

The RIKKE-MATHILDA WideStore

by

Jens Kristian Kjærgård

DAIMI MD-42

November 1980

Computer Science Department

AARHUS UNIVERSITY

Ny Munkegade - DK 8000 Aarhus C - DENMARK

Telephone: 06 - 12 83 55



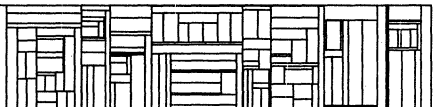
The RIKKE-MATHILDA WideStore

This paper describes WideStore, which is the common memory component of the Rikke Mathilda system.

It is intended to be the firmware reference manual and to be used by the advanced microprogrammer of the Rikke Mathilda system.

DAIMI MD 42
November 1980
Jens Kristian Kjærgård.

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



Contents

1. General description	1
2. Function of the memory	2
2.1. Dataports	2
2.2. Addressports	3
2.3. Blocktransfers	6
3. Using WideStore	7
3.1. Using WideStore from Rikke	7
3.2. Using WideStore from Mathilda	8
3.3. Using WideStore from Device 3	9
3.4. Using WideStore from the Disk	9
4. References	10

Appendices:

A. Exampels using WideStore	11
A.1. Reading/Writing 16-bits words from Rikke	11
A.2. Reading/Writing single words from Mathilda	12
A.3. Block transfer	13
B. WideStore front panel	14



1. General description.

WideStore, WS, is the common memory component of the Rikke-Mathilda system. It is a core memory with 64 bits wordlength, presently with a capacity of 32K words, but designed for 64K words. WideStore is partitioned in two banks with interleaved memory cycles (designed to include four banks). The banks are selected by the least significant bit(s) of the address.

The operation of the memory is managed by an intelligent controller. This controller allows data to the memory to be obtained from four different sources and to be delivered to four different destinations. Input to and output from memory uses independent ports, thus the system has a total of eight dataports. Rikke, Mathilda and the disk occupies two ports each, one set of ports is presently unassigned.

Each transfer can be either a single word transfer (64 bits) corresponding to a single address, or a block transfer obtained or delivered word by word, where the memory address controller automatically increments the memory address.

Addresses can be delivered from three different sources and be used in any of the eight possible datatransfers.

The memory detects parity errors. If an error is found, and the parity switch is turned on, the memory stops.

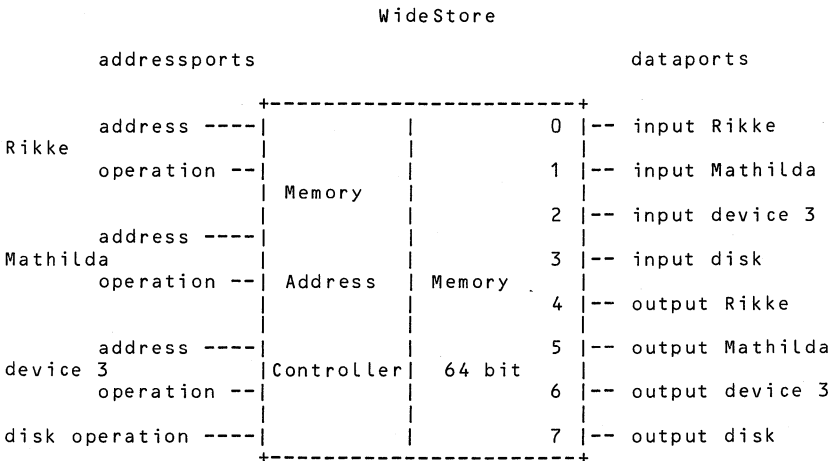


figure 1.1.

2. Function of the memory.

2.1. Dataports.

WideStore has eight 64-bits dataports, four for output from memory (reading from memory), and four for input to memory (writing into memory).

Each inputport can be partitioned into four 16-bits fields, and the port contains a write-bit for each such field. During a write operation a 16-bit field in memory can either be filled with the contents of the buffer or the old contents of the memory location can be rewritten, depending on the state of the write bit. This facility is particular usefull when writing from Rikke.

