MEC - A Microprogrammable Computer for the Fisher/GEC-Elliot Camac System Crate

D. Notz and K. Rehlich

Abstract

In large experiments the readout of different components and the formatting of data in a computer becomes more and more timeconsuming. It is therefore necessary to use I/O devices with intelligence so that data are prepared in such a way that no further formatting is needed. We describe in this paper a microprocessor of 200 nsec cycletime which reads out part of the equipment of the TASSO experiment at PETRA. The processor resides in a Fisher/GEC-Elliot system crate and is able to readout and format complete events keeping the online computer free for monitoring services.

Zusammenfassung

In großen Experimenten nimmt die Zeit zum Auslesen und Formatieren der verschiedenen Komponenten immer mehr zu. Es ist daher notwendig, Ein-/Ausgabegeräte mit Intelligenz zu benutzen, die die Daten soweit aufbereiten, daß keine weiteren Umformungen mehr nötig sind. Wir beschreiben in dieser Arbeit einen Mikroprozessor mit 200 nsec Taktzeit, der einen Teil des TASSO-Experiments ausliest. Der Prozessor befindet sich in einem Fisher/GEC-Elliot system crate und kann komplette Ereignisse auslesen und aufbereiten. Dadurch wird der Hauptrechner frei für andere Überwachungsfunktionen.

Content

I.1 Introduction

The TASSO detector at PETRA is a large solenoidal detector which allows the measurement of charged particles and photons in almost the full solid angle [1].

Fig. 1 shows a side view of the TASSO detector. It consists of a large magnetic solenoid, 440 cm long and 270 cm in diameter. The field is about 0.5 Tesla parallel to the beam axis. The solenoid is filled with tracking chambers and time-of-flight counters. The energy of photons and electrons is measured by liquid argon counters surrounding the solenoid on top, on the bottom and in the forward direction. The hadron arms are used for particle identification at higher momenta: they are equipped with plane drift chambers, Čerenkov counters, time-of-flight and shower counters. 50 % of the solid angle is covered by muon chambers behind 60 cm of iron. A forward detector allows both measurement of the luminosity by small angle Bhabha scattering and the detection of $\gamma\gamma$ scattering.

The 15 different components in the experiment are summarized in Table 1. The total number of channels or addresses is of the order of 27756. These data must be controlled and formatted into separate blocks or banks to make further analysis easier.

In the TASSO experiment standard CAMAC controllers (A1,A2) and branch highways are used. This standarization is needed to allow each of the 9 collaborating institutes to test their equipment at home. Special read only crate controllers could not be used because they have the disadvantage that one cannot write data for thresholds or corrections to the readout electronics.
Looking at the electronics of the different components one can distinguish between 8 different readout systems:

Device	CAMAC module
1) Pattern units (Latches)	EGG/ORTEC C144 24 bits/unit, 2 slots/unit
2) Camac Addressable ADCs	LRS2249 12 channels/unit, 1slot/unit
3) ADCs with own controller Single channels are not addressable by CAMAC	LRS2280 48 channels/unit, 1 slot/unit All channels controlled by one processor
4) Camac addressable TDCs	LRS2228, EGG/ORTEC TD811 8 channels/unit, 1 slot/unit

5) Drift chamber TDCs	LRS 2770A 96 channels/unit, 3 slots/unit
6) Proportional chamber readout system also for muon chambers	RHEL 540 All channels connected to one unit. 2 slots/unit
7) ADCs for barrel liquid Argon counters	DESY system CADAS 48 units with 320 channels/unit 1 slot/unit
8) ADCs for endcap liquid Argon counter	Aachen system 5 units with max 1024 channels/unit, 1 slot/unit

Only part of the electronic information can be transferred to the computer via the standard DMA which operates in two modes [2].

a) The data are read out from one CAMAC station until there is no Q response from CAMAC. In this mode the proportional chambers and the ADCs with controller (LRS2280) can be read out.

b) The DMA can address one CAMAC slot after another. In this mode up to 16 addresses maximum per slot can be selected by the DMA. Only the pattern units and direct addressable ADCs and TDCs can be read out. But for these devices we are interested only in those ADCs and TDCs for which the corresponding bit is set in the pattern unit.

To conclude: All devices apart from the proportional chamber and the LRS2280 ADCs require a special readout which needs a lot of readout and formatting time in the online computer.

The MEC microprocessor should read out the whole information for an event and format different banks for the various components. These banks (Tab. 2) will not be changed by the analysis programs on online and offline computers. An example of a bank is shown in Table 3. Internally a bank contains pointers and length information of different groups, drift chamber cylinders or muon chambers and within a group the wire addresses start from zero. The offline programs can therefore extract quickly the no. of addresses in a certain chamber and compute coordinates.

In the second chapter the interface and the structure of the microprocessor is explained. Programming of the processor is explained in chapter three, testing and program development in chapter four and a program example is presented in chapter five. Three appendices give more information about a program to read part of the experiment, the monitor program to test the processor and the hardware realization.

I.2 The TASSO online computer configuration

The online computer and its periphery is shown in Fig. 2. We are using a NORD10/50 computer (512Kbytes memory) with two 66 Mbyte discs, floppy discs, card reader and terminals. The system and all user files are stored on one disc, and a copy of the system and a buffer area for data are placed on the second disc. The experiment is monitored by two colour TVs and steered by a touch panel on which all relevant commands are shown to the experimenter (therefore one does not have to learn the commands). With a tracker ball and a cursor one can select histograms or part of the event display to see an enlarged frame. Results are printed at the end of each run on an electrostatic printer/plotter which is also connected to two graphic terminals.

Data are taken via the CAMAC I/O port, buffered on the second big disc and then transferred to the computer centre. Here data are stored on a disc and then copied to tape.

I.3 The CAMAC interface

Interfacing of CAMAC to the online computer is done by the commercial System Crate produced by Fisher/GEC-Elliot. It mainly consists of three components:

1) The executive controller handles CAMAC requests from different source modules. In our case the online computer, the DMA module and the MEC microcomputer have access to all CAMAC branches and are "source" modules.
2) The branch couplers connect a standard CAMAC highway with 7 crates maximum to the system crate.
3) The interface of the online computer. This interface is computer dependent whereas the branch couplers and the executive controller are standard modules.

In our experiment the NORD computer is connected to the system crate with 2 modules for programmed transfer, one module for interrupt handling and three modules for DMA.

The branches and crates are standard CAMAC which is commercially available. This enables the collaborating institutes to built and test their equipment in their own workshops using available infrastructure and test aids. In the experiment 5 branches with 24 crates are needed (Fig. 3). The crates are positioned near the readout electronics in 4 different areas: central electronics, north arm, south arm and control room. The distance between these areas and the computer varies from 30 m to 70 m.

I.4 Demands for the microcomputer

The total number of channels, wires, bits which contain the information of a single event could be rather large. (Typical event length 3000 - 4000 words). Hence a large memory for parameters and data storage is needed. We use a memory of 8k words with 16 bit word length. The frequency of triggering at design luminosity of PETRA is estimated to be 20 Hz approximately. Therefore the time for readout, formatting of data and transfer to the online computer must be less than 50 msec. The microcomputer must be fast; a cycle time of 200 nsec is chosen. In experiments of this magnitude the overall configuration is not fixed: some components get improved electronics, and other equipment is added or replaced. A readout system must be flexible or - in other words - programmable.

The MEC microprocessor has the following properties:

1) Programmable. Programs are stored in a PROM or in a separate card in a RAM. Storage size for the microprogram: 1K words, 64 bits/word
2) 200 nsec cycletime
3) Memory for parameters and data: 8k words, 16 bit/word
4) Two slot wide CAMAC module with PROMs or three slots, if microprogram is stored in a separate RAM.

II. Structure of the microcomputer

The system of the microcomputer is shown in Fig. 4. It is built using four-bit slices manufactured by Advanced Micro Devices [3]. One can distinguish between several blocks:

1) The arithmetic-logical unit (ALU) performs the arithmetic and logical operations.
2) The sequencer computes the next microprogram address depending on internal or external conditions.
3) A pipeline register increases the speed of the processor because the information of the microprogram memory is available at the rising edge of the clock cycle. During execution of one instruction the following instruction is placed into the pipeline register. Hence conditional jumps can be executed in the following instruction at the earliest.
4) The microprogram memory contains the microcode which can be stored into a PROM or into a RAM on an additional CAMAC card.
5) Data and parameters are stored in a memory which can be accessed by the microcomputer and via CAMAC by the online computer to store parameters and to read event information.
6) CAMAC command and data register contain the CAMAC function, CAMAC addresses BCNA and results of the CAMAC cycle.
7) The internal bus connects all these blocks to transfer information.

II.1 The arithmetic and logical unit ALU

In the following we assume that the reader is familiar with the slice processors of the AMD 2900 series [3-5]. We repeate here only the main features. The position of the ALU inside the MEC microprocessor is shown in Fig. 4. Four AM2901 slices are connected forming a 16 bit processor. The internal structure of a single AM2901 can be seen in Fig. 5. The ALU has two input paths R and S which are combined by the operation (R + S, R - S, S - R, $R \vee S$, $R \wedge S$,...). The result F can be strobed onto the external bus and internally written to one of the 16 registers or to the shift register Q. Before storing the data they may be shifted by one position. The information of the most and least significant bit depends on the four selectors 74253. This allows logical shifts, arithmetic shifts and rotations (Fig. 6). double shifts with a combined Q register for multiplication and division are also possible. The carry is generated by a lookahead carry generator AM2902 avoiding time delays caused by ripple carry.

II.2 The sequencer

The address of the next instruction is prepared by the sequencer and may depend on external or internal conditions. The size of the microprogram memory is 1024 words long. Three AM2911 sequencer modules with four bits each are needed for addressing. The AM2911 contains a multiplexer, an incrementer, a microprogram counter and a stack of four words depth (Fig.7). In some computers the return addresses in subroutine calls are stored in the first word of the subroutine or in a save area file. This scheme does not work for microcomputers because the program can also be stored in a non writable PROM. All microcomputers dispose of a stack for storing of addresses. The depth of four in our case restricts subroutine calls to the fourth level. The next address multiplexer can take the next address from four sources as input:

1) Microprogram counter. This counter is incremented by one at each clock cycle.
2) Stack register: The stack is loaded with return addresses of subroutines or with addresses of a beginning of a loop.
3) Data from the internal bus used in jump operation. The address is provided by the pipeline register
4) Address from the mapping PROM.

The three sequencer modules are steered by a 29811 address controller. It pushes/pops data on to the stack and gates the multiplexer input (Fig. 8).

Its action may depend on one test input. This input is connected via an inverter and a multiplexer to all possible external conditions: (FALSE, $A \geq B$, $A \leq B$, EVENT START, CAMAC Q, CAMAC DATA ready, F = 0, COUNTER = 0...). All these conditions may or may not be inverted. The conditions on results produced by the ALU ($A \gtrless B$, F = 0,...) are set by the previous instruction. Therefore two microcycles are needed for a conditional branch.

The sequencer is further equipped with a hardware counter (Three 74163). This counter is very useful for loops. The number of loops is stored into the counter taking the information from the pipeline register. Then a set of commands is repeated just by a conditional branch and testing whether the counter has reached zero. When loading the counter with a negative loop index care must be taken of the fact that the 74163 produces a carry out when reaching -1. The final carry is produced by a ripple carry for the three 74163's and is stable only after some delay: so a one level pipelining is done here. As a consequence the carry bit is stable at the next cycle. The loop index must therefore be decreased by two and it's complement value should be stored into the counter.

II.3 The pipeline register

In a memory the data are stable at the output after some delay (~70 nsec). This delay is not negligible compared to a cycletime of 200 nsec; so to avoid a slow-down the microinstruction which is about to be executed is stored in a pipeline register and, simultaneously, the address of the next microinstruction is applied by the sequencer to the microprogram memory. The contents of that word are then set up at the input of the pipeline register.

The pipeline register is built up by different chips. 3 AM2918's strobe the data onto the internal bus to load the ALU or the sequencer with the next address. A 74175 is used for the test multiplexer and one 74174 for the sequencer controller. Two 74374's latch the instructions to the ALU and a 74273 is used for the shift selectors, carry selectors and other selectors or control bits.

II.4 The microprogram memory

The memory of the microcomputer has a capacity of 1024 words with 64 bits/word. The microprogram is stored in 8 82S181 PROMs. Using PROMs has the advantage that it is not necessary to load the memory after each power fail. The disadvantage is the decreased flexibility and the impossibility of testing the device. We therefore produced in addition an external memory of the same size which can be loaded and controlled by the online computer. This memory (PROM simulator) can operate in two modes:

1) Seen from the online computer it is a 4k word memory with 16 bits/word which can be loaded and read out. In this mode the memory can also be used to test CAMAC transfer, DMA, etc.

2) Via special cables the memory is connected to the microprocessor's PROM sockets. The processor selects 64 bits/word. The current address of the processor is also readable by the main computer and indicated by LEDs. This configuration is the essential tool for testing the microprocessor and developing programs. See chapter IV for more details.

II.5 The data memory

For data and parameters a 8k words, 16 bits/word memory is used (Fairchild 93471). Parameters for CAMAC addresses and data structure are stored into the memory by the online computer. When the memory is accessed via CAMAC the microcomputer has to be stopped. The access to the memory is controlled by a counter which is incremented after each read or write cycle allowing DMA transfer. When the microcomputer wants to transfer data to the memory it has first to set the memory address register by a previous instruction.

II.6 CAMAC command and data register

Before requesting a CAMAC cycle the CAMAC command must be loaded into the command register. This register is 24 bits long and contains the CAMAC address and function.

Crate bus	F F F F F W W W 16 8 4 2 1 14 13 12	W W W W W W 10 9 8 7 6 5	W W W W A A A A 4 3 2 1 4 3 2 1
Meaning	F F F F F B B B 16 8 4 2 1 4 2 1	C C C N 3 2 1 16	N N N N A A A A 8 4 2 1 8 4 2 1

F = CAMAC function, B = Branch address, C = Crate address, N = Slot number, A = Subaddress

After loading the register the microcomputer gives a request to the system crate controller via the daisy chained arbitration highway. If no module with higher priority need the CAMAC periphery the executive controller allows

the MEC to be MASTER on the crate. Then the command register is strobed onto the data way and one CAMAC cycle is generated.

The result of the transfer is stored into 24 bit data registers and a data ready bit is set. This bit can be tested by the microprogram and is cleared if the data are transferred to the ALU or to the memory.

II.7 The internal bus

The structure of the internal bus is shown in Fig. 4 . It has 5 sources

1. ALU
2. Pipeline register
3. Camac data register low
4. Camac data register high
5. Data memory

and 5 destinations

1. ALU
2. Data memory
3. Memory address register
4. Camac command register low
5. Camac command register high

Data can be transferred between all sources and destinations allowing a lot of flexibility. See chapter III.7 for more details. Within one cycle during bus transfer the counter of the sequencer and an ALU's registers can be loaded independently.

III Programming the microprocessor

The microprocessor's memory for microcode is 1024 words long with a word length of 64 bits. Small programs of the order of 10-30 instructions (2000 bits) can be written directly in binary code whereas programs of the order of 200 or more instructions (10 000 bits) should be developed with the help of an assembler. In this chapter we describe an assembler which is written in FORTRAN and is therefore machine independent. The microprogram is written on a bigger computer using well known editors and file systems and afterwords translated. The address computation and syntax checks are done by the assembler which also produces a listing and a binary dump. The binary code is then loaded into a PROM Simulator and the program can be tested by a monitor program. In chapter IV more details are given how programs are developed.

III.1 The microprogram word

The structure of the microprogram word is shown in Table 4. As we have seen in Figs. 4, 6 and 8 one can distiguish different blocks of the microprocessor. Each block is connected to some part of the microprogram word. The different fields are mainly:

1) Field for the ALU: Function and registers
2) Field for the sequencer: Branch instructions
3) Field for the bus: Destination and source
4) Fields for test and shift multiplexers
5) Field for constants: Addresses, counter or numerical constants
6) Bits for status of the processor.

In the assembler these fields are grouped in a way that the instructions which are used frequently are placed in the beginning of a line with 72 columns. Each line starts with a label field followed by the fields for the ALU, bus destination and source, sequencer information etc.

```
        A L U                 B U S  CCU                    carry test
label dest. func. sa reg.  dest. sa. inst. const.  shift  out in MPX
LLLLL DDDD FFFF SS B A DDD SSS IIII LNNNNNN SSSS OO II TTTT M W C E B M
```

If no instruction is inserted in a field, code 0 is inserted in the microcode apart from the sequencer field where the code 14_8 (CONTINUE) is used.

III.2 The label field

The first five columns may contain labels which can be used as addresses in branch operations. These labels are arbitrary decimal numbers like those in FORTRAN. If the star is printed in the first column the line is a comment and ignored by the assembler. A typical listing of a program is shown in Fig. A3

Load memory address register. The 7 is in the constant field and loaded from the pipeline register. No operations are performed in the ALU.

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
NOP             MAR PIP     D7                      S W
NOP             MEM PIP     D4
```

Increment register 2, which is used as address register and store into that location a 3:

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
LRF  ADD  ZB 2   MAR ALS                CH
```

- Add zero+carry to register 2
- Store the newly computed address into register 2
- the result of the ALU is strobed on to the bus and loaded into the memory address register
- Carry bit for incrementing

Store the 3 from the pipeline register into memory

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
NOP             MEM PIP     D3                      S W
```

Memory select bits

Camac transfer:

Registers 1 and 2 contain the information for the CAMAC command registers (Function-Branch, Crate-Station-Subaddress). The CAMAC data should be loaded into registers 3 and 4 :


```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
NOP  OR   ZB 1   CHM ALS                                          H
NOP  OR   ZB 2   CML ALS                                          request CAMAC
```

- ALU data strobed to the bus
- Load CAMAC command registers

Now the program has to wait until CAMAC is finished
Repeat the instruction if data are not ready

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
2    NOP                        CJP  L2                    DNTR
```

- Label 2
- Conditional jump
- CAMAC not ready

Load CAMAC result. The data at the CAMAC bus in the crate are inverted (0V = logical "1"; 5V = logical "0"). Therefore the data must be inverted using an exclusive NOR

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
LRF  EXNO DZ 3   ALU CAL
LRF  EXNO DZ 4   ALU CAH
```

- CAMAC data registers
- Data are loaded into the ALU
- Exclusive NOR of data with "ZERO": $\overline{D + 0}$
- Load data into register 3 and 4

The CAMAC ready flag will be cleared by the read CAMAC register low (CAL) command.

III.9 The field for the sequence controller

Columns	32 - 35	
Bitposition	10 - 13	
Code	Mnemonic	
0	JZ	Jump to Address Zero
1	CJS	Conditional Jump to Subroutine with Jump Address in Pipeline Register
2	JMAP	Jump to Address at Mapping PROM Output
3	CJP	Conditional Jump to Address in Pipeline Register
4	PUSH	Push Stack and Conditionally Load Counter
5	JSRP	Jump to Subroutine with Starting Address Conditionally Selected from Am2911 R-Register or Pipeline Register
6	CJV	Conditional Jump to Vector Address (Not realized in our hardware)
7	JRP	Jump to Address Conditionally Selected from Am2911 R-Register or Pipeline Register
8	RFCT	Repeat Loop if Counter is not Equal to Zero
9	RPCT	Repeat Pipeline Address if Counter is not Equal to Zero
10	CRTN	Conditional Return from Subroutine
11	CJPP	Conditional Jump to Pipeline Address and Pop Stack
12	LDCT	Load Counter and Continue
13	LOOP	Test End of Loop
14	CONT	Continue to Next Address
15	JP	Jump to Pipeline Register Address 29811 only

In the field for the sequence controller code 14 (continue) is default if the field contains a blank.

JZ jump to address zero is normally used at the end of a task to wait for the next interrupt or event.

CJS pushes the next address on the stack file and jumps to the address presented in the label field of the pipeline register if the condition is fulfilled (subroutine call).

CRTN pop stack and jump to the address given by the stack (Return).

Example:

Jump to subroutine label L15 if sum of register 4 + register 5 = 0

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
      LRF  ADD  AB 4 5                          CL
      NOP                CJS  L15               FEQZ
15    LRF  ADD  AB B 9
      LRF  SUBR AB 1 2   CRTN                   TRUE
                         └Return                └──────condition = 'true'
```


JMAP can be used to start another program the address of which is loaded into the mapping prom and selected by the online computer via a CAMAC command.

CJP Jump to the address given by the pipeline register.

