DELPHI-87-70-DAS-58

DELPHI-87-70-DAS-58
July 1st 1987

Fermilab Library
0 1160 0035404 7

FDDP
A Fast Digital Data Processor
for the FEMC trigger.
(VME version)

User Manual, rev. 1.0

FERMILAB
MAR 17 1994
LIBRARY

D. Crosetto [1)]

Abstract

In high energy physics experiments, where sophisticated trigger decisions are based on a large number of signals, fast low cost programmable devices have become a necessity. For decisions that have to be taken in a time longer than 10 usec, it is conceivable to use fast Digital Signal Processors (DSP) working in parallel, instead of hardware decision modules.

The FDDP is a modular system implemented on VME and FASTBUS boards, consisting of multiple "Logical Units" per board, which can process in parallel several digitized analog signals to perform a programmable trigger decision. Each "Logical Unit" has a Digital Signal Processor (DSP Texas TMS32010) which executes the majority of its intructions in 200 nsec. A VME board contains two "Logical Units", a FASTBUS board under development contains six "Logical Units". A system of 24 "Logical Units" will be used for the trigger in the Forward ElectroMagnetic Calorimeter (FEMC) of DELPHI detector, to calculate sector energy in 14.8 usec for the 2nd level trigger, and perform a cluster search in 150 – 250 usec for the 3rd level trigger. Other possible applications are discussed.

[1)] Istituto Nazionale Fisica Nucleare Sez. Torino, ITALY

CONTENTS

1. INTRODUCTION 2
   1.1 Software 3
      1.1.1 TMS32010 program debugging facilities. 4
      1.1.2 Application program monitoring facilities. 4
   1.2 Hardware 4

2. FEATURES 5

3. LOGICAL UNIT 6
   3.1 Digital Signal Processor (DSP). 7
      3.1.1 DSP limitations. 7
   3.2 Asynchronous communications 10
      3.2.1 Logical Units Intercommunication 10
      3.2.2 Logical Unit communication with Crate Controller 10
         3.2.2.1 Crate Controller communication with dual port Program memory 10
         3.2.2.2 Crate Controller communication with dual port Data memory 10
         3.2.2.3 Crate Controller with TMS32010 through MAIL-BOX registers. 11
         3.2.2.4 Crate Controller communication with TMS32010 through Out-FIFO's. 12
         3.2.2.5 Interrupts to the Crate Controller. 12
         3.2.2.6 Interrupts to the TMS32010. 14
         3.2.2.7 Software triggers. 14
         3.2.2.8 Software reset to the TMS32010. 14
      3.2.3 Logical Unit communication with External Input Devices 14
      3.2.4 Logical Unit communication with External Output Devices. 15
   3.3 Memory Map & I/O Functions 15
   3.4 Command Register (CMD). 17
   3.5 Status Register (STS). 18
      3.5.1 Status Register Flag clearance. 18
   3.6 FIFO board 20

4. USER SIGNAL DESCRIPTION 21

5. JUMPER SELECTION 25
   5.1 J1/J16 - MEMORY MAP-SELECT 25

6. FRONT PANEL CONNECTORS 26
   6.1 TRIGGER 26
   6.2 ABORT 26
   6.3 PL9A & PL9B "Output Ready (OR) of Out-FIFO's" signals 26
   6.4 PL0A & PL0B "Unclock of Out-FIFO's" signals 26
   6.5 P3A "16-bit Out-FIFO's" signals 27

7. BACK-PLANE CONNECTORS 29
   7.1 P1 "VME backplane connector" signals. 29
   7.2 P2 "VME backplane connector" signals 30

1. TRADEMARKS i

INDEX ii

TABLES

1. INTERRUPT REGISTER MODEL 12

2. INTERRUPT LEVEL 13

3. MEMORY MAP & I/O FUNCTIONS 16

4. COMMAND REGISTER 17

5. STATUS REGISTER 19

6. BASE ADDRESS JUMPER SELECTION 25

7. P3A "Front panel Out – FIFO's DATA – BUS". Pin assignment 27

8. P1 back plane connector VME – BUS. Pin assignment 29

9. P2 back plane connector VME – BUS. Pin assignment 30

10. P6A & P7A CONTROL LINES for ANIN board 32

11. P6D & P7D ADDRESS LINES for ANIN board 32

12. P6B & P7B TMS32010 DATA BUS for ANIN board 33

13. P6F & P7F POWER SUPPLY for ANIN board 33

14. P6C & P7C / P6E & P7E CONTROL LINES for FIFO board 35

15. P8A & P8C DATA – BUS of TMS32010A & TMS32010B for FIFO board 35

16. P8B DATA – BUS from transreceivers to VME for FIFO board 36

17. P8D Out – FIFO's DATA – BUS for FIFO board 36

18. P8E TMS32010A & TMS32010B DATA – BUS to CW & CCW FIFO's 37

19. Screen page of FEMC monitoring program. 47

20. TMS32010 assembly routine for analysis of a track in a drift chamber 52

21. PL1A/PL8A & PL1B/PL8B Lemo connector ANALOG INPUTS (ANIN1 board). 61

22. P6H & P7H 20 pin connector ANALOG INPUTS (ANIN1 board) 62

8. ON BOARD CONNECTORS 32
8.1 FDDP connectors to piggy – back ANIN boards 32
8.2 FDDP connectors to piggy – back FIFO board 35

9. APPLICATION EXAMPLES 38
9.1 FDDP – A modular system unit 38
9.2 Examples of different personalized ANIN boards. 38
9.3 FDDP System Configuration for Development and Applications 41
9.4 Application example to calorimeters (FEMC – DELPHI) 42
9.4.1 Development and test. 47
9.4.2 FEMC – DELPHI trigger monitoring facilities. 47
9.5 Application example to straw tubes. 48
9.6 Application examples to drift chambers 50
9.6.1 Example of hit center of gravity and total energy calculation 50
9.6.2 Guidelines to configure a FDDP system for drift chamber analysis. 55

Acknowledgments 60

Appendix A: ANIN1 BOARD 61

A.1 Analog Input board for FEMC – DELPHI trigger. 61
A.1.1 PL1A/PL8A & PL1B/PL8B – ANALOG INPUTS (ANIN1 board) 61
A.1.2 P6H & P7H – ANALOG INPUTS (ANIN1 board) 62

Appendix B: TMS32010 Assembly routines 64

B.1 TMS32010 software support routine (tracing program). 64
B.2 TMS32010 software support routine (monitoring program). 68

Appendix C: MC 68000 Assembly routines 71

C.1 MC68000 interrupt routine (for TMS32010 tracing and monitoring). 71
C.2 MC68000 display and command interpreting routines for TMS3210 tracing. 74
C.3 MC68000 display and command interpreting routines for TMS3210 monitorin 75
C.4 MC68000 test & application example routines. 76

Appendix D: TMS32010 DSP: documentation 94

D.1 TMS32010 Instruction Set Summary 95
D.2 TMS32010 Cross – Assembler Directives 96
D.3 TMS32010 Cross Software Support 98

Appendix E: MC68000 documentation 100

E.1 MC68000 Instruction Set Summary 100
E.2 MC68000 Cross Software Support 102

References 104

23. Screen page of TMS32010 trace program 64

24. TMS32010 Interrupt trace routine. 65

25. TMS32010 Interrupt monitoring routine. 69

26. MC68000 interrupt routine for TMS32010 tracing and monitoring. 71

27. MC68000 display routine for TMS32010 tracing. 74

28. MC68000 display routine for TMS32010 monitoring. 75

29. TMS32010 INSTRUCTION SET SUMMARY 95

30. TMS32010 CROSS – ASSEMBLER DIRECTIVES 96

31. TMS32010 CROSS SOFTWARE SUPPORT 98

32. MC68000 INSTRUCTION SET SUMMARY 100

33. MC68000 CROSS SOFTWARE SUPPORT 103

FIGURES

1. LOGICAL UNIT block diagram 8

2. LOGICAL UNIT intercommunication through FIFO's block diagram 9

3. The MAIL – BOX registers block diagram 11

4. FIFO onboard connectors layout 20

5. FDDP jumpers layout 24

6. FDDP front panel and back plane connectors layout 28

7. FDDP onboard connectors layout for ANIN and FIFO boards 34

8. FDDP as a modular system for Fast Digital Signal Processing 39

9. Examples of different personalized ANIN boards 40

10. FDDP System Configuration for Development and Applications 41

11. The FEMC 2nd and 3rd level trigger with 24xTMS32010. 43

12. Detail of FEMC system trigger. 44

13. FEMC – DELPHI calorimeter segmentation 45

14. Typical timing diagram of FEMC second level trigger 46

15. Typical timing diagram of FEMC third level trigger. 46

16. Example of ANIN board that interfaces data from FERA (Le Croy) unit 48

17. Example of FDDP processing data from straw – tubes 49

18. Algorithm flowchart of hit center of gravity and total Energy 51

19. Analysis of a track in a drift chamber. 54

20. Track resolution using derivative on signals 56

21. FDDP mother board 57

22. ANIN1 and FIFO piggy back boards 58

23. FDDP board assembled with ANIN and FIFO boards. 59

24. ANIN1 onboard connectors layout 62

25. ANIN1 board. Block diagram 63

26. Logical Unit Program Memory Map 67

27. TMS32010 Functional block diagram 94

1. TRADEMARKS

TMS32010 is a trade trademark of TEXAS INSTRUMENTS; MicroVAX, VAX, VMS, are trademarks of Digital Equipment Corp.; IBM – PC, XT, AT are trademarks of IBM; MS – DOS, is a trademark of Microsoft; PAL, PALASM are trademarks of Monolithic Memories Inc. MC68000 is a trademark of Motorola Inc.; Unix is a trademark of Bell Laboratories. I apologise in advance for any failure to acknowledge other trademarks.

1. INTRODUCTION

The Fast Digital Data Processor is a slave module which performs fast parallel digital processing. It has been developped for the preliminary data reduction of the trigger signals from the Forward Electromagnetic Calorimeter (FEMC) of Delphi [1].

Independent of the FEMC trigger, the FDDP card can be regarded as a rather general purpose board, which can perform the following tasks, working in a slave parallel mode:

- fetch digitalized signals from a separate analog – to – digital conversion board (ANIN1)[1]
- process the digital data by means of a fast microprocessor;
- store the results of the processing into FIFO memories, readable either through a front panel connector or through the back plane bus;
- exchange data with adjacent processors through FIFO memories.

In the present configuration the FDDP board contains two logical units. Each unit (see Fig. 1) consists of:

- one ANIN board;
- 1 microprocessor (Texas TMS 32010) with relative program memory, registers and ancillary logic;
- a set of output FIFO memories (OUT FIFO) to store the results;
- two sets of FIFO's to exchange data with clockwise and counterclockwise adjacent processors (CW and CC FIFO).

The completely assembled FDDP board consists of a 'mother board', hosting the two TMS32010 processors, program memories, buffers, registers, ancillary logic, etc., and of three piggy back cards. On two piggy back cards – called 'ANIN1' – are located the analog circuits (operational amplifiers, Flash ADC), for the acquisition. On the third piggy back card are located the OUT FIFO's and the two sets of CW and CC FIFO's.

In the present version, the mother board, housing two "Logical Units", is a standard VME card, to be inserted in a VME crate, while the final board, under development, with six "Logical Unit" will be in FASTBUS (see Fig. 8).

A crate microprocessor (Motorola MC68000), working as master, is needed to load programs directly into the TMS32010 program memory, to monitor its activity and eventually to read the results.

A trigger pulse starts acquisition, conversion and processing, while an abort pulse can stop processing at any time.

Debugging, maintenance of the board and simplicity in writing the software profit from a modular approach to the architecture of the CPU, peripherals and control lines. All the functions that can be generated by the TMS32010 processor can also be generated by the crate master microcomputer through the VME bus. These features are very convenient during hardware debugging.

The basic description of the Logical Unit is contained in section 3. The core of the unit is represented by the different means of communication between the various subsystems. They are discussed in section 3.2 in a way which is not very orthodox but it is conducive to appreciate the overall architecture. There is some redundancy between sections in order to make this document useful for reference.

---

[1] for more details refer to the chapter: application examples and ANIN description in Appendix A.

1.1 Software

Software development for the TMS32010 (see Appendix B and D) and MC68000 (see Appendix C and E) requires a host computer[2] for file storage and cross software preparation. The software, in object code (hexadecimal output file of cross linker in Motorola S format) must then be loaded into the "Crate Master Controller" microprocessor memory (e.g. via a RS232 connection). A transfer program from the "Crate Master Controller" microprocessor will then copy the code into the TMS32010 program memory[3] (see also Fig. 10, Fig. 26, Tab. 19 and Tab. 23).

Seen from the TMS32010 side, the program memory is located between addresses 0 and $7FF (see Table 3). A reset command stops the TMS32010 and allows the master to access the TMS32010 program memory. For the TMS32010, the memory locations from 0 to 7 are reserved as input/output ports.

From the MC68000 side, each FDDP logical unit is seen as a memory area starting at a given base address, greater than $100000($ means hexadecimal), and extending to a displacement of $201E.[4] The base address is selected by setting of jumpers on the FDDP board.[5] Of the two logical units which reside on the same board, one corresponds to the selected base address, the second to base address + $4000.

To develop or modify the software, important information is found in:

- "Logical unit" block diagram: Fig. 1
- Memory & I/O Functions List (Table 3)
- Command Register (Table 4)
- Status Register (Table 5)
- Trigger signal (section 6.1)
- Abort signal (section 6.2)
- Interrupt registers (section 3.2.2.5)

The overall software development is the following:

- a set of drivers (see Appendix C), component test and function test routines are being written in MC68000 Assembly language. The cross – assembler used for these routines is: Lattice [17], Quelo [18] and MC68mil [4]. The driver routines mentioned above are then called from an high level language for writing more complex programs.
- For writing these higher level programs, we chose Real Time Fortran (RTF) for the applications since it is well known and widely used by physicists, and "C" for the basic software, since "C" has good Input/Output and is found on practically all personal computers and all operating systems. Both high level languages are calling the same assembly routines mentioned above (see Appendix C).

---

[2] Cross Software for MC68000 and TMS32010 is available under MS – DOS, VMS, UNIX Operating Systems. The "Host Computer" can then be: IBM PC or uVAX or VAX, etc.

[3] see "Block move routine from 68000 to TMS32010" listed in Appendix C

[4] see more details in Section "Memory map & I/O Functions"

[5] for more details refer to Section "Jumper Selection"

- Efforts are being made to write also different console driver routines for different VME CPU's. At present they are available for the CPU – 1B from Force Computer.
- The stack layout follows the model described in the M68MIL Cross Macro Assembler by H. Von Eicken [4].

1.1.1 TMS32010 program debugging facilities.

A trace facility allowing execution of one or N instruction per step, or continuous running is provided for TMS32010 programs. After execution of these commands, all TMS32010 registers are displayed with their actual value (see Table 23).

This facility is available by loading into the VME Crate Controller memory the programs described in Appendix C1 and C2, and into the TMS32010 program memory the programs described in Appendix B1.

1.1.2 Application program monitoring facilities.

In order to make visible the dynamic changes of the state of the Logical Units to the user during application program execution, the hardware design provides each Logical Unit with MAIL – BOXES.

An example of monitoring the different phases and status of each Logical Unit during calculation of II and III level trigger is described in section 9.4.2 (see Table 19).

This facility is available by loading into the VME Crate Controller memory the programs described in Appendix C3, and into the TMS32010 program memory the programs described in Appendix B2.

1.2 Hardware

The mother board circuit is a standard VME card (Fig. 21), implemented on a 6 layer printed circuit board. The block diagram of one Logical Unit is shown in Fig. 1 The ANIN1[6] (Fig. 25 and Fig. 22) piggy back cards, which receive the analog inputs, are connected to the mother board through connectors: P6A, P6B, P6D, P6F, for ANIN of TMS32010A; and P7A, P7B, P7D, P7F, for ANIN TMS32010B (see Fig. 7). All the analog circuits are optically insulated from the digital circuits and from the mother board. The FIFO piggy back card (Fig. 22) is connected to the mother board through the connectors: P8A, P8B, P8C, P8D, P8E (see Fig 7).

---

[6] for more details see "ANIN description" in Appendix A.

2. FEATURES

1. Flexible triggering mode: external, internal.

2. Easy program modification, effected by giving to the Crate Controller access to the TMS32010 program memory.

3. Fast digital processing of 8 analog channels by each of two TMS32010 microprocessors (for example 16 x 16 – bit multiply in 200 nsec).

4. Data intercommunication between the two TMS32010 microprocessors and the adjacent modules, through FIFO's at 2.5 Mbyte/sec.

5. Output data available asychronously from 16x16 – bit Out – FIFO's, through the VME bus at the Crate Controller CPU maximum speed, or through the front panel connector at 20 Mbyte/sec.

6. The Logical Units can be paralleled without limits.

7. Possibility to design a personalized ANalog INput board (ANIN) to suit better the application needs.

8. Dual port memory (switching technique) for onboard data storage.

9. Messages and data communication between TMS32010 microprocessors and Crate Controller on the VME through the MAIL – BOX registers, without need to stop the TMS32010 activity.

10. Software maskable TMS32010 interrupt sources.

11. VME bus interrupt capability:

    a. by software selectable interrupt vector,

    b. by software selectable interrupt levels (from IRQ1 to IRQ7),

    c. by software maskable interrupt sources:

    - from Out – FIFO's "OR" (output ready signal),
    - from TMS32010 microprocessors, (at Mail – Box write operation).

12. External "ABORT" line for the TMS32010 microprocessors.

13. "BROADCAST" capability of all VME bus write functions to all modules in the crate.

14. From the VME Crate Controller side, each board needs 2x8Kwords of 16 – bit (32Kbytes) of contiguous space selectable through jumpers in the range of 0 to 4 Gbytes. Thus there are essentially 4K words of program space and 4K words of I/O space for each DSP.

3. LOGICAL UNIT

The Logical Unit (see Fig. 1) is the basic processing element of a system configuration. Based on this modularity, regardless of the board used (VME version with two Logical Units or FASTBUS version with six Logical Units), a very powerful system of several hundred MIPS can be built. Each Logical Unit has an independent processing capability of almost 5 MIPS (DSP Texas TMS32010), since the majority of the instructions require only one 200 nsec cycle.

The principal features of the Logical Unit are:

- it can fetch data via its own Input/Output or from an adjacent Logical Unit.
- it can communicate asynchronously to the adjacent units, to the Crate Controller or to an external device through a front panel data bus.
- a Crate Controller processor (see Fig. 10 and Fig. 11) is required to download the TMS32010 programs to correlate the Logical Units activity and also supervise the entire system (any VME−CPU or FASTBUS master board can accomplish this task). Beside this, the Crate Controller can do also other important tasks such as: debugging the hardware and simulating the algorithms at lower speed (using the MC68000 instruction set instead of the TMS32010 instruction set). These features have been made available by paying particular attention in designing the hardware so that from the software point of view, all functions that can be generated by the TMS32010, can also be generated by the Crate Controller processor (see Tab. 3). Considering that normally there is more software development support for the MC68000 family than for DSP's, this hardware design consideration has been a valuable help during debugging and testing.
- in order to give more flexibility to the system, the conversion circuit module from Analog to Digital called ANIN (ANalog INput module) has been built in a separate board giving thus the possibility to design different personalized analog modules to fulfill better the application needs.

A typical sequence of operation in the Logical Unit is the following:

- the Crate Controller processor takes mastership of the Logical Unit Data Bus by resetting the TMS32010,
- downloads the TMS32010 program in the Program Memory,
- releases the Logical Unit Bus
- monitors asynchronously the activity of the Logical Unit by reading the MAIL−BOX in polling mode or in interrupt mode (normally every time there is a change in the status of the Logical Unit, the TMS32010 copies the STATUS REGISTER of its Logical Unit into the MAIL−BOX)
- the Crate Controller processor may also collect asynchronously DSP results at its maximum speed, without stopping the activity of the TMS32010, through the Out−FIFO's.

When the Crate processor has released the Logical Unit bus, the TMS32010 starts execution of the instruction in position 0 of the Program Memory. A fast means to synchronize an external event (e.g. Trigger) with the TMS32010 program is to use the 'BIOZ' instruction that tests the 'BIO' signal (connected to the Trigger in the FEMC – DELPHI application). This type of interrupt is preferable when performing time critical loops because it has a Maximum Latency Time (MLT) of 1 usec. Normally then the program fetches input data from the ANIN module or from the CW and CCW fifo's, performs the calculation of the algorithm and writes the result to the Out – FIFO's. At the same time, asynchronously, it can communicate data to the adjacent Logical Units, to the Crate Controller processor, update the Status Register and enable/disable signals through the Command Register.

3.1 Digital Signal Processor (DSP).

Digital signal processing is concerned with real – time processing of digitized analog signals, which are discrete in both amplitude and time. The real – time requirement has forced the DSP to be very fast, normally by having data and instructions in separate address spaces. As well their wide use has resulted in a low unit cost. Thus the combined characteristics of high speed and low cost make these devices very attractive for trigger processing in high energy physics experiments [8].

The primary advantage of digital signal processing is that it uses discrete algorithms instead of analog function and thus extends the advantage of digital circuitry (high density, precision, stability, and testability) to those parts of systems previously served by analog components.

Finally, digital signal processing can solve application problems in a more fexible way then analog signal processing.

3.1.1 DSP limitations.

Major limitations are due to rates.

Some DSP applications involve sampling rates of up to 100 MHz and can require hundreds of million of multiply/accumulate operations per second. In the case of the FDDP board this limitation has been solved by employing several Logical Units. The parallelism thus obtained can yield systems of several hundreds of Million of Instructions Per Second (MIPS).

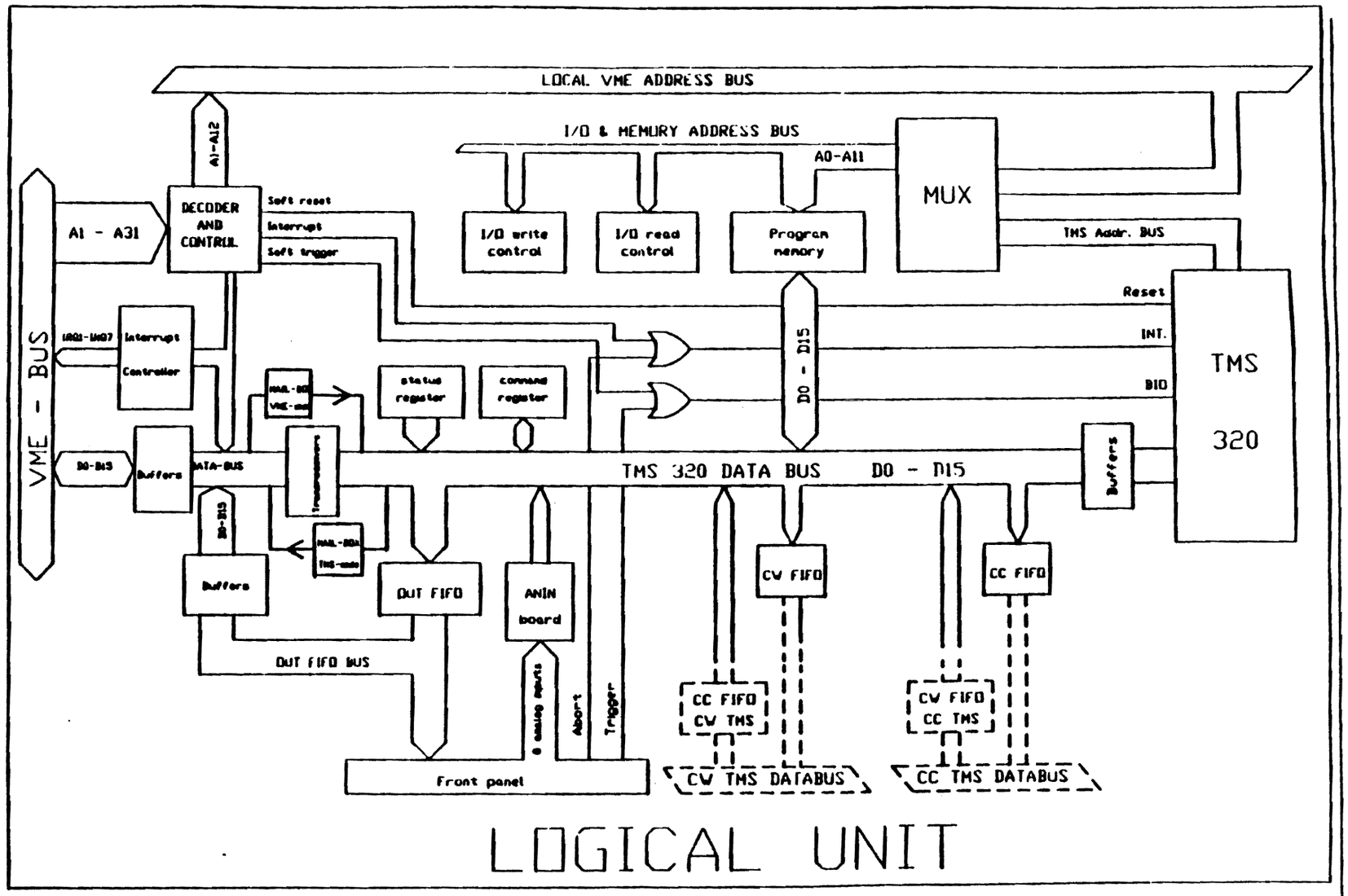


Figure 1: LOGICAL UNIT block diagram

Notation in ( ) refers to STATUS REGISTER of Table 5.
Notation [ ] refers to Memory Map & I/O Function of Table 3.

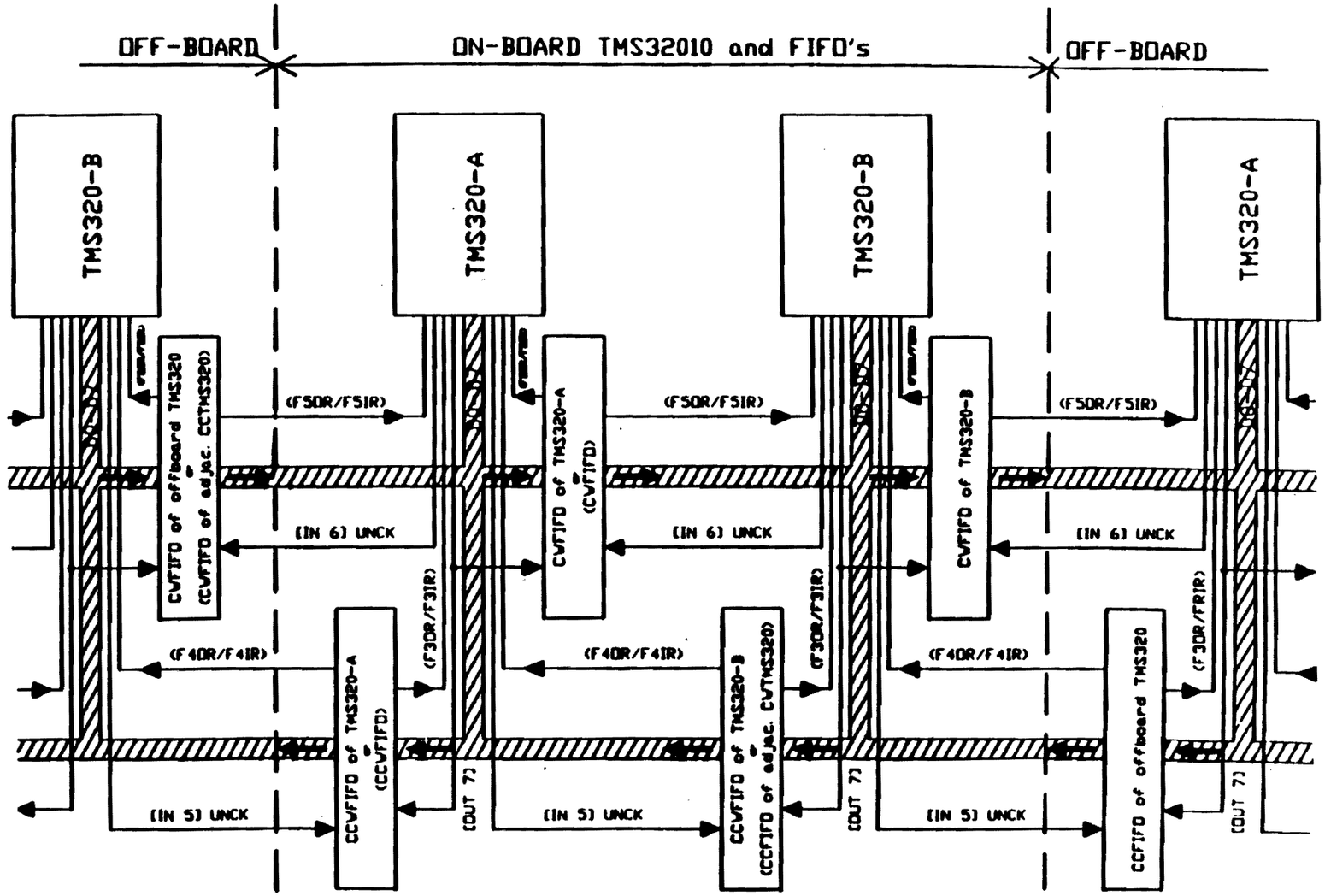


Figure 2: LOGICAL UNIT intercommunication through FIFO's block diagram

3.2 Asynchronous communications

All communications are asynchronous.

Some communications from the Crate Controller Processor to the TMS32010 Program Memory require the TMS32010 to be in the reset state (see below). All other communications from the Crate Controller side or between Logical Units do not require stopping the activity of other Units. If interrupts on a given Logical Unit are enabled, then that Logical Unit may switch from one task to another without releasing the bus lines.

3.2.1 Logical Units Intercommunication

Each Logical Unit can exchange 8 – bit data with the adjacent Logical Units (two Units: Clock Wise and Counter Clock Wise) at the maximum speed of 2.5 Mbyte/sec. The intercommunication scheme is shown in Fig. 2 Elements carrying an "A" label are specific of the first of the two "Logical Units" located on the board, elements carrying a "B" label are specific of the second onboard Logical Unit.

The communication is asynchronous as there is no need to stop the activity of the other Logical Units during transfer. Each Logical Unit is informed if there are data from other Logical Units by reading the STATUS REGISTER (see Tab. 5). (For diagnostic purpose it is also possible to write and read the CW and CC FIFO's from the Crate Controller after having stopped the TMS32010 activity).

Read and Write Commands, as well as FIFO reset, from the TMS32010 or from the Crate Controller are described in Tab. 3 (Memory Map & I/O Functions).

3.2.2 Logical Unit communication with Crate Controller

Logical Units and Crate Controller can communicate at different levels.

Some communications from the Crate Controller require the TMS32010 to be in the reset mode (no activity) because they require to use data or adress busses of the Logical Unit. They are listed in table 3 with a "*" indicator and have offset address smaller than $2010.

The other communications can be performed also while the TMS32010 is active: they are listed in table 3 and have an offset address greater than or equal to $2010.

A broadcast to all Logical Units at the same time is also forseen.

The possible communication channels are listed below.

3.2.2.1 Crate Controller communication with dual port Program memory

It requires stopping the TMS32010 activity.
The Crate Controller takes mastership of the Logical Unit bus just by writing 'D0 = 1' at base address + $2010. It can then read and write the Program Memory (see Tab. 3 for address). It can release the Logical Unit bus by writing 'D0 = 0' at the same address.

3.2.2.2 Crate Controller communication with dual port Data memory

It requires stopping the TMS32010 activity.

If the ANIN board is equipped with Data Memory, the Crate Controller takes mastership of the Logical Unit bus as for the Program Memory. (See Tab. 3). In the case of FEMC – DELPHI Calorimeter Trigger application, this memory is not provided on the ANIN1 board.
The Data memory can range from 2kiloword to several kilowords by using the bank switching technique with the limitation that for subsequent access to memory locations belonging to different banks, an extra time (400 nsec for the DSP TMS32010) is necessary. There will be no difference for each memory access (400 nsec for TMS32010) within the same 2kword memory bank.

3.2.2.3 Crate Controller with TMS32010 through MAIL – BOX registers.

The TMS32010 or the Crate Processor can write to or read from a 16 – bit MAIL – BOX register (see Fig. 3 and Tab. 3). The operation is asynchronous and does not require stopping the activity on the other side.

Writing the MAIL – BOX from the TMS32010 side automatically generates an interrupt to the Crate Controller as explained in section 3.2.2.5.

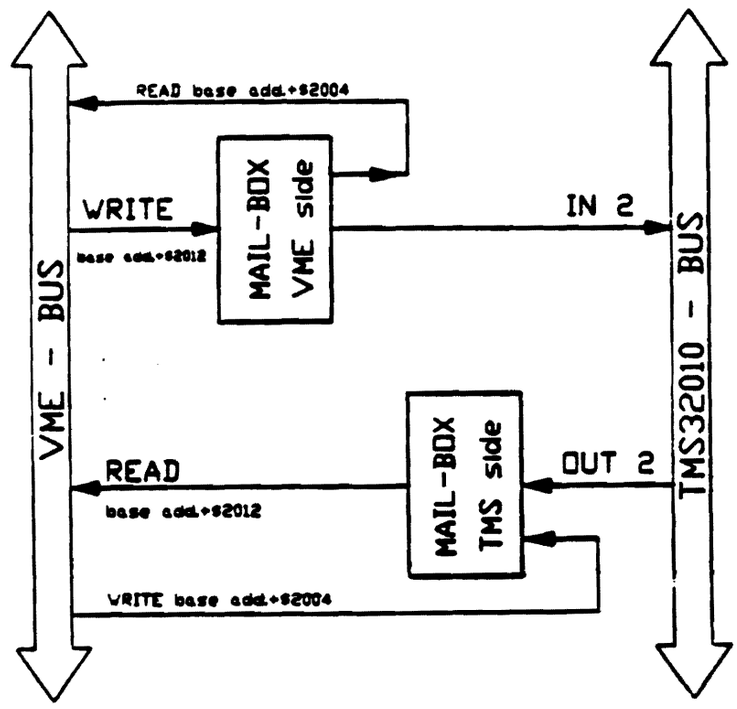


Figure 3: The MAIL – BOX registers block diagram

3.2.2.4 Crate Controller communication with TMS32010 through Out – FIFO's.

This unidirectional communication from the Logical Unit to the Crate Controller at the maximum speed of the Crate Controller CPU does not require stopping the TMS32010 activity.

Writing the Out – FIFO's from the TMS32010 automatically generates an interrupt to the Crate Controller as described in next section.
(For diagnostic purpose it is also possible to write the Out – FIFO's from Crate Controller, at offset $200C, after having stopped the TMS32010 activity).

3.2.2.5 Interrupts to the Crate Controller.

The interrupt is automatically generated by a write operation to the Out – FIFO's or to the MAIL – BOX on the TMS32010 side.

Within six VME cycles after receiving the acknowledge from the VME bus, the interrupt controller located on the Logical Unit will recognize the Interrupt Level and sends to the VME bus the Interrupt Vector for the service routine.

For each TMS32010 logical unit there are 4 programmable read/write registers. There are two interrupt Control Registers (CTRL REG0 – CTRL REG1) and two Vector Register (VECT REG0 – VECT REG1) that contain the vector data used during an interrupt acknowledge cycle. Interrupt control is based upon the MC68153 interrupt controller (see Table 3 for Interrupt Register Address).
Table 1 illustrates the interrupt register model.

Table 1: INTERRUPT REGISTER MODEL

REGISTER BIT	CTRL REG0 Out – FIFO	CTRL REG1 Soft.I.	VECT REG0 Out – FIFO	VECT REG1 Soft.I.
	base add. + 2018 Hex	base add. + 201A Hex	base add. + 201C Hex	base add. 201E Hex
bit – 7	F	F	V7	V7
bit – 6	DFAC	FAC	V6	V6
bit – 5	X/IN	X/IN	V5	V5
bit – 4	IRE	IRE	V4	V4
bit – 3	IRAC	IRAC	V3	V3
bit – 2	L2	L2	V2	V2
bit – 1	L1	L1	V1	V1
bit – 0	L0	L0	V0	V0

F = Flag Is a flag forseen to be used in conjunction with the test and set instruction of the MC68000. It can be changed without affecting other interrupt control operations. It is useful for processor-to-processor comunication and resource allocation.

FAC = Flag Auto-Clear If FAC is set, the Flag bit is automatically cleared during an interrupt acknowledge cycle.

X/IN = External(1)/Internal(0) Determines the response of the MC68153 during an interrupt acknowledge cycle. If the X/IN bit is cleared (low level) the MC68153 will respond with vector data and /DTACK signal, i.e., an internal response. If X/IN bit is set, the vector is not supplied and no /DTACK is given by the MC68153, i.e., an external device should respond.

IRE = Interrupt Enable This field must be set (high level) to enable the bus interrupt request associated with the control register. Thus, if the interrupt request from Out-FIFO's or from Mail-Box is asserted and the IRE is cleared, no interrupt request (/IRQ) will be asserted.

IRAC = Interrupt Auto-Clear If the IRAC is set (Bit 3), IRE (Bit 4) is cleared during an interrupt acknowledge cycle responding to this request. This action of clearing IRE disables the interrupt request. To renable the interrupt associated with this register, IRE must be set again by writing the control register.

The least significant 3 bit field of the register determines the level at which an interrupt will be generated:

Table 2: INTERRUPT LEVEL

L2	L1	L0	IRQ LEVEL
0	0	0	Disabled
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

Each interrupt input has its own associated vector register. Each register is 8-bit wide and supplies a data byte during its interrupt acknowledge cycle if the associated External/Internal (X/IN) con-

trol register bit is clear. This data can be status, identification, or address information depending on system usage. The information is programmed by the user.

When a software reset is generated by the Crate Controller, the Control Registers are set to all zeros (low). The vector registers are set to $0F. This value is the MC68000 vector for the uninitialized interrupt vector.

3.2.2.6 Interrupts to the TMS32010.

Two types of interrupts are implemented in the TMS32010: a conventional single level interrupt and a 'BIO' input signal.

The conventional interrupt must be enabled by setting Command Register bit 4. Three different interrupts are forseen, which are recognized by reading the Status Register.
– A software interrupt from Crate Controller can be generated by writing D1 = 1 at Board – Base – Address + $2010. This interrupt is maskable by bit 4 of COMMAND REGISTER.
– A software interrupt from TMS32010 (used in the trace routine) can be generated with an "IN PA7" instruction when bit 4 = 1 of Command Register (see Table 3 and Table 4).
– A hardware, non maskable interrupt is generated from the front panel line ABORT.

The 'BIO' input signal is a particular type of interrupt which is preferable when performing time critical loops. It is also called 'external polling input for bit test and jump operation'. In conjunction with the 'BIOZ' instruction that tests the input of the 'BIO' signal, it is possible to synchronize an external device to the program. This is in fact the way used to sense the external trigger in the FEMC – DELPHI trigger application.

3.2.2.7 Software triggers.

A software trigger from the VME – BUS can be generated any time, even during TMS32010 process by writing D2 = 1 at the Board – Base – Address + $2010 (see Table 3 and Table 4).
A software trigger can be generated by the TMS32010 itself with an "IN PA7" instruction (It must be enabled by bit 5 of Command Register)

3.2.2.8 Software reset to the TMS32010.

– The TMS32010 is forced to the reset state from the VME – BUS by writing D0 = 1 at the Board – Base – Address + $2010.
– The TMS32010 is released from the reset state by a writing D0 = 0 at the Board – Base – Address + $2010 (see Table 3).

