ISSN 0105-8525

Running a microprogram on RIKKE-MATHILDA

by

Flemming Wibroe

DAIMI MD-41
October 1980

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade - DK 8000 Aarhus C - DENMARK
Telephone: 06 - 12 83 55

Running a microprogram on Rikke-Mathilda

This paper describes how to load and execute a microprogram on Rikke and Mathilda.

DAIMI MD-41, October 1980

Flemming Wibroe

Computer Science Department
AARHUS UNIVERSITY
Ny Munkegade – DK 8000 Aarhus C – DENMARK
Telephone: 06 – 12 83 55

Contents

1. Physical configuration.

The machine configuration is as follows:

Configuration

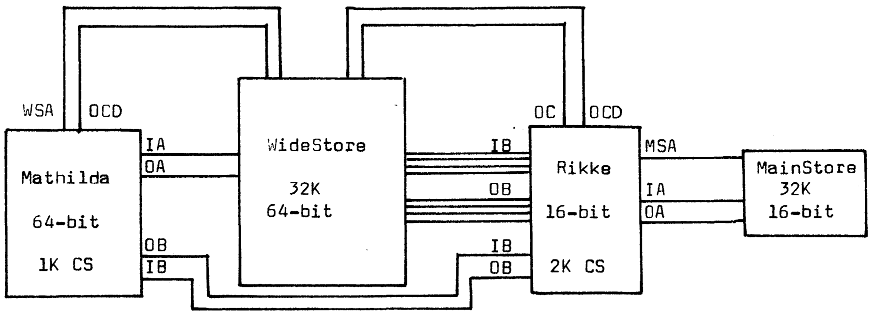

fig. 1.1.

Both Rikke and Mathilda are connected to a 32K 64-bit memory called WideStore(WS). Data is transferred between Rikke an WS through 8 16-bits dataports, 4 for reading and 4 for writing, where a special writeoperation from Rikke allows writing of individual 16-bit groups in WS [6].
Data is transferred between Mathilda and WS through 2 64-bits dataports.

WS is controlled through OC/OCD-ports on Rikke, and through WSA/OCD-ports on Mathilda, details are given in [6].

Rikke and Mathilda can communicate directly through 2 16-bits dataports IB/OB.
Rikke is furthermore connected to a 32K 16-bits local memory called MainStore through IA/OA, where MSA is the address register.

The drawing in figure 1.1 does not represent the full physical Rikke/Mathilda system. Rikke is also connected to other I/O-devices, such as TTY, lineprinter, papertape and Disk-controller, and the DEC-10-system, but for the purpose of this paper these are left out.

2. The microprogram.

2.1. Preparation.

Preparation of a microprogram for Rikke or Mathilda is done using the assemblers and simulators on the DEC-10 [1].
The following example shows how to get the binary microprogram to the Rikke file-system.

Assuming the DEC-10-file ADD.LUI is a source file of a microprogram for Mathilda:
The assembler will produce 2 files:

ADD.LPT : a listning of the program
ADD.PTP : the binary microprogram

If the program is to be simulated on the DEC-10 instead of actually executed on Mathilda, the assembler will produce the file ADD.MTS instead of ADD.PTP.

To be executed on Mathilda, the file ADD.PTP must now be transported to the Rikke file-system, where it must reside in a directory with the extension .MAT (e.g under the name ADD.MAT) [2].
This can be done in two ways:

1. Punch the file on the DEC-10 papertape puncher, and use the command 'readptr ADD.MAT' to read it into a directory on Rikke.

2. Use the transmission-system between DEC-10 and Rikke to transmit the file. The transport must be initiated on both machines:

   on Rikke : readdec ADD.MAT
   on DEC-10 : copy RIKOUT: = ADD.PTP/I

   The "/I" after ADD.PTP is neccessary because of the DEC-10 file-format for binary files, and because the transmission system can be used to send text-files too.

So we have:

```
                       ADD.LUI
assembler             /       \              DEC-10
                 ADD.LPT     ADD.PTP
                               |
                               |           file-transmission
                               v
                             ADD.MAT         Rikke
```

If the file is a microprogram for Rikke, it must be given the extension .MIC on Rikke.

