Removing the Configuration File from GSM (Windows)

1. Introduction

This document describes the changes required to GSM (Windows NT) to remove the requirement for the Global Configuration files on GSM (Windows NT) configurations. All the options currently held in the Configuration File will be replaced by new settings in the Global section of the registry. All these new "configuration file" registry settings are identified by an initial "+" character in the ValueName (e.g. +NumberOfFileChannels).

This document is intended for internal use to implement and maintain the necessary changes to the Steering Routine. In addition, it will be published externally as a White Paper. However, no technical details will be suppressed. External readers should skip over the very detailed technical information (normally identified by a smaller font), particularly Appendix A.

Removing the need for a configuration file (i.e. ++5661xJ or ++5663xJ for the Global Client; ++5669BB or ++5669CC for the Global Server) is desirable for a number of reasons:

- It simplifies the procedure used to add or amend screens and printers in a GSM (Windows NT) configuration. The existing procedure involves changes to BOTH the Global configuration file AND the Windows registry;

- Despite copious documentation there is still some confusion regarding the name and locations of the various configuration files present on a GSM (Windows NT) configuration;

- The Configuration File concept is one of many "non-Windows" technique required by users of GSM (Windows NT). Removing configuration files, and the use of CFUPDATE, will make GSM more Windows-compliant;

- The extra parameters required for the fully "open" filing system cannot be included in the Global configuration file. Removing the use of the configuration file is a necessary first step to the "open file" implementation;

- The requirement for the Global Server to read the GL-%-IPL.DLV file does not fit in well with the Windows "service" paradigm. Removing the use of the configuration file is a necessary first step to convert GLSERVER.EXE to a Windows NT service;

- The Global configuration file is only Global file required by the GSM (Windows NT) loader mechanism. Removing the configuration file should simplify the GSM (Windows NT) loader.

However, the mixed configuration-file/registry customisation mechanism for GSM (Windows NT) does include several important advantages over a registry-only mechanism:

- The configuration file can be easily renamed in a situation where a "safe", backup alternative to a live, experimental configuration is required. Although this can be currently emulated using Windows REGEDIT to save/restore the various registry keys & valuenames, changes to GLREGED.EXE will be required in the medium-term;

- Some complex configurations include two, or more, alternative configuration files. For example, the ++5699*xx* configuration file for server "A" on GL-A-IPL.DLV may be different from the ++5669*xx* configuration file for server "B" on GL-B-IPL.DLV. Similarly, the various Global Clients in an SMC configuration may use the /IN=*config_file_name* command line option to select a particular configuration file. Any selection of alternative configurations in the Global registry will require extensive changes to GLREGED.EXE, GLOBAL.EXE and GLSERVER.EXE;

- The Customisation Configuration File option of $CUS (i.e. $CUSA) patches the configuration file directly. This option should detect that the Global format configuration file is not being used and should display an appropriate warning message;

- Global Configurator (i.e. CFUPDATE) includes some "intelligence" and a number of useful shortcuts. For example, the default parameters for a new controller are taken from the previous controller of the same name (if available). Also, the special keystroke <CTRL B> can be used to quickly duplicate an existing controller. Neither of these techniques are currently available in GLREGED.EXE;

- Global Configurator (i.e. CFUPDATE) includes some rudimentary parameter validation at "compile time" (i.e. the 2nd phase of CFUPDATE). This logic must be moved to GLREGED.EXE and/or to the code in the Steering Routine that validates the new "configuration file" section of the registry.

Although the precise details are outside the scope of this document, a number of changes to GLREGED.EXE will be required to fully compensate for the removal of CFUPDATE (see IN202).

It is tempting to remove the Controller Number concept entirely (i.e. entries to the relevant C function could be simply plugged into the primary control blocks). However, certain sub-routines (e.g. DEVIN$) expect an essentially BOS-style DA/PA block, including the controller number information. Consequently, the Controller Number notion will be retained. Similarly, the Controller Index concept should be retained.

A primary aim of this design is to avoid any changes to executives or controllers thus minimising the risk of destabilising either GLOBAL.EXE or GLSERVER.EXE. The changes to the Steering Routine have been successfully implemented without ANY changes to executives, controllers or any of the primary control blocks.

This note only considers the changes to the GSM (Windows NT) nucleus. However, the same changes could be implemented for GSM (Unix) by changing GLCONFIG and the layout of the Shared Memory (SHM) data structures.

Although the configuration file is mainly used at GSM load time it is also used at install time and software generation time. The changes only affect the run-time use of the configuration file (i.e. a configuration file is still required during the software generation process and during software installation from BACRES). However, other changes to the Software Distribution and Installation Mechanism (see IN183) will remove both the generation and installation procedures entirely.

The 2 phase nature of CFUPDATE has been replaced by a combination of GLREGED.EXE and the extended functionality of the Steering Routine: The extra ValueNames included in the GLMACH.TLT file (see below) have resulted in GLREGED.EXE effectively replacing the Edit phase of CFUPDATE (i.e. $INST1). The extra functionality in the Steering Routine to generate internal control blocks dynamically has effectively replaced the Generate phase of CFUPDATE (i.e. $INST2).

1.1 Enabling the Configuration File removal

By default, for the current revision of GSM (Windows NT) (i.e. V2.9K) the option to ignore the configuration file will be **disabled**. This is to provide compatibility with all existing installations. To enable the new option (i.e. to ignore the configuration file) and to use the registry settings for **ALL** nucleus customisations for the Global Client (i.e. GLOBAL.EXE) the following registry setting must established:

..\Global\Client\UseConfigurationFile=Off

The default setting of the UseConfigurationFile ValueName is “On”.

To enable the new option (i.e. to ignore the configuration file) and to use the registry settings for **ALL** nucleus customisations for the Global Server (i.e. GLSERVER.EXE) one of the following registry settings must established:

..\Global\Servers\UseConfigurationFile=Off (all servers default)

or:

..\Global\Servers\%\UseConfigurationFile=Off (server specific)

(where % is the server letter between A and Z).

The server-specific option, if established, is used in preference to the “all servers” option, for the relevant server. This technique, of supplying an "all servers" default option in addition to a "server specific" option, is used throughout the design.

The default setting of the UseConfigurationFile ValueName is “On”.

1.2 Enabling diagnostics

The internal control block generation and emulation process, which effectively sythesises a Configuration File, is highly complex. Consequently, copious diagnostics are provided to troubleshoot problems. The diagnostic option, which generates a large (e.g. 100Kb) text-file in the Global “log” directory, is enabled by a number of registry settings. To enable diagnostics for the Global Client (i.e. GLOBAL.EXE) the following registry setting must established:

..\Global\Client\LogConfigurationSynthesis=On

When Client diagnostics are enabled the log-file “config.log” is produced.

To enable diagnostics for the Global Server (i.e. GLSERVER.EXE) one of the following registry settings must established:

..\Global\Servers\ LogConfigurationSynthesis=On (all servers default)

or:

..\Global\Servers\%\ LogConfigurationSynthesis=On (server specific)

(where % is the server letter between A and Z).

The server-specific option, if established, is used in preference to the “all servers” option, for the relevant server.

When Server diagnostics are enabled the log-file "config%.log" is produced (where % is the server letter between A and Z).

1.3 Compatibility

Although several new registry settings are available to fully emulate the Configuration File, a large amount of effort has been expended to provide sensible defaults for the "+" options that have not been specified in the registry. In some cases, the default merely involves a hard-coded "fall back" value. However, in other cases, particularly in the Direct Access controller section, entire registry structures are emulated based on the existing de facto registry settings (see section 3 for full details).

IN GENERAL, IT SHOULD BE POSSIBLE TO REMOVE THE USE OF THE CONFIGURATION FILE MERELY BY SETTING THE UseConfigurationFile SETTING TO "Off". NO BULK ADDITION OF "+" VALUENAMES SHOULD BE REQUIRED.

The individual sections (below) describe the default handling for all the new registry settings.

1.4 Error Handling

Both phases of CFUPDATE include several types of error checking and validation. Until more validation and error-checking is added to GLREGED.EXE all error checking must be performed at run-time by the Steering Routine as GLOBAL.EXE or GLSERVER.EXE are initialising. Any error condition will be signaled by the appearance of a Dialogue Box describing the precise details. In general, the Global Client or Global Server will continue to load after an error condition. However, all messages should be treated as errors (i.e. rather than warnings) and the Global Client or Global Server should be immediately closed to allow the registry to be updated to correct the error condition.

All error messages are written to the log-file (see section 1.2).

1.5 Document Structure

The remainder of this document describes the changes in complete detail. Each section corresponds to a section in the CFUPDATE dialogue. Note to external readers: CFUPDATE behaves slightly different when used with an "internal serial number". Some of the prompts and concepts described below are suppressed when running CFUPDATE on an "external serial number".

2. The MACHINE NAME & BOOTSTRAP MESSAGE section

The external dialogue in the MACHINE NAME section consists of a single prompt for the 15-character Machine Name, I0LDES (see Appendix A). For the Global Client, the Machine Name is emulated by the following setting:

..\Global\Client\Nucleus\+ConfigurationDescription

A default name of "Windows NT" is assumed.

This section of CFUPDATE also establishes the Machine Code ("W1"), Sub-Code (1), Architecture Code ("W") and default System Manager Code (4). All of these values are effectively hard-coded in the Steering Routine and cannot be altered by registry settings. Note that the System Manager code may be modified by the LAN Controller (see Appendix A). Note also that the Machine Code is set to "W2" when the Steering Routine is running within GLSERVER.EXE. This parameter is never used externally but must be set up internally to ensure the correct entries in the controller load table are used.

The internal fields that log the date, time and operator-id of the last update to the configuration file are not emulated. However, the Configuration File name, which appears in the $S report, is emulated by the following setting:

..\Global\Client\Nucleus\+SynthesizedConfigurationFileName

A default name of "++566SYN" is assumed.

Note that an internal, undocumented, flag (SYBIF1) contains a bit that indicates whether the Configuration File was used when GSM was loaded (i.e. reflecting the setting of the UseConfigurationFile option). This flag, rather than the $$CONF System Variable will be used by $CUSA and $S to warn that the Global configuration file has been ignored.

Although the 400 character (i.e. 10 lines, each of 40 characters) bootstrap message is largely ignored on GSM (Windows NT) it will still appear if the "D" option is used as the $MONITOR "Target System". For completeness, the Bootstrap Message is emulated by the following settings:

..\Global\Client\Nucleus\+InitialMessage0
..\Global\Client\Nucleus\+InitialMessage1
..\Global\Client\Nucleus\+InitialMessage2
..\Global\Client\Nucleus\+InitialMessage3
..\Global\Client\Nucleus\+InitialMessage4
..\Global\Client\Nucleus\+InitialMessage5

..\Global\Client\Nucleus\+InitialMessage6
..\Global\Client\Nucleus\+InitialMessage7
..\Global\Client\Nucleus\+InitialMessage8
..\Global\Client\Nucleus\+InitialMessage9

The default for each setting is <blank>.

The first line of the Bootstrap Message is validated by CFUPDATE. However, no validation is performed by the Steering Routine. Consequently, the first text-string of the first line of the Bootstrap Message does not have to include the "BOS Level Number" (sic) keyword or the "LEVEL*n*" keyword. Since all GSM (Windows NT) configurations are level 9, this field (IGLEVN), which is only used at software generation and installation time (see Appendix A), does not have to be emulated.

Similarly, the "Maximum Number of Screens" value (i.e. the last text-string on the first line of the Bootstrap Message) which is used at software generation time and is moved to I0MSCR (see Appendix A) is ignored. This field is only used to prevent external users adding more screens to the configuration file than the run-time code allows.

Important Note: The removal of the Maximum Number of Screens option means that users can add more than the maximum number of permitted screens to the registry. This condition will no longer result in a configuration error (i.e. from CFUPDATE) but will result in a fatal run-time $57 INITIATION ERROR 363 or 369. All $57 errors and warnings should appear in a windows Dialogue Box and/or log-file (ref. W0-1093).

For the Global Server, GLSERVER.EXE, all of the above options (none of which, with the exception of the Machine Code, are actually required) are hard-coded in the Steering Routine and consequently no external options appear.

3. The DIRECT ACCESS CONTROLLER section

The current structure of the "Data" key of the registry is not readily expandable. No drive-specific keys exist beneath the main "DiscreteDataFiles", "IntegratedDataFiles" and "RAMDisk" keys. The "Diskette" entry in the registry is a ValueName not a key. Within this structure any expansion can only be achieved by adding further single-level ValueNames with the drive number tagged on the end (e.g. DDF0, DDF1 etc.). Unlike the somewhat milder issues with the "Screens" and "Printers" keys (see sections 4 and 5) the restrictions in the current registry structure for the "Data" key cannot be avoided by performing a multi-pass analysis on the registry entries. A fundamental restructure of the registry entries beneath the "Data" key is required is allow the configuration file to be correctly emulated

and, more importantly, to allow some of the more far-reaching File Executive changes to be implemented (see IN201).

However, the registry changes described in section 3.1 are NOT mandatory if the "UseConfigurationFile" option has been enabled. The rules described in section 3.2 create a new format "Data" registry key from the existing settings.

3.1 The existing and proposed "Data" registry key structure

This section considers the structures of each of the sub-keys below the "Data" key. Only the keys and ValueNames that are pertinent to the Steering Routine are described in this document. A number of low-level, controller-specific ValueNames, which will remain unaltered by the proposed changes, are ignored in this document.

Note for internal readers: The Drive% limits for the various controllers must agree with the corresponding "Maximum Drive Number" options in the A.W1 Action File.

3.1.1 The "IntegratedDataFiles" key

The "IntegratedDataFiles" key currently contains the following ValueNames:

IntegratedDataFiles\IDF%	(where % = 0 to 9)

An additional drive-number specific key, Drive% (where % = 0 to 9) is required, under which a number of drive-specific ValueNames can be added:

IntegratedDataFiles\Drive%\+DriveDescription	(where % = 0 to 9)
IntegratedDataFiles\Drive%\+VolumeFormat	(where % = 0 to 9)
IntegratedDataFiles\Drive%\+UnitNumber	(where % = 0 to 9)

Note that this registry structure enforces the rule that only a single IDF Volume Format can be associated with a single IDF drive-number.

3.1.2 The "DiscreteDataFiles" key

The "DiscreteDataFiles" key currently contains the following ValueNames:

DiscreteDataFiles\DDF%	(where % = 0 to 9)

An additional drive-number specific key, Drive% (where % = 0 to 9) is required, under which a number of drive-specific ValeuNames can be added:

DiscreteDataFiles\Drive%\+DriveDescription	(where % = 0 to 9)

DiscreteDataFiles\Drive%\+VolumeFormat	(where % = 0 to 9)
DiscreteDataFiles\Drive%\+UnitNumber	(where % = 0 to 9)
DiscreteDataFiles\Drive%\+NumberOfSubUnits	(where % = 0 to 9)
DiscreteDataFiles\Drive%\+MaximumNumberOfFiles	(where % = 0 to 9)

Note that this registry structure enforces the rule that only a single DDF Volume Format can be associated with a single DDF drive-number. Note also the need to specify both the Maximum Number of Sub-Units and the Maximum Number of Files.

3.1.3 The "Diskette" ValueName

The "Diskette" ValueName is a multi-value flag that can take the following values:

0	Diskette not supported
1 - 4	Diskette supported (with various controller-specific options)

A new "Diskettes" key, in addition to a sub-ordinate drive-number specific key,,Drive% (where % = 0 to 9), are required, under which a number of drive-specific ValueNames can be added:

Diskettes\Drive%\+DriveDescription	(where % = 0,1; $ = 0 to 9)
Diskettes\Drive%\+VolumeFormat$	(where % = 0,1; $ = 0 to 9)
Diskettes\Drive%\+UnitNumber$	(where % = 0,1; $ = 0 to 9)

Note the multiple ValueNames for the Volume Format and Unit Number settings which allow multiple diskette Volume Formats to be associated with a single Diskette drive-number.

3.1.4 The "RAMDisk" key

The highly-specialised "RAMDisk" key currently only contains a single controller-specific ValueName.

An additional drive-number specific key, Drive0, is required, under which a number of drive-specific ValueNames can be added:

RAMDisk\Drive0\+DriveDescription
RAMDisk\Drive0\+VolumeFormat
RAMDisk\Drive0\+UnitNumber

Note that this registry structure enforces the rule that only a single RAMDisk controller can be specified.

3.2 Emulating the new "Data" registry key structure

The new registry structure is only recognised for a particular controller if the corresponding new "Drive*N*" key is defined in the registry:

```
IntegratedDataFiles\Drive%          (% = 0 to 9)
DiscreteDataFiles\Drive%            (% = 0 to 9)
Diskettes\Drive%                    (% = 0 or 1)
RAMDisk\Drive0
```

It is possible to mix the new registry structure with the old registry structure for different types of controllers (e.g. IntegratedDataFiles\Drive0 present but DiscreteDataFiles\Drive0 absent) but this technique is not recommended.

If a "Drive%" key is defined then all the mandatory ValueNames beneath it (see below) **must** be defined otherwise the associated controller will be dynamically removed from the configuration. It is not possible to have "hanging" Drive keys.