3.2.3 Logical Unit communication with External Input Devices

The ANIN board is connected to the Logical Unit bus through optocouplers (see Appendix A for ANIN1 description).

Two other signals are present on the front panel and are described in the chapter 6 'front panel connectors'. The TRIGGER signal latches analog input data into the ANIN board and starts program execution, the ABORT signal interrupts calculation and clears buffers and FIFO's.

3.2.4 Logical Unit communication with External Output Devices.

A 20-pin connector on the front panel, provides the digital output (16-bit word data) on a three-state TTL unidirectional BUS. On these lines are available the result written in the Out-FIFO's.
As soon as data is present on the Out-FIFO's, a Low TTL signal 'OR' is placed on the 'PL9' connector. The external unit detects this 'OR' signal and sends out an unclock to the connector 'PL0'. At the falling edge of 'PL0', the Logical Unit places 16-bit data on 'P3A' connector (see section 6.3, 6.4, Table 7 and Fig. 6).

3.3 Memory Map & I/O Functions

Two types of external memories are foreseen, accessible both to the VME and to the TMS32010: the "program memory" and the "data memory".
The "program memory" is a 2 kiloword memory, located on the mother board: it contains the TMS32010 program and main data.
The "data memory" can range from 2kiloword to several kilowords by using the bank switching technique with the limitation that for subsequent access to memory locations belonging to different banks, an extra time (400 nsec for the DSP TMS32010) is necessary. There will be no difference for each memory access (400 nsec for TMS32010) within the same 2kword memory bank. It can be located on the ANIN board for additional data storage and/or direct data storage from the flash ADC's.
The addresses of the two memories are given in Table 3

The TMS32010 reads or writes into external memories by "TABLR" or "TABLW" instructions.
On the VME side, the addresses given in Table 3 represent the displacement with respect to the base address of the board.
The base address is selected through onboard jumpers, J1/J16, described in detail in section 5. For the logic unit B a fixed additional displacement of $4000 must be added.

In addition to conventional memory read/write instructions, there are a few instructions needed to read or write external registers, FIFO memories, etc.
On the TMS32010 side, these instructions exploit the input/output possibility through the 8 parallel I/O ports ("IN n" or "OUT n" for input or output, through port n). With the "IN n" instruction the content of the data lines is written in the addressed internal memory of the TMS32010, the opposite holds for the "OUT n" instruction.

On the VME side, all these functions are treated as conventional reading or writing instructions, starting at a displacement, with respect to the board base address, of $2000.

The detailed list and the correspondence with the TMS32010 functions is given in Table 3 A "R", "W" or "R/W" added in parenthesis near the displacement indicates whether input, output, or input/output function is implemented while a "*" indicates that the TMS32010 must be in the reset state.

Table 3: MEMORY MAP & I/O FUNCTIONS

NAME	VME addr. displac.	TMS addr. or I/O func.	DESCRIPTION
PMEM	0 (*)	$0	1st address of Program Memory
"	$FFF(*)	$7FF	Last address of Program Memory
DMEM	$1000(*)	$800	1st address of External Data Memory
"	$1FFF(*)	$FFF	Last address of External Data Memory
CMD	$2000(W*)	OUT 0	Write 8-bit Command Register
CMD	$2000(R*)	IN 0	Read 8-bit Command Register
AIN	$2002(R*)	IN 1	Read next value from ANIN (reset BIO)
BOXT	$2004(W*)	OUT 2	Write 16-bit Mail Box Reg. TMS32010 side
BOXT	$2004(R*)	IN 2	Read 16-bit Mail Box Reg. VME side
STS	$2006(R*)	IN 3	16-bit Status Reg. (Tab. 5)
ICW	$200A(R*)	IN 5	8-bit CW FIFO's
ELO	$200C(W*)	OUT 6	bits 0/15 of OUT FIFO's
ICCW	$200C(R*)	IN 6	8-bit CC FIFO's
OCCCW	$200E(W*)	OUT 7	8-bit CW & CC FIFO's
STRIF	$200E(R*)	IN 7	Reset lost trigger flag or software trigger if CMD bit 5 = 1 or interrupt of TMS if CMD bit 4 = 1
RSTRI	$2010(W)		TMS32010 Reset Command (D0 = 1) Interrupt to TMS32010 (D1 = 1) or software trigger (D2 = 1)
BOX6	$2012(W)		Write 16-bit Mail Box Reg. VME side
BOX6	$2012(R)		Read 16-bit Mail Box Reg. TMS32010 side. TMS must be active
RDFI	$2014(R)		16-bit OUT FIFO's (read)
ICR0	$2018(R/W)		Interrupt Control Reg.0 to VME (OUT FIFO's)
ICR1	$201A(R/W)		Interrupt Control Reg.1 to VME (Mail Box Reg.)
IVR0	$201C(R/W)		Interrupt Vector of OUT FIFO's to VME
IVR1	$201E(R/W)		Interrupt Vector of soft interrupt from Mail Box Reg. to VME

3.4 Command Register (CMD).

The Command Register is an 8 – bit register that can be read/written from VME with an I/O base address + $2000 memory location and from TMS32010 with IN PA0 or OUT PA0 instructions. (See Table 3)

Table 4: COMMAND REGISTER

BIT	NAME	ST	DESCRIPTION
0	not used		
	not used		
1	CRSI	1	Reset pointer ANIN
		0	Enable fetch from ANIN
2	ERSF	1	Enable Reset all FIFO's by software
		0	Enable Reset all FIFO's by Abort
3	CRSF	1	Reset all FIFO's
		0	Enable writing to FIFO's
4	EIN3	1	Enable interrupt of TMS32010
		0	Disable interrupt of TMS32010
5	EITR	1	Enable Internal Trigger
		0	Disable Internal Trigger
6	FITR	1	BIOZ from FIFO's Full
		0	BIOZ from Trigger
7	OULA	1	Out – FIFO's generate BIO
		0	C.C.W. & C.W. FIFO's generate BIO

Bit 1: when set to 0, allows reading the 8 input channels. The reading is done by consecutive IN 1 TMS32010 instruction or by "READ" base addr + $2002 from VME – BUS. These instructions, whether from TMS32010 or from the Crate Controller, reset also the 'BIO' level and increment automatically the ANIN pointer.
When set to 1, resets the pointer to the first of the analog input channels on the ANIN card without waiting for a hardware ABORT signal.

Bit 2 : enables the software reset of all FIFO's.

Bit 3: when set to 1 resets all FIFO's, when set to 0 allows writing to the FIFO's.

Bit 6 & bit 7 provide the possibility to check rapidly if there is still space on the FIFO's before writing into them and prevents thus from losing data. This is done by swapping the BIO function from TRIGGER (bit 6 set to 0) to FIFO's (bit 6 set to 1 and bit 7 signalling whether Out or CW – CC FIFO's).

The program could be written in this form:

- The program loops on a BIOZ instruction until a Trigger is generated
- then the program switches bit 6 of COMMAND REGISTER to "1"
- the TMS32010 processes the data.
- Before writing to the FIFO's, the program checks by a BIOZ instruction if there is space in the FIFO memory and eventually loops until space has been freed.
- At the end of the analysis, the trigger is enabled by setting bit 6 to 0.

3.5 Status Register (STS).

The 16-bit status register can be read by VME with the instruction read base address + $2006 or by TMS32010 with a "IN 3" instruction (see Table 3).

3.5.1 Status Register Flag clearance.

- Bits 0/5. The FIFO's can be cleared: by software using bit 2 & bit 3 of Command Register; by an hardware ABORT, if enabled by bit 2 of Command Register.
- Bits 6/9 cannot be cleared since they don't belong to this TMS32010
- A TMS32010 instruction "IN 3" clears the following bits:
  bit 10 Reset Flag (Reset at TMS32010, set from VME-BUS)
  bit 12 Interrupt from VME-BUS Flag
  bit 13 Interrupt from TMS32010 (Trace)
  bit 15 ABORT Status Flag
- A TMS32010 intruction "IN 7" clears the "Lost trigger from front panel Flag" (bit 11).

Table 5: STATUS REGISTER

BIT	NAME	ST	DESCRIPTION
0	F1OR	1	16-bit Out-FIFO's "OUTPUT READY" ("OR")
		0	16-bit Out-FIFO's "EMPTY"
1	F1IR	1	16-bit Out-FIFO's "FULL"
		0	16-bit Out-FIFO's "INPUT READY" ("IR")
2	F2OR	1	8-bit C.W.FIFO's "OUTPUT READY"
		0	8-bit C.W.FIFO's "EMPTY"
3	F2IR	1	8-bit C.W.FIFO's "FULL"
		0	8-bit C.W.FIFO's "INPUT READY"
4	F3OR	1	8-bit C.C.W.FIFO's "OUTPUT READY"
		0	8-bit C.C.W.FIFO's "EMPTY"
5	F3IR	1	8-bit C.C.W.FIFO's "FULL"
		0	8-bit C.C.W.FIFO's "INPUT READY"
6	F4OR	1	8-bit C.C.W.FIFO's "OR" of adjacent C.W.TMS32010
		0	8-bit C.C.W.FIFO's "EMPTY" of adjacent C.W.TMS32010
7	F4IR	1	8-bit C.C.W.FIFO's "FULL" of adjacent C.W.TMS32010
		0	8-bit C.C.W.FIFO's "IR" of adjacent C.W. TMS32010
8	F5OR	1	8-bit C.W.FIFO's "OR" of adjacent C.C.W.TMS32010
		0	8-bit C.W.FIFO's "EMPTY" of adjacent C.C.W.TMS32010
9	F5IR	1	8-bit C.W.FIFO's "FULL" of adjacent C.C.W.TMS32010
		0	8-bit C.W.FIFO's "IR" of adjacent C.C.W.TMS32010
10	REOC	1	Reset at TMS32010 from VME-BUS occurred
		0	No Reset occurred
11	LOTR	1	Lost triggers from front panel connector
		0	No triggers occurred during data analysis
12	I68	1	Interrupt from VME-BUS to TMS32010
		0	No Interrupts from VME-BUS occurred
13	ITR	1	Interrupt from TMS32010 'IN PA7' (Trace)
		0	No Interrupt from TMS32010 occured.
14			Not used
15	ABOC	1	ABORT occurred
		0	No ABORT occurred

3.6 FIFO board

On this piggy back board are located the Clock – Wise, Counter – Clock – Wise and Out – FIFO's memories. The layout of connectors with the mother board is given in Fig. 4 The corresponding connectors sitting on the mother board are shown in Fig. 7 and described in section 8.2.

The reason for having a separate board for the FIFO memories, is only due to cost, availability of components at the beginning of the 1986 and organization of parallel in, parallel out FIFO's. If there arise a need in the future to redesign the FDDP six layer printed – circuit mother board for some reason, certainly this FIFO board will not exist any more because with the higher density FIFO's existing now on the market at lower price it will be possible to put all components on a single mother board.

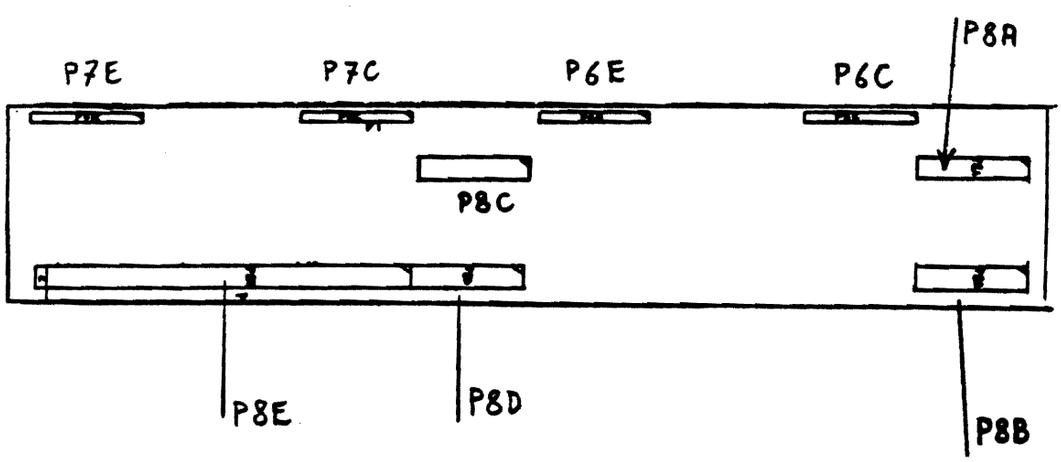


Figure 4: FIFO onboard connectors layout

4. USER SIGNAL DESCRIPTION

SIGNAL MNEM.	LOCATION	SIGNAL NAME AND DESCRIPTION
A01 – A15	P1	ADDRESS BUS (BITS 1/15) VME – BUS. Three – state driven address lines that are used to broadcast a short address.
A16 – A23	P1	ADDRESS BUS (BITS 16/23) VME – BUS. Three – state driven address lines that are used in conjunction with A01 – A15 to broadcast a standard address.
A24 – A31	P2	ADDRESS BUS (BITS 24/31) VME – BUS. Three – state driven bus expansion address lines.
/ABORT	ABORT	ABORT PROCESSING ON TMS32010 (A & B). The Abort Signal generates an interrupt to TMS32010 (A & B) and resets the ANIN board. See also "Front Panel Signals Description".
/ATRIG	P6A,P7A	TRIGGER TO THE ANIN BOARD. Trigger signal from the mother board to the ANIN board.
AM0 – AM5	P1	ADDRESS MODIFIER (BITS 0 – 5) VME – BUS. Three – state driven lines that provide additional information about the address bus, such as size, cycle type, and/or DTB master identification. Allowed configurations are 09 – 0D – 39 – 3D
/AS	P1	ADDRESS STROBE VME – BUS. Three – state driven signal that indicates a valid address has been placed on the address bus.
/BIO(A – B)	TMS320	TMS320 EXTERNAL POLLING INPUT External polling input of TMS32010 for bit test and jump operation. When a low is present on this pin, executing the "BIOZ" TMS32010 instruction activates the branch Four external sources are connected to this line and selected by the COMMAND REGISTER setting: – External trigger – Internal Trigger – Out – FIFO's full – C.W. & C.C.W.FIFO's full
D00 – D15	P1	DATA BUS (BITS 0 – 15) VME – BUS. Three state driven bidirectional data lines that provide a data path between the DTB master or slave.
/DEN	TMS320	Data enable indicates the TMS32010 accepting input data on D15 – D0 for its data memory.

/DS0	P1	DATA STROBE 0 VME – BUS. Three state driven signal that indicates during byte and word transfers that a data transfer will occur on D0 – D7 data bus lines.
/DS1	P1	DATA STROBE 1 VME – BUS. Three state driven signal that indicates during byte and word transfers that a data transfer will occur on D8 – D15 data bus lines.
/DTACK	P1	DATA TRANSFER ACKNOWLEDGE VME – BUS. Open – collector driven signal generated by a slave. The falling edge of this signal indicates that valid data is available on the data bus during a read cycle, or that data has been accepted from the data bus during a write cycle. The rising edge indicates when the SLAVE has released the data bus at the end of a READ cycle.
FxIR (A – B)	P6E,P7E	FIFO'S INPUT READY High when the FIFO's are not FULL and FIFO's Load Clock input is high. Low when the FIFO's are FULL. – available on bit 1, 3, 5, 7 and 9 of STS – it can be used to prevent further trigger processing if FIFO's are full.
/F1OR (A – B)	P6C,P7C	OUTPUT – READY OUT FIFO'S Generated when there is a data present on the 16 – bit OUT – FIFO's – interrupts the VME via the MC68153 interrupt controller – intended to request fetch by external Energy Sum Processor (see below /FRF) via front panel connector PL9.
FxOR (A – B)	P6C,P7C	FIFO'S OUTPUT READY High when a data is present on the FIFO's – available on bit 2, 4, 6 and 8 of STS
/FRF (A – B)	PL9A PL9B	FRONT PANEL READ "OUT – FIFO'S" Generated by the external device Energy Sum Processor, it unclocks the 16 – bit result from the "OUT – FIFO's" of TMS32010 (A or B) and sends it through buffers to the Front Panel Out – FIFO's – BUS.
/IACK	P1	INTERRUPT ACKNOWLEDGE VME – BUS An open – collector or three – state driven signal used by the INTERRUPT HANDLER MC68153 acknowledging an interrupt request.
/IACKIN	P1	INTERRUPT ACKNOWLEDGE IN VME – BUS A totem – pole driven signal. IACKIN* and IACKOUT* signals form a daisy – chain. The IACKIN* signal indicates to the VME bus board receiving it that an INTERRUPT ACKNOWLEDGE CYCLE is in progress.
/IACKOUT	P1	INTERRUPT ACKNOWLEDGE OUT VME BUS

		A totem-pole driven signal. IACKIN* and IACKOUT* signals form a daisy-chain. The IACKOUT* signal is sent by a board to indicate to the next board in the daisy-chain that an INTERRUPT ACKNOWLEDGE CYCLE is in progress.
/IRQ1-7	P1	INTERRUPT REQUEST (1-7) VME-BUS. Open-collector driven signals, generated by the interrupt controller MC68153, which carry prioritized interrupt requests. Level seven is the highest priority.
/MEN	TMS320	TMS320-DATA ENABLE ON PROGRAM MEMORY Memory enable indicates that D15-D0 will accept external program memory word.
MR	P6A,P7A	READ/WRITE ENABLE FOR ANIN MEMORY Read/Write signal for the Program or Data memory on the ANIN board
MSX	P6A,P7A	CHIP SELECT FOR ANIN MEMORY Selects the Program or Data Memory on the ANIN board.
R1	P6A,P7A	INPUT DIGITIZED DATA FROM ANIN A pulse on this line fetches data from Flash ADC converters.
Rx	P6E,P7E	READ CW & CC FIFO's. Read Clock-wise and Counter-clock-wise FIFO's.
RES		0 = TMS32010 Activity 1 = VME Activity
RFILW	P8B	READ 16-bit Out-FIFO's. Output Enable signal to the Out-FIFO's buffers.
ACRSI	P6A,P7A	RESET POINTER ANIN Signal from the mother board to the ANIN board to reset the pointer of the 8 Flash ADC.
RSF	P6C,P7C	RESET ALL FIFO's Signal from the mother board to the FIFO board that resets Clock-wise, Counter-lock-Wise and Out-FIFO's
SLOT	P6A,P7A	LOST TRIGGER SIGNAL Two consecutive trigger occured while the TMS32010 was processing. Thus data of the first trigger are memorized in the Flash ADC buffers and the second will be lost. Signal from the ANIN board to the mother board.
SYSCLK	P1	SYSTEM CLOCK A constant 16-MHz clock signal that is independent of processor speed or timing. This signal is used for general system timing use.
SYSR	P6A,P7A	SYSTEM RESET TO THE ANIN BOARD

/SYSRESET	P1	SYSTEM RESET Open – collector driven signal which, when low, will cause the system to be reset.
UEF	P6C,P7C	UNCLOCK FIFO's DATA. Signal from mother board to FIFO board corresponding to RDFI function of table 3.
/WE	TMS320	TMS320 WRITE ENABLE TO PROGRAM MEMORY Write enable indicates valid data on D15 – D0.
/WRITE	P1	WRITE Three – state driven signal that specifies the data transfer cycle in progress to be either read or write. A high level indicates a read operation; a low level indicates a write operation.
W3,W4	P6A,P7A	WRITE FUNCTION TO ANIN Two write function listed in table 3. (Not used in FEMC – DELPHI)
Wx	P6E,P7E	WRITE FUNCTION TO FIFO's. Signals from the mother board to the FIFO board. (See table 3)

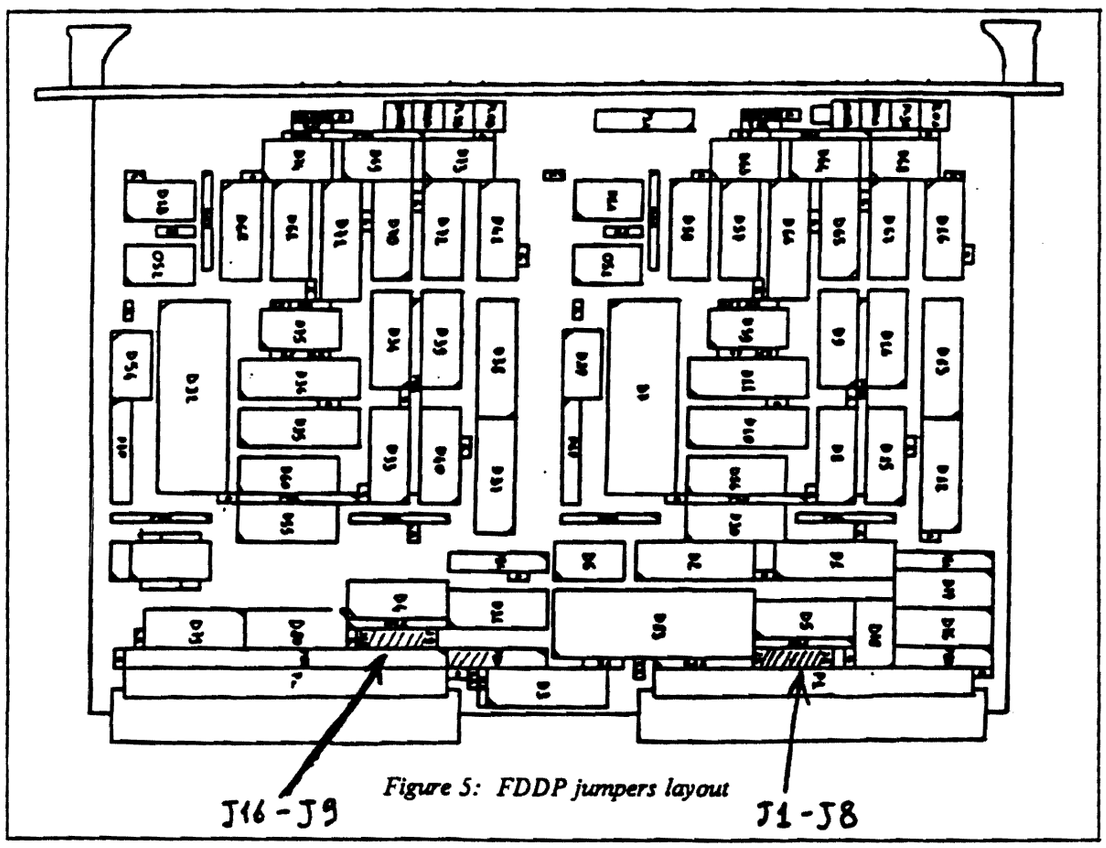


Figure 5: FDDP jumpers layout

5. JUMPER SELECTION

5.1 J1/J16 – MEMORY MAP–SELECT

Table 6: BASE ADDRESS JUMPER SELECTION

ADDR.	STAT.	JUMPER	CONDITION
A31	0	J16	INSERTED
A30	0	J15	INSERTED
A29	0	J14	INSERTED
A28	0	J13	INSERTED
A27	0	J12	INSERTED
A26	0	J11	INSERTED
A25	0	J10	INSERTED
A24	0	J9	INSERTED
A23	0	J1	INSERTED
A22	0	J2	INSERTED
A21	1	J3	REMOVED
A20	0	J4	INSERTED
A18	0	J5	INSERTED
A17	0	J6	INSERTED
A16	0	J7	INSERTED
A15	0	J8	INSERTED

A19 – 1 BROADCAST
– 0 SELECT ONLY ONE BOARD

A14 – 1 SELECT TMS32010B
– 0 SELECT TMS32010A

A13 – 1 SELECT I/O FUNCTIONS
– 0 SELECT MEMORY FUNCTIONS

(The previous example of decoding is for ADDR = 00200000 Hex)

6. FRONT PANEL CONNECTORS

6.1 TRIGGER

- 30 nsec from the falling edge of the TTL TRIGGER signal, the 16 input analog values are sampled simultaneously. They are converted by sixteen 8 – bit FLASH ADC converters.
- At the Trigger falling edge, a "BIO" signal is sent to the two TMS32010 processors. Each TMS32010 will read sequencially N outputs of the ADC converters with a squence of IN 1 instructions. The IN 1 instruction automatically reset the 'BIO' level.
  Input values in the TMS32010 memory are then processed to give a result in a 16 – bit word Out – FIFO's.
  When the result is present on the Out – FIFO's, an interrupt is generated to the VME – BUS, a level is set on the "Status Register – bits" and on the "Front Panel OR – line". A master external device (Energy Sum Processor for the Delphi trigger application) connected to the front panel connector "P3" can then fetch asynchronously the results at the maximum speed of 20 Mbyte/sec.
- During acquisition time (= conversion time 200 nsec plus "IN" from TMS320 Nx400 nsec) further triggers are disabled (see section 9.3).
- Further triggers occurring during the acquisition time are flagged by the "Lost – Trigger – Flag" (bit 11 of STS of Table 5).
- Lemo connector (see Fig 6)
- Ground and Signals are referred to the Digital Power Supply.

6.2 ABORT

- A TTL falling edge signal on this line stops the process (see also section 9.3).
- Within 25 nsec from the falling edge of "ABORT", the board is ready to make the acquisition of other 16 analog input channels
- It resets the pointer of ANIN – board.
- It resets "Lost – Trigger – Flag" settings.
- If enabled by bit 2 of Command Register it clears all FIFO's.
- Lemo (see Fig. 6)
- Ground and Signals are referred to the Digital Power Supply.

6.3 PL9A & PL9B "Output Ready (OR) of Out – FIFO's" signals

- When a result is present on the Out – FIFO's, a TTL level – 1 is set on these lines. The master external device (Energy Sum Processor for the Delphi trigger application) connected to the "P3" front panel DATA – BUS connector, reading this line can then fetch at any time the results from the Out – FIFO's.
- Lemo (see Fig. 6)
- Ground and signals are referred to the Digital Power Supply.

6.4 PL0A & PL0B "Unclock of Out – FIFO's" signals

- At the maximum rate of 10 MHz an "UNCLOCK" TTL pulse can be sent from an external master device (Energy Sum Processor for the Delphi trigger application) in order to read the results from the Out – FIFO's of each module (see Fig 10, Fig. 11 and section 9.4).
- Lemo (see Fig. 6)

- Ground and signals are referred to the Digital Power Supply (see Table 7).

6.5 P3A "16 – bit Out – FIFO's" signals

It is a three – state bus carrying the TTL levels from the Out – FIFO's. Any external device can forward the Out – FIFO's data onto this bus by sending the Out – FIFO unclock signal.

Table 7: P3A "Front panel Out – FIFO's DATA – BUS". Pin assignment

1	NC	NC	11
2	12P8D – D0	2P8D – D8	12
3	13P8D – D1	3P8D – D9	13
4	14P8D – D2	4P8D – D10	14
5	15P8D – D3	5P8D – D11	15
6	16P8D – D4	6P8D – D12	16
7	17P8D – D5	7P8D – D13	17
8	18P8D – D6	8P8D – D14	18
9	19P8D – D7	9P8D – D15	19
10	GND	GND	20

- 16 pin connectors (see Fig. 6)
- Ground and signals are referred to the Digital Power Supply.

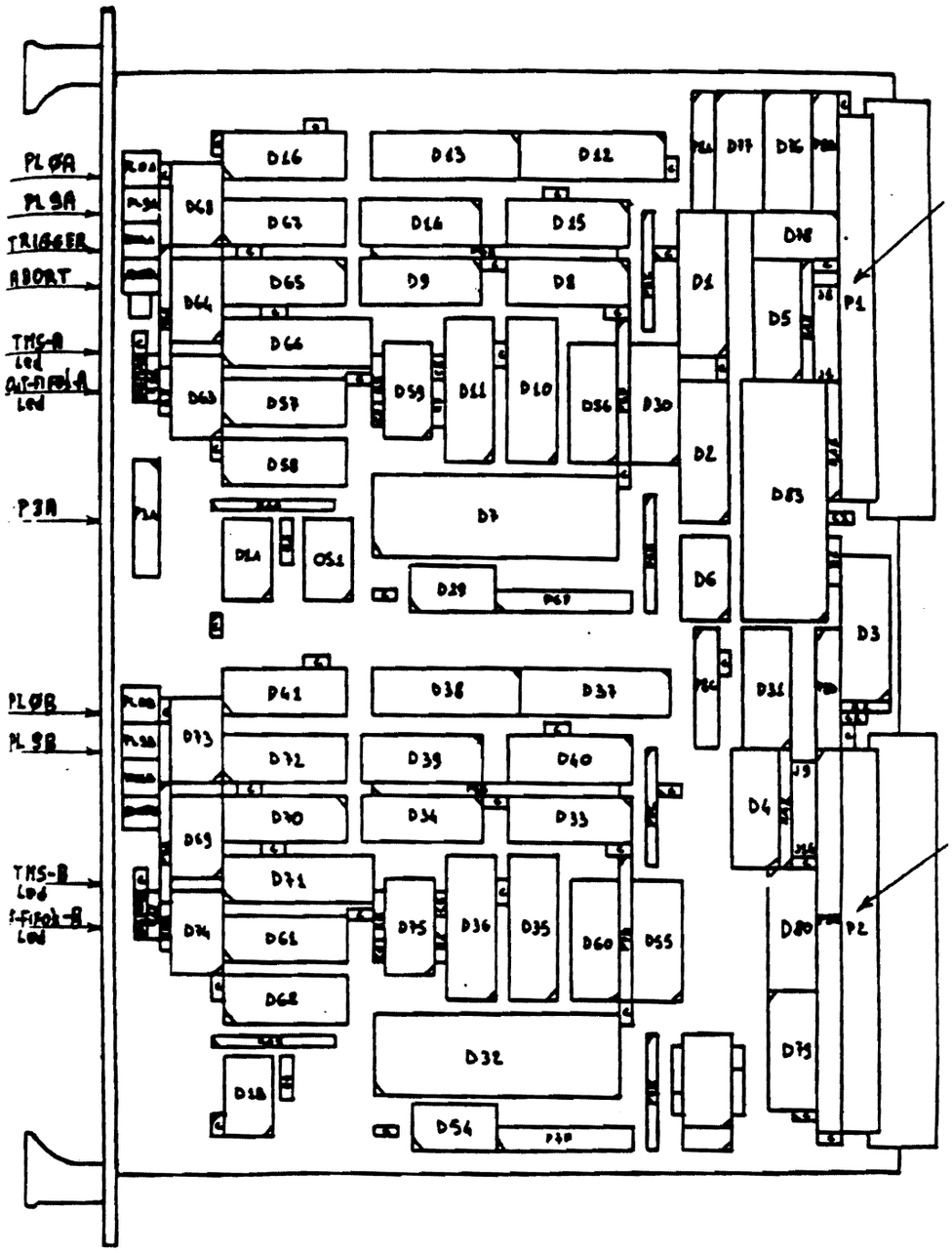


Figure 6: FDDP front panel and back plane connectors layout

7. BACK – PLANE CONNECTORS

7.1 P1 "VME backplane connector" signals.

All lines are referred to the Digital Power Supply.
This appendix identifies the VME – BUS backplane P1 (see Fig. 6) connector pin assignments. The following table lists the pin assignments by pin number order. (The connector consists of three rows of pins labeled rows 'a', 'b', and 'c'. See Tab. 8).
The description of each line used on this board is listed in alphabetical order in chapter 4 "User Signal description". The lines not used on this board are described in the VME – BUS Specification Manual.

Table 8: P1 back plane connector VME – BUS. Pin assignment

PIN	ROW a SIGNAL MNEMONIC	ROW b SIGNAL MNEMONIC	ROW c SIGNAL MNEMONIC
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFALL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK (1)	A17
22	IACKOUT*	SERDAT (1)	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	– 12V	+ 5V STDBY	+ 12V
32	+ 5V	+ 5V	+ 5V

7.2 P2 "VME backplane connector" signals

The 32-lines of the central row "b" of this connector are the extension of the VME-BUS for 32 bit DATA and Address. The 32-lines of side rows "a" and "c" are reserved for external analog ground, supply voltage and C.W.FIFO/C.C.W.FIFO's comunication signals (see Tab. 9 and Fig. 6).

Table 9: P2 back plane connector VME-BUS. Pin assignment

PIN	ROW a SIGNAL MNEMONIC	ROW b SIGNAL MNEMONIC	ROW c SIGNAL MNEMONIC
1	P8E > D0-A	+5 Volts	33-P8E > D0-BCWF
2	P8E > D1-A	GND	34-P8E > D1-BCWF
3	P8E > D2-A	RESERVED	35-P8E > D2-BCWF
4	P8E > D3-A	A24	36-P8E > D3-BCWF
5	P8E > F5IR-B>	A25	37-P8E > F5IR-B
6	P8E > F5OR-B>	A26	38-P8E > F5OR-B
7	P8E < IN 6-A	A27	39-P8E < IN 6-A<
8	P8E > D4-A	A28	40-P8E > D4-BCWF
9	P8E > D5-A	A29	41-P8E > D5-BCWF
10	P8E > D6-A	A30	42-P8E > D6-BCWF
11	P8E > D7-A	A31	43-P8E > D7-BCWF
12	NC	GND	NC
13	A-	+5 Vots	A-
14	A-	D16	A-
15	A-	D17	A-
16	AGND	D18	AGND
17	AGND	D19	AGND
18	AGND	D20	AGND
19	A+	D21	A
20	A+	D22	A
21	A+	D23	A
22	P8E < D0-ACCWF	GND	54-P8E < D0-B
23	P8E < D1-ACCWF	D24	55-P8E < D1-B
24	P8E < D2-ACCWF	D25	56-P8E < D2-B
25	P8E < D3-ACCWF	D26	57-P8E < D3-B
26	P8E < F4IR-A	D27	58-P8E < F4IR-A<
27	P8E < F4OR-A	D28	59-P8E < F4OR-A<
28	P8E > IN 5-B>	D29	60-P8E > IN 5-B
29	P8E < D4-ACCWF	D30	61-P8E < D4-B
30	P8E < D5-ACCWF	D31	62-P8E < D5-B
31	P8E < D6-ACCWF	GND	63-P8E < D6-B
32	P8E < D7-ACCWF	+5 Volts	64-P8E < D7-B

D0−A = D0 of TMS32010A data bus.
D0−B = D0 of TMS32010B data bus.
D0−ACCWF = D0 of Counter Clock Wise FIFO data bus of TMS32010A
D0−BCWF = D0 of Clock Wise FIFO data bus of TMS32010B
IN 6−A = IN PA6 from TMS32010A; IN 6−A< = IN PA6 from off board TMS32010A

All lines, except the ones listed below are referred to the Digital Power Supply.
4 AGND −lines carry the analog ground
4 A+ −lines carry the +5 Volt analog supply.
4 A− −lines carry the −5 Volt analog supply.

Information flow direction are indicated by ">" and "<" angular simbols (see also Fig.2).

8. ON BOARD CONNECTORS

8.1 FDDP connectors to piggy – back ANIN boards

These connectors are provided for the analog input modules "ANIN"

Table 10: P6A & P7A CONTROL LINES for ANIN board

1	R1
2	ATRIG
3	SLOT
4	ACRSI
5	ABORT
6	MSX
7	MR
8	SYSR
9	W4
10	W3

Table 11: P6D & P7D ADDRESS LINES for ANIN board

1	A11
2	A10
3	A9
4	A2
5	A5
6	A6
7	A7
8	A1
9	A3
10	A0
11	A4
12	A8

Table 12: P6B & P7B TMS32010 DATA BUS for ANIN board

1	D0
2	D1
3	D2
4	D3
5	D4
6	D5
7	D6
8	D7
9	NC
10	NC
11	NC
12	NC
13	D8
14	D9
15	D10
16	D11
17	D12
18	D13
19	D14
20	D15

Table 13: P6F & P7F POWER SUPPLY for ANIN board

1	+5	+5	11
2	+5	+5	12
3	GND	GND	13
4	GND	GND	14
5	A+	A+	15
6	A+	A+	16
7	AGND	AGND	17
8	AGND	AGND	18
9	A−	A−	19
10	A−	A−	20

All lines, except the ones listed below, are referred to the Digital Power Supply.
GNDA 4−lines for analog ground.
+5VA 4−lines for analog power supply
−5VA 4−lines for analog power supply

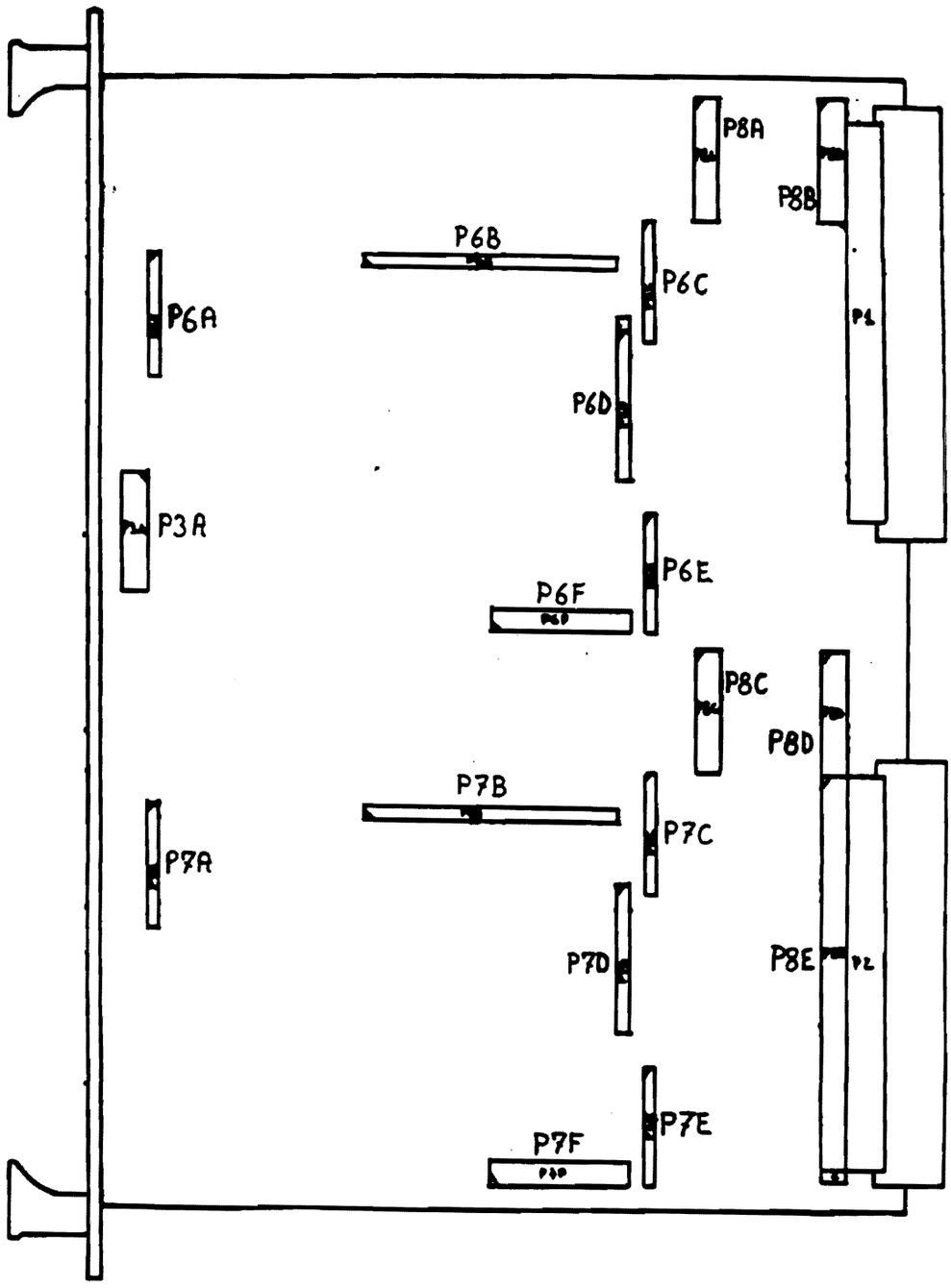


Figure 7: FDDP onboard connectors layout for ANIN and FIFO boards

8.2 FDDP connectors to piggy – back FIFO board

Table 14: P6C & P7C / P6E & P7E CONTROL LINES for FIFO board

1	F5OR	W7
2	F4OR	W6
3	F3OR	W5
4	F2OR	R6
5	/F1OR	R5
6	UEF	F5IR
7	NC	F4IR
8	RSF	F3IR
9	NC	F2IR
10	NC	F1IR

Table 15: P8A & P8C DATA – BUS of TMS32010A & TMS32010B for FIFO board

1	GND	GND	11
2	D0	D8	12
3	D1	D9	13
4	D2	D10	14
5	D3	D11	15
6	D4	D12	16
7	D5	D13	17
8	D6	D14	18
9	D7	D15	19
10	+5	+5	20

Table 16: P8B DATA – BUS from transreceivers to VME for FIFO board

1	D0	D8	11
2	D1	D9	12
3	D2	D10	13
4	D3	D11	14
5	D4	D12	15
6	D5	D13	16
7	D6	D14	17
8	D7	D15	18
9	NC	NC	19
10	RFILW	NC	20

Table 17: P8D Out – FIFO's DATA – BUS for FIFO board

1	GND	GND	11
2	D8 12P3A	D0 2P3A	12
3	D9 13P3A	D1 3P3A	13
4	D10 14P3A	D2 4P3A	14
5	D11 15P3A	D3 5P3A	15
6	D12 16P3A	D4 6P3A	16
7	D13 17P3A	D5 7P3A	17
8	D14 18P3A	D6 8P3A	18
9	D15 19P3A	D7 9P3A	19
10	+5	+5	20

Table 18: P8E TMS32010A & TMS32010B DATA – BUS to CW & CCW FIFO's

for FIFO board

1	1P2A	33P2C	33
2	2P2A	34P2C	34
3	3P2A	35P2C	35
4	4P2A	36P2C	36
5	5P2A	37P2C	37
6	6P2A	38P2C	38
7	7P2A	39P2C	39
8	8P2A	40P2C	40
9	9P2A	41P2C	41
10	10P2A	42P2C	42
11	11P2A	43P2C	43
12	NC	NC	44
13	NC	NC	45
14	NC	NC	46
15	NC	NC	47
16	NC	NC	48
17	NC	NC	49
18	NC	NC	50
19	NC	NC	51
20	NC	NC	52
21	NC	NC	53
22	22P2A	54P2C	54
23	23P2A	55P2C	55
24	24P2A	56P2C	56
25	25P2A	57P2C	57
26	26P2A	58P2C	58
27	27P2A	59P2C	59
28	28P2A	60P2C	60
29	29P2A	61P2C	61
30	30P2A	62P2C	62
31	31P2A	63P2C	63
32	32P2A	64P2C	64

9. APPLICATION EXAMPLES

9.1 FDDP – A modular system unit

FDDP is a modular system based on the 'LOGICAL UNIT' (see description in chapter 3).