2.2. Structure of the microprogram.

The communication between a BCPL-program and a user microprogram can be done through the library functions, described in the next sections:
This standard communication demands the parameters to be setup in a vector, pvec, and has the following conventions about entry and exit from the user microprogram:

At entry:

```
    LR[LRP] =  pvec,     the address of the parameter  vector
    DS      =  pvec!0 ,  1. parameter
    VS      =  pvec!1 ,  2. parameter
```

At exit:

```
    DS      =  result, the contents of  DS is  written  back
               to the caller
    RB+1       must be used to return from the microprogram.
               This indicates, that the value of RBP  before
               exit must be the same as at entry, so the RB-
               stack must be used carefully.
               The RA-stack can be used freely.
```

Origin:
To avoid overwriting of system-microcode the origins for user microprograms must be greater than:

```
    on Rikke     : 1024 (decimal),  400 (hexadecimal)
    on Mathilda  :  100 (decimal),   64 (hexadecimal)
```

An example of a microprogram for Mathilda that obeys these rules is the following ADD.LUI:

```
 LOUISE VERSION 1.7. PDP-10  17 OCTOBER 1980   ADD.LUI   PAGE 1

 LINENO CS ADDRESS

     ************************************************************
              . ADD: PROGRAM TO ADD 2 INTEGERS
              . 3-9-80 FLEMMING WIBROE
     ************************************************************
     *RADIX  =  16
     *ORIGIN = 109

        109 ADD:   LR:=DS ; ALF:=A+B       ;      . 1 PARAMETER
        10A        AS:=VS ;                ;      . 2 PARAMETER
        10B        DS:=AL ;                ;      . DS=RESULT
        10C               ;                ;RB+1  . RETURN

     ************************************************************

     *ENTRY=ADD
```

3. Microprogramming Rikke.

A microprogram can either be loaded and called under control from a user BCPL-program, or from an interactive system-program. The interactive execution will be described in section [5], here we describe how to handle a microprogram from a user BCPL-program.

The way in which microprograms are executed, differs somewhat on Rikke and Mathilda, primarily because Rikke is host for I/O-nucleus and the OCODE-machine, and secondarily because of the difference in datapath-width, 16-bit for Rikke and 64-bit for Mathilda.
We start by describing how to run a microprogram on Rikke.

3.1. BCPL-library functions.

The library is named "RCSLib.REL" and is located in directory ">SysAdmin>SysUser>RikCS", together with the GET-file "RikHdr.GET". These 2 files must be linked to CurrentDirectory before use.

RCSLib contains the following functions:

```
SetupRikkeCS[silence]
LoadRikkeCS[filename]
DefineRikkeEntry[entry,offset]
CallRikkeCS[entry,pvec]
ResetRikkeEntry[entry]
ResetRikkeCS[]
```

RikHdr.GET contains the globals corresponding to these functions, 200-210, and the manifests used to communicate the results of the functions.

SetupRikkeCS[silence]

This routine must be called before any load of microcode to initiate system-tables etc. The value of silence must be NOSILENCE or SILENCE. If silence=SILENCE, the warnings reported by any of the routines in RCSLib will be suppressed. The errors detected by any of the routines will always be reported regardless of silence.

LoadRikkeCS[fn]

A function, which loads the microcode on file fn.MIC in CurrentDirectory. If the microprogram contains any VALUE-statements, these will be executed by LoadRikkeCS.
The result of the function is:

NOTFOUND : fn.MIC is not in CurrentDirectory.

NOGOOD : an overwrite of existing microcode was attempted, or format-error, sum-error or EOF on fn.MIC.

else an entry-point 'entry', which can be used by CallRikkeCS[entry,pvec]

In any of the two first cases, LoadRikkeCS displays an appropiate message on the console before returning.

DefineRikkeEntry[entry,offset]

In the current assembler [1] it is only possible to specify one entrypoint by "*ENTRY=nn". If, in a large microprogram, an alternative entrypoints is desired, this function can be used to define a new entrypoint to be nn+offset.

The result of DefineRikkeEntry is:

NOGOOD : offset does not specify an entrypoint inside the microprogram, referenced by 'entry'

else as LoadRikkeCS.

CallRikkeCS[entry,pvec]

Calls the microcode, identified by 'entry'. pvec is the address of a communication-area in WideStore, which must be allocated from the calling BCPL-program.
pvec!i must contain the i'th parameter to the microprogram [2.2].

