
Title:

RC 8005 CPU Test

Users Manual



RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No:	30-M 58
Edition:	May 1978
Author:	Flemming Hansen Finn Grønning

Keywords:

RC 8000, CPU Test, Diagnostic Test

Abstract:

This manual describes two CPU testprograms, which test the CPU 801 microprogram and the CPU 802, CPU 803 hardware respectively.

Contents:

	Pages
0. Introduction.	2
0.1. Changes from previous version.	
0.2. Configuration, applicable documents.	
0.3. Storage and addressing.	
0.4. Parameter specification.	
1. Extended RC 4000 test (test a).	7
1.1. Testing strategy.	
1.2. Core layout.	
1.3. Looping.	
1.4. Limitregister test.	
1.5. Test errormessages.	
1.6. Examples test a.	
2. CPU microprogram test (test b).	4
2.1. Purpose.	
2.2. Testing strategy.	
2.3. Test errormessages.	
2.4. Operator messages.	
2.5. TCP messages.	
Appendix 1. CPU801 logic diagrams (test b)	3
Appendix 2. Interrupt 0 treatment (test b)	2
Appendix 3. Escape treatment (test b)	2
Appendix 4. Survey of testloop content (test b)	5

0. Introduction.

The CPU-test consist of two diagnostic testprograms, where Test a is an extended RC 4000 test, and Test b is a RC 8000 microprogram test.

0.1. Changes from previous version.

Test a: Can now run in a 32K memory.
Extended with a limit register test.

Test b: All test loops are included.

0.2. Configuration, applicable documents.

The configuration required is the minimum, on which the test system can be operated.

The following documents may be used for reference:

RC 8005 Technical Manual, RCSL: 30-M73

RC 8000 Testprogram System, Users Manual, RCSL: 30-M124

Testprogram listing, RCSL: 30-M127

RC 4000 Diagnostic Program for CPU, program description

RCSL: 51-VB 1172.

0.3. Storage and addressing.

The current version of "testadm" loads the testprogram from physical address 6166. The size of the CPU-test is 44 segments.

The parameter segment is loaded from cell 5654-6164. The first cell in this is addressed as logical address 20, that is, the testprogram operate with an addressing base of 5634. ($=5654-20$)

Lower limit is set to 5654, so any obstruction of the testadm due to test-malfunction is avoided.

0.4. Parameter specification.

Testprogram a (extended RC 4000 test)

paramno	text	default	min	max
000	select testprogram	b	a	b
001	number of runs=	10000	1	2xx23-1
004	errormessages?	yes	-	-
006	logical insertaddress=	0	0	2xx23-1
007	octal contents=	15100000	-	-

Testprogram b (RC 8000 microprogram test)

paramno	text	default	min	max
000	select testprogram:	b	a	b
001	number of runs=	10000	1	2xx23-1
003	tcp on?	no	-	-
004	errormessages?	yes	-	-
005	loop no.=	- 1	- 1	679

1. Extended RC 4000 test (test a).

This test is the RC 4000 CPU diagnostic test adjusted for the RC 8000.

As a consequence, the checkout is not exhaustive, but in many cases relevant diagnostic information can be obtained.

It is extended with a limit register test (test 81).

1.1. Testing strategy.

The testprogram is designed on the "start small" principle. However, this is partially irrelevant, because the testadm has to be running, to load and start the CPU testprogram.

The first test will be test 7, as the testnumbers are the same as in RC 4000.

It is checked, if the correct test sequence is obeyed, and deviations from the chronological order implies the error message: "SEQUENCE ERROR: <TEST IDENTIFIER>"

If an error is detected an errormessage will be printed on selected output.

For test 7-80 it is not possible to suppress the errormessage by setting the parameter "errormessages?" to no. It is, however, possible to loop in a specific test, with or without errormessages, by means of the parameters "logical insert address=" and "octal contents=".