A system can be configured with many parallel processing Logical Units with a personalized ANIN (analog input module) capable of executing almost 5 MIPS each.

At present a 30 MIPS system consisting of six Logical Units has been built in VME and works satisfactorily.

9.2 Examples of different personalized ANIN boards.

The ANIN board is the interface between the Logical Unit and the detector and is therefore detector dependent.

The task of this board is to transform signals from analog to digital by means of Flash ADC.

There are essentially two types of applications: either a single digitalization of the signal is needed or the entire shape of the signal is required.

In the first case (as in Delphi trigger application, see Fig. 9 ANIN1), a single register is sufficient for each input line. The task of the DSP is to process the data (e.g. subtract pedestals, correct with calibration constants, sum data, etc.) and eventually correlate the data with neighbor Units by means of CW and CC FIFO's.

In the second case (see Fig. 9 ANIN2) the ANIN samples the analog input at given rates and store results in a data memory. The task of the DSP is to read the memory and make a pulse shape analysis (taking derivatives of the pulse, finding the maximum, minimum, time duration or center of gravity, subtracting pedestals, etc.).

Figure 8 illustrates two possible configurations of FDDP systems for the single register case: the VME version with two Logical Units associated with two ANIN boards, and the FASTBUS version with six of each. The same ANIN boards are used in the two versions except for different analog input connectors. On the VME version the inputs can be affected through distinct LEMO connectors or with coaxial flat. On the FASTBUS version, space limitation on the front panel allows only 16–way coaxial flat connectors. Here the analog inputs to the ANIN boards (see Fig. 8) are carried between ground layers in the inner layers of the printed circuit.

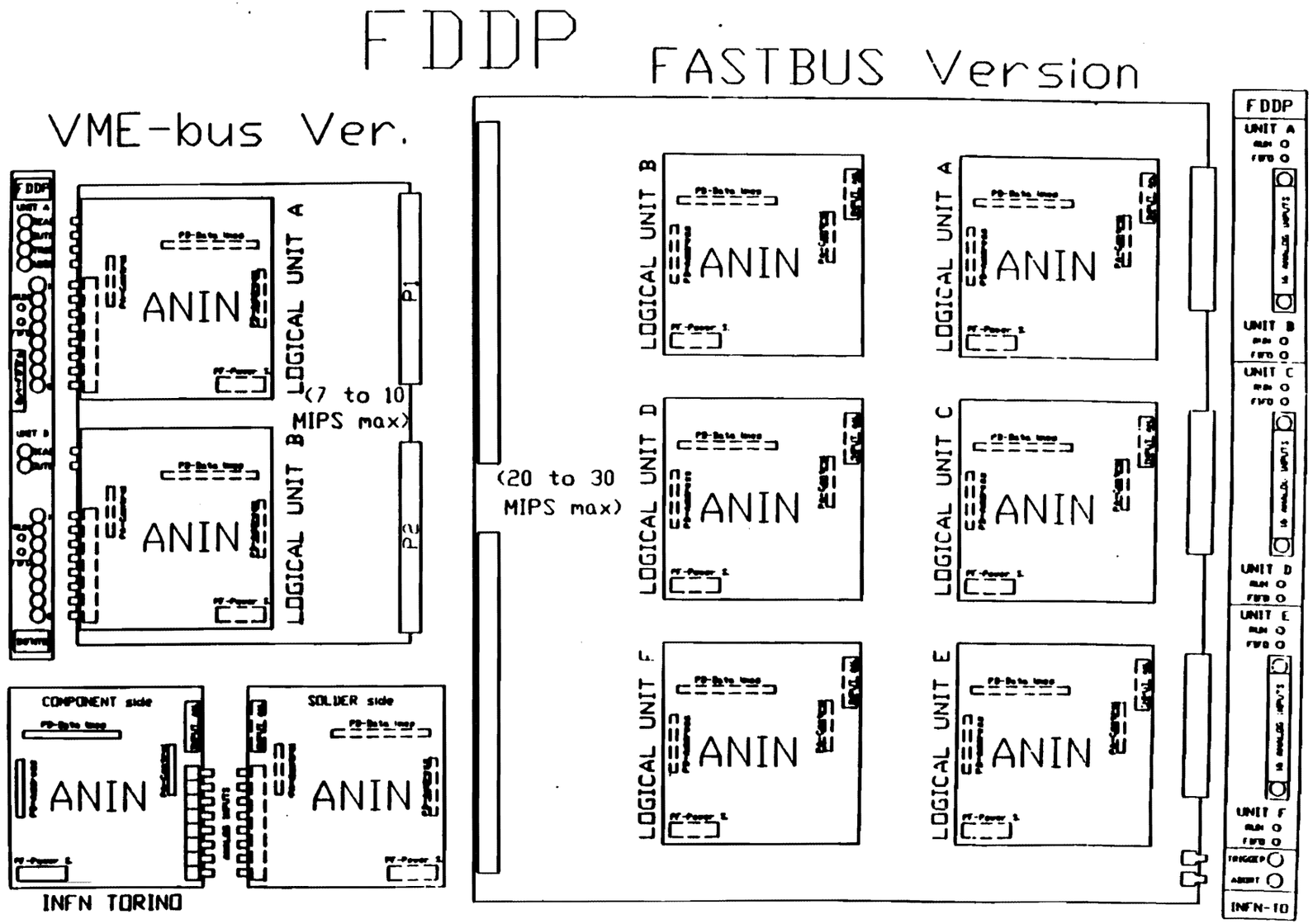
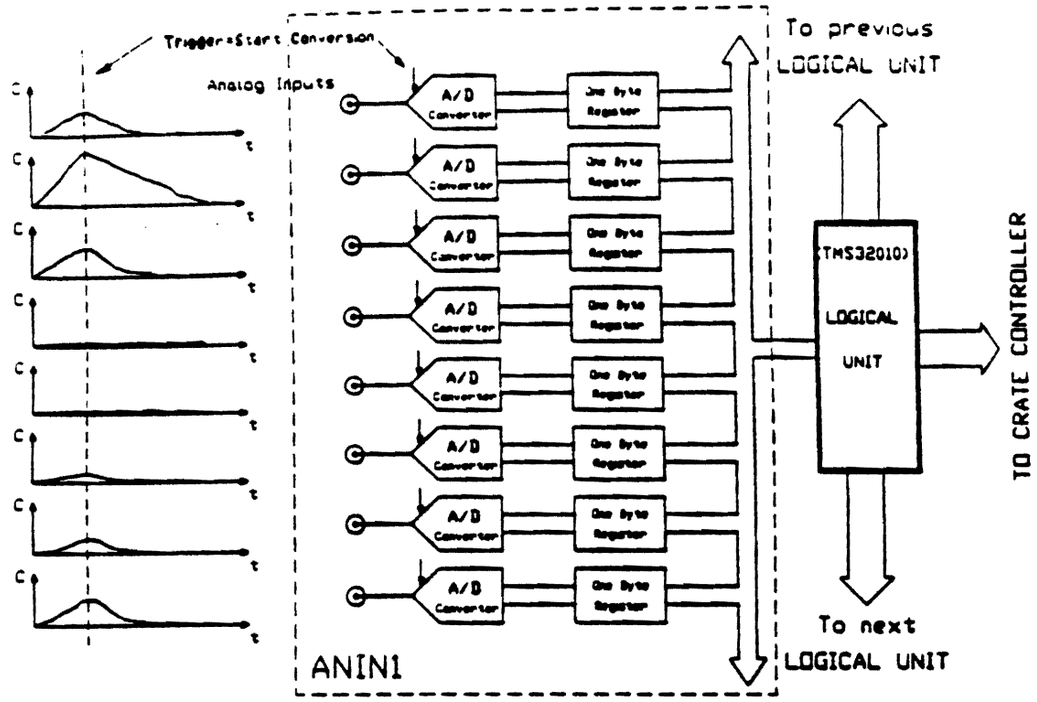


Figure 4: FDDP as a modular system for Fast Digital Signal Processing algorithms



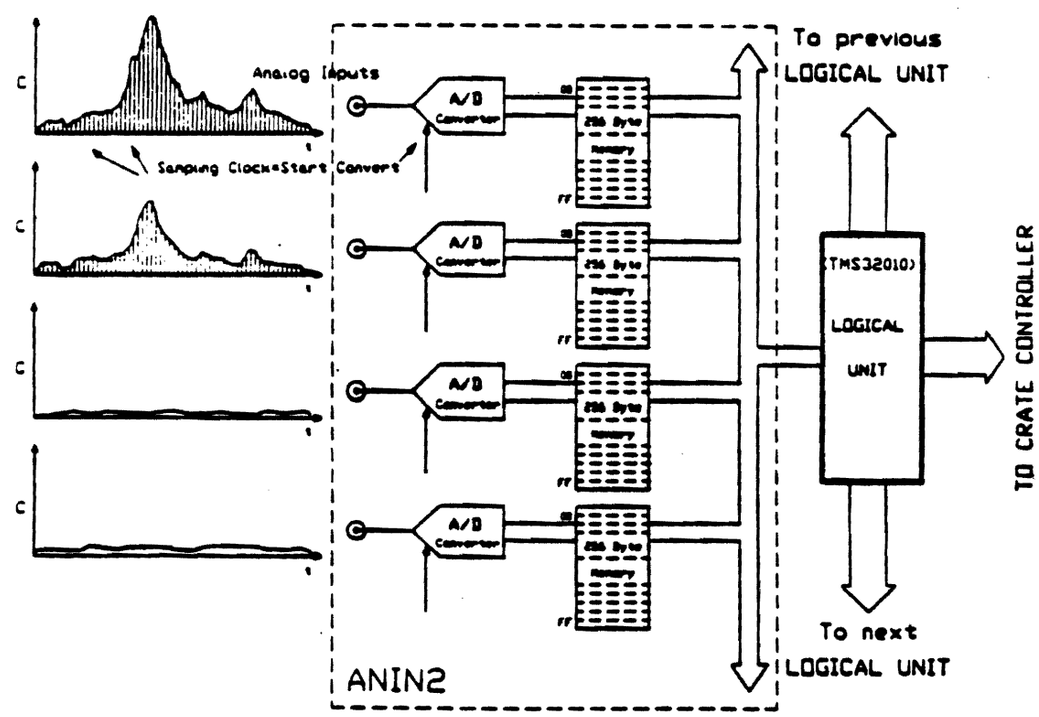


Figure 9: Examples of different personalized ANIN boards

9.3 FDDP System Configuration for Development and Applications

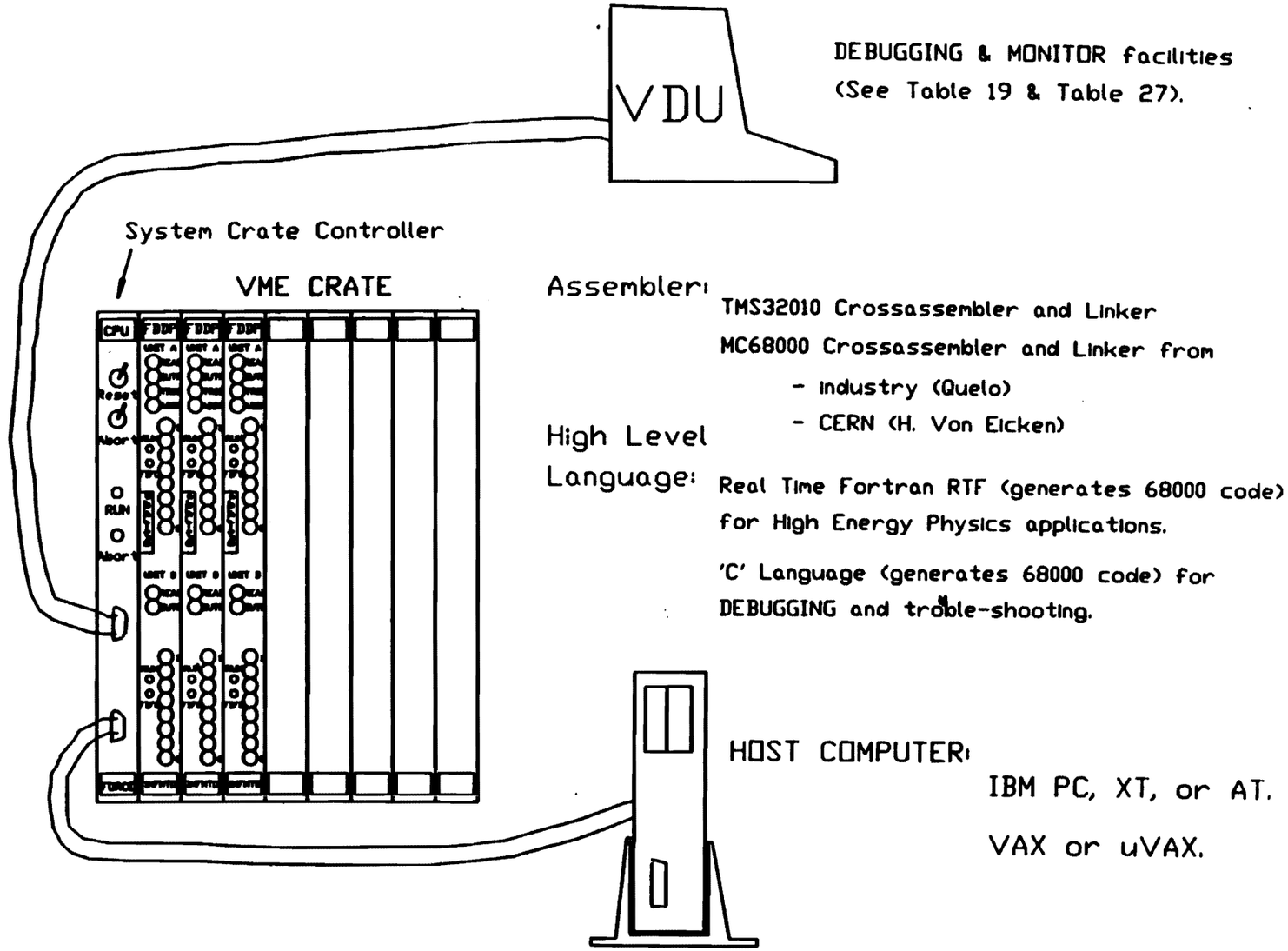


Figure 10: FDDP System Configuration for Development and Applications

9.4 Application example to calorimeters (FEMC – DELPHI)

The Forward Electromagnetic Calorimeter of the DELPHI detector consists of two large fine grained lead glasses walls, equipped with a new type of low gain, magnetic field tolerant, one – stage vacuum phototriode [9]. Each wall consist of about 4000 lead glasses, phototriodes, charge preamplifiers and shapers.

The analog signals derived from the shapers are sent to the readout electronics [10] and to the trigger electronic system [1].

The trigger scheme proposed for DELPHI [11] requires for the FEMC:

- multiplicity data for the first level (3 usec)
- total energy information for the second level (within 39 usec, See Fig. 14).
- cluster finding with the associated information of cluster energy and cluster center of gravity, for the third level (see Fig. 15).

The 4000 shaper signals are reduced in the FEMC system trigger to 1000 (blocks) and 192 (superblocks) by means of analog sums (see Fig. 13). The 192 superblock signals are grouped in 24 sectors and are sent to analog comparators for the first level trigger and to the FDDP board. For each sector there is a Logical Unit with 8 superblock signals as inputs (see Fig. 11 and Fig. 12). The ANIN board is of the type shown in Fig. 9 (ANIN1) and described in Appendix A (see also Fig. 25).

Each Logical Unit at each sector, in parallel with the others, does the following tasks:

1. calculates sector energy for the second level trigger (see timing diagram Fig. 14); for this task the Unit:

- inputs 8 superblock values from ANIN1 module (Ai)
- subtracts pedestals
- compares with threshold
- multiplies Ai x calibration constant of superblocks above threshold, to obtain the superblock Energy Ei
- adds superblock Energies to yield Sector Energy
- writes Sector Energy to Out – FIFO's

2. finds the clusters [12] for the third level trigger (see timing diagram Fig. 15); for this task the Unit:

- reads from Clock Wise and Counter Clock Wise FIFO's
- finds Clusters
- writes Cluster Center of Gravity to Out – FIFO's
- writes Cluster Energy Ec to Out – FIFO's.

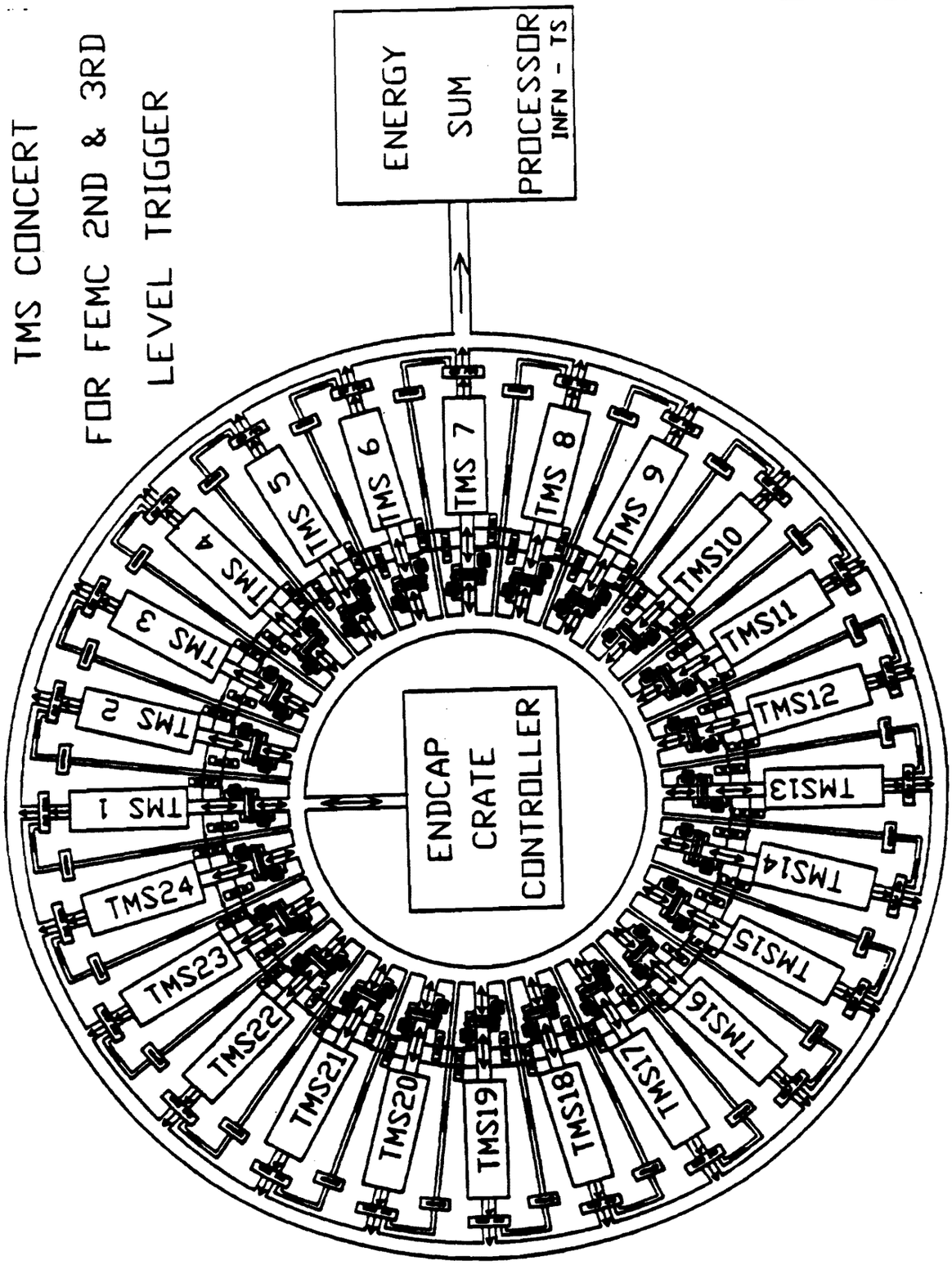


Figure 11: The FEMC 2nd and 3rd level trigger with 24xTMS32010.

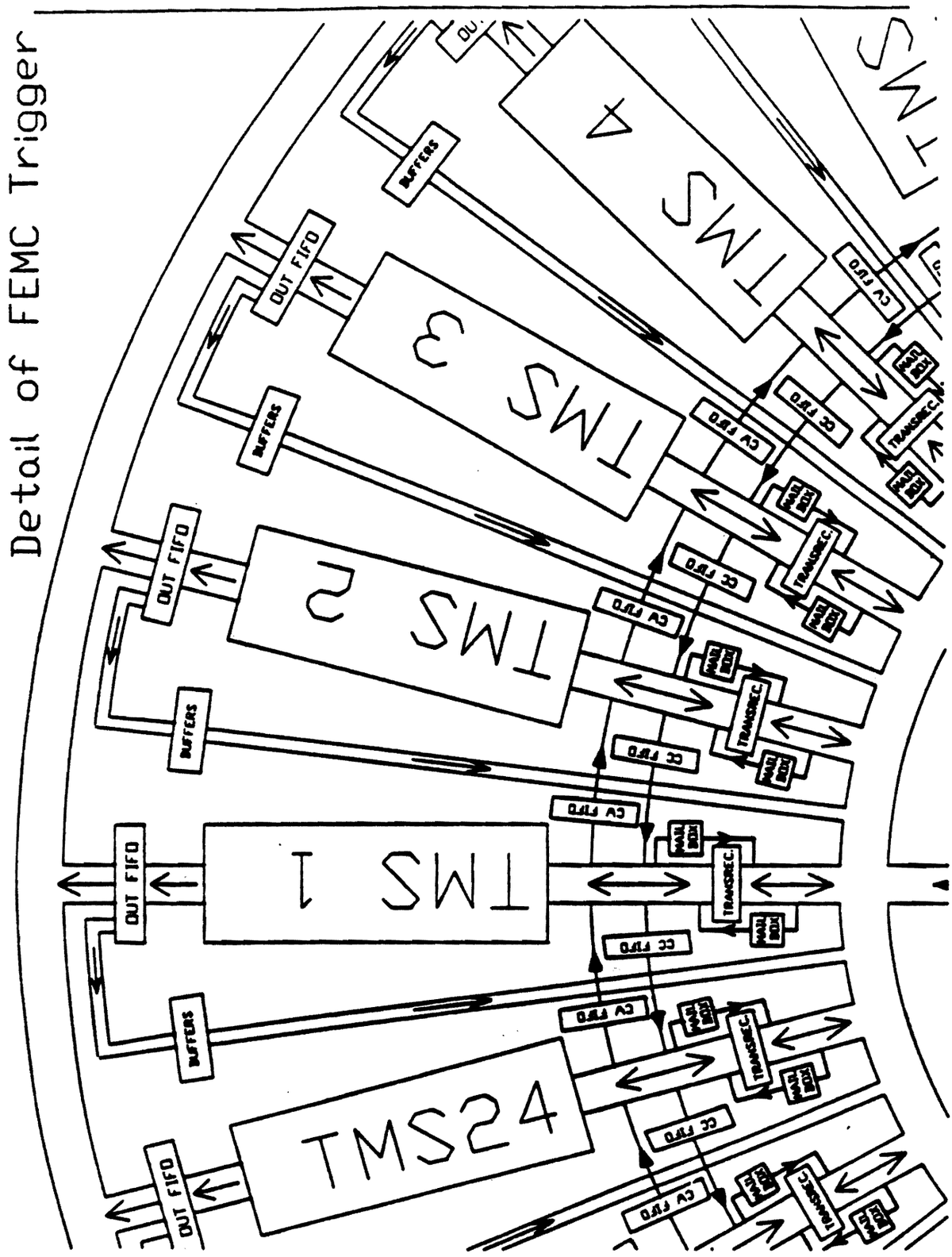


Figure 8: Detail of FEMC system trigger.

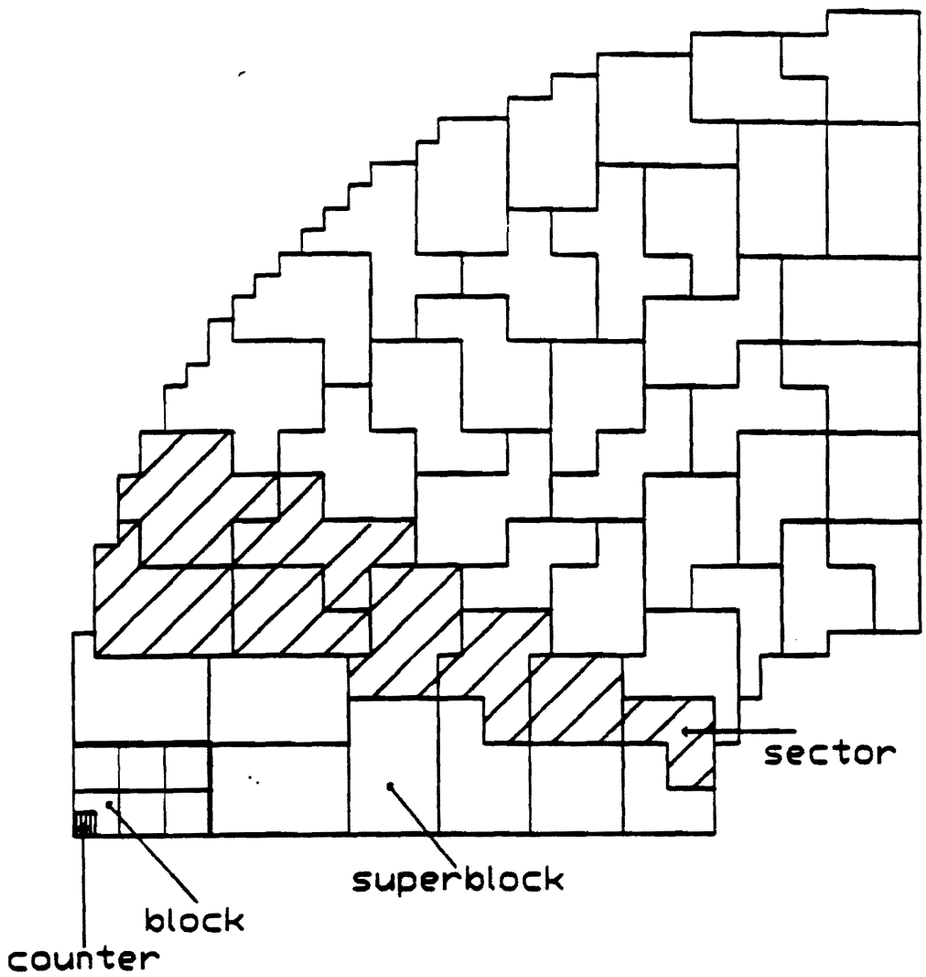


Figure 13: FEMC – DELPHI calorimeter segmentation in Quadrant, Sector, Superblock, Block, Counter.

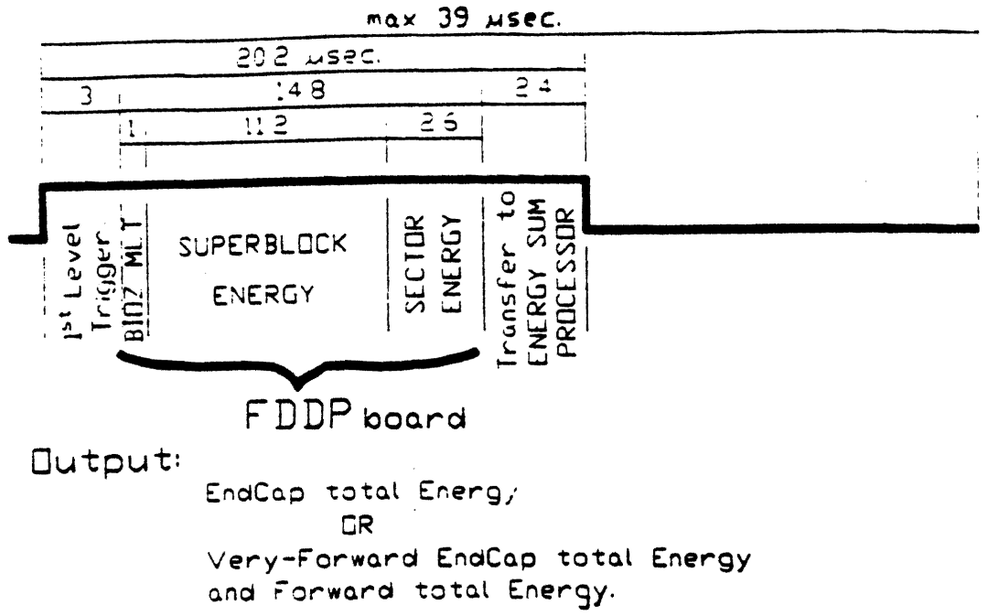


Figure 14: Typical timing diagram of FEMC second level trigger

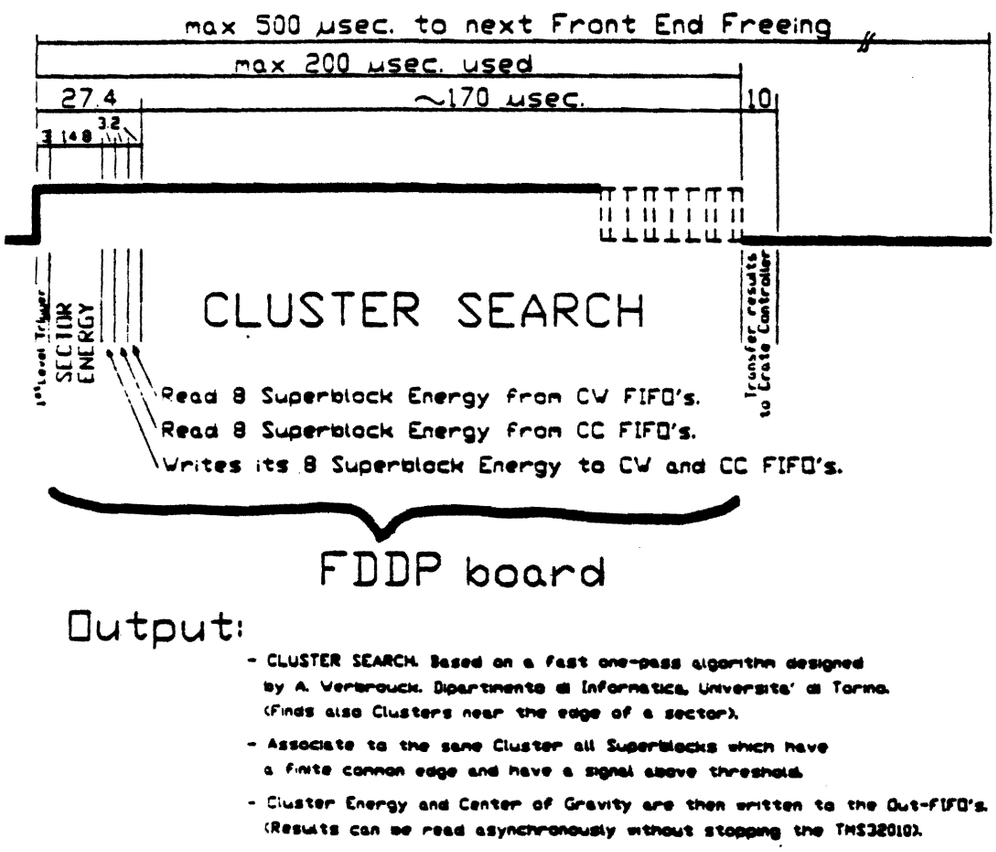


Figure 15: Typical timing diagram of FEMC third level trigger.

9.4.1 Development and test.

Three VME version FDDP's with a total of 6 Logical Units were fully tested with constant input levels by March 27th 1987. The first dynamic test was then made on the complete electronic chain: phototriodes, charge preamplifiers, shapers analog adders and FDDP. An electronic test – in signal of about 40 mVolt was used to simulate the signal generated by the beam. This signal corresponding to 1 Gev electrons gives typically 5 digital counts.

On April 30th a test was made at CERN on the electronic chain: phototriods, charge preamplifiers, shapers, analog adders and FDDP. The results were compared with data obtained by the conventional data acquisition chain (Le Croy Camac modules ADC converters) and full agreement was found.

9.4.2 FEMC – DELPHI trigger monitoring facilities.

To monitor the data flow on the 6 Logical Units, a monitoring routine was developped which displays, upon request, the status of each Logical Unit and of the associated TMS32010. A typical display is shown in table 19 . Rows 1 to 14 contains the STS bits for the last three display requests, line 15 is the actual CMD register content, the last rows show the TMS32010 registers. To call the monitoring routine, TMS32010 user application program must have breakpoints in crucial points, significant to monitor the flow of the program. One step causes execution of the program to the next breakpoint, "n" steps causes execution through n breakpoints (see also Appendix B2 and C3).

Table 19: Screen page of FEMC monitoring program.

		UNIT1 new/old	UNIT2 new/old	UNIT3 new/old	UNIT4 new/old	UNIT5 new/old	UNIT6 new/old
	F1OR OutF ready	001	010	000	100	000	001
L	F1IR OutF full	000	000	000	000	000	000
O	F2OR cwff ready	100	010	010	001	110	000
G	F2IR cwff full	000	000	000	000	000	000
I	F3OR ccff ready	000	111	011	100	110	100
C	F3IR ccff full	000	000	000	000	000	000
A	F4OR ajccw ready	001	010	100	000	111	011
L	F4IR ajccw full	000	000	000	000	000	000
	F5OR ajcw ready	011	000	001	010	100	001
U	F5IR ajcw full	010	010	010	010	010	010
N	REOC rst TMS	000	000	000	000	000	000
I	LOTR lost trig.	000	001	010	100	110	010
T	I68IRQ from VME	010	110	000	111	011	000
	ABOC Abort occ.	011	101	000	101	101	101
	CMD reg.	00	84	7E	55	FE	E4
T	Acc. high	547E	0000	0324	4530	E56F	0E42
M	Acc. low	59AB	6729	B34A	239B	58BA	46BA
S	STATUS REG.	0000	0000	0000	0000	0000	0000
3	AUX REG.0	040	006	050	080	000	00A
2	AUX REG.1	007	106	070	053	29B	06F
0	Prog. Counter	030	073	0B8	012	050	03A

M = Mon., T = next Test, C = Cont., N = number steps, Any key = One step

9.5 Application example to straw tubes.

The description of the proposal of the second level trigger based on straw tube signals in the FERMI – LAB experiment E760 is described elsewhere [13].