The result of CallRikkeCS is

NOGOOD : 'entry' does not identify a loaded microprogram

else the result of the microprogram, i.e the contents of DS upon exit.

ResetRikkeEntry[entry]

If 'entry' is the result from LoadRikkeCS[fn], this routine discards the microcode, specified by 'entry', i.e disables calling of the microcode and allows loading of new microcode in the same ControlStore locations.

ResetRikkeCS[]

Discards all loaded microcode.

3.2. Example.

The following is an example of a program, which can load and call the program ADD.MIA. This program is a Rikke equivalent of ADD.LUI [2.2].

RIKADD.BCPL:

```
 get "SysHdr"                                    // BCPL-library
 get "RikHdr"                                    // microcode library

 manifest    $( NUMBPARAMS = 2 $)

 let Start() be
 $(S
    let add,pvec,res = 0,0,0
    Load["RCSLib",CurrentDirectory]              // load the library

    // initiate BCPL/microcode communication:
    SetupRikkeCS[NOSILENCE]

    // load the microcode:
    add:=LoadRikkeCS["ADD"]                      // load ADD.MIC
    switchon add into
    $(sw
       case NOTFOUND :
       case NOGOOD   : GiveUp["load aborted"]
                       endcase
       default       : endcase                   // loaded ok
    $)sw

    // initiate pvec:
    pvec:=NewVec[NUMBPARAMS-1]
    pvec!0:=PromptN["1. operand - "]
    pvec!1:=PromptN["2. operand - "]

    // perform call:
    res:=CallRikkeCS[add,pvec]
    if res=NOGOOD then GiveUp["call aborted"]

    // display result:
    OutF[Console,"result of %N+%N = %N*n",pvec!0,pvec!1,res]

    // clean up
    ReturnVec[pvec,NUMBPARAMS-1]
 $)S
 .
```

The library RCSLib.REL is loaded by the program RIKADD.BCPL, but the load,addload,go construction, or the 'combine' program [2] could be used instead.

3.3. Microcoded Library functions.

Microcoded library functions can be used from a user microprogram via the XTERNAL declaration [1].

On Rikke 2 functions are available, the entry-points are specified in hexadecimal addresses:

```
WSREAD  =  21 : read a 16-bit word  from  WideStore.
                The address must be in AS,
                the result is on IB.
                Destroys: AS(15)S, IB, IBD, ALF, OCD and OC.

WSWRITE =  22 : write a 16-bit  word  to  WideStore.
                The address must be in  AS,
                and the value to write in LR[LROP].
                Destroys: AS(15)S, OB, OBD, OCD and OC.
```

For both functions, the AS-address is a 16-bit-word address, as used by the OCODE-machine, so addresses passed as parameters from a BCPL-program can be used immediately (remember WideStore is a 64-bit memory). WSREAD and WSWRITE will do the actual conversions to 64-bit-word addresses and the selection of the correct port-number.

This has as consequence, that WSREAD and WSWRITE only can be used in the OCODE-machines address-space, the lowest half of WideStore, 64K 16-bits words, which equals 16K 64-bits words.
Routines to access the upperhalf of WideStore must be supplied by the user microprograms.

The routines are called as subroutines, using the RA-stack, e.g

```
START:                  .
                        .
           AS:=...      ; RA!           """"" """""       ; R-WSREAD
CONTINUE:  VS:=IB       .
                        .
```

4. Microprogramming Mathilda.

The routines for loading and calling a microprogram in Mathilda are basically identical to those of Rikke, however there are some differences due to:

- the asynchrounous operation of the calling and the called processor
- the difference in datapath-width

Point 1 leads to a slightly different calling-sequence, whereas point 2 gives some complications, when communicating parameters and results.

4.1. BCPL-Library functions.

The library is named "MCSLib.REL" and is located in directory ">SysAdmin>SysUser>MatCS", together with the GET-file "MatHdr.GET". These 2 files must be linked to CurrentDirectory before use.

MCSLib contains the following functions:

```
SetupMatCS[silence]
LoadMatCS[filename]
DefineMatEntry[entry,offset]
MatParVec[n]
SetMatPar[pvec,i,v3,v2,v1,v0]
ReturnMatVec[pvec]
CallMatCS[entry,pvec]
ResetMatEntry[entry]
ResetMatCS[]
InMat64[buf]
OutMat64[buf]
MatDA[]
MatSA[]
MatDeadStart[]
```

MatHdr.GET contains the globals corresponding to these functions, 110-130, and the manifests used to communicate the results of the functions.

