

ApplCore Version 3

Commands Reference

Contents

Contents	2
1 Audio Commands.....	3
2 Bitmap Command	6
3 Control Commands.....	6
4 Device Commands.....	10
5 Executive Commands.....	14
6 eXternal Commands	27
7 File Commands	36
8 GPI Commands	36
9 Html Commands.....	39
10 Info Command	41
11 Jump Commands.....	46
12 Key Commands.....	50
13 List Commands	60
14 Memory Commands	72
15 Number Commands	76
16 Panel Commands	83
17 Query Commands	85
18 Router Commands.....	88
19 String Commands	93
20 Text Commands.....	99
21 UMD Commands	107
22 Video Commands	108
23 Workstation Commands.....	112
V3 Appendix I – Keyboard Tables.....	114
V3 Appendix II – Keyboard Tables.....	133
24 V3 Version History	153

1 Audio Commands

1.1 Audio Beep

From version 3.02.00

Syntax : **AB**

Description:-

Produces an audible beep from the default audio device. This is by default the PC (internal) speaker. If you have a sound card fitted and enabled then the beep will play the WAV file assigned in the 'sound' section of the Windows Control Panel application.

It should be noted that the PC speaker is driven directly by the CPU and as such causes a dramatic slow down in panel performance. Panel application processing stops for the duration of the 'beep'. Use **AB** sparingly.

With a sound card fitted the processing of the sound is handled by on board hardware and as causes very little deterioration in system performance.

Technical Details: The code in ApplCore is just a call to MessageBeep(MB_OK). The MessageBeep() function plays a waveform sound. The waveform sound for each sound type is identified by an entry in the [sounds] section of the registry and may be selected by settings in the systems control panel (in Windows XP: Sounds and Audio Devices -> Default Beep).

1.2 Audio Key

From version 3.02.00

Syntax : **AK <UP> <Filename.WAV\$> or AK <DOWN> <Filename.WAV\$>**

Description:-

If you have a sound card fitted and enabled then this command will play a file when any key is pressed or released. Can be used to simulate key clicks when set to play an appropriate WAV file. <Filename.WAV\$> can be a full filepath and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons.

From version 3.08.05

If <UP> or <DOWN> will NULL filename specification the operation is cancelled.

1.3 Audio MCI

From version 3.02.00

Syntax : **AM <Var\$>**

Description:-

Sends string <Var\$> directly to the MCI driver. A full list of available commands is beyond the scope of this manual. This command is intended for future development of existing ApplCore systems. The MCI 'alias' name for the audio control is 'applcore'. Examples of MCI commands include:

- AK 'pause applcore' to pause the playback
- AK 'resume applcore' to resume the playback

1.4 Audio Play

From version 3.02.00

Syntax : **AP <Filename.WAV\$> [<Offset%>] [<OnErrorOffset%>]**

Description:-

Plays a WAV file **<Filename.WAV\$>** via the external sound system if available. When the sound system has finished playing the file stringtable offset **<Offset%>** is executed. If the sound system is already playing or recording then that previous activity is aborted.

<Filename.WAV\$> can be a full filepath and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons. Optional parameter **<OnErroroffset%>** stringtable offset executes should an error occur (such as file not found).

From version 3.08.06

Syntax : **AP <Filename.WAV\$> [<Offset%>] [<OnErrorOffset%>] [<ImmediateFlag%>]**

Issuing the **AP** command with the **<ImmediateFlag%>** set to 1 while ApplCore is currently playing a file will now cause a restart of audio play. Previous versions the behaviour was non-deterministic, generally stopped the play operation; but often caused audio noise bursts rather than restarting the operation.

1.5 Audio Record

From version 3.02.00

Syntax : **AR <Filename.WAV\$>**

Description:-

Records a WAV file **<Filename.WAV\$>** via the external sound system if available. If the sound system is already playing or recording then that previous activity is aborted. If **<Filename.WAV\$>** already exists then it will be overwritten. **<Filename.WAV\$>** can be a full filepath and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons.

1.6 Audio Stop

From version 3.02.00

Syntax : **AS**

Description:- Stops the external sound system playing or recording.

2 Bitmap Command

2.1 Bitmap Put

From version 3.02.00

Syntax : **BP** <Panel No%> <Control ID%> <xPos%> <yPos%>

Description:-

Draws a bitmap on a control. The control must have the style 'owner draw'. The text block of the control stores the bitmap name. By putting a bitmap name in the text block using resource workshop means that the bitmap will appear immediately when the panel is displayed and does not require an entry in the startup string table. <xPos%> and <yPos%> determine the position of the bitmap within the control. This will normally be 0.

3 Control Commands

3.1 Control Adjust

From version 3.02.00

Syntax : **CA** <Panel No%>> <ScrollBarId%>> <Var%>

Description:-

Sets the position of a scroll bar <ScrollBarId%> on <Panel No%> to <Var%>. <Var%> should be an integer between 0 and 255.

3.2 Control Check

From version 3.02.00

Syntax : **CC** <Panel No%> <Button Id%>

Description:-

Puts a check mark next to a check control or radio button **<Button Id%>** on panel **<Panel No%>**.

3.3 Control Disable

From version 3.05.18

Syntax : **CD <Panel No%> <Button Id%> (<Max Button Id%>)**

Description:-

Disables control **<Button ID%>** on panel **<Panel No%>** by "greying out" the control and therefore making it clear which controls have been disabled. Works also with text blocks/menus etc. Optional (**<Max Button Id%>**) parameter extends the function to a range of controls starting with **<Button ID%>** through **<Max Button Id%>**.

3.4 Control Enable

From version 3.05.18

Syntax : **CE <Panel No%> <Button Id%> (<Max Button Id%>)**

Description:-

Enables control **<Button Id%>** on panel **<Panel No%>** by un-"greying out" the control and therefore making it visible which controls are active. Works also with text blocks/menu. Optional (**<Max Button Id%>**) parameter extends the function to a range of controls starting with **<Button ID%>** through **<Max Button Id%>**.

3.5 Control Focus

From version 3.02.00

Syntax : **CF <Panel No%> <Button Id%>**

Description:-

Sets the input focus to be control **<Button Id%>** on panel **<Panel No%>** Primarily intended for Edit controls where any subsequent text entered from the keyboard will appear in the specified control.

3.6 Control Hide

From version 3.08.38

Syntax : **CH** <Panel No%> <Button Id%> (<Max Button Id%>)

Description:-

Hides, but does not destroy, control <Button ID%> on panel <Panel No%> Optional (<Max Button Id%>) parameter extends the function to a range of controls starting with <Button ID%> through <Max Button Id%>.

Show again using CV command

3.7 Control IdChange

From version 3.02.00

Syntax : **CI** <PanelId%> <OldCtlId%> <NewCtlId%>

Description:-

Changes the control ID of a control. This is an extremely useful command but should be used with great care as it is possible to lose track of the ID number control.

3.8 Control Move

From version 3.02.00

Syntax : **CM** <PanelId%> <CtlId%> <NewXpos%> <NewYpos%>

Description:-

Moves control <CtlId%> from its current position to <NewXpos%> <NewYpos%> on <PanelId%> .

3.9 Control Name

From version 3.02.00

Syntax : **CN** <Driver%> <Source/Dest Flag> <Min%> <Max%> <PanelId%>
<FirstCtlId%>

Description:-

Extracts database names from router **<Driver%>** for range **<Min%>** through **<Max%>** and places the names into buttons on panel **<PanelId%>** beginning at control number **<FirstCtlId%>**

The **<Source/Dest Flag>** can be 0 to 9 inclusive, if 0 then source names are extracted, if 1 destination names are extracted, if 2 database 2 names are extracted etc.

3.10 Control Return

From version 3.02.00

Syntax : **CR <Line%>**

Description:-

Specifies a stringtable line to be executed when the keyboard **<Return>** key is pressed. The default value is zero. i.e. no line will be executed. Particularly useful when used with the BBC Edit control in BBC_CC05.DLL when entering text from the keyboard.

3.11 Control Size

From version 3.02.00

Syntax : **CS <PanelID%><CtlID%><New X size%><New Y size%>**

Description:-

Resizes the Control **<CtlID%>** on the Panel **<PanelID%>** to have a new horizontal size of **<New X size%>** and a new vertical size of **<New Y size%>**.

3.12 Control Text

From version 3.02.00

Syntax : **CT <String\$> <PanelId%> <Min%> <Max%>**

Description:-

Places **<String\$>** into range of buttons between **<Min%>** and **<Max%>** on panel **<PanelId%>**

This command can be used to preload a range of buttons with the same text, particularly useful to set a range of buttons to a particular colour scheme using the '/xxxxxx' escape code scheme as defined in the Applcore manual.

3.13 Control Uncheck

From version 3.02.00

Syntax : **CU <Panel No%> <Button ID%>**

Description:-

Remove a check mark from a check control or radio button

3.14 Control Visible (Show)

From version 3.08.38

Syntax : **CV <Panel No%> <Button Id%> (<Max Button Id%>)**

Description:-

Shows control **<Button ID%>** on panel **<Panel No%>** Optional (**<Max Button Id%>**) parameter extends the function to a range of controls starting with **<Button ID%>** through **<Max Button Id%>**.

Hide using CH command

4 Device Commands

4.1 Device Enable

From version 3.02.00

Syntax : **DE <0/1> <Workstation%> <Device%> <ReturnOffset%>**

Description:-

Enables **<1>** or disables **<0>** a device **<Device%>** on a remote workstation **<Workstation%>** and signals via **<ReturnOffset%>** success or failure.

4.2 Device GetData

From version 3.02.00

Syntax : **DG** <Device%> <Index%> <Var\$>

Description:-

Retrieves integer or string data from device <Device%>, destination or slot <Index%> and puts the result in <Var\$>. If the cache is not turned on or is invalid <Var\$> returns '- -'.

4.3 Device Index

From version 3.02.00

Syntax : **DI** <Device%> <Min%> <Max%> <Index\$> <Var%>

Description:-

This command searches for strings or integers in the CSI cache for device <Device%> between destinations or slots <Min%> and <Max%> and puts the result in <Var%>. If the string or integer cannot be found <Var%> returns '0'. If the cache is not turned on or is invalid <Var%> returns '-1'.

4.4 Device List

From version 3.02.00

Syntax : **DL** <Panel Id%> <ListBoxId%>

Description:-

Gets the status of all known drivers from their associated dev_xxx.ini files on the local PC and delivers it in the form of a string to the list box <Panel Id%> <ListBoxId%>. The String contains six pieces of information in the format '<Driver ID> <Driver Version> <Driver Type> <Driver Sources> <Driver Destinations> <Driver DataBases>'. The Driver ID is the device number of the driver, the Driver Version is 1 for version 1 drivers and 2 for version 2. The Driver Type is used with version 1 drivers only, 1 signifies a GRD, 2 a GPI and 3 an InfoDriver, for version 2 drivers this is always 0. Driver Source and Driver Destinations are the max number of sources and destinations.

The Driver Databases is a comma delimited list of the databases associated with this driver.

For example a returned string of 100 2 0 128 128 0,1,2,3 represents Driver 100 that is a version 2 driver with 128 sources and destination and has databases 0,1, 2 and 3 associated with it.

4.5 Device Mode

From version 3.02.00

Syntax : **DM <Device%> <Workstation%> <TxRxMode%> <Error%>**

Description:-

Each time a status message is sent from a driver on the network your panel will automatically execute string table line 32400. As this is an automated process and you don't need to register to receive these messages (just like the Database Change Mechanism).

If you have issued a Device Mode command with the above syntax on line 32400 then when a status message is received **<Device%>** and **<Workststaion%>** will contain the Device number and Workstation number of the driver issuing the message. **<TxRxMode%>** will be 1 for a device in RX only mode and 2 if it is in TxRX Mode. **<Error%>** should return a 0 is all is OK, anything else would indicate that some form of error has occurred. "Comms failure in a driver may be signalled using a single NB message (RxOnly_Broken), the status message received will contain a **<TxRxMode%>** value of 1 and an **<Error%>** value of 4.

4.6 Device Profile

From version 3.02.00

Syntax : **DP <Driver%> <Size0%> <Size1%> <Reassert%>**

Description:-

Instructs all CSI's in the system to create or modify a '**DEV_xxx.INI**' file where 'xxx' is **<Driver%>**. The database section is dimensioned to be **<Size0%>** by **<Size1%>** and CSI then loads the default contents into its database cache. If **<Reassert%>** is set to 1 any existing entries will be overwritten, whereas if **<Reassert%>** is set to 0 only missing entries will be filled in.

Please Note if **<Reassert>** is set to 0 then Database_0 will not be updated as it already exists.

4.7 Device Status

From version 3.02.00

Syntax : **DS** <Driver%> <VersionVar%> <TypeVar%> <SourcesVar%>
<DestsVar%>

Description:-

Gets the current status for <Driver%> and puts the results in <VersionVar%> <TypeVar%> <SourcesVar%> <DestsVar%>. <VersionVar%> is the driver version, either 1 or 2. If <VersionVar%> is 1 then <TypeVar%> is 1 for a GRD, 2 for a GPI or 3 for an InfoDriver. If <VersionVar%> is 2 then <TypeVar%> will always be zero.

For a GRD <SourcesVar%> and <DestsVar%> will return the number of sources and destinations. For GPIs <SourcesVar%> will be 2 and <DestsVar%> will be the number of inputs or outputs. For InfoDrivers <SourcesVar%> will be 0 and <DestsVar%> is the number of slots.

The information comes from CSIs local copy of the INI files.

5 Executive Commands

5.1 Exec Application

From version 3.02.00

Syntax : **EA <Application\$>**

Description:-

Launches another application with command line parameters if supplied. **<Application\$>** can be a full file path and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons.

e.g.. EA C:/MRCTA/PANELMAN.EXE 44

From Version 3.03.05

Syntax : **EA <Application\$> <ErrorBoxFlag%> <TranslateFlag%>**

The **<ErrorBoxFlag%>** parameter affects whether a message box is displayed in the event that the requested application cannot be launched. If set to '0' then the message box will be not displayed and the failure to launch the application ignored.

If **<TranslateFlag%>** is set to '1' then any occurrences of '/' in the file path will not be translated to '\\'.

From Version 3.06.08

The **EA** command now supports both the original **8 + 3** filename form as well as "**long file names**" for the **Application\$** parameter. The **Application\$** parameter is used to form the *process name* of the ApplCore panel (visible in the taskbar and process viewer). The *process name* is constructed using the following form:

<ApplCore panel process name>.<workstation number>

The *process name* is also used to signal between this ApplCore panel and the application to be launched – setting the command line parameters in **S\$** and bringing the panel to the foreground (in the case where the application to be launched is already running).

The **Application\$** parameter panel name is padded to a length of eight using underscore characters. The name length does not include the three in the extension of the workstation number.

See also: **CSI.INI** entry: **Workstation=###**

For example if the ApplCore panel name is **ABCD.EXE** and the **CSI.INI** entry **Workstation=119**, then the process name is **ABCD____.119** (This is what is shown in the taskbar and process manager).

Long file names support means, an ApplCore panel which exceed the previously enforced panel name length limit of eight, such as example name:

ThisApplCorePanelHasALongFileName.EXE And produces a process name of:
ThisApplCorePanelHasALongFileName.119

Please note: Versions 3.06.01 through 3.06.07 are not compatible with 3.06.08 forward due to problems in those older versions.

5.2 Exec Backpanel

From version 3.02.00

Syntax : **EB** <Var%> <XSize%> <YSize%> <Bitmap\$>

Description:-

Modifies the state of the backpanel. The backpanel is a matte background that hides any other Windows applications if your panel application does not have a panel which is full screen. The three states are selected by <Var%> as follows :

<0> Backpanel is off

<1> Backpanel is grey

<2> Backpanel is black

<3> Backpanel is a bitmap within your RC file called <Bitmap\$>

For all values of <Var%> other than 0 the <XSize%> and <YSize%> values specify the size of the backpanel in pixels.

5.3 Exec Caption

From Version 3.05.07

Syntax : **EC** <Mode%>

Description:-

If <Mode%> is '0' the main window which forms the backpanel of the panel application has it's origin at screen coordinates 0,0 and is the full width and depth of the screen. The window will have a caption bar at the top showing the name of the application. The first part of the name will come from the filename and the second part after the full stop will be the workstation number. e.g. ROUTER.032

If <Mode%> is '1' the origin becomes 0, -20 and the total window size is the full width and depth of the screen. This is to allow the use of a child window with many overlaid sub panels by pushing the caption bar outside the viewport.

If <Mode%> is '2' the caption bar becomes unlocked and you can move the backpanel with the mouse. Child windows will follow.

If <Mode%> is '3' the caption bar is locked and cannot be moved with the mouse.

If <Mode%> is '4' turns off the caption bar and border (different from EC 1 which just moves the caption bar off screen).

If <Mode%> is '5' turns on the caption bar and border.

5.4 Exec Debug

From version 3.05.09

Syntax : **ED <Mode%>**

Description:-

If **<Mode%>** is '1' the debug box will be displayed and if **<Mode%>** is '0' it will be removed.

If **<Mode%>** is '2' turns off the Debug window.

If **<Mode%>** is '3' turns on the Debug window.

If **<Mode%>** is '4' turns off the output to the Windows Debug Stream.

If **<Mode%>** is '5' turns on the output to the Windows Debug Stream.

If **<Mode%>** is '6' turns off the Parser Trace output.

If **<Mode%>** is '7' turns on the Parser Trace output.

If **<Mode%>** is '8' turns off the Debug output.

If **<Mode%>** is '9' turns on the Debug output.

5.5 Exec Error

From version 3.02.00

Syntax : **EE <On/Off%> <Var\$> <DisplayTime%> <OnLine%> <OffLine%>**

Description:-

Normally if there is an error in the string able code a message box is displayed detailing the error and the application then waits for the user to clear the message box. The Exec Error command can be used to put the error string into a variable **<Var\$>** and execute a stringtable line **<OnLine%>**.

5.6 Exec File

From version 3.02.00

Syntax : **EF <Section\$ or Section %> <Entry\$ or Entry %> <Var\$ or Var%> [**<Read/Write%>**] [**<Filename\$>**]**

Description:-

Searches for **<Entry\$>** in the **<Section\$>** of the text based file **<Filename\$>** and assigns either the string found there to **<Var\$>** or the integer to **<Var%>**. **<Filename\$>** can contain an explicit file path.

Parameters **<Entry\$>** and **<Section\$>** may be either a string or numeric value. The **<Read/Write%>** parameter allows entries to be written to or read from the file, the **<Read/Write%>** parameter is 0 (default) for a read and 1 for a write.

If no explicit path is given, then the file path will default to "ConfigPath" as specified in file: bncs_config.ini <ConfigPath>\<Filename\$>

From version 3.08.08

EF command will resolve the path to indicated file using the following:

Source file name and path	File Path Searched
\\ws\c\file.ini	\\ws\c\file.ini
c:\path\file.ini	c:\path\file.ini
file.ini	<ConfigPath>\file.ini
sub\file.ini	< ConfigPath >\sub\file.ini
.\file.ini	<CWD>\file.ini
.\sub\file.ini	<CWD>\sub\file.ini
..\sub\file.ini	<CWD>\..\sub\file.ini

Note: "CWD" is Current Working Directory.

Filename and path syntax:

If an explicit path is used then colons (:) must be replaced by semi-colons(;) and backslashes (\) by forward slashes (/).

Example:

EF 'Router' 'Source 1' A% 0 'C:/BNCS/MyFile.ini' will read the **Source 1=** entry in the **[Router]** section of the **MyFile.ini** file located in the **C:\BNCS** folder and assign it to **A%**.

If <Filename\$> is not specified the default filename "ApplCore.INI" is used.

NOTE: It is important to note the **EF** command accepts non-quoted strings and numerics for the <Section\$> and <Entry\$> parameters. This is different from other ApplCore commands with string parameters, however quoted strings are recommended. For example, EF accepts these parameter sets:

- EF FooBarSection FooBarEntry A\$ 0 'foobar.ini'
- EF t% FooBarEntry A\$ 0 'foobar.ini'
- EF 'FooBarSection' B% A\$ 0 foobar.ini
- EF A\$ B\$ A\$ 0 foobar.ini

5.7 Exec Get

From version 3.02.00

Syntax : **EG** <Var%> <Offset%>

Description:-

Places the stringtable line number on which the current command is executing into <Var%>. The value of <Offset%> is then added to <Var%>. Negative values of offset are accepted.

5.8 Exec Hide

From version 3.02.00

Syntax : **EH** <PanelId%> <Switch%>

Description:-

If <Switch%> is 0 the panel application is pushed to the bottom of all other panel applications. If <Switch%> is 1 the panel is brought to the top.

If <PanelId%> is 0 and <Switch%> is 0 then the main panel (the whole application) is moved to the bottom.

If <PanelId%> is 0 <Switch%> is 1 then all panels are moved to the top in numeric order of the <PanelId%>, i.e. the whole application is moved to the top.

5.9 Exec Ignore

From version 3.02.00

Syntax : **EI**

Description:-

This command and any following text is ignored on the stringtable line, until a : is encountered when another command may be added.

Used to add comments into stringtable lines. Note however that EI is seen as a valid command when executing sequential lines, for example during STARTUP or during a SubRoutine.

5.10 Exec Kill

From version 3.02.00

Syntax : **EK** <Min resources%>

Description:-

If system resources are less than <Min resources%> this command searches for the least recently used ApplCore application and instructs it to terminate.

It is intended for use in menu bar applications which launch new applications. By terminating the least used application first it ensures that there are sufficient Windows resources to allow the new application to run.

5.11 Exec Log

From version 3.02.00

Syntax : **EL** <Filename\$> <Var\$>

Description:-

Adds the string <Var\$> to the end of the file <Filename\$>. If <Filename\$> does not exist then it is created. <Filename\$> can be a full file path and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons.

5.12 Exec Message

From version 3.02.00

Syntax : **EM** <Title\$> <Message\$> [<Mode%>] [<TimerParameter%>]

Description:-

Displays a message box with the heading <Title\$> and displaying the message <Message\$>. Pressing the 'OK' button will remove the message box.

If <Mode%> parameter is missing or 0 (default) a ApplCore style message window pops up and command execution continues.

If <Mode%> parameter is 1, a Windows message box window pops up and all panel activity, all command execution is halted whilst the message box is displayed. When the "OK" button is pressed, then program execution continues.

From version 3.08.01

If optional parameter <TimerParameter%> is present a timer is set to **TimerParameter%** milliseconds (requires <Mode%> parameter be specified as 0). If the timer expires, the message window is removed and program execution continues. When the "OK" button is pressed, then program execution continues.

CRITICALLY IMPORTANT NOTE:

All panel activity is halted whilst the Windows style message box of type <Mode%> = 1 is displayed.

This includes handling of events such as timers, revertives, and clock updates stop (i.e. if using time 1, T\$ - clock updates).

Therefore, use of the EM command in UNATTENDED critical operation panels is strongly discouraged.

5.13 Exec Name Day

From version 3.02.00

Syntax : **EN** <Day%> <Mon%> <Year%> <Var%><Var\$>

Description:-

Returns the day of the week as number in <Var%> and as a string in <Var\$>. Sunday is the first day of the week. The string names returned are "**Sun**", "**Mon**", "**Tue**", "**Wed**", "**Thu**", "**Fri**", "**Sat**".

5.14 Exec Override

From version 3.00.00

Syntax : **EO** <EOMin %> <EOMax%> <EOLine%> <EOIdVar%>

Description:-

The Exec Override command allows the setting of overrides for execution Id's between <EOMin%> and <EOMax%>. That is, setting of overrides from between <EOMin%> and <EOMax%> to go to line <EOLine%>.

The alternate execution values are set if <EOMin%> and <EOMax%> are between 0 and 32767, and <EOMin%> is less than <EOMax%> (then the supplied values are set for execute override).

Default values:

EOMin = 32767

EOMax = 0

EOLine = 0

EOIdVar = A% (The return variable is assigned to A% by default)

Note: This facility was added to ApplCore from the first versions of V3 (3.00.00). The V3 command "EO" was reassigned to Exec Override (EO) and the original command facility Exec Debug Output String was deleted. The original V2 Exec Debug output string (EO) command function has been reassigned to "EZ" (Exec Z Debug Output String) command in V3 ApplCore starting with version 3.07.02.

5.15 Exec Panel

From version 3.02.00

Syntax : **EP** <0/1>

Description:-

If '0' the application is minimised. If '1' it is maximised.

5.16 Exec Quit

From version 3.02.00

Syntax : **EQ** <Var%> <LineNumberValue%>

Description:-

Causes the panel application to execute the **CLOSEDOWN** stringtable line (31000) and quit if Windows resources are less than <Var%>. This can be used to close down an application in the **SLEEP** stringtable line (30500) to make way for another one to startup without being short of resources.

EQ 100 will always cause the application to terminate since system resource will always be less than 100%. For values of <Var%> less than or equal to 100, the parameter <LineNumberValue%> is ignored.

Example stringtable lines:

```
140, 'EQ 100      :EI normal closedown :JA 141'  
141, 'EI EO TEST :EI next line      '
```

```
31000, 'EI CLOSEDOWN MSG :JS 31100 :EI returned from JS '  
31001, 'EI line 31001 returned from call of 31100'  
31002, 'EI line 31002 '  
31100, 'EI return :JR'
```

The button executes stringtable line 140, first the EQ command then the EI then the JA to line 141. Execution continues with line 141. The posted "close down" message to the ApplCore execution queue is realized and executed. This starts ApplCore execution at line 31000, and continues until the last ApplCore command (on 31000 and subsequent lines).

ApplCore execution ceases with the end of line 31002.

ApplCore then executes "close down" within ApplCore and exits from ApplCore "main()".

From version 3.07.11

Adds the parameter <LineNumberValue%>.

EQ has been enhanced to optionally allow for an immediate closedown. Original EQ behaviour is to post a **CLOSEDOWN** message to the ApplCore execution queue, this means that ApplCore will continue executing until the **CLOSEDOWN** message is realized in the ApplCore execution queue.

If <Var%> is a value equal to 1000, then the **CLOSEDOWN** action is immediate. Current stringtable line execution does not continue. The **CLOSEDOWN** stringtable line is not executed, if parameter <LineNumberValue%> is specified then the stringtable line is executed.

In the following example, ApplCore execution ceases at the "EQ 1000" ApplCore instruction, **CLOSEDOWN** begins immediately. The "EI" and "JA" commands on stringtable line 140 are not executed. Line 31000 is not executed.

