TITLE:

CR80 AMOS
MASTERCLEAR UTILITIES (AMU)
USER's MANUAL

DOCUMENT NO : CSS/395/USM/0040

PREPARED BY : FINN RITSLEV

AUTHORIZED BY: FINN RITSLEV

DISTRIBUTION : SW-group, GB, EB, KM, JA (test)

APPROVED BY:

AUTHORITY	DATE	SIGNATURE
Jørgen Høg		

ISSUE: 2

DATE: 811101

400-810-1

CR80 AMOS, MASTERCLEAR UTILITIES (AMU) USER's MANUAL	sign/date FR/811101	page
	repl	project

PAGE RECORD AND ISSUE LOG.

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
67								
68								
69								
70								
71								
72								
73								
74								
75								
76								
77								
78								
79								
80								
81								
82								
83								
84								
85								
86								
87								
88								
89								
90								
91								
92								
93								
94								
95								
96								
97								
98								
99								
100								

ISSUE	DATE	PREPARED BY	APPROVED BY	AUTHORIZED BY
1	791023	FR	FR	JHØ
2	811101	FR	FR	JHØ

A

LIST OF CONTENTS

1. SCOPE

The purpose of this document is to describe the use of the CR80 AMOS Masterclear Utilities (AMU) program product. The AMU program is identified as CSS/395.

The AMU is a PROM-resident CR80 program which provides the user of the CR80 system with a range of utilities which can be used both stand-alone and in connection with other programs. The program supports CR80 configurations with one or more CPU's, though only one is master at any given time. The utilities are intended to be used by both programmers and maintenance personnel.

The following set of utilities is available to the user:

- Automatic Start-Up Facilities
- Boot direct/indirect via DMA from Floppy Disk
- Copy from one memory area to another
- Dump memory area
- Copy and Execute
- IO Utilities:
  - read
  - write
  - sense
  - control
- Load direct/indirect via DMA from Floppy Disk
- Memory Check
- Options (onlay applicable to IO/CPU test)
- Patch
- Switch CPU
- Return Calling Process
- Search Pattern (Incl. Mask)
- Negated Search
- Test CPU (only if CSM/100 is present)
- Unit Mapper
- Wait for Interrupt
- Execute
- Set Memory Parity
- Instruction Loop Facility

The document has been organized as follows:

Chapter 2 contains a general overview as well as the general syntax rules which apply to the AMU program. Chapter 3 contains a detailed description of each command. Section 3.1 of tis chapter describes general commands, i.e. commands whose utilization destroy but a limited part of memory. Section 3.2 describes special commands, i.e. commans which use a larger part of memory or other CPU-resources. Lastly section 3.3 provides a description of the automatic start-up facilities provided by the AMU. Chapter 4 gives a short AMU reference list.

Chapter 5 contains more specific uses of the AMU progfram as well as the description of how other programs may link to the AMU program.

This document applies from version o4 of the AMU program.

2. OVERVIEW AND SYNTAX RULES

2.1 Overview

Upon master clear the AMU program shall write the following on the Operator's Console:

```
CSS/395/04      811001
CPU 0
>
```

or after an emergency action

```
EMERGENCY ACTION BY 0200
>
```

The AMU program will now be awaiting the first command from the user.

If automatic mode from eg. a discette has been selected, no user input is expected and the first of the above "messages" will not be printed out. In this case boot load will automatically be attempted from the selected device.

In a multiprocessor configuration the CPU which is fastest will be selected as master CPU, while the remaining will be "parked" with base ≠ 0020 waiting for a CPU interrupt. This implies that locations ≠ 0020 through ≠ 0032 are destroyed by the first CPU interrupt.

The AMU program requires the following device addresses:

1 - AV24 I/F or SCM operator's console
2 - Floppy Disk I/F (might be overrided)
7 - DMA I/F to "external" system

Only locations # 0200 through # 0221 inclusive shall be destroyed as long as the user restricts himself to the set of general commands:

C - Copy
D - Dump
I - I/O - Utilities
M - Memory Check
O - Options
P - Patch
R - Return Calling Process
S - Search
X - Execute
Z - Set Memory Parity

While all other memory locations are left unchanged.

The remaining utilities except the CPU-test will claim from # 0000 through # 0247 inclusive.

