```
 ------------------------
/                        \
\                        /
/      H E R M E S       \
\                        /
/                        \
\                        /
 ------------------------
```

Proposal for a master task to run application programs on the PDP-11/70

copies can be obtained from:

20-Sept-78

J. P. Terlouw

K 309

Latest update:
22-Nov-78

"Hermes was de god, die de wegen en het verkeer daarlangs beschermde. Hij was dan ook de god van de reizigers en de handel, zelfs van de dieven, die immers ook hun geluk langs de wegen zoeken. Hij was de bode der goden, toegerust met vleugelschoenen en herautstaf, die op bevel van Zeus boodschappen overbracht en mensen veilig naar hun doel voerde; ook geleidde hij de doden naar de onderwereld."

Uit: Ernst Hoffmann:
"Goden- en Heldensagen"
bewerkt door Dr. J.H.Croon

Contents.

1. Introduction.

Our data processing system on the PDP-11/7Ø (sometimes referred to as GIPSY) consists of a number of separate "tasks". The concept of a task is explained in the appropriate RSX11M reference manuals. In short: a RSX11M task is an independent load module ("core image") that communicates with the outside world only through the RSX11M operating system.

In our system we have three kinds of tasks:

1. s e r v a n t tasks that do the actual work, e.g. loading a tape onto the disk, clean a map or produce a display.
2. s l a v e tasks, of which both application programmers and users of the system are unaware. An example is a task that receives device-independent plot instructions (vectors) and translates them into instructions for a particular device, like the Gould electrostatic plotter or a graphics plane of the M7Ø image computer.
3. o n e m a s t e r task, which is in control of the whole system and responsible for things like starting up servant tasks, communication with both the user of the system and the servant tasks, performing the necessary bookkeeping. The present document concerns this task.

2. Communication with the human user of the system.

The "interface" between the GIPSY system and the user-astronomer is a video display terminal. Its display is an output device through which information from the system can reach the user; its keyboard is an input device via which the user can enter information in order to "steer" the reduction process. The master task proposed in this document allows for the following kinds of user-supplied information for which the format is described in more detail in chapter 2.2.:

- commands to run a servant task (more than one servant task can be active at one time; the maximum is determined by a parameter established when HERMES is built)
- commands to obtain information from the system or from a servant task (e.g. ask for the input parameters for a servant to be displayed)
- commands that change the state of the system or the state of a servant task. (e.g. abort a servant; stop the system)
- input parameters for a servant task

As the proposed master task allows for more than one servant task to run concurrently, care has been taken to prevent one servant task from blocking others when it is communicating with the master, even when it requests input from the user.

There will also be a facility that enables a servant task to assume the role of the human "typist" (and "looker at"). This will open the possibility to write "programs" for automatic data reduction.

2.1. Display organization.

The terminal display screen will be divided into three sections, each with its own purpose. (fig 1.)