1.2. Core layout

Logical addr.	
578	Entry test a Procedures for initialization and output
1230	Testprograms 7 to 8
1816	Central actions for interrupt error selection of next testprogram operator communication
3672	Testprograms 9 to 80
9540	Testprogram 81 Limit register test
11324	

1.3. Looping.

By means of the two parameters

P006 logical insertaddress (decimal) =

P007 octal contents (8 digits) =

it is possible to make the machine run in a specific loop. For example, the operator may insert a JL. -14 instruction in cell 5506 to exercise subtest 2 in test 30.

This is done by setting:

P006 logical insertaddress (decimal) = 5506

P007 octal contents (8 digits) = 15107762 (= JL. -14)

Then the testprogram must be restarted by the "start" command, because the inserting is performed in the beginning of the test.

In this way it is possible to overwrite as many words in the test as wanted, as long as only the "start" command is used. The original test may be called upon with a "new" command.

The testprogram listing, RCSL: 30-M127 will give the logical insertaddress.

1.4. Limitregister test (test 81)

The limit register- and comparator circuits on the CPU803 will be tested in the same way as the CDU801 (checkout and diagnostic unit) does.

This means that certain predefined digital errors at IC-level will be detected. An erroneous position is localized within a few MSI circuits.

Test errormessages:

If an error is detected, an errormessage, telling the erroneous circuit position, is printed on the selected output device. Further the last testpattern used, is repeated with breakpoints enabled. This feature might give some diagnostic information in case, that other errors is detected by the limitregister test.

First time an error is detected, a guidance in the interpretation of the breakpoints, is given on the selected output device.

The test errormessages will be skipped if the parameter "errormessages?" is set to no.

It is possible to loop in test 81 by help of

P006 logical insertaddress (decimal) = 9660

P007 octal contents (8 digits) = 15107652 (JL. -86)

Loop if error is possible by help of

P006 logical insertaddress (decimal) = 10064

P007 octal contents (8 digits) = 15107236 (JL. -354)

1.5. Test errormessages:

"sequence error: < Test identifier >"

the test named by < Test identifier > deviate from the chronological order, in which the different test's are executed.

"< letter >: < 24-bit pattern >"

Error in test 7. < letter > is the name of the subtest, which has failed, and the pattern gives the result of the test.

"< number >: < 24-bit pattern >"

Error in test 8. < number > is the name of the subtest, which has failed, and the pattern gives the result of the test.

"< 0 - ext >: < received result of 12 ext SB(12) con SB(12:23) >"

"< 1 - ext >: < received result of 12 ext SB(12) con SB(12:23) >"

Error in test 9, in either 0 or 1 extention.

"< program name >"

Error in test 10-22. The name of the test and evt. the subtest number and result will be printed.

"< test no > : < test identifier > < subtest no >

received: < received pattern >

expected: < expected pattern >"

Error in test 23-80. The testnumber, the name of the test and the subtest number is printed together with the received and expected pattern.

"LL-error in CPU 803 around pos. < pos. no >"

"UL-error in CPU 803 around pos. < pos. no >"

Error in test 81. Lower Limit or Upper Limit error.
The pos. no of the comparator is printed.

****Breakpoint No: < no >

< register > = < Decimal > < octal > < binary >"

Error in test 81. Each error will cause a repetition with breakpoints enabled.

Might be used for diagnostic purpose, in case that other errors is detected by the limitregister test.

1.6. Examples test a.

RC8000 CPU TEST FEB. 78

```

000 SELECT TESTPROGRAM: A
001 NUMREP OF RUNS = 1
004 ERRORMESSAGES? YES
006 LOGICAL INSERTADDRESS (DECIMAL) = 0
007 OCTAL CONTENTS (8 DIGITS) = 15100000

```

LL-ERROR IN CPU 803 AROUND POS. 87

```

***BREAKPOINT NO:      1
W0  =      1  0000 0001  ..... 1
W1  =      1  0000 0001  ..... 1
W2  = -524288  7600 0000  11111...
W3  = -1048576  7400 0000  1111....