The CPU-test will further claim # 1000 through # 1FFF, which will contain a copy of the CPU-testprogram.

The detailed memory allocation is shown on figure 2-1.

```
# 0000.............     Wait_for_interrupt
:
# 0003

# 0007.............     Units/Switch_cpu (Q)
:
# 000F

# 0020.............     Units/Switch_cpu (Q)/dummy process
:
# 003F

# 0100.............     Boot/Load buffer area
:
# 013F

# 0180.............     Wait_interupt (Interrupt Table)
:
# 01C0.............     Dummy waiting process
:
# 01DF

# 0200.............     AMU process base
:                           master clear/emergency action
:
# 0222....                  Boot/load/test_cpu
:
# 0247

# 0300.............     Test_cpu program work jparea
:
# 1000.............     Test-cpu program (RAM resident copy)
:
# 1FFF

# F800.............     Prommed AMU program
:
# FFE0.............     Entry prpc to AMU program
:
# FFFE.............     Program address emergency address
# FFFF
```

Figure 2-1. AMU MEMORY ALLOCATION

2.2 General Syntax Rules

A command to the AMU program consists of one character followed by command dependent parameters.

In the detailed description of the commands in the succeeding sections, the following definitions are used:

<command>::=	B/C/D/I/E/L/M/O/P/Q/R/S/T/U/W/X/Z/*/
<page>::=	0/1/2/3 is the page (memory section)
<hexa_numbers>::=	<hexa_number> <delimiter><hexa_numbers>
<hexa_number>::=	<hexa> <hexa_number> if more than four hexa characters are entered only the last four are considered valid. If less than four hexa characters are entered zeroes are assumed preceeding the entered characters. <DELETE> or <RUB_OUT> following immediately after a <hexa_number> will result in its cancellation.
<hexa>::=	<digit>/A/B/C/D/E/F
<digit>::=	0/1/2/3/4/5/6/7/8/9
<delimiter>::=	Any non-hexadecimal character except <CR>
<blinds>::=	<blind> <blinds>
<blind>::=	<SP>/<NULL>
<SP>::=	is the space character (ASCII 32/#20)
<NULL>::=	is the null character (ASCII 0)
<CR>::=	is the carriage return character which is used as end of command (ASCII 13/#0D)

<starting-address>::= <hexa_number> .<off-set>
the resulting start address is made up as the sum of the hexa_number (the base-address) and the off-set.

<off-set>::= <hexa_number> .<off-set>

<size>::= <hexa_number>
hexadecimal count specifying the size of the memory area under consideration. Note that +0 implies 64K, not zero.

<repeats>::= <hexa_number>_ <hexa_number>
interpreted as a long integer

<BREAK>::= pressing BREAK during a print-out or the CPU-test will result in the cancellation of the command in question and the program will be waiting for its next command

::=. optional parameter

3. INDIVIDUAL COMMANDS DESCRIPTION

3.1 General Commands

This chapter describes those commands which can be used without destroying but the lower ≠ 0022 words of the current AMU process.

- copy
- dump
- IO utilities
- memory check
- patch
- return calling process
- search (incl. negated search)
- set parity
- instruction loop facility

Each section is organized as follows:

- description
- remarks
- syntax
- examples

3.1.1 Copy

Description

The Copy utility enables the user to copy one memory area to another desrespective of their respective page (memory section).

Remarks

None

Syntax

```
  <page>                                        +<size>
C   0    <sp><source_starting_address> +1

                                  /page
     sp<object_start_addres> /0
```

<source_start_address>::= <base_address> .<offset>

<object_start_address>::= <base_address>

Examples

- create copy of AMU process descriptor

```
>C FFE0+20  400
>
```

- copy AMU program to page 1 starting from location 0

```
>C F800+800 0/1
```

3.1.2 Dump

Description

The Dump Utility enables the user to dump a specified area on the Operator's Console. Areas might be from any memory section (page).

A facility is included which allows the same memory location to be re-read a specified number of times.

Remarks

The utility will allways round the starting address downwards to an address divideable by 16. This will not have any impact on the ending address.

The parity has to be set properly, otherwise a local action with cause 2 will result from dumping a memory location with parity error.

