CERN 84-03
Experimental Physics Division
21 March 1984

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

BST – PINK PANTHER
An Intelligent CAMAC Crate Controller

Dominik A. Tröster

GENEVA
1984

CERN - Service d'Information scientifique - RD/628 - 1300 - Mars 1984.

ABSTRACT

A technical and functional description of the PINK system for intelligent, distributed data acquisition, data formatting, and data reduction is presented.

The system has been developed to bypass some of the constraints of CAMAC when collecting data with the high-resolution π^0 spectrometers of the Basel-Stockholm-Thessaloniki (BST) Collaboration at CERN.

CONTENTS

FREQUENTLY USED ABBREVIATIONS

ADC	Analog-to-Digital Converter
ALU	Arithmetic Logic Unit
ASM	Algorithmic State Machine
BRA	Branch (Module)
BST	Basel-Stockholm-Thessaloniki Collaboration
BSY	Busy (Output Signal)
CAMAC	Computer-Aided Measurement And Control
CE	Card Enable
CLRBUSY	Clear Busy
CMD	Command (Module)
CS	Command Strobe
CSC	Clear Slave Crate
DMA	Direct Memory Access (Data Transfer)
DTA	Data (Module)
DTL	Diode Transistor Logic
ECL	Emitter-Coupled Logic
EEN	Examine Event Number
EPP	Examine Program Pointer
ERR	Error (LED)
FFS	FIFO Strobe
FIFO	First-In/First-Out (Memory)
GOBRA	GOTO or Branch
HSH	Handshake (LED)
IC	Integrated Circuit
ICC	Intelligent Crate Controller
IDL	Idle (LED)
INCA	Increment Address
INIT	Initialize
ISC	Initialize Slave Crate
LAM	Look At Me
LED	Light-Emitting Diode
LSB	Least Significant Bit
MSB	Most Significant Bit
NAF	Station Number, Subaddress, and Function Code (Module)
NIM	Nuclear Instruments and Methods (Logic Standard)
OC	Open Collector (A TTL Option)

OS	Operating System
PFD	Prepare FIFO Dump
PGM	Program (Mode and Command to enter it)
PON	Power On
RAM	Random Access Memory
RFF	Read FIFO
ROM	Read Only Memory
RPP	Read Program at Pointer
RUN	Run (Mode and Command to enter it)
SEN	Set Event Number
SPP	Set Program Pointer
TDC	Time-to-Digital Converter
TTL	Transistor-Transistor Logic
VALCO	Valid Command
VAX	Trademark of Digital Equipment
WALAM	Wait for Any LAM
WE	Write Enable
WMTR	Wait for Master Trigger
WPP	Write Program at Pointer
WTLAM	Wait for This LAM

1. INTRODUCTION

1.1 Concept

Standard equipment in high-energy physics experiments is usually based on CAMAC. Although CAMAC means "Computer-Aided Measurement And Control", it is not at all adapted to a modern computer system with Virtual Address extension such as a Digital VAX11/750.
There are many CAMAC controllers commercially available, but either they are not flexible enough or they are too slow to cope with all the requirements mentioned below which may be specific to the PS182 experiment. Therefore, a new controller had to be developed. The concept behind the BST-PINK PANTHER is to have a device capable of sampling data independently of the main computer, which would then be free to carry out more rewarding tasks than collecting data across a CAMAC system. The list of requirements is straightforward:

- Random programmability for flexible response to changes in equipment, trigger conditions, etc.
- Fast operation.
- Delivery of a ready-mixed event structure that can be written to magnetic tape with no need of additional processing; this means less dead-time for the computer and similar programs for on-line control and off-line data evaluation.
- Handshake lines for easy interface to the experiment, and comprehensive status display.
- In an extended system with many crates: parallel operation to cut readout time.
- Buffering of several events in a memory: one DMA (Direct Memory Access) transfer from CAMAC to VAX for some dozen events to hundreds will allow readout time to be optimized.
- Readout of only relevant data with minimum overhead.

These goals have been reached with ease.

1.2 Realization

For reasons that will be explained in Section 7, it is clear that the microprocessor or bit-slice approach had to be dropped. To build random logic (conventional, "classical") was no way out, because of the risk of hang-up. Therefore, a dedicated microprocessing device had to be created. In addition, a comprehensive programming language was created and a compiler written for it.

The device is a dedicated, microprogrammable machine with horizontal microword structure. This means that it does everything related to taking one data item in one single command. In addition, pedestal subtraction is done, if desired, at the same time. The data buffer holds up to 64K words of 16 bit each at a time. Since the machine is an interface between CAMAC and CAMAC (yes!), it had to be split into two devices which are linked by a multipole bus. The economy and reliability of this 2 × 25-pole cable have to be compared with those of the traditional CAMAC 132-pole cable. A distance of up to 10 m can be bridged with no problem.

The two devices are characterized as follows: the first is the interface CAMAC/ICC plus the data buffer; the buffer is of the FIFO type (First-In-/First-Out). The second is the intelligent crate controller, referred to as the ICC, with its programmability (see Fig. 1). The system is called PINK PANTHER because of its colour and speed.

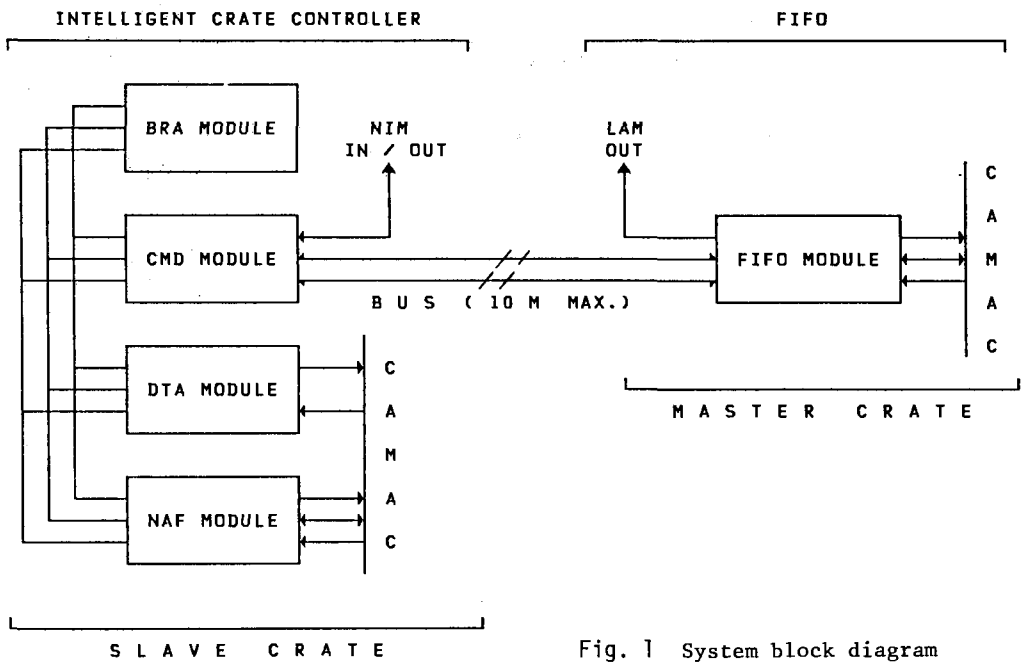

Fig. 1 System block diagram

The devices consist of five distinct modules:

- FIFO : First-In/First-Out.
- NAF : Station Number, Subaddress, and Function Code.
- DTA : Data.
- CMD : Command.
- BRA : Branch.

