


```

; Return the number of the button pushed (1,2,3...).
;
; 4-26-94 DSF Add keyword PARENT= to position the Ask widget in
; the middle of the top-level-base.
; 7-05-94 DSF Add keyword CHOICES to pass labels as an array. May also
; include arguments as well.
; 10-17-94 DSF First argument may be array of strings so message may be
; multiple lines.
; 11-29-94 DSF Support any number of choices. Up to ten as arguments, but
; unlimited number if you use CHOICES.
; 2-22-95 DSF Add keyword ROW and COLUMN to allow enforcement of format.
; 10-17-95 DSF Add keyword BEEP.
; 11-27-95 DSF Add keyword INPUT_FOCUS to specify button to have focus.
; 1-11-96 DSF Add keyword BASE_ALIGN_CENTER to main widget_base()
; call to center text and buttons.
; 3-11-96 DSF Add keyword WIDGET_ID to have the TLB widget
; returned, and when this is specified allow no labels.
; This is for popup info widgets that can be removed later
; by the calling program.
; 4-02-96 DSF Center on-screen (no PARENT) taking into account the
; size of the widget.
; 5-14-97 DSF Update for IDL 5.0. Use POS_WIDGET() to position widget.
; 1-12-99 DSF Add /LEFT_JUSTIFY to left-justify the message text in the widget.

```

```

;-----
; procedure ASK_EVENT
;
; Event handler for ask() function.
;-----

```

```

PRO ask_event, event

```

```

common ask_common, rtnval, buttons

```

```

for i = 0, n_elements(buttons) - 1 do begin
  if (event.id eq buttons(i)) then begin
    rtnval = i + 1
    widget_control, event.top, /DESTROY
  endif
endfor

```

```

END

```

```

;-----
; function ASK

```

```

;
; Routine to create widgets for prompt.
;-----
FUNCTION ask, text, label1,label2,label3,label4,label5,label6,label7, $
  label8,label9,label10,PARENT=parent, CHOICES=choices, $
  TITLE=title, ROW=row, COLUMN=column, BEEP=beep, $
  INPUT_FOCUS=input_focus, WIDGET_ID=widget_id, $
  LEFT_JUSTIFY=left_justify

```

```

common ask_common, rtnval, buttons

```

```

if (xregistered("ask")) then begin
  message, 'Only one instance may be running', /continue
  return, -1
endif

```

```

rtnval = 0
params = n_params()
n_labels = params + n_elements(choices) - 1
if ( n_labels lt 1 and not keyword_set( WIDGET_ID ) ) then begin
  message, 'At least 1 label required', /continue
  return, -1
endif

```

```

if ( n_labels ge 1 ) then begin
  labels = STRARR(n_labels)
  if (n_elements(label1)) then labels(0) = label1
  if (n_elements(label2)) then labels(1) = label2
  if (n_elements(label3)) then labels(2) = label3
  if (n_elements(label4)) then labels(3) = label4
  if (n_elements(label5)) then labels(4) = label5
  if (n_elements(label6)) then labels(5) = label6
  if (n_elements(label7)) then labels(6) = label7
  if (n_elements(label8)) then labels(7) = label8
  if (n_elements(label9)) then labels(8) = label9
  if (n_elements(label10)) then labels(9) = label10
endif

```

```

if (keyword_set(CHOICES)) then begin
  for i = params-1, (params-1) + n_elements(choices)-1 do begin
    labels(i) = choices(i - (params-1))
  endfor
endif

```

```

if (not keyword_set(TITLE)) then title = ' '
if (not keyword_set(PARENT)) then parent = 0L

```

```
; Left-justify text if specified
```

```
if (keyword_set(LEFT_JUSTIFY) and n_elements(text) ge 2) then begin  
  text2 = left_justify(text)  
endif else begin  
  text2 = text  
endelse
```

```
; Improve appearance of buttons
```

```
if ( n_labels ge 1 ) then begin  
  for i = 0, n_labels - 1 do begin  
    labels(i) = ' ' + labels(i) + ' ' ; To make buttons pretty  
    len = strlen(labels(i)) ; Labels will be at least 8 characters long  
    if (len lt 8) then begin  
      for j = 1, (8-len)/2 do labels(i) = ' ' + labels(i) + ' '  
      if (len mod 2 ne 0) then labels(i) = ' ' + labels(i)  
    endif  
  endfor  
endif
```

```
if (keyword_set(COLUMN)) then begin ; Set to COLUMN or ROW  
  set_column = 1 ; according to keywords  
  set_row = 0 ; (default to /ROW)  
endif else if (keyword_set(ROW)) then begin  
  set_column = 0  
  set_row = 1  
endif else begin  
  set_column = 0  
  set_row = 1  
endelse
```

```
if ( widget_info(parent, /valid_id) eq 1 and not keyword_set(WIDGET_ID) ) then $  
  begin  
    base = widget_base( /column, xpad=10, ypad=10, title=title, $  
      /base_align_center, group_leader=parent, /modal )  
  endif else begin  
    base = widget_base( /column, xpad=10, ypad=10, title=title, $  
      /base_align_center )  
  endelse
```

```
n_text = n_elements(text2)  
if (n_text gt 1) then junk = WIDGET_BASE(base, /column, /frame)
```

```
for i = 0, n_elements(text2) - 1 do begin  
  if (n_text eq 1) then begin  
    junk = widget_label(base, value=text2(i))
```

```
endif else begin
  junk2 = widget_label(junk, value=text2(i))
endelse
endifor
```

```
if (n_labels gt 4 and not keyword_set(ROW)) then begin ; Default to column for
  set_column = 1 ; > 4 items
  set_row = 0
endif
```

```
if ( n_labels ge 1 ) then begin
  junk = widget_base(base, row=set_row, column=set_column, /frame)
  buttons = lonarr(n_labels)
  for i = 0, n_labels - 1 do begin
    buttons(i) = widget_button(junk, value=labels(i))
  endifor
endif
```

```
if (keyword_set(BEEP)) then $
  print, format='($,a1)', string(7B)
```

