# Multitasking

das Paket MTASK erlaubt kooperatives Multitaskting.

Laden mit

```
1 GET MTASK
1 5 THRU
```

Glossar

```
(PAUSE)   ( -- )
RESTART   ( -- )
6 CONSTANT INT#
LOCAL   ( base adr -- adr' )
@LINK   ( -- adr )
!LINK   ( adr -- )
SLEEP   ( adr -- )
WAKE    ( adr -- )
STOP    ( -- )
SINGLE   ( -- )
MULTI   ( -- )
TASK:  ( size -- )
SET-TASK   ( IP Task -- )
ACTIVATE   ( Task -- ) Activate a Task
BACKGROUND:   ( -- )   Create a Background task
```

aus L/P-F83-Doku:

```
Multitasking low level
(PAUSE)   (S -- )
    Puts a task to sleep by storing the IP and the RP on the
    parameter stack.  It then saves the pointer to the
    parameter stack in the user area and jumps to the code
    pointed at by USER+3, switching tasks.
RESTART     (S -- )
    Sets the user pointer to point to a new user area and
    restores the parameter stack that was previously saved
    in the USER area.  Then pops the RP and IP off of the
    stack and resumes execution.   The inverse of PAUSE.

Initialize current User area to a single task.

Manipulate Tasks
LOCAL  Map a User variable from the current task to another task
@LINK  Return a pointer the the next tasks entry point
!LINK  Set the link field of the current task (perhaps relative)
SLEEP  makes a task pause indefinitely.
WAKE  lets a task start again.
```

```
STOP  makes a task pause indefinitely.
SINGLE  removes the multi-tasker's scheduler/dispatcher loop.
MULTI
   installs the multi-tasker's scheduler/dispatcher loop.
   By patching the appropriate INT vector and enabling PAUSE.

MULTI starts the multi-tasker loop running. SINGLE stops it.
Then type XXX WAKE to start the XXX task.
To put the XXX on hold, use  XXX SLEEP
To restart it, use XXX WAKE

In general, executing the name of a task leaves the address of
its user area on the stack. Words like sleep and wake use that
address.

\ Activate a Task
TASK:  Name, initialize, and allocate a new task.
  Copy the USER Area.  I point to where he pointed.
  He points to me.
  Set initial stack pointers.
  Set dictionary pointer.
  Make task ready to execute. Allocate task in host dictionary.
SET-TASK  assigns an existing task to the code at ip.
  Get top of stack of the task to be used.
  Put IP and RP values on its stack.
  Set its saved stack pointer.

ACTIVATE  assigns an existing task to the following code,
  and makes it ready to execute.

\ Create a Background Task
BACKGROUND:
  Create a new task of default size. Initialize it to execute
  the following code.

Examples:
 The task named counter executes an infinite loop, so STOP is not
 required. Note that you MUST use PAUSE, or no other tasks will
 be executed. PAUSE is built in to all words which do I/O, so
 tasks which do I/O ( like spooler ) do not need to use PAUSE
 explicitly.
```

# Beispiel

```
VARIABLE COUNTS
BACKGROUND: COUNTER  BEGIN
    PAUSE 1 COUNTS +!  AGAIN ;
COUNTER WAKE  MULTI
```

```
COUNTS ?
COUNTS ?
COUNTS ?
SINGLE
```

## Funktionsweise

Das Multitasking-System basiert auf dem des Laxen/Perry-F83 und wurde geringfügig ans FG FORTH83 angepasst. Die Beschreibung zur Funktionsweise und Nutzung des Multitasking-Systems entspricht daher der Dokumentation in C.H. Thing „Inside F83“ (http://forth.org/OffeteStore/1003_InsideF83.pdf).

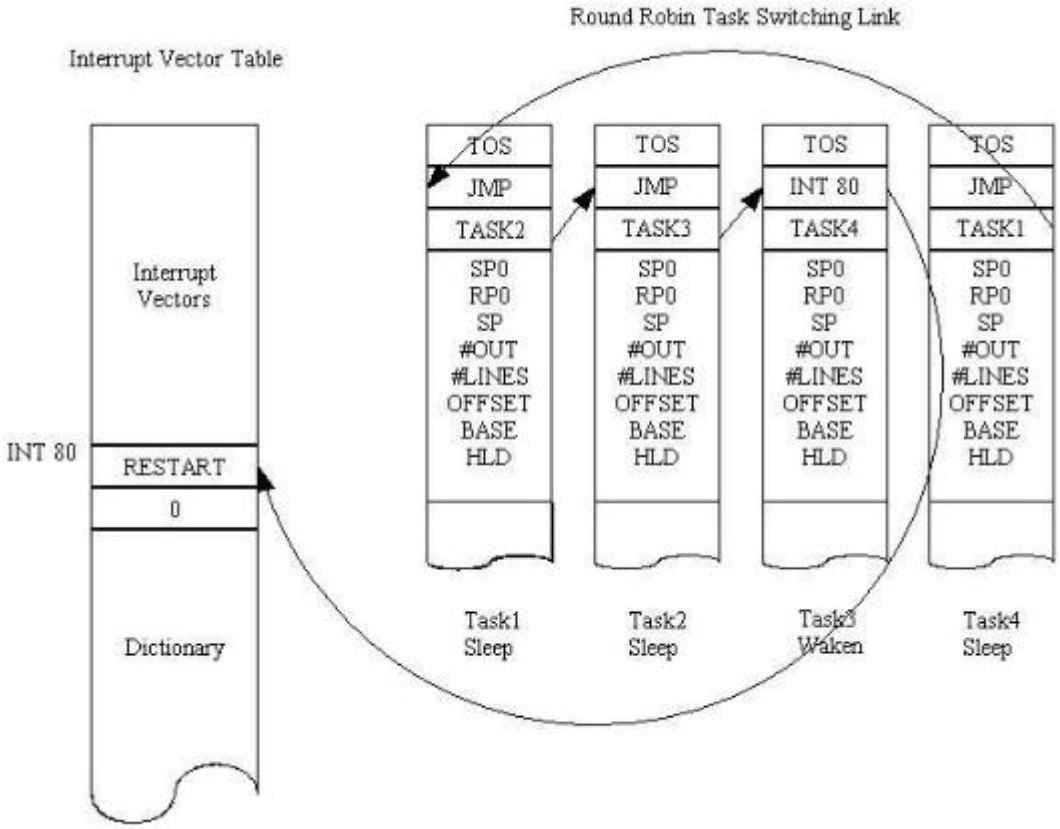


Bild aus InsideF83. In FG FORTH83 wird RST30 statt INT 80 genutzt (Z80 vs. 8086)

Abweichungen:

- Beim Z80 wird statt INT80 der RST30 (RST 6) genutzt.
- instruction pointer ist Register BC
- return stack pointer ist Register IY