Figure 1 shows their relation. The modules are described in some detail in the next chapter. The functional description of the system is given in Section 5.

Programs which have to run on the BST-PINK PANTHER are written on the host computer and can have a maximum length of 2047 microwords. They are downloaded into the ICC and started by means of special utility programs. A comprehensive, high-level programming language has been defined in order to make full use of the BST-PINK PANTHER. This is called ICCOLA (Intelligent Crate COntroller LAnguage), and a compiler for it exists. It is written in FORTRAN 77 and implemented on a VAX11/750 [1].

It is possible to make a tree structure: one ICC controls several FIFOs. This boosts the data-taking. Redundancy is provided in the event structure in order to detect and trace transmission errors. A typical application, but on a small scale, can be found in Section 6.

The FIFO is two CAMAC stations wide, and the ICC is three stations wide. The power consumption density is five times less than for commercial equipment; this enhances the reliability of the system. The ICC has a power consumption of 15 W; the FIFO needs 10 W. A 200 W CAMAC crate has sufficient power to supply as many ICCs and FIFOs as you can put in it. However, only the combination with one ICC at the far right of the crate (the rest being FIFOs) makes sense -- or less FIFOs of course, if this is enough.

A knowledge of CAMAC standard[2-4] is assumed for the reading of this report.

2. TECHNICAL DESCRIPTION OF THE MODULES

2.1 FIFO Module

The expression FIFO characterizes a type of memory. Another expression for this concept is "Circular Buffer". The main characteristic of the FIFO is the fact that data are output in the same sequence as they have been put in. Another important aspect of the FIFO is its flexibility, since data can be written into and read out of it almost independently, provided that the memory space is sufficient. It is mostly used to adapt devices with high intermittent data rates to devices with lower but continuous ones.

A FIFO can be realized in different ways.

- There is the pure software approach that many computers use in their OS (Operating System). A FIFO of this kind can handle data rates up to some 10^4 words per second depending on the interrupt response of the computer. The size of this FIFO is only limited by practical considerations, usually some 10^3 words.
- There is the pure hardware approach used for very high data rates (10^7 words per second) but usually short sequences (some 10-200 words), since the amount of hardware is proportional to the storage capacity.
- And finally, there is the firmware approach. A dedicated FIFO controller uses standard RAM (Random Access Memory). Since, for a given access time, RAM capacity is (still?) growing from year to year, it becomes very easy to realize fast (2×10^6 words per second) FIFOs of large (2^{16} words) size. If necessary, the size could be increased by a factor of 16 without severe loss of speed.

The FIFO as realized in the BST-PINK PANTHER is of the third type (see Fig. 2). Besides the mandatory Read and Write signals, it accepts some CAMAC commands. The FIFO can be defined

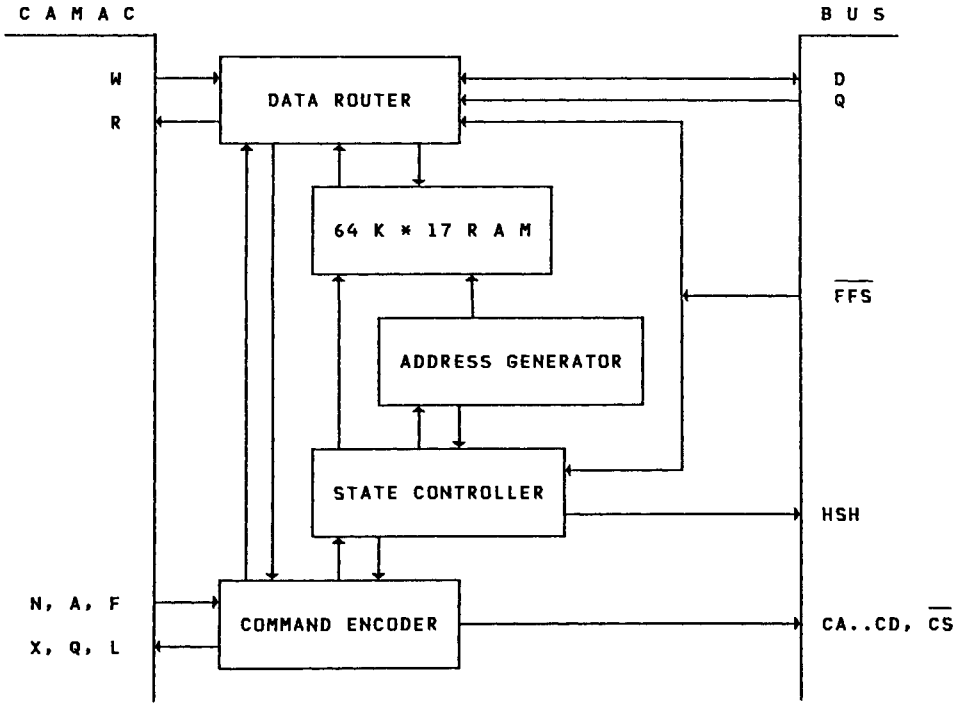

Fig. 2 FIFO block diagram

as being full in order to dump it. This can be useful for retrieving erroneously read data, since they are usually lost when a FIFO is read. The FIFO can be defined to be empty in order to get only fresh data after a change in the system.

Some characteristic problems of the FIFO concept cannot be resolved. If the FIFO is empty, further attempts to read it will only reproduce the last data read. After PON (Power ON), this is random data. At PON, the FIFO is automatically defined to be empty; however, there is a CAMAC command for testing whether it is empty or not.

If the FIFO is neither full nor busy writing, a handshake signal is asserted. This is required for further data input.

The details are discussed in subsection 3.1.

2.2 NAF module

The NAF module contains all the hardware related to the addressing of CAMAC, including the Function code to be executed (see Fig. 3). On the other hand, it generates control signals for the CMD module according to the state of some CAMAC lines (e.g. LAM).