The FDDP in this case will be interfaced to an existing acquisition conversion system (Le Croy CAMAC modules FERA 4300B) with a simple ECL – TTL interface in place of the ANIN board as shown in Fig. 16

The complete system layout, that reduces the trigger rate by calculating in about 40 usec the total charge and charge partitioning of the straw tube signals above threshold, is shown in Fig. 17

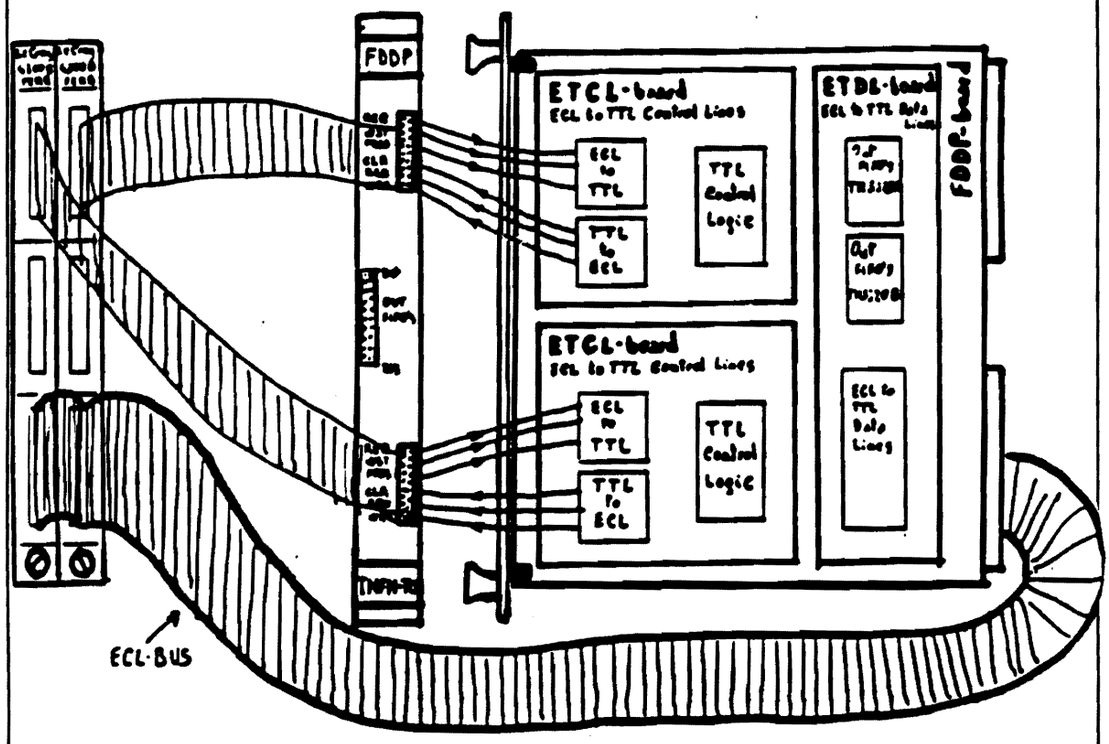


Figure 16: Example of ANIN board that interfaces data from FERA (Le Croy) unit

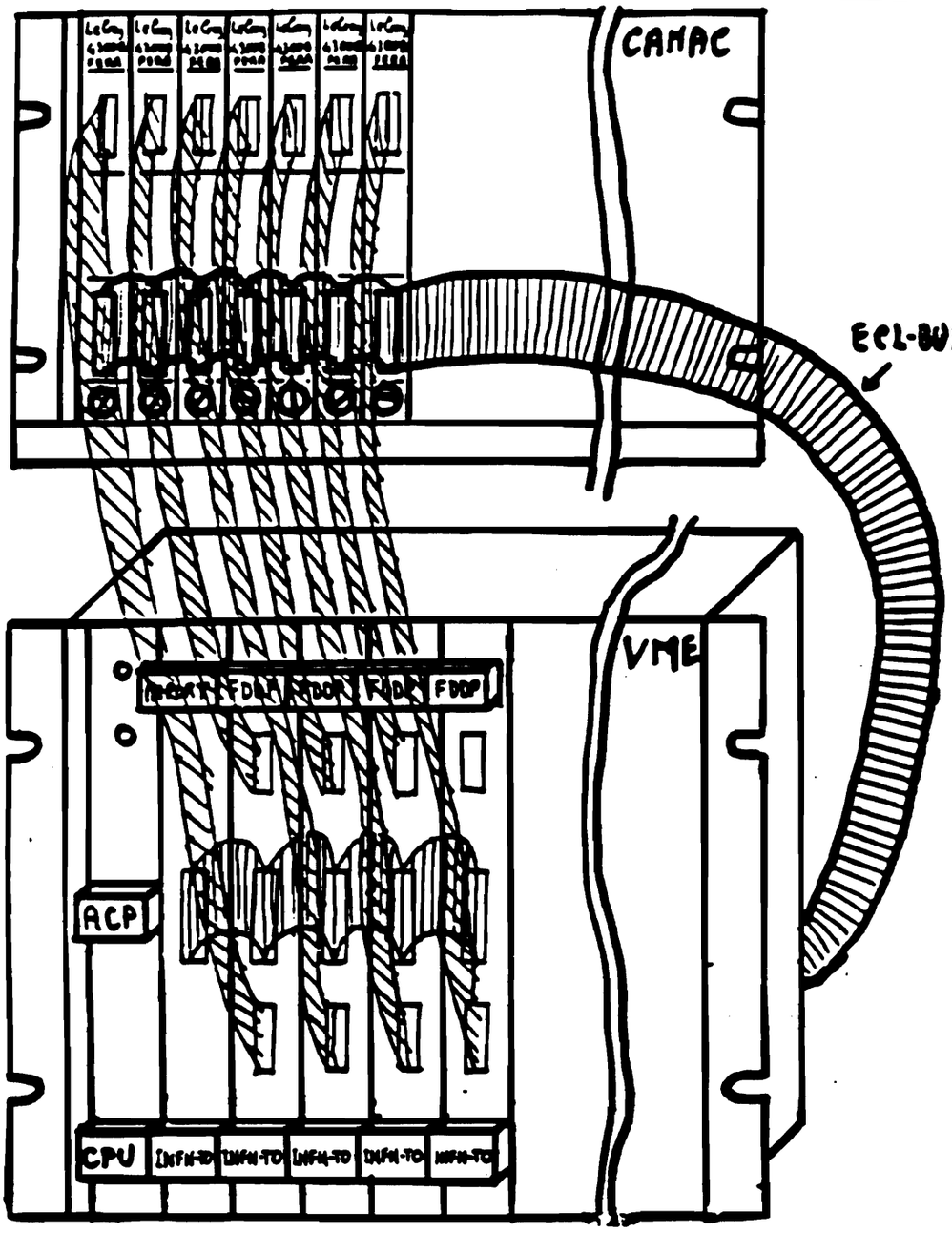


Figure 17: Example of FDDP processing data from straw – tubes

9.6 Application examples to drift chambers

A rapid pulse shape analysis of drift chamber signals is needed for many trigger applications.

The accurate reproduction of the signal shape in digital form by Flash ADC yields good time and amplitude resolution and allows using digital processing algorithms in place of hardware analog functions.

The enormous amount of data produced in a large system can effectively be reduced on line by an FDDP parallel system that effects zero suppression and shape analysis.

Without loss of relevant information (signal time and charge), the data are compressed within the tight time constraints with a processing time usually obtained only with analog readout.

9.6.1 Example of hit center of gravity and total energy calculation

In order to have a precise idea of the time needed to calculate the center of gravity and total energy of a track in a drift chamber, an example is given for a simple request of recognizing an isolated track.

Let us suppose a data acquisition and conversion system that for each wire or group of wires, samples and converts the analog input 150 times per event (the digital sampling rate, of no importance for the digital algorithm, can range from 10 nsec to hundreds of nsec, See Fig. 19)

that the maximum time duration of a pulse (also called: hit, cluster or track) is of maximum 64 consecutive samples above threshold.

The digital algorithm task for each wire is then (see Fig. 18):

- find the cluster
- calculate its energy
- calculate its Center of Gravity

This algorithm, coded in TMS32010 assembly language as shown in Tab. 20, calculates the above parameters for one cluster in 250 usec.

Further and similar analysis can yield:

- coordinates from cluster analysis
- time and amplitude calibration
- individual signal correction (pedestal and slope)
- double track separation, by rapid algorithms based on signal derivation (see Fig. 20).

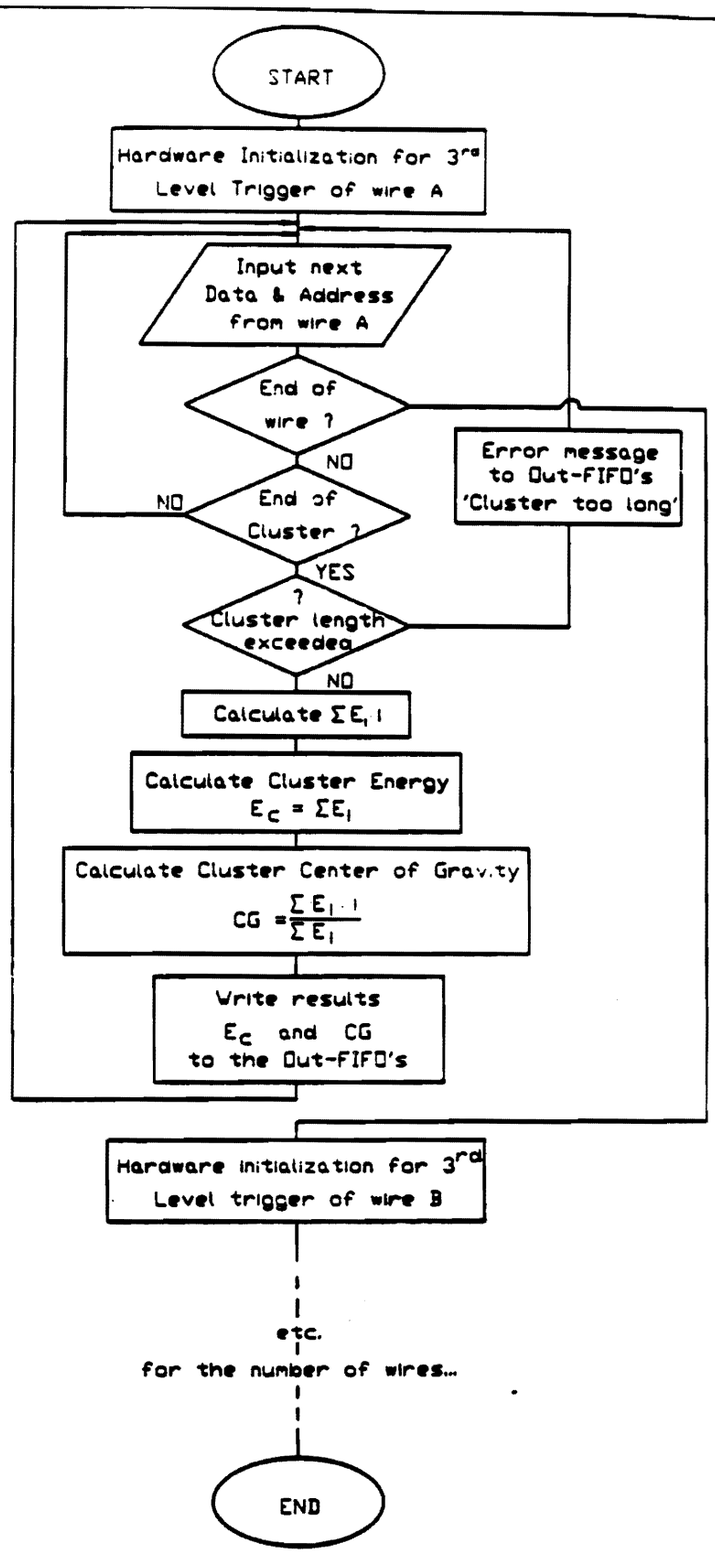


Figure 18: Algorithm flowchart of hit center of gravity and total Energy of a digitized drift chamber pulse.

Table 20: TMS32010 assembly routine for analysis of a track in a drift chamber

```
DRIFT        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    01:54:34  08-28-87
                                                                   PAGE 0001

0001                *
0002                * AUTHOR: D. CROSETTO
0003                * DATE:   July 1st 1987
0004                *
0005                * TMS32010 track analysis program in a drift chamber
0006                *
0007                       IDT     'DRIFT'
0008                       DEF     DRIFT
0009                *
0010          0000  ADDM   EQU     0          Address memory
0011          003D  TEMP1  EQU     ADDM+61    Variable
0012          003E  MAXCL  EQU     ADDM+62    Maximum cluster length
0013          003F  CLUSEP EQU     ADDM+63    Cluster separation
0014          0040  DATM   EQU     ADDM+64    Data memory
0015          007C  FRACU  EQU     DATM+60    Fractional accuracy of division
0016          007D  ENGWIA  EQU     DATM+61   Energy cluster of wirwe A
0017          007E  NUMERA EQU     DATM+62    Index multiply energy sums
0018          007F  CGCLU  EQU     DATM+63    Cluster center of gravity
0019 0000                  PSEG
0020                *
0021                *  Initialization
0022                *
0023 0000                  AORG    0
0024                *
0025 0000 7F8A     DRIFT   ROVM               Reset overflow mode
0026 0001 6E00             LDPK    0          Data memory pointer (0 - 127)
0027 0002 7140             LARK    1,DATM     AR1=64 (Data memory pointer)
0028 0003 7000             LARK    0,ADDM     AR0=0 (Address memory pointer)
0029 0004 6880             LARP    0          Current aux reg. = AR0
0030 0005 7E03             LACK    3
0031 0006 503F             SACL    CLUSEP     Cluster separation = 3
0032 0007 7E3C             LACK    60
0033 0008 503E             SACL    MAXCL      Max cluster lenght = 60
0034 0009 7E02             LACK    2
0035 000A 507C             SACL    FRACU      Fractional accuracy of division
0036                *
0037                *
0038                * here starts calculation of energy for the second level tri
0039                *
0040                *
0041 000B F600     TRIG2   BIOZ    START      Wait for trigger
     000C 000F
0042 000D F900             B       TRIG2
     000E 000B
0043                *
0044 000F 43A8     START   IN      *+,3       Fetch energy of wire A
0045 0010 44A8             IN      *+,4       Fetch energy of wire B
0046 0011 45A8             IN      *+,5       Fetch energy of wire C
0047 0012 7F89             ZAC
0048 0013 6198     LOOP1   ADDS    *-         Calculate total energy
0049 0014 F400             BANZ    LOOP1
     0015 0013
0050                *
0051 0016 503D             SACL    TEMP1      Save total energy
0052                *        .
0053                *        .                 Compare total energy with threshold
0054                *        .                    "     "     "     "     "
```

```
DRIFT        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    01:54:34  08-28-87
                                                                   PAGE 0002

0055                *       OUT     TEMP1,1    Write result to Out-FIFO's
0056                *
0057                *
0058 0017 4088     TRIG3   IN      *,0        Load command register of Logical Uni
0059 0018 7E01             LACK    1          Modify command register
0060 0019 7A88             OR      *          CMD bit0 = 1
0061 001A 5088             SACL    *
0062 001B 4888             OUT     *,0        Store command register
0063 001C 4B88             OUT     *,3        Reset counters A-B-C-D
0064 001D 4888             OUT     *,0        Increment & write Add. memory A-B-C-
0065 001E 4A88             OUT     *,2        Reset Pointers A-B-C-D
0066                *
0067                * here starts reading data from the wire A
0068                *
0069 001F 4081     WIREA   IN      *,0,1      Fetch dummy Addr. memory
0070 0020 4180             IN      *,1,0       "    "    Data  memory
0071 0021 40A1             IN      *+,0,1     Fetch first Addr. memory to (AR0)
0072 0022 41A0             IN      *+,1,0      "    "    Data memory to (AR1)
0073 0023 4091     LOOP2   IN      *-,0,1      "    next Addr. memory to (AR0+1)
0074 0024 41A0             IN      *+,1,0      "     "   Data memory to (AR1+1)
0075 0025 66A0             ZALS    *+,0       Load previous Addr. memory to ACC.
0076 0026 FF00             BZ      NXTWIR     Check if end of wire
     0027 005B
0077 0028 63A0             SUBS    *+,0       Subtract next address memory
0078 0029 613F             ADDS    CLUSEP     Check cluster separation
0079 002A FC00             BGZ     LOOP2      If not end of cluster fetch next inp
     002B 0023
0080 002C 6891             MAR     *-,1       Pointers to last input
0081 002D 6890             MAR     *-,0
0082 002E 303D             SAR     AR0,TEMP1  Save pointer of last input
0083 002F 663D             ZALS    TEMP1      Check if maximum cluster length
0084 0030 633E             SUBS    MAXCL      exceeded
0085 0031 FC00             BGZ     ERROR      If yes, go to error
     0032 005A
0086 0033 7F89             ZAC
0087 0034 6A91             LT      *-,1       Load last Addr. memory to "T" reg.
0088 0035 6D90             MPY     *-,0       Multiply with last Data memory
0089 0036 6C91     LOOP3   LTA     *-,1       Load previous Addr. mem. to "T" reg
0090                *                          P+ACC > ACC
0091 0037 6D90             MPY     *-,0       T x (AR1) > P
0092 0038 F400             BANZ    LOOP3      Check if end of data
     0039 0036
0093 003A 507E             SACL    NUMERA     Save index mult. by energy sums
0094 003B 393D             LAR     AR1,TEMP1  Load pointer of last input
0095 003C 7F89             ZAC
0096 003D 6198     LOOP4   ADDS    *-         Calculate energy
0097 003E F400             BANZ    LOOP4      (sums Data memory)
     003F 003D
0098 0040 507D             SACL    ENGWIA     Save energy wire A
0099 0041 497D             OUT     ENGWIA,1   Out results to Out-FIFO's
0100 0042 6A7E     DIV     LT      NUMERA     Get sign of quotient
0101 0043 6D7D             MPY     ENGWIA
0102 0044 7F8E             PAC
0103 0045 583D             SACH    TEMP1      Save sign of quotient
0104 0046 207D             LAC     ENGWIA
0105 0047 7F88             ABS
0106 0048 507D             SACL    ENGWIA     Make denominator positive
```

```
DRIFT        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    01:54:34  08-28-87
                                                                  PAGE 0003

0107 0049 7E0F          LACK    15
0108 004A 007C          ADD     FRACU
0109 004B 507C          SACL    FRACU     Compute loop count
0110 004C 207E          LAC     NUMERA    Align numerator
0111 004D 7F88          ABS               Make numerator positive
0112 004E 387C          LAR     0,FRACU
0113 004F 647D  LOOP5   SUBC    ENGWIA    16+FRACU cycle divide loop
0114 0050 F400          BANZ    LOOP5
     0051 004F
0115 0052 507F          SACL    CGCLU
0116 0053 203D          LAC     TEMP1
0117 0054 FD00          BGEZ    CONT1     Done if sign is positive
     0055 0059
0118 0056 7F89          ZAC
0119 0057 107F          SUB     CGCLU
0120 0058 507F          SACL    CGCLU     Negative quotient if negative
0121 0059 497F  CONT1   OUT     CGCLU,1   Write result to Out-FIFO's
0122            *
0123            *
0124 005A 7F80  ERROR   NOP     *
0125            *       .
0126            *       .
0127            *       .
0128 005B 7F80  NXTWIR  NOP     *
0129            *       .
0130            *       .
0131            *
0132            *
0133                    END
NO ERRORS, NO WARNINGS
```

```
NO$IDT       32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    01:54:40  08-28-87
                                                                  PAGE 0004

0001 0000
0002 0000
0003 0000
0004 0000
0005 0000
0006 0000
0007 0000
0008 0000
NO ERRORS, NO WARNINGS
```

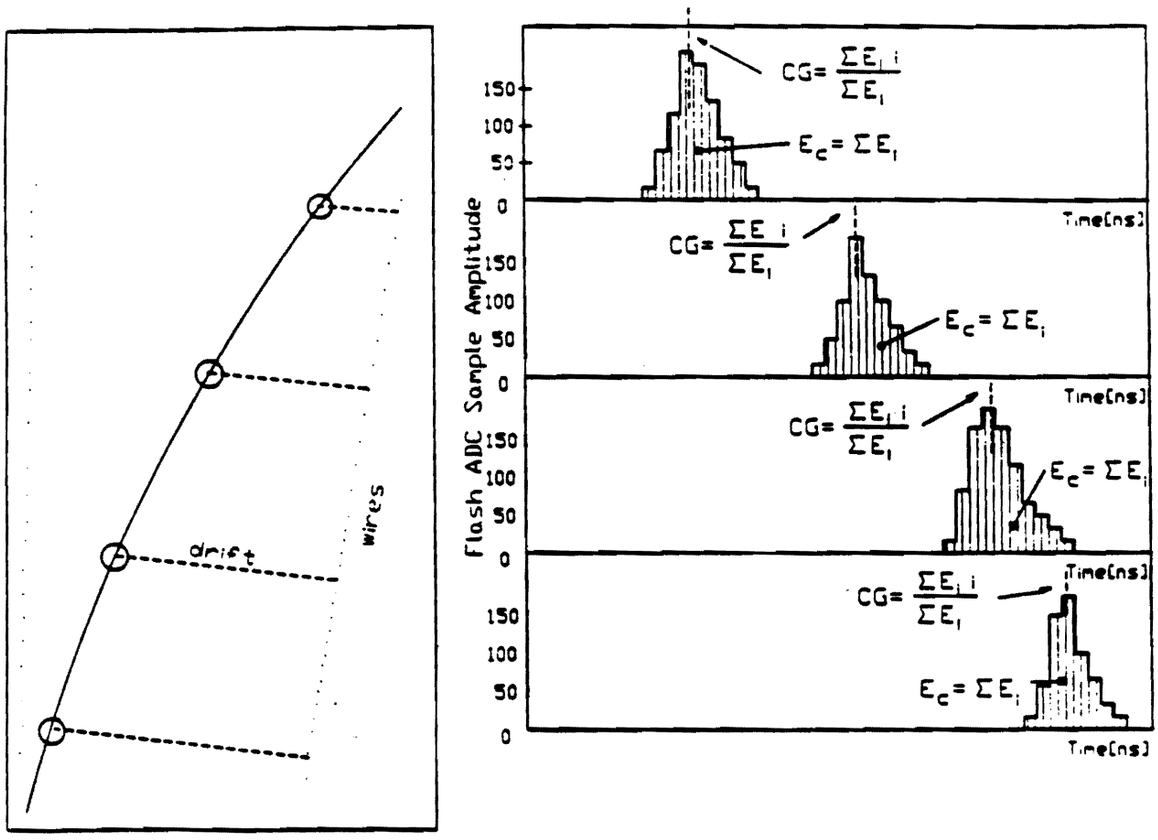


Figure 19: Analysis of a track in a drift chamber.

9.6.2 Guidelines to configure a FDDP system for drift chamber analysis.

In configuring a system of parallel processors based on Logical Units (FDDP boards) and in designing the personalized ANIN board (a very general indication of an ANIN2 board for drift chambers is given in Fig. 9), the following considerations must be made:

1. in designing the personalized ANIN board:

    a. analog signals should be optically isolated from digital signals

    b. drift time and Flash ADC sampling rate should be optimized (memory size per channel)

    c. Flash ADC resolution could be optimized by allowing also non linear gain to extend dynamic range

2. in configuring the parallel system processor, optimization can be done on:

    a. typical number of samplings per hit

    b. typical number of hits/channel

    c. time for algorithms needed for the:

    - linearization of inputs
    - detection of hits
    - calculation of hit center of gravity
    - calculation of amplitude
    - double tracks separation (derivation and zero – cross analysis).

3.

    The optimization must also take into account the maximum time available for data reduction. The latter will essentially determine how many channels a Logical Unit can process and therefore the level of parallelism needed.

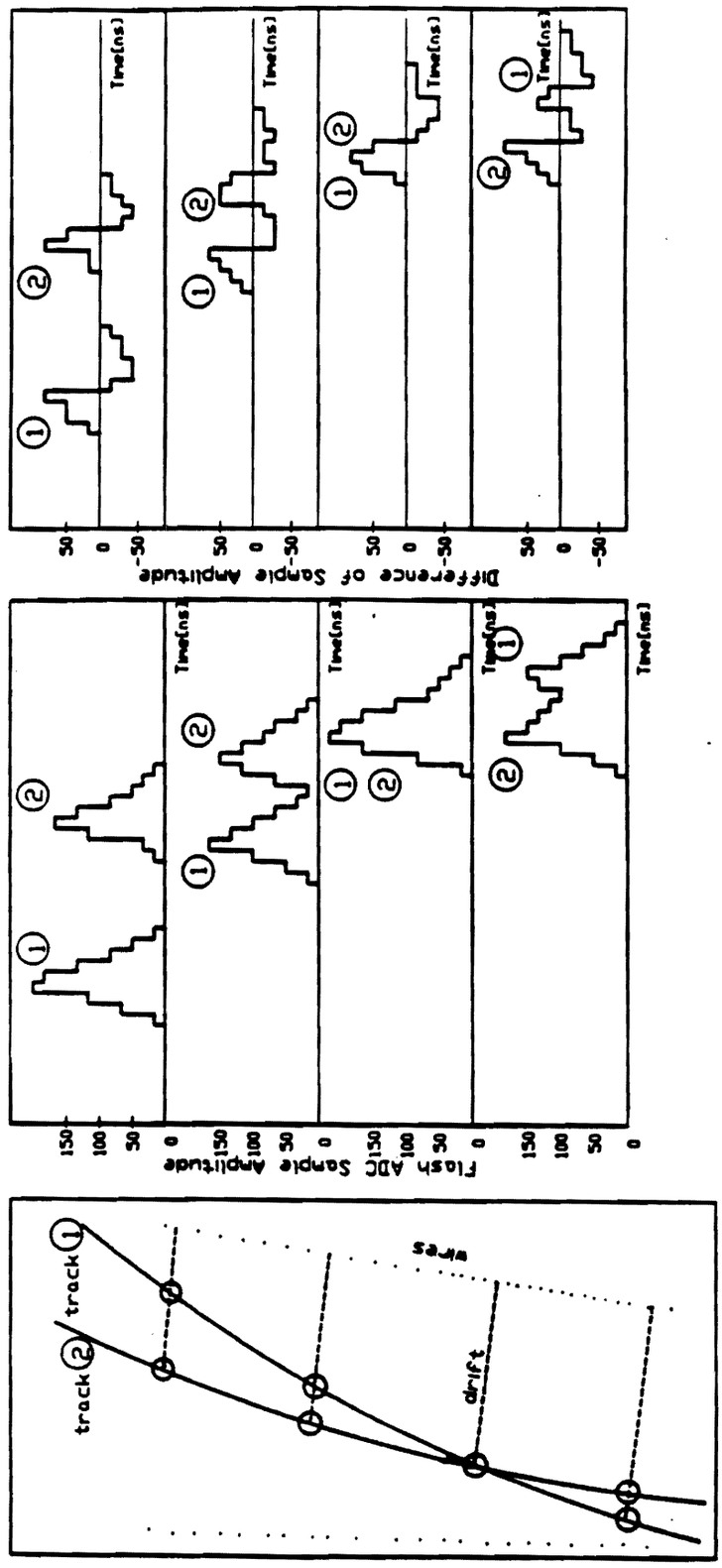


Figure 20: Track resolution using derivative on signals

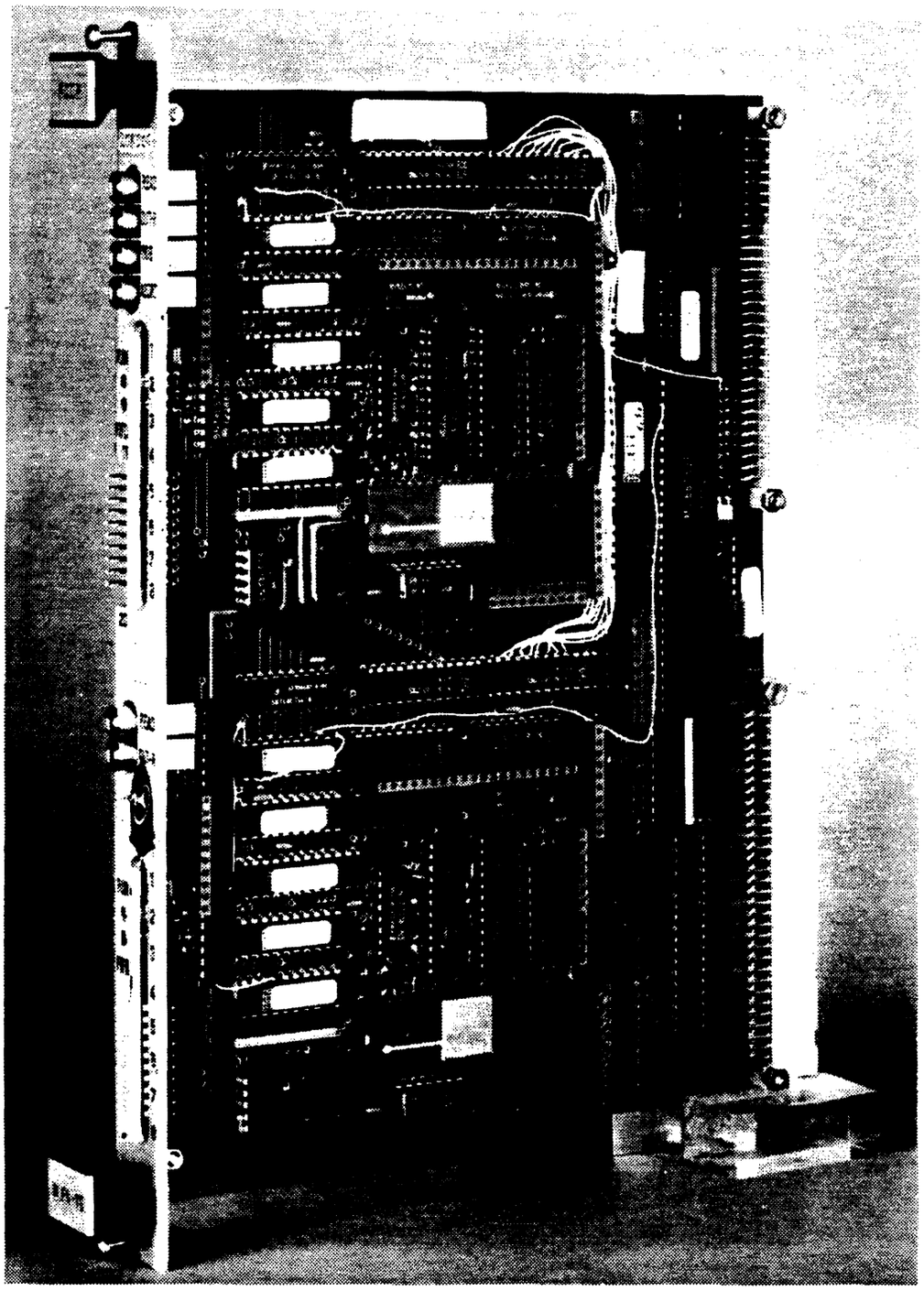


Figure 21: FDDP mother board

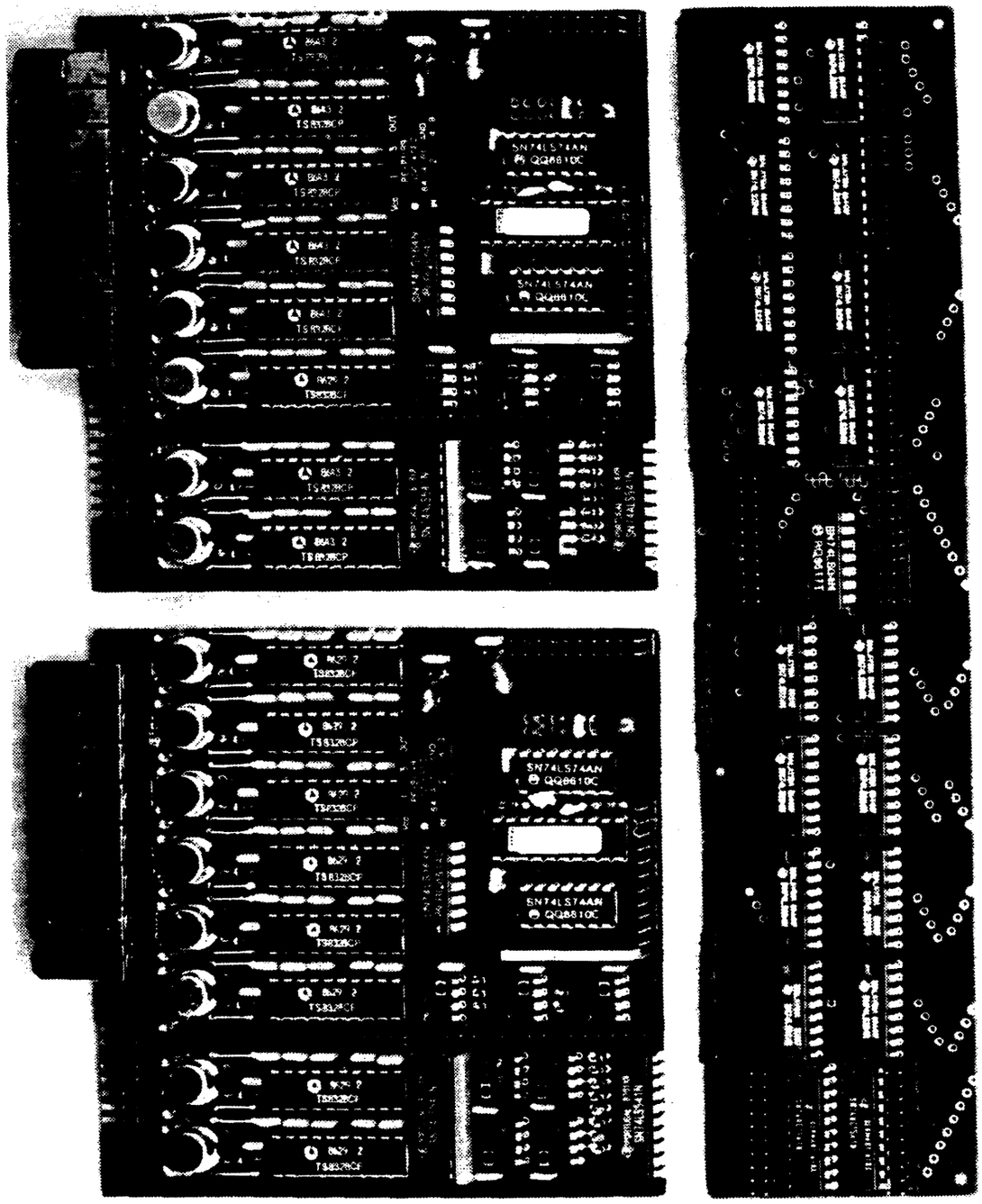


Figure 22: ANIN1 and FIFO piggy back boards

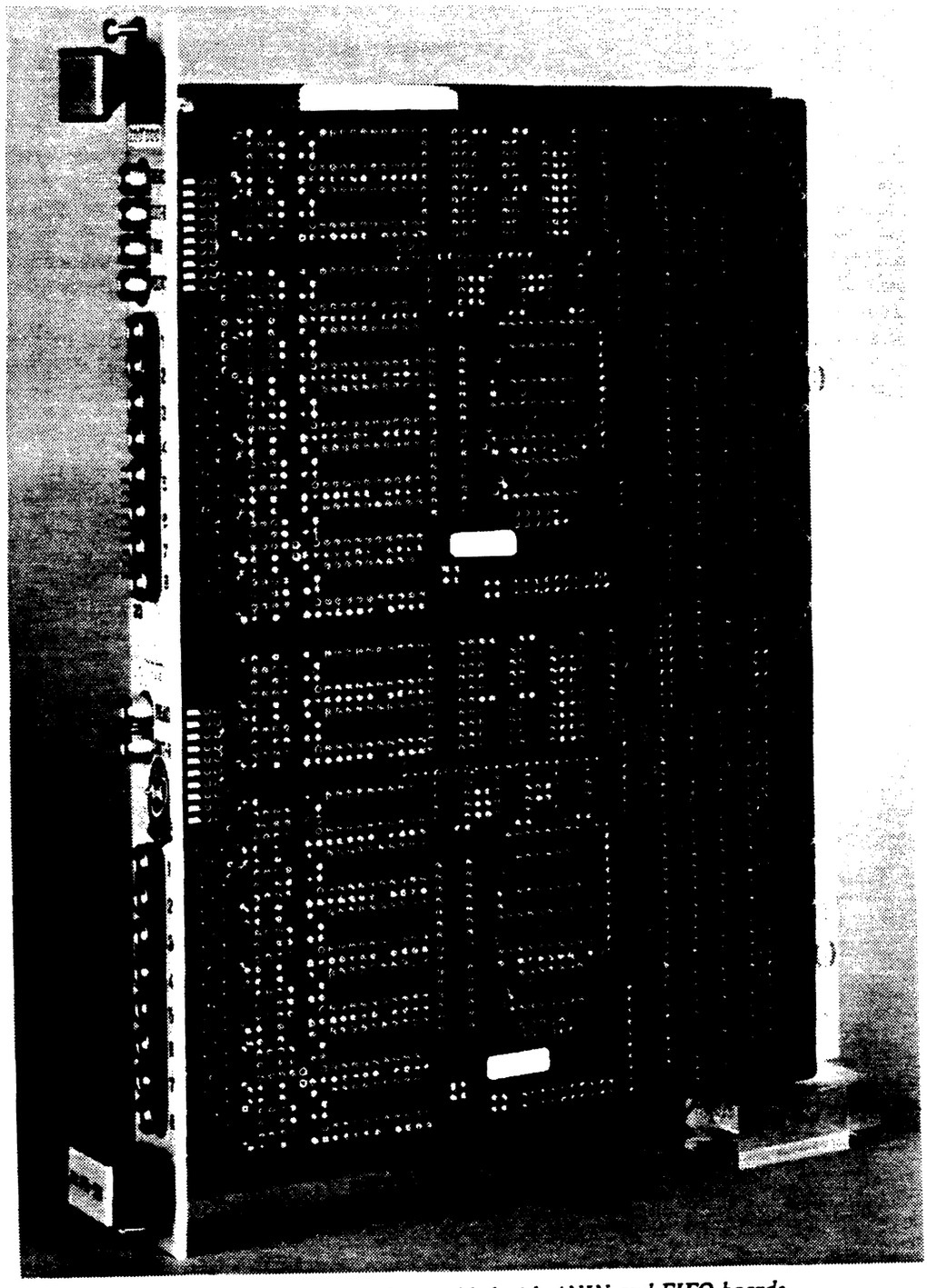


Figure 23: FDDP board assembled with ANIN and FIFO boards.

Acknowledgments

I would like to thank prof. G. Rinaudo and prof. A. Werbrouck. During all the developmment phase of these modules, there has been a very close cooperation, continuous support, elaborate criticism and many fruitfull suggestions from both of them. Starting from a basic concept of intercommunication between DSP of prof. G. Rinaudo, most of the architecture and features of the modular system has been defined after evenings of discussion with prof. A. Werbrouck. The relatively short realization period of the first 6 logical units has been possible thanks to the partecipation of prof. G. Rinaudo in debugging and testing the prototipe and thanks to J. Allaby, P. Gavillet and H. Muller for giving us the possibility to use the facilities of the DELPHI FASTBUS laboratory.
It is also a pleasure to aknowledge the contributions of all persons who have worked on this project, in particular for the drawings of the six-layer Printed-Circuit and Shao Bei Bei who has helped in testing in the last two months.

APPENDIX A

ANIN1 BOARD

A.1 Analog Input board for FEMC – DELPHI trigger.

The ANIN – board provides electrical isolation between the Analog Circuits and Digital Circuit by means of the"HPCL – 2630" dual TTL compatible optocouplers. Attention must be paid when connecting I/O cables to the connectors on the board. It is important to use the appropriate ground signal for each connector and maintain the power supply isolation as described in the chapters "Front Panel Connectors" and "Back – Plane Connectors".

Simultaneously 8 different Flash ADC convert 8 analog input signals. A strobe signal starts the conversion (Trigger falling edge pulse delayed by 30 nsec). A two pulse train propagates the converted value to the D – Type FF inside the "TS8328" Flash ADC (see fig. 25). The three – state output D – Type Registers are connected all together and, through an output enable at each Flash ADC, the data are placed on the bus one at a time.

The digital values are then fetched by the TMS32010 with N consecutive "IN" instructions.

A.1.1 PL1A/PL8A & PL1B/PL8B – ANALOG INPUTS (ANIN1 board)

Table 21: PL1A/PL8A & PL1B/PL8B Lemo connector ANALOG INPUTS (ANIN1 board).

LEMO	SIGNAL	LEMO	SIGNAL
PL1A	CH1 – TMS32010A	PL1B	CH1 – TMS32010B
PL2A	CH2 – TMS32010A	PL2B	CH2 – TMS32010B
PL3A	CH3 – TMS32010A	PL3B	CH3 – TMS32010B
PL4A	CH4 – TMS32010A	PL4B	CH4 – TMS32010B
PL5A	CH5 – TMS32010A	PL5B	CH5 – TMS32010B
PL6A	CH6 – TMS32010A	PL6B	CH6 – TMS32010B
PL7A	CH7 – TMS32010A	PL7B	CH7 – TMS32010B
PL8A	CH8 – TMS32010A	PL8B	CH8 – TMS32010B

- 2*8 analog inputs to Flash ADC.
- Lemo (see Fig. 24)
- Ground and signals are referred to the Analog Power Supply.