If a "Drive%" key is not defined then the various controller settings are emulated according to the following rules by creating internal control blocks within the Steering Routine. **The registry is never updated by this process.**

3.2.1 Emulation Rules for the "IntegratedDataFiles" key

The processing for the "IntegratedDataFiles" key is as follows:

```
DO FOR % = 0 to 9                   * Consider all 10 drives
    IF IDF% key present THEN        * Ignore if IDF% is not defined
        IF Drive% key present THEN
            Process ValueNames          * See section 3.1.1
        ELSE                        * May need to emulate registry
            IF % = 0                * Only emulate Drive0
                Emulate Drive0      * See below
            ENDIF                   * Ignore all other drives
        ENDIF
    ENDIF
ENDDO
```

If the "UseConfigurationFile" option is enabled then for all drive numbers higher than 0, both the IDF% and a Drive% keys must be present in the registry. For drive 0, only the

IDF% key is mandatory. Note that the IDF% key is always mandatory anyway in order for the IDF controller to recognise the associated device.

If the Drive0 key is absent, a single IDF with the following parameters is emulated:

IntegratedDataFiles\Drive0\+DriveDescription	1st Integrated Data File
IntegratedDataFiles\Drive0\+VolumeFormat	Z151Z
IntegratedDataFiles\Drive0\+UnitNumber	110

3.2.2 Emulation Rules for the "DiscreteDataFiles" key

The processing for the "DiscreteDataFiles" key is as follows:

```
DO FOR % = 0 to 9                          * Consider all 10 drives
    IF DDF% key present THEN          * Ignore if DDF% is not defined
        IF Drive% key present THEN
            Process ValueNames              * See section 3.1.2
        ELSE                                * May need to emulate registry
            IF % = 0                        * Only emulate Drive0 or Drive1
                IF DDF1 defined         * Emulate 59 sub-volume Drive0
                    Emulate Drive0 with 59 sub-volumes
                ELSE                    * Emulate 99 sub-volume Drive0
                    Emulate Drive0 with 99 sub-volumes
                ENDIF
            ENDIF
            IF % = 1
                Emulate Drive0 with 39 sub-volumes
            ENDIF                       * Ignore all other drives
        ENDIF
    ENDIF
ENDDO
```

If the "UseConfigurationFile" option is enabled then for all drive numbers higher than 1, both the DDF% and a Drive% keys must be present in the registry. For drives 0 and 1, only the DDF% key is mandatory. Note that the DDF% key is always mandatory anyway in order for the DDF controller to recognise the associated device.

If the Drive0 key is absent, a single DDF with the following parameters is emulated if the DDF1 key is not defined:

DiscreteDataFiles\Drive0\+DriveDescription	1st Discrete Data File
DiscreteDataFiles\Drive0\+VolumeFormat	T259Z
DiscreteDataFiles\Drive0\+UnitNumber	200
DiscreteDataFiles\Drive0\+NumberOfSubUnits	99
DiscreteDataFiles\Drive0\+MaximumNumberOfFiles	250

If the Drive0 key is absent, a single DDF with the following parameters is emulated if the DDF1 key is defined:

DiscreteDataFiles\Drive0\+DriveDescription	1st Discrete Data File
DiscreteDataFiles\Drive0\+VolumeFormat	T259Z
DiscreteDataFiles\Drive0\+UnitNumber	200
DiscreteDataFiles\Drive0\+NumberOfSubUnits	59
DiscreteDataFiles\Drive0\+MaximumNumberOfFiles	250

If the Drive1 key is absent, a single DDF with the following parameters is emulated:

DiscreteDataFiles\Drive1\+DriveDescription	2nd Discrete Data File
DiscreteDataFiles\Drive1\+VolumeFormat	T259Z
DiscreteDataFiles\Drive1\+UnitNumber	260
DiscreteDataFiles\Drive1\+NumberOfSubUnits	39
DiscreteDataFiles\Drive1\+MaximumNumberOfFiles	250

3.2.3 Emulation Rules for the "Diskettes" key

The processing for the "Diskettes" key is as follows:

```
DO FOR % = 0 to 1                          * Consider 2 drives
    IF Diskette valuename nonzero THEN     * Diskette value must be > 0
        IF Drive% key present THEN
            Process ValueNames                 * See section 3.1.3
        ELSE                               * May need to emulate registry
            IF % = 0                       * Only emulate Drive0
                Emulate Drive0             * See below
            ENDIF                          * Ignore Drive1
        ENDIF
    ENDIF
ENDDO
```

If the "UseConfigurationFile" option is enabled then for drive 1, the "Diskette" ValueName must be non-zero and the Drive1 key must be present in the registry. For drive 0, only a

nonzero value for the "Diskette" ValueName is required. Note that the "Diskette" ValueName must be nonzero anyway in order for the Diskette controller to recognise the associated device.

If the Drive0 key is absent, a single Diskette drive with the following parameters is emulated:

Diskettes\Drive0\+DriveDescription	1st Diskette Drive (A:)
Diskettes\Drive0\+VolumeFormat0	O2A
Diskettes\Drive0\+UnitNumber0	140
Diskettes\Drive0\+VolumeFormat1	O2B
Diskettes\Drive0\+UnitNumber1	142
Diskettes\Drive0\+VolumeFormat2	B3B
Diskettes\Drive0\+UnitNumber2	170

3.2.4 Emulation Rules for the "RAMDisk" key

The processing for the "RAMDisk" key is as follows:

```
IF RAMDisk key present                  * RAMDisk key required
        IF Drive0 key present THEN
                Process ValueNames              * See section 3.1.4
        ELSE                            * Emulate registry
                Emulate Drive0          * See below
        ENDIF
ENDIF
```

If the "UseConfigurationFile" option is enabled then the "RAMDisk" key must be present. This key must be present anyway in order for the RAMDisk controller to recognise the associated device.

If the Drive0 key is absent, a single RAMDisk drive with the following parameters is emulated:

RAMDisk\Drive0\+DriveDescription	Windows NT RAM Disk
RAMDisk\Drive0\+VolumeFormat	P00A
RAMDisk\Drive0\+UnitNumber	109

3.3 Validating the new "Data" registry key structure

CFUPDATE includes a considerable amount of logic to process and validate the Direct Access Devices section of the Configuration File. Most of this logic must be emulated in

the Steering Routine. However, there are some important differences between CFUPDATE and the Steering Routine emulation.

The most important difference concerns the use of the ISAM Volume Types Action File, A.VTYPE. This file contains definitions of all the Volume Types that are available for inclusion in a Configuration File. The A.VTYPE Action File is architecture independent (unlike A.W0 and A.W1) to ensure portability between all the various GSM implementations. Although an early design did involve the creation of a Windows data file derived from the Global A.VTYPE file, the final design is far simpler and involves details of the following volume formats hard-coded into the Steering Routine:

Volume type	*Associated controller*
O2A	Diskettes
O2D	Diskettes
O2B	Diskettes
B3B	Diskettes
T259Z	DDF
Z151Z	IDF
P00A	RAMDisk

NO OTHER VOLUME FORMATS CAN BE EMULATED UNLESS A NEW GLOBAL.EXE, OR GLSERVER.EXE, MODULE IS RECOMPILED.

A slightly more subtle issue also involves the A.VTYPE Action File. The default Format Byte and Access Option parameters in the A.VTYPE Action File can be over-ridden by options in the A.W1 Action File. Again, a very simple approach has been taken (since formatting is not supported by any GSM (Windows NT) controllers; and the "Access Option" concept disappeared with the withdrawal of 8" diskettes !!!). These parameters are hard-coded, on a per Volume Type basis, in the Steering Routine.

Finally, the CFUPDATE prompts that appear to allow the size of an IDF Discrete Logical Volume (DLV) to be specified are not emulated since the IDF controller always dynamically resizes the volume according to the size of the Windows .DLV file.

None of the above differences between the Steering Routine and CFUPDATE are expected to cause any run-time problems. However, in order to prevent invalid configurations and run-time errors the Steering Routine must include a certain amount of the validation logic that is currently within CFUPDATE.

The following sections describe the specific validation for each controller type.

3.3.1 Validating the "IntegratedDataFiles" key

No validation is performed on the +DriveDescription key. However, this string will be truncated to 25 characters. The default Drive Description is "1st Integrated Data File", "2nd Integrated Data File" etc.

The only allowed "+VolumeFormat" is Z151Z (see above).

The "+UnitNumber" must be in the range 104 to 199. The default unit for drive 0 is 110 incrementing by 1 for each drive number.

Validation is performed to ensure that the Unit Number does not clash with any other Direct Access devices.

3.3.2 Validating the "DiscreteDataFiles" key

No validation is performed on the +DriveDescription key. However, this string will be truncated to 25 characters. The default Drive Description is "1st Discrete Data File", "2nd Discrete Data File" etc.

The only allowed "+VolumeFormat" is T259Z (see above).

The "+UnitNumber" of the domain must be in the range 110 to 299 (note that during implementation this minimum domain unit number was reduced from 120 to 110). The default unit for drive 0 is 200. The Unit Numbers of all sub-volumes must be less than 299.

Validation is performed to ensure that the Unit Number of the domain, and all sub-volumes, do not clash with any other Direct Access devices.

The "+MaximumNumberOfFiles" must be either 99 or 250. No other values are allowed. The default is 250.

The "+NumberOfSubUnits" must be between 1 and 99. The default number of sub-volumes for Drive0 is 99, unless Drive1 is present in which case the default is 59. The default number of sub-volumes for Drive1 is 39. The default number of sub-volumes for all other drives is 29.

Important note: A further validation included in CFUPDATE is also performed by the Steering Routine: The check to limit the number of sub-volumes to between 1 and two less than the number of files in the directory is performed. In practice this test ensures that the

maximum number of sub-volumes is 97 for a 99-file per directory domain. Note that the theoretical maximum number of sub-volumes for a 250-file per directory domain (i.e. 249) exceeds the practical maximum of 99 anyway.

3.3.3 Validating for the "Diskettes" key

No validation is performed on the +DriveDescription key. However, this string will be truncated to 25 characters. The default Drive Description for Drive0 is "1st Diskette Drive (A:)". The default Drive Description for Drive1 is "2nd Diskette Drive (B:)".

The only allowed "+VolumeFormat" settings are O2A ,O2D, O2B and B3B (see above).

The "+UnitNumber" must be in the range 104 to 199. The default units are as follows:

Volume Type	*Default unit for Drive0*	*Default unit for Drive 1*
O2A	140	141
O2B	142	143
O2D	144	145
B3B	170	171

Validation is performed to ensure that the Unit Number does not clash with any other Direct Access devices.

Neither CFUPDATE nor the Steering Routine prevent a Volume Format from being defined more than once for a particular diskette drive.

3.3.4 Validating the "RAMDisk" key

No validation is performed on the +DriveDescription key. However, this string will be truncated to 25 characters. The default Drive Description is "Windows NT RAM Disk".

The only allowed "+VolumeFormat" is P00A (see above).

The "+UnitNumber" must be in the range 104 to 199. The default unit is 109.

Validation is performed to ensure that the Unit Number does not clash with any other Direct Access devices.

3.4 Processing the new "Data" registry key structure

After the various registry keys have been validated, the control block generation phase of CFUPDATE is emulated by the GSM (Windows NT) Steering Routine:

- A dummy AZ structure is allocated;
- The I0DEXC[1] and I0LEXC[1] fields in the Executive Array are set to the address of the AZ structure and the length of the AZ structure, respectively;
- The fields in the AZ structure are initialised as follows:

AZNDAS	Number of DA-blocks allocated (see below)
AZADA	Pointer to the first DA-block;
AZSBUF	Maximum sector size across all the volumes. This value is also moved to I0MAXS (see Appendix A);
AZNBUF	Calculated from I0MAXF using the following algorithm: AZNBUF = ((((I0MAXF + 2) * 8) - 1) / I0MAXS) + 2
AZNDBS	Copied from I0NBUF which is obtained from the +NumberOfFileBuffers registry setting (see Appendix A). Both I0NBUF and AZNDBS must be at least as high as AZNBUF;
AZNDCS	Copied from I0NCHA which is obtained from the +NumberOfFileChannels registry setting (see Appendix A). May be altered during the Console Controller validation (see section 4);
AZNDFS	Copied from I0NFIL which is obtained from the +NumberOfFileBlocks registry setting (see Appendix A). May be altered during the Console Controller validation (see section 4);
AZNLKS	Copied from BTLOCK which is obtained from the +NumberOfLockTableEntries registry setting (see Appendix A);
AZSLKS	Obtained from the +NumberOfSharedLockTableEntries registry setting (see Appendix A);

The AZNOPS, AZDOA and AZNSG fields are not required by the File Executive.

- The following fields in the BT/I0/IG-block are initialised as follows:

I0MAXF	Maximum number of files across all units (see Appendix-A);
I0MAXS	Maximum sector size across all the volumes (see Appendix A);
I0CBSZ	Maximum track size across all the volumes (see Appendix A);
I0NBUF	Obtained from the +NumberOfFileBuffers registry setting (see Appendix A);
I0NCHA	Obtained from the +NumberOfFileChannels registry setting (see Appendix A). May be altered during the Console Controller validation (see section 4);
I0NFIL	Obtained from the +NumberOfFileBlocks registry setting (see Appendix A). May be altered during the Console Controller validation (see section 4);
BTLOCK	Obtained from the +NumberOfLockTableEntries registry setting (see Appendix A);

- A number of DA structures are allocated, one DA-block per unit number defined in the registry. For the "IDF", "Diskette" and "RAMDisk" controllers only a single DA-block is allocated per unit.

  For the "DDF" controller *N*+2 DA-blocks are allocated per unit (where *N* is the number of sub-volumes). One of the extra DA-blocks is for the domain itself. The other extra DA-block is for a notional Bad-Track table (which is not actually used by the GSM (Windows NT) File Executive);

- A DV-block is allocated for every Volume Format defined in the registry. Unlike CFUPDATE, no attempt is made to optimise the control block allocation by sharing a single DV-block between multiple diskette volume types on different drives. Note that

it is essential that a separate DV-block is allocated for each DDF and IDF controller (because the controller may dynamically update the DVBYTE field);

- A DP-block is allocated for every unique drive/controller-code combination (i.e. a single DP-block is allocated for multiple devices on the same Diskette controller). Note that a DP-block extension is not required.

  The byte immediately before the DP-block contains the length of the Drive Description (up to 25 characters). The Drive Description text is held immediately before the length byte so that the last character of the text occupies the byte position immediately before the length field.

Offset from DP-block	*Description*
0	Start of DP-block
-1	Length of Drive Description text, set to *N*
-(1+*N*)	Start of Printer Description text, for *N* characters

- The following example illustrates the DA, DV, DP-block allocation for the following devices:

Controller	*Drive number*	*Volume Types*	*Unit Number*	*Sub-units*
DDF	0	T259Z	200	49
DDF	1	T259Z	250	49
IDF	0	Z151Z	110	
IDF	1	Z151Z	111	
Diskette	0	O2A	140	
		O2B	142	
		B3B	170	
Diskette	1	O2A	141	
		O2B	143	
		B3B	171	
RAMDisk	0	P00A	109	

A total of 111 DA-blocks will be allocated (i.e. one for each of the 9 discrete units; and 51 for each of the domains). This DA-block allocation is identical to the DA-block allocation within CFUPDATE.

A total of 7 DP-blocks will be allocated (i.e. 1 for each unique drive/controller-code combination). This DP-block allocation is identical to the DP-block allocation within CFUPDATE.

A total of 11 DV-blocks will be allocated (i.e. one for each volume type). This DV-block allocation is different from the DV-block allocation within CFUPDATE, which would only generate 8 DV-blocks (the 3 DV-blocks for the diskette volumes would be shared between each drive).

- The various control blocks are initialised as follows:

Field	*T259Z*	*Z151Z*	*P00A*	*O2A*	*O2B*	*B3B*	*O2D*
DADEV	Note-1	Note-1	Note-1	Note-1	Note-1	Note-1	Note-1
DACLA	Note-2	#00	#00	#00	#00	#00	#00
DAADV	DV-block	DV-block	DV-block	DV-block	DV-block	DV-block	DV-block
DAADP	DP-block	DP-block	DP-block	DP-block	DP-block	DP-block	DP-block
DAFLAG	#40	#40	#40	#40	#40	#40	#40
DAACO	#01	#01	#01	#01	#01	#01	#01
DASTA	#00008000	#00001000	#00000000	#00002400	#00002400	#00002400	#00002400
DAFIL	Note-3	#003FC200	#0000EC00	#00165000	#00162E00	#000B1000	#00165000
DADRIV	Note-4	Note-4	#00	Note-4	Note-4	Note-4	Note-4
DASINT	#00	#00	#00	#00	#00	#00	#00
DPCONT	#0000	#0000	#0000	#0000	#0000	#0000	#0000
DPDEVA	Note-5	Note-5	#0000	Note-5	Note-5	Note-5	Note-5
DPVECT	#0000	#0000	#0000	#0000	#0000	#0000	#0000
DPCNCD	#03	#02	#04	#01	#01	#01	#01
DPMULT	#01	#01	#01	#01	#01	#01	#01
DPCAL	#01	#01	#01	#01	#01	#01	#01
DPRTRY	#03	#03	#07	#01	#01	#01	#01
DPRTCT	#0000	#0000	#0000	#0000	#0000	#0000	#0000
DPRCCT	#0000	#0000	#0000	#0000	#0000	#0000	#0000
DPREM	#01	#00	#00	#00	#00	#00	#00
DPODD	#00	#00	#00	#01	#01	#01	#01
DPLOCK	#00	#00	#00	#00	#00	#00	#00
DPEXTZ	#00	#00	#00	#00	#00	#00	#00
DPEXT	None	None	None	None	None	None	None
DVVTYP	#1E	#1C	#1E	#12	#17	#10	#12
DVLTYP	#00	#00	#00	#00	#00	#00	#00
DVLPS	#0B	#0B	#05	#0B	#0B	#0B	#0B
DVMAXF	Note-6	#FA	#63	#3F	#FA	#3F	#3F
DVDIR	Note-7	#2E00	#1400	#0C00	#2E00	#0C00	#0C00
DVSEC	#0200	#0200	#0100	#0200	#0200	#0200	#0200
DVSPT	#40	#08	#40	#12	#12	#09	#12
DVHPC	#04	#02	#01	#02	#02	#02	#02
DVBYTE	#02000000	#00400000	#00010000	#00168000	#00168000	#000B4000	#01680000

DVACO	#01	#01	#01	#01	#01	#01	#01
DVDIV	#09	#09	#08	#09	#09	#09	#09
DVDENS	#00	#00	#00	#10	#10	#10	#10
DVFORM	#80	#80	#80	#80	#80	#80	#80
DVDESC	"T259Z"	"Z151Z"	"P00A"	"O2A"	"O2B"	"B3B"	"O2D"

Note-1 DADEV is set to 100 less than the unit number (e.g. DADEV for discrete unit 110 is #0A; DADEV for domain unit 200 is #64; DADEV for sub-volume 201 is #65 etc.)

