MD-43 Kjærgård: The Rikke Processor

ISSN 0105-8525

The Rikke Processor - Described as Rikke-Mathilda differences

by

Jens Kristian Kjærgård

DAIMI MD-43

December 1980

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

The Rikke Processor -
Described as Rikke-Mathilda differences.

The Rikke Microprogrammable processor was built as a 16 bit prototype of the 64 bit Mathilda processor. Apart from the architectural differences due to the width of datapaths and registers, the two processors differ in a few places, where the design was changed after the Rikke prototype was built. Also a few facilities of Mathilda have not been implemented in Rikke, e.g. the BitEncoder.

The purpose of this paper is to describe the differences between the two processors in such a way that it together with the full documentation on Mathilda given in [5], can form a sufficient background for programming Rikke.

The paper consists of a listing of differences between the two processors, structured as comments on the individual sections in section 2 of [5], together with comprehensible tables of microoperations and conditions available in Rikke.

DAIMI MD 43
December 1980
Jens Kristian Kjærgård

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

Contents

1. Summary of differences.

The major differences between Rikke and Mathilda are :

1. The datapath (MDP) is only 16-bits wide, and so are all associated registers.

2. There is no Status Port (SP) on Rikke. Instead the LR registers can be used directly as Bus source.

3. There is no BitEncoder (BE) on Rikke.

4. There are no Condition Registers on Rikke (CR).

5. There is no External Register (EX) on Rikke. Whenever EX is used in Mathilda as input to a Standard Group Pointer (SGP) on Mathilda the OD register is used on Rikke instead.

6. There are no bus latches on Rikke. This implies, that bustransfers using the same register as source, as well as destination will not be secure. For further information see the comments in section 2.20.1.

7. CAP, CBP, WAGSP, WAUSP, WBGSP, WBUSP have no save registers, and cannot be loaded. They can only be cleared, incremented and decremented.

8. LA, LB, MA, MB, PA and PB are loaded with the inverted contents, of what is to be used as mask.

9. SETALFA, SETALFALLOS and SETALFALL1S do not exist on Rikke.

10. The following loads of standard group pointers must be remarked.

```
WAU:= CM | OD(3:0)   | SB(3:0)   | SG
WAG:= CM | OD(7:4)   | SB(7:4)   | SG
WBU:= CM | OD(11:8)  | SB(11:8)  | SG
WBG:= CM | OD(15:12) | SB(15:12) | SG
LAP:= CM | OD(15:12) | S1 | S2
LBP:= CM | OD(7:4)   | S1 | S2
```

2. Comments on each entity.

This chapter should be viewed as a partial substitution of the corresponding sections in chapter 2 in [5].
The exampels in [5] are in general illegal because of the different datapath width.
Symbols and abbreviations are as in [5].

2.1. The Register Group and Standard Group.

As in Mathilda.

2.2. Counter A.

CA is loaded from OD instead of EX. ie.

CA:= CM | OD | SB | SG

CAP has no save registers, and CAP cannot be loaded, only cleared, incremented and decremented.

2.3. Main Data Path Transport.

BUS is 16 bits wide only. See further comments in section 2.20.

2.4. Working Registers.

See comments in section 2.24.

2.5. The Bus Shifter.

The BS is a 16 bits cyclic shifter. The amount of shift can be selected from one of three possible sources :

1. A data field in the CM.

2. The least significant 4 bits of the OD-register.

3. An element of a the 4 bits wide BSSG.

BSS:= 'CM' | 'OD' | 'SG'

A 16-bits left cyclic shifter and a 16 bits right cyclic shifter are related by the expression lshift = 16-rshift.

2.6. Bus Masks.

BM (MA\/MB) is 16 bits wide only.

MA and MB have to be loaded with the inverted masks, ie. if SB = 11...101 then MA:= gives MA = 00...010

2.7. Postshift Masks.

PM (PA\/PG) is 16 bits wide only.

PG generates only 32 masks, and the postshift mask generation data can come from 5 bits of CM, OD or PGSG.

PGS:= 'CM' | 'OD' | 'SG'

mask generator data decimal	binary	mask binary
0	00000	1111 1111 1111 1111
1	00001	0111 1111 1111 1111
2	00010	0011 1111 1111 1111
3	00011	0001 1111 1111 1111
4	00100	0000 1111 1111 1111
5	00101	0000 0111 1111 1111
6	00110	0000 0011 1111 1111
7	00111	0000 0001 1111 1111
8	01000	0000 0000 1111 1111
9	01001	0000 0000 0111 1111
10	01010	0000 0000 0011 1111
11	01011	0000 0000 0001 1111
12	01100	0000 0000 0000 1111
13	01101	0000 0000 0000 0111
14	01110	0000 0000 0000 0011
15	01111	0000 0000 0000 0001
16	10000	0000 0000 0000 0000
17	10001	1000 0000 0000 0000
18	10010	1100 0000 0000 0000
19	10011	1110 0000 0000 0000
20	10100	1111 0000 0000 0000
21	10101	1111 1000 0000 0000
22	10110	1111 1100 0000 0000
23	10111	1111 1110 0000 0000
24	11000	1111 1111 0000 0000
25	11001	1111 1111 1000 0000
26	11010	1111 1111 1100 0000
27	11011	1111 1111 1110 0000
28	11100	1111 1111 1111 0000
29	11101	1111 1111 1111 1000
30	11110	1111 1111 1111 1100
31	11111	1111 1111 1111 1110

2.8. The Arithmetical and Logical Unit.

AL is 16 bits wide only.

SETALFA, SETALFALLOS and SETALFALL1S do not exist.

Condition AL(15) replaces AL(63).

2.9. The Local Registers.

LR is 16 bits wide only.

LR can be chosen as source for a bustransfer directly in Rikke.

Be carefull when using LR as destination for a MDP-transfer see comments in section 2.20.1.

2.10. The Accumulator Shifter.

AS is 16 bits wide only.

AS(V)S is 4 bits wide only.