Syntax

D <page> (0) <sp> <start_address> <sp> <hexa_end_address> (+<size> (+1))

<hexa_end_address>::= <hexa_number>

dump same memory location

D <page><sp><start_address>*<repeats>

(print-out may be disable by means of the command OPTIONS prior to this command)

Examples

- dump relative

```
>D F800.7E0+20

FFE0  FDA8  0020  0000  0200  0000  FDE0  0000  0000
FFE8  0200  0200  F800  FD7D  7FFF  E000  FFE0  0030
FFF0  0000  FFFF  0000  8000  0000  FFFF  0000  0000
FFF8  1000  0000  0000  0000  0000  0000  F854  0000
>
```

- dump registers

```
>D 207
0200  FDA8  0020  0000  0200  0000  FDE0  0000 0000
>D 20.3.4
()
0200  FDA8  0020  0000  0200  0000  FDE0  0000 0000
>D 200+7
()
0200  FDA8  0020  0000  0200  0000  FDE0  0000 FDE0 0000
>D 200 207
()
0200  FDA8  0020  0000  0200  0000  FDE0  0000 0000
>D 200.7 207
()
0200  FDA8  0020  0000  0200  0000  FDE0  0000 0000
>
```

- dump other page

```
>D1 7FE0+20

7FE0  FDA8  0020  0000  0200  0000  FDE0  0000  0000
7FE8  0200  0200  F800  FD7D  7FFF  E000  FFE0  0030
7FF0  0000  FFFF  0000  8000  0000  FFFF  0000  0000
7FF8  1000  0000  0000  0000  0000  0000  F854  0000
>
```

<- repeatitive dump

```
>P 1000+E000 0
>P 1000.8.4x8_0 AAAA
>D 1000.E

1000  0000  0000  0000  0000  0000  0000  0000  0000
1008  0000  0000  0000  0000  AAAA  0000  0000
>D 1000.Ex6
0000  0000  0000  0000  0000  0000
>D 1000.Cx4
AAAA  AAAA  AAAA  AAAA
>0
PRINT OUT ?>N
NO OF TESTS ><CR>
>D 1000.Cx8_0

>O
PRINT OUT  ?>Y
NO OF TESTS >1
>
```

3.1.3 IO-Utilities

Description

The IO-Utilities allows the operator directly to exercise any IO-module connected to the CR80-system. All IO-commands can be issued:

Read	R
Write	W and V
Sense	S
Control	C and K

For Read and Sense the program prints the result while the operator for the remaining IO-utilities has to specify the data to be output to the device. For W and C the AMU will repeat outputting the entered hexa pattern. For V and K the operator will have to enter <repeat> hexa pattern which then one by one are executed by the AMU.

Remarks

It is possible to disable the print-out of Read and Sense results by activating 'Options' prior to the IO-Utilities. This is a useful tool to technical personnel in test situations.

Pressing break will terminate the test sequence.

Syntax

I R S <sp><modify_device_repeats><delimiter>

I W C <sp><modify_device_repeats><sp><hexa_pattern><delimiter>

I V C <sp><modify_device_repeats><sp><hexa_pattern><delimiter>

I V K <sp><modify_device_repeats><sp><hexa_pattern><delimiter>

<modify_device_repeats>::=

(modify_pattern)<sp>+<repeats> <device_number> +<repeats> +1

<modify_pattern>::= <hexa_number>

<repeats>::= <hexa_number>_<hexa_number>

Examples

```
>IO-S 1<delimiter 3F>033F
>IO-S 1+7<CR>
033F 033F 033F 033F 033F 033F 033F
>IO-W 1+2_0 30
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000
>
```

- supress output

```
>O
PRINT-OUT ? N
NO OF TESTS 1
>IO-S 1+F_0
()

>O
PRINT-OUT ? Y
NO OF TESTS 1
>IO-S 1 033F
>
```

- multiple IO

```
>IO-W 1+1_0
>IO-W 1+4_0 41
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
>IO-R 1 FF20
>IO-R 1WFF57
>IO-R 1+4WFF57 FF57 FF57 FF57
>
```

3.1.4 Memory Check

Description

The Memory Check Utility allows the operator to verify the proper functioning of any RAM; though, only a simple test is performed.

The utility works non-destructive.

Remarks

This utility can not be interrupted.

