The WINDOW Body

The window body contains details of the text, fields and buttons that are to be displayed and accepted within the window, and may contain options that control its general layout. It follows the Window Options and is before the optional ROUTINES SECTION. The format of the window body is as follows:

```
LI CL Longname | LI CL Name Pic Options...

(Detail Lines)
```

The window body may be introduced by a comment line that may be helpful in laying out the source code. The compiler will ignore any line at the start of the window body that begins with the characters LI. This is then followed by the window's detail lines.

The detail lines define both the window's form (i.e. fixed text items with their boxes) and data items (i.e. variables that are to be displayed and accepted). A detail line consists of one or more text items, and/or one or more data-items and/or one or more buttons. Text items specify fixed text which will be displayed at a fixed position on the screen. Data-items specify variable data that may be accepted and displayed on the screen in various positions. Buttons specify the button to be displayed for windows running under GX.

1. Text Items

Text items are used to display fixed text on the screen at a given line and column position. The general form of a text-item is as follows:

```
line col "text" [options]
```

where *line* is the line number (counting the top line as 1) at which the text is to be displayed, *col* is the column number at which the text is to be displayed, and *text* is the text to be displayed at that position, as offset by the BASE AT window option statement. For example, coding:

```
10 20 "Customer #"
```

would cause the characters **Customer #** to be displayed on the screen at line 10, column 20.

Fixed text items are normally displayed on the screen when the frame is executed. The basic screen form, composed of all the text items, boxes and lines of all the coded windows within the frame is displayed in this way, providing the operator with an "empty" screen. Text items are always displayed at the coded line and column positions, and are therefore unaffected by the SCROLL verb.

Line and column numbers must both be unsigned positive integer literals. Line numbers may be coded from 2 to 46 inclusive. Column numbers may be coded from 2 to 127 inclusive, so long as the item being displayed, as offset by the BASE AT statement, would not cause column 131 to be exceeded. Note that is the application is to be run on a screen in text mode, you must ensure that the screen is wide enough.

Only the display attribute option described below for the field options are allowed on text fields.

2. Data Items

A data-item specifies a variable that is to be displayed and/or accepted from the screen. The general form of a data item is:

```
    line col name [options]
```

Where *line col* is the line and column number at which the item is to be displayed, as offset by the BASE AT statement, *name* is the variable name of the item to be displayed and/or accepted, and *options* consists of one or more option clauses which specify how the item should be processed.

Line and Column Numbers must be unsigned positive integer literals. Line numbers may be coded in the range of 2 to 46 inclusive. Column numbers may be coded in the range 2 to 127 inclusive, as long as the item displayed would not cause column 131 to be exceeded. It should be noted that the line and column position coded will be modified at run-time by the operation of the SCROLL verb, as documented earlier in this chapter, and offset by the BASE AT statement.

2.1 Variable Name

The Variable Name is a **reference** to an existing variable that has previously been declared within the Data Division or by an ACCESS statement.

2.2 Picture Clause

A picture clause may b coded for a field without the preceding PIC clause. Picture clauses or fields that are defined outside the window do not need to be included though it is preferable to do so. Data items declared within the window must have a name that begins with the window-id. For data items declared outside the window the picture clause must be the same as that of the declaration with the exception of PIC X fields under GX. For these fields it is possible to declare a field display area within the window that is smaller than the actual field size. GX will allow a windowing of this longer field within the display length, allowing the use of the cursor keys to move to the start and end of the field.

2.3 Field Options

Field Options may be coded following the picture clause in order to affect the way the field is processed at run-time These are:

DIS	Display only field, the field is only displayed
PRO	Protected field, the field cannot be entered or amended
NOE	No-edit field, the field may not be amended
NUL	Null allowed (i.e. the field may be left blank)
TAB	Stops the cursor at this field after a <SKIP>
CHK	Perform duplicate record check after this field
UF1	Enable UF1 when this field is accepted
UF2	Enable UF2 when this field is accepted
UF3	Enable UF3 when this field is accepted

NSC	The field is non-scrolled in an otherwise scrolled window
CNV	Lower-case characters to be converted to upper case
YN	Yes/No validation to be performed on field input
YNC	Yes/No validation tick box
TXT	A text item specified in the form of a variable
RJF	Right-justify field
FMT	Specifies the display formatting to be applied to the field
HOT	Specifies field auto-inputs without need to key <RET>.
TTL	Display field using display attribute 5 (Titles).
ERR	Display field using display attribute 8 (Error message).
A12	Display field using attribute 12.
A13	Display field using attribute 13.
A14	Display field using attribute 14.
PAS	Password field
NSL	Non-scrolled label
SCL	Scrolled label
OPT	Optional field
PDT	Pop-up date field
PDA	Pop-up data field (auto-accept)
CON	Concatenate

These options are discussed below.

2.3.1 DIS Option

DIS (Display) is similar to a protected field in that the operator will not be able to modify it. The display option is used, however, to indicate that the field contains valid data and should not be initialised during ADD mode. If despite this, a default routine has been coded, then this will nevertheless be called.

