EUROPEAN ORGANISATION FOR NUCLEAR RESEARCH

PS/CO/Note 81-11
18.6.1981

# ENVIRONMENT POUR PROGRAMMES RT-PASCAL DANS LES CONSOLES

F. Perriollat, A. Gagnaire

## Table des matières

I. Introduction

II. Généralités

III. Règle d'introduction de nouveaux programmes compilés

IV. Fichier de déclaration de l'environnement Console

V. Fonctions appelables par ICCI Call

Annexe 1 : Appel des fonctions

Annexe 2 : Environnement console

Annexe 3 : Exemple

## I. Introduction

RT-PASCAL est disponible pour écrire des programmes (compilés) pour les consoles.

Deux types de programmes peuvent s'exécuter dans les consoles:

a) Des programmes non interactifs. Se sont des programmes de types SRT pour le cas compilé ainsi que "File-driven-NODAL" ou "remote-call" pour le cas NODAL. Ces programmes disposent des outils non interactifs des consoles.

b) Des programmes interactifs. Ces programmes s'exécutent sous un context xIP donné. Dans ce cas un "programme compilé" qui doit être défini comme "switch-program", s'exécute comme un overlay de l'xIP qui le démarre (par START) tout en conservant complètement le contexte de cet xIP. Il a donc l'entier accès à tous les outils qui sont alloués (de façon fixe) à cet xIP. En fait au point de vue de Sintran III ce programme utilise la même RT-discription et les mêmes ressources que l'xIP appelant.

Nous décrivons ci-dessous les moyens d'accès aux outils de consoles depuis RT-PASCAL.

## II. Généralités

L'ensemble des facilités des consoles sont disponibles. La disposition de ces outils se fait:

i) à travers un ensemble de routines appelables par ICCI call;

ii) par un fichier PASCAL de déclaration de l'environnement console.

La flexibilité d'utilisation des programmes compilé est moindre que pour des programmes Nodal (interprété). Ceci est la conséquence des remarques ci-contre.

i) Dans la version actuel de RT-PASCAL, lors d'un appel ICCI le stack Nodal généré a une longueur fixe (déterminé par un symbole d'assemblage). Ceci a pour conséquence que les activités qui demandent des tailles de stacks plus grande que cette valeur ne peuvent pas être exécutés.

ii) Les concaténations dynamiques à l'appel d'un routine n'ont aucun sens en compilé. Donc les appels de routines par ICCI call doivent toujours se faire avec des chaines de caractères précédemment construites.

iii) La résolution des symbols externes se fait au moment du chargement des segments. Afin que ces références (inter-segment) soient toujours cohérantes il importe d'observer strictement les règles ci-dessous d'introduction de nouveaux programmes compilés.

## III. Règle d'introduction de nouveaux programmes compilés

1.) Tout programme compilé doit être au préalable annoncé "système supervisor" des consoles et MCR (A. Gagnaire ou F. Perriollat) qui distribue les ressources nécessaires pour cela. (No. de segment, et autres ressources eventuellement pour les activités non interactives).

2.) Les "MODE-FILE" de chargement seront après test transmit à la section console qui introduira dans les procédures automatiques de rechargement après HENT, ou modification de segments références. Toutes modifications de ces "mode-file" devra se faire alors par une demande au "système supervisor".

3.) Le segment 100 (octal) est en permanence réservé aux tests.

## IV. Fichier de déclaration de l'environnement Console

Le fichier disponible sur le PRDEV (CONSOLE-SYS)PAS-CON-ENVIRONT:SYMB comporte 2 classes de symboles déclarés:

i) Des symboles correspondants aux constantes de Nodal pour les consoles. Les mêmes noms sont utilisés dans ce cas (maximum 6 caractères).

ii) Des symboles propres pour les programmes compilés. Ces symboles sont définis par 2 mots.

Il s'agit

- de symboles définissant la valeur de "FLAG" dans les appels ICCI.

- d'une valeur d'un numéro d'untié logiques (LUN) sur lequel il est toujours possible d'écrire sans réservation préalable ERROR-TERMINAL. Ce LUN suit en fait "error-device" de Sintran III.

- du code d'un caractère sans effet sur les écrans: DUMMY-CHAR. Ce caractère doit être utilisé en fin de toute chaine de caractères pour laquelle des espaces à la fin doivent être significatif pour des appels ICCI.

Un listing de ce fiction est donné en Annexe 2.

## V. Fonctions appelables par ICCI Call

- Toutes les fonctions des facilités consoles sont disponibles.
Sauf : BUTTON (qui va disparaître: cf. PS/CO/Note 81-6).

- Le nom par lequel est appelé ses facilités est en règle général le nom Nodal tronqué à 5 caractères (limitation du language BRF).
Les exceptions proviennent des noms réservés des divers languages utilisés.
Les principales exceptions sont :

| | | |
|---|---|---|
| WRITE | -----> | VIDEO |
| LINE | -----> | LINEP |
| POINT | -----> | POIN |

- Les facilités Nodal des consoles sont de type très divers (et non pas seulement le type 12 suivant la stricte conversion ICCI).

Ceci a pour conséquence que 1 ou 2 paramètres explicites doivent être rajoutés à la liste des paramètres explicites de Nodal. Ces paramètres supplémentaires couvrent les paramètres implicites des appels Nodal. A savoir:

i) FLAG, qui supporte à l'entrée le code du type d'appel (read, write, call, et qui retourne la condition d'exécution (code d'erreur ou 0).

ii) La valeur de la fonction : soit un réel soit une chaîne de caractère.

De plus toute fonction qui est appelée avec FLAG différent de CALL-FLAG ne peut pas être appelé par un REMOTE call.
Mais en général un remote call aux facilités des consoles n'est pas significatif car le contexte n'est pas défini de façon stable. En effet un "Remote-call" est exécute par l'un des esclaves Nodal du reseau.

L'ensemble des facilités est décrite dans l'annexe 1 (classement par ordre alphabétique des noms Nodal).

## VI. Annexes

Annexe 1 : description alphabéthique des fonctions appelables par ICCI

Annexe 2 : déclaraction d'environnement console

Annexe 3 : exemple de processus interactif (MIP) compilé : HANOI

- mode file de changement
- programme RT-PASCAL

## ANNEXE 1 : Appel des fonctions

NODAL name: AN1LUN

Pascal call ICCI('AN1LU', 'CONF', WO LUN, RW FLAG)

LUN: real
FLAG: integer

entry: FLAG: = read flag

Brief description :

Return the LUN of the 1st analogue touch-panel video.

Reference: :

NODAL name: AN2LUN

Pascal call ICCI('AN2LU', 'CONF1', WO LUN, RW FLAG)

LUN : real

FLAG: integer

entry: FLAG:= real_flag

Brief description :

Return the LUN of the 2nd analogue touch panel video.

Reference: :

NODAL name: ANAGR

Pascal call ICCI('ANAGR', 'CONF1', WO GR, RW FLAG)

```
GR   : real
FLAG : integer

entry : FLAG:=  real_flag
```

Brief description :

Return the analgoue touch panel group number.

Reference: :

NODAL name: ANALEG

Pascal call ICCI('ANALE', 'CONF1', RO STR, RW FLAG, RO BUTNB)

```
STR   : array of char
FLAG  : integer
BUTNB : integer

entry :   FLAG :=  write_flag
```

Brief description :

Write a legend to a button of analogue touch panel.

Reference: : PS/CO/Note 79-11

NODAL name: BALLGR

Pascal call ICCI('BALLG', 'CONF1', WO GR, RW FLAG)

```
GR  : real
FLAG: integer

entry : FLAG:= read_flag
```

Brief description :
Return the ball identification button group number.

Reference: :PS/CO/Note 79-11

NODAL name: BALLST

Pascal call ICCI('BALLS', 'CONF1', WO LUN, RW FLAG)

```
LUN : real
FLAG: integer

entry : FLAG:= read_flag
```

Brief description :

Return the devide to which the ball is connected.

Reference: :PS/CO/Note 79-11

NODAL name: BANDW

Pascal call ICCI('BANDW', 'CONF1', WO LUN, RW FLAG, RO INDEX)

```
LUN  : real
FLAG : integer
INDEX: integer


entry :  FLAG:= read_flag
```

Brief description :

Return the LUN of the black and write video device.

Reference: :PS/CO/Note 79-11

NODAL name: BLINK

Pascal call ICCI('BLINK', 'GRAF1', RW FLAG, RO OBJECT)

FLAG : integer
OBJECT: integer

entry : FLAG:= call_flag

Brief description :

Set the object blinking

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

<u>NODAL name:</u> BRITRG

<u>Pascal call</u> ICCI('BRITR', CONF1', RW VAL, RW FLAG, RO SCOPE)

```
VAL : real
FLAG : integer
SCOPE : integer

entry :  FLAG:= read-flag  for read
         FLAG:= write-flag for setting
```

<u>Brief description</u> :

Set or read the bright-up counter

<u>Reference:</u> : PS/CO/Note 81-23

NODAL name: BUTS

Pascal call ICCI('BUTS', 'CONF1', WO VAL, RW FLAG, RD GR)

```
VAL  : real
FLAG : integer
GR   : integer

entry :  FLAG :=  read_flag
```

Brief description :

Immediate read of a group

Reference: : PS/CO/Note 78-20 & PS/CO/Note 79-11

NODAL name: CHNSTA

Pascal call ICCI('CHNST', 'GRAF1', RW FLAG, RO CHANNEL, WO DATA)

```
FLAG : integer
CHANNEL: integer
DATA   : array of integer


entry :  FLAG:= call_flag
```

Brief description :

Read the graphic channel status.

Reference: :PS/CO/Note 80-33

NODAL name: CHSIZE

Pascal call  ICCI('CHSIZ', 'GRAF1', RW FLAG, RO CHS)

```
FLAG : integer
CHS  : integer

entry :  FLAG:= call_flag
```

Brief description :

Define the character size and orientation

Reference: :PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: COLOUR

Pascal call ICCI('COLOU', 'CONF1', WO LUN, RW FLAG)

LUN : real
FLAG: integer

entry : FLAG:= read _flag

Brief description :

Return the LUN of the colour video device

Reference: : PS/CO/Note 79-11

NODAL name: COLUMN

Pascal call ICCI('COLUM', 'CONF1', WO STR, RW FLAG, RO XVAL)

```
STR : array of char
FLAG: integer
XVAL: integer
```

```
entry : FLAG:= read_flag
```

Brief description :

Return a string to do a cursor column positioning on a video device.

Reference: : PS/CO/Note 77-16

<u>NODAL name:</u> DELAST

<u>Pascal call</u> ICCI('DELAS', 'GRAF1', RW FLAG, RO X1, RO Y1, RO X2, RO Y2)

FLAG:= integer
X1, Y1, X2, Y2 : real

entry : FLAG:= call_flag

<u>Brief description</u> :

Connect an elastic broken line between the specified point to the ball.

