GX and VIEW$

1. Introduction

This note describes some aspects of the special GX interface for the general-purpose VIEW$ routine and the VIEW$ interface used by the 32-bit GX spooler. Other documents describe the related GX interfaces for the 32-bit equivalents of $INSPECT and $SDE.

This interface is used by VIEW$ when it detects the application is running in "GX True Window" mode (i.e. when the application has been compiled without the "CB" option).

A series of Call-And-Do operations (see below) are required for VIEW$ to interface with GX. In order for VIEW$ to use these Call-And-Do operations efficiently, and to avoid an initial delay during any sort of "intermediate file building process", VIEW$ merely builds a table containing the file offsets and line numbers at the top of all new pages within the report file. The logic that scans the file, converting a block requested from GX to a series of text lines, involves the removal of alignment patterns, expansion of Print Control Bytes and processing of Sub-Headings.

Note that the conversion of the report file to an intermediate file might be a lengthy process so a "keep the user happy" message box is displayed (using the GXMSG$ sub-routine) if the intermediate file contains 1000 lines, or more.

2. Operator Interface Changes

There are 3 main sections to the VIEW$ screen:

- the title area;
- the scrolled region;
- the base line prompt.

These are readily mapped to a GX window with the base line prompt being replaced by a set of buttons. Note that these buttons are effectively hard-coded into GX and cannot be changed by the VIEW$ sub-routine.

The title line includes a page number which indicates the page of the line at the top of the scrolled area. Note that this page number is the page number as indicated by the listing file and not the scrolled area page number.

GX creates as large a window as it can on the given screen and creates horizontal and vertical scroll bars as appropriate. The various buttons (see below) are hard-coded in GX and cannot be controlled by the VIEW$ code.

When VIEW$ is operating in text mode a page break is indicated by drawing a horizontal line immediately above the 1st line of the new page. GX can determine the 1st line of a page and draw non-intrusive horizontal line (i.e. one that does not take up a character row) to indicate a page break.

3. The VIEW$ Interface

VIEW$ does not currently have a flag to indicate if it is running from the spooler or not. (It currently differentiates a spool unit from a work unit which may not necessarily indicate the spooler). If the spooler is running on GX then VIEW$ will be run in GX mode (using the results of the GXMOD$ call to determine the GX operating mode). The spooler calls VIEW$ with an (undocumented) second parameter that specifies the timeout associated with the View Window.

The title passed to GX will be the title in PRTITL. The total number of records can be calculated from the file size and record length. The record length can be derived from the file label. A table of page numbers is sent to GX which GX uses to display the relevant page number.

The only buttons that will be required are as follows:

- Next
- Back
- Goto
- Find
- Find next
- Close

The Find string button will accept a string to look for. The string may be terminated by a """ to allow for the search string to include trailing spaces.

Goto will cause a window to pop up which will allow a particular page number, or line number, to be displayed.

The Columns function is still supported, albeit in a reduced capacity. Any columns set via the VIEW$ sub-routine are indicated by the "I" character. However, the column positions cannot be adjusted by the operator. (<BCK> etc are indicated by standard keys.)

4. GX Operations

The following GX Call-And-Do operations are required for the VIEW$ interface:

- Initialise window
- Set page start lines
- Send sub-heading text

- Send page
- Accept

See the Call-And-Do note "Text View Window (opcode 111)" for full details.

5. Differences Between text-mode VIEW$ and GX VIEW$

This section describes the differences between the "traditional" text-mode VIEW$ and the GX mode VIEW$:

5.1 Title & Sub-Title Information

The title line information, which normally occupies the top text line, is displayed in the caption area of the GX VIEW$ window. The current page number on the caption bar is updated as required.

Note that neither the text-mode VIEW$, nor the GX-mode VIEW$, display the "current line number".

Sub-headings are displayed after the main heading in a darker grey box within the Window Heading.

The title includes the page Number.

5.2 Baseline and Buttons

The dynamic, and somewhat confusing, text-mode VIEW$ base-line has been replaced by a series of buttons in the GX-mode VIEW$.

The **relative** "go to" line-number (nnn/-nnn) has been replaced by the GoTo button (followed by the selection of an **absolute** line number). Furthermore, the GX-mode VIEW$ includes a "Goto Page" function, which is not available with the text-mode version.

The "Columns" option is **NOT** supported in the GX-mode VIEW$. However, a horizontal scroll bar is displayed if the line-width is larger than the current window width. [Note the the column interface is still supported by the VIEW$ sub-routine. Columns are indicated by a "|" character. However, the column positions cannot be adjusted by the operator.]

The "Find" option is supported and has been extended to support a Find from current/top position and an option for a case-insensitive search. The "Find Next" button is only enabled after a successful previous find.

The "Resume" (to application e.g. spooler) option has been replaced by a Close button. Note the standard windows close box and the <ESC> key also close the GX-mode VIEW$ window.

The "Next" and "Back" baseline options have been replaced by Next and Back buttons.

The "CR for Next" or "CR for Back" function of the <CR> key is not supported in the GX-mode VIEW$ window.

The PAGE-UP, PAGE-DOWN, UP-ARROW and DOWN-ARROW keys are supported in GX -mode to allow navigation through the file. Note that GX-mode VIEW$ also supports the standard vertical scroll bar functions.