```

```

***BREAKPOINT NO:     -1004
W0  =     -1004  7777 6024  11111111 111111.. ...1.1..
W1  =      1  0000 0001  ..... 1
W2  = -529922  7576 4776  1111.111 111.1..1 1111111..
W3  =      8  0000 0010  ..... 1
STATUS= -8388602  4000 0006  1..... 11..
TC   =     9904  0002 3260  ..... ..1..11. 1.11....
CAUSE =      26  0000 0032  ..... ..11.1.
SB   =    -2022  7777 4032  11111111 11111.. ...11.1.
CPA  =      0  0000 0000  .....
BASE =     5634  0001 3002  ..... ..1.11. ....1.
LLC  =     5654  0001 3026  ..... ..1.11. ....1.11.
HLC  =    262144  0100 0000  .....1..
TLFVC =    24635  0006 0073  ..... .11..... ..111.11

```

LL-ERROR IN CPU 803 AROUND POS. 87

DIAGNOSTIC INFORMATION:

AFTER AN ERRORMESSAGE IS PRINTED ON THE OUTPUT DEVICE,
THE LAST TESTPATTERN IS REPEATED WITH BREAKPOINTS ENABLED.
USE THE BREAKPOINTS IN THE FOLLOWING WAY:

1. LOOK AT BREAKPOINT NO 1 AND NOTICE THE IOADDR AND LIMIT.
W2:=IOADDR, W3:=LIMIT.
2. FROM IOADDR AND LIMIT DECIDE WHICH BREAKPOINTS, IF ANY,
YOU EXPECT TO FOLLOW.

IOADDR IS FIRST COMPARED TO LL=LIMIT, UL=MAX (MAX=-1).
IOADDR IS THEN COMPARED TO LL=MIN, UL=LIMIT (MIN=0).

E A C H COMPARISON RESULTS IN ONE OF THE FOLLOWING ACTIONS:

BREAKPOINT NO -1000: INTERRUPT NO 0 AS A CONSEQUENCE OF LIMITVIOLATION
BREAKPOINT NO -1003: INTERRUPT NO 3 AS A CONSEQUENCE OF LIMITVIOLATION
BREAKPOINT NO -1004: INTERRUPT NO 4, BUS TIME OUT BECAUSE IOADDR > CORE SIZE
NO BREAKPOINTS: NO LIMITVIOLATION OR BUS TIME OUT

IF THE ERROR AGAIN ARISE, THE SAME ERRORMESSAGE THAT LEAD UP TO
THE REPETITION, WILL BE PRINTED ON THE OUTPUT DEVICE.

2. CPU microprogram test (test b)

2.1. Purpose

The test examines the microprogram address in the CPU 801. This address may originate from different sources (see appendix 1):

1. The address table prom for address calculation (this table is selected by the microprogram modification A = next instruction).
2. The address table prom for instruction execution (this table is selected by the microprogram modification x = execute instruction).
3. The microaddress register (last address incremented by one).
4. The subreturn register.
5. The jump address register.

The address table proms content is not hardware checked for parity, so the address may be degenerated and thereby select an erroneous part of the microprogram.

In the first part of the test (testloops 1 - 256), the address calculation prom is exercised and in the next part (testloops 257 - 512, 599 - 679), the instruction execute prom is exercised. Simultaneously the address sources in 3., 4. and 5. is examined.

It is not possible to predict the errorreaction due to an erroneous prom-content because of the $2 \times 12 = 4096$ possible output combinations for each position (there are $2 \times 8 = 256$ positions), so an error may cause a strange behaviour of the testprogram.