The verification on one memory section takes a few minutes.

Syntax

```
  <page>                          +<size>
M         <sp><start_address>
    0                             +1

     <sp><test_pattern1><sp> <test_pattern2> <CR>
              5555                  AAAA
```

<test_pattern$_1$>::= <hexa_number>: the primary pattern is inserted and verified in all locations of the specified area.

<test_pattern$_2$>::= <hexa_number>: the secondary pattern is inserted and verified in every second location of the specified area while the primary test pattern is used in the remaining.

Examples

```
>M 0+B
PARITY ERROR AT 0000
PARITY ERROR AT 0004
PARITY ERROR AT 0006
>Z, MEM PARITY ERROR
M 0+1000 CCCC 3333
M 0+1000 BEBE
>
```

3.1.5 Patch

Description

The Patch Utility enables the user to patch (modify) the content of any memory location (RAM-resident) from Operator's Console.

The utility includes a multipatch facility to enable initialization of the specified area with the operator defined pattern.

A facility is included which allows in same memory location to be re-written a specified number of times with the same pattern.

Remarks

The multipatch facility is a very convenient way of initializing a given area with the same pattern.

Negated search may be used to determine discrepancies from the above pattern.

Note in this context that +0 indicates 64K, i.e. a full memory section (page).

Syntax

P <page> / 0 <sp><start_address> <sp><hexa_numbers> + <size><sp><hexa_number><delimiter>

re-write

p<page><sp><start_address>*<repeats><sp><pattern><delimiter>

Examples

- conventional Patch

```
>P 408  400  400  F800  FDA8
>
```

or

```
>P 400.8  400  400  F800  FDA8
>
```

- multipatch: Fill page 1 with # CCCC

```
>P1 0+0 CCCC
>
```

- re-write location # 100C 300.000 times with AAAA's

```
>P 1000+E000 0
>P 1000.84+8_9 AAAA
>D 1000.E

1000  0000  0000  0000  0000  0000  0000  0000  0000
1008  0000  0000  0000  0000  AAAA  0000  0000
>
```

3.1.6 Return Calling Process

Description

The Return Utility will reload any calling process provided the link still exists. The link is retained by the following utilities.

C	-	Copy
D	-	Dump
I	-	IO-utilities
M	-	Memory-Check
O	-	Options
P	-	Patch
S	-	Search
X	-	Execute
Z	-	Set-Parity

Remarks

The utilities of AMU might be envoked by generating a process in RAM (48 words) and usign the contents of register 0 (XR0) of the prom-resident built-in AMU process descriptor located at # FFE0 as PRPC (program counter). (Refer Section 5.1 Linkage with other programs). It is the responsibility of the user to keep himself to the above mentioned utilities as others will either claim parts of memory outside the allowed process descriptor and/or destroy the link.

Syntax

R

Examples

```
>R
CSS/395/04 811101
CPU 0
>
```

3.1.7 Search

Description

The Search Utility enables the user to search for a specified binary pattern in any memory section. All memory locations which contain the specified pattern will be printed.

A negated search facility is included.

The search may include a mask which is and'ed to the memory contents before the comparison is made.

Remarks

Aside from the usefulness of searching for a given pattern the utility can be search for parity errors; the operator in that case searches the whole page for any infrequent test pattern.

Syntax

```
                                        +<size>
S  <page>  <sp><start_address>
     0                                  +1

                        <sp><hexa_pattern>/<mask>
```

negated search

```
                                        +<size>
S  <page>  <sp><start_address>
     0                                  +1

                        <sp><hexa_pattern> /0
```

Examples

- search upper 4K of page 0 for ≠ FFE0

```
>S F000+1000
FFE0 MATCH AT FFEE
>
```

- search page 1 for parity errors

```
>S1 0+1+ 8642

LOCAL INTERRUPT, CAUSE = 0002
>
```

- masked search

```
>P 200+EE00  FFFF
>P 7777 FFFE  FFEE  FEEE  EEEE
>S 1000+E000  0E00/0F00
MATCH AT 7779
MATCH AT 777A
>S 1000+E000  00E/F
MATCH AT 7777
MATCH AT 7778
MATCH AT 7779
MATCH AT 777A
>
```

3.1.8 Set Parity

Description

The Set_Parity Utility resets the parity of all RAM locations of memory. The utility works non-destructive. If any parity error has been detected this will be stated.

