GX Programming Notes

Master Class - 7th December 2000

Table of Contents

Demo of GX itself

The following examples and coding techniques are all features of the released 32-bit speedbase language and the GX interface. I have provided example programs of all the techniques to be discussed and where possible I will also demonstrate how we have used these features in the Global 3000 packages.

Brief run through of a typical speedbase window running on GX

Re-size, change fonts, remembers positions, about boxes, on-line help and how it links to application, click on fields to move to that field.

Windows can be wider than 80 cols, deeper than 24 lines.

To Come.... Moveable fields, Sizeable scroll areas, Customisation........

GX Features

Email address recognition features (CO routine).
Web address recognition features (CO routine).

The GX interface will recognise an email or web address embedded in a piece of data, when displayed in a data field. The data is displayed in blue (though due to screen sizing no underline is drawn) and clicking on the data item will invoke the local email or browser software.

Show the V5.1 email and web address fields feature on the contact records.

TYPE Verb

This new window statement allows us to define a window as belonging to one of several classes. Each class has it own set of ini file options that control all aspects of the window display attributes.

In our example program a null response to a field displays an information window. An invalid response (in this case an account number of "FRED") is used to trigger an error window.

```
FRAME TYPE "TYPE Window Statement"
*************************************************************************
*                                                                       *
*                      TYPE - WINDOW STATEMENT                          *
*                      =======================                          *
* This window statement can be used to define common display attributes *
* for like windows groiuped by window usage. For example you may want   *
* all windows that display help text to be displayed in a different     *
* colour to make the user visually aware that he is looking at a help   *
* window.                                                               *
*                                                                       *
*************************************************************************

DATA DIVISION

WINDOW DIVISION

WINDOW W0
TYPE NORMAL
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  2  6
02 02 "[ Select Account                      ]"
03 04 "Account to be reported on"
                03 32 W0ACC        X(7)  NUL CNV
ROUTINES SECTION
V-W0ACC.
        IF W0ACC = ""
           ENTER WINDOW W1
           IGNORE EXCEPTION
           EXIT WITH 1
        END

        IF W0ACC = "FRED"
           ENTER WINDOW W2
           IGNORE EXCEPTION
           EXIT WITH 1
        END
EXIT

ENDWINDOW

WINDOW W1
TYPE INFO
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  4 12
02 02 "[ Please select account        ]"
03 04 "Accounts with a zero balance"
```

```
04 04 "will not be included on this"
05 04 "report."
   06 05 W1NUL  X(0)    NUL
ENDWINDOW

WINDOW W2
TYPE ERROR
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  9 25
02 02 "[ Error                        ]"
03 04 "Accounts with a zero balance"
04 04 "cannot be selected for this"
05 04 "report."
06 05 W2NUL       X(0)         NUL
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W0
        CALL TITLES USING W1
        CALL TITLES USING W2

        DISPLAY WINDOW W0
        ENTER WINDOW W0
        IGNORE EXCEPTION
EXIT

ENDFRAME
ENDSOURCE
```

Standard settings for "Normal" windows from GX.INI are mapped to the `Application' ini file settings.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                     ;
;         Application window colour settings                          ;
;                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ApplicationWindowBackground=192,192,192
ApplicationScrolledBackground=255,255,255
ApplicationLabelText=0,0,0
ApplicationFieldBackground=255,255,255
ApplicationFieldGreyedBackground=212,212,212
ApplicationFieldGreyedText=0,0,0
ApplicationFieldHighlightBackground=0,127,0
ApplicationFieldHighlightText=255,255,255
ApplicationFieldWarningBackground=127,127,0
ApplicationFieldWarningText=0,0,0
ApplicationFieldErrorBackground=255,0,0
ApplicationFieldErrorText=255,255,255
ApplicationFieldCurrentRecordBackground=0,0,255
ApplicationFieldCurrentRecordText=255,255,255
ApplicationFieldOtherRecordMusicBackground=196,253,200
ApplicationFieldAcceptBackground=252,254,188
ApplicationFieldAcceptText=0,0,0
```

For each other window type a set of ini file settings can be specified to override the above.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                     ;
;        Select (Search/Browse) window colour settings                ;
;                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

SelectFieldGreyedBackground=212,212,212
SelectFieldBackground=212,212,212
SelectFieldOtherRecordMusicBackground=196,253,200

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                     ;
;                    Info window colour settings                      ;
;                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

InfoWindowBackground=94,255,249
InfoFieldGreyedBackground=84,248,239
InfoFieldBackground=84,248,239
InfoFieldOtherRecordMusicBackground=196,253,200

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                     ;
;               Error window colour settings                          ;
;                                                                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

ErrorWindowBackground=250,204,194
ErrorFieldGreyedBackground=250,204,194
ErrorFieldOtherRecordMusicBackground=196,253,200
```