AS(V)S:= CM | OD | SB | SG

source no	AS(15) input	AS(0) input
0	0	0
1	1	1
2	AS(0)	AS(15)
3	AS(15)	BUS(15)
4	spare	SB(15)
5	DS(V+1)	DS(V+1)
6	AS(V)	AS(V)
7	VS(V)	VS(V)

Condition AS(15) replaces AS(63).

2.11. The Variable Width Shifter.

VS is 16 bits wide only.

VS(V)S is 4 bits wide only.

VS(V)S:= CM | OD | SB | SG

source no	VS(15) input	VS(0) input
0	0	0
1	1	1
2	VS(0)	VS(15)
3	VS(15)	BUS(14)
4	spare	SB(14)
5	DS(V)	DS(V)
6	VS(V)	VS(V)
7	AS(V)	AS(V)

Condition VS(15) replaces VS(63).

2.12. The Double Shifter.

DS is 16 bits wide only.

DS(V)S is 4 bits wide only

DS(V)S:= CM | OD | SB | SG

source no	DS(15) input	DS(14) input	DS(1) input	DS(0) input
0	0	0	0	0
1	1	1	1	1
2	DS(1)	DS(0)	DS(15)	DS(14)
3	DS(15)	DS(15)	BUS(15)	BUS(14)
4	spare	spare	SB(15)	SB(14)
5	DS(V+1)	DS(V)	DS(V+1)	DS(V)
6	AS(V)	VS(V)	AS(V)	VS(V)
7	BUS(1)	BUS(0)	spare	spare

2.13. The AVD Shifter Standard Group.

AVDSG is 4 bits wide only.

AVDP:= CM | OD | S1 | S2

2.14. Loading Masks.

LA and LB are 16 bits wide only.

LA and LB are loaded with the inverted masks.

LAP:= CM | OD(15:12) | S1 | S2

LBP:= CM | OD(7:4) | S1 | S2

2.15. The BUS Parity Generator.

The parity of the 16 bits wide BUS.

2.16. The Bit Encoder.

Not implemented on Rikke.

2.17. The Status Port.

Not implemented on Rikke.

2.18. Input Facility.

IA and IB are 16 bits wide only.

2.19. Output Facility.

OA, OB , OC and OD are 16 bits wide only.

2.20. The MDP Structure.

Rikke Main Data Path

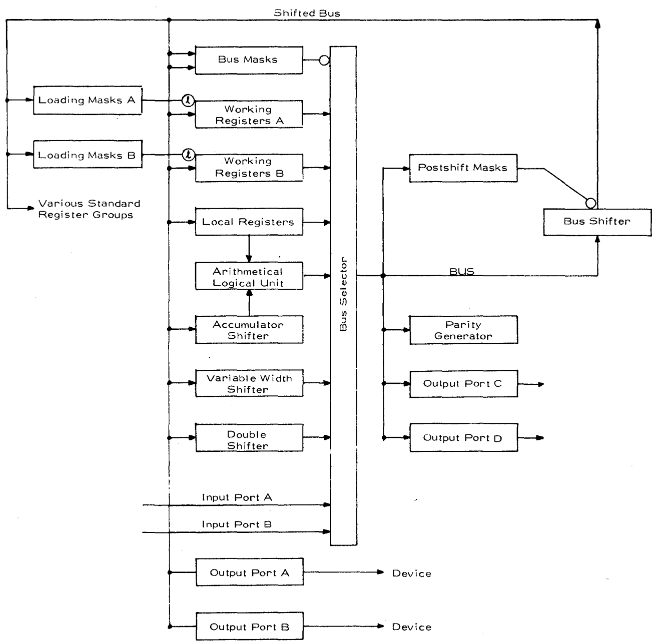

figure 2.1.

The LR register can be used as input to the Bus Selector, and there exist no SP on Rikke.

So we have the following sources for a MDP transfer.

WA
WB
LR
AL
VS
DS
IA
IB

2.20.1. The Bus Latch and the Shifted Bus Latch.

Rikke has neither a Bus Latch nor a Shifted Bus Latch. This has two serious implications.

1. It is not possible to force the BUS to be ALL1S or to force the SB to be ALLOS. This is the reason why the masks MA, MB, PA, LA and LB must be inverted by load, since it otherwise had been impossible to initialize the masks at deadstart.

2. Using the same register as both source and destination for a MDP transport is only allowed for the shift registers. (AS, VS and DS)
   So there are the following restrictions on the pair of destination and source.

   1. When WA is used as destination, WA must not be used as source.

   2. When WB is used as destination WB must not be used as source.

   3. When LR is used as destination none of the following situations must occur.

      1. LR is used as source and LRIP=LROP.

      2. AL is used as source and LRIP=LROP and ALF function involves A (LR).

2.21. The Control Unit.

As in Mathilda.

2.21.1. Microinstruction Sequencing.

As in Mathilda.

2.21.2. The Control Unit Arithmetical Logical Unit.

As in Mathilda.

2.21.3. Return Jump Stack Facilities A and B.

As in Mathilda.

2.21.4. The Save Address Register.

As in Mathilda.

2.21.5. The External Register.

Not implemented on Rikke.

2.21.6. The Force 0 Address Capability.

Not implemented on Rikke.

2.21.7. The Microinstruction Address Bus.

As in Mathilda.

2.21.8. Control Store Loading.

Because of the 16 bits datapathwidth each CS location is loaded in four parts (most significant bits first) and a modulo four counter, named Load Counter, LC, is automatically incremented by a CSLOAD. LC can be cleared by the microinstruction LCC. OB[0] is used as databuffer for the Control Store Load (instead of OC as in Mathilda).

2.22. Condition Registers.

Rikke has no Condition Registers.

2.22.1. Short and Long Cycle.

Rikke can operate in short cycle mode only.

2.23. Auxiliary Control Facilities.

2.23.1. Counter B.

CB cannot be loaded from BE

CB:= CM | SB | SG

2.23.2. The Snooper Facility.

Rikke has no Snooper Facility.

2.24. An Alternative View of the Working Registers.

WAGSP, WAUSP, WBGSP and WBUSP have no save registers and cannot be loaded only cleared, incremented and decremented.