Note-2 For a domain DA-block DACLA is set to the number of sub-volumes multiplied by -1 (e.g. #C5 for a 59 sub-volume domain). For a sub-volume DA-block, DACLA is set to the sub-volume number (e.g. #01 for unit 201). The structure of the DA-block for the Bad Track Table entry is outside the scope of this document.

Note-3 The value of DAFIL is calculated from the following formula:

DVBYTE = DAFIL + DASTA + DVDIR + (DVSEC * DVSPT * 6)

(i.e. #01FC6C00 for 99 file/directory domains; #01FC5200 for 250 file/directory domains).

Note-4 The byte-value DADRIV contains the drive number (from 0 to N)

Note-5 In the configuration file the word-value DPDEVA contains the drive number (from 0 to N). However, DPDEVA, DPCONT and DPVECT are ignored by the GSM (Windows NT) Direct Access controllers

Note-6 The value of DVMAXF for domains reflects the +MaximumNumberOfFiles registry setting (i.e. #63 or #FA)

Note-7 The value of DVDIR is calculated from the following formula:

DVDIR = ((DVMAXF / DVLPS) + 1) * DVSEC

(i.e. #1400 for 99 file/directory domains; #2E00 for 250 file/directory domains).

3.5 The Global Server (GLSERVER.EXE)

The general considerations described above for the "DiscreteDataFiles", "IntegratedDataFiles" and "Diskettes" keys apply to the Global Server, GLSERVER.EXE. Note that a "RAMDisk" cannot be configured on a Global Server.

Obviously, the technique of supplying an "all servers" default option in addition to a "server specific" option does NOT apply to the intrinsically server-specific Direct Access parameters.

4. The CONSOLE CONTROLLER section

The structure of the "Screens" key of the registry is fundamentally different from the CONSOLE CONTROLLER section of the Configuration File. The **order** of the various types of Console Controllers (i.e. GUI, NETWORK and SERIAL) in the configuration file determines the User Number allocation. [Note that this statement is strictly true for the "fixed" GUI and SERIAL controllers but is not entirely true for the current version of the NETWORK controller where a fixed User Number allocation is only possible by a kludge to the configured Port numbers. This issue with the NETWORK controller is being addressed - see IN193 for further details]. However, in the registry the Screen Controller name (i.e. GUI, Network, Serial) is the primary key with the notional "console index" as a secondary key within the controller type key.

Rather than change the hierarchy of the keys in the registry, which would involve concomitant changes to the various GSM (Windows NT) screen controllers, an optional "Console Index" ValueName has been defined and the Steering Routine processes the "Screens" key in three passes. The first and second passes simply build a list of screen/console definitions together with their associated Console Indexes, checking for duplicate Console Indexes. The final pass generates the required control blocks from the screen/console unit list and the various registry options.

The following options in the CONSOLE CONTROLLER section of the Configuration File must be emulated:

TYPE-AHEAD BUFFER LENGTH	CATALN
DISPLAY BUFFER LENGTH	CABLEN
FUNCTION KEY BUFFER LENGTH	CAFLEN
SCREEN IMAGE WIDTH	CAIMGW
SCREEN IMAGE DEPTH	CAIMGD
NUMBER OF STORED ATTR' BYTES	CAIMGA
NUMBER OF VIRTUAL PARTITIONS	CANCB
CHARACTER TRANSLATION ENABLED	CAXLAT
CONSOLE EXECUTIVE FLAG BYTE	CAFLAG2

Under every Screens Controller key:

..\Global\Client\Screens\GUI
..\Global\Client\Screens\Serial*nn*
..\Global\Client\Screens\Network*nn*

where *nn* is between 00 and 99 the following settings emulate the various Console Controller options:

..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+NumberOfPartitions
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+TypeAheadBufferLength
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+DisplayBufferLength
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+FunctionKeyBufferLength
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+ScreenImageWidth
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+ScreenImageDepth
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+NumberOfAttributeBytes
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+CharacterTranslation
..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+ConsoleExecFlagByte

where *xxxxxxxx* is either "Network", "Serial" or "GUI".

Note the asymmetry between the "Serial" and "Network" keys, both of which require a lower-level controller-specific index number; and the "GUI" key which is not followed by a subsidiary numeric key.

In addition, the following registry setting is available to define the Screen Index Number:

..\Global\Client\Screens*xxxxxxxx*\[*nn]*\+ScreenIndexNumber

Note the difference in scope between the new, generic Screen Index Number **ValueName** and the existing controller-specific index number **key** under the "Serial" and "Network" keys. The controller-specific index number only has a scope **within** the relevant controller key (i.e. "Serial" or Network"; whereas the Screen Index Number applies to **all** Screen controllers (i.e. "Serial", Network" and "GUI"). The Screen Index Number has been devised to allow total control over the order of the various Screen/Console controllers so that, for example, screens could be defined in the following order (Serial/02, Network/02, GUI, Network/01, Serial/01).

The Screen Index Number allocation is a two-pass process. In the first pass, all screens **with** an explicit +ScreenIndexNumber value are processed and allocated the relevant Index Number. Any duplicate Screen Index Numbers generate a warning dialogue box and the screen is ignored. In the second pass, all screens **without** an explicit +ScreenIndexNumber are processed and allocated the "next available" Index Number. In the second pass, the screens are considered in the following order:

..\Global\Client\Screens\GUI\
..\Global\Client\Screens\Network\01\
..\Global\Client\Screens\Network\02\

…
..\Global\Client\Screens\Network\99\
..\Global\Client\Screens\Serial\01\
..\Global\Client\Screens\Serial\02\
…
..\Global\Client\Screens\Serial\99\

For the “Network” and “Serial” controllers if any of the above “screen number specific” settings are not configured in the registry the equivalent “screen class” setting is used:

..\Global\Client\Screens*xxxxxxxx*\+NumberOfPartitions
..\Global\Client\Screens*xxxxxxxx*\+TypeAheadBufferLength
..\Global\Client\Screens*xxxxxxxx*\+DisplayBufferLength
..\Global\Client\Screens*xxxxxxxx*\+FunctionKeyBufferLength
..\Global\Client\Screens*xxxxxxxx*\+ScreenImageWidth
..\Global\Client\Screens*xxxxxxxx*\+ScreenImageDepth
..\Global\Client\Screens*xxxxxxxx*\+NumberOfAttributeBytes
..\Global\Client\Screens*xxxxxxxx*\+CharacterTranslation
..\Global\Client\Screens*xxxxxxxx*\+ConsoleExecFlagByte

where *xxxxxxxx* is either “Network” or “Serial”.

Obviously, a "screen class" +ScreenIndexNumber is meaningless.

If an option is not configured as either a “screen number specific” setting or a “screen class” setting the “generic screen” setting is used:

..\Global\Client\Screens\+NumberOfPartitions
..\Global\Client\Screens\+TypeAheadBufferLength
..\Global\Client\Screens\+DisplayBufferLength
..\Global\Client\Screens\+FunctionKeyBufferLength
..\Global\Client\Screens\+ScreenImageWidth
..\Global\Client\Screens\+ScreenImageDepth
..\Global\Client\Screens\+NumberOfAttributeBytes
..\Global\Client\Screens\+CharacterTranslation
..\Global\Client\Screens\+ConsoleExecFlagByte

Again, a generic +ScreenIndexNumber is meaningless.

If an option is not configured as either a "screen number specific" setting, a "screen class" setting or a "generic screen" setting, a default hard-coded in the Steering Routine is used.

In addition to the above "generic screen" settings the following registry setting is available:

..\Global\Client\Screens\+MaximumPartitions

This value must be set to either 99 or 250 (the default is 99).

The +MaximumPartitions registry setting is required to emulate a relatively new feature of Global Configurator. CFUPDATE includes two upper limits for the number of screens configured. The "Maximum Number of Partitions" (i.e. equivalent to the external User Number) is calculated by simply adding all the partitions configured for each console. The "Screen Number" (an internal field that is used by the Console Executive) is calculated by multiplying the Console Number by the Maximum Number of Partitions for any one screen. An example should make this clear:

Console Number	*Partitions*	*User Number(s)*	*Screen Number(s)*
1	4	1,2,3,4	1,2,3,4
2	1	5	9
3	8	6,7,8,9,10,11,12,13	17,18,19,20,21,22,23,24

For early versions of GSM V8.1 (i.e. before GSM V8.1j) the total number of partitions (i.e. the highest User Number) must less than, or equal to, 99. For GSM V8.1j, and later, this limit has been extended to 250 (for GSM (Windows NT) and GSM (Unix)). CFUPDATE displays an appropriate warning message if the configuration file exceeds the maximum User Number count.

This validation must be emulated by the Steering Routine. The +MaximumPartition setting allows the upper-limit (i.e. 99 or 250) to be specified. If this upper-limit is exceeded, a Warning Dialogue Box will appear and the number of partitions for the current screen/console will be reduced to avoid exceeding the limit. The number of partitions for all subsequent screens will be set to 0

The maximum Screen Number is hard-coded at 255 in both CFUPDATE and the Steering Routine. If this limit is exceeded, a Warning Dialogue Box will appear and the Number of Partitions for **ALL** screens will be set to 1.

The TYPE-AHEAD BUFFER LENGTH configuration file option is replaced by the +TypeAheadBufferLength registry setting (default 500). This setting is used to initialise CATALN.

The DISPLAY BUFFER LENGTH configuration file option is replaced by the +DisplayBufferLength registry setting (default 2000). This setting is used to initialise CABLEN.

The FUNCTION KEY BUFFER LENGTH configuration file option is replaced by the +FunctionKeyBufferLength registry setting (default 0). This setting is used to initialise CAFLEN.

The SCREEN IMAGE WIDTH configuration file option is replaced by the +ScreenImageWidth registry setting (default 132). This setting is used to initialise CAIMGW.

The SCREEN IMAGE DEPTH configuration file option is replaced by the +ScreenImageDepth registry setting (default 24). This setting is used to initialise CAIMGD.

The NUMBER OF STORED ATTR' BYTES configuration file option is replaced by the +NumberOfAttributeBytes registry setting (default 1). This setting is used to initialise CAIMGA.

The NUMBER OF VIRTUAL PARTITIONS configuration file option is replaced by the +NumberOfPartitions registry setting (default 4). This setting is used to initialise CANCB.

The CHARACTER TRANSLATION ENABLED configuration file option is replaced by the +CharacterTranslation registry setting (default "On"). This setting is used to initialise CAXLAT.

The CONSOLE EXECUTIVE FLAG BYTE configuration file option is replaced by the +ConsoleExecFlagByte registry setting (default 0). This setting is used to initialise CAFLAG2.

The following changes to the GSM (Windows NT) Steering Routine have been made to emulate the Console Controller section of the configuration file:

- A dummy BZ structure is allocated;

- The I0DEXC[2] and I0LEXC[2] fields in the Executive Array are set to the address of the BZ structure and the length of the BZ structure, respectively;

- A number of CA structures are allocated, one CA-block per screen defined in the registry. Note that that a CP block **is** required (see below). The BZCA pointer is set to the address of the first CA structure;

- The fields in the BZ structure are initialised as follows:

BZNCAS	Number of screens/consoles defined in the registry. This value is also moved to BTCSCR (see Appendix A);
BZCA	Pointer to the first CA-block;
BZNCBS	Total number of partitions. This value is also moved to BTNUSE (see Appendix A).

- The following fields in the BT/I0/IG-block are initialised as follows:

BTCSCR	See Appendix-A;
BTNUSE	See Appendix-A;
BTCPAR	This field is set to the highest partition number (see Appendix A).

- The following "File Executive" fields in the BT/I0/IG-block are validated:

I0NFIL	This value must be at least as high as: (SQR(BTNUSE * 64)) + 1 or the absolute value 30 (whichever is the highest). A Warning Dialogue Box will appear and I0NFIL (and AZNDFS) will be increased accordingly if the +ValidateNumberOfFileBlocks registry setting is set to "On" etc.
I0NCHA	This value must be at least as high as:

BTNUSE * 8

or the value of I0NFIL; or the absolute value 64 (whichever is the highest). A Warning Dialogue Box will appear and I0NCHA (and AZNDCS) will be increased accordingly if the +ValidateNumberOfFileChannels registry setting is set to "On" etc.

- In addition to the CA-block fields derived from the registry settings described above the following fields in the CA-block are also intialised by the Steering Routine:

CACNCD	Screen controller code: #01 GUI controller #02 Serial controller #03 Network controller
CADISP	Initialised to 0
CAXOFF	Initialised to 0
CANBLK	Initialised to 0
CALBLK	Initialised to 0
CALENG	Initialised to 128
CACP	Pointer to a 1 byte CP-block containing the value of the Screen Number key for "Serial" and "Network" controllers).

- A single-byte CP-block is required for both the "Serial" and "Network" screen controllers. This control block contains the value of the Screen Number key for that screen.

The CONSOLE controller section has no relevance for the Global Server configuration.

5. The TAPE CONTROLLER section

Although on some GSM architectures the Tape Controller TA-block is part of the configuration data, this is not the case for the GSM (Windows NT), or GSM (Unix),

architectures. The I0DEXC[3] and I0LEXC[3] fields in the Executive Array are NOT initialised by Global Configurator. The $TAPE TA-block is actually part of the DATA DIVISION of $TAPE.

Consequently, no changes to the GSM (Windows NT) registry have been made to emulate the Tape Controller section of the Global Client configuration file. The various options under the following registry key:

..\Global\Client\Tape

are used privately by the Tape Controller module.

Furthermore, no changes to the GSM (Windows NT) Steering Routine have been made to emulate the Tape Controller section of the Global Client configuration file.

The TAPE controller section has no relevance for the Global Server configuration.

6. The PRINTER CONTROLLER section

The structure of the "Printers" key of the registry is fundamentally different from the PRINTER CONTROLLER section of the Configuration File. In the Configuration File the "primary key" is the Printer Unit Number (e.g. 500, 501, etc.) with the Printer Controller (i.e. DOSPRINT, WINPRINT, $AUXPRINT) considered as a secondary option. However, in the registry the Printer Controller (i.e. DOSPrint, WinPrint, AuxPrint) is the primary key with the Unit Number (i.e. 500 to 599) as the secondary key. Rather than change the hierarchy of the keys in the registry, which would involve concomitant changes to the various GSM (Windows NT) printer controllers, the Steering Routine processes the "Printers" key in two passes. The first pass simply builds a list of printer units with their associated controllers checking for duplicate unit numbers in the process (i.e. duplicate Printer Unit numbers are detected at run-time). The second pass generates the required control blocks from the printer unit list and the various registry options.

The following options in the PRINTER CONTROLLER section of the Configuration File must be emulated:

HARDWARE FORMFEED	PAPSZ
PAGE DEPTH	PAPSZ
MAXIMUM PAGE WIDTH	PAWIDE
TIME-OUT IN TENS OF SECONDS	PADC2 (bottom 6 bits)
SPOOLER CONTROL BITS	PADC2 (top 2 bits)
PRINTER EXECUTIVE FLAG BYTE	PADC3

DEVICE CHARACTERISTICS	PADC1

In addition, the following options in the NUCLEUS OPTIONS section of the Configuration File pertain to printers:

NUMBER OF PRINT BUFFERS	I0NPB
LENGTH OF PRINT BUFFERS	I0SPB
NUMBER OF PRINT XLATION TABLES	I0NPT

The Nucleus options are emulated by the following settings:

..\Global\Client\Printers\+NumberOfPrinterBuffers
..\Global\Client\Printers\+LengthOfPrinterBuffers

The default Printer Buffer Length is 250. The default number of Printer Buffers is set to the actual number of printers configured. The Printer Executive always configures a Printer Translation Table for every printer so there is no need to configure this option in the registry.