2.2. Testing Strategy.

The test is divided into 679 independent loops. Selection of the next loop is determined by the "loop no. = " parameter or, if the "tcp on?" parameter is yes, by the switch-setting on the DSWR on the TCP 801. This "loop no. = " or DSWR value is called loopno, and it determines the mode of the operation.

loopno = -1:

The testloops are executed one by one, starting with loopno = 1 and continuing until loopno = 679. This is repeated until "no of runs" parameter is reached. The test continues even if an error is detected.

loopno = 0:

The testloops are executed one by one, starting with loopno = 1 and continuing until an error is detected. The loop in which the error is detected will now be run in loopmode.

loopno = 1,2,...,n,...,679:

The testloop no n will be executed in loopmode.

If a loopno greater than 679 is specified, the message "loopno too big" is printed. Now a correct loopno should be specified.

If an error is detected an errormessage will be printed on selected output. This errormessage may be suppressed if the parameter "errormessages?" is set to no. If the "tcp on?" parameter is set to yes, the regsel (4) on the TCP 801 determines the printing of errormessages. Regsel (4) = 0 corresponds to errormessages = yes.

2.3. Test errormessages.

Four kinds of errormessages from the test is given:

1. "testno:< testno>, sequenceerror"

An illegal testno is selected. In the continue mode, the new testno is expected to be the old one incremented by one. This is examined in each testloop.

2. "testno:< testno >, resulterror, w0 = < wreg0 >, w1 = < wreg1 >"

An escape or interrupt action has not been executed successfully. This action should increment the instruction counter (=IC) and bypass the call of an errormessage routine. (in the program called "error1")

3. "testno:< testno >, unwanted interrupt no: < no >, IC = < ic >, SB = < sb >"

In the instruction pointed out by IC - 2 and SB, an illegal interrupt has occurred. The interrupt number is given.

4. "testno:< testno >, unwanted escape call, CAUSE = < cause >, IC = < ic >, SB = < sb >"

In the instruction pointed out by IC and SB, an illegal escape call has occurred. Cause consists of the function field and the working register in the above mentioned instruction.

	Function code						wreg	
cause:	16	17	18	19	20	21	22	23
	128	64	32	16	8	4	2	1

After the test messages 1, 3 and 4 the test is reinitialized.