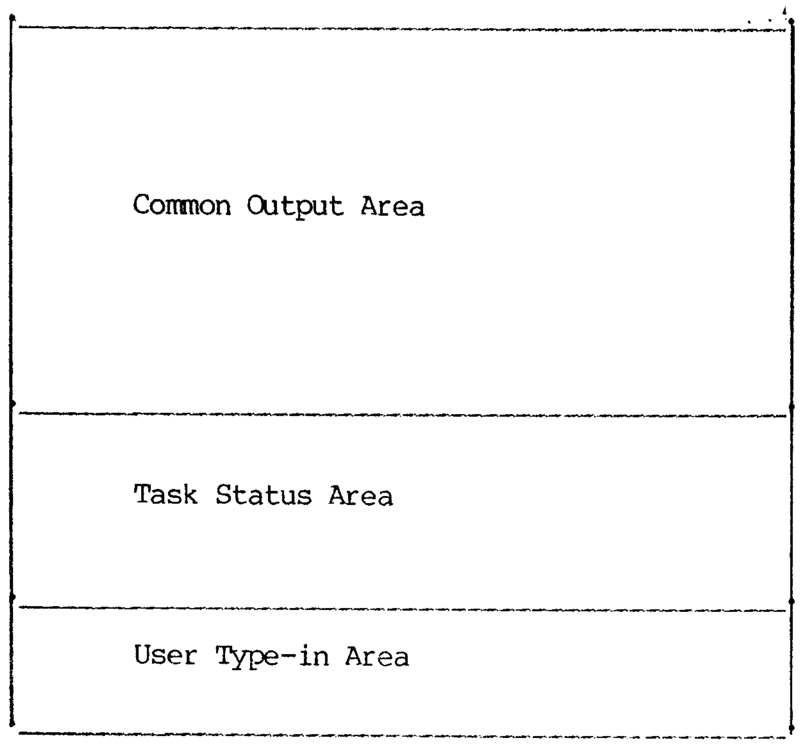


fig. 1

a. Common Output Area (length: what is left by b. and c.)
This section is used to display information that is outputted by servant tasks and meant to reach the user-astronomer via the terminal display.

b. Task Status Area (length: max. number of active tasks)
This section contains one line for every active servant task consisting of the taskname and additional status information, which is updated dynamically.
When a servant task requires user intervention to continue, the status information is preceded by a blinking asterisk. In addition to this, the terminal will "beep" when a task comes into such a state.
The format of a task status entry is one of the following (asterisks shown blink):
- When the task has been activated, but is not running yet:

```
taskname  WAITING TO BE RUN
```

- When the task is running:
  ```
  taskname  RUNNING
  ```

- When the task is pausing:
  ```
  taskname  *PAUSING [optional message]
  ```

- When a task requires user input:
  ```
  taskname  *input request message
  ```

- When a task has finished processing normally:
  ```
  taskname  *ENDED NORMALLY
  ```

- When a task has crashed:
  ```
  taskname  *** CRASHED ***
  ```

The last two displays of course do not require user-intervention. The messages will disappear when the user enters a line or just presses the carriage-return key.

c. User Type-in Area (length: 2 lines)
This section is used to "echo" whatever the user types in. See chapter 2.2.

2.2. User type-in formats.

Wherever a comma is shown in these formats, a blank may also be used.

2.2.1. Activate a servant task.

a. taskname

```
                         example: XVPROF
```

This activation doesn't contain pre-specified parameters, so the servant task will ask everything it needs.

b. taskname,keyword=values,....,keyword=values

```
                         example: RDMAPS,NAME=M1Ø1,CH=1,2,3
```

Pre-specified parameters are "consumed" without asking the user.

2.2.2. Supply parameters to a servant task.

Parameters can be given to a task at any time, even when it is not requesting input. However, they will only become effective when the servant task explicitly requests those parameters.

format: taskname,keyword=values,....,keyword=values
(essentially the same format as 2.2.1 sub b)

2.2.3. Control- and informational commands (CIC's).

This list is not complete. More CIC's are available, primarily for program debugging.

a. Not associated with servant task:

```
/END
```

causes HERMES to stop as soon as any servant tasks are no longer active.

b. CIC's associated with a servant task:

```
taskname,/INP
```

displays all input parameters for a task in the "Common Output Area" on the terminal display screen.

```
taskname,/WAIT
```

causes the task to enter the "PAUSING" state. Note that a task can also enter this state on its own initiative.

```
     taskname,/GO
```

causes the task to leave the "PAUSING" state.

```
     taskname,/ABO
```

aborts the task.

```
     taskname,/SET,n
     taskname,/CLR,n
```

sets or clears switch in task. Only implemented for compatiblity reasons. New programs should not use this facility.

2.3. "Shorthand" notation.

a. The ESC key will automatically provide the

```
taskname,
```

display in the User Type-in Area. Every time ESC is hit, HERMES steps through the active tasks and increments taskname. The user need only complete the line.

b. The space bar will automatically provide the

```
taskname,keyword=
```

display. Every time a space is typed, HERMES steps through the tasks requesting input and changes the display accordingly. The user need only complete the line.

c. The ^U and the DEL keys have their normal effects: ^U deletes the whole type-in; DEL deletes one character.
In addition to this, ^U will also cause the type-in to disappear from the screen.

d. When only one servant task is active, CIC's (chapter 2.2.3.) can be entered without the "taskname," part.
The /GO command can be entered without the "taskname," part if only one task is in the "PAUSING" state, even if there are also other tasks active.

e. For compatibility with WSRT-PDP-9, ^X is a synonym for /ABO .

