

MANUAL DE FUNCIONES ALV

1.- CONCEPTOS GENERALES.	3
2.- CONCEPTOS PARA LA CREACION DE UN REPORT USANDO FUNCIONES ALV..	4
2.1.- LÓGICA EN UN REPORT ALV.-	4
3.- REALIZACIÓN DE UN REPORT ALV.-	5
3.1.- IT_FIELDCAT.....	5
3.2.- IT_EVENTS.	8
3.3.- <i>Eventos</i>	8
3.3.- I_SAVE.....	10
3.4.- I_CALLBACK_USER_COMMAND.	13
3.5.- IS_LAYOUT.	14
3.6.- IT_SORT.....	15
4.- EJEMPLO DE REUSE_ALV_LIST_DISPLAY Y REUSE_ALV_GRID_DISPLAY. ..	18
5.- EJEMPLO DE REUSE_ALV_HIERSEQ_LIST_DISPLAY.	26

1.- CONCEPTOS GENERALES.

Utilizando las funciones ALV para la generación de reports se obtienen muchas ventajas y utilidades que vienen ya implementadas por la propia función.

Hay 3 tipos de ALV cuyas funciones principales son:

- REUSE_ALV_GRID_DISPLAY
- REUSE_ALV_HIERSEQ_LIST_DISPLAY
- REUSE_ALV_LIST_DISPLAY

Para cualquiera de estas funciones que esté siendo utilizada existen una serie de funciones que deben ser utilizadas adicionalmente. Estas se irán viendo a lo largo del documento.

2.- CONCEPTOS PARA LA CREACION DE UN REPORT USANDO FUNCIONES ALV.

Cuando se crea un report utilizando funciones ALV es necesario:

1. *Incluir en el programa el tipo SLIS que tiene definidas todas las estructuras de los parámetros de entrada y de salida de esta función.*
2. *Definir correctamente el catálogo de campos.*
3. *Definir en el programa todas las subrutinas que implementarán el comportamiento de éste ante la ejecución de determinados eventos o ante la interacción del usuario.*
4. *Pasarle los datos que deben ser mostrados en el report a la función utilizando tablas internas.*

2.1.- Lógica en un Report ALV.-

Utilizando las funciones ALV, ninguna sentencia WRITE aparecerá en el programa. Se informa a la correspondiente función de los datos que van a ser mostrados, la definición y características de cada uno de éstos datos y de la apariencia y todo es implementado por la función. En el momento que llamamos a una de las funciones ALV para que imprima el report, perdemos el control del programa. Esto es, cualquier evento del programa como 'Nueva página', 'Top of page', 'End of page', etc. será controlado e implementado por la función, a no ser que le indiquemos a la función qué eventos queremos que sean implementadas en el programa.

3.- REALIZACIÓN DE UN REPORT ALV.-

Se llamará a la función, informándole obligatoriamente los siguientes parámetros:

- I_CALLBACK_PROGRAM:
Se informa a la función con el nombre del programa que está llamando a dicha función.
- IT_FIELDCAT:
Este parámetro se informará con una tabla interna que contiene la definición de los campos que aparecerán en el listado.
- T_OUTTAB:
Nombre de la tabla interna que contiene toda la información recogida por el programa y la cual debe ser impresa en el report.

Los demás parámetros podrán ser informados según el uso que se le vaya a dar a la función dentro del programa.

Ejemplo:

```
DATA: L_REPID TYPE SY-REPID.
      L_REPID = SY-REPID.

CALL FUNCTION 'REUSE_ALV_LIST_DISPLAY'
EXPORTING
  I_CALLBACK_PROGRAM           = L_REPID
  I_CALLBACK_PF_STATUS_SET    = 'STATUS_SET'
  I_CALLBACK_USER_COMMAND     = 'USER_COMMAND'
  IT_FIELDCAT                 = D_FIELDCAT
  IT_SORT                     = D_SORTCAT
  I_SAVE                      = 'A'
  IS_VARIANT                  = I_VARIAN
  IT_EVENTS                   = D_EVENTCAT

* IMPORTING
*   E_EXIT_CAUSED_BY_CALLER    =
*   ES_EXIT_CAUSED_BY_USER    =
TABLES
  T_OUTTAB                     = I_BSEG
EXCEPTIONS
  PROGRAM_ERROR                = 1
  OTHERS                       = 2.
```

3.1.- IT_FIELDCAT.

El parámetro IT_FIELDCAT se usa para diseñar el contenido de un report ALV. Se definirá una tabla interna de tipo SLIS_FIELDCAT_ALV, el cual está definido en el type-pool SLIS incluido al inicio del programa. Se construirá la tabla interna en el código del programa.

De todos los campos de esta tabla, las siguientes entradas son suficientes para cada registro:

- FIELDNAME:

Nombre del campo en la tabla interna que contendrá los datos a mostrar.

- REF_TABNAME:
Nombre de una tabla o estructura del diccionario de datos a la cual el campo a mostrar está referenciado. Solo se podrá usar si el campo actual está definido según el campo de una tabla del diccionario en SAP.

Descripción de los demás campos de esta tabla:

- ROW_POS:
Solo se informará este campo en el caso en el que el report de salida sea multilínea. Valores: 0, 1, 2 y 3.
- COL_POS:
Solo es relevante en el caso que se quiera que la salida en el listado tenga un orden distinto a su posición en la tabla interna de datos.
- TABNAME:
Nombre de la tabla interna que contiene los datos a mostrar en el listado.
- REF_FIELDNAME:
Nombre del campo de la tabla del diccionario de datos al cual está referenciado el campo de la tabla interna (en el caso en el que lo esté).
- CFIELDNAME:
Nombre del campo de la tabla interna de datos que contiene la moneda para el campo importe.
- CTABNAME:
Este dato sólo es relevante informarlo para listados con más de una cabecera.
- QFIELDNAME:
Nombre del campo de la tabla interna de datos que contiene la unidad de medida para el campo cantidad.
- QTABNAME:
Este dato solo es relevante informarlo para listados con más de una cabecera.
- OUTPUTLEN:
Siempre que el campo a mostrar en el listado de salida no esté referenciado a ningún campo de tabla del diccionario, se especificará aquí la longitud de salida del campo.
- KEY:
El campo del registro actual se tomará como clave si se marca este campo. No podrá ocultarse en el listado de salida este campo.
- KEY_SEL:
El campo será clave, pero podrá no salir en el listado de salida.
- NO_OUT:
El campo no será mostrado en el listado de salida.

- TECH:
Si se marca este campo, el campo actual no puede nunca salir en el listado de salida.
- EMPHASIZE:
Para mostrar el campo de algún color.
- HOTSPOT:
El campo aparece como hotspot.
- FIX_COLUMN:
Si se marca, este campo aparecería como columna fija.
- DO_SUM:
Se calculará la suma de este campo.
- NO_SUM:
No se calculará la suma de este campo.
- ICON:
El campo será mostrado como un icono.
- SYMBOL:
El campo será mostrado como un símbolo.
- JUST:
Justificación del campo. R= justificado a la derecha, L= justificado a la izquierda, C= centrado.
- LZERO:
Mostrará los campos de tipo NUMC justificados a la derecha y sin ceros a la izquierda.
- NO_SIGN:
Mostrará el campo sin signos.
- NO_ZERO:
Suprime los 0's.
- EDIT_MASK:
Mostrará el campo con una máscara.
- SELTEXT_L:
Se mostrará en la cabecera de la columna el texto largo de la definición del campo en la tabla del diccionario de datos.
- SELTEXT_M:
Se mostrará en la cabecera de la columna el texto medio de la definición del campo en la tabla del diccionario de datos.