The complete set of valid window types and our suggested uses are NORMAL (the default if the type statement is omitted), HEADER for batch header or similar summary type windows, INFO for information and help windows, ERROR for error windows, ENQ for enquiry frames, SELECT for standard searches etc, TRAILER for trailer type windows e.g. "balance to allocate..." etc.

NSL field attribute (non-scrolled label)

Replaces use of the NS-TXT routine, correctly removes fields without leaving bits behind as per GUI 1. This routine also works on TAP 711.

In the example program the second prompt is set to either Customer or supplier dependent upon what is keyed into the first field.

```
FRAME NSL "NSL Field Attribute"
************************************************************************
*                                                                      *
*                         NSL FIELD ATTRIBUTE                          *
*                         ===================                          *
*  This field level attribute causes a data item to be displayed as if *
*  it was a non-scrolled text item.                                    *
*                                                                      *
************************************************************************

DATA DIVISION

77 Z-LABEL      PIC X(10)

WINDOW DIVISION

WINDOW W1
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  4 12
02 02 "[ NSL Attribute               ]"
03 03 "Debtors or Creditors"
                03 26 W1TYPE       X       CNV
                04 03 Z-LABEL      X(10)   NSL
                04 26 W1ACC        X(7)    NUL
ROUTINES SECTION
V-W1TYPE.
        IF  W1TYPE NOT = "D"
        AND W1TYPE NOT = "C"
            ERROR "Must be D or C only"
            EXIT WITH 1
        END
EXIT

B-Z-LABEL.
        IF W1TYPE = "D"
           MOVE "Customer" TO Z-LABEL
        ELSE
           MOVE "Supplier" TO Z-LABEL
        END
EXIT
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1
        ENTER WINDOW W1
        IGNORE EXCEPTION
EXIT

ENDFRAME
ENDSOURCE
```

OPT field attributes (optional field)

The OPT field attribute allows us much greater control over optional fields in windows than the NS-TXT technique it replaces. An EXIT WITH 1 from the B-routine of a field causes the input or display of that field to be skipped and the field removed from the window. Under GUI1 this was often leaving bits of the chiseled boxes behind, or `holes' in the window. 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. This technique is normally used in conjunction with an NSL field to allow the field prompt to also be suppressed.

The example program suppresses the second prompt dependent on the response to the first field. This can be used in conjunction with the NSL option to allow a field and it's label to either be displayed or removed accordingly.

Show how order entry differs between V4.5 and V5.0 as far as the optional fields go when entering a stocked or non-stocked product.

```
FRAME OPT "OPT Field Attribute"
*************************************************************************
*                                                                       *
*                         OPT FIELD ATTRIBUTE                           *
*                         ===================                           *
*  This field level attribute causes a data item to be suppressed if    *
*  an EXIT with 1 is performed in the B- of the field. This replaces    *
*  the technique in GUI1 where to suppress a field you would have to    *
*  blank out the field contents and then use CALL NS-TXT to turn it     *
*  into a label. This technique was not always reliable.                *
*                                                                       *
*************************************************************************

DATA DIVISION

77 Z-LABEL      PIC X(10)
77 Z-TEXT       PIC X(9)
                VALUE "Reference"

WINDOW DIVISION

WINDOW W1
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  4 12
02 02 "[ OPT Attribute               ]"
03 03 "Debtors or Creditors"
                03 26 W1TYPE      X            CNV
                05 03 Z-LABEL     X(10)        NSL
                04 03 Z-TEXT      X(9)         NSL OPT
                04 26 W1REF       X(8)         OPT NUL
                05 26 W1ACC       X(7)         NUL
ROUTINES SECTION
*
* Do not prompt for second field if "C" was chosen at first field.
*
B-Z-TEXT.
B-W1REF.
```

```
        IF W1TYPE = "C" EXIT WITH 1
EXIT

V-W1TYPE.
        IF  W1TYPE NOT = "D"
        AND W1TYPE NOT = "C"
            ERROR "Must be D or C only"
            EXIT WITH 1
        END
EXIT

B-Z-LABEL.
        IF W1TYPE = "D"
           MOVE "Customer" TO Z-LABEL
        ELSE
           MOVE "Supplier" TO Z-LABEL
        END
EXIT

ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1

        DISPLAY WINDOW W1
        ENTER WINDOW W1
        IGNORE EXCEPTION
EXIT
ENDFRAME
ENDSOURCE
```

SCL field attribute (scrolled label)

Replaces use of the SC-TXT routine, as per above.