2.4. Operator messages.

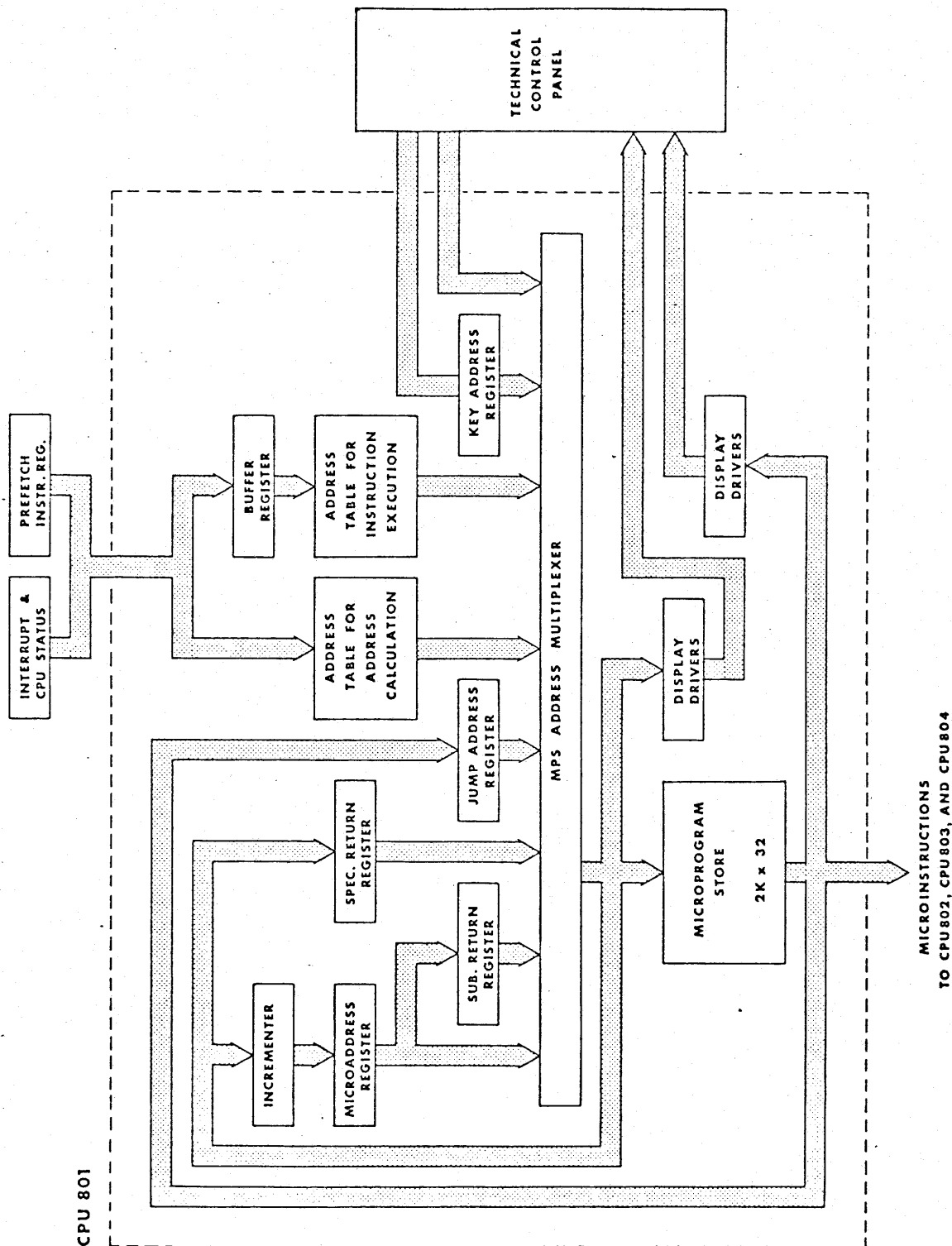
"xxx loopno too big"

A testno greater than 679 has been specified.

2.5. TCP messages.

The actual testno is continuously displayed on the TCP 801. This may be a great help if the test, due to an error, is terminated without giving any test error messages.

Appendix 1. CPU801 logic diagrams (test b)