The loads of WAU, WAG, WBU, WBG are :

WAU:= CM | OD(3:0) | SB(3:0) | SG

WAG:= CM | OD(7:4) | SB(7:4) | SG

WBU:= CM | OD(11:8)| SB(11:8)| SG

WBG:= CM |OD(15:12)|SB(15:12)| SG

Notice that WBG and WBU load are from the most significant 8 bits.

2.25. Alternative View of the Postshift Masks.

PA, PB and PG are 16 bits wide and the masks PA and PB are loaded with the inverted mask.

2.26. Main Store Address.

Rikke has a 16 bits 32K word local memory called MainStore. It is connected to Rikke in the same way as WS to Mathilda, through the IA and OA ports. The address register is called MSA, and the associated SG is called MSASG.

3. Microinstruction tables.

3.1. Microoperation tables.

This chapter corresponds to section 3.3 in [5].
The meaning of XX, VV, YY, ZZ, EE are changed according to the architecture of Rikke.

XX = SG | SB

VV = SG | SB

YY = SB | S1 | S2

ZZ = S1 | S2

EE = SB

Inside the table, Θ means, that the microinstruction is not implemented in that field.

Outside the table, Θ means, that the microoperation is not yet implemented.

MICROOPERATIONS FOR Arithmetic Logical Unit, AL

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3	S3	M D_4	F4	MICROOPERATION	
2					M	ALP :=	ZZ CM	D	d d d d	Load the AL SG Pointer from CM|OD|S1|S2	
2					M	ALP +1				Increment AL SG Pointer	
2					M	ALP -1				Decrement AL SG Pointer	
2					M	ALPC				Clear AL SG Pointer	
2			M	ALPS1:=	M	ALPS1:=	ZZ CM	D	d d d d	Load the AL SG Save1 register from CM|OD|S1|S2	
1	ALPS2:=ALP									Load the AL SG Save2 register from the AL SG Pointer	
1			M	ALSG:=SB						Load the AL SG with SB(5:0)	
2	ALF:=	XX CM	D	d d d d d d						Load the AL Function register from CM|OD|SB|SG	
2			M	SET ALF +						Set AL Function to A+B (≡ LR+AS)	
2			M	SET ALF -						Set AL Function to A-B (≡ LR-AS)	
2			M	SET ALF A				M	SETALF A	Set AL Function to A (≡ LR)	⊖
2			M	SETALF +1						Set AL Function to A+1 (≡ LR+1)	
2			M	SETALF B				M	SETALF B ⊖	Set AL Function to B (≡ AS)	
2			M	SETALF ALL0S				M	SETALF ALL0S	Set AL Function to generate 00....0	⊖
2			M	SETALF ALL1S				M	SETALF ALL1S	Set AL Function to generate 11....1	⊖

MICROOPERATIONS FOR Accumulator Shifter, AS

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3	S3	M D_4	F4	MICROOPERATION
2	AS(0)S:=	XX CM	D	d d d				M	AS(0):=	Load the AS(0) Source register from CM|OD|SB|SG
2	AS(15)S:=	XX CM	D	d d d				M	AS(63)S:=	Load the AS(15) Source register from CM|OD|SB|SG
2	AS(V)S:=	XX CM	D	d d d d				M	AS(V)S:=	Load the AS(V) Selection register from CM|OD|SB|SG
2								M	ASLL	Set the AS to a logical left shift
2								M	ASLR	Set the AS to a logical right shift
2					M	AS(V)SC				Clear the AS(V) Selection register
2					M	AS(V)S +1				Increment the AS(V) Selection register
2					M	AS(V)S -1				Decrement the AS(V) Selection register

MICROOPERATIONS FOR AVD (AS, VS, DS) Standard Group and parallel mops

7	2	1	7	1	7	2	1	7

C_p	F1	S1	MD_2	F2	MD_3	F3	S3	MD_4	F4	MICROOPERATION
2					M	AVDP:=	ZZ CM	D	d d d d	Load the AVD SG Pointer from CM|OD|S1|S2
2					M	AVDP +1				Increment the AVD SG Pointer
2					M	AVDP -1				Decrement the AVD SG Pointer
2					M	AVDPC				Clear the AVD SG Pointer
2			M	AVDPS1:=	M	AVDPS1:=	ZZ CM	D	d d d d	Load the AVDP Save1 register from CM|OD|S1|S2
1	AVDPS2:= AVDP				M	AVDPS2:= AVDP				Load the AVDP Save2 register from the CS Pointer
1			M	AVDSG:=SB						Load the AVD SG from SB(3:0)
2								M	AVDLL	Set AS, VS and DS to logical left shift
2								M	AVDLR	Set AS, VS, and DS to logical right shift
2					M	AVD(V)SC				Clear AS, VS, and DS Variable Bit Selection register
2	AVD(0)S:=	XX CM	D	d d d						Load AS(0), VS(0), and DS(1:0) Source register from CM|EX|SB|SG
2	AVD(15)S:=	XX CM	D	d d d						Load AS(15), VS(15), and DS(15:14) Source register from CM|OD|SB|SG
2	AVD(V)S:=	XX CM	D	d d d d						Load AS(V), VS(V), and DS(V) Selection register from CM|OD|SB|SG

MICROOPERATIONS FOR Bus Shifter, BS

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION	
2	BSS:=		M	BSS:=	M	BSS:=	dd			Load the BS Selection register with dd, dd='CM'|'OD'|'SG'|	
2								M	BSS +1	Increment the BS Selection register	Θ
2								M	BSS -1	Decrement the BS Selection register	Θ
2								M	BSSC	Clear the BS Selection register	
					D	dddd				(THIS DATA IS REQUIRED WHENEVER THE BUSSHIFTER CONTROL IS USING CM AS DATA)	
2	BSP:=	ZZ CM	D	dddd						Load BS SG pointer from CM|OD|S1|S2	
2	BSP +1									Increment BS SG Pointer	
2	BSP -1									Decrement BS SG Pointer	
2	BSPC									Clear BS SG Pointer	
2	BSPS1:=	ZZ CM	D	dddd						Load BSP Save1 register from CM|OD|S1|S2	
1					M	BSPS2:=BSP				Load BSP Save2 register from BS SG Pointer	
1								M	BSSG:=SB	Load BS SG from SB(3:0)	

