Title:

SOFTWARE PRODUCTION for IML701.

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 43-GL7966

Edition: Oktober 1978

Author: Dan Andersen

Keywords:

RC3600, Autoload, Image load, IML701, F102, software production description.

Abstract:

This manual is a description of how to generate stand alone programs or RC3600 systems for the IML701 (F102).

CONTENTS PAGE

1. GENERAL. 1.

The IML701 consist of a EPROM memory expandible in 2K 16 bits increments to 32K 16 bits words, which can be transferred to the RC3600 main memory at autoloade time.

An erased EPROM memory may also be programmed with the contents of the RC3600 main memory by applying external power and activation of the PROGRAM button.

This manual deals with the preparation of software, which is going to be programmed into the EPROM's from a core-image generated in the RC3600 main memory.

2. AUTOLOAD.

When the IML701 is requested, by the autoload program, to transfer data from the EPROM memory to the main memory with an I/O-START pulse, the data is transferred word by word through the data channel.

The transfer is started at main memory address 400_8, and continued to the maximum address defined by switch setting on the controller board. After this transfer, the page zero is filled from address 0 to 377_8 (see fig 2-1).

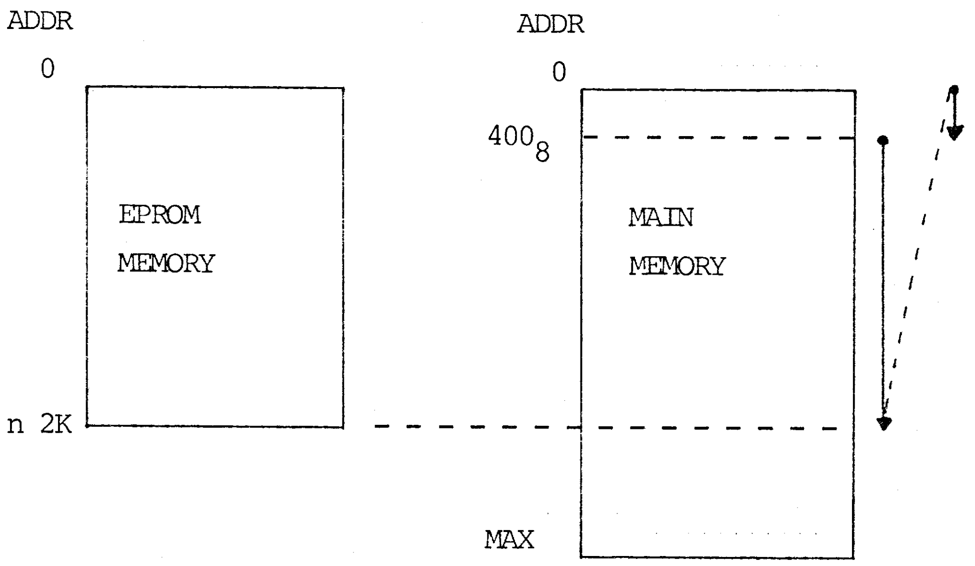


Fig 2-1: Transfer of EPROM memory to main memory from address 400_8 and on, and then from address 0 to 377_8.

Programs designed for autoload via DMA channel can then be used without modification, if only the device busy flag is tested in the delay loop, which delays the transfer of program control until the whole core image is read into the memory.

As word 377_8 is the last word written in the IML data transfer the transfer of program control is delayed by keeping the CPU busy, executing JMP 377 in location 377_8, until the last data word defines a new instruction in word 377_8.

3. CORE IMAGE GENERATION 3.

Before the programming of the EPROM's is started, a core image of the actual system/program must be created in the RC3600 main memory.

The core image is created from an absolute binary file which is the resulting output from an absolute assembler, a linkage editor or a relocatable loader.

As transfer of program control to the resulting system is not wanted before EPROM programming, the absolute binary output must be produced with a startblock which tells the core image generator (binary loader) not to transfer control to the loaded system.

