STELLA/II Project:
Console, Log, and Timer processes
The TALK Aplication

Nedo Celandroni
Erina Ferro

Report C83-20

Istituto CNUCE
PISA
October 1983

Final Draft

1. The CONSOLE process

The CONSOLE process can be activated on a task request or on operator request.

In the first case, the Console process is invoked in order to print a task message on the operator console. The ITCALL routine is used to enqueue to the Console process a block containing:

```
+------------+
|    LINK    |
+------------+
|  C.WRITE   |
+------------+
|    MSGN    |
+------------+
|    PAR1    |
+------------+
|    PAR2    |
+------------+
|    PAR3    |
+------------+
| FROM PROC. |
+------------+
```

Where:

C.WRITE : is the operation code
MSGN : is the message number
PAR1,PAR2,PAR3,: are the parameters necessary to print the message.

The BYTASK routine is entered and the requested message is printed on the operator console via the BCPL "WRITEF" routine.

In the second case, the CONSOLE process is directly invoked by the operator by typing CON <CARRIAGE RETURN> on the terminal. The Console process in this case prints

CON>

and the command can be entered.

If <CNT/Z> is typed, the command is no more expected. At the moment, the Console process handles the following commands:

1) EDP : the DASM process is requested to display the Enable Block (ENABLE DISPLAY)

2) ES : the DASM process is requested to send the Enable Block (ENABLE SEND)

3) ECB(*) : the DASM process is requested to change the content of a byte in the Enable Block (ENABLE CHANGE BYTE)

4) ECW(*) : the DASM process is requested to change the content of a word in the Enable Block (ENABLE CHANGE WORD)

5) DQL : the QLIST is displayed with the format: NAME : semaphore(decimal), head-word(octal), tail-word(octal), counter(decimal) (DISPLAY QLIST)

6) FDP : the FLAGWD content is displayed (in hexadecimal) (FLAGWORD DISPLAY)

7) GO : to send the <go-ahead> command (GO)

8) RC : the DASM process is requested to restart the CIM. (RESTART CIM)

9) DF : the DASM process is requested to finish. (DASM FINISH)

10) LF : the statistic LOG process is requested to finish. (LOG FINISH)

11) MASTER : the STELLAII system runs as master in NORMAL-mode

12) MASTERTEST : the STELLAII system runs as master in TEST-mode

13) SLAVE : the STELLAII system runs as slave in NORMAL-mode

14) SLAVETEST: the STELLAII system runs as slave in TEST-mode

15) LOCKON : a certain number of LOCKONs are requested to the CIM via the DASM process.

16) S LINES : INTERNET task is requested to <u>start</u> all the line-handlers

17) H LINES : INTERNET task is requested to <u>stop</u> all the line-handlers.

18) D LINES : INTERNET task is requested to <u>display</u> the line-handlers status

The following 4 commands may have <u>S</u> or <u>H</u> or <u>D</u> as keyword. We will generally indicate it with an 'X'. The meaning of the command is respectively "start" (X=S), "stop" (X=H) or "display the status of" (X=D) the specified line-handler.

19) X X25LINE: INTERNET is requested to "X" the X25 line-handler

20) X CRLINE : INTERNET is requested to "X" the CAMBRIDGE RING line-handler

21) X CERLINE: INTERNET is requested to "X" the LKDRV (CERNET) line-handler

22) X SATLINE: INTERNET is requested to "X" the SATELLITE line-handler

23) IF : INTERNET process is requested to finish.

24) DEV : the devices used in STELLA2 are displayed on the operator console.

25) DLS : the current large-data slot size assigned to the station for accessing to satellite is displayed.

26) TF : the timer process is requested to finish.

27) TRAFFIC : the INTERNET RX/TX traffic is diplayed.

28) FALL : the finish command is issued to the following tasks: INTERNET, DASM, LOGPROCESS, SYNCTIM and TALK.

(*) After entering ECB or ECW, the byte/word displacement number is requested (in decimal); after entering DISPLACEMENT, the exadecimal value is requested.