<u>Reference:</u> : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: DTMTRG

Pascal call ICCI('DTMTR', 'CONF1', RO VAL, RW FLAG,
RO SCOPE, RO DELAI_TYPE)

VAL : real
FLAG: integer
SCOPE, DELAI_TYPE : integer

entry : FLAG:= write_flag

Brief description :

Incremental write a scope-trigger counter

Reference: : PS/CO/Note 81-23

NODAL name: ELASTI

Pascal call ICCI('ELAST', 'GRAF1', RW FLAG, RO X, RO Y)

```
FLAG : integer
X, Y : real


entry :  FLAG:= call_flag
```

Brief description :

Connect an elastic vector to the ball.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: EVENT

Pascal call

```
ICCI('EVENT', 'CONF1', WO GR, RW FLAG,
RO MACHINE, RO PULSE, RO PLS)

GR : real
FLAG : integer
MACHINE : integer
PULSE : integer
PLS : integer


entry :  FLAG:= read_flag
```

Brief description :

Activate the specified event, and return the machine event group number.

Reference: : PS/CO/Note 78-20 & PS/CO/Note 79-11.

NODAL name:  FLUSH

Pascal call  ICCI('FLUSH', 'GRAF1', RW FLAG)

FLAG : integer

entry : FLAG:= call_flag

Brief description :

Graphic channel initialisation.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: FUNSTA

Pascal call ICCI('FUNST', 'GRAF1', RW FLAG, WO DATA)

```
FLAG : integer
DATA : array of integers
```

entry: FLAG:= call_flag

Brief description :

Read the graphic-primitives statistic.

Reference: : PS/CO/Note 80-33

NODAL name: GAPPEA

Pascal call ICCI('GAPPE', 'GRAF1', RW FLAG)

FLAG : integer

entry : FLAG:= call_flag

Brief description :

Make the cursor visible on the graphic screen.

Reference: : PS/CO/Note 80-33

NODAL name: GERASE

Pascal call ICCI('GERAS', 'GRAF1', RW FLAG)

FLAG : integer

entry : FLAG:= call_flag

Brief description :

Erase completely the graphic screen.

Reference: : PS/CO/Note 79-14 & PS/CO/note 80-33

NODAL name: GERMES

Pascal call ICCI('GERME', 'GRAF1', WO STG, RW FLAG, RO ERROR_CODE)

```
STG : array of char
FLAG : integer
ERROR_CODE : integer

entry : FLAG:= read_flag
```

Brief description :

Return the graphic error message corresponding to ERROR_CODE.

Reference: : PS/CO/Note 81-6

NODAL name:  GFA

Pascal call  ICCI('GFA', 'CONF1', RW FLAG, RO GFA, RO GFA_DATA)

```
FLAG : integer
GFA  : integer
GFA_DATA : array of integers
```

```
entry : FLAG:=  call_flag
```

Brief description :

Load a console GFA display.

Reference: : PS/CO/Note 81-23

NODAL name: GFABTG

Pascal call

```
ICCI('GFABT', 'CONF1', RW VAL, RW FLAG, RO GFA,
RO MARQUEUR)

VAL  : real
FLAG : integer
GFA, MARQUEUR : integer

entry : FLAG:= read_flag for read
        FLAG:= write_flag for setting
```

Brief description :

Set or read a GFA bright-up counter.

Reference: : PS/CO/Note 81-23

<u>NODAL name:</u> GFATRG

<u>Pascal call</u>

```
ICCI('GFATR', 'CONF1', RW VAL, RW FLAG;
RO GFA)

VAL : real
FLAG: integer
GFA : inters

entry : FLAG:= read_flag for read
        FLAG:= write_flag for setting
```

<u>Brief description</u> :

Set or read a GFA start counter.

<u>Reference:</u> : PS/CO/Note 81-23

NODAL name: GMODE

Pascal call ICCI('GMODE', 'GRAF1', RW FLAG, RO MODE)

```
FLAG : integer
MODE : integer


entry : FLAG:= call_flag
```

Brief description :

Define the graphique mode.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name:  GRAGR

Pascal call  ICCI/('GRAGR', 'CONF1', WO GR; RW FLAG)

```
GR : real
FLAG : integer


entry : FLAG:= read_flag
```

Brief description :

Return the graphic identification button group number.

Reference: :

NODAL name:  GRAPH

Pascal call  ICCI('GRAPH', 'CONF1', WO LUN, RW FLAG)

```
LUN  : real
FLAG : integer


entry. FLAG:= read_flag
```

Brief description :

Return the LUN  of the channel to the graphic system.

Reference: : PS/CO/Note 79-11

NODAL name: GRASK

Pascal call ICCI('GRASK', 'GRAF', WO VALUE, RW FLAG, RO OBJECT)

```
VALUE : real
FLAG :  integer
OBJECT: integer
```

```
entry :  FLAG:= read_flag
```

Brief description :

Edit the object, and return the evaluated value.

Reference: : PS/CO/Note 80-33

NODAL name: GRCHK

Pascal call ICCI('GRAF1', WO ERSTA, RW FLAG)

```
ERSTA : real
FLAG  : integer


entry :  FLAG:=  read_flag
```

Brief description :

Wait, if necessary, for completion of last graphic—system call, and return the status.

Reference: : PS/CO/Note 80-33

NODAL name: GREDIT

Pascal call ICCI('GREDI', 'GRAF1', WO STG, RW FLAG, RO OBJECT)

```
STG :  array of char
FLAG:  integer
OBJECT : integer
```

```
entry : FLAG:= read_flag
```

Brief description :

Edit and return the content of the object.

Reference: : PS/CO/Note 79-16 & PS/CO/Note 80-33

NODAL name: GRERR

Pascal call ICCI('GRERR', 'GRAF1', WO ERVAL, RW FLAG)

```
ERVAL : real
FLAG  : integer
```

entry : FLAG:= read_flag

Brief description :

Return the last graphic error status.

Reference: : PS/CO/Note 80-33

NODAL name: GWRITE

Pascal call ICCI('GWRIT', 'GRAF1', RW FLAG; RO STG)

```
FLAG : integer
STG  : array of char


entry : FLAG :=  call_flag
```

Brief description :

Write a string of characters on the screen.

Reference: : PS/CO/Note 80-33

NODAL name: IDFOBJ

Pascal call ICCI('IDFOB', 'GRAF1', WO OBJECT; RW FLAG)

```
OBJECT :  real
FLAG   :  integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Read the identified object.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: IMAGTR

Pascal call ICCI('IMAGT', 'CONF1', RW FLAG; RO PICTURE,
WO CONVERTED_PICTURE)

FLAG : integer
PICTURE, CONVERTED_PICTURE: array of integer

entry : FLAG := call_flag

Brief description :

Convert a colour picture into a back-and-white picture and vice-versa.

Reference: : PS/CO/Note 79-11

NODAL name:  IMGRES

Pascal call  ICCI('IMGRE', 'GRAF1, RW FLAG, RO DATA)

```
FLAG : integer
DATA : array of integer


entry :  FLAG := call_flag
```

Brief description :

Restore a complete graphic screen image.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name:  IMGSAV

Pascal call  ICCI('IMGSA', 'GRAF1', RW FLAG, WO DATA)

FLAG : integer
DATA : array of integer

entry :  FLAG := call_flag

Brief description :

Save the complete graphic screen image.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name:  KAPPEA

Pascal call  ICCI('KAPPE', 'CONF1', RW FLAG, RO LUN)

```
FLAG : integer
LUN  : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Turn on the cursor

Reference: : PS/CO/Note 79-11, PS/CO/Note 79-14
& PS/CO/Note 80-33

NODAL name: KCURC

Pascal call ICCI('KCURC', 'CONF1', RW VAL, RW FLAG)

```
VAL  : real
FALG : integer


entry :  FLAG := read_flag  for read
entry :  FLAG := write_flag for setting
```

Brief description :

Set or read the viedo cursor column position.

Reference: : PS/CO/Note 79-11

NODAL name:  KCURL

Pascal call  ICCI('KCURL', 'CONF1', RW VAL, RF FLAG)

```
VAL : real
FLAG: integer
```

```
entry :  FLAG := read_flag  for read
         FLAG := write_flag for setting
```

Brief description :

Set or read the viedo cursor line position.

Reference: : PS/CO/Note 79-11

NODAL name: KDSABL

Pascal call ICCI('KDSAB', 'CONF1', RW FLAG, RO LUN)

```
FLAG : integer
LUN  : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Disable and disconnect the cursor.

Reference: : PS/CO/Note 79-11, PS/CO/Note 79-14
& PS/CO/Note 80-33

NODAL name: KENABL

Pascal call ICCI('KENAB', 'CONF1', RW FLAG, RO LUN)

```
FLAG : integer
LUN  : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Connect the ball to the device.

Reference: : PS/CO/Note 79-11, PSD/CO/Note 79-14
& PS/CO/Note 80-33

NODAL name: KINIT

Pascal call

```
ICCI('KINIT', 'CONF1', RO VAL, RW FLAG,
RO KNB, RO MODE, RO VMAX, RO VMIN,
RO BBW, RO EDPT, RO UNST, RO TIST)
```

```
VAL : real
FLAG : integer
KNB, MODE, BBW, EDPT : integer
VMAX, VMIN : real
UNST, TIST : array of char
```

```
entry :  FLAG := write_flag
```

Brief description :

Initialize the knob.

Reference: : PS/CO/Note 79-02 & PS/CO/Note 79-11

NODAL name: KNOB

Pascal call ICCI('KNOB', 'CONF1', WO VAL, RW FLAG, RO KNOB)

```
VAL : real
FLAG: integer
KNB : integer
```

entry : FLAG:= read_value

Brief description :

Read the current value of knob

Reference: PS/CO/Note 79-02 & PS/CO/Note 79-11

NODAL name: KNVGR

Pascal call ICCI('KNVGR', 'CONF1', WO GR, RW FLAG)

```
GR  : real
FLAG: integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Return the knob validation group number.

Reference: : PS/CO/Note 79-11

NODAL name: KPURGE

Pascal call ICCI('KPURG', 'CONF1', RW FLAG, RO KNB)

```
FLAG : integer
KNB  : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Purge the knob.

Reference: : PS/CO/Note 79-02 & PS/CO/Note 79-11

NODAL name: KWRITE

Pascal call ICCI('KWRIT', 'CONF1', RO STG, RW FLAG, RO KNB)

STG : array of char
FLAG: integer
KNB : integer

entry : FLAG := write_flag

Brief description :

Write the text line on the knob display.

Reference: : PS/CO/Note 79-02 & PS/CO/Note 79-11

NODAL name: LEGEND

Pascal call ICCI('LEGEN', 'CONF1', RO STR, RW FLAG, RO BUTNB)

```
STR : array  of char
FLAG: integer
BUTNB : integer
```

```
entry :  FLAG := write_flag
```

Brief description :

Write a legend to a button of touch-panel.