The system programmed into the EPROM's must contain an instruction in address 377_8, which transfer program control to the system/program entry.

When the absolute binary file is generated the core image can be created in the main memory in two ways:

1) load of a papertape containing the system in absolute binary format, with the RC3600 Binary loader.

When the loader halts after input of the tape the CPU must be running in a dummy loop, as the data channel can not function if the CPU halts.

This can be done by insert of a dummy instruction JMP .+0 (400_8) in any unused memory location, and start of the CPU in this address by means of the technical panel.

Programming can then be started (Appendix B)

2) Abs. binary load of the system from a DOMUS disc file by means of the command BOOT <filename>.

   In this case the maximum size of the system allowed is approximately 60000_8 words, and if greater a size error is returned by the DOMUS absolute binary loader.

When the DOMUS system halts after the load the CPU must be running executing the dummy instruction JMP .+0 (400_8) in any unused memory word, as the data channel can not function if the CPU halts. This can be done by means of the technical panel before programming is started. (Appendix B)

4. STAND ALONE PROGRAMS. 4.

If the stand alone program is designed for DMA load, and fulfils the specifications given in section 3, the core image can be created as described.

If, however, word 377_8 in the program is not used as entry point some modifications must be carried out before use of the IML is possible.

The modifications can be done on the loaded core image by means of the technical panel, or on source level, but in this case must the absolute binary file be created from scratch.

The modifications necessary are shown in example 4-1.

```
ADDRESS (oct)

376          PIP            ; Address of new entry point
377          JMP @  .-1     ;
             .
             .
             .
             .
                            ; unused words:
      PIP:   LDA  3  ORG1;  new entry point
             STA  3   376;  restore memory word
             LDA  3  ORG2;  376 and 377
             STA  3   377;
             JMP    entry;  jump  to original entry
      ORG1:  XXX          ; original content
      ORG2:  YYY          ; of word 376 and 377
```

5. RC3600 MUS SYSTEMS.

The MUS system consist basically of a number of seperate relocatable binary files, which can be transformed to a single absolute binary file by means of a linkage editor program (fig 5-1)