Remarks

Note that the Boot and Load Utilities automatically resets the parity when activated prior to any load from the DMA or floppy disk.

Syntax

Z

Examples

```
>Z, MEM PARITY ERROR
>Z
>
```

3.2 Special Commands

This chapter describes those commands which destroy other than the basic AMU-memory claims:

- Copy and execute
- Boot from floppy discette (via DMA)
- Load from floppy
- Test CPU
- Unit mapper
- Wait for interrupt
- Execute
- Switch CPU

Each section of this chapter is organized as follows:

- Description
- Remarks
- Syntax
- Examples

3.2.1 Copy and Execute

Description

The AMU program supports activation of other programs. The aim of this utility is to enable other loaders to co-exist with AMU.

Remarks

The external program to be activated is subject to the following constraints:

- The program part must be the top of the module
- The first word must be the "MODC -1"-instruction, which is used by the automatic boot-loader to verify existence of the external module
- Word with offset AMUPROCESS (## 0004) in the external module must contain the offset to the built-in process descriptor
- Word with offset AMUPROCESS (## 0005) in the external module must contain the size of the external module

The AMU copies the external program module from the designated location to the memory specified as new prog (for auto load the memory just below the AMU itself).

The contents of the built-in process descriptor are modified as follows:

REG3	new_base
REG5	new_prog (AMU_program_start-program_size)
XBASE	new_base
XM0D	new_base
XPROG	new_prog
XPRPC	new_prog+(old_prpc-old_prog)

The copied program is activated by the LDP-instruction.

AMU itself follows the above conventions.

This utility provides a means for copying e.g. the AMU itself to RAM for patching or debugging purposes.

Syntax

```
   <page>
E          <sp><program_start>
     0
           <sp><new_base>/<new_prog><delimiter>
```

Examples

create new AMU with prog = F000 and base = 100

```
>E F800  100/F000
CSS/395/04  811101
CPU 0, DISK ERROR = 0800, DISK ERROR = 0800?
>P F7E4 0
>X F7E0
CSS/395/04  811101
CPU 0
>
```

3.2.2 Boot from Floppy Disk (Load and Execute)

Description

The Boot from Floppy Disk enables the user to load a boot file into memory either directly from the floppy disk or indirectly via a DMA-channel.

The loaded program is activated provided the load was successful.

If the load was not successful the status of the device is given.

Remarks

The Boot utility automatically sets the parity.

The boot file might start from # 0248 and upwards in page 0 (memory section 0) and from # 0000 in the three other pages. Crossing a page will result in continuation from # 0000 on the next page.

Syntax

```
                              0
   F                              1
                               to
B     #     <device_address>. 2
                              3
```

The first parameter specifies whether the load is

be direct (F-floppy) or indirect via a DMA-channel (X-external) while the second parameter specifies which is to be used.

Examples

```
>BOOT FD:  1, MEM PARITY ERROR, DISK ERROR = 0800
>BOOT FD:  0, START = 0248, BASE = 14E2

>BOOT FD:  0, MEM PARITY ERROR, START = 0248, BASE = 14E2
```

ERROR MESSAGES (FLOPPY DISK)

STATUS	DESCRIPTION
0001	Seek Complete - seek commands was executed successfuly
0002	ID Data Check - CRC failed to compare in the ID field
0004	Program error - invalid command
0008	System Error - Write and Read lines active simultaneously
0010	Data Check - CRC check failed on the data field
0020	ID Not Found - search of the ID-field was not successful after two revolutions
0040	Equipment Check - resident micro diagnostics failed
0080	Ext. Status not 0 - disk change, deleted record or overrun found
0100	Write Protect - selected drive is write protected
0200	No AM found - one of the following three types of address marks was not found: ID, record, or deleted record
0400	Track Overflow - index was detected with Write or Read control line active
0800	Drive Not Ready - selected drive is not ready
1000	Seek Error - upon completion of the Seek command, the track addreass does not compare
2000	Recalibration Error - track 00 was not detected in response to a Restore command
4000	Device Error - failure to sense track 00. During Write format command, this status code alone indicates that the discette rotational speed is too slow, if this status code occurs with Track Overflow, then the diskette rotational speed is too fast
8000	Write Error - index was detected while head write current was on
FFFF	AMU floppy driver looping more than 16 times in order to restore diskette. Try again -