Reference: : PS/CO/Note 79-11

NODAL name:  LEGLUN

Pascal call  ICCI('LEGLU', 'CONF1', WO LUN, RW FLAG)

```
LUN  : real
FLAG _ integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Return the LUN of the touch panel video.

Reference: : PS/CO/Note 79-11

NODAL name: LGREAD

Pascal call ICCI('LGREA', 'CONF1', WO STR, RW FLAG, RO BUTNB)

STR : array of chart
FLAG : integer
BUTNB : integer

Note : only available for MIP

Brief description :

Read the label of a button of the user touch-panel,

Reference: : PS/CO/Note 81-06

NODAL name: LINE

Pascal call ICCI('LINEP', 'CONF1', WO STR, RW FLAG, RO YVAL)

```
STR  : array of char
FLAG : integer
YVAL : integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Return a string to do a cursor line positioning on a video device.

Reference: : PS/CO/Note 77-16

NODAL name: LISTOB

Pascal call ICCI('LISTO', 'GRAF1', RW FLAG, WO DATA)

```
FLAG : integer
DATA : array of integer


entry :  FLAG := call_flag
```

Brief description :

Return the list of recognizable objects.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: MARKER

Pascal call ICCI('MARKE', 'GRAF1', RW FLAG, RO MAR)

```
FLAG : integer
MAR  : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Define the graphique marker sign.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: MODBRI

Pascal call ICCI('MODBR', 'CONF1', RO MODE, RW FLAG, RO SCOPE)

MODE : real
FLAG : integer

entry : FLAG := write_flag

Brief description :

Set-up the bright-up mode

Reference: : PS/CO/Note 81-23

NODAL name: MOGFAB

Pascal call ICCI('MOGFA', 'CONF1', RO MODE, RW FLAG, RO GFA)

```
MODE : real
FLAG : integer
GFA  : integer
```

```
entry :  FLAG := write_flag
```

Brief description :

Set-up the GFA bright-up mode.

Reference: : PS/CO/Note 81-31

NODAL name: MONOPL

Pascal call ICCI('MONOP', 'GRAF1', RW FLAG)

Flag : integer

entry : FLAG := call_flag

Brief description :

Monopolise all the graphic resources for the interactive channel.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: MOVE

Pascal call ICCI('MOVE', 'GRAF1', RW FLAG, RO X, RO Y)

```
FLAG : integer
X, Y : real

entry :  FLAG := call_flag
```

Brief description :

Move the beam to the defined point.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: MPXBRI

Pascal call ICCI('MPXBR', 'CONF1', RO PULSE, RW FLAG, RO SCOPE)

```
PULSE : real
FLAG  : integer
SCOPE, TYPE : integer
```

```
entry :  FLAG := write_flag
```

Brief description :

Set-up the bright-up multiplexor.

Reference: : PS/CO/Note 81-23

<u>NODAL name:</u> MPXGFA

<u>Pascal call</u> ICCI('MPXGF', 'CONF1', RO PULSE, RW FLAG, RO SCOPE, RO TYPE)

```
PULSE : real
FLAG  : integer
SCOPE, TYPE : integer
```

```
entry :  FLAG := write_flag
```

<u>Brief description</u> :

Set-up the start-gfa multiplexor

<u>Reference:</u> : PS/CO/Note 81-23

NODAL name: MPXTRG

Pascal call ICCI('MPXTR', 'CONF1, RO PULSE, RO FLAG, RO SCOPE, RO TYPE)

```
PULSE : real
FLAG  : integer
SCOPE, TYPE : integer
```

```
entry :  FLAG := write_flag
```

Brief description :

Set-up the trigger multiplexor

Reference: : PS/CO/Note 81-23

<u>NODAL name:</u> MPXVID

<u>Pascal call</u> ICCI('MPXVI', 'CONF1', RO SOURCE, RW FLAG, RO SCREEN)

| | | |
|---|---|---|
| SOURCE | : | real |
| FLAG | : | integer |
| SCREEN | : | integer |

entry : FLAG := write_flag

<u>Brief description</u> :

Setup a video multiplexor channel.

<u>Reference:</u> : PS/CO/Note 81-23

NODAL name: MWAIT

Pascal call ICCI('MWAIT', 'CONF1', WO BUTVAL, RW FLAG, WO GR)

```
BUTVAL : real
FLAG   : integer
GR     : real
```

```
entry :  FLAG := read_flag
```

Brief description :

Wait for event, and return the event identifier.

Reference: : PS/CO/Note 79-11 & PS/CO/Note 81-0

<u>NODAL name:</u> NEWS

<u>Pascal call</u> ICCI('NEWS', 'CONF1', RW FLAG, RO STAR)

FLAG : integer
STR : array of char

<u>Brief description</u> :

Print a line of the newspaper printer.

<u>Note</u>: only available on MCR computer,
can be called by REMOTE call.

<u>Reference:</u> : PS/CO/Note 81-06

NODAL name: OBJALC

Pascal call ICCI('OBJAL', 'GRAF1', RW FLAG, RO SIZE)

```
FLAG : integer
SIZE : integer


entry :  FLAG := call_flag
```

Brief description :

Allocate graphic elements to the currently opened object.

Reference: : PS/CO/Note 80-33.

NODAL name: OBJCLS

Pascal call ICCI('OBJCL', 'GRAF1', RW FLAG)

FLAG : integer

entry : FLAG := call_flag

Brief description :

Close any opened object.

Reference: : PS/CO/Note 80-33.

NODAL name: OBJECT

Pascal call ICCI('OBJEC', 'GRAF1', RW FLAG, RO OBJECT, RO TYPE, RO ADDRESS)

```
FLAG  : integer
OBJECT: integer
TYPE, ADDRESS : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Create and open a new object.

Reference: : PS/CO/Note 89-14 & PS/CO/Note 80-33.

NODAL name: OBJERS

Pascal call ICCI('OBJER', 'GRAF1', RW FLAG, RO OBJECT)

FLAG : integer
OBJECT : integer

entry : FLAG := call_flag

Brief description :

Erase and delete the object.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJEXT

Pascal call ICCI('OBJEX', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Open the object for append.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJIDF

Pascal call ICCI('OBJID', 'GRAF1', RW FLAG, RO OBJECT, RO X1, RO X2, RO Y1, RO Y2)

FLAG : integer
OBJECT : integer
X1, X2, Y1, Y2 : real

entry : FLAG := call_flag

Brief description :

Put the object into the reconnaisable object list.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJMDF

Pascal call ICCI('OBJMD', 'GRAF1', RW FLAG, RO OBJECT,
RO TICKV)

```
FLAG : integer
OBJECT : integer
TICKV  : real (1)
```

```
entry :  FLAG := call_flag
```

Brief description :

Open the object at the specified tick position for modification.

Note (1) in fact used as a triple word.

Reference: : PS/CO/Note 80-33.

NODAL name: OBJMOV

Pascal call ICCI('OBJMO', 'GRAF1', RW FLAG, RO DX, RO DY)

```
FLAG : integer
OBJECT : integer
DX, DY : real

entry :  FLAG := call_flag
```

Brief description :

Move the object.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJNID

Pascal call ICCI('OBJNI', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer

entry :  FLAG := call_flag
```

Brief description :

Remove the object from the recognizable list.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: OBJREF

Pascal call ICCI('OBJRF', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer


entry :  FLAG := call_flag
```

Brief description :

Open for redefine the object.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJRES

Pascal call ICCI('OBJRE', 'GRAF1', RW FLAG, RO OBJECT, RO DATA)

```
FLAG : integer
OBJECT : integer
DATA   : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Restore the object from the array.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJSAV

Pascal call ICCI('OBJSA', 'GRAF1', RW FLAG, RO OBJECT, WO DATA)

```
FLAG : integer
OBJECT : integer
DATA   : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Save the object into the array.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33.

NODAL name: OBJSIZ

Pascal call ICCI('OBJSI', 'GRAF1', WO SIZE, RW FLAG, RO OBJECT)

SIZE : real
FLAG : integer
OBJECT : integer

entry : FLAG := call_flag

Brief description :

Return the total space allocated to the object.

Reference: : PS/CO/Note 80-33

NODAL name: OBJSPA

Pascal call ICCI('OBJSP', 'GRAF1', WO SPACE, RW FLAG)

```
SPACE : real
FLAG  : integer


entry :  FLAG := read_FLAG
```

Brief description :

Return the free space in the object.

Reference: : PS/CO/Note 80-33

NODAL name: OBJSTA

Pascal call ICCI('OBJST', 'GRAF1', RW FLAG, RO OBJECT, RO CHANNEL, WO DATA)

```
FLAG : integer
OBJECT, CHANNEL : integer
DATA : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Read the object status.

Reference: PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: OBJWIN

Pascal call ICCI('OBJWI', 'GRAF1', RW FLAG, RO OBJECT, RO WINDOW_STATUS)

```
FLAG : integer
OBJECT : integer
WINDOW_STATUS : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Appear, disappear the reconnition window.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: ODEV

Pascal call ICCI('ODEV', 'CONF1', WO LUN, WO COCO)

```
LUN : real
COCO: integer
```

Brief description :

Return the current value of ODEV
(read only function).

Reference:

<u>NODAL name:</u> PIXIN

<u>Pascal call</u> ICCI('PIXIN', 'CONF1', RW FLAG, RO LUN,
WO

```
FLAG : integer
LUN  : integer
PICTURE : array of integer
```

```
entry :  FLAG := call_flag
```

<u>Brief description</u> :

Save a complete video image into PICTURE

<u>Reference:</u> : PS/CO/Note 79-11

NODAL name: PIXOUT

Pascal call ICCI('PIXOU', 'CONF1', RW FLAG, RO LUN, RO PICTURE)

```
FLAG : integer
LUN  : integer
PICTURE : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Send a complete image to video device

Reference: : PS/CO/Note 79-11

NODAL name: PLS

Pascal call ICCI('PLS', 'CONF1', RW FLAG, RO MACHINE, WO TELEG)

```
FLAG : integer
MACHINE : integer
TELEG : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Read the last PLS telegram for the specified machine.

Reference: : PS/CO/Note 79-11

NODAL name: POINT

Pascal call ICCI('POIN', 'GRAF1', RW FLAG, RO X, RO Y)

```
FLAG : integer
X, Y : real


entry :  FLAG := call_flag
```

Brief description :

Draw a point at the specified position.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

<u>NODAL name:</u> POSIT

<u>Pascal call</u> ICCI('POSIT', 'CONF1', WO STR, RW FLAG, RO XVAL, RO YVAL)

STR : array of char
FLAG : integer
XVAL, YVAL : integer

entry : FLAG := read_flag

<u>Brief description</u> :

Return a string to do a cursor positioning on a video device.