Included in above demonstration. This attribute is intended for scrolled labels, i.e. the labels displayed above the scrolled columns.

HEADING Verb

Adds text items to the status line. Show examples from V5.0

Show example frame and also how this has been used in V5.0 for the Ledgers, GL and Cash Manager.

Please note that all items displayed using the heading command must either be fixed text or PIC X type data items. Any numeric information must first be moved to a text data item.

```
FRAME HEADIN "HEADING command"
*************************************************************************
*                                                                       *
*                         HEADING Command                               *
*                         ===============                               *
*  This new procedural statement allows a speedbase program to update   *
*  the window title bar with items of data.                             *
*                                                                       *
*************************************************************************

DATA DIVISION

77 Z-YEAR       PIC X(4)
                VALUE "2000"
77 Z-PERD       PIC X(2)
                VALUE "12"

WINDOW W0
TYPE NORMAL
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  2  6
02 02 "[ Select Account                      ]"
03 04 "Account to be reported on"
                03 32 W0ACC        X(7)  NUL CNV
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W0

        DISPLAY WINDOW W0
        ENTER WINDOW W0
        IGNORE EXCEPTION
EXIT

LOAD DIVISION
        HEADING ""
        HEADING "Current Period: "
        HEADING Z-YEAR
        HEADING "/"
        HEADING Z-PERD
EXIT

ENDFRAME
ENDSOURCE
```

YNC boxes (Check boxes)

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.

Show example from CL Aged Transaction report for V5.1 as well as the following example.

```
FRAME YNC "YNC Field Attribute"
*************************************************************************
*                                                                       *
*                YNC FIELD ATTRIBUTE - YES/NO CHECKBOX                  *
*                =====================================                  *
*  This field level attribute can be used instead of the normal YN      *
*  field attribute. It replaces the one character input field with      *
*  a standard window tickbox. A tick represens "Y" and an empty box     *
*  represent "N".                                                       *
*                                                                       *
*************************************************************************

WINDOW DIVISION

WINDOW W1
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  4 12
02 02 "[ Report Options               ]"
03 04 "Include future dated?"
       03 30 W1OPT1     X       YNC
04 04 "Include zero balances?"
       04 30 W1OPT2     X       YNC
05 04 "Include over limit?"
       05 30 W1OPT3     X       YNC
06 04 "Include foreign currency?"
       06 30 W1OPT4     X       YNC
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1

        MOVE "Y" TO W1OPT1
        MOVE "Y" TO W1OPT2
        MOVE "Y" TO W1OPT3
        MOVE "Y" TO W1OPT4

        DISPLAY WINDOW W1
        ENTER WINDOW W1
        IGNORE EXCEPTION
EXIT
ENDFRAME
ENDSOURCE
```

PAS field attribute

Allows password style fields to be entered displaying an "*" for each character entered.

Show example from the web frame software

```
FRAME PAS "PAS Field Attribute"
*************************************************************************
*                                                                       *
*                PAS FIELD ATTRIBUTE - PASSWORD                         *
*                ==============================                         *
*  This field level attribute causes a data item to be accepted with an *
*  * displayed on screen for each character typed.                      *
*                                                                       *
*************************************************************************

DATA DIVISION

WINDOW DIVISION

WINDOW W1
EDT
SBOX
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  4 12
02 02 "[ PAS Field Attribute          ]"
03 06 "Logon-id"
                03 18 W1TYPE      X(10)        CNV
04 06 "Password"
                04 18 W1ACC       X(10)        PAS
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1
        ENTER WINDOW W1
        IGNORE EXCEPTION
EXIT

ENDFRAME
ENDSOURCE
```