In figure 2.1 an input port, which write all 4 fields at a time, is shown. (e.g. connected to Mathilda). Notice that the write bit acts as the busy condition too, and all four bits are set and reset simultaneously. A memory cycle is initiated, when the dataport has been activated, and an address is present for that port. Initiated memory transfers are served such that lowest port number has the highest priority. This implies that writeoperations are always served before readoperations.

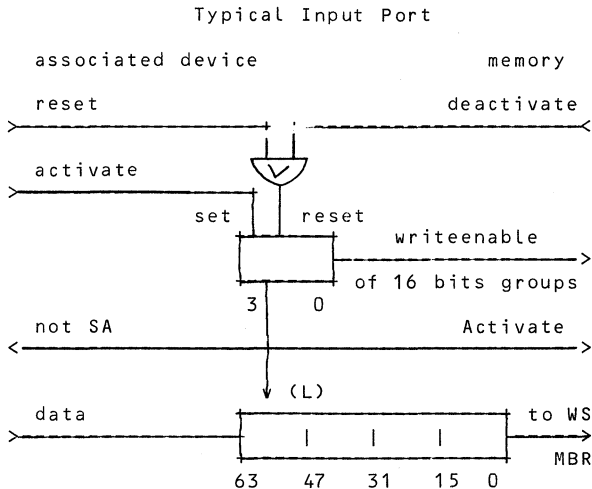


figure 2.1.

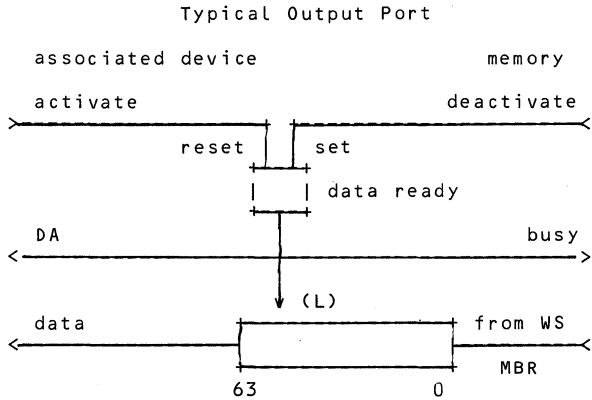


figure 2.2.

2.2. Addressports.

WideStore has three address/control ports, and a control port for the disk.

Port 0 : Rikke

Port 1 : Mathilda

Port 2 : device 3

Port 3 : Disk / only control port

The address port consist of a 16-bits address buffer (allowing memory extension to 64K).

The control port consist of a 4-bit operation buffer. Bit 2-0 is used to select the dataport to be used. So bit 2 distinguish between input and output.

If bit 3 in the operation buffer is set, the address is used as the start address of a block transfer, otherwise for a single memory operation.