ERROR MESSAGES (DISK)

STATUS	DESCRIPTION
0001	busy
0002	was seeking
0004	parity error
0008	unexpected drive status
0010	check or sync datafield
0020	check or sync address field
0040	address field
0080	sector status
0100	illegal sector
0200	illegal unit
0400	missing clock
0800	time out
1000	sector timing
2000	subbus overrun
4000	not used
8000	not used

ERROR MESSAGES (DMA)

STATUS	DESCRIPTION
0020	not end of block
0040	stop DMA
0080	time out remote
0100	time out local
0200	parity error

3.2.3 Load from Floppy Disk

The load from floppy disk enables the user to load a boot file into memory either directly from floppy disk or indirectly via a DMA-channel.

If the load was not successful the status of the device is given.

Remarks

The load utility automatically sets the parity.

The boot file might start from # 0248 and upwards in page 0 (memory section 0) and from # 0000 in the three other pages. Crossing a page will result in continuation from # 0000 on the next page.

Syntax

L F/X # <device_address> 0/1/2/3

The first parameter specifies whether the load is to be direct (F-floppy) or indirect via a DMA-channel (X-external) while the second parameter specifies which is to be used.

Examples

```
>LOAD FD:  1, MEM PARITY ERROR, DISK ERROR=0800
>LOAD FD:  0, START=0248, BASE=14E2

>LOAD FD: 0, MEM PARITY ERROR, START=248, BASE=14E2
```

Errors

Refer 3.2.2

3.2.4 Options

This facility provides a means for suppression of print-outs for the following utilities.

- Dump Repeatitive
- Memory Check
- IO-Utilities
- CPU-test

Furthermore it provides the means by which the number of CPU-tests to be performed is specified (as a long integer).

Remarks

None

Syntax

```
0
PRINT OUT?  Y/N
NO OF TESTS > <CPU_test_repeats>
```

where

<CPU_test_repeats::= <hexa>_ <hexa>

Examples

```
>0
PRINT OUT? N
NO OF TESTS  <CR>
>D 1000*F_000

>0
PRINT OUT? N
NO OF TESTS F_O
>T3 F000
CSM/100/01  791101
***--------------------*
>
```

3.2.5 Test CPU

Refer to CSM/100/USM (TBD)

```
>T3 F000
CSM/100/01  791101
************************************************
>T
************************************************
>
```

3.2.6 Unit Mapper

Description

The Unit Mapper utility maps the following units of a CR80 configuration.

- o CPU's
- o RAM/PROM (allocation of 4K modules)
- 0 IO-modules

Remarks

1 existing unit
. missing unit

Syntax

U

Example

```
                page 0    page 1          page 2      page 3
>U
CPU:  1.....
RAM:  11111111 1111111/11111111 11111111/......../......../
                                         ........ ........1/
IO:   111.11.. 1......./........ ......./........ ........./
                                         ........ ........./
                IO: 0-F          10-1F          20-2F    30-3F
```

3.2.7 Wait for Interrupt

Description

The Wait for Interrupt Utility enables the operator to verify proper action of a given I/O module to any previous IO-command.

The utility can wait for an interrupt from a specified device or from all devices.

Return to normal mode is obtained by break.

The AMU runs masked against I/O interrupts except during execution of this command.

Remarks

The utility is also useful in verifying proper action by the CPU.

Syntax

```
                <device_address>
W<sp>  All                        <CR>
```

device_address::= hexa_number, only the lower 6 bits are used, i.e. 0 device 63.

Examples

```
>W 2
IO interrupt, dev = 0002, prio = 0
IO interrupt, dev = 0002, prio = 0
>W

>W
IO interrupt, dev = 0002, prio = 0
IO interrupt, dev = 0002, prio = 0
```

3.2.8 Execute

Description

The Execute Utility enables the operator to start the execution of any process resident in memory.

Remarks

None.

Syntax

X <hexa_address><delimiter>

<hexa_address>::= base of process to be activated by a Load-process (LDN-instruction).