Before communicating, Mathilda must be deadstarted, see [2]. The deadstart-loader loads the bootstrap-loader, and hands over control to this. After normalising Mathilda, the bootstrap-loader is ready to load and execute another microprogram.

SetupMatCS[silence]
The purpose of this routine is to initialise the communication between Rikke and Mathilda, and to initialise the tables on Rikke administering the Mathilda ControlStore. The value of silence must be NOSILENCE or SILENCE.
Before SetupMatCS is called, Mathilda must be deadstarted.

To establish a communication between Rikke and Mathilda, a microprogrammed monitor, which can load and execute user microprograms, must be loaded. SetupMatCS loads this communication monitor.
The monitor contains microprogrammed library functions as described in section 4.3, and assures, that the conventions for communication from section 2.2 are obeyed. The text of the current (27/10-80) communication monitor can be found in appendix B.

As with SetupRikkeCS, silence=SILENCE suppresses the warnings given by any of the routines in MCSLib, the errors are always reported.

LoadMatCS[fn]
A function, which loads the microcode on file fn.MAT in CurrentDirectory. If the microcode contains any VALUE-statements, these will be executed by LoadMatCS.
The result of the function is:

NOTFOUND : fn.MAT is not in CurrentDirectory.

NOGOOD : an overwrite of existing microcode was attempted, or format-error, sum-error or EOF on fn.MAT

else an entry-point 'entry', which can be used by CallMatCS[entry,pvec]

In any of the two first cases, LoadMatCS displays an appropiate message on the console before returning.

DefineMatEntry[entry,offset]
equivalent to DefineRikkeEntry

MatParVec[n]
Delivers a vector, pvec, of size 4*n, such that pvec rem 4 = 0.

SetMatPar[pvec,i,v3,v2,v1,v0]
Equivalent to
```
$( let k = (i-1)*4
   pvec!(k+3),pvec!(k+2),pvec!(k+1),pvec!k:=v3,v2,v1,v0
$)
```

ReturnMatVec[pvec]
Returns the vector allocated by MatParVec.

CallMatCS[entry,pvec]

Calls the microcode identified by 'entry'. pvec is the address of the communication area in WideStore as seen from Rikke:

Because of the difference in datapath-width on Rikke and Mathilda, this communication-area is treated in a special way:

1. Parameters are seen as 64-bits words from Mathilda, but must be handled as 4 16-bits words from Rikke.
2. The address of a parameter is a 64-bits word address from Mathilda and a 16-bits address from Rikke, so rikaddress=4*mataddress.
3. As a consequence of this, each parameter, which must be setup from Rikke, consists of 4 16-bits words, where the address of the first word must be divisible by 4.

The call pvec:= MatParVec[n] allocates a communication vector in WideStore of size 4*n, such that pvec rem 4 = 0.
The pvec must now be initialised as follows:

```
pvec!0 = bit 15...0 of 1. parameter
pvec!1 = bit 31..16 of 1. parameter
pvec!2 = bit 47..32 of 1. parameter
pvec!3 = bit 63..48 of 1. parameter

pvec!4 = bit 15...0 of 2. parameter
pvec!5 = bit 31..16 of 2. parameter
pvec!6 = bit 47..32 of 2. parameter
pvec!7 = bit 63..48 of 2. parameter

pvec!8 = bit 15...0 of 3. parameter
   .
   .
   .
```

After the call CallMatCS[entry,pvec], the conventions of section 2.2 means that upon entry to the microprogram

```
LR = pvec/4,          the address of the parameter vector
DS = 1.parameter:     pvec!3::pvec!2::pvec!1::pvec!0
VS = 2.parameter:     pvec!7::pvec!6::pvec!5::pvec!4
```

CallMatCS(entry,pvec) does not wait for any result from the Mathilda microprogram, it merely starts execution and waits only for the Mathilda monitor to reply with an accept of the call to ensure that the microprogram is started, and then returns to the calling BCPL-program with the result ACCEPTED.
The result from the microprogram, the content of DS, can then be obtained by calling InMat64.
If 'entry' does not identify a loaded microprogram, or pvec rem 4 \=0, NOGOOD is returned.