<u>Reference:</u> : PS/CO/Note 77-16 & PS/CO/Note 79-11

NODAL name: POST

Pascal call ICCI('POST', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer


entry :  FLAG := call_flag
```

Brief description :

Turn visible the object.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

<u>NODAL name:</u> REDCUR

<u>Pascal call</u> ICCI('REDCU', 'GRAF1', RW FLAG, WO X, WO Y)

```
FLAG : integer
X, Y : real
```

```
entry :  FLAG := call_flag
```

<u>Brief description</u> :

Read the cursor position.

<u>Reference:</u> : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: RELEAS

Pascal call ICCI('RELEA', 'CONF1', RW FLAG,
RO IODIR)

FLAG : integer
LUN, IODIR : integer

entry : FLAG := call_flag

Brief description :

Release the resource.

Note: only for non-interactive

Reference: : PS/CO/Note 79-11

NODAL name: RESERV

Pascal call ICCI('RESER', 'CONF1', RW FLAG, RO LUN,
RO IODIR)

FLAG : integer
LUN, IODIR : integer

entry : FLAG := call_flag

Brief description :

Reserve the resource without waiting.

Note : only for non-interactive

Reference: : PS/CO/Note 79-11

NODAL name: SCREEN

Pascal call ICCI('SCREE', 'GRAF1', RW FLAG, RO X1, RO X2, RO Y1, RO Y2)

FLAG : inter
X1, X2, Y1, Y2 : integer

entry : FLAG := call_flag

Brief description :

Define the screen window space.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: SDMIN

Pascal call ICCI('SDMIN', 'CONF1', RW FLAG, RO LUN, WO SYMBOLS)

```
FLAG : integer
LUN  : integer
SYMBOLS : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Save the user-defined symbols.

Reference: : PS/CO/Note 79-11

NODAL name: SDMOUT

Pascal call ICCI('SDMOU', 'CONF1', RW FLAG, RO LUN,
RO SYMBOLS)

```
FLAG : integer
LUN  : integer
SYMBOLS : array of integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Load the user-defined symbols into
a colour video device.

Reference: : PS/CO/Note 79-11

NODAL name: SYSGR

Pascal call ICCI('SYSGR', 'CONF1', WO GR, RW FLAG)

```
GR  : real
FLAG: integer


entry :  FLAG := read_flag
```

Brief description :

Return the system button group number

Reference: : PS/CO/Note 79-11

<u>NODAL name:</u> TICKV

<u>Pascal call</u> ICCI('TICKV', 'GRAF1', WO TICKV, RW FLAG)

```
TICKV : real    (1)
FLAG  : integer
```

```
entry :  FLAG := read_flag
```

<u>Brief description</u> :

Pick up a position marker for future modification.

(1)Note: in fact used as a triple word.

<u>Reference:</u> : PS/CO/Note 80-33

NODAL name: TIMODE

Pascal call ICCI('TIMOD', 'CONF1', RW FLAG, RO SCOPE, RO PLS_LINE, RO TIME_BASE)

```
FLAG : integer
SCOPE, PLS_LINE, TIME_BASE : integer

entry :  FLAG := call_flag
```

Brief description :

Set up the pulse counter mode.

Reference: : PS/CO/Note 81-23

NODAL name: TIMGR

Pascal call ICCI('TIMGR', 'CONF1', WO GR, RW FLAG)

```
GR  :  real
FLAG:  integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Return the timing buttons group number.

Reference: : PS/CO/Note 81-23

<u>NODAL name:</u> TIMTRG

<u>Pascal call</u> ICCI('TIMTR', 'CONF1', RW VAL, RW FLAG,
RO SCOPE, RO DELAY_TYPE)

```
VAL  :  real
FLAG :  integer
SCOPE, DELAY_TYPE : integer
```

```
entry :  FLAG := read_flag   for read
         FLAG := write_flag  for setting
```

<u>Brief description</u> :

Set or read a scope-trigger delay

<u>Reference:</u> : PS/CO/Note 81-23

NODAL name: TPGR

Pascal call ICCI('TPGR', 'CONF1', WO GR, RW FLAG)

```
GR  : real
FLAG: integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Return the touch-panel group number.

Reference: : PS/CO/Note 79-11

NODAL name: TRIG

Pascal call ICCI('TRIG', 'CONF1', RO VAL, RW FLAG,
RO TRGNB)

```
VAL  : real
FLAG : integer
```

```
entry :  FLAG := read_flag
```

Brief description :

Send a software trigger.

Reference: : PS/CO/Note 78-20 & PS/CO/Note 79-11

NODAL name: TVASK

Pascal call
```
ICCI('TVASK', 'CONF1', WO VAL, RW FLAG,
RO LUN, RO LINE, RO COLUMN, RO WINDOW)
```

```
VAL  : integer
FLAG : integer
LUN  : integer
LINE, COLUMN, WINDOW : integer       (1)
```

```
entry :  FLAG := read_flag
```

Brief description :

Edit and read the edited value.

Note (1): default value are flagged by a -1 value

Reference: : PS/CO/Note 79-11 & PS/CO/Note 81-05

NODAL name: TVEDIT

Pascal call ICCI('TVEDI', 'CONF1', WO STG, RW FLAG, RO LUN, RO LINE, RO COLUMN, RO WINDOW)

```
STG  : array of char
FLAG : integer
LUN  : integer
LINE; COLUMN, WINDOW : integer    (1)
```

```
entry :  FLAG := read_flag
```

Brief description :

Edit and return the content of the window.

Note (1) : défault values are flagged by a -1 value

Reference: : PS/CO/Note 79-11 & PS/CO/Note 81-05

NODAL name: TVREAD

Pascal call ICCI('TVREA', 'CONF1', WO STG, RW FLAG, RO LUN, RO LINE)

```
STG  : array of char
FLAG : integer
LUN  : integer
LINE : integer                    (1)
```

```
entry :  FLAG := read_flag
```

Brief description :

Read the content of the specified line.

Note (1) : default value is flagged by a -1 value.

Reference: : PS/CO/Note 81-06

NODAL name: UBLINK

Pascal call ICCI('UBLIN', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer
```

```
entry :  FLAG := call_flag
```

Brief description :

Reset the blinking status of the object.

Reference: : PS/CO/Note 80-33

NODAL name: UNPOST

Pascal call ICCI('UNPOS', 'GRAF1', RW FLAG, RO OBJECT)

```
FLAG : integer
OBJECT : integer


entry :  FLAG := call_flag
```

Brief description :

Turn the object invisible.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: VECTOR

Pascal call ICCI('VECTO', 'GRAF1', RW FLAG, RO X, RO Y)

```
FLAG : integer
X, Y : real


entry :  FLAG := call_flag
```

Brief description :

Draw a vector from the current beam position to the specified position.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: VIDGR

Pascal call ICCI('VIDGR', 'CONF1', WO GR, RW FLAG)

```
GR   : real
FLAG : integer


entry :  FLAG := read_flag
```

Brief description :

Return the video identification buttons group number.

Reference: : PS/CO/Note 79-11

NODAL name: WINDOW

Pascal call

```
ICCI('WINDO', 'GRAF1', RW FLAG, RO X1, RO X2,
RO Y1, RO Y2)

FLAG : integer
X1, X2, Y1, Y2 : real


entry :  FLAG := call_flag
```

Brief description :

Define the user window space.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: WRESER

Pascal call ICCI('WRESE', 'CONF1', RW FLAG, RO LUN,
RO IODIR)

```
FLAG : integer
LUN, IODIR : integer
```

```
entry :    FLAG:= call_flag
```

Brief description :

Reserve the resource with waiting.

Note : only for non-interactive

Reference: : PS/CO/Note 79-11

NODAL name: WRITE

Pascal call ICCI('VIDEO', 'CONF1', RO LUN, RO STR, WO COCO)

|  |  |
|---|---|
| LUN : | integer |
| STR : | array of char |
| COCO: | integer |

Note : the maximum size of array of char is limited to 128.
The DUMMY_CHAR must be used as the last character if previous "space" character must be sent to the device.

Brief description :

Write string to LUN.
Completion code returned into COCO.

Reference: : PS/CO/Note 77-16

NODAL name: XBAR

Pascal call ICCI('XBAR', 'GRAF1', RW FLAG, RO XARRAY, RO YARRAY; RO X=)

FLAG : integer
XARRAY, YARRAY : real

entry : FLAG := call_flag

Brief description :

Draw a X-bar graphic.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: XBAR

Pascal call ICCI('XBAR', 'GRAF1', RW FLAG, RO XARRAY, RO YARRAY; RO X=)

FLAG : integer
XARRAY, YARRAY : real

entry : FLAG := call_flag

Brief description :

Draw a X-bar graphic.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

<u>NODAL name:</u>  XGRAF

<u>Pascal call</u>  ICCI('XGRAF', 'GRAF1', RW FLAG, RO XARRAY, RO Y, RO YSTEP)

FLAG : integer
XARRAY : array of real
Y, YSTEP : real

entry :  FLAG := call_flag

<u>Brief description</u> :

Draw a X-graphic

<u>Reference:</u> :  PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: XMARK

Pascal call ICCI('XMARK', 'GRAF1', RW FLAG, RO XARRAY,
RO Y, RO YSTEP)

```
FLAG : integer
XARRAY : array of real
Y, YSTEP : real
```

```
entry :  FLAG := call_flag
```

Brief description :

Draw a X point graphic

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: XYGRAF

Pascal call ICCI('XYGRA', 'GRAF1', RW FLAG, RO XARRAY;
RO YARRAY)

```
FLAG : integer
XARRAY, YARRAY: array of real


entry :  FLAG := call_flag
```

Brief description :

Draw a broken line.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: XYMARK

Pascal call ICCI('XYMAR', 'GRAF1', RW FLAG, RO XARRAY, RO YARRAY)

FLAG : integer
XARRAY, YARRAY : array of real

entry : FLAG := call_flag

Brief description :

Draw a point plot.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: YAXIS

Pascal call

```
ICCI('YAXIS', 'GRAF1', RW FLAG, RO Y1, RO Y2,
RO X, RO YA, RO DYA, RO TICK)

FLAG : integer
Y1, Y2, X, YA, DYA : real
TICK : integer



entry :  FLAG := call_flag
```

Brief description :

Draw a Y axis on the screen.

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name:  YBAR

Pascal call  ICCI('YBAR', 'GRAF1', RW FLAG, RO XARRAY,
RO YARRAY, RO Y=)

FLAG : integer
XARRAY, YARRAY : array of real
Y= : real

entry :  FLAG := call_flag

Brief description :

Draw a Y bar-graphic.

