sequence.

(Not applicable to PAWS Studio)

Prior to this release, RTS did not perform an IFC/DCL Sequence when a Fatal Unload Error (1030) was received from the CEM Module.

With this release, RTS performs an IFC/DCL Sequence when a Fatal Unload Error is received from the CEM Module.

This Problem Report has been closed.

2.3 CEM Help

Update to Miscellaneous CEM Macro Group

AbortUserAction()

Update to Bus Configuration CEM Macro Group

GetCurCtlMLA()
GetCurCtlMTA()
GetCurCtlNam()
GetCurDevCtl()
GetCurDevMLA
GetCurDevMSA
GetCurDevMTA()

Name: AbortUserAction

Type: Miscellaneous

Usage:
```
int  nStatus;
AbortUserAction( nStatus );
```

Description: This Macro causes control to be returned immediately to the CEM Kernel at the same location where control would have been returned if the most-recently-called CEM User Function had been allowed to execute a return. This allows CEM Drivers to abnormally terminate their processing at a low level without having to return control to each preceeding Subroutine.

The specified Status (in **`nStatus`**) is not used by the CEM Kernel but is printed to the CEM Log File "CEM.log" if Environment Variable **`TYXCEMFDEBUG`** is set to a number greater then or equal to 1.

See Also:

Name: GetCurCtlMLA

Type: Bus Configuration

Usage:
```
int  nCtlMLA;
nCtlMLA = GetCurCtlMLA();
```

Description: This Macro returns the Listen Bus Address (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File) of the Current Controller. (The Current Controller is defined to be the Controller associated with the Current Device.)

If the Bus Address is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurCtlMTA()*, *GetCurCtlNam()* and *CntLsnAdd()*.

Name: GetCurCtlMTA

Type: Bus Configuration

Usage:
```
int  nCtlMTA;
nCtlMTA = GetCurCtlMTA();
```

Description: This Macro returns the Talk Bus Address (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File) of the Current Controller. (The Current Controller is defined to be the Controller associated with the Current Device.)

If the Bus Address is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurCtlMLA()*, *GetCurCtlNam()* and *CntTlkAdd()*.

Name: GetCurCtlNam

Type: Bus Configuration

Usage:
```
char  *pcCtlNam;
pcCtlNam = GetCurCtlNam();
```

Description: This Macro returns a Character Pointer to the File Name (exactly as specified in the Bus Configuration File) of the Current Controller. (The Current Controller is defined to be the Controller associated with the Current Device.)

If the File Name is not available or if an error is detected, a Character Pointer to the String "Channel" is returned.

See Also: CEM Macros *GetCurCtlMLA()* and *GetCurCtlMTA()*.

Name: GetCurDevCtl

Type: Bus Configuration

Usage:
```
int  nDevCtl;
nDevCtl = GetCurDevCtl();
```

Description: This Macro returns the Controller Number (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File following the Keyword "BUS") of the Controller associated with the Current Device.

If the Controller Number is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurDevMLA()*, *CurDevMSA()* and *GetCurDevMTA()*.

Name: GetCurDevMLA

Type: Bus Configuration

Usage:
```
int  nDevMLA;
nDevMLA = GetCurDevMLA();
```

Description: This Macro returns the Listen Bus Address (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File) of the Current Device.

If the Bus Address is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurDevCtl()*, *CurDevMSA()*, *GetCurDevMTA()* and *DevLsnAdd()*.

Name: GetCurDevMSA

Type: Bus Configuration

Usage:
```
int  nDevMSA;
nDevMSA = GetCurDevMSA();
```

Description: This Macro returns the Secondary Bus Address (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File) of the Current Device.

If the Bus Address is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurDevCtl()*, *CurDevMLA()*, *GetCurDevMTA()* and *DevSecAdd()*.

Name: GetCurDevMTA

Type: Bus Configuration

Usage:
```
int  nDevMTA;
nDevMTA = GetCurDevMTA();
```

Description: This Macro returns the Talk Bus Address (e.g., 1, 2, 3, etc., exactly as specified in the Bus Configuration File) of the Current Device.

If the Bus Address is not available or if an error is detected, a negative number (currently minus one (-1)) is returned.

See Also: CEM Macros *GetCurDevCtl()*, *CurDevMLA()*, *GetCurDevMSA()* and *DevTlkAdd()*.

2.4 Error Codes and Associated RTS Actions

Error Code	Warning Message	Unload ATLAS Program	IFC/DCL Sequence	Halt	Measured Value	MAX-TIME
0					Maximum	
1					Minimum	
2	Yes				Zero	
3	Yes				Maximum	TRUE
4	Yes				Minimum	TRUE
5	Yes				Zero	TRUE
6	Yes					TRUE
7	Yes			Yes		
8	Yes			Yes		TRUE
9 to 27	Yes		Yes			
1030	Yes	Yes	Yes			
All Others	Yes		Yes	Fail		