Under every Printer Controller key:

..\Global\Client\Printers*xxxxxxxx**nnn*

where *xxxxxxxx* is one of "DOSPrint", "Winprint" or "AuxPrint" and *nnn* is between 500 and 599 the following settings emulate the various printer options:

..\Global\Client\Printers*xxxxxxxx**nnn*\+PrinterDescription
..\Global\Client\Printers*xxxxxxxx**nnn*\+HardwareFormFeed
..\Global\Client\Printers*xxxxxxxx**nnn*\+PageDepth
..\Global\Client\Printers*xxxxxxxx**nnn*\+MaximumPageWidth
..\Global\Client\Printers*xxxxxxxx**nnn*\+PrinterExecTimeout
..\Global\Client\Printers*xxxxxxxx**nnn*\+SpoolerControlBits
..\Global\Client\Printers*xxxxxxxx**nnn*\+PrinterExecFlagByte
..\Global\Client\Printers*xxxxxxxx**nnn*\+DeviceCharacteristics
..\Global\Client\Printers*xxxxxxxx**nnn*\+PrinterPool

Note that the +PrinterPool option (see below) only applies to the "DOSPrint" and "WinPrint" controllers, it does NOT apply to the "AuxPrint" controller.

If any of the above “printer number specific” settings is not configured in the registry the equivalent “printer class” setting is used:

..\Global\Client\Printers*xxxxxxxx*\+PrinterDescription
..\Global\Client\Printers*xxxxxxxx*\+HardwareFormFeed
..\Global\Client\Printers*xxxxxxxx*\+PageDepth
..\Global\Client\Printers*xxxxxxxx*\+MaximumPageWidth
..\Global\Client\Printers*xxxxxxxx*\+PrinterExecTimeout
..\Global\Client\Printers*xxxxxxxx*\+SpoolerControlBits
..\Global\Client\Printers*xxxxxxxx*\+PrinterExecFlagByte
..\Global\Client\Printers*xxxxxxxx*\+DeviceCharacteristics
..\Global\Client\Printers*xxxxxxxx*\+PrinterPool

Note that the +PrinterPool option (see below) only applies to the “DOSPrint” and “WinPrint” controllers, it does NOT apply to the “AuxPrint” controller.

If an option is not configured as either a “printer number specific” setting or a “printer class” setting the “generic printer” setting is used:

..\Global\Client\Printers\+HardwareFormFeed
..\Global\Client\Printers\+PageDepth
..\Global\Client\Printers\+MaximumPageWidth
..\Global\Client\Printers\+PrinterExecTimeout
..\Global\Client\Printers\+SpoolerControlBits
..\Global\Client\Printers\+PrinterExecFlagByte
..\Global\Client\Printers\+DeviceCharacteristics
..\Global\Client\Printers\+PrinterPool

Note the absence of a "generic printer" +PrinterDescription ValueName.

If an option is not configured as either a “printer number specific” setting, a “printer class” setting or a “generic printer” setting a default hard-coded in the Steering Routine is used.

The 25-character +PrinterDescription is moved to a data block that is allocated immediately before the generated PA-block (see below). This string is displayed by $U. The default +PrintDescription is either "DOS Printer", "Windows Printer" or "Auxiliary Printer" depending on the printer class.

The HARDWARE FORMFEED configuration file option is replaced by the +HardwareFormFeed registry setting. The PAGE DEPTH configuration file option is

replaced by the +PageDepth registry setting. If the +HardwareFormFeed setting is set to "On", "Yes" etc. (the default setting), a value of 0 is moved to PAPSZ and the +PageDepth setting is ignored. If the +HardwareFormFeed setting is set to "Off", "No" etc., the +PageDepth setting (default 66) is used to initialise PAPSZ. This processing illustrates one feature of CFUPDATE that cannot be emulated by GLREGED.EXE: The HARDWARE FORMFEED prompt is an "Auto Jump" prompt that causes CFUPDATE to avoid the PAGE DEPTH prompt if the reply is "Y".

The MAXIMUM PAGE WIDTH configuration file option is replaced by the +MaximumPageWidth registry setting (default 132). This setting is used to initialise PAWIDE.

The TIME-OUT IN TENS OF SECONDS configuration file option is replaced by the +PrinterExecTimeout registry setting (default 2). The SPOOLER CONTROL BITS configuration file setting option is replaced by the +SpoolerControlBits registry setting (default 0). These setting are combined to initialise the PADC2 flag (i.e. Spooler Control Bits in the top 2 bits; and Timeout value in the bottom 6 bits). Note that the value of the +SpoolerControlBits setting (i.e. 0, 1, 2 or 3) is shifted up by 6 bits and OR'ed with the +PrinterExecTimeout value, so that:

SPOOLER CONTROL BITS value	*Equivalent +SpoolerControlBits setting*
#00	0
#40	1
#80	2
#C0	3

The PRINTER EXECUTIVE FLAG BYTE configuration file option is replaced by the +PrinterExecFlagByte registry setting (default #FF). This setting is used to initialise PADC3.

The DEVICE CHARACTERISTICS configuration file option is replaced by the +DeviceCharacteristics registry setting (default 0). This setting is used to initialise the top 4 bits of PADC1. The +PrinterPool setting specifies the optional Printer Pool number. If this setting is specified the #08 bit of PADC1 is set and the Printer Pool number (between 0 and 7) is moved to the bottom 3 bits of PADC1. Note that the default value of the PADC1 field in the configuration file is #40, even though the #40 bit is not recognised by any of the GSM (Windows NT) printer controllers. However, the default value of the PADC1 flag when the configuration data is synthesized is #00.

The following changes to the GSM (Windows NT) Steering Routine have been made to emulate the Printer Controller section of the configuration file:

- A dummy EZ structure is allocated;
- The I0DEXC[5] and I0LEXC[5] fields in the Executive Array are set to the address of the EZ structure and the length of the EZ structure, respectively;
- A number of PA structures are allocated, one PA-block per printer defined in the registry. Note that that a PP block is not required (but see below). The EZPA pointer is set to the address of the first PA structure;
- The fields in the EZ structure are initialised as follows:

EZNPAS (EZNDEV)	Number of printers defined in the registry;
EZPA	Pointer to the first PA-block;
EZPB	Not initialised. This field is initialised by the Printer Executive to the first Printer Buffer (i.e. PB-block) initialised by the Printer Executive;
EZNPBS (EZNBUF)	Number of printer buffers. This field is set to the value of the +NumberOfPrinterBuffers registry setting (or to the actual number of printers configured, if this registry setting is not established). Note that the I0NPB field is also set to this value, for completeness only (see Appendix A);
EZBSZ	Length of the printer buffers. This field is derived from the value of the +LengthOfPrinterBuffers registry setting. If this registry setting is not established a default of 250 is used. The value must be between 132 and 254, inclusive. The value is also validated to be at least as high as the PAWIDE setting for each printer configured. The actual length is moved to the I0SPB field (see Appendix A). The EZBSZ field is derived from I0SPB by adding 5 to the length and rounding if necessary to obtain an even result.

Note that I0NPT is not initialised. It is not used by the Printer Executive.

EZODEV — Cleared to 0 by the Steering Routine. Initialised by the Printer Executive.

EZMAXB — Cleared to 0 by the Steering Routine. Initialised by the Printer Executive.

- In addition to the PA-block fields derived from the registry settings described above the following fields in the PA-block are also intialised by the Steering Routine:

  PACNCD — Printer controller code:

  #01 DOSPrint controller
  #02 WinPrint controller
  #50 AuxPrint controller

  PADEV — Printer unit number (minus 500)

  PAPP — Pointer to <NULL> PP-block (see below);

  PABAUD — Initialised to 0

  PAUSER — Initialised to #FF

- Although a PP-block is not required by any of the Printer Controllers (i.e. any controller specific information is read directly from the registry) the PAPP pointer must be established to provide a pointer to the Printer Description text. The baroque mechanism within CFUPDATE to allocate the Printer Description text is emulated:

  The area of memory pointed to by PAPP is used as follows:

Offset from PAPP	*Description*
0	Start of notional PP-block (not allocated)
-1	Length of PP-block (low), set to #00
-2	Length of PP-block (high), set to #00
-3	Length of Printer Description text, set to *N*
-(3+*N*)	Start of Printer Description text, for *N* characters

- All Printer Buffers and Printer Translation Tables are initialised by the Printer Executive.

The PRINTER controller section has no relevance for the Global Server configuration.

7. The LAN CONTROLLER section

The LAN Controller section of the configuration file is only partly used on the GSM (Windows NT) architecture. Although, unlike GSM (Unix) a "true" network interface has been implemented to connect the GSM (Windows NT) Global Client(s) to the Global Server(s), none of the low-level (i.e. Arcnet specific) options are required by the client-server interface. The following options in the LAN CONTROLLER section of the Global Client configuration file are all ignored:

POLLS TO WAIT BETWEEN ACCESSES
NORMAL TIMEOUT IN SECONDS
TIMEOUT ON 1ST ACCESS IN SECS
LAN EXECUTIVE RETRY COUNT
WAN MASTER NODE-ID
LAN EXECUTIVE FLAG BYTE

The following options in the NUCLEUS CONTROLLER section of the Global Client configuration file are also ignored:

NUMBER OF FULL LAN BUFFERS
NUMBER OF SHORT LAN BUFFERS

Similarly, the following options in the LAN CONTROLLER section of the Global Server configuration file are also ignored:

POLLS TO WAIT BETWEEN ACCESSES
NORMAL TIMEOUT IN SECONDS
TIMEOUT ON 1ST ACCESS IN SECS
LAN EXECUTIVE RETRY COUNT

The following options in the NUCLEUS CONTROLLER section of the Global Server configuration file are also ignored:

NUMBER OF FULL LAN BUFFERS
NUMBER OF SHORT LAN BUFFERS

Consequently, no changes to the GSM (Windows NT) registry have been made to emulate the LAN Controller section of the configuration file.

The following changes to the GSM (Windows NT) Steering Routine have been made to emulate the LAN Controller section of the configuration file:

- A dummy FZ structure is allocated. Note that the FZ op-code table is NOT moved or created by the Steering Routine - it is hard-coded in the LAN Executive;

- The I0DEXC[6] and I0LEXC[6] fields in the Executive Array are set to the address of the FZ structure and the length of the FZ structure, respectively;

- A dummy NA structure is allocated (note that an NP block is not required). The FZNA pointer is set to the address of the NA structure;

- To satisfy the Controller Loading section of the Steering Routine the NACNCD field (i.e. LAN Controller Code) is set to #01. The rest of the NA-block is ignored by the LAN Executive and LAN Controller. The NP-block is not used by either the LAN Executive or the LAN Controller;

- No LAN Buffers (Full or Short) are used by the LAN Executive.

The LAN controller section has no relevance for the Global Server configuration.

8. The NUCLEUS OPTIONS section

Internal readers should refer to Appendix A when reading this section.

The NUCLEUS SECTION of the Configuration File has the effect of establishing the various disparate fields in the I0-block. The I0-block is actually split into 3 sections:

BT-block	Bootstrap Information;
I0-block	Initialisation infromation;
IG-block	Generation and Distribution information.

However, since the inception of the I0-block the clear distinction between the 3 sections has blurred considerably.

The fields in the BT/I0/IG-block are established by, and used by, a variety of modules. Fields in the I0-block are **established** by the following software modules:

- CFUPDATE from hard-coded defaults;
- CFUPDATE from internal, invisible fields in header blocks within the A.W0 and A.W1 Action Files;
- CFUPDATE from prompt fields in the MACHINE NAME, BOOTSTRAP MESSAGES, NUCLEUS OPTIONS and DISTRIBUTION OPTIONS sections;
- CFUPDATE from calculations based on the results of processing the other sections of the configuration file (e.g. the CONSOLE CONTROLLERS section);
- The Steering Routine itself;
- The initialisation code of various Executives, Controllers and Nucleus Modules called by the Steering Routine;

Fields in the I0-block are **used** by the following software modules:

- CFUPDATE itself when validating and printing reports;
- The Steering Routine;
- The initialisation code of various Executives, Controllers and Nucleus Modules called by the Steering Routine;
- The $MONITOR Initiation functions;
- Various run-time components (fields from the I0-block are used to establish some (internal and external) System Variables (e.g. $$SYSM).

The BT/I0/IG-block created by CFUPDATE is in a "packed" Cobol format control block. This "packed" block must be unpacked to a "C" structure (bt,i0,ig) for subsequent use by the Steering Routine and the initialisation code of various Executives, Controllers and Nucleus Modules called by the Steering Routine. However, when the BT/I0/IG-block is presented to the Initiation code of $MONITOR (via a technique that is outside the scope of this document) it must be packed back from the "unpacked" structure to the "packed" control block.

When the configuration data is synthesized by the Steering Routine the "unpack" function, at the start of the Steering Routine processing, is **replaced** by a "synthesize" function that initialises the **important** fields established by CFUPDATE. The "important" fields are those that are used by software modules further along the GSM load process. Furthermore, the existing "pack" function, towards the end of the Steering Routine processing, is **supplemented** by an "extra pack" routine that establishes the additional fields in the BT/I0/IG-block that are required by $MONITOR but are not otherwise used by the Steering Routine and the initialisation code of various Executives, Controllers and Nucleus Modules called by the Steering Routine.

8.1 Global Client (GLOBAL.EXE)

Appendix A contains a detailed analysis of every field in the BT/I0/IG block. The remainder of this section just describes the "visible" fields. The following options in the NUCLEUS OPTIONS section of the Global Client configuration file must be considered:

SET BTFLAG TO #72 FOR GSM V8.1	Ignored – see Appendix A
DYNAMIC DC/DF BLOCK ALLOCATION	Ignored – see Appendix A
DYNAMIC LOCK TABLE ALLOCATION	Ignored – see Appendix A
LARGEST SECTOR SIZE	Set by Steering Routine – see below
NUMBER OF FILE CHANNELS	New registry option – see below
NUMBER OF FILE BUFFERS	New registry option – see below
NUMBER OF FILE BLOCKS	New registry option – see below
NUMBER OF LOCK TABLE ENTRIES	New registry option – see below
NUMBER OF EXTRA ASSIG$ TABLES	New registry option – see below
RAM HIGH ADDRESS (KB)	Set by Steering Routine – see below
MAXIMUM MEMORY ALLOCATION	Set by Steering Routine – see below
RAM DISK START ADDRESS (KB)	Set by Steering Routine – see below
CACHE BUFFER SIZE	Set by Steering Routine – see below
CACHE START ADDRESS (KB)	Set by Steering Routine – see below
DYNAMIC FILE BUFFER ALLOCATION	Ignored – see Appendix A
DYNAMIC LAN BUFFER ALLOCATION	Ignored – see Appendix A
DYNAMIC CONSOLE BUFFER ALLOCATION	Ignored – see Appendix A
DYNAMIC CS-BLOCK ALLOCATION	Ignored – see Appendix A
TARGET BOOTSTRAP STRATEGY	Ignored – see Appendix A
TARGET STARTUP STRATEGY	Ignored – see Appendix A
IS $REMOTE SUPPORTED?	Ignored – see Appendix A
$REMOTE DEVICE ADDRESS	Ignored – see Appendix A
$REMOTE BAUD RATE	Ignored – see Appendix A
$REMOTE TIME-OUT	Ignored – see Appendix A

BASIC CLOCK (MICROSECONDS)	New registry option – see below
CLOCK INTERVAL (MILLISECONDS)	New registry option – see below
SWAP INTERVAL (MILLISECONDS)	New registry option – see below
Load timer	Set by Steering Routine – see below
Load automatic OPID/TERM	Set by Steering Routine – see below
Load $BYE	Set by Steering Routine – see below
Set SYSYSM to 4 for Windows NT	Set by Steering Routine – see below
Auto date/time sign-on	New registry option – see below

The registry settings that emulate options in the NUCLEUS OPTIONS section of the configuration file are all in the following registry keys:

..\Global\Client\Nucleus	(see below)
..\Global\Client\Printers	(see section 6)

This section only considers the entries in the ..\Global\Client\Nucleus registry key. See section 6 for the details of the entries in the ..\Global\Client\Printers registry key.

The NUMBER OF FILE CHANNELS configuration file option is replaced by the +NumberOfFileChannels registry setting. The default value is 1000.

The NUMBER OF FILE BUFFERS configuration file option is replaced by the +NumberOfFileBuffers registry setting. The default value is 10.

The NUMBER OF FILE BLOCKS configuration file option is replaced by the +NumberOfFileBlocks registry setting. The default value is 100.

The NUMBER OF LOCK TABLE ENTRIES configuration file option is replaced by the +NumberOfLockTableEntries registry setting. The default value is 100. Note also that the related +NumberOfLockSharedTableEntries registry setting (default value 2000) is also included in this section..

The NUMBER OF EXTRA ASSIG$ TABLES configuration file option is replaced by the +NumberOfExtraASSIG$Tables registry setting. The default value is 1. The Number of Extra ASSIG$ Tables is held in the bottom 2 bits of the BTASSIG field. The #08 bit of the BTASSIG field is flag that indicates if the Dynamic Unit Mapping option (see document SJ307 & refs. GSM-6451 & W0-937) has been enabled. The registry setting +EnableUnitMapping (default “Off”) can be used to set bit #08 of BTASSIG.

The BASIC CLOCK (MICROSECONDS) configuration file option is replaced by the +BasicClockMicroSeconds registry setting. The default value of 100,000 **SHOULD NOT BE CHANGED.**

The CLOCK INTERVAL (MILLISECONDS) configuration file option is replaced by the +ClockIntervalMilliSeconds registry setting. The default value of 100 **SHOULD NOT BE CHANGED.**

The SWAP INTERVAL (MILLISECONDS) configuration file option is replaced by the +SwapIntervalMilliSeconds registry setting. The default value of 100 **SHOULD NOT BE CHANGED.**

The 3 "visible" Clock and Swap fields are used to calculate the "internal" Clock and Swap fields using the same algorithm as CFUPDATE (as described in Appendix A).