Reference: :  PS/CO/Note 79-14 & PS/CO/Note 80-33

<u>NODAL name:</u> YGRAF

<u>Pascal call</u> ICCI('YGRAF', 'GRAF1', RW FLAG, RO YARRAY, RO X, RO XSTEO)

```
FLAG : integer
YARRAY : array of real
Y, XSTEP . real
```

```
entry :  FLAG := call_flag
```

<u>Brief description</u> :

Draw a Y-graphic

<u>Reference:</u> : PS/CO/Note 79-14 & PS/CO/Note 80-33

NODAL name: YMARK

Pascal call ICCI('YMARK', 'GRAF1', RW FLAG, RO YARRAY, RO X, RO XSTEP)

```
FLAG : integer
YARRAY : array of real
X, XSTEP : real

entry :  FLAG := call_flag
```

Brief description :

Draw a Y points-graphic

Reference: : PS/CO/Note 79-14 & PS/CO/Note 80-33

## ANNEXE 2 : Environnement console

A1

```
(* FILE NAME : (CONSOLE-SYSTEM)PAS-CON-ENVIRONT *)

(*STANDARD CONSOLE DECLARATION FOR RT-PASCAL PROGRAMS IN CONSOLES*)

(*CONSOLE TEAM, MARS 1981*)

(* UPDATE :
*)

(* SYMBOL DEFINITION FOR FLAG VALUE OF ICCI CALL *)
(* ========================================== *)

CONST
        READ_FLAG   =  1;
        WRITE_FLAG  = -1;
        CALL_FLAG   =  0;


(* ERROR TERMINAL LOGICAL UNIT NUMBER *)
(* ================================= *)

(* NOTICE : CAN BE ACCESSED FOR OUTPUT WITHOUT RESERVATION *)

CONST
        ERROR_TERMINAL = 2;


(* SYMBOLS FOR RESER WRESE RELEA ICCI CALL *)
(* ===================================== *)

(*      NON-INTERACTIVE CALL ONLY : I/O DIRECTION      *)

CONST
        INPFLG = 0;             (* INPUT DIRECTION  *)
        OUTFLG = 1;             (* OUTPUT DIRECTION *)


(* TARGET SOFTWARE-TRIGGER IDENTIFICATION FOR TRIG ICCI CALL *)
(* ======================================================== *)

CONST
        TRGTIP = 1;             (* TRIGGER TO TIP *)
        TRGMIP = 2;             (* TRIGGER TO MIP *)
        TRGVIP = 3;             (* TRIGGER TO VIP *)
        TRGSIP = 4;             (* TRIGGER TO SIP *)
        TRGLIP = 5;             (* TRIGGER TO LIP *)


(* GROUP NUMBER FOR EVENT IDENTIFICATION *)
(* ==================================== *)

(*      FOR MWAIT AND BUTS ICCI CALL       *)

CONST
        TRIGER = 255;           (* SOFTWARE TRIGGER GROUP *)
        EVGRUP = 254;          (* PROCESS-EVENT GROUP     *)


  (* VIDEO SCREEN CONTROL CHARACTERS DEFINITION *)

  (*    TO BE USED LIKE : CHAR(SYMBOL-NAME)        *)

  CONST
        RED    = 0;            (* RED COLOUR *)
        GREEN  = 1;            (* GREEN COLOUR *)
        BLUE   = 2;            (* DARK BLUE COLOUR *)
        WHITE  = 3;            (* WHITE COLOUR *)
        LBLUE  = 4;            (* LIGHT BLUE COLOUR *)
        MAGENT = 5;            (* MAGENTA COLOUR *)
        YELLOW = 6;            (* YELLOW COLOUR *)
        BLACK  = 7;            (* BLACK COLOUR *)

        BACKGR = &20;          (* BACKGROUND COLOUR INTO NEXT CHARACTER *)
        INVERT = &17;          (* SWAP BACKGROUND AND FOREGROUND COLOUR *)
        ERASE  = &14;          (* ERASE AND RESET SCREEN *)
        DUMMY_CHAR = &33;      (* DUMMY CAHARACTER FOR END OF SPACE CHARACTERS STRING *)
```

## ANNEXE 3 : Exemple

```
@CC    (  START OF MODE-HANOI-SW
@CC MODE FILE TO LOAD HANOI-SWITCH PROGRAM
@CC
@CC FILE NAME : (F:C-NEW)MODE-HANOI-SW:SYMB
@CC FABIEN PERRIOLLAT 19-FEB-1981
@CC

@RT-L
SET-SEG-FIL 1

PART-CLEAR-RTFIL 77
CL-SEG 77
NEW-SEG 77 2 DM,,
FIX-SYMB 173
SET-LOA-ADD 77 40000
LOAD (F:CON-NEW)HANOI:BRF,,,
LOAD PASCAL-LIBRARY,,,
LOAD DEBUG-ERRO,,,
SEARCH 201,,
SEARCH 37,,
WR-REF TER
WHAT-IS 1ICCI

DELETE-NON-REENTRANT
UNFIX 173
UNFIX 201
UNFIX 37
END-LOAD
EX

@CC    END OF MODE-HANOI-SW )
```

