Msg_Equates $Insert Record

```
compile insert Msg_Equates
* used by Msg()
********************************************************************************
*
* Product    :  OpenInsight
* Version    :  3.5
*
* History    :  (date, initials, notes)
*   12/15/95  apk  Original programmer.
*   02/26/96  cp   Changed OK return value to "" from 0 (backwards compat.)
*   03/28/96  cp   Added MHELP$, MREQRESP$, and MBEEP$ fields
*                  Commented message structure and instructions
*   06/24/97  cp   Added G (gauge) type, removed unused equates
********************************************************************************
declare subroutine Msg          ;* Msg(Parent, MsgDef [, MsgKey, Instruction, Params])
declare function   Msg          ;* Ans = Msg(Parent, MsgDef [, MsgKey, Instruction, Params])
* message structure field definitions
equ MTEXT$          to 1        ;* the text to display in the message, multiple lines delimited by @tm, cr/lf,
or "|"
equ MTYPE$          to 2        ;* the message type, defaults to "BO" (see below)
equ MMODAL$         to 3        ;* modality of message, defaults to "A" (see below)
equ MICON$          to 4        ;* icon to display (see below)
equ MDEFBTN$        to 5        ;* default button (1 for first, 2 for second, etc.)
equ MCOL$           to 6        ;* message h-pos in pixels, or -2 (center screen, the default), -1 (center
parent)
equ MROW$           to 7        ;* message v-pos in pixels
equ MJUST$          to 8        ;* justification:  T (text, the default), L (left), R (right), C (center)
equ MBKCOLOR$       to 9        ;* background color (RGB value, @vm-delimited), see Utility("CHOOSECOLOR")
equ MFGCOLOR$       to 10       ;* foreground color (RGB value, @vm-delimited), see Utility("CHOOSECOLOR")
equ MTEXTWIDTH$     to 11       ;* the message width (or the response field width for response messages)
equ MCAPTION$       to 12       ;* the message title
equ MVALID$         to 13       ;* for response messages, this is the validation pattern (ie. (MD0) for integer)
equ MDEFINPUT$      to 14       ;* for reponse messages, this is the default response
equ MMASKINPUT$     to 15       ;* boolean, true for password (masked) input, false (default) for readable input
equ MBITMAP$        to 16       ;* name of a bitmap registered in repository (as it appears in the outliner)
equ MCLIPBMP$       to 17       ;* boolean, true to clip bitmaps, false to resize (see IMAGECLIP property)
equ MFONT$          to 18       ;* font structure for the text of the message
equ MLITERAL$       to 19       ;* boolean, defaults to false, true specifies that default value is a function
(see below)
equ MHELP$          to 20       ;* help button, Type:@vm:Specifier:@vm:Text (see below), defaults to null (no
help)
equ MREQRESP$       to 21       ;* boolean, for type "R" messages, false allows nulls (default) while true
doesn't
equ MBEEP$          to 22       ;* integer, specifies beep (see MessageBeep in the Windows API)
equ MEXTENT$        to 23       ;* integer, specifies extent of the gauge (number of items to process for type="
G")
equ MNUMFIELDS$     to 23
* MTYPE$ details:
*
* there are six base types, B (buttons), R (response), U (up), and D (down),
* T (timed), and G (gauge)
*
* the button type has several pre-defined button sets which are localized
* using entries from the SYSTEM_RESOURCES record in the SYSENV table; for
* non-standard labels, the buttons can be specified in a comma-delimited
* list, like "B&One,&Two,&Three" (where the & specifies the accelerator)
*
*   B Type  Description
*   ------  --------------------
*   BO      OK
*   BOC     OK/Cancel
*   BNY     Yes/No
*   BNYC    Yes/No/Cancel
*   BRC     Retry/Cancel
*   BAR     Abort/Retry
*   BARI    Abort/Retry/Ignore
*   B{list} User-defined buttons
```

```
*
* the response type displays an edit field and OK and Cancel buttons;
* the optional sub-types are C (upper-case only) and E (escape or
* cancel button returns escape character instead of default response);
* for example, the following types are valid:  "R", "RC", "RE", "RCE"
*
* for B and R types, the type can be preceded with an N to specify that
* the default value for the message is to be returned without the message
* being displayed; this is one way to change messages from interactive
* to non-interactive for batch processes; for example, instead of "BARI",
* pass "NBARI" (meaning don't display the abort/retry/ignore message)
*
* to display a message while processing, use the "U" type:
*
*   Def = ""
*   Def<MTEXT$> = "Processing..."
*   Def<MTYPE$> = "U"
*   MsgUp = Msg(@window, Def)   ;* display the processing message
*   ...
*   Msg(@window, MsgUp)         ;* take down the processing message
*
* to display a message for a specific length of time, use the "T" type:
*
*   Def = ""
*   Def<MTEXT$> = "Waiting..."