Example:

Jump unconditionally to label 33

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
      NOP                CJP  L33               TRUE
33    NOP
```

PUSH The next address is pushed on to the stack and the loop counter is loaded with the complement of number of loops-2 from the pineline register. This command together with

RFCT repeat loop if counter is not equal to zero is used for loops.

Example:

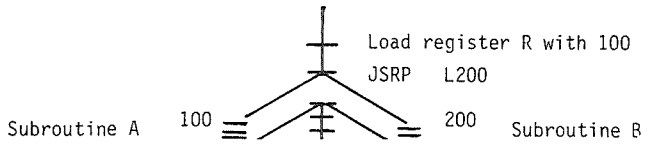
Run the following instructions 8 times. The 10_8 must be reduced by two ($=6_8$) and inverted (= -7) because the LS163 counter gives a carry at -1 (One's complement presentation) and the carry is pipelined by one level allowing an internal delay for the ripple carry in the three LS163 counters

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
      NOP                PUSH 01771             TRUE
                         │    │                 └── set condition
                instruction  └ -7(= (8 - 2))
```

The following instructions are repeated 8 times

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
      LRF  ADD  AB 4 5                          CL
      LRF  ADD  AB 3 4   RFCT                   CNEZ
                                                │
                                                Counter not equal zero
```

JSRP,JRP are two commands to branch to two different addresses in the next cycle. With the previous instruction the internal register R of the sequencer is loaded (mapping prom, pipeline register). The address of the next instruction is taken either from the internal R register or the pipeline register depending on the condition:



RPCT In a previous instruction the counter is set with the

LDCT command and the instructions between this address and the address given by the pipeline register are repeated.

Example:
Shift the Q register together with register 5 by 8 positions to the right

Load the counter ($8 = 1000_2$, $6 = 110_2$, $\overline{6} = 1111111001_2 = 1771_8$)

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
22   NOP                                  LDCT 01771
     LRQD OR    ZB 5                      RPCT L22      RRSL      CL CNEZ
```

Shift Q and register 5

Repeat counter

Label

shift multiplexer setting

Carry low

counter not equal zero

CJPP If one does not want to return from a subroutine but jump to an other address the stack is popped by this instruction.

LOOP This command for loop testing works in a similar way to RFCT but the counter is not incremented but the test nultiplexer with another condition input is used.

CONT is the default to use the next instruction.

JP jumps unconditionally to the address given in the pipeline register.

III.10 The field for constants:

Columns	37 - 43
Bit position	0 - 9
Letter in column 37	
O	Octal value (Default)
D	Decimal value
L	Address or Label constant

This field contains numerical constants, addresses for subroutines and loop indices to load the counter. The field is 10 bits wide covering the whole address space. If more bits are needed for masks and constants one must shift the word in the next instruction or invert the information in the ALU to set the most significant bits.

III.11 The shift linkage control field

Shift-/Rotate Operations

Columns 45 - 48
Bitposition 42 - 47

SSSSSS 543210	ALU Destination	mne-monics	meaning		B-Register	Q-Register
001100	F/2=B	SRSS	Shift right short		0 →	
001000	F/2=B	SRAS	Shift right arith. short	OVR¼	F15 →	
001100	F/2=B,Q/2=Q	SRSL	Shift right long		0 →	→
001000	F/2=B,Q/2=Q	SRAL	Shift right arith. long	OVR¼	F15 →	→
000011	2·F=B	SLSS	Shift left short		← 0	
000010	2·F=B	SLAS	Shift left carry short		← C	
110000	2·F=B,2·Q=Q	SLSL	Shift left long		←	← 0
100000	2·F=B,2·Q=Q	SLAL	Shift left carry long		←	← C
000100	F/2=B	RRSS	Rotate right short			
001000	F/2=B	RRAS	Rotate right arith. short	OVR¼	F15 →	
000000	F/2=B,Q/2=Q	RRSL	Rotate right long			→
001000	F/2=B,Q/2=Q	RRAL	Rotate right arith. long	OVR¼	F15 →	→
000001	2·F=B	RLSS	Rotate left short			
000010	2·F=B	RLAS	Rotate left carry short		← C	
010000	2·F=B,2·Q=Q	RLSL	Rotate left long			←
100000	2·F=B,2·Q=Q	RLAL	Rotate left carry long			← C

The contents of the carry bit also depends on the carry multiplexer. In our configuration the commands SRAS/RRAS and SRAL/RRAL are identical.

The bits for the shift linkage control field are connected to the multiplexers shown in Fig. 6 to do the shifting and rotation. The assembler does not check whether the shift control, ALU destination and carry multiplexers are combined correctly because the programmer may combine them differently.

Examples:

Shift register 4 one position right and test with the next instruction if the rightmost bit was set

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
     LRQD OR   ZB 4                    SRSS RR
F/2=B                                  |    |_____ The rightmost bit
                                       |          is selected by the
                                       |__Selector for   carry out multiplexer
                                          shift/rotate

NOP                  CJP  L4           CART
                      |                 |_____ Test on
                      |                        carry of pre-
Goto label 4                                   vious in-
if test is true                                struction
```

Shift register 5 together with register Q to 8 positions to the right.

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
     NOP
272  LRD  OR   ZB 5          LDCT 01771
                             RPCT L272\    SRSL    CL CNEZ
                                       \
                                        Load counter with -7 = (8-2)
```

III.12 The carry out multinlexer field

Columns	50 - 51	
Bitposition	39 - 41	
Code	Mnemonic	Carry out
0	HI	High
1	CA	Carry of previous cycle
2	LO	Low
3	RQ	Bit 0 of Q-register
4	LR	Bit 15 of RAM
5	SC	Sign bit 15 of ALU output
6	RR	Bit 0 of RAM
7	CO	Carry out

The output of the carry multiplexer is strobed to the test multiplexer which can be tested in the next cycle and to the shift control multiplexers to shift the information into an internal register. The input may be low, high, bit 0 of Q or RAM register, carry F15 or bit 15 of the RAM register.

III.13 The carry-in multiplexer field

Columns	53 - 54	
Bitposition	37 - 38	
Code	Mnemonic	Carry in
0	CH	High
1	CC	Carry out
2	CB	$\overline{\text{Carry out}}$
3	CL	Low

The carry-in information is used in arithmetic instruction. An add with carry-in high increments the results by one, a subtract with carry-in low decreases the result by one.

III.14 The test multiplexer field

Columns 56 - 59
Bitpositions 14 - 18

Code	Mnemonic	Test
0	TRUE	True
1	FALS	False
2	-	
3	-	
4	AGEB	A ≥ B
5	ALTB	A < B
6	ALEB	A ≤ B
7	AGTB	A > B
8	-	
9	-	
10	EVSP	Event not start
11	EVNT	Event start
12	XTRU	Camac X true
13	XNOT	Camac X not set
14	QTRU	Camac Q true
15	QNOT	Camac Q not set
16	DNTR	Camac data word not ready
17	DARD	Camac data word ready
18	POSI	Positiv sign
19	NEGA	Negativ sign
20	OVFN	No overflow
21	OVFL	Overflow
22	CALN	Low byte = 0
23	CALB	Low byte ≠ 0
24	CAHN	High byte = 0
25	CAHB	High byte ≠ 0
26	FNEZ	F ≠ 0
27	FEQZ	F = 0
28	CARN	Carry-out multiplexer inverted
29	CART	Carry-out multiplexer
30	CNEZ	Counter ≠ 0
31	CEQZ	Counter = 0

Bit 14 of the instruction is used to invert the information at the test input. The inputs to the test multiplexer are shown in Fig. 8. Event start can be used to start the processor with an external event flag: if the processor is in a dead loop waiting for it. CAMAC flags X, Q and the data word ready are cleared by the CAL read CAMAC data low instruction in the field for the source of the internal bus. Low byte is the wired 'OR' of the two least significant slices's F = 0, high byte of the two most significant slices.

Example:

Go to label 4 if register 4 = register 5

```
*LLLL DEST FUNC DS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
*
      LRF  SUBR AB 4 5                              CL
      NOP                      CJP  L4                FEQZ
```

III.15 Memory select

Column 61
Bitposition 57

Code	Mnemonic	
0	blanc	Memory not selected
1	S	Memory select.

If MEM or MMS is given as destination or source for the internal bus the memory must be selected using S in column 61.

III.16 Memory write

Column 63
Bitposition 58

Code	Mnemonic	
0	blanc	Read flag
1	W	Write flag for memeory

If MEM is given also the write flag W must be set.

III.17 CAMAC request

Column 65		
Bitposition 55		
CODE	Mnemonic	
0	blanc	Slave
1	H	Host/Master
		CAMAC request to system crate

For requesting a Camac cycle from the system this bit must be set.

III.18 Error indicator

Column 67		
Bitposition 54		
Code	Mnemonic	
0	blanc	No error
1	E	Error, give LAM2

The error can be set if a CAMAC module does not give an X or Q response or if some parameters are wrong. A CAMAC LAM2 interrupt is produced.

III.19 Busy indicator

Column 69		
Bitposition 56		
Code	Mnemonic	
0	N	Not densy
1	B	Busy
	blanc	Always the last state busy/not busy is used

This bit is directly connected to the run light at the front panel of the processor.

III.20 Multiply bit

Column 71		
Bitposition 19		
Code	Mnemonic	
0	blanc	No multiply
1	M	Multiply

The multiply bit influences the ALU source field. If it is one AB or AQ will be replaced by ZB or ZQ depending on to Q0 bit avoding an extra instruction for testing that bit and deciding to add register A or a zero.

Example: Multiply two numbers $583_{10} * 253_{10}$

```
 1 0000 *LLLL DDDD FFFF SS B A BDB BSB CCUU LOOOOOO SHFT CO CI TMUX S W H E B M
 2 0000 *-------------------------------------------------------------------------
 3 0000 *          TWO'S COMPLEMENT MULTIPLICATION
 4 0000 *          REGISTER 0 CONTAINS THE MULTIPLIER
 5 0000 *          REGISTER 1 THE MULTIPLICANT
 6 0000 *
 7 0000 *          LOAD REGISTERS 0 AND 1
 8 0000       LRF  OR   DZ 0   ALU PIP      D583
 9 0001       LRF  OR   DZ 1   ALU PIP      D253
10 0002 *          MOVE MULTIPLIER TO Q REGISTER
11 0002       LQ   OR   ZA 0
12 0003 *          CLEAR REGISTER 3 AND MOVE COMPL(15-2) INTO COUNTER
13 0003       LRF  AND  ZB 3          LDCT 01762
14 0004 *          CONDITIONAL ADD AND SHIFT 15 TIMES
15 0004 22    LRQD ADD  AB 3 1        RPCT L22     SRAL    CL CNEZ          M
16 0005 *
17 0005 *          CONDITIONAL SUBTRACT AND SHIFT (IN THE LAST CYCLE THE SIGN
18 0005 *          OF THE MULTIPLIER IS EXAMINED AND A SUBTRACTION RATHER THAN A
19 0005 *          ADDITION IS THE RIGHT OPERATION TO GET A CORRECT
20 0005 *          TWO'S COMPLEMENT PRODUCT
21 0005 *
22 0005       LRQD SUBR AB 3 1                      SRAL    CH               M
23 0006 *          MOVE LEAST SIGNIFICANT PART OR PRODUCT TO REGISTER 2
24 0006       LRF  OR   ZQ 2
25 0007 *          FINAL RESULTS IN REGISTERS 2 AND 3
26 0007       NOP  OR   ZB 3
27 0010       NOP  OR   ZB 2          JZ
```

Program flow

```
000000: 035107  170000  000015  000004 DATA BUS :  001107
000001: 034375  170400  000015  000004 DATA BUS :  001107   585
000002: 034000  140000  000001  000000 DATA BUS :  000375   253
000003: 031762  031400  000016  000000 DATA BUS :  001107
000004: 162004  011437  020160  000000 DATA BUS :  000000
000004: 162004  011437  020160  000000 DATA BUS :  000375
000004: 162004  011437  020160  000000 DATA BUS :  000573
000004: 162004  011437  020160  000000 DATA BUS :  000672
000004: 162004  011437  020160  000000 DATA BUS :  000335
000004: 162004  011437  020160  000000 DATA BUS :  000156
000004: 162004  011437  020160  000000 DATA BUS :  000067
000004: 162004  011437  020160  000000 DATA BUS :  000430
000004: 162004  011437  020160  000000 DATA BUS :  000214
000004: 162004  011437  020160  000000 DATA BUS :  000106
000004: 162004  011437  020160  000000 DATA BUS :  000440
000004: 162004  011437  020160  000000 DATA BUS :  000220
000004: 162004  011437  020160  000000 DATA BUS :  000110
000004: 162004  011437  020160  000000 DATA BUS :  000044
000004: 162004  011437  020160  000000 DATA BUS :  000022
000005: 034000  111430  020020  000000 DATA BUS :  000011
000006: 034000  121000  000015  000000 DATA BUS :  000004
000007: 034000  131460  000005  000000 DATA BUS :  040053
000010: 000000  131000  000005  000000 DATA BUS :  000002
```

Register 3 = 16427 = 16427
Register 2 = 2*65536 = 131072
583 × 253 = 147499

IV. Developing programs on the microcomputer

After having built the processor and checked the hardware one needs tools to develop programs. First tests were performed using an exorcisor manufactured by Motorola which is equipped with one teletype and two floppy discs. The PROMs are replaced by the exorcisor's internal memory. The binary code is typed into the memory using an editor. The output and behaviour of the processor is checked with a logic analyser. This procedure is cumbersome because one has to type in binary code and whenever an instruction is inserted all addresses must be recalculated. Note that the program for reading pattern units, ADCs and TDCs is in our case 246 64-bit instructions long giving a total of 15744 bits so that the chance for errors increases. The advantages are that one can test the processor at design speed because the memory is fast enough and that one can produce PROMs with this machine.

It soon turned out that one needs an assembler and a medium size computer with lineprinter and discs to write and test programs. Therefore the exorcisor was replaced by the online computer and the PROMs were replaced by a PROM simulator. The simulator is connected to the PROM sockets in the microcomputer and contains an address register (10 bits wide), one data register (64 bits wide) and one connection to clock the processor. In this configuration the program is edited in a readable assembler format with comments on a known computer, relieving the user of the need to learn a new editor and operating system. Then the program is checked and translated by the assembler and printed on the lineprinter. The online computer then loads the first instruction into the prom simulator, produces one clock cycle and reads the next address from the address register. The microcode for this address is then loaded for the next cycle. In addition the online computer can read via CAMAC the information of the microprocessor's internal bus. The whole program flow can be printed on a lineprinter and checked. An example of program flow is given in Fig. 9.
With this method it is possible to check the logic of a program but one cannot test the behaviour of the microprocessor under realtime conditions with design speed. The one-register promsimulator was therefore replaced by a simulator with a complete memory which can be loaded and controlled by the online computer in the same way as described above.

When the development of the program is finished, it can be run at full speed by using the internal clock of the microcomputer. If problems occur at this stage one can check the program addresses on the LEDs and easily modify the routines.

V. An example: Program to read pattern units, ADCs and TDCs

In the experiment we have to read out via Camac latches which are set by phototubes or meantimers. Some phototubes are connected to ADCs or TDCs or both. Therefore we define different groups of counters according to the different parts of the detector. For each group the number of members, number of ADCs and TDCs per member and the different Camac addresses are defined in a parameter list PARLIS.

The microprocessor is the first readout device which is started by the experiment computer after an event interrupt. It might be that the conversion of the ADCs and TDCs has not yet finished when the program is started. Therefore the pattern units are read out first in one block and stored into the memory (600 µsec). The bit pattern is then tested together with the information of the parameter list and only those ADCs/TDCs which have a corresponding bit in the bit map are read.

The cycle time of the scanner is 200 nsec which is fast compared to the CAMAC read out time in a branch (~3 µsec). To save time the CAMAC cycles and the computation of the next CAMAC addresses are overlapped.

CAMAC errors do not cause a hang up of the system. If a CAMAC ready signal does not appear within 5 µsec the readout is ignored, a zero is stored into memory, the actual CAMAC address is copied into memory and the number of errors is increased. This allows the main computer to check the system and to warn the experimenter.

The program is $366_8 (=246_{10})$ microinstructions long; the number of cycles and the read out time for a certain no. of bits set in each 24 bit word is:

	No. of cycles	Time for readout
All pattern units with 0 bits on (No ADCs,TDCs read)	3028	0.63 msec
" " " " 6 " "	10234	2.1 "
" " " " 12 " "	17257	3.5 "
" " " " 16 " "	23833	4.8 "
" " " " 24 " "	30037	6.4 "

More details about this program are given in Apendix A.

VI. Outlook

The processor was produced in two versions by an outside company [6]:

1) Wire wrap version with wire wrapping on the same side as the ICs. The cycle time of this processor was 400 nsec and didn't run very reliably. Most problems came from bad contacts or wires broken at the wrapp pins.

2) Layout version. After removing some layout errors this version runs with PROMs at a cycle time of 200 nsec. From time to time the program does not work correctly and overwrites the parameters. After checking the timing and replacing some chips by faster ones there are no more problems.

In future it might be useful to use two microprocessors to read out complete events. After an event has been read and digested by the first processor the gates can be opened for the following event to be read by the second processor while data are sent to the online computer from the first one.

If this device were to be built over again, the following modifications should be kept in mind:

1. The processor should also be able to write information to CAMAC.
2. It should have one or two autoincrement memory registers so that copying could be done with half the instructions,
3. The bits for memory select or write enable should not be part of the micro-instruction but should be decoded from the bus source or destination.
4. The processor's structure is copied from the structure of a bigger machine where several microinstructions are executed for one instruction in a fixed order. In our situation the whole program is written in microcode and the sequence of microinstructions should be variable. In the present processor it is not possible to load the next microcode address from the parameter memory to the sequencer or to load the hardware counter (74LS163) from the ALU. The Counter and the addresses register of the sequencer also should be connected to the internal bus.
5. The pipeline register for constants should be 16 bits long.

We would like to thank Dr. H.-J. Stuckenberg (DESY F56), Dr. B. Struck and H. Ülschläger (Company Dr. B. Struck) for numerous and fruitful discussions. Special thanks are due to Mrs. E. Hell for her efforts with the manuscript.

References

1) TASSO Collaboration, R. Brandelik et al., Phys. Lett. 83B (1979) 261

2) A Modular CAMAC System Controller for the NORD-10 Computers using the GEC-Elliot System Crate Philosophy, J. P. Vanuxem
CERN CAMAC Note 60-00, Nov. 1976

3) Advanced Micro Devices
The Am 2900 Family, Data Book

4) Advanced Micro Devices
Microprogramming Handbook

5) Advanced Micro Devices
A Microprogrammed 16-Bit Computer

6) Company Dr. Bernd Struck, Dorfstraße 163, D2000 Tangstedt/Hamburg
Tel. 04109/6252

Appendix A

A1 Program to read pattern units, ADCs and TDCs

In our experiment we want to read from different components the scintillation counter information.
Example: For the inner time-of-flight counter system we read for each of the 48 counters one latch for the mean timer and on each end one latch, one ADC and one TDC. This defines in our terminology three groups:

1. Read 48 latches without ADCs and TDCs
2. Read one side with 48 latches with one ADC and one TDC
3. Read other side with 48 latches with one ADC and one TDC.

For each group or component we therefore use the following information:

1. No. of latches per group
2. No. of ADCs to be read for each latch
3. No. of TDCs to be read for each latch
4. CAMAC address for the first pattern unit in the group. If this field is zero the CAMAC addresses of the previous field are continued
5. CAMAC address of the first ADC
6. CAMAC address of the first TDC

An example of the parameters is given in Fig. A1. Dummy groups with zero number of latches are inserted for future development.

The final data will be stored in two different banks, one bank (BITS) for the pattern units and one bank (ATDC) for the ADCs and TDCs. After an event interrupt the microprocessor is started and creates the BITS bank. As the latch information is available immediately after the interrupt this information is read out in one block. Then the ATDC bank will be created and the ADCs and TDCs can be read after computing the actual CAMAC address. In order to save computer time the CAMAC cycles and the computation of the next addresses are overlapped. Fig. A2 shows a flow chart of the program. Each box corresponds to one subroutine with a label. The actual coding of the pattern unit readout part is shown in Fig. A3. This source text is then compiled to produce an absolute binary code shown in Fig. A4 which can be placed into a PROM or a RAM.

Before starting the program the parameters must also loaded into the parameter memory. This is done by a separate program at the beginning of each run. The parameters are reformatted and organized in a way that the computation is fast. First all parameters for the BITS bank and CAMAC addresses for the latches are stored followed by the ATDC bank and ADC/TDC addresses. The final parameters and the result of the readout is shown in Fig. A5.

A2 Parameters in memory and registers used by the program

In this chapter we describe the parameters in memory and the usage of the 16 registers.

Parameters:

```
 0
 :
 :     Not used
 :
27
28     Camac subaddress A for ADCs of current group
29     Number of subaddresses per ADC module (12)
30     Actual CNA Camac address
31     Actual branch address
32     Camac subaddress A for TDCs of current group
33     Number of subaddresses per TDC module (8)
34     Actual CNA Camac address
35     Actual branch address

36     Camac CNA address of current Camac transfer
37     Camac branch address + Camac function*8
38     Number of errors of Camac transfers
39     Last Camac CNA address causing an error
40     Last Camac branch causing an error

41     Address of ATDC bank (points to second part of length word)
42     Address of BITS data-1
43
 :     Not used
 :
48
49     Address of length word for all data
50     No. of total words if no bits were sat
51     Address of bits bank
 :     Parameters of BITS bank
 :     Parameters for readout: CNA
 :     B
 :     CNA  }
 :     B    }    for each 24 bit pattern unit
 :
 :     0      At end of bit readout
 :
 :     Parameters of ATDC bank
 :     No. of bits in first group
 :     CNA  }
 :     B    }  of ADC.  If no ADC is used for this group = 0
 :     CNA  }
 :     B    }  of TDC.  If no TDC is used for this group = 0
 :     :
 :     :
       -1     At end of ADC readout.
```

Registers used by the program

0 Not used

1 No. of previous bit in the group. This register is used together with register 15 to compute the next CNA address for ADCs or TDCs.