2.4. Macro's.

A macro processing feature similar (but not identical) to the STER macro facility will be provided.
(See appendix A: excerpt from the STER user's guide)

2.5. <u>Defaults.</u>

In many cases it is not necessary that the user supply all the "steering" information for a servant task. For this reason a "default system" has been designed.

There are three classes of defaults:

a. Local defaults.
   These are defaults that apply only to one task. Every task may have (but is not required to have) a set of local defaults.

b. Global defaults.
   They apply to every task and contain very general information like: the device on which the data-base resides, etc.

c. Program-determined defaults.
   Whenever a servant task requests parameter input from HERMES, and HERMES is unable to give it a n d the servant task can find out a sensible value, the latter is used.

There will be facilities to edit both local and global defaults.

The parameter hierarchy will now be:
1. user-supplied parameters
2. local defaults
3. global defaults
4. program-determined defaults

3. Servant-master communication.

This chapter is only useful for people who intend to write application programs for use under HERMES.

a. Declare the task running to HERMES:

```
CALL INIT('text')
```

b. Enable control commands from HERMES.

```
CALL RDAST
```

(required if USRINP is used)

c. Obtain parameters:

```
CALL USRINP(.....)
```

d. Enter the "PAUSING" state:

```
CALL PAUSE('text')
```

(note that the Fortran PAUSE statement is forbidden)

e. Obtain a switch setting:

```
IF ( SWITCH (n) ) .......
```

(note that new programs should not use this facility)

f. Declare the task to be ended normally:

```
CALL FINIS
```

g. Error handling:

```
CALL ERROR ( severity , 'text' )
```

h. Send text to both logfile and display:

```
TYPE...
 or
WRITE(5,...) ....
```

i. Send text to specified (HERMES-controlled) devices:

```
            TYPE...
             or
            WRITE(5,...)....
                             (special control characters precede text)

            CALL ANYOUT(....)


            WRITE(1,...) ....
               .
               .
               .
            WRITE(1,...) ....
            CALL EASYOU(....)
                             (text is outputted as one
                                     contiguous block  )
```

4. References.

1. RSX11M Executive Reference Manual
2. RSX11M I/O Drivers Reference Manual
3. Note 270 SRZM: RSX11M virtual MCR driver
4. STER user's guide

BEFORE A COMMAND LINE IS EXECUTED ALL MACRO CALLS, DEFINITIONS AND DELET
OUTSIDE THE PARAMETERLISTS ARE PROCESSED.
BEFORE ANYDATA GIVES THE INFORMATION TO THE REQUESTING TASK, MACRO'S ARE
PROCESSED.
A MACRO IS A KIND OF ABBREVIATION WHICH CAN BE USED AS INPUT TO STER OR
ANYDATA ONCE IT HAS BEEN DEFINED.
A MACRO NAME IS A HOLLERITH FIELD WHICH BEGINS WITH A PERIOD AND IS NOT
LONGER THAN 9 CHARACTERS.

THE MACRO PROCESSING IN STER CAN BE DIVIDED INTO THREE SECTIONS.:
1. DEFINE A MACRO
2. DELETE A MACRO
3. EVALUATE A MACRO

AD 1.
FORMAT:

```
          .+ MACRONAME MACROBODY ;
                     .+ MEANS "DEFINE"
                     ; CLOSES THE DEFINITION
```

THE MACRO BODY CAN CONSIST OF ANY TYPE OF INPUT, NUMBERS AND CHARACTER S
, INCLUDING OTHER MACRO NAMES AND -DEFINITIONS.
SPECIAL ITEMS ARE THE CHARACTER STRINGS P1 THROUGH P9 WHICH WILL BE SUBS
D BY PARAMETERS AFTER THE MACRO CALL.
WHEN THE DEFINITION IS STORED IN THE SYSTEM THE WHOLE DEFINITION IS REMO
FROM THE LINE CONTAINING IT.

AD 2.
FORMAT:

```
          .- MACRONAME
                     .- MEANS "DELETE"
```

WHEN A MACRO IS DELETED FROM THE SYSTEM, THE DELETE COMMAND IS REMOVED F
THE LINE CONTAINING IT.