MICROOPERATIONS FOR Counter A, CA

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION	
2	CA:=	XX CM	D	ddddddd	D	ddddddd	dd	M	CA:=	Load CA from CM (16 bits), SB (16 bits), OD (16 bits), or CASG (16 bits)	
2	CA +1				M	CA +1		M	CA +1	Increment CA	
2	CA -1				M	CA -1		M	CA -1	Decrement CA	
2	CAC				M	CAC		M	CAC	Clear CA	
2	CAP:=	YY CM	D	dddd						Load the CA SG Pointer from CM|SB|S1|S2	Θ
2	CAP +1 Θ		M	CAP +1						Increment CA SG Pointer	
2	CAP -1 Θ		M	CAP -1						Decrement CA SG Pointer	
2	CAPC Θ		M	CAPC						Clear CA SG Pointer	
1			M	CASG:=CA				M	CASG:=CA Θ	Load CA SG from CA	
2	CAPS1:=	YY CM	D	dddd						Load CAP Save1 register from CM|SB|S1|S2	Θ
1					M	CAPS2:=CAP				Load CAP Save2 register from CA SG Pointer	Θ

MICROOPERATIONS FOR Counter B, CB

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION	
2	CB:=	VV CM	D	ddddddd	D	ddddddd	dd	M	CB:=	Load CB from CM (16 bits), SB (16 bits), or CBSG(16 bits)	
2	CB +1		M	CB +1				M	CB +1	Increment CB	
2	CB -1		M	CB -1				M	CB -1	Decrement CB	
2	CBC		M	CBC				M	CBC	Clear CB	
2					M	CBP:=	YY CM	D	dddd	Load the CB SG Pointer from CM|SB|S1|S2	Θ
2					M	CBP +1				Increment CB SG Pointer	
2					M	CBP -1				Decrement CB SG Pointer	
2					M	CBPC				Clear CB SG Pointer	
1	CBSG:=CB Θ				M	CBSG:=CB				Load CB SG from CB	
2					M	CBPS1:=	YY CM	D	dddd	Load CBP Save1 register from CM|SB|S1|S2	Θ
1			M	CBPS2:=CBP						Load CBP Save 2 register from CB SG Pointer	Θ

MICROOPERATIONS FOR Control Unit, CU

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3	S3	M D_4	F4	MICROOPERATION
1			M	SA:=SB						Load Save Address register from SB(11:0)
1								M	SA +1	Increment Save Address
1								M	SA -1	Decrement Save Address
1								M	SAC	Clear Save Address
1					M	CUALF:=		D	d d d d d	Load CU AL Function register with d d d d d
1					M	SET CU ALF +				Set CU AL Function register to A+B
1	SETCUALF B									Set CU AL Function register to B
1			M	RA ↑						Decrement RA Pointer
1	RA ↓		M	RA ↓	M	RA ↓				Increment RA Pointer and then Load RA
1			M	RAPC						Clear RA Pointer
1	RB ↑									Decrement RB Pointer
1	RB ↓		M	RB ↓	M	RB ↓				Increment RB Pointer and then Load RB
1	RBPC									Clear RB Pointer
1			M	CS LOAD						Load control store and then choose A+1 as the address of the next microinstruction
1			M	STOP A				M	STOP A	if the corresponding console switch A or B is turned on then stop execution else do nothing
1								M	STOP B	
1								M	CYL	Sets the mode of the processor to be in Long resp. Short cycle, starting with the execution of the next instruction.
1								M	CYS	
1	NOOP1		M	NOOP2	M	NOOP3		M	NOOP4	No Operation (dummy)
1					M	LCC				Clear Load Counter
2			M	WSMC						Master clear of Wide Store

MICROOPERATIONS FOR Switches KC, KD

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3		M D_4	F4	MICROOPERATION
1 1			M M	SETKC KCC						KC:= true KC:= false
S*	KC:=<SC>		M	KC:=<SC>						Load KC with the current Selected Condition
1 1								M M	SETKD KDC	KD:= true KD:= false
S*	KD:=<SC>							M	KD:=<SC>	Load KD with the current Selected Condition
	S* = special depending on short or long cycle to give the value of the condition used in sequencing.									

Microinstruction tables

MICROOPERATIONS FOR Double Shifter, DS

7	2	1	7	1	7	2	1	7

<table>
<tr><th>Cp</th><th>F1</th><th>S1</th><th>M/D2</th><th>F2</th><th>M/D3</th><th>F3</th><th>S3</th><th>M/D4</th><th>F4</th><th>MICROOPERATION</th></tr>
<tr><td>2</td><td>DS(1:0)S :=</td><td>XX
CM</td><td>D</td><td>d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load DS(1:0) Source register from CM|OD|SB|SG</td></tr>
<tr><td>2</td><td>DS(15:14)S :=</td><td>XX
CM</td><td>D</td><td>d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load DS(15:14) Source register from CM|OD|SB|SG</td></tr>
<tr><td>2</td><td>DS(V)S :=</td><td>XX
CM</td><td>D</td><td>d d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load DS(V) Selection register from CM|OD|SB|SG</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>M</td><td>DSLL</td><td>Set the DS to logical left shift</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>M</td><td>DSLR</td><td>Set the DS to logical right shift</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td>M</td><td>DS(V)SC</td><td></td><td></td><td></td><td>Clear DS(V) Selection register</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td>M</td><td>DS(V)S +1</td><td></td><td></td><td></td><td>Increment DS(V) Selection register</td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td>M</td><td>DS(V)S -1</td><td></td><td></td><td></td><td>Decrement DS(V) Selection register</td></tr>
</table>