2 Address register for data written into memory. The Camac wait and readout routine use this register to store data into memory.

3 Address register to read parameters from the memory

4 Contains the bit data word. This register is shifted to test for bits

5 Sratch register

6 Address register pointing to Camac CNA

7 Scratch register

8 Register indicates whether an ADC or TDC should be read out and if a Camac cycle is started

9 This register keeps the number of bits in the group

A Number of shifts executed with register 4. This register is used to check if the next bitword should be used

B Scratch register

C Address register to pointers in ATDC bank

E Not used

F Counter inside the group

Appendix B

The monitor and test program

In this section we describe briefly the possibilities and commands of the program for loading and checking the microprocessor.
The program name is SIM-PROG and can be loaded by any user. It has the micro-program and a standard parameter set as default in a BLOCK DATA.

READ-MIC Read microprogram from disc file. If no filename is given. the previous file name will be used.

READ-PAR Read parameter list PARLIS from disc

INITCAM Initialize the microprocessor. The parameters and the microprogram are loaded into memory. The processor is set to wait.

START The processor is started. If it is connected to the internal clock the microprogram is executed

RESETMEC Resets the microprocessor to zero address. This command is executed only if the processor is running with it's own clock. Otherwise you have to execute one cycle by pressing the return button

READ-ADR Reads the actual address of the microprocessor. With this command one can check whether the processor has finished. It can be given if the processor is running or waiting

TEST-LAM Test LAM1 or LAM2 interrupts
PERMANEN The processor is started. After 40 msec it is reset, LAMs are cleared and then started again

The following commands are useful for program developing. The processor is clocked externally under program control.

RUN-MEC Gives one clock signal after another to processor and reads the micro code address. If this address is equal to the break-point no more pulses are generated, and the last address and no. of cycles are printed

BREAKPNT Set an octal address to finish the running of the processor

LOOP For a given number of loops one clock pulse is generated after another. At each cycle the address, the microcode and information of the internal bus is printed

CONTINUE (Return)	Generates one clock cycle,reads the next address and the internal bus
RAMCAMAC	Set the CAMAC address for the RAM-prom-simulator
MECCAMAC	Set the CAMAC address for the MEC-prom-simulator
READ-MEM	Read the parameter and data memory of the processor from lower to upper address and print the information
WRITEMEM	Write test data into the memory
READ-RAM	Read the loaded microcode from the promsimulator
UPDATMIC	Change microcode in octal format without using the assembler. If only some bits or addresses need to be replaced this command can be used. It asks for the octal line number, prints out the old contents and reads in octal format the undated information. Afterwords an INIT must be given to load the processor with the updated code
UPDATPAR	Change parameter list PARLIS. First one can change the number of groups and then edit each line. If the line number is zero this task returns. An INIT must be given afterwards
LIST-MIC	Prints the microprogram in binary format
LIST-PAR	" " parameters PARLIS
PROTOCOL	One can change the output file with this command to LINE-PRINTER or TERMINAL
MEM-MEC	A pattern is loaded into the processor's memory and verified
MEM-RAM	The prom-simulator's memory is checked
OUTBLOCK	A FORTRAN BLOCK DATA program with the microcode in DATA statements is generated
OUTPARDA	The parameters are written to a direct access file.

Appendix C

We describe in this chapter the circuit diagrams and the hardware realization of the following three devices:

1. Microprocessor
2. PROM simulator with single register
3. PROM simulator with complete memory

The CAMAC instructions for all three devices are summarized in Table C1.

C1 MEC Microcomputer

The MEC microcomputer is a two slot wide CAMAC module which resides inside the Fisher/GEC-Elliot system crate. Apart from the standard CAMAC connections and the arbitration highway it has one Lemo input to indicate an event and another one to clear the device. Fig. C.1 shows the photograph of the processor equipped with PROMs. For program development and testing the PROMs can be replaced by a PROM simulator via cables.

On one board we have the arithmetic and logic unit ALU (Fig. C.2), the sequencer with test multiplexer and hardware counter (Fig. C.3) and the PROMs (Fig. C.4). This board is connected via 4 flat cables to the memory part: Fig. C.5 shows the memory for parameters and data which can be accessed by the processor or via CAMAC. Part of this board is the CAMAC standard decoder (Fig. C.6). The bus management for the arbitration highway, the CAMAC data register and the circuits for the CAMAC functions are presented in Fig. C.7. The arrangement of the electrical parts on both plates can be seen in Fig. C.8 and C.9.

C2 PROM simulator with single register

This module is three CAMAC slots wide with sockets on the front for the PROMs. Using this module the online computer reads the PROM address, transfers a micro-instruction to the registers and generates one clock cycle. Because the timing in this nearly static mode is not critical the cables between this simulator and the processor can be long (~1 m). Fig. C.10 shows the circuit diagram of this simple device.

C3 The PROM simulator with full memory

This PROM simulator is a single slot wide unit and contains a 1K/64 bit word memory which can be accessed by the microprocessor via short (20 cm) cables replacing the PROMs. Short cables are necessary because the processor should run with this memory also at full speed (200 nsec cycle time). If a microprocessor with more than 64 bits/word is tested one can use two PROM simulators in parallel. A photograph of the simulator together with the cables is shown in Fig. C.11 and a circuit diagram in Fig. C.12. Memory chips are available over a wide speed range from 70 nsec to 450 nsec. Seen from the online computer the memory is organized as 4K/16 bit word and is addressed via a counter so that DMA transfers can be used. A LEMO connector can be used to clock the microprocessor. The address of the microprocessor is displayed on LEDs and readable via CAMAC. The arrangement of the electronic circuits can be seen in Fig. C.13 and the no. of pieces needed to build the simulator in Table C2.

Table 1: Components in the TASSO experiment and their readout electronics

	Component	Readout electronics
1)	4 Beam pipe counters with a phototube at each end	8 Pattern units 8 ADCs 8 TDCs
2)	Proportional chamber with 4 layers of anode wires (4 x 480 = 1920 anodes) and 8 layers of cathode strips (8 x 120 = 960 cathode strips)	2880 wire addresses
3)	Drift chamber with 15 layers with 72 to 240 wires	2340 drift chamber TDCs
4)	48 inner time-of-flight counters with a phototube at each end and one mean timer	144 Pattern units 96 ADCs 96 TDCs
5)	24 endcap time-of-flight counters	48 Pattern units 48 ADCs 48 TDCs
6)	8 barrel liquid Argon submodules with 156 big towers, 636 small towers, 504 z-strips, 72 Φ-strips, (1368 x 8 = 10944 channels)	10944 ADCs
7)	2 liquid Argon endcaps with 216 big towers, 864 small towers, 288 Ψ-strips, 522 R-strips (1890 x 2 = 3780 channels)	3780 ADCs
8)	4 planar dirft chambers (128 + 32 wires)	640 drift chamber TDCs
9)	2 Čerenkov counters (Aerogel: 192 channels, CO_2: 64 channels, Freon: 64 channels	320 ADCs (LRS2280)
10)	2 x 48 hadron arm time-of-flight counters with one phototube on each end and one meantimer	240 Pattern units 192 ADCs (LRS2280) 192 TDCs
11)	2 x 80 shower counters	160 ADCs (LRS2280)
12)	Muon chambers (720 + 720 + 336 x 4 + 480 x 2)	3744 addresses
13)	2 x 16 scintillation counters forward detector	32 Pattern units
14)	4 forward detector proportional chambers with 384 channels	1536 addresses
15)	Leadglass blocks forward detector	96 Pattern units 96 ADCs

Table 2: Format of an event after formatting in the online computer

The information is grouped into Banks

Bankname	Contents
'EVNT'	Indicates an event. Run number, day, time ....
'PROP'	Information of the cylindrical proportional chamber
'DRFK'	Driftchamber information (cylindrical and plane chamber)
'PRFO'	Forward detector proportional chamber
'MUON'	Muon chamber information
'BITS'	Pattern units from TOFs, Forward detector, ...
'ATDC'	ADCs and TDCs
'ADCN'	Northarm LRS2280 ADCs
'ADCS'	Southarm LRS2280 ADCs
'LIAR'	Barrel liquid argon
'LIAE'	Endcap liquid argon
'CAMC'	Counter and timing information
'LUMI'	Luminosity monitor

Table 3: Example for a typical bank

Word	Contents	
-6	'AT'	Bank information
-5	'DC'	
-4	0	
-3	1	
-2	0	
-1	0	
+0	0	
1	Length of the bank in 32 bit words	
2	No. of double reals	Information for conversion routines
3	No. of 32 bit integers	
4	No. of characters	
5	No. of 16 bit integers	
6	0	
7	No. of words read	
8	No. of groups	
9	Pointer to first group	
10	Pointer to second group	
⋮		
41	Pointer to last group	
42	Pointer to end of data + 1	
→43	Member no. inside group	
44	ADC	
45	TDC	
→46	Member no. inside group	
47	ADC	
48	TDC	
⋮		

Table 4: Table of bits in the microinstruction word

Bit no.	Contents
0 ... 9	Label field for addresses in loops, subroutine calls or for constants used by the program
10 ... 13	Field for instructions of the sequencer CCU. These bits are connected to the AM29811 which is connected to the AM2911, the counter and test multiplexer
14	The bit inverts the information of the test multiplexer
15 .. 18	This field selects the condition in the test multiplexer like A ≥ B, 'true', CAMAC Q, Data ready, F = 0, Carry,...
19	Multiply bit to control the ALU function in multiply operations
20 .. 23	Address of A register for the ALU
24 .. 27	Address of B register for the ALU
28 .. 30	Field for the ALU source. A-register, B-register, Q-register, Data, 'logical zero'
31 .. 33	Field for the ALU operation like add, subtract, and, or, exclusive or
34 .. 36	Destination of the ALU result Data can be stored into the register file and into the Q-register. Before storing them they can be shifted
37 .. 38	Multiplexer of carry in. This may be high, low, the carry and $\overline{\text{carry}}$
39 .. 41	Multiplexer for carry out. The output of this multiplexer can be tested or shifted to the ALU. The inputs are high, last carry, low, bit 0 of Q-register, the sign bit, ...
42 .. 47	Shift control bits. These bits are connected to the shift multiplexer S1...S6 in Fig.6
48 .. 50	These bits control the sources of the internal bus (ALU, Memory, Camac data register)
51 .. 53	Destination of the internal bus (ALU, Memory, Camac command register)
54	Error bit to signal a LAM2
55	Bit indicating a request to the system crate
56	Busy bit
57	Memory select bit if data are transferred to or from memory
58	Write enable bit to write data to memeory
59 .. 63	Not used.

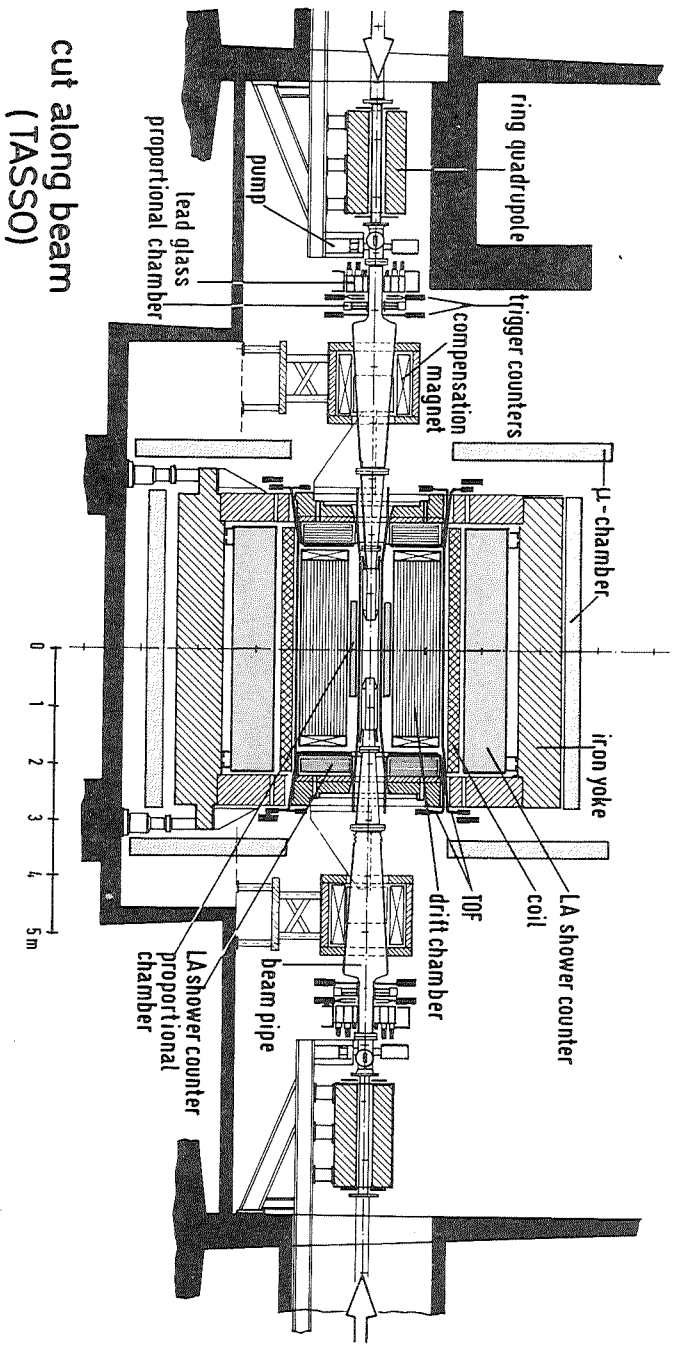


Fig. 1 Side view of the TASSO experiment. Positrons are coming from the left, electrons from the right. The interaction region is surrounded by proportional chambers, a drift chamber and time-of-flight counters. Outside the coil are liquid Argon shower counters. The forward detector is used for luminosity measurement and electron detection for $\gamma\gamma$ physics.

Mnemonic	Mnemonic	AQ	AB	ZQ	ZB	ZA	DA	DQ	DZ
	I_{210} OCTAL	0	1	2	3	4	5	6	7
	OCTAL 543 / ALU Source / ALU Function	A, Q	A, B	0, Q	0, B	0, A	D, A	D, Q	D, 0
ADD	0 — C_n = L, R Plus S, C_n = H	A+Q A+Q+1	A+B A+B+1	Q Q+1	B B+1	A A+1	D+A D+A+1	D+Q D+Q+1	D D+1
SUBR	1 — C_n = L, S Minus R, C_n = H	Q−A−1 Q−A	B−A−1 B−A	Q−1 Q	B−1 B	A−1 A	A−D−1 A−D	Q−D−1 Q−D	−D−1 −D
SUBS	2 — C_n = L, R Minus S, C_n = H	A−Q−1 A−Q	A−B−1 A−B	−Q−1 −Q	−B−1 −B	−A−1 −A	D−A−1 D−A	D−Q−1 D−Q	D−1 D
OR	3 — R OR S	A ∨ Q	A ∨ B	Q	B	A	D ∨ A	D ∨ Q	D
AND	4 — R AND S	A ∧ Q	A ∧ B	0	0	0	D ∧ A	D ∧ Q	0
NOTR	5 — $\overline{R}$ AND S	$\overline{A} \wedge Q$	$\overline{A} \wedge B$	Q	B	A	$\overline{D} \wedge A$	$\overline{D} \wedge Q$	0
EXOR	6 — R EX-OR S	A ∀ Q	A ∀ B	Q	B	A	D ∀ A	D ∀ Q	D
EXNO	7 — R EX-NOR S	$\overline{A \forall Q}$	$\overline{A \forall B}$	$\overline{Q}$	$\overline{B}$	$\overline{A}$	$\overline{D \forall A}$	$\overline{D \forall Q}$	$\overline{D}$

+ = Plus, − = Minus, ∨ = OR, ∧ = AND, ∀ = EX-OR

Source Operand and ALU Function Matrix.

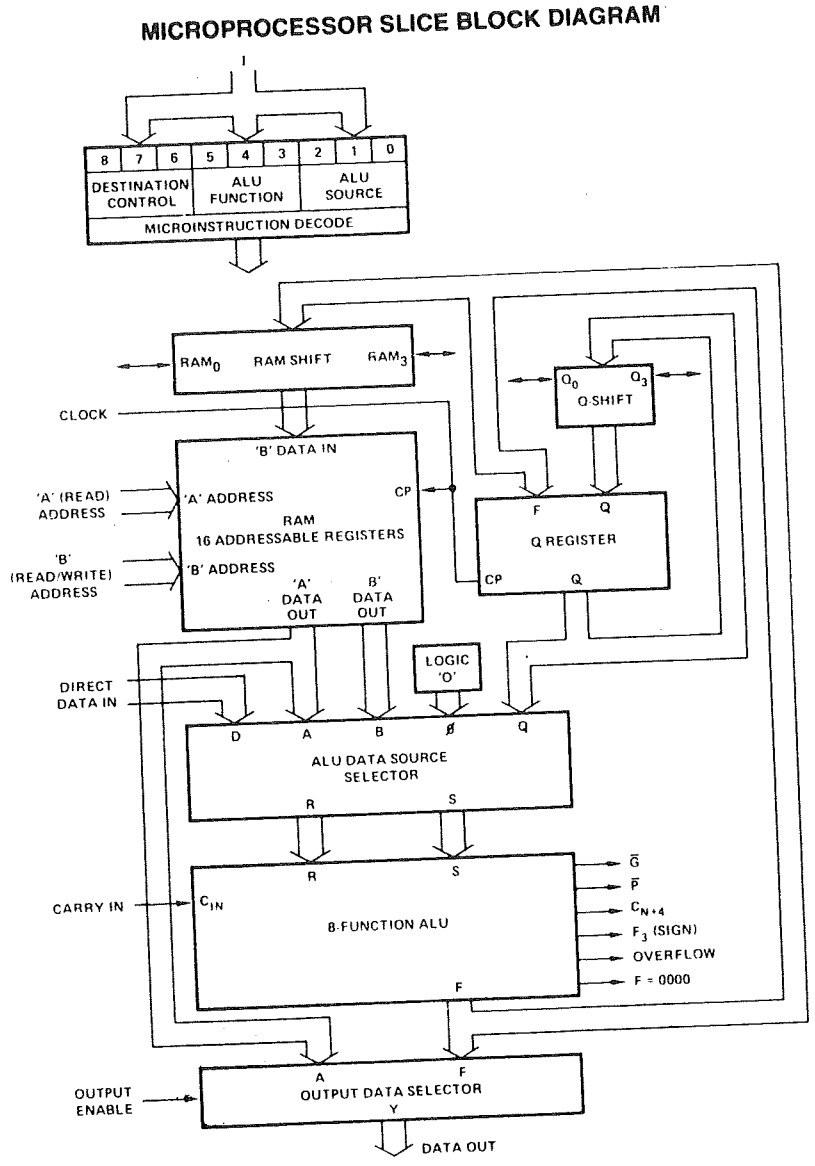


Fig. 5 Arithmetic and logical unit.

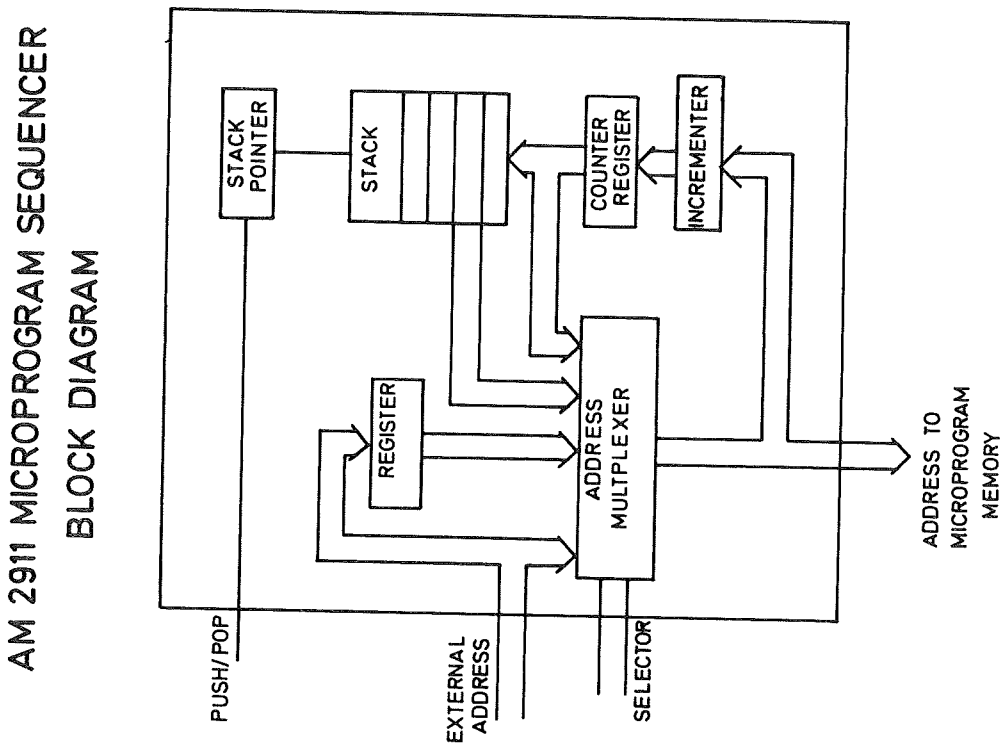


Fig. 7 The next address in the microprogram may be taken from the pipeline register (external address), the stack or the internal counter.