It also generates all the necessary timing for CAMAC. For details, see subsection 3.2.

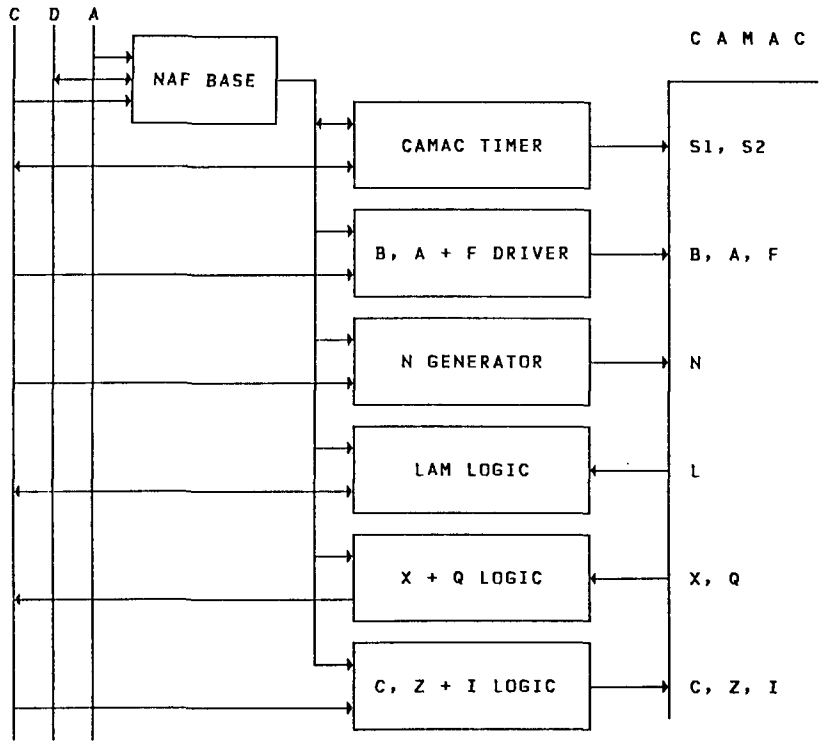

Fig. 3 The NAF module block diagram

2.3 DTA module

The DTA module contains all the hardware necessary for data manipulation, including logical tests (see Fig. 4 for the block diagram). It is able to subtract pedestals from data read [e.g. ADCs (Analog-to-Digital Converters)] with no additional delay, and it is used to write data into CAMAC. For details, see subsection 3.3.

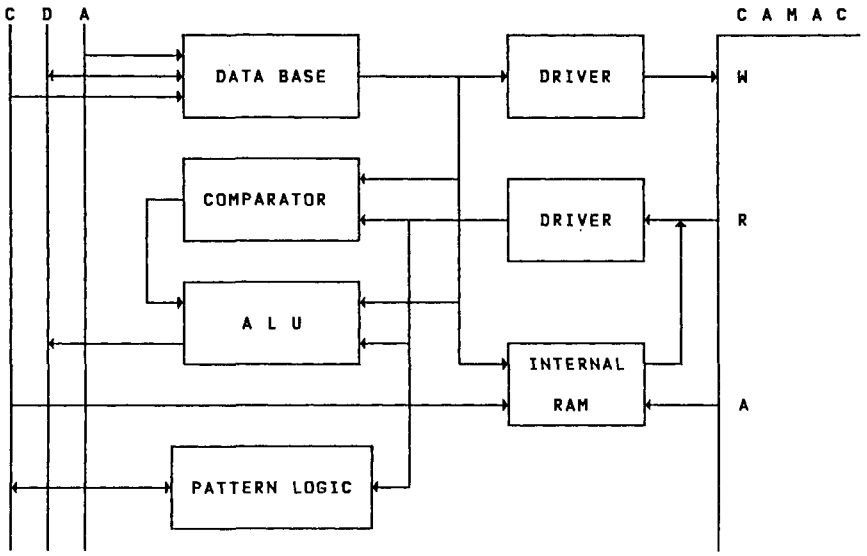

Fig. 4 DTA module block diagram

2.4 CMD module

The CMD module contains a microprogrammed controller to interpret and execute the program loaded previously.

Figure 5 shows the block diagram. The behaviour of the CMD module is influenced by the NAF module as well as by the FIFO. It can also sense four NIM inputs. Furthermore, it contains the hardware used for event formatting. Additional information can be found in subsection 3.4.

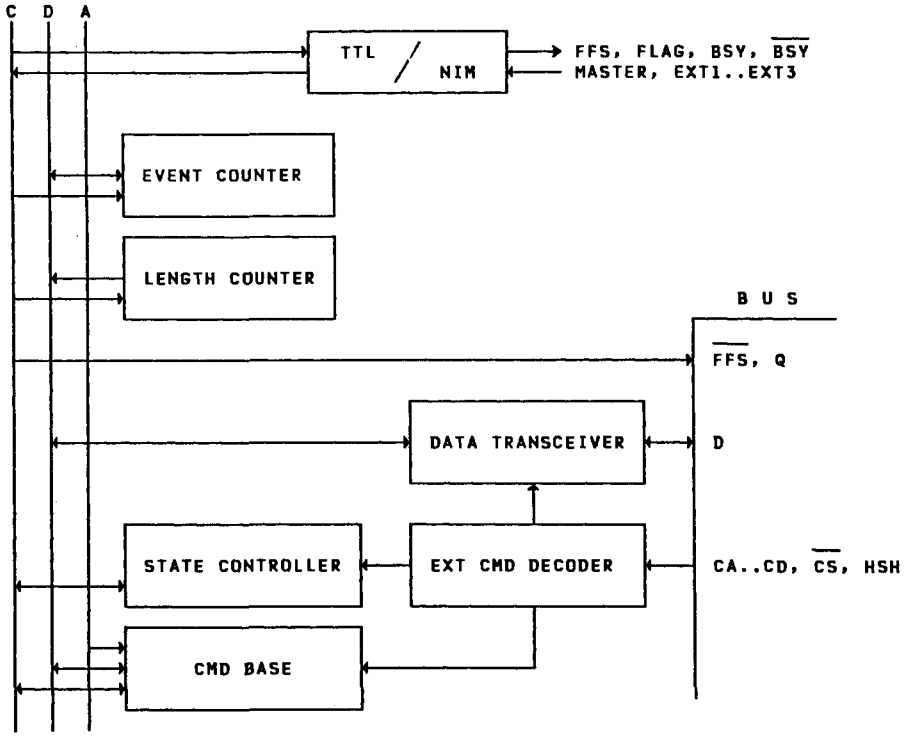

Fig. 5 The CMD module block diagram

2.5 BRA module

The BRA module (Fig. 6) contains all the hardware needed for generating the system address, including GOTO and IF ... THEN ... (ELSE ...). In PGM (Program) mode it is used to select the module which is to be programmed, by setting the two most significant address bits. In RUN mode all modules are enabled. For more information, see subsection 3.5.