A.1.2 P6H & P7H – ANALOG INPUTS (ANIN1 board)

Table 22: P6H & P7H 20 pin connector ANALOG INPUTS (ANIN1 board)

1	NC	NC	11
2	NC	NC	12
3	CH8	GND	13
4	CH7	GND	14
5	CH6	GND	15
6	CH5	GND	16
7	CH4	GND	17
8	CH3	GND	18
9	CH2	GND	19
10	CH1	GND	20

Ground and signals are refferd to the Analog Power Supply.

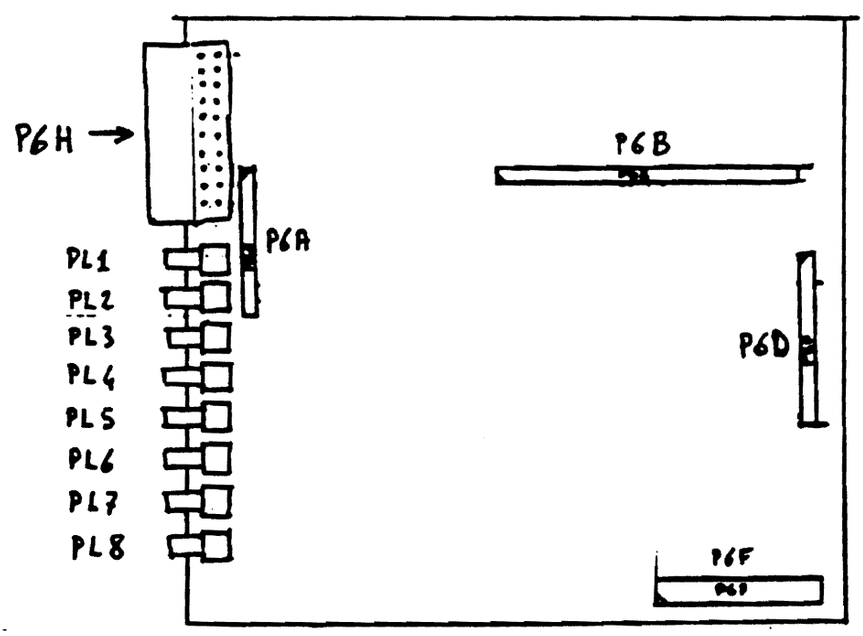


Figure 24: ANIN1 on board connectors layout

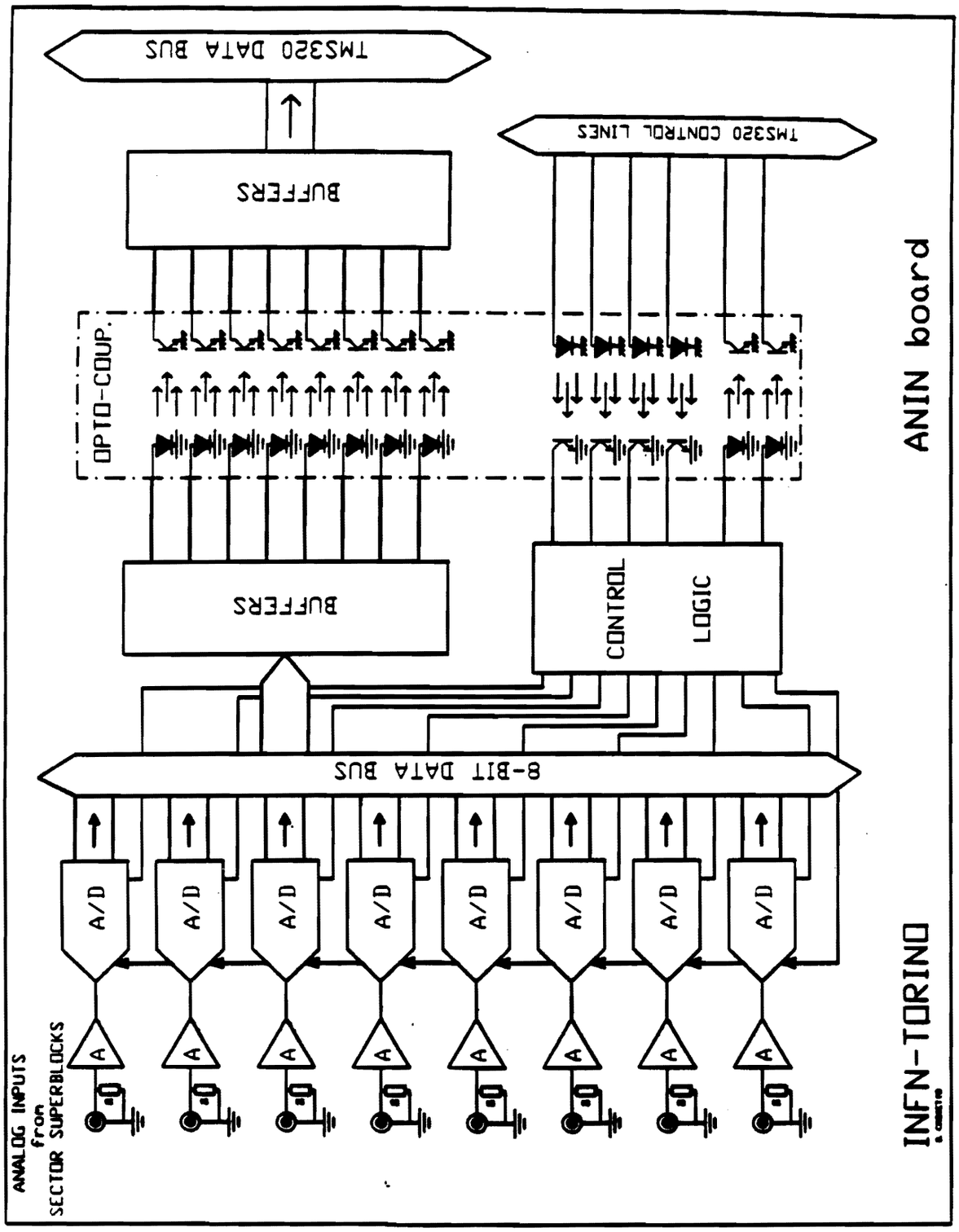


Figure 25: ANIN1 board. Block diagram

APPENDIX B

TMS32010 ASSEMBLY ROUTINES

B.1 TMS32010 software support routine (tracing program).

As described in section 1.1.1 a trace facility of TMS32010 programs is available, which does not require modifications of the user program.

This facility requires configuring the TMS32010 program memory as described in Fig. 26

The trace interrupt routine is listed in Table 24
Instructions at address 8 – 11 are needed to initialize the trace routine and must be loaded on the top of the user application program.
The interrupt tracing routine saves the TMS32010 register context, reads through the Mail – Box the command given by the user, executes it and writes the data described in Table 23 into the Out – FIFO's.
The MC68000 programs which read the data from the Out – FIFO's is described in Appendix C1, the display routine is described in appendix C2. The commands available are listed in table 23; they are given through the keyboard and interpreted by the MC68000 program described in Appendix C2.

Table 23: Screen page of TMS32010 trace program

```
                    *** TMS32010 TRACE UTILITY ***

            HEX         BINARY
Acc.        xxxx.xxxx   xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
P reg.      xxxx.xxxx   xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
T reg.      xxxx        xxxxxxxx.xxxxxxxx          AuxReg0: xxxx AuxReg1: xxxx DP: x
STS reg.    xxxx        xxxxxxxx.xxxxxxxx          ARP: x  OVM: x  OV: x
PC reg.     xxx         xxxx.xxxxxxxx              Stack0: xxx  Stack1: xxx  Stack2: xxx

                        *** Logical Unit Trace ***
STS reg.    xxxx        xxxxxxxx.xxxxxxxx          CMD reg. xx    xxxx.xxxx

M = Mon., T = next Test, C = Cont., N = number of steps, Any key = One step

                    *** TMS32010 TRACE UTILITY ***

            HEX         BINARY
Acc.        xxxx.xxxx   xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
P reg.      xxxx.xxxx   xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx
T reg.      xxxx        xxxxxxxx.xxxxxxxx          AuxReg0: xxxx AuxReg1: xxxx DP: x
STS reg.    xxxx        xxxxxxxx.xxxxxxxx          ARP: x  OVM: x  OV: x
PC reg.     xxx         xxxx.xxxxxxxx              Stack0: xxx  Stack1: xxx  Stack2: xxx

                        *** Logical Unit Trace ***
STS reg.    xxxx        xxxxxxxx.xxxxxxxx          CMD reg. xx    xxxx.xxxx
```

Table 24: TMS32010 Interrupt trace routine.

```
TRACE        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:57:16  08-25-87
                                                                  PAGE 0001

0001                *
0002                * AUTHOR: D. CROSETTO
0003                * DATE:   July 1st 1987
0004                *
0005                * TMS32010 TRACE program for FDDP
0006                *
0007                       IDT     'TRACE'
0008                       DEF     TRACE
0009                *
0010          0000  SAVSTS  EQU     0         TMS32010 STATUS REGISTER
0011          0001  STATUS  EQU     SAVSTS+1  LOGICAL UNIT STATUS REGISTER
0012          0002  TEMP    EQU     STATUS+1  Temporary variable
0013          0003  UNITY   EQU     TEMP+1    Unity variable
0014          0004  NUMSTP  EQU     UNITY+1   Number of steps
0015          0005  SAVACH  EQU     NUMSTP+1  Accumulator high
0016          0006  SAVACL  EQU     SAVACH+1  Accumulator low
0017          0007  SAVAR0  EQU     SAVACL+1  AR0
0018          0008  SAVAR1  EQU     SAVAR0+1  AR1
0019          0009  SAVPRH  EQU     SAVAR1+1  "P" register high
0020          000A  SAVPRL  EQU     SAVPRH+1  "P" register low
0021          000B  SAVTRG  EQU     SAVPRL+1  "T" register
0022          000C  SAVPC   EQU     SAVTRG+1  Program Counter
0023          000D  SAVST0  EQU     SAVPC+1   Stack 0
0024          000E  SAVST1  EQU     SAVST0+1  Stack 1
0025          000F  SAVST2  EQU     SAVST1+1  Stack 2
0026                *
0027 0000                  PSEG
0028 0000                  AORG    >0
0029                *
0030 0000 F900    START    B       MAIN
     0001 0008
0031 0002 F900             B       TRACE
     0003 0700
0032                *
0033 0008                  AORG    >8
0034                *                                                  *****
0035 0008 7F80    MAIN     NOP             Initialize     * Add this   *   *
0036 0009 6E01             LDPK    1       to one         * instructions * E *
0037 000A 7E00             LACK    0       step           * to the     * X *
0038 000B 5004             SACL    NUMSTP                 * user       * A *
0039 000C 7F82             EINT            Enable int.    * program    * M *
0040 000D 4702             IN      TEMP,7  Set 1st int.   * for tracing * P *
0041 000E 7F80             NOP                                         * L *
0042 000F 7F80             NOP                                         * E *
0043 0010 7F80             NOP                                         *   *
0044 0011 7F80             NOP                                         *   *
0045 0012 7F80             NOP                                         * O *
0046 0013 7F80             NOP                                         * F *
0047 0014 7F80             NOP                                         *   *
0048 0015 7F80             NOP                                         * U *
0049 0016 7F80             NOP                                         * S *
0050 0017 7E00    ZERO     LACK    0       Load accumulator with 0     * E *
0051 0018 F900             B       SKIP                                * R *
     0019 001B
0052 001A 7E2E    LVALUE   LACK    46      Load accumulator with 01010101 *   *
0053                *                                                  *   *
0054                *                                                  *   *
```

```
TRACE        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:57:16  08-25-87
                                                                  PAGE 0002

0055                *      Move Accumulator to Auxiliary Register      * P *
0056                *                                                  * R *
0057 001B 500A    SKIP     SACL    10,0    Store AC to internal mem. 10 * O *
0058 001C 380A             LAR     AR0,10  Re-load AR0                 * G *
0059 001D 6880             LARP    0       Load AR0 pointer            * R *
0060                *                                                  * A *
0061                *      Move Auxiliary Register to Accumulator      * M *
0062                *                                                  *   *
0063 001E 300A             SAR     AR0,10  Save AR0                    *****
0064 001F 200A             LAC     10,0    Load into accumulator
0065 0020 FF00             BZ      LVALUE
     0021 001A
0066 0022 F900             B       ZERO
     0023 0017
0067                *
0068 0700                  AORG    >700
0069                *
0070 0700 6E01    TRACE    LDPK    1
0071 0701 7C00             SST     SAVSTS  Save current context
0072 0702 5805             SACH    SAVACH
0073 0703 5006             SACL    SAVACL
0074 0704 3007             SAR     0,SAVAR0
0075 0705 3108             SAR     1,SAVAR1
0076 0706 7E01             LACK    1       Initialize TEMP1 = 1
0077 0707 5003             SACL    UNITY
0078 0708 7F89             ZAC
0079 0709 7F8F             APAC
0080 070A 5809             SACH    SAVPRH
0081 070B 500A             SACL    SAVPRL  Save "P" register
0082 070C 6D03             MPY     UNITY
0083 070D 7F8E             PAC
0084 070E 500B             SACL    SAVTRG  Save "T" register
0085 070F 7F9D             POP
0086 0710 500C             SACL    SAVPC   Save Program Counter
0087 0711 7F9D             POP
0088 0712 500D             SACL    SAVST0  Save Stack 0
0089 0713 7F9D             POP
0090 0714 500E             SACL    SAVST1  Save Stack 1
0091 0715 7F9D             POP
0092 0716 500F             SACL    SAVST2  Save Stack 2
0093 0717 7F9C             PUSH            Restore Stack 2
0094 0718 200E             LAC     SAVST1
0095 0719 7F9C             PUSH            Restore Stack 1
0096 071A 200D             LAC     SAVST0
0097 071B 7F9C             PUSH            Restore        Stack 0
0098 071C 200C             LAC     SAVPC
0099 071D 7F9C             PUSH            Restore Return Address
0100                *
0101 071E 4301    RDSTS    IN      STATUS,3
0102 071F 7E20             LACK    >20     Check interrupting device
0103 0720 5002             SACL    TEMP
0104 0721 2802             LAC     TEMP,8
0105 0722 7901             AND     STATUS
0106 0723 FF00             BZ      ABO68K  Abort, 68000 or trace ?
     0724 0757
0107 0725 2004             LAC     NUMSTP  Check if executing several steps
0108 0726 FE00             BNZ     TRACEC
```

```
TRACE        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:57:16  08-25-87
                                                                  PAGE 0003
     0727 0770
0109              *
0110 0728 4202  MAILB  IN      TEMP,2   Input Commande Code from Mail-Box
0111 0729 2002         LAC     TEMP
0112 072A FA00         BLZ     CONT     Is Continue ?
     072B 0750
0113 072C 5004         SACL    NUMSTP   Is trace several steps ?
0114 072D FE00         BNZ     TRACEC   Is single step ?
     072E 0770
0115              *
0116 072F 4002  STEP   IN      TEMP,0   Command Reg. to Out-FIFO's
0117 0730 4E02         OUT     TEMP,6   It is single step
0118 0731 4E01         OUT     STATUS,6
0119 0732 4E00         OUT     SAVSTS,6 then
0120 0733 4E05         OUT     SAVACH,6
0121 0734 4E06         OUT     SAVACL,6 transfer local parameters
0122 0735 4E07         OUT     SAVAR0,6
0123 0736 4E08         OUT     SAVAR1,6 to the Out-FIFO's
0124 0737 4E09         OUT     SAVPRH,6
0125 0738 4E0A         OUT     SAVPRL,6
0126 0739 4E0B         OUT     SAVTRG,6
0127 073A 4E0C         OUT     SAVPC,6
0128 073B 4E0D         OUT     SAVST0,6
0129 073C 4E0E         OUT     SAVST1,6
0130 073D 4E0F         OUT     SAVST2,6
0131 073E 7E80         LACK    >80
0132 073F 5002         SACL    TEMP
0133 0740 2802         LAC     TEMP,8
0134 0741 5002         SACL    TEMP
0135 0742 4E02         OUT     TEMP,6   end of transfer message
0136              *
0137 0743 F600  USER   BIOZ    EXIT     Wait for user response
     0744 0747
0138 0745 F900         B       USER     command selection
     0746 0743
0139              *
0140 0747 6E01  EXIT   LDPK    1
0141 0748 6505         ZALH    SAVACH   Restore Accumulator High
0142 0749 6106         ADDS    SAVACL   Restore accumulator low
0143 074A 3807         LAR     0,SAVAR0 Restore AR0
0144 074B 3908         LAR     1,SAVAR1 Restore AR1
0145 074C 7B00         LST     SAVSTS   Restore Status
0146 074D 7F82         EINT             Enable subsequent interrupts
0147 074E 4702         IN      TEMP,PA7 Generate a delaied interrupt
0148 074F 7F8D         RET
0149              *
0150 0750 6E01  CONT   LDPK    1
0151 0751 6505         ZALH    SAVACH   Restore Accumulator High
0152 0752 6106         ADDS    SAVACL   Restore Accumulator Low
0153 0753 3807         LAR     0,SAVAR0 Restore AR0
0154 0754 3908         LAR     1,SAVAR1 Restore AR1
0155 0755 7B00         LST     SAVSTS   Restore Status
0156 0756 7F8D         RET
0157              *
0158 0757 7E40  ABO68K LACK    >40      Check interrupting device
0159 0758 5002         SACL    TEMP
0160 0759 2802         LAC     TEMP,8
```

```
TRACE        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:57:16  08-25-87
                                                                  PAGE 0004
0161 075A 7901         AND     STATUS   is from 68000 ?
0162 075B FF00         BZ      ABORT
     075C 0763
0163 075D 7F80         NOP
0164 075E 7F80         NOP              Interrupt service routine
0165 075F 7F80         NOP
0166 0760 7F80         NOP              for 68000
0167 0761 F900         B       CONT
     0762 0750
0168              *
0169 0763 7E80  ABORT  LACK    >80
0170 0764 5002         SACL    TEMP     Check interrupting device
0171 0765 2802         LAC     TEMP,8
0172 0766 7901         AND     STATUS   is from ABORT ?
0173 0767 FF00         BZ      CLRINT   if not, clear interrupt.
     0768 076E
0174 0769 7F80         NOP
0175 076A 7F80         NOP              Interrupt service routine
0176 076B 7F80         NOP
0177 076C 7F80         NOP              for ABORT
0178 076D 7F80         NOP
0179 076E F900  CLRINT B       CONT
     076F 0750
0180              *
0181 0770 1003  TRACEC SUB     UNITY    Decrement trace counter
0182 0771 5004         SACL    NUMSTP
0183 0772 FE00         BNZ     EXIT
     0773 0747
0184 0774 F900         B       STEP
     0775 072F
0185              *
0186              *
0187                           END
NO ERRORS, NO WARNINGS
```

```
NO$IDT       32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:57:24  08-25-87
                                                                  PAGE 0005
0001 0000
0002 0000
0003 0000
0004 0000
0005 0000
0006 0000
0007 0000
0008 0000
NO ERRORS, NO WARNINGS
```

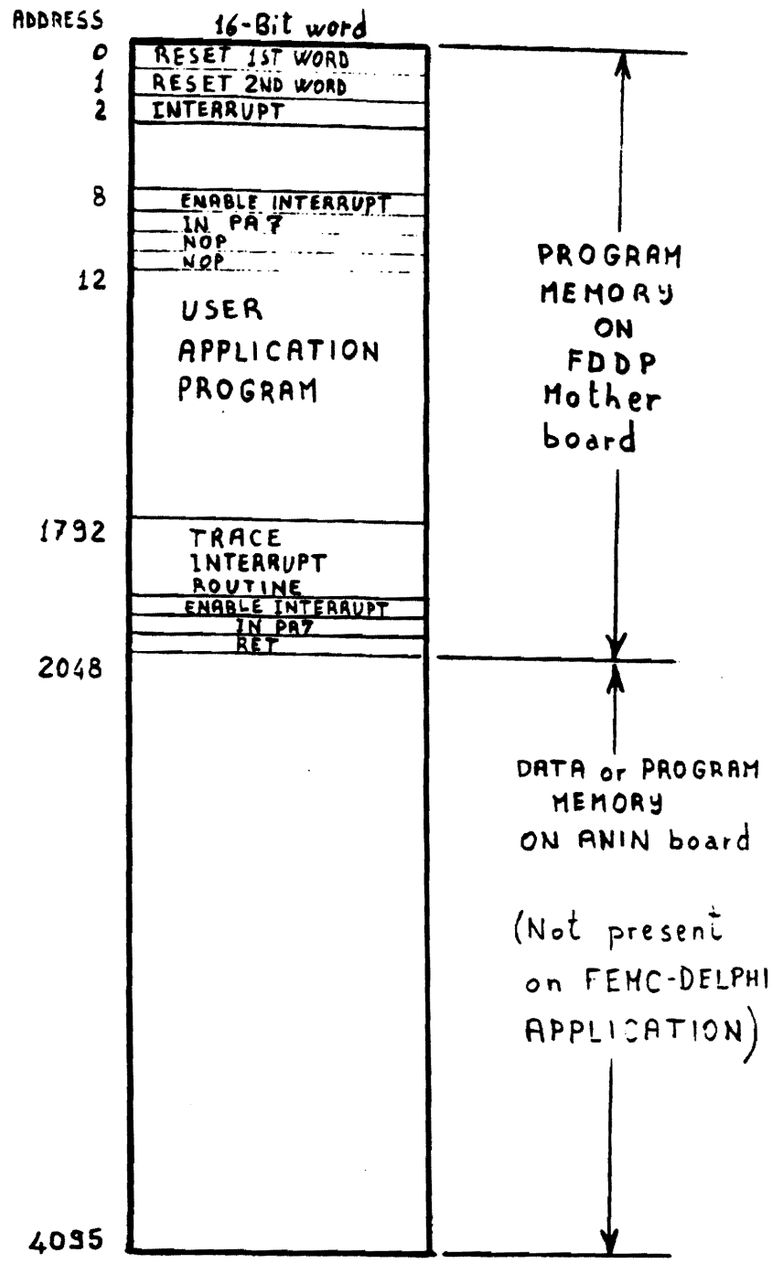


Figure 26: Logical Unit Program Memory Map

B.2 TMS32010 software support routine (monitoring program).

The monitoring routines are intended to give a global view of the entire system and monitor all the active Logical Units at the same time.

This monitoring program requires insertion of break points in the crucial points of the TMS32010 user program.

At the break point, contents of the Status Register, Command Register and TMS32010 context of the active Logical Units are displayed on the sreen (see Table 19).

The interrupt monitoring routine is listed in Table 25 The routine saves the TMS32010 register context, reads through the MAIL-BOX the command given by the user, executes it and writes the data described in Table 19 into the Out-FIFO's. The MC68000 programs which read the data from the Out-FIFO's is described in Appendix C1, the display routine is described in appendix C3. The commands available are listed in table 19; they are given through the keyboard and interpreted by the MC68000 program described in Appendix C3.

Table 25: TMS32010 Interrupt monitoring routine.

```
MONIT        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:56:51  08-25-87
                                                                   PAGE 0001

0001                *
0002                * AUTHOR: D. CROSETTO
0003                * DATE:   July 1st 1987
0004                *
0005                * TMS32010 LOGICAL UNIT monitoring program for FDDP
0006                *
0007                       IDT     'MONIT'
0008                       DEF     MONIT
0009                *
0010        0000    SAVSTS  EQU     0         TMS32010 STATUS REGISTER
0011        0001    STATUS  EQU     SAVSTS+1  LOGICAL UNIT STATUS REGISTER
0012        0002    TEMP    EQU     STATUS+1  Temporary variable
0013        0003    UNITY   EQU     TEMP+1    Unity variable
0014        0004    NUMSTP  EQU     UNITY+1   Number of steps
0015        0005    SAVACH  EQU     NUMSTP+1  Accumulator high
0016        0006    SAVACL  EQU     SAVACH+1  Accumulator low
0017        0007    SAVAR0  EQU     SAVACL+1  AR0
0018        0008    SAVAR1  EQU     SAVAR0+1  AR1
0019        0009    SAVPC   EQU     SAVAR1+1  Program Counter
0020                *
0021 0000                  PSEG
0022 0000                  AORG    >0
0023                *
0024 0000 F900      START  B       MAIN
     0001 0008
0025 0002 F900             B       MONIT
     0003 0700
0026                *
0027 0008                  AORG    >8
0028                *                                                  *****
0029 0008 7F80      MAIN   NOP             Initialize     * Add this   *   *
0030 0009 6E01             LDPK    1       to one         * instructions * E *
0031 000A 7E00             LACK    0       step           * to the     * X *
0032 000B 5004             SACL    NUMSTP                 * user       * A *
0033 000C 7F82             EINT            Enable int.    * program    * M *
0034 000D 4702             IN      TEMP,7  Set 1st int.   * for        * P *
0035 000E 7F80             NOP                            * monitoring * L *
0036 000F 7F80             NOP                                         * E *
0037 0010 7F80             NOP                                         *   *
0038 0011 7F80             NOP                                         * O *
0039 0012 7F80             NOP                                         * F *
0040 0013 7F80             NOP                                         *   *
0041 0014 7F80             NOP                                         * U *
0042 0015 7F80             NOP                                         * S *
0043 0016 7E00      ZERO   LACK    0       Load accumulator with 0     * E *
0044 0017 F900             B       SKIP                                * R *
     0018 001C
0045                ***********************************************************
0046 0019 4702             IN      TEMP,7  This instr. insert break point *   *
0047                *                       to line SACL 10,0              *
0048                ***********************************************************
0049 001A 7E2E      LVALUE LACK    46      Load accumulator with 01010101 *   *
0050                *                                                      *   *
0051                ***********************************************************
0052 001B 4702             IN      TEMP,7  This instr. insert break point *   *
0053                *                       to line LAR AR0,10             *
0054                ***********************************************************
```

```
MONIT        32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107    00:56:51  08-25-87
                                                                   PAGE 0002

0055                *      Move Accumulator to Auxiliary Register          * P *
0056                *                                                      * R *
0057 001C 500A      SKIP   SACL    10,0    Store AC to internal mem. 10    * O *
0058 001D 380A             LAR     AR0,10  Re-load AR0                     * G *
0059 001E 6880             LARP    0       Load AR0 pointer                * R *
0060                *                                                      * A *
0061                *      Move Auxiliary Register to Accumulator          * M *
0062                *                                                      *   *
0063 001F 300A             SAR     AR0,10  Save AR0                        *****
0064 0020 200A             LAC     10,0    Load into accumulator
0065 0021 FF00             BZ      LVALUE
     0022 001A
0066 0023 F900             B       ZERO
     0024 0016
0067                *
0068 0700                  AORG    >700
0069                *
0070 0700 6E01      MONIT  LDPK    1
0071 0701 7C00             SST     SAVSTS  Save current context
0072 0702 5805             SACH    SAVACH
0073 0703 5006             SACL    SAVACL
0074 0704 3007             SAR     0,SAVAR0
0075 0705 3108             SAR     1,SAVAR1
0076 0706 7E01             LACK    1       Initialize TEMP1 = 1
0077 0707 5003             SACL    UNITY
0078 0708 7F9D             POP
0079 0709 5009             SACL    SAVPC   Save Program Counter
0080 070A 7F9C             PUSH            Restore Return Address
0081                *
0082 070B 4301      RDSTS  IN      STATUS,3
0083 070C 7E20             LACK    >20     Check interrupting device
0084 070D 5002             SACL    TEMP
0085 070E 2802             LAC     TEMP,8
0086 070F 7901             AND     STATUS
0087 0710 FF00             BZ      ABO68K  Abort, 68000 or monitor ?
     0711 073D
0088 0712 2004             LAC     NUMSTP  Check if executing several steps
0089 0713 FE00             BNZ     MONITC
     0714 0756
0090                *
0091 0715 4202      MAILB  IN      TEMP,2  Input Commande Code from Mail-Box
0092 0716 2002             LAC     TEMP
0093 0717 FA00             BLZ     CONT    Is Continue ?
     0718 0736
0094 0719 5004             SACL    NUMSTP  Is monitor several steps ?
0095 071A FE00             BNZ     MONITC  Is single step ?
     071B 0756
0096                *
0097 071C 4002      STEP   IN      TEMP,0  Command Reg. to Out-FIFO's
0098 071D 4E02             OUT     TEMP,6  It is single step
0099 071E 4E01             OUT     STATUS,6
0100 071F 4E00             OUT     SAVSTS,6 then
0101 0720 4E05             OUT     SAVACH,6
0102 0721 4E06             OUT     SAVACL,6 transfer local parameters
0103 0722 4E07             OUT     SAVAR0,6
0104 0723 4E08             OUT     SAVAR1,6 to the Out-FIFO's
0105 0724 4E09             OUT     SAVPC,6
```

```
MONIT       32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107     00:56:51  08-25-87
                                                                   PAGE 0003

0106 0725 7E80          LACK    >80
0107 0726 5002          SACL    TEMP
0108 0727 2802          LAC     TEMP,8
0109 0728 5002          SACL    TEMP
0110 0729 4E02          OUT     TEMP,6    end of transfer message
0111              *
0112 072A F600  USER    BIOZ    EXIT      Wait for user response
     072B 072E
0113 072C F900          B       USER      command selection
     072D 072A
0114              *
0115 072E 6E01  EXIT    LDPK    1
0116 072F 6505          ZALH    SAVACH    Restore Accumulator High
0117 0730 6106          ADDS    SAVACL    Restore accumulator low
0118 0731 3807          LAR     0,SAVAR0  Restore AR0
0119 0732 3908          LAR     1,SAVAR1  Restore AR1
0120 0733 7B00          LST     SAVSTS    Restore Status
0121 0734 7F82          EINT              Enable subsequent interrupts
0122 0735 7F8D          RET
0123              *
0124 0736 6E01  CONT    LDPK    1
0125 0737 6505          ZALH    SAVACH    Restore Accumulator High
0126 0738 6106          ADDS    SAVACL    Restore Accumulator Low
0127 0739 3807          LAR     0,SAVAR0  Restore AR0
0128 073A 3908          LAR     1,SAVAR1  Restore AR1
0129 073B 7B00          LST     SAVSTS    Restore status
0130 073C 7F8D          RET
0131              *
0132 073D 7E40  ABO68K  LACK    >40       Check interrupting device
0133 073E 5002          SACL    TEMP
0134 073F 2802          LAC     TEMP,8
0135 0740 7901          AND     STATUS    is from 68000 ?
0136 0741 FF00          BZ      ABORT
     0742 0749
0137 0743 7F80          NOP
0138 0744 7F80          NOP               Interrupt service routine
0139 0745 7F80          NOP
0140 0746 7F80          NOP               for 68000
0141 0747 F900          B       CONT
     0748 0736
0142              *
0143 0749 7E80  ABORT   LACK    >80
0144 074A 5002          SACL    TEMP      Check interrupting device
0145 074B 2802          LAC     TEMP,8
0146 074C 7901          AND     STATUS    is from ABORT ?
0147 074D FF00          BZ      CLRINT    if not, clear interrupt.
     074E 0754
0148 074F 7F80          NOP
0149 0750 7F80          NOP               Interrupt service routine
0150 0751 7F80          NOP
0151 0752 7F80          NOP               for ABORT
0152 0753 7F80          NOP
0153 0754 F900  CLRINT  B       CONT
     0755 0736
0154              *
0155 0756 1003  MONITC  SUB     UNITY     Decrement trace counter
0156 0757 5004          SACL    NUMSTP
```

```
MONIT       32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107     00:56:51  08-25-87
                                                                   PAGE 0004

0157 0758 FE00          BNZ     EXIT
     0759 072E
0158 075A F900          B       STEP
     075B 071C
0159              *
0160              *
0161                    END
NO ERRORS, NO WARNINGS
```

```
NO$IDT      32010 FAMILY MACRO ASSEMBLER   PC2.1 84.107     00:56:58  08-25-87
                                                                   PAGE 0005

0001 0000
0002 0000
0003 0000
0004 0000
0005 0000
0006 0000
0007 0000
0008 0000
NO ERRORS, NO WARNINGS
```

APPENDIX C

MC 68000 ASSEMBLY ROUTINES

C.1 MC68000 interrupt routine (for TMS32010 tracing and monitoring).

This routine initializes the interrupt controllers of all the Logical Units (level and vector). It then loads the start address of the interrupt service routine on the Crate Controller CPU and enables interrupts.

At each interrupt request from a given Logical Unit, the corresponding interrupt servicing routine reads data from the Logical Unit Out – FIFO's and writes them into the corresponding buffer.

Table 26: MC68000 interrupt routine for TMS32010 tracing and monitoring.

```
        TTL     *** Out-FIFO's INTERRUPT ***
*+INTFI   IDNT  1,1     Utility routines                                *+
*****************************************************************************
*** MODULE:     INTFI   Out-FIFO's Interrupt                            **
*****************************************************************************
***                                                                     **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                    **
***                                                                     **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                     **
*** DESCRIPTION: Initialize Out-FIFO's Interrupt Vector and Level of the **
***     interrupt controllers for n  FDDP Logical Units.                **
***         When an interrupt from the Out-FIFO's of FDDP occurs, the Crate**
***     Controller CPU reads the interrupting Out-FIFO and saves the value **
***     on the CPU-Stack.                                               **
***                                                                     **
***        The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<####" may change the configuration **
***     of the board or of the system.                                  **
***                                                                     **
*** ENTRY POINT:                                                        **
*+        xdef    Intfi                                                 **
***                                                                     **
*** INTERNALLY DEFINED PROCEDURES:                                      **
***     xdef    ivvct1,ivvct2,ivvct3,ivvct4,ivvct5,ivvct6               **
***                                                                     **
*** INTERNALLY DEFINED DATA:                                            **
***                                                                     **
*** EXTERNAL PROCEDURES REFERENCED:                                     **
***                                                                     **
*** EXTERNAL DATA REFERENCED:                                           **
*+      xref    base,units,fifobf,offset,rdfi,aicr0,aivr0               *+
***                                                                     **
*** SYSTEM INCLUDE FILES:                                               **
*** ENTRY CONDITION: Interrupt controllers on FDDP boards are not initialize**
***                                                                     **
*** EXIT CONDITION: Out-FIFO's GLOBAL VARIABLES have the updated value.  **
***                                                                     **
*** PARAMETER PASSING:  Through STACK.                                  **
***                                                                     **
*** GLOBAL VARIABLES: (stack)                                           **
***             fifobf(a5) = Out-FIFO of Logical Unit 1                 **
***             fifobf+2(a5) = Out-FIFO of Logical Unit 2               **
***             fifobf+4(a5) = Out-FIFO of Logical Unit 3               **
***             fifobf+6(a5) = Out-FIFO of Logical Unit 4               **
***             fifobf+8(a5) = Out-FIFO of Logical Unit 5               **
***             fifobf+10(a5) = Out-FIFO of Logical Unit 6              **
***                                                                     **
*** LOCAL VARIABLE:                                                     **
***                                                                     **
*****************************************************************************


*** REVISION HISTORY:                                                   **
***                                                                     **
***   DATE          AUTHOR                  CHANGES                     **
*** --------    ----------------    ------------------------------------ **
*** 02/18/86    Dario Crosetto      Initial coding                      **
*** 05/29/87    Shao Bei Bei        extend to six Logical Units         **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                     **
*****************************************************************************
***                                                                     **
*
*+wfivct  equu   $160            <#### FIFO vector address
*+level1  equu   $11             <#### interrupt level 1                 *+
*
```

```
intfi   lea.l   msgfi0(pc),a4
        bsr     outstg
*
ficlr   move.l  #base+cmd,a0
        move.b  #units,d2
ficlr1  move    #1,$10(a0)      Reset TMS
        move    #4,(a0)
        move    #$0C,(a0)       Clear FIFO with pulse
        move    #4,(a0)
        add.l   #offset,a0
        sub.b   #1,d2
        bne     ficlr1
*
        move.b  #units,d7
        move.l  #wfivct,a3      jmp table
        lea.l   vctbf(pc),a2    vector table
nvctf   move.l  (a2)+,(a3)+
        subq.b  #1,d7
        bne     nvctf
*
        move.b  #units,d7
        move.l  #base,a3
        move    #wfivct,d6
        lea.l   vctbf(pc),a2
        lsr     #2,d6                   divide by 4
nxtvf0  move.b  #level1,aicr0(a3)        fifo lever ctrl addr.
        move.b  d6,aivr0(a3)             fifo vector address
        adda.l  #offset,a3
        addq.b  #1,d6
        subq.b  #1,d7
        bne     nxtvf0
*
        lea.l   fifo01(a5),a1   initial tms fifo buffer pointer
        move.b  #units,d7
nxpti   move.l  a1,a0           (a1)=pointer
        addq.l  #8,a0           a0 point to first word
        move.l  a0,(a1)
        sub.l   #40,a1          next unit's fifo buffer
        sub.b   #1,d7
        bne     nxpti
*
        lea.l   fifo01+4(a5),a1  initial vme fifo buffer pointer
        move.b  #units,d7
nxptj   move.l  a1,a0           (a1)=pointer
        addq.l  #4,a0           a0 point to first word
        move.l  a0,(a1)
        sub.l   #40,a1          next unit's fifo buffer
        sub.b   #1,d7
        bne     nxptj

        move    #$2000,sr               Enable interrupt
*
        lea.l   msgfi1(pc),a4
        bsr     outstg
*
        rts
*
msgfi0  dc.b    '  FIFO interrupt initialed. '
        dc.l    $0d0a0404
msgfi1  dc.b    '  Interrupt tested. '
        dc.l    $0d0a0404
*
vctbf   dc.l    vvct01
        dc.l    vvct02
        dc.l    vvct03
        dc.l    vvct04
        dc.l    vvct05
        dc.l    vvct06
        dc.l    vvct07
```

```
        dc.l    wvct08
        dc.l    wvct09
        dc.l    wvct10
        dc.l    wvct11
        dc.l    wvct12
        dc.l    wvct13
        dc.l    wvct14
        dc.l    wvct15
        dc.l    wvct16
        dc.l    wvct17
        dc.l    wvct18
        dc.l    wvct19
        dc.l    wvct20
        dc.l    wvct21
        dc.l    wvct22
        dc.l    wvct23
        dc.l    wvct24
*
*  Interrupt subroutines
*
wvct01  move.l  a2,-(a7)                    push a2,a3
        move.l  a3,-(a7)
        lea.l   fifo01+40(a5),a2          pointer limit
        move.l  fifo01(a5),a3             current pointer
        move    base+rdfi,(a3)+
        bmi     rtpt1
        cmpa.l  a3,a2
        bcs     rtpt1
rtpn1   move.l  a3,fifo01(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2                    pop  a2,a3
        rte
rtpt1   lea.l   fifo01+8(a5),a3
        bra     rtpn1
*
wvct02  move.l  a2,-(a7)
        move.l  a3,-(a7)
        lea.l   fifo02+40(a5),a2
        move.l  fifo02(a5),a3
        move    base+offset*1+rdfi,(a3)+
        bmi     rtpt2
        cmpa.l  a3,a2
        bcs     rtpt2
rtpn2   move.l  a3,fifo02(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2
        rte
rtpt2   lea.l   fifo02+8(a5),a3
        bra     rtpn2
*
wvct03  move.l  a2,-(a7)
        move.l  a3,-(a7)
        lea.l   fifo03+40(a5),a2
        move.l  fifo03(a5),a3
        move    base+offset*2+rdfi,(a3)+
        bmi     rtpt3
        cmpa.l  a3,a2
        bcs     rtpt3
rtpn3   move.l  a3,fifo03(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2
        rte
rtpt3   lea.l   fifo03+8(a5),a3
        bra     rtpn3
*
wvct04  move.l  a2,-(a7)
        move.l  a3,-(a7)
        lea.l   fifo04+40(a5),a2
        move.l  fifo04(a5),a3
        move    base+offset*3+rdfi,(a3)+
        bmi     rtpt4
        cmpa.l  a3,a2
        bcs     rtpt4
rtpn4   move.l  a3,fifo04(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2
        rte
rtpt4   lea.l   fifo04+8(a5),a3
        bra     rtpn4
*
wvct05  move.l  a2,-(a7)
        move.l  a3,-(a7)
        lea.l   fifo05+40(a5),a2
        move.l  fifo05(a5),a3
        move    base+offset*4+rdfi,(a3)+
        bmi     rtpt5
        cmpa.l  a3,a2
        bcs     rtpt5
rtpn5   move.l  a3,fifo05(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2
        rte
rtpt5   lea.l   fifo05+8(a5),a3
        bra     rtpn5
*
wvct06  move.l  a2,-(a7)
        move.l  a3,-(a7)
        lea.l   fifo06+40(a5),a2
        move.l  fifo06(a5),a3
        move    base+offset*5+rdfi,(a3)+
        bmi     rtpt6
        cmpa.l  a3,a2
        bcs     rtpt6
rtpn6   move.l  a3,fifo06(a5)
        move.l  (a7)+,a3
        move.l  (a7)+,a2
        rte
rtpt6   lea.l   fifo06+8(a5),a3
        bra     rtpn6
*
wvct07  rte
wvct08  rte
wvct09  rte
wvct10  rte
wvct11  rte
wvct12  rte
wvct13  rte
wvct14  rte
wvct15  rte
wvct16  rte
wvct17  rte
wvct18  rte
wvct19  rte
wvct20  rte
wvct21  rte
wvct22  rte
wvct23  rte
wvct24  rte
*
*+      end
```

C.2 MC68000 display and command interpreting routines for TMS3210 tracing.