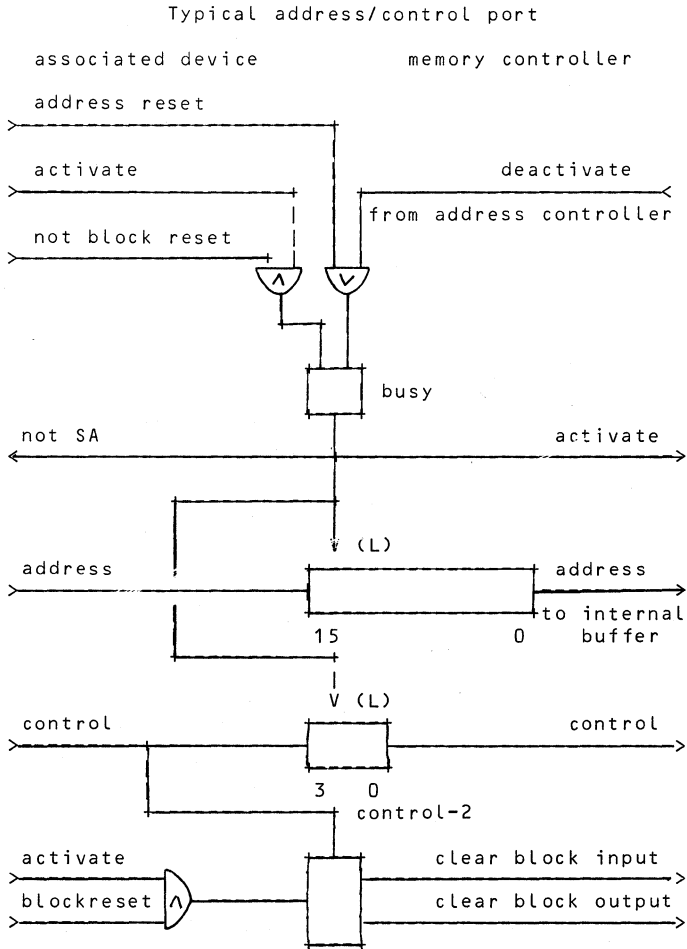


figure 2.3.

The memory address controller has an internal address buffer for each dataport. An address is present, when it has been loaded through some address port into that internal buffer, which was selected by bit 2-0 of the controlport.

The internal buffers are loaded in the following order :

1. Block addresses. (the incremented address from the previous memory cycle of a block-transfer)
2. Rikke addresses.
3. Mathilda addresses.
4. Device 3 addresses.

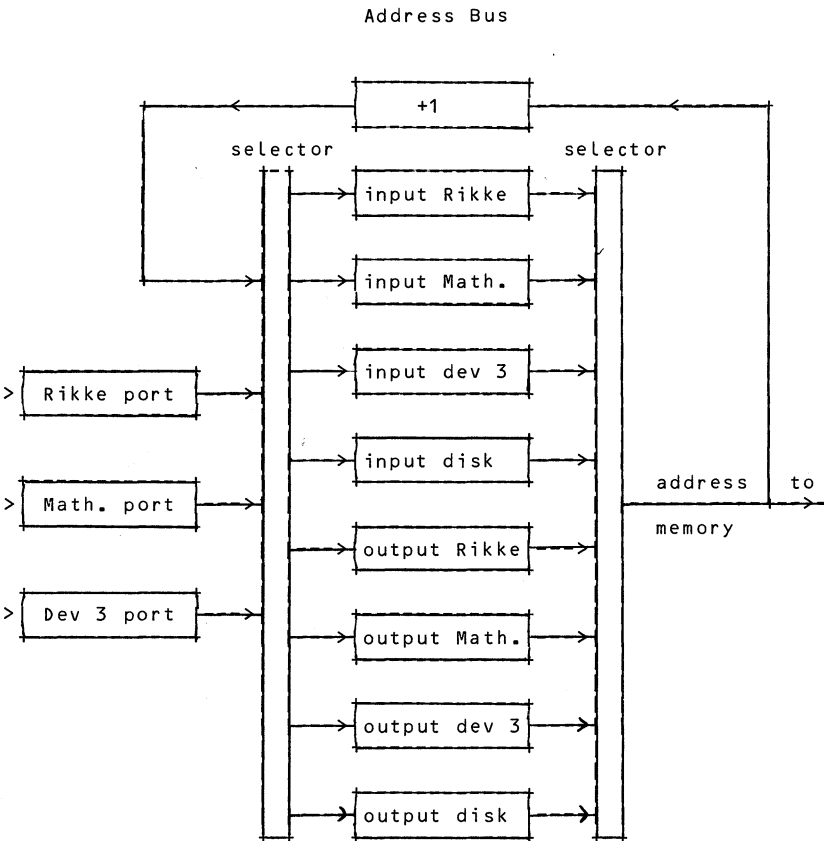


figure 2.4.

2.3. Blocktransfers.

WideStore facilitates simultaneous blocktransfers on each of the eight dataports.

When a blocktransfer has been requested through the operation field, the address will be incremented by one each time data is delivered to/ received from memory.