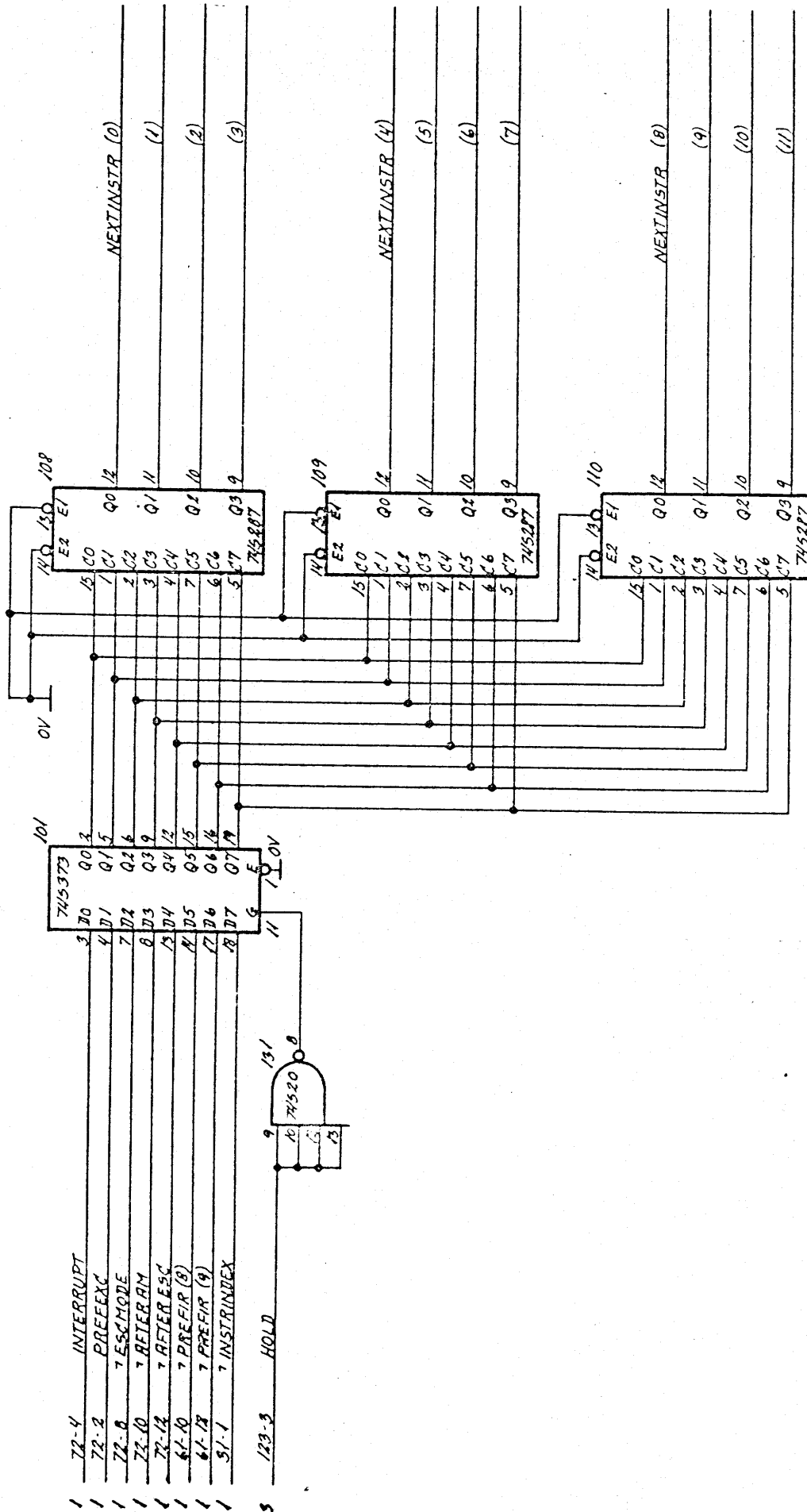
CPU801 Block diagram

R 11385

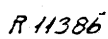
760V23 ARJ 760V23 ROB

CPU801

R12015



ADDRESS TABLE FOR START OF NEXT INSTRUCTION



ADDRESS TABLE FOR START OF INSTRUCTION EXECUTION

Appendix 2. Interrupt 0 treatment. (test b)

Detailed description of interrupt no zero treatment.

The interrupt no zero is treated as an escape action. When the microprogram detects an illegal instruction a microprogrammed interrupt-0 routine is called. This routine reads the contents of INF +3. (INF is the information register, general register no 15). In the program, the contents of INF +3 is equal to int-0-reg%, where % means the starting address of the array int-0-reg (0:10). The microprogram now stores w0, w1, w2, w3, status, ic, cause and sb into int-0-reg (0), int-0-reg (1),, int-0-reg (7), and then starts execution of slang instructions from int-0-reg (8).

int-0-reg	(0) = w0
	(1) = w1
	(2) = w2
	(3) = w3
	(4) = STATUS
	(5) = IC
	(6) = CAUSE
	(7) = SB
	(8) = RL. w2 + 4
	(9) = JL. INT-0-ACTION.
	(10) = STACKBASE

fig. use of int-0-reg.

In int-0-reg (8) w2 is reinitialized to stackbase (= startaddress of variables in the language PATAC). Hereafter int-0-reg (9) is executed, and it is a jump to the int-0-action procedure.