ResetMatEntry[entry]
If 'entry' is the result of LoadMatCS[fn], this routine discards the microcode, specified by 'entry', i.e disables calling of the microcode and allows loading of new microcode in the same ControlStore locations.

ResetMatCS[]
Discards all loaded microcode.

InMat64[buf]
Reads a 64-bit words, send from Mathilda through the direct connection between Rikke and Mathilda, to the 4-word vector 'buf' such that:

```
buf!0  =  bit 15...0
buf!1  =  bit 31..16
buf!2  =  bit 47..32
buf!3  =  bit 63..48
```

This routine is used after CallMatCS to wait for the result of the Mathilda microprogram.
Note: Mathilda sends the word in 4 * 16-bits, so 'buf' need not be divisible by 4, as with pvec.

OutMat64[buf]
Writes a 4*16-bits word to Mathilda through the direct connection in the same format as InMat64. If Mathilda is not ready to read from Rikke, Rikke will be hung up.

MatDA[]
A boolean functions, Mathilda Data Available, which is true, if Mathilda has send a word to Rikke through the direct connection, and Rikke has not read this value yet.

MatSA[]
A boolean function, Mathilda Space Available, which is true, if Mathilda has read the last word send from Rikke through the direct connection.

MatDeadStart[]
If Mathilda is in a welldefined state after having executed a user microprogram, it can be deadstarted from a BCPL-program by calling this routine, i.e unload the communication monitor and the user microprograms, and return control to the bootstraploader.

This routine should be called upon normal exit from the user BCPL-program.

4.2. Example.

The following is an example of a program, which can load and call the program ADD.LUI[2.2].

MATADD.BCPL:

```
 get "SysHdr"                                      // BCPL-library
 get "MatHdr"                                      // microcode library

 manifest $( NUMBPARAMS = 2 $)

 let Start() be
 $(S
    let add,res = 0,0
    and pvec = 0
    and buf = vec 3
    Load["MCSLib",CurrentDirectory]                // load the library

    // initiate BCPL/microcode communication:
    SetupMatCS[NOSILENCE]

    // load the microcode:
    add:=LoadMatCS["ADD"]                          // load ADD.MAT
    switchon add into
    $(sw
       case NOTFOUND :
       case NOGOOD   : GiveUp["load aborted"]
                       endcase
       default       : endcase                     // loaded ok
    $)sw

    // initiate communication area:
    pvec:=MatParVec[NUMBPARAMS]                    // a vector of size 8

    SetMatPar(pvec,1, 0,0,1,0)                     // 1.par = 65536
    SetMatPar(pvec,2, 0,0,0,1)                     // 2.par = 1

    // perform call:
    res:=CallMatCS[add,pvec]
    switchon res into
    $(sw
       case NOGOOD   : GiveUp["call aborted"]
                       endcase
       case ACCEPTED : endcase                     // Mathilda started
       default       : GiveUp["system-error"]//should not occur
                       endcase
    $)sw
```

```
   // wait for Mathilda
   until MatDA[] do
   $(
      // Do some sensible work. This loop need not be here,
      // since InMat64 will wait for Mathilda.
   $)

   // read the result
   InMat64(buf)                                   // the result in 'buf'

   // display result:
   OutF[Console,"result: %U::%U::%U::%U*n",buf!3,buf!2,buf!1,buf!0]

   // clean up:
   ReturnMatVec(pvec)
   MatDeadStart[]
$)S
.
```

4.3. Microcoded library functions.

The following 4 functions are available in the standard communication monitor, and can be used from the user microprograms via the XTERNAL declaration. The addresses are given in hexadecimal:

```
WSREAD    =   8 : read a 64-bit word  from  WideStore.
                  The address must be in AS,
                  the result is on IB.
                  Destroys: ALF, WSA, OCD, IA

WSWRITE   =   9 : write a 64-bit  word  to  WideStore.
                  The address must be in  AS,
                  and the value to write in LR[LROP].
                  Destroys: ALF, WSA, OCD, OA

RIKREAD   =   A : read a full 64-bit  word  send  from
                  Rikke by OutMat64.
                  The result is on AL.
                  Requires: LRIP=LROP
                  Destroys: CA, ALF, LR[LRP], IB

RIKWRITE =    B : write a full 64-bit word  to  Rikke,
                  to be recieved by InMat64.
                  The value to write must be in VS.
                  Destroys: CA, VS, OB
```

The AS-addresses for WSREAD and WSWRITE are 64-bits-word WideStore addresses, so these two functions can be used to access the whole WideStore.