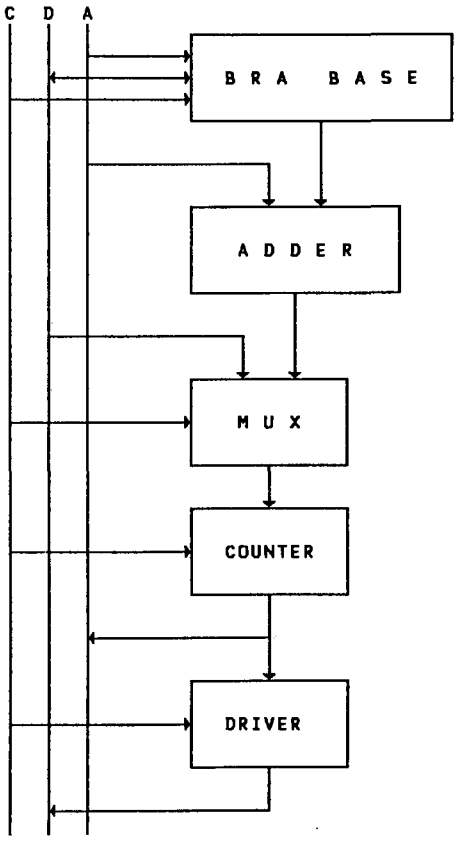

Fig. 6 The BRA module block diagram

3. TECHNICAL AND FUNCTIONAL DESCRIPTION OF THE ELEMENTS[1)]

3.1 FIFO module

In the following subsections, the constituents of the FIFO module are presented in more detail.

3.1.1 Command encoder

The command encoder performs the translation between CAMAC commands and ICC commands. The Function (F)-lines and the Subaddress (A)-lines are fully decoded, which means that all the bits are used. The F/A combinations that are meaningful are detected by OR-gates. They are listed in Table 1. They are encoded by a priority encoder to form ICC commands, which are transmitted through the bus (see subsection 3.6) if they are allowed in the current system mode [RUN or PGM]. The CS (Command Strobe) is generated to be either long or short, depending on the command type. This is shown in Fig. 7. CS(A) is done for RPP (Read Program at

Table 1

CAMAC functions defined for the Command Encoder

• Concerning the FIFO

F(0)	A(0)	read FIFO
F(0)	A(7)	prepare FIFO dump
F(8)		test LAM
F(9)	S2	clear FIFO and LAM
F(10)		clear LAM
F(24)		disable LAM
F(26)		enable LAM

• Concerning the ICC

		Mnemonic	Description
F(0)	A(1)	RPP	Read Program at Pointer
F(1)	A(1)	EPP	Examine Program Pointer
F(1)	A(7)	EEN	Examine Event Number
F(16)	A(1)	WPP	Write Program at Pointer
F(17)	A(1)	SPP	Set Program Pointer
F(17)	A(7)	SEN	Set Event Number
F(17)	A(12)	ISC	Initialize Slave Crate
F(17)	A(13)	CSC	Clear Slave Crate
F(17)	A(14)	PGM	Enter Program Mode
F(17)	A(15)	RUN	Enter Run Mode

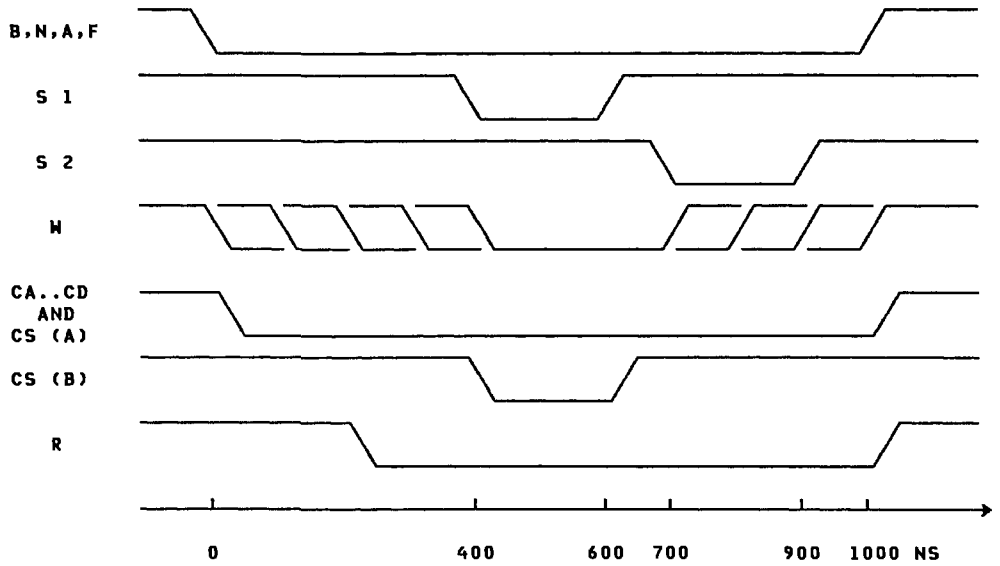

Fig. 7 Timing diagram of the command encoder

Table 2

CAMAC response of the Command Encoder

L:	LAM is active if enabled and FIFO more than half full.
X:	An X-response is generated if the command is accepted.
Q:	A Q-response is generated if there is - either F(8) and LAM active, - or any other F and not an End-Of-Event condition, and the FIFO is not empty.

Pointer), EPP (Examine Program Pointer), and EEN (Examine Event Number); CS(B) for all the other commands. Commands for the data router are simultaneously generated. Upon PON, the command encoder is switched to PGM mode, and an INIT (Initialize) signal is sent to the FIFO controller. The LAM (Look-At-Me), the X-response, and the Q-response are generated according to the conditions in Table 2. Two TTL (Transistor-Transistor Logic) signals are available on the front panel: LAM and "EARLY"; the former is active (low) if the FIFO is more than half full, the latter if more than a jumper-selectable number of words are stored in the FIFO. However, the FIFO can be read out whenever it is not empty.