RJF field attribute

In the following example the first window shows how a proportional spaced label does not correctly align to the end of the column of data it represents. The second window shows the same window but with the RJF attribute in place.

```
FRAME RJF "RJF Field Attribute"
*************************************************************************
*                                                                       *
*                          RJF FIELD ATTRIBUTE                          *
*                          ===================                          *
*  This field level attribute causes a data item to be displayed right  *
*  aligned rather than left aligned.                                    *
*                                                                       *
*************************************************************************

ACCESS STOCK:SP

WINDOW DIVISION

WINDOW W1 USING SP
ENQ
AUTOPGE
SEQUENCE EXIT, EXIT
BASE AT  2  2
SCROLL  8 BY  1 SPLIT  1 OFFSET  1
02 02 "Product Code"
                03 02 SPPROD       X(15)
02 18 "Description"
                03 18 SPDESC       X(30)
02 49 "Standard Cost"
                03 49 SPNEWC      S9(9,2) C
ENDWINDOW

WINDOW W2 USING SP
ENQ
AUTOPGE
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  2  2
SCROLL  8 BY  1 SPLIT  1 OFFSET  1
02 02 "Product Code"
                03 02 SPPROD       X(15)
02 18 "Description"
                03 18 SPDESC       X(30)
02 49 "Standard Cost" RJF
                03 49 SPNEWC      S9(9,2) C
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1
        ENTER WINDOW W1
        IGNORE EXCEPTION

        CALL TITLES USING W2
        ENTER WINDOW W2
        IGNORE EXCEPTION
EXIT

ENDFRAME
ENDSOURCE
```

CON Field attribute

BERR, BWARN and B$HILI routines

Show example from enquiries in V5.0 (Credit limit field)
Show how they are configured via the ini file settings

```
FRAME ERR "Field Colours"

WINDOW DIVISION

WINDOW W1
EDT
POP-UP
SBOX
SEQUENCE EXIT, EXIT
BASE AT  2  4
02 02 "[ Warning Attribute   ]"
03 03 "Warning"
                03 11 W1TEXT      X(13)         CNV DIS
04 03 "Prompt"
                04 11 W1PRMT      X(13)         CNV NUL
ROUTINES SECTION
B-W1TEXT.
        MOVE "** WARNING **" TO W1TEXT
        CALL B$WARN
EXIT

ENDWINDOW

WINDOW W2
EDT
POP-UP
SBOX
SEQUENCE EXIT, EXIT
BASE AT 09  04
02 02 "[ Hilight Attribute   ]"
03 03 "Hilight"
                03 11 W2TEXT      X(13)         CNV DIS
04 03 "Prompt"
                04 11 W2PRMT      X(13)         CNV NUL
ROUTINES SECTION
B-W2TEXT.
        MOVE "** HILIGHT **" TO W2TEXT
        CALL B$HILI
EXIT

ENDWINDOW

WINDOW W3
EDT
POP-UP
SBOX
SEQUENCE EXIT, EXIT
BASE AT  16 4
02 02 "[ Error Attribute     ]"
03 03 "Error"
                03 11 W3TEXT      X(12)         CNV DIS
04 03 "Prompt"
                04 11 W3PRMT      X(12)         CNV NUL
ROUTINES SECTION
B-W3TEXT.
        MOVE "** DANGER **" TO W3TEXT
        CALL B$ERR
EXIT
```

```
ENDWINDOW

PROCEDURE DIVISION

        CALL TITLES USING W1
        CALL TITLES USING W2
        CALL TITLES USING W3

        DISPLAY WINDOW W1
        ENTER WINDOW W1
        IGNORE EXCEPTION
        *
        DISPLAY WINDOW W2
        ENTER WINDOW W2
        IGNORE EXCEPTION
        *
        DISPLAY WINDOW W3
        ENTER WINDOW W3
        IGNORE EXCEPTION
EXIT

ENDFRAME
ENDSOURCE
```

Baseline useage

1. Convert all baseline accepts into windows.
2. Remove all DISPLAY..AT and ACCEPT..AT.
3. Do no mix display and error messages to the baseline.
4. Use compiler option GUI to identify all problem areas.

Buttons

BTN attribute
Unn attribute

New way of thinking when designing software for GX only
UF1 - drop down arrow on search fields

Show three examples:

Currently buttons are only available on selection windows.

A button is defined by the presence of the BTN attribute and the Unn attribute where nn is a number between 50 and 99 which is the value returned in $FUNC if the button has been pressed.