The Auto date/time sign-on configuration file option is replaced by the +AutoDateTimeSignOn registry setting. The default option is "On".

The new +AutoOperatorID registry setting allows the Operator-ID for the first console to be defined via the "Machine" hive of the registry (i.e. rather than the "User" hive).

In addition to those options listed above that are derived from registry settings, the Steering Routine also initialises the following variables:

BTHIGH	Initialised to 0
BTBANK	Initialised to 0
BTBND	Initialised to 0
BTLSTK	Initialised to 0
BTPSTK	Initialised to 0
BTSTMB	Initialised to 0
BTLNID	Initialised to 0
BTMBSZ	Initialised to 0
BTMAXA	Initialised to 0
I0SDCK	Initialised to 0
I0NCAB	Initialised to 0
I0HMBK	Initialised to 0
I0SPRE	Initialised to 0
I0XLAN	Initialised to 1
BTSYSM	Initialised to 4
BTARCH	Initialised to "W"

BTFORM	Initialised to 1
BTMCD	Initialised to "W1"
BTMSCD	Initiliased to 1
BTPROC	Initialised to 2
BTLAN	Initialised to 1
IGDOM	Initialised to 0 or 1 (depending on the number of domains)

A number of these fields (e.g. BTLNID) may be modified during the GSM loading process.

The following entries are inserted into the I0NLST "Nucleus Component" array to allow the Nucleus component load section of the Steering Routine to remain unaltered:

SV9	Load timer
AUT	Load automatic OPID/TERM
BYE	Load $BYE

A number of "derived fields" are calculated during the processing of the DIRECT ACCESS CONTROLLER and CONSOLE CONTROLLER sections (see above):

I0MAXF	See section 3
I0MAXS	See section 3
I0CBSZ	See section 3
I0NBUF	See section 3
I0NCHA	See section 3
I0NFIL	See section 3
BTLOCK	See section 3
BTNUSE	See section 4
BTCSCR	See section 4
BTCPAR	See section 4

Furthermore, a number of user-specified fields are validated during the processing of the DIRECT ACCESS CONTROLLER, CONSOLE CONTROLLER and PRINTER CONTROLLER sections (see above):

I0NBUF	See section 3
I0NFIL	See section 4
I0NCHA	See section 4
I0NPB	See section 6
I0SPB	See section 6

As explained in section 4 the strict validation of the Number of File Channels can be enabled by switching the +ValidateNumberOfFileChannels registry setting to “On”. Similarly, the strict validation of the Number of File Blocks can be enabled by switching the +ValidateNumberOfFileBlocks registry setting to “On”.

8.2 Global Server (GLSERVER.EXE)

The Global Server (GLSERVER.EXE) requires a cut-down configuration file with just enough components and parameters defined to allow the File Executive to operate. This is reflected in the configuration file emulation which only considers those parameters that directly affect the File Executive.

The registry settings that emulate options in the NUCLEUS OPTIONS section of the configuration file are all in the following registry keys:

..\Global\Servers\ (all servers default)

or:

..\Global\Servers*x*\ (server specific)

(where *x* is the server letter between A and Z).

The server-specific option, if established, is used in preference to the “all servers” option, for the relevant server.

The NUMBER OF FILE CHANNELS configuration file option is replaced by the +NumberOfFileChannels registry setting. The default value is 1000.

The NUMBER OF FILE BUFFERS configuration file option is replaced by the +NumberOfFileBuffers registry setting. The default value is 10.

The NUMBER OF FILE BLOCKS configuration file option is replaced by the +NumberOfFileBlocks registry setting. The default value is 100.

The NUMBER OF LOCK TABLE ENTRIES configuration file option is replaced by the +NumberOfLockTableEntries registry setting. The default value is 100. Note also that the related +NumberOfLockSharedTableEntries registry setting (default value 2000) is also included in this section..

There is no requirement to emulate the NUMBER OF EXTRA ASSIG$ TABLES configuration file option by a combination of the +NumberOfExtraASSIG$Tables and +EnableUnitMapping registry settings.

There is no requirement to emulate the BASIC CLOCK (MICROSECONDS) configuration file option by the +BasicClockMicroSeconds registry setting. A value of 100,000 is assumed. There is no requirement to emulate the CLOCK INTERVAL (MILLISECONDS) configuration file option by the +ClockIntervalMilliSeconds registry setting. A value of 100 is assumed. There is no requirement to emulate the SWAP INTERVAL (MILLISECONDS) configuration file option by the +SwapIntervalMilliSeconds registry setting. A value of 100 is assumed. The standard CFUPDATE algorithm is used to calculate the "internal" Clock and Swap fields (although these fields are not actually used at run-time by the Global Server).

There is no requirement to emulate the Auto date/time sign-on configuration file option by the +AutoDateTimeSignOn registry setting. Similarly, there is no requirement to consider an +AutoOperatorID registry setting.

In addition to those options listed above that are derived from registry settings, the Steering Routine also initialises the following variables:

BTHIGH	Initialised to 0
BTBANK	Initialised to 0
BTBND	Initialised to 0
BTLSTK	Initialised to 0
BTPSTK	Initialised to 0
BTSTMB	Initialised to 0
BTLNID	Initialised to 0
BTMBSZ	Initialised to 0
BTMAXA	Initialised to 0
I0SDCK	Initialised to 0
I0NCAB	Initialised to 0
I0HMBK	Initialised to 0
I0SPRE	Initialised to 0
I0XLAN	Initialised to 1
BTSYSM	Initialised to 4
BTARCH	Initialised to "W"
BTFORM	Initialised to 1
BTMCD	Initialised to "W2"
BTMSCD	Initiliased to 1
BTPROC	Initialised to 2

BTLAN Initialised to 1
IGDOM Initialised to 0 or 1 (depending on the number of domains)

Note 1: The value of BTMCD is set to "W1" for the Global Client; and "W2" for the Global Server. For the Global Client this field is used by the Steering Routine when loading the rest of the nucleus AND is passed to $MONITOR to establish the SYMCD System Variable (see Appendix A). For the Global Server this field is only used internally by the Steering Routine when loading the rest of the nucleus (i.e. the bootstrap process for a Global Server configuration never reaches the point that loads and initialises $MONITOR).

Note 2: BTLNID is NOT modified during the Global Server loading process.

The following entry is inserted into the I0NLST "Nucleus Component" array to allow the Nucleus component load section of the Steering Routine to remain unaltered:

SV9 Load timer

Note 3: The "AUT" and "BYE" Nucleus Components are NOT required for a Global Server configuration.

A number of "derived fields" are calculated during the processing of the DIRECT ACCESS CONTROLLER section (see above):

I0MAXF See section 3
I0MAXS See section 3
I0CBSZ See section 3

Furthermore, a number of user-specified fields are validated during the processing of the DIRECT ACCESS CONTROLLER section (see above):

I0NBUF See section 3

Note 4: The strict validation of the Number of File Channels is NOT performed for the Global Server. The Global Server is a "User-less" configuration so any algorithm that uses the number of screens and/or number of users is meaningless). Similarly, the strict validation of the Number of File Blocks is NOT performed for the Global Server. Consequently, the +ValidateNumberOfFileChannels and +ValidateNumberOfFileBlocks registry settings do not appear in the "Servers" registry keys.

9. The DISTRIBUTION OPTIONS section

This section of the Configuration File is reserved for fields in the IG-portion of the tri-purpose BT/I0/IG block (see section 8 and Appendix A) that are only used by the software generation process. In general, fields in the IG-block are not used when GSM (Windows NT) is loaded or at run-time.

The following options in the DISTRIBUTION OPTIONS section of the Global Client configuration file are all ignored:

SOFTWARE LEVEL NUMBER	Ignored – see Appendix A
SERIAL CONSOLES ATTACHED?	Ignored – see Appendix A
PRERELEASE?	Ignored – see Appendix A
MONITOR SUFFIX	Ignored – see Appendix A
STARTER BOOTSTRAP STRATEGY	Ignored – see Appendix A
DISTRIBUTION UNIT ADDRESS	Ignored – see Appendix A
IPL UNIT ADDRESS	Ignored – see Appendix A
BACNAT format	Ignored – see Appendix A
Number of BACNAT diskettes	Ignored – see Appendix A
STARTER INSTALLATION STRATEGY	Ignored – see Appendix A

Consequently, no changes to the GSM (Windows NT) registry have been made to emulate these configuration file options.

No changes to the GSM (Windows NT) Steering Routine have been made to emulate these configuration file options.

However, the field IGDOM must be established by the Steering Routine. The $MONITOR initialisation code moves this field to the SYDOM System Variable which IS used at run-time (by both $FIND and GEN). Thus, the IGDOM field must be established in the "packed BT/I0/IG-block" that is passed to $MONITOR by the Steering Routine (see section 8). IGDOM is set to 1 if one, or more, DDF domains are configured in the registry (see section 3).

The DISTRIBUTION OPTIONS section has no relevance for the Global Server configuration.

APPENDIX A - NOTES ON THE BT, I0 & IG BLOCKS

This Appendix is for internal use only. It contains a detailed description of every field in the tri-purpose BT/I0/IG-block. This Appendix should be used in conjunction with section 8.

A.1 I0BT at the start of the I0/BT-block (BT = Bootstrap information)

BTUEA	**PIC 9(4) COMP**	**PART.ADDR USER AREA**
Compiled offset	#0000	
Typical value	#0000	
BT/I0-block	BTUE 0	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	Packed after memory allocation completed	
Nucleus usage	No	
$MONITOR usage	Moved to SYCUE	
How emulated	No changes (i.e. this field is packed in a special way)	

BTHIGH	**PIC PTR**	**ADDR HIGHEST BYTE+1**
Compiled offset	#0002	
Typical value	#0000	
BT/I0-block	BTHIGH 2	
BT/I0 structure	i0.high	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTHIGH to i0.high	
Steer Rtn pack	i0.high to BTHIGH	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0m.c, read/write	
$MONITOR usage	Moved to SYBOHIGH & MIHIGH	
How emulated	No attempt to unpack BTHIGH Field i0.high initialised to #0000 Packed as normal	

BTARCH	**PIC X**	**ARCHITECTURE CODE**
Compiled offset	#0004	
Typical value	#57 = "W"	
BT/I0-block	BTARCH 4	
BT/I0 structure	No	
Action file	No	
Configurator	Copied from AMMBOS	
Steer Rtn unpack	No	

Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to SYARCH
How emulated	New field bt.arch_syn required Field bt.arch_syn initialised to hard-coded "W" Field bt.arch_syn must be packed back to BTARCH

BTFLAG	**PIC 9(2) COMP**	**0 (INDICATES V5.0)/6 (INDICATES V6.0)**
Compiled offset	#0005	
Typical value	#72	
BT/I0-block	BTFLAG 5	
BT/I0 structure	i0.flag	
Action file	SET BTFLAG TO #72 FOR GSM V8.1	
Configurator	Action File value used by CFUPDATE	
Steer Rtn unpack	BTFLAG to i0.flag	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	vc0b.c, read	
$MONITOR usage	No	
How emulated	No attempt to unpack BTFLAG Not packed back Packed BT-block value will be #00 rather than #72 Also, remove use of this obsolete field from the Console Executive	

BTBANK	**PIC 9(2) COMP**	**NO. OF MEMORY BANKS**
Compiled offset	#0006	
Typical value	#00	
BT/I0-block	BTBANK 6	
BT/I0 structure	io.bank	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTBANK to i0.bank	
Steer Rtn pack	i0.bank to BTBANK	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0m.c, write	
$MONITOR usage	Moved to SYBOBANK & SYBANK	
How emulated	No attempt to unpack BTBANK Field i0.bank initialised to #0000 Packed as normal	

BTFORM	**PIC 9(2) COMP**	**1=M6800 ADDR.,2=PDP**
Compiled offset	#0007	
Typical value	#01	
BT/I0-block	BTFORM 7	
BT/I0 structure	No	
Action file	No	
Configurator	Copied from AMFORM	

Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Move to SYFORM
How emulated	New field bt.form_syn required Field bt.form_syn initialised to hard-coded #01 Field bt.form_syn must be packed back to BTFORM

BTBOUND	**PIC 9(4) COMP**	**SWAP AREA BOUND ALIGN**
Compiled offset	#0008	
Typical value	#0000	
BT/I0-block	BTBND 8	
BT/I0 structure	i0.bnd	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTBND to i0.bnd	
Steer Rtn pack	i0.bnd to BTBND	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0m.c, write	
$MONITOR usage	Moved to SYMSIZE	
How emulated	No attempt to unpack BTBND Field i0.bnd initialised to #0000 Packed as normal	

BTMCD	**PIC X(2)**	**0Z=PDP,0S=S/1**
Compiled offset	#000A	
Typical value	#5731 = "W1"	
BT/I0-block	BTMCD 10	
BT/I0 structure	No	
Action file	No	
Configurator	Copied from AMMARC	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked to ZMCD for use when building controller names	
Nucleus usage	No	
$MONITOR usage	Moved to SYMCD	
How emulated	New field bt.mcd_syn required Field bt.mcd_syn initialised to hard-coded "W1" Field bt.mcd_syn must be packed back to BTMCD	

BTMSCD	**PIC 9(2) COMP**	**MACH. SPEC. SUBCODE.**
Compiled offset	#000C	
Typical value	#01	
BT/I0-block	BTMSCD 12	
BT/I0 structure	No	
Action file	No	
Configurator	Copied from AMMSUB	

Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to SYMSCD
How emulated	New field bt.mscd_syn required Field bt.mscd_syn initialised to hard-coded #01 Field bt.mscd_syn must be packed back to BTMSCD

BTNVER	**PIC 9(2) COMP**	**ASS. NUCLEUS VER.**
Compiled offset	#000D	
Typical value	#03	
BT/I0-block	BTMVER (*sic*) 13	
BT/I0 structure	No	
Action file	No	
Configurator	Set to 3	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #00 rather than #03	

BTNUSE	**PIC 9(2) COMP**	**TOTAL NUMBER OF USERS**
BTNUSX	**PIC X**	**TOTAL NUMBER OF USERS**
Compiled offset	#000E	
Typical value	#1D	
BT/I0-block	BTNUSE14	
BT/I0 structure	i0.nuse	
Action file	No	
Configurator	Calculated from the Number of Partitions	
Steer Rtn unpack	BTNUSE to i0.nuse	
Steer Rtn pack	i0.nuse to BTNUSE	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0x6.c, read; vc0m.c, read	
$MONITOR usage	Moved to SYBOUSER & MIUSER	
How emulated	No attempt to unpack BTNUSE Field i0.nuse is calculated during Console Executive processing Algorithm: The total number of partitions (i.e. CB-blocks). Packed as normal	

BTINT	**PIC 9(2) COMP**	**MAX. NO. INTACV USER**
Compiled offset	#000F	
Typical value	#00	
BT/I0-block	BTINT 15	
BT/I0 structure	No	
Action file	No	
Configurator	No (pre-V6.0 only)	

Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to SYBOINT
How emulated	No (this field is now obsolete) Packed BT-block value will continue to be #00

BTNAM **PIC 9(4) COMP** **ADDR OF PROCESSOR NAM**

Compiled offset	#0010
Typical value	#0000
BT/I0-block	BTNAM 16
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed at I0+LDES
Nucleus usage	No
$MONITOR usage	Pointer to bootstrap name
How emulated	No changes (i.e. this field is packed in a special way)

BTBTMS **PIC 9(4) COMP** **ADDR OF SIGNON MESSAGS**

Compiled offset	#0012
Typical value	#0000
BT/I0-block	BTBTMS 18
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed at I0+MESS
Nucleus usage	No
$MONITOR usage	Used when displaying Nucleus Messages
How emulated	No changes (i.e. this field is packed in a special way)

BTERRM **PIC 9(4) COMP** **ADDR FILE-IDS/ERR CODE**

Compiled offset	#0014
Typical value	#0000
BT/I0-block	BTERRM 20
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed at Error Messages
Nucleus usage	No
$MONITOR usage	Used when displaying Error Messages

How emulated	No changes (i.e. this field is packed in a special way)

FILLER **PIC 9(4) COMP** **ONCE &&SPRE**

Compiled offset	#0016
Typical value	#0000
BT/I0-block	BTSPRE 22
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00

BTSVCP **PIC 9(4) COMP** **PTR TO SVC TABLE**

Compiled offset	#0018
Typical value	#0000
BT/I0-block	BTSVCP 24
BT/I0 structure	bt.svcp
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed at SVC-table
Nucleus usage	vc0x6.c, write; vc5ta.c, write
$MONITOR usage	Used when establishing SVC pointers
How emulated	No changes (i.e. this field is packed in a special way)

FILLER **PIC X(2)** **ONCE $CP/SWAP ASSNMNT**

Compiled offset	#001A
Typical value	#0000

BTEXZ **PIC 9(4) COMP** **ADDR XZ ADDR ARRAY**

Compiled offset	#001C
Typical value	#110E
BT/I0-block	BTEXZ 28
BT/I0 structure	No
Action file	No
Configurator	Pointer past last xZ block in Configuration file
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #0000 rather than non-zero value

BTPROC	**PIC 9(2) COMP**	**0=S-PROC ELSE=M-PROC**
Compiled offset	#001E	
Typical value	#02	
BT/I0-block	BTPROC	30
BT/I0 structure	No	
Action file	No	
Configurator	0,1 or 2 depending on AMMP and BTLAN	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SYBOPROC	
How emulated	New field bt.proc_syn required Field bt.proc_syn initialised to hard-coded #02 Field bt.proc_syn must be packed back to BTPROC	