3.1.2 FIFO state controller

The FIFO state controller consists mainly of an ASM (Algorithmic State Machine). A description of the ASM can be found in the Appendix. In order to facilitate the use of the ASM, the states are decoded by demultiplexers. AND gates and flip-flops generate timing and control signals. The state diagram is shown and other details are discussed in the Appendix. The data transfer signals FFS (FIFO Strobe) and RFF (Read FIFO) trigger on the raising (i.e. trailing) edge and are synchronized, whilst INIT and PFD (Prepare FIFO Dump) are asynchronous. (For details, please refer to Fig. A.3.) A Write cycle is done as soon as data have been written into the input latch if the FIFO is not full, and a Read cycle is done as soon as the output latch has been read if the FIFO is not empty. The clock generator consists of a nine-stage ring oscillator. Two ICs (Integrated Circuits) are used. They can be exchanged to vary the speed according to memory access time and the throughput need, as well as the desired safety. The obtainable speed range is shown in Table 3.

3.1.3 FIFO address generator

The FIFO address generator contains two counters, the first one for the Write pointer, and the second one for the Read pointer. The pointer required is selected and latched, and within half an ASM clock cycle it is incremented. The latched address is sent through bus drivers to the FIFO RAM. On the other hand, the new difference between the pointers is calculated. If the difference is zero, the FIFO is either full or empty. This signal is defined through the last operation: if it arrives during a Write operation, then the FIFO is full; empty is sensed if the signal activates during a Read operation. The most significant bit of the difference is used to generate a LAM (meaning that half the available space is used). A gate in DTL (Diode Transistor Logic) senses the seven most significant bits of

Table 3

Cycle time range of the ASM clock generator

Time (ns)	IC 1	IC 2
121	74N00	74N04
106	74LS00	74N04
97	74S00	74N04
93	74N00	74LS04
89	74F00	74N04
82	74N00	74S04
77	74LS00	74LS04
70	74N00	74F04
67	74S00	74LS04
65	74LS00	74S04
59	74F00	74LS04
55	74S00	74S04
53	74LS00	74F04
48	74F00	74S04
42	74S00	74F04
34	74F00	74F04

the difference plus "FULL" in order to announce that more than 512 data items are present (as an early LAM). This feature is particularly useful when running with a low event rate.

3.1.4 Data router

The data router is a switchboard built of bus drivers and latches. The drivers are needed to interface the unidirectional CAMAC data Read (R) and Write (W) lines to the bidirectional bus, whereas the latches are complementary devices to the FIFO RAM. The FIFO controller keeps the input latch free of data if possible, i.e. the FIFO is not full, and it writes data into the output latch as soon as possible, i.e. when the FIFO is not empty. This procedure is chosen in order to minimize the response time of the FIFO.

3.1.5 FIFO RAM

The FIFO RAM is an all-purpose fast static RAM card in CAMAC outline. Figure 8 shows the layout of the connectors.

Its inherently high speed (80 ns approx.) makes it suitable for many applications requiring large space (64K words, 17-bit wide)*). Several cards may be wired in parallel in order to increase the size. However, care must be taken that only one card is selected at a time, for the current drive capability of the address drivers and WE (Write Enable) (0.7 mA per card, both high and low level), and for the decrease in speed due to the raise in capacitive loading of the lines.

*) 16 bits are used for data, one bit is used to mark the end of one event (see subsection 3.4.5).

GND	A 0
GND	A 1
GND	A 2
GND	A 3
GND	A 4
GND	A 5
GND	A 6
GND	A 7
GND	A 8
GND	A 9
GND	A 10
GND	A 11
GND	A 12
GND	A 13
GND	A 14
GND	A 15
GND	$\overline{\text{INIT}}$
GND	$\overline{\text{WD}}$
GND	$\overline{\text{WS}}$
GND	$\overline{\text{CE}}$

OUT	IN	15
OUT	IN	14
OUT	IN	13
OUT	IN	12
OUT	IN	11
OUT	IN	10
OUT	IN	9
OUT	IN	8
OUT	IN	7
OUT	IN	6
OUT	IN	5
OUT	IN	4
OUT	IN	3
OUT	IN	2
OUT	IN	1
OUT	IN	0
OUT	IN	Q
GND	GND	
GND	GND	
GND	GND	

CA	GND
CB	GND
CC	GND
CD	GND
$\overline{\text{CS}}$	GND
$\overline{\text{FFS}}$	GND
D 15	GND
D 14	GND
D 13	GND
D 12	GND
D 11	GND
D 10	GND
D 9	GND
D 8	GND
D 7	GND
D 6	GND
D 5	GND
D 4	GND
D 3	GND
D 2	GND
D 1	GND
D 0	GND
Q	GND
	GND
HSH	GND

Fig. 8 Connectors in the FIFO

3.2 NAF module

In the following subsections, the constituents of the NAF module are presented in more detail.

3.2.1 NAF base

The NAF base is built around a 2K × 16-bit RAM. This RAM can be accessed in PGM mode only; in RUN mode it is isolated from the internal data lines (see Fig. 9 for the basic concept). The RAM contains the full Station Number (N), Subaddress (A), and Function Code (F) in one single word, hence the name NAF (see also Fig. 13). Of course, care must be taken that N is only assigned a value in the range (1 ... 22) for CAMAC transfers to external stations. The value N = 0 is used for dummy CAMAC cycles, or if an unaddressed command is executed (initialize or clear). The value N = 23 accesses the internal RAM (see 3.3.3).

Whilst A and F concern CAMAC only -- they are sent to CAMAC while the CAMAC cycle is active -- the N information is used internally as well: for the internal RAM, for the LAM logic, and for the N generator. The two MSBs (Most Significant Bits) are not used and are reserved for future applications.

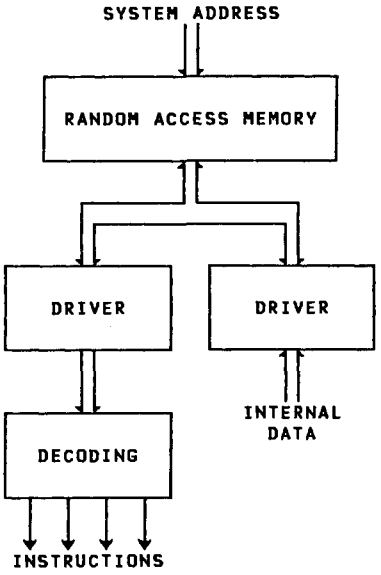

Fig. 9 The instruction base

3.2.2 CAMAC timer

The CAMAC timer is essentially a set of monostables. It can be programmed to provide a long (standard) CAMAC cycle, or a short (abbreviated) CAMAC cycle, or an internal cycle not involving CAMAC at all. See Fig. 10 for details.