Hence, to use the CONSOLE process via terminal:

```
con      <C.R.>
CON > command <C.R.>
      - reply the eventual questions
CON > <CTN/Z> (when necessary) to exit from the
      Console environment.
```

After that a task command or an operator command has been executed, the receiving queue of the Console process is again scanned looking for a block enqueued. If yes, the task command is executed. If no, the Console process exits.

1.1 Blocks enqueued by the CONSOLE process via ITCALL:

a) In case of commands 1,2,7,8,9,11,12,13,14,15 and 25, a block is enqueued to the DASM task containing:

```
+--------------+
|     LINK     |
+--------------+
| COMMAND CODE |
+--------------+
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|   CONSOLE    |
+--------------+
```

b) In case of commands 3 and 4 a block is enqueued to the DASM task containing:

```
+--------------+
|     LINK     |
+--------------+
| COMMAND CODE |
+--------------+
| DISPLACEMENT |  in the Enable block
+--------------+
| NEW EXADEC.  |
|    VALUE     |
+--------------+
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|   CONSOLE    |
+--------------+
```

c) Commands 5, 6 25 and 27 are directly handled by the Console process which displays the common areas defined in the resident STELLALIB library. The QLIST, FLAGWD and STATISTIC contents are printed.

d) In case of commands 10, 23 and 26 a block like that of point a) is enqueued respectively to the LOG task or to the INTERNET task or to SYNCTIM (TIMER) task.

e) In case of commands 16, 17 and 18, a block as that of point a) is enqueued to the INTERNET task.

f) In case of commands 19, 20, 21 and 22, a block is enqueued to the INTERNET task containing:

```
+--------------+
|     LINK     |
+--------------+
| COMMAND CODE |
+--------------+
|    NUMBER    |   corresponding to the
+--------------+   the correct LINE HANDLER
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|    SPARE     |
+--------------+
|   CONSOLE    |
+--------------+
```

NUMBER will be used by INTERNET as offset in its line-handler table (NTRAVEC) to select the correct line-handler routine address (see STELLA/CNUCE/83/09).

g) In case of command 28, a block as that of point a) is enqueued to the following tasks:

INTERNET
DASM
LOGPROCESS
SYNCTIM
TALK

2. The LOG process

The LOG process is the statistic process in the STELLAII system. It receives statistic information from the other tasks and records them in the STATISTIC.LOG disk file.

The basic idea in order to have not too big disk files, is that a message is recorded in the STATISTIC.LOG file only at the first occurrency. If the same message is consecutively repeated "n" times, it is no more printed but a counter is increased. As soon as a new message, different from the previous one, is received, the string "** REPEATED 'n' TIMES" is written regarding to the previous message, and the same procedure is used on the new message.

In order to close in any moment the STATISTIC.LOG file, the command "LF" (Log Finish) may be entered from the operator console via the CON process. The STATISTIC.LOG file is closed with all the statistics updated till the moment the LF command was entered. The message "---LOG PROCESS FINISH--" is also printed on the operator console.

The present statistics recorded in the STATISTIC.LOG files are:

1) errors on data TX operations (for the INTERNET, DASM
2) errors on data RX operations (for the INTERNET, DASM and X25LINE tasks)
3) LOCKON values
4) null reference bursts received
5) errors on PIO RX operations
6) errors on PIO TX operations
7) small-slot changed: start/stop times as assigned in the allocation list
8) large-slot changed: start/stop times as assigned in the allocation list
9) slave-down indication
10) small-window deleted indication
11) large-window deleted indication
12) master station indication
13) LOCKON requested by operator
14) ADDON value for the station
15) "minimum" and "requested" values for the large slot (as sent from the slave and as received from the master)
16) residual time available for large slots
17) THROPUT value sent by a certain station

18) HELLO messages sent by a slave station
19) HELLO messages received by the master station
20) error code #331 (buffer not found) from the
21) information related to the logical unit which was not connected by the X25LINE task.