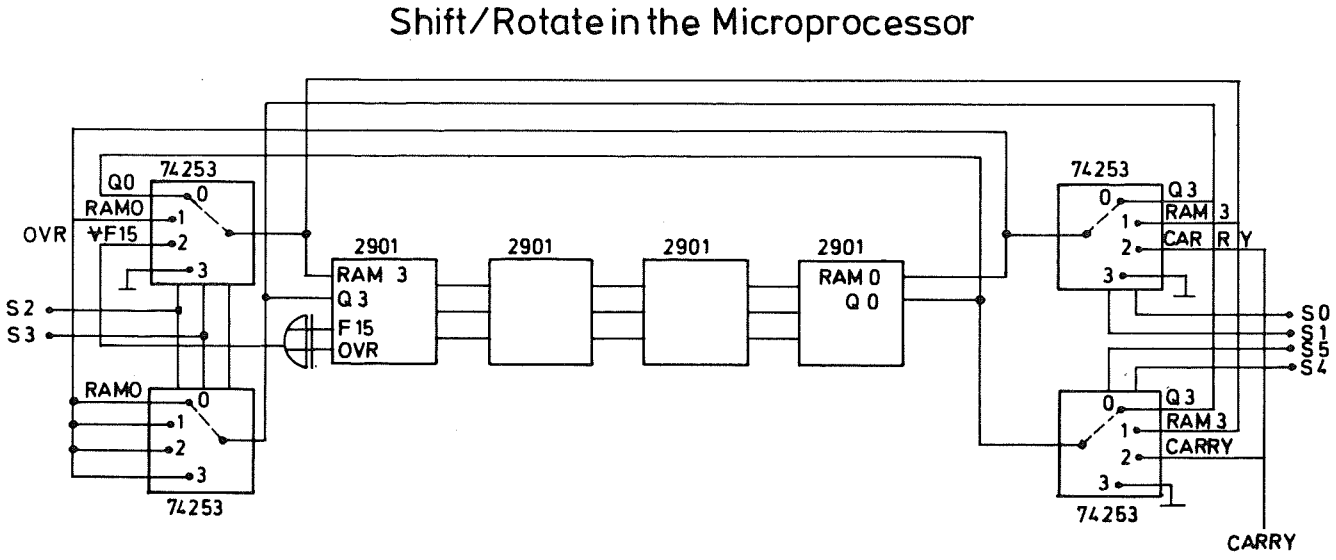


Fig. 6 Shift/rotate wiring in the microprocessor. The contents of the most or least significant bit in the registers depend on the selectors steered by the microcode bits 50-55.

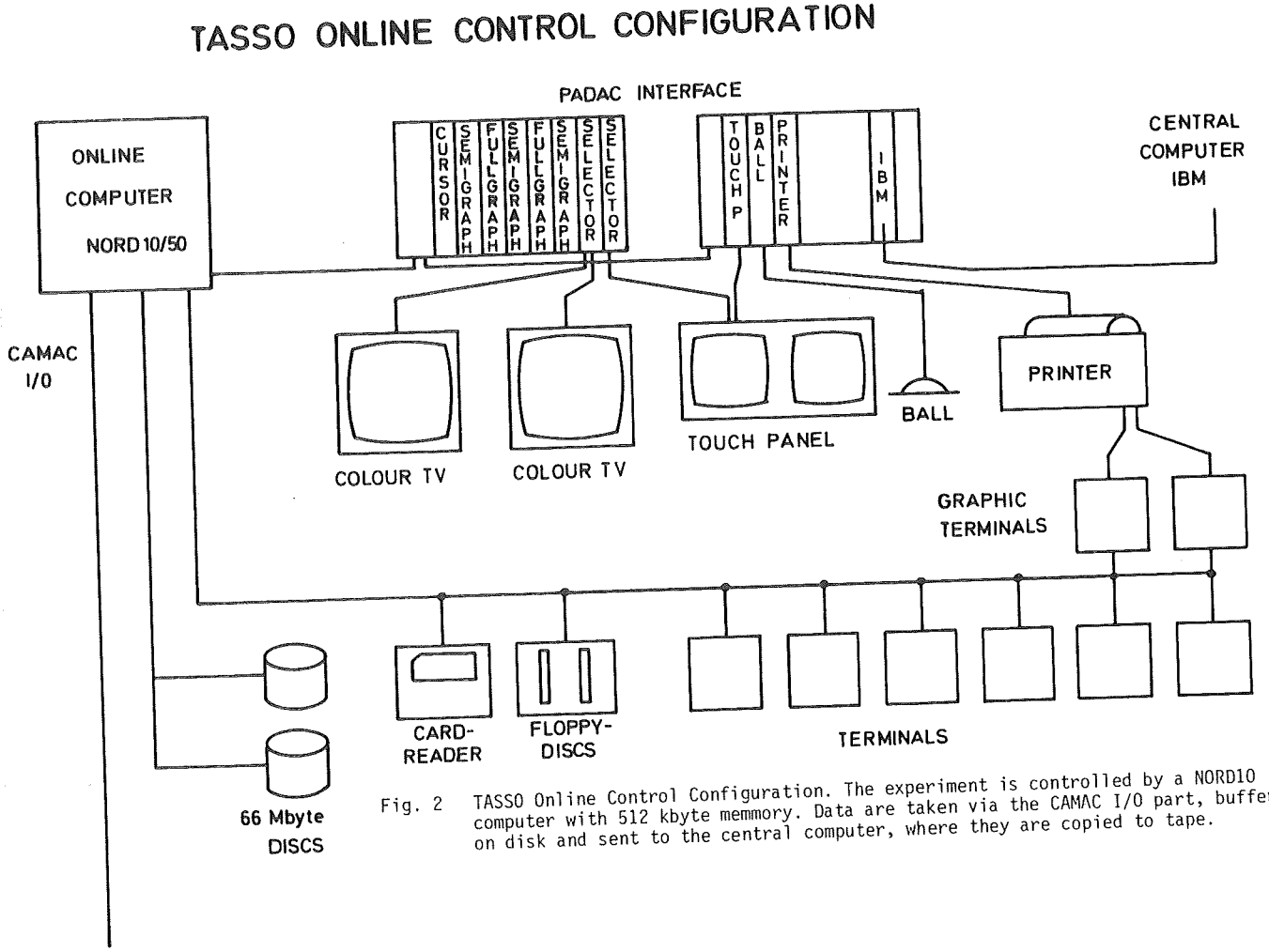


Fig. 2 TASSO Online Control Configuration. The experiment is controlled by a NORD10 computer with 512 kbyte memmory. Data are taken via the CAMAC I/O part, buffered on disk and sent to the central computer, where they are copied to tape.

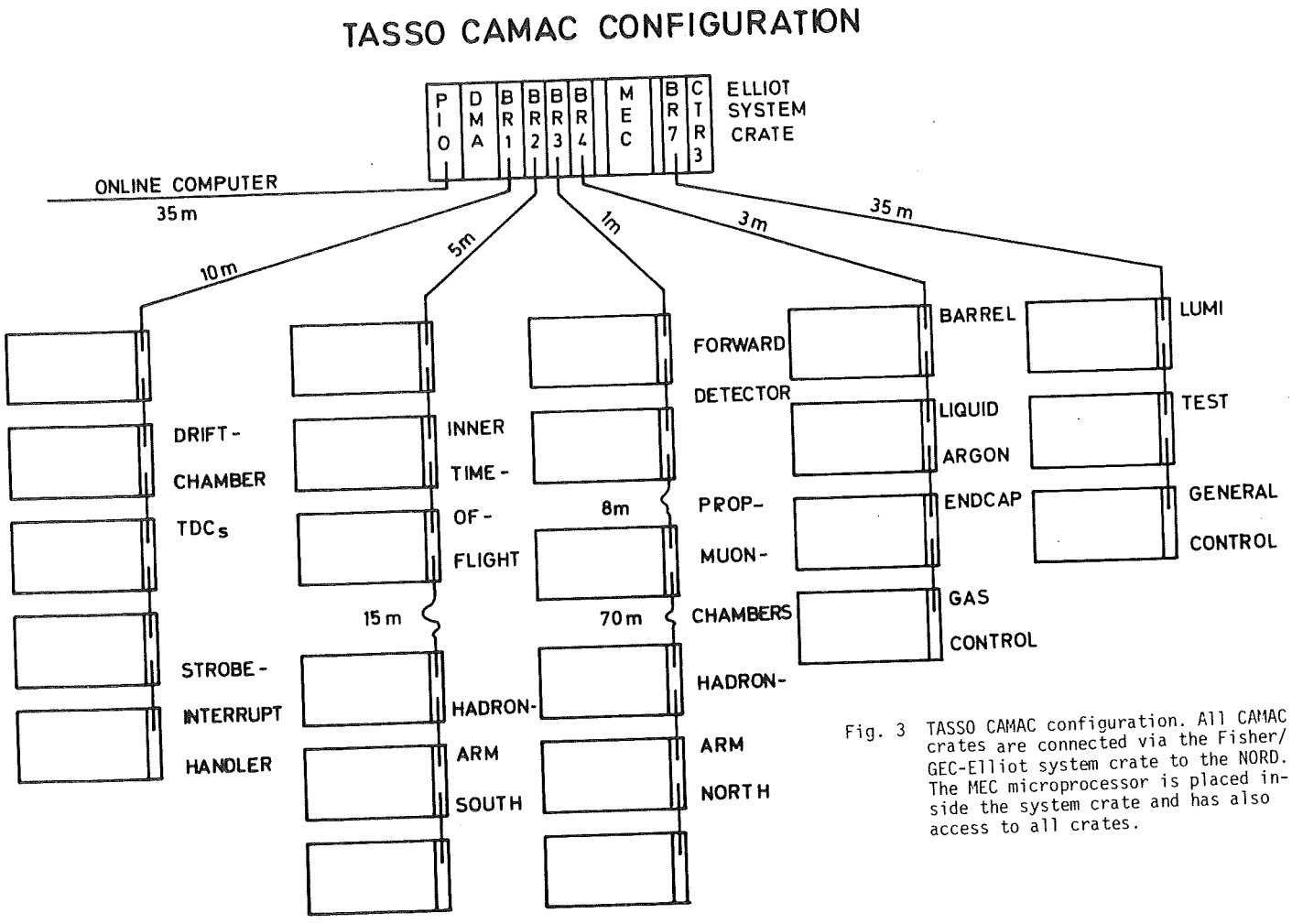


Fig. 3 TASSO CAMAC configuration. All CAMAC crates are connected via the Fisher/GEC-Elliot system crate to the NORD. The MEC microprocessor is placed inside the system crate and has also access to all crates.

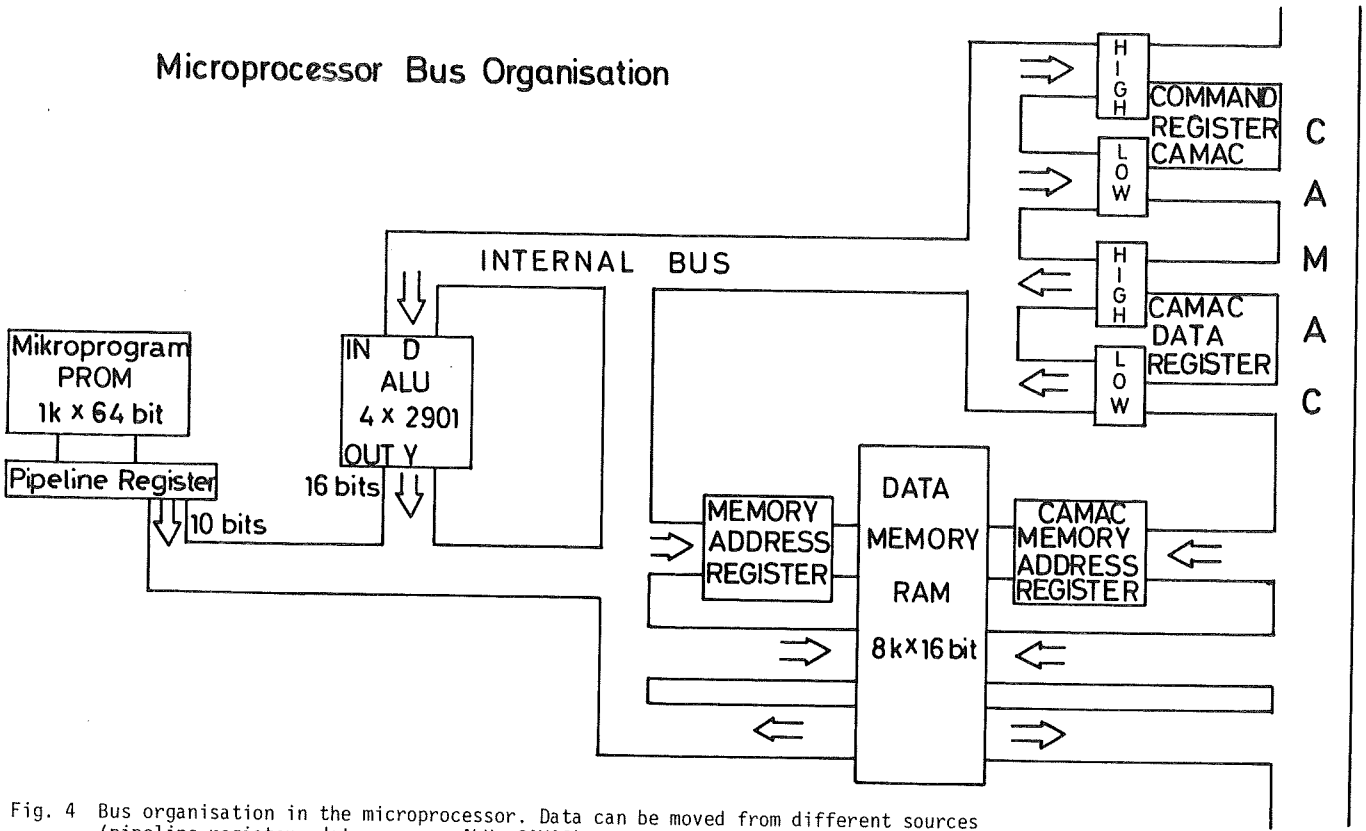
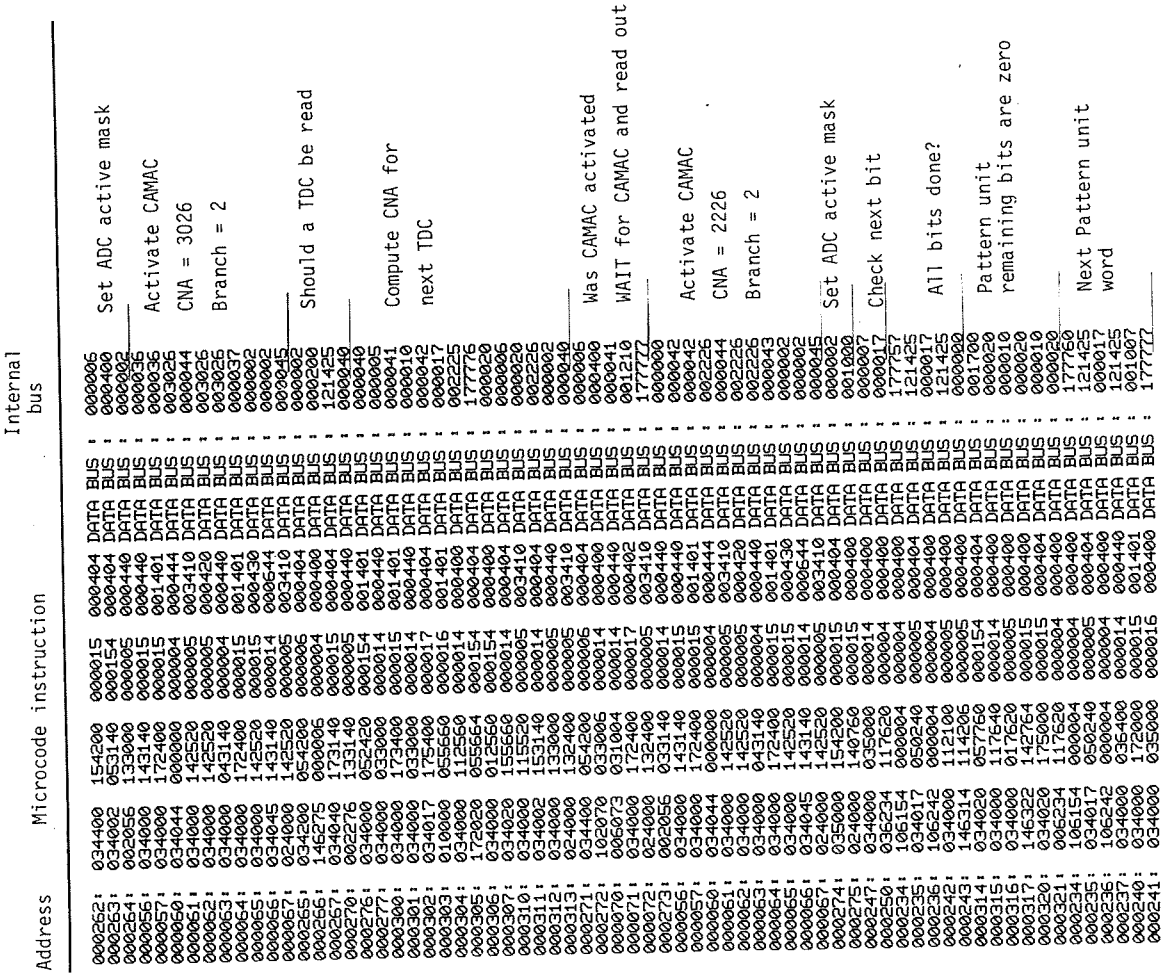


Fig. 4 Bus organisation in the microprocessor. Data can be moved from different sources (pipeline register, data memory, ALU, CAMAC) to various destinations (ALU, memory, address and CAMAC command registers).



Address	Microcode instruction				Internal bus	
000262:	034400	154200	000015	000404 DATA BUS	000006	Set ADC active mask
000263:	034002	053140	000154	000404 DATA BUS	000400	
000264:	002056	133000	000005	000440 DATA BUS	000002	
000056:	034000	143140	000015	000440 DATA BUS	000036	Activate CAMAC
000057:	034000	172400	000015	001401 DATA BUS	000036	
000250:	034044	000000	000004	000444 DATA BUS	003026	CNA = 3026
000061:	034000	142520	000005	003410 DATA BUS	000044	
000062:	034000	142520	000005	000420 DATA BUS	003026	
000063:	034000	043140	000004	000440 DATA BUS	003026	Branch = 2
000064:	034000	172400	000015	001401 DATA BUS	000037	
000065:	034000	142520	000015	000430 DATA BUS	000002	
000066:	034045	143140	000014	000644 DATA BUS	000002	
000067:	024000	142520	000005	003410 DATA BUS	003045	
000265:	034200	054200	000006	000404 DATA BUS	000002	Should a TDC be read
000266:	146275	000006	000004	000400 DATA BUS	000200	
000267:	034040	173140	000015	000404 DATA BUS	121425	
000270:	002276	133140	000005	000440 DATA BUS	000040	
000276:	034000	052420	001154	001401 DATA BUS	000005	
000277:	034000	033000	000014	000440 DATA BUS	000041	Compute CNA for
000300:	034000	173400	000015	001401 DATA BUS	000010	
000301:	034000	033000	000014	000440 DATA BUS	000042	next TDC
000302:	034017	175400	000017	000404 DATA BUS	000017	
000303:	010000	055660	000016	001401 DATA BUS	002225	
000304:	034000	112560	000014	000400 DATA BUS	177776	
000305:	172020	055664	000154	000404 DATA BUS	000020	
000306:	034000	012560	000154	000400 DATA BUS	000006	
000307:	034020	155660	000014	000404 DATA BUS	002226	
000310:	034000	115520	000005	003410 DATA BUS	000020	
000311:	034002	153140	000014	000404 DATA BUS	000002	
000312:	034000	133000	000005	000440 DATA BUS	000040	Was CAMAC activated
000313:	024000	132400	000005	003410 DATA BUS	000006	
000271:	034400	054200	000006	000404 DATA BUS	000400	
000272:	102070	031004	000014	000440 DATA BUS	000041	WAIT for CAMAC and read out
000070:	006073	031004	000017	000402 DATA BUS	001210	
000071:	034000	172400	000005	003410 DATA BUS	177777	
000072:	024000	132400	000005	003410 DATA BUS	000000	
000273:	002056	031140	000014	000440 DATA BUS	000042	Activate CAMAC
000056:	034000	143140	000015	000440 DATA BUS	000042	
000057:	034000	172400	000015	001401 DATA BUS	002226	
000060:	034044	000000	000004	000444 DATA BUS	000044	CNA = 2226
000061:	034000	142520	000005	003410 DATA BUS	002225	
000062:	034000	142520	000014	000420 DATA BUS	000044	Branch = 2
000063:	034000	043140	000004	000440 DATA BUS	002225	
000064:	034000	172400	000015	001401 DATA BUS	000043	
000065:	034000	142520	000015	000430 DATA BUS	000002	
000066:	034045	143140	000014	000644 DATA BUS	000002	
000067:	024000	142520	000005	003410 DATA BUS	000045	
000274:	035000	154200	000015	000404 DATA BUS	000002	Set ADC active mask
000275:	024000	140760	000015	000400 DATA BUS	001000	
000247:	034000	035000	000014	000400 DATA BUS	000007	Check next bit
000250:	036234	117620	000004	000400 DATA BUS	000017	
000234:	106154	000004	000004	000400 DATA BUS	177757	
000235:	034017	050240	000005	000404 DATA BUS	121425	
000236:	106242	000004	000004	000400 DATA BUS	000017	All bits done?
000242:	146314	112100	000005	000400 DATA BUS	121425	
000243:	034020	114206	000005	000404 DATA BUS	000000	
000314:	034020	057760	000154	000404 DATA BUS	001700	Pattern unit
000315:	034000	117640	000014	000400 DATA BUS	000020	remaining bits are zero
000316:	034000	017620	000005	000400 DATA BUS	000010	
000317:	146322	142764	000015	000400 DATA BUS	000020	
000320:	034020	175000	000015	000404 DATA BUS	000010	
000321:	106154	006234	000004	000400 DATA BUS	000020	
000234:	106154	117620	000004	000400 DATA BUS	177760	
000235:	034017	050240	000005	000404 DATA BUS	121425	Next Pattern unit
000236:	106242	000004	000004	000400 DATA BUS	000017	word
000237:	034000	036400	000014	000400 DATA BUS	121425	
000240:	034000	172000	000015	001401 DATA BUS	001007	
000241:	034000	035000	000016	000400 DATA BUS	177777	

Fig. 9 Example for program flow.
The processor is clocked by the online computer. After each cycle the address and the internal bus of the processor is read and listed together with the microcode instruction.