MICROOPERATIONS FOR Input Port A, and Input Port B, IA and IB

7	2	1	7	1	7	2	1	7

<table>
<tr><th>Cp</th><th>F1</th><th>S1</th><th>M/D2</th><th>F2</th><th>M/D3</th><th>F3</th><th>S3</th><th>M/D4</th><th>F4</th><th>MICROOPERATION</th></tr>
<tr><td>1</td><td>IAD:=</td><td>EE
CM</td><td>D</td><td>d d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load IA Device register from CM|OD|SB</td></tr>
<tr><td>2</td><td>IAA</td><td></td><td>M</td><td>IAA</td><td></td><td></td><td></td><td>M</td><td>IAA</td><td>Activate device, i.e. initiate read</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IADC</td><td></td><td></td><td></td><td></td><td></td><td>Clear IA Device register</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IAD +1</td><td></td><td></td><td></td><td></td><td></td><td>Increment IA Device register</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IAD -1</td><td></td><td></td><td></td><td></td><td></td><td>Decrement IA Device register</td></tr>
<tr><td>1</td><td>IBD:=</td><td>EE
CM</td><td>D</td><td>d d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load IB Device register from CM|OD|SB</td></tr>
<tr><td>2</td><td>IBA</td><td></td><td>M</td><td>IBA</td><td></td><td></td><td></td><td>M</td><td>IBA</td><td>Activate device, i.e. initiate read</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IBDC</td><td></td><td></td><td></td><td></td><td></td><td>Clear IB Device register</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IBD +1</td><td></td><td></td><td></td><td></td><td></td><td>Increment IB Device register</td></tr>
<tr><td>1</td><td></td><td></td><td>M</td><td>IBD -1</td><td></td><td></td><td></td><td></td><td></td><td>Decrement IB Device register</td></tr>
</table>

MICROOPERATIONS FOR Loading Mask Registers A, LA

7	2	1	7	1	7	2	1	7

<table>
<tr><th>Cp</th><th>F1</th><th>S1</th><th>M/D2</th><th>F2</th><th>M/D3</th><th>F3</th><th>S3</th><th>M/D4</th><th>F4</th><th>MICROOPERATION</th></tr>
<tr><td>2</td><td>LAP :=</td><td>ZZ
CM</td><td>D</td><td>d d d d</td><td></td><td></td><td></td><td></td><td></td><td>Load LASG Pointer from CM|OD(15:12)|S1|S2</td></tr>
<tr><td>2</td><td>LAP +1</td><td></td><td>M</td><td>LAP +1</td><td></td><td></td><td></td><td>M</td><td>LAP +1</td><td>Increment LASG Pointer</td></tr>
<tr><td>2</td><td>LAP -1</td><td></td><td>M</td><td>LAP -1</td><td></td><td></td><td></td><td>M</td><td>LAP -1</td><td>Decrement LASG Pointer</td></tr>
<tr><td>2</td><td>LAPC</td><td></td><td></td><td></td><td></td><td></td><td></td><td>M</td><td>LAPC</td><td>Clear LASG Pointer</td></tr>
<tr><td>2</td><td>LAPS1 :=</td><td>ZZ
CM</td><td>D</td><td>d d d d</td><td></td><td></td><td></td><td>M</td><td>LAPS1:=</td><td>Load LAP Save1 register from CM|OD|S1|S2</td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td>M</td><td>LAPS2:=LAP</td><td></td><td></td><td></td><td>Load LAP Save2 register from LA Pointer</td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>M</td><td>LA := SB</td><td>Load LA from SB(15:0)</td></tr>
</table>

MICROOPERATIONS FOR Loading Mask Registers B, LB

Cp	F1	S1	M/D	F2	M/D	F3	S3	M/D	F4	MICROOPERATION
	7	2	1	7	1	7	2	1	7	
2					M	LBP :=	ZZ CM	D	d d d d	Load LBSG Pointer from CM¦OD(7:4)¦S1¦S2
2			M	LBP +1	M	LBP +1		M	LBP +1	Increment LBSG Pointer
2			M	LBP -1	M	LBP -1		M	LBP -1	Decrement LBSG Pointer
2					M	LBPC		M	LBPC	Clear LBSG Pointer
2			M	LBPS1:=	M	LBPS1:=	ZZ CM	D	d d d d	Load LBP Save1 register from CM¦OD¦S1 S2
1	LBPS2:=LBP									Load LBP Save 2 register from LB Pointer
1			M	LB := SB						Load LB from SB(15:0)
2			M	LPC						Clear LASG and LBSG pointer

MICROOPERATIONS FOR Local Registers, LR

Cp	F1	S1	M/D	F2	M/D	F3	S3	M/D	F4	MICROOPERATION
	7	2	1	7	1	7	2	1	7	
2	LRIP := DS									Load LR Input Pointer with DS(V+1:V)
2	LRIP +1									Increment LR Input Pointer
2	LRIP -1									Decrement LR Input Pointer
2	LRIPC									Clear LR Input Pointer
2								M	LROP := DS	Load LR Output Pointer with DS(V+1:V)
2								M	LROP +1	Increment LR Output Pointer
2								M	LROP -1	Decrement LR Output Pointer
2								M	LROPC	Clear LR Output Pointer
2			M	LRP := DS	M	LRP := DS				Load both LRIP and LROP with DS(V+1:V)
2			M	LRPC	M	LRPC				Clear both LRIP and LROP
2			M	LRP +1	M	LRP +1				Increment both LRIP and LROP
2			M	LRP -1	M	LRP -1				Decrement both LRIP and LROP

MICROOPERATIONS FOR Bus Mask Registers, MA and MB

Cp	F1	S1	M/D	F2	M/D	F3	S3	M/D	F4	MICROOPERATION
	7	2	1	7	1	7	2	1	7	
2	MAP :=	XX CM	D	d d d d				M	MAP :=	Load MA Pointer from CM¦OD¦SB¦SG
2	MAP +1		M	MAP +1				M	MAP +1	Increment MA Pointer
2	MAP -1		M	MAP -1				M	MAP -1	Decrement MA Pointer
2	MAPC		M	MAPC				M	MAPC	Clear MA Pointer
2	MBP :=	XX CM	D	d d d d				M	MBP :=	Load MB Pointer from CM¦OD¦SB¦SG
2	MBP +1				M	MBP +1		M	MBP +1	Increment MB Pointer
2	MBP -1				M	MBP -1		M	MBP -1	Decrement MB Pointer
2	MBPC				M	MBPC		M	MBPC	Clear MB Pointer
2					M	BMP :=	ZZ CM	D	d d d d	Load BM SG Pointer from CM¦OD¦S1¦S2
2					M	BMP +1				Increment BM SG Pointer
2					M	BMP -1				Decrement BM SG Pointer
2					M	BMPC				Clear BM SG Pointer
2			M	BMPS1 :=	M	BMPS1 :=	ZZ CM	D	d d d d	Load BMP Save1 register from CM¦OD¦S1¦S2
1	BMPS2 := BMP									Load BMP Save2 register from the BMPP
1			M	BMSG:=SB						Load BM SG with SB(3:0)