3. The TIMER process

The task TIMER handles the timer service requests for the tasks of the STELLAII system.

When a task "A" wants to receive a timer-interrupt at a certain time "x" or every "y" seconds, he sends (via an ITCALL) to the TIMER process a block with the following information:

```
+------------+
| LINK WORD  |
+------------+
| REQUEST-ID |
+------------+
|  SECONDS   |
+------------+
|            |
+------------+
|            |
+------------+
|            |
+------------+
|FROM PROCESS|
|     = A    |
+------------+
```

where the REQUEST-ID is the identification of the timer-service-request.
Bit 15 on in the Request-id means that the request is valid only for one time.
Bit 15 off in the REQUEST-ID means that if the request must be repeated at every specified interval time till task A decides to stop the request or the TIMER process is requested to finish.

SECONDS is the number of seconds of the interval time between one time interrupt and the other.

The TIMER process uses two queues: FREETIMQ and TIMERQ.

FREETIMQ is the queue of the free blocks used by the TIMER when he receives a timer-service request.

TIMERQ is the queue of the active timer-service requests.

At the very beginning, the two queues are so initialized:

```
FREETIMQ:
   TQ.HEAD TQ.TAIL
   +-------+-------+
   |       |       |
   +--|----+--|----+
      V       |
   +------+   |
+-->-     |   |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|             |
|  +------+   |
+-->-    ->-+ |
   +------+ | |
   |      | | |
   +------+ | |
   |      | | |
   +------+ | |
   |      | | |
   +------+ | |
   |      | | |
   +------+ | |
            | |
   +------+ | |
+-->-    ->-+ |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|  |      |   |
|  +------+   |
|     .       |
.     .       |
.     .       |
.     .       |
.     .       |
|  +------+   |
+-->- Ø -<----+
   +------+
   |      |
   +------+
   |      |
   +------+
   |      |
   +------+
   |      |
   +------+
```

```
TIMERQ:
        TQ.HEAD  TQ.TAIL
       +--------+---------+
       |   Ø    |         |
       +---|----+---|-----+
           |        V
           +---<----+
```

Every time the TIMER process receives a timer-service request from a task, a block (5 words) is dequeued from the top of FREETIMQ and enqueued to the bottom of TIMEREQ. In the block the following information is stored:

```
+--------+
|  LINK  |  T.LINK
+--------+
|  CODE  |  T.CODE
+--------+
|SECONDS |  T.TIM
+--------+
|SAVE TIM|  T.SAVTIM
+--------+
|  USER  |  T.USER
+--------+
```

As soon as the TIMER process is started, it initializes the FREETIMQ and TIMERQ queues and enters in a wait state by issuing an

CODE:=ITCALL(GOFREE+EXTRAEVENTS,Ø,MRKEF,Ø,Ø,Ø,Ø)

where MRKEF is the other global event flags associated to the task in addition to the global event flag (EFG) of the task handled by the ITCALL routine (i.e. it is EFG+1).
MRKEF is the EF specified when the mark-time directive is issued and it is set on by the system when the timer-interrupt event occurs.
No mark-time directive is issued by the TIMER task until the first timer request is enqueued to it by a task.
When the TIMER task exit from the wait state, the CODE of the ITCALL routine is tested.
If CODE = PRIVATEEVENT, no request is enqueued by any task, but a timer interrupt occurred. <u>The interval for the timer interrupt is chosen to 1 sec.</u>

The actions performed are:

1) If <u>no timer-service request</u> block is enqueued in TIMERQ, the cancel mark-time directive is issued and the wait state entered (it is useless to receive timer interrupts if no process needs timer services!).

2) If TIMERQ is not empty, the T.CODE word of the block is decreased by one, and, if equal to Ø the user of this timer-service request is alerted by enqueuing to its receiving queue (via ITCALL) a block containing:

```
+-----------+
|    LINK   |
+-----------+
| TIMER-CODE|
+-----------+
|    SPARE  |
+-----------+
|    SPARE  |
+-----------+
|    SPARE  |
+-----------+
|    SPARE  |
+-----------+
|TIMEPROCESS|
+-----------+
```

3) If T.CODE = Ø, a check is made to verify if the timer-service request was valid only once or must be repeated.