Example stringtable lines:

```
140, 'EQ 1000 :EI special closedown :JA 141'
141, 'EI TEST :EI next line
```

In the following example, ApplCore execution ceases at the "EQ 1000" ApplCore instruction, **CLOSEDOWN** begins immediately. The "EI" command on stringtable line 140 is not executed. Line 33333 is executed.

Example stringtable lines:

```
140, 'EQ 1000 33333 :EI special closedown'
141, 'EI TEST :EI next line
33333, 'EI Special close down vector'
```

5.17 Exec Reboot

From version 3.02.04

Syntax : **ER 'REBOOT'**

Description:-

Causes the panel to restart Windows. The word '**REBOOT**' after the ER command is to make sure that the panel programmer meant to use this command rather than spend many hours chasing a syntax error with machine rebooting every time the panel is run. Additional parameters of '**LOGOFF**', '**LOGOUT**' and '**KILLCSI**'.

5.18 Exec Status

From version 3.02.04

Syntax : **ES <DeviceID%> <DeviceWorkstation%> <Parameter = Value\$> <RevertiveLine%>**

Description:-

Generates a network message that is sent directly to a driver, to control its function and obtain status information.

The **<DeviceID%>** is the device to send the message to, and **<DeviceWorkstation%>** is the machine hosting the device.

The **<Parameter = Value\$>** is the property of the device being requested or set – see options below.

The **<RevertiveLine%>** is the stringtable line to be executed when the device responds. When the revertive line is called, S% will contain the device Id and S\$ will contain the status information.

Note: No response on **<RevertiveLine%>** will be received if driver is not running on target workstation.

The Parameter Value \$ options are:

'TxRx=?' - enquire current device status

'TxRx=1' - causes the device to go RxOnly and return 'TXRX=1'

'TxRx=2' - causes the device to go TxRx and return 'TXRX=2'. Note: another network instance of the device in TxRx may issue an objection forcing the changed device back to RxOnly and any further client operations based on the returned status message could then be invalid.

'TxRx=3' - causes the device to issue a Network Objection to force any other instances RxOnly before it becomes the undisputed TxRx device, returns 'TXRX=2'.

The driver will respond depending upon whether it supports the parameter or not. Device registration is not required to receive the revertive status message.

Example: ES 123 1 'TxRx=1' 3001 will switch Device 123 on Workstation 1 to RxOnly and the return status string containing TXRX=1 will be received as a revertive in S\$ on stringtable line 3001.

Note: the Parameter \$ is not case sensitive, mixed case examples used here to identify the client command message from the Uppercase status message returned by the device when seen in Caplog.

See also the documentation on Resilience.

5.19 Exec Timer

From version 3.02.00

Syntax : **ET** <TimerNumber%> <TimeDelay%> [<OffSet1%>] [<Offset2%>]

Description:-

Controls the enabling of event timers. <TimerNumber%> may be any one of 16 timers (1 - 16).

If the time delay parameter <TimeDelay%> or either of the offsets is zero the timer is turned off. Any other offset will cause the stringtable entry at that offset to be called at a rate one half of the <TimeDelay%> parameter value. The stringtable entry at <Offset2%> is 180 degrees out of phase with <Offset1%>, i.e. the timer service "flip flops" between <Offset1%> and <Offset2%>. Both parameters <Offset1%> and <Offset2%> are optional.

The time delay is in milliseconds. When timer 1 is enabled the time of day in 'HH:MM:SS' format is available in the string variable **T\$** and the current date in the form '21 Mar 2000' is in **U\$**.

Timers are a limited resource within the Windows environment. Avoid excessive use of timers whereby the response of the panel might be effected. Timers used for on screen clocks and any other non-critical timers should be disabled in the SLEEP stringtable and re-enabled in the WAKEUP string table which will minimise the possibility of running out of timers. The maximum <TimeDelay%> is 32767. For delays longer than 30 seconds set a 1 second timer and then increment a variable until the required number of seconds has elapsed.

5.20 Exec Unyield

From version 3.02.00

Syntax : **EU** <Var%>

Description:-

If <Var%> is 1 then ApplCore will not allow itself to be closed down by an Exec Kill instruction from another panel. If <Var%> is 0 then the panel may yield to the request to close down from another panel if resources are low. The default is 0.

5.21 Exec Variable

From version 3.02.00

Syntax : **EV** <String\$>

Description:-

Executes the contents of the string variable <String\$>. A very powerful command enabling remote control of applications. EV takes the contents of a variable and executes it as if it were on a line in a stringtable.

Note.

- 1) V2 Applcore EV will only accept a single command.
- 2) The EV command copies the **<String\$>** parameter to StringTable line 32768. Then StringTable line 32768 is executed by the ApplCore parser.

5.22 Exec Workstation

From version 3.02.00

Syntax : **EW <Var%>**

Description:-

Returns the workstation ID number as set in the APPLCORE.INI file and places it in **<Var%>**.

5.23 Exec eXecute

From version 3.05.26

Syntax : **EX <FunctionType%> <DLLName\$> <EntryPoint\$> <Result%> <Result\$> [<Param1\$> <Param2\$> <Param3\$> <Param4\$> <Param5\$>]**

Description:-

The EX command enables 'C' functions to be called in external DLLs. The DLLs can have any name and any number of functions. They are loaded and released each time the command is executed. Returns values into **Result%** and **Result\$**.

<FunctionType%> This parameter has a value of 0 or 1. Function type 0 executes the DLL function using a single parameter string. Function type 1 passes up to 5 individual strings. See below.

<DLLName\$> This parameter is the name of the DLL, such as: mathlib.dll

Starting with version 3.07.11, the load search path rules are as follows:

When a full path is not specified in the EX command ---

- Search in the local directory
- Then the SystemPath (bncs_config.ini)

When the DLL file name does not include the ".DLL" file extension, EX will concatenate a ".DLL" file extension prior to the file search.

<EntryPoint\$> This parameter is the name of the function to be called, such as: Add

<Result%> This is the ApplCore integer you wish the functions return value to be placed in.

<Result\$ >This is the ApplCore string value you wish the functions return date to be placed in. Up to 1024 characters can be returned.

<Param1\$> to **<Param5\$>** These are string values or string representation of integer values to be passed to the DLL function.

If **<FunctionType%>** is 0, then **<Param1\$>**, **<Param2\$>** and **<Param3\$>** are concatenated using intermediate spaces into a single parameter passed to the DLL.

If **<FunctionType%>** is 1, then the **<ParamN\$>** strings are built into an argument list array with the item count and string array passed to the DLL.

Example :- EX 0 mathlib.dll Add X% X\$ 23 19 0

5.24 Exec Z Output Debug String

From version 3.07.02

Syntax : **EZ <String\$>**

Description:-

Simply sends **<String\$>** to the Windows default debug IO stream. Use DBWIN or DBWIN32 to view the messages.

From version 3.08.01

Syntax : **EZ <Format\$> [<Var\$> or <Var%>...]**

EZ has been extended to include the text formatting functions provided in the SC command. See SC for formatting capabilities.

Note: The Exec Debug output string (EO) command function was supported in V2 ApplCore, but was deleted from the first versions of V3 (3.00.00). In V3 the command "EO" was reassigned to Exec Override starting with version 3.00.00. The original V2 Exec Debug output string (EO) command function has been reassigned to "EZ" (Exec Z Debug Output String) command in V3 ApplCore starting with version 3.07.02

6 eXternal Commands

This section describes a set of ApplCore commands which support driver external interfacing. These commands are used to communicate with the drivers directly without the delay of network access.

For additional information see document file: V3Applcore eXternal Driver Interfacing.pdf

6.1 eXternal Crosspoint

From version 3.05.28

Syntax : **XC <Driver%> <Source%> <Destination%>**

Description:-

Instructs a router driver **<Driver%>** to route **<Source%>** to **<Destination%>**. This command is similar in function to 'Router Crosspoint'.

Driver developer note: XC sends message BBC_CTRLCLIENTSTR to the driver.

From version 3.05.28

Syntax : **XC <DriverVar%> <Source%> <Destination%> [**<UpdateTallyTableOnly%>**]**

Description:-

The **<UpdateTallyTableOnly%>** parameter is optional and can be used to enable/disable network revertives. If **<UpdateTallyTableOnly%>** is 0 (default) the switch command is sent to the GRD and a revertive will be issued to the network, if non-zero, only the client external and GRD internal tally table are updated.

Driver developer note: XC sends message BBC_CTRLCLIENTSTR to the driver if **<UpdateTallyTableOnly%>** parameter is zero, if **<UpdateTallyTableOnly%>** parameter is non-zero message BBC_GRDSETDESTONLY is sent.

From version 3.08.21

Syntax : **XC <DriverVar%> <Source%> <Destination%>
[<UpdateTallyTableOnly%>] [<Mask\$>] [<WorkstationNumber%>]**

Description:-

Starting with version 3.08.21 optional parameters **<Mask\$>** and **<WorkstationNumber%>** are added. This requires that parameter **<UpdateTallyTableOnly%>** be specified. The addition of the **<Mask\$>** and **<WorkstationNumber%>** parameters results in the sending of a network style command to an "external" driver or automatic.

Example with four parameters **<UpdateTallyTableOnly%>** value of 0:

For a **<DriverVar%>** value of 960, **<Source%>** value of 10, **<Destination%>** value of 20, **<UpdateTallyTableOnly%>** value of 0 the following will be sent:

XC 960 10 20 0 (sends control client string message)

Example with four parameters - **<UpdateTallyTableOnly%>** value of 1:

For a **<DriverVar%>** value of 960, **<Source%>** value of 10, **<Destination%>** value of 20, **<UpdateTallyTableOnly%>** value of 1 the following will be sent:

XC 960 10 20 1 (sends GRD SET DEST message)

Example with six parameters:

For a **<DriverVar%>** value of 960, **<Source%>** value of 10, **<Destination%>** value of 20, **<UpdateTallyTableOnly%>** value of 0, **<Mask\$>** is 'REVFACS' and **<WorkstationNumber%>** value of 990, the following will be sent:

XC 960 10 20 0 'REVFACS' 990 (sends control client string message)

6.2 eXternal Data

From version 3.02.00

Syntax : **XD** [<DriverVar%>] [<IndexVar%>] [<DataVar%>] [<DataVar\$>]
[<WorkstationVar%>]

Description:-

When a panel has registered an external connection with a driver using the **eXternal Register** command and then used the **eXternal Intercept** command then the panel will receive notification of the driver receiving a command. The eXternal Data command allows the command data to be retrieved and processed in any way desired. The following example illustrates how it might be used.

The example driver is a GRD with an Id of 321 and 64 destinations. First register with the driver for a min max range of 10 to 20.

XR 321 10 20

After this command the panel will receive reverts from the driver for destinations 10 to 20. No commands will be intercepted and the GRD will process them as if the external control panel is not there. However the **eXternal Intercept** command is used to specify a stringtable line to execute when the driver receives a command as follows:

XI 321 8000

After executing the above command any **Router Crosspoint** commands received by the driver from the network, which lie between the min/max range specified previously with the **XR** command will result in line 8000 being executed. The eXternal Data command can then be used to extract the command information. If the incoming command did not lie between the min/max range then the stringtable line is not executed and the command is passed immediately back to the driver for processing as normal.

XD D% I% X% X\$ W%

If the above command is executed **D%** will be the driver number, **I%** will be the destination number **X%** will be the source, **X\$** will be the mask and **W%** will be the workstation number that sent the command.

For InfoDriver commands **X%** is a numerical representation of the slot value or '0' if the content is not numeric and **X\$** will be the slot contents.

Lock commands (RL, IL and GL) cannot be intercepted by Applcore and are passed back to the driver for processing as normal.

6.3 eXternal GetInfo

From version 3.02.00

Syntax : **XG** <Driver%> <Index%> <InfoVar\$>

Description:-

Requests <Driver%> to directly return in <InfoVar\$> the information for destination (or GPI IO, or InfoDriver slot) <Index%>.

6.4 eXternal Intercept

From version 3.02.00

Syntax : **XI** <Driver%> <Offset%>

Description:-

Specifies a stringtable line to be called when it is desired to intercept command to the driver for pre-processing, The <Offset%> line is called when ever a command has been intercepted and the eXternal Data command can be used to retrieve the details. See the eXternal Data command for a complete example.

6.5 eXternal redundancy Notify

From version 3.07.00

Syntax : **XN** <Driver%> <Offset%>

Description:-

Specifies a stringtable line to be called when the external client TxRx status changes. Use the **DM** command to retrieve the status.

Requires V3infodriver version 3.05.00 or later.

6.6 eXternal Poll

From version 3.02.00

Syntax : **XP <Driver%> <Min%> <Max%>**

Description:-

Polls **<Driver%>** for the status of all destinations between **<Min%>** and **<Max%>**. The information will be delivered into the stringtable offset that was set up when the eXternal Registration command was executed. This command is similar to the three commands 'Router Poll', 'GPI Poll' and 'Infodriver Poll'.

For Router drivers: **S%** will contain the source number and **S\$** will contain the name of the source as defined in the INI file for that router driver.

For GPI drivers: **S%** will contain the 0/1 state of the I/O Line. **S\$** will contain the name of the I/O line as defined in the INI file for that GPI driver.

For Info drivers: **S%** will contain the slot number and **S\$** will contain the slot contents.

6.7 eXternal Register

From version 3.05.36

Syntax : **XR <Driver%> <Min%> <Max%> <Offset%> <Offset Mode%> [<Dest%>] [<'ADD'>]**

Description:-

Registers with driver **<Driver%>**. Revertive information will be sent to your application every time the status of a destination between **<Min%>** and **<Max%>** changes. This command is similar to the three commands 'Router Register', 'GPI Register' and 'Infodriver Register'.

The source index will be placed in **S%** and stringtable offset **<Stringtable Offset%>** executed if **<OffsetMode%>** is 0 (STATIC). If **<OffsetMode%>** is 1 (DYNAMIC) then **<Stringtable Offset%>** is added to the input or output index and the resulting stringtable line is executed.

If the optional parameter variable **<Dest%>** is included then whenever a revertive occurs this variable will contain the index/slot value of the destination.

If the optional word 'ADD' is included then the registration is added to any existing registration for the device. If the 'ADD' is omitted then any previous registration is overwritten.

Note: If the optional word 'ADD' is used then the parameter variable **<Dest%>** must also be included.

6.8 eXternal Switch

From version 3.02.00

Syntax : **XS <Driver%> <I/O Line%> <StateVar%>**

From version 3.05.28

Syntax : **XS <Driver%> <I/O Line%> <StateVar%> [<UpdateTallyTableOnly%>]**

Description:-

Instructs a GPI driver to change the state of an I/O Line (indicated by the **<I/O Line%>** parameter). This command is similar in function to 'GPI Switch'.

Parameter **<StateVar%>** may be a value of 1 or 0. The **<UpdateTallyTableOnly%>** parameter is optional and can be used to enable/disable network revertives. If **<UpdateTallyTableOnly%>** is 0 (default) the switch command is sent to the GPID and a revertive will be issued to the network, if non-zero, only the client external and GPID internal tally table are updated.

From version 3.08.21

Syntax : **XS <Driver%> <I/O Line%> <StateVar%> [<UpdateTallyTableOnly%>] [<WorkstationNumber%>]**

Description:-

Starting with version 3.08.21 the optional parameter **<WorkstationNumber%>** is added. This requires that optional parameter **<UpdateTallyTableOnly%>** be specified. The addition of the **<WorkstationNumber%>** parameter results in the sending of a network style command to an "external" driver or automatic of the following format:

XS Driver% I/O Line% 0/1 UpdateTallyTableOnly% WorkstationNumber%

Example with four parameters - **<UpdateTallyTableOnly%>** value of 0:

For a **<DriverVar%>** value of 960, **<I/O Line%>** value of 10, **<StateVar%>** value of 1, **<UpdateTallyTableOnly%>** value of 0 the following will be sent:

XS 960 10 1 0

(sends control client string message)

Example with four parameters - **<UpdateTallyTableOnly%>** value of 1:

For a **<DriverVar%>** value of 960, **<I/O Line%>** value of 10, **<StateVar%>** value of 1, **<UpdateTallyTableOnly%>** value of 1 the following will be sent:

XS 960 10 1 1 (sends GPID SET I/O ONLY message)

Example with five parameters:

For a **<DriverVar%>** value of 960, **<I/O Line%>** value of 10, **<StateVar%>** value of 1, **<UpdateTallyTableOnly%>** value of 0 and **<WorkstationNumber%>** value of 990, the following will be sent:

XS 960 10 1 0 990 (sends control client string message)

6.9 eXtra Timer

From version 3.02.00

Syntax : **XT <Switch%> <var%>**

Description:-

If **<Switch%>** is 1 the system timer is started. If **<Switch%>** is 0 the timer is stopped and the result in milliseconds is placed in **<var%>**.

6.10 eXternal Unregister

From version 3.02.00

Syntax : **XU <Driver%> [<Min%>] [<Max%>]**

ApplJack : Yes

Description:-

Following this command no further status updates will be returned to the application for the **<Driver%>**. This command is similar in function to the three commands 'Router Unregister', 'GPI Unregister' and 'Infodriver Unregister'.

6.11 eXternal Write

From version 3.02.00

Syntax : **XW <Driver%> <Var\$> <Slot%>**

From version 3.05.28

Syntax : **XW <Driver%> <Var\$> <Slot%> [<UpdateTallyTableOnly%>]**

Description:-

Instructs an InfoDriver to place the contents of <Var\$> in <Slot%>. This command is similar in function to the command 'Infodriver Write'.

The <UpdateTallyTableOnly%> parameter is optional and can be used to enable/disable network revertives. If <UpdateTallyTableOnly%> is 0 (default) the slot message is sent to the Infodriver and a revertive will be issued to the network, if non-zero, only the client external and Infodriver slot are updated.

From version 3.08.21

Syntax : **XW <Driver%> <Var\$> <Slot%> [<UpdateTallyTableOnly%>] [<WorkstationNum%>]**

Description:-

Starting with version 3.08.21 optional parameter <WorkstationNumber%> is added. This requires that parameter <UpdateTallyTableOnly%> be specified. The addition of the <WorkstationNumber%> parameters results in the sending of a network style command to an "external" driver or automatic of the following format:

XW Driver% Var\$ Slot% UpdateTallyTableOnly% WorkstationNumber%

Example with four parameters - <UpdateTallyTableOnly%> value of 0:

For a <DriverVar%> value of 960, <Var\$> is 'ThisIsVarValue', <Slot%> value of 100, <UpdateTallyTableOnly%> value of 0 the following will be sent:

XW 960 'ThisIsVarValue' 100 0 (sends control client string message)

Example with four parameters - <UpdateTallyTableOnly%> value of 1:

For a <DriverVar%> value of 960, <Var\$> is 'ThisIsSlotValue', <Slot%> value of 100, <UpdateTallyTableOnly%> value of 1 the following will be sent:

XW 960 'ThisIsSlotValue' 100 1 (sends ID SET SLOT ONLY message)

Example with five parameters:

For a **<DriverVar%>** value of 960, **<Var\$>** is 'ThisIsVarValue', **<Slot%>** value of 100, **<UpdateTallyTableOnly%>** value of 1 and **<WorkstationNumber%>** value of 990, the following will be sent:

XW 960 'ThisIsVarValue' 100 1 990 (sends ID SET SLOT ONLY message)

For a **<DriverVar%>** value of 960, **<Var\$>** is 'ThisIsVarValue', **<Slot%>** value of 100, **<UpdateTallyTableOnly%>** value of 0 and **<WorkstationNumber%>** value of 990, the following will be sent:

XW 960 'ThisIsVarValue' 100 0 990 (sends control client string message)

7 File Commands

7.1 File Config Info

Syntax : **FC** [<ConfigDirectory\$>] [<SystemDirectory\$>] [<TempDirectory\$>]

Description

The File Config Info command returns the contents of the "ConfigPath", "SystemPath" and "TempPath" in the specified variables <ConfigDirectory\$>, <SystemDirectory\$> and <TempDirectory\$> from the [Config] section of the file: c:\bncs_config.INI

The command returns a max of 255 chars even though a directory path may be larger (512).

7.2 File Directory

Syntax : **FD** <directory\$> <PaneId%> <ListboxId%> <ResultVar%>

Description

Fills a list box with file names from a directory

8 GPI Commands

8.1 GPI File

From version 3.02.00

Syntax : **GF** <Driver%>

Description:-

Instructs the GPI driver <Driver%> to reload its Line Inversion table from the drivers INI file. Any changes you have made to any of the inversion bits will be immediately reflected in the system. In other words if you have changed the inversion bit for an i/o line then as soon as the table is reloaded the logical state of that line will change and the driver will reflect that change in state by sending a message to the network.

8.2 GPI Lock

From version 3.02.00

Syntax : **GL** <Driver%> <Min%> <Max%> <Switch%>

Description:-

If <Switch%> = 1 then all outputs between <Min%> and <Max%> are locked and cannot be toggled again without first unlocking them with <Switch%> = 0 or changing the lock state manually at the driver.

The GPI Lock command cannot be intercepted by the Applcore eXternal Commands and is always passed to the driver for processing as normal.

8.3 GPI Poll

From version 3.02.00

Syntax : **GP** <Driver%> <Min%> <Max%>

Description:-

Polls <Driver%> for status on all inputs or outputs between <Min%> and <Max%>. The information will be delivered into the stringtable offset that was set up when the GPI Register command was executed. All workstations receive the data returned so this method should be used sparingly on small ranges of data. Use GPI Query for larger ranges.

8.4 GPI Query

From version 3.02.00

Syntax : **GQ** <Driver%> <Min%> <Max%>

Description:-

Polls <Driver%> for status on all inputs or outputs between <Min%> and <Max%>. The information will be delivered into the stringtable offset that was set up when the GPI Register command was executed. The data transfer is carried out on a private session between the driver and the workstation requesting the data.

8.5 GPI Register

From version 3.06.01

Syntax : **GR** <Driver%> <Min%> <Max%> <Stringtable Offset%> <Offset Mode%> [<DestVar%>] ['ADD'] [<DriverVar%>]

Description:-

Registers with a GPI driver <Driver%>. Update information will be sent to your application every time the status of an input or output between <Min%> and <Max%> changes. The status of an input or output will be placed in S% and stringtable offset <Stringtable Offset%> executed if <OffsetMode%> is 0. If <OffsetMode%> is 1 then <Stringtable Offset%> is added to the input or output index and the resulting stringtable line is executed. If the optional parameter variable <DestVar%> is included then whenever a revertive occurs this variable will contain the index value of the input or output that has changed. If the word 'ADD' is appended as a 7th parameter then the registration is added to any existing registration for the device. If the 'ADD' is omitted then any previous registration is overwritten.

An example of using the 'ADD' feature should look something like this:

GR A% 1 16 21000 0 r%:GR A% 53 54 21000 0 r% 'ADD'

GR for device 'A%' indices 1 to 16 calling line 21000 'statically' when a revertive is received, plus **GR** for device 'A%' indices 53 to 54 also calling line 21000 'statically'. 'r%' will contain the GPI index number and could be used to filter an individual revertive using the appropriate code on line 21000.

If the optional <DriverVar%> variable was specified at registration time, the variable is filled with the value of <Driver%> then the stringtable line is called.

8.6 GPI Switch

From version 3.02.00

Syntax : **GS** <Driver%> <Index%> <0 or 1> <Switch%>

Description:-

Turns the given <Index%> on the GPI either on or off providing it has been configured at the driver as an output. <Driver%> is the number of the GPI you want to use.

If <Switch%> is 0 the command is buffered along with any other commands waiting to be sent until CSI's next regular transmit cycle. If <Switch%> is 1 the command is sent immediately along with any buffered commands already waiting.

8.7 GPI Unregister

From version 3.02.00

Syntax : **GU** <Driver%> (<Min%> <Max%>)

Description:-

Following this command no further status updates will be returned to the application for the <Driver%>. Selective deregistration of individual or sub-ranges of destinations is possible if the optional <Min%> <Max%> are added to the command.

If <Driver%> is 0, then ALL devices are unregistered for this panel.

9 Html Commands

9.1 Html Close

From version 3.02.00

Syntax : **HC** <Identifier%>

Description:-

Appends the required HTML closing syntax and closes the file.

9.2 HTML Delete

From version 3.02.00

Syntax : **HD** <Filename\$>

Description:-

Deletes the specified <Filename\$>, the file must be closed first.

9.3 HTML Open

From version 3.02.00

Syntax : **HO** <Identifier%> <Filename\$> <Title\$> <Refresh%>

Description:-

Creates and opens a new HTML file with the name **<Filename\$>** and a title of **<Title\$>**. The **<Filename\$>** should be a maximum of 8 characters, (not including the extension), should specify the file extension and can contain an explicit file path. If an explicit file path is included then backslashes must be replaced by forward slashes and colons by semi-colons. e.g. C:/bncs/my_htm.htm

An automatic refresh period may be specified in the **<Refresh%>** parameter otherwise no automatic update of the page will occur in the browser. The **<Identifier%>** value can be between 1 and 32 and this is used to uniquely identify up to 32 open files. The **<Identifier%>** must be used in all other HTML commands.

If a file already exists with the same name then the old file will be overwritten. If an identifier is already being used on an open file and you use the HO command with the same identifier then the file previously using that identifier will be closed and the new file opened.

9.4 HTML Rename

From version 3.02.00

Syntax : **HR <Filename\$1> <Filename\$2>**

Description:-

Renames **<Filename\$1>** to be **<Filename\$2>**.

9.5 HTML Write

From version 3.02.00

Syntax : **HW <Identifier%> <String\$>**

Description:-

Appends the **<String\$>** to the HTML file opened using **<Identifier%>**. Strings containing spaces should be enclosed in single quotes.

10 Info Command

10.1 Info Directory

From version 3.02.00

Syntax : **ID** <RemoteWS%> <RemoteDir\$> <LocalFile\$> <Index%>

Description:-

Requests from remote workstation <RemoteWS%> a directory listing of <RemoteDir\$> and that the resulting file be placed in a local file <LocalFile\$>. Success or failure of the request will be indicated by a call to stringtable offset <Index%>. **S%** contains the error number and **S\$** contains an error string. A zero indicates success. <RemoteDir\$> should be of the form 'C:/bncs/*.txt'

Example Syntax:-

ID 45 'C:/bncs/*.txt' 'C:/backup/Listing.txt' 2000

This will copy a directory listing from workstation 45's C:\bncs directory and put it in a file called Listing.txt on your local machine under C:\Backup\. The program will then execute line 2000

10.2 Info File

From version 3.02.00

Syntax : **IF** <Driver%> <Switch%> <Filename\$><Var\$>

Description:-

Requests <Driver%> to send the string <Vars\$> to the file <Filename\$>. If <Switch%> is 0 the string will be added to the end of <Filename\$> if it exists, otherwise it will be created. If <Switch%> is 1 a new file will be created with the name <Filename\$>. Any previous file with the same name will be overwritten.