MICROOPERATIONS FOR MDP-transport

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3		M D_4	F4	MICROOPERATION
					D	d d d d				THIS DATA IS REQUIRED WHENEVER BUS SHIFTER IS ENABLED AND BSS='CM' (=0)
			D	d d d d d						THIS DATA IS REQUIRED WHENEVER MASK GENERATOR IS ENABLED AND PGS='CM'(=0)

MICROOPERATIONS FOR Output Ports A, B, C, and D, OA, OB, OC, and OD

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M D_2	F2	M D_3	F3	S3	M D_4	F4	MICROOPERATION
1					M	OAD:=	EE CM	D	d d d d	Load OA Device register from CM|OD|SB
2	OAA	d			M	OAA		M	OAA	Activate device i.e. initiate write with DM_1 := d
2			M	OAR						Deactivate device (Reset)
1					M	OAD +1				Increment OA Device register
1					M	OAD -1				Decrement OA Device register
1					M	OADC				Clear OA Device register
1					M	OBD:=	EE CM	D	d d d d	Load OB Device register from CM|OD|SB
2	OBA	d			M	OBA		M	OBA	Activate device i.e. initiate write with DM_1 := d
2			M	OBR						Deactivate device (Reset)
1			M	OBD +1						Increment OB Device register
1			M	OBD -1						Decrement OB Device register
1			M	OBDC						Clear OB Device register
1					M	OCD:=	EE CM	D	d d d d	Load OC Device register from CM|OD|SB
2	OCA	d			M	OCA		M	OCA	Activate device i.e. initiate write with DM_1 := d
2			M	OCR						Deactivate device (Reset)
1					M	OCD +1				Increment OC Device register
1					M	OCD -1				Decrement OC Device register
1					M	OCDC				Clear OC Device register
1	OC:=BUS ⊖		M	OC:=BUS						Load OC from BUS(15:0)
1					M	ODD:=	EE CM	D	d d d d	Load OC Device register from CM|OD|SB
2	ODA	d	M	ODA				M	ODA	Activate device i.e. initiate write with DM_1 := d
2			M	ODR						Deactivate device (Reset)
1			M	ODD +1						Increment OD Device register
1			M	ODD -1						Decrement OD Device register
1			M	ODDC						Clear OD Device register
1			M	OD:=BUS						Load OD from BUS(15:0)

MICROOPERATIONS FOR Postshift Masks (PA, PB) and PMSG

7	2	1	7	1	7	2	1	7

C_p	F1	S1	MD2	F2	MD3	F3	S3	MD4	F4	MICROOPERATION
2	PAP:=	XX CM	D	d d d d						Load PA Pointer from CM|OD|SB|SG
2	PAP +1				M	PAP+1		M	PAP +1	Increment PA Pointer
2	PAP -1				M	PAP-1		M	PAP -1	Decrement PA Pointer
2	PAPC							M	PAPC	Clear PA Pointer
1					M	PA:=BUS				Load PA RG from BUS(15:0)
2	PBP:=	XX CM	D	d d d d						Load PB Pointer from CM|OD|SB|SG
2	PBP +1							M	PBP +1	Increment PB Pointer
2	PBP -1							M	PBP -1	Decrement PB Pointer
2	PBPC							M	PBPC	Clear PB Pointer
1					M	PB:=BUS				Load PB RG from BUS(15:0)
2					M	PMP:=	ZZ CM	D	d d d d	Load PM SG Pointer from CM|OD|S1|S2
2					M	PMP +1				Increment PM SG Pointer
2					M	PMP -1				Decrement PM SG Pointer
2					M	PMPC				Clear PM SG Pointer
2			M	PMPS1:=	M	PMPS1:=	ZZ CM	D	d d d d	Load PMP Save1 register from CM|OD|S1|S2
1	PMPS2:=PMP									Load PMP Save2 register from PMSG Pointer
1			M	PMSG:=SB						Load PMSG from SB(3:0)
2					M	PABC				Clear PA and PB Pointer

MICROOPERATIONS FOR Postshift Mask Generator, PG

7	2	1	7	1	7	2	1	7

C_p	F1	S1	MD2	F2	MD3	F3	S3	MD4	F4	MICROOPERATION
2					M	PGS:=	dd			Mask Generator Control Source Selection register is set to dd, dd='CM'|'OD'|'SG'
2			M	PGS +1				M	PGS +1	Increment PG Selection register
2			M	PGS -1				M	PGS -1	Decrement PG Selection register
2			M	PGSC				M	PGSC	Clear PG Selection register
0			D	d d d d d						THIS DATA IS REQUIRED WHENEVER THE MASK GENERATOR CONTROL IS USING CM AS DATA
2					M	PGP:=	ZZ CM	D	d d d d	Load PG SG Pointer from CM|OD|S1|S2
2					M	PGP +1				Increment PG SG Pointer
2					M	PGP -1				Decrement PG SG Pointer
2					M	PGPC				Clear PG SG Pointer
2			M	PGPS1:=	M	PGPS1:=	ZZ CM	D	d d d d	Load PG Save1 register from CM|OD|S1|S2
1	PGPS2:=PGP									Load PG Save2 register from PGP
1			M	PGSG:=SB						Load PG SG from SB(4:0)