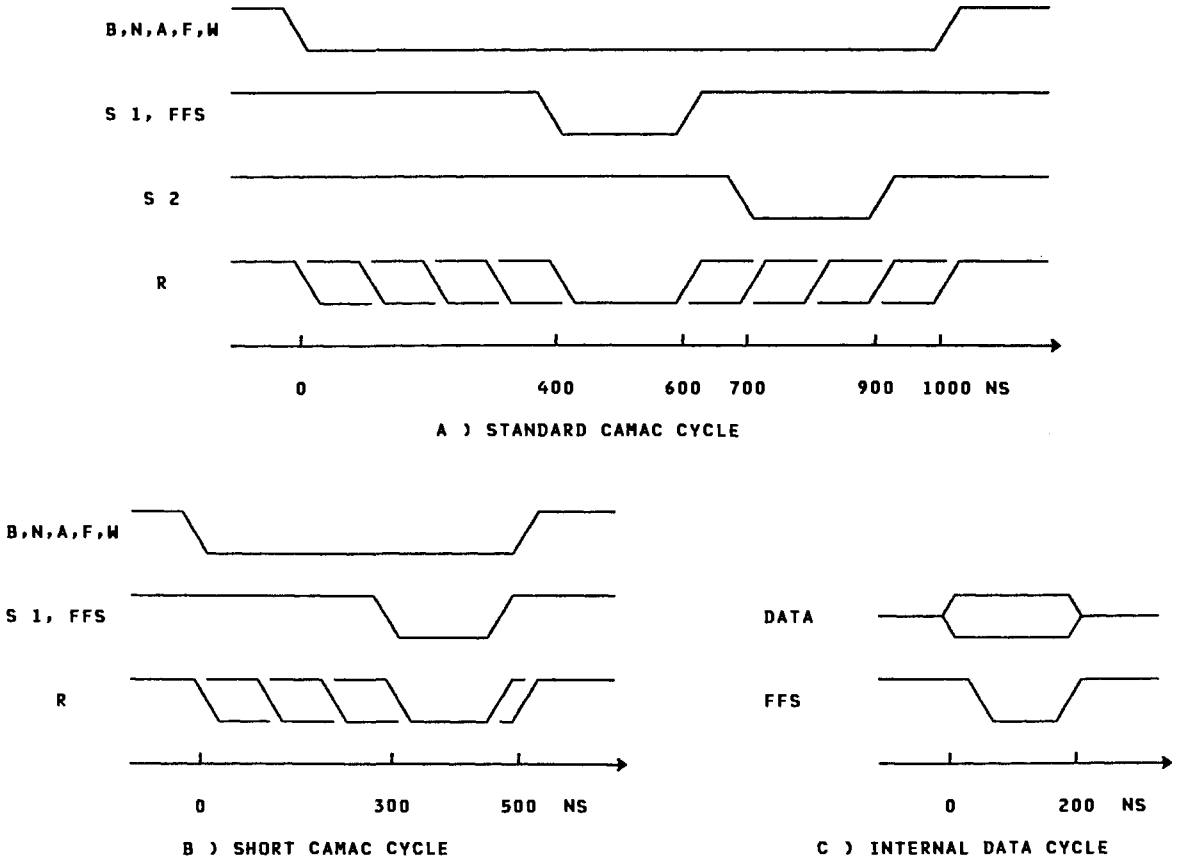

Fig. 10 CAMAC timer options

All the time values can be varied by turning CERMET multiturn potentiometers. Since the dynamic range is +100%/-50%, an optimum compromise between speed and reliability can be found. The short CAMAC cycle is available for operating the units which are fast enough and also for the commands not using the S2 strobe. Here, a compromise between speed and efficiency has to be found: the faster the cycle, the less the modules are able to respond correctly. S1 should not be adjusted below 120 ns since data might be lost on the way to the FIFO! However, the time can only be tuned when working with an extender.

3.2.3 B, A, and F driver

The B, A, and F driver sends these signals to CAMAC, provided that a CAMAC cycle is active. The B signal is just a copy of the CAMAC timer's output "CAMAC Busy"; A and F are the values given by the NAF base. The driver has OC (Open Collector) outputs according to CAMAC standard, and exchangeable pull-up resistors. These resistors (560 Ω) can be replaced by 330 Ω resistors in order to accelerate the trailing edge of the signals, but with a penalty in power consumption.

3.2.4 N generator

The N generator is a set of decoders, activating one output corresponding to the value of N. The decoders are enabled with a shortened B signal in order to fulfil the timing requirement of CAMAC, shown in Fig. 10. They have OC outputs according to CAMAC specifications, and exchangeable pull-up resistors.

3.2.5 LAM logic

The LAM logic provides two signals to the CMD module. The first signal is active if any LAM is active, i.e. at least one enabled CAMAC station wants to communicate. This signal is used to resume ICC operation when it has been waiting for any LAM (ICCOLA command: WALAM) [see the ICCOLA User's Manual[1)]]. The other signal is active only if the selected station gives a LAM. The current N value determines the station to be selected. This signal is used to resume ICC operation when it has been waiting for this LAM (ICCOLA command: WTLAM). Both signals can also be activated by front panel input MASTER (for details, see subsection 3.4.7).

At first sight it seems there is a contradiction between CAMAC standard and the above-mentioned operation mode: the standard defines that a selected station shall not activate its LAM line. It should rather give a Q-response when LAM is tested. However, since the CAMAC cycle has not yet started when WTLAM or WALAM are tested, the station does not know it is selected. This is one of the advantages of using an ASM. Thus the following codes are equivalent:

- FORTRAN

```
      IMPLICIT INTEGER (A-Z)
      CALL CDREG (EXT,1,2,5,8)
10    CALL CSSA (8,EXT,DATA,Q)
      IF (Q.EQ.0) GOTO 10
      CALL CSSA (0,EXT,DATA,Q)
```

```
BUFF(J) = DATA-38
IF (BUFF(J).LT.0)) BUFF(J) = 0
J = J+1
LENGTH = LENGTH+1
```

- ICCOLA

```
PED = 38  N = 5  A = 8  F = 0 WTLAM READ
```

3.2.6 X and Q logic

The X and Q logic consists of two flip-flops which latch the value of the CAMAC X and Q lines at the end of S1 of every CAMAC cycle. Before another CAMAC cycle is issued, the user has the possibility to test the state of these flip-flops.

3.2.7 C, Z, and I logic

The C, Z, and I logic can be activated in both PGM and RUN mode (see Table 4 for details). The I line is active when either C or Z is active. A long CAMAC cycle is generated automatically, irrespective of the mode.

Table 4

Clear and initialize CAMAC

PGM	RUN	Description
CSC	CAMCLR	Clear CAMAC Crate
ISC	CAMINI	Initialize CAMAC Crate

3.3 DTA module

In the following subsections, the constituents of the DTA module are presented in more detail.