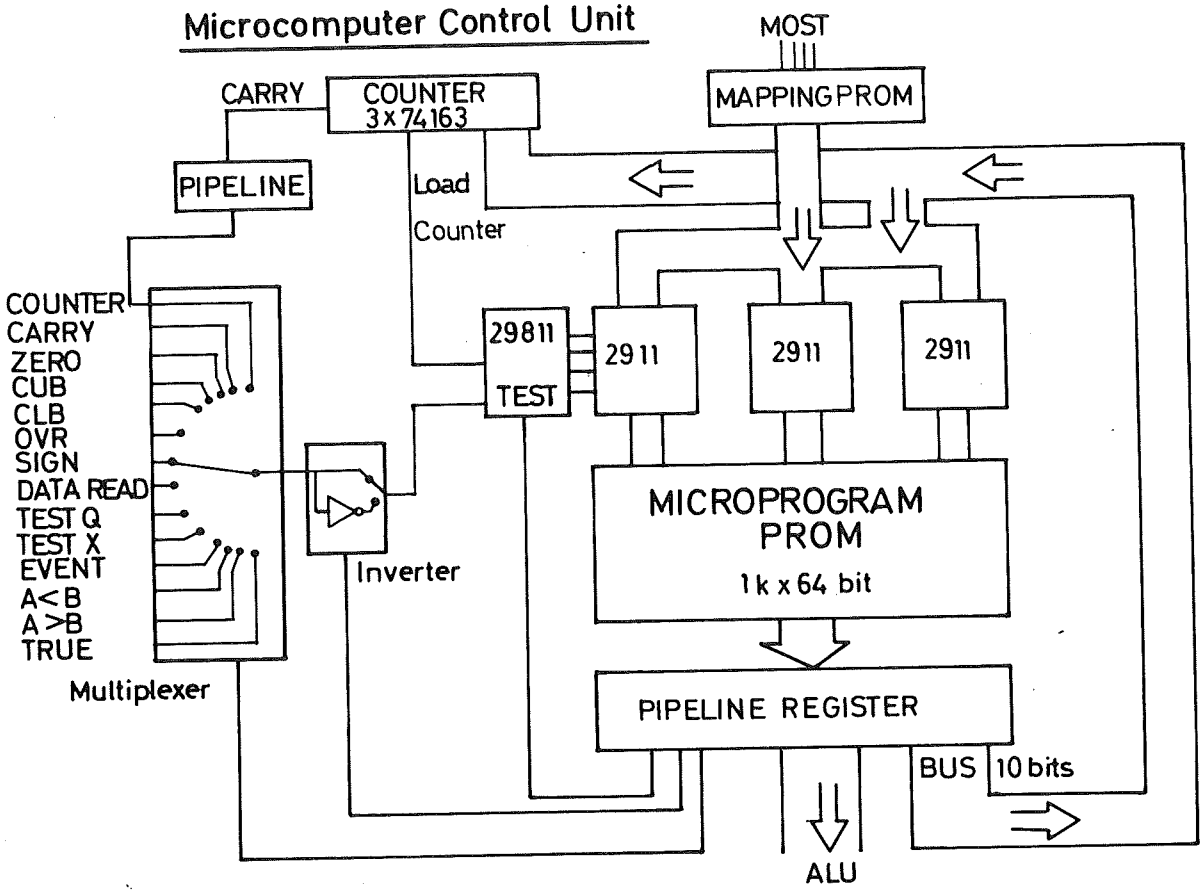


Fig. 8 Three AM2911 sequencers are used in the processor. The next instruction may depend on the test selected by the multiplexer. A hardwired counter can be used for loops and shifts.

33 groups are used to read out

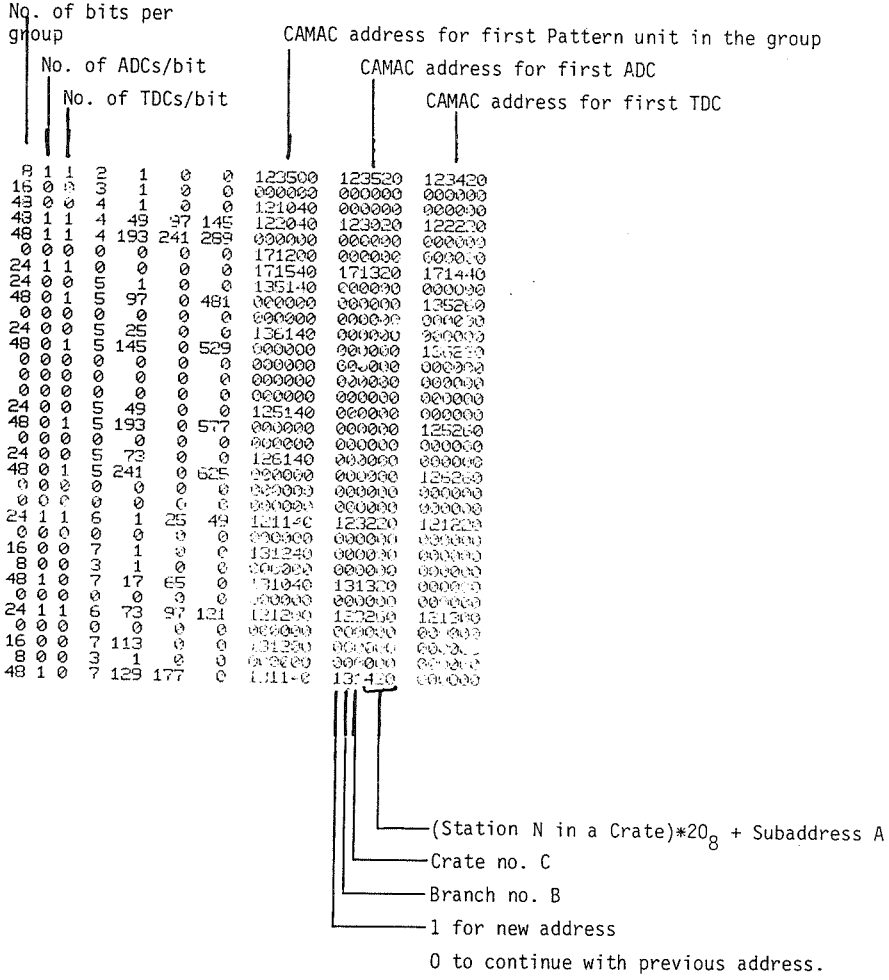


Fig. A1 Example of parameters to read Pattern units, ADCs and TDCs. These parameters will be reformatted and loaded into the memory at the beginning of each run.

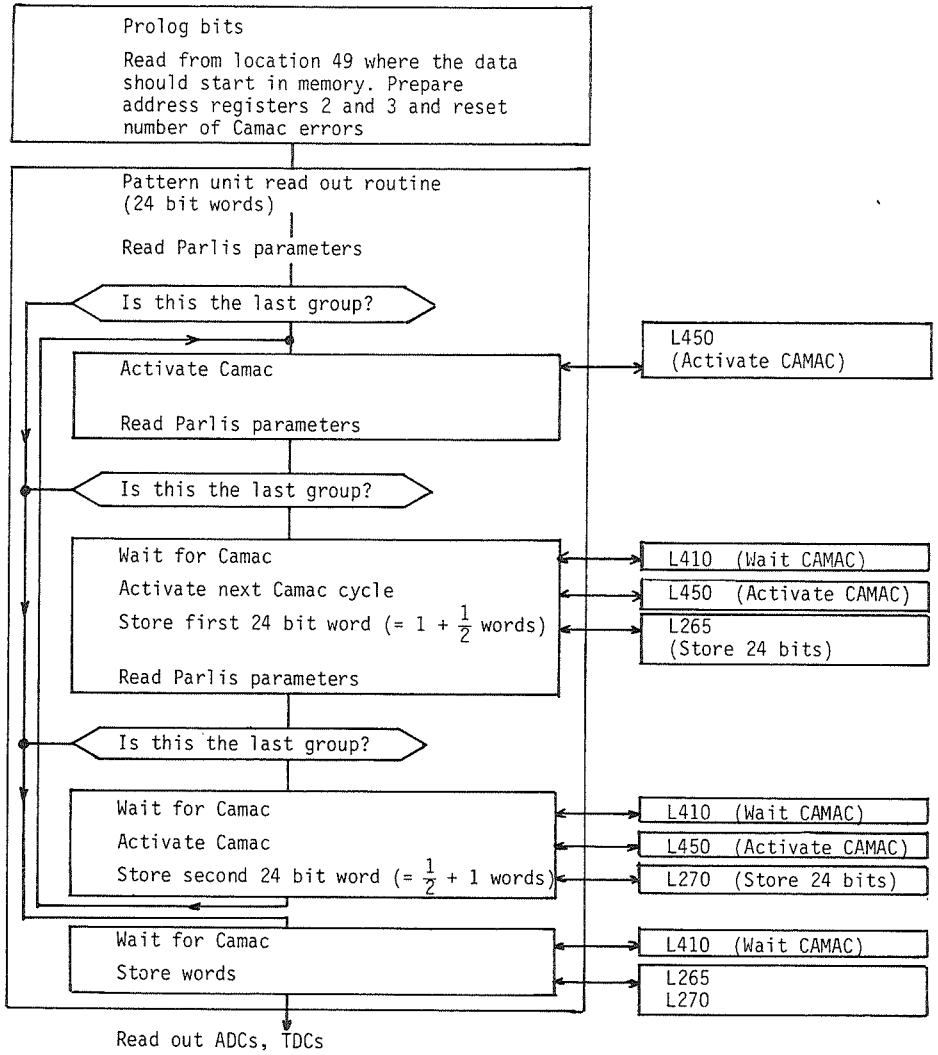


Fig. A2a Flow chart of a program to read pattern units, ADCs and TDSc. Each box corresponds to a subroutine, CAMAC cycles and program execution are overlapped (Activate CAMAC, Wait for CAMAC).

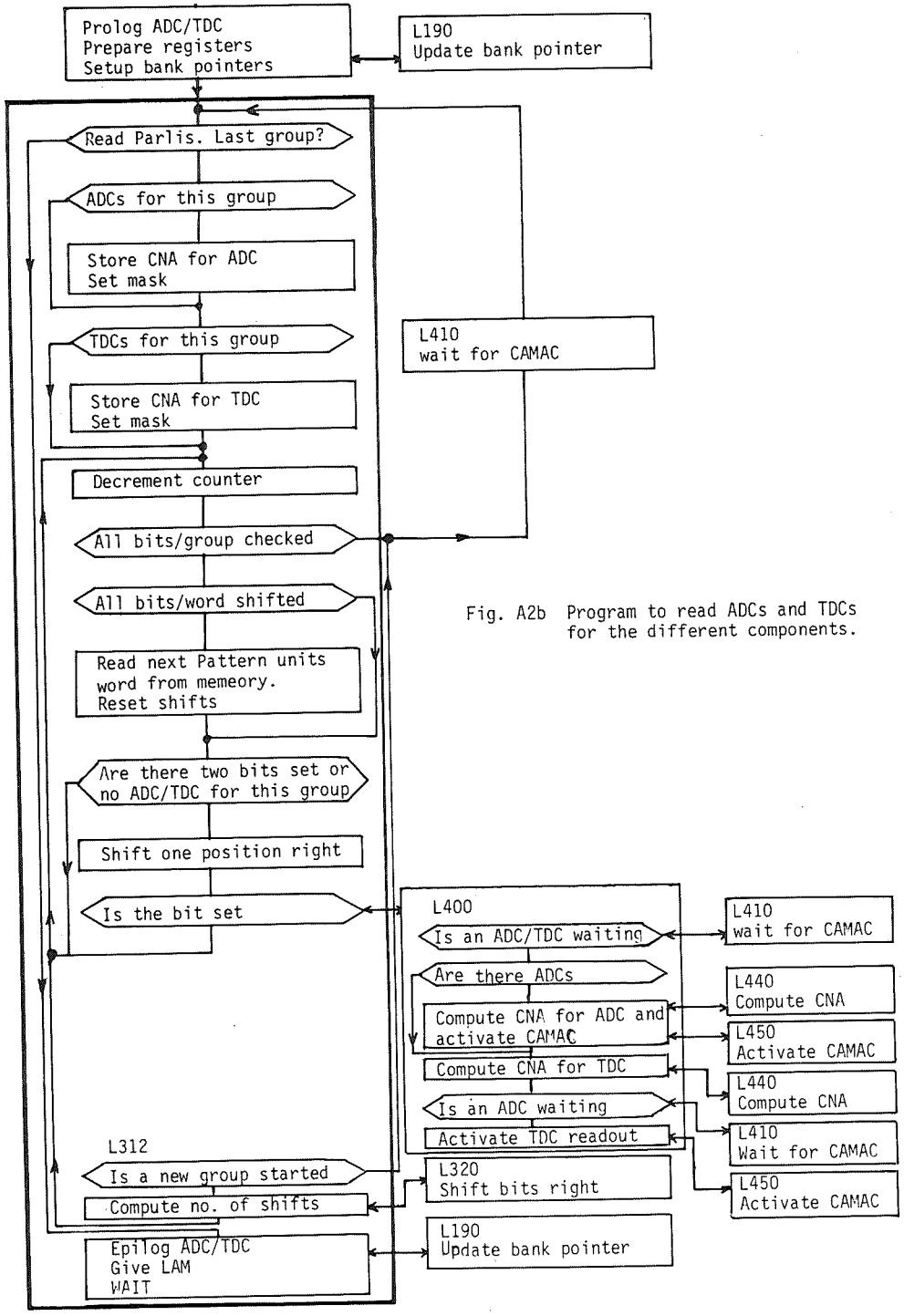


Fig. A2b Program to read ADCs and TDCs for the different components.

```
0015 *
0015 *          SUBROUTINE TO READ 24 BIT WORD BITS
0015 *-------------------------------------------------------------------
0015 *
0015 *          INCREMENT PARLIS POINTER
0015 250   LRF  ADD  ZB 3   MAR ALS                          CH
0016 *          READ CNA
0016       LRF  OR   DZ 5   ALU MMS                              S
0017 *          IF ZERO WE REACHED THE END OF DATA
0017       LRF  OR   ZA 6 3 MAR ALS CJP  L252                FEQZ
0020 *          REGISTER 6 POINTS NOW TO CNA
0020 *          ACTIVATE CAMAC
0020       NOP                    CJS  L450                  TRUE
0021 *          INCREMENT REGISTER 3
0021 251   LRF  ADD  DA 3 3 ALU PIP      D2                  CL
0022 *          READ NEXT PARLIS
0022       LRF  OR   ZA 6 3 MAR ALS
0023 *          PARLIS INTO REGISTER 5
0023       LRF  OR   DZ 5   ALU MMS                              S
0024 *          IS THE END OF DATA REACHED
0024       NOP                    CJP  L254                  FEQZ
0025 *          WAIT FOR CAMAC
0025       NOP                    CJS  L410                  TRUE
0026 *          READ HIGH ORDER DATA FROM CAMAC
0026       LQ   EXNO DZ     ALU CAH
0027 *          MASK OFF FIRST BITS
0027       LQ   AND  DQ     ALU PIP      0377
0030 *          SAVE REGISTER 5 (CAMAC INFORMATION) AND ACTIVATE NEXT CAMAC
0030       LRF  OR   ZA 7 5              CJS  L450             TRUE
0031 *          RESTORE INFORMATION IN REG. 5 AND STORE 1. 24 BIT WORD INTO
0031 *          MEMORY
0031       LRF  OR   ZA 5 7              CJS  L265
0032 *          INCREMENT REGISTER 3
0032       LRF  ADD  DA 3 3 ALU PIP      D2                  CL
0033 *          READ NEXT PARLIS
0033       LRF  OR   ZA 6 3 MAR ALS                              S
0034       LRF  OR   DZ 5   ALU MMS
0035 *          HAVE WE REACHED THE END OF PARLIS
0035       NOP                    CJP  L256                  FEQZ
0036 *          WAIT FOR CAMAC
0036       NOP                    CJS  L410                  TRUE
0037 *          READ HIGH ORDER FROM CAMAC
0037       LQ   EXNO DZ     ALU CAH
0040 *          MASK OFF FIRST BITS
0040       LQ   AND  DQ     ALU PIP      0377
0041 *          STORE REG.5 AND ACTIVATE CAMAC
0041       LRF  OR   ZA 7 5              CJS  L450             TRUE
0042 *          STORE CAMAC INFORMATION INTO MEMORY
0042       LRF  OR   ZA 5 7              CJS  L270             TRUE
0043 *          NEXT PARAMETER
0043       NOP                    CJP  L251                  TRUE
0044 *
0044 *
0044 *----------
0044 *          END OF THE GAME
0044 *          WAIT FOR CAMAC
0044       NOP                    CJS  L4.)                  TRUE
0045 *          READ HIGH ORDER PART
0045       LQ   EXNO DZ     ALU CAH
0046 *          MASK OFF FIRST BITS
0046       LQ   AND  DQ     ALU PIP      0377
0047 *          STORE CAMAC
0047       NOP                    CJS  L265                  TRUE
0050 *          JUMP TO EXIT
0050       NOP                    CJP  L252                  TRUE
0051 *
0051 *          WAIT FOR CAMAC

0051 256   NOP                    CJS  L410                  TRUE
0052 *          READ HIGH ORDER DATA FROM CAMAC
0052       LQ   EXNO DZ     ALU CAH
0053 *          MASK OFF FIRST BITS
0053       LQ   AND  DQ     ALU PIP      0377
0054 *          STORE THE DATA
0054       NOP                    CJS  L270                  TRUE
0055 *
0055 252   NOP                    CJP  L301                  TRUE
0056 *----------
0056 *-------------------------------------------------------------------
```

Fig. A3 Microprogram to read out 24 bit pattern units as shown in Fig. A2a.