MICROOPERATIONS FOR Main Store Address, MSA

7	2	1	7	1	7	2	1	7

C_p	F1	S1	MD_2	F2	MD_3	F3		MD_4	F4	MICROOPERATION
2	MSA :=	XX CM	D	d d d d d d d	D	d d d d d d d	d d			Load MSA from CM|OD|SB|SG
2	MSA + 1		M	MSA + 1				M	MSA + 1	Increment MSA
2	MSA – 1		M	MSA – 1				M	MSA – 1	Decrement MSA
2	MSAC		M	MSAC				M	MSAC	Clear MSA
2	MSAP :=	ZZ CM	D	d d d d						Load MSA SG Pointer from CM|OD|S1|S2
2	MSAP + 1									Increment MSA SG Pointer
2	MSAP – 1									Decrement MSA SG Pointer
2	MSAPC									Clear MSA SG Pointer
1	MSASG := MSA		M	MSASG:=MSA				M	MSASG:=MSA	Load MSA SG from MSA
2	MSAPS1 :=	ZZ CM	D	d d d d				M	MSAPS1 :=	Load MSAP Save 1 register from CM|OD|S1|S2
1					M	MSAPS2:= MSAP				Load MSAP Save 2 register from MSA SG Pointer

MICROOPERATIONS FOR Variable Width Shifter, VS

7	2	1	7	1	7	2	1	7

C_p	F1	S1	MD_2	F2	MD_3	F3	S3	MD_4	F4	MICROOPERATION
2	VS(0)S :=	XX CM	D	d d d				M	VS(0)S :=	Load the VS(0) Source register from CM|OD|SB|SG
2	VS(15)S :=	XX CM	D	d d d				M	VS(63)S :=	Load the VS(15) Source register from CM|OD|SB|SG
2	VS(V)S :=	XX CM	D	d d d d				M	VS(V)S :=	Load the VS(V) Selection register from CM|OD|SB|SG
2			M	VSLL						Set the VS to a logical left shift
2			M	VSLR						Set the VS to a logical right shift
2	VS(V)SC									Clear the VS(V) Selection register
2	VS(V)S +1									Increment the VS(V) Selection register
2	VS(V)S –1									Decrement the VS(V) Selection register

MICROOPERATIONS FOR Working Registers, WA

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION
2	WAU :=	XX CM	D	d d d d						Load WA Unit pointer from CM|OD|SB|SG
2	WAU +1							M	WAU +1	Increment WA Unit pointer
2	WAU -1							M	WAU -1	Decrement WA Unit pointer
2	WAUC							M	WAUC	Clear WA Unit pointer
2					M	WAG :=	XX CM	D	d d d d	Load WA Group pointer from CM|OD(7:4)|SB(7:4)|SG
2					M	WAG +1				Increment WA Group pointer
2					M	WAG -1				Decrement WA Group pointer
2					M	WAGC				Clear WA Group pointer
2	WAP :=	XX CM	D	d d d d			XX CM	D	d d d d	Load WA Unit pointer from CM|OD|SB|SG AND load WA Group pointer from CM|OD|SB|SG
2	WAPC									Clear WA Unit pointer and WA Group pointer *)
1								M	WAPCOUPLE	Couple WA Unit and Group pointers to form an 8 bit counter
1								M	WAP-UNCOUPLE	Uncouple WA Unit and Group pointers to form two independent 4 bit counters

*) WAP +1 is equivalent to WAU +1, and WAP -1 to WAU -1, assuming that the unit and group pointers are coupled.

MICROOPERATIONS FOR WA Unit and Group Standard Groups, WAUS and WAGS

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION	
1	WAUS:=WAU ⊖							M	WAUS:=WAU	Load WA Unit SG with WAU	
2					M	WAUSP:=	YY CM	D	d d d d	Load WAUS Pointer from CM|SB|S1|S2	⊖
2								M	WAUSP +1	Increment WA Unit SG Pointer	
2								M	WAUSP -1	Decrement WA Unit SG Pointer	
2								M	WAUSPC	Clear WA Unit SG Pointer	
2					M	WAUSPS1:=	YY CM	D	d d d d	Load WAUSP Save 1 register from CM|SB|S1|S2	⊖
1			M	WAUSPS2:= WAUSP						Load WAUSP Save2 register from WAUSP	⊖
1	WAGS:=WAG							M	WAGS:=WAG ⊖	Load WA Group SG with WAG	
2	WAGSP:=	YY CM	D	d d d d						Load WAGSP from CM|SB|S1|S2	⊖
2	WAGSP +1									Increment WA Group SG Pointer	
2	WAGSP -1									Decrement WA Group SG Pointer	
2	WAGSPC									Clear WA Group SG Pointer	
2	WAGSPS1:=	YY CM	D	d d d d						Load WAGSP Save1 register from CM|SB|S1|S2	⊖
1	.				M	WAGSPS2:= WAGSP				Load WAGSP Save 2 register from WAGSP	⊖
1					M	WAPS:=WAP ⊖		M	WAPS:=WAP	Load WA Unit and WA Group SG with WAU and WAG respectively	
2	WAPSP:=	YY CM	D	d d d d			YY CM	D	d d d d	Load WAUPSP and WAGSP from CM|SB|S1|S2	⊖
2								M	WAPSP +1	Increment WA Unit and WA Group SG Pointers	
2								M	WAPSP -1	Decrement WA Unit and WA Group SG Pointers	
2								M	WAPSPC	Clear WA Unit and WA Group SG Pointers	
2	WAPSPS1:=	YY CM	D	d d d d			YY CM	D	d d d d	Load WAUSP and WAGSP Save1 registers from CM|SB|S1|S2	⊖
1								M	WAPSPS2:= WAPSP	Load WAUSP and WAGSP Save2 registers from WAUSP and WAGSP respectively	⊖

MICROOPERATIONS FOR Working Registers, B, WB

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION
2					M	WBU :=	XX CM	D	d d d d	Load WB Unit pointer from CM|OD(11:8)|SB(11:8)|SG
2			M	WBU +1				M	WBU +1	Increment WB Unit pointer
2			M	WBU -1				M	WBU -1	Decrement WB Unit pointer
2			M	WBUC				M	WBUC	Clear WB Unit pointer
2	WBG :=	XX CM	D	d d d d						Load WB Group pointer from CM|OD(15:12)|SB(15:12)|SG
2			M	WBG +1						Increment WB Group pointer
2			M	WBG -1						Decrement WB Group pointer
2			M	WBGC						Clear WB Group pointer
2	WBP :=	XX CM	D	d d d d			XX CM	D	d d d d	Load WB Unit pointer from CM|OD(11:8)|SB(11:8)|SG AND load WB Group pointer from CM|OD(15:12)|SB(15:12)|SG
2			M	WBPC						Clear WB Unit pointer and WB Group pointer *)
1					M	WBPCOUPLE				Couple WB Unit pointer and Group pointers to form an 8 bit counter
1					M	WBP-UNCOUPLE*				Uncouple WB Unit pointer and Group pointer to form two independent 4 bit counters

*) WBP +1 is equivalent to WBU +1, and WBP -1 to WBU -1, assuming that the unit and group pointers are coupled.