- SELTEXT_S:
Se mostrará en la cabecera de la columna el texto corto de la definición del campo en la tabla del diccionario de datos.
- DDICTXT:
Especificará mediante L, M y S el texto largo, medio o corto que se utilizará en los títulos de las cabeceras.
- DATATYPE:
Se especificará cuando el campo de la tabla interna de datos no esté referenciado a ningún campo de tabla del diccionario de datos.
- DDIC_OUTPUTLEN:
Especifica la longitud de salida. Para campos no referenciados al diccionario de datos.
- SP_GROUP:
Si se marca este campo, el campo actual pertenecerá a un grupo que se especificará después en el parámetro IT_SPECIAL_GROUPS.
- REPREP:
El campo será tomado como un criterio de selección.

Nota: si queremos hacer la suma por alguno de los campos, este campo debe ser o bien tipo CURR o QUAN, debe existir otro campo en la tabla interna que contenga bien la moneda o bien la unidad de medida

3.2.- IT_EVENTS.

En esta tabla se especificarán ante qué eventos la respuesta será de nuestro programa y no implementado por la función. Para obtener los eventos que pueden ser codificados por el programa, es decir, aquellos para los cuales la función nos devuelve el control, basta con ejecutar la función REUSE_ALV_EVENTS_GET, importando una tabla interna en el parámetro ET_EVENTS, definida como SLIS_ALV_EVENT, estructura que viene definida en el type-pool SLIS incluido al inicio del programa.

Cuando se llame a la función que va a mostrar el report se le pasará esta tabla interna con el nombre de los eventos y el nombre de la subrutina donde irán implementados dichos eventos.

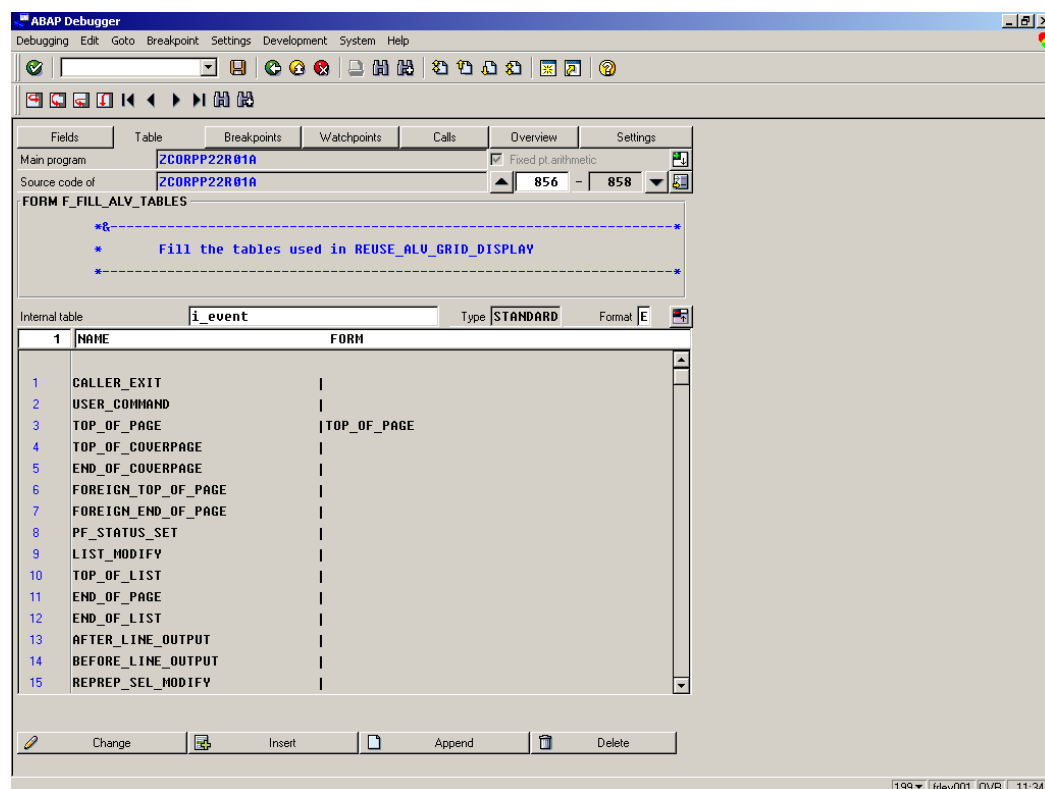
3.3.- Eventos.

De todos los eventos que podemos implementar en nuestro programa, los más utilizados o los más comunes son dos: 'Top-of-page' y 'At user command'.

- 'Top-of-page':

Este evento se ejecutará cada vez que se vaya a imprimir una nueva página en el listado. En el caso de utilizar una función GRID display, la cabecera sólo aparecerá en la primera página, puesto que en un GRID hay una sola página.

La forma de implementar el Top-of-page utilizando la función es sencilla. Definiremos una subrutina cuyo nombre se le pasará a la función en el contenido de la tabla interna I_EVENTS en el parámetro IT_EVENT.



En esta subrutina codificaremos el contenido de la cabecera del report. Existe una diferencia a la hora de implementar el top-of-page según la función que se esté utilizando:

- Si se trata de una función REUSE_ALV_LIST_DISPLAY o REUSE_ALV_HIERSEQ_LIST_DISPLAY se codificará con sentencias 'Write' de una forma normal.
- Si se trata de una función REUSE_ALV_GRID_DISPLAY se tendrá que utilizar una función específica para codificar el top-of-page. La función en particular es REUSE_ALV_COMMENTARY_WRITE. A dicha función se le pasará una tabla interna de tipo SLIS_LISTHEADER en el parámetro IT_LIST_COMMENTARY con toda la información que queremos que aparezca en la cabecera del listado.

```
*&-----
*
*&      Form  F_FILL_I_TITLE
*&-----
*      Build I_TITLE for header
*-----
FORM F_FILL_I_TITLE.

* Local variables and constants
```

```

DATA: L_DATUM(10),
      L_TIME(8).

CONSTANTS: C_ACTION LIKE WA_TITLE-TYP VALUE 'A',
           C_TITLE LIKE WA_TITLE-TYP VALUE 'H',
           C_HEADER LIKE WA_TITLE-TYP VALUE 'S'.

CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_TITLE.
WA_TITLE-KEY = SPACE.
WA_TITLE-INFO = TEXT-001.
APPEND WA_TITLE TO I_TITLE.
CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_HEADER.
WA_TITLE-KEY = TEXT-002.
WA_TITLE-INFO = L_DATUM.
APPEND WA_TITLE TO I_TITLE.
CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_HEADER.
WA_TITLE-KEY = TEXT-003.
WA_TITLE-INFO = L_TIME.
APPEND WA_TITLE TO I_TITLE.
CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_ACTION.
WA_TITLE-KEY = SPACE.
WA_TITLE-INFO = TEXT-004.
APPEND WA_TITLE TO I_TITLE.
CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_HEADER.
WA_TITLE-KEY = TEXT-005.
WA_TITLE-INFO = SY-REPID.
APPEND WA_TITLE TO I_TITLE.
CLEAR WA_TITLE.

* Title of report
WA_TITLE-TYP = C_ACTION.
WA_TITLE-KEY = SPACE.
WA_TITLE-INFO = SY-UNAME.
APPEND WA_TITLE TO I_TITLE.

ENDFORM.                " F_FILL_I_TITLE

```

- 'At user command':

Este evento se ejecutará una vez esté mostrado el report en pantalla, ante cualquier interacción del usuario.

3.3.- I_SAVE.

En muchas ocasiones, cuando se nos pide implementar un report utilizando funciones ALV, se nos pide mostrar en la pantalla de selección un parámetro que sea el Layout del Report. Esto es una utilidad de estas funciones. Una vez el listado esté en pantalla, el usuario puede definir su propio layout del report, mostrando y ocultando campos (columnas), calculando subtotales e incluso cambiando las columnas de sitio. Una vez adaptado el layout a sus necesidades, puede grabarlo, quedando así disponible para otras ejecuciones del programa.