A3-2

```
#################################################################################################
#
#              P A S C A L    P4-SYSTEM  CERN V79.1    DATE: 26/MAR/81   TIME: 10:44:16
#
#################################################################################################

INC IL LINE LV VARLC

           1  0     20 (*$RT HANOI 10;M=4K*)
           2  0     20 PROGRAM TOWERS_OF_HANOI;
           3  0     20
           4  0     20 (*
           5  0     20     1ST VERSION : ROBERT CAILLIAU (NON INTERACTIVE VERSION)
           6  0     20     2ND VERSION : FABIEN PERRIOLLAT (INTERACTIVE VERSION)
           7  0     20      LAST EDIT  : 25-MAR-1981
           8  0     20      FILE NAME  : PAS-HANOI:SYMB
           9  0     20 *)
          10  0     20
          11  0     20
          12  0     20 CONST RT_STSZ = 15;
          13  0     20          F_STSZ=59;
          14  0     20
          15  0     20 TYPE PIPETYPE=INTEGER;
          16  0     20      BUFFTYPE=INTEGER;
          17  0     20
          18  0     20 (*$I (RT-PASCAL)PAS-RT-ENVIRONT;L-
 *)
   20 0  174  0     20 (*$I (CONSOLE-SYS)PAS-CON-ENVIRONT*)

    1 1  175  0     20 (* FILE NAME : (CONSOLE-SYSTEM)PAS-CON-ENVIRONT *)
    2 1  176  0     20
    3 1  177  0     20 (*STANDARD CONSOLE DECLARATION FOR RT-PASCAL PROGRAMS IN CONSOLES*)
    4 1  178  0     20
    5 1  179  0     20 (*CONSOLE TEAM, MARS 1981*)
    6 1  180  0     20
    7 1  181  0     20 (* UPDATE :
    8 1  182  0     20 *)
    9 1  183  0     20
   10 1  184  0     20
   11 1  185  0     20 (* SYMBOL DEFINITION FOR FLAG VALUE OF ICCI CALL *)
   12 1  186  0     20 (* ============================================ *)
   13 1  187  0     20
   14 1  188  0     20 CONST
   15 1  189  0     20          READ_FLAG   =  1;
   16 1  190  0     20          WRITE_FLAG  = -1;
   17 1  191  0     20          CALL_FLAG   =  0;
   18 1  192  0     20
   19 1  193  0     20
   20 1  194  0     20
   21 1  195  0     20 (* ERROR TERMINAL LOGICAL UNIT NUMBER *)
   22 1  196  0     20 (* ================================= *)
   23 1  197  0     20
   24 1  198  0     20 (* NOTICE : CAN BE ACCESSED FOR OUTPUT WITHOUT RESERVATION *)
   25 1  199  0     20
   26 1  200  0     20 CONST
   27 1  201  0     20          ERROR_TERMINAL = 2;
   28 1  202  0     20
   29 1  203  0     20
   30 1  204  0     20
   31 1  205  0     20 (* SYMBOLS FOR RESER WRESE RELEA ICCI CALL *)
 32 1  206  0    20 (* ==================================== *)
 33 1  207  0    20
 34 1  208  0    20 (*     NON-INTERACTIVE CALL ONLY : I/O DIRECTION     *)
 35 1  209  0    20
 36 1  210  0    20 CONST
 37 1  211  0    20       INPFLG = 0;              (* INPUT DIRECTION  *)
 38 1  212  0    20       OUTFLG = 1;              (* OUTPUT DIRECTION *)
 39 1  213  0    20
 40 1  214  0    20
 41 1  215  0    20
 42 1  216  0    20 (* TARGET SOFTWARE-TRIGGER IDENTIFICATION FOR TRIG ICCI CALL *)
 43 1  217  0    20 (* ======================================================== *)
 44 1  218  0    20
 45 1  219  0    20 CONST
 46 1  220  0    20       TRGTIP = 1;              (* TRIGGER TO TIP *)
 47 1  221  0    20       TRGMIP = 2;              (* TRIGGER TO MIP *)
 48 1  222  0    20       TRGVIP = 3;              (* TRIGGER TO VIP *)
 49 1  223  0    20       TRGSIP = 4;              (* TRIGGER TO SIP *)
 50 1  224  0    20       TRGLIP = 5;              (* TRIGGER TO LIP *)
 51 1  225  0    20
 52 1  226  0    20
 53 1  227  0    20
 54 1  228  0    20 (* GROUP NUMBER FOR EVENT IDENTIFICATION *)
 55 1  229  0    20 (* ==================================== *)
 56 1  230  0    20
 57 1  231  0    20 (*     FOR MWAIT AND BUTS ICCI CALL      *)
 58 1  232  0    20
 59 1  233  0    20 CONST
 60 1  234  0    20       TRIGER = 255;            (* SOFTWARE TRIGGER GROUP *)
 61 1  235  0    20       EVGRUP = 254;            (* PROCESS-EVENT GROUP    *)
 62 1  236  0    20
 63 1  237  0    20
 64 1  238  0    20
 65 1  239  0    20 (* VIDEO SCREEN CONTROL CHARACTERS DEFINITION *)
 66 1  240  0    20
 67 1  241  0    20 (*     TO BE USED LIKE : CHAR(SYMBOL-NAME)        *)
 68 1  242  0    20
 69 1  243  0    20 CONST
 70 1  244  0    20       RED    = 0;              (* RED COLOUR *)
 71 1  245  0    20       GREEN  = 1;              (* GREEN COLOUR *)
 72 1  246  0    20       BLUE   = 2;              (* DARK BLUE COLOUR *)
 73 1  247  0    20       WHITE  = 3;              (* WHITE COLOUR *)
 74 1  248  0    20       LBLUE  = 4;              (* LIGHT BLUE COLOUR *)
 75 1  249  0    20       MAGENT = 5;              (* MAGENTA COLOUR *)
 76 1  250  0    20       YELLOW = 6;              (* YELLOW COLOUR *)
 77 1  251  0    20       BLACK  = 7;              (* BLACK COLOUR *)
 78 1  252  0    20
 79 1  253  0    20       BACKGR = &20;            (* BACKGROUND COLOUR INTO NEXT CHARACTER *)
 80 1  254  0    20       INVERT = &17;            (* SWAP BACKGROUND AND FOREGROUND COLOUR *)
 81 1  255  0    20       ERASE  = &14;            (* ERASE AND RESET SCREEN *)
 82 1  256  0    20       DUMMY_CHAR = &33;        (* DUMMY CAHARACTER FOR END OF SPACE CHARACTE

 21 0  257  0    20
```

```
22 0   258   0     20
23 0   259   0     20 (* DEFINITIONS FOR PROGRAM AND MODULE OF TOWERS OF HANOI *)
24 0   260   0     20
25 0   261   0     20 CONST
26 0   262   0     20    MAXDISC = 10;  NODISC = 0;
27 0   263   0     20    EMPTY = 0; ONE = 1; FULL = 10;
28 0   264   0     20
29 0   265   0     20 TYPE
30 0   266   0     20    PILE = (LEFT, MIDDLE, RIGHT);
31 0   267   0     20    DISC = 0 .. MAXDISC;
32 0   268   0     20    PILESIZE = EMPTY..FULL;
33 0   269   0     20
34 0   270   0     20 CONST
35 0   271   0     20     TRIGGER_GROUP = TRIGER;            (* SOFTWARE TRIGGER GROUP NUMBER *)
36 0   272   0     20     MIN_PILE = 1;                      (* KNOB VALUE FOR FIRST(PILE) *)
37 0   273   0     20
38 0   274   0     20 TYPE
39 0   275   0     20     KNOB_INDEX = (DUMMY, K1, K2, K3, K4);
40 0   276   0     20     KNOB_MODE = RECORD
41 0   277   0     20        VAL : REAL;
42 0   278   0     20        KFLAG : INTEGER;
43 0   279   0     20        KNB : KNOB_INDEX;
44 0   280   0     20        MODE : INTEGER;
45 0   281   0     20        VMAX : REAL;
46 0   282   0     20        VMIN : REAL;
47 0   283   0     20        BBW : INTEGER;
48 0   284   0     20        EDPT : INTEGER;
49 0   285   0     20        UNST : ARRAY [0..3] OF CHAR;
50 0   286   0     20        TIST : ARRAY [0..15] OF CHAR;
51 0   287   0     20     END;
52 0   288   0     20
53 0   289   0     20
54 0   290   0     20 VAR MIP_TERMINAL :INTEGER;
55 0   291   0     21     ADISC: DISC;
56 0   292   0     22     SOURCE_PEG, DESTINATION_PEG : PILE;
57 0   293   0     24     HIGHTH : PILESIZE;
58 0   294   0     25     TP_GROUP, KNOB_GROUP : INTEGER;
59 0   295   0     27     COLOUR : INTEGER;
60 0   296   0     28     FLAG : INTEGER;
61 0   297   0     29     TEMP_REAL : REAL;
62 0   298   0     32     TXT : RT_STRING;
63 0   299   0     48     KNOB : KNOB_INDEX;
64 0   300   0     49     KNOB_INIT : KNOB_MODE;
65 0   301   0     83     LINE : INTEGER;
66 0   302   0     84     TWO_ERROR : BOOLEAN;
67 0   303   0     85     GROUP, BUTTON : INTEGER;
68 0   304   0     87     DELAY : REAL;
69 0   305   0     90     MAX_PILE, DELTA_PILE : INTEGER;
70 0   306   0     92     ALL_PILE : INTEGER;
71 0   307   0     93
```

```
 72 0  308  0    93 (* INITIALIZATION ROUTINES *)
 73 0  309  0    93
 74 0  310  0    93 PROCEDURE STOP_HANOI;
 75 0  311  1     9          FORWARD;
 76 0  312  0    93
 77 0  313  0    93
 78 0  314  0    93 PROCEDURE ERASE_TP;
 79 0  315  1     9
 80 0  316  1     9 CONST TITLE_INDEX = 0;
 81 0  317  1     9 TYPE NUL_STRING = ARRAY[0..0] OF CHAR;
 82 0  318  1     9
 83 0  319  1     9 VAR
 84 0  320  1     9      NUL_STG : NUL_STRING;
 85 0  321  1    10      TITLE_IX : INTEGER;
 86 0  322  1    11
 87 0  323  1    11 BEGIN
 88 0  324  1       NUL_STG := '';
 89 0  325  1       TITLE_IX := TITLE_INDEX;
 90 0  326  1       FLAG := WRITE_FLAG;
 91 0  327  1       ICCI('LEGEN','CONF1',RO NUL_STG,RW FLAG,RO TITLE_IX); LINE := LNR;
 92 0  328  1       IF (FLAG <> 0) THEN STOP_HANOI;
 93 0  329  1       END;
 94 0  330  0    93
 95 0  331  0    93
 96 0  332  0    93 PROCEDURE CLEAR_KNOB(KNOB_NUMBER : KNOB_INDEX);
 97 0  333  1    10
 98 0  334  1    10 VAR KN : INTEGER;
 99 0  335  1    11
100 0  336  1    11 BEGIN
101 0  337  1       KN := ORD(KNOB_NUMBER);
102 0  338  1       FLAG := CALL_FLAG;
103 0  339  1       ICCI('KPURG','CONF1',RW FLAG,RO KN); LINE := LNR;
104 0  340  1       IF (FLAG <> 0) THEN STOP_HANOI;
105 0  341  1       END;
106 0  342  0    93
107 0  343  0    93
108 0  344  0    93 PROCEDURE INITIALIZE;
109 0  345  1     9
110 0  346  1     9 VAR T_PILE : PILE;
111 0  347  1    10          VAR PLSAR : ARRAY [1..16] OF INTEGER;
112 0  348  1    26              MACHINE : INTEGER;
113 0  349  1    27
114 0  350  1    27 BEGIN
115 0  351  1       RTTERMINAL(ERROR_TERMINAL);
116 0  352  1       MIP_TERMINAL := ERROR_TERMINAL;
117 0  353  1       COLOUR := ERROR_TERMINAL;
118 0  354  1       MACHINE := 1;
119 0  355  1       FLAG := READ_FLAG;
120 0  356  1       ICCI('PLS','CONF1',RW FLAG, RO MACHINE, WO PLSAR); LINE := LNR;
121 0  357  1       IF (FLAG <> 0) THEN STOP_HANOI;
122 0  358  1       FLAG := READ_FLAG;
123 0  359  1       ICCI('ODEV','CONF1',WO TEMP_REAL,RW FLAG); LINE := LNR;