BTVDAT	**PIC 9(2) COMP**	**1=VIEWDATA KEYPAD ONLY**
Compiled offset	#001F	
Typical value	#00	
BT/I0-block	BTVDAT31	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	If zero, then 1 moved to SYVDAT	
How emulated	No. Packed BT-block value will continue to be #00	

BTLSTK	**PIC 9(4) COMP**	**LENGTH OF STACK**
Compiled offset	#0020	
Typical value	#0000	
BT/I0-block	BTLSTK	32
BT/I0 structure	i0.lstk	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTLSTK to i0.lstk	
Steer Rtn pack	i0.lstk to BTLSTK	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0m.c, write	
$MONITOR usage	If nonzero, used to create System Stack (MNSYSL, MNSIZE)	
How emulated	No attempt to unpack BTLSTK Field i0.lstk initialised to #0000 Packed as normal	

BTPSTK	**PIC PTR**	**ADDR. STACK**
Compiled offset	#0022	

Typical value	#0000
BT/I0-block	BTPSTK 34
BT/I0 structure	i0.pstk
Action file	No
Configurator	No
Steer Rtn unpack	BTPSTK to i0.pstk
Steer Rtn pack	i0.pstk to BTPSTK
Steer Rtn usage	Packed & unpacked
Nucleus usage	vc0m.c, write
$MONITOR usage	If BTLSTK nonzero, used to create System Stack (SYSYST)
How emulated	No attempt to unpack BTPSTK Field i0.pstk initialised to #0000 Packed as normal

BTSWAP	**PIC PTR**	**ADDR TO SWAP FROM**
Compiled offset	#0024	
Typical value	#0000	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SYSELE	
How emulated	No. Packed BT-block value will continue to be #0000	

BTFSWAP	**PIC 9 COMP**	**0=USE BOBOUND,1=BOSWAP**
Compiled offset	#0026	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SY11-L	
How emulated	No. Packed BT-block value will continue to be #00	

BTIPLD	**PIC 9(2) COMP**	**IPL DEVICE NO**
Compiled offset	#0027	
Typical value	#00	
BT/I0-block	BTIPLD 39	
BT/I0 structure	No	
Action file	No	

Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	DADEV moved to BTIPLD
Nucleus usage	No
$MONITOR usage	May be used instead of CPILID when loading P.$MON
How emulated	This field is essentially obsolete because the CPILID field in the $MONITOR is always valid (i.e. nonzero) on an installed system. However, BTIPLD MUST still be established when loading a starter system (i.e. the $MONITOR on BACRES does not contain a valid IPL unit, from which P.$MON is loaded). The option to remove the configuration file must NOT be enabled for a Starter System (i.e. when loading from BACRES). Not packed back. Therefore, the packed BT-block value will be #00 rather than a non-zero value

BTFIX **PIC 9(9) COMP** **ABS ADDR FIXED AREA**

Compiled offset	#0028
Typical value	#00000000
BT/I0-block	BTFXDA40
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to SYFIX
How emulated	No. Packed BT-block value will continue to be #00

BTSTMB **PIC 9(9) COMP** **ABS ADDR 1st MEM BANK**

Compiled offset	#002C
Typical value	#00000000
BT/I0-block	BTSTMB 44
BT/I0 structure	i0.stmb
Action file	No
Configurator	No
Steer Rtn unpack	BTSTMB to i0.stmb
Steer Rtn pack	i0.stmb to BTSTMB
Steer Rtn usage	Packed & unpacked
Nucleus usage	vc0m.c, write
$MONITOR usage	Moved to SYSTMB
How emulated	No attempt to unpack BTSTMB Field i0.stmb initialised to #0000 Packed as normal

BTLAN **PIC 9 COMP** **1 = BOS/LAN**

Compiled offset	#0030
Typical value	#01

BT/I0-block	BTLAN 48
BT/I0 structure	No
Action file	No
Configurator	0 if no LAN controller; 1 if LAN controller present
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	New field bt.lan_syn required Field bt.lan_syn initialised to hard-coded #01 Field bt.lan_syn must be packed back to BTLAN

BTLNID	**PIC X**	**LOCAL NODE-ID**
Compiled offset	#0031	
Typical value	#00	
BT/I0-block	BTLNID 49	
BT/I0 structure	i0.lnid	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTNLID to i0.nlid	
Steer Rtn pack	i0.nlid to BTNLID	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc5fa.c, write	
$MONITOR usage	Moved to SYLNID, if CPLNID is zero	
How emulated	No attempt to unpack BTLNID Field i0.lnid initialised to #00 Packed as normal	

BTLOCK	**PIC 9(4) COMP**	**NUMBER OF LOCKS IN FE**
Compiled offset	#0032	
Typical value	#0064	
BT/I0-block	BTLOCK50	
BT/I0 structure	No	
Action file	NUMBER OF LOCK TABLE ENTRIES	
Configurator	Validated	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	Used for LK-block creation	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	New field bt.lock_syn required Field bt.lock_syn initialised to value of registry key: **+NumberOfLockTableEntries** Default value = 100 Field bt.lock_syn used by Steering Routine instead of BTLOCK Field bt.lock_syn packed back to BTLOCK (for completeness only)	
Also emulated	New field bt.slck_syn required	

	Field bt.slck_syn initialised to value of registry key: **+NumberOfSharedLockTableEntries** Default value = 2000 Field bt.slck_syn used by Steering Routine Field bt.slck_syn NOT packed back

BTIFID	**PIC X(8)**	**FILE-ID OF INST. FILE**
Compiled offset	#0034	
Typical value	"++5663XJ"	
BT/I0-block	BTIFID 52	
BT/I0 structure	No	
Action file	No	
Configurator	File name of Configurator file	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	Set up from RLNAME (as used by the loader)	
Nucleus usage	No	
$MONITOR usage	Last 6 characters moved to SYCONF	
How emulated	New field bt.ifid_syn required Field bt.ifid_syn initialised to value of registry key: **+SythesizedConfigurationFileName** Default value = "++566SYN" No need to establish from RLNAME Field bt.ifid_syn packed back to BTIFID	

BTCSCR	**PIC 9 COMP**	**NUMBER OF CBOS SCREENS**
Compiled offset	#003C	
Typical value	#06	
BT/I0-block	BTCSCR 60	
BT/I0 structure	i0.cscr	
Action file	No	
Configurator	Calculated number of screens	
Steer Rtn unpack	BTCSCR to i0.cscr	
Steer Rtn pack	i0.cscr to BTCSCR	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	Moved to SYBOCSCR & MICSCR	
How emulated	No attempt to unpack BTCSCR Field i0.cscr is calculated during Console Executive processing Algorithm: The number of screen controllers (i.e. CA-blocks). Packed as normal	

BTCPAR	**PIC 9 COMP**	**CBOS PART. PER SCREEN**
Compiled offset	#003D	
Typical value	#09	
BT/I0-block	BTCPAR61	
BT/I0 structure	i0.cpar	
Action file	No	

Configurator	Calculated Maximum Number of Partitions per Screen
Steer Rtn unpack	BTCPAR to i0.cpar
Steer Rtn pack	No
Steer Rtn usage	Unpacked only
Nucleus usage	vc0b.c, read
$MONITOR usage	Moved to SYBOCPAR & MICPAR
How emulated	No attempt to unpack BTCPAR Field i0.cpar is calculated during Console Executive processing Algorithm: Highest partition number on any screen. Must be packed

BTFGRQ	**PIC 9 COMP**	**0=F/G ON SERIAL SCREENS**
Compiled offset	#003E	
Typical value	#00	
BT/I0-block	BTFGRQ 62	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

FILLER	**PIC X**	**KEEP NEXT FIELDS EVEN**
Compiled offset	#003F	
Typical value	#00	

BTMBSZ	**PIC 9(9) COMP M.BANK SIZE (FOR INDX**
Compiled offset	#0040
Typical value	#00000000
BT/I0-block	BTMBSZ 64
BT/I0 structure	i0.mbsz
Action file	No
Configurator	No
Steer Rtn unpack	BTMBSZ to i0.mbsz
Steer Rtn pack	i0.mbsz to BTMBSZ
Steer Rtn usage	Packed & unpacked
Nucleus usage	vc0m.c, write
$MONITOR usage	Moved to SYMBSZ
How emulated	No attempt to unpack BTMBSZ Field i0.mbsz initialised to #00000000 Packed as normal

BTMAXA	**PIC 9(9) COMP**	**HIGHEST M/C ADDRESS**
Compiled offset	#0044	
Typical value	#00000000	

BT/I0-block	BTMAXA	68
BT/I0 structure	i0.maxa	
Action file	No	
Configurator	No	
Steer Rtn unpack	BTMAXA to i0.maxa	
Steer Rtn pack	i0.maxa to BTMAXA	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0m.c, write	
$MONITOR usage	Moved to SYMHIG	
How emulated	No attempt to unpack BTMAXA Field i0.maxa initialised to #00000000 Packed as normal	

BTOPID	**PIC X(4)**	**AUTO OPERATOR ID**
Compiled offset	#0048	
Typical value	#00000000	
BT/I0-block	BTOPID	72
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SYAUOP	
How emulated	New field bt.opid_syn required Field bt.opid_syn initialised to value of registry key: **+AutoOperatorID** Default value = #00000000 Field bt.opid_syn packed back to BTOPID	

BTMSCV	**PIC X**	**MACH. SPEC. VERSION**
Compiled offset	#004C	
Typical value	#00	
BT/I0-block	BTMSCV	76
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

BTSYSM	**PIC 9 COMP**	**1=SYSTEM MANAGER**
Compiled offset	#004D	
Typical value	#04	

BT/I0-block	BTSYSM 77
BT/I0 structure	i0.sysm
Action file	Set SYSYSM to 4 for Windows NT
Configurator	No
Steer Rtn unpack	BTSYSM to i0.sysm
Steer Rtn pack	i0.sysm to BTSYSM
Steer Rtn usage	Packed & unpacked
Nucleus usage	vc5fa.c, write
$MONITOR usage	Moved to SYSYSM
How emulated	No attempt to unpack BTSYSM Field i0.sysm initialised to 4. [Note that in the original design the registry key **+SetSYSYSMFlag**, with a default value = 4, was suggested but a hard-coded value set in the Steering Routine, that may be patched by the LAN controller, is more appropriate]. Packed as normal

BTAUDT	**PIC 9 COMP**	**AUTO DATE/TIME S-ON**
Compiled offset	#004E	
Typical value	#01	
BT/I0-block	BTAUDT 78	
BT/I0 structure	i0.audt	
Action file	Auto date/time sign-on	
Configurator	No	
Steer Rtn unpack	BTAUDT to i0.audt	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	No	
$MONITOR usage	Moved to SYAUDT	
How emulated	No attempt to unpack BTAUDT Field i0.audt initialised to value of registry key: **+AutoDateTimeSignOn** Default value = "Y" (i.e. 1) Field i0.audt must be packed back to BTAUDT	

BTBACV	**PIC 9(4) COMP**	**BACNAT VERSION * 1000 E.G 3.111 = 3111**
Compiled offset	#004F	
Typical value	#0800	
BT/I0-block	BTBACV79	
BT/I0 structure	i0.bacv	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	i0.bacv to BTBACV	
Steer Rtn usage	Packed	
Nucleus usage	vc5na.c, write	
$MONITOR usage	Moved to SYBACV	
How emulated	No changes (i.e. synthesised and packed as normal)	

BTASSIG	**PIC 9(2) COMP**	**REDEFINITION OF BTBACV (NO OF EXTRA ASSIG$'S)**
Compiled offset	#004F	
Typical value	#08	
BT/I0-block	BTASSIG 79	
BT/I0 structure	i0.assig	
Action file	NUMBER OF EXTRA ASSIG$ TABLES	
Configurator	No	
Steer Rtn unpack	BTASSIG to i0.assig	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	vc0x6.c, read	
$MONITOR usage	No	
How emulated	No attempt to unpack BTASSIG Field i0.assig initialised to value of registry key: **+Number OfExtraASSIG$Tables** Default value = 1 (bottom 2 bits only i.e. #03) Also, registry key: **+EnableUnitMapping** Default value = "Off" (set bit #08 of i0.assig) Not packed (as normal)	

BTDCX	**PIC 9 COMP**	**1 EXTENDED DC BLOCK/AVAILABLE (>99 USERS)**
Compiled offset	#0051	
Typical value	#00	
BT/I0-block	BTDCX 81	
BT/I0 structure	i0.dcx	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	i0.dcx to BTDCX	
Steer Rtn usage	Packed	
Nucleus usage	No	
$MONITOR usage	Moved to SYDCX	
How emulated	No changes. The Steering Routine will continue to move 1 to io.dcx which will be packed as normal.	

BTBIF1	**PIC 9 COMP**	**EXTENDED SVC 80 AVAILABLE**
Compiled offset	#0052	
Typical value	#00	
BT/I0-block	BTBIF1 82	
BT/I0 structure	i0.bif1	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	i0.bif1 to BTBIF1	
Steer Rtn usage	Packed	
Nucleus usage	No	
$MONITOR usage	Moved to SYBIF1	

How emulated — No changes. The Steering Routine will continue to initialised i0.bif1 as follows (which must be packed as normal):

Bit	*How established*
#01	Set if extended SVC-80 operations are supported
#02	Set if registry value **CentralPCFName** enabled
#04	Reserved, registry value **SYBIF1ReservedBit04**
#08	Reserved, registry value **SYBIF1ReservedBit08**
#10	Reserved, registry value **SYBIF1ReservedBit10**
#20	Set if SVC-79 MXOPC 5, FileMapping supported
#40	Set if Configuration File emulation enabled
#80	Set if registry value **ExtendedSharedLocks** enabled

BTSV85N	**PIC X**		**RESERVED FOR SVC-85 (BOS/DOS EXTRA MEMORY)**
Compiled offset	#0053		
Typical value	#00		
BT/I0-block	BTSV85N	83	
BT/I0 structure	No		
Action file	No		
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will continue to be #00		

BTSV85S	**PIC X(2)**		**RESERVED FOR SVC-85 (BOS/DOS EXTRA MEMORY)**
Compiled offset	#0054		
Typical value	#0000		
BT/I0-block	BTSV85S	84	
BT/I0 structure	No		
Action file	No		
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will continue to be #0000		

BTEPTT	**PIC X**		**RESERVED FOR EXTENDED PRINTER XLAT TABS**
Compiled offset	#0056		
Typical value	#00		
BT/I0-block	BTEPTT	86	
BT/I0 structure	No		

Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00

FILLER **PIC X(43)** **FILLER AREA**

Compiled offset	#0057
Typical value	#00's

Main part of I0-block

I0DEXC **OCCURS 10 PIC S9(4) COMP**
REDEFINED AS I0EXPTR OCCURS 10 PIC PTR

Compiled offset	#0082
Typical value	Various pointers
BT/I0-block	I0DEXC 130
BT/I0 structure	i0.dexc
Action file	No
Configurator	Array of xZ pointers
Steer Rtn unpack	I0DEXC to i0.dexc
Steer Rtn pack	No
Steer Rtn usage	Unpacked
Nucleus usage	vc5na.c, read, i0.dexc[2]
$MONITOR usage	No
How emulated	No changes (all executives assumed when no configuration file) Not packed back. Packed BT-block values will be #0000 rather than #*xxxx*

I0LEXC **OCCURS 10 PIC S9(4) COMP** **TABLE OF Xz CONTROL BLOCK LENGTHS**

Compiled offset	#0096
Typical value	Various lengths
BT/I0-block	I0LEXC 150
BT/I0 structure	i0.lexc
Action file	No
Configurator	Array of xZ lengths
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No changes (the lengths in i0.lexc are set when the xZ blocks are dynamically allocated) Not packed back. Packed BT-block values will be #0000 rather than #*xxxx*

I0MAXS **PIC S9(4) COMP** **MAX SECTOR SIZE**

Compiled offset	#00AA
Typical value	#0200
BT/I0-block	I0MAXS 170
BT/I0 structure	i0.maxs
Action file	LARGEST SECTOR SIZE
Configurator	Maximum AVSEC value
Steer Rtn unpack	I0MAXS to i0.maxs
Steer Rtn pack	No
Steer Rtn usage	Unpacked
Nucleus usage	vc0a.c, read
$MONITOR usage	No
How emulated	No attempt to unpack I0MAXS Field i0.maxs is calculated during File Executive processing Algorithm: Maximum Sector size value Must be packed (for completeness)

I0MAXF PIC S9(4) COMP MAX NO OF FILES ON DLV

Compiled offset	#00AC
Typical value	#00FA
BT/I0-block	I0MAXF 172
BT/I0 structure	No
Action file	No
Configurator	Maximum AVMAXF value
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	Continue not to attempt to unpack I0MAXF Field i0.maxf_syn is calculated during File Executive processing Algorithm: Maximum Number Of Files per Subvolume This field is used internally for validating the value of I0NBUF. Packed for completeness.