MICROOPERATIONS FOR WB Unit and Group Standard Groups, WBUS and WBGS

7	2	1	7	1	7	2	1	7

C_p	F1	S1	M/D_2	F2	M/D_3	F3	S3	M/D_4	F4	MICROOPERATION	
1			M	WBUS:=WBU	M	Θ WBUS:=WBU				Load WB Unit SG from WBU	
2	WBUSP:=	YY CM	D	d d d d						Load WBUS Pointer from CM|SB|S1|S2	Θ
1			M	WBUSP +1						Increment WB Unit SG Pointer	
2			M	WBUSP -1						Decrement WB Unit SG Pointer	
2			M	WBUSPC						Clear WB Unit SG Pointer	
2	WBUSPS1:=	YY CM	D	d d d d						Load WBUSP Save1 register from CM|SB|S1|S2	Θ
1								M	WBUSPS2:= WBUSP	Load WBUSP Save2 register from WBUSP	Θ
1			M	Θ WBGS:=WBG	M	WBGS:=WBG				Load WB Group SG from WBG	
2					M	WBGSP:=	YY CM	D	d d d d	Load WBGS Pointer from CM|SB|S1|S2	Θ
2					M	WBGSP +1				Increment WB Group SG Pointer	
2					M	WBGSP -1				Decrement WB Group SG Pointer	
2					M	WBGSPC				Clear WB Group SG Pointer	
2					M	WBGSPS1:=	YY CM	D	d d d d	Load WBGSP Save1 register from CM|SB|S1|S2	Θ
1	WBGSPS2:= WBGSP									Load WBGSP Save2 register from WBGSP	Θ
1			M	Θ WBPS:=WBP	M	WBPS:=WBP				Load WB Unit and WB Group SG with WBU and WBG respectively	
2	WBPSP:=	YY CM	D	d d d d			YY CM	D	d d d d	Load WBUS and WBGS Pointers from CM|SB|S1|S2	Θ
2					M	WBPSP +1				Increment WB Unit and WB Group SG Pointers	
2					M	WBPSP -1				Decrement WB Unit and WB Group SG Pointers	
2					M	WBPSPC				Clear WB Unit and WB Group SG Pointers	
2	WBPSPS1:=	YY CM	D	d d d d			YY CM	D	d d d d	Load WBUSP and WBGSP Save1 registers from CM|SB|S1|S2	Θ
1								M	WBPSPS2:= WBPSP	Load WBUSP and WBGSP Save2 registers from WBUSP and WBGSP respectively	Θ
2								M	WCU + 1	Increment WAU and WBU	
2								M	WCU - 1	Decrement WAU and WBU	
1			M	WCUS:=WCU						Load WA and WB Unit SG from WAU and WBU	
1					M	WCGS:=WCG				Load WA and WB Group SG from WAG and WBG	

3.2. Condition tables.

Conditions.

AL(0)	DS(0)	KA	VS(0)
AL(15)	DS(1)	KB	VS(15)
AL	DS(2)	KC	VS(V)
ALOV	DS(3)	KD	WA(0)
AS(0)	DS(4)	LR(0)	WA(15)
AS(15)	DS(5)	LR(15)	WACS
AS(V)	DS(6)	MSAB	WAGOV
BP	DS(7)	MSAOR	WAGSPOV
BUS	DS(8)	MSASPOV	WAPOV
BUSPAR	DS(9)	OASA	WAPSPOV
CA(3)	DS(10)	OBSA	WAUOV
CA(4)	DS(11)	OCSA	WAUSPOV
CA(5)	DS(12)	ODSA	WB(0)
CA(6)	DS(13)	ONE	WB(15)
CA	DS(14)	RAPOV	WBCS
CASPOV	DS(15)	RAPUN	WBGOV
CB(3)	DS(V)	RBPOV	WBGSPOV
CB(4)	DS(V+1)	RBPUN	WBPOV
CB(5)	FALSE	SB(0)	WBPSPOV
CB(6)	IADA	SB(1)	WBUOV
CB	IADM	SB(14)	WBUSPOV
CBSPOV	IBDA	SB(15)	ZERO
	IBDM	TRUE	

4. Rikke I/O Ports.

OA and IA are dedicated ports for MainStore.

OB and IB have full device selection, and the various devices are connected presently as follows :

OB:	IB:
0: CS	0: INPUTSTATUS
1: TTYMULTIPLEXER	1: TTYMULTIPLEXER
2: DEC10	2: DEC10
3: PRINTER	3: PTR2
4: DISKIN	4: DISKOUT
5: CONSOLE	5: TIMER
6: DISKSEEKCTRL	6: DISKSTATUS
7: DISKRWCTRL	7: MULTIPLEX STATUS
8: EXTERNAL	8: OUTPUTSTATUS
9: MINIPRINTER	9: KEYBOARD
10: PTP	10: PTR1
11: MATHILDAOUT	11: MATHILDAIN

12-15: WIDE-STORE PORTS (in and out)

OC is dedicated to WideStore Control.

OD is used as input register to various selectors.

5. 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

[9]: Flemming Wibroe
Running a microprogram on RIKKE-MATHILDA
DAIMI MD-41, October 1980

Kjoergard, Jens Kristian.
The Rikke processor--described as Rikke-Mathilda differences / by Jens Kristian Kjoergard.-- Aarhus, Denmark: Computer Science Department, Aarhus University, 1980.
(DAIMI; MD-43)

I. Title.