124 0  360  1       IF (FLAG <> 0) THEN STOP_HANOI;
125 0  361  1       MIP_TERMINAL := TRUNC(TEMP_REAL);
126 0  362  1       RTTERMINAL(MIP_TERMINAL);
127 0  363  1       TWO_ERROR := FALSE;
128 0  364  1       LINE := LNR;
129 0  365  1       ERASE_TP;
130 0  366  1       FOR KNOB := K1 TO K4 DO CLEAR_KNOB(KNOB);
```

```
131 0  367  1
132 0  368  1        FLAG := READ_FLAG;
133 0  369  1        ICCI('COLOU','CONF1',WO TEMP_REAL,RW FLAG); LINE := LNR;
134 0  370  1        IF (FLAG <> 0) THEN STOP_HANOI;
135 0  371  1        COLOUR := TRUNC(TEMP_REAL);
136 0  372  1        FLAG := READ_FLAG;
137 0  373  1        ICCI('TPGR','CONF1',WO TEMP_REAL,RW FLAG); LINE := LNR;
138 0  374  1        IF (FLAG <> 0) THEN STOP_HANOI;
139 0  375  1        TP_GROUP := TRUNC(TEMP_REAL);
140 0  376  1        FLAG := READ_FLAG;
141 0  377  1        ICCI('KNVGR','CONF1',WO TEMP_REAL,RW FLAG); LINE := LNR;
142 0  378  1        IF (FLAG <> 0) THEN STOP_HANOI;
143 0  379  1        KNOB_GROUP := TRUNC(TEMP_REAL);
144 0  380  1
145 0  381  1        DELTA_PILE := MIN_PILE - ORD(FIRST(PILE));
146 0  382  1        MAX_PILE := ORD(LAST(PILE)) + DELTA_PILE;
147 0  383  1        WITH KNOB_INIT DO
148 0  384  1        BEGIN;
149 0  385  2            MODE := 2;
150 0  386  2            BBW := -1;
151 0  387  2            EDPT := 0;
152 0  388  2          END;
153 0  389  1
154 0  390  1        ALL_PILE := 0;
155 0  391  1        FOR T_PILE := FIRST(PILE) TO LAST(PILE) DO
156 0  392  1            ALL_PILE := ALL_PILE + ORD(T_PILE);
157 0  393  1
158 0  394  1        END;
159 0  395  0    93
```

```
160 0  396  0    93 (* PROCEDURE TO WRITE TO COLOUR SCREEN *)
161 0  397  0    93
162 0  398  0    93 CONST BASE=16;
163 0  399  0    93
164 0  400  0    93 VAR
165 0  401  0    93    DISCIMAGE: ARRAY[1..MAXDISC] OF STRING;
166 0  402  0   703    PEGIMAGE: ARRAY[0..MAXDISC] OF STRING;
167 0  403  0  1374
168 0  404  0  1374
169 0  405  0  1374 VAR
170 0  406  0  1374    BOARD: ARRAY [PILE] OF RECORD
171 0  407  0  1374                           DISCS:   ARRAY [ONE..FULL] OF DISC;
172 0  408  0  1374                           TOPONE:  PILESIZE
173 0  409  0  1374                           END;
174 0  410  0  1407
175 0  411  0  1407
176 0  412  0  1407 PROCEDURE WRITESTRING(VAR F: TEXT; S: STRING; W: INTEGER); EXTERNAL '0WRST
177 0  413  0  1407
178 0  414  0  1407
179 0  415  0  1407 PROCEDURE POSIT(LINE,COLUMN: INTEGER);
180 0  416  1    11 BEGIN
181 0  417  1       WRITE(SCREEN, CHAR(9), CHAR(COLUMN), CHAR(11), CHAR(LINE))
182 0  418  1       END;
183 0  419  0  1407
184 0  420  0  1407 FUNCTION FLASH  : CHAR;   BEGIN FLASH  :=CHAR(19) END;
185 0  421  0  1407 FUNCTION NOFLASH: CHAR;   BEGIN NOFLASH:=CHAR(18) END;
186 0  422  0  1407 FUNCTION ALTCH  : CHAR;   BEGIN ALTCH  :=CHAR(17) END;
187 0  423  0  1407
188 0  424  0  1407
189 0  425  0  1407
190 0  426  0  1407 PROCEDURE CLEARBOARD;
191 0  427  1     9
192 0  428  1     9 VAR P: PILE;  LDISC: DISC;  J,K,L,R: INTEGER;
193 0  429  1    15 VAR BLANK_LINE : ARRAY [1..65] OF CHAR;
194 0  430  1    80
195 0  431  1    80 BEGIN
196 0  432  1       FOR P:=LEFT TO RIGHT DO BOARD[P].TOPONE:=EMPTY;
197 0  433  1       FOR LDISC:=1 TO MAXDISC DO BEGIN
198 0  434  2          WITH DISCIMAGE[LDISC] DO BEGIN
199 0  435  3             C[0]:=CHAR(RED); C[1]:=CHAR(INVERT);
200 0  436  3             FOR J:=2 TO 1+2*LDISC DO C[J]:=' ';
201 0  437  3             L:=2*LDISC+2
202 0  438  3             END;
203 0  439  2          WITH PEGIMAGE[LDISC] DO BEGIN
204 0  440  3             C[0]:=CHAR(BACKGR); C[1]:=CHAR(LBLUE);
205 0  441  3             FOR J:=2 TO LDISC DO C[J]:=' ';
206 0  442  3             C[LDISC+1]:=CHAR(BLUE); C[LDISC+2]:=CHAR(BACKGR); C[LDISC+3]:=CHAR(L
207 0  443  3             C[LDISC+4]:=ALTCH; C[LDISC+5]:=CHAR(128);
208 0  444  3             C[LDISC+6]:=ALTCH; C[LDISC+7]:=CHAR(129);
209 0  445  3             FOR J:=LDISC+8 TO LDISC*2+6 DO C[J]:=' ';
210 0  446  3             L:=LDISC*2+7
211 0  447  3             END
212 0  448  2          END;
213 0  449  1       WITH PEGIMAGE[0] DO BEGIN
214 0  450  2          C[0]:=CHAR(BLUE); C[1]:=CHAR(BACKGR); C[2]:=CHAR(LBLUE);
215 0  451  2          C[3]:=ALTCH; C[4]:=CHAR(130); C[5]:=ALTCH; C[6]:=CHAR(131);
216 0  452  2          L:=7
217 0  453  2          END;
218 0  454  1       FOR J :=1 TO 64 DO BLANK_LINE[J] := ' ';
219 0  455  1        BLANK_LINE[65] :=CHAR(DUMMY_CHAR);
220 0  456  1        RTTERMINAL(COLOUR);
221 0  457  1        WRITE(SCREEN, CHAR(ERASE), CHAR(BLACK), CHAR(BACKGR), CHAR(LBLUE));
222 0  458  1        FOR J:=0 TO 23 DO BEGIN;
223 0  459  2             ICCI('VIDEO','CONF1',RO COLOUR,RO BLANK_LINE,WO FLAG); LINE := LNR;
224 0  460  2             IF (FLAG <> 0) THEN STOP_HANOI;
225 0  461  2        END;
226 0  462  1        POSIT(3,22); WRITE(SCREEN,'-=* TOWERS OF HA','NOI *=-');
227 0  463  1        POSIT(BASE,1);
228 0  464  1        WRITE(SCREEN, CHAR(YELLOW), CHAR(INVERT));
229 0  465  1        FOR J:=BASE TO BASE+1 DO FOR K:=0 TO 63 DO WRITE(SCREEN,' ');
230 0  466  1        WRITE(SCREEN, CHAR(INVERT));
231 0  467  1        POSIT(BASE-12,11); WRITESTRING(SCREEN, PEGIMAGE[0], 1);
232 0  468  1        POSIT(BASE-12,32); WRITESTRING(SCREEN, PEGIMAGE[0], 1);
233 0  469  1        POSIT(BASE-12,53); WRITESTRING(SCREEN, PEGIMAGE[0], 1);
234 0  470  1        FOR J:=BASE-11 TO BASE-1 DO BEGIN
235 0  471  2           POSIT(J,11); WRITESTRING(SCREEN, PEGIMAGE[1], 1);
236 0  472  2           POSIT(J,32); WRITESTRING(SCREEN, PEGIMAGE[1], 1);
237 0  473  2           POSIT(J,53); WRITESTRING(SCREEN, PEGIMAGE[1], 1)
238 0  474  2           END;
239 0  475  1        POSIT(23,13); WRITE(SCREEN,CHAR(BLACK),CHAR(BACKGR),CHAR(LBLUE),
240 0  476  1                          'T()3 )3 ! PASCAL',' ).4%2!#4)6% MIP',' 02/'2!-');
241 0  477  1        RTTERMINAL(MIP_TERMINAL);
242 0  478  1        END;
243 0  479  0  1407
244 0  480  0  1407
245 0  481  0  1407 PROCEDURE PLACE(WHICH: DISC; WHERE: PILE);
246 0  482  1    11 VAR J: INTEGER;
247 0  483  1    12
248 0  484  1    12 BEGIN
249 0  485  1        RTTERMINAL(COLOUR);
250 0  486  1        WITH BOARD[WHERE] DO BEGIN
251 0  487  2           TOPONE:=SUCC(TOPONE);
252 0  488  2           DISCS[TOPONE]:=WHICH;
253 0  489  2           CASE WHERE OF
254 0  490  2           LEFT: J:=12-DISCS[TOPONE];
255 0  491  2           MIDDLE: J:=33-DISCS[TOPONE];
256 0  492  2           RIGHT: J:=54-DISCS[TOPONE]
257 0  493  2              END;
258 0  494  2           WITH BOARD[WHERE] DO BEGIN
259 0  495  3              POSIT(BASE-TOPONE,J); WRITESTRING(SCREEN, DISCIMAGE[DISCS[TOPONE]],
260 0  496  3              END
261 0  497  2           END;
262 0  498  1        RTTERMINAL(MIP_TERMINAL);
263 0  499  1        END;
264 0  500  0  1407
265 0  501  0  1407 PROCEDURE TAKE(VAR WHICH: DISC; WHERE: PILE);
266 0  502  1    11 VAR J: INTEGER;
267 0  503  1    12 BEGIN
268 0  504  1        RTTERMINAL(COLOUR);
269 0  505  1        WITH BOARD[WHERE] DO BEGIN
270 0  506  2           WHICH:=DISCS[TOPONE];
271 0  507  2           CASE WHERE OF
272 0  508  2           LEFT: J:=12-DISCS[TOPONE];
273 0  509  2           MIDDLE: J:=33-DISCS[TOPONE];
274 0  510  2           RIGHT: J:=54-DISCS[TOPONE]
275 0  511  2           END;
276 0  512  2           WITH BOARD[WHERE] DO BEGIN
277 0  513  3              POSIT(BASE-TOPONE,J); WRITESTRING(SCREEN, PEGIMAGE[DISCS[TOPONE]],
278 0  514  3              END;
```

```
 279 0   515  2              TOPONE:=PRED(TOPONE)
 280 0   516  2             END;
 281 0   517  1          RTTERMINAL(MIP_TERMINAL);
 282 0   518  1          END;
 283 0   519  0   1407
 284 0   520  0   1407
 285 0   521  0   1407 PROCEDURE PEG_NAME (PEG : PILE);
 286 0   522  1     10
 287 0   523  1     10 BEGIN
 288 0   524  1          CASE PEG OF
 289 0   525  1              LEFT     : WRITE(SCREEN,'LEFT');
 290 0   526  1              MIDDLE   : WRITE(SCREEN,'MIDDLE');
 291 0   527  1              RIGHT    : WRITE(SCREEN,'RIGHT');
 292 0   528  1            END;
 293 0   529  1          END;
 294 0   530  0   1407
```

```
295 0  531  0  1407 (* OPERATOR EVENTS ROUTINES *)
296 0  532  0  1407
297 0  533  0  1407
298 0  534  0  1407 PROCEDURE WAIT_EVENT ( VAR GROUP_NB, BUTTON_NB : INTEGER);
299 0  535  1    11
300 0  536  1    11 VAR BUT_VAL : REAL;
301 0  537  1    14
302 0  538  1    14 BEGIN
303 0  539  1       FLAG := READ_FLAG;
304 0  540  1       ICCI('MWAIT','CONF1',WO BUT_VAL,RW FLAG,WO TEMP_REAL); LINE := LNR;
305 0  541  1       IF (FLAG <> 0) THEN STOP_HANOI;
306 0  542  1       GROUP_NB := TRUNC(TEMP_REAL); BUTTON_NB := TRUNC(BUT_VAL);
307 0  543  1       END;
308 0  544  0  1407
309 0  545  0  1407
310 0  546  0  1407 FUNCTION WAIT_KNOB : KNOB_INDEX;
311 0  547  1     9
312 0  548  1     9 VAR GROUP, BUTTON : INTEGER;
313 0  549  1    11
314 0  550  1    11 BEGIN
315 0  551  1       REPEAT
316 0  552  2           BEGIN
317 0  553  3           WAIT_EVENT(GROUP,BUTTON);
318 0  554  3           IF (GROUP = TRIGGER_GROUP) OR (GROUP = TP_GROUP) THEN STOP_HANOI;
319 0  555  3           END;
320 0  556  2       UNTIL GROUP = KNOB_GROUP;
321 0  557  1       WAIT_KNOB := KNOB_INDEX(BUTTON);
322 0  558  1       END;
323 0  559  0  1407
324 0  560  0  1407
325 0  561  0  1407 FUNCTION TEST_GROUP (GROUP_NB : INTEGER) : BOOLEAN;
326 0  562  1    10
327 0  563  1    10 VAR TEMP : REAL;
328 0  564  1    13
329 0  565  1    13 BEGIN
330 0  566  1       FLAG := READ_FLAG;
331 0  567  1       ICCI('BUTS','CONF1',WO TEMP,RW FLAG,RO GROUP_NB); LINE := LNR;