El parámetro de la pantalla de selección nos servirá para elegir distintos layouts grabados en anteriores ejecuciones del programa, nos mostrará cuáles hay definidos y verificará la existencia de éstos. Para llevar a cabo estas funcionalidades, utilizaremos las siguientes funciones:

- REUSE_ALV_VARIANT_EXISTENCE pasándole en el parámetro CS_VARIANT el valor introducido por el usuario en la pantalla de selección. Nos dirá si el nombre del layout introducido por el usuario para ese report existe o no.
- REUSE_ALV_VARIANT_F4 hará las funciones del matchcode para ese campo. Hay que informar los siguientes parámetros:

- o IS_VARIANT:

Se le pasará una estructura definida de tipo DISVARIANT. Esta estructura deberá llevar informado el nombre del report que ejecuta la función. Se almacenará el valor devuelto por la función en el parámetro ES_VARIANT.

- o I_SAVE:

Este parámetro podrá tomar 3 valores distintos: 'X', seleccionará aquellos layouts grabados como standares, 'U', seleccionará aquellos layouts específicos de usuario, y 'A', seleccionará todo tipo de layouts definidos para el programa, tanto standares como específicos de usuario.

Cuando se utiliza esta utilidad de la función, hay que tener en cuenta que cuando se llame a la función para imprimir el report, hay que informar el parámetro I_SAVE con uno de los valores anteriormente definidos, ya que por defecto este parámetro toma como valor SPACE, que significa que no puede grabarse ningún layout en el report.

Definiremos a continuación un layout para un programa de ejemplo en el que se ha utilizado una función REUSE_ALV_GRID_DISPLAY. Este es el aspecto del report:

Production rework cost analysis

List Edit Goto Views Settings System Help

THEMIS DANONE

Date: 11/03/2002
Time: 17:36:06
Name: ZCORPP22R01A

Production rework cost analysis
MONTERRO

Plant	Production Li...	Order Type	Order Number	Material Group	Material Grou...	Material	Material Des...	Confirmed Q...	Scrapped Q...	Quantity Unit	Material Com...	Machine Costs
Z001	TITI LINE	Z001	100000833	1010100	WHEAT FLO...	34	YOGHURT x...	96.000,000	0,000	PC	0,00	5,00
Z001	TITI LINE	Z001	100000834	1010100	WHEAT FLO...	34	YOGHURT x...	189.999,984	0,000	PC	17.416,66	0,00
Z001	TITI LINE	Z001	100000835	1010100	WHEAT FLO...	34	YOGHURT x...	90.000,000	0,000	PC	207,00	0,00

199 | filev001 | DVR | 18:32

Eligiendo el botón 'Change layout' definiremos la apariencia que queremos que tenga el listado y lo grabará con el nombre '/Prueba' en la que solo se mostrarán 2 campos:

Change layout

Columns Sort order Filter View Display

Columns

Column Name	Material	Material Description
Material		
Material Description		

Column set

File Name

Save layout

Save as... Save with

Layout	Descript.	DS
ZTEST1	Test	
TEST USER	User specific	

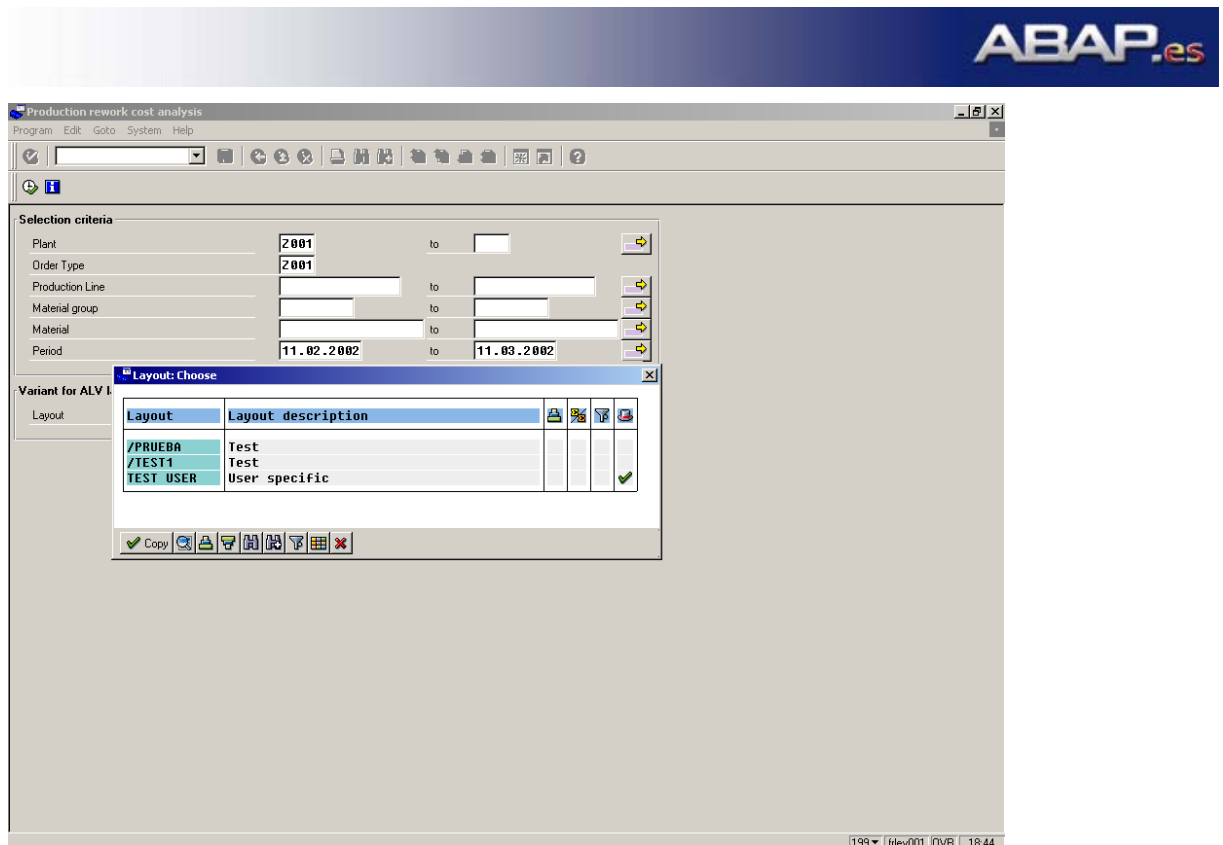
Save layout: /Prueba

Name: Test

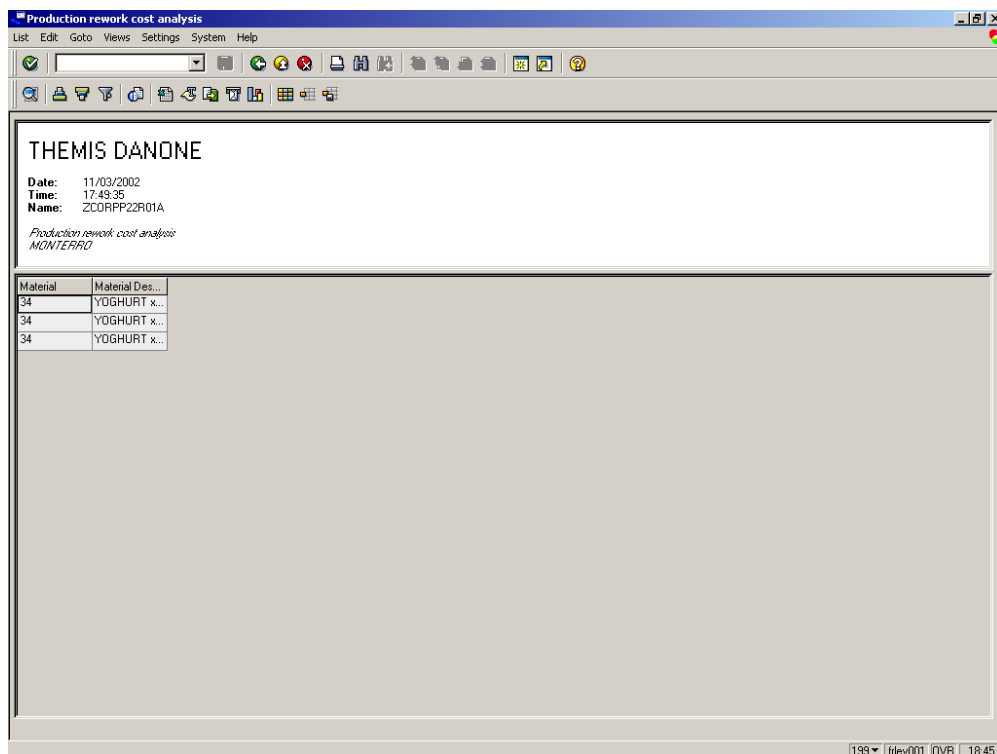
☐ User-specific ☐ Default setting

199 | filev001 | DVR | 18:43

En una ejecución posterior del programa, ya tenemos nuestro layout disponible para ser seleccionado:



Al elegir el layout '/Prueba' y ejecutar el programa, la nueva apariencia del report es la definida por nosotros en el layout /Prueba:



3.4.- I_CALLBACK_USER_COMMAND.

En este parámetro se le informará a la función del nombre de la subrutina donde irán codificadas todas las interacciones del usuario con el Report. En esta subrutina se implementará la respuesta del programa a ciertos códigos de función ejecutados.

La subrutina tendrá dos parámetros de entrada:

- a. L_UCOMM like SY-UCOMM.
En este parámetro vendrá informado el código de función ejecutado por el usuario.
- b. L_SELFIELD type SLIS_SELFIELD.
En este parámetro vendrá informado el registro actual del listado seleccionado por el usuario para interactuar sobre él.

Ejemplo:

Imaginemos que la subrutina se llamará 'USER_COMMAND' y que lo que queremos es visualizar el documento en el cual el usuario hará doble-click. Tendremos que codificarla en nuestro programa de la siguiente manera:

```
FORM USER_COMMAND USING L_UCOMM      LIKE SY-UCOMM
                        L_SELFIELD TYPE SLIS_SELFIELD.

  READ TABLE I_BSEG INDEX L_SELFIELD-TABINDEX.
  IF SY-SUBRC = 0.
    CASE L_UCOMM.
      WHEN '&IC1'.
        " Doble click
        SET PARAMETER ID 'BLN' FIELD I_BSEG-BELNR.
        CALL TRANSACTION 'FB03' AND SKIP FIRST SCREEN.
    ENDCASE.
  ENDIF.
ENDFORM.
```

3.5.- IS_LAYOUT.

En este parámetro podemos pasarle una tabla interna informando de la apariencia del report. Esta apariencia podrá luego ser modificada desde el propio report.

La tabla interna tendrá que ser declarada del tipo SLIS_LAYOUT_ALV. Entre otros, se puede definir si mostrar líneas verticales y horizontales (éstas últimas solo en el LIST, en el GRID parece no funcionar bien), los colores de los campos, si el report debe ser scrollable, etc. Entre ellos, uno de los más importantes o que más se suele pedir es mostrar el report sin mostrar los valores repetidos para cierto campo en distintos registros. Esto se hará marcando con 'X' el campo CELL_MERGE de esta tabla interna. Para que esto funcione, el listado debe estar ordenado por este campo. Aquí hay una breve descripción de algunos valores de esta tabla:

NO_COLHEAD	:	No habrá títulos en las columnas.
NO_HOTSPOT	:	Las cabaceras no serán como hotspot.
ZEBRA	:	Aparecerá una línea blanca otra gris, y así.
NO_VLINE	:	Oculto las líneas verticales.
CELL_MERGE	:	No suprime los campos repetidos.
EDIT	:	Hace el listado editable.

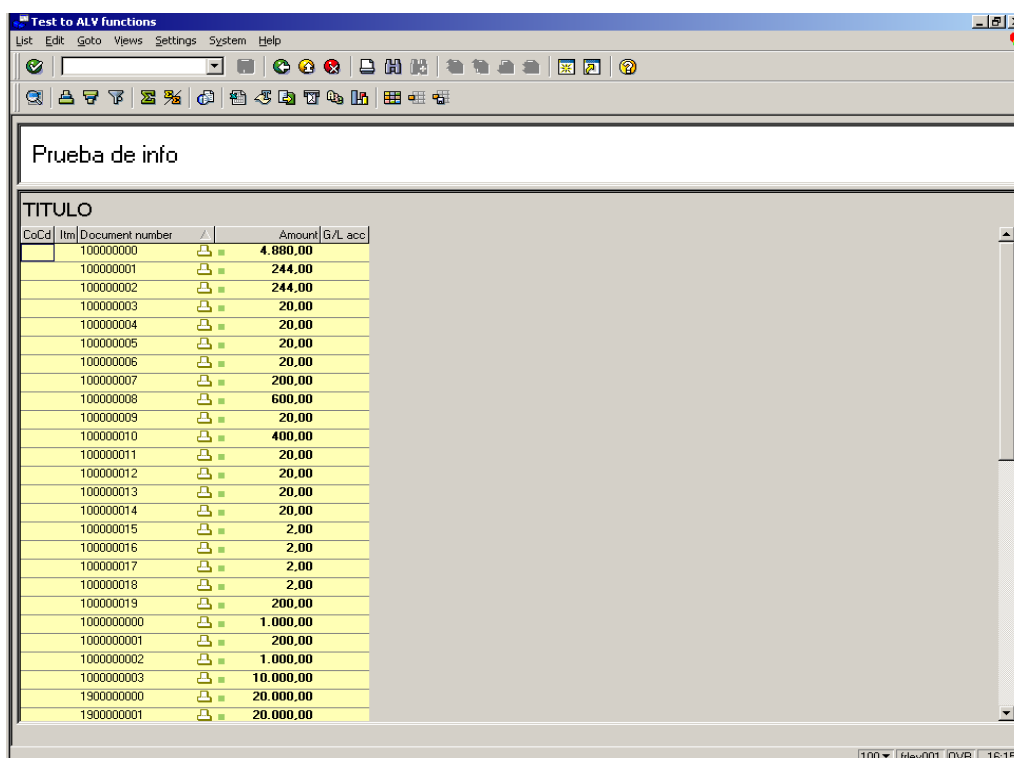
COLWIDTH_OPTIMIZE: Para cada registro, muestra el ancho de columna apropiado.

3.6.- IT_SORT.