2.3.2 PRO Option

PRO (Protected) specifies that the field may not be modified by the operator under any circumstances. During ADD mode, the field will be initialised to binary zeros, spaces or "0". If a default routine has been coded, it will be called to initialise the field instead.

2.3.3 NOE Option

NOE (No Edit) prohibits the user from modifying the field in MNT and EDT modes, but allows normal access to the field during ADD mode. It is used to protect fields that may not be changed after the record has been written.

2.3.4 NUL Option

NUL (Null allowed) specifies that the field may be left blank, in the case of a character field, or zero, in the case of a numeric field. In the case of dates, this option allows a null date, " / / ", to be entered. If this option is not coded the window manager will re-input the field if any blank or zero values are entered, or defaults accepted by the operator.

2.3.5 TAB Option

TAB causes the cursor to stop at the current field whenever the <SKP> key is used, and therefore acts as a tab-stop. This allows fields to be arranged in blocks within the window, allowing the operator to skip from one block to the next. If the TAB option is not used in the window, then the <SKP> operation will automatically jump to the last field. This option has no effect when used on GX.

2.3.6 CHK Option

CHK causes a duplicate-record check to take place during processing of the associated field in ADD and INS modes. This option is used to check that the record being created does not already exist on the database. In effect, a duplicate key check takes place using the index key entered so far. If this key is found and ENQ mode is enabled, the baseline message:

    Record key already exists - Enquire?

is displayed. The operator may then enquire on the duplicate record. This option should be coded **after** all primary index key fields have been entered. This option is particularly recommended for maintenance programs, since it allows the operator a fast way of enquiring on records without having to use the ENQ key to switch between modes.

2.3.7 UF1 Option

UF1 enables the use of the <UF1> key when the associated field is accepted. If coded, a validation routine should be written within the routines section, which should examine system variable $FUNC to check whether this key had been entered, and if so, take the appropriate action. In GX setting the UF1 function causes an associated button to appear adjacent to the field if it is not a scrolled field. Pressing of this button is equivalent to keying <UF1> from the keyboard.

2.3.8 UF2 Option

UF2 enables the use of the <UF2> key when the associated field is accepted.

2.3.9 UF3 Option

UF3 enables the use of the <UF3> key when the associated field is accepted.

2.3.10 NSC Option

NSC specifies that the field is not to be scrolled within a window coded using the SCROLL clause. If the SCROLL clause has not been specified, all fields are automatically non-scrolled, and there is therefore no point in coding this option. Option NSC is used to define those fields that will be displayed in the non-scrolled portion of the window (see WINDOW options).

2.3.11 CNV Option

CNV specifies that any lower-case characters input are to be converted to upper case before the field is validated, or any other processing is carried out.

2.3.12 YN Option

YN causes the field to be re-input unless it is Y or N.

2.3.13 YNC Option

YNC specifies a check box for windows displayed on GX. The YNC attribute can be used instead of the YN attribute and replaces the one character input field with a window tick box instead. A tick is mapped to "Y" and a blank is mapped to "N".

It is very important to note that any YNC field should be initialised to either Y or N before entering the window. Otherwise the box will display as empty but the input field present within Speedbase will not contain an "N" and you will not be able to move past the field without either ticking the box or blanking out what already appears to be a blank box. All rules that apply to YN fields also apply to YNC fields and you can still key "Y" or "N" using the keyboard.

2.3.14 TXT Option

TXT specifies that the field variable is to be displayed as text in the scrolled-text attribute unless the TXT option is immediately followed by the NSC option, in which case it is displayed in the non-scrolled-text attribute. This option is used where the text to be displayed varies under program control. Once a text field is displayed on a window it will not be redisplayed even if its contents have been changed. In particular the field may not be the subject of the HIDE or SHOW verbs. If you want to redisplay the field then you must used the NSL option.

2.3.15 RJF Option

RJF causes the field to be accepted, stored and displayed in right-justified form.

2.3.16 FMT Option

FMT "*options*" specifies the output formatting to be applied to a numeric field. The table lists the five groups of *options*.

Option	Condition	Output Formatting	As input	As output
B	*Field* = 0	**Blank** when zero	00.00	
C	*Field* < 0	Trailing **CR**	-12.34	12.34CR
<		Enclosed in **()**		(12.34)
-		Trailing **-**		12.34-
D	*Field* > 0	Trailing **DR**	+56.78	56.78DR
>		Enclosed in **()**		(56.78)
+		Trailing +		56.78+
,	None	**Comma** insertion	90123	90,123
$	None	Leading **dollar**	45.67	$45.67
0		**Zero** fill	8.90	0008.90
*		**Asterisk** fill	1.23	***1.23
L	Date only	**Long** date	DD/MM/YYYY	DD/MM/YYYY
8	Date only	Short date **8** byte input	DDMMYYYY	DD/MM/YYYY
6	Date only	Short date **6** byte input	DDMMYY	DD/MM/YY

The Output Formatting Options

You may specify no more than one option from each group in any set of options. Examples of invalid options would include:

```
BC<+,$
->+*
BCD,0*
```

Examples of valid sets of options and the resultant output formatting are as follows:

Options	Field as Input	Field as Output
BCD,	-678.90	678.90CR
	0.00	
	+1234.56	1,234.56DR
B<,	+789.01	789.01
	-7890.12	(7,890.12)
B-+$,	+345.67	$345.67+
	$890.12	$890.12-
-+0	+345.67	00345.67+
	-8901.23	08901.23-
CD*	+456.78	**456.78DR
	-901.23	**901.23CR

Examples of Output Formatting

A field with a DATE clause is always considered to be of FMT “L”/

2.3.17 HOT Option

HOT Field entry terminates as if <RET> entered when last byte of the field is keyed. This saves having to key <RET> for the field.