There is currently a known bug in that if $FUNC is left set to a value greater than 50 the next window processed fails to recognise any valid keystrokes. This is why the sample button programs set $FUNC to 0.

```
FRAME BUTT01 "Buttons"
*************************************************************************
*                                                                       *
*                          Buttons                                      *
*                          =======                                      *
*                                                                       *
*************************************************************************

ACCESS STOCK:SP

DATA DIVISION

WINDOW DIVISION

WINDOW W1 USING SP
SEL
REPEAT UNTIL CURRENT RECORD
SEQUENCE EXIT, EXIT
BASE AT  2  2
SCROLL  8 BY  1 SPLIT  1 OFFSET  1
02 02 "Product Code"
                03 02 SPPROD      X(15)
02 18 "Description"
                03 18 SPDESC      X(30)
02 49 "Lookup"
                03 49 SPLOOK      X(13)
12 02 " Button 1"                             BTN U51
12 18 " Button 2"                             BTN U52
12 34 " Button 3"                             BTN U53
ENDWINDOW

PROCEDURE DIVISION
        CALL TITLES USING W1
        DO
           ENTER WINDOW W1
           ON EXCEPTION FINISH
           IF $FUNC = 51
              ERROR "Button 1 pressed on product "
              ERROR SPPROD SAMELINE
              ERROR ""
           END
           IF $FUNC = 52
              ERROR "Button 2 pressed on product "
              ERROR SPPROD SAMELINE
              ERROR ""
           END
           IF $FUNC = 53
              ERROR "Button 3 pressed on product "
              ERROR SPPROD SAMELINE
              ERROR ""
           END
           MOVE 0 TO $FUNC
        ENDDO
EXIT

ENDFRAME
ENDSOURCE
```

The DL425 program that has a dialogue window to trigger searches etc and shows how multiple buttons can be used. As this frame can only ever operate with the GX interface it also uses a window that is wider than the normal 80 columns and deeper than 24 lines.

The updated demonstration software shows how a typical windows based application can be emulated in full now in speedbase.

Browser integration

See separate document for an example program and details on routines to be used.

Configure ini file
Creating the temp file using the new DLM routines
Invoking the browser

Send email from application- EMAIL$

```
FRAME EMAIL

DATA DIVISION

01 ST
   02 STSUB       OCCURS 10       * Up to 10 recipients
      03 STRLEN   PIC 9(4) COMP   * Length of each recipient
      03 STRPTR   PIC PTR         * Pointer to recipient text
   02 STSLEN      PIC 9(4) COMP   * Length of subject text
   02 STSPTR      PIC PTR         * Pointer to subject text
   02 STTLEN      PIC 9(4) COMP   * Length of text block
   02 STTPTR      PIC PTR         * Pointer to text block
   02 STATT       OCCURS 20       * List of up to 20 attachments
      03 STALEN   PIC 9(4) COMP   * Length of attachment
      03 STAPTR   PIC PTR         * Pointer to attachment pathname
   02 STCCS       OCCURS 10       * List of up to 10 CC's
      03 STCLEN   PIC 9(4) COMP   * Length of CC recipient
      03 STCPTR   PIC PTR         * Pointer to CC recipient text
   02 STBCCS      OCCURS 10       * List of up to 10 BCC's
      03 STBLEN   PIC 9(4) COMP   * Length of BCC recipient
      03 STBPTR   PIC PTR         * Pointer to BCC recipient text

77 Z-RECIPIENT    PIC X(17)
                  VALUE "djf@tissoft.co.uk"
77 Z-SUBJECT      PIC X(30)
                  VALUE "Some subject"
77 Z-TEXT         PIC X(50)
                  VALUE "Some text to be emailed....."

PROCEDURE DIVISION

        MOVE  17         TO STRLEN(1)
        POINT STRPTR (1) AT Z-RECIPIENT

        MOVE  30     TO STSLEN
        POINT STSPTR AT Z-SUBJECT

        MOVE  50     TO STTLEN
        POINT STTPTR AT Z-TEXT

        CALL EMAIL$ USING ST 0
EXIT

ENDFRAME
ENDSOURCE
```

The second parameter passed to the email$ routine is a 'mode'. 0 means display the email software dialogue window and allow amendment of the message details before manually sending. A mode of 1 means send the message with no further user input. If any mandatory fields have been left blank, the email dialogue window will appear and you will have to finish and send the email by hand, just as if using mode 0.