Define los criterios de ordenación con el que se mostrarán los datos en el listado, así como aquellos campos por los cuales debe calcularse totales y subtotales. Se da el nombre del campo por el cual se va a ordenar, el criterio de ordenación, si ascendente o descendente, la posición de la columna que ocupa, si se van a calcular subtotales por ese campo o no, si queremos que los subtotales aparezcan comprimidos o expandidos, etc. Por ejemplo, poniendo estos valores conseguiremos que nuestro report ordene y subtotalize por el campo número de documento:

```
IT_SORT-FIELDNAME    = 'BELNR'.
IT_SORT-SPOS         = 2.
IT_SORT-UP           = 'X'.
IT_SORT-SUBTOT       = 'X'.
IT_SORT-EXPA        = 'X'.
```

El resultado de esto es el siguiente:



CoCd	Itm	Document number	Amount	G/L acc
		100000000	4.880,00	
		100000001	244,00	
		100000002	244,00	
		100000003	20,00	
		100000004	20,00	
		100000005	20,00	
		100000006	20,00	
		100000007	200,00	
		100000008	600,00	
		100000009	20,00	
		100000010	400,00	
		100000011	20,00	
		100000012	20,00	
		100000013	20,00	
		100000014	20,00	
		100000015	2,00	
		100000016	2,00	
		100000017	2,00	
		100000018	2,00	
		100000019	200,00	
		1000000000	1.000,00	
		1000000001	200,00	
		1000000002	1.000,00	
		1000000003	10.000,00	
		1900000000	20.000,00	
		1900000001	20.000,00	

De forma que si pinchamos en cualquier punto verde, desplegamos la lista de valores que pertenecen a ese número de documento:

Manual para funciones ALV V 1.0
Página 16 de 31

Manual para funciones ALV V 1.0
Página 17 de 31

4.- EJEMPLO DE REUSE_ALV_LIST_DISPLAY Y REUSE_ALV_GRID_DISPLAY.

A continuación se indica un pequeño ejemplo de cómo realizar report utilizando bien una LIST Display bien una GRID Display.

El report sería el siguiente:

```
*-----*
*          *****
*          >
*          Accenture
*          * Enterprise Business Solution Center SAP *
*          *****
*-----*
*-----*
```

***** DATA AREA (Begin) *****

*- INCLUDE DE TIPOS

TYPE-POOLS: slis. " ALV Type Pool

*- TABLAS DDIC

TABLES : bkpfbseg.

DATA:

WA_TITLE TYPE SLIS_LISTHEADER.

*- TABLAS INTERNAS

DATA: BEGIN OF i_bseg OCCURS 0,

bukrs LIKE bseg-bukrs,

belnr LIKE bseg-belnr,

buzei LIKE bseg-buzei,

wrbtr LIKE bseg-wrbtr,

waers LIKE bkpfbseg,

hkont LIKE bseg-hkont,

END OF i_bseg.

DATA gflg_rule.

*- CONSTANTES

CONSTANTS : c_form_top_of_page TYPE slis_formname VALUE 'TOP_OF_PAGE',

c_form_before_line TYPE slis_formname VALUE
'BEFORE_LINE_OUTPUT'.

*- VARIABLES DEL REPORT

* Cabecera report (STD)

DATA: d_anchoinforme TYPE i, " ANCHO DEL INFORME

d_tituloinforme1(50), " TITULO INFORME SUPERIOR

d_tituloinforme2(50), " TITULO INFORME INFERIOR

D_BUKRS LIKE BSEG-BUKRS.

*- Vars del ALV

DATA: d_fieldcat TYPE slis_t_fieldcat_alv,

d_fieldcat_ln LIKE LINE OF d_fieldcat,

d_sortcat TYPE slis_t_sortinfo_alv,

d_sortcat_ln LIKE LINE OF d_sortcat,

d_eventcat TYPE slis_t_event,

d_eventcat_ln LIKE LINE OF d_eventcat,

D_LIST TYPE slis_t_listheader,

```
I_TITLE      TYPE SLIS_LISTHEADER OCCURS 0,
D_LAYCAT     TYPE SLIS_LAYOUT_ALV. " OCCURS 0.
```

```
DATA: d_colpos TYPE i.
```

```
DATA: I_VARIAN LIKE DISVARIANT OCCURS 0 WITH HEADER LINE,
      E_VARIAN LIKE DISVARIANT OCCURS 0 WITH HEADER LINE,
      I_EXTAB  TYPE SLIS_EXTAB OCCURS 0.
```

```
PARAMETER:
```

```
RB_UNO TYPE C RADIOBUTTON GROUP AB1,
RB_DOS TYPE C RADIOBUTTON GROUP AB1 DEFAULT 'X'.
```

```
PARAMETER: P_VARIAN LIKE DISVARIANT-VARIANT.
```

```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_VARIAN.
```

```
I_VARIAN-REPORT = SY-REPID.
```

```
I_VARIAN-USERNAME = SY-UNAME.
```

```
APPEND I_VARIAN.
```

```
CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
```

```
EXPORTING
```

```
IS_VARIANT          = I_VARIAN
```

```
IT_DEFAULT_FIELDCAT = D_FIELDCAT
```

```
I_SAVE              = 'A'
```

```
IMPORTING
```

```
ES_VARIANT          = I_VARIAN
```

```
EXCEPTIONS
```

```
NOT_FOUND           = 1
```

```
PROGRAM_ERROR       = 2
```

```
OTHERS              = 3
```

```
IF SY-SUBRC <> 0.
```

```
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
```

```
* WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
```

```
ELSE.
```

```
P_VARIAN = i_VARIAN-VARIANT.
```

```
ENDIF.
```

```
TOP-OF-PAGE.
```

```
WRITE: / SY-TITLE.
```

```
WRITE: / 'Día:',
```

```
SY-DATUM,
```

```
SPACE,
```

```
'Página:',
```

```
SY-PAGNO.
```

```
***** DATA AREA ( End ) *****
```

```
START-OF-SELECTION.
```

```
***** SELECT DATA ( Begin ) *****
```

```
CLEAR BKPF.
```

```
SELECT * FROM bkp
```

```
where bukrs = '0002'
```

```
and gjahr = '2002'.
```

```
SELECT BUKRS BUZEI BELNR WRBTR WAERS
```

```
APPENDING TABLE I_BSEG
```

```
FROM bseg WHERE bukrs = bkp-bukrs AND
```

```
belnr = bkp-belnr AND
```

```
gjahr = bkp-gjahr.
```

ENDSELECT.

***** SELECT DATA (END) *****

CHECK NOT I_BSEG IS INITIAL.

**** 1.- CONSTRUCCIÓN CATÁLOGO DE DATOS

PERFORM f_field_kat.

**** 2.- CONSTRUCCIÓN CATÁLOGO DE ORDENACIÓN

PERFORM f_field_sort.

**** 3.- CONSTRUCCIÓN CATÁLOGO DE EVENTOS

PERFORM f_field_event.

**** 4.- CONSTRUCCIÓN DEL LAYOUT

PERFORM F_FIELD_LAYOUT.

WA_TITLE-TYP = 'H'.

WA_TITLE-KEY = ' '.

WA_TITLE-INFO = 'Prueba de info'.

append WA_title TO I_TITLE.

IF NOT RB_UNO IS INITIAL.

***** START A L V (Begin) *****

PERFORM f_start_alv.

***** START A L V (End) *****

ELSE.

PERFORM F_START_GRID_ALV.

ENDIF.

PERFORM status_set using I_EXTAB.

END-OF-SELECTION.

***** FORM ROUTINES *****

&-----

*& Form f_field_kat

&-----

FORM f_field_kat.