2.3.18 PAS Option

PAS allows password style fields to be entered displaying an "*" for each character entered. This option is only available on windows displayed by GX.

2.3.19 NSL Option

NSL causes a data item to be displayed as if it was a non-scrolled text label item. This option is only available on windows displayed by GX. A non -

2.3.20 SCL Option

SCL causes a data item to be displayed as it were a scrolled text label item, that is the labels displayed above scrolled columns, This option is only available on windows displayed by GX.

2.3.21 OPT Option

OPT allows much greater control over optional fields in windows. An EXIT WITH 1 from the B- routine of a field causes the input or display of that field to be skipped. (see ROUTINES section). The 'OPT' attribute tells GX that a field is optional and that if it is suppressed from a B- routine it needs to be removed from the window. The OPT option can be used in conjunction with an NSL field to allow the field prompts to also be suppressed if required. This option is only available on windows displayed by GX.

2.3.22 CON Option

CON concatenates so that they appear as one field in a window displayed under GX.

2.3.23 TTL, ERR, A12, A13, A14 Options

These are field colour display attribute options. If the first field of a window is a text field or text item with the A12 attribute, for windows displayed under GX this field is taken as the window caption,

2.3.24 PDT Option

PDT on a field indicates a pop-date field that is only available on GX. This attribute can be used on date field or character fields that could contain a date value. It must not be used in conjunction with the PDA or UF1 options.

If a field has the PDT option set, and pop-up calendars are available under GX, then a button will appear adjacent to the window on the field if it is in the non-scrolled area. On pressing this button or keying <UF1>on a scrolled field, a pop-up calendar will be shown. For the standard pop-up calendar, GXCAL.EXE, the user can scroll through the years and months to select a specific date, date selection can be made in a number of ways:

- You can use the right mouse button, select to copy the date to the windows clip board (in either dd-mm-yy or dd-mm-yyy format), then exit the calendar look-up and the selected date is returned to the application,
- You can press the F2 key to return the selected date (dd-mm-yy format),
- You can press the F3 key to return the selected date (dd-mm-yyyy format),
- You can 'double' click on the selected date.

Please note, in the majority of cases the format in which the date is selected (i.e. dd-mm-yy or dd-mm-yyyy) has no bearing on the application as it automatically converts dates from dd-mm-yy to dd-mm-yyyy format.

2.3.24.1 GX Settings to Enable Date Pop-up

The following options in the [general] section of the GX.INI file enable and configure the Calendar Look-Up option. Please note, only the first ini file setting (EnableDatePopup) in the list below is required to configure your system to use the calendar look-up feature.

EnableDatePopup

This setting enables/disables the Calendar Look-Up option.

If this setting is "On" a pop-up iconic button will be displayed to the left of the input field. When this button is clicked, or when the <UF1> key is pressed, the Calendar Look-Up Program (see below) will run. If the field contains an initial date value GX will pass this as a parameter to the Calendar Look-Up Program. The Calendar Look-Up

Program will default to the passed date, if one is supplied. When the Calendar Look-Up Program exits the selected date will be returned to the Speedbase field via the clipboard.

The default setting is “Off”.

DatePopupProgram
This setting allows the name of the Calendar Look-Up Program to be specified. The default setting is GXCAL.EXE.

The guidelines described in section 4 of technical note in366-speedbase-calendar-pop-up should be followed when writing a bespoke Calendar Look-Up Program.

DatePopupBitmap
This setting allows the icon associated with the active “Calendar look-up” button to be changed. For most applications the default button displayed by GX should be adequate.

DatePopupBitmapGreyed
This setting allows the icon associated with the greyed-out “Calendar look-up” button to be changed. For most applications the default button displayed by GX should be adequate.

DatePopupLegacyTabCRProcessing
This setting should agree with the “LegacyTabCRProcessing” setting in the “..\Global\Client\Customisations” key of the registry on the server running GLOBAL.EXE (or GlobalClientService.exe).

The default setting is “Off”.

DatePopupCenturyStartDate
This setting, which is only supported by GX V4.3, and later, should agree with the “Century Start Date” setting in $CUS.

The default is 60.

Special Considerations for Microsoft Vista

The “MonthView” control required by GXCAL.EXE is not available on Vista. The appropriate file, MSCOMCT2.OCX is available from the Windows web-site. To use GXCAL.EXE on Microsoft Vista download this file from the Microsoft web-site and register the control.

See http://support.microsoft.com/kb/297381 for more details.

To register the control simply run REGSVR32.EXE and specify the path/name of the control.