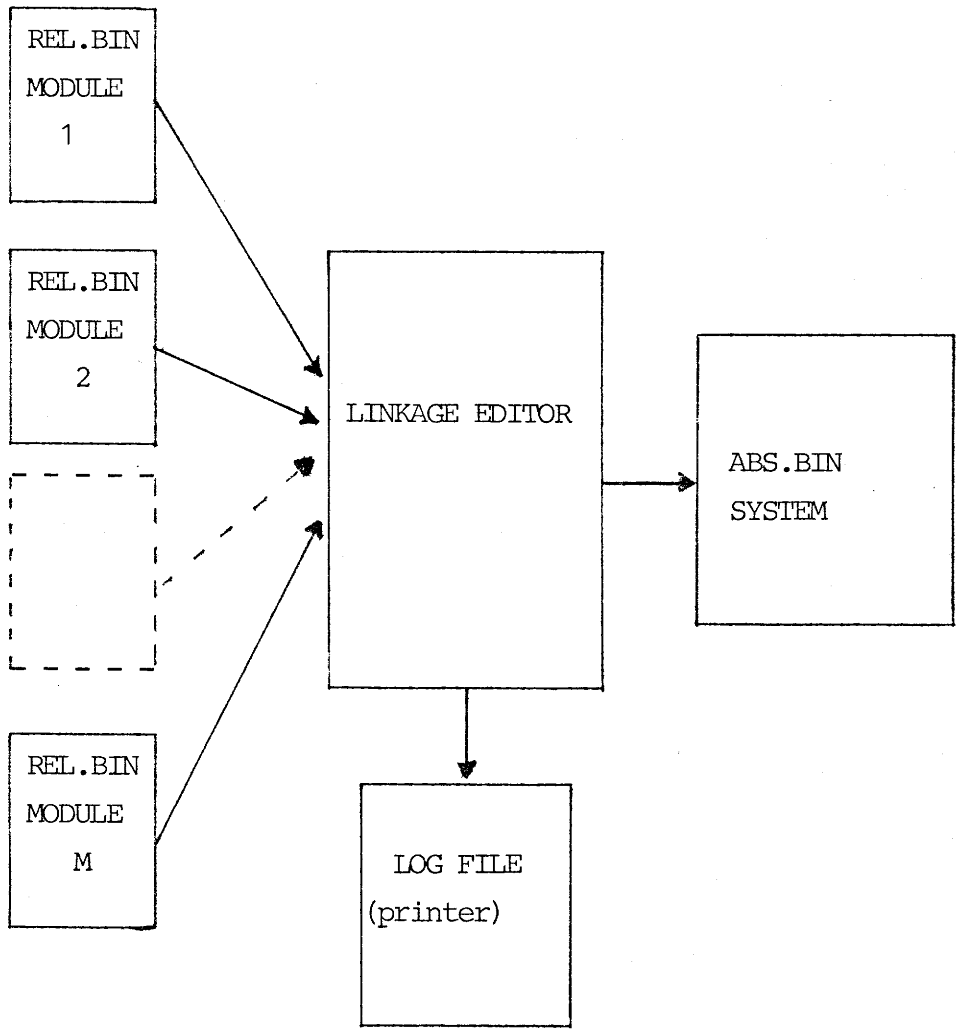


Fig. 5-1: Linkage Editing of rel. binary files creating a absolute binary file and some log information.

In the RC3600 DOMUS system the utility program LINK can do the creation of absolute binary files. The LINK program can furthermore place some system information in the resulting core image, which is used by the MUS-system initialization in the startup fase after autoload (IML image load).

The information is primarily the start addresses of all process descriptions in the resulting system, which are defined in the rel. binary files by the binary startblocks. The start addresses are placed in the system from memory address 402_8 and on, terminated by the value 177777_8.

When the MUS-system is created by the Linkage Editor the modules to link must be given in a right sequence, which is:

1) The MUS-Monitor module MUMXX
2) All non-process modules (I/O Procedures)
3) All process modules (drivers, application, etc).
4) The Operating system S, if present
5) The MUS-System Initialization module MUIXX

The format parameter FORM. must be set to N ie. creation of a absolute binary MUS-Basic system, which is not autostarting.

The log output from the Linkage Editor contains an information necessary for documentation of the created system f. ex. the titles of all linked modules.

The following is two examples of Linkage Editor calls in the DOMUS-system:

1) MUS-System with TTY and MT:

```
LINK ABS LOG.$LPT CHECK.NO FORM.N IN.MUMXX MUUXX!
!MUBXX MUCXX MURXX INTXX TTXXX MTXXX!
!SSXXX MUIXX
```

2) MUS-System with support of TTY and PTR. No MUSIL interpreter is included and an application program with driver AMX are included.

```
LINK ABS1 LOG.$SP CHECK.NO FORM.N IN.MUMXX MUUXX!
!MUBXX MUCXX MURXX TTXXX PRXXX APC AMX SSXXX!
!MUIXX
```

The modules given as parameters in the examples are the mnemonic names of the MUS system modules. The XX is the version number.

Appendix A contains a description of the MUS-System modules.

Before programming of the IML is performed the resulting system should be tested. This can be done by use of the DOMUS command BOOT, and start of the system in address 377_8.

Remember to reload the system before programming, and start the CPU in a dummy instruction.

In the MUS-system memory word 0 is 0, and the CPU can then just be started in address 0.

APPENDIX A, MUS-System modules

All MUS-System modules are given a unique name defined by the .TITL directive to the assembler. The title is transferred to the rel.binary title block in the memory file.

The titles are two or three letters with a trailing version number. The version number is replaced by XX or XXX in the following description.