* Paso 1 - Se definen los campos clave (key = 'X')

* Sociedad

d_colpos = d_colpos + 1.

d_fieldcat_ln-ref_tabname = 'BSEG'. " Tabla Interna

d_fieldcat_ln-fieldname = 'BUKRS'. " Campo de la Tabla Interna

d_fieldcat_ln-key = space. " Clave = 'X' / No clave = space

d_fieldcat_ln-do_sum = space. " Acepta Sumatorio = 'X' /

" No acepta sumatorio = space

d_fieldcat_ln-col_pos = d_colpos. " Posición de la columna

d_fieldcat_ln-no_out = space. " Columna invisible = 'X' /

" Columna visible = space

d_fieldcat_ln-qfieldname = SPACE.

d_fieldcat_ln-hotspot = space. " Hotspot on = 'X' /

" Hotspot off = space

APPEND d_fieldcat_ln TO d_fieldcat. " Traspasamos los datos

* POS

d_colpos = d_colpos + 1.

d_fieldcat_ln-ref_tabname = 'BSEG'.

d_fieldcat_ln-fieldname = 'BUZEI'.

d_fieldcat_ln-key = 'X'.

d_fieldcat_ln-do_sum = space.

d_fieldcat_ln-col_pos = d_colpos.

```
d_fieldcat_ln-no_out      = space.
d_fieldcat_ln-qfieldname  = space.
d_fieldcat_ln-hotspot     = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* N° DOC

```
d_colpos                  = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BSEG'.
d_fieldcat_ln-fieldname   = 'BELNR'.
d_fieldcat_ln-key         = space.
d_fieldcat_ln-do_sum      = space.
d_fieldcat_ln-col_pos     = d_colpos.
d_fieldcat_ln-no_out      = space.
d_fieldcat_ln-qfieldname  = space.
d_fieldcat_ln-hotspot     = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* IMPORTE

```
d_colpos                  = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BSEG'.
d_fieldcat_ln-fieldname   = 'WRBTR'.
d_fieldcat_ln-key         = space.
d_fieldcat_ln-do_sum      = 'X'.
d_fieldcat_ln-col_pos     = d_colpos.
d_fieldcat_ln-no_out      = space.
d_fieldcat_ln-qfieldname  = SPACE.
d_fieldcat_ln-hotspot     = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* MONEDA

```
d_colpos                  = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BKPF'.
d_fieldcat_ln-fieldname   = 'WAERS'.
d_fieldcat_ln-key         = space.
d_fieldcat_ln-do_sum      = space.
d_fieldcat_ln-col_pos     = d_colpos.
d_fieldcat_ln-no_out      = 'X'.
d_fieldcat_ln-qfieldname  = space.
d_fieldcat_ln-hotspot     = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* HKONT

```
d_colpos                  = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BSEG'.
d_fieldcat_ln-fieldname   = 'HKONT'.
d_fieldcat_ln-key         = space.
d_fieldcat_ln-do_sum      = space.
d_fieldcat_ln-col_pos     = d_colpos.
d_fieldcat_ln-no_out      = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

ENDFORM. " f_field_kat

```
*&-----*
*& Form f_field_sort
*&-----*
```

```
FORM f_field_sort.
  DATA: ls_sort TYPE slis_sortinfo_alv.
```

```
ls_sort-fieldname = 'BELNR'.
ls_sort-spos      = 2.
ls_sort-up        = 'X'.
ls_sort-subtot    = 'X'.
LS_SORT-EXPA      = 'X'. " Sirve para que salgan los subtotales
                        plegados
```

```

APPEND ls_sort TO d_sortcat.

ENDFORM.                " f_field_sort

*&-----*
*&   Form f_field_event
*&-----*
FORM f_field_event.
  DATA: ls_event TYPE slis_alv_event.
  * Para que la función entre dentro de los eventos standares del
  * reporting standar tenemos 2 opciones :

  * OPCIÓN 1.- Escoger que eventos deseamos que se produzcan.
  * " EJEMPLO
  * d_eventcat_ln-name = 'TOP_OF_PAGE'.
  * APPEND d_eventcat.

  * OPCIÓN 2.- Dejar a Sap que nos recoja todos los eventos que
  * posee un report y volcarlos en nuestra tabla de
  * eventos.
  CALL FUNCTION 'REUSE_ALV_EVENTS_GET'
    EXPORTING
      i_list_type = 0
    IMPORTING
      et_events  = d_eventcat.

  * Una vez tenemos los eventos , debemos asignar a cada uno de
  * aquellos que nos interese el form que lo controlará
  READ TABLE d_eventcat WITH KEY name = slis_ev_top_of_page
    INTO ls_event.
  IF sy-subrc = 0.

    MOVE c_form_top_of_page TO ls_event-form.
    APPEND ls_event TO d_eventcat.
  ENDIF.

  CLEAR ls_event.

  READ TABLE d_eventcat WITH KEY name = slis_ev_before_line_output
    INTO ls_event.
  IF sy-subrc = 0.

    MOVE c_form_before_line TO ls_event-form.
    APPEND ls_event TO d_eventcat.
  ENDIF.

ENDFORM.                " f_field_event

*&-----*
*&   Form f_start_alv
*&-----*
FORM f_start_alv.
  DATA : l_repid TYPE sy-repid.
  l_repid = sy-repid.
  CALL FUNCTION 'REUSE_ALV_LIST_DISPLAY'
    EXPORTING
      i_callback_program           = L_REPID
      i_callback_user_command      = 'USER_COMMAND'
      IS_LAYOUT                    = D_LAYCAT
      it_fieldcat                  = D_FIELDCAT
      it_sort                      = D_SORTCAT
      i_save                       = 'A'
      IS_VARIANT                   = I_VARIAN
      it_events                    = D_EVENTCAT
  TABLES

```

```

t_outtab                                = I_BSEG
EXCEPTIONS
  program_error                          = 1
  OTHERS                                = 2
.
IF sy-subrc <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*   WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.

ENDFORM.                                " f_start_alv

*-----*
*   FORM F_USER_COMMAND                  *
*-----*
* Podemos controlar todos los user_command activados por el usuario *
*-----*
* --> L_UCOMM                            *
* --> L_SELFIELD                         *
*-----*
FORM user_command USING l_ucomm LIKE sy-ucomm
                        l_selfield TYPE slis_selfield.

READ TABLE i_bseg INDEX l_selfield-tabindex.
IF sy-subrc = 0.
  CASE l_ucomm.
    WHEN '&IC1'.                        " HotSpot
      SET PARAMETER ID 'BLN' FIELD I_BSEG-BELNR.
      CALL TRANSACTION 'FB03' AND SKIP FIRST SCREEN.
  ENDCASE.
ENDIF.
ENDFORM.

*-----*
*   FORM status_set                     *
*-----*
* Podemos definir nuestro PF-STATUS. Incluso podemos aprovechar el
* que nos da el ALV y añadir los que nos interese, o eliminar aquellos
* del ALV que no nos interesen
*-----*
* --> RT_EXTAB                           *
*-----*
FORM status_set USING rt_extab TYPE slis_t_extab.
data: l_status(20) type c.

SET PF-STATUS 'STANDARD' EXCLUDING rt_extab.
ENDFORM.

*-----*
*   FORM top_of_page                    *
*-----*
*
*-----*
FORM top_of_page.
d_anchoinforme = 80.
WRITE: /  'INFORME DE PRUEBA',
        50 'PAGINA:',
        60 SY-PAGNO.

CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
  EXPORTING
    it_list_commentary      = I_TITLE[]
*   I_LOGO                  =
*   I_END_OF_LIST_GRID      =

```

ENDFORM.

```

**-----*
**  FORM before_line_output      *
**-----*

```

FORM before_line_output USING p_ll TYPE kkblo_lineinfo.

```

IF D_BUKRS NE I_BSEG-BUKRS.
  D_BUKRS = I_BSEG-BUKRS.
  IF P_LL-TABINDEX GT 1.
    ULINE / (45).
    NEW-PAGE.
  ENDIF.
ENDIF.

```

ENDFORM.

```

*&-----*
*&  Form F_START_GRID_ALV      *
*&-----*

```

FORM F_START_GRID_ALV.