The installation/registration procedure is as follows:

- Copy MSCOMCT2.OCX to a suitable directory, e.g. C:\WINDOWS\system32.

- Register the control using REGSVR32 C:\WINDOWS\system32\MSCOMCT2.OCX.

Important Note: The full path name is essential. Running the command REGSVR32 MCCOMCT2.OCX is not sufficient even though REGSVR32 indicates the registration has been successful.

2.3.25 PDA Option

PDA is a data pop-up option similar to the PDT option above except that it does an automatic accept, as with the HOT option on the field.

2.4 Data Item Processing

The processing applied to each data item depends on the mode the window is operating in at the time, the field options in force, and the operation of the field level routines within the routines section. The usual order of events is as follows:

1. The before-routine is executed to suppress the field
2. The field is initialised, to spaces or low-values, if it is being processed for the first time in ADD or INS modes
3. The default routine is executed to allow a new value to be placed in the field
4. The field is displayed and accepted
5. The validation routine is executed
6. If the CHK option has been coded for the field, a duplicate key check takes place at this time

The field options cause some steps to be omitted. The **DIS** option disables field initialisation, step 2, and stops the field from being accepted in all modes other than ENQ. The **PRO** option prevents the field from being accepted in all modes other than ENQ. The **NOE** option prevents the field from being accepted in MNT and EDT modes. The **CHK** option causes step 6 to take place. The **UF1**, **UF2** and **UF3** options enable the corresponding keys if the field is accepted.

Consider the effect the various modes have on the processing of data items:

ENQ	starts an enquiry and data entry of index key fields
DSP	displays existing records from the database
MNT	causes modification of an existing record
DEL	causes deletion of an existing record
ADD	causes creation of a new record
INS	causes creation and insertion of a new record

EDT causes creation of a new record by editing an existing one

In **ENQ** mode the before-routine is not called, since index key fields may not be suppressed at run time. The current index fields are accepted irrespective of options NOE, DSP and PRO.

In **DSP** mode, fields within the record are display-only. The before-routine is called to suppress display of certain fields, the validation and default routines being ignored. If it is necessary to default certain fields prior to display, this should be done using a fetch routine in the routines section. The field options have no effect on processing during DSP mode.

In **MNT** mode, fields coded with NOE, DSP and PRO options are not accepted. It should be noted, however, that the default routine is called. This feature may be used to re-calculate a protected field, such as a line total, during the maintenance process. If a field is neither accepted nor re-defaulted using its default routine, then it will **not be redisplayed** during processing.

In **DEL** mode, no field-level processing takes place.

In **ADD** and **INS** modes, fields coded with DIS and PRO options are not accepted. Fields coded with the DIS option are also not cleared at step four. If the CHK option has been coded, a duplicate key check will take place at step six.

In **EDT** mode, fields coded with DIS, PRO and NOE options are not accepted.

3. Buttons

A button, which is only available on GX, is coded as a text string or data item with the field option BTN and Unn as follows:

```
li cl Button-Label BTN Unn
```

where *li* is the Line number where the button is to appear, *cl* is the column number where the button is to appear, *Button-Label* is a text literal or a text variable to be displayed on top of the button, BTN is a clause indicating the segment is a button, and U*nn* indicates the function number the button is to generate, where *nn* is in the ranges 1-21, and 50-99 inclusive.

3.1 Button Label

Button-Label can be a Text Literal or a character variable. Note that Buttons are created when the window is displayed, so it is essential that you set the variable up before the window is displayed or entered. If a Button title is null, then it regarded as being not displayable, which allows you to suppress unnecessary buttons. Simply move spaces to the variable containing the Button Title before displaying or entering the window. There are GX functions available to modify button captions.

3.2 Button Function

The function number is the value returned in system variable $FUNC when the button is clicked. Each button should therefore have a separate function number to allow the button click to be identified. Numbers in the range 1-21 are Standard System Functions such as <DEL> ($FUNC = 13), and <INS> ($FUNC = 17). Functions in the range 50-99 are Applications Function, which allow you to define up to 50 further functions for any given window.

Window Manager automatically manages Buttons for System Functions (U01-U21). If, for example, you code an explicit button for Delete (using the BTN U13 Clause>, then the user will be able to delete a record by clicking on your button, by using the <DEL> key (Usually mapped to F7 on the keyboard), or by clicking on the Toolbar Delete Button. As Window Manager already knows how to process the <DEL> function, you do not need to write any other code. Note however that you cannot use the ENABLE or DISABLE verbs to enable or disable system functions, as these at all times operate directly under Window Manager's control.

When you create a button for an Application function (U50-U99), then you will need to write some code in the R-FUNC entry-point to process your function (See ROUTINES SECTION). You will probably also want to use the R-START entry-point (See ROUTINES SECTION) to enable or disable the buttons depending on the state of processing.

When an Application Button is clicked, Window Manager simply calls the R-FUNC routine within the routines section to allow the application to deal with the operation. If no R-FUNC routine exists, or no code has been written to deal with the function, then no action occurs.

Note that whenever you explicitly define one or more buttons for a window, then Window Manager automatically suppresses the default <OK> and <CANCEL> buttons.