3.3.1 DTA base

The DTA base is built around a 2K × 16-bit RAM. This RAM can be accessed in PGM mode only. In RUN mode it is normally isolated from the internal data lines (see Fig. 9 for the basic concept, and also Fig. 13). Its function covers a wide range, according to the command being executed: Firstly, in a CAMAC Read operation, the content of the actual memory location is defined to be a pedestal that has to be subtracted from the data read. Secondly, in a CAMAC Write operation, the content of the actual memory location is defined to be the word to be transferred into CAMAC. The only exceptions to this case are the ICCOLA commands OUTNUM and OUTLEN, which send the event number and the event length, respectively, into CAMAC (e.g. to a display unit). Thirdly, the command GOTO defines the content of the actual memory location to be the address to be loaded into the program pointer.

3.3.2 Pedestal subtraction

The pedestal subtraction is always done when data are read from CAMAC. In order to suppress this feature, the pedestal has to be set to zero. If the value of data read is less

than the pedestal, then the result will not be negative, but clear (zero). The comparison of data and pedestal, including the subtraction, is done within S1 of the CAMAC cycle by a dedicated ALU (Arithmetic Logic Unit) and comparators, thus without any increase in dead-time. The ERR LED (Light-Emitting Diode) indicates whether the pedestal is less than the data (green) or not (red).

3.3.3 Internal RAM

The internal RAM, for scratch pad use, is built of fast 16 × 4-bit TTL TAM ICs. It is accessed as station number 23 using long or short CAMAC cycles. Station N = 23 is one of the locations occupied by the ICC itself. The RAM responds to the CAMAC functions F = 0 (read) and F = 16 (write). The subaddresses A = 0 ... 15 can be used. The data source is either the DTA base, or the event number, or the event length. The memory behaves in the same way as any CAMAC register. However, neither the X-response nor the Q-response is asserted.

3.3.4 Pattern logic

In parallel with the pedestal subtraction, a latch is connected for data read. This latch offers several possibilities. It is loaded with the original data read if this is desired, i.e. without pedestal subtraction. This is no severe restriction, since it is supposed to help reduce the data stream using tests on pattern units. Each bit corresponds to one ADC: if the bit is set, then the ADC has to be read out. Arithmetic on this bit pattern is not believed to be meaningful.

Tests can be done on this latch:

- whether it is empty (all bits are clear) or not,
- whether a specific bit is set or not.

Whilst the test on empty is straightforward (see ICCOLA), the bit test is more subtle: a 4-bit counter selects, through a multiplexer, one of the 16 latched bits. This counter can be cleared via the command RESTORE, which does not affect the data latched, or via the command STORE, which simultaneously latches data read. The counter can be incremented by using the command NEXTBIT. The command LOAD latches data read without affecting the counter. The counter starts counting with the right-most bit, which is the LSB (Least Significant Bit) as well. It should be noted that data are inevitably written into the FIFO when they are latched.

3.4 CMD module

In the following subsections, the constituents of the CMD module are presented in more detail.

3.4.1 External command decoder

The external command decoder is controlled by the bus lines CA ... CD and CS, and by a flip-flop (see subsection 5.1). The flip-flop is used to remember the mode (RUN/PGM). In RUN mode, only the command PGM is accepted. The decoder generates the signals for the data transceiver as well. When loading or verifying a program, an INCA (INCrement Address) signal is generated, in order to achieve better performance (loading and reading can be done in DMA mode, i.e. as fast as CAMAC allows). PON sets the flip-flop to PGM state.

3.4.2 Data transceiver

The data transceiver is made of bidirectional three-state bus-drivers with Schmitt trigger inputs and a high fan-out. This is needed in order to compensate for the signal deterioration along the bus cable. Pull-up resistors are provided on both sides: 1.5 kΩ on the bus side to provide a defined state when the connection to the FIFO fails, and 1.5 kΩ on the ICC side to increase the speed and to provide a defined state when all the drivers connected are in high-impedance state.

3.4.3 CMD base

The CMD base is built around a 2K × 16-bit static RAM. The RAM can be accessed in PGM mode only; in RUN mode it is isolated from the internal data lines. In the latter mode, the content is interpreted by the state controller (see subsection 3.4.4 and Table 5). The C word indicates the CAMAC action and is always decoded. The J word, together with G and I bits, is decoded and interpreted simultaneously, but the result is strobed with state 7 (see next subsection). The outcome of this interpretation is used by the State Controller at state 8 only. The W word is used by the ASM at state 8 as well. The WAIT signal, however, is active not only if the ICC is actually waiting for a MASTER trigger or a LAM but also if the FIFO is either full, or is busy writing, or is not powered, or if the contact is lost (cable missing or interrupted). The E word is interpreted when VALCO (VALid COmmand) is active only, for safety. The NEXTBIT command sets a flip-flop and is executed while the next command is being fetched. At state 9, the following commands are executed:

- SETFLAG
- SETBUSY
- CAMINI
- CAMCLR
- If applicable, a CAMAC cycle is started.

At state B, the following commands are executed:

- CLRFLAG
- CLRBUSY
- The GOTO and BRANCH flip-flops are cleared, as well.

Internal and external commands concerning the event number and the event length are encoded by a priority encoder and sent via the front connector (see Fig. 11c) to the appropriate elements. The encoding is done in order to save lines.

Commands generating data to be stored in the FIFO are decoded and strobed with appropriate signals, thus activating the FFS. For instance, a CAMAC read operation is strobed with the S1 signal of the CAMAC timer to produce FFS (see Fig. 10).

3.4.4 State controller

This state controller is merely a copy of the one described in subsection 3.1.2. The differences are found mainly in its programming (for details, see the Appendix), and in the input and output lines. Since, in RUN mode, it is the master of the BST-PINK PANTHER system, no input line needs to be synchronized with flip-flops. It is easy enough to modify the program in order to adapt it to slower/faster RAM. There are even unused steps available.

a)

A 9	A 10
A 7	A 8
A 5	A 6
A 3	A 4
A 1	A 2
A 0	$\overline{\text{INCA}}$
R/$\overline{\text{W}}$	$\overline{\text{RUN}}$
$\overline{\text{NOMEM}}$	$\overline{\text{W}}$
$\overline{\text{C DTA}}$	$\overline{\text{C CMD}}$
$\overline{\text{NEXTBIT}}$	$\overline{\text{READ}}$
$\overline{\text{CLRCNTR}}$	$\overline{\text{BIT}}$
$\overline{\text{LOAD}}$	$\overline{\text{EMPTY}}$
D 14	D 15
D 12	D 13
D 10	D 11
D 8	D 9
D 6	D 7
D 4	D 5
D 2	D 3
D 0	D 1

b)