AD 3.
FORMAT:

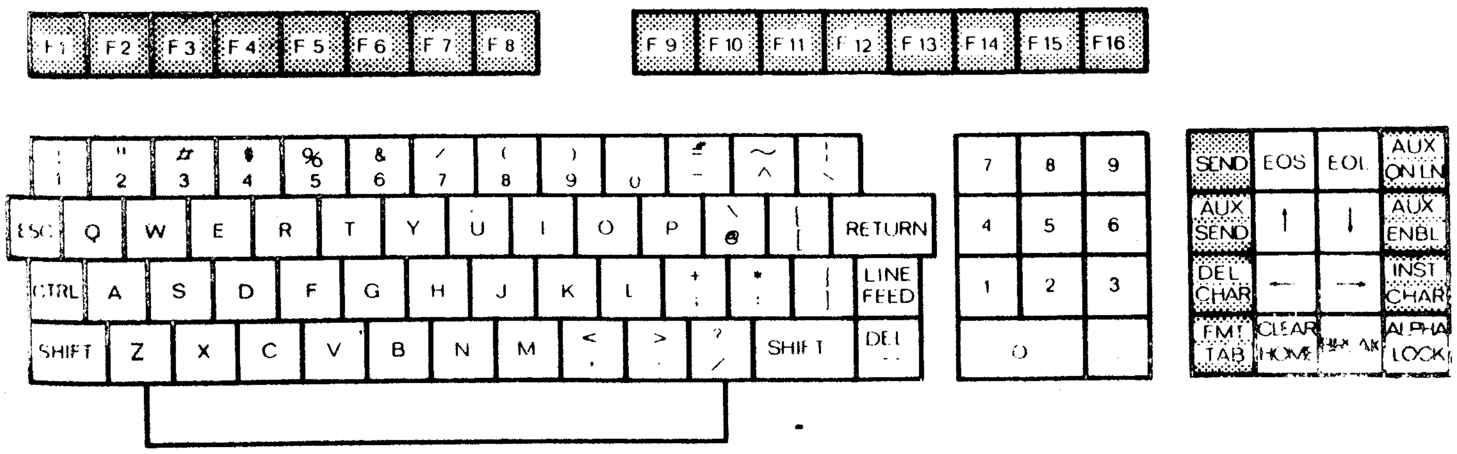
```
          MACRONAME
```

THIS NAME IS REPLACED BY THE DEFINITION FOUND IN THE SYSTEM

```
      EXAMPLES
.+ .ABC THIS IS MACRO TEXT ;
                   DEFINE .ABC

.ABC
         RESULTS IN:
         THIS IS MACRO TEXT

.- .ABC
                   DELETE .ABC

.ABC
                   CALL .ABC AGAIN
         SINCE .ABC DOESN'T EXIST ANYMORE, THIS CALL
         WILL RESULT IN AN ERROR MESSAGE

.+ .DEF .PQR .STU ;
.+ .PQR 1 2 3 ;
.+ .STU 7 8 9 ;

.PQR
         RESULTS IN:
          1 2 3

.DEF
         RESULTS IN:
         1 2 3 7 8 9

.+ .ABUSE P1 IS A TWIT ;
                   A PARAMETER IS INVOLVED

.ABUSE NOBODY
         RESULTS IN:
         NOBODY IS A TWIT

.+ .FDGE P1 P2 P3 P2 P1 ;

.FDGE A B C
         RESULTS IN:
         A B C B A

RECURSIVE MACRO'S WILL RESULT IN A TIME LIMIT.
         EXAMPLE:  .+ .C .C ;
                   .C
```

Appendix B

Figure 1-2 KEYBOARD CONFIGURATION



F1 F2 F3 F4 F5 F6 F7 F8
F9 F10 F11 F12 F13 F14 F15 F16
ESC Q W E R T Y U I O P RETURN
CTRL A S D F G H J K L LINE FEED
SHIFT Z X C V B N M SHIFT DEL
7 8 9 4 5 6 1 2 3 0
SEND EOS EOL AUX ON LN
AUX SEND AUX ENBL
DEL CHAR INST CHAR
FMT TAB CLEAR HOME BREAK ALPHA LOCK

Shaded Areas Denote Optional Keys