332 0  568  1       IF (FLAG <> 0) THEN STOP_HANOI;
333 0  569  1       IF (TRUNC(TEMP) <> 0) THEN
334 0  570  1           TEST_GROUP := TRUE
335 0  571  1         ELSE
336 0  572  1           TEST_GROUP := FALSE;
337 0  573  1       END;
338 0  574  0  1407
```

```
339 0  575  0  1407 (* KNOB ROUTINES *)
340 0  576  0  1407
341 0  577  0  1407
342 0  578  0  1407 FUNCTION READ_KNOB (KNB : KNOB_INDEX) : REAL;
343 0  579  1    10
344 0  580  1    10 VAR TEMP : REAL;
345 0  581  1    13     KN : INTEGER;
346 0  582  1    14
347 0  583  1    14 BEGIN
348 0  584  1       KN := ORD(KNB);
349 0  585  1       FLAG := READ_FLAG;
350 0  586  1       ICCI('KNOB','CONF1',WO TEMP,RW FLAG,RO KN); LINE := LNR;
351 0  587  1       IF (FLAG <> 0) THEN STOP_HANOI;
352 0  588  1       READ_KNOB := TEMP;
353 0  589  1       END;
354 0  590  0  1407
355 0  591  0  1407
356 0  592  0  1407 PROCEDURE NEW_KNOB (NKNB : KNOB_MODE);
357 0  593  1    44
358 0  594  1    44 VAR KN : INTEGER;
359 0  595  1    45
360 0  596  1    45 BEGIN
361 0  597  1       WITH NKNB DO
362 0  598  1          BEGIN;
363 0  599  2           KN := ORD(KNB);
364 0  600  2           KFLAG := WRITE_FLAG;
365 0  601  2           LINE := LNR; ICCI('KINIT','CONF1',RO VAL,RW KFLAG,RO KN,RO MODE,
366 0  602  2                    RO VMAX,RO VMIN,RO BBW,RO EDPT,RO UNST,RO TIST);
367 0  603  2           FLAG := KFLAG;
368 0  604  2           IF (FLAG <> 0) THEN STOP_HANOI;
369 0  605  2          END;
370 0  606  1       END;
371 0  607  0  1407
372 0  608  0  1407
373 0  609  0  1407 PROCEDURE INIT_KNOBS;
374 0  610  1     9
375 0  611  1     9 BEGIN;
376 0  612  1       WITH KNOB_INIT DO
377 0  613  1       BEGIN;
378 0  614  2           KNB := K1;
379 0  615  2           VMAX := MAX_PILE;
380 0  616  2           VMIN := MIN_PILE;
381 0  617  2           VAL := VMIN;
382 0  618  2           UNST := 'PEGS';
383 0  619  2           TIST := 'START-PEG';
384 0  620  2       END;
385 0  621  1       NEW_KNOB(KNOB_INIT);
386 0  622  1
387 0  623  1       WITH KNOB_INIT DO
388 0  624  1       BEGIN;
389 0  625  2           KNB := K2;
390 0  626  2           VAL := VMAX;
391 0  627  2           TIST := 'DEST-PEG';
392 0  628  2       END;
393 0  629  1       NEW_KNOB(KNOB_INIT);
394 0  630  1
395 0  631  1       WITH KNOB_INIT DO
396 0  632  1       BEGIN;
397 0  633  2           KNB := K3;
```

```
 398 0   634   2           VMAX := MAXDISC;
 399 0   635   2           VMIN := ONE;
 400 0   636   2           VAL  := VMAX;
 401 0   637   2           UNST := 'DISC';
 402 0   638   2           TIST := 'NUMBER OF DISCS';
 403 0   639   2       END;
 404 0   640   1       NEW_KNOB(KNOB_INIT);
 405 0   641   1       END;
 406 0   642   0   1407
```

```
407 0  643  0  1407 (* INTERACTION TO DEFINE RUNNING CONDITIONS *)
408 0  644  0  1407
409 0  645  0  1407 PROCEDURE READ_CONDITION;
410 0  646  1     9
411 0  647  1     9 VAR
412 0  648  1     9      BUTTON_NB, MX : INTEGER;
413 0  649  1    11      ACTIVE_KNOB : SET OF KNOB_INDEX;
414 0  650  1    19      USED_PEG : SET OF PILE;
415 0  651  1    27
416 0  652  1    27 BEGIN
417 0  653  1
418 0  654  1         BUTTON_NB := 7;
419 0  655  1         TXT := '\STOP';
420 0  656  1         FLAG := WRITE_FLAG;
421 0  657  1         ICCI('LEGEN','CONF1',RO TXT, RW_FLAG, RO BUTTON_NB); LINE := LNR;
422 0  658  1         IF (FLAG <> 0) THEN STOP_HANOI;
423 0  659  1
424 0  660  1         INIT_KNOBS;
425 0  661  1
426 0  662  1         USED_PEG := [];
427 0  663  1         ACTIVE_KNOB := [K1,K2,K3];
428 0  664  1         RTTERMINAL(COLOUR); WRITE(SCREEN,CHAR(RED),CHAR(BACKGR),CHAR(LBLUE));
429 0  665  1         REPEAT
430 0  666  2              KNOB := WAIT_KNOB;
431 0  667  2              IF (KNOB IN ACTIVE_KNOB) THEN
432 0  668  2                CASE KNOB OF
433 0  669  2                    K1 :
434 0  670  2                      BEGIN;
435 0  671  3                      SOURCE_PEG := PILE(ROUND(READ_KNOB(KNOB)) - DELTA_PILE);
436 0  672  3                      IF NOT(SOURCE_PEG IN USED_PEG) THEN BEGIN
437 0  673  4                          CLEAR_KNOB(KNOB);
438 0  674  4                          ACTIVE_KNOB := ACTIVE_KNOB - [KNOB];
439 0  675  4                          USED_PEG := USED_PEG + [SOURCE_PEG];
440 0  676  4                          POSIT(19,8); WRITE(SCREEN,'STARTING PEG    ',': ');
441 0  677  4                          PEG_NAME(SOURCE_PEG);
442 0  678  4                          END;
443 0  679  3                      END;
444 0  680  2                    K2 :
445 0  681  2                      BEGIN;
446 0  682  3                      DESTINATION_PEG := PILE(ROUND(READ_KNOB(KNOB)) - DELTA_PILE);
447 0  683  3                      IF NOT(DESTINATION_PEG IN USED_PEG) THEN BEGIN
448 0  684  4                          CLEAR_KNOB(KNOB);
449 0  685  4                          ACTIVE_KNOB := ACTIVE_KNOB - [KNOB];
450 0  686  4                          USED_PEG := USED_PEG + [DESTINATION_PEG];
451 0  687  4                          POSIT(20,8); WRITE(SCREEN,'DESTINATION PEG ',': ');
452 0  688  4                          PEG_NAME(DESTINATION_PEG);
453 0  689  4                          END;
454 0  690  3                      END;
455 0  691  2                    K3 :
456 0  692  2                      BEGIN;
457 0  693  3                      HIGHTH := ROUND(READ_KNOB(KNOB));
458 0  694  3                      CLEAR_KNOB(KNOB);
459 0  695  3                      ACTIVE_KNOB := ACTIVE_KNOB - [KNOB];
460 0  696  3                      POSIT(21,8); WRITE(SCREEN,'NUMBER OF DISCS ',': ',HIGHTH:1);
461 0  697  3                      END;
462 0  698  2                    OTHERWISE BEGIN END;
463 0  699  2                  END;
464 0  700  2         UNTIL ACTIVE_KNOB = [];
465 0  701  1         RTTERMINAL(MIP_TERMINAL);
```

```
0   702  1
0   703  1         KNOB := K4;                              (* SPEED CONTROL KNOB *)
0   704  1         WITH KNOB_INIT DO BEGIN
0   705  2             KNB := KNOB;
0   706  2             MODE := 0;
0   707  2             VMAX := 5.0;
0   708  2             VMIN := 0.0;
0   709  2             VAL  := 0.4;
0   710  2             BBW  := -3;
0   711  2             UNST := 'SEC';
0   712  2             TIST := 'EXECUTION-SPEED';
0   713  2           END;
0   714  1         NEW_KNOB(KNOB_INIT);
0   715  1
0   716  1         END;
0   717  0  1407
```

```
  482 0   718   0  1407 (* PURGE AND EXIT PROCEDURE *)
  483 0   719   0  1407
  484 0   720   0  1407 PROCEDURE STOP_HANOI;
  485 0   721   1     9
  486 0   722   1     9 BEGIN
  487 0   723   1         IF NOT TWO_ERROR THEN
  488 0   724   1              BEGIN;
  489 0   725   2              RTTERMINAL(MIP_TERMINAL);
  490 0   726   2              IF (FLAG <> 0 ) THEN BEGIN
  491 0   727   3                  WRITELN(SCREEN);
  492 0   728   3                  WRITE(SCREEN,'ERROR (NODAL-COD','E) : ',FLAG,' AT LINE : ',LINE);
  493 0   729   3                  WRITELN(SCREEN);
  494 0   730   3                  END;
  495 0   731   2              TWO_ERROR := TRUE;
  496 0   732   2              ERASE_TP;
  497 0   733   2              FOR KNOB := K1 TO K4 DO CLEAR_KNOB(KNOB);
  498 0   734   2              RTTERMINAL(COLOUR); WRITE(SCREEN,CHAR(ERASE)); RTTERMINAL(MIP_TERMINAL
  499 0   735   2              END;
  500 0   736   1         STOP;
  501 0   737   1         END;
  502 0   738   0  1407
```

```
503 0   739   0   1407 (* PROCEDURE TO FIND THE TEMPORARY PILE FOR MOVING DISCS *)
504 0   740   0   1407
505 0   741   0   1407 FUNCTION REMAININGPILE ( A_PILE : PILE; B_PILE : PILE ) : PILE;
506 0   742   1     11
507 0   743   1     11 BEGIN;
508 0   744   1        REMAININGPILE := PILE(ALL_PILE - ORD(A_PILE) - ORD(B_PILE));
509 0   745   1        END;
510 0   746   0   1407
511 0   747   0   1407
512 0   748   0   1407
513 0   749   0   1407 (* RECURCIVE PROCEDURE FOR MOVING DISCS *)
514 0   750   0   1407
515 0   751   0   1407 PROCEDURE MOVE(FROMPILE: PILE; TOPILE: PILE; HOWMANY: PILESIZE);
516 0   752   1     12
517 0   753   1     12 VAR ADISC: DISC;
518 0   754   1     13
519 0   755   1     13 BEGIN
520 0   756   1        IF HOWMANY=1 THEN BEGIN
521 0   757   2           IF TEST_GROUP(TRIGGER_GROUP) OR TEST_GROUP(TP_GROUP) THEN STOP_HANOI;
522 0   758   2           DELAY := READ_KNOB(KNOB);
523 0   759   2           TWAIT(DELAY);
524 0   760   2           TAKE(ADISC,FROMPILE);
525 0   761   2           PLACE(ADISC,TOPILE)
526 0   762   2           END
527 0   763   1        ELSE BEGIN
528 0   764   2           MOVE(FROMPILE,REMAININGPILE(FROMPILE,TOPILE),PRED(HOWMANY));
529 0   765   2           MOVE(FROMPILE,TOPILE,1);
530 0   766   2           MOVE(REMAININGPILE(FROMPILE,TOPILE),TOPILE,PRED(HOWMANY))
531 0   767   2           END
532 0   768   1        END;
533 0   769   0   1407
```

```
 534 0  770  0  1407 (* MAIN BODY OF HANOI PROGRAM *)
 535 0  771  0  1407
 536 0  772  0  1407 BEGIN
 537 0  773  0        INITIALIZE;
 538 0  774  0        CLEARBOARD;
 539 0  775  0        READ_CONDITION;
 540 0  776  0        FOR ADISC := HIGHTH DOWNTO 1 DO PLACE(ADISC,SOURCE_PEG);
 541 0  777  0        MOVE(SOURCE_PEG,DESTINATION_PEG,HIGHTH);
 542 0  778  0        REPEAT
 543 0  779  1            WAIT_EVENT(GROUP,BUTTON);
 544 0  780  1        UNTIL (GROUP = TRIGGER_GROUP) OR (GROUP = TP_GROUP);
 545 0  781  0        STOP_HANOI
 546 0  782  0        END.

COMPILATION SUCCESSFUL
```