3.3 GX Search Buttons

GX Search Buttons are only available on GX and are special Speedbase buttons that are appended to a normal field and displayed in the form of an <F1> drop-down.

A Search Button is identified by a "normal" button (i.e. BTN type) with text-string consisting of a single "#" character. The button field must be placed at exactly the same start location as the associated field. The data field MUST be a non-scrolled item. For example:

```
13 02 DATE        D          NSC
13 02 "#"                    BTN U50
```

The Search Button is handled as any other general-purpose button (i.e. the function number must be trapped in the R-FUNC routine, and can be enabled/disabled as required).

3.4 Interrupt Button

The interrupt button of a window can be used to emulate the keying of <CTRL G> TO interrupt a program. This button is defined with the

```
BTN U-7
```

Clause for example

    12 02 “Break   “   BTN U-7

On normal display of the window this button will be disabled. The B$GX-7 routine can be used while such a window is displayed (but not while in the window) to enable the U-7 button and disable all other buttons. The program can continue in a loop containing a TEST$ call outside the window itself. If the user presses the U-7 button then TEST$ will behave as if <CTRL G> has been pressed. To restore the window either re-enter or redisplay it.

For example W1 is a window that contains a U-7 button with text of “Stop Printing”,

```
DO
      DISPLAY WINDOW W1
      ENTER WINDOW W1
      ON EXCEPTION                  * Window not cleared
            IF $$COND = 4           * Print button pressed
                  CALL B$GX-7 * Enable stop printing button
                  DO
                        PERFORM BA-PRINT-A-RECORD
                        ON EXCEPTION
                              PERFORM BB-STOP-PRINTING
                              IGNORE EXCEPTION
                              FINISH
                        END
                        CALL TEST$
                        ON EXCEPTION      * User wishes to terminate print
                              PERFORM BB-STOP-PRINTING
                              IGNORE EXCEPTION
                              FINISH
                        END
                  ENDDO
            END
      END
ENDDO
```

3.5 Iconic Buttons

GX supports Iconic Buttons (i.e. Buttons where the text on the BTN field is replaced by a bitmap with a caption). Establishing the following special format text label specifies an Iconic Button:

"~I*awwwddd*"

where:

I	Iconic button label specifier
a	Accelerator character
www	Button width (in pixels)
ddd	Button depth (in pixels)

Note that Iconic Buttons do not conform to the normal practice in GX where window items are sized in characters. In particular each button is not assumed to occupy a single lines (as all the other window items are) and can in fact span several lines. This means care must be taken when positioning these buttons so as not to interfere with other items on the window, and if the buttons are placed at the bottom of the window some "dummy" labels must be included to allow for the increased depth of the window.

Once the buttons have been configured on the window the icons to be used are setup using the BTICN$ call. Note that an icon file (i.e. .ico, .icn) and **not** a bitmap file (i.e. .bmp) must be specified for the buttons. This requirement is because icon files have a special invisible colour that allows them to blend into any Windows background colour.

You may optionally also use J, K and L instead of I as above to indicate an iconic button. They differ from the I iconic button as follows:

Letter identifier	**Right-justified**	**Field Depth Height**
I	No	No
J	Yes	No
K	No	Yes
L	Yes	Yes

4. Special Embedded GX Options

These options provide an interface to GX to instruct GX to build special constructs or to give special instructions. They are usually embedded in the window as a text string starting with the "~" character as follows:

```
li cl text
```

where *li* is the Line number, *cl* is the column number and *text* is the GX text instruction that may be literal or a field followed by the TXT option.

The special GX options available are:

Option	**Function**
B	Bitmap

C	Fixed list combo box
D	Editable combo boxes
G	Button only group box
H	Button Group Boxes
I	Label identifier
L	Radio buttons – left text
M	Field new character accept timeout
P	Progress windows
R	Radio button – right text
T	Window Timeout
W	Window width modification

4.1 Bitmap (B)

The Bitmap Image Area is defined by a label within the window. The position of the first character in the label defines the top left hand corner of the bitmap. The text of the label defines the bitmap index number and the size of the bitmap, as follows:

"~B*idd*.........."

where ~B indicates an Bitmap Image Area, *i* is the Bitmap Index number (between 1 and 9); and dd is a two digit numeric value indicating the depth (in lines) of the Bitmap Image Area.

The total width of the text in the label specifies the width of the Bitmap Image Area. Thus, the text in the label may have to be padded with extra characters to achieve the correct length.

You must insure that the window has enough depth to include the entire image area. To achieve this you may need to display a dummy X(0) field on the line(s) following the Bitmap Image Area.

The bitmap files (.bmp) to be displayed are held on the PC that is running GX in a series of folders that are specified via the section [images] in the GX.INI file. Each setting in this section of the GX.INI file is of the form:

Folder*N*=*folder name of bitmap files*

For example:

```
[images]
Folder1=f:\bitmaps\products\
Folder2=f:\bitmaps\faces\
```

4.2 Fixed List Combo Box (C)

A GX Fixed List Combo Box is implemented by a combination of a special label field and the ACMBO$, CCMBO$ and DCMDO$ sub-routines.