Before the modules are linked be sure that the newest versions are used.

MUMXX	*	MUS-System Monitor
MUUXX	*	MUS-System Utility Procedures
MUBXX	*	MUS-System Basic I/O Procedures
MUCXX	*	MUS-System Character I/O Procedures
MURXX	*	MUS-System Record I/O Procedures
MUIXX	*	MUS-System Initialization module
INTXX		MUSIL Interpreter
TTXXX		Operator console driver (TTY)
MTXXX		Magnetic tape driver (MT0)
CRXXX		Cardreader driver
PRXXX		Papertape reader driver
FDXXX		Flexible disc driver (RC3650)
FLXXX		Flexible disc driver (RC3751)
FMXXX		Preprocess module to the flexible disc driver (RC3751)
SSXXX		Operating System S
SSAXX		Operating System S, which has automatic interpretation of file SSYSI after autoload.

The modules marked with "*" must always be present in the system.

If the Operating System is included an operator console driver and an input driver must be present too.

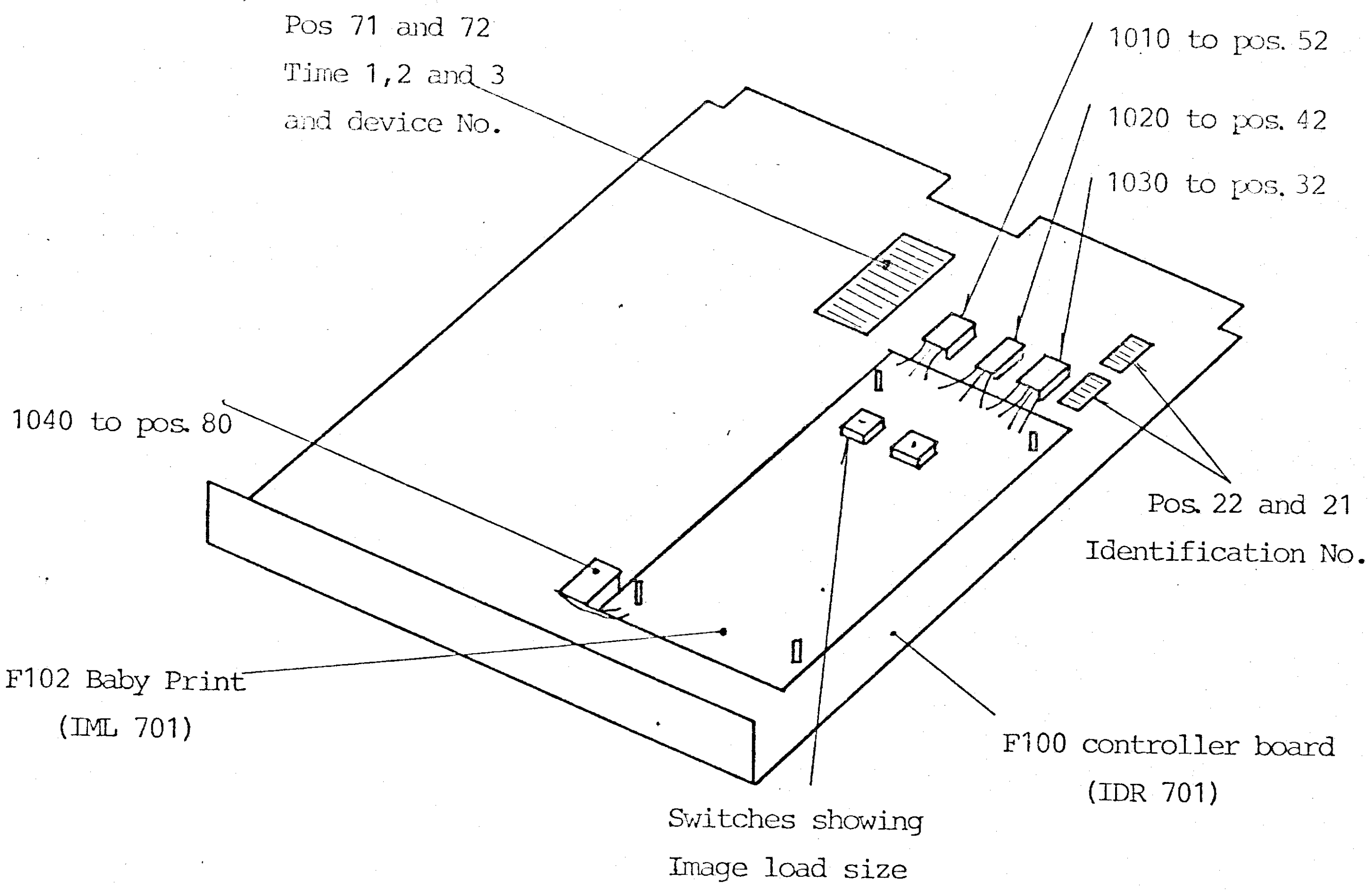
The Operating System performs an automatic initialization before start, and must be the last module in the system (highest core address) before the MUS-Initialization module.