The display routine reads the tracing data stored in the buffer as described in Appendix C1 and display them on the screen in the format shown in Table 23 .

The command interpreting routine, selects the Logical Unit on which the tracing will be done. During tracing, it accepts from keyboard the following commands:

- M = return to the MC68000 monitor
- T = exit from TMS32010 program
- C = continue without tracing
- N = execute N instructions
- Any key = execute one instruction.

Table 27: MC68000 display routine for TMS32010 tracing.

```
*                                                    SEE MODULES IN APPENDIX C.4
*
dsp3     equu     *
         bsr      crt_trac                                          MODULE CRT_TRA
dsp3a    equ      *
*        bsr      otrace   copy TMS and Log. U. reg. to crt buff MODULE OUTBUF
         bsr      dsply
         bsr      swich2                                            MODULE SUBLIB
         beq      dsp3b
         rts
dsp3b    equ      *
*        bsr      otrace                                            MODULE OUTBUF
         bsr      dsply1
         bsr      swich3                                            MODULE SUBLIB
         beq      dsp3b
         rts
*
dsply    equ      *
         bsr       otcrlf
         lea.l     crtbf(a5),a4                                     MODULE SUBLIB
         bsr       outstg                                           MODULE IOCPU1B
         rts
*
dsply1   equ      *
         bsr      otcrlf
         lea.l    hlfcrt(a5),a4                                     MODULE SUBLIB
         bsr      outstg                                            MODULE IOCPU1B
         rts
*
```

C.3 MC68000 display and command interpreting routines for TMS3210 monitorin

The monitoring routines are intended to give a global view of the entire system and monitor all the active Logical Units at the same time.

The display routine reads the monitoring data stored in the buffer as described in Appendix C1 and display them on the screen in the format shown in Table 19 .

The command interpreting routine, accepts from keyboard the following commands:

- M = return to the MC68000 monitor
- T = exit from TMS32010 program
- C = continue without monitoring
- N = execute to the Nth break point
- Any key = execute to the next beak point.

Table 28: MC68000 display routine for TMS32010 monitoring.

```
                                                        SEE MODULES IN APPENDIX C.4
*
*
dsp2     equu     *
         bsr      crt_sts    format CRT for monitoring          MODULE CRT.STS
dsp2a    equ      *
         bsr      newold     copy new status to old status      MODULE OUTBUF
         bsr      omlsts     copy 1st fifo data to crt in bin   MODULE OUTBUF
         bsr      oenergy    copy 7 data fifo to crt buffer     MODULE OUTBUF
         bsr      dsply                                         MODULE SUBLIB
         bsr      swich1
         beq      dsp2a
         rts
*
dsply    equ      *
         bsr       otcrlf                                       MODULE SUBLIB
         lea.l     crtbf(a5),a4
         bsr       outstg                                       MODULE IOCPU1B
         rts
*
```