Since the hardware does not maintain a wordcount, the receiver / sender attached to the dataport has to stop the blocktransfer. This is performed by a clear operation on the appropriate control port. This clear operation must take place before the last data transfer, since the incrementation of the address takes place immediately after the memory read/write cycle.

Notice that it is the receiver / sender of data, who is responsible for stopping a blocktransfer, irrespective of who requested the transfer.

Address overflow in a blocktransfer is detected by the disk controller, and the memory stops if an overflow occur.

3. Using WideStore.

3.1. Using WideStore from Rikke.

OC is used as address.

OCD is used as control.

OCD = 0-7 single word transfer.

OCD = 8-15 blocktransfers on port number OCD mod 8

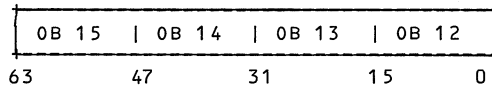
OCSA new address and operation can be loaded.

OCA1 address and control is transferred to the memory address controller.

OCA0 block transfer is cleared, bit 2 of operation (OCD) indicates, whether input or output is cleared. Bits number 0-1 and 3 of control are not used.

OCR clears activation of address port (sets OCSA true).

OB ports are used for data to be written to WideStore, and therefore connected to WideStore port 0.
The following field mapping is used.



OBA0 transfers information in OB to the selected devicebuffer. OBA0 sets the writeenable bit for the particular 16-bits field, which corresponds to the contents of OBD. If all four devicebuffers thereby have been loaded, a write will be initiated.

OBA1 transfer information in OB to the selected device buffer, sets the write enable bit for the selected 16-bits field, and initiates a write, which overwrites those fields, which have been enabled, and rewrites the contents of disabled fields.

OBSA Indicates that data can be loaded at the selected 16-bit field. An enabled field will have the corresponding OBSA false until a writeoperation is completed either by activating with OBA1 or because all four fields have been enabled.

Those devicebuffers, which have not been enabled, will also be set busy (not OBSA) during a write cycle.

OBR resets the enable bit for that field.

- IB port 12-15 receives data from WideStore dataport 4 with the same field mapping as for OB 12-15. Reading from memory will always bring up a full 64 bit word on ports 12 - 15.
- IBA on any of the ports 12-15 makes all dataports ready for reading.
- IBDA indicates that data are available. The condition becomes false on all ports 12, 13 ,14, 15, by an activation on any one of the ports.
- WSMC is a firmware Masterclear of WideStore. It resets all data and address buffers.

3.2. Using WideStore from Mathilda.

- WSA is used as address.
- OCD is used as control.
OCD = 0-7 gives single word transfer.
OCD = 8-15 gives blocktransfer on port number OCD mod 8.
- OCA1 address and operation is transferred to the memory address controller.
- OCA0 block transfer is cleared. Bit 2 of operation (OCD) decides whether input or output is cleared. Bits number 0-1 and 3 are not used.
- OCSA new address and operation can be loaded.
- OA is dedicated to WideStore output (write) and is connected to WideStore dataport 1.
- OAA1 request a write with all four 16-bits fields enabled.
- OAA0 request a (dummy) write with no 16-bit field enabled, having the effect of removing an address from the port.
- OASA indicates that data can be loaded.
- OAR resets databuffer (sets OASA)
- IA is dedicated to WideStore read and is connected to WideStore dataport 5.
- IAA request a read.
- IAA indicates that data has been read (is available).

3.3. Using WideStore from Device 3.

This device has not been specified yet.

3.4. Using WideStore from the Disk.

The disk ports have no address port and the WideStore operation must be initiated from one of the other ports 0,1 or 2 (as a block transfer). So the disk address/operation port is only used for clearing a block transfer. This is done automatically by the disk-controller.