A GX Combo Box is defined as a data field that is overlayed (has the same row and column) with a button containing the following special text:

"~C*ndd*"

where:

C	GX Fixed List Combo Box specifier
n	GX Combo Box index number (1 - 9; A - Z)
dd	Depth of list box in lines (a leading 0 is required if less than 10)

When this combination of button and data field definitions are detected by GX, the normal button and edit control displays are suppressed and a combo box is created instead. The creation of a Fixed List Combo Box means that only information in the list can be selected (i.e. the combo box doesn't act as a normal field where free-format text can be supplied). The list is normally created in the B- routine for the data-field, by calling ACMBO$ repeatedly to build the list, and in the R-FUNC routine when the GX Combo Box button is selected.

4.3 Editable Combo Box (D)

A GX Editable Combo Box is implemented by a combination of a special label field and the ACMBO$, CCMBO$ and DCMDO$ sub-routines.

A GX Editable Combo Box is defined as data field that is overlayed (same row and column) with a button containing the following special text:

"~D*ndd*"

where:

D	GX Editable Combo Box specifier

n	GX Combo Box index number (1 - 9; A - Z)
dd	Depth of list box in lines (a leading 0 is required if less than 10)

When this combination of button and data field definitions are detected by GX, the normal button and edit control displays are suppressed and a combo box is created instead. The creation of an Editable Combo Box means that a response that is not present in the drop-down list can be entered by the operator. The list is normally created in the B- routine for the data-field, by calling ACMBO$ repeatedly to build the list, and in the R-FUNC routine when the GX Combo Box button is selected.

4.4 General Purpose Group Box (G) and Button Group Boxes(H)

The BOX window option and the related special label text can be used successfully to create "general-purpose" GX Group Boxes that contain fields, labels, radio buttons etc., but cannot be used to contain groups of buttons. The button group box must be used for this. The specialised Button Group boxes are positioned and sized using the same algorithm as buttons themselves. Any buttons placed inside a Button Group Box will always remain at the same relative position inside the box despite any changes to the font or other window parameters. Note that Button Group Boxes should only contain BTN labels.

The label format of the Button group Box is almost identical to the label format on the "general-purpose" Group Box except the "G" identifier is replaced by "H", as follows:

"~H*wwwhhtext*

where:

H	Group box modifier
www	Group box width
hh	Group box height
text	Group box title

Both the width and height parameters must be specified with **exactly** 2 digits (including a leading 0 for values less than 10). If no Label is specified then the group box has no title and an unbroken box is drawn. The group box specifying label should be placed on the line above the set of fields/labels to be grouped together and the height parameter set to the number of lines to be enclosed (excluding the top label line).

4.5 Label Indentifies (I)

By default all Speedbase labels are "anonymous". However, for some specialised sub-routines it is necessary to identify a particular label. The following special label format has been implemented to allow an index value to be associated with a label:

"~I*nnText*"

where:

I	Indicates this is an indexed label
nn	label index number (with a leading 0 if less than 10)

text The true label text

The subroutine, GXSLA$, is available to allow an attribute number and/or enabled/disabled flag to be specified for a particular label (identified by its index number). The attribute number must be in the range 1 to 64 and specifies one of the extended attribute colour combinations.

Version Requirements

```
GX V2.8f
```

4.6 Radio Buttons (L) and (R)

The GX interface supports up to 9 groups of radio buttons for a single Speedbase window. Radio buttons are supported using a set of normal labels together with an associated field that is used to return the currently selected button. Each Radio Button label contains special text to indicate that it should be displayed as part of a radio button.

The limit of 9 groups of radio buttons per Speedbase window has been extended to 35 groups of radio buttons per Speedbase window for GX V3.2v, and later.

The special label text takes one of two forms. Either:

"~R*ns*Text"

or:

"~L*ns*Text"

where:

R	Display text to the right of the button
L	Display text to the left of the button
n	Group number (1 to 9 for GX pre-V3.2v; 1 to 9 or A to Z for GX V3.2v, and later)
s	Selection character (must be unique in the group)

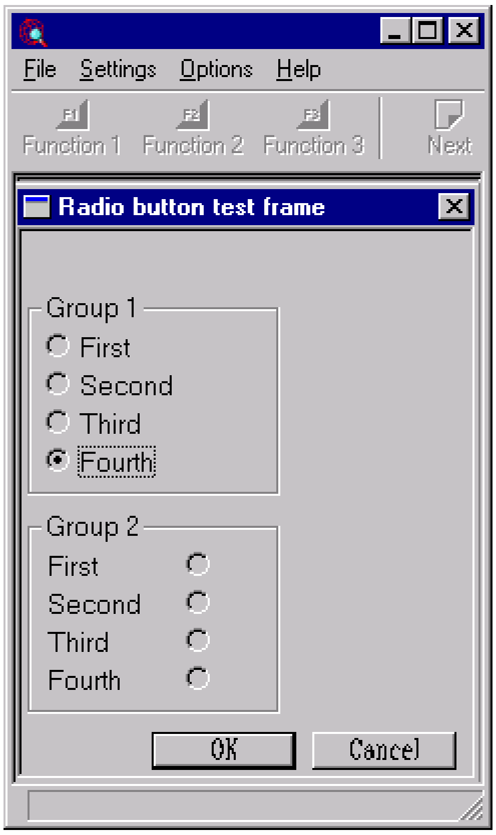
Up to 9 groups of radio buttons can be specified for GX pre-V3.2v. Up to 35 groups of radio buttons can be specified for GX pre-V3.2v. Each Radio Button group acts independently of all the others. The selection character of the currently selected button is placed in the associated field and this value is returned when an accept takes place on another field on the window (i.e. clicking on a radio button does not send back the value immediately).