4) If bit 15 of the T.CODE word is on, the request was valid only once. The block is dequeued from TIMERQ and returned to FREETIMQ.

5) If bit 15 of the T.CODE word is off, the request must be repeated. Into the T.TIM word, the content of the T.SAVTIM word is copied just to reset the original timeout value.

6) In any case, the same actions and controls are performed on the next block (if any) till the end of the queue (T.LINK word = Ø) is reached.

7) When also the last block of TIMERQ was processed, the mark-time system directive is issued again and the wait state entered.

If CODE ≠ PRIVATEEVENT, a new timer-service request is enqueued to the TIMER process by a task.

The actions performed are:

a) the request code is controlled.

b) If request code = STOPTIM, the TIMER process is requested to finish. The cancel-mark-time directive is issued; the TIMER receiving queue in the QLIST is purged and the process finishes.

c) If request code = STOPONEREQ, a particular timer-service request of the sending task must be cancelled. TIMERQ is scanned in order to find the particular request of that task (contained in RESULT!R.VAL2). If the queue is empty or the request is not found, the TIMER enters again in the wait state.

   If the request is found, the block is dequeued by TIMERQ and enqueued to FREETIMQ. The wait state is entered again.

d) If request code = STOPALLREQ, all the timer-service requests from that task must be cancelled. TIMERQ is scanned in order to find all the blocks with the T.USER word equal to the ID of that task. If the queue is empty or no blocks of that tasks are found, the TIMER process enters again in the wait state. Otherwise, each block of that task is dequeued from TIMERQ and enqueued to FREETIMQ. The wait state is then entered again.

e) If it is a normal new timer-service request, a block is dequeued from the top of FREETIMQ and enqueued at the bottom of TIMERQ. The block is so filled:

```
T.LINK   word  =  Ø
T.CODE   word  =  timer service request code
T.SEC    word  =  number of seconds for the interval
                  time
T.SAVTIM word  =  T.SEC
T.USER   word  =  id-of the task sending this timer
                  service request.
```

f) If it is the very first request in TIMERQ, the mark-time directive (with a timeout = 1 sec) is issued.

g) If it is not the very first request in TIMERQ, and <u>in any case</u>, the wait state is entered again.

4. The TALK application

The TALK process is a very simple application on the top of the internetworking used to exchange operator messages between two gateways of the STELLA/II system. It is seen from the INTERNET task as a transport manager to whom the TALK.TM.INTERFACE routine is associated (see STELLA/CNUCE/83/09).

The TALK process creates the TRG region inside which the TX buffer pool is allocated to send messages. Each buffer in the pool is 64 words long.
The destination of the message is identified by typing the gateway name into $ characters. A 4 characters destination name is allowed. At the moment, allowed destinations are:

$PISA$	(PISA)
$GENE$	(GENEVA)
$FRAS$	(FRASCATI)
$ISPR$	(ISPRA)
$DUBL$	(DUBLIN).

Once specified a destination, it remains valid till that a new $XXXX$ is entered; it means that all messages will be sent to the last specified destination until a new destination is specified.

The internet header is added by the TALK process itself on the top of the message; the source and destination fields are so filled:

- --source network : number of the talk net corresponding to the current gateway
- --source local addr. : 4 characters identifying the current gateway
- --destination network : number of the talk net corresponding to the destination gateway
- --destination local addr.: the 4 characters specified between the $ characters.

Data are the passed to the INTERNET task to be sent to the selected destination gateway. The communication subnetwork to reach the destination is chosen by INTERNET.
When a message is received, it is printed on the operator console with the following format:

--FROM XXXX : text of the message

where XXXX is the name of the source gateway sending the message.
The TALK application is not able, at the moment, to send messages in a broadcast way.