Examples

```
>X 200
CSS/395/01  791015
CPU 0
>X FFE0
CSS/395/01  791015
CPU 0
>
```

3.2.9 Switch-CPU

Description

The Switch-CPU Utility enables the operator to change from one CPU to any other provided the other runs the CR80 standard instruction set. The user will continue with the selected CPU waiting for the next user command.

Remarks

The specified CPU has to exist, as no checks what so ever is made by the utility.

Whether a given CPU exist or not can be verified by the unit-mapper U.

Which CPU is currently active may be determined by the command "X FFE0".

Syntax

Q<cpu-no>
<cpu-no>::= 0 1 2 3 4 5 6 7

Examples

```
>U
CPU: 11......
RAM: 11111111 11111111/11111111 ......../........ ......../........
     ......../
IO:  1111.... 1......./........ ......../........ ......../........
     ......../
>Q0
>Q1
>Q0
>Q1
>Q0
>
```

- illegal CPU
  - >Q5

3.3 Automatic Start-Up Facilities

AMU includes facilities which enables CO-existence with other more special loaders, e.g. disk loader or LTU down line loader.

Furthermore it allows automatic start-up via a prommed parameter:

- auto boot from floppy diskette
- auto boot from floppy diskette via DMA
- auto boot from external program module
- auto boot from floppy diskette, if unsuccessful from external program

External programs must obey the rules set forward in section 3.2.1 Copy and Execute program.

Remarks

Location # F808 of the AMU contains the program start address of 'external program', which has to be a multiple of 16 to which is added the page (default is page 3 memory # 0000).

REG2 of the built-in process descriptor contains the basic timing element used during initialization (default 256). It is essential that this element results in a timing delay which exeeds the internal CPU initialization of the slowest CPU.

REG4 of the built-in processes descriptor (i.e. location # FFE4). determines which boot option AMU will follow. The following options exsist:

0 - utilities
1 - floppy diskette drive 0, then 1
2 - indirect floppy diskette drive 1, then 0
3 - external loader located page 3 from memory location 0
4 - floppy diskette drive 0, then 1 then external loader.

4. AMU REFERENCE SHEETS

5. SPECIAL USE

5.1 Linkage with Other Programs

The utilities of the AMU program can be made available to the user also when the AMU itself is no longer active. In this mode of operation all general commands of the AMU program is available to the user at no risk for destroying but the area determined by the start of the allocated (new) AMU process plus # 0022 words. The AMU will however always have to be operated from the "master" terminal, i.e. the terminal connected to the AV24 if or SCM with module address 1 (default value).

Any other program can activate (call) the AMU program provided the following is followed:

1) the PROM'ed AMU process decriptor (# FFE0 - # FFFF) is copied to own process area and the base and modify registers adjusted accordingly.

2) the PRPC which should be used is contained as the first word of the AMU process descriptor, i.e. # FFE0.

3) The AMU program is activated by the LPD-instruction from the program.

4) Only the class of general commands should be used. Otherwise memory areas other than those intended are destroyed.

5) return to the calling process is done by the command "R" to the AMU.

Example - Test AMU Auto External Facility

```
CSS/395/04  811101
CPU 0, MEM PARITY ERROR, DISK ERROR = 0800, DISK ERROR = 0800?
>U
CPU: 1.......
RAM: 11111111 11111111/11111111 11111111/........ ......../........
                                                          ......../
IO:  1111..1. 11....../11..1... ......../........ ......../........
                                                          ......../

>C F800+800 0/1
>E F800 200/F000
CSS/395/04 811101
CPU 0, DISK ERROR=0800, DISK ERROR=0800?
>P F008 1
>P F7E4 3
>X F7E0
CSS/395/04 811101
CPU 0
CSS/395/04 811101
CPU 0, DISK ERROR=0800, DISK ERROR=0800?
>D 200+20

0200  FF04  0020  0100  0200  0001  E800  0000  0000
0208  0200  0200  E800  EE6F  7FFF  E000  0200  0042
0210  012D  0003  FFFF  8000  0001  FFFF  0000  0000
0218  1000  01E8  0111  01F0  01D4  0472  0472  0002
>
```

CR80 AMOS, MASTERCLEAR UTILITIES (AMU) USER's MANUAL	sign/date FR/811101	page
	replace	project