The routines are called as subroutines, using the RA-stack as described in section 3.3.

5. Interactive execution of a microprogram.

If the user only wants to load and call a simple microprogram, i.e a program which only communicate with the calling BCPL-program and only give one value, a word, as its result, this can be done with the programs "Rikke" and "Mathilda".

They consists of respectively "RCSLib" and "MCSLib" together with some input/output routines, and a microprogram catalog.

The commands they accept are:

- load name : load a microprogram from file name.MIC (Rikke) or name.MAT (Mathilda).
- call name : call of the microprogram 'name'. The program asks for the number of parameters and the parameters.
  For Rikke, the parameters can only be given in decimal.
  For Mathilda they can be given as:

		e.g	
decimal	:	102	16-bit
hexadec.	:	XA7B42C	64-bit
octal	:	0713132	64-bit
binary	:	B101001	64-bit

  The result of the microcode call is displayed in decimal, and for Mathilda in hexadecimal too.
- delete name : discards the microprogram 'name'. If 'name'=all all loaded microprograms are discarded.
- list : gives a list of all callable microprograms.
- help : type a help text on the Console.
- end : terminate the program. In case of Mathilda, a MatDeadStart[] will be executed.

The example-programs ADD.MIA and ADD.LUI can both be executed by these two programs.

The two programs reside in SystemDirectory, and are invoked as normal systemprograms by typing their name [2].

6. Restrictions on microprograms.

When running a microprogram on Rikke or Mathilda, some rules about the environment must be obeyed, especially on Rikke, since the I/O-nucleus and the OCODE-machine both are microprogrammed [3],[4], and therefore (possibly) uses the same registers, masks etc. as the user microprogram.

Both Rikke and Mathilda must be left in a normalised state, when the user microprogram terminates. This means:

```
MA[0] = LA[0] = LB[0] = PA[0] = NOMASK    (11...111)
MA[1] = PA[1] = PB[0]         = FULLMASK  (00...000)
MAP   = LAP   = LBP   = PAP   = PBP   = 0
BSS   = PGS   = CM
CUALF = A+B
```

If any of these are omitted, the processor (Rikke or Mathilda) will probably die, when trying to execute the next microprogram, which on Rikke is the OCODE-machine itself.

On the other hand, the user microprograms may also assume, that both Rikke and Mathilda are normalised, when entering the microprogram.

On Mathilda a user microprogram can use all the resources in the machine, the register groups, pointers etc, with the exceptions as mentioned above, and assume that the values are unchanged, when re-entering the microprogram from the BCPL-system, except for the following, which are used by the library routines and the communication monitor:

ALF	CA	CB	LRP	LR[1]
AS	DS	VS	OC	SA
IB	IB	OA	OB	
OCD	WSA			

On Rikke the user microprograms are more restricted. The I/O-nucleus and the OCODE-machine uses some permanent resources, and these must not be changed by the user microprograms. These are:

```
WA[0] :             the OCODE-machine registers
WA[11]-WA[13]       used by I/O-nucleus

WB[0] : 0-15 =      0,1,2,3,,,14,-1      , the constants
WB[2] :             used by the disk-controller
WB[4] - WB[7]       used for OCODE-decoding

MB:5-9              used by the disk-controller
MB:12-14            used by I/O-nucleus and OCODE-machine
LA:14-15,           used by the OCODE-machine
```

MainStore:
MS: 0-256 used by I/O-nucleus

Apart from these, the I/O-nucleus and the OCODE-machine uses some resources, when running, so these cannot be assumed to be unchanged, when re-entering a user microprogram. The following resources are not used, and can be assumed to be left unchanged by the I/O-nucleus and the OCODE-machine:

ALSG [5] - [15]	AVDSG [5] - [15]
BSSG [5] - [15]	BMSG [5] - [15]
CASG [5] - [15]	CBSG [5] - [15]
LA [5] - [13]	LB [5] - [15]
MA [5] - [15]	MB [1] - [4]
PA [5] - [15]	PB [5] - [15]
PMSG [5] - [15]	PGSG [5] - [15]
MSASG[5] - [15]	WB [8] - [15]

The free WA-groups must not be used uncontrolled, since the I/O-nucleus also uses these for device-records. If a WA-group is needed, it must be allocated and deallocated by the calling BCPL-program:

```
group := AllocDB[]          // allocate
DeAllocDB[group]            // deallocate
```

For both Rikke and Mathilda, using VALUE-statements on any of the above permanent resources or the resources concerning the normalised machine, will of course have disastrous consequences too.