; Account for size of widget and center accordingly, then realize

```
ret = pos_widget( base, parent=parent )
widget_control, base, /realize
```

```
if ( n_labels ge 1 ) then begin
  if (keyword_set(INPUT_FOCUS)) then begin
    if (input_focus gt n_elements(buttons)) then $
      input_focus = 1
    endif else begin
      input_focus = 1
    endelse
  widget_control, buttons(input_focus-1), /input_focus
endif
```

; Register this widget and start event processing, unless keyword
; WIDGET_ID is specified, in which case we just realize the widgets,
; return the TLB as the keyword, and return.

```
if ( keyword_set( WIDGET_ID ) ) then begin
  widget_id = base
endif else begin
  xmanager, "ask", base
endelse
```

```
return, rtnval
```

END

ASK

Use ASK to present the user with a question and from one to ten options for response, in a small widget. The function returns the number corresponding to the button pressed {1,2,3...}. Use Ask for warnings or when you require an answer before continuing. It is a MODAL function, in that it will be the only widget processing events until it returns (unless /WIDGET_ID is used).

Pass the text of the query and the labels for the buttons as arguments. You may optionally pass labels as an array, using the keyword CHOICES.

You may also use ASK to present a text message which may be removed by the calling program later using the widget id returned as keyword.

Calling Sequence

```
Answer = ASK(Message, answer1 [[,answer2], ..., answer10])
```

Arguments

Message

The message (or warning, etc) to be presented. This argument may be an array if more than one line is required for the message. Type: STRING or STRARR.

Answer1 ... Answer10

The labels for the buttons. These are the possible responses to the above Message. At least one is required unless using the keyword WIDGET_ID. Type: STRING.

Outputs

Returns the number of the button pressed as an answer, from 1 to 10. If there are not enough parameters (you must pass at least one Answer) returns -1.

Keywords

BEEP

Set this keyword to cause the console to beep.

CHOICES

Use this keyword to pass an array of labels. This can be used instead of or in combination with other labels as arguments. These labels will appear after any included as arguments. See the examples.

COLUMN

Set this to force the choices to be arranged in a column. Normally defaults to a row for four or less items, column otherwise.

INPUT_FOCUS

Use this keyword to designate which button will have input focus. The values begin with one, like the return value. When not set the first button gets input focus by default.

LEFT_JUSTIFY

Set this keyword to left-justify the message text in the widget. Otherwise, by default each line of text is centered within the text widget.

PARENT

This keyword is required under IDL 5.0 and later, to make ASK a modal widget.

Set to the widget ID of a top-level widget base, and the ASK widget will be positioned in the center of that widget.

ROW

Set this to force the choices to be arranged in a row. Normally defaults to a row for four or less items, column otherwise.

TITLE

The title of the ASK widget. Defaults to no title.

WIDGET_ID

Set this to a defined variable which will return the top-level-base widget id of the ASK widget. In this case, the widget is realized, but is not registered with XMANAGER.

This is intended for popup message widgets that will remain for some

period of time, and then will be deleted by the calling program, using the widget id returned.

Note that this is the only case when no MessageN parameters AND no CHOICES keyword may be specified.

Examples

```
answer = ASK('Exit Program?','Yes','No')
case (answer) of
  1: print, 'User said Yes'
  2: print, 'User said No'
endcase
```

```
answer = ASK('What is your favorite color?', $
  'Red','Blue','Magenta','Other')
```

```
answer = ASK('Operation complete', 'Ok')
```

```
options = ['Second', 'Third', 'Fourth']
answer = ASK('Buttons appear in the order:', $
  'First', CHOICES=options)
```

```
; Example of presenting an informational message and removing
; it later after a time-consuming operation.
```

```
ret = ASK('Reading image volume...', WIDGET_ID=id0, $
  TITLE='Please wait...')
widget_control, /hourglass
status = read_image_volume()
if (widget_info(id0, /valid_id) eq 1) then $
  widget_control, id0, /destroy
```

File Attachments

- 1) [ask.pro](#), downloaded 131 times
 - 2) [ask.doc](#), downloaded 134 times
-