*   Def<MTYPE$> = "T2"          ;* 2-second message
*   Msg(@window, Def)
*
* the optional sub-type for T type is A (asynchronous), which displays
* the message and returns (allowing processing to continue) and takes
* the message down after the specified period of time; since this relies
* on a timer event, your event code must either complete within the
* specified period of time or you must regularly use Yield() to allow
* the processing of posted events (like the timer):
*
*   Def = ""
*   Def<MTYPE$> = "TA5"         ;* 5-second splash-screen
*   Msg(@window, Def, "SPLASHSCREEN")
*   loop
*     Done = AppLogonProcessing()
*     Yield()
*   until Done
*   repeat
*
* to display a gauge (percent bar), use the "G" type; sub-types are C (show
* cancel button) and Y (yield on each cycle):
*
*   Def = ""
*   Def<MCAPTION$> = "Processing Orders..."
*   Def<MTYPE$   > = "GC"
*   Def<MEXTENT$ > = OrderCnt
*   MsgUp = Msg(@window, Def)
*   for Order = 1 to OrderCnt
*     gosub ProcessOrder
*   * update the gauge and check if cancel was pressed
*   while Msg(@window, MsgUp, Order, MSGINSTUPDATE$)
*   next Order
*   Msg(@window, MsgUp)         ;* take down the gauge
* MMODAL$ detail:
*
*   Code         Modality     Description
*   -----------  -----------  ----------------------------
*   W            Window       only the parent is disabled
*   A (default)  Application  all OI windows are disabled
*   S            System       all applications are disabled
* MICON$ detail:
*
*   Code  Icon
*   ----  ---------------
*   null  None
*   *     Asterisk (Info)
```

```
*   ?     Question
*   !     Exclaim (Warning)
*   H     Halt (Stop sign)
*   B     User-specified bitmap (specified in MBITMAP$ field)
* MLITERAL$ details (applies only to response type messages):
*
* if MLITERAL$ is true, the Msg() function assumes that the default value is
* the name of a function which returns the default value for the message;
* for example, if you wrote a function called CURRENTUSER which returned
* the user name of the current user, you could specify CURRENTUSER as the
* default value (MDEFINPUT$) and set MLITERAL$ to true, so that the current
* user name would be the default value for the message; parameters are
* passed to the specified function depending on the number of parameters
* that are supported by the function:
*
*   # Params   Values Passed
*   ---------  --------------
*   0          None
*   1          MsgKey
*   2 or more  MsgKey, MsgDef
* MHELP$ details:
*
*   Type  Description          Specifier
*   ----  -------------        -----------------
*   Q     QuickHelp (AppNote)  Name of AppNote
*   M     Message              Name of Message
*   H     WinHelp              HelpFile,HelpID
*   S     Stored Procedure     ProcName[,Param1]
*
*   Note:  Specify the AppNote, Message, HelpFile, or ProcName as it appears
*          in the repository outliner.  For example, the OINSIGHT.HLP file is
*          registered as OINSIGHT, so specify the HelpFile as "OINSIGHT" (look
*          in the outline under "General", "Windows Components", "Help Files")
*
*   Text defaults to "&Help" or a localized equivalent
* Replaceable message parameters:
*
*   Msg(@window, "Hello, %1%, how are you %2%?", "", "", @username: @fm: "today")
* Msg() function instruction values
equ MSGINSTSTART$   to 1         ;* (default instruction)
equ MSGINSTREAD$    to 2         ;* bErr = Msg("", MsgDef, MsgID, MSGINSTREAD$)
equ MSGINSTWRITE$   to 3         ;* bErr = Msg("", MsgDef, MsgID, MSGINSTWRITE$)
equ MSGINSTLOCK$    to 4         ;* bErr = Msg("", "", MsgID, MSGINSTLOCK$)
equ MSGINSTUNLOCK$  to 5         ;* bErr = Msg("", "", MsgID, MSGINSTUNLOCK$)
equ MSGINSTBUTTON$  to 6         ;* reserved
equ MSGINSTCREATE$  to 7         ;* reserved
equ MSGINSTRESPCHG$ to 8         ;* reserved
equ MSGINSTCLOSE$   to 9         ;* reserved
equ MSGINSTTIMER$   to 10        ;* reserved
equ MSGINSTHELP$    to 11        ;* reserved
equ MSGINSTUPDATE$  to 12        ;* (see above)
* Msg() function return values from MSGINSTSTART$
equ RET_OK$         to ""
equ RET_CANCEL$     to \1B\    ;* escape character
equ RET_YES$        to 1
equ RET_NO$         to 0
equ RET_ABORT$      to 1
equ RET_RETRY$      to 2
equ RET_IGNORE$     to 3
* misc strings
equ MSGTYPE$        to "MSG"
equ MSGCLASS$       to ""
```

See Also

Msg()