View image file - IMAGE$

Can be used as a simple picture viewer until GX supports images directly.

```
FRAME IMAGE "IMAGE$ Routine"

DATA DIVISION

77 Z-IMAGE      PIC X(13)
                VALUE "C:\acmods.GIF"

01 ST
   03 STLEN     PIC 9(4) COMP
   03 STURL     PIC PTR


WINDOW DIVISION

WINDOW WW
EDT
SEQUENCE EXIT CLW, EXIT CLW
POP-UP
SBOX
BASE AT  5  2
02 02 "[ IMAGE$ Subroutine            ]"
03 03 "Prompt"
                03 10 WWFLD        X    NUL UF1
03 13 "Press F1 for picture"
ROUTINES SECTION

V-WWFLD.
        IF $FUNC = 1
           POINT STURL AT Z-IMAGE
           MOVE 13 TO STLEN
           CALL IMAGE$ USING ST
        END
EXIT

ENDWINDOW


PROCEDURE DIVISION

        CALL TITLES USING WW

        DISPLAY WINDOW WW
        ENTER WINDOW WW
        IGNORE EXCEPTION
EXIT
ENDFRAME
ENDSOURCE
```

Text edit box - EDTTX$

```
FRAME EDTTX "EDTTX Windows text edit box"
*************************************************************************
*                                                                       *
*                   EDTTX - Windows text edit box                       *
*                   =============================                       *
*  This routine generates a standard windows-style text edit box for    *
*  keying in multi-line text data.                                      *
*                                                                       *
*************************************************************************

DATA DIVISION

01 TEXT                         * Text block to be updated
   03 FILLER    PIC X(240)
                VALUE "This is some default text to b"
                VALUE "e edited."

01 CB
   02 CBLEN     PIC 9(4) COMP   * Length of text area
   02 CBLIN     PIC 9(4) COMP   * Line to position window at
   02 CBCOL     PIC 9(4) COMP   * Column to position window at
   02 CBWID     PIC 9(4) COMP   * Initial width
   02 CBDEP     PIC 9(4) COMP   * Initial length
   03 CBTIT     PIC X(132)      * Caption text
   03 CBID      PIC X(4)        * Window id
   03 CBNAME    PIC X(8)        * Window name
   03 CBDISP    PIC 9    COMP   * 0 = Update, 1 = Display Only

PROCEDURE DIVISION

        MOVE 240               TO CBLEN
        MOVE   1               TO CBLIN
        MOVE   1               TO CBCOL
        MOVE  40               TO CBWID
        MOVE   5               TO CBDEP
        MOVE  "Customer Notes" TO CBTIT
        MOVE "W1__"            TO CBID
        MOVE "EDTTX___"        TO CBNAME

        MOVE   0  TO CBDISP

        CALL EDTTX$ USING CB TEXT
EXIT

ENDFRAME
ENDSOURCE
```

The line and column to position the window at are relative to the GX window. The window id and window name are used to tie the text window into the on-line help and also so the GX has a key by which it can remember the window position and size etc.

File browse dialogue box - FILBR$

```
FRAME FILBR "File Browse Demo"
*************************************************************************
*                                                                       *
*                          FILE BROWSE SUBROUTINE                       *
*                          ======================                       *
*                                                                       *
* This routine allows a speedbase program to call the windows file      *
* browse dialogue box.                                                  *
*                                                                       *
*************************************************************************

DATA DIVISION

77 FILE-FILTER  PIC X(17)
                VALUE "Text file (*.txt)"

01 FI-BLOCK
   02 FIFLG1    PIC 9(2) COMP   * 0 = "Open" file dialogue box
                                * 1 = "Save As" file dialogue box
   02 FIFLG2    PIC 9(2) COMP   * 0 = Return full path name
                                * 1 = Return file name
                                * 2 = Return directory name
                                * 3 = Return file name, <CR>, directory
   02 FILENG    PIC 9(4) COMP   * Length of file filter text
                VALUE 17

WINDOW W1
EDT
SEQUENCE EXIT CLW, EXIT CLW
BASE AT  7 14
02 02 "[ File browse                                  ]"
04 03 "File"
                04 08 W1PATH      X(40)        UF1 NUL
                05 09 W1NULL      X(0)         NUL
ROUTINES SECTION

V-W1PATH.
        IF $FUNC = 1
           CALL FILBR$ USING FI-BLOCK FILE-FILTER
           ON EXCEPTION EXIT WITH 1
           CALL ECHO
        END
EXIT
ENDWINDOW

LOAD DIVISION
        CALL TITLES USING W1
EXIT

ENDFRAME
ENDSOURCE
```