It is recommended to use the above given sequence of the MUS-System modules because debugging is eased considerately in this case.

APPENDIX B IML701 PROGRAMMING

The programming of the Image Load is made in the following way:

1. Supply the Image Load IML701 with the correct numbers of un-programmed EPROM's of the type Intel 2716 (RC number EPROM ROM490). Set the switches on the IML701 to show the size of the Image Load used. Connect the 4 plugs from IML 701 to IDR 701 and connect IDR 701 to the RC3600 BUS.



2. Connect 28 Volt ±1V from a lab power supply or from Erasure Box EEB701 to the jack connectiors on the front panel. Max. current consumption from the power supply is 0.25 AMP. (Without this power supply programming is impossible).

3. Load the program, which is wanted in the Image Load, into the main memory of the RC3600, and start the CPU in a unused memory address executing the dummy instruction JMP .+0 (400_8)

4. Push the PROGRAM button on the front panel and the programming of the EPROM's starts from location zero, and ends in case of no errors, when the size selected with the switches on IML 701, is reached. Under programming the PROGRAM light is on.

5. Under programming the indicators IDENTIFIER (0-15) shows the address being programmed or tested. After a successfull programming the IDENTIFIER (0:15) shows the top address, which has been programmed. The program identicator stays on and no of the error indicators are turned on. Programming takes about 1 minute for each 1 K word, so 32 K words takes about ½ hour.

The microcomputer inside the unit checks the programming, and tries to reprogram in case of failures. If the programming fails in any way, the program light stays on and one or bothe of the Error lights turn on. These indicaters, together with the IDENTIFIER indicators, show which EPROM fails. This is described in details in the Technical Manual.

When the micro-computer stops the failed EPROM is found the following way:

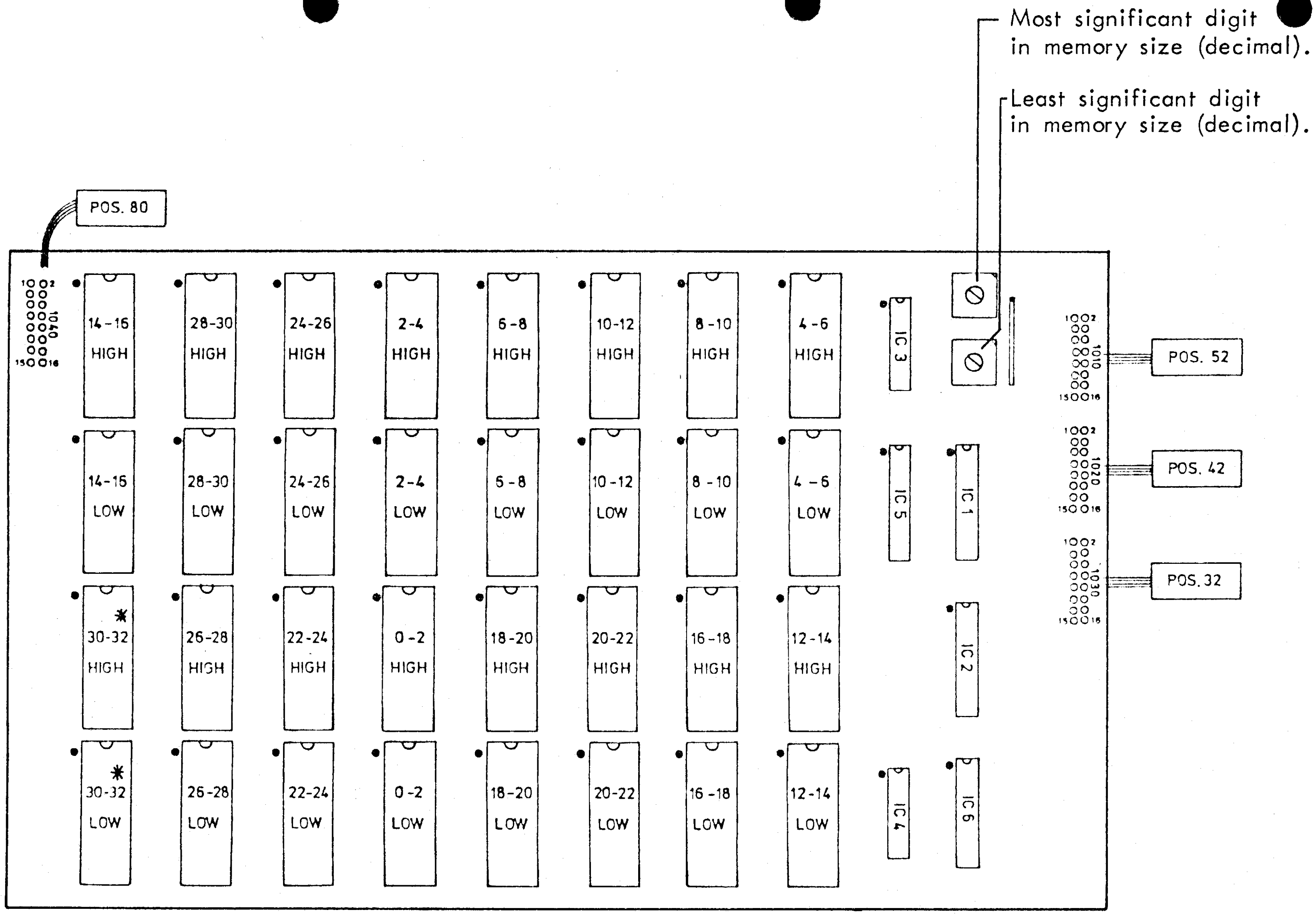
0	1 — 4	5									15
X	—— n ——	X	X	X	X	X	X	X	X	X	X

IDENTIFIER (0–15)

Failed EPROM = (2   n) - (2   n + 2)

Example: The programming stops and n = 1010 and HIGH ERROR Light is on.

Failed EPROM = (2 x 10) - (2 x 10 + 2) = 20 - 22 K word, and the EPROM marked 20 - 22 and High is found using the Assembly Drawing on next page.



* The EPROM in this positions make makes the memory from 30 K word to 32 K word.
High means most significant byte (bit 0:8).
Low means least significant byte (bit 9:15).

Appendix C References.

[1] DOMUS User's Guide, Part I
Philippe Gauguin, July 1976.

[2] DOMUS Linkage Editor, Revision 01
Marie Louise Møller, October 1977.

[3] RC3600 Binary Loader
Jens Falkenberg Andersen, November 1972.

[4] General Information for F100, F101, F102 and F103
Mogens V. Petersen, October 1978.