The single character field is associated with the group of buttons by placing it at the start location of one of the labels in the group. The field is not displayed on screen and the overlaid label is registered as the default. Note that the default button can also be specified from within the Speedbase application by initialising the field to one of the selection characters in the group.

4.6.1 Example

```
PROGRAM RADIO
DATA DIVISION
01      MESSAGE
```

```
  02    FILLER  PIC X(?)
        VALUE   "1st response = "
  02    GR1     PIC X
  02    FILLER  PIC X(?)
        VALUE   " 2nd response = "
  02    GR2     PIC X
WINDOW W1
*
BASE AT 5 16
*
BOX 04 02 14 05 "Group 1"
BOX 10 02 14 05 "Group 2"
*
01 02 "Radio button test frame"  A12
*
05 02 "~R1FFirst "      NSC
06 02 "~R1SSecond"      NSC
07 02 "~R1TThird "      NSC
08 02 "~R1OFourth"      NSC
08 02 GR1       X       NSC
*
11 02 "~L2FFirst "      NSC
12 02 "~L2SSecond"      NSC
13 02 "~L2TThird "      NSC
14 02 "~L2OFourth"      NSC
12 02 GR2       X       NSC
*
ROUTINES SECTION
*
ENDWINDOW
*
PROCEDURE DIVISION
*
    ENTER WINDOW W1
    ON EXCEPTION
    END
    DISPLAY MESSAGE
    SUSPEND 10
    EXIT
*
ENDPROG
ENDSOURCE
```



4.7 Character Accept Timeout (M)

To define a time out for a field that times out if a new character has not been keyed, you need to set a text entry for the field at the same line column number of the field as follows:

Ln col "~M*nnnn*" where *nnnn* is the time out in milliseconds.

4.8 Progress Windows (P)

Although the GXBAR$ and GXBRU$ routines allow a single progress bar to be displayed within a special Progress Bar Window they do not allow a progress bar to be displayed within a standard application window. This note describes a technique that allows one, **or more**, progress bars to be displayed on a standard application window.

Furthermore, the format of the Progress Bar Window displayed by the combination of the GXBAR$ and GXBRU$ sub-routines is very rigid (e.g. only a small, fixed number of text lines can be displayed above and below the actual bar). The technique described in this document allows complete control over the fields and labels displayed around the "move-able" progress bar.

A Progress Bar is added to a standard application window by coding a special label of the form:

"~P*ndddxxxxx*"

where:

P	Progress Bar control specifier
n	Index value in the range 1 to 9
ddd	Depth (in pixels of the control)
xxxxx	Filler text to set the width of the control

Important Note: The depth specified in pixels thus the progress bar control could be displayed over multiple lines in the application window. Care must be taken to ensure there is sufficient space to allow for the progress bar control.

The range of values (minimum and maximum) that the progress control displays is specified using the GXPRG$ routine If the GXPRG$ routine is not used the control automatically defaults the minimum and maximum values to 0 and 100, respectively.

The position of the progress bar is updated by updating a PIC 9(9) COMP data field that is defined **at the same position in the window** as the label containing the "~P*ndddxxxxx*" text. For example:

```
      03 03 "Units"                       * Description of 1st progress bar
      03 15 "~P10257890"                  * Label Specifying 1st progress bar
      03 15 Z-D1      9(9) C  NSC         * Data field to update 1st progress bar
      *
      05 03 "Tens"                        * Description of 2nd progress bar
      05 15 "~P20257890"                  * Label Specifying 2nd progress bar
      05 15 Z-D2      9(9) C  NSC         * Data field to update 2nd progress bar
      *
      07 03 "Hundreds"                    * Description of 3rd progress bar
      07 15 "~P30257890"                  * Label Specifying 3rd progress bar
      07 15 Z-D3      9(9) C  NSC         * Data field to update 3rd progress bar
```

If the value in the associated data field (e.g. Z-D1) when it is displayed (typically by using the SHOW verb) is below the minimum value specified using the GXPRG$ routine, the minimum value will be used. Similarly, if the value in the associated data field (e.g. Z-D1) when it is displayed (typically by using the SHOW verb) is above the maximum value specified using the GXPRG$ routine, the maximum value will be used.