Furthermore it should be noted, that VALUE-statements are executed by LoadRikCS and LoadMatCS, see 3.1 and 4.1, and that this execution uses some pointers and registers, when initialising the register-group and pointers.

This means, that all the microcode for a user microprogram should be loaded, before calling any of the microcode, if the microcode assumes any register-group or pointer to be left unchanged by the BCPL-system.

Appendix A: References

[1]: I.H.Sørensen, E.Kressel:
Rikke-Mathilda microassemblers and simulators
DAIMI MD-28, December 1977

[2]: J.K.Kjærgaard and Flemming Wibroe
The RIKKE-BCPL system
DAIMI MD-38, September 1980

[3]: E.Kressel, I.H.Sørensen
The I/O-nucleus on RIKKE-1
DAIMI MD-21, October 1975

[4]: O.Sørensen
The emulated OCODE-machine for the support of BCPL
DAIMI PB-45, April 1975

[5]: P.Kornerup,B.Shriver
A description of the MATHILDA system
DAIMI PB-52, September 1980

[6]: J.K.Kjærgaard
The RIKKE-MATHILDA WideStore
DAIMI MD-42, November 1980

[7]: J.K.Kjærgaard, I.H.Sørensen
The RIKKE-BCPL compiler
DAIMI MD-36, August 1980

[8]: J.K.Kjærgaard, I.H Sørensen
The RIKKE editor
DAIMI MD-37, August 1980

Appendix B: Mathilda monitor

The following is the source text for the Mathilda communication monitor as of 27/10-80. The text, and thereby the addresses is likely to change in the future, but the functions should remain unchanged.