ADDR	BIT	WSBHE	BSD	BSS	SHIFT	CM	CI	DST	FUN	SRC	REGB	REGA	M	TMUX	CCU	PIPELINE
0000	0000000100	000	100	000000	000	00	011	011	111	0011	0000	0	00000	1110	0000110001
0001	0000000100	100	000	000000	000	00	001	011	011	0011	0000	0	00000	1110	0000000000
0002	0000001100	000	001	000000	000	00	011	011	111	0010	0000	0	00000	1110	0000000000
0003	0000000100	100	000	000000	000	00	011	000	011	0011	0000	0	00000	1110	0000000000
0004	0000001100	000	001	000000	000	00	011	011	111	0101	0000	0	00000	1110	0000000000
0005	0000000100	100	000	000000	000	00	001	011	011	0010	0000	0	00000	1110	0000000000
0006	0000011100	001	000	000000	000	00	001	011	011	0101	0000	0	00000	1110	0000000000
0007	0000000100	000	100	000000	000	11	011	000	101	0010	0010	0	00000	1110	0000001110
0010	0000000100	100	100	000000	000	00	001	011	111	0000	0000	0	00000	1110	0000101010
0011	0000011100	001	000	000000	000	00	001	011	011	0010	0000	0	00000	1110	0000000000
0012	0000000100	100	100	000000	000	00	001	011	111	0000	0000	0	00000	1110	0000100110
0013	0000011100	001	000	000000	000	00	001	100	011	0000	0000	0	00000	1110	0000000000
0014	0000000100	100	000	000000	000	00	011	000	011	0011	0000	0	00000	1110	0000000000
0015	0000000100	100	000	000000	000	00	011	000	011	0011	0000	0	00000	1110	0000000000
0016	0000001100	000	001	000000	000	00	011	011	111	0101	0000	0	00000	1110	0000000000
0017	0000000100	100	000	000000	000	00	011	011	100	0110	0011	0	11011	0011	0000101101
0020	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0000101110
0021	0000000100	000	100	000000	000	11	011	000	101	0011	0011	0	00000	1110	0000000010
0022	0000000100	100	000	000000	000	00	011	011	100	0110	0011	0	00000	1110	0000000000
0023	0000001100	000	001	000000	000	00	011	011	111	0101	0000	0	00000	1110	0000000000
0024	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	11011	0011	0000100100
0025	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0000111000
0026	0000000100	000	011	000000	000	00	000	111	111	0000	0000	0	00000	1110	0000000000
0027	0000000100	000	100	000000	000	00	000	100	110	0000	0000	0	00000	1110	0011111111
0030	0000000100	000	000	000000	000	00	011	011	100	0111	0101	0	00000	0001	0000101110
0031	0000000100	000	000	000000	000	00	011	011	100	0101	0111	0	00000	0001	0001001101
0032	0000000100	000	100	000000	000	11	011	000	101	0011	0011	0	00000	1110	0000000010
0033	0000000100	100	000	000000	000	00	011	011	100	0110	0011	0	00000	1110	0000000000
0034	0000001100	000	001	000000	000	00	011	011	111	0101	0000	0	00000	1110	0000000000
0035	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	11011	0011	0000101001
0036	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0000111000
0037	0000000100	000	011	000000	000	00	000	111	111	0000	0000	0	00000	1110	0000000000
0040	0000000100	000	100	000000	000	00	000	100	110	0000	0000	0	00000	1110	0011111111
0041	0000000100	000	000	000000	000	00	011	011	100	0111	0101	0	00000	0001	0000101110
0042	0000000100	000	000	000000	000	00	011	011	100	0101	0111	0	00000	0001	0001010000
0043	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0011	0000010001
0044	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0000111000
0045	0000000100	000	011	000000	000	00	000	111	111	0000	0000	0	00000	1110	0000000000
0046	0000000100	000	100	000000	000	00	000	100	110	0000	0000	0	00000	1110	0011111111
0047	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0001001101
0050	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0011	0000101101
0051	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0000111000
0052	0000000100	000	011	000000	000	00	000	111	111	0000	0000	0	00000	1110	0000000000
0053	0000000100	000	100	000000	000	00	000	100	110	0000	0000	0	00000	1110	0011111111
0054	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0001	0001010000
0055	0000000100	000	000	000000	000	00	001	000	000	0000	0000	0	00000	0011	0001011011
0056	0000000100	100	000	000000	000	00	011	011	100	0110	0110	0	00000	1110	0000000000
0057	0000001100	000	001	000000	000	00	011	011	111	0101	0000	0	00000	1110	0000000000
0060	0000000100	100	100	000000	000	00	001	000	000	0000	0000	0	00000	1110	0000100100

Fig. A4 Binary dump of the assembler. Lines 15-55 correspond to the program shown in Fig. A3.

000000	000001	177777	177777	177777	177777	177777	177777	177777	177777	177777	
000012	177777	177777	177777	177777	177777	177777	177777	177777	177777	177777	
000024	177777	177777	177777	177777	177777	177777	177777	177777	000000	000014	Parameters 28 - 42
000036	001420	000003	000000	000010	001300	000002	006407	000002	000005	005407	
000050	000002	001063	001001	177777	177777	177777	177777	177777	177777	000763	
000062	000151	000764	000001	000002	041111	052123	000001	000030	000026	000054	BITS Parameters
000074	000000	000051	003500	000002	001040	000002	001100	000002	002040	000002	
000106	002100	000002	002140	000002	002200	000002	005140	000003	005200	000003	CAMAC addresses for pattern units
000120	005240	000003	006140	000003	006200	000003	006240	000003	005140	000002	
000132	005200	000002	005240	000002	006140	000002	006200	000002	006240	000002	
000144	001140	000002	001240	000003	001040	000003	001100	000003	001200	000002	
000156	001300	000003	001140	000003	001200	000003	000000	000001	000003	040524	ATDC Parameters
000170	042103	000001	000013	000011	000022	000000	000042	000041	000010	003520	
000202	000002	003420	000002	000020	000000	000000	000060	000000	000000	000060	CAMAC addresses and information for ADCs and TDCs
000214	003020	000002	002220	000002	000060	003120	000002	002360	000002	000000	
000226	000000	000030	000000	000000	000060	000000	005260	000003	000000	000030	
000240	000000	000000	000060	000000	006260	000003	000000	000000	000000	000030	
000252	000000	000000	000060	000000	005260	000002	000000	000030	000000	000000	
000264	000060	000000	006260	000002	000000	000000	000030	003220	000002	001220	
000276	000002	000000	000020	000000	000000	000010	000000	000000	000060	001320	
000310	000003	000000	000000	000030	003260	000002	001300	000002	000000	000020	
000322	000000	000000	000010	000000	000000	000060	001420	000002	000000	177777	
000334	177777	177777	177777	177777	177777	177777	177777	177777	177777	177777	
000752	177777	177777	177777	177777	177777	177777	177777	177777	177777	000704	
000764	041111	052123	000000	000001	000000	000000	000000	000030	000000	000026	
000776	000000	000054	000000	000051	177777	000000	000000	000000	000000	002000	BITS Bank
001010	000000	000000	000000	000000	000000	177400	177777	000377	177400	177777	
001022	177777	177777	177777	177777	177777	177777	177777	177777	177777	000377	
001034	177400	177777	000377	000000	000000	000000	000000	000000	000000	000000	
001046	000000	000000	000000	000000	000000	000001	040524	042103	000000	000001	ATDC Bank
001060	000000	000000	000000	000302	000000	000300	000000	000576	000000	000576	
001072	000041	000053	000103	000103	000103	000106	000106	000106	000106	000106	Pointer
001104	000206	000206	000206	000346	000346	000346	000346	000346	000506	000506	
001116	000506	000606	000606	000606	000606	000606	000606	000606	000606	000606	
001130	000606	000606	000606	000606	000606	000001	000005	004524	000002	000011	
001142	004524	000003	000005	004524	000004	000006	004524	000005	000001	004524	Addresses ADCs, TDCs
001154	000006	000002	004524	000007	000001	004524	000010	000007	004524	000023	
001166	000030	005042	000001	002455	000002	002430	000003	002343	000004	002301	
001200	000005	002330	000006	002332	000007	002435	000010	002343	000011	002371	
001212	000012	002350	000013	002411	000014	002462	000015	002456	000016	002462	
001224	000017	002371	000020	002361	000021	002351	000022	002260	000023	002346	
001236	000024	002312	000025	002230	000026	002252	000027	002331	000030	002246	
001250	000051	002543	000052	002527	000053	002523	000054	002526	000055	002545	
001262	000056	002520	000057	002551	000060	000000	000001	002401	000002	002354	
001274	000003	002357	000004	002310	000005	002345	000006	002245	000007	002312	
001306	000010	002262	000011	002326	000012	002251	000013	002341	000014	002277	
001320	000015	002257	000016	002331	000017	002324	000020	002314	000021	002372	
001332	000022	002472	000023	002377	000024	002312	000025	002307	000026	002221	
001344	000027	002321	000030	002227	000031	002403	000032	002441	000033	002330	
001356	000034	002350	000035	002343	000036	002327	000037	002334	000040	002360	
001370	000041	002337	000042	002353	000043	002321	000044	002314	000045	002263	
001402	000046	002247	000047	002272	000050	002264	000051	002243	000052	002350	
001414	000053	002261	000054	002234	000055	002277	000056	002272	000057	002207	
001426	000060	002207	000001	002202	000002	002505	000003	002406	000004	002376	
001440	000005	002433	000006	002446	000007	002475	000010	002501	000011	002400	
001452	000012	002534	000013	002522	000014	002426	000015	002461	000016	002376	
001464	000017	002420	000020	002430	000021	002434	000022	002552	000023	002521	
001476	000024	002415	000025	002416	000026	002434	000027	002450	000030	002455	
001510	000031	003057	000032	003016	000033	003003	000034	002750	000035	002763	
001522	000036	003012	000037	003117	000040	003062	000041	003027	000042	002775	
001534	000043	003050	000044	002772	000045	002763	000046	002666	000047	003021	
001546	000050	002702	000051	002530	000052	002455	000053	002477	000054	002543	
001560	000055	002412	000056	002474	000057	002474	000060	000000	000021	002642	
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
001604	000027	002623	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	000033	[illegible]	
001616	000034	002560	000035	002454	000036	002531	000037	002564	000040	002434	
001630	000041	002541	000042	002526	000043	002466	000044	002446	000045	002371	
001642	000046	002506	000047	002427	000050	002452	000051	002621	000052	002610	
001654	000053	002730	000054	002615	000055	002744	000056	002652	000057	002652	
001666	000060	002652	000041	000000	000042	002513	000043	002456	000044	002456	
001700	000045	002412	000046	002507	000047	002437	000050	002432	000051	002624	
001712	000052	002626	000053	002732	000054	002625	000055	002740	000056	002646	

Fig. A5 Parameters and result in the memory of the MEC.

CAMAC commands for the processor

A2 F26	Set processor to wait
A2 F24	Reset wait and start processor
A2 F0	Read data/parameters from memory and increment address register
A2 F16	Write address into memory address register
A2 F17	Write data into memory and increment address register Reading/writing to memory is possible only in wait state
A4,A5 F8	Test LAM1, LAM2
A4,A5 F10	Clear LAM1, LAM2
A4,A5 F24	Disable LAM1, LAM2
A4,A5 F26	Enable LAM1, LAM2
A6 F1	Read microprogram address
A7 F1	Read the internal bus
A9 F0	Read error register (Error, address error, 1, busy)

CAMAC commands for single word PROM-simulator

A0 F0	Read Microprogram address
A0 F16	Write Microprogramm instruction bits 0 - 15
A1 F16	" " " 16 - 31
A2 F16	" " " 32 - 47
A3 F16	" " " 48 - 63
A4 F16	Generate Clock Cycle
A5 F16	Reset Microprocessor

CAMAC single word commands for the PROM-simulator with full memory

A0 F0	Read data and increment address register
A0 F16	Write data and increment address register
A0 F17	Write address to address register
A0 F24	Disable CAMAC transfer/Enable microprocessor access
A0 F25	Perform one clock cycle for microprocessor
A0 F26	Enable CAMAC transfer/Disable micrprocessor address

Table C1. CAMAC instructions

1	24 pin Socket	
10	14 pin "	
5	16 pin "	
16	18 pin "	
21	20 pin "	
5	16 pin "Rundkontakt"socket	
1	Lemo socket	
16	4045, 2114 or 2148	Memory ICs
2	LS 244	
10	S 240	
9	LS 240	
3	LS 05	
4	LS 08	
2	LS 32	
3	LS 191	
2	LS 138	
1	LS 154	
1	LS 74	
11	LED	CQY65 (Layer No. 19/011)
2	330	Widerstandsnetzwerke single in line
1	1kΩ	
5	1KΩ	
1	68 pF	
1	39Ω	
1	220 nF	
6	22 µF	Tantal Kondensatoren
10	10 µF	Tantal Kondensatoren
34	10 µF	Keramikkondensatoren
7	Breitbanddrossel Valvo	
2	Dioden MRS10	

Table C2: No. of peaces needed for the PROM simulator

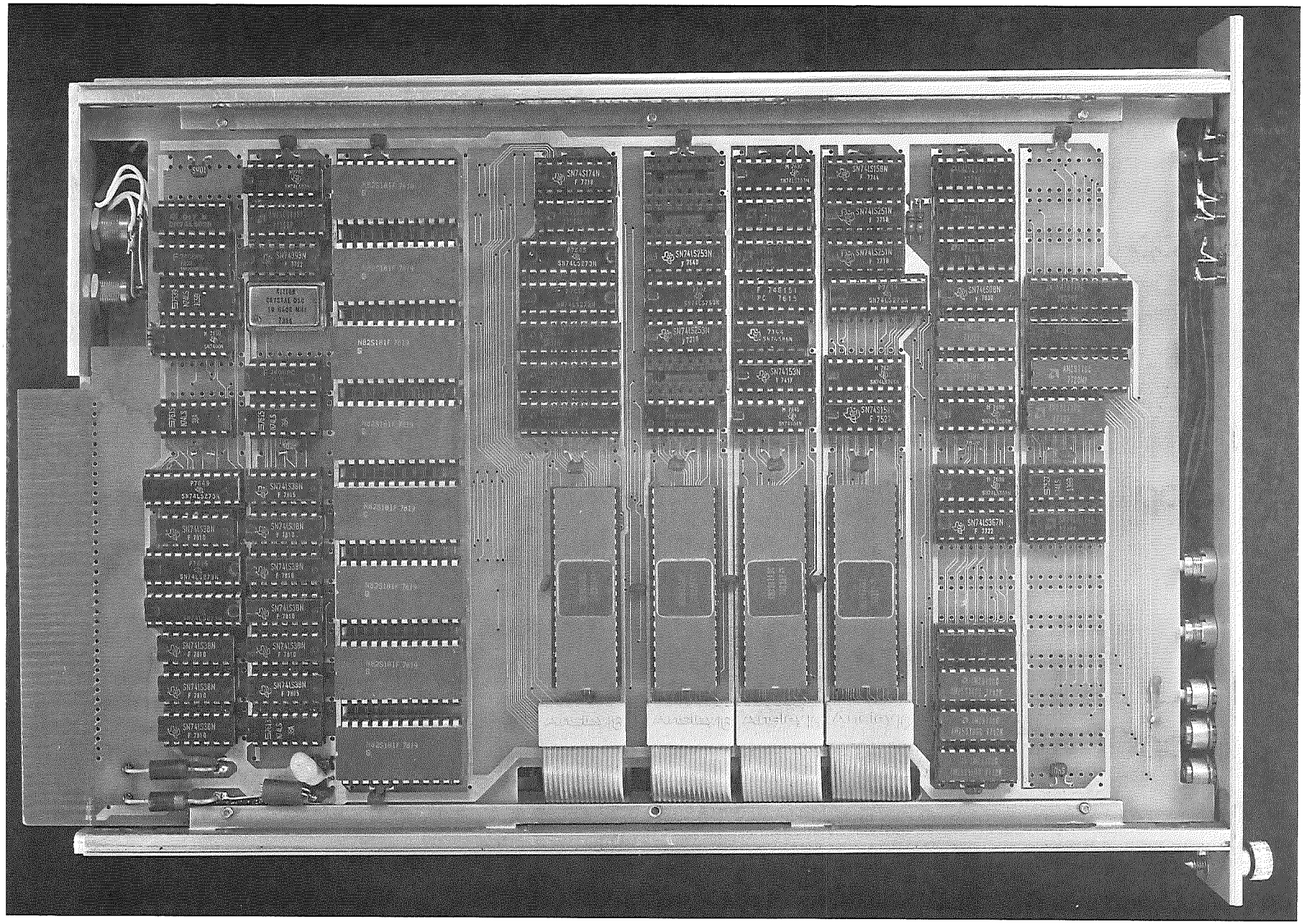


Fig. C1 Layout of the MEC processor.

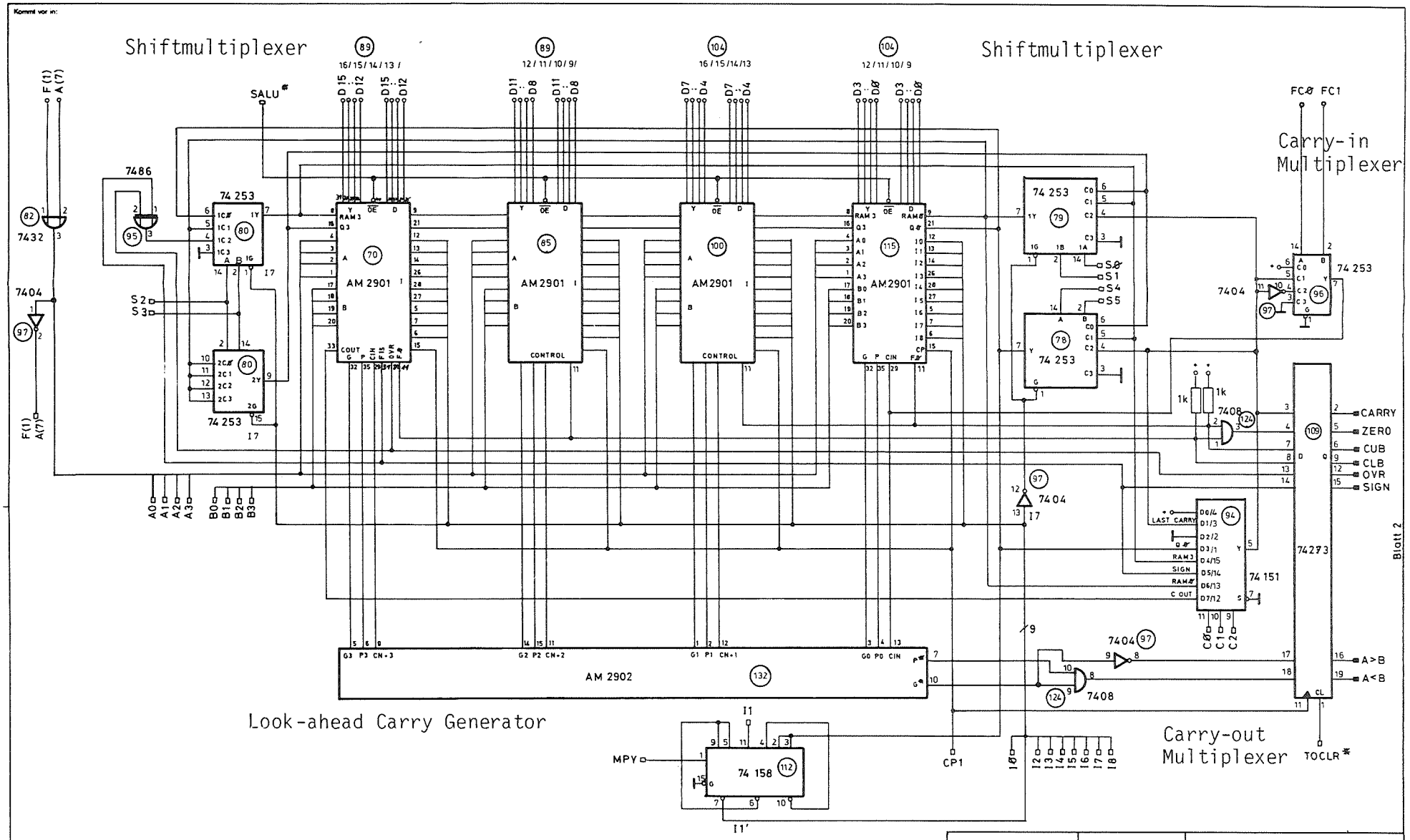


Fig. C2 Arithmetic and Logic Unit

Maße ohne Toleranzangabe DIN 7168	Arbeitspause	Maßstab	
Bearb. 73.9.78 Gepr. Norm	Datum / Name	Benennung: MEC - Prozessor ALU	
DR. B. STRUCK 2000 Tangstedt / HH		Zeichnungsnummer	Blatt 1 / 3 Bl.
Zust. Änderung	Datum Name	Ers. f.	

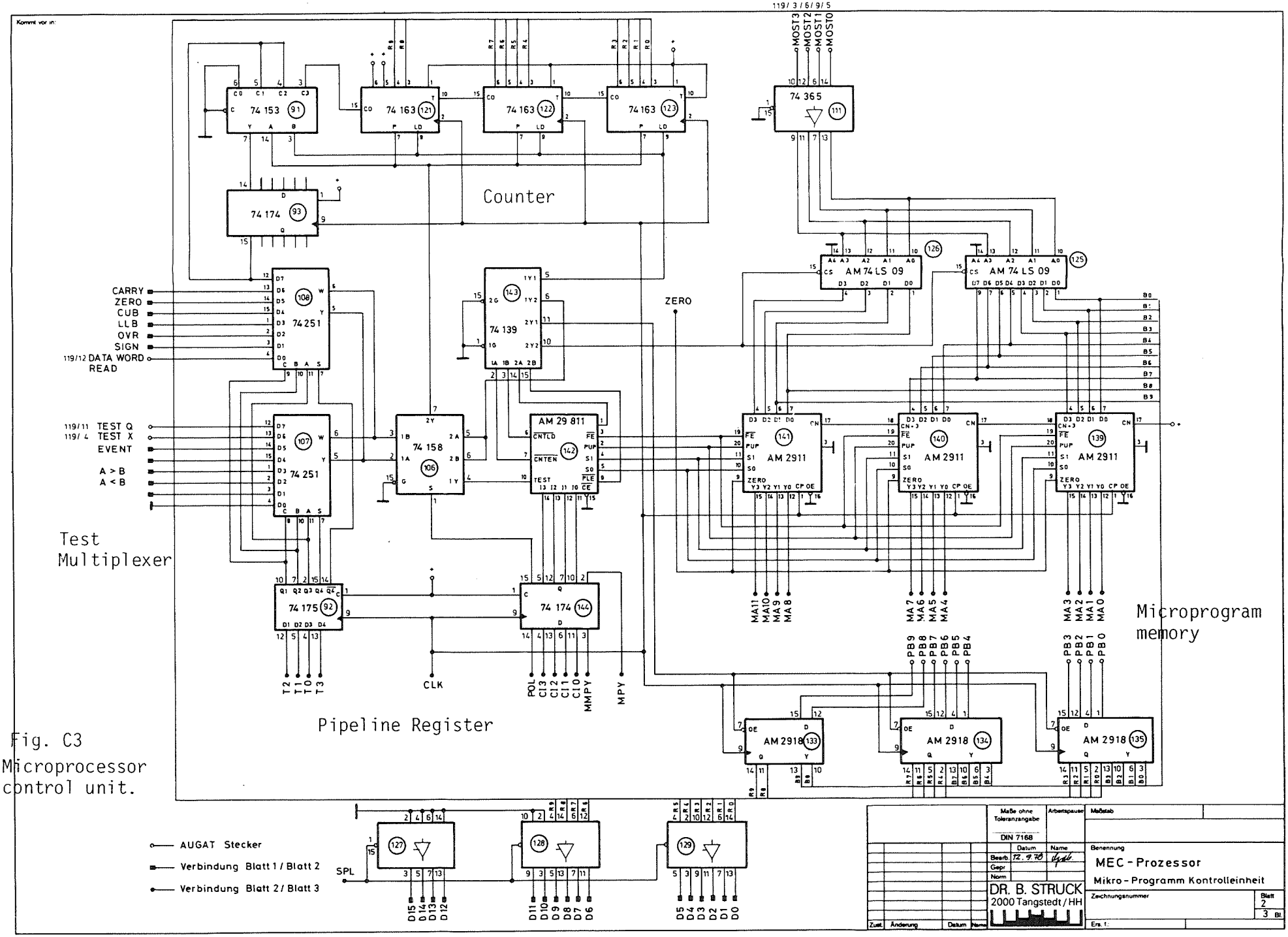


Fig. C3
Microprocessor control unit.