4. 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, I.H.Sørensen
The RIKKE-BCPL compiler
DAIMI MD-36, August 1980
- [7]: J.K.Kjærgaard, I.H.Sørensen
The RIKKE editor
DAIMI MD-37, August 1980
- [8]: Flemming Wibroe
Running a microprogram on RIKKE-MATHILDA
DAIMI MD-41, October 1980

Appendix A: Exampels using WideStore

A.1. Reading/Writing 16-bits words from Rikke.

The microcode in this exampel reads and writes single 16-bits words in WideStore from Rikke.

The code is actually the read and write routines used by the 0-Code machine on Rikke, and also the library routines mentioned in [8] section 3.3.

Rikke microassembly language is used.

VERSION 4.7. PDP-10 27 NOVEMBER 1980 14:14:17

DIARW.MIA

PAGE 1

```

*****
.      IB:=WS[AS]      ( as 16 bits address )
*****
WAITOCR:      ;
              ;AS>,  IBD:= 14                      ;IF OCSA THEN R-PORTSEL ELSE HERE
              ;      AS(15)S:=0,  OCD:= 4          ;IF OCSA THEN R-PORTSEL ELSE HERE-1
              ;      IBD:= 13                      ;IF AS(0) THEN HERE-1
              ;AS>,  IP,  SETALFB                    ;UNLESS OCSA THEN R-WAITOCR
              ;      OC:=BUS,OCA1                    ;IF AS(0) THEN R-RP1415
RP1213: AL    ;      IBD-1  *****                ;
              ;      OC:=BUS,OCA1                    ;R-WAITREAD
RP1415: AL    ;      IBD+1  *****                ;
              ;      OC:=BUS,OCA1                    ;R-WAITREAD
WAITP:        ;      ;                              ;IF IBDA THEN RA+1 ELSE HERE
*****
.      WS[AS]:=LR      ( 16-bits address )
*****
WRITE:
WP1214: AL    ;      SETALFB,OCD                    ;IF AS(0) THEN R-WP1315
              ;AS> AS(15)S:=0,  OBD:= 13          ;IF NOT OCSA THEN HERE
              ;      *****                        ;IF NOT SB(1) THEN R-WP1213 ELSE R-WP1213+1
WP1315: AL    ;AS> AS(15)S:=0,  OBD:= 14          ;IF NOT OCSA THEN HERE
              ;      *****                        ;IF SB(1) THEN R-P1415
              ;AS>,  OBD-1                          ;R+2
WP1213:      ;AS>,  OBD+1                          ;
P1415:       ;AS>,  OC:=BUS,OCA1  *****          ;IF NOT OBSA THEN R-WAITOB
AL          ;      ;                              ;RA+1
OB:=LR      ;      OBA1                          ;IF OBSA THEN HERE-1 ELSE HERE
*****
ASSEMBLY CORRECT

```


A.2. Reading/Writing single words from Mathilda.

Single 64 bits words are read and written between Mathilda and WideStore. It is actually the read and write routines used by the Mathilda micromonitor and hence the library routines mentioned in [8].

Mathilda microassembly language is used.

VERSION 1.7. PDP-10 27 NOVEMBER 1980 14:21:27 MRW.LUI PAGE 1

```

*****
.      IA:=WS[AS]
*****
MEMREAD:      ;      SETALFB      ; IF NOT OCSA THEN HERE
              ;      WSA:=SB,IAA,   OCD:= 5      ;
              ;      OCA1          ;
              ;                      ; IF IADA THEN RA+1 ELSE HERE
*****
.      WS[AS]:=LR
*****
MEMWRITE:    ;      SETALFB      ; IF NOT OCSA THEN HERE
              ;      WSA:=SB,   OCD:= 1      ; IF NOT OBSA THEN HERE
              ;      OCA1, SETALFA ;
              ;      OAA1          ; RA+1
*****
ASSEMBLY CORRECT

```

A.3. Block transfer.

This example shows a block transfer to and from a WA group on Mathilda.

Mathilda microassembly language is used.