I0NCHA PIC S9(4) COMP NUMBER OF FILE CHANNELS

Compiled offset	#00AE
Typical value	#0040
BT/I0-block	I0NCHA174
BT/I0 structure	i0.ncha
Action file	NUMBER OF FILE CHANNELS
Configurator	Calculated, depending on BTNUSE
Steer Rtn unpack	I0NCHA to i0.ncha
Steer Rtn pack	i0.ncha to I0NCHA
Steer Rtn usage	Packed & unpacked
Nucleus usage	No
$MONITOR usage	No
How emulated	No attempt to unpack I0NCHA Field i0.ncha initialised to value of registry key:

+NumberOfFileChannels
Default value = 1000
Also, optionally validated by Steering Routine
Algorithm: I0NCHA >= 8 * BTNUSE; I0NCHA>I0NFIL; I0NCHA>64
The validation is enabled by setting the following option to "on":
+ValidateNumberOfFileChannels
Packed as normal (for completeness)

I0NBUF	**PIC S9(4) COMP**	**NUMBER OF FILE BUFFERS**
Compiled offset	#00B0	
Typical value	#0006	
BT/I0-block	I0NBUF	176
BT/I0 structure	i0.nbuf	
Action file	NUMBER OF FILE BUFFERS	
Configurator	Calculated, depending on I0MAXF and I0MAXS	
Steer Rtn unpack	I0NBUF to i0.nbuf	
Steer Rtn pack	i0.nbuf to BTNBUF	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	vc0a.c, read	
$MONITOR usage	No	
How emulated	No attempt to unpack I0NBUF	

Field i0.nbuf initialised to value of registry key:
+NumberOfFileBuffers
Default value = 10
Also, validated by Steering Routine
Algorithm: I0NBUF >= 1 + (8 * (I0MAXF + 1) /I0MAXS) RND UP
Packed as normal (for completeness)

I0NFIL	**PIC S9(4) COMP**	**NUMBER OF FILE BLOCKS**
Compiled offset	#00B2	
Typical value	#001E	
BT/I0-block	I0NFIL	178
BT/I0 structure	i0.nfil	
Action file	NUMBER OF FILE BLOCKS	
Configurator	Calculated, depending on BTNUSE	
Steer Rtn unpack	I0NFIL to i0.nfil	
Steer Rtn pack	i0.nfil to I0NFIL	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0NFIL	

Field i0.nfil initialised to value of registry key:
+NumberOfFileBlocks
Default value = 100
Also, optionally validated by Steering Routine
Algorithm: I0NFIL >= (8 * SQRT(*number of users*)) ROUNDED UP
The validation is enabled by setting the following option to "on":
+ValidateNumberOfFileBlocks

Packed as normal (for completeness)

I0BCLK	**PIC 9(6,3) COMP**	**BASIC CLOCK (mS)**
Compiled offset	#00B4	
Typical value	#000186A0	
BT/I0-block	I0EDIT 180	
	I0BCLK 180	
BT/I0 structure	No	
Action file	BASIC CLOCK (MICROSECONDS)	
Configurator	Only calculated if I0FREQ set	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	Continue not to unpack I0BCLK	
	New field io.bclk_syn required	
	Field i0.bclk_syn initialised to value of registry key:	
	+BasicClockMicroSeconds	
	Default value = 100000	
	Field i0.bclk_syn packed back to I0BCLK (for completeness)	

I0FREQ	**PIC 9(4) COMP**	**CLOCK FREQUENCY (Hz)**
Compiled offset	#00B8	
Typical value	#0000	
BT/I0-block	I0FREQ 184	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0CUP	**PIC 9(4) COMP**	**CLOCK UPDATE PERIOD (mS)**
Compiled offset	#00BA	
Typical value	#0064	
BT/I0-block	I0CUP 186	
BT/I0 structure	No	
Action file	CLOCK INTERVAL (MILLISECONDS)	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	

How emulated	Continue not to attempt to unpack I0CUP New field io.cup_syn required Field i0.cup_syn initialised to value of registry key: **+ClockIntervalMilliSeconds** Default value = 100 Field i0.cup_syn packed back to I0CUP (for completeness)

I0SWIN	**PIC 9(4) COMP**	**SWAP INTERVAL**
Compiled offset	#00BC	
Typical value	#0064	
BT/I0-block	I0SWIN 188	
BT/I0 structure	No	
Action file	SWAP INTERVAL (MILLISECS)	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	Continue not to attempt to unpack I0SWIN New field io.swin_syn required Field i0.swin_syn initialised to value of registry key: **+SwapIntervalMilliSeconds** Default value = 100 Field i0.swin_syn packed back to I0SWIN (for completeness)	

I0CUV	**PIC 9(6,3) COMP**	**CLOCK UPDATE VALUE (mS)**
Compiled offset	#00BE	
Typical value	#000186A0	
BT/I0-block	I0CUV 190	
BT/I0 structure	i0.cuv	
Action file	No	
Configurator	Calculated from I0BCLK	
Steer Rtn unpack	I0CUV to i0.cuv	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0CUV I0.cuv calculated by Steering Routine Algorithm: I0CUV = I0BCLK * I0CUCT Packed (for completeness)	

I0CUV2	**PIC 9(4) COMP**	**DITTO AS 2 BYTE FIELD**
Compiled offset	#00C2	
Typical value	#0064	
BT/I0-block	I0CUV2 194	
BT/I0 structure	i0.cuv2	

Action file	No
Configurator	Calculated from I0BCLK
Steer Rtn unpack	I0CUV2 to i0.cuv2
Steer Rtn pack	No
Steer Rtn usage	Unpacked
Nucleus usage	vc5ta.c, read
$MONITOR usage	No
How emulated	No attempt to unpack I0CUV2 I0.cuv2 calculated by Steering Routine Algorithm: I0CUV2 = I0BCLK Packed (for completeness)

I0CUCT **PIC 9(4) COMP** **CLOCK UPDATE COUNT**

Compiled offset	#00C4
Typical value	#0001
BT/I0-block	I0CUCT 196
BT/I0 structure	i0.cuct
Action file	No
Configurator	Calculated from I0BCLK
Steer Rtn unpack	I0CUCT to i0.cuct
Steer Rtn pack	No
Steer Rtn usage	Unpacked
Nucleus usage	No
$MONITOR usage	No
How emulated	No attempt to unpack I0CUCT I0.cuct calculated by Steering Routine Algorithm: I0CUCT = I0CUP/I0BCLK Packed (for completeness)

I0SWCT **PIC 9(4) COMP** **SWAP COUNT**

Compiled offset	#00C6
Typical value	#0001
BT/I0-block	I0SWCT 198
BT/I0 structure	i0.swct
Action file	No
Configurator	Calculated from I0BCLK
Steer Rtn unpack	I0SWCT to i0.swct
Steer Rtn pack	No
Steer Rtn usage	Unpacked
Nucleus usage	vc5ta.c, read
$MONITOR usage	No
How emulated	No attempt to unpack I0SWCT I0.swct calculated by Steering Routine Algorithm: I0SWCT = I0SWIN / I0BCLK Packed (for completeness)

I0PARMS **PIC X(8)** **MACHINE DEPENDANT INFO.**

Compiled offset	#00C8

Typical value	#0000000000000000
BT/I0-block	I0PARM 200
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00

I0OPID **PIC X(4)** **OPERATOR ID OF LAST UPDATE**

Compiled offset	#00D0
Typical value	"XXXX"
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	Set to $$OPID
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00000000 rather than #*xxxxxxxx*

FILLER **PIC X(6)**

Compiled offset	#00D4
Typical value	#000000000000

I0END **PIC S9(4) COMP** **NUMBER OF EDIT AREAS**

Compiled offset	#00DA
Typical value	#0047
BT/I0-block	I0END 218
BT/I0 structure	No
Action file	No
Configurator	Number of Edit Areas
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #0000 rather than #*xxxx*

I0FORM **PIC S9(2) COMP** **ADDRESS FORMAT**

Compiled offset	#00DC
Typical value	#01
BT/I0-block	I0FORM220

BT/I0 structure	No
Action file	No
Configurator	Copied from AMFORM
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01

I0IBM	**PIC S9(2) COMP**	**IBM DIRECTORY LABEL SUPPORT REQUIRED**
Compiled offset	#00DD	
Typical value	#00	
BT/I0-block	I0IBM	221
BT/I0 structure	No	
Action file	No	
Configurator	Set to 1 if any AVLTYP = 1	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0DVER	**PIC S9(2) COMP**	**CONTROL BLOCK VERSION (OBSOLETE)**
Compiled offset	#00DE	
Typical value	#02	
BT/I0-block	I0DVER	222
BT/I0 structure	No	
Action file	No	
Configurator	1 if V5.0; 2 if V6.0	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #00 rather than #02	

I0SERV	**PIC 9 COMP**	**1 = FILE SERVER SYSTEM**
Compiled offset	#00DF	
Typical value	#00	
BT/I0-block	I0SERV	223
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	

Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00

I0DATE	**PIC S9(6) COMP**	**DATE OF LAST UPDATE**
Compiled offset	#00E0	
Typical value	#0F4370	
BT/I0-block	I0DATE	224
BT/I0 structure	No	
Action file	No	
Configurator	Set to $$DATE	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #000000 rather than #*xxxxxx*	

I0NDRI	**PIC S9(4) COMP**	**NO. OF UNIT DESCRIPTIONS**
Compiled offset	#00E3	
Typical value	#0000	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0NSEG	**PIC S9(4) COMP**	**CENTRAL SEGS FOR FILE EXEC.**
Compiled offset	#00E5	
Typical value	#0000	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0DFBA	**PIC 9 COMP**	**1=DYNAMIC DC/DF-BLOCK ALLOC.**
Compiled offset	#00E7	

Typical value	#01
BT/I0-block	No
BT/I0 structure	No
Action file	DYNAMIC DC/DF-BLOCK ALLOCATION
Configurator	Used during control block allocation
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01 DC/DF block allocation is always dynamic

I0TIME	**PIC X(8)**		**TIME OF LAST UPDATE**
Compiled offset	#00E8		
Typical value	"HH:MM:SS"		
BT/I0-block	I0TIME	232	
BT/I0 structure	No		
Action file	No		
Configurator	Set to current time		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will be #00's rather than non-zero		

I0DESC	**PIC X(8)**		**SHORT M/C DESC**
Compiled offset	#00F0		
Typical value	"CLIENT "		
BT/I0-block	I0DESC	240	
BT/I0 structure	No		
Action file	No		
Configurator	Set to AMDESC		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will be #00's rather than non-zero		

I0LDES	**PIC X(15)**		**LONG M/C DESC**
Compiled offset	#00F8		
Typical value	"WINDOWS NT "		
BT/I0-block	I0LDES	248	
BT/I0 structure	No		
Action file	No		
Configurator	Set to AMLDES		

Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed at by BTNAM
Nucleus usage	No
$MONITOR usage	No
How emulated	New field i0.ldes_syn required Field i0.ldes_syn initialised to value of registry key: **+ConfigurationDescription** Default value = "Windows NT" Field i0.ldes_syn packed back to I0LDES

I0DGEN **PIC S9(2) COMP** **GENERATED CONFIG FILE FLAG**

Compiled offset	#0107
Typical value	#01
BT/I0-block	I0DGEN 263
BT/I0 structure	No
Action file	No
Configurator	Always set to 1
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01

I0NLIST **OCCURS 20 PIC X(3)** **NUCLEUS FILE CODES NOT INCLUDING 000**

Compiled offset	#0108		
Typical value	See below (note that the 1st entry is not used)		
BT/I0-block	I0NLST 264		
BT/I0 structure	No		
Action file	Load timer	0267	SV9
	Load automatic OPID/TERM	0270	AUT
	Load $BYE	0273	BYE
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	Used when loading nucleus components		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	3 entries patched into I0NLIST and then used as normal		

I0MESS **OCCURS 10 PIC X(40)** **ALSO PART R/DEF AS I0MSCR PIC 9(4) MAX NO SCRNS**

Compiled offset	#0144
Typical value	10 lines of 40 characters per line
BT/I0-block	I0MESS 324
BT/I0 structure	No
Action file	No
Configurator	Accepted as bootstrap message

Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	Pointed to by BTBTMS
Nucleus usage	No
$MONITOR usage	No
How emulated	New fields i0.mess0_syn to i0.mess9_syn required Fields i0.mess0_syn initialised to value of registry key: **+InitialMessage0** **+InitialMessage1** **+InitialMessage2** **+InitialMessage3** **+InitialMessage4** **+InitialMessage5** **+InitialMessage6** **+InitialMessage7** **+InitialMessage8** **+InitialMessage9** Default value = SPACE Fields i0.mess0_syn etc. packed back to I0MESS

I0MSCR	**PIC 9(4)**	**MAXIMUM NUMBER OF SCREENS**
Compiled offset	#0168	
Typical value	" 7"	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Compared with BTCSCR	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No	

I0SMEM	**PIC X(15)**	**FILLER – NOT USED**
Compiled offset	#02D4	
Typical value	#00000000000000000000000000000000	
BT/I0-block	I0SMEM	724
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0DDAA **PIC 9 COMP** **1=DYNAMIC DA-BLOCK ALLOC.**

Compiled offset	#02E3
Typical value	#00
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	Set if 300, 400 or 700 units present
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00. This field has always been reserved for future use.

I0HDCF **PIC 9 COMP** **1=DC/DF-BLOCKS IN HIGH MEMORY**

Compiled offset	#02E4
Typical value	#00
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00 The DC/DF blocks are always in high-memory.

I0DLKA **PIC 9 COMP** **1=DYNAMIC LOCK-TABLE ALLOC.**

Compiled offset	#02E5
Typical value	#01
BT/I0-block	No
BT/I0 structure	No
Action file	DYNAMIC LOCK TABLE ALLOCATION
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01 Lock Table allocation is always dynamic

I0NUCS **PIC 9(4) COMP** **SIZE OF NUCLEUS ON DISK IN K (IF > 130 K)**

Compiled offset	#02E6
Typical value	#0000

BT/I0-block	I0NUCS 742
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #0000

I0NPB	**PIC S9(4) COMP**	**NUMBER OF PB ENTRIES**
Compiled offset	#02E8	
Typical value	#0004	
BT/I0-block	I0NPB 744	
BT/I0 structure	No	
Action file	NUMBER OF PRINT BUFFERS	
Configurator	Validated to be > 3	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	New field i0.npb_syn required Field i0.npb_syn initialised to value of registry key: **+NumberOfPrinterBuffers** Default value = Number of printers configured Field i0.npb_syn used by Steering Routine to set ez.npbs Field i0.npb_syn packed back to I0NPB (for completeness only)	

I0SPB	**PIC S9(4) COMP**	**LENGTH OF PB (EXCL. PREFX)**
Compiled offset	#02EA	
Typical value	#0084	
BT/I0-block	I0SPB 746	
BT/I0 structure	No	
Action file	LENGTH OF PRINT BUFFERS	
Configurator	Validated to be between 132 and 254	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SYSPB	
How emulated	New field i0.spb_syn required Field i0.spb_syn initialised to value of registry key: **+LengthOfPrinterBuffers** Default value = 250 Field i0.spb_syn used by Steering Routine to set ez.bsz Field i0.spb_syn must be packed back to I0SPB	

I0NLB	**PIC S9(4) COMP**	**NUMBER OF FULL LAN BUFFERS**
Compiled offset	#02EC	
Typical value	#0003	
BT/I0-block	I0NLB	748
BT/I0 structure	i0.nlb	
Action file	NUMBER OF FULL LAN BUFFERS	
Configurator	Validated for LAN configuration	
Steer Rtn unpack	I0NLB to i0.nlb	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0NLB	
	No action required (this is not used by the LAN Executive)	
	Packed BT-block value will be #0000 rather than non-zero	

I0NLSB	**PIC S9(4) COMP**	**NUMBER OF SHORT LAN BUFFERS**
Compiled offset	#02EE	
Typical value	#0003	
BT/I0-block	I0NLSB	750
BT/I0 structure	i0.nlsb	
Action file	NUMBER OF SHORT LAN BUFFERS	
Configurator	Validated for LAN configuration	
Steer Rtn unpack	I0NLSB to i0.nlsb	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0NSLB	
	No action required (this is not used by the LAN Executive)	
	Packed BT-block value will be #0000 rather than non-zero	

I0SLB	**PIC S9(4) COMP**	**DATA LENGTH OF FULL BUFFERS**
Compiled offset	#02F0	
Typical value	#0000	
BT/I0-block	I0SLB	752
BT/I0 structure	i0.slb	
Action file	No	
Configurator	Used to calculate FZBLEN	
Steer Rtn unpack	I0SLB to i0.slb	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0SLB	
	No action required (this is not used by the LAN Executive)	
	Packed BT-block value will be #0000 rather than non-zero	

I0BEND	**PIC X(4)**	**ADDRESS OF END RTN IN BOOT CONT**
Compiled offset	#02F2	
Typical value	#00000000	
BT/I0-block	I0BEND 754	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00000000	

I0XLAN	**PIC 9 COMP**	**1=EXTENDED LAN USER NOS.**
Compiled offset	#02F6	
Typical value	#01	
BT/I0-block	I0XLAN 758	
BT/I0 structure	i0.xlan	
Action file	No	
Configurator	Always set if LAN configuration	
Steer Rtn unpack	I0XLAN to i0.xlan	
Steer Rtn pack	No	
Steer Rtn usage	Unpacked	
Nucleus usage	vc0a.c, read	
$MONITOR usage	Moved to SYEXCB	
How emulated	No attempt to unpack I0XLAN Field i0.xlan initialised to #01 Field i0.xlan must be packed back to I0XLAN	

I0DCSA	**PIC 9 COMP**	**1=DYNAMIC CS-BLOCK ALLOC.**
Compiled offset	#02F7	
Typical value	#01	
BT/I0-block	I0DCSA 759	
BT/I0 structure	No	
Action file	DYNAMIC CS-BLOCK ALLOCATION	
Configurator	Used during control block allocation	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #00 rather than #01 CS-block allocation is always dynamic	