C.4 MC68000 test & application example routines.

```
        plen    66,6,6
        TTL     *** FDDP main program ***
FDDP    IDNT    1,1     Interactive FDDP test program
*****************************************************************
*** MODULE:     FDDP    Interactive test program                **
*****************************************************************
***                                                             **
*** ENVIROMENT: FDDP VME version. Any 68xxx VME-CPU              **
***                                                             **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                             **
*** DESCRIPTION: This software package, permits testing interactively **
***     the FDDP board.                                         **
***        A set of routines have been implemented and grouped in several **
***     modules. This modular structure of the software makes it relatively**
***     easy to modify or add new test routines, as well as calling them **
***     from Higher Level Languages.                            **
***                                                             **
***        Detailed description of Functions and hardware setting is **
***     available in the "FDDP User's manual Vers. 1.1".        **
***                                                             **
***        A good hardware test of the board can be done from the VME bus **
***     because all Functions generated by the TMS32010 DSP microprocessor **
***     can also be generated (at lower speed) by a 68xxx CPU through VME bus
***                                                             **
***        There are no PROMS on board containing the TMS32010 test program**
***     In order to execute some programs from the TMS32010, the routine **
***     that transfers the program code from the 68xxx memory to the TMS320**
***     memory must be executed first.                          **
***                                                             **
***        The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                          **
***                                                             **
*** ENTRY POINT:                                                **
        xdef    main                                            **
***                                                             **
*** INTERNALLY DEFINED PROCEDURES:                              **
        xdef    dsp1,dsp2,dsply                                 **
***                                                             **
*** INTERNALLY DEFINED DATA:                                    **
        xdef    base,units,mlsts,linebf,fifo01,broadc,offset    **
        xdef    pmem,dmem,cmd,ain,boxt,sts,icw,elo,iccw,occcw,strif **
        xdef    rstri,box6,rdfi,aicr0,aicr1,aivr0,aivr1         **
***                                                             **
*** EXTERNAL PROCEDURES REFERENCED:                             **
        xref    inch,onkey,outch,outstg                         **
***     xref    fftst,antst,test8,asci2,memory,hxasc6,copy,hexdcm,wait **
***     xref    switch,mailb,tmsvme,vmetms,mwait,tms32k,load,twait,crt_ml **
***     xref    readmail,rdanin,dspyen,putmail,inlcrt,intml,intfi **
***                                                             **
*** EXTERNAL DATA REFERENCED:                                   **
***                                                             **
*** SYSTEM INCLUDE FILES:                                       **
***     include fifotst.a68                                     **
***     include anintst.a68                                     **
***     include memory.a68                                      **
***     include sublib.a68      subroutines lib.                **
***     include mboxtst.a68                                     **
***     include tms32k.a68                                      **
***     include crt_tri.a68                                     **
***     include rdmail.a68                                      **
***     include outbuf.a68                                      **
***     include crt_sts.a68                                     **
***     include intml.a68                                       **
***     include intfi.a68                                       **
***     include crt_trac.a68                                    **
***                                                             **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED **
***                                                             **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR) **
***                                                             **
***    D:                                                       **
***                                                             **
***    A:                                                       **
***                                                             **
*** ENTRY CONDITION: System not initialized.                    **
***                                                             **
*** EXIT CONDITION:  Interactive test program running (This one) **
***                                                             **
*** PARAMETER PASSING:  Through STACK.                          **
*****************************************************************
***                                                             **
*** REVISION HISTORY:                                           **
***                                                             **
***  DATE          AUTHOR                  CHANGES               **
*** --------     ----------------     ------------------------- **
*** 02/06/86     Dario Crosetto       Initial codin             **
*** 02/18/86     Dario Crosetto       Memory Map, block transfer, interrupt **
*** 05/29/87     Shao Bei Bei         extend to six Logical Units. Stack layout
*** 06/19/87     Dario Crosetto       Labele revision and modular structure **
***                                                             **
*****************************************************************
***                                                             **
*
units   equu    6               <#### number of Logical Units Maximum=24
base    equ     $200000         <###### FDDP board jumper selection
*
*****************************************************************
***  OFFSET BETWEEN TMS32010 UNITS                              **
*****************************************************************
***                                                             **
offset  equ     $4000           Offset between Logical Units    **
***                                                             **
*****************************************************************
***  MAMORY MAP FOR BROADCAST                                   **
*****************************************************************
***                                                             **
broadc  equ     $280000         <###### Base address for BROADCAST **
***                             Use BROADCAST only for "Write" Functions **
*****************************************************************
***  MEMORY MAP INSIDE EACH LOGICAL UNIT                        **
*****************************************************************
*** TMS32010 MUST BE IN THE RESET STATE BEFORE GENERATING THE FOLLOWING **
*** FUNCTIONS FROM THE VME BUS. (See FDDP User's manual)        **
***                                                             **
pmem    equ     0               Offset of 1st address of Program memory **
dmem    equ     $1000           Offset of 1st address of Input Data memory **
***                             "pmem" and "dmem" are 4kbytes each, organized
***                             as 2k x 16-bit Read/Write words **
cmd     equ     $2000           Offset of Command Reg. (8-bit Read/Write) **
ain     equ     $2002           Offset of IN next value from ANIN. 8-bit Read
boxt    equ     $2004           Offset of Mail-Box TMS320 side, 16-bit R/W **
sts     equ     $2006           Offset of Logical Unit STATUS REG. (16-bit R)
icw     equ     $200A           Offset of CW-FIFO. 8-bit Read. **
elo     equ     $200C           Offset of Out-FIFO's. 16-bit Write. **
iccw    equ     $200C           Offset of CC-FIFO's. 8-bit Read. **
occcw   equ     $200E           Offset of CW & CC-FIFO's. 8-bit Write. **
```

```
strif   equ     $200E           Offset of Reset Lost Trigger Flag  (Read)  **
***                             or Software Trigger if CMD bit 5 = 1       **
***                             or Interrupt to TMS320 if CMD bit 4 = 1    **
***                                                                        **
***  TMS320 CAN BE IN ANY STATE DURING THE FOLLOWING FUNCTIONS GENERATED   **
***  FROM THE VME BUS                                                      **
***                                                                        **
rstri   equ     $2010           Offset of TMS320 RESET command (D0 = 1)    **
***                             Interrupt to TMS32010 (D1 = 1)             **
***                             and SOFTWARE-TRIGGER to TMS320 (D2 = 1)    **
***                             Write.                                     **
box6    equ     $2012           Offset of MAIL-BOX register VME-BUS sids   **
***                             16-bit, Read/Write.                        **
***                             The TMS32010 must not be in the "RESET"    **
***                             state while the 68xxx CPU Reads the MAIL-BOX
rdfi    equ     $2014           Offset of Out-FIFO's. 16-bit, Read         **
aicr0   equ     $2019           Offset of Interrupt Control Register 0     **
***                             Out-FIFO's to VME request.                 **
aicr1   equ     $201B           Offset of Interrupt Control Register 1     **
***                             Mail-Box to VME request.                   **
aivr0   equ     $201D           Offset of Out-FIFO's interrupt vector      **
aivr1   equ     $201F           Offset of Mail-Box interrupt vector        **
***                                                                        **
**************************************************************************
***                                                                        **
usrvct  equ     $100            for mailbox 24 vectors
wfivct  equ     $160            for FIFO
***                                                                        **
level1  equ     $11             interrupt level 1
level2  equ     $12             interrupt level 2
level3  equ     $13             interrupt level 3
level4  equ     $14             interrupt level 4
level5  equ     $15             interrupt level 5
level6  equ     $16             interrupt level 6
level7  equ     $17             interrupt level 7
***                                                                        **
*** STACK LAYOUT:                                                         **
heap    equ     $7000           Program size limit
stklmt  equ     heap+256        for safe area
stack   equ     heap+$5000      Stack size
stklp   equ     stack-128       for high lever lang.
bfsize  equ     132             line buffer size
***                                                                        **
*** GLOBAL VARIABLES: (stack)                                              **
***                                                                        **
glbbgn  equ     -16             Globals defined here:                      **
crtbf   equ     glbbgn-80*24    CTR buffer                                 **
ln3d    equ     crtbf+80*2+20   3th line                                   **
mail1   equ     crtbf+80*5+20   6th line                                   **
data1   equ     crtbf+80*7+14   8th line                                   **
blfcrt  equ     crtbf+80*12     13th line
ln16d   equ     crtbf+80*15+20  16th line                                  *
ln17d   equ     crtbf+80*16+20  17th line                                  **
ln18d   equ     crtbf+80*17+20  18th line                                  *
mlsts   equ     crtbf-units*2   mailbox data                               **
dataan  equ     mlsts-2*8*units anin data                                  **
ancnt   equ     dataan-2        array A counter                            **
energy  equ     ancnt-2*7*units Energy array                               **
encnt   equ     energy-2        array E counter                            **
linebf  equ     encnt-80        ascii integer buffer                       **
lnbuff  equ     linebf-bfsize   read line buffer                           **
fifo01  equ     lnbuff-2*20     fifo data buffer                           **
fifo02  equ     fifo01-2*20     fifo data buffer                           **
fifo03  equ     fifo02-2*20     fifo data buffer                           **
fifo04  equ     fifo03-2*20     fifo data buffer                           **
fifo05  equ     fifo04-2*20     fifo data buffer                           **
fifo06  equ     fifo05-2*20     fifo data buffer                           **
fifo07  equ     fifo06-2*20     fifo data buffer                           **
fifo08  equ     fifo07-2*20     fifo data buffer                           **
fifo09  equ     fifo08-2*20     fifo data buffer                           **
fifo10  equ     fifo09-2*20     fifo data buffer                           **
fifo11  equ     fifo10-2*20     fifo data buffer                           **
fifo12  equ     fifo11-2*20     fifo data buffer                           **
fifo13  equ     fifo12-2*20     fifo data buffer                           **
fifo14  equ     fifo13-2*20     fifo data buffer                           **
fifo15  equ     fifo14-2*20     fifo data buffer                           **
fifo16  equ     fifo15-2*20     fifo data buffer                           **
fifo17  equ     fifo16-2*20     fifo data buffer                           **
fifo18  equ     fifo17-2*20     fifo data buffer                           **
fifo19  equ     fifo18-2*20     fifo data buffer                           **
fifo20  equ     fifo19-2*20     fifo data buffer                           **
fifo21  equ     fifo20-2*20     fifo data buffer                           **
fifo22  equ     fifo21-2*20     fifo data buffer                           **
fifo23  equ     fifo22-2*20     fifo data buffer                           **
fifo24  equ     fifo23-2*20     fifo data buffer                           **
glbend  equ     fifo24          globals end    maximum 3660 bytes          **
***                                                                        **
main    equ     *
        move.l  #stklp,a6       stack top=prg.org+stack
        lea.l   (pc,a6.l),a5    stack limit pointer
        move    #glbend,a6
        lea.l   (a5,a6),a7      sys stack pointer
        move.l  a5,a6           a5=global mark
*                               a6=local stack
        bsr     fftst           fifo test
        bsr     antst           anin test
        bsr     memory          memory test
        bsr     mailbx          mailbox test
        bsr     tms32k          TMS32010 test
        bsr     intml           mail-box intrrupt initial.
        bsr     intfi           Out-FIFO's interrupt initial and  test
*
dspall  equ     *
        bsr     dsp1            FEMC Trigger
        bsr     dsp2            6 logical units monitor        APPENDIX C.3
        bsr     dsp3            TMS32010 Trace Utility         APPENDIX C.2
        bra     dspall
*
dsp1    equ     *                                              APPENDIX C.4
        bsr     crt_tri         format CRT                     MODULE CRT_TRI
dsp1a   equ     *
        bsr     omlstsl         copy mail box status to crt buf. I  MODULE
        bsr     odatani         copy analog inputs to crt buffer I
        bsr     ofifobf         copy fifo data to crt buffer     I  OUTBUF
        bsr     dsply                                          MODULE SUBLIB
        bsr     switch                                         MODULE SUBLIB
        beq     dsp1a
        rts
        include fifotst.a68   ok!
        include anintst.a68   ok!
        include memory.a68    ok!
        include sublib.a68    subroutines lib.
        include mbxtst.a68
        include tms32k.a68
        include crt_tri.a68
        include outbuf.a68
        include crt_sts.a68
        include intml.a68       ok!
        include intfi.a68       ok!
        include crt_trac.a68
*
        end     main
*
```

```
***                                                                    **
        plen    66,6,6
        TTL     *** Format screen for TMS32010 trace routine ***
*+CRT_TRAC IDNT 1,1     Utility routines                               *+
*************************************************************************
*** MODULE:      CRT_TRAC  Format screen for TMS32010 trace routine   **
*************************************************************************
***                                                                    **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                   **
***                                                                    **
*** LANGUAGE:    Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                    **
*** DESCRIPTION: Format the screen on the video terminal with diagnostic **
***     messages for the "TMS32010 TRACE" program.                     **
***         Writes on the crt out buffer the variables names:          **
***     copy crtmsgl to crtbf.                                          *
***                                                                    **
***        The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                 **
***                                                                    **
*** ENTRY POINT:                                                       **
***     xdef    crt_trac                                               **
***                                                                    **
*** INTERNALLY DEFINED PROCEDURES:                                     **
***                                                                    **
*** INTERNALLY DEFINED DATA:                                           **
***                                                                    **
*** EXTERNAL PROCEDURES REFERENCED:                                    **
***                                                                    **
*** EXTERNAL DATA REFERENCED:                                          **
***                                                                    **
*** SYSTEM INCLUDE FILES:                                              **
***                                                                    **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED   **
***                                                                    **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)     **
***                                                                    **
***    D:                                                              **
***                                                                    **
***    A:                                                              **
***                                                                    **
*** ENTRY CONDITION: Any state of the TMS32010                         **
***                                                                    **
*** EXIT CONDITION: Any state of the TMS32010.                         **
***                                                                    **
*** PARAMETER PASSING:  Through STACK.                                 **
***                                                                    **
*** GLOBAL VARIABLES: (stack)                                          **
***     crtbf                                                          **
***                                                                    **
*** LOCAL VARIABLE:                                                    **
***                                                                    **
*************************************************************************
***                                                                    **
*** REVISION HISTORY:                                                  **
***                                                                    **
***   DATE          AUTHOR                CHANGES                      **
*** --------    ----------------    ------------------------------    **
*** 05/29/87    Shao Bei Bei        Initial coding                    **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                    **
*************************************************************************
***                                                                    **
```

```
crt_trac equ    *
         lea.l  crtmsgl(pc),a4
         lea.l  crtbf(a5),a3
nextchl  move.b (a4)+,d0
         cmp.b  #4,d0
         beq    finishl
         cmp.b  #9,d0
         beq    spacel
         move.b d0,(a3)+
         bra    nextchl
spacel   move.b (a4)+,d1
         move.b #$20,d0
nspacel  move.b d0,(a3)+
         sub.b  #1,d1
         bne    nspacel
         bra    nextchl
finishl  move   #$0404,(a3)              eof
         rts
*
crtmsgl  dc.l     $094e0d0a        78 space and  crlf line1
         dc.l     $094e0d0a        78 space and  crlf line2
         dc.l     $094e0d0a        78 space and  crlf line3
         dc.l     $094e0d0a        78 space and  crlf line4
         dc.l     $094e0d0a        78 space and  crlf line5
         dc.l     $094e0d0a        78 space and  crlf line6
         dc.l     $094e0d0a        78 space and  crlf line7
         dc.l     $094e0d0a        78 space and  crlf line8
         dc.l     $094e0d0a        78 space and  crlf line9
         dc.l     $094e0d0a        78 space and  crlf line10
         dc.l     $094e0d0a        78 space and  crlf line11
         dc       $0912
         dc.b  '***  TMS32010  TRACE  Utility  ***' 44 char.
         dc.l     $09100d0a                                          line12
         dc.l  $094e0d0a        blank                                line13
         dc.b  '                  HEX              BINARY' 40 char.
         dc.l  $09260d0a                                             line14
         dc.b  'ACCUMULATOR:  xxxx.xxxx   '  26 char
         dc.b  'xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.' 36 char
         dc.l     $09100d0a                                          line15
         dc.b  '"p" REGISTER: xxxx.xxxx   '  26 char                 line16
         dc.b  'xxxxxxxx.xxxxxxxx.xxxxxxxx.xxxxxxxx.' 36 char
         dc.l  $09100d0a
         dc.b  '"T" REGISTER:      xxxx   '  26 char                 line17
         dc.b  'xxxxxxxx.xxxxxxxx AUXREG0: xxxx AUXREG1: xxxx DP:x ' 52
         dc    $0d0a
         dc.b  'STATUS REGISTER:   xxxx   '  26 char                 line18
         dc.b  'xxxxxxxx.xxxxxxxx ARP:x         OVM:x         OV:x ' 52
         dc    $0d0a
         dc.b  'PROGRAM COUNTER:    xxx   '  26 char                 line19
         dc.b  '    xxxx.xxxxxxxx STACK0:xxx stack1:xxxx stack2:xxx'
         dc    $0d0a
         dc.l  $094e0d0a        78 space and  crlf                  line20
         dc.b  '                     ***  Logical Unit Trace  *** '  line21
         dc.l  $091e0d0a
         dc.b  'STATUS REGISTER:   xxxx   '  26 char                 line22
         dc.b  'xxxxxxxx.xxxxxxxx  CMD REISTER: xx  xxxx.xxxx      '
         dc    $0d0a
         dc.l  $094e0d0a        blank  line                          line23
         dc       $0404         end of text
*                                                                      *+
*+       end
*
```

```
        TTL     *** Format screen for LOGICAL UNIT STATUS monitorin ***
*+CRT_STS IDNT  1,1     Utility test routines                          *+
*****************************************************************************
*** MODULE:      CRT_STS Format screen for LOICAL UNIT STATUS monitoring  **
*****************************************************************************
***                                                                       **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                      **
***                                                                       **
*** LANGUAGE:    Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                       **
*** DESCRIPTION: Format the screen on the video terminal with diagnostic  **
***     messages for the monitoring of six LOGICAL UNITS STATUS registers. **
***         Writes on the crt out buffer the variables names:             **
***     copy crtmlm to crtbf.                                             **
***                                                                       **
***         The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                    **
***                                                                       **
*** ENTRY POINT:                                                          **
***     xdef    crt_sts                                                   **
***                                                                       **
*** INTERNALLY DEFINED PROCEDURES:                                        **
***                                                                       **
*** INTERNALLY DEFINED DATA:                                              **
***                                                                       **
*** EXTERNAL PROCEDURES REFERENCED:                                       **
***                                                                       **
*** EXTERNAL DATA REFERENCED:                                             **
***                                                                       **
*** SYSTEM INCLUDE FILES:                                                 **
***                                                                       **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED      **
***                                                                       **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)        **
***                                                                       **
***    D:                                                                 **
***                                                                       **
***    A:                                                                 **
***                                                                       **
*** ENTRY CONDITION: Any state of the TMS32010                            **
***                                                                       **
*** EXIT CONDITION: Any state of the TMS32010.                            **
***                                                                       **
*** PARAMETER PASSING:  Through STACK.                                    **
***                                                                       **
*** GLOBAL VARIABLES: (stack)                                             **
***     crtbf                                                             **
***                                                                       **
*** LOCAL VARIABLE:                                                       **
***                                                                       **
*****************************************************************************
***                                                                       **
*** REVISION HISTORY:                                                     **
***                                                                       **
***   DATE          AUTHOR                  CHANGES                       **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87     Shao Bei Bei        Initial coding                       **
*** 06/19/87     Dario Crosetto      Labels revision and modular structure **
***                                                                       **
*****************************************************************************
crt_sts equ     *
        lea.l   crtmlm(pc),a4
        lea.l   crtbf(a5),a3
nextml  move.b  (a4)+,d0
        cmp.b   #4,d0
        beq     finiml
        cmp.b   #9,d0
        beq     spacem
        move.b  d0,(a3)+
        bra     nextml
```



```
spacem  move.b  (a4)+,d1
        move.b  #$20,d0
nspacm  move.b  d0,(a3)+
        sub.b   #1,d1
        bne     nspacm
        bra     nextml
finiml  move    #$0404,(a3)   eof
        rts
crtmlm  dc      $0912
        dc.b    '    Unit 1'    Line1
        dc.b    '    Unit 2'
        dc.b    '    Unit 3'
        dc.b    '    Unit 4'
        dc.b    '    Unit 5'
        dc.b    '    Unit 6'
        dc.w    $0d0a
        dc.w    $0912          line2
        dc.b    '   new/old'
        dc.b    '   new/old'
        dc.b    '   new/old'
        dc.b    '   new/old'
        dc.b    '   new/old'
        dc.b    '   new/old'
        dc.w    $0d0a          Line3
        dc.b    '   F1OR FIFO ready' line6  18 char.
        dc.l    $093c0d0a
        dc.b    'L  F1IR outFF full' line4
        dc.l    $093c0d0a
        dc.b    'O  F2OR CWFF ready' line5
        dc.l    $093c0d0a
        dc.b    'G  F2IR CWFF  full' line6
        dc.l    $093c0d0a
        dc.b    'I  F3OR CCFF ready' line7
        dc.l    $093c0d0a
        dc.b    'C  F3IR CCFF  full' line8
        dc.l    $093c0d0a
        dc.b    'A  F4OR  CCW ready' line9
        dc.l    $093c0d0a
        dc.b    'L  F4IR AjCCW full' line10
        dc.l    $093c0d0a
        dc.b    '   F5OR AjCW ready' Line11
        dc.l    $093c0d0a
        dc.b    'U  F5IR AdjCW full' line12
        dc.l    $093c0d0a
        dc.b    'N  REOC reset TMS ' line13
        dc.l    $093c0d0a
        dc.b    'I  LOTR lost trig.' line14
        dc.l    $093c0d0a
        dc.b    'T  I68IRQ from VME' line15
        dc.l    $093c0d0a
        dc.b    '  ABOC Abort occur' line16
        dc.l    $093c0d0a
        dc.b    '     CMD Register ' line17
        dc.l    $093c0d0a
        dc.b    'T    Acc.HIGH     ' line18
        dc.l    $093c0d0a
        dc.b    'M    Acc.LOW      ' line19
        dc.l    $093c0d0a
        dc.b    'S    STATUS REG.  ' line20
        dc.l    $093c0d0a
        dc.b    '3    AUX REG.0    ' line21
        dc.l    $093c0d0a
        dc.b    '2    AUX REG.1    ' line22
        dc.l    $093c0d0a
        dc.b    '0    Prog. Counter' line23
        dc.l    $093c0d0a
        dc      $0404         end of text
*
*+      end                                                               *+
*
```

```
        TTL     *** Format screen for FEMC Trigger monitoring ***
*+CRT_TRI IDNT  1,1     Utility test routines                          *+
*****************************************************************************
*** MODULE:     CRT_TRI Format screen for FEMC Trigger monitoring      **
*****************************************************************************
***                                                                    **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                   **
***                                                                    **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                    **
*** DESCRIPTION: Format the screen on the video terminal with diagnostic **
***     messages for the monitoring of FEMC 2nd and 3rd level trigger. **
***         Writes on the crt out buffer the variables names:          **
***     copy crtmsg to crtbf.                                          **
***                                                                    **
***         The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                 **
***                                                                    **
*** ENTRY POINT:                                                       **
***     xdef    crt_tri                                                **
***                                                                    **
*** INTERNALLY DEFINED PROCEDURES:                                     **
***                                                                    **
*** INTERNALLY DEFINED DATA:                                           **
***                                                                    **
*** EXTERNAL PROCEDURES REFERENCED:                                    **
***                                                                    **
*** EXTERNAL DATA REFERENCED:                                          **
***                                                                    **
*** SYSTEM INCLUDE FILES:                                              **
***                                                                    **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED   **
***                                                                    **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)     **
***                                                                    **
***    D:                                                              **
***                                                                    **
***    A:                                                              **
***                                                                    **
*** ENTRY CONDITION: Any state of the TMS32010                         **
***                                                                    **
*** EXIT CONDITION: Any state of the TMS32010.                         **
***                                                                    **
*** PARAMETER PASSING:  Through STACK.                                 **
***                                                                    **
*** GLOBAL VARIABLES: (stack)                                          **
***     crtbf                                                          **
***                                                                    **
*** LOCAL VARIABLE:                                                    **
***                                                                    **
*****************************************************************************
***                                                                    **
*** REVISION HISTORY:                                                  **
***                                                                    **
***   DATE          AUTHOR                  CHANGES                    **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                     **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                    **
*****************************************************************************
crt_tri equ     *                                                      **
        lea     crtmsg(pc),a4
        lea     crtbf(a5),a3
nextch  move.b  (a4)+,d0
        cmp.b   #4,d0
        beq     finish
        cmp.b   #9,d0
        beq     space
        move.b  d0,(a3)+
space   move.b  (a4)+,d1
        move.b  #$20,d0
nspace  move.b  d0,(a3)+
        sub.b   #1,d1
        bne     nspace
        bra     nextch
finish  move    #$0404,(a3)             eof
        rts
*
crtmsg  dc.l    $094e0d0a line1 78 space and  crlf
        dc      $0912     line2 18 space
        dc.b  '1987 Summer Test on FEMC Trigger at CERN  '  42 char.
        dc.l    $09120d0a
        dc.l    $094e0d0a line3
        dc      $090a     line4
        dc.b    '    Unit 1'
        dc.b    '    Unit 2'
        dc.b    '    Unit 3'
        dc.b    '    Unit 4'
        dc.b    '    Unit 5'
        dc.b    '    Unit 6'
        dc.l    $09080d0a
        dc.l    $094e0d0a line5
        dc.b      ' Mailbox:'  line6
        dc.l    $09440d0a
        dc.l    $094e0d0a  line7
        dc.b    '  Tot.EG  ' line8
        dc.l    $09440d0a
        dc.b    '     Ch.1  ' line9
        dc.l    $09440d0a
        dc.b    '     Ch.2  ' line10
        dc.l    $09440d0a
        dc.b    '     Ch.3  ' line11
        dc.l    $09440d0a
        dc.b    '     Ch.4  ' line12
        dc.l    $09440d0a
        dc.b    '     Ch.5  ' line13
        dc.l    $09440d0a
        dc.b    '  EG.CL1   ' line14
        dc.l    $09440d0a
        dc.b    '  I. CL1   ' line15
        dc.l    $09440d0a
        dc.b    '  J. CL1   ' line16
        dc.l    $09440d0a
        dc.b    '  EG.CL2   ' line17
        dc.l    $09440d0a
        dc.b    '  I. CL2   ' line18
        dc.l    $09440d0a
        dc.b    '  J. CL2   ' line19
        dc.l    $09440d0a
        dc.b    '  EG.CL3   ' line20
        dc.l    $09440d0a
        dc.b    '  I. CL3   ' line21
        dc.l    $09440d0a
        dc.b    '  J. CL3   ' line22
        dc.l    $09440d0a
        dc.l    $094e0d0a   line23
        dc      $0404       end of text
*
*+      end                                                            *+
*
```

```
        TTL     *** Copy the Global variables values to the CRT out buffer **
*+OUTBUF IDNT   1,1     Utility test routines
*****************************************************************************
*** MODULE: OUTPUF Copy the Global variable values to the CRT out buffer   **
*****************************************************************************
***                                                                         **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                        **
***                                                                         **
*** LANGUAGE:  Assembler (Quelo or M68NIL-CERN). Position independent code **
***                                                                         **
*** DESCRIPTION: Copy the Stack Global variables values of Logical Units   **
***     (mail1, data1, engy1, mlsts, dataan, energy) to the CRT buffer      **
***                                                                         **
***        The locations indicated by  "<@@@@@@ are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<@@@@ determine the configuration    **
***     of the board or of the system.                                      **
***                                                                         **
*** ENTRY POINT:                                                            **
***                                                                         **
*** INTERNALLY DEFINED PROCEDURES:                                          **
***     xref    omlsts,odataen,oenergy,omlsts1,oldnev                       **
***                                                                         **
*** INTERNALLY DEFINED DATA:                                                **
***                                                                         **
*** EXTERNAL PROCEDURES REFERENCED:                                         **
***                                                                         **
*** EXTERNAL DATA REFERENCED:                                               **
***                                                                         **
*** SYSTEM INCLUDE FILES:                                                   **
***                                                                         **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED        **
***                                                                         **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)          **
***                                                                         **
***    D:                                                                   **
***                                                                         **
***    A:                                                                   **
***                                                                         **
*** ENTRY CONDITION:    TMS3210 in any state                               **
***                                                                         **
*** EXIT CONDITION:     TMS32010 in any state                              **
***                                                                         **
*** PARAMETER PASSING:  Through STACK.                                      **
***                                                                         **
*** GLOBAL VARIABLES: (stack)                                               **
***     mail1                                                               **
***     mlsts                                                               **
***     data1                                                               **
***     engy1                                                               **
***     dataan                                                              **
***     energy                                                              **
***     fifo01                                                              **
*** LOCAL VARIABLES:                                                        **
***                                                                         **
*****************************************************************************
***                                                                         **
*** REVISION HISTORY:                                                       **
***                                                                         **
***   DATE          AUTHOR                  CHANGES                         **
*** --------    ----------------    ------------------------------------    **
*** 05/29/87    Shao Bei Bei        Initial coding                          **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure   **
***                                                                         **
*****************************************************************************
***
*
newold  equ     *
        lea.l   ln3d+4(a5),a4           ^line3 position 22
        move.b  #14,d7                  14 lines
frash0  move.b  #6,d6                   6 units
frash1  move.b  (a4),2(a4)              shift 2
        adda.l  #10,a4                  next column
        sub.b   #1,d6
        bne     frash1
        adda.l  #20,a4                  next line
        sub.b   #1,d7
        bne     frash0
        rts
*
omlsts  equ     *
        lea.l   fifo01+8(a5),a3         first fifo data
        lea.l   ln3d+4(a5),a4
        move.b  #6,d6                   6 units
pnxtu   move    (a3),d5
        move    d5,d4
        lsr     #2,d4                   bit15 goes to bit13
        and     #$2000,d4
        and     #$1fff,d5
        or      d4,d5
        move.b  #14,d7                  14 bits
sftnx   lsr     #1,d5
        bcc     out0
        move.b  #$31,d0
putbf   move.b  d0,(a4)
        adda.l  #80,a4
        sub.b   #1,d7
        bne     sftnx
        suba.l  #80*14-10,a4            next unit
        sub.l   #40,a3
        sub.b   #1,d6
        bne     pnxtu
        rts
*
omlsts1 equ     *
        move.b  #units,d2
        move.l  #base,a0
        lea     mail1(a5),a4            ^5th line buffer unit 1
        lea     mlsts(a5),a3
nexts   move    (a3)+,d3
        bsr     hxasc8
        sub.b   #1,d2
        bne     nexts
        rts
*
hxasc8  move.b  #4,d5                   4 ascii to hex
hloopm  move    d3,d4
        and.b   #$0F,d4
        cmp.b   #9,d4
        bhi     atofm
        add.b   #$30,d4
outlb   move.b  d4,-(a4)                to 3rd line buffer
        lsr     #4,d3
        sub.b   #1,d5
        bne     hloopm
        adda.l  #14,a4                  next position
        rts
atofm   add.b   #$37,d4
        bra     outlb
```

```
*
out0    move.b  #$30,d0
        bra     putbf
**
odatani equ     *
        clr     d7                      offset
        lea.l   dataan(a5),a1           48*2 array begins here
        lea.l   data1(a5),a3            CRT beffer possition
        move.b  #8,d4                   8 lines
nextu   lea.l   (a1,d7),a0              data pointer
        move.b  #6,d6                   6 colums
nextdt  move    (a0)+,d0                data
        and     #$ff,d0                 8 bits only
        bsr     hexdcm                  hex in d0 to ascii in line buffer
        lea     linebf(a5),a2
        move.l  (a2)+,(a3)+             6 char.
        move    (a2)+,(a3)+
        addq.l  #4,a3                   next unit
        sub.b   #1,d6
        bne     nextdt
        add.l   #20,a3                  next line
        add     #16,d7
        sub.b   #1,d4
        bne     nextu
        rts
*
*
oenergy equ     *
        lea.l   fifo01+10(a5),a1        second fifo data
        lea.l   ln17d+8(a5),a4            CRT buffer position
        move.b  #7,d2                   7 lines
nxtu    clr     d7                      offset
        move.b  #6,d6                   6 colums
nxtdt   move    (a1,d7),d3
        bsr     hxasc8
*nxtdt   move    (a1,d7),d0                 data
*       bsr     hexdcm                  hex in d0 to ascii in line buffer
*       lea.l   linebf(a5),a2
*       move.l  (a2)+,(a3)+             6 char.
*       move    (a2)+,(a3)+
*       addq.l  #4,a3                   next unit position
        sub.l   #40,d7                  next unit fifo data
        sub.b   #1,d6
        bne     nxtdt
        addq.l  #2,a1
        clr     d7
        add.l   #20,a4                  next line
        sub.b   #1,d2
        bne     nxtu
        rts
*
ofifobf equ     *
        clr     d7                      offset
        lea.l   fifo01+8(a5),a1         FIFO data here
        lea.l   data1(a5),a3            CRT buffer position
        move.b  #6,d4           6 colums
nxtufb  lea.l   (a1,d7),a0              data pointer
        move.b  #15,d6          15 line
nxtdfb  move    (a0)+,d0                data
        bsr     hexdcm                  hex in d0 to ascii in line buffer
        lea.l   linebf(a5),a2
        move.l  (a2)+,(a3)+             6 char.
        move    (a2)+,(a3)+
        add.l   #74,a3          next line
        sub.b   #1,d6
        bne     nxtdfb
        sub.l   #1190,a3        next colum
        sub     #40,d7                  next units fifo
        sub.b   #1,d4
        bne     nxtufb
        rts
*
*+      end
*
```

```
***                                                                     **
        plen    66,6,6
        TTL     *** Utility Subroutines ***
*+SUBLIB IDNT   1,1     Utility subroutines
***********************************************************************
*** MODULE:     SUBLIB   Utility subroutines of general use from other module
***********************************************************************
***                                                                     **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                    **
***                                                                     **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position indipendent code**
***                                                                     **
***********************************************************************
***                                                                     **
*** DESCRIPTION:  Tests Program Memory of TMS320 from VME bus and leaves **
***     the content unchanged.                                          **
***         Reads complesents the value, writes. The above operation + check*
***     is done twice on each memory location from 0 to $FFF of Program **
***     Memory.                                                         **
***                                                                     **
***         The locations pointed by "<######" are HARDWARE JUMPER SELECTABLE
***         The locations pointed by "<####" may change the configuration **
***     of the board or of the system.                                  **
***                                                                     **
*** ENTRY POINT:                                                        **
***                                                                     **
***-INTERNALLY DEFINED PROCEDURES:                                      **
        xdef    copy,hexdcm,hxasc6,wait,switch                          **
***                                                                     **
*** INTERNALLY DEFINED DATA:                                            **
***                                                                     **
*** EXTERNAL PROCEDURES REFERENCED:                                     **
        xref    monitor,otcrlf                                          **
***     xref    switch,hxasc6,copy,ocrlf                                **
***                                                                     **
*** EXTERNAL DATA REFERENCED:                                           **
*+      xref    base,units,offset,rstri,broadc                          *+
***                                                                     **
*** SYSTEM INCLUDE FILES:                                               **
***                                                                     **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED    **
***                                                                     **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)      **
***                                                                     **
***    D:                                                               **
***                                                                     **
***    A:                                                               **
***                                                                     **
*** ENTRY CONDITION: 50 ohm Lemo on Trigger connector                   **
***                                                                     **
*** EXIT CONDITION: ANIN modules tested from VME bus                    **
***                                                                     **
*** PARAMETER PASSING:  Through STACK.                                  **
***                                                                     **
*** GLOBAL VARIABLES: (stack)                                           **
***                                                                     **
*** LOCAL VARIABLES: (stack)                                            **
***                                                                     **
***********************************************************************
***                                                                     **
*** REVISION HISTORY:                                                   **
***                                                                     **
***   DATE          AUTHOR                  CHANGES                     **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                      **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                     **
***********************************************************************
***                                                                     **
*
trstep  equ     0               Trace one step
trcont  equ     $8000           Trace. Continue until program ends
masktr  equ     $7fff           Mask for Tracing many steps
maskun  equ     $7
*
copy    equ     *
        move    (a0)+,d0        copy a0^ string
        move    d0,(a1)+        to   a1^
        cmp.b   #$04,d0
        bne     copy
        rts
*
***********************************************************************
*
*  Subroutine hexadecimal conversion
*   hex in d0, decimal in linebf
*  R. used:d1,d2,a4
*
hexdcm  equ     *
        lea.l   linebf(a5),a4
        move.b  #$20,(a4)+              1 space
        move.l  #10000,d1
        bsr     dcm
        move.l  #1000,d1
        bsr     dcm
        move.l  #100,d1
        bsr     dcm
        move.l  #10,d1
        bsr     dcm
        add.b   #$30,d0
        move.b  d0,(a4)+
        move.l  #$0D0A0404,(a4)
*
        lea.l   linebf(a5),a4
        add.l   #1,a4
        move.b  #4,d2                   take off at most 4 zero
if0     move.b  (a4),d1
        cmp.b   #$30,d1
        bne     rtsd
        move.b  #$20,(a4)+
        sub.b   #1,d2
        bne     if0
rtsd    rts
*
dcm     move.b  #$30,d2
        bclr    #31,d0                  force to pluse
dcm1    add.b   #1,d2
        sub.l   d1,d0
        bge     dcm1
        sub.b   #1,d2
        add.l   d1,d0
        move.b  d2,(a4)+
        rts
***********************************************************************
*
*  Subroutine wait 4 second
*
wait    move    #15,d1                  0.27*15 = 4 second
```

```
wait0   bsr     wait1
        sub     #1,d1
        bne     wait0
        rts
wait1   move    #0,d0                   wait 0.27 second
wait2   add     #1,d0
        cmp     #0,d0
        bne     wait2
        rts
*****************************************************************************
*
*  Subroutine switch
*
switch  equ *
        lea.l   swhmsg(pc),a4           call switch
        bsr     outstg                  beq  retest
        bsr     inch                    bne  continue
        cmp.b   #$0d,d0
        beq     cntn
        cmp.b   #$0a,d0
        beq     cntn
        cmp.b   #$20,d0
        beq     cntn
        cmp.b   #'M',d0
        beq     monitor
        cmp.b   #'R',d0
        beq     rtst
        bra     switch
cntn    move    #0,ccr
        rts
rtst    move    #4,ccr
        rts
*
swhmsg  dc.b    '  Type: space or return for continue'  36
        dc.b    ';  R for retest;  M to Monitor :'      32
        dc.l    $0d0a0404
*
swich1  equ     *
        lea.l   sw1msg(pc),a4           call swich1
        bsr     outstg                  beq  retest
        bsr     inch                    bne  continue
        cmp.b   #'T',d0
        beq     ntest
        cmp.b   #'M',d0
        beq     monitor
        cmp.b   #'N',d0
        beq     nstep
        cmp.b   #'C',d0
        beq     cntn1
anykey  move.l  #broadc,a0              One step of TMS
        move    #trstep,box6(a0)        and repeat last test
        move    #4,rstri(a0)
        move    #4,ccr
        rts
ntest   move    #0,ccr
        rts
*
cntn1   move.l  #broadc,a0              Execute TMS prg. until end
        move    #trcont,box6(a0)        and repeat test
        move    #4,rstri(a0)
        move    #4,ccr
        rts
*
nstep   bsr     otcrlf
        lea.l   sw2msg(pc),a4           'N' steps TMS prg.
        bsr     outstg                  and repeat test
        bsr     rddcm
        bcs     nstep
        and     #masktr,d0
        move.l  #broadc,a0
        move    d0,box6(a0)
        move    #4,rstri(a0)
        move    #4,ccr
        rts
*
sw1msg  dc.b  'Type: M=Monitor, T=next test, C=Continue,'
        dc.b  ' M=Number of steps, Anykey=one step. '
        dc      $0404
sw2msg  dc.b  ' How many steps?(0--32767): '
        dc      $0404
*
swich2  equ     *
nstep2  bsr     otcrlf
        lea.l   sw3msg(pc),a4
        bsr     outstg
        bsr     rddcm
        bcs     nstep2                  on error again
        and     #maskun,d0
        beq     nstep2
        cmp     #units,d0
        bgt     nstep2
        subq    #1,d0
        mulu    #offset,d0
        move.l  #base,a1
        lea.l   (a1,d0),a0
swich3  equ     *
        lea.l   sw1msg(pc),a4           call swich1
        bsr     outstg                  beq  retest
        bsr     inch                    bne  continue
        cmp     #'T',d0
        beq     ntest
        cmp.b   #'M',d0
        beq     monitor
        cmp.b   #'N',d0
        beq     rptinp
        cmp.b   #'C',d0
        beq     cntn1
anyky2  move.l  #broadc,a0              One step of TMS
        move    #trstep,box6(a0)        and repeat last test
        move    #4,ccr
        rts
*
cntn2   move.l  #broadc,a0              Execute TMS prg. until end
        move    #trcont,box6(a0)        and repeat test
        move    #4,ccr
        rts
*
rptinp  bsr     otcrlf
        lea.l   sw2msg(pc),a4
        bsr     outstg
        bsr     rddcm
        bcs     rptinp
        and     #masktr,d0
        move    d0,box6(a0)
        move    #4,rstri(a0)
        move    #4,ccr
        rts
*
```

```
sw3msg  dc.b    'Which Logical Unit?(1--6):'
        dc      $0404
*
*****************************************************************************
*
*  Subroutine Hexadecimal to Ascii conversion
*
hxasc6  equ     *
        move    #6,d5           d3 to 6 hex ascii
hloop   move.l  d3,d4           and output
        and     #$f,d4
        cmp     #9,d4
        bhi     atof
        add     #$30,d4
outasc  move.b  d4,-(a7)        push result
        lsr.l   #4,d3
        sub     #1,d5
        bne     hloop
        move    #6,d5
oloop   move.b  (a7)+,d4        pop to output
        bsr     outch
        sub     #1,d5
        bne     oloop
        rts
*
atof    add     #$37,d4
        bra     outasc
*
*****************************************************************************
*
*    subroutine readln read line to linebf first word is the counter.
*
readln  equ     *
        lea.l   lnbuff+2(a5),a4
        clr     d1              as bytes counter
inloop  bsr     inch
        addq    #1,d1
        cmp     #bfsize-2,d1
        bge     monitor         too many char. error
        move.b  d0,(a4)+
        cmp.b   #$0d,d0
        beq     inrts
        cmp.b   #$0a,d0
        beq     inrts
        cmp.b   #$08,d0         back space
        beq     bksps
        cmp.b   #$20,d0
        blt     ctrlc
        move.b  d0,d4
        bsr     outch           echo
        bra     inloop
ctrlc   move.b  #'?',d4
        bsr     outch
        bra     inloop
bksps   subq    #1,d1
        bcs     readln
        move.b  d0,d4
        bsr     outch
        move.b  #$20,d4
        bsr     outch
        move.b  d0,d4
        bsr     outch
        bra     inloop
inrts   subq    #1,d1
        move    d1,lnbuff(a5)
        bsr     otcrlf
        rts
*****************************************************************************
*
*  Subrutine rddcm read number 0 to 65535, data in word in d0
*    if error set c.
*
rddcm   equ     *
        bsr     readln
        clr     d0
        clr     d2
        move    #1,d3
        lea.l   lnbuff(a5),a0
        move    (a0),d1         counter
        cmp     #5,d1
        bgt     rderrs          read error
        lea.l   2(a0,d1),a0     point to  last char.+1
nxdcm   move.b  -(a0),d2
        cmp.b   #'0',d2
        blt     rderrs
        cmp.b   #'9',d2
        bgt     rderrs
        and     #$0f,d2
        mulu    d3,d2
        cmp.l   #$10000,d2
        bge     rderrs
        mulu    #10,d3
        add.l   d2,d0
        cmp.l   #$10000,d0
        bge     rderrs
        subq    #1,d1
        bne     nxdcm
        move    #0,ccr          clear c as ok flag
        rts
rderrs  move    #1,ccr          set c as error flag
        rts
..
dsply   equ     *                                                       *+
        bsr     otcrlf
        lea.l   crtbf(a5),a4
        bsr     outstg
        rts
*a
*+      end
```

```
***                                                                     **
        plen    66,6,6
        TTL     *** MAIL-BOX INTERRUPT ***
*+INTML  IDNT   1,1     Utility routines                                *+
*************************************************************************
*** MODULE:     INTML   MAIL-BOX Interrupt                              **
*************************************************************************
***                                                                     **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                    **
***                                                                     **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                     **
*** DESCRIPTION: Initialize MAIL-BOX Interrupt Vector and Level of the  **
***     interrupt controllers for n  FDDP units.                        **
***         When an interrupt from the MAIL-BOX of a FDDP occurs, the Crate**
***     Controller CPU reads the interrupting Mail-Box and saves the value **
***     on the CPU-Stack.                                               **
***                                                                     **
***         The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                  **
***                                                                     **
*** ENTRY POINT:                                                        **
        xdef    Intml                                                   **
***                                                                     **
*** INTERNALLY DEFINED PROCEDURES:                                      **
***     xdef    ivct1,ivct2,ivct3,ivct4,ivct5,ivct6                    **
***                                                                     **
*** INTERNALLY DEFINED DATA:                                            **
***                                                                     **
*** EXTERNAL PROCEDURES REFERENCED:                                     **
***                                                                     **
*** EXTERNAL DATA REFERENCED:                                           **
*+      xref    base,units,mlsts,offset,box6,aicr1,aivr1                *+
***                                                                     **
*** SYSTEM INCLUDE FILES:                                               **
***                                                                     **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED    **
***                                                                     **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)      **
***                                                                     **
***    D:                                                               **
***                                                                     **
***    A:                                                               **
***                                                                     **
*** ENTRY CONDITION: Interrupt controllers on FDDP boards are not initialize*
***                                                                     **
*** EXIT CONDITION: Mail-Box GLOBAL VARIABLES have the updated value.   **
***                                                                     **
*** PARAMETER PASSING:  Through STACK.                                  **
***                                                                     **
*** GLOBAL VARIABLES: (stack)                                           **
***             mlsts(a5) = Mail-Box of Logical Unit 1                  **
***             mlsts+2(a5) = Mail-Box of Logical Unit 2                **
***             mlsts+4(a5) = Mail-Box of Logical Unit 3                **
***             mlsts+6(a5) = Mail-Box of Logical Unit 4                **
***             mlsts+8(a5) = Mail-Box of Logical Unit 5                **
***             mlsts+10(a5) = Mail-Box of Logical Unit 6               **
***                                                                     **
*** LOCAL VARIABLE:                                                     **
***                                                                     **
*************************************************************************
***                                                                     **
*** REVISION HISTORY:                                                   **
***                                                                     **
*** DATE        AUTHOR              CHANGES                             **
*** --------    ----------------    ------------------------------------ **
*** 02/18/86    Dario Crosetto      Initial coding                      **
*** 05/29/87    Shao Bei Bei        extend to six Logical Units         **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                     **
*************************************************************************
***                                                                     **
*+usrvct  equ     $100                    <#### user vector address
*+level1  equ     $11                     <#### interrupt level 1
*
*
intml   lea.l   msgmi(pc),a4
        bsr     outstg
*
        move.b  #units,d7
        move.l  #usrvct,a3
        lea.l   vctbl(pc),a2
nvctm   move.l  (a2)+,(a3)+
        subq.b  #1,d7
        bne     nvctm
*
        move.b  #units,d7
        move.l  #base,a3
        move    #usrvct,d6
        lsr     #2,d6                   divide by 4
nxtv    move.b  #level1,aicr1(a3)       mailbox level ctrl addr.
        move.b  d6,aivr1(a3)            mailbox vector address
        adda.l  #offset,a3
        addq.b  #1,d6
        subq.b  #1,d7
        bne     nxtv
*
        lea.l   mlsts(a5),a3            Clear mail buffer
        move.b  #units,d7
mlclr   clr     (a3)+
        subq.b  #1,d7
        bne     mlclr
*
        rts
*
*
msgmi   dc.b    '  Mailbox interrupet vectors initialed'
        dc.l    $0D0A0404
*
*
vctbl   dc.l    ivct1
        dc.l    ivct2
        dc.l    ivct3
        dc.l    ivct4
        dc.l    ivct5
        dc.l    ivct6
        dc.l    ivct7
        dc.l    ivct8
        dc.l    ivct9
        dc.l    ivct10
        dc.l    ivct11
        dc.l    ivct12
        dc.l    ivct13
        dc.l    ivct14
        dc.l    ivct15
        dc.l    ivct16
        dc.l    ivct17
```

```
        dc.l    ivct18
        dc.l    ivct19
        dc.l    ivct20
        dc.l    ivct21
        dc.l    ivct22
        dc.l    ivct23
        dc.l    ivct24
*
ivct1   move    base+box6,mlsts(a5)     read mailbox
        rte
ivct2   move    base+offset+box6,mlsts+2(a5)
        rte
ivct3   move    offset*2+box6+base,mlsts+4(a5)
        rte
ivct4   move    offset*3+box6+base,mlsts+6(a5)
        rte
ivct5   move    offset*4+box6+base,mlsts+8(a5)
        rte
ivct6   move    offset*5+box6+base,mlsts+10(a5)
        rte
ivct7   rte
ivct8   rte
ivct9   rte
ivct10  rte
ivct11  rte
ivct12  rte
ivct13  rte
ivct14  rte
ivct15  rte
ivct16  rte
ivct17  rte
ivct18  rte
ivct19  rte
ivct20  rte
ivct21  rte
ivct22  rte
ivct23  rte
ivct24  rte
*
*+      end
```

```
        TTL     *** Read Mail-Box test from VME bus, without interrupt ***
*+RDMAIL IDNT   1,1     Utility test routines
*****************************************************************************
*** MODULE: RDMAIL Read Mail-Box TMS32010 side without interrupt          **
*****************************************************************************
***                                                                       **
*** ENVIROMENT: FDDP VME version. Any 68xxx VME-CPU                       **
***                                                                       **
*** LANGUAGE:  Assembler (Quelo or M68MIL-CERN). Position independent code **
***                                                                       **
*** DESCRIPTION: Reads the Mail-Box (TMS32010 side) from VME bus for the  **
***     number of units specified in the MAIN program.                    **
***        Set TMS32010 on activity, reads the Mail-box of Loical Units,  **
***     writes results to STACK Global Variables.                         **
***
***        The locations indicated by  "<###### are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<#### determine the configuration  **
***     of the board or of the system.                                    **
***                                                                       **
*** ENTRY POINT:                                                          **
        xdef    rdmail                                                    **
***                                                                       **
*** INTERNALLY DEFINED PROCEDURES:                                        **
***                                                                       **
*** INTERNALLY DEFINED DATA:                                              **
***                                                                       **
*** EXTERNAL PROCEDURES REFERENCED:                                       **
***                                                                       **
*** EXTERNAL DATA REFERENCED:                                             **
*+      xref    units,base,rstri,box6                                     **
***                                                                       **
*** SYSTEM INCLUDE FILES:                                                 **
*** ENTRY CONDITION:     TMS3210 in any state                             **
***                                                                       **
*** EXIT CONDITION:      TMS32010 actyve                                  **
***                                                                       **
*** PARAMETER PASSING:  Through STACK.                                    **
***                                                                       **
*** GLOBAL VARIABLES: (stack)                                             **
***     mlsts                                                             **
***                                                                       **
*** LOCAL VARIABLES:                                                      **
*+  ***                                                                   **
*****************************************************************************
***                                                                       **
*** REVISION HISTORY:                                                     **
***                                                                       **
***   DATE          AUTHOR                  CHANGES                       **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87     Shao Bei Bei        Initial coding                       **
*** 06/19/87     Dario Crosetto      Labels revision and modular structure **
***                                                                       **
*****************************************************************************
***                                                                       **
rdmail  equ     *
        move.l  #base,a0
        move    #0,rstri(a0)            Enable TMS
        move.b  #units,d2
        lea     mlsts(a5),a3            mailbox status
        move    box6(a0),d1             read mailbox form VME
nextr   move    d1,(a3)+
        add.l   #offset,a0              offset to next unit
        sub.b   #1,d2
        bne     nextr
        rts
*
*+      end
```

```
***                                                                    **
        plen    66,6,6
        TTL     *** Program Memory test of Logical Units from VME bus ***
*+MEMORY IDNT   1,1     Utility Test routines
*************************************************************************
*** MODULE:     MEMORY   Program memory test of Logical Units from VME bus **
*************************************************************************
***                                                                    **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                   **
***                                                                    **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                    **
*** DESCRIPTION:  Tests Program Memory of TMS320 from the VME bus and leaves*
***     the content unchanged.                                         **
***         Reads, the value and writes it back complemented. This     **
***     operation is carried out twice on each memory location from 0 to **
***     $FFF of Program Memory and then the memory contents are checked **
***     for invariance.                                                **
***                                                                    **
***        The locations indicated by "<$$$$$$" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<$$$$" determine the configuration **
***     of the board or of the system.                                 **
***                                                                    **
*** ENTRY POINT:                                                       **
***     xdef    memory                                                 **
***                                                                    **
*** INTERNALLY DEFINED PROCEDURES:                                     **
***                                                                    **
*** INTERNALLY DEFINED DATA:                                           **
***                                                                    **
*** EXTERNAL PROCEDURES REFERENCED:                                    **
***     xref    outstg                                                 **
***     xref    switch,hxasc6,copy,ocrlf                               **
***                                                                    **
*** EXTERNAL DATA REFERENCED:                                          **
*+      xref    base,units,offset,rstri,broadc                         *+
***                                                                    **
*** SYSTEM INCLUDE FILES:                                              **
***                                                                    **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED   **
***                                                                    **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)      **
***                                                                    **
***    D:                                                              **
***                                                                    **
***    A:                                                              **
***                                                                    **
*** ENTRY CONDITION: 50 ohm Lemo on Trigger connector                  **
***                                                                    **
*** EXIT CONDITION: ANIN modules tested from VME bus                   **
***                                                                    **
*** PARAMETER PASSING:  Through STACK.                                 **
***                                                                    **
*** GLOBAL VARIABLES: (stack)                                          **
***                                                                    **
*** LOCAL VARIABLES: (stack)                                           **
lnbfan equ   -84-12     line buffer                                    **
tmpban equ   lnbfan-2   temporary buffer                               **
asbfan equ   tmpban-2   ascii cord buffer                              **
***                                                                    **
*************************************************************************
***                                                                    **
*** REVISION HISTORY:                                                  **
***                                                                    **
```

```
***   DATE          AUTHOR                CHANGES                      **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                     **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                    **
*************************************************************************
***                                                                    **
*
cmsg1   equ     -84-12
*
memory  link    a6,#cmsg1           local line buffer
        lea     mmsg1(pc),a0        rom
        lea     cmsg1(a6),a1        data area
        bsr     copy
        move    #1,broadc+rstri     Reset all TMS
        move.l  #base,a0            Base memory addr.
        move.l  #base+$1000,a1      Memory end+1
        move    #units,d2           6 units
mloop   not     (a0)
        move    (a0),d0
        not     (a0)
        move    (a0)+,d1
        not     d1
        cmp     d0,d1
        bne     merr                Report error
nextm   cmp.l   a0,a1
        bgt     mloop
        lea     cmsg1(a6),a4
        add.b   #1,6(a4)
        bsr     outstg
        add     #offset-$1000,a0    Next unit
        add     #offset,a1
        sub     #1,d2
        bne     mloop
        unlk    a6
        bsr     switch
        beq     memory
mrts    rts
merr    lea     mmsg2(pc),a4
        bsr     outstg
mpadd   move.l  a0,d3
        sub.l   #2,d3
        bsr     hxasc6              Hex to ascii
        bsr     otcrlf
        bra     nextm
*
mmsg1   dc.b    'Unit  0 Memory tested.'  22 char.
crlf    dc.w    $0d0a
        dc.w    $0404
*
mmsg2   dc.b    ' Memorey error at:  '
        dc.w    $0404
**
*+      end                                                            *+
*
```

```
        TTL     *** Mail-Box test from VME bus, without interrupt ***
*+MBOXTST IDNT  1,1     Utility test routines                          *+
*****************************************************************************
*** MODULE:      MBOXTST Mail-Box test from VME bus, without interrupt     **
*****************************************************************************
***                                                                       **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                      **
***                                                                       **
*** LANGUAGE:    Assembler (Quelo or M68MIL-CERN). Position independant code**
***                                                                       **
*** DESCRIPTION: Writes and Reads the Mail-Box (VME side and TMS side) from**
***     the VME bus for the number of units specified in the MAIN program. **
***         Stops activity of the TMS32010 with a software reset, tests   **
***     the Mail-Box TMS side (from TMS32010 to VME). Checks for errors   **
***         Set TMS32010 on activity, tests the Mail-Box VME side (from   **
***     VME to TMS32010). Checks for errors.                              **
***                                                                       **
***         The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                    **
***                                                                       **
*** ENTRY POINT:                                                          **
        xdef    mailbx                                                    **
***                                                                       **
*** INTERNALLY DEFINED PROCEDURES:                                        **
***     xref    tmsvme,vmetms                                             **
***                                                                       **
*** INTERNALLY DEFINED DATA:                                              **
***     xref    rstri,boxt,box6                                           **
*** EXTERNAL PROCEDURES REFERENCED:                                       **
        xref    outch,outstg                                              **
***     xref    switch                                                    **
***                                                                       **
*** EXTERNAL DATA REFERENCED:                                             **
*+      xref    base,units,offset,rdfi,rstri,elo                          *+
***                                                                       **
*** SYSTEM INCLUDE FILES:                                                 **
***                                                                       **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED      **
***                                                                       **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)        **
***                                                                       **
***    D:                                                                 **
***                                                                       **
***    A:                                                                 **
***                                                                       **
*** ENTRY CONDITION: Any state of the FIFO's                              **
***                                                                       **
*** EXIT CONDITION: FIFO's tested and cleared on exit.                    **
***                                                                       **
*** PARAMETER PASSING:  Through STACK.                                    **
***                                                                       **
*** GLOBAL VARIABLES: (stack)                                             **
***                                                                       **
*** LOCAL VARIABLE:                                                       **
***                                                                       **
*****************************************************************************
***                                                                       **
*** REVISION HISTORY:                                                     **
***                                                                       **
***   DATE          AUTHOR                  CHANGES                       **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                       **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                       **
*****************************************************************************
***                                                                       **
```

```
cmsgt   equ     -84-12                  line buffer 1
cmsgv   equ     cmsgt-84                line buffer 2
cmsg    equ     cmsgv-84                line buffer 3
mailbx  link    a6,#cmsg                local var. pointer
        lea     bmsgt(pc),a0
        lea     cmsgt(a6),a1
        bsr     copy                    to data area
        lea     bmsgv(pc),a0
        lea     cmsgv(a6),a1
        bsr     copy                    a0^ string to a1^
        lea     bmsg(pc),a0
        lea     cmsg(a6),a1
        bsr     copy
        move    #units,d2               Units number
        move.l  #base,a0                Base address
bloop   move    #$5555,d0               Test with 010101...
        bsr     ajstnb                  adjust unit number
        bsr     tmsvme                  Test TMS toVME
        bsr     vmetms                  VME to TMS
        lea     cmsg(a6),a4
        bsr     outstg
        add.l   #offset,a0              offset to next unit
        sub     #1,d2
        bne     bloop
        unlk    a6
        bsr     switch
        beq     mailbx
        rts
*
tmsvme  bsr     tmsvm1                  Test twice
tmsvm1  move    #1,rstri(a0)            Reset TMS
        move    d0,boxt(a0)             Write from TMS
        move    #0,rstri(a0)            Enable TMS
        move    box6(a0),d1             Read from VME
        cmp     d0,d1
        bne     bxerr                   Report error
tmsvm2  not     d0                      Test with 101010...
        rts
*
vmetms  bsr     vmetm1
vmetm1  move    d0,box6(a0)             Write from VME
        move    #1,rstri(a0)            Reset TMS
        move    boxt(a0),d1             Read from TMS
        cmp     d0,d1
        bne     bxerrv
vmetm2  move    #0,rstri(a0)            Enable TMS
        not     d0
        rts
*
bxerr   lea     cmsgt(a6),a4            report error
        bsr     outstg
        bra     tmsvm2
*
bxerrv  lea     cmsgv(a6),a4            report error
        bsr     outstg
        bra     vmetm2
ajstnb  lea     cmsgt(a6),a4            adjust number
        add.b   #1,6(a4)
        lea     cmsgv(a6),a4
        add.b   #1,6(a4)
        lea     cmsg(a6),a4
        add.b   #1,6(a4)
        rts
*
bmsgt   dc.b    'Unit  0 mailbox TMS to VME failed.',$0d,$0a,$04,$04   38char.
bmsgv   dc.b    'Unit  0 mailbox VME to TMS failed.',$0d,$0a,$04,$04   38char.
*
bmsg    dc.b    'Unit  0 mailbox tested. ',$0d,$0a,$04,$04
*
*+      end                                                             *+
*
```

```
        TTL     *** TMS32010 test ***
*+TMS32K  IDNT  1,1     Utility test routines                                  *+
*****************************************************************************
*** MODULE:      TMS32K  Test the TMS32010 functionality                      **
*****************************************************************************
***                                                                          **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                          **
***                                                                          **
*** LANGUAGE:    Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                          **
*** DESCRIPTION: Loads a TMS32010 assembly program into the LOGICAL UNIT      **
***     program memory, executes it from TMS32010 and checks for errors.     **
***         Tests the number of LOGICAL UNITS specified in the MAIN Program**
***         Writes a message to Mail-Box (VME side), loads a program into    **
***     the TMS32010 Program Memory. Set TMS32010 active that execute the    **
***     program (reads value from Mail-Box VME side and writes to memory     **
***     location of Loical Unit). Resets TMS32010 to no activity, reads      **
***     from VME bus the memory location modified and checks for error       **
***                                                                          **
***        The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<####" determine the configuration    **
***     of the board or of the system.                                       **
***                                                                          **
*** ENTRY POINT:                                                             **
***     xdef    tms32k                                                       **
***                                                                          **
*** INTERNALLY DEFINED PROCEDURES:                                           **
***     xdef    load                                                         **
*** INTERNALLY DEFINED DATA:                                                 **
***                                                                          **
*** EXTERNAL PROCEDURES REFERENCED:                                          **
***     xref    outch,outstg                                                 **
***     xref    switch                                                       **
***                                                                          **
*** EXTERNAL DATA REFERENCED:                                                **
*+      xref    base,units,offset,rstri,box6                                 *+
***                                                                          **
*** SYSTEM INCLUDE FILES:                                                    **
***                                                                          **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED         **
***                                                                          **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)           **
***                                                                          **
***    D:                                                                    **
***                                                                          **
***    A:                                                                    **
***                                                                          **
*** ENTRY CONDITION: Any state of the TMS32010                               **
***                                                                          **
*** EXIT CONDITION: TMS32010 tested and no active                            **
***                                                                          **
*** PARAMETER PASSING:  Through STACK.                                       **
***                                                                          **
*** GLOBAL VARIABLES: (stack)                                                **
***                                                                          **
*** LOCAL VARIABLE:                                                          **
***                                                                          **
*****************************************************************************
***                                                                          **
*** REVISION HISTORY:                                                        **
***                                                                          **
***   DATE            AUTHOR                      CHANGES                    **
*** 05/29/87    Shao Bei Bei        Initial coding                           **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure    **
***                                                                          **
*****************************************************************************
***
*
tms32k  equ     *
        move.l  #base,a0
        move.b  #units,d2
        move.b  #'1',d3
        move    #$5555,d1
        move.l  #base,a1                Base RAM address
tloop   move    #1,rstri(a0)                    Reset TMS
        bsr     load                    to TMS memory
        move    #0,rstri(a0)                    Enable TMS
        move    d1,box6(a0)             Write to mailbox
        move    #1,rstri(a0)                    Reset TMS
        move    $20(a1),d0              read from RAM
        cmp     d1,d0
        bne     terror
        lea.l   tmsg0(pc),a4
        bsr     outstg
        move.b  d3,d4
        bsr     outch
        lea.l   tmsg1(pc),a4
        bsr     outstg
tnext   add.l   #offset,a0              offset to next unit
        add.l   #offset,a1
        add.b   #1,d3
        sub.b   #1,d2
        bne     tloop
        bsr     switch
        beq     tms32k
        rts
*
load    lea.l   pmsg(pc),a2
        move.l  a1,a3
        move    #5,d6                   5 words program
load1   move    (a2)+,(a3)+             to RAM
        sub     #1,d6
        bne     load1
        rts
*
terror  lea.l   tmsg0(pc),a4
        bsr     outstg
        move.b  d3,d4
        bsr     outch
        lea.l   tmsg2(pc),a4
        bsr     outstg
        bra     tnext.
*
pmsg    dc.w    $7E10                   loop  LACK  >10
        dc.w    $4203                   IN    3,2
        dc.w    $7D03                   TBLW  3
        dc.w    $F900                   JMP   loop
        dc.w    $0000
*
tmsg0   dc.b    ' Unit '
        dc      $0404
tmsg1   dc.b    ' TMS32010 is ok!'      16char.
        dc.l    $0D0A0404
tmsg2   dc.b    '  ****TMS32010 does not work **** '    34char.
        dc.l    $0D0A0404
*
*+      end                                                                  *+
*
```

```
***                                                                    **
        plen    66,6,6
        TTL     *** Out-FIFO's test from VME bus, without interrupt ***
*+FIFOTST IDNT  1,1     Utility routines                               *+
*************************************************************************
*** MODULE:     FIFOTST Out-FIFO's test from VME bus, without interrupt ***
*************************************************************************
***                                                                    **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                   **
***                                                                    **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                    **
*** DESCRIPTION: Writes and Reads the Out-FIFO's from the VME bus for the **
***     number of units specified in the MAIN program.                 **
***         Stops activity of the TMS32010 with a software reset, turns on **
***     all leds of the Out-FIFO's, writes and reads the Out-FIFO'S,   **
***     checks for errors.                                             **
***                                                                    **
***         The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                 **
***                                                                    **
*** ENTRY POINT:                                                       **
***     xdef    fftst                                                  **
***                                                                    **
*** INTERNALLY DEFINED PROCEDURES:                                     **
***                                                                    **
*** INTERNALLY DEFINED DATA:                                           **
***                                                                    **
*** EXTERNAL PROCEDURES REFERENCED:                                    **
***     xref    outch,outstg                                           **
***     xref    switch                                                 **
***                                                                    **
*** EXTERNAL DATA REFERENCED:                                          **
*+      xref    base,units,offset,rdfi,rstri,elo                       *+
***                                                                    **
*** SYSTEM INCLUDE FILES:                                              **
***                                                                    **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED   **
***                                                                    **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)      **
***                                                                    **
***    D:                                                              **
***                                                                    **
***    A:                                                              **
***                                                                    **
*** ENTRY CONDITION: Any state of the FIFO's                           **
***                                                                    **
*** EXIT CONDITION: FIFO's tested and cleared on exit.                 **
***                                                                    **
*** PARAMETER PASSING:  Through STACK.                                 **
***                                                                    **
*** GLOBAL VARIABLES: (stack)                                          **
***                                                                    **
*** LOCAL VARIABLE:                                                    **
***                                                                    **
*************************************************************************
***                                                                    **
*** REVISION HISTORY:                                                  **
***                                                                    **
***   DATE          AUTHOR                  CHANGES                    **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                     **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                    **
*************************************************************************
***                                                                    **
```

```
*
fftst   equ     *
        move.l  #base,a0
        move.b  #units,d2
redled  move    #1,rstri(a0)            Reset TMS
        move    d2,elo(a0)              red led on
        add.l   #offset,a0
        sub.b   #1,d2
        bne     redled
        lea.l   fmsg0(pc),a4
        bsr     outstg
        bsr     switch
        beq     fftst
*
fftst1  move.l  #base,a0
        move.b  #units,d2
        move.b  #'1',d3
floop   move    #1,rstri(a0)            Reset TMS
        move    #4,cmd(a0)
        move    #$0C,cmd(a0)            CLR FIFO with pulse
        move    #4,cmd(a0)
        move    #$5555,d0               test with 0101...
        move    d0,elo(a0)              Write FIFO
        not     d0                      test with 1010...
        move    d0,elo(a0)              Write FIFO
        not     d0
        move    #0,rstri(a0)            Set TMS
        move    rdfi(a0),d1             Read FIFO
        cmp     d0,d1
        bne     ferror                  Check for error
        not     d0
        move    rdfi(a0),d1             Read FIFO
        cmp     d0,d1
        bne     ferror
        lea.l   fmsg1(pc),a4
        bsr     outstg
        move.b  d3,d4
        bsr     outch
        lea.l   fmsg2(pc),a4
        bsr     outstg
fnext   add.l   #offset,a0              offset to next unit
        add.b   #1,d3
        sub.b   #1,d2
        bne     floop
        bsr     switch
        beq     fftst1
        rts
*
ferror  lea.l   fmsg1(pc),a4
        bsr     outstg
        move.b  d3,d4
        bsr     outch
        lea.l   fmsg3(pc),a4
        bsr     outstg
        bra     fnext
*
fmsg0   dc.b    ' FIFO test... Are all the reds leds on? '  40char.
        dc.l    $0D0A0404
fmsg1   dc.b    ' Unit '
        dc.w    $0404           EOT
fmsg2   dc.b    ' FIFO tested. '
        dc.l    $0D0A0404
fmsg3   dc.b    '  ****FIFO Read ERROR!****'
        dc.l    $0D0A0404
*
*+      end                                                            *+
```

```
        TTL     *** ANIN module test from VME bus ***
*+ANINTST IDNT  1,1     Utility routines                                *+
*****************************************************************************
*** MODULE:     ANINTST ANIN module test from VME bus.                   **
*****************************************************************************
***                                                                      **
*** ENVIRONMENT: FDDP VME version. Any 68xxx VME-CPU                     **
***                                                                      **
*** LANGUAGE:    Assembler (Qualo or M68NIL-CERN). Position independent code**
***                                                                      **
*** DESCRIPTION:  Reads from the VME bus the digitized input values of ANIN**
***     module for the number of units specified in the MAIN program.    **
***         Stops activity of the TMS32010 with a software reset, reads  **
***     the digitized values from the ANIN modules and display their values**
***     on the terminal.                                                 **
***                                                                      **
***         The locations inicated by "<######" are HARDWARE JUMPER SELECTABL
***         The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                   **
***                                                                      **
*** ENTRY POINT:                                                         **
        xdef    antst                                                    **
***                                                                      **
*** INTERNALLY DEFINED PROCEDURES:                                       **
        xdef    asci2                                                    **
***                                                                      **
*** INTERNALLY DEFINED DATA:                                             **
***                                                                      **
*** EXTERNAL PROCEDURES REFERENCED:                                      **
        xref    outstg                                                   **
***     xref    switch                                                   **
***                                                                      **
*** EXTERNAL DATA REFERENCED:                                            **
*+      xref    base,units,offset,rstri,broadc,cmd,strif,ain             *+
***                                                                      **
*** SYSTEM INCLUDE FILES:                                                **
***                                                                      **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED     **
***                                                                      **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)       **
***                                                                      **
***    D:                                                                **
***                                                                      **
***    A:                                                                **
***                                                                      **
*** ENTRY CONDITION: 50 ohm Lemo on Trigger connector                    **
***                                                                      **
*** EXIT CONDITION: ANIN modules tested from VME bus                     **
***                                                                      **
*** PARAMETER PASSING:  Through STACK.                                   **
***                                                                      **
*** GLOBAL VARIABLES: (stack)                                            **
***                                                                      **
*** LOCAL VARIABLES: (stack)                                             **
*+lnbfan equ   -84-12     line buffer                                    **
*+tmpban equ   lnbfan-2   temporary buffer                               **
*+asbfan equ   tmpban-2   ascii cord buffer                              **
***                                                                      **
*****************************************************************************
***                                                                      **
*** REVISION HISTORY:                                                    **
***  DATE          AUTHOR                  CHANGES                       **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87    Shao Bei Bei        Initial coding                       **
*** 06/19/87    Dario Crosetto      Labels revision and modular structure **
***                                                                      **
*****************************************************************************
***                                                                      **
*
antst   equ     *
        link    a6,#asbfan              for local VAR.
        move    #1,broadc+rstri         reset all TMS
        lea.l   lnbfan(a6),a4           Fill with
        move.b  #40,d0                  80 spaces
anxtw   move    #$2020,(a4)+
        sub.b   #1,d0
        bne     anxtw
        move.l  #$0d0a0404,(a4)
*
anmain  move.b  #units,d2
        move.l  #base,a0
        lea.l   dataan(a5),a2           anin data array
        lea.l   lnbfan(a6),a3           line buffer
        move.l  anmsg1(pc),(a3)
        move    anmsg1+4(pc),4(a3)
anxtt   add     #1,4(a3)                unit x +1
        lea.l   4(a3),a4
        bsr     test8                   Read 8 channels
        lea.l   lnbfan(a6),a4
        bsr     outstg
        add.l   #offset,a0              offset to next unit
        sub.b   #1,d2
        bne     anxtt
        unlk    a6
        lea.l   anmsg(pc),a4
        bsr     outstg
        bsr     switch
        beq     antst
        rts
*
test8   add.l   #4,a4
        move    #$20,cmd(a0)            Enable interal trig.
        move    strif(a0),d0            A/D trigger
        move.b  #8,d1                   8 channels
nxtcn   move    ain(a0),d0              Read a data
        move    d0,(a2)+                Save data
        bsr     asci2                   to 2 ascii
        sub.b   #1,d1
        bne     nxtcn                   next channel
        move    #2,cmd(a0)              Reset pointer
        rts
*
asci2   move.b  d0,tmpban(a6)
        and     #$0F,d0                 Mask high bits
        cmp.b   #9,d0
        bhi     ascla
        add.b   #$30,d0                 0 to 9
asclal  move.b  d0,asbfan+1(a6)         save low byte
        move.b  tmpban(a6),d0
        lsr     #4,d0                   mask low bits
        cmp.b   #9,d0
        bhi     ascha
        add.b   #$30,d0
aschal  move.b  d0,asbfan(a6)           save high byte
tolban  move    asbfan(a6),6(a4)        to line buffer
        add.l   #6,a4                   next position
        rts
ascla   add.b   #$37,d0
        bra     asclal
ascha   add.b   #$37,d0
        bra     aschal
*
anmsg   dc.b    '  The all ANNI channels have been tested. '  42 char.
        dc.l    $0D0A0404
anmsg1  dc.b    'Unit 0'
*
*+      end                                                             *+
*
```

```
***                                                                    **
        plen    66,6,6
        TTL     *** Input/Output Driver for CPU-1B of Force Computer ***
IOCPU1B IDNT    1,1     Utility  routines                              *+
*************************************************************************
*** MODULE:     IOCPU1B  Input/Output Driver for CPU-1B of Force Computer **
*************************************************************************
***                                                                    **
*** ENVIRONMENT: FDDP VME version. CPU-1B from Force Computer          **
***                                                                    **
*** LANGUAGE:   Assembler (Quelo or M68MIL-CERN). Position independent code**
***                                                                    **
*** DESCRIPTION:  Input/Output Driver for the CPU-1B of Force Computer **
***                                                                    **
***          This module provides the necessary routines to input characters**
***     (returns data on Reg. D0) and output character (sends data from D4)**
***     from the Console Serial port RS-232C of CPU-1B.                **
***                                                                    **
***                                                                    **
***        The locations indicated by "<######" are HARDWARE JUMPER SELECTABL
***        The locations indicated by "<####" determine the configuration **
***     of the board or of the system.                                 **
***                                                                    **
*** ENTRY POINT:                                                       **
***                                                                    **
*** INTERNALLY DEFINED PROCEDURES:                                     **
        xdef    inch,onkey                                             **
        xdef    outch,outstg,otcrlf                                    **
        xdef    monitor                                                **
***                                                                    **
*** INTERNALLY DEFINED DATA:                                           **
***                                                                    **
*** EXTERNAL PROCEDURES REFERENCED:                                    **
        xref    linebf                                                 **
***                                                                    **
*** EXTERNAL DATA REFERENCED:                                          **
***                                                                    **
*** SYSTEM INCLUDE FILES:                                              **
***                                                                    **
*** REGISTER USAGE: (A)RGUMENT  (D)ESTROYED  (P)RESERVED  (R)ETURNED   **
***                                                                    **
***         0   1   2   3   4   5   6   7     SR hi    SR lo (CCR)      **
***                                                                    **
***    D:                                                              **
***                                                                    **
***    A:                                                              **
***                                                                    **
*** ENTRY CONDITION:                                                   **
***                                                                    **
*** EXIT CONDITION:                                                    **
***                                                                    **
*** PARAMETER PASSING:  Through STACK.                                 **
***     linebf                                                         **
***                                                                    **
*** GLOBAL VARIABLES: (stack)                                          **
***                                                                    **
*** LOCAL VARIABLES: (stack)                                           **
***                                                                    **
*************************************************************************
***                                                                    **
*** REVISION HISTORY:                                                  **
***                                                                    **
***   DATE          AUTHOR                  CHANGES                    **
*** --------    ----------------    ------------------------------------ **
*** 05/29/87     Shao Bei Bei        Initial coding                     **
*** 06/19/87     Dario Crosetto      Labels revision and modular structure **
***                                                                    **
*************************************************************************
***                                                                    **
*                                                                      **
acias   equ     $0c0080         address of RS-232C host interface (status) **
aciad   equ     acias+2         address of RS-232C host interface (data)   **
*
monitor trap    #14             monitor interrupt vector
*
outstg  move.b  (a4)+,d4        out a string a4^
        cmp.b   #4,d4           d4  used
        beq     orts
        bsr     outch
        bra     outstg
*
outch   btst.b  #1,acias        get status
        beq     outch
        move.b  d4,aciad
orts    rts
*
otcrlf    move.b  #$0d,d4
        bsr     outch
        move.b  #$0a,d4
        bsr     outch
        rts
*
inch    btst.b  #0,acias        read a char.
        beq     inch            from keyboard
inch1   move.b  aciad,d0        char. in d0
        rts
*
onkey   btst.b  #0,acias        if key typed
        bne     inch1           char. in d0
        clr     d0              else d0=0
inrts   rts
*
        end
*                                                                      *+
```