10.3 Info Get

From version 3.02.00

Syntax : **IG** <RemoteWS%> <RemoteFile\$> <LocalFile\$> <Index%>

Description:-

Requests from remote workstation <**RemoteWS%**> the file <**RemoteFile\$**> to be copied to the local file <**LocalFile\$**>. Success or failure of the request will be indicated by a call to stringtable offset <**Index%**>. **S%** contains the error number and **S\$** contains an error string. A zero indicates success.

10.4 Info Hardcopy

From version 3.02.00

Syntax : **IH <Driver%> <Var\$> <Switch%>**

Description:-

Requests <**Driver%**> to print <**Var\$**> on the logging printer. If <**Switch%**> is 0 the string will be added to a buffer. When sufficient lines of strings are held in the buffer to fill a page then the whole buffer will be sent to the printer. If <**Switch%**> is 1 it forces the buffer to be sent to the printer regardless of the number of lines it contains.

10.5 Info Lock

From version 3.02.00

Syntax : **IL <Driver%> <Min%> <Max%> <Var%>**

Description:-

If <**Var%**> is 1 then all slots on <**Driver%**> between <**Min%**> and <**Max%**> are locked. The contents of a slot can not be changed when it is locked.

Use <**Var%**> set to 0 to unlock a slot or range of slots.

The Info Lock command cannot be intercepted by the Applcore eXternal Commands and is always passed to the driver for processing as normal.

Note: "EnableLocks" must be set to 1 in the InfoDriver .INI file (Dev_xxx.ini) for this command to affect the specified InfoDriver. From the InfoDriver documentation:

EnableLocks=0

If set to 1 infodriver slots can be locked, by default locking is disabled.

10.6 Info Poll

From version 3.02.00

Syntax : **IP <Driver%> <Min%> <Max%>**

Description:-

Requests **<Driver%>** to send all information contained between slots **<Min%>** and **<Max%>**.. The information is delivered to the stringtable offset supplied as a parameter when the IR command was used. All workstations receive the data returned so this method should be used sparingly on small ranges of data. Use Info Query for larger ranges.

10.7 Info Query

From version 3.02.00

Syntax : **IQ <Driver%> <Min%> <Max%>**

Description:-

Requests **<Driver%>** to send all information contained between slots **<Min%>** and **<Max%>**.. The information is delivered to the stringtable offset supplied as a parameter when the IR command was used. The data transfer is carried out on a private session between the driver and the workstation requesting the data.

10.8 Info Register

From version 3.06.01

Syntax : **IR <Driver%> <Min%> <Max%> <Offset%><Offset Mode%> [<Slotvar%>] [<'ADD'>] [<DriverVar%>]**

Description:-

Requests the **<Driver%>** to register the calling application to be informed whenever strings in slots between **<Min%>** and **<Max%>** change.

The information in a slot will be placed in S\$ and the stringtable offset **<Offset%>** will be executed if **<OffsetMode%>** is 0. For example if the **<Offset%>** is 2000 then stringtable line 2000 will be executed. This is known as 'static registration'.

If **<OffsetMode%>** is 1 then **<Offset%>** is added to the InfoDriver slot number and the resulting stringtable line is executed.

For example if the **<Offset%>** is 2000 and the slot number is 15 then stringtable line 2015 will be executed. This is known as 'dynamic registration'.

If the optional parameter variable **<SlotVar%>** is included then whenever a revertive occurs this variable will contain the Infodriver slot number. If the word **'ADD'** is appended as a 7th parameter then the registration is added to any existing registration for the device.

Note if using the **'ADD'** parameter then both instances of the registration need to be static i.e. always calling the same single stringtable line. If the **'ADD'** is omitted in the second registration then any previous registration is overwritten.

An example of using the **'ADD'** feature should look something like this:

IR A% 1 16 21000 0 r%:IR A% 53 54 21000 0 r% 'ADD'

IR for device **'A%'** slots 1 to 16 calling line 21000 'statically' when a revertive is received, plus IR for device **'A%'** slots 53 to 54 also calling line 21000 'statically'. **'r%'** will contain the InfoDriver slot number and could be used to filter an individual revertive using the appropriate code on line 21000.

If the optional **<DriverVar%>** variable was specified at registration time, the variable is filled with the value of **<Driver%>** then the stringtable line is called.

10.9 Info Send

From version 3.02.00

Syntax : **IS <LocalFile\$> <RemoteWS%> <RemoteFile\$> <Index%>**

Description:-

Requests that the local file **<LocalFile\$>** be copied to the remote file **<RemoteFile\$>** on workstation **<RemoteWS%>**. Success or failure of the request will be indicated by a call to stringtable offset **<Index%>**. **S%** contains the error number and **S\$** contains an error string. A zero indicates success.

10.10 Info Unregister

From version 3.02.00

Syntax : **IU** <Driver%> ([<Min%>] [<Max%>])

Description:-

Following this command no further status updates will be returned to the application for the specified device. Selective deregistration of individual or sub-ranges of destinations is possible if the optional [<Min%>] [<Max%>] are added to the command.

If <Driver%> is 0, then ALL devices are unregistered for this panel.

10.11 Info Write

From version 3.02.00

Syntax : **IW** <Driver%> <Var\$> <Index%> <Switch%>

Description:-

Requests the <Driver%> to store the string <Var\$> in slot <Index%>.

If <Switch%> is 0 the command is buffered along with any other commands waiting to be sent until CSI's next regular transmit cycle. If <Switch%> is 1 the command is sent immediately along with any buffered commands already waiting.

11 Jump Commands

11.1 Jump Always

From version 3.02.00

Syntax : **JA** <Line%>

Description:-

Jumps directly to line <Line%>. If the line number is zero then execution continues on the same line.

11.2 Jump Compare

From version 3.02.00

Syntax : **JC** <Var1\$> <Var2\$> <Var3%> <Var4%>

Description:-

If <Var1\$> is equal to <Var2\$> then stringtable line <Var3%> will be executed after the current line has been completed otherwise stringtable line <Var4%> will be executed. A If the line number is zero then execution continues on the same line. The comparison is case sensitive (for case insensitive comparison copy/convert both params to lower case using TE command before using in JC)

11.3 Jump Date

From version 3.02.00

Syntax : **JD** <Date1\$> <Date2\$> <Var3%> <Var4%> <Var5%>

Description:-

Compares <Date1\$> with <Date2\$>. Jumps to <Var3%> if equal, <Var4%> if less and <Var5%> if greater. The format of the date should be the same as provided by the ApplCore clock, i.e. 14-April-2005. The format 14/April/2005 can also be used. If the line number is zero then execution continues on the same line.

11.4 Jump Equal

From version 3.02.00

Syntax : **JE** <Var1%> <Var2%> <Var3%> <Var4%>

Description:-

If <Var1%> is equal to <Var2%> then stringtable line <Var3%> will be executed after the current line has been completed otherwise stringtable line <Var4%> will be executed. If the line number is zero then execution continues on the same line.

11.5 Jump Goto

From version 3.02.00

Syntax : **JG** <Index%> <Offset%> <VarList3\$>

Jump Goto gets item number <Index%> from comma delimited list <Var\$>, adds the number found there to <Offset%> and jumps to that line.

11.6 Jump Instring

From version 3.02.00

Syntax : **JI** <Var1\$> <Var2\$> <Var3%> <Var4%>

Description:-

If <Var1\$> is contained within <Var2\$> then stringtable line <Var3%> will be executed after the current line has been completed otherwise stringtable line <Var4%> will be executed. If the line number is zero then execution continues on the same line.

11.7 Jump Less

From version 3.02.00

Syntax : **JL** <Var1%> <Var2%> <Var3%> <Var4%>

Description:-

If <Var1%> is less than <Var2%> then stringtable line <Var3%> will be executed after the current line has been completed otherwise stringtable line <Var4%> will be executed. If the line number is zero then execution continues on the same line.

11.8 Jump Offset

From version 3.02.00

Syntax : **JO** <Var1%> <Var2%>

Description:-

<Var1%> is added to <Var2%> and the resulting line number is executed. If the line number is zero then execution continues on the same line.

11.9 Jump Return

From version 3.02.00

Syntax : **JR**

Description:-

If a subroutine is being executed the **JR** command will return the interpreter to the next command on the line (or next line) after the Jump Subroutine command which invoked the subroutine.

11.10 Jump Subroutine

From version 3.02.00

Syntax : **JS** <Var1%>

Description:-

Jump to subroutine at line <Var1%>. Commands on that line and all subsequent lines of commands will be executed until either a blank line, Jump Return command or another Jump Subroutine command is encountered.

If a Jump Return command is found then the interpreter jumps back to the command following the Jump Subroutine command that preceded the return command. If there were more commands on the line after the JS command then execution will continue with the next command on the line.

If another Jump Subroutine is encountered then that subroutine will be executed and then, after it has returned, the rest of the current subroutine will be completed.

There can be up to 1000 nested subroutines.

If a blank line is found then execution stops and all pending Jump Returns are cleared.

Note: You should not use another type of jump command to leave a subroutine.

11.11 Jump Time

From version 3.02.00

Syntax : **JT** <Time1\$> <Time2\$> <Var3%> <Var4%> <Var5%>

Description:-

Compares <Time1\$> with <Time2\$>. Jumps to <Var3%> if equal, <Var4%> if less and <Var5%> if greater. The format of the time string can be HH:MM or HH:MM:SS or HHMMSS. Jump Time **only** compares the hours and minutes fields of <Time1\$> <Time2\$>. The seconds are ignored. If the line number is zero then execution continues on the same line.

12 Key Commands

12.1 KeyRegister

From version 3.02.00

Syntax : **KR** **<Offset%>** **<Char\$>** **<Ascii%>** [**<ControlMode%>**]

Description:-

This is used to allow key presses to be placed one at a time into variables. **<Offset%>** is the stringtable offset to be called when a key is pressed.

During the key events Variable **<Char\$>** will contain the character and variable **<Ascii%>** will contain the ASCII value.

This function will remain active until the Key Unregister command is used. Whilst Key Register is active a timer continually sets the input focus to be ApplCore's main window, hence any control which has the focus will lose it. The characters arriving in the variables can be assembled into complete strings using other commands.

Parameter [**<ControlMode%>**] is added starting in version 3.07.02 (explain below).

From version 3.05.39

Documentation using V3 version 3.05.39 of the **KR** command:

The return codes for the Control keys, Modifier keys, Function keys are documented in the following table. Keycodes 0 - 31 are delivered to the application as these correspond to ASCII Control "@" through Control "_". Additionally control and meta keys are supported with the following values returned via the **<Ascii%>** parameter.

Complete key tables for mode 0 & 1 operation are defined in **Appendix I – Keyboard Tables**.

Control Key (s)	Value
Control @	0
Control A	1
Control B	2
Control C	3
Control D	4
Control E	5
Control F	6
Control G	7
Control H also <Backspace>	8
Control I (TAB)	9
Control J	10
Control K	11
Control L	12
Control M also <Return>	13
Control N	14
Control O	15
Control P	16
Control Q	17
Control R	18
Control S	19
Control T	20
Control U	21
Control V	22
Control W	23
Control X	24
Control Y	25
Control Z	26
Control [27
Control BACK SPACE	28
Control]	29
Control ^ carrot / carat character	30
Control _ underscore	31
Control 0 thru Control 9	48 thru 57

Function Keys	VALUE
Left Arrow	129
Right Arrow	130
Up arrow	131
Down Arrow	132
Insert	133
Delete	134
Home	135
End	136
Prior (Page Up)	137
Next (Page Down)	138
Num Lock	144
Scroll Lock	145
F1	146
F2	147
F3	148
F4	149
F5	150
F6	151
F7	152
F8	153
F9	154
F10	N/A (ControlMode 0 & 1 155 (ControlMode 2 & 3))
F11	156
F12	157

From version 3.07.02

Syntax : **KR** <Offset%> <Char\$> <AsciiValue%> [<ControlMode%>]

The **KR** command has been enhanced to better support keyboard key capture - while maintaining the original **KR** function. The new optional fourth parameter [<ControlMode%>] has been added to support specifying additional operational modes (below).

Parameter <Char\$> is the character return string variable such as **K\$** -- that is if the key press is an ASCII key value between the values of 'A' to 'Z', 'a' to 'z', '0' to '9' and the various punctuation characters. If a meta key, modifier key or VK key is pressed the decimal value is returned. For example pressing a SHIFT key, the string variable is NULL, the value is 16.

Parameter <AsciiValue %> is the character return value variable such as **K%** -- that is if the key press is an ASCII value if applicable (see following table for various meta and VK keys). If a meta key, modifier key or VK key is pressed the decimal value is returned. For example pressing a SHIFT key, the value variable returned is 16 (for SHIFT).

The state of the modifier keys (Shift, Control, Alt, CapLock keys) are maintained and returned in the **S%** variable, each time a key event is signalled the **S%** variable contains the modifier keys state. The modifier keys values are defined in the following table:

Modifier Keys	AsciiValue	Modifier Flags S% BIT VALUE
Shift	16	1 (bit 0, 0x01) Shift bit
Control (Ctrl)	17	2 (bit 1, 0x02) Control bit
Alt	18	4 (bit 2, 0x04) Alt bit
Alt Gr (for U.K. keyboard)	18	8 (bit 3, 0x08) Alt Gr bit
Cap Lock	20	n/a

The new optional parameter <ControlMode%> offers FOUR modes of operation for **KR**:

0 – Sets the original operational mode (default).

1 – Sets operational mode and forces the main window to foreground and focus.
New support for reporting key UP and for VK and modifier keys (Shift CapLock, etc.)

key down: call at <Offset%> LineNumber,
key up: call at <Offset%> LineNumber + 200.

Modifier keys returned in S%.

2 – Sets operational mode scheme which allows for capture of keystrokes regardless of focus or Z order position. This mode of operation also includes support for such keys as the **F10** key and **Alt Gr** key (ALT Graphics). This mode requires DLL file: KeyHkAC.dll be located in the “windows” directory along with other BNCS DLL’s.

key down: call at **<Offset%>** LineNumber,
key up: call at **<Offset%>** LineNumber + 200.

3 – The same as 2, but alters certain key mappings.

Modifier Key Explanation Tables for <ControlMode%> 2 and 3:

ControlMode 2				
Modifier Key State	Key Pressed	Returned: k\$	Returned: k%	Returned: S%
None (caps lock off)	a	'a'	97	0
Shift (caps lock off)	a	'A'	65	Shift Bit Set
None (caps lock on)	a	'A'	65	0
Shift (caps lock on)	a	'a'	97	Shift Bit Set
Control	a	"	1	Control Bit Set
Alt	a	"	97	Alt bit set
Alt Gr	a	"	97	Alt Gr bit set + Control bit set + Alt bit set
Virtual key reporting				
Virt Key (F1, Up, etc)	F1	"	146	0
Virt Key + Shift	F1	"	146	Shift Bit set
Virt Key + Alt	F1	"	146	Alt bit set
Virt Key + Control	F1	"	146	Control Bit set
Virt Key + Alt Gr	F1	"	146	Alt Gr bit set + Control bit set + Alt bit set

ControlMode 3				
Modifier Key State	Key Pressed	Returned: k\$	Returned: k%	Returned: S%
None (caps lock off)	a	'a'	97	0
Shift (caps lock off)	a	'A'	97	Shift Bit Set
None (caps lock on)	a	'A'	97	0
Shift (caps lock on)	a	'a'	97	Shift Bit Set
Control	a	"	97	Control Bit Set
Alt	a	"	97	Alt bit set
Alt Gr	a	"	97	Alt Gr bit set
Virtual key reporting				
Virt Key (F1, Up, etc)	F1	"	146	0
Virt Key + Shift	F1	"	146	Shift Bit set
Virt Key + Alt	F1	"	146	Alt bit set
Virt Key + Control	F1	"	146	Control Bit set
Virt Key + Alt Gr	F1	"	146	Alt Gr bit set + Control bit set + Alt bit set

NOTES for <ControlMode> 2 & 3 operation:

Library KeyHkAC.dll is required for <ControlMode%> 3. Version: 3.2.0.0, Dated: 08-Feb-2011

Locking Keys: The **CapLock**, **NumLock** and **ScrollLock** keys generate a key down event during the start-up of a KR 3 command if they are set. No event is sent if they are not set. The **CapLock** key operates a toggle lock action, while the **Shift**, **ALT**, etc. are active only while depressed.

No Character returns: The notation in the above table " (two single quotes) signifies the returned string is of zero length, i.e. no character returned.

ALT GR: The **Alt Gr** key requires special consideration.

When **Alt Gr** is pressed, first a "**Control**" (keycode: 17) is emitted, then an "**Alt**" (keycode: 18) is emitted, the **modifier flags** are set as **Control** (2) + **Alt** (4) + **Alt Gr** (8) for a value of 14 (HEX 0x0E).

When **Alt Gr** is released, first a "**Control**" (keycode: 17) is emitted, then an "**Alt**" (keycode: 18) is emitted, the **modifier flags** are reset to zero

.

NOTES for <ControlMode> 2 & 3 operation (continued):

KEYBOARD TYPES: Only US and UK standard keyboard types are supported at this time. Other keyboard special character arrangements will require engineering review. Complete key tables for mode 2 & 3 operation are defined in **Appendix II– Keyboard Tables**.

On the standard UK keyboard there are a number of special keys, including the EURO symbol, and two EBCDIC symbols located on the Grave Accent key in the Shift and Alt Gr positions, and different key / glyph arrangements relative to the US standard keyboard. For example:

Special Key	Grave accent		Mode 2	Mode 3	
Modifier Key State	Key Pressed	Returned k\$	Returned k%	Returned k%	Returned S%
None (caps lock off)	`	``	96	96	0
Shift (caps lock off)	` (note 1)	UK: `¬ US: ~	UK: 172 US: 126	96	Shift Bit Set
None (caps lock on)	`	``	96	96	0
Shift (caps lock on)	` (note 1)	UK: `¬ US: ~	UK: 172 US: 126	96	Shift Bit Set
Control	`	"	96	96	Control Bit Set
Alt	`	"	96	96	Alt bit set
Alt Gr	` (note 2)	' '	166	96	Alt Gr bit set + Control bit set + Alt bit set

Note 1: Grave Accent + Shift refers to EBCDIC symbol decimal number 95 (¬ Logical NOT).

Note 2: Grave Accent + Alt Gr refers to EBCDIC decimal symbol number 106 (¦ broken bar / split vertical bar) alternatively may be EBCDIC decimal symbol number 79 (| vertical bar).

Special Key	Number 3		Mode 2	Mode 3	
Modifier Key State	Key Pressed	Returned k\$	Returned k%	Returned k%	Returned S%
None (caps lock off)	3	'3'	51	51	0
Shift (caps lock off)	3 (note 1)	UK: '£' US: '#'	UK: 163 US: 35	51	Shift Bit Set
None (caps lock on)	3	'3'	51	51	0
Shift (caps lock on)	3 (note 1)	UK: '£' US: '#'	UK: 163 US: 35	51	Shift Bit Set
Control	3 (note 2)	"	51	51	Control Bit Set
Alt	3	"	51	51	Alt bit set
Alt Gr	3	^	51	51	Alt Gr bit set + Control bit set + Alt bit set

Note 1: Number 3 + Shift refers to the British Pound symbol £ with UK keyboard and # for US keyboard.

Note 2: Not defined, returned value is the control key.

Special Key	Number 4		Mode 2	Mode 3	
Modifier Key State	Key Pressed	Returned: k\$	Returned: k%	Returned: k%	Returned: S%
None (caps lock off)	4	'4'	52	52	0
Shift (caps lock off)	4 (note 1)	'\$'	36	52	Shift Bit Set
None (caps lock on)	4	'4'	52	52	0
Shift (caps lock on)	4 (note 1)	'\$'	36	52	Shift Bit Set
Control	4	"	52	52	Control Bit Set
Alt	4	''	52	52	Alt bit set
Alt Gr	4 (note 2)	'€'	128	52	Alt Gr bit set + Control bit set + Alt bit set

Note 1: Number 4 + Shift refers to the Dollar symbol \$.

Note 2: Number 4 + Alt Gr refers to the Euro symbol €.

12.2 Key Unregister

From version 3.02.00

Syntax : **KU**

Description:-

Removes any key registration.

13 List Commands

13.1 List All

From version 3.02.00

Syntax : **LA** <String\$> <PanelId%> <ListBoxId%> <MemLoc%>

Description:-

Searches the list box <ListBoxId%> on panel <PanelId%> for <String\$>. The number of instances found are put in <MemLoc%>. Indices placed in <MemLoc%> + 1 ect. E.G. if three instances of <String\$> are found then their indices would be placed in <MemLoc%> + 1, <MemLoc%> + 2 and <MemLoc%> + 3.

13.2 List BlankErase

From version 3.02.00

Syntax : **LB** <PanelId> <ListBoxId%>

Description:-

Removes all blank lines from <PanelId> <ListBoxId %>.

13.3 List Clear

From version 3.02.00

Syntax : **LC** <PanelId%> <ListBoxId%>

Description:-

Deletes the entire contents of the listbox.

13.4 List Delete

From version 3.02.00

Syntax : **LD** <PanelId%> <ListBoxId%>

Description:-

Deletes the currently selected text from the listbox.

13.5 List Erase

From version 3.02.00

Syntax : **LE <String\$> <PanelId%> <ListBoxId%> <Mode%>**

Description:-

If **<Mode%>** = 0 then the first instance of **<String\$>** is erased from the listbox. If **<Mode%>** = 1 then all instances of **<String\$>** are erased.

13.6 List Find

From version 3.02.00

Syntax : **LF <String\$> <PanelId%> <ListBoxId%> <ReturnVar%> <Sub%> [<FoundCount%>]**

Description:-

Searches for the list box entry **<String\$>** in the listbox pointed to by **<PanelId%>** and **<ListBoxId%>**. If **<Sub%>** is set to '0' then the search is for an exact compare of the string, if set to '1' then the search is for a substring and if set to '2' it searches for a listbox entry that starts with **<String\$>**. If the string is not found **<ReturnVar%>** will be zero, otherwise it will contain the entry position of the string in the listbox.

Version 3.5.24 adds **<Sub%>** with values of 3, 4 & 5, which perform the same test as 0, 1 & 2 but without checking for string case.

Search commands values for <Sub%>:

- 0 search for an exact compare
- 1 search for a substring
- 2 searches for a entry starting with **<String\$>**

- 3 search for an exact compare (case insensitive)
- 4 search for a substring (case insensitive)
- 5 searches for a entry starting with **<String\$>** (case insensitive)

From version 3.07.02

New Search commands: Starting with version 3.07.02, LF has been enhanced to support the following search functions:

- Case sensitive
- Reducing list search
- Search from start / current
- Search within
- Search Descending / Ascending
- Alphabetical / Insert
- Exact match / partial match

In order for these new search types to be backward compatible (with versions prior to 3.07.02), they are being introduced via "new searches" command tag value for **<Sub%>** with the new types specified as bits in the bit maps. That is, the original search commands are specified by **<Sub%>** values 0 to 5 as specified above.

The new searches are specified by **<Sub%>** being set to a **command tag** which is a value of 0x40000000 (hexadecimal), plus one or more of the search specifications (OR'd together with the command tag).

The new individual searches types are coded:

- 0x40000001 -- Bit 0 = **Case Sensitive** (bit set to 1)
Case Insensitive (bit set to 0)
- 0x40000002 -- Bit 1 = **Reducing list search** (bit set to 1)
Non-Reducing list search (bit set to 0)
- 0x40000004 -- Bit 2 = **Search from current**** (bit set to 1)
Search from start (bit set to 0)
- 0x40000008 -- Bit 3 = **Exact match** (bit set to 1)
Inexact (partial) match (bit set to 0)
- 0x40000010 -- Bit 4 = **Search within string** (bit set to 1)
Start of string (bit set to 0)
- 0x40000020 -- Bit 5 = **Descending** (bit set to 1)
Ascending (bit set to 0)
- 0x40000040 -- Bit 6 = **Alphabetical** (bit set to 1)
Index (insert) order (bit set to 0)
- 0x40000080 -- Bit 7 = **Space Free Search** (bit set to 1)
Default behaviour (bit set to 0)
- 0x40001000 -- Bit 12 = **Reload ListBox** original data then perform search (bit set to 0)
Use current ListBox contents (bit set to 1)

** Search from current - search starts from current selected list element.

Detailed Descriptions:

The default search, i.e. a search with all of the following bits set to zero plus the command tag (0x40000000 hexadecimal) -- produces a search for the specified string with these attributes:

Case Insensitive + Non-Reducing list search + Search from start + Inexact (partial) match

The default case is for the search to start at the beginning of the list, at entry 1. If the search string is cleared ("", i.e. zero length name), the ListBox will re-populate with the full list of original data list entries.

Case Sensitive (bit set to 1) 0x40000001 -- Bit 0
Case Insensitive (bit set to 0)

This search style selects whether the case of the alpha characters are considered during the search. The default is 0, "**Case Insensitive**". If bit 0 is set to 1 then the search considers the case of the alpha characters.

Example list entries:

- "abcdefghijklmn"
- "abc123"
- "abc456"
- "foobar123"
- "foobar456"

Searching "**Case Sensitive**" for "ABC" will return NO entries identified. Searching "**Case Insensitive**" for "ABC" will return THREE entries identified and reduce the list to just the first THREE entries.

Reducing list search (bit set to 1) 0x40000002 -- Bit 1
Non-Reducing list search (bit set to 0)

This search style selects whether the list is reduced when characters are matched in search.

If bit 1 is set to zero then the search does not reduce the number of entries in the list on match during the search.

If bit 1 is set to one then during the search non-matching entries in the list are deleted. The default is 0, "**Non-Reducing list search**".

Example list entries:

- "abcdefghijklmn"
- "abc123"
- "abc456"
- "foobar123"
- "foobar456"

Searching **Non-Reducing** for "abc" will return NO entries identified.

Searching **Reducing + Inexact** for "abc" will return THREE entries identified and reduce the list to just the first THREE entries.

Searching **Reducing + Inexact** for "abcd" will return ONE entries identified and reduce the list to just the first entry.