I0DPBA	**PIC 9 COMP**	**1=DYNAMIC PRINT BUFFER ALLOC.**
Compiled offset	#02F8	

Typical value	#01		
BT/I0-block	I0DPBA	760	
BT/I0 structure	No		
Action file	NUMBER OF PRINT XLATION TABLES		
Configurator	No (was dynamic control blocks pre V6.0)		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will be #00 rather than #01 Print Buffer allocation is always dynamic		

I0DLBA	**PIC 9 COMP**		**1=DYNAMIC LAN BUFFER ALLOC.**
Compiled offset	#02F9		
Typical value	#01		
BT/I0-block	I0DLBA	761	(actually defined as I0LBA, this is a bug)
BT/I0 structure	No		
Action file	DYNAMIC LAN BUFFER ALLOCATION		
Configurator	Used during buffer allocation		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will be #00 rather than #01 LAN buffer allocation is always dynamic		

I0CSIZ	**PIC 9(4) COMP**		**SIZE OF CACHE ENTRIES**
Z-FIX2	**PIC X(2)**		**SIZE OF CACHE ENTRIES**
Compiled offset	#02FA		
Typical value	#8000		
BT/I0-block	I0CBSZ	762	
BT/I0 structure	i0.cbsz		
Action file	CACHE BUFFER SIZE		
Configurator	Set to largest track size		
Steer Rtn unpack	I0CBSZ to i0.cbsz		
Steer Rtn pack	No		
Steer Rtn usage	Unpacked		
Nucleus usage	No		
$MONITOR usage	Moved to SYCSIZ		
How emulated	No attempt to unpack I0CBSZ Field i0.cbsz initialised to largest track size Field i0.cbsz must be packed back to I0CBSZ		

I0HMEM	**PIC X(4)**	**CURRENT HIGH MEMORY- BYTE ADDR**
Compiled offset	#02FC	
Typical value	#00000000	

BT/I0-block	I0HMEM	764
BT/I0 structure	i0.hmem	
Action file	No	
Configurator	No	
Steer Rtn unpack	I0HMEM to i0.hmem	
Steer Rtn pack	i0.hmem to I0HMEM	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0HMEM Field i0.hmem initialised to value of registry key: **MaximumMemory** Default value = 32Mb Packed as normal (for completeness)	

I0SDCK	**PIC 9(4) COMP**	**START CACHE BUFFERS IN K**
Compiled offset	#0300	
Typical value	#0000	
BT/I0-block	I0SDCK 768	(a bug in vc20.h defines this offset as 767)
BT/I0 structure	i0.sdck	
Action file	CACHE START ADDRESS (KB)	
Configurator	No	
Steer Rtn unpack	I0SDCK to i0.sdck	
Steer Rtn pack	i0.sdck to I0SDCK	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0SDCK Field i0.sdck initialised to #0000 Packed as normal (for completeness)	

I0IBFG	**PIC 9 COMP**	**1=INITIALISED BUFFERS (5.0)**
Compiled offset	#0302	
Typical value	#00	
BT/I0-block	I0IBFG 770	
BT/I0 structure	No	
Action file	No	
Configurator	Only used for V5.0	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0DBFG	**PIC 9 COMP**	**1=DYNAMIC FILE BUFFER ALLOC**
Compiled offset	#0303	
Typical value	#01	

BT/I0-block	I0DBFG	771
BT/I0 structure	No	
Action file	DYNAMIC FILE BUFFER ALLOCATION	
Configurator	Used during buffer allocation	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #00 rather than #01 File buffer allocation is always dynamic	

I0DBST	**PIC 9(9) COMP**	**FILE BUFFER AREA START**
Compiled offset	#0304	
Typical value	#00000000	
BT/I0-block	I0DBST	772
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00000000	

I0DBSZ	**PIC 9(9) COMP**	**FILE BUFFER AREA SIZE**
Compiled offset	#0308	
Typical value	#00000000	
BT/I0-block	I0DBSZ	776
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00000000	

I0NCAB	**PIC 9(4) COMP**	**NUMBER OF CACHE BUFFERS ALLOC.**
Compiled offset	#030C	
Typical value	#0000	
BT/I0-block	I0NCAB	780
BT/I0 structure	i0.ncab	
Action file	No	
Configurator	No	
Steer Rtn unpack	I0NCAB to i0.ncab	

Steer Rtn pack	i0.ncab to I0NCAB
Steer Rtn usage	Packed & unpacked
Nucleus usage	No
$MONITOR usage	Moved to SYNCAB
How emulated	No attempt to unpack I0NCAB Field i0.ncab initialised to #0000 Packed as normal

I0DCBA PIC 9 COMP 1=DYNAMIC CONSOLE BUFFR ALLOC

Compiled offset	#030E
Typical value	#01
BT/I0-block	I0DCBA 782
BT/I0 structure	No
Action file	DYNAMIC CONSOLE BUFFER ALLOC'
Configurator	Used during buffer allocation
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01 Console buffer allocation is always dynamic

I0BBCM PIC 9 COMP 0=BOS-BOS COMM ($REMOTE)

Compiled offset	#030F
Typical value	#01
BT/I0-block	I0BBCM 783
BT/I0 structure	No
Action file	IS $REMOTE SUPPORTED?
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will be #00 rather than #01 This field is only used at generation time.

I0HMBK PIC 9(4) COMP HIGH ADDR FOR MEM BANKS (KB)

Compiled offset	#0310
Typical value	#0000
BT/I0-block	I0HMBK784
BT/I0 structure	i0.hmbk
Action file	RAM DISK START ADDRESS (KB)
Configurator	No
Steer Rtn unpack	I0HMBK to i0.hmbk
Steer Rtn pack	No
Steer Rtn usage	Unpacked

Nucleus usage	vc0m.c, read
$MONITOR usage	No
How emulated	No attempt to unpack I0HMBK Field i0.hmbk initialised to #0000 Packed back (for completeness)

I0MEMK	**PIC 9(4) COMP**	**HIGH MEMORY ADDR (KB)**
Compiled offset	#0312	
Typical value	#0000	
BT/I0-block	I0MEMK 786	
BT/I0 structure	i0.memk	
Action file	MAXIMUM MEMORY ALLOCATION RAM HIGH ADDRESS (KB)	
Configurator	No	
Steer Rtn unpack	I0MEMK to i0.memk	
Steer Rtn pack	i0.memk to I0MEMK	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0MEMK Field i0.memk calculated by Steering Routine although this field is not used. Packed as normal (for completeness)	

I0REMP	**OCCURS 5 PIC 9(4) COMP**	**$REMOTE PARAMETERS**
Compiled offset	#0314	
Typical value	#00000000258000000000	
BT/I0-block	I0REMP 788	
BT/I0 structure	No	
Action file	TIME-OUT (T=)	0788
	DEVICE ADDRESS (D=)	0790
	BAUD RATE (B=)	0792
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will be #00's rather than non-zero These fields are only used at generation time.	

I0MBNO	**PIC X**	**MEMORY BANK VERSION CODE**
Compiled offset	#031E	
Typical value	#00	
BT/I0-block	I0MBNO 798	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	

Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed BT-block value will continue to be #00

I0MP	**PIC S9(2) COMP**	**MULTI-PROC FLAG N/0**
Compiled offset	#031F	
Typical value	#00	
BT/I0-block	I0MP	799
BT/I0 structure	No	
Action file	No	
Configurator	Set to value of AMMP (used during control block allocation)	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed BT-block value will continue to be #00	

I0SPRE	**PIC 9(2) COMP**	**SCREEN PRE-SCALE COUNT**
Compiled offset	No	
Typical value	#00	
BT/I0-block	I0SPRE	900
BT/I0 structure	i0.spre	
Action file	No	
Configurator	No	
Steer Rtn unpack	I0SPRE to i0.spre	
Steer Rtn pack	i0.spre to I0SPRE	
Steer Rtn usage	Packed & unpacked	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No attempt to unpack I0SPRE Field i0.spre initialised to #0000 although this field is not used. Packed as normal (for completeness)	

I0UCIS	**PIC X(2)**	**C-ISAM FLAG**
Compiled offset	No	
Typical value	#0000	
BT/I0-block	I0UCIS	902
BT/I0 structure	i0.ucis	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	

How emulated	No. Packed BT-block value will continue to be #0000

I0UCIC	**PIC X(2)**		**C-ISAM FLAG**
Compiled offset	No		
Typical value	#0000		
BT/I0-block	I0UCIC	904	
BT/I0 structure	i0.ucic		
Action file	No		
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will continue to be #0000		

I0UCIF	**PIC X(2)**		**C-ISAM FLAG**
Compiled offset	No		
Typical value	#0000		
BT/I0-block	I0UCIF	906	
BT/I0 structure	i0.ucif		
Action file	No		
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will continue to be #0000		

I0UCIR	**PIC X(2)**		**C-ISAM FLAG**
Compiled offset	No		
Typical value	#0000		
BT/I0-block	I0UCIR	908	
BT/I0 structure	i0.ucir		
Action file	No		
Configurator	No		
Steer Rtn unpack	No		
Steer Rtn pack	No		
Steer Rtn usage	No		
Nucleus usage	No		
$MONITOR usage	No		
How emulated	No. Packed BT-block value will continue to be #0000		

A.3 IG at the end of the I0-block (IG = Generation Control Information)

IGARCH	**PIC X**	**ARCHITECTURE CODE**
Compiled offset	#0320	

Typical value	"W"
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	Copied from AMMBOS
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGMU	**PIC X**	**MULTI-USER FLAG Y/SP**
Compiled offset	#0321	
Typical value	"Y"	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to "Y" if BTNUSE > 1	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGMP	**PIC X**	**MULTI-PROC FLAG Y/SP**
Compiled offset	#0322	
Typical value	#20	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to "Y" if I0MP nonzero	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGSTAR	**PIC 9 COMP**	**STARTER BOOT STRATEGY**
Compiled offset	#0323	
Typical value	#01	
BT/I0-block	No	
BT/I0 structure	No	
Action file	STARTER BOOTSTRAP|INSTALLATION STRATEGY	
Configurator	No	
Steer Rtn unpack	No	

Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to ZGSTAR, used when installing
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGTARG	**PIC 9 COMP**	**TARGET BOOT STRATEGY**
Compiled offset	#0324	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	TARGET BOOTSTRAP|STARTUP STRATEGY	
Configurator	Validated to be < 3	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGIPLA	**PIC X(3)**	**IPL UNIT ADDRESS**
IHIPLA	**PIC 9(3)**	**IPL UNIT ADDRESS**
Compiled offset	#0325	
Typical value	"260"	
BT/I0-block	No	
BT/I0 structure	No	
Action file	IPL UNIT ADDRESS	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Used when installing (if IGSTAR < 3), copied to SYVF1	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGINSA	**PIC X(3)**	**NOT USED**
IHINSA	**PIC 9(3)**	**NOT USED**
Compiled offset	#0328	
Typical value	#000000	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	

How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGRESA PIC X(3) DISTRIBUTION UNIT ADDRESS
IHRESA PIC 9(3) DISTRIBUTION UNIT ADDRESS
Compiled offset #032B
Typical value "260"
BT/I0-block No
BT/I0 structure No
Action file DISTRIBUTION UNIT ADDRESS
Configurator No
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage Used when installing (if IGSTAR > 2), copied to SYVF1
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGIPLV PIC S9(2) COMP IPL VOL TYPE
Compiled offset #032E
Typical value #1E
BT/I0-block No
BT/I0 structure No
Action file No
Configurator Set to AVVTYP
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage No
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGRESV PIC S9(2) COMP DISTRIBUTION VOL TYPE
Compiled offset #032F
Typical value #1E
BT/I0-block No
BT/I0 structure No
Action file No
Configurator Set to AVVTYP
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage No
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGTPED PIC S9(2) COMP 1=DIST. ON TAPE;0=NOT
Compiled offset #0330
Typical value #00

BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGUTL	**PIC 9 COMP**	**1 = SERIAL CONSOLES**
Compiled offset	#0331	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	SERIAL CONSOLES ATTACHED ?	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGTAPE	**PIC 9 COMP**	**1=TAPE NOT SUPPORTED**
Compiled offset	#0332	
Typical value	#01	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to 0 if region 33 configured; set to 1 otherwise	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGBDX	**PIC 9 COMP**	**1=BDX NOT SUPPORTED**
Compiled offset	#0333	
Typical value	#01	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to 0 if A1M or D1F formats configured, set to 1 otherwise	
Steer Rtn pack	No	
Steer Rtn usage	No	

Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGMON PIC X(2) MONITOR SUFFIX

Compiled offset	#0334
Typical value	"OR"
BT/I0-block	No
BT/I0 structure	No
Action file	MONITOR SUFFIX
Configurator	Set to "OR" normally; or "LS" if no console configured
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGMNGS PIC X(2) MONITOR GAP START

Compiled offset	#0336
Typical value	#0000
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGMNGL PIC X(2) MONITOR GAP LENGTH

Compiled offset	#0338
Typical value	#0000
BT/I0-block	No
BT/I0 structure	No
Action file	No
Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGNATF PIC X(6) NATIVE VOLUME FORMAT
IGNATX OCCURS 6 PIC X NATIVE VOLUME FORMAT

IGNATC	**OCCURS 6 PIC 9**	**NATIVE VOLUME FORMAT**
Compiled offset	#033A	
Typical value	"TAPE "	
BT/I0-block	No	
BT/I0 structure	No	
Action file	BACNAT format	
Configurator	Trailing alphabetic character removed	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGTAP	**OCCURS 4 PIC X(4)**	**TAPS REQUIRED**
Compiled offset	#0340	
Typical value	#00000000000000000000000000000000	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGBANK	**PIC 9 COMP**	**DEFAULT MEMORY BANK SIZE**
Compiled offset	#0350	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGFIXS	**PIC 9 COMP**	**1 = FIXED STACK ALLOCATION**
Compiled offset	#0351	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	

Configurator	No
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGIPLF	**PIC X(6)**	**IPL FORMAT CODE**
Compiled offset	#0352	
Typical value	"T259Z "	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to AVDESC	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGRESF	**PIC X(6)**	**DISTRIBUTION FORMAT CODE**
Compiled offset	#0358	
Typical value	"T259Z "	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to AVDESC	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGI0MP	**PIC 9 COMP**	**COPY OF I0MP**
Compiled offset	#035E	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Set to AMMP	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	

How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGSMON PIC X(2) SLAVE MONITOR SUFFIX
Compiled offset #035F
Typical value #0000
BT/I0-block No
BT/I0 structure No
Action file No
Configurator No
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage No
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGSINF PIC X(8) SLAVE INSTALL FILE
Compiled offset #0361
Typical value #0000000000000000
BT/I0-block No
BT/I0 structure No
Action file No
Configurator No
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage No
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGPREL PIC 9 COMP 1 = PRERELEASE
Compiled offset #0369
Typical value #00
BT/I0-block No
BT/I0 structure No
Action file PRERELEASE ?
Configurator No
Steer Rtn unpack No
Steer Rtn pack No
Steer Rtn usage No
Nucleus usage No
$MONITOR usage No
How emulated No. Packed IG-block value will be #00 rather than non-zero value

IGDOM PIC 9 COMP 1 = DOMAIN SUPPORTED
Compiled offset #036A
Typical value #01
BT/I0-block No

BT/I0 structure	No
Action file	No
Configurator	Flag set if domain in Configurator file
Steer Rtn unpack	No
Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	Moved to SYDOM and used by GEN and $FIND
How emulated	New field ig.dom_syn required Field ig.dom_syn initialised to 1 if a domain is configured Field ig.dom_syn must be packed back to IGDOM

IGLEVN	**PIC 9 COMP**	**BOS LEVEL NUMBER**
Compiled offset	#036B	
Typical value	#09	
BT/I0-block	No	
BT/I0 structure	No	
Action file	SOFTWARE LEVEL NUMBER	
Configurator	Level number from Bootstrap Message	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	Moved to SYVF2	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGFMTW	**PIC 9 COMP**	**1=WINCHESTER MUST BE FORATTED**
Compiled offset	#036C	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGPNID	**PIC 9 COMP**	**1=PROMPT FOR NODE-ID**
Compiled offset	#036D	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	

Steer Rtn pack	No
Steer Rtn usage	No
Nucleus usage	No
$MONITOR usage	No
How emulated	No. Packed IG-block value will be #00 rather than non-zero value

IGSEPR	**PIC 9 COMP**	**1=RAM DISK IN SEPARATE MEMORY**
Compiled offset	#036E	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGRESC	**PIC X**	**DISTRIBUTION CAPACITY (L, M, H, S OR X)**
Compiled offset	#036F	
Typical value	"S"	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	Calculated from IGRESV	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGNATN	**PIC 9(2) COMP**	**NUMBER OF BACNAT DISKETTES**
Compiled offset	#0370	
Typical value	#02	
BT/I0-block	No	
BT/I0 structure	No	
Action file	Number of BACNAT diskettes	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

IGISAM	**PIC 9(2) COMP**	**1 = C-ISAM UCI AVAILABLE**
Compiled offset	#0371	
Typical value	#00	
BT/I0-block	No	
BT/I0 structure	No	
Action file	No	
Configurator	No	
Steer Rtn unpack	No	
Steer Rtn pack	No	
Steer Rtn usage	No	
Nucleus usage	No	
$MONITOR usage	No	
How emulated	No. Packed IG-block value will be #00 rather than non-zero value	

FILLER	**PIC X(18)**	**TOTAL LENGTH IS 900 BYTES**
Compiled offset	#0372	
Typical value	#00's	