4.10.1 Example

```
FRAME PROGRS
DATA DIVISION
*
77 Z-C1      PIC 9(9) COMP
77 Z-C2      PIC 9(9) COMP
77 Z-C3      PIC 9(9) COMP
77 Z-COUNT   PIC 9(9) COMP
*
77 Z-WHDL    PIC 9(4) COMP
77 Z-INDX    PIC 9(4) COMP
77 Z-PRM1    PIC 9(9) COMP
77 Z-PRM2    PIC 9(9) COMP
*
77 Z-D1      PIC 9(9) COMP
77 Z-D2      PIC 9(9) COMP
77 Z-D3      PIC 9(9) COMP
*
01     P1
  02   P1VER PIC 9(4) COMP
       VALUE 1
  02   P1ID  PIC 9(2) COMP
  02   P1MIN PIC 9(9) COMP
  02   P1MAX PIC 9(9) COMP
*
WINDOW W1
```

```
*
BASE AT 5 5
*
01 02 "Progress control display window" A12
*
03 03 "Units"
03 15 Z-D1  9(9) C      NSC
03 15 "~P10257890"
05 03 "Tens"
05 15 Z-D2  9(9) C      NSC
05 15 "~P20257890"
07 03 "Hundreds"
07 15 Z-D3  9(9) C      NSC
07 15 "~P30257890"
*02 02 W1NULL     X(0)        NUL
*
09 20 "  Break    "           BTN U-7
*
ENDWINDOW
*
WINDOW W2
*
BASE AT 10 10
*
02 02 "Progress control window"     A12
*
04 02 W2TEMP      X(4)              NSC NUL
*
06 06 "  Start    " BTN U50
*
ROUTINES SECTION
*
R-FUNC.
    IF $FUNC < 50 EXIT
    IF $FUNC > 99 EXIT

    IF $FUNC = 50
       PERFORM CE-START-TEST
       ON EXCEPTION
        CLEAR WINDOW W1
       END
    END

    EXIT

PROCEDURE DIVISION
SECTION AA-MAIN.
*
    ENTER WINDOW W2
    IGNORE EXCEPTION
    EXIT
*
SECTION CA-SET-RANGE
*
      MOVE Z-INDX TO P1ID
      MOVE Z-PRM1 TO P1MIN
      MOVE Z-PRM2 TO P1MAX
      CALL GXPRG$ USING P1          * ASSUME CURRENT WINDOW
      EXIT
*
SECTION CE-START-TEST
*
    DISPLAY WINDOW W1
    IGNORE EXCEPTION

    CALL B$GX-7

    MOVE 0 TO Z-WHDL
    MOVE 1 TO Z-INDX
    MOVE 0 TO Z-PRM1
    MOVE 10 TO Z-PRM2
    PERFORM CA-SET-RANGE
    MOVE 2 TO Z-INDX
    PERFORM CA-SET-RANGE
    MOVE 3 TO Z-INDX
    PERFORM CA-SET-RANGE

    MOVE 0 TO Z-C1
    MOVE Z-C1 TO Z-D1

    MOVE 0 TO Z-C2
    MOVE Z-C2 TO Z-D2

    MOVE 0 TO Z-C3
    MOVE Z-C3 TO Z-D3

    DISPLAY WINDOW W1

    DO FOR Z-COUNT = 1 TO 999
       CALL TEST$
       ON EXCEPTION EXIT WITH 1
       ADD 1 TO Z-C1
       IF Z-C1 = 10
        MOVE 0 TO Z-C1
        ADD 1 TO Z-C2
        IF Z-C2 = 10
           MOVE 0 TO Z-C2
           ADD 1 TO Z-C3
           MOVE Z-C3 TO Z-D3
           SHOW WINDOW W1 FIELD Z-D3
        END
        MOVE Z-C2 TO Z-D2
        SHOW WINDOW W1 FIELD Z-D2
       END
       MOVE Z-C1 TO Z-D1
       SHOW WINDOW W1 FIELD Z-D1
    ENDDO
    CLEAR WINDOW W1
    EXIT
*
ENDFRAME
ENDSOURCE
```

4.9 Window Timeout (T)

The following special label text allows a timeout to be specified on a window:

"~T*n*"

where:

T	Timeout label Identifier
n	An integer between 1 and 9999 which defines the time-out in seconds

Note that the timeout will only be activated on a zero length accept.

4.10 Window Width Alteration (W)

It is sometimes necessary to be able to modify the width of the window generated by GX under application program control. The GX Window Width alteration can be achieved by defining a label containing the following text within the window:

"~W+*nn*"
"~W-*nn*"

where:

W	Window Width label identifier
+	Increase Window Width by *nn* character positions
-	Decrease Window Width by *nn* character positions
nn	Window Width alteration factor in character positions

5. See Also

WINDOW Statement
WINDOW Options
ROUTINES SECTION

ACMBO$	Add to combo box
CCMBO$	Clear combo box
DCMBO$	Delete combo box
BTICN$	Iconic button bit map
GXBAR$	Progress bar
GXBRU$	Update Progress Bar on GX
GXPRG$	Specify GX Progress Bar Range
GXSLA$	Set the attributes of a "Dynamic Label"
WIBMP$	Display Bitmap in Window
WIDE$	Wide mode on text screens
SHOW verb	Show window field
HIDE verb	Hide window field
in366	Speedbase-calendar-pop-up