Returns: **<ReturnVar%>** set to the first (matching) element in the reduced list.

Search from current (bit set to 1) 0x40000004 -- Bit 2
Search from start (bit set to 0)

This search style selects whether the list is searched from the current beginning (entry number 1) in the list or whether the search starts from the current cursor position (high-lighted entry).

If bit 2 is set to zero then the search starts at the beginning.

If bit 2 is set to one the search starts at the current cursor position. The default is 0, **"Search from start"**.

Exact match (bit set to 1) 0x40000008 -- Bit 3
Inexact (partial) match (bit set to 0)

This search style selects whether the list is searched for exact matches or a partial match.

If bit 3 is zero then the search is **"Inexact (partial) match"**.

If bit 3 is set to one the search is **"Exact match"**. The default is 0, **"Inexact (partial) match"**.

Example list entries:

- "abcdefghijklmn"
- "abc123"
- "abc456"
- "foobar123"
- "foobar456"

Searching exact for "abc" will return NO entries identified. Searching inexact for "abc" will return THREE entries identified.

Search within string (bit set to 1) 0x40000010 -- Bit 4
Start of string (bit set to 0)

This search style selects whether an entry in the list is searched from the start of the string name of a list entry or whether substring searches are performed (i.e. the string to be searched for may be located anywhere within the list entry).

If bit 4 is set to zero then the search is **"Start of string"**.

If bit 4 is set to one the search is **"Search within string"**. The default is 0, **"Start of string"**.

Ascending (bit set to 0) 0x40000020 -- Bit 5
Descending (bit set to 1)

This search command reorganizes the presentation of the list such that the list is presented in **Ascending** order or **Descending** order.

If **Ascending** is selected then the original list entries data in insert order with the beginning of the list presented first, i.e. at the top of the list.

If **Descending** is selected then the original list entries data in insert order with the end of the list presented first, i.e. at the top of the list display.

If bit 5 is set to zero then **Ascending** order is selected.

If bit 4 is set to one the list is presented in **Descending** order. The default is 0, **Ascending**.

Alphabetical (bit set to 1) 0x40000040 -- Bit 6
Index (insert) order (bit set to 0)

This search style selects whether an entry in the list is searched for from the start of the string name of an entry or whether substring searches are performed.

If bit 6 is zero then the search is "**Start of string**".

If bit 6 is set to one the search is "**Search within string**". The default is 0, "**Index (insert) order**".

Space Free Search (bit set to 1) 0x40000080 -- Bit 7
Default (bit set to 0)

This search style removes spaces from an entry in the list and the from the string name before the comparison is performed.

Example: A search for "BBC1" will be return "BBC 1" if "**Space Free Search**" is set.

If bit 7 is one then the search is "**Space Free Search**".

If bit 7 is set to zero the search compare includes any space characters in the list entry and the string name. The default is 0.

Reload ListBox (reload) (bit set to 0) 0x40001000 -- Bit 12 (don't reload) (bit set to 1)

This search command causes a reload of the ListBox with the original list box contents / data.

If bit 8 is set to one this is a no-op, that is, if the bit is set then the list box is not reloaded.

If bit 8 is set to zero the list box contents are restored ("reload") and then any other specified searches are performed. The default is 0, "reload".

An optional parameter is returned [**<FoundCount%>**] which indicates the number of matching entries found. A value of zero would indicate that no entries match. A value of one would indicate that a unique match has been found.

Because these search types are coded as a bitmap, they can be "mixed and matched", thus a values of 0x40000003 encodes a search for "**Reducing list search**" plus "**Case Sensitive**".

A value of 0x4000000F encodes a search for "**Reducing list search**" with "**Case Sensitive**", "**Search from current**" and "**Exact match**".

Example Usage:

The following searches the list currently loaded in ID 12001 for "Stoke". The parameter **<Sub%>** is loaded with *command tag* value and *Reducing Search* command.

```
TL '0x40000002' N$           :EI note 0x40000000 = 1073741824.
NC N$ N%                   :EI convert hexadecimal string to integer numeric value
LF 'Stoke' P% 12001 C% N% D% :EI return results in C% and D%
```

13.7 List Get

From version 3.02.00

Syntax : **LG** <PanelId%> <ListBoxId%> <Var\$> <ReturnVar%>

Description:-

Copies the selected text string from a listbox control and puts it in a <Var\$>. The index of the selected text is placed in <ReturnVar%>.

13.8 List Hardcopy

From version 3.02.00

Syntax : **LH** <PanelId%> <ListBoxId%>

Description:-

Copies the entire contents of a listbox control to the printer.

13.9 List Item

From version 3.02.00

Syntax : **LI** <Switch%> <ListBoxItem%> <PanelId%> <ListBoxId%>
<Position%>

Description:-

If <Switch%> is 0 any the highlight or blue selection bar will be removed and <ListBoxItem%> is ignored.

If <Switch%> is 1 then <ListBoxItem%> will be highlighted.

<ListBoxItem%> is the index number of the selected item in the list box. Range 1 to 32766. If there are more items in the listbox that can be displayed at one time then the <Position%> parameter can be used to make sure that the highlighted item is actually visible in the list box. <Position%> is the number of lines down in the listbox that the highlighted item should appear. Range 0 to 65535

13.10 List Load

From version 3.02.00

Syntax : **LL** <Filename\$> <Min%> <Max%> <PanelId%> <ListBoxId%>

Description:-

Loads lines from a text file into a list box. All lines between <Min%> and <Max%> are loaded. Blank lines are omitted. The <Filename\$> can include the full path, however colons should be replaced by semi-colons and back slashes replaced by forward slashes. Eg. C:/BNCS/FILE.TXT

13.11 List Null

From version 3.02.05

Syntax : **LN** <PanelId%> <ListBoxId%>

Description:-

This effectively clears the keyboard edit buffer by momentarily switching the keyboard focus away from the listbox and back to it again.

13.12 List Put

From version 3.02.00

Syntax : **LP** <Variable\$> <PanelId%> <ListBoxId%>

Description:-

Puts text from <Variable\$> into a listbox. If the text box was configured in the dialog box editor to be an 'Auto-Sort' list box the new text will be inserted in its correct alphabetical position in the list. If the list box is not set to 'Auto-Sort' the new text will be added to the bottom of any existing text.

13.13 List Quantity

From version 3.02.00

Syntax : **LQ <PanelId%> <ListBoxId%> <Var%>**

Description:-

Looks for the number of entries contained in the listbox **<PanelId%> <ListBoxId%>** and returns the result in **<Var%>**.

13.14 List Router

From version 3.02.00

Syntax : **LR <Router%> <Switch%> <Min%> <Max%> < PanelId%>
<ListBoxId%> [<Index List\$>]**

Description:-

List Router will extract all the source or destination names for the given **<Router%>** between **<Min%>** and **<Max%>** and place the information in the listbox pointed to by **< PanelId%>** and **<ListBoxId%>**

<Switch%> can be 0 to 9 inclusive, if it is set to 0 then source names will be extracted, 1 destination names, 2 names out of database 2 etc.

If **<Min%>** =0 and **<Max%>** =0 the list of indices to use will come from the comma delimited list supplied in optional parameter **<Index List\$>**

13.15 List Save

From version 3.02.00

Syntax : **LS <Filename\$> <PanelId%> <ListBoxId%>**

Description:-

Save a listbox to a text file. The **<Filename\$>** can include the full path, however colons should be replaced by semi-colons and back slashes replaced by forward slashes. Eg. C:/BNCS/FILE.TXT

13.16 List Trim

From version 3.02.00

Syntax : **LT <String\$> <PanelId%> <ListBoxId%> <Mode%>**

Description:-

Please Note - This requires BBC_CC17.dll to be installed.

Searches a listbox for a string **<String\$>** and if:

<Mode%> = All text will be removed except any text which **exactly** matches
0 **<String\$>**

<Mode%> = All text will be removed except any strings which **start with** **<String\$>**
1

<Mode%> = All text will be removed except any strings which **contain** **<String\$>**
2

13.17 List Up

From version 3.02.00

Syntax : **LU <Switch%> <PanelId%> <ListBoxId%>**

Description:-

Shows (**<Switch%>** = 1) or hides (**<Switch%>** = 0) the contents of a combo listbox pointed to by **<PanelId%> <ListBoxId%>**

13.18 List Vertical

From version 3.02.00

Syntax : **LV <PanelId%> <ListBoxId%> <Vert%>**

Description:-

Adds a space of size **<Vert%>** in pixels between lines of text in the listbox pointed to by **<PanelId%> <ListBoxId%>**.

13.19 List Width

From version 3.02.00

Syntax : **LW <PanelId%> <ListBoxId%> <Width%>**

Description:-

Sets the gap between columns in a multi-column list box to **<Width%>** pixels.

13.20 List eXtract

From version 3.02.00

Syntax : **LX <PanelId%> <ListBoxId%> <index%> <var\$>**

Description:-

Extracts the string at index position **<index%>** from list box **<ListBoxId%>** on panel **<PanelId%>** and puts the result in **<var\$>**.

14 Memory Commands

14.1 Memory Ascii

From version 3.02.00

Syntax : **MA <Min%> <Max%> <Var\$>**

Description:-

Takes the contents of all memory locations between **<Min%>** and **<Max%>** and produces a space delimited string in **<Var\$>**.

14.2 Memory Decrement

From version 3.02.00

Syntax : **MD <Loc%>**

Description:-

Decrements memory location **<Loc%>** by 1.

14.3 Memory Find

From version 3.02.00

Syntax : **MF** <Find%> <Min%> <Max%> <Var%>

Description:-

Searches for the first instance of integer <Find%> between memory locations <Min%> and <Max%>. If found the memory location number is placed in integer variable <Var%>.

i.e. MF 10 1 3000 A%

14.4 Memory Get

From version 3.02.00

Syntax : **MG** <Number%> <Var%>

Description:-

Gets the contents of memory <Number%> and puts it into <Var%>.

14.5 Memory Increment

From version 3.02.00

Syntax : **MI** <Loc%>

Description:-

Increments memory location <Loc%> by 1.

14.6 Memory Logic

From version 3.02.00

Syntax : **ML** <Function\$> <Min%> <Max%> <Var%>

Description:-

Perform logic function **<Function\$>** on a range of memory locations between **<Min%>** and **<Max%>** and returns the result in **<Var%>**. **<Function\$>** can be **AND, NAND, OR** or **NOR, MIN** or **MAX**.

i.e. ML NAND 11 20 A%

This will perform a 10 input logical **NAND** on the contents of memory locations 11 to 20 inclusive and put the result in **A%**

14.7 Memory Move

From version 3.02.00

Syntax : **MM <Min%> <Max%> <NewStart%>**

Description:-

Takes the contents of all memory locations between **<Min%>** and **<Max%>** and places them incrementally starting at **<NewStart%>**

14.8 Memory Put

From version 3.02.00

Syntax : **MP <Var%> <Number%>**

Description:-

Gets the contents **<Var%>** of and puts it into memory **<Number%>**.

14.9 Memory Read

From version 3.02.00

Syntax : **MR <Filepath+name> <Min%> <Max%>**

Description:-

Retrieves integer memory locations between **<Min%>** and **<Max%>** from the file **<Filepath+name>**

Subject to the following:

- Min% need not be the same as the saved location, so you can load a range of previously saved memory locations back into any location.
- If the range **Max%-Min%** is less than the saved range for the given filename, only the range is loaded.
- If the range exceeds the stored range, the spare locations are set to zero.
- Hence you only alter the specified range of locations when executing the command.

Example:

MR C:/CTRLSYS/mem.DAT 100 123 will read file mem.DAT into memory locations 100 thru 123, constraints as above

Note that forward slashes and semicolons should be used instead of backslash and colon respectively.

14.10 Memory Set

From version 3.02.00

Syntax : **MS <Var%> <Min%> <Max%>**

Description:-

Sets the contents of all memory locations between **<Min%>** and **<Max%>** to **<Var%>**

14.11 Memory Write

From version 3.02.00

Syntax : **MW <Filepath+name> <Min%> <Max%>**

Description:-

Stores integer memory locations between **<Min%>** and **<Max%>** in the file pointed to by **<Filepath+name>**. If the file doesn't exist it is created, else it is overwritten.

Example:

MW C:/CTRLSYS/mem.DAT 100 123 will store integer memory locations between 100 and 123 in a file called mem.DAT in the C:\CTRLSYS\ directory.

Note that forward slashes and semicolons should be used instead of backslash and colon respectively.

15 Number Commands

Regarding numeric ranges:

The limits of numeric conversions are 32 bit integer signed arithmetic. The values are 2147483647 (positive) and -2147483648 (negative).

Therefore the limits for the NL command are:

```
:NL 2147483647 L%  
:NL -2147483648 L%
```

15.1 Number Add

From version 3.02.00

Syntax : **NA** <Var1%> <Var2%> <Var3%>

Description:-

Adds <Var1%> to <Var2%> and places the result in <Var3%>

15.2 Number BIT

From version 3.07.02

Syntax : **NB** <Value1%> <Operator\$> <Value2%> <Result%>

Description:-

Facility for setting, clearing and performing various bit-wise operations on numeric variables.

Supported operator keywords: And, Or, Not, Sr, Sl, Xor, Set, Clr and Test

Operators:

Supported operators keywords are **And**, **Or**, **Not**, **Sr**, **Sl**, **Xor**, **Set**, **Clr** and **Test** and are case insensitive, i.e. **AND** can be written as **AND**, **And**, etc.

Additionally, the "C" language standard operators are supported and may be substituted as follows:

- **'AND' '&'** (bitwise AND)
- **'OR' '|'** (bitwise OR)
- **'XOR' '^'** (bitwise exclusive OR operator)
- **'NOT' '~'** (bitwise complement)
- **'SR' '>>'** (bitwise Shift right)
- **'SL' '<<'** (bitwise Shift left)

Example: An AND operation:

NB A% 'AND' B% C%

or

NB A% '&' B% C%

Additionally, there are bit operators for: set, clear and test:

- **'SET'** (bitwise SET bit operator)
- **'CLR'** (bitwise CLEAR bit operator)
- **'Test'** (bitwise test bit operator)

Range of Values:

The values to be operated upon are 32 bits unsigned integer, have a value range of 0x0 to 0xFFFFFFFF (hexadecimal).

In ApplCore, values are indicated in the value registers and when written to a dialog as "signed integers".

Values greater than 0x80000000 (hexadecimal) are displayed as negative integers.

They are displayed as signed integer ranges of 0 to 2147483647 (decimal) i.e. 0 to 0x7FFFFFFF (hexadecimal) for positive values and -2147483648 to -1 (decimal) for negative values 0x80000000 to 0xFFFFFFFF (hexadecimal).

Operations with signed and unsigned values can cause programmer confusion. The results of the bit operations are displayed as signed integers though the values operated on are unsigned.

Parameters which specify a bit number have a range of values of 0 to 31. Referring to the parameter to be operated on:

- Bit 31 is the most significant bit (MSB).
- Bit 0 is the least significant bit (LSB).

Detail NB Command Descriptions:

NB 'And' (bitwise AND)

Alias NB "&"

NB <Value1%> <Operator\$> <Value2%> <Result%>

NB 55 'AND' 909 C% The result (C%) is value of 5

NB 'Or'(bitwise OR)

Alias NB "|"

NB <Value1%> <Operator\$> <Value2%> <Result%>

NB 55 'OR' 909 C% The result (C%) is value of 959

NB 'Not' (bitwise complement)

Alias NB "~"

NB <Value1%> <Operator\$> <WordSize%> <Result%>

NB 55 'NOT' 4 C% The result (C%) is value of 201

The <WordSize%> parameter specifies the NOT operator to operate on less than the full 32 bit word. The default is 32 (also 0) specify the full word width of 32 bits. A value of 2 would specify that the NOT operator operate on the lower (least significant) two bits in the word (bits 0 and 1).

Notes:

NOT 0 is indicated as -1 (0xFFFFFFFF)

NOT 1 is indicated as -2 (0xFFFFFEE)

Example:

NB 5 'NOT' 4 C% The result (C%) is a value of 10

NB 5 'NOT' 3 C% The result (C%) is a value of 2

NB 5 'NOT' 32 C% The result (C%) is a value of -6

NB 5 'NOT' 0 C% The result (C%) is a value of -6

The operator NOT applied to 32 bits:

NOT 0 is indicated as -1 (corresponding to a HEX value of 0xFFFFFFFF).

NOT 1 is indicated as -2 (corresponding to a HEX value of 0xFFFFFEE).

NB 'SR'(bitwise Shift Right)

Alias NB ">>"

NB <Value1%> <Operator\$> <ShiftCount%> <Result%>

The <ShiftCount%> parameter supports a range of values of 0 to 31.

NB 55 'SR' 2 C% The result (C%) is a value of 13.

NB 'SL'(bitwise Shift Left)

Alias NB "<<"

NB <Value1%> <Operator\$> <ShiftCount%> <Result%>

The <ShiftCount%> parameter supports a range of values of 0 to 31.

NB 55 'SL' 3 C% The result (C%) is a value of 220.

NB 'XOR' (bitwise eXclusive OR operator)

Alias NB "^"

NB <Value1%> <Operator\$> <Value2%> <Result%>

NB 55 'XOR' 909 C% The result (C%) is a value of 109.

NB 'SET' (bitwise SET bit operator)

NB <Value1%> <Operator\$> <bitToSet%> <Result%>

The <bitToSet%> parameter supports a range of values of 0 to 31.

NL 10 C%

NB C% 'SET' 0 C% The result (C%) is a value of 11.

NB 'CLR' (bitwise CLear bit operator)

NB <Value1%> <Operator\$> <bitToClear%> <Result%>

The <bitToClear%> parameter supports a range of values of 0 to 31.

NL 11 C%

NB C% 'CLR' 0 C% The result (C%) is a value of 10.

NB 'TEST' (bitwise bit TEST operator)

NB <Value1%> <Operator\$> <bitToTest%> <Result%>

The <bitToTest%> parameter supports a range of values of 0 to 31.

<Result%> contains 1 if bit set and zero if bit not set.

NL 10 C%

NB C% 'TEST' 0 C% The result (C%) is a value of 0.

15.3 Number Convert

From version 3.02.00

Syntax : **NC <Var%> <Var2\$> or NC <Var\$> <Var%>**

Description:-

Converts string variables to integer variables and vice versa

NC A% B\$ will place a string representation of the integer variable in **A%** into string variable **B\$**. Conversely, **NC A\$ B%** will take the numerical value of **A\$** and place it into integer variable **B%**

Note: Both parameters must be variables, not literals.

From version 3.07.02

Support for hexadecimal notation numeric values conversion added. Hexadecimal notation numeric values are signified by a prefix of "0x". Example:

"TL '0x1002' A\$:NC A\$ A%" Result: 4098 (decimal) in A%

15.4 Number Divide

From version 3.02.00

Syntax : **ND <Var1%> <Var2%> <QuotientVar3%> <RemainderVar4%>**

Description:-

Divides integers **<Var1%>** by **<Var2%>** and puts the quotient in **<QuotientVar3%>** and the remainder in **<RemainderVar4%>**

i.e. ND 20 8 A% B%

This will result in A% containing 2 and B% containing 4.

15.5 Number Format

From version 3.02.00

Syntax : **NF** <Num%> <Format\$> <Pad\$> <Var\$>

Description:-

Inserts the number supplied in <Num%> using the string supplied in <Format\$> which contains "#" as placeholder characters indicating the position into which the text should be inserted and <Pad\$> will replace the remaining characters. The resulting string is returned in <Var\$>. For example :-

NF 12 'DEV_###.INI' '0' I\$ will result in I\$ containing 'DEV_012.INI'

NF 1420 'SPKG_####.INI' '0' I\$ will result in I\$ containing 'SPKG_1420.INI'

15.6 Number Get

From version 3.02.00

Syntax : **NG** <Panel%> <Control%> <Var%>

Description:-

Gets the contents of <Control%> on <Panel%> and places its integer value <Var%>

15.7 Number Load

From version 3.02.00

Syntax : **NL** <Number%> <Variable%>

Description:-

Puts a numeric value into a numeric variable. <Variable%> can be between A% and Z%. <Number%> can also be a variable in which case it would copy one variable to another.

15.8 Number Multiply

From version 3.02.00

Syntax : **NM** <Var1%> <Var2%> <Var3%>

Description:-

Multiplies integers **<Var1%>** by **<Var2%>** and puts the result in **<Var3%>**

15.9 Number Put

From version 3.02.00

Syntax : **NP <Var%> <PanelId%> <ControlId%>**

Description:-

Puts a numeric value into a control.

15.10 Number Random

From version 3.02.00

Syntax : **NR <Min%> <Max%> <Variable % or \$>**

Description:-

Generates a random number between **<Min>** and **<Max>** and places it a variable. The variable can be either an integer or string.

15.11 Number Subtract

From version 3.02.00

Syntax : **NS <Var1%> <Var2%> <Var3%>**

Description:-

Subtracts **<Var2%>** from **<Var1%>** and places the result in **<Var3%>**

15.12 Number Transfer

From version 3.02.00

Syntax : **NT <Var1%> <Var2%>**

Description:-

Copies <Var1%> to <Var2%>. <Var1%> remains the same.

16 Panel Commands

16.1 Panel De-Initialise

From version 3.02.00

Syntax : **PD** <PanelId%>

Description:-

Similar to PR (Panel Remove) except that the panel is destroyed and information can no longer be sent to it. Panels should always be de-initialised when you no longer require them to be displayed unless they require updating constantly.

16.2 Panel Initialise

From version 3.02.00

Syntax : **PI** <PanelId%>

Description:-

Initialises a panel without showing it on the screen. Initialising a panel means that it can be updated by the control system in its hidden state. Panel No. is an integer 1...64.

16.3 Panel Move

From version 3.02.00

Syntax : **PM** <PanelId%> <NewXPos%> <NewYPos%>

Description:-

Moves panel **<PanelId%>** to a new location on the screen so that the top left hand corners of the panel are at **<NewXPos%>** **<NewYPos%>**. If dialog% is 0, it moves the backpanel rather than a dialog.

16.4 Panel Remove

From version 3.02.00

Syntax : **PR <PanelId%>**

Description:-

Removes panel **<PanelId%>**. **<PanelId%>** is an integer 1 and 64.

16.5 Panel Show

From version 3.02.00

Syntax : **PS <PanelId%>**

Description:-

Shows panel **<PanelId%>**. **<PanelId%>** is an integer 1 and 64.

16.6 Panel View

From version 3.02.00

Syntax : **PV <Switch%> <PanelId%> <Icon\$>**

Description:-

If **<Switch%>** is '0' the panel **<PanelId%>** will be shown minimised at the bottom of the screen. If the **<Icon\$>** parameter is supplied then the minimised panel will display any icon with that name found in the applications resources. If the panel dialog box already has a name then this will be overwritten by **<Icon\$>**. If **<Icon\$>** is not supplied the minimised panel takes its name from the dialog box name.

If **<Switch%>** is '1' the panel **<PanelId%>** will be shown normal size.

16.7 Panel size

From version 3.05.07

Syntax : **PZ** <PanelId%> <cx%> <cy%>

Description:-

Sizes dialog <PanelId%> to size <cx%>, <cy%>. If <PanelId%> is 0, it sizes the backpanel rather than a dialog.

17 Query Commands

17.1 Child

From version 3.04.00

Syntax : **QC** <dialog%> <control%> <var%>

Description:-

Gets the window handle of control <control%> on <dialogue%> into <var%>

17.2 Query Desktop

From version 3.04.00

Syntax : **QD** <width%> <height%>

Description:-

Gets the desktop screen size into the specified variables <width%> <height%>

17.3 Queue Flush

From version 3.08.05

Syntax : **QF**

Description:-

The QF command directs CSI32 "send queued commands/revertives now".

17.4 Query Handle

From version 3.04.00

Syntax : **QH <PanelId%> <var%>**

Description:-

Gets the window handle of dialogue **<PanelId%>** into **<var%>** if **< PanelId %>** is 0, it gets the window handle of the backpanel.

17.5 Query Parent

From version 3.04.00

Syntax : **QP <PanelHandle%> <var%>**

Description:-

Gets the window handle of the parent of window handle **<PanelHandle%>** into **<var%>**

17.6 Query System

From version 3.08.08

Syntax : **QS <Selector\$> [<ReturnVars%>] [<ReturnVars\$>]**

Description:-

Gets the characteristics of the running ApplCore environment. The following **<Selector\$>** are implemented: 'D', 'K', 'M', 'P', 'T' and 'V'

Get DATE - QS 'D' <DateTimeRetVar\$>

Get Keyboard ID - QS 'K' <KbdIdRetVar%>

Get Number of Memories - QS 'M' <InteMemCountRetVar%> <StrMemCountRetVar%>

Get Panel Information - QS 'P' <PanelId%> <XRetVar%> <YRetVar%>

Get System MS time - QS 'T' <RetVarMillisecondTime%>

Get Version - QS 'V' <VersionRetVar\$>

18 Router Commands

18.1 Router Crosspoint

From version 3.02.00

Syntax : **RC** <Driver%> <Source%> <Destination%> <Mask\$> <Switch%>

Description:-

Instructs a router driver <Driver%> to route <Source%> to <Destination%>. The <Mask\$> and <Switch%> are optional.

If <Switch%> is 0 the command is buffered along with any other commands waiting to be sent until CSI's next regular transmit cycle. If <Switch%> is 1 the command is sent immediately along with any buffered commands already waiting.

<Mask\$> is used to define *how* the route is to be made. This is often used where the router is not a physical device but is the presentation of a multi-level or complex device such as a packager/combiner or Virtual Router Driver (VRD). Use of this parameter is specific to these applications.

18.2 Router Database

From version 3.02.00

Syntax : **RD** <DriverVar%> <Source/DestFlag%> <Index%> <MappedId%>

Description:-

When a database change occurs as a result of the '**Router Modify**' command the router driver will validate the source or destination name that has changed. If valid the change will be signalled to all instances of CSI in the system. Each CSI will save the change in the local database and signal all panel applications that a change has occurred by executing the DATABASE line (32000) of the stringtable.

Immediately after executing this command <DriverVar%> will hold the driver number whose database has changed. <Source/DestFlag%> will be '0' if the change is a source and '1' for a destination. <Index%> will hold the index value for the source or destination that has changed. <MappedId%> will normally contain the same Id as <DriverVar%>. However, if the DEV file for <DriverVar%> shows that the level being notified is mapped to another database Id then <MappedId%> will the contain the redirected Id.