$\overline{\#\ B}$	GND
$\overline{\#\ 9}$	GND
$\overline{\#\ 8}$	GND
$\overline{\#\ 7}$	GND
$\overline{\#\ 6}$	GND
$\overline{\#\ 0}$	GND
$\overline{\text{GOBRA}}$	GND
$\overline{\text{SKIP}}$	GND
$\overline{\text{WAIT}}$	GND
$\overline{\text{BUSY}}$	GND
HSH	GND
$\overline{\text{EPP}}$	GND
$\overline{\text{SPP}}$	GND
CCA	GND
CCB	GND
$\overline{\text{CCS}}$	GND
$\overline{\text{FFS}}$	GND
NAF 9	GND
NAF 10	GND
NAF 11	GND
NAF 12	GND
NAF 13	GND
$\overline{\text{VALCO}}$	GND
$\overline{\text{GOTO}}$	GND
$\overline{\text{N(23)}}$	GND

c)

CA	GND
CB	GND
CC	GND
CD	GND
$\overline{\text{CS}}$	GND
$\overline{\text{FFS}}$	GND
D 15	GND
D 14	GND
D 13	GND
D 12	GND
D 11	GND
D 10	GND
D 9	GND
D 8	GND
D 7	GND
D 6	GND
D 5	GND
D 4	GND
D 3	GND
D 2	GND
D 1	GND
D 0	GND
Q	GND
	GND
HSH	GND

Fig. 11 Connectors in the ICC

The ASM basically works itself through the ASM chart (see Fig. A.4). At state 0 the new address is calculated. For details, see the description of the BRA Base (subsection 3.5). At state 6 the RAMs are believed to give stable information. The VALCO line is activated. During state 7 the command is decoded. At state 8 it is decided whether a further execution of the command is necessary or if the next command can be fetched. If the HSH (HandSHake) line is not asserted, the ASM will wait for it. The same is true if it is waiting for a Master Trigger or a LAM. State 9 starts the cycle, be it a long/short CAMAC cycle or an internal command. The CAMAC timer activates the BSY (Busy) line. During this time, the ASM loops at state A. After completion of the cycle, state B is done to de-activate some of the other elements, e.g. in the CMD module.

If the PGM line is activated, the ASM will go immediately and unconditionally into the state F (break). PON will produce the same result: the ICC will be halted. When PGM is removed, first the state E occurs, then the ASM resumes normal operation.

3.4.5 Length counter

This 16-bit counter is used for event formatting (see Fig. 12). It is set to 1 when a new event starts, i.e. the command HEADER is processed. It is incremented through the signal FFS, which is active at any data transfer to the FIFO and thus keeps pace with the actual event length. The length itself is transferred to the FIFO by the command LENGTH, and it is written into CAMAC by using the command OUTLEN. The occurrence of the command LENGTH means that the end of an event has been reached. This information is sent to the FIFO by holding the Q-line of the bus low. If the length is read out of the FIFO, no Q-response will be given, thus enabling a DMA readout in Q stop mode.

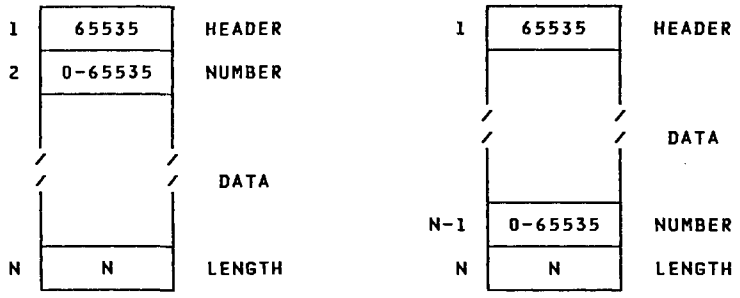

Fig. 12 Prototype structured event

3.4.6 Event counter

This 16-bit counter is used for event formatting. In PGM mode, the counter can be set (SEN) and examined (EEN). In RUN mode, it is incremented by the command HEADER, i.e. when a new event starts. The counter contains the actual event number. When it has an overflow, it continues with 0. In a system using several ICCs, this feature provides important redundancy in order to ensure data integrity. The event number is transferred to the FIFO by the command NUMBER. It can be transferred into CAMAC by using the command OUTNUM. The command NUMBER can be used anywhere in the program and as often as necessary.

3.4.7 TTL/NIM/TTL

The TTL/NIM/TTL converter performs several duties. Four input lines and four output lines exist. The first input line is called MASTER and is used in conjunction with the command WMTR (Wait for Master TRigger). This command causes the ASM to loop at state 8 until the input becomes active. The response time is of the order of 50 ns. The ASM will loop for an indefinite time if the MASTER signal is missing. For safe operation, however, the signal should not be shorter than 1 µs.

For the commands WTLAM (Wait for This LAM) and WALAM (Wait for Any LAM), the ASM will loop as well. However, if the LAM signal is late, it can be overridden by MASTER, thus providing a timeout feature.

The other three inputs EXT1 ... EXT3 are just switches which can be tested in the program. Here it is left to the user to provide a timeout or not. Signals applied should not be shorter than 1 µs, and should be used if possible with a handshake, using one of the outputs described below.

The four outputs have very different behaviour. The first one is called FFS, and is synchronous with the FFS on the bus. It can be used to monitor the data rate. The second output is called FLAG; it can be set and cleared under program control for handshake, or to generate test pulses, or to indicate special conditions. It is not affected by a change of the ICC mode (RUN/PGM).

The two remaining outputs are complementary. They are called BSY and $\overline{\text{BSY}}$, respectively. They can be set and cleared under program control in much the same way as FLAG. However, BSY is activated by switching to PGM mode in order to prevent further events from being generated. It is automatically set by the command HEADER for the same reason. The ICCOLA sequence for making use of this handshake is as follows:

```
CLRBUSY        * enables triggers to be generated
WMTR HEADER    * waits for trigger to start new event and to disable triggers
```

If MASTER is already active when the ASM is at state 8, operation will continue immediately, thus introducing no additional delay at all.

3.4.8 The LEDs

The four LEDs give some information about the system status. The HSH LED comes on when the FIFO is either full, or is busy writing, or is not powered, or if the bus cable is interrupted. When the ICC is transmitting data at a high rate, the LED can be seen to glow faintly. This has no influence on the transfer rate, since the ICC cannot generate data as fast as the FIFO can store them.

The RUN LED is red when in PGM mode and green when in RUN mode. It is red after PON.

The IDL LED is red when the ASM is idling; otherwise it is green.

The ERR LED is red when the pedestal is greater than the data read; otherwise it is green.

3.5 BRA base

The BRA base contains a 2K × 8-bit RAM. This base generates the address for the ICC. The RAM can be accessed in PGM mode only; in RUN mode it is isolated from the data lines (see Fig. 9 for the basic concept