INT-0-ACTION

If this procedure is entered due to an unwanted interrupt the error message:

"testno: < testno >, unwanted interruptno: < no >, IC = < ic >,
SB = < sb > will be printed.

If the interrupt was expected, 2 is added to int-0-reg (5) (=ic) thereby bypassing the instruction following the instruction, which caused the int-0. The bypassed instruction is a call of an error routine (error1) printing the message:

"testno: < testno >, resulterror, w0 = < wreg0 >, w1 = < wreg1 >".

Now all bits in int-0-reg (4) (= STATUS) except monitormode is cleared.

Hereafter the RE. INT-0-REG. (return from _escape) instruction is executed. This instruction restores w0, w1, w2, w3, STATUS, IC, CAUSE, SB with the content of int-0-reg (0:7) and continues the execution as pointed out by IC.

Appendix 3. Escape treatment. (test b)

Detailed description of escape treatment.

The microprogram enters an escaperoutine if the three conditions mentioned below is fulfilled:

1. The escape mode bit is set in CPUT.
2. At least one of bits 6 - 11 in STATUS is set.
3. An instruction is executed with a type belonging to one of the bits 6 - 11 in status.

The microprogrammed escaperoutine reads the content of the information (INF) register +5. In the CPU test INF +5 is initialized to escapereg % or escape-1-reg%, where % means the starting address of the array escapereg (0:10) or escape-1-reg (0:10). The microprogram stores w0, w1, w2, w3, status, ic, cause, sb in escapereg (0:7) or escape-1-reg (0:7) and then starts execution of slang instructions in reg (8).

escapereg (0) = w0	escape-1-reg (0) = w0
(1) = w1	(1) = w1
(2) = w2	(2) = w2
(3) = w3	(3) = w3
(4) = STATUS	(4) = w4
(5) = IC	(5) = IC
(6) = CAUSE	(6) = CAUSE
(7) = SB	(7) = SB
(8) = RL. w2 + 4	(8) = RL. w2 + 4
(9) = JL. ESCAPEROUTINE.	(9) = JL. ESCAPE-1-ROUTINE.
(10) = STACKBASE	(10) = STACKBASE

fig. use of escapereg (0:10) and escape-1-reg (0:10).

In reg (8) w2 is reinitialized to stackbase (=startaddress of variables in the language PATAc).

Hereafter reg (9) is executed, and it is a jump to either the escaperoutine or the escape-1-routine procedure.

Escaperoutine or escape-1-routine

If this procedure is entered due to an unwanted escape call the following message is printed:

"testno: < testno >, unwanted escape call, CAUSE = < cause >, IC = < ic >, SB = < sb >".

If the escape was expected, then

1. escape-1-routine

escape-1-reg (5) (=ic) is incremented by 4, thereby bypassing the next instruction (call of the error1 procedure) and status (6:11):= all ones and all other bits are cleared.

Hereafter a return from escape (RE. ESCAPE-1-REG.) to escape-1-reg is executed.

2. Escaperoutine

All escapereg (4) (=status) bits are cleared except monitormode, after escape and after am bits.

Hereafter a return from escape (RE. ESCAPEREG.) to escapereg is executed.

The RE. ESCAPEREG. restores w0, w1, w2, w3, STATUS, IC, CAUSE, SB with the content of escapereg (0:7) and continues the execution as pointed out by IC.

Appendix 4. Survey of testloop content (test b)

Test 1 - 256 is a test of the address calculation prom

testno

1	IXR	I = indirect
2	IR	R = relative
3	XR	X = index
4	R	D = direct
5	IX	
6	ID	
7	X	
8	D	

address

9 - 16	address roms	48 - 55	after escape
17 - 24	"	40 - 47	after am
25 - 32	"	32 - 39	after escape+after am
33 - 40	"	24 - 31	escape mode
41 - 48	"	16 - 23	escape mode+after escape
49 - 56	"	8 - 15	escape mode+after am
57 - 64	"	0 - 7	escape mode+afteram+ after escape
65 - 128	"	64 - 127	prefetch exceptions
129 - 192	"	128 - 191	external interrupt
193 - 256	"	192 - 255	prefetch exc+ext interrupt