APPENDIX D

TMS32010 DSP: DOCUMENTATION

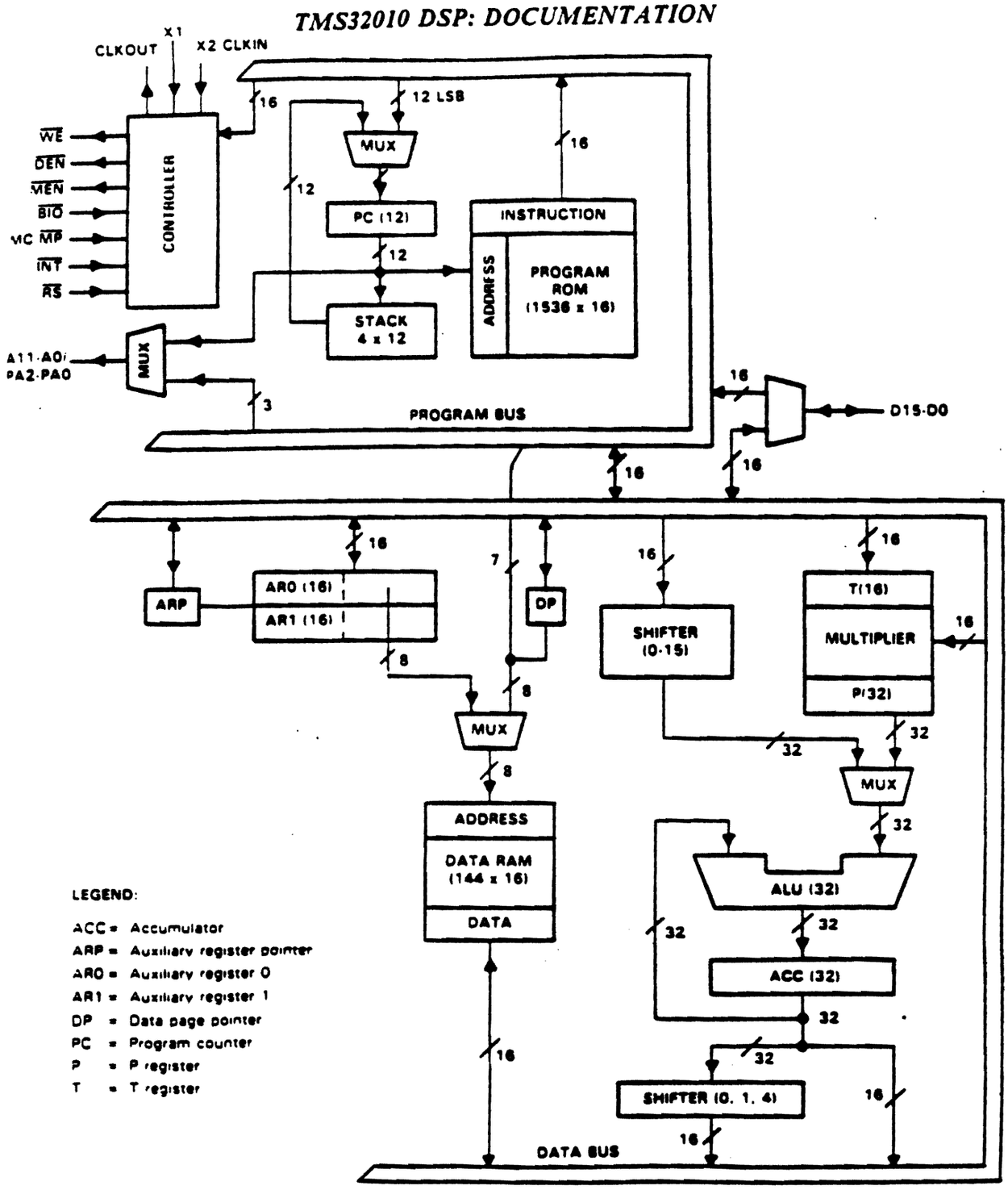


Figure 27: TMS32010 Functional block diagram

D.1 TMS32010 Instruction Set Summary

MNEMONIC	DESCRIPTION	NO CYCLES	NO WORDS	OPCODE INSTRUCTION REGISTER (15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0)
ABS	Absolute value of accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0
ADD	Add to accumulator with shift	1	1	0 0 0 0 ← S → I ← D →
ADDH	Add to high-order accumulator bits	1	1	0 1 1 0 0 0 0 0 I ← D →
ADDS	Add to accumulator with no sign extension	1	1	0 1 1 0 0 0 0 1 I ← D →
AND	AND with accumulator	1	1	0 1 1 1 1 0 0 1 I ← D →
APAC	Add P Register to accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
B	Branch unconditionally	2	2	1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BANZ	Branch on auxiliary register not zero	2	2	1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGEZ	Branch if accumulator ≥ 0	2	2	1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BGZ	Branch if accumulator > 0	2	2	1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BIOZ	Branch on (BIO) = 0	2	2	1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLEZ	Branch if accumulator ≤ 0	2	2	1 1 1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BLZ	Branch if accumulator < 0	2	2	1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BNZ	Branch if accumulator ≠ 0	2	2	1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BV	Branch on overflow	2	2	1 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
BZ	Branch if accumulator = 0	2	2	1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
CALA	Call subroutine from accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0
CALL	Call subroutine immediately	2	2	1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ← BRANCH ADDRESS →
DINT	Disable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1
DMOV	Copy contents of data memory location into next location	1	1	0 1 1 0 1 0 0 1 I ← D →
EINT	Enable interrupt	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 0
IN	Input data from port	2	1	0 1 0 0 0 ← PA → I ← D →
LAC	Load accumulator with shift	1	1	0 0 1 0 ← S → I ← D →
LACK	Load accumulator immediate	1	1	0 1 1 1 1 1 1 1 0 ← K →
LAR	Load auxiliary register	1	1	0 0 1 1 1 0 0 R I ← D →
LARK	Load auxiliary register immediate	1	1	0 1 1 1 0 0 0 R ← K →
LARP	Load auxiliary register pointer immediate	1	1	0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 K
LDP	Load data memory page pointer	1	1	0 1 1 0 1 1 1 1 1 ← D →
LDPK	Load data memory page pointer immediate	1	1	0 1 1 0 1 1 1 0 0 0 0 0 0 0 0 K
LST	Load status register	1	1	0 1 1 1 1 0 1 1 I ← D →
LT	Load T Register	1	1	0 1 1 0 1 0 1 0 I ← D →
LTA	LTA combines LT and APAC into one instruction	1	1	0 1 1 0 1 1 0 0 I ← D →
LTD	LTD combines LT, APAC, and DMOV into one instruction	1	1	0 1 1 0 1 0 1 1 I ← D →
MAR	Modify auxiliary register and pointer	1	1	0 1 1 0 1 0 0 0 I ← D →
MPY	Multiply with T Register, store product in P Register	1	1	0 1 1 0 1 1 0 1 I ← D →
MPYK	Multiply T Register with immediate operand, store product in P Register	1	1	1 0 0 ← K →
NOP	No operation	1	1	0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
OR	OR with accumulator	1	1	0 1 1 1 1 0 1 0 I ← D →
OUT	Output data to port	2	1	0 1 0 0 1 ← PA → I ← D →
PAC	Load accumulator from P Register	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0
POP	Pop stack to accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1
PUSH	Push stack from accumulator	2	1	0 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0
RET	Return from sub routine	2	1	0 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1
ROVM	Reset overflow mode	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0
SACH	Store high order accumulator bits with shift	1	1	0 1 0 1 1 ← R → I ← D →
SACL	Store low order accumulator bits	1	1	0 1 0 1 0 0 0 0 I ← D →
SAR	Store auxiliary register	1	1	0 0 1 1 0 0 0 R I ← D →
SOVM	Set overflow mode	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 1
SPAC	Subtract P Register from accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 1 0 0 0 0
SST	Store status register	1	1	0 1 1 1 1 1 0 0 I ← D →
SUB	Subtract from accumulator with shift	1	1	0 0 0 1 ← S → I ← D →
SUBC	Conditional subtract (for divide)	1	1	0 1 1 0 0 1 0 0 I ← D →
SUBH	Subtract from high order accumulator bits	1	1	0 1 1 0 0 0 1 0 I ← D →
SUBS	Subtract from accumulator with no sign extension	1	1	0 1 1 0 0 0 1 1 I ← D →
TBLR	Table read from program memory to data RAM	3	1	0 1 1 0 0 1 1 1 I ← D →
TBLW	Table write from data RAM to program memory	3	1	0 1 1 1 1 1 0 1 I ← D →
XOR	Exclusive OR with accumulator	1	1	0 1 1 1 1 0 0 0 I ← D →
ZAC	Zero accumulator	1	1	0 1 1 1 1 1 1 1 1 0 0 0 1 0 0 1
ZALH	Zero accumulator and load high-order bits	1	1	0 1 1 0 0 1 0 1 I ← D →
ZALS	Zero accumulator and load low-order bits with no sign extension	1	1	0 1 1 0 0 1 1 0 I ← D →

SYMBOL	MEANING
D	Data memory address field
I	Addressing mode bit
K	Immediate operand field
PA	3 bit port address field (PA0 through PA7 are predefined assembler symbols equal to 0 through 7, respectively)
R	1 bit operand field specifying auxiliary register
S	4 bit left shift code
X	3 bit accumulator left-shift field

Table 29: TMS32010 INSTRUCTION SET SUMMARY

Table 30:

D.2 TMS32010 Cross – Assembler Directives

ABSOLUTE ORIGIN **AORG**

AORG places a value in the location counter and defines the succeeding locations as absolute.

Syntax: [<label>] AORG [<wd-exp>] [<comment>]

RELOCATABLE ORIGIN **RORG**

RORG places a value in the location counter and defines the succeeding locations as program relocatable.

Syntax: [<label>] RORG [<exp>] [<comment>]

DUMMY ORIGIN **DORG**

DORG places a value in the location counter and defines the succeeding locations as a dummy section. No object code is generated in a dummy section.

Syntax: [<label>] DORG <exp> [<comment>]

BLOCK STARTING WITH SYMBOL **BSS**

BSS first assigns the label, if present, and then advances the location counter by the value of the expression.

Syntax: [<label>] BSS <wd-exp> [<comment>]

BLOCK ENDING WITH SYMBOL **BES**

BES first advances the location counter by the value of the expression and then assigns the label, if present.

Syntax: [<label>] BES <wd-exp> [<comment>]

DATA SEGMENT **DSEG**

DSEG places a value in the location counter and defines succeeding locations as data relocatable.

Syntax: [<label>] DSEG [<comment>]

DATA SEGMENT END **DEND**

DEND terminates a block of data-relocatable code by placing a value in the location counter and defining succeeding locations as program-relocatable.

Syntax: [<label>] DEND [<comment>]

COMMON SEGMENT **CSEG**

CSEG places a value in the location counter and defines succeeding locations as common-relocatable (i.e., relocatable with respect to a common segment).

Syntax: [<label>] CSEG ['<string>' [<comment>]]

COMMON SEGMENT END **CEND**

CEND terminates the definition of a block of common-relocatable code by placing a value in the location counter and defining succeeding locations as program-relocatable.

Syntax: [<label>] CEND [<comment>]

PROGRAM SEGMENT **PSEG**

PSEG places a value in the location counter and defines succeeding locations as program-relocatable.

Syntax: [<label>] PSEG [<comment>]

PROGRAM SEGMENT END **PEND**

PEND places a value in the location counter and defines succeeding locations as program-relocatable. (Since PEND properly appears only in program-relocatable code, the relocation type of succeeding locations remains unchanged.)

Syntax: [<label>] PEND [<comment>]

OUTPUT OPTIONS **OPTION**

OPTION selects several options for the assembler listing output.

Syntax: [<label>] OPTION <option-list> [<comment>]

PROGRAM IDENTIFIER **IDT**

IDT assigns a name to the object module produced.

Syntax: [<label>] IDT '<string>' [<comment>]

PAGE TITLE **TITL**

TITL supplies title to be printed in the heading of each page of the source listing.

Syntax: |<label>| TITL '<string>' |<comment>|

RESTART SOURCE LISTING **LIST**

LIST restores printing of the source listing

Syntax: |<label>| LIST |<comment>|

STOP SOURCE LISTING **UNL**

UNL halts the source listing output until the occurrence of a LIST directive.

Syntax: |<label>| UNL |<comment>|

EJECT PAGE **PAGE**

PAGE causes the assembler to continue the source program listing on a new page. The PAGE directive is not printed in the source listing, but the line counter increments.

Syntax: |<label>| PAGE |<comment>|

INITIALIZE WORD **DATA**

DATA places one or more values in one or more successive words of memory.

Syntax: |<label>| DATA <exp>|,<exp>| |<comment>|

INITIALIZE TEXT **TEXT**

TEXT places one or more characters in successive words of memory.

Syntax: |<label>| TEXT |'|'<string>' |<comment>|

DEFINE ASSEMBLY-TIME CONSTANT **EQU**

EQU assigns a value to a symbol.

Syntax: <label> EQU <exp> |<comment>|

EXTERNAL DEFINITION **DEF**

DEF makes one or more symbols available to other programs for reference.

Syntax: |<label>| DEF <symbol>|,<symbol>| |<comment>|

EXTERNAL REFERENCE **REF**

REF provides access to one or more symbols defined in other programs.

Syntax: |<label>| REF <symbol>|,<symbol>| |<comment>|

SECONDARY EXTERNAL REFERENCE **SREF**

SREF provides access to one or more symbols defined in other programs.

Syntax: |<label>| SREF <symbol>|,<symbol>| |<comment>|

FORCE LOAD **LOAD**

LOAD is similar to REF, but the symbol does not need to be used in the module containing the LOAD. The symbol used in LOAD must be defined in some other module. LOADs are used with SREFs.

Syntax: |<label>| LOAD <symbol>|,<symbol>| |<comment>|

PROGRAM END **END**

END terminates the assembly. The last source statement of a program is the END directive.

Syntax: |<label>| END |<symbol>| |<comment>|

COPY SOURCE FILE **COPY**

COPY changes the source input for the assembler.

Syntax: |<label>| COPY <file name> |<comment>|

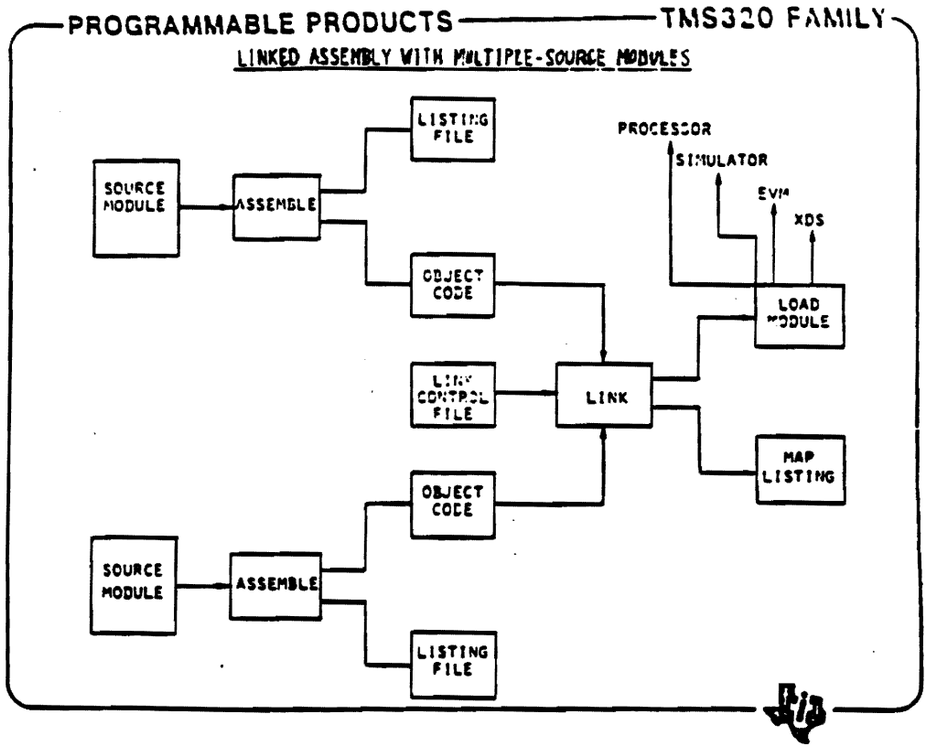
DEFINE MACRO LIBRARY **MLIB**

MLIB provides the name of a library containing macro definitions.

Syntax: |<label>| MLIB '<pathname>' |<comment>|

D.3 TMS32010 Cross Software Support

Table 31:



PROGRAMMABLE PRODUCTS — TMS320 FAMILY

TMS32010

CROSS SOFTWARE SUPPORT

	HOST COMPUTER	OPERATING SYSTEM
MACROASSEMBLER/LINKER -----	TI 990/10	DX-10
	TI BS	DX10/DNOS
	DEC VAX	VMS
	DEC VAX	BERKELEY UNIX 4.1
	DEC VAX	BERKELEY UNIX 4.2
	IBM 43XX	MVS/CMS
	TI-PC	MS-DOS
	IBM-PC	MS-DOS
SIMULATORS ----------	DEC VAX	VMS
	TI-PC	MS-DOS
	IBM-PC	MS-DOS
SOFTWARE LIBRARY ----------	DEC VAX	VMS

(CONT'D)

PROGRAMMABLE PRODUCTS — TMS320 FAMILY

THIRD - PARTY SUPPORT

PRODOTTO	SISTEMA	DENOMINAZIONE	PRODUTTORE
TMS32010 CROSS ASSEMBLER	CP/M-80 SYSTEMS	SYSTEM-320	A. ASHLEY
TMS32010 CROSS ASSEMBLER	MS-DOS PC-DOS	MAC320	A. ASHLEY
TMS32001 CROSS-ASSEMBLER	APPLE II CPM-80	TASSM	COMPUTALKER
TMS32010 SIMULATOR	APPLE II CPM-80	S320	COMPUTALKER
TMS32010 CROSS-ASSEMBLER LPC PRGS	VAX (VMS) PDP-11 (RT-11) TI/IBM PC	DSPS-3	DIGITAL SIGNAL PROCESSOR SOFTWARE IN
TMS32010 CROSS-ASSEMBLER TMS32010 FFT PRGS	VAX (VMS) PDP-11 (RT-11) TI/IBM PC	DSPS-2	DIGITAL SIGNAL PROCESSOR SOFTWARE IN

(CONT'D)

PROGRAMMABLE PRODUCTS — TMS320 FAMILY

THIRD - PARTY SUPPORT (CONT'D)

PRODOTTO	SISTEMA	DENOMINAZIONE	PRODUTTORE
TMS32010 CROSS-ASSEMBLER DOWNLOADER	TEKTRONICS 3560 DEC PDP-11 VAX (UNIX) 8080-Z80 SYSTEMS (CPM/TURBO-DOS)	PH ASS. TMS320 S/W	PH ASSOCIATION
TMS32010 CROSS-ASS. LINKER	INTELLEC II/30 INTELLEC MODEL 800 INTELLEC II/85	XI MACRO ASSEMBLER	PROCESSOR INNOVATIONS
TMS32010 DEV. HARDWARE	XI-90/30	XI CORE DEVELOPMENT	PROCESSOR INNOVATION
TMS32010 CROSS-ASS.LINKER	HP64000 LOGIC DEVELOPMENT SYSTEM	HP64000 ASS/LNK	HEWLETT-PACKARD
SKY320 DSP	DEC PDP-11 (RT-11)	SMOS	SKY COMPUTER INC.

APPENDIX E

MC68000 DOCUMENTATION

E.1 MC68000 Instruction Set Summary

Table 32:

Integer Arithmetic Operations

Instruction	Operand Size	Operation
ADD	8, 16, 32	Dn + (EA) → Dn (EA) + Dn → EA (EA) + #xxx → EA
	16, 32	An + (EA) → An
ADDX	8, 16, 32	Dx + Dy + X → Dx
	16, 32	Ax@ - Ay@ - + X → Ax@
CLR	8, 16, 32	0 → EA
CMP	8, 16, 32	Dn - (EA) (EA) - #xxx Ax@ + - Ay@ +
	16, 32	An - (EA)
DIVS	32 ÷ 16	Dn/(EA) → Dn
DIVU	32 ÷ 16	Dn/(EA) → Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 * 16 → 32	Dn * (EA) → 32 Dn
MULU	16 * 16 → 32	Dn * (EA) → Dn
NEG	8, 16, 32	0 - (EA) → EA
NEGX	8, 16, 32	0 - (EA) - X → EA
SUB	8, 16, 32	Dn - (EA) → Dn (EA) - Dn → EA (EA) - #xxx → EA
	16, 32	An - (EA) → An
SUBX	8, 16, 32	Dx - Dy - X → Dx Ax@ - - Ay@ - - X → Ax@
TAS	8	(EA) - 0, 1 → EA [7]
TST	8, 16, 32	(EA) - 0

NOTE
( ) = bit number

Logical Operations

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn ∧ (EA) → Dn (EA) ∧ Dn → EA (EA) ∧ #xxx → EA
OR	8, 16, 32	Dn ∨ (EA) → Dn (EA) ∨ Dn → EA (EA) ∨ #xxx → EA
EOR	8, 16, 32	(Ea) ⊕ Dy → EA (EA) ⊕ #xxx → EA
NOT	8, 16, 32	~ (EA) → EA

NOTE:
~ = invert

System Control Operations

Instruction	Operation
Privileged	
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
MOVE USP	Move user stack pointer
ORI to SR	Logical OR to status register
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
Trap Generating	
CHK	Check register against bounds
TRAP	Trap
TRAPV	Trap on overflow
Status Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
MOVE SR to EA	Store status register
ORI to CCR	Logical OR to condition codes

CC — carry clear
CS — carry set
EQ — Equal
F — Never True
GE — Greater or Equal
GT — Greater Than
HI — High
LE — Less or Equal
LS — Lower or same
LT — Less Than
MI — Minus
NE — Not Equal
PL — Plus
T — Always True
VC — No Overflow
VS — Overflow

Program Control Operations

Instruction	Operation
Conditional	
Bcc	Branch conditionally (14 conditions), 8- and 16-bit displacement
DBcc	Test condition, decrement and branch 16-bit displacement
Scc	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always, 8- and 16-bit displacement
BSR	Branch to subroutine, 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

Shift and Rotate Operations

Instruction	Operand Size	Operation
ASL	8, 16, 32	X:C ← 0
ASR	8, 16, 32	→ X:C
LSL	8, 16, 32	X C ← 0
LSR	8, 16, 32	0 → X:C
ROL	8, 16, 32	C ←
ROR	8, 16, 32	→ C
ROXL	8, 16, 32	C ← X ←
ROXR	8, 16, 32	→ X → C

Bit Manipulation Operations

Instruction	Operand Size	Operation
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z ~bit of (EA) → bit of EA

Binary Coded Decimal Operations

Instruction	Operand Size	Operation
ABCD	8	$D_{x10} + D_{y10} + X \rightarrow D_x$ $A_{x@-10} + A_{y@-10} + X \rightarrow A_{x@}$
SBCD	8	$D_{y10} - D_{y10} - X \rightarrow D_x$ $A_{x@-10} - A_{y@-10} - X \rightarrow A_{x@}$
NBCD	8	$0 - (EA)_{10} - X \rightarrow EA$

Instruction	Operand Size	Operation
MOVEP	16, 32	d(An) → Dn Dn → d(An)
MOVEQ	8	∅xxx → Dn
PEA	32	EA → −(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	(An) → Sp (SP)+ → An

NOTES:

s = source
d = destination
[ ] = bit numbers
−( ) = indirect with predecrement
( )+ = indirect with postdecrement
∅ = immediate data

Instruction	Operand Size	Operation
EXG	32	$R_x \leftrightarrow R_y$
LEA	32	EA → An
LINK	–	(An) → −(SP) (SP) → An (SP) + displacement → SP
MOVE	8, 16, 32	$(EA)_s \rightarrow EA_d$
MOVEC	32	(Rn) → Cr (Cr) → Rn
MOVEM	16, 32	(EA) → An, Dn (An, Dn) → EA
MOVES	8, 16, 32	(EA) → Rn (Rn) → EA

ASCII REFERENCE TABLE

	00	10	20	30	40	50	60	70
00	NUL	DLE	SP	0	@	P	`	p
01	SOH	DC1	!	1	A	Q	a	q
02	STX	DC2	"	2	B	R	b	r
03	ETX	DC3	#	3	C	S	c	s
04	EOT	DC4	$	4	D	T	d	t
05	ENQ	NAK	%	5	E	U	e	u
06	ACK	SYN	&	6	F	V	f	v
07	BEL	ETB	'	7	G	W	g	w
08	BS	CAN	(	8	H	X	h	x
09	HT	EM	)	9	I	Y	i	y
0A	LF	SUB	*	:	J	Z	j	z
0B	VT	ESC	+	;	K	[	k	{
0C	FF	FS	,	<	L	\	l	|
0D	CR	GS	-	=	M	]	m	}
0E	SO	RS	.	>	N	^	n	~
0F	SI	US	/	?	O	_	o	DEL

E.2 MC68000 Cross Software Support

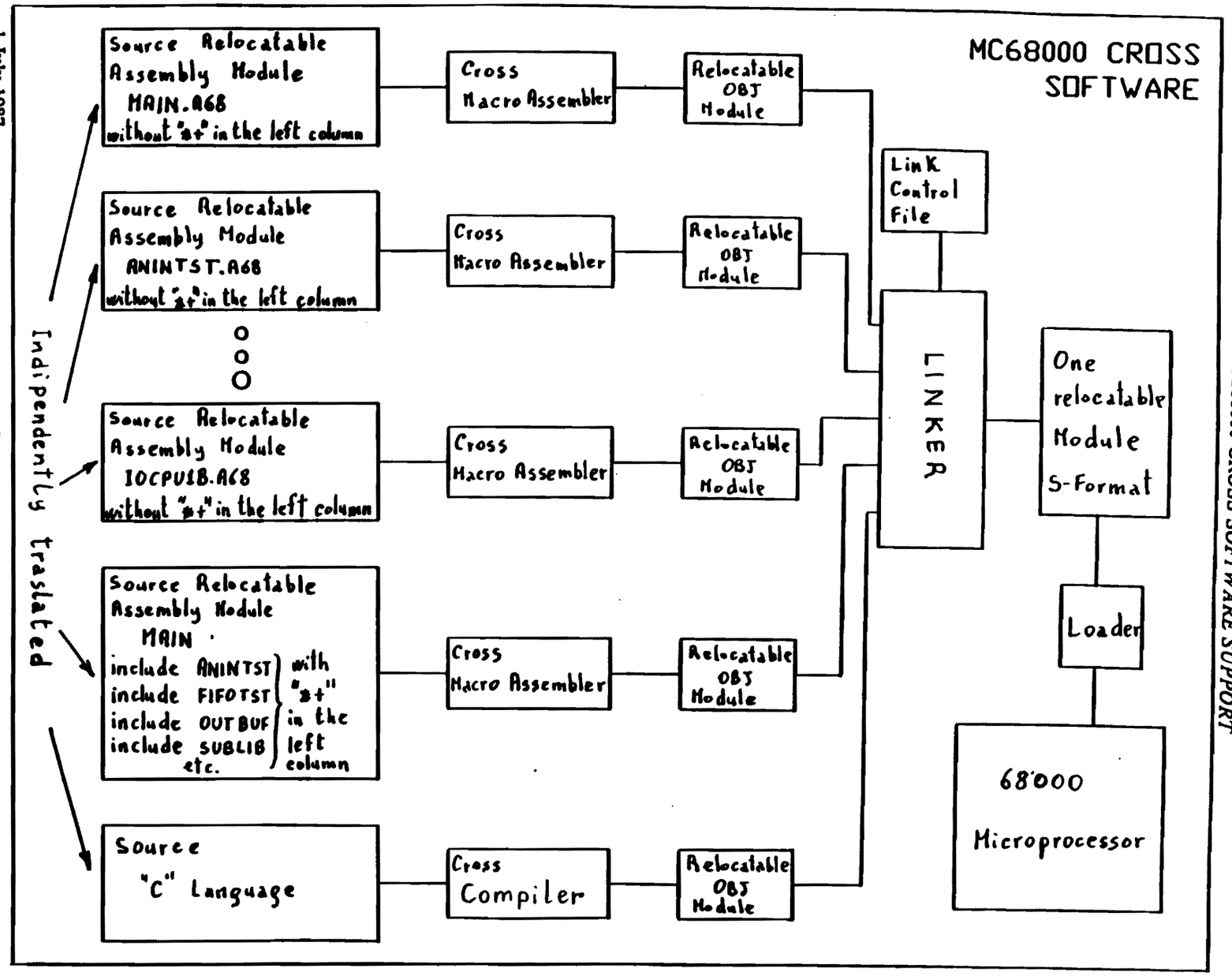
Most of the software is written in MC68000 Assembly language. The crossassembler used are: Lattice, Quelo, MC68mil (CERN). Some routines are forseen to be called also from High Level Languages, in particular RTF, for the physics applications, and "C" from Quelo and Lattice, for monitoring and debugging.

All the code is position indipendent to facilitate future installation in the framework of a Resident Operating System with the possibility of using System Calls.

In Table 33 a typical flow of cross software operations is shown, to go from different source files to one relocatable object module in Motorola S Format.

The source code can be in High Level Language or a module in assembly code. The module can consist of a set of routines which can either be independently assembled or included as part of another module by commenting a few lines of source code (those indicated with "*+" in the listing, see e.g. listing given in Appendix C).

Table 33: MC68000 CROSS SOFTWARE SUPPORT



REFERENCES

[1] D. Bulfone, D. Crosetto, F. Fratnik, L. Lanceri, E. Menichetti, G. Rinaudo: Proposal for the FEMC Trigger; DELPHI 86-100 DAS-42

[2] VME-BUS specifications

[3] VMX-BUS specifications

[4] H. Von Eicken: Software support for Motorola 68000 Microprocessors at CERN, M68MIL Cross Macro Assebler; CERN DD 83-12.

[5] MC68000UM(AD3) Motorola edition: 16-bit microprocessor user's manual Motorola Edition.

[6] Monolithic Memories Inc.: PAL programmable array logic handbook

[7] TEXAS INSTRUMENTS: TMS32010 User's Guide, 16/32-bit Digital Signal Processor; Edited by Paul Strzelecki, B. Sc.

[8] A. V. Oppenheim and R. W. Schafer: Digital Signal Processing; Prentice-Hall, Englewood Cliffs, N. J., 1975.

[9] DELPHI Thecnical Proposal, CERN/LEPC/83-3
DELPHI Progress Report, CERN/LEPC/84-16
DELPHI Progress Report, CERN/LEPC/85-33

[10] G. Baricello et al. A low-noise amplification chain for the vacuum photo-triode read-out of DELPHI lead glass calorimeter; DFPD 86-3.

[11] F. Udo. Proposal for the first level trigger. DELPHI 84-31 DAS-2
F. Udo. The second level trigger of Delphi; DELPHI 84-35 DAS-3
S.O. Holmgreen et al. The first and second level trigger; DELPHI 87-12 DAS-48.

[12] In preparation by A. Werbrouck: Cluster finding with DSP Dipartimento di informatica Universita' di Torino Italy.

[13] D. Crosetto, E. Menichetti, G. Rinaudo: Proposal for a second level trigger based on straw tube signals. FERMI-LAB Exp. E760 March 1987

[14] G. Eckerlin, et al. Parallel data analysis in a multichannel Flash ADC system; HD-PY-87/2.

[15] G. Eckerlin, et al.. 'Front End Processing for a 100 MHz Flash ADC system; HD-PY-86/3.

[16] W. Farr, et al. 'Readout of drift chambers with 100 MHz Flash ADC system. IEEE Transaction on Nuclear Science, Vol. NS-30, No. 1, February 1983.

[17] LATTICE MC68000 Cross Macro Assembler and 'C' compiler for MS-DOS operating system.

[18] QUELO MC68000 Cross Macro Assembler and 'C' compiler for MS-DOS operating system.

INDEX

ABORT ... 26
Analog Input board for FEMC – DELPHI trigger. ... 61
ANIN1 BOARD ... 61
Application example to calorimeters (FEMC – DELPHI) ... 42
Application example to straw tubes. ... 48
APPLICATION EXAMPLES ... 38
Application examples to drift chambers ... 50
Application program monitoring facilities. ... 4
Asynchronous communications ... 10

BACK – PLANE CONNECTORS ... 29

Command Register (CMD). ... 17
Crate Controller communication with dual port Data memory ... 10
Crate Controller communication with dual port Program memory ... 10
Crate Controller communication with TMS32010 through Out – FIFO's. ... 12
Crate Controller with TMS32010 through MAIL – BOX registers. ... 11

Development and test. ... 47
Digital Signal Processor (DSP). ... 7
DSP limitations. ... 7

Example of hit center of gravity and total energy calculation ... 50
Examples of different personalized ANIN boards. ... 38

FDDP – A modular system unit ... 38
FDDP connectors to piggy – back ANIN boards ... 32
FDDP connectors to piggy – back FIFO board ... 35
FDDP System Configuration for Development and Applications ... 41
FEATURES ... 5
FEMC – DELPHI trigger monitoring facilities. ... 47
FIFO board ... 20
FRONT PANEL CONNECTORS ... 26

Guidelines to configure a FDDP system for drift chamber analysis. ... 55

Hardware ... 4

Index ... 6
Interrupts to the Crate Controller. ... 12
Interrupts to the TMS32010. ... 14
INTRODUCTION ... 2

JUMPER SELECTION ... 25
J1/J16 – MEMORY MAP – SELECT ... 25

LOGICAL UNIT ... 6
Logical Unit communication with Crate Controller ... 10
Logical Unit communication with External Input Devices ... 14
Logical Unit communication with External Output Devices. ... 15
Logical Units Intercommunication ... 10

MC 68000 Assembly routines ... 71
MC68000 Cross Software Support ... 102
MC68000 display and command interpreting routines for TMS3210 monitorin ... 75
MC68000 display and command interpreting routines for TMS3210 tracing. ... 74
MC68000 documentation ... 100
MC68000 Instruction Set Summary ... 100
MC68000 interrupt routine (for TMS32010 tracing and monitoring). ... 71
MC68000 test & application example routines. ... 76
Memory Map & I/O Functions ... 15

ON BOARD CONNECTORS ... 32

PL0A & PL0B "Unclock of Out – FIFO's" signals ... 26
PL1A/PL8A & PL1B/PL8B – ANALOG INPUTS (ANIN1 board) ... 61
PL9A & PL9B "Output Ready (OR) of Out – FIFO's" signals ... 26
P1 "VME backplane connector" signals. ... 29
P2 "VME backplane connector" signals ... 30
P3A "16 – bit Out – FIFO's" signals ... 27
P6H & P7H – ANALOG INPUTS (ANIN1 board) ... 62