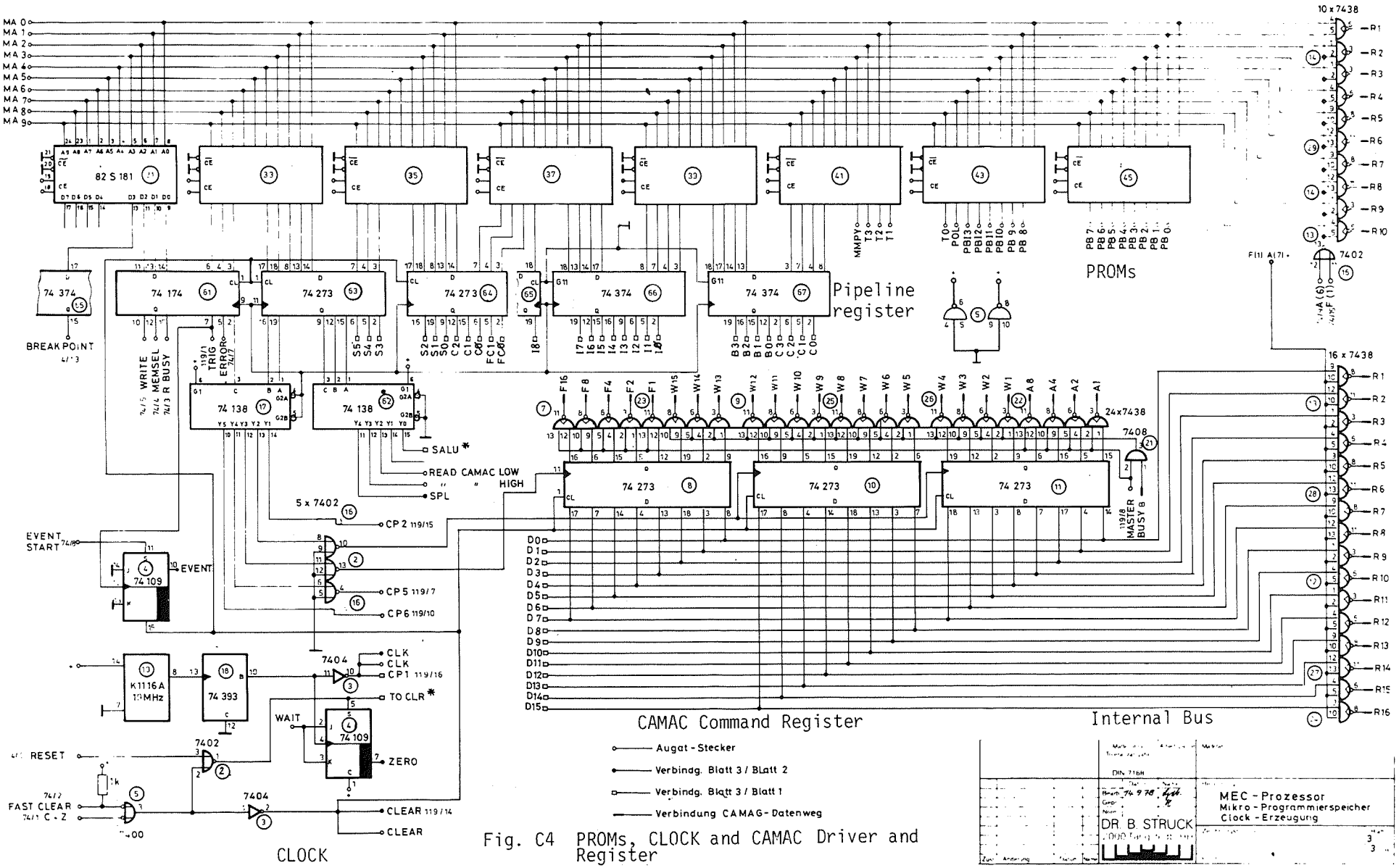
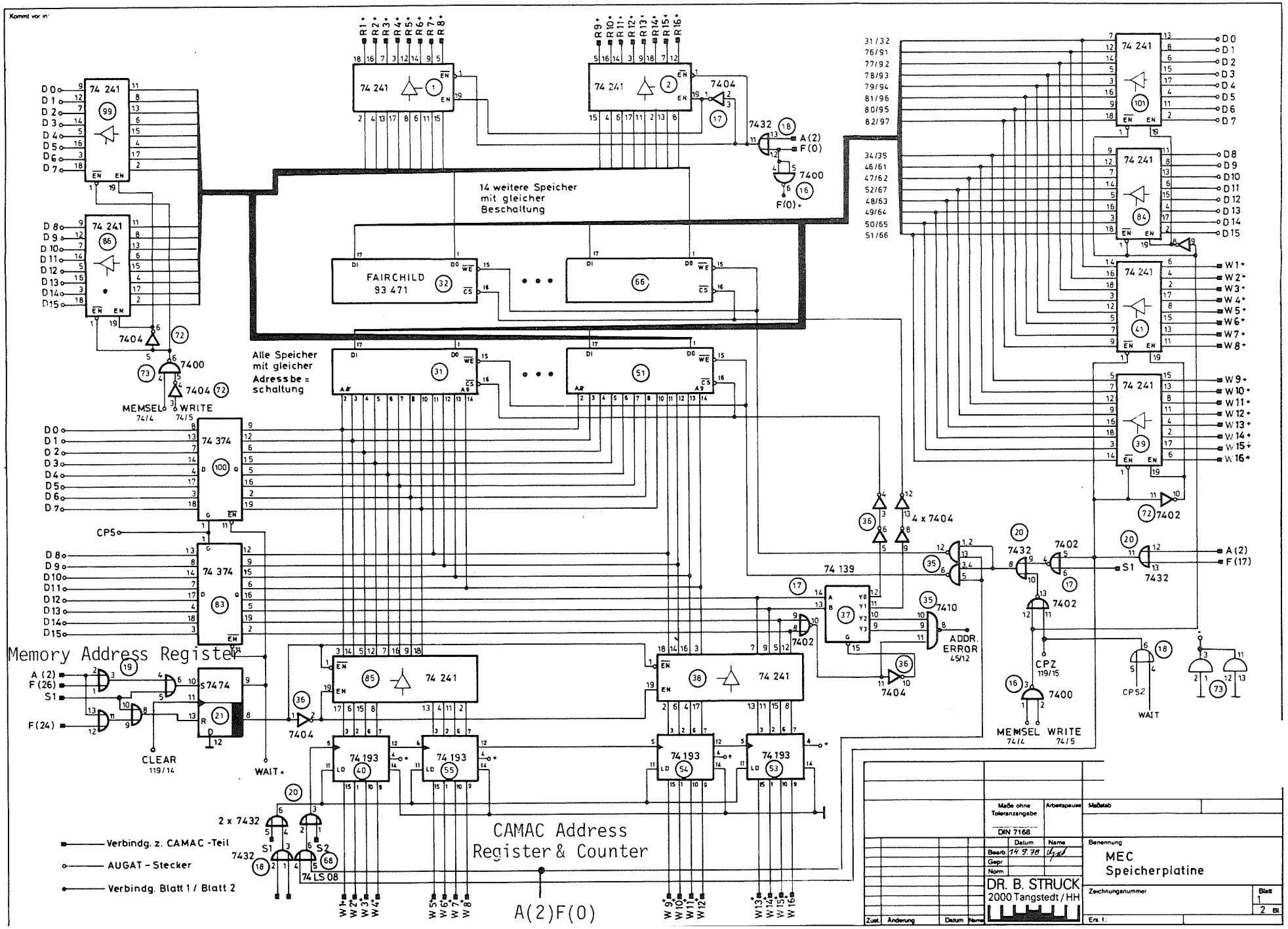


Fig. C4 PROMs, CLOCK and CAMAC Driver and Register


Kommt vor in:
14 weitere Speicher mit gleicher Beschaltung
FAIRCHILD 93 471
Alle Speicher mit gleicher Adressbe = schaltung
MEMSEL 74/4
WRITE 74/5
4 x 7404
74 139
ADDR. ERROR 45/12
CPZ 119/15
CPS2
WAIT
CLEAR 119/14
Memory Address Register
2 x 7432
74 LS 08
CAMAC Address Register & Counter
A(2)F(0)
Verbindg. z. CAMAC -Teil
AUGAT - Stecker
Verbindg. Blatt 1 / Blatt 2
DR. B. STRUCK
2000 Tangstedt / HH
MEC
Speicherplatine

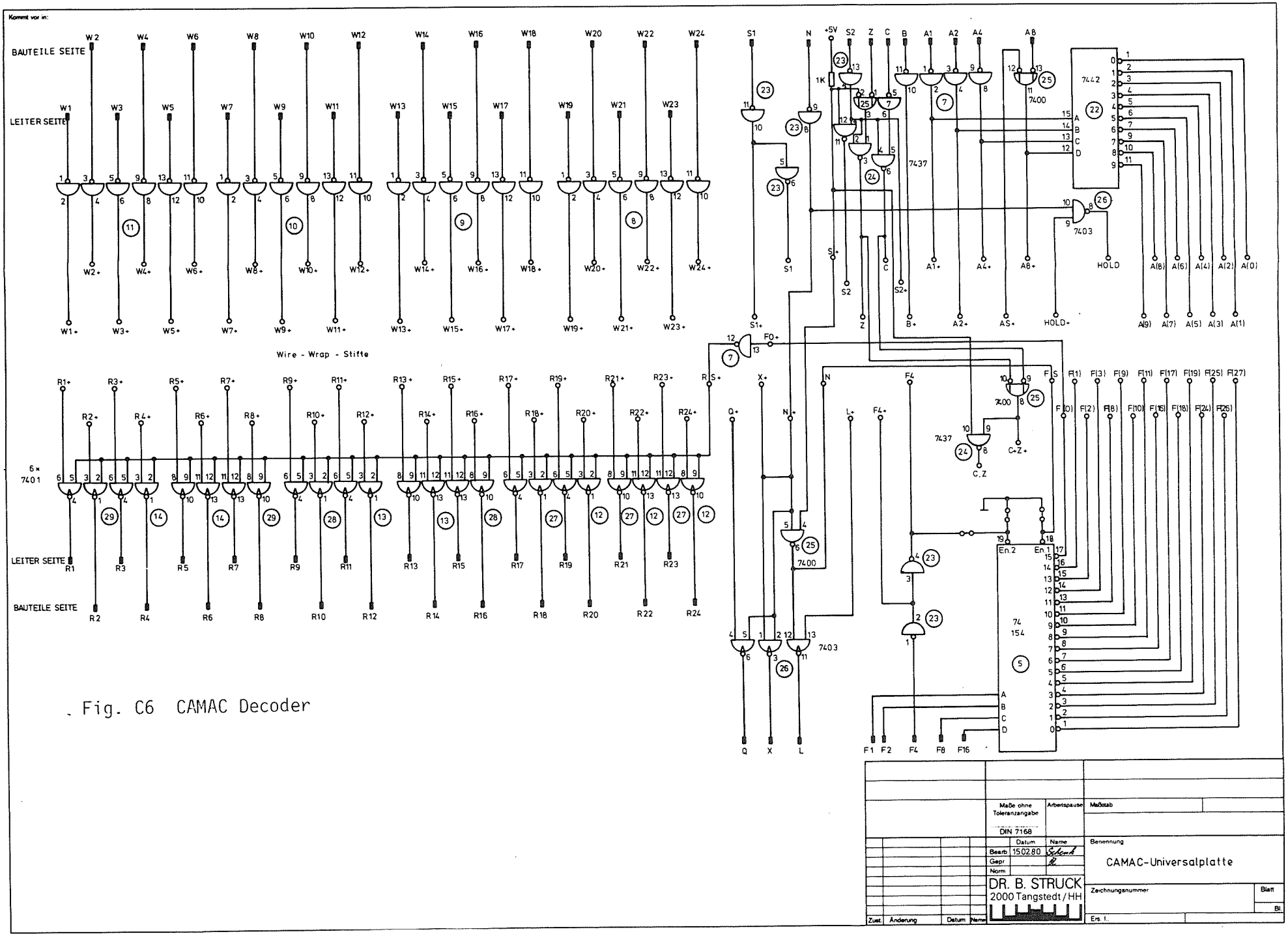


Fig. C6 CAMAC Decoder

		Maße ohne Toleranzangabe	Arbeitspause	Maßstab	
		DIN 7168			
			Datum	Name	Benennung
		Bearb	15.02.80	Schmidt	CAMAC-Universalplatte
		Gepr			
		Norm			
		DR. B. STRUCK 2000 Tangstedt / HH			Zeichnungsnummer / Blatt
Zust.	Änderung	Datum	Name		Ers. f. / Bl.