Please Note: All 4 variables must be supplied even if they are not required in the panel.

The DATABASE line is executed once for every single database change.

An example of **<MappedId%>**, lets say that **RD A% B% C% D%** returns **100 4 15 200** for the four variables. This means that in the **[Database]** section of the **DEV_100.INI** file the **DatabaseFile_4** entry is pointing to the database referred to in **DEV_200.INI**

18.3 Router File

From version 3.02.00

Syntax : **RF <Driver%>**

Description:-

Requests **<Driver%>** to reload its database file.

This command is superseded by the functionality of the dynamic database mechanism provided using the Router Modify and Router Database commands.

18.4 Router Index

From version 3.02.00

Syntax : **RI <Driver%> <DataBaseNumber%> <DataBaseName\$> <Var%>**

Description:-

Returns the Index for a source or destination on **<Driver%>**. If the name given **<DataBaseName\$>** is a source then the **<DataBaseNumber%>** must be 0 and for a destination it must be 1. **<Var%>** is the variable in which the index value is to be placed.

From version 3.07.02

Starting with version 3.07.02 **<DataBaseNumber%>** has expanded range access to all data bases (DB's) is supported, i.e. 0-9. Version 3.07.02 ApplCore requires support from CSI32 to access DB's 0-9 and in file: CSI.ini section, entry: UseMemoryDatabaseOnly(no_V1_support) must be set to 1.

File: CSI.ini

Section: [Database]

Entry: UseMemoryDatabaseOnly(no_V1_support)=1

Note : This command cannot be used on databases 2 through 9 on older versions of ApplCore.

18.5 Router Lock

From version 3.02.00

Syntax : **RL** <Driver%> <Min%> <Max%> <Var%>

Description:-

If <Var%> is 1 then all destinations on <Driver%> between <Min%> and <Max%> are locked. A destination can not be changed when it is locked. Use <Var%> set to 0 to unlock a destination or range of destinations.

The Router Lock command cannot be intercepted by the Applcore eXternal Commands and is always passed to the driver for processing as normal.

18.6 Router Modify

From version 3.02.00

Syntax : **RM** <Driver%> <Database%> <Index%> <Name\$> <Flag%>

Description:-

Instructs router driver <Driver%> to change its database name <Database%> 0-9. <Index%> is the index of the source or destination name that is changing and <Name\$> is the new name.

If <Flag%> is 1 it signals to the router driver to send a complete tally table dump in order to update any revertive source names that may have changed. If several database changes are being made this flag should be set to '0' and only be set to '1' for the very last command. If only destination names are changing then no revertive update is needed.

Some drivers support a value of 2 for the <Flag%> parameter. Changing a source name (database 0 or database 2) results in a dump of tallies solely for those destinations which have that source selected rather than a complete tally dump.

18.7 Router Name

From version 3.02.00

Syntax : **RN <Driver%> <DataBaseNumber%> <Index%> <Var\$>**

Description:-

Returns the name of a given database **<Index%>** on driver number **<Driver%>**. The **<DataBaseNumber%>** can be 0 to 9. If it is 0 then it will search for a source name, if 1 a destination name, if 2 a database 2 name etc. If a name is found it placed in **<Var\$>**.

The range of **<Index%>** is -1 to 4096 (65535 for LAWO).

Support of the LAWO router requires addressable indices beyond 4096. The RN command has an index limit of 65535.

18.8 Router Poll

From version 3.02.00

Syntax : **RP <Driver%> <Min%> <Max%>**

Description:-

Polls **<Driver%>** for the status of all destinations between **<Min%>** and **<Max%>**. The information will be delivered into the stringtable offset that was set up when the the Router Registration command was executed. **S%** will contain the source number and **S\$** will contain the name of the source as defined in the INI file for that router driver. All workstations receive the data returned so this method should be used sparingly on small ranges of data. Use Router Query for larger ranges.

18.9 Router Query

From version 3.02.00

Syntax : **RQ <Driver%> <Min%> <Max%>**

Description:-

Polls **<Driver%>** for the status of all destinations between **<Min%>** and **<Max%>**. The information will be delivered into the stringtable offset that was set up when the the Router Registration command was executed. **S%** will contain the source number and **S\$** will contain the name of the source as defined in the INI file for that router driver. The data

transfer is carried out on a private session between the driver and the workstation requesting the data.

18.10 Router Register

From version 3.06.01

Syntax : **RR <Driver%> <Min%> <Max%> <Offset%> <Offset Mode%> [<DestVar%>] [<'ADD'>] [<DriverVar%>]**

Description:-

Registers with router driver **<Driver%>**. Revertive information will be sent to your application every time the status of a destination between **<Min%>** and **<Max%>** changes. The source index will be placed in S% and stringtable offset **<Stringtable Offset%>** executed if **<OffsetMode%>** is 0. If **<OffsetMode%>** is 1 then **<Stringtable Offset%>** is added to the input or output index and the resulting stringtable line is executed.

If the optional parameter variable **<DestVar%>** is included then whenever a revertive occurs this variable will contain the index value of the destination. If the optional word **'ADD'** is appended as a 7th parameter then the registration is added to any existing registration for the device. If the 'ADD' is omitted then any previous registration is overwritten.

An example of using the **'ADD'** feature should look something like this:

RR A% 1 16 21000 0 r%:RR A% 53 54 21000 0 r% 'ADD'

RR for device **'A%'** destinations 1 to 16 calling line 21000 'statically' when a revertive is received, plus RR for device **'A%'** destinations 53 to 54 also calling line 21000 'statically'. **'r%'** will contain the destination number and could be used to filter an individual revertive using the appropriate code on line 21000.

If the optional **<DriverVar%>** variable was specified at registration time, the variable is filled with the value of **<Driver%>** then the stringtable line is called.

18.11 Router Unregister

From version 3.02.00

Syntax : **RU <Driver%> (<Min%> <Max%>)**

Description:-

Following this command no further status updates will be returned to the application for the **<Driver%>**. Selective deregistration of individual or sub-ranges of destinations is possible if the **<Min%>** **<Max%>** are added to the command.

If **<Driver%>** is 0, then ALL devices are unregistered for this panel.

19 String Commands

19.1 String AppName

From version 3.02.00

Syntax : **SA <Var\$>**

Description:-

Puts the name of the current ApplCore application into **<Var\$>**

19.2 String Compose

From version 3.03.11

Syntax : **SC <Output\$> <Format\$> [<Var\$> or <Var%> ...]**

Description:-

This command allows strings to be formatted using a syntax similar to the format specifier in C. **<Output\$>** is the target string variable to hold the string to be formatted.

<Format\$> is a C sprintf() compatible string. **[<Var\$> or <Var%> ...]** are a variable number of string and/or numeric constants/variables and are optional.

Note the format string may use almost any string/numeric format specifier but %c DOES NOT work

19.3 String Date

From version 3.02.00

Syntax : **SD <Start Date\$> <Days Offset%> <New Date\$> <New Day%> <New Month%> <New Year%>**

Description:-

Calculates a new date based upon a start date placed in **<Start Date\$>** and a plus or minus offset placed in **<Days Offset%>**. The new date calculated is placed in **<New Date\$>** and the day, month and year integers placed in **<New Day%>**, **<New Month%>** and **<New Year%>** respectively. Dates are calculated with reference to the number of days that have elapsed since the first of January 1900 taking into account all the leap years. **<Start Date\$>** must be of the form 14-Apr-1999. This is the format returned by ApplCore's current date function delivered by timer 0 into **U\$**. **<New Date\$>** is returned in the same format.

19.4 String Environment

From version 3.02.00

Syntax : **SE <Search\$> <Variables\$>**

Description:-

This command allows you to retrieve the Environment Variable **<Search\$>** from your local machine and place it in string **<Variable\$>**.

19.5 String Find

From version 3.02.00

Syntax : **SF <Find\$> <Min%> <Max%> <Var%>**

Description:-

Searches for the first instance of string **<Find\$>** between memory locations **<Min%>** and **<Max%>**. If found the memory location number is placed in integer variable **<Var%>**.

i.e. SF 'hello there' 1 3000 A%

19.6 String Get

From version 3.02.00

Syntax : **SG <Number%> <Var\$>**

Description:-

Gets the contents of memory **<Number%>** and copies it into **<Var\$>**.

19.7 String Hex

From version 3.02.00

Syntax : **SH <Number%> <Target\$> <Zeros%> <CaseSwitch%>**

Description:-

Converts decimal **<Number%>** into a string **<Target\$>**. **<Zeros%>** specifies the number of leading zeros. If **<CaseSwitch%>** is zero then the string will be lower case, otherwise if it is a '1' then the string will be converted to upper case.

19.8 String Interval

From version 3.05.06

Syntax : **SI <NewDate\$> <NewTime\$> <OldDate\$> <OldTime\$> <Result\$>**

Description:-

This command measures the elapsed interval of time between **<NewDate\$>** **<NewTime\$>** and **<OldDate\$>** **<OldTime\$>** and puts the result in the form of HH:MM:SS in **<Result\$>**. The dates and times are in the same format as those provided by ApplCore timer 0 in **T\$** and **U\$**

19.9 String List

From version 3.02.00

Syntax : **SL <Min%> <Max%> <Delimiter\$> <Result\$> <Switch%>**

Description:-

Gets the contents of either integer or string memory locations between **<Min%>** and **<Max%>** and creates a delimited list, using first character of **<Delimiter\$>** as the delimiter. The result is put in **<Result\$>**. If **<Switch%>** is zero then integer memories are used otherwise string memories are used. A NULL delimiter can be used, just use the single quotes with nothing between them.

19.10 String Message

From version 3.02.00

Syntax : **SM <Var\$>**

Description:-

The string **<Var\$>** is sent to the network as a verbatim universal revertive message. CSI's in the system will ignore it, but CAPLOG.EXE will pick it up and store it in the log files. This command is useful for keeping track of panel usage, errors and events.

19.11 String NameApp

From version 3.02.00

Syntax : **SN <NewName\$>**

Description:-

Renames the application (panel) to be **<NewName\$>.exe**. This is very useful if you want to run multiple instances of the same panel. You could supply the new name via a command line parameter. The panel will initially start up with its standard name and then take its new name when it processes the **SN <NewName\$>** command.

Example:-

If you have the command **SN S\$** in your Startup stringtable and start the panel with a command line parameter eg **'MyPanel.exe NewName'** your panel will initially run up as **'MyPanel.exe'** until the **SN S\$** command is executed at which point you panel will be named **'NewName.exe'**.

19.12 String Put

From version 3.02.00

Syntax : **SP <Var\$> <Number%>**

Description:-

Gets the contents **<Var\$>** of and copies it into memory **<Number%>**.

19.13 String Query Application Name

From version 3.07.02

Syntax : **SQ** <FileName\$> <AdditionalNameCharacters\$> <Result%>

Description:-

Queries for the existence of the named application program consisting of the <FileName\$> concatenated to <AdditionalNameCharacters\$>.

If the application is running SQ returns 1, if not running SQ returns 0.

Example: Given a file name root name of NTHDEST, SQ enables easy checking to see if these application names are running:

```
NTHDEST 1|A
NTHDEST 2\A
```

19.14 String Read

From version 3.02.00

Syntax : **SR** <Filepath+name\$> <Min%> <Max%>

Description:-

Retrieves lines of strings from the file <Filepath+name> and places these in memory locations between <Min%> and <Max%>

Subject to the following:

- **Min%** need not be the same as the saved location, so you can load a range of previously saved memory locations back into any location.
- If the range **Max%-Min%** is less than the saved range for the given filename, only the range is loaded.
- If the range exceeds the stored range, the spare locations are set to NULL.
- Hence you only alter the specified range of locations when executing the command.

Example:

SR C:/CTRLSYS/mem.TXT 100 123 will read file mem.TXT line by line into memory locations 100 thru 123, constraints as above

Note that forward slashes and semicolons should be used instead of back slash and colon respectively.

19.15 String Set

From version 3.02.00

Syntax : **SS** <Var\$> <Min%> <Max%>

Description:-

Copies the contents of <Var\$> to all memory locations between <Min%> and <Max%>.

19.16 String Timecode

From version 3.02.00

Syntax : **ST** <Switch%> <Var1\$> <Var2\$>

Description:-

This command applies various conversion to strings in TimeCode format. The actual function performed is determined by the <Switch%>.

<Switch%>	Function
0	Converts the number of fields contained in <Var1\$> into a Timecode string in <Var2\$>
1	Converts the Timecode string contained in <Var1\$> into fields in <Var2\$>.
2	The number of fields in <Var1\$> is added to the TimeCode string in <Var2\$> and the result is placed in <Var2\$>. Adding 1 to 23.59.59.24 will wrap the clock back to 00.00.00.00.
3	The number of fields in <Var1\$> is subtracted from the TimeCode string in <Var2\$> and the result is placed in <Var2\$>. Subtracting 1 from 00.00.00.00 will wrap the clock back to 23.59.59.24
4	The current system time is placed in <Var1\$>. <Var2\$> is ignored.

Note 1: do not use the ST command to convert the system time variable **T\$** directly to a time field as the format of **T\$** is not compatible with a timecode string. Instead use '**ST 4** <Var1\$> <Var2\$>' which will convert the system time to time code format and place it in <Var1\$>.

19.17 String Write

From version 3.02.00

Syntax : **SW <Filepath+name\$> <Min%> <Max%>**

Description:-

Stores strings from memory locations between **<Min%>** and **<Max%>** in the file pointed to by **<Filepath+name\$>**. If the file doesn't exist it is created, else it is overwritten.

Example:

SW C:/CTRLSYS/mem.TXT 100 123 will store strings from memory locations between 100 and 123 in a file called mem.TXT in the C:\CTRLSYS\ directory.

Note that forward slashes and semicolons should be used instead of backslash and colon respectively.

20 Text Commands

20.1 Text Break

From version 3.02.12

Syntax : **TB <String\$> <NumCharacters%> <TargetString\$>**

Description:-

Takes the string **<String\$>** and adds the line break character '|' (pipe) after **<NumCharacters%>** number of characters and places the resulting string into **<TargetString\$>**

20.2 Text Copy

From version 3.02.00

Syntax : **TC <Panel1%> <Id1%> <Panel2%> <Id2%>**

Description:-

Copies the text from **<Panel1%> <Id1%>** and places it in **<Panel2%> <Id2%>** leaving the source text unchanged.

20.3 Text Divide

From version 3.02.00

Syntax : **TD <String\$> <Delimiter\$> <Part1\$> <Part2\$>**

Description:-

Divides a string into two at the point where the delimiter occurs. The delimiter should be a single character. After the command is executed **<Part1\$>** contains the string up to the delimiter and **<Part2\$>** contains the remainder. The delimiter is discarded.

20.4 Text Edit

From version 3.02.00

Syntax : **TE <Command\$> <Var\$> <Options%>**

Description:-

The Text Edit commands permit manipulation of text via functions similar to those found in text editors. All commands modify the supplied string **<Var\$>**. The **<Command\$>** may be one of the following and the **<Option%>** parameter is dependent upon this command.

<Command\$>	<Option%>
BACKSPACE	Characters specified in <Option%> are removed from the end of <Var\$> .
DELETE	Characters specified in <Option%> are removed from the beginning of <Var\$> .
SLASH	1 = a forward slash is inserted at the start of the string. 0 = any forward slash at the start of the string is removed.
CASE	0 = the string is converted to lower case. 1 = the string is converted to UPPER

CASE
2 = the string is converted to Title
Case (i.e. first convert to lowercase
then make first letter of each word
capital)

20.5 Text Filter

From version 3.02.00

Syntax : **TF** <String\$>

Description:-

Removes all carriage returns, line feeds and single quotation marks from <String\$>.

20.6 Text Get

From version 3.02.00

Syntax : **TG** <Panel1%> <Control ID%> <Var\$>

Description:-

Collects the text from <Panel%> <Control ID%> and places this text in a <Var\$>.

20.7 Text Horizontal Tab

From version 3.02.00

Syntax : **TH** <String\$> <Delimiter> <Switch%>

Description:-

If <Switch\$> is 0 then the function scans the string <String\$> and replaces any occurrence of the first character in the <Delimiters\$> string with a Tab character (0x09). If <Switch\$> is 1 then the any tabs are replaced with the first character in the <Delimiters\$> string.

20.8 Text Insert

From version 3.02.00

Syntax : **TI** <Var1\$> <Var2\$>

Description:-

Adds <Var1\$> to the end of <Var2\$> and places the result in <Var2\$>.

20.9 Text Load

From version 3.02.00

Syntax : **TL** <Text\$> <Var\$>

Description:-

Loads text into a string variable.

20.10 Text Move

From version 3.02.00

Syntax : **TM** <Panel1%> <Control ID1%> <Panel2%> <Control ID2%>

Description:-

Copies the text from <Panel1%> <Control ID1%> and places it in <Panel2%> <Control ID2%> and then deletes the text in <Panel1%> <Id1%>.

20.11 Text Put

From version 3.02.00

The Text Put command has two uses:

1. It is used to write text to a Control ID
2. It is used to set the Custom Control Attributes for a Control ID.

20.11. 1 Adding text to a Control ID

Syntax : **TP <Var\$> <Panel%> <Control ID%>**

Description:-

Collects the text from a local variable **<Var\$>** and places this text in **<Panel%> <Control ID%>**.

20.11. 2 Changing the Custom Control Attributes for a Control ID

Syntax : **TP '/ABCDEF' <Panel%> <Control ID%>**

This syntax of the Text Put Command is used to set the Custom Control Attributes of Controls such as the button colour and text colour. For a full list of attributes see the [Custom Control Attributes](#) document (cc_attris.pdf).

Positions A-E are used to changes Attributes within a Control and position F is used to change the Text Block Switch for that Control ID. Some more complicated Controls, such as a Destination Buttons or Fader Controls, have a number of Sub-ID's. This is analogous to a house being split into a number of flats and each flat having it's own sub-address within the house. Position F is used to give this sub-address or Sub-ID.

Please note: once the Text Block Switch has been changed then **all** subsequent commands sent to that Control will go to the new Sub-ID until another Switch Block is sent. The default Switch Block is 0.

20.12 Text Quantity

From version 3.02.00

Syntax : **TQ <Var\$> <Varl%>**

Description:-

Returns the length of string **<Var\$>** in user variable **<Var%>**

20.13 Text Remove

From version 3.02.00

Syntax : **TR <Var1\$> <Var2\$>**

Description:-

Removes the first occurrence of **<Var1\$>** from **<Var2\$>** and places the result in **<Var2\$>**.

20.14 Text Split

From version 3.02.00

Syntax : **TS** <Var\$> <Delimiters\$> <IntMemLoc%> (<StringMemLoc%>)

Description:-

Splits up a delimited list of integer variables contained in <Var\$> using <Delimiters\$> and puts the numbers into memory locations starting at <IntMemLoc%>+1. The number of integers extracted is put in <IntMemLoc%>. The optional fourth parameter, <StringMemLoc%>, can be used to put the split up string members into memory locations starting at <StringMemLoc%>. Using a zero for either <IntMemLoc%> or <StringMemLoc%> will inhibit that particular split.

From version 3.08.05

If <Var\$> is zero length command returns the number extracted to be 0.

20.15 Text Transfer

From version 3.02.00

Syntax : **TT** <Var1\$> <Var2\$>

Description:-

Copies <Var1\$> to <Var2\$>. <Var1\$> remains the same.

20.16 Text Write

From version 3.02.00

Syntax : **TW** <Var1\$> <Var2\$>

Description:-

Writes the string <Var1\$> to file path <Var2\$>. <Var2\$> can be a full file path and filename. Remember to substitute forward slashes for backward slashes and semi-colons for colons.

20.17 Text eXchange

From version 3.02.00

Syntax : **TX <Var\$> (<Switch%>)**

Description:-

Scans the string **<Var\$>** and replaces any occurrence of certain characters with other characters – see list and rules following.

< Switch %>	Source character	Replaced by character	Comments
0	;	:	Semicolon (ASCII 59.) changed to colon (ASCII 58.)
0	/	\\	One "forward" slash (ASCII 47.) changed to two "back" slashes (ASCII 92.)
0	 	<CR>	Vertical Bar (ASCII 124.) changed to carriage return (ASCII 13.) If environment variable TxVerticalBarCRLF in ApplCore.ini is non-zero then add a <LF> (ASCII 10.) following the <CR>.
0	`	'	Single back quote / Grave Accent (ASCII 96.) changed to single forward quote / Apostrophe (ASCII 39.)
1	:	;	Colon (ASCII 58.) changed to Semicolon (ASCII 59.)
1	\\	/	Two "back" slashes (ASCII 92.) changed to one "forward" slash (ASCII 47.)
1	<CR>	 	Carriage return (ASCII 13.) changed to Vertical Bar (ASCII 124.) If environment variable TxVerticalBarCRLF in ApplCore.ini is non-zero then delete the extra <LF> (ASCII 10.) following the <CR>.
1	'	`	Single "forward" quote (ASCII 39.) changed to single "back" quote (ASCII 96.)

21 UMD Commands

21.1 UMD Enable

From version 3.02.00

Syntax : **UE <Var%>**

Description:-

If **<Var%>** is 1 the UMD driver within ApplCore is enabled. If **<Var%>** is 0 then the driver is disabled.

21.2 UMD Put

From version 3.02.00

21.2. 1 Syntax for Pro-Bel and TSL UMD's

UP <Var\$> <Chain%> <Port%> <CueLamps%> <Brightness%>

Description:-

For Pro-Bel and TSL UMD's the format of the command

Sets UMD on **<Chain%>**, **<Port%>** to display **<Var\$>**. The **<CueLamps%>** variable can be between 0 and 4 and has the following meaning.

<CueLamp%>	Left	Right
0	Off	Off
1	On	Off
2	Off	On
3	On	On
4	See note below	See note below

Note: If **<CueLamp%>** is 4 the cue lamps are controlled by the presence or absence of a '/' character. If the first character in a string is a '/' both cue lamps will be turned on. If the first character is not a '/' then both cue lamps will be turned off.

<Brightness%>	Effect
0	Off
1	1/7
2	1/2
3	Full

Note : **<Brightness%>** only applies to TSL UMD's

21.2. 2 Syntax for Amazon UMD's

Amazon UMDs do not have cue lamps, but they do have tri-colour displays. The last two parameters in the UP command set the colour of the left and right displays respectively. For 8 character displays only the left colour value is used. If using 8 character UMDs remember to set the 'UMDSize' parameter in the ApplCore INI file as the default is 16. The values are as follows:-

UP <Var\$> <Chain%> <Port%> <LeftColour%> <RightColour%>

<LeftColour%> & <RightColour%>	Effect
0	Green
1	Amber
2	Red
3	Red with amber underline

22 Video Commands

22.1 Video Adjust

From version 3.02.00

Syntax : **VA <BoardNum%> <Control\$> <Value%>**

Description:-

Adjusts the video characteristics of the Win-Tv or other MCI compliant board number **<BoardNum%>**. **<Control\$>** is one of the following: BRIGHTNESS, CONTRAST , COLOUR or TINT. This **<Control\$>** parameter is not case sensitive. **<Value%>** must be between 0 and 255.

22.2 Video Channel

From version 3.02.00

Syntax : **VC <BoardNum%> <Channel%>**

Description:-

Changes the UHF channel on the Hauppauge Win-Tv board. **<Channel%>** is between 1 and 69.

22.3 Video Disable

From version 3.02.00

Syntax : **VD <BoardNum%> <ControlId%>**

Description:-

Turns off overlaid video on **<BoardNum%>** using **<ControlId%>**.

22.4 Video Enable

From version 3.02.00

Syntax : **VE <BoardNum%> <ControlId%> <CaptureName\$>**

Description:-

Turns on overlaid video for **<BoardNum%>** using **<ControlId%>**. The VI command must have been used either as a result of a button being pressed or in the STARTUP stringtable. **<CaptureName\$>** specifies the application to be "WinExec'd". ApplCore.ini contains the application path "in variable VideoCaptureApplication" in the "System" section.

Notes: Some video cards demonstrate a start-up problem in initially displaying the overlay image. There is no correction to V3 ApplCore, but there is a work-around, where the VE

command is placed in a start-up timer service. This strategy allows the initial ApplCore start-up to complete and then the video overlay button is enabled.

22.5 Video Fit

From version 3.02.00

Syntax : **VF** <BoardNum%> <PanelNo%> <ControlID%>

Description:-

Places overlaid video from <BoardNum%> onto control correctly scaled.

22.6 Video Initialise

From version 3.02.00

Syntax : **VI** <BoardNum%>

Description:-

Initialises video board <BoardNum%>. This is only needed for the M2C board

22.7 Video Position

From version 3.02.00

Syntax : **VP** command obsolete

22.8 Video Source

From version 3.02.00

Syntax : **VS** <BoardNum%> <Source%>

Description:-

Selects the video input source <Source%> to video board <BoardNum%>. This is only for the Hauppauge Win-Tv board.

22.9 Video Write

From version 3.02.00

Syntax : **VW** **<BoardNum%>** **<Filename\$>** **<FileType%>**

Description:-

This command takes a snapshot of the video being captured by **<BoardNum%>** and stores it in **<Filename\$>** which can contain a path. Remember to substitute colons with semi-colons and backward slashes for forward slashes. The **<FileType%>** determines the format of the file and can be one of the fourteen options listed below. The extension that you supply for the **<Filename\$>** does not affect on the format of the file actually saved. This is because the same extension can be used for variants of a basic file type.

<FileType%>	Description of file format
0	Windows DIB 24 bit true color
1	Windows DIB 8 bit palettized
2	Windows DIB 8 bit gray-scale
3	Windows DIB 4 bit dithered
4	Targa 32 bit true color
5	Targa 24 bit true color
6	Targa 16 bit true color
7	IBM MMotion format 4:1:1 YUV
8	TIFF 24 bit true color
9	TIFF 8 bit palettized
10	TIFF 8 bit gray-scale
11	PCX 8 bit palettized
12	PCX 8 bit gray-scale
13	PCX 4 bit dithered

14	GIF 8 bit palettized
15	GIF 8 bit gray-scale
16	JPEG compressed 4:2:2 YUV
17	BMP monochrome
18	BMP 16 colors palettized

23 Workstation Commands

23.1 Workstation Enable

From version 3.02.00

Syntax : **WE**

Description:-

This command tells CSI to enable its cache.

23.2 Workstation Disable

From version 3.02.00

Syntax : **WD**

Description:-

This command tells CSI to disable its cache.

23.3 Workstation Flush

From version 3.02.00

Syntax : **WF** <**Device%**>

Description:-

This command tells CSI to flush its cache for device <**Device%**>.