VERSION 1.7. PDP-10 27 NOVEMBER 1980 14:03:53 BLOKEX.LUI PAGE 1

```

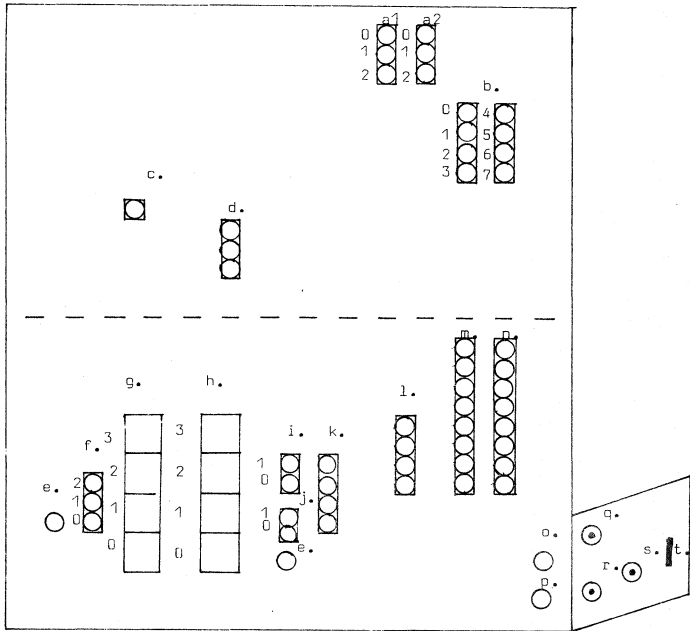
*****
.      WACWAG]=WSEVS] - WSEVS+15]
*****
WStoWA:      ;CA:=          14,      WAUC      ;IF NOT OCSA THEN HERE
VS           ;WSA:=SB,IAA,   OCD:=    13      ;
           ;OCA1          ;
SALOOP:      ;CA-1          ;IF NOT IADA THEN HERE
WA:=IA       ;              ;IF NOT CA THEN HERE-1
           ;              ;IF NOT IADA THEN HERE
           ;OCA0          ;
           ;IAA           ;
WA:=IA       ;              ;IF IADA THEN RA+1 ELSE HERE
*****
.      WSEVS] - WSEVS+15]=WACWAG]
*****
WAtoWS:      ;CA:=          15,      WAUC      ;IF NOT OCSA THEN HERE
VS           ;WSA:=SB,     OCD:=     9      ;
           ;OCA1          ;
ASLOOP:      ;CA-1          ;IF NOT OASA THEN HERE
OA:=WA       ;WAU+1, OAA    ;IF NOT CA THEN HERE-1
           ;OCA0          ;
           ;              ;IF OASA THEN HERE
OA:=WA       ;OAA          ;RA+1
*****

ASSEMBLY CORRECT

```

Appendix B: WideStore front panel

Upper half of WideStore Cabinet








-  = Group of light diodes
-  = Single light diode
-  = Hexadecimal digit
-  = Single pulse button.
-  = Switch

figure B.1.

- a1. write WS from disk.
 - 0. disk busy.
 - 1. WS busy.
 - 2. transfer enabled.
- a2. read WS from disk.
 - 0. disk busy.
 - 1. WS busy.
 - 2. transfer enabled.
- b. blockmark port 0-7
- c. address overflow
- d. address port busy
- e. clocks
- f. selected dataport for last initiated memory cycle.
- g. address of last initiated memory cycle.
- h. address of last read word.
- i. dataport for last read word.
- j. bank for last read word.
- k. parity errors for each bank.
- l. bank ready.
- m. address ready in internal buffer.
- n. dataport ready.
- o. powerfail.
- p. run.
- q. reset (MasterClear of WideStore).
Clears all address- and dataports and blockmarks.
- r. run.
- s. step.
- t. parity switch (down=on up=off).



Micro
Archives
2-54

Kjoergard, Jens Kristian.
The RIKKE-MATHILDA WideStore / Jens
Kristian Kjoergard.-- Aarhus [Denmark]:
Computer Science Department, Aarhus
University, 1980.
"DAIMI MD 42."

I. Title.