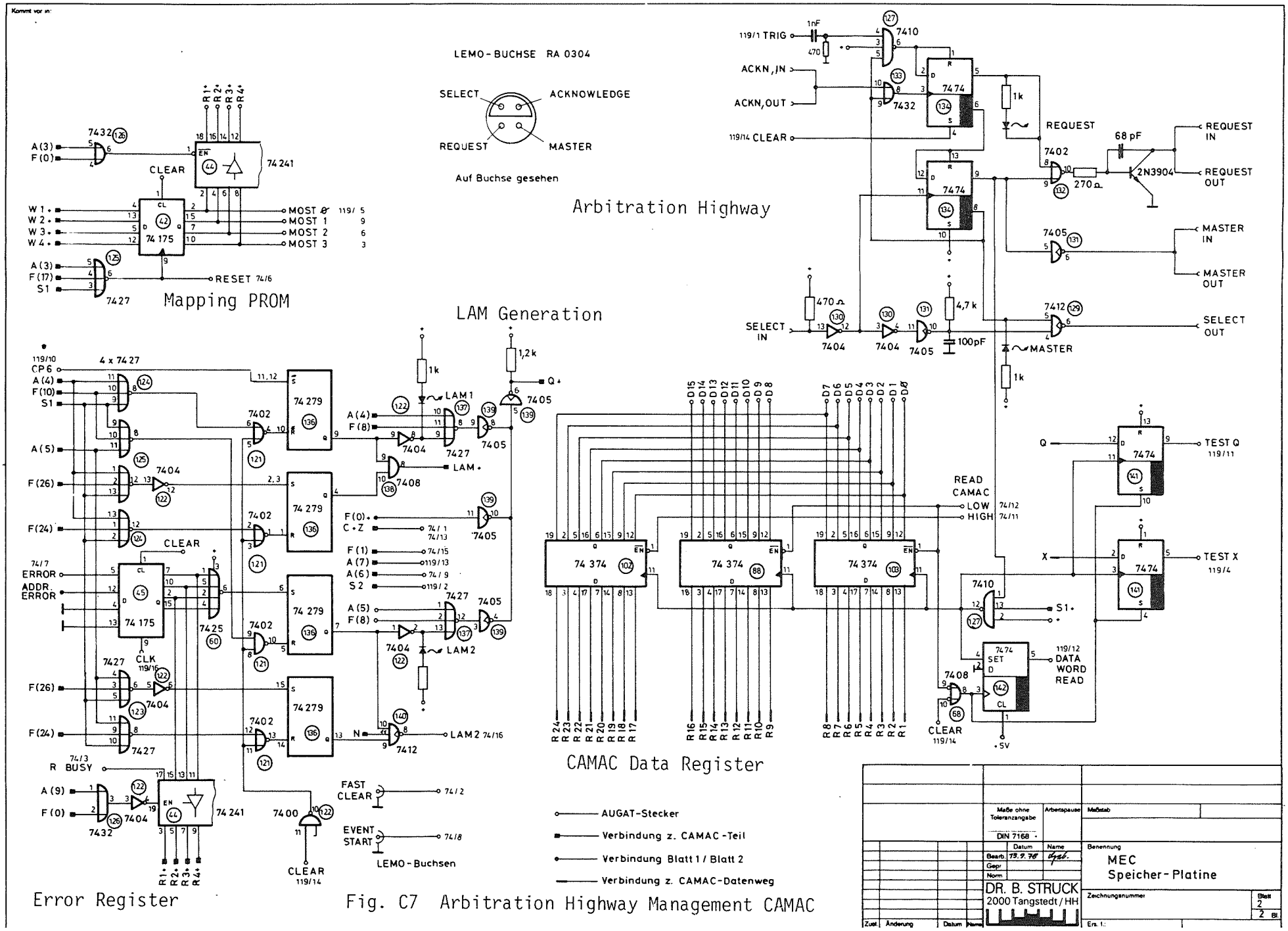


Fig. C7 Arbitration Highway Management CAMAC

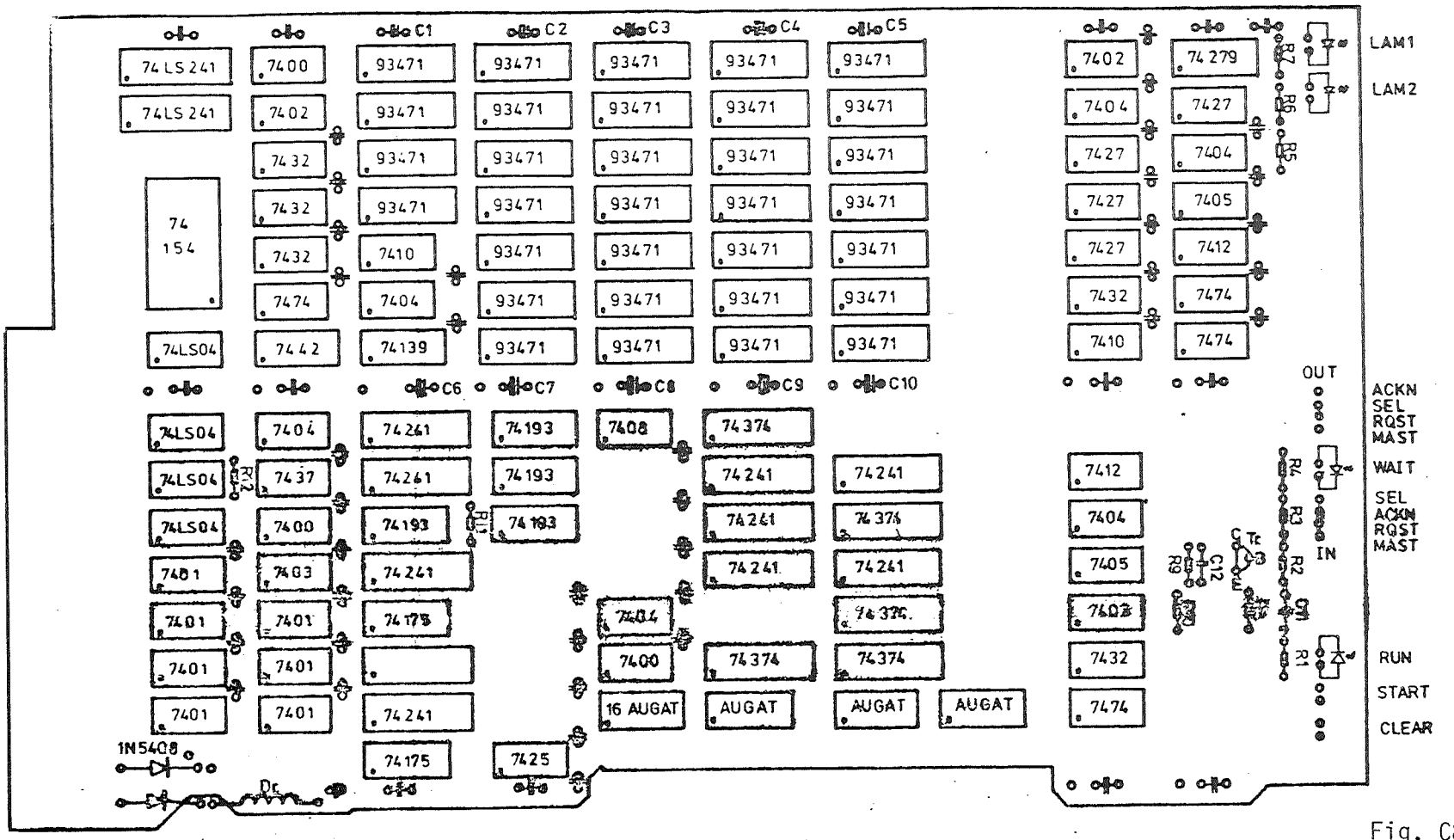


Fig. C8 Layout of MEC memory

10nF

10nF

C1 - C10 4,7uF/6,3V

C11 100pF

C12 82pF

R1	1K	R5 - R7	1K	R11	1K
R2	4,7K	R8	560	R12	1K
R3	470	R9	270		
R4	1K	R10	560		

Maßstab

Maße ohne Toleranzangabe

DIN 7168

Datum

Name

Bearb 14.02.80

Gepr

Norm

DR. B. STRUCK
2000 Tangstedt / HH

Benennung

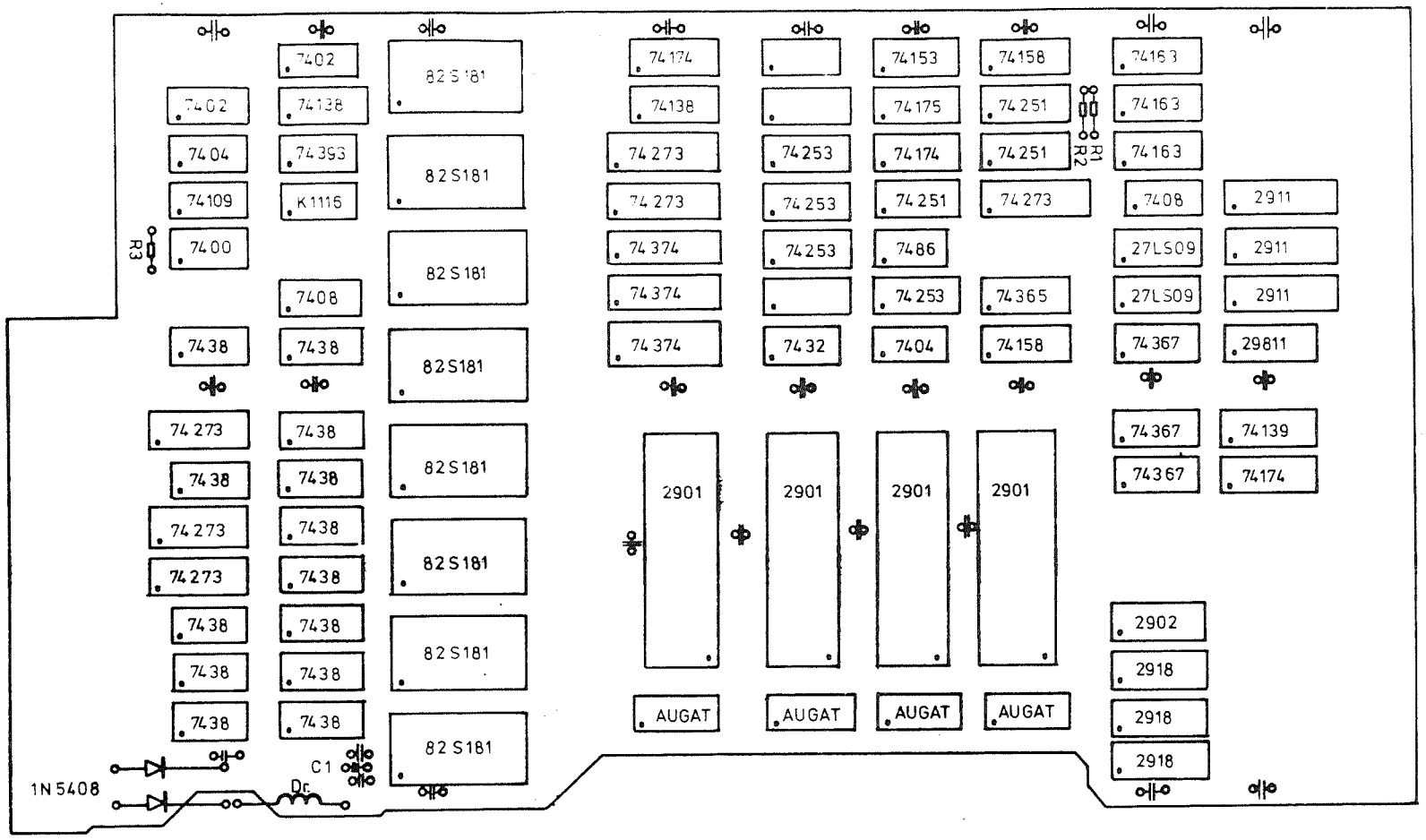
MEC- SPEICHER

Zeichnungsnummer

Blatt

Änderung

Datum



10nF

C 1 100µF/6,3V

R 1 R2 R3 1K2

Fig. C9 Layout of MEC processor

Maße ohne Toleranzangabe DIN 7168	Arbeitspause	Maßstab	
	Datum	Name	Benennung
Bearb	14.02.80	Schenk	MEC-PROZESSOR
Gepr			
Norm			
DR. B. STRUCK 2000 Tangsted		Zeichnungsnummer	Blatt

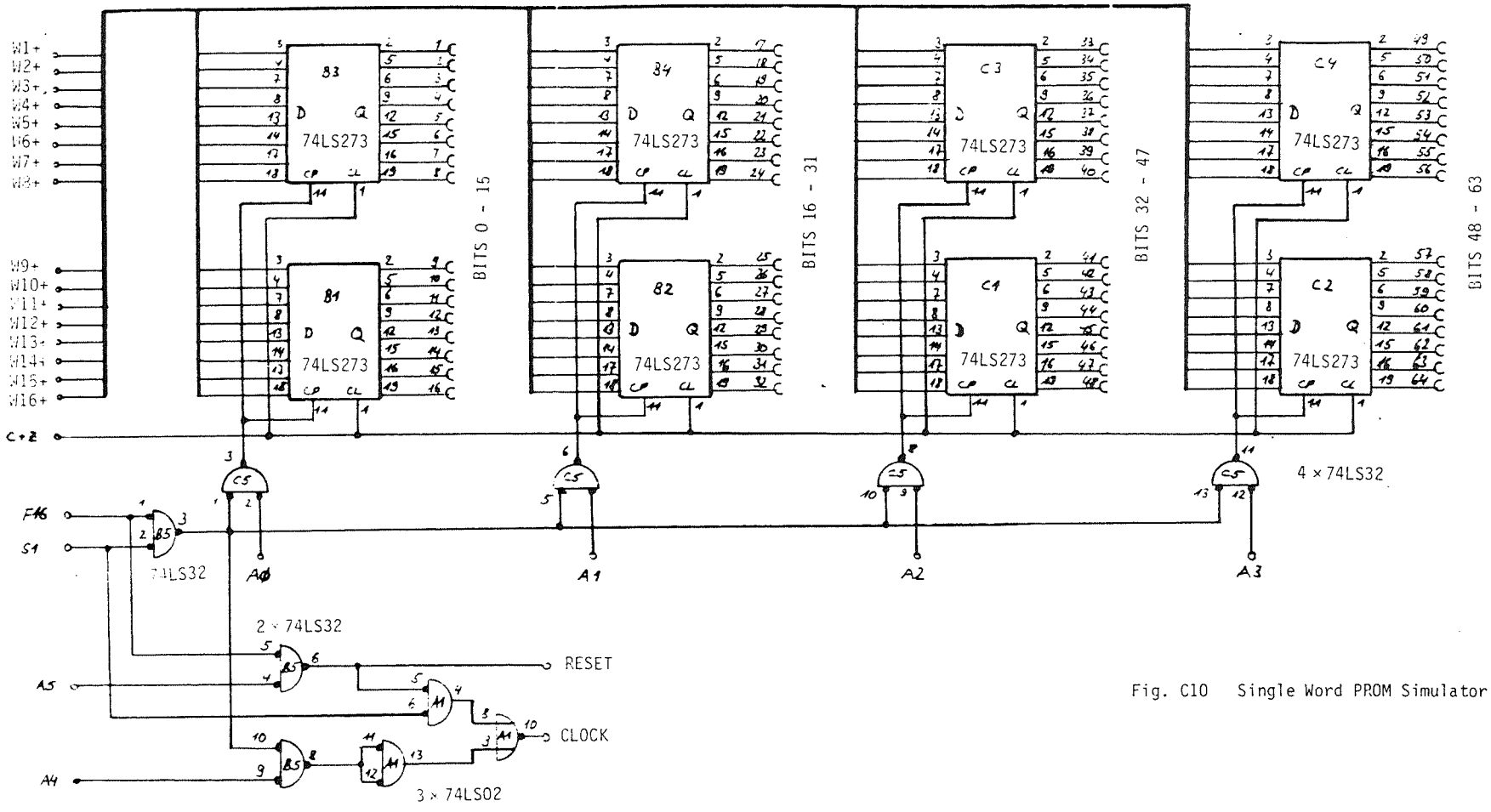


Fig. C10 Single Word PROM Simulator

CAMAC-COMMANDS

F(0)	A(0)	READ p-ADDRESS
F(16)	A(0)	WRITE BITS 0 - 15
F(16)	A(1)	" " 16 - 31
F(16)	A(2)	" " 32 - 47
F(16)	A(3)	" " 48 - 63
F(16)	A(4)	GENERATE CLOCK
F(16)	A(5)	" RESET

Maße ohne Toleranzangabe
DIN 7168
Maßstab
Datum
Name
Bearb.
Gepr.
Norm
Benennung
MEC Testmodul
DR. B STRUCK
2000 Tangstedt / HH
Zeichnungsnummer
Zust. Änderung Datum Name
Ers. f.

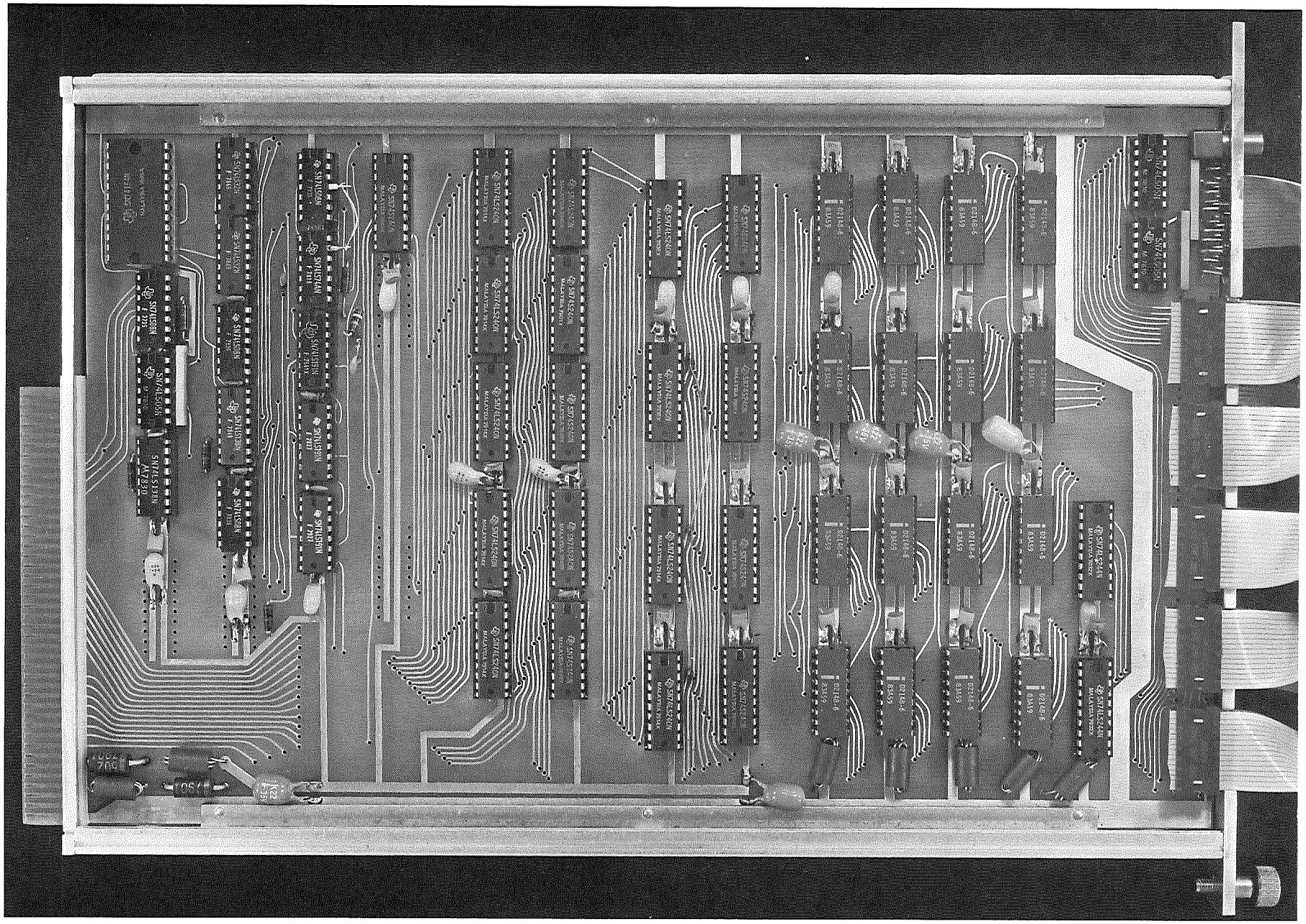


Fig. C11 Layout of the PROM simulator with full 1K 64 bit word memory. The cables on the rigth hand side are connected to the PROM sockets of the processor.

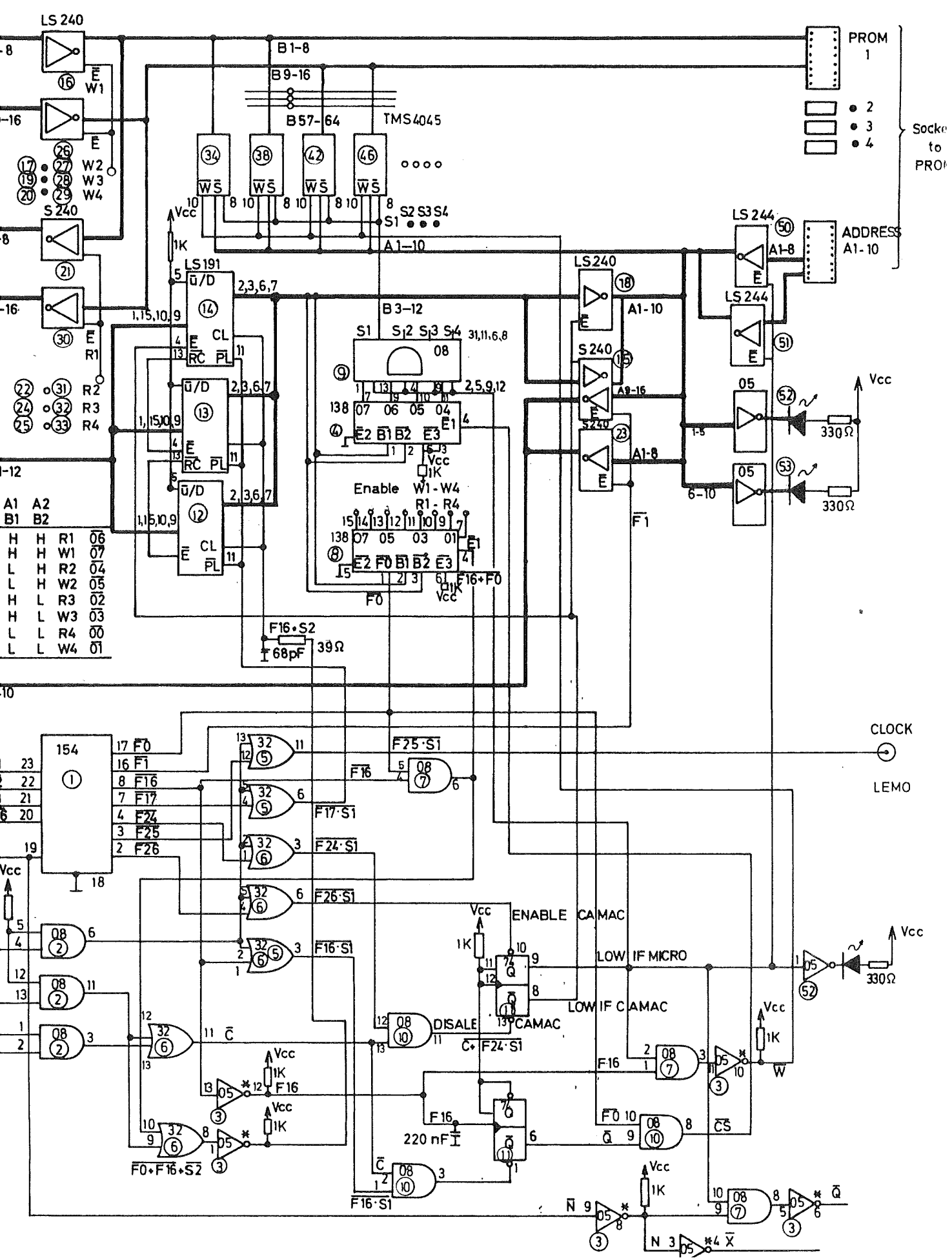


Fig. C12 PROM simulator with full memory. The PROM's of the processor are replaced via cables by this module. The address of the processor is indicated by LEDs.

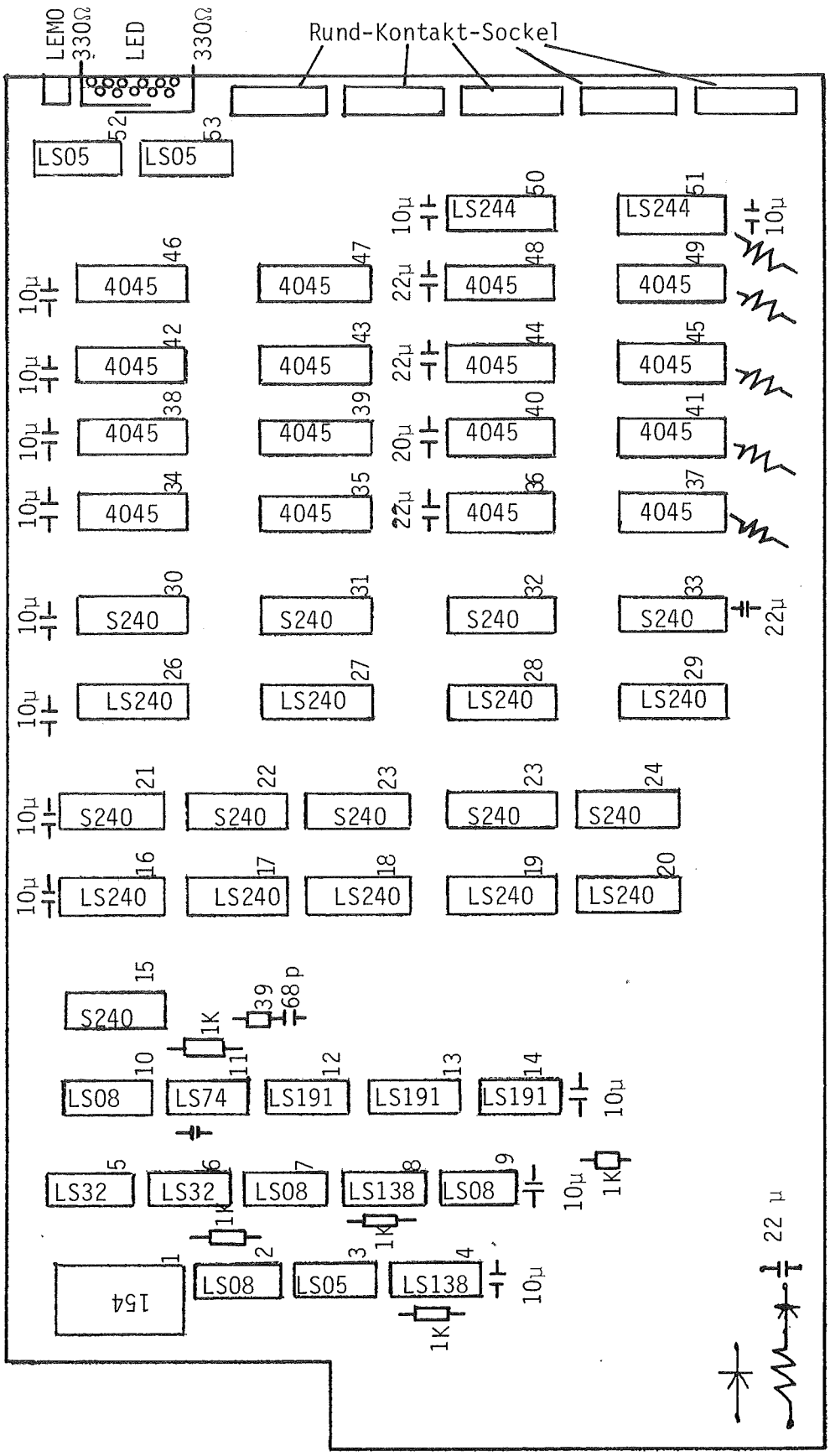


Fig. C13 ICs on microprogram memory 2790/1 (PROM simulator)