```

DATA: L_REPID LIKE SY-REPID.
L_REPID = SY-REPID.

```

```

CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
  EXPORTING
    I_CALLBACK_PROGRAM      = L_REPID
    I_CALLBACK_USER_COMMAND = 'USER_COMMAND'
    I_GRID_TITLE            = 'TITULO'
    IS_LAYOUT               = D_LAYCAT
    IT_FIELDCAT             = D_FIELDCAT
    IT_SORT                 = D_SORTCAT
    I_SAVE                  = 'A'
    IS_VARIANT              = I_VARIAN
    IT_EVENTS               = D_EVENTCAT
  TABLES
    T_OUTTAB                I_BSEG
  EXCEPTIONS
    PROGRAM_ERROR           = 1
    OTHERS                  = 2
.

```

IF SY-SUBRC <> 0.

```

* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
*   WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.

```

ENDFORM. " F_START_GRID_ALV

```

*&-----*
*&  Form F_FIELD_LAYOUT        *
*&-----*
*  Definir el layout de salida
*-----*

```

Form F_FIELD_LAYOUT.

DATA: LS_LAYCAT TYPE SLIS_LAYOUT_ALV.

```

CLEAR LS_LAYCAT.
D_LAYCAT-no_colhead      = SPACE.
D_LAYCAT-no_hotspot      = SPACE. " headings not as hotspot
D_LAYCAT-Zebra           = 'X'.   " striped pattern
D_LAYCAT-no_vline        = 'X'.   "SPACE.
D_LAYCAT-cell_merge      = 'X'.   " not suppress field replication

```

```
* D_LAYCAT-Edit          = SPACE.  
* D_LAYCAT-edit_mode     = space.
```

```
Endform.                " F_FIELD_LAYOUT
```

5.- EJEMPLO DE REUSE_ALV_HIERSEQ_LIST_DISPLAY.

***** DATA AREA (Begin) *****

*- INCLUDE DE TIPOS

TYPE-POOLS: slis, " ALV Type Pool
KKBLO.

*- TABLAS DDIC

TABLES : bkpfbseg.

DATA:

WA_TITLE TYPE SLIS_LISTHEADER.

*- TABLAS INTERNAS

DATA: BEGIN OF i_bseg OCCURS 0,

bukrs LIKE bseg-bukrs,

belnr LIKE bseg-belnr,

buzei LIKE bseg-buzei,

wrbtr LIKE bseg-wrbtr,

waers LIKE bkpfbseg-waers,

hkont LIKE bseg-hkont,

END OF i_bseg.

DATA: BEGIN OF i_HEADER OCCURS 0,

bukrs LIKE bseg-bukrs,

BELNR LIKE bseg-BELNR,

END OF i_header.

DATA gflg_rule.

*- CONSTANTES

CONSTANTS : c_form_top_of_page TYPE slis_formname VALUE 'TOP_OF_PAGE',
c_form_before_line TYPE slis_formname VALUE
'BEFORE_LINE_OUTPUT'.

*- VARIABLES DEL REPORT

* Cabecera report (STD)

DATA: d_anchoinforme TYPE i, " ANCHO DEL INFORME

d_tituloinforme1(50), " TITULO INFORME SUPERIOR

d_tituloinforme2(50), " TITULO INFORME INFERIOR

D_BUKRS LIKE BSEG-BUKRS.

*- Vars del ALV

DATA: d_fieldcat TYPE slis_t_fieldcat_alv,

d_fieldcat_ln LIKE LINE OF d_fieldcat,

d_sortcat TYPE slis_t_sortinfo_alv,

d_sortcat_ln LIKE LINE OF d_sortcat,

d_eventcat TYPE slis_t_event,

d_eventcat_ln LIKE LINE OF d_eventcat,

D_LIST TYPE slis_t_listheader,

* I_TITLE LIKE D_LIST OCCURS 0.

I_TITLE TYPE SLIS_LISTHEADER OCCURS 0,

D_LAYCAT TYPE SLIS_LAYOUT_ALV,

D_KEYINFO TYPE slis_keyinfo_alv.

DATA: d_colpos TYPE i.

DATA: I_VARIAN LIKE DISVARIANT OCCURS 0 WITH HEADER LINE,

```
E_VARIAN LIKE DISVARIANT OCCURS 0 WITH HEADER LINE,
I_EXTAB TYPE SLIS_EXTAB OCCURS 0.
```

PARAMETER: P_VARIAN LIKE DISVARIANT-VARIANT.

AT SELECTION-SCREEN ON VALUE-REQUEST FOR P_VARIAN.

```
I_VARIAN-REPORT = SY-REPID.
I_VARIAN-USERNAME = SY-UNAME.
APPEND I_VARIAN.
```

```
CALL FUNCTION 'REUSE_ALV_VARIANT_F4'
EXPORTING
  IS_VARIANT      = i_VARIAN
  IT_DEFAULT_FIELDCAT = d_fieldcat
  I_SAVE          = 'A'
IMPORTING
  ES_VARIANT      = I_VARIAN
EXCEPTIONS
  NOT_FOUND       = 1
  PROGRAM_ERROR   = 2
  OTHERS          = 3
```

```
IF SY-SUBRC <> 0.
* MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
* WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ELSE.
  P_VARIAN = i_VARIAN-VARIANT.
ENDIF.
```

```
TOP-OF-PAGE.
WRITE: / SY-TITLE.
WRITE: / 'Día:',
       SY-DATUM,
       SPACE,
       'Página:',
       SY-PAGNO.
***** DATA AREA ( End ) *****
```

```
START-OF-SELECTION.
***** SELECT DATA ( Begin ) *****
CLEAR BKPF.
```

```
SELECT * FROM bkpj
where gjahr = '2002'.

SELECT * FROM bseg WHERE bukrs = bkpj-bukrs AND
              belnr = bkpj-belnr AND
              gjahr = bkpj-gjahr.
  i_header-bukrs = bseg-bukrs.
  I_BSEG-BUKRS = BSEG-BUKRS.
  i_bseg-buzei = bseg-buzei.
  i_bseg-belnr = bseg-belnr.
  i_HEADER-belnr = bseg-belnr.
  i_bseg-wrbtr = bseg-wrbtr.
  i_bseg-waers = bkpj-waers.
  I_BSEG-HKONT = BSEG-HKONT.
  APPEND i_bseg.
  append i_header.
ENDSELECT.
ENDSELECT.
***** SELECT DATA ( END ) *****
```

CHECK NOT I_BSEG IS INITIAL.

**** 1.- CONSTRUCCIÓN CATÁLOGO DE DATOS

PERFORM f_field_kat.

**** 3.- CONSTRUCCIÓN CATÁLOGO DE EVENTOS

PERFORM f_field_event.

**** 4.- CONSTRUCCIÓN DEL LAYOUT

PERFORM F_FIELD_LAYOUT.

WA_TITLE-TYP = 'H'.

WA_TITLE-KEY = ''.

WA_TITLE-INFO = 'Prueba de info'.

append WA_title TO I_TITLE.

PERFORM f_start_alv.

PERFORM status_set using I_EXTAB.

END-OF-SELECTION.

***** FORM ROUTINES *****

&-----

*& Form f_field_kat

&-----

FORM f_field_kat.

* Paso 1 - Se definen los campos clave (key = 'X')

* Sociedad

d_colpos = d_colpos + 1.

d_fieldcat_ln-ref_tabname = 'BSEG'. " Tabla Interna

D_FIELDCAT_LN-TABNAME = 'I_HEADER'.

d_fieldcat_ln-fieldname = 'BUKRS'. " Campo de la Tabla Interna

d_fieldcat_ln-key = space. " Clave = 'X'/No clave = space

d_fieldcat_ln-do_sum = space. " Acepta Sumatorio = 'X' /

" No acepta sumatorio = space

d_fieldcat_ln-no_out = space. " Columna invisible = 'X' /

" Columna visible = space

d_fieldcat_ln-qfieldname = SPACE.

d_fieldcat_ln-hotspot = space. " Hotspot on = 'X' /

" Hotspot off = space

APPEND d_fieldcat_ln TO d_fieldcat. " Traspasamos los datos

* POS

d_colpos = d_colpos + 1.

d_fieldcat_ln-ref_tabname = 'BSEG'.

D_FIELDCAT_LN-TABNAME = 'I_BSEG'.

d_fieldcat_ln-fieldname = 'BUZEI'.

d_fieldcat_ln-key = 'X'.

d_fieldcat_ln-do_sum = space.

d_fieldcat_ln-col_pos = d_colpos.

d_fieldcat_ln-no_out = space.

d_fieldcat_ln-qfieldname = space.

d_fieldcat_ln-hotspot = space.

APPEND d_fieldcat_ln TO d_fieldcat.

* N° DOC

d_colpos = d_colpos + 1.

d_fieldcat_ln-ref_tabname = 'BSEG'.

D_FIELDCAT_LN-TABNAME = 'I_HEADER'.

d_fieldcat_ln-fieldname = 'BELNR'.

d_fieldcat_ln-key = space.