23.4 Workstation Test

From version 3.02.00

Syntax : **WT** <**Device%**> <**Min%**> <**Max%**> <**Var%**>

Description:-

This command tests the CSI cache validity for device <**Device%**> for indices between <**Min%**> and <**Max%**>, returning the result in <**Var%**>.

V3 Appendix I – Keyboard Tables

Key tables for modes 0 and 1:

This table complements the tables listed in KR command description. Some information is duplicated in presentation. Please review the notes following the tables.

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
` (Grave Accent) UK: ¬ (logical not) US: ~ (tilde) UK Alt-Gr: (broken bar / split vertical bar)	None	``'	96	96
	Shift (caps lock: off)	UK: `¬' US: " ~"	UK: 172 US:126	UK: 172 US:126
	CapLock: on	``'	96	96
	Shift + Caplock on	UK: `¬' US: " ~"	UK: 172 US:126	UK: 172 US:126
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	' '	166	166
1 ! (exclamation mark)	None	`1'	49	49
	Shift (caps lock: off)	`!'	33	33
	CapLock: on	`1'	49	49
	Shift + Caplock on	`!'	33	33
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	49	49
2 UK: " US: @ (UK: double quotes, US: At sign)	None	`2'	50	50
	Shift (caps lock: off)	UK: `"" US: "@"	UK:34 US:64	UK:34 US:64
	CapLock: on	`2'	50	50
	Shift + Caplock on	UK: `"" US: "@"	UK:34 US:64	UK:34 US:64
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	50	50
3 UK: £ US: # (British Pound Sterling, Number sign / hash)	None	`3'	51	51
	Shift (caps lock: off)	UK: `£' US: "#"	UK: 163 US: 35	UK: 163 US: 35
	CapLock: on	`3'	51	51
	Shift + Caplock on	UK: `£' US: "#"	UK: 163 US: 35	UK: 163 US: 35
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	51	51

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
4 \$ UK Alt Gr: € (Dollar sign, Euro)	None	`4'	52	52
	Shift (caps lock: off)	`\$'	36	36
	CapLock: on	`4'	52	52
	Shift + Caplock on	`\$'	36	36
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	'€'	128	128
5 % (percent sign)	None	`5'	53	53
	Shift (caps lock: off)	`%'	37	37
	CapLock: on	`5'	53	53
	Shift + Caplock on	`%'	37	37
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	53	53
6 ^ (Circumflex accent or Caret)	None	`6'	54	54
	Shift (caps lock: off)	`^'	94	94
	CapLock: on	`6'	54	54
	Shift + Caplock on	`^'	94	94
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	54	54
7 & (Ampersand)	None	`7'	55	55
	Shift (caps lock: off)	`&'	38	38
	CapLock: on	`7'	55	55
	Shift + Caplock on	`&'	38	38
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	55	55

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
8 * (Asterisk)	None	'8'	56	56
	Shift (caps lock: off)	'*'	42	42
	CapLock: on	'8'	56	56
	Shift + Caplock on	'*'	42	42
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	56	56
9 ((open parenthesis / round bracket)	None	'9'	57	57
	Shift (caps lock: off)	'('	40	40
	CapLock: on	'9'	57	57
	Shift + Caplock on	'('	40	40
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	57	57
0) (close parenthesis / round bracket)	None	'0'	48	48
	Shift (caps lock: off)	')'	41	41
	CapLock: on	'0'	48	48
	Shift + Caplock on	')'	41	41
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	48	48
- (minus sign) _ (underscore)	None	'_'	45	45
	Shift (caps lock: off)	'_'	95	95
	CapLock: on	'_'	45	45
	Shift + Caplock on	'_'	95	95
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	45	45
= (equal) + (plus)	None	'='	61	61
	Shift (caps lock: off)	'+'	43	43
	CapLock: on	'='	61	61
	Shift + Caplock on	'+'	43	43
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	61	61

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
a	None	`a`	97	97
	Shift (caps lock: off)	`A`	65	65
	CapLock: on	`A`	65	65
	Shift + Caplock on	`a`	97	97
	Control ⁽²⁾	'A'	1	1
	Alt	"	No-op	No-op
	Alt Gr / Alt Gr Shift	`á` / `Á`	225 / 193	225 / 193
b	None	`b`	98	98
	Shift (caps lock: off)	`B`	66	66
	CapLock: on	`B`	66	66
	Shift + Caplock on	`b`	98	98
	Control ⁽²⁾	'B'	2	2
	Alt	"	No-op	No-op
	Alt Gr	"	98	98
c	None	`c`	99	99
	Shift (caps lock: off)	`C`	67	67
	CapLock: on	`C`	67	67
	Shift + Caplock on	`c`	99	99
	Control ⁽²⁾	'C'	3	3
	Alt	"	No-op	No-op
	Alt Gr	"	99	99
d	None	`d`	100	100
	Shift (caps lock: off)	`D`	68	68
	CapLock: on	`D`	68	68
	Shift + Caplock on	`d`	100	100
	Control ⁽²⁾	'D'	4	4
	Alt	"	No-op	No-op
	Alt Gr	"	100	100
e	None	`e`	101	101
	Shift (caps lock: off)	`E`	69	69
	CapLock: on	`E`	69	69
	Shift + Caplock on	`e`	101	101
	Control ⁽²⁾	'E'	5	5
	Alt	"	No-op	No-op
	AltGr/AltGrShift	`é` / `É`	233 / 201	233 / 201

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
f	None	`f`	102	102
	Shift (caps lock: off)	`F`	70	70
	CapLock: on	`F`	70	70
	Shift + Caplock on	`f`	102	102
	Control ⁽²⁾	`F`	6	6
	Alt	"	No-op	No-op
	Alt Gr	"	102	102
g	None	`g`	103	103
	Shift (caps lock: off)	`G`	71	71
	CapLock: on	`G`	71	71
	Shift + Caplock on	`g`	103	103
	Control ⁽²⁾	`G`	7	7
	Alt	"	No-op	No-op
	Alt Gr	"	103	103
h	None	`h`	104	104
	Shift (caps lock: off)	`H`	72	72
	CapLock: on	`H`	72	72
	Shift + Caplock on	`h`	104	104
	Control ⁽²⁾	`H`	8	8
	Alt	"	No-op	No-op
	Alt Gr	"	104	104
i	None	`i`	105	105
	Shift (caps lock: off)	`I`	73	73
	CapLock: on	`I`	73	73
	Shift + Caplock on	`i`	105	105
	Control ⁽²⁾	`I`	9	9
	Alt	"	No-op	No-op
	Alt Gr / Alt Gr Shift	`í` / `Í`	237 / 205	237 / 205
j	None	`j`	106	106
	Shift (caps lock: off)	`J`	74	74
	CapLock: on	`J`	74	74
	Shift + Caplock on	`j`	106	106
	Control ⁽²⁾	`J`	10	10
	Alt	"	No-op	No-op
	Alt Gr	"	106	106

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
k	None	`k'	107	107
	Shift (caps lock: off)	`K'	75	75
	CapLock: on	`K'	75	75
	Shift + Caplock on	`k'	107	107
	Control ⁽²⁾	'K'	11	11
	Alt	"	No-op	No-op
	Alt Gr	"	107	107
l	None	`l'	108	108
	Shift (caps lock: off)	`L'	76	76
	CapLock: on	`L'	76	76
	Shift + Caplock on	`l'	108	108
	Control ⁽²⁾	'L'	12	12
	Alt	"	No-op	No-op
	Alt Gr	"	108	108
m	None	`m'	109	109
	Shift (caps lock: off)	`M'	77	77
	CapLock: on	`M'	77	77
	Shift + Caplock on	`m'	109	109
	Control ⁽²⁾	'M'	13	13
	Alt	"	No-op	No-op
	Alt Gr	"	109	109
n	None	`n'	110	110
	Shift (caps lock: off)	`N'	78	78
	CapLock: on	`N'	78	78
	Shift + Caplock on	`n'	110	110
	Control ⁽²⁾	'N'	14	14
	Alt	"	No-op	No-op
	Alt Gr	"	110	110

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
o	None	`o'	111	111
	Shift (caps lock: off)	`O'	79	79
	CapLock: on	`O'	79	79
	Shift + Caplock on	`o'	111	111
	Control ⁽²⁾	'O'	15	15
	Alt	"	No-op	No-op
	Alt Gr / Alt Gr Shift	'ó' / 'Ó'	243 / 211	243 / 211
p	None	`p'	112	112
	Shift (caps lock: off)	`P'	80	80
	CapLock: on	`P'	80	80
	Shift + Caplock on	`p'	112	112
	Control ⁽²⁾	'P'	16	16
	Alt	"	No-op	No-op
	Alt Gr	"	112	112
q	None	`q'	113	113
	Shift (caps lock: off)	`Q'	81	81
	CapLock: on	`Q'	81	81
	Shift + Caplock on	`q'	113	113
	Control ⁽²⁾	'Q'	17	17
	Alt	"	No-op	No-op
	Alt Gr	"	113	113
r	None	`r'	114	114
	Shift (caps lock: off)	`R'	82	82
	CapLock: on	`R'	82	82
	Shift + Caplock on	`r'	114	114
	Control ⁽²⁾	'R'	18	18
	Alt	"	No-op	No-op
	Alt Gr	"	114	114

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
s	None	's'	115	115
	Shift (caps lock: off)	'S'	83	83
	CapLock: on	'S'	83	83
	Shift + Caplock on	's'	115	115
	Control ⁽²⁾	'S'	19	19
	Alt	"	No-op	No-op
	Alt Gr	"	115	115
t	None	't'	116	116
	Shift (caps lock: off)	'T'	84	84
	CapLock: on	'T'	84	84
	Shift + Caplock on	't'	116	116
	Control ⁽²⁾	'T'	20	20
	Alt	"	No-op	No-op
	Alt Gr	"	116	116
u	None	'u'	117	117
	Shift (caps lock: off)	'U'	85	85
	CapLock: on	'U'	85	85
	Shift + Caplock on	'u'	117	117
	Control ⁽²⁾	"	21	21
	Alt	'U'	No-op	No-op
	Alt Gr / Alt Gr Shift	'ú ' / 'Ú'	250 / 218	250 / 218
v	None	'v'	118	118
	Shift (caps lock: off)	'V'	86	86
	CapLock: on	'V'	86	86
	Shift + Caplock on	'v'	118	118
	Control ⁽²⁾	'V'	22	22
	Alt	"	No-op	No-op
	Alt Gr	"	118	118

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
w	None	'w'	119	119
	Shift (caps lock: off)	'W'	87	87
	CapLock: on	'W'	87	87
	Shift + Caplock on	'w'	119	119
	Control ⁽²⁾	'W'	23	23
	Alt	"	No-op	No-op
	Alt Gr	"	119	119
x	None	'x'	120	120
	Shift (caps lock: off)	'X'	88	88
	CapLock: on	'X'	88	88
	Shift + Caplock on	'x'	120	120
	Control ⁽²⁾	'X'	24	24
	Alt	"	No-op	No-op
	Alt Gr	"	120	120
y	None	'y'	121	121
	Shift (caps lock: off)	'Y'	89	89
	CapLock: on	'Y'	89	89
	Shift + Caplock on	'y'	121	121
	Control ⁽²⁾	'Y'	25	25
	Alt	"	No-op	No-op
	Alt Gr	"	121	121
z	None	'z'	122	122
	Shift (caps lock: off)	'Z'	90	90
	CapLock: on	'Z'	90	90
	Shift + Caplock on	'z'	122	122
	Control ⁽²⁾	'Z'	26	26
	Alt	"	No-op	No-op
	Alt Gr	"	122	122
, < (comma, less than angle bracket)	None	','	44	44
	Shift (caps lock: off)	'<'	60	60
	CapLock: on	','	44	44
	Shift + Caplock on	'<'	60	60
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	44	44

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
. > (period, greater than angle bracket)	None	`.'	46	46
	Shift (caps lock: off)	`>'	62	62
	CapLock: on	`.'	46	46
	Shift + Caplock on	`>'	62	62
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	46	46
/ ? (forward slash, question mark)	None	`/'	47	47
	Shift (caps lock: off)	`?'	63	63
	CapLock: on	`/'	47	47
	Shift + Caplock on	`?'	63	63
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	47	47
; : (semi colon, colon)	None	`;'	59	59
	Shift (caps lock: off)	`:'	58	58
	CapLock: on	`;'	59	59
	Shift + Caplock on	`:'	58	58
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	59	59
` UK: @ US: `` (Apostrophe, UK: At sign, US: double quotes)	None	``'	39	39
	Shift (caps lock: off)	UK: `@' US: `'''	UK: 64 US: 34	UK: 64 US: 34
	CapLock: on	``'	39	39
	Shift + Caplock on	UK: `@' US: `'''	UK: 64 US: 34	UK: 64 US: 34
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
# (Number sign / hash) ~ (Tilde) UK: ~ (Tilde) US: N/A (UK keyboard only)	None	`#'	35	35
	Shift (caps lock: off)	`~'	126	126
	CapLock: on	`#'	35	35
	Shift + Caplock on	`~'	126	126
	Control	"	28	28
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op
[{ (open square bracket, open curly bracket)	None	`['	91	91
	Shift (caps lock: off)	`{'	123	123
	CapLock: on	`['	91	91
	Shift + Caplock on	`{'	123	123
	Control	"	27	27
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
] } (close square bracket, close curly bracket)	None	'] '	93	93
	Shift (caps lock: off)	' }] '	125	125
	CapLock: on	'] '	93	93
	Shift + Caplock on	' }] '	125	125
	Control	"	29	29
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op
\ (backslash, vertical bar 'pipe')	None	' \ '	92	92
	Shift (caps lock: off)	' '	124	124
	CapLock: on	' \ '	92	92
	Shift + Caplock on	' '	124	124
	Control	"	28	28
	Alt	"	No-op	No-op
	Alt Gr	"	92	92
<space> (space bar)	None	' ' '	32	32
	Shift (caps lock: off)	' ' '	32	32
	CapLock: on	' ' '	32	32
	Shift + Caplock on	' ' '	32	32
	Control	"	32	32
	Alt	"	32	32
	Alt Gr	"	32	32
<Esc> (Escape)	None	"	27	27
	Shift (caps lock: off)	"	27	27
	CapLock: on	"	27	27
	Shift + Caplock on	"	27	27
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	27	27
<Enter> <Return>	None	"	13	13
	Shift (caps lock: off)	"	13	13
	CapLock: on	"	13	13
	Shift + Caplock on	"	13	13
	Control	"	10	10
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
<Backspace> < ← >	None	"	27	27
	Shift (caps lock: off)	"	27	27
	CapLock: on	"	27	27
	Shift + Caplock on	"	27	27
	Control	"	127	127
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op
<Tab> < → > < ← >	None	"	9	9
	Shift (caps lock: off)	"	9	9
	CapLock: on	"	9	9
	Shift + Caplock on	"	9	9
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
Function Keys				
Left Arrow	None	"	129	129
	Shift (caps lock: off)	"	129	129
	CapLock: on	"	129	129
	Shift + Caplock on	"	129	129
	Control	"	129	129
	Alt	"	No-op	No-op
	Alt Gr	"	129	129
Right Arrow	None	"	130	130
	Shift (caps lock: off)	"	130	130
	CapLock: on	"	130	130
	Shift + Caplock on	"	130	130
	Control	"	130	130
	Alt	"	No-op	No-op
	Alt Gr	"	130	130

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
Up Arrow	None	"	131	131
	Shift (caps lock: off)	"	131	131
	CapLock: on	"	131	131
	Shift + Caplock on	"	131	131
	Control	"	131	131
	Alt	"	No-op	No-op
	Alt Gr	"	131	131
Down Arrow	None	"	132	132
	Shift (caps lock: off)	"	132	132
	CapLock: on	"	132	132
	Shift + Caplock on	"	132	132
	Control	"	132	132
	Alt	"	No-op	No-op
	Alt Gr	"	132	132

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
Insert	None	"	133	133
	Shift (caps lock: off)	"	133	133
	CapLock: on	"	133	133
	Shift + Caplock on	"	133	133
	Control	"	133	133
	Alt	"	No-op	No-op
	Alt Gr	"	133	133
Delete	None	"	134	134
	Shift (caps lock: off)	"	134	134
	CapLock: on	"	134	134
	Shift + Caplock on	"	134	134
	Control	"	134	134
	Alt	"	No-op	No-op
	Alt Gr	"	134	134
Home	None	"	135	135
	Shift (caps lock: off)	"	135	135
	CapLock: on	"	135	135
	Shift + Caplock on	"	135	135
	Control	"	135	135
	Alt	"	No-op	No-op
	Alt Gr	"	135	135
End	None	"	136	136
	Shift (caps lock: off)	"	136	136
	CapLock: on	"	136	136
	Shift + Caplock on	"	136	136
	Control	"	136	136
	Alt	"	No-op	No-op
	Alt Gr	"	136	136

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
Prior (Page Up)	None	"	137	137
	Shift (caps lock: off)	"	137	137
	CapLock: on	"	137	137
	Shift + Caplock on	"	137	137
	Control	"	137	137
	Alt	"	No-op	No-op
	Alt Gr	"	137	137
Next (Page Down)	None	"	138	138
	Shift (caps lock: off)	"	138	138
	CapLock: on	"	138	138
	Shift + Caplock on	"	138	138
	Control	"	138	138
	Alt	"	No-op	No-op
	Alt Gr	"	138	138
Num Lock	None	"	144	144
	Shift (caps lock: off)	"	144	144
	CapLock: on	"	144	144
	Shift + Caplock on	"	144	144
	Control ₁₀	"	19	19
	Alt	"	No-op	No-op
	Alt Gr	"	144	144
Scroll Lock	None	"	145	145
	Shift (caps lock: off)	"	145	145
	CapLock: on	"	145	145
	Shift + Caplock on	"	145	145
	Control ₁₁	"	3	3
	Alt	"	No-op	No-op
	Alt Gr	"	145	145

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
F1	None	"	146	146
	Shift (caps lock: off)	"	146	146
	CapLock: on	"	146	146
	Shift + Caplock on	"	146	146
	Control	"	146	146
	Alt	"	No-op	No-op
	Alt Gr	"	146	146
F2	None	"	147	147
	Shift (caps lock: off)	"	147	147
	CapLock: on	"	147	147
	Shift + Caplock on	"	147	147
	Control	"	147	147
	Alt	"	No-op	No-op
	Alt Gr	"	147	147
F3	None	"	148	148
	Shift (caps lock: off)	"	148	148
	CapLock: on	"	148	148
	Shift + Caplock on	"	148	148
	Control	"	148	148
	Alt	"	No-op	No-op
	Alt Gr	"	148	148
F4	None	"	149	149
	Shift (caps lock: off)	"	149	149
	CapLock: on	"	149	149
	Shift + Caplock on	"	149	149
	Control	"	149	149
	Alt	"	No-op	No-op
	Alt Gr	"	149	149

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
F5	None	"	150	150
	Shift (caps lock: off)	"	150	150
	CapLock: on	"	150	150
	Shift + Caplock on	"	150	150
	Control	"	150	150
	Alt	"	No-op	No-op
	Alt Gr	"	150	150
F6	None	"	151	151
	Shift (caps lock: off)	"	151	151
	CapLock: on	"	151	151
	Shift + Caplock on	"	151	151
	Control	"	151	151
	Alt	"	No-op	No-op
	Alt Gr	"	151	151
F7	None	"	152	152
	Shift (caps lock: off)	"	152	152
	CapLock: on	"	152	152
	Shift + Caplock on	"	152	152
	Control	"	152	152
	Alt	"	No-op	No-op
	Alt Gr	"	152	152
F8	None	"	153	153
	Shift (caps lock: off)	"	153	153
	CapLock: on	"	153	153
	Shift + Caplock on	"	153	153
	Control	"	153	153
	Alt	"	No-op	No-op
	Alt Gr	"	153	153

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 0	Returned k% Mode 1
F9	None	"	154	154
	Shift (caps lock: off)	"	154	154
	CapLock: on	"	154	154
	Shift + Caplock on	"	154	154
	Control	"	154	154
	Alt	"	No-op	No-op
	Alt Gr	"	154	154
F10	None	"	No-op	No-op
	Shift (caps lock: off)	"	No-op	No-op
	CapLock: on	"	No-op	No-op
	Shift + Caplock on	"	No-op	No-op
	Control	"	No-op	No-op
	Alt	"	No-op	No-op
	Alt Gr	"	No-op	No-op
F11	None	"	156	156
	Shift (caps lock: off)	"	156	156
	CapLock: on	"	156	156
	Shift + Caplock on	"	156	156
	Control	"	156	156
	Alt	"	No-op	No-op
	Alt Gr	"	156	156
F12	None	"	157	157
	Shift (caps lock: off)	"	157	157
	CapLock: on	"	157	157
	Shift + Caplock on	"	157	157
	Control	"	157	157
	Alt	"	No-op	No-op
	Alt Gr	"	157	157

V3 Appendix II – Keyboard Tables

Key tables for modes 2 and 3:

This table complements the tables listed in KR command description. Some information is duplicated in presentation. Please review the notes following the tables.

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
` (Grave Accent) UK: ¬ (logical not) US: ~ (tilde) UK Alt-Gr: (broken bar / split vertical bar)	None	``	96	96
	Shift (caps lock: off)	UK: `¬' US: " ~"	UK: 172 US: 126	96
	CapLock: on	``	96	96
	Shift + Caplock on	UK: `¬' US: " ~"	UK: 172 US: 126	96
	Control	"	96	96
	Alt	"	96	96
	Alt Gr	' '	166	96
1 ! (exclamation mark)	None	`1'	49	49
	Shift (caps lock: off)	`!'	33	49
	CapLock: on	`1'	49	49
	Shift + Caplock on	`!'	33	49
	Control	"	49	49
	Alt	"	49	49
	Alt Gr	"	49	49
2 UK: " US: @ (UK: double quotes, US: At sign)	None	`2'	50	50
	Shift (caps lock: off)	UK: `"" US: "@"	UK: 34 US: 64	50
	CapLock: on	`2'	50	50
	Shift + Caplock on	UK: `"" US: "@"	UK: 34 US: 64	50
	Control	"	50	50
	Alt	"	50	50
	Alt Gr	"	50	50
3 UK: £ US: # (British Pound Sterling, Number sign / hash)	None	`3'	51	51
	Shift (caps lock: off)	UK: `£' US: "#"	UK: 163 US: 35	51
	CapLock: on	`3'	51	51
	Shift + Caplock on	UK: `£' US: "#"	UK: 163 US: 35	51
	Control	"	51	51
	Alt	"	51	51
	Alt Gr	"	51	51

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
4 \$ UK Alt Gr: € (Dollar sign, Euro)	None	'4'	52	52
	Shift (caps lock: off)	'\$'	36	52
	CapLock: on	'4'	52	52
	Shift + Caplock on	'\$'	36	52
	Control	"	52	52
	Alt	"	52	52
	Alt Gr	'€'	128	52
5 % (percent sign)	None	'5'	53	53
	Shift (caps lock: off)	'%'	37	53
	CapLock: on	'5'	53	53
	Shift + Caplock on	'%'	37	53
	Control	"	53	53
	Alt	"	53	53
	Alt Gr	"	53	53
6 ^ (Circumflex accent or Caret)	None	'6'	54	54
	Shift (caps lock: off)	'^'	94	54
	CapLock: on	'6'	54	54
	Shift + Caplock on	'^'	94	54
	Control	"	54	54
	Alt	"	54	54
	Alt Gr	"	54	54
7 & (Ampersand)	None	'7'	55	55
	Shift (caps lock: off)	'&'	38	55
	CapLock: on	'7'	55	55
	Shift + Caplock on	'&'	38	55
	Control	"	55	55
	Alt	"	55	55
	Alt Gr	"	55	55

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
8 * (Asterisk)	None	'8'	56	56
	Shift (caps lock: off)	'*'	42	56
	CapLock: on	'8'	56	56
	Shift + Caplock on	'*'	42	56
	Control	"	56	56
	Alt	"	56	56
	Alt Gr	"	56	56
9 ((open parenthesis / round bracket)	None	'9'	57	57
	Shift (caps lock: off)	'('	40	57
	CapLock: on	'9'	57	57
	Shift + Caplock on	'('	40	57
	Control	"	57	57
	Alt	"	57	57
	Alt Gr	"	57	57
1) (close parenthesis / round bracket)	None	'0'	48	48
	Shift (caps lock: off)	')'	41	48
	CapLock: on	'0'	48	48
	Shift + Caplock on	')'	41	48
	Control	"	48	48
	Alt	"	48	48
	Alt Gr	"	48	48
- (minus sign) _ (underscore)	None	'_'	45	45
	Shift (caps lock: off)	'_'	95	45
	CapLock: on	'_'	45	45
	Shift + Caplock on	'_'	95	45
	Control	"	45	45
	Alt	"	45	45
	Alt Gr	"	45	45
= (equal) + (plus)	None	'='	61	61
	Shift (caps lock: off)	'+'	43	61
	CapLock: on	'='	61	61
	Shift + Caplock on	'+'	43	61
	Control	"	61	61
	Alt	"	61	61
	Alt Gr	"	61	61

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
a	None	`a'	97	97
	Shift (caps lock: off)	`A'	65	97
	CapLock: on	`A'	65	97
	Shift + Caplock on	`a'	97	97
	Control ⁽²⁾	"	1	97
	Alt	"	97	97
	Alt Gr	"	97	97
b	None	`b'	98	98
	Shift (caps lock: off)	`B'	66	98
	CapLock: on	`B'	66	98
	Shift + Caplock on	`b'	98	98
	Control ⁽²⁾	"	2	98
	Alt	"	98	98
	Alt Gr	"	98	98
c	None	`c'	99	99
	Shift (caps lock: off)	`C'	67	99
	CapLock: on	`C'	67	99
	Shift + Caplock on	`c'	99	99
	Control ⁽²⁾	"	3	99
	Alt	"	99	99
	Alt Gr	"	99	99
d	None	`d'	100	100
	Shift (caps lock: off)	`D'	68	100
	CapLock: on	`D'	68	100
	Shift + Caplock on	`d'	100	100
	Control ⁽²⁾	"	4	100
	Alt	"	100	100
	Alt Gr	"	100	100
e	None	`e'	101	101
	Shift (caps lock: off)	`E'	69	101
	CapLock: on	`E'	69	101
	Shift + Caplock on	`e'	101	101
	Control ⁽²⁾	"	5	101
	Alt	"	101	101
	Alt Gr	"	101	101