The source text of the Rikke and Mathilda bootstrap loaders can be found in [1].

```
     LOUISE VERSION 1.7. PDP-10    27 OCTOBER 1980    12:38:53                    LOADER.LUI        PAGE 1

LINENO CS ADDRESS
              ****************************************************************************************************
              .
              .   MATHILDA MICRO-PROGRAMS RUN-TIME ENVIRONMENT

              .   9-2-79 :  Exec: LR[0] unchanged, LRP=1
              .                     LR[1] = pointer to parameter vector
              .
              .   9-5-80 :  MEMREAD,MEMWRITE: READ/WRITE DIRECTLY FROM WS
              ****************************************************************************************************
              .
              *0=16
              .
              ****************************************************************************************************
              .
              .   CONTROLSTORE LOAD MODULE
              .
       10     MATLOAD:         ;         RA!                      """"""   ; R-READ             .START-ADDRESS
       11             AL       ;         RA!,     SA:=SB          """"""   ; R-READ             .LOAD-COUNT
       12             AL       ;         CB:=SB, RA!              """"""   ; R-READ             .FIRST WORD
       13     NEWLOAD:AL       ;         OC:=BUS                           ;
       14                      ;                  CSLOAD                   ; SA
       15             DS:=ALLOS;         CB-1              """"""          ; IF CB THEN RB+1    .RETURN TO EXEC
       16                      ;         RA!                      """"""   ; R-READ             .NEXT WORD
       17                      ;                                  SA+1     ;
       18                      ;                                  """"""   ; R-NEWLOAD
              .
              ****************************************************************************************************
              .
              .   CONTROL AND PARAMETER TRANSFER MODULE.
              .   UPON ACTIVATION OF USER MICRO-PROGRAMS
              .      LR POINTS TO THE PARAMETER VECTOR,
              .      DS CONTAINS THE 1. PARAMETER
              .      VS CONTAINS THE 2. PARAMETER
              .
              .      UPON TERMINATION OF A USER-SPECIFIED FIRMWARE FUNCTION
              .      THE CONTENTS OF DS IS AUTOMATICALLY
              .      WRITTEN "BACK" TO THE CALLING SYSTEM ROUTINE, AS
              .      THIS CONTENTS IS ASSUMED TO BE THE RESULT OF THE
              .      FUNCTION CALL
              .
       19     EXEC:            ;                  LRPC                     ; IF KA THEN HERE
       1A                      ;         RA!,     LRP+1           """"""   ; R-READ             .START-ADDRESS
       1B             AL       ;         RA!,     SA:=SB          """"""   ; R-READ             .PARAMETER VECTOR
       1C             AS,LR:=AL;                                           ;
       1D             VS:=ALLOS;         RA!               """""" """"""   ; R-WRITE            .ACCEPT TO RIKKE
       1E                      ;         RA!                      """"""   ; R-MEMREAD          .1.PARAMETER
       1F             DS:=IA   ;                  SETALF+1                 ;
       20             AS:=AL   ;         RA!                      """"""   ; R-MEMREAD          .2.PARAMETER
       21             VS:=IA   ;                                           ; IF KA THEN HERE
       22                      ;         RB!                               ; SA                 .EXECUTE PROGRAM
       23             VS:=DS   ;         RA!                      """"""   ; R-WRITE            .RESULT TO RIKKE
       24                      ;                                  """"""   ; R-EXEC             .READY AGAIN
              .
              ****************************************************************************************************
              *P
```

```
LOUISE VERSION 1.7. PDP-10     27 OCTOBER 1980    12:38:57                          LOADER.LUI          PAGE 2

LINENO CS ADDRESS
              ****************************************************************************************************

              .    LIBRARY ROUTINES

              ****************************************************************************************************
              .
              .   READ A FULL 64 BIT WORD SEND FROM A BCPL PROGRAM ON RIKKE
              .
       25     READ:           ;          ALF:=    ALLOS                          ;
       26                     ;          CA:=     """"""    3                   ;
       27            AS:=AL,<16;                         """"""                  ;
       28                     ;          ALF:=    A&B                            ; IF IBDA THEN HERE+1 ELSE HERE
       29            LR:=IB   ;          CA-1,    IBA                 """"""     ; IF CA THEN HERE+1 ELSE R-2
       2A                     ;                                                  ; RA+1
              .
              ****************************************************************************************************
              .
              .   WRITE A FULL 64 BIT WORD TO A BCPL PROGRAM ON RIKKE
              .
       2B     WRITE:          ;          CA:=     """"""    3                    ;
       2C                     ;                                                  ; IF OBSA THEN HERE+1 ELSE HERE
       2D            VS,OB:=VS,<16;  CA-1,               """"""    OBA           ; IF CA THEN RA+1 ELSE HERE-1
              .
              ****************************************************************************************************
              .
              .   WIDESTORE READ FUNCTION: IA:=WS[AS]
              .
       2E     MEMREAD:        ;                  SETALFB                         ; IF NOT OCSA THEN HERE
       2F            AL       ;          WSA:=SB,IAA,      OCD:=   5             ;
       30                     ;          OCA1                                    ;
       31                     ;                                                  ; IF IADA THEN RA+1 ELSE HERE
              .
              ****************************************************************************************************
              .
              .   WIDESTORE WRITE ROUTINE: WS[AS]:=LR
              .
       32     MEMWRITE:       ;                  SETALFB                         ; IF NOT OCSA THEN HERE
       33            AL       ;          WSA:=SB,          OCD:=   1             ; IF NOT OBSA THEN HERE
       34                     ;          OCA1,    SETALFA                        ;
       35            OA:=AL   ;          OAA1                                    ; RA+1
              .
              ****************************************************************************************************

              .   THESE ROUTINES A CALLABLE THROUGH A TABLE:

              ****************************************************************************************************

              *0=8

       8      RIKREAD:        ;                                       """"""     ; R-READ
       9      RIKWRITE:       ;                                       """"""     ; R-WRITE
       A      WSREAD:         ;                                       """"""     ; R-MEMREAD
       B      WSWRITE:        ;                                       """"""     ; R-MEMWRITE

              ****************************************************************************************************

              *ENTRY = EXEC

              ****************************************************************************************************

              ASSEMBLY CORRECT
```

Running a microprogram on Rikke-Mathilda

Wibroe, Flemming.
Running a microprogram on Rikke-Mathilda / Flemming Wibroe.-- Aarhus, Denmark: Computer Science Department, Aarhus University, 1980.
(DAIMI; MD-41)

I. Title.