```
d_fieldcat_ln-do_sum    = space.
d_fieldcat_ln-col_pos   = d_colpos.
d_fieldcat_ln-no_out    = space.
d_fieldcat_ln-qfieldname = space.
d_fieldcat_ln-hotspot   = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* IMPORTE

```
d_colpos                = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BSEG'.
d_fieldcat_ln-fieldname   = 'WRBTR'.
D_FIELDCAT_LN-TABNAME    = 'I_BSEG'.
d_fieldcat_ln-key        = space.
d_fieldcat_ln-do_sum     = 'X'.
d_fieldcat_ln-col_pos    = d_colpos.
d_fieldcat_ln-no_out     = space.
d_fieldcat_ln-qfieldname = SPACE.
d_fieldcat_ln-hotspot    = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* MONEDA

```
d_colpos                = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BKPF'.
D_FIELDCAT_LN-TABNAME    = 'I_BSEG'.
d_fieldcat_ln-fieldname   = 'WAERS'.
d_fieldcat_ln-key        = space.
d_fieldcat_ln-do_sum     = space.
d_fieldcat_ln-col_pos    = d_colpos.
d_fieldcat_ln-no_out     = 'X'.
d_fieldcat_ln-qfieldname = space.
d_fieldcat_ln-hotspot    = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

* HKONT

```
* d_colpos                = d_colpos + 1.
d_fieldcat_ln-ref_tabname = 'BSEG'.
D_FIELDCAT_LN-TABNAME    = 'I_BSEG'.
d_fieldcat_ln-fieldname   = 'HKONT'.
d_fieldcat_ln-key        = space.
d_fieldcat_ln-do_sum     = space.
* d_fieldcat_ln-col_pos    = d_colpos.
d_fieldcat_ln-no_out     = space.
APPEND d_fieldcat_ln TO d_fieldcat.
```

```
ENDFORM.                " f_field_kat
```

```
*&-----*
```

```
*&   Form f_field_event
```

```
*&-----*
```

```
FORM f_field_event.
  DATA: ls_event TYPE slis_alv_event.
  CALL FUNCTION 'REUSE_ALV_EVENTS_GET'
    EXPORTING
      i_list_type = 0
    IMPORTING
      et_events   = d_eventcat.
```

```
READ TABLE d_eventcat WITH KEY name = slis_ev_top_of_page
  INTO ls_event.
```

```
IF sy-subrc = 0.
```

```
  MOVE c_form_top_of_page TO ls_event-form.
  APPEND ls_event TO d_eventcat.
```

```

ENDIF.

CLEAR ls_event.

READ TABLE d_eventcat WITH KEY name = slis_ev_before_line_output
      INTO ls_event.
IF sy-subrc = 0.

    MOVE c_form_before_line TO ls_event-form.
    APPEND ls_event TO d_eventcat.
ENDIF.

ENDFORM.          " f_field_event
*&-----*
*&   Form f_start_alv
*&-----*
FORM f_start_alv.
    DATA : l_repid TYPE sy-repid.
    l_repid = sy-repid.

    SORT l_HEADER.
    DELETE ADJACENT DUPLICATES FROM l_HEADER.

    SORT l_BSEG.

    D_keyinfo-header01 = 'BUKRS'.
    * D_keyinfo-item01  = 'BUKRS'.
    D_keyinfo-header02 = 'BELNR'.
    * D_keyinfo-item02  = 'BELNR'.

    CALL FUNCTION 'REUSE_ALV_HIERSEQ_LIST_DISPLAY'
      EXPORTING
        l_CALLBACK_PROGRAM      = l_REPID
        l_CALLBACK_USER_COMMAND = 'USER_COMMAND'
        is_LAYOUT               = D_LAYCAT
        it_FIELDCAT             = D_FIELDCAT
        l_SAVE                  = 'A'
        is_VARIANT              = l_VARIAN
        it_EVENTS               = D_EVENTCAT
        i_tabname_header        = 'l_HEADER'
        i_tabname_item          = 'l_BSEG'
        is_keyinfo              = D_KEYINFO
      tables
        t_outtab_header        = l_HEADER
        t_outtab_item          = l_BSEG
      EXCEPTIONS
        PROGRAM_ERROR          = 1
        OTHERS                  = 2
      .
    IF sy-subrc <> 0.
    * MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
    *   WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
    ENDIF.

ENDFORM.          " f_start_alv

*&-----*
*&   FORM F_USER_COMMAND
*&-----*
* Podemos controlar todos los user_command activados por el usuario *
*&-----*
* --> l_UCOMM
* --> l_SELFIELD
*&-----*
FORM user_command USING l_ucomm LIKE sy-ucomm
      l_selfield TYPE slis_selfield.

```

```

READ TABLE i_bseg INDEX l_selfield-tabindex.
IF sy-subrc = 0.
  CASE l_ucomm.
    WHEN '&IC1'.
      " HotSpot
      SET PARAMETER ID 'BLN' FIELD l_BSEG-BELNR.
      CALL TRANSACTION 'FB03' AND SKIP FIRST SCREEN.
    ENDCASE.
  ENDIF.
ENDFORM.

*-----*
*   FORM status_set                               *
*-----*
* Podemos definir nuestro PF-STATUS. Incluso podemos aprovechar el
* que nos da el ALV y añadir los que nos interese, o eliminar aquellos
* del ALV que no nos interesen
*-----*
* --> RT_EXTAB                                   *
*-----*
FORM status_set USING rt_extab TYPE slis_t_extab.
  data: l_status(20) type c.

  SET PF-STATUS 'STANDARD' EXCLUDING rt_extab.
ENDFORM.

* EVENTOS DEL REPORT -----*
*-----*
*   FORM top_of_page                               *
*-----*
*   .....                                         *
*-----*
FORM top_of_page.
  d_anchoinforme = 80.
  WRITE: / 'INFORME DE PRUEBA',
         50 'PAGINA:',
         60 SY-PAGNO.

CALL FUNCTION 'REUSE_ALV_COMMENTARY_WRITE'
  EXPORTING
    it_list_commentary = l_TITLE[]
  .

ENDFORM.

*&-----*
*&   Form F_FIELD_LAYOUT                          *
*&-----*
* Definir el layout de salida
*-----*
form F_FIELD_LAYOUT.
  DATA: LS_LAYCAT TYPE SLIS_LAYOUT_ALV.

  CLEAR LS_LAYCAT.
  D_LAYCAT-no_colhead = SPACE.
  D_LAYCAT-no_hotspot = SPACE.      " headings not as hotspot
  D_LAYCAT-Zebra = 'X'.            " striped pattern
  D_LAYCAT-no_vline = SPACE.
  D_LAYCAT-cell_merge = 'X'.        " not suppress field replication
endform.      " F_FIELD_LAYOUT

```