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
f	None	`f`	102	102
	Shift (caps lock: off)	`F`	70	102
	CapLock: on	`F`	70	102
	Shift + Caplock on	`f`	102	102
	Control ⁽²⁾	"	6	102
	Alt	"	102	102
	Alt Gr	"	102	102
g	None	`g`	103	103
	Shift (caps lock: off)	`G`	71	103
	CapLock: on	`G`	71	103
	Shift + Caplock on	`g`	103	103
	Control ⁽²⁾	"	7	103
	Alt	"	103	103
	Alt Gr	"	103	103
h	None	`h`	104	104
	Shift (caps lock: off)	`H`	72	104
	CapLock: on	`H`	72	104
	Shift + Caplock on	`h`	104	104
	Control ⁽²⁾	"	8	104
	Alt	"	104	104
	Alt Gr	"	104	104
i	None	`i`	105	105
	Shift (caps lock: off)	`I`	73	105
	CapLock: on	`I`	73	105
	Shift + Caplock on	`i`	105	105
	Control ⁽²⁾	"	9	105
	Alt	"	105	105
	Alt Gr	"	105	105
j	None	`j`	106	106
	Shift (caps lock: off)	`J`	74	106
	CapLock: on	`J`	74	106
	Shift + Caplock on	`j`	106	106
	Control ⁽²⁾	"	10	106
	Alt	"	106	106
	Alt Gr	"	106	106

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
k	None	`k'	107	107
	Shift (caps lock: off)	`K'	75	107
	CapLock: on	`K'	75	107
	Shift + Caplock on	`k'	107	107
	Control ⁽²⁾	"	11	107
	Alt	"	107	107
	Alt Gr	"	107	107
l	None	`l'	108	108
	Shift (caps lock: off)	`L'	76	108
	CapLock: on	`L'	76	108
	Shift + Caplock on	`l'	108	108
	Control ⁽²⁾	"	12	108
	Alt	"	108	108
	Alt Gr	"	108	108
m	None	`m'	109	109
	Shift (caps lock: off)	`M'	77	109
	CapLock: on	`M'	77	109
	Shift + Caplock on	`m'	109	109
	Control ⁽²⁾	"	13	109
	Alt	"	109	109
	Alt Gr	"	109	109
n	None	`n'	110	110
	Shift (caps lock: off)	`N'	78	110
	CapLock: on	`N'	78	110
	Shift + Caplock on	`n'	110	110
	Control ⁽²⁾	"	14	110
	Alt	"	110	110
	Alt Gr	"	110	110

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
o	None	`o'	111	111
	Shift (caps lock: off)	`O'	79	111
	CapLock: on	`O'	79	111
	Shift + Caplock on	`o'	111	111
	Control ⁽²⁾	"	15	111
	Alt	"	111	111
	Alt Gr	"	111	111
p	None	`p'	112	112
	Shift (caps lock: off)	`P'	80	112
	CapLock: on	`P'	80	112
	Shift + Caplock on	`p'	112	112
	Control ⁽²⁾	"	16	112
	Alt	"	112	112
	Alt Gr	"	112	112
q	None	`q'	113	113
	Shift (caps lock: off)	`Q'	81	113
	CapLock: on	`Q'	81	113
	Shift + Caplock on	`q'	113	113
	Control ⁽²⁾	"	17	113
	Alt	"	113	113
	Alt Gr	"	113	113
r	None	`r'	114	114
	Shift (caps lock: off)	`R'	82	114
	CapLock: on	`R'	82	114
	Shift + Caplock on	`r'	114	114
	Control ⁽²⁾	"	18	114
	Alt	"	114	114
	Alt Gr	"	114	114

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
s	None	`s'	115	115
	Shift (caps lock: off)	`S'	83	115
	CapLock: on	`S'	83	115
	Shift + Caplock on	`s'	115	115
	Control ⁽²⁾	"	19	115
	Alt	"	115	115
	Alt Gr	"	115	115
t	None	`t'	116	116
	Shift (caps lock: off)	`T'	84	116
	CapLock: on	`T'	84	116
	Shift + Caplock on	`t'	116	116
	Control ⁽²⁾	"	20	116
	Alt	"	116	116
	Alt Gr	"	116	116
u	None	`u'	117	117
	Shift (caps lock: off)	`U'	85	117
	CapLock: on	`U'	85	117
	Shift + Caplock on	`u'	117	117
	Control ⁽²⁾	"	21	117
	Alt	"	117	117
	Alt Gr	"	117	117
v	None	`v'	118	118
	Shift (caps lock: off)	`V'	86	118
	CapLock: on	`V'	86	118
	Shift + Caplock on	`v'	118	118
	Control ⁽²⁾	"	22	118
	Alt	"	118	118
	Alt Gr	"	118	118

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
w	None	'w'	119	119
	Shift (caps lock: off)	'W'	87	119
	CapLock: on	'W'	87	119
	Shift + Caplock on	'w'	119	119
	Control ⁽²⁾	"	23	119
	Alt	"	119	119
	Alt Gr	"	119	119
x	None	'x'	120	120
	Shift (caps lock: off)	'X'	88	120
	CapLock: on	'X'	88	120
	Shift + Caplock on	'x'	120	120
	Control ⁽²⁾	"	24	120
	Alt	"	120	120
	Alt Gr	"	120	120
y	None	'y'	121	121
	Shift (caps lock: off)	'Y'	89	121
	CapLock: on	'Y'	89	121
	Shift + Caplock on	'y'	121	121
	Control ⁽²⁾	"	25	121
	Alt	"	121	121
	Alt Gr	"	121	121
z	None	'z'	122	122
	Shift (caps lock: off)	'Z'	90	122
	CapLock: on	'Z'	90	122
	Shift + Caplock on	'z'	122	122
	Control ⁽²⁾	"	26	122
	Alt	"	122	122
	Alt Gr	"	122	122
, < (comma, less than angle bracket)	None	','	44	44
	Shift (caps lock: off)	'<'	60	44
	CapLock: on	','	44	44
	Shift + Caplock on	'<'	60	44
	Control	"	44	44
	Alt	"	44	44
	Alt Gr	"	44	44

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
. > (period, greater than angle bracket)	None	`.'	46	46
	Shift (caps lock: off)	`>'	62	46
	CapLock: on	`.'	46	46
	Shift + Caplock on	`>'	62	46
	Control	''	46	46
	Alt	''	46	46
	Alt Gr	''	46	46
/ ? (forward slash, question mark)	None	`/'	47	47
	Shift (caps lock: off)	`?'	63	47
	CapLock: on	`/'	47	47
	Shift + Caplock on	`?'	63	47
	Control	''	47	47
	Alt	''	47	47
	Alt Gr	''	47	47
; : (semi colon, colon)	None	`;'	59	59
	Shift (caps lock: off)	`:'	58	58
	CapLock: on	`;'	59	59
	Shift + Caplock on	`:'	58	58
	Control	''	59	59
	Alt	''	59	59
	Alt Gr	''	59	59
` UK: @ US: `` (Apostrophe, UK: At sign, US: double quotes)	None	``'	39	39
	Shift (caps lock: off)	UK: '@' US: '``'	UK: 64 US: 34	39
	CapLock: on	``'	39	39
	Shift + Caplock on	UK: '@' US: '``'	UK: 64 US: 34	39
	Control	''	39	39
	Alt	''	39	39
	Alt Gr	''	39	39

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
# (Number sign / hash) ~ (Tilde) UK: ~ (Tilde) US: N/A (UK keyboard only)	None	`#'	35	35
	Shift (caps lock: off)	`~'	126	35
	CapLock: on	`#'	35	35
	Shift + Caplock on	`~'	126	35
	Control	"	35	35
	Alt	"	35	35
	Alt Gr	"	35	35
[{ (open square bracket, open curly bracket)	None	`['	91	91
	Shift (caps lock: off)	`{'	123	91
	CapLock: on	`['	91	91
	Shift + Caplock on	`{'	123	91
	Control	"	27	91
	Alt	"	91	91
	Alt Gr	"	27	91

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
] } (close square bracket, close curly bracket)	None	'] '	93	93
	Shift (caps lock: off)	' }] '	125	93
	CapLock: on	'] '	93	93
	Shift + Caplock on	' }] '	125	93
	Control	"	29	93
	Alt	"	93	93
	Alt Gr	"	29	93
\ (backslash, vertical bar 'pipe')	None	' \ '	92	92
	Shift (caps lock: off)	' '	124	92
	CapLock: on	' \ '	92	92
	Shift + Caplock on	' '	124	92
	Control	"	28	92
	Alt	"	92	92
	Alt Gr	"	92	92
<space> (space bar)	None	' '	32	32
	Shift (caps lock: off)	' '	32	32
	CapLock: on	' '	32	32
	Shift + Caplock on	' '	32	32
	Control	"	32	32
	Alt	"	32	32
	Alt Gr	"	32	32
<Esc> (Escape)	None	"	27	27
	Shift (caps lock: off)	"	27	27
	CapLock: on	"	27	27
	Shift + Caplock on	"	27	27
	Control	"	27	27
	Alt	"	27	27
	Alt Gr	"	27	27
<Enter> <Return>	None	"	13	13
	Shift (caps lock: off)	"	13	13
	CapLock: on	"	13	13
	Shift + Caplock on	"	13	13
	Control	"	10	10
	Alt	"	13	13
	Alt Gr	"	13	13

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
<Backspace> < ← >	None	"	27	27
	Shift (caps lock: off)	"	27	27
	CapLock: on	"	27	27
	Shift + Caplock on	"	27	27
	Control	"	27	27
	Alt	"	27	27
	Alt Gr	"	No-op	No-op
<Tab> < → > < ← >	None	"	9	9
	Shift (caps lock: off)	"	9	9
	CapLock: on	"	9	9
	Shift + Caplock on	"	9	9
	Control	"	No-op	No-op
	Alt	"	9	9
	Alt Gr	"	No-op	No-op

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
Function Keys				
Left Arrow	None	"	129	129
	Shift (caps lock: off)	"	129	129
	CapLock: on	"	129	129
	Shift + Caplock on	"	129	129
	Control	"	129	129
	Alt	"	129	129
	Alt Gr	"	129	129
Right Arrow	None	"	130	130
	Shift (caps lock: off)	"	130	130
	CapLock: on	"	130	130
	Shift + Caplock on	"	130	130
	Control	"	130	130
	Alt	"	130	130
	Alt Gr	"	130	130

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
Up Arrow	None	"	131	131
	Shift (caps lock: off)	"	131	131
	CapLock: on	"	131	131
	Shift + Caplock on	"	131	131
	Control	"	131	131
	Alt	"	131	131
	Alt Gr	"	131	131
Down Arrow	None	"	132	132
	Shift (caps lock: off)	"	132	132
	CapLock: on	"	132	132
	Shift + Caplock on	"	132	132
	Control	"	132	132
	Alt	"	132	132
	Alt Gr	"	132	132

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
Insert	None	"	133	133
	Shift (caps lock: off)	"	133	133
	CapLock: on	"	133	133
	Shift + Caplock on	"	133	133
	Control	"	133	133
	Alt	"	133	133
	Alt Gr	"	133	133
Delete	None	"	134	134
	Shift (caps lock: off)	"	134	134
	CapLock: on	"	134	134
	Shift + Caplock on	"	134	134
	Control	"	134	134
	Alt	"	134	134
	Alt Gr	"	134	134
Home	None	"	135	135
	Shift (caps lock: off)	"	135	135
	CapLock: on	"	135	135
	Shift + Caplock on	"	135	135
	Control	"	135	135
	Alt	"	135	135
	Alt Gr	"	135	135
End	None	"	136	136
	Shift (caps lock: off)	"	136	136
	CapLock: on	"	136	136
	Shift + Caplock on	"	136	136
	Control	"	136	136
	Alt	"	136	136
	Alt Gr	"	136	136

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
Prior (Page Up)	None	"	137	137
	Shift (caps lock: off)	"	137	137
	CapLock: on	"	137	137
	Shift + Caplock on	"	137	137
	Control	"	137	137
	Alt	"	137	137
	Alt Gr	"	137	137
Next (Page Down)	None	"	138	138
	Shift (caps lock: off)	"	138	138
	CapLock: on	"	138	138
	Shift + Caplock on	"	138	138
	Control	"	138	138
	Alt	"	138	138
	Alt Gr	"	138	138
Num Lock	None	"	144	144
	Shift (caps lock: off)	"	144	144
	CapLock: on	"	144	144
	Shift + Caplock on	"	144	144
	Control₁₀	"	144	144
	Alt	"	144	144
	Alt Gr	"	144	144
Scroll Lock	None	"	145	145
	Shift (caps lock: off)	"	145	145
	CapLock: on	"	145	145
	Shift + Caplock on	"	145	145
	Control₁₁	"	145	145
	Alt	"	145	145
	Alt Gr	"	145	145

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
F1	None	"	146	146
	Shift (caps lock: off)	"	146	146
	CapLock: on	"	146	146
	Shift + Caplock on	"	146	146
	Control	"	146	146
	Alt	"	146	146
	Alt Gr	"	146	146
F2	None	"	147	147
	Shift (caps lock: off)	"	147	147
	CapLock: on	"	147	147
	Shift + Caplock on	"	147	147
	Control	"	147	147
	Alt	"	147	147
	Alt Gr	"	147	147
F3	None	"	148	148
	Shift (caps lock: off)	"	148	148
	CapLock: on	"	148	148
	Shift + Caplock on	"	148	148
	Control	"	148	148
	Alt	"	148	148
	Alt Gr	"	148	148
F4	None	"	149	149
	Shift (caps lock: off)	"	149	149
	CapLock: on	"	149	149
	Shift + Caplock on	"	149	149
	Control	"	149	149
	Alt	"	149	149
	Alt Gr	"	149	149

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
F5	None	"	150	150
	Shift (caps lock: off)	"	150	150
	CapLock: on	"	150	150
	Shift + Caplock on	"	150	150
	Control	"	150	150
	Alt	"	150	150
	Alt Gr	"	150	150
F6	None	"	151	151
	Shift (caps lock: off)	"	151	151
	CapLock: on	"	151	151
	Shift + Caplock on	"	151	151
	Control	"	151	151
	Alt	"	151	151
	Alt Gr	"	151	151
F7	None	"	152	152
	Shift (caps lock: off)	"	152	152
	CapLock: on	"	152	152
	Shift + Caplock on	"	152	152
	Control	"	152	152
	Alt	"	152	152
	Alt Gr	"	152	152
F8	None	"	153	153
	Shift (caps lock: off)	"	153	153
	CapLock: on	"	153	153
	Shift + Caplock on	"	153	153
	Control	"	153	153
	Alt	"	153	153
	Alt Gr	"	153	153

Key Pressed	Modifier Key State	Returned k\$	Returned k% Mode 2	Returned k% Mode 3
F9	None	"	154	154
	Shift (caps lock: off)	"	154	154
	CapLock: on	"	154	154
	Shift + Caplock on	"	154	154
	Control	"	154	154
	Alt	"	154	154
	Alt Gr	"	154	154
F10	None	"	155	155
	Shift (caps lock: off)	"	155	155
	CapLock: on	"	155	155
	Shift + Caplock on	"	155	155
	Control	"	155	155
	Alt	"	155	155
	Alt Gr	"	155	155
F11	None	"	156	156
	Shift (caps lock: off)	"	156	156
	CapLock: on	"	156	156
	Shift + Caplock on	"	156	156
	Control	"	156	156
	Alt	"	156	156
	Alt Gr	"	156	156
F12	None	"	157	157
	Shift (caps lock: off)	"	157	157
	CapLock: on	"	157	157
	Shift + Caplock on	"	157	157
	Control	"	157	157
	Alt	"	157	157
	Alt Gr	"	157	157

Key tables for modes 0, 1, 2 and 3 - Notes:

1. Reference is made to ASCII see also: UNICODE C0 Basic LATIN coding standard (www.unicode.org - <http://www.unicode.org/charts/PDF/U0000.pdf>)
2. For modes 0 & 1, (Control + key) returns the ASCII character in the specified **Char\$**.
3. Control + Shift and Control + Caplock combinations are defined as follows: The Shift and Caplock to not alter the values returned for "control" character mode as by definition the "control" codes are defined as the first two columns in the ASCII code chart decimal values 0-31 – and are the "control" of ASCII characters in columns 4 and 5 decimal values 64-95 '@','A' – 'Z', '[', '\', ']', '^' and '_' (Underscore).
4. ALT + Shift and ALT + Caplock combinations are not currently defined.
5. Alt is not currently defined for modes 0 & 1.
6. Grave Accent + Shift refers to EBCDIC symbol decimal number 95 (¬ Logical NOT).
7. Grave Accent + Alt Gr refers to EBCDIC symbol decimal number 106 (¦ broken bar / split vertical bar) alternatively may be EBCDIC symbol decimal number 79 (| vertical bar).
8. Number 3 + Shift refer to the British Pound symbol £ with UK keyboard and # for US keyboard.
9. Number 4 + Shift refer to the Dollar symbol \$. Number 4 + Alt Gr refer to the Euro symbol €.
10. Single and double quote are shown quoted – but with extra "white" space for clarity.
11. Control + Num Lock – The keyboard for modes 0 & 1 emits a key code 19.
12. Control + Scroll Lock – The keyboard for modes 0 & 1 emits a key code 3.

24V3 Version History

APPLCORE VERSION	Release Date of this document	Support for 4.5
3.08.41	29/07/2016	Corrections for Bug 2798 in LE command – multiple to be deleted did not work correctly after changes for version 3.08.40.
3.08.40	28/07/2016	Corrections for Bug 2798 in LE command (noted problem in LF after a LE command - same issue as found in LB).
3.08.39	26/07/2016	Corrections for Bug 2798 – A bug in the LB command which would manifest itself interacting with the LF command.
3.08.38	06/00/2016	Updated LF command documentation to include a usage example.
3.08.38	08/02/2016	Adds CV and CH commands
3.08.35	02/11/2015 10/07/2015 11/08/2015	Correction in Jump Time description. Changes for Bug 2704. Version 3.08.35, Dated: 09-July-2015 requires CSI32 version: 1.2.54 or later. If CSI configuration file: (for V4 bnCS_system.ini, V3 CSI.ini) entry DistributeBBC_NETTXRXSTATUS is set to zero (DistributeBBC_NETTXRXSTATUS=0) then Applcore receives Device Modify messages from CSI only if there is Applcore code on Stringtable line 32400.
3.08.34	11/06/2015	Correction for XW parameter checking issue. Bug 2692
3.08.33	06/05/2015	Clarified the description of the ET command. Bug 2642.
3.08.33	20/03/2015	Support for 4.5 environment added. Bugs 2637 & 2639.
3.08.31	24/09/2014	Correction for problem in EH command. Bug 2573.
3.08.30	23/09/2014	Correction for problem in KU command. Bug 2566.
3.08.29	18/02/2014	Updated IL cmd documentation regarding dev_XXX.ini entry "EnableLocks".
3.08.29	18/02/2014	Version 3.08.29 released. Bug 2518 correction for XR issue.
3.08.28	16/02/2014	Spelling correction "memorys" to memories" in QS cmd.
3.08.28	04/02/2014	Minor formatting correction to EV command. Added notes to the EV command.
3.08.28	09/01/2014	Version 3.08.28 released. Bug 2504 correction for crasher noted in revertive handler for RM command (in the Borland version). Correction of a number of Cx, Ex, Nx, Px, Rx & Qx command crashers noted while expanding regression tests. Correction of several buffer and stack issues noted while pushing the code base through Microsoft VS2005 & VS2013).
3.08.27	07/12/2013	Updated ND return variables names to be more descriptive <QuotientVar3%> and <RemainderVar4%>.
3.08.27	06/12/2013	Version 3.08.26 released correction for bug 2491 ND parameter 3 & 4 missing not detected during parameter inspection.

3.08.26	13/11/2013	Version 3.08.26 released correction for bug 2475 NA missing third (results target) parameter not emitting msgs when parameter missing. Correction for error in debug msg parameter set "XS Sending message: ". Correction for bug 2484 XR "ADD" issues.
3.08.25	04/09/2013	Notes added to the VE command regarding video overlay issues identified in bug 2222. Corrections for issues in Xx commands for bug 2381. 3.08.22 – 3.08.25 were pre-release integration & testing releases supplied only internally for test.
3.08.24	29/08/2013	
3.08.23	24/08/2013	
3.08.22	14/08/2013	
3.08.22	30/07/2013	Correction for broken "debug" trace messaging system. This was broken in the fixes for bug 2205 (SA & SC limits).
3.08.21	30/05/2013	Added enhancement features to Xx commands requested in bug 2381 (XC, XS & XW). Corrected Xx command descriptions, added XC command description (was missing).
3.08.20	08/05/2013	Correction to version numbers (version 3.08.19 released to BNCS support web site had errors in the file version properties versus the V3 ApplCore About Box version).
3.08.19	23/03/2013	Version 3.08.19 released to BNCS support web site. Release build of changes in version 3.08.18.
3.08.18	29/03/2013	Integer memories extended from 10,000 to 65,536. Added doc regarding the QS command – which was added in version 3.08.08 (22-August-2013). Version 3.08.18 was a release candidate.
3.08.17	19/03/2013	Corrected AP command documentation and V3 ApplCore parameter testing regarding parameters [<Offset%>] [<OnErrorOffset%>] both of which have always been optional. Corrected documentation for ET command parameters <Offset1%> and <Offset2%> are optional. Corrections to the Jx Line Number parameters. If line number is 0 then continues on same line. Corrected PZ documentation. LF command "reload" description improved.
3.08.16	25/01/2013	Updated RN description, index range is -1 to 4096 (65535 for LAWO). Updated IU command description.
3.08.15	21/11/2012	Updated description of LI cmd. Updated TS documentation.
3.08.15	20/11/2012	Support of the LAWO router requires addressable indices beyond 4096. The RN command has an index limit of 65535.
3.08.14	15/11/2012	Added source file parameter presence checking for IS, EL, HO, LL, LS, MR, SR, SW, and TW cmds. Correction for bug 2313 multiple IS commands in subroutine not executing response vectors. Now marking replies as "InSequence" and queued into the AuxProc as are other msgs from CSI. Correction for bug 2324 parameter issues in LR cmd causing crash (classic strtok() & atol() NULL ptr issues).
	03/11/2012	Updated GU, IU & RU commands regarding <Device%> parameter specified as 0. This facility has been present since (at least) V2 2.03.05 (March-2003) and V3 3.01.01 (April-2005).
3.08.13	15/10/2012	Correction for the "A% overwrite by S%" bug 2250. The problem existed when the optional return variables were not specified. The return variable index was not reset when subsequent XR & RR cmds were executed. XR was not correctly checking for return variables present and RR was not resetting the return variables index.

3.08.12	05/10/2012	The change in version 3.08.12 of V3 ApplCore relative to version 3.08.11 is a correction for bug 2309. The "NB" command 'NOT' operator was not operating correctly. Example: "NB 0 'NOT' 1 A%" should result in A%=1 (i.e. NOT 0 (1 bit operation)) but actually returns A%=2.
3.08.11	28/11/2012	Correction for bug 2307. The "IS" command parameter checking was limiting the <Index%> parameter to slot maximum value rather than line number maximum value.
3.08.10	11/09/2012	Corrected error msgs typo in "TP" cmd. Added return variable over-write protection to: XT Corrected a param check bug in PV. Was looking at param 0 instead of 1. Corrected typo in LF cmd, old search, case #3 not working (broken since Nov-2010 3.07.02). Now building with Borland 3.52 compiler. Correction for EF relative path adding extra back-slash i.e. "c:\some\path\.\my.ini"
3.08.09	28/08/2012	Updated EF documentation (bugs: 2216, 2235 & 2285). The following is a list of the paths tested. Source File Name and Path --- File Path Searched \\ws\c\file.ini \\ws\c\file.ini c:\path\file.ini c:\path\file.ini file.ini <ConfigPath>\file.ini sub\file.ini <ConfigPath>\sub\file.ini .\file.ini <CWD>\file.ini .\sub\file.ini <CWD>\sub\file.ini ..\sub\file.ini <CWD>\..\sub\file.ini
3.08.08	22/08/2012	Updated EF documentation including: <Section\$> and <Entry\$> parameters can be numerics (bugs: 2216, 2235 & 2285). Clarified EO command defaults. Correction for bugs 2216, 2235, 2285 - 'EF' redesign (see email trail). Corrected parameter testing / validation issues covering a number of commands. Correction for bugs 2250 - A% and A\$ variables over-write issues. Added return variable over-write protection to EF, DI, DS, Nx, RD, TQ, XD, and XG commands. Corrected crash problem in XR cmd (Borland only) discovered with regression testing. Correction for bugs 2286 Correction Panel Create VC6/VS version only displays MessageBox on create failure. STRING TABLE lines longer than 256 are flagged with a debug message. STRING TABLE lines containing special case single quote characters 0x91 (-110) or 0x92 (-111) are flagged and converted to ASCII 39. (ASCII single quote).
3.08.06	28/06/2012	Corrected description for RI, and typo's in FD.

3.08.06	24/05/2012	Added <ImmediateFlag%> parameter to AP command. Corrections for bug 2250 GR/IR/RR/XR commands - corrections applied to check for the presence of the optional return variable parameters <IdxRetVar%> and <DriverVar%>. Correction for ND "Quotient" and "Remainder" returned vars were not being checked for existence so A% was being crashed if "Remainder" was not specified. Correction for bugs 2134 & 2245 AP audio curtailed when restarting replay out by issuing another AP cmd. The AP command now has an added parameter <ImmediateMode%>. Set this to '1' for immediate restart of play out when currently playing a clip. Correction for bug 2207 - V3 Applcore TE command wrong variable written. A list of commands where a NULL parameter value is acceptable was produced and the ApplCore parameter inspection system to no longer WARN for String parameters equal to NULL to the following commands: EM, JC, JG, LG, LP, LT, NF, SD, ST, TB, TW
3.08.05	05/05/2012	Updated AM command description.
3.08.05	30/04/2012	Corrected parameter spelling error in LQ command.
3.08.05	18/04/2012	Updated ES when targeted device not running. Clarifies use of <UpdateTallyTableOnly%> parameter with eXternal commands XC, XS & XW. Clarifies use of optional parameters with XR command.
3.08.05	17/04/2012	KR command - corrected typo in Locking Keys description
3.08.05	12/04/2012	Updated parameter names in Lx commands. Updated ES command regarding RevertiveLine% parameter.
3.08.05	27/03/2012	Updated AK operation description. Added missing documentation for JG commad (was presenting Quick Ref doc.). Updated documentation regarding version when QC, QD, QH and QP commands were added (3.04.00). Added new QF command documentation. Correction for TG doesn't work. Correction for bug 2203 parameter parsing problem reported in regarding the use of variable A% in IR/GR/RR cmds. Correction for bug 2205: (continuing effort) Added more parameter evaluation and range checking of Numeric Memory, filenames, PanelID, ControlID, Device Numbers & Data Base, and Line Number parameters. Correction for bug 2221. Correction for NB shift left or right zero, (1 >> 0) and (1 << 0) both were returning 0, should be 1. Correction for bug 2228: Correction for TS 1st parameter being zero length return 0 in the return memory. Correction for bug 2229: Debug Variables Windows Height tweaked and added applcore.ini environment variables to aid in customization: [System] VariablesWindowPositionX=200 VariablesWindowPositionY=20 VariablesWindowSizeW=280 VariablesWindowSizeH=450 Corrrrection for bug 2233: Added support for QF QueueFlush() directs csi32 to "send queued commands/revertives now". Requires csi32.exe version: 1.1.33, dated: 11/07/11 or newer.
3.08.04	24/02/2012	Correction for an issue in 'EM' when no timer value parameter is present - CORRECTED. Correction for a very old "corner case" crasher issue in 'SE' when the env var does not exist.

3.08.02	21/02/2012	Corrected DI command documentation fourth parameter is a string. Corrected EE parameters list. VP command has been obsoleted since V2. Corrected warning message for IW command parameter 2 issue. Correction for TL return string incorrect NULL termination.
3.08.01	20/02/2012	Updated EZ command documentation to reflect additional capabilities. As requested in bug 2205. Updated EM command documentation. There is substantial run-time parameter check starting with version 3.08.01
3.07.13		<p>Corrections for bug 2198, correction for LG missing String return var over writing random. Added filter code to LG, LX, NF, TR & TE. Corrections for bug 2198, evaluation of parameters where target is NULL, is indicated as "<appname> WARNING: line:" rather than "<appname> ERROR: line:". Corrections for bug 2205, eval of SC params where targets result in over size output result, ApplCore crash. Applied to other commands also. Now length range check will produce a message: "WARNING: line: %d, cmd: 'SC' resulting string length exceeds 255. Truncating."</p> <p>Corrections for bug 2207, eval of TL params where wrong syntax is used (TL 0 o%) crashes O\$ - parameter evaluation.</p> <p>Corrections for bug 2119, EZ outputs formatted text as SC cmd (feature request). Corrections for bug 2198, additional corrections for LG, LX, NF, TR, TE parameter validation, filtering, etc. Enhanced function in GetArguments(). Now trapping NON-ASCII characters in the string table. Had noted an apostrophe of -122 decimal in the TL cmd test panel and the parser did not find the literal string because of it. Parameter checking calls added to all commands. Added environment variables to enable/disable parameter inspection and warning and error messages.</p> <p>ApplCore.ini environment variables are:</p> <p style="padding-left: 40px;">EnableParameterInspection=1 EnableParameterInspectMsgs=1</p> <p>Corrections for bug 2206, LF space-free search documentation Corrections for bug 2205, SC string length range check. Corrections for bug 2197, & 2090 EM command does not stop program flow.</p>
3.07.12	14/02/2012	Corrected typos in parameter names of SI cmd. Added missing <ControlId%> parameter in VD & VE cmds. Updated VE documentation. Corrected LF documentation issue: missing documentation for bit 7 Space Free Search. Correction for bug 2180, for EM execution stack problem after messagebox loss of focus. Correction for bug 2203, parameter parsing for parameters 5,6 & 7 in IR, GR & RR correction of return param corruption
3.07.12	03/02/2012	Updated IR / GR / RR syntax regarding optional parameters, thus: [<DestVar%>] [<'ADD'>] [<DriverVar%>]
3.07.11	06/01/2012	Updated document to reflect the change to ATOS. Now using Verdana fonts. Updated documentation for Ax commands bug 2134. Parameter 3 of EM [<Mode%>] is optional.

3.07.11	23/12/2011	Correction for bug 2112 - EX additional issue noted and enhanced EX now performs std. path searches for DLL's. Enhancement AP audio curtailed, added audio thread scheme to support restart - bug 2134. Enhancement to EQ command, i.e. EQ 1000. Correction for bug 2163. Updated ES documentation. Correction for bug 2112, Another enhancement EX performs std. path searches for DLL. Enhancements work for Bug 2134, AP audio curtailed (Ax thread added). Correction for bug 2117, TS did not get integrated from the 3.07.10 test release to the 3.07.10 release.
3.07.10	25/10/2011	Correction for bug 2112 EX - extensive rework of parameter handler/counting scheme. Correction for bug 2117 - TS command failing to populate zero location memory when zero length string specified and was over-writing the integer memory location when zero length string specified. Correction for bug 2130 - KR command 0 & 1 does not need keyboard DLL (KeyHkAC.dll). Correction for bug 2133 - EF section names with spaces issues
3.07.09	09/09/2011	Removed code changes made for bug 2117 TS command in version 3.07.08 (code rolled back to original design).
3.07.08	08/09/2011	Correction for bug 2117 - TS command failing to populate zero location memory. This is a VERY old problem dating back to at least V2 in 2002. Removed changes for bug 2112 EX from version 3.07.08.
3.07.07	08/09/2011	Correction for JS in AC_CLOSEDOWN & AC_DATABASE vector - bugs 2030 & 2105. Correction for bug 2112 EX parameter issues. Correction for bug introduced in 3.07.06 associated with bug fix of TD/TE command (bug 2120). Version 3.07.07 was not release to the BNCS support web site.
3.07.06	23/08/2011	Added timestamp debug msgs in panel destroy. Correction for LP command - bug 2088 - removed crashing debug msg. Correction for TE command - bug 2089 - added parameter validation & warning messages to debug when target is not a string variable, no longer crashes memory when string literal is passed as parameter.
3.07.05	29/03/2011	Correction for LF command old style compatibility bug 2007
3.07.04	22/08/2011	Correction of XD parameter spec. errors. Corrected BP parameter list and description errors. Corrected parameter description errors in EH.
3.07.04	19/07/2011	Added documentation to the EM command.
3.07.04	08/07/2011	Removes duplicate Q series command description from section 16
3.07.04	30/03/2011	Correction to discription of EF.
3.07.04	10/03/2011	Corrections in KR command description.
3.07.04	09/03/2011	Corrections in KR Appendix I & II Keyboard Tables and notation of note 2.
3.07.04	08/03/2011	Corrections in KR documentation (modes 2 & 3), including Appendix I Keyboard Tables.
3.07.03	03/03/2011	Updated TX command description.
3.07.03	24/02/2011	Corrections to DM description.

3.07.02	16/02/2011	Corrections regarding bug #1769 V3GRD with ExternalsAdded - comments in the following sections: 6.2 eXternal Data. 6.4 eXternal Intercept, 10.5 Info Lock, 18.5 Router Lock
3.07.02	10/02/2011	Updated KR description, (KeyHkAC32M library Version: 3.2.0.0, Dated: 08-Feb-2011 supports multiple ApplCore instances – all using KR (2 or 3).
3.07.02	10/02/2011	Removes section and information regarding start-up behaviour and adds to Applcore user ref guide. Bug 1800
3.07.02	07/02/2011	Added SQ - String Query Application Name command.
3.07.02	04/02/2011	Added new section to KR command for external keyboard capture. Supports US & UK keyboards and modifier keys including ALT GR.
3.07.02	04/01/2011	Added section and information regarding start-up behaviour. Bug 1800
3.07.02	30/12/2010	Integrated documentation for KR command (from version 3.05.39). Added notes regarding C:\bncs_config.ini Added support for hexadecimal notation values conversion added to NC Tidy up of various typo's, etc.

3.07.02	02/12/2010	<p>Added enhancements to LF command ListBox support search functions and keyboard linkage enable/disable - BUG 1620.</p> <p>Added NB (Number bit operations) - BUG 1620.</p> <p>Bug fix in init re: pApplDllCall not initialized - causing exception during init with debug CC libs.</p> <p>Disabled DEBUG variables call to BringWindowToTop(hWndDebug) so no updates on debug variables windows so keyboard stays in focus on editboxes, etc. - BUG 1620.</p> <p>This corrects a problem (BUG 1693) dating to the earlier versions of ApplCore - where the --- time/date timer (#1) is stopped --- the panel is De-initialized --- the panel is Re-initialized --- the panel is then shown --- time/date timer (#1) is then started.</p> <p>The bug is that the timer DID NOT start up and therefore there was no timer activity to T\$ and U\$ date/time -- no date/time events to the panel. for example: ET 1 0 0 0 :PD 1 :PI 1 :PS 1 :ET 1 1000 9000 9001 :ET 1 1000 9000 9001</p> <p>Added support V3 EZ (v2 EO) command for outputting debug strings from V2 ApplCore version 2.02.29 03/07/02</p> <p>Added notes to Audio Beep (AB) command.</p> <p>Added debug window size and position variables in [Debug] section of ApplCore.ini – DebugWindowPositionX, DebugWindowPositionY, DebugWindowSizeW, DebugWindowSizeH. Defaults are: 20, 0, 640, 480 respectively.</p> <p>Added documentation for ApplCore.ini file</p> <p>Added support for access to DB 2-9 in RI command. Bug 1598 requires use of CSI32 (usesBBC_GETDBASEINDEXEXT).</p>
3.07.01	10/11/2010	Updated EO documentation.
3.07.01	04/10/2010	Added note in EF regarding default search file.
3.07.01	15/09/2010	Correction of EF ref. to ApplCore.ini, LF description, LQ spelling error, PI, PR, PS max panels is 64, EX description was old version – updated. General clean up of many errors in this version history.
3.07.01	27/08/2010	Correction to EA -- now handles file: bncs_monitor.exe and c:/bncs/clients/bncs_montiro.exe correctly. Bug 1625
3.07.00	22/04/2010	Added XN command to get TxRx status to external client.

3.06.09	10/03/2010	Additional corrections for EF and EA long file name problem. Bug 1585 (EA wd.exe VS EA c:/bncs/clients/wd.exe) Noted missing separator between command code and parameter does not work correctly as per V1/V2 (example: :TL'This is ' j\$) -- added detection and correction code in GetArguments(). bug #1587.
3.06.08	19/01/2010	Correction for EA 8+3 VS long file names issues bug #1468. Correction for EF parameter number three String VS Numeric issues. bug #1356.
3.06.07	16/11/2009	Correction for EA with NULL parameters, single space for parameters crasher. bug #1468/1519. Correction EA S\$ not translating in "1" case. bug #1520
3.06.06	02/11/2009	Fix for EF correction as specified for bugs identified. "NewStartUp" ApplCore.ini env var is not longer created in ApplCore.ini, and the default is set to 1. (bug #1468).
3.06.05	20/10/2009	Fix for EA /Panel manager issue (bug 1468). Minimum applicationname is eight, padded with underscore characters followed by three numbers of workstation ID number (bug #1420).
3.06.04	30/09/2009	Fixes for additional EA issues. Fixes for EH command (bug #1420). Added ApplCore.INI "System", "TxVerticalBarCRLF", default is disabled (0). Fix for off by one problem in the number of panels which can be started, MaxExecWindows=3 could only run 2 panels. Fix for EH 0 0 & EH 0 1 cases. bug #1437.
3.06.03	06/08/2009	Corrections for EF issues noted since moveable "windows" directory support added, i.e. bncs/windows directory support. Changes to support long file names in EA. Added 'V2CompatibleApplNames' to Applcore.ini in order to force the "old" V2 compatible application name convention -- for support of mixed V2 & V3 development environments.
3.06.02	01/09/2009	Corrections for EA issues regarding long and short file names (less than 8.3).
3.06.01	04/08/2009	Enhancement to Registration commands RR / IR / GR, added <DriverVar%> which contains the driver number in the specified var% when the revertive handler is called [Bug 1270]. Correction for EH issue -- show top window after an EH panel 0 cmd. [Bug 1306]. New STARTUP functionality. Start line (30000) is started via post message, this corrects the uncertain start up behaviour noted in XR and XP test panel (in start up lines) which were causing significant delay. [Bug 1268]. Corrected bugs in read FIFO incr./decr code. Addition of diagnostics messages for FIFO levels and HWM [West 1 issues]. Correction of the Accelerator Keys for "&Parser Trace" and "&Debug". Fix for right click menu position in both main and dialog panels.
3.05.41	15/05/2009	Version 3.05.41, dated: 15/05/09 Corrections for XP/XR & TB (very old V2 applcore issues). Correction for problem noted in MM and EQ, (EQ 0 should not immediately exit). Another correction for EQ issues related to correct number of ApplCore panels running.

3.05.40	29/03/2009	Corrections for MR & MS, very old V2 applcore issues. Correction for QD parameter crashing bug, which did not get included in the version 3.05.37. Diagnostic startup "Time stamping" disabled by setting ApplCore.ini file entry default value to zero.
3.05.39	16/03/2009	3.05.39 02/04/09 Keyboard KR corrections. Various Fn key mappings added as in V2 (several were missing, note: F1 emits 146., F12 157.). Added third parameter to "EM" command, if value is 1, then puts up a MessageBox() and halts execution until "OK", a va
3.05.38	06/02/2009	Default Workstation number no longer set to -1 in CSI.ini, rather set to a value of 999 (MAX_ID).
3.05.37	02/02/2009	Implementation of a "resource management" scheme for EK & EQ. Start-up delays 500ms then runs a through the Z order of ApplCore panels - limiting to ApplCore.ini: MaxExecWindows, default is: 24. Correction for QD parameter crashing bug. Correction for CF and <TAB> control issues not fully sorted in version 3.05.35
3.05.36	12/01/2009	Enhancement to XR/XU commands to be like the GR/IR/RR commands, i.e. added additional parameters [<slotvar%>;] ['ADD']. Additionally, limit debug msgs in LG cmd. on LB_ERR returned by LB_GETCURSEL.
3.05.35	30/12/2008	Bug fix for CF command not functioning since version 3.03.14
3.05.34	16/12/2008	Correction for problems with Keyboard input and arrow/pg up/down key controls.
3.05.33	22/10/2008	fix to CN command allows max character strings to be passed from data bases. Was previously limited to 20 - was 20 in V2ApplCore. Correction for ApplCore.INI file vars: "DebugMode" and "VideoCard" are not set to OBSOLETE_PARAMETER if they do not exist in
3.05.32	26/09/2008	3.05.32 26/09/08 Bug fix to XD command to allow one character strings to be passed.
3.05.31	21/09/2008	Bug fix to CD and CE commands for when Min/Max range not specified. Bug fix to JD command to handle '-'.
3.05.30	05/09/2008	Added INI file parameter to optionally cycle Amazon UMD protocol.
3.05.29	25/08/2008	Modified XR command to use normal SendMessage first and then connect again using 32 bit direct This overcomes the problem whereby broadcasting 32 bit messages causes problems in some MS applications like Outlook
3.05.28	17/08/2008	Removed old 16 bit ApplCore.def file Added Amazon UMD protocol from V2 Modified XC, XS and XW commands to have optional parameter to set paramters without network update iInterceptData changed to be long int Changed UMDProtocol config entry to use acPath[] instead of APPLCORE.INI
3.05.27	14/07/2008	Correction for LG command problem paired with the issue in BNCS_CC17.DLL. The correction in version 3.01.24 31/10/05 was incomplete.
3.05.26	13/05/2008	Added support of navigation from V2ApplCore (2.05.15 17/03/06) Modification to KR handler to allow navigation keys to return string data. Merge of minor DIRECTV changes.
3.05.25	25/03/2008	Disabled "timestamping" dbwin messages.

3.05.24	10/03/2008	Adding VE command support for reading the INI file at each VE execution. Correction for LR command. Enhancement to LF command '0,1,2,' added 3,4,5 (same as 0,1,2 but no-case compare).. Correction for SA command - SA always does an upcase on the workstation/application name before it is returned.
3.05.23	02/01/2008	Correction for WAKEUP & SLEEP JS and also the start up "wakeup" first time run problem corrected in V1/V2. (folded in to this version from 2.05.20 05/12/06 Fix to prevent WAKEUP stringtable being called on application start).
3.05.22	29/08/2007	Correction for IR not functioning with A% as the index target var.
3.05.21	29/06/2007	Correction for problem identified in UMD system with UE & UP command.
3.05.20	26/06/2007	Correction to V3 ApplCore startup when CSI is not in Client Mode.
3.05.19	18/06/2007	Correction for another TS command bug.
3.05.18	31/05/2007	Enhancement of the CD/CE command per request. Added range parameter. See new V3 ApplCore documentation. TS & SL cmd bugs fixes. RN correction -- 128 name limit changed to 256.
3.05.17	23/05/2007	fix for another bug in EV command.
3.05.16	22/05/2007	bug fix in EV command.
3.05.15	02/05/2007	bug fix in ED command
3.05.14	01/05/2007	bug fix for WAKEUP & SLEEP JS in wakeup lines causing the following lines to not execute.
3.05.13	30/04/2007	ET 1 starts up with T\$ and U\$ set to current time.
3.05.11	26/04/2007	Added the menu and .INI file entries for all of the new ED modes. Bug fix to SI command for midnight wrap around. Corrected several missing changes to support alternate BNCS System Directory.
3.05.10	05/04/2007	Correction for initial state supporting the ED command, i.e. if debug state is set in the INI file.
3.05.09	02/04/2007	Added ED support modes: 0/1 for Variables, 2/3 for Debug, 4/5 for OutputDebugString, 6/7 for Parser Trace, 8/9 for Diagnostics. Correction of debug window scrolling memory over-write problem reported corrupted data in InfoDriver polling.
3.05.08	21/03/2007	Changes to printer support. Printer support on some machines causes a slow start up. PrinterSupport =0 in Applcore.ini disables all printer support.Changes to the ParserTrace -- outputs all info to OutputDebugString() also.
3.05.07	16/03/2007	Completed implementation of changes to the EC command to support request. Added EC 4 and EC 5 requested function. Changes the AP command added line number vector for file missing. XT command added back into V3. Was removed during the V2 to V3 port.
3.05.05	02/03/2007	Change the EV command - Recommendation from Simon is to raise STM_MAXMEMORIES to 32769 and change EXECVARTEMPMEM from 4097 to 32768.
3.05.04	27/02/2007	SYSTEMDELAY was 50 changed to 5 to improve performance - Simon recommended change.
3.05.03	27/02/2007	Bugfix for the SC command -- memory over write bug
3.05.01	06/02/2007	Allows parser string to go to the debug stream without diagnostics being turned on.

3.04.00	17/01/2007	Bug fix QH command to filter out of range parameter numbers. Modified QH command to return backpanel window handle if panel '0' specified. Modified PM command to allow backpanel to be moved if panel '0' specified. Implemented PZ command (Panel siZe) so that panels can be sizes as well as moved. Implemented QP command (Query Parent). QP A% B% returns the parent window of the Hwnd from A% into B%. Implemented QD command (Query Desktop). QD X% Y% returns desktop screen width, height into X% and Y%. SC modified to not lower the case of the format string. Bug fix to TS command to prevent garbage appearing in string memories when NULL string supplied SE modified to convert search string to uppercase. Bug fix to PZ and TS commands. Implemented QC command (Query Child) QC A% B% C% returns handle of control B% on panel A% in C%.
3.03.20	14/01/2007	Bug fix to prevent exception when JD parameters contain NULL strings
3.03.19	09/01/2007	Added QH command to query dialog handle
3.03.18	02/01/2007	Control external intercept modified to uses 32 bit messaging for GRDs & GPIDs
3.03.17	29/12/2006	Bug fix to correct fifo revertive handling for control externals WAKEUP stringtable no longer called if application closing down
3.03.16	24/11/2006	Bug fix to ensure EA command compatibility prior to 3.03.11
3.03.15	21/11/2006	Entry in APPLCORE.INI if enables allows System Menu exit
3.03.14	13/11/2006	Pause on left mouse click added to diagnostic window.
3.03.13	10/11/2006	Restored functionality to stringtable editor
3.03.12	03/10/2006	Fix to JD command. EA command no longer converts ';' to ':' when in 'no translate' mode. EV now parses a line rather than a single command.
3.03.11	01/09/2006	Maximum string sized in creased from 128 to 256 in ACXexec_ExecApplic(). Added third parameter to the EA command for optional translation of the escape character. >EX command no longer returns !Error!. Added 'String Compose' command
3.03.10	19/07/2006	Bug fix to SI command
3.03.09	12/07/2006	Bug fix to prevent ODS call with string length greater than 256 bringing up message box
3.03.08	18/06/2006	Revertives as a result of XR command now passed through fifo Can now scroll back debug window
3.03.07	16/06/2006	This corrects the XR bug
3.03.06	31/05/2006	Added SI command
3.03.05	19/05/2006	Extra parameter added to EA command to optionally inhibit error box on launch failure
3.03.04	11/05/2006	EG command now appends '0' as a default offset
3.03.03	10/05/2006	Increased acTrace to be MAXSTRINGSIZE*2 in GetArguments()
3.03.02	12/04/2006	Fix to EF command
3.03.01	10/03/2006	Modification to XR command to use BBC_CTRLCONNECT32. Modification to ACDxSup_Disconnect to use BBC_CTRLDISCONNECT32. MAX_EX_DATASIZE set to 1025. GetConfigDirectory() now GetBNCSConfigDirectory(). Added GetBNCSSystemDirectory to get path to DLLs for LoadLibrary() to use.
3.03.00	27/02/2006	Now uses GetConfigDirectory

3.03.00	28/02/2006	Modified EA command to use CreateProcess() and enhanced error message reporting
3.03.00	01/03/2006	Added Exec eExecute command
3.02.12	03/03/2006	The following new commands have been added to V3 Applcore recently: Device Mode, List Null, List Trim, List Vertical, String Hex and Text Break. The following has been modified: Exec Reboot. See the V3 Applcore Command Ref for full details.
3.02.12	10/02/2006	Added Text Break command
3.02.11	06/02/2006	Completed initial video overlay support in Video Enable and Video Disable commands
3.02.10	24/01/2006	Bug fix to ACXumd_UmdPutTslVirtual() & ACXumd_UmdPutProbel()
3.02.09	24/01/2006	Bug fix to ACXumd_UmdPutTslNormal() Began implementing 32 bit video overlay support
3.02.08	13/01/2006	Bug fix to new BBC_DATABASECHANGE handler
3.02.07	11/01/2006	BBC_DATABASECHANGE, BBC_NETTXRXSTATUS, BBC_DEVSTATUSMSG now processed via FIFO
3.02.06	06/01/2006	Correction for ER - reboot problems on NT/XP - now calls AdjustTokenPrivileges(), etc. to get priv. on NT systems.
3.02.05	21/12/2005	Added List Null command
3.02.04	12/12/2005	Modification to ER command to ensure Win9x and NTx compatibility
3.02.03	12/12/2005	Bug fix to filter out listbox notifications when receiving focus. Bug fix to ACXroute_RouterUnregister() default Min Max range is appended.
3.02.02	09/12/2005	Masked wParam in ACPanel_DialogProc() to filter off HIWORD Paint problem fixed in diagnostics window EA Command : SendMessage() changed to BNCS32SendMessage() ER command modified to include LOGOFF and KILLCSI options
3.02.01	06/12/2005	Uncommented-out the DestroyWindow, in ACPanel_Destroy() so that BNCS_CCxx libraries are notified that ApplCore is shutting down the window of interest -- allows the DLL's to deregister / clean up.
3.02.00	23/11/2005	Added resilience message handlers. Added Device Mode command. Added Exec Status command Function prototypes for WndProcs changed
3.01.30	06/12/2005	Uncommented-out the DestroyWindow, in ACPanel_Destroy() so that BNCS_CCxx libraries are notified that ApplCore is shutting down the window of interest -- allows the DLL's to deregister / clean up.
3.01.29	21/11/2005	Added String Hex command 22/11/05 'iUmdSize' entry renamed 'UmdSize' Now closes down if no STARTUP line found when error boxed is acknowledged Trace now expands variables.
3.01.28	17/11/2005	Added comms support library to allow re-enabling of UMD commands
3.01.27	15/11/2005	Correction to ACPanel_DialogProc() function prototype.
3.01.26	02/11/2005	Correction for IW broken.
3.01.25	01/11/2005	Correction for handling "--" lines. "--" lines are used because "" lines are not allowed in the 32bit Borland env.(creates defective .RES file) and MSFT doesn't like such either.
3.01.24	31/10/2005	Correction for EF command problem - unwound above "fix". It was incorrect. Correction for LG command problem

3.01.23	28/10/2005	Correction for EF problem introduced in V3 (corrected using V2 code).
3.01.22	19/10/2005	IP/IR/IW now filter for device > 999 (MAX_DEVICES).Corrected "EQ" command problems. Extensive debugging on Win98 with MSFN VC++. Re-enabled Debug dialog for release ApplCore version. Correction for MA command problem reported by Paul R.
3.01.22	24/10/2005	Added LT & LV commands from Simons V2 source code.
3.01.21	29/09/2005	Added debug text format printing
3.01.20	28/09/2005	Correction to ACSTMSup_SetMem() routine. Only free memory not NULL memory pointers. Disable Debug window on Win98/Win95 systems.
3.01.19	26/09/2005	Enhancement to unknown command pop-up
3.01.18	15/09/2005	Improved trapping algorithm. Addition of Min/Max to ML function
3.01.17	13/09/2005	Trapping on bad strings data ala Paul R's GFX_1.rc
3.01.16	13/09/2005	Added support for detecting defective strings.
3.01.15	12/09/2005	Changed GlobalAlloc()'s to 256 or size + 256 buffer space.
3.01.14	08/09/2005	Corrections in Cbncs_strtok
3.01.13	02/09/2005	KEYREGTIMER handler reinstated in main window procedure. Cbncs_strtok class now used for stringtable parser
3.01.12	30/08/2005	Memory size changes
3.01.11	18/08/2005	Test version to localise ACSTMSup_SetMem() problem
3.01.10	16/08/2005	Now exits straight away if CSI is not present
3.01.09	15/08/2005	Bug fix to LS command
3.01.08	10/08/2005	Compiled with all debug options on
3.01.07	14/07/2005	Added String Environment command
3.01.06	13/07/2005	EQ 100 now always quits

AtoS IT Solutions and Services Limited
Faraday House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
Tel.: +44 (0)1276 696000