Test 257 - 512 is a test of the execute instruction prom

testno

address

257 - 320	execute proms	64 - 127	escapemode+monitormode
321 - 384		193 - 255	escapemode
		0 - 63	monitormode
		128 - 192	normal

last testno corresponds to execution in monitormode

385, 449	00
386, 450	DO
387, 451	BL
388, 452	HL
389, 453	LA
390, 454	LO
391, 455	LX
392, 456	WA
393, 457	WS
394, 458	AM
395, 459	WM
396, 460	AL
397, 461	RI
398, 462	JL
399, 463	JD
400, 464	JE
401, 465	XL
402, 466	BS
403, 467	BA
404, 468	BZ
405, 469	RL
406, 470	SP
407, 471	RE
408, 472	RS
409, 473	WD
410, 474	RX
411, 475	HS
412, 476	XS
413, 477	GG
414, 478	DI
415, 479	30
416, 480	31
417, 481	CI
418, 482	AC
419, 483	NS
420, 484	NO
421, 485	AS
422, 486	AD
423, 487	LS
424, 488	LO
425, 489	SH
426, 490	SL
427, 491	SE
428, 492	SN
429, 493	SO
430, 494	SZ
431, 495	SX

432, 496	GP
433, 497	FA
434, 498	FS
435, 499	FM
436, 500	51 = KS
437, 501	FD
438, 502	CF
439, 503	DL
440, 504	DS
441, 505	AA
442, 506	SS
443, 507	58
444, 508	59
445, 509	60
446, 510	61
447, 511	62
448, 512	63

testloops 513 - 598 are not used.

testno	microaddress
599	0106 - 0114
600	0425 - 0433
601	0154, 01521, 01522, 10524
602	0174
603	0201
604	0207
605	0214
606	0226, 0233, 0222
607	0253
608	0257 - 0265
609	0302 - 0310
610	0311 - 0312
611	0326 - 0332
612	0346 - 0352
613	0363 - 0364
614	0375 - 0376
615	0407 - 0410
616	0414 - 0413
617	0434 - 0436
618	0437
619	0447 - 0450
620	0456 - 0455
621	0456

testno	microaddress
622	0466 - 0467
623	0472 - 0474
624	0475
625	0553 - 0555
626	0556
627	0557 - 0560
628	0567 - 0575
629	0600 - 0603
630	0614 - 0617
631	0610
632	0631 - 0633
633	0654 - 0656
634	0663 - 0667
635	0670 - 0676
636	0677 - 0700
637	0702
638	0707
639	0731
640	0732
641	0733
642	0736
643	0745
644	0753, 0755
645	0770 - 01002
646	01003 - 1004 - 01016 - 01023
647	01066 - 1077
648	01040 - 01051
649	01121 - 01130
650	01131 - 01135
651	01143
652	01243 - 01253 - ,01262 - 01263
653	01260, 01264 - 01270
654	01275
655	01375
656	01467 - 01466, 01516
657	01467 - 01473
658	01523
659	01666 - 01673, 01634
660	01774 - 01775, 01674 - 01677, 01625
661	01703 - 01710
662	01711 - 01714
663	01715 - 01716
664	01734 - 01736

testno	microaddress
665	01747 - 01752
666	01772 - 01773
667	02005 - 02006
668	02012 - 02016
669	02051 - 02053
670	02113 - 02115, 2504 - 2506, 2102 - 2105
671	02176 - 02205, 02143 - 02145
672	02146 - 02152, 02174
673	02155 - 02164, 02175
674	02171 - 02173
675	02250 - 02257, 02372 - 02373, 02375 - 02377
676	02313 - 02323, 02344 - 02345
677	02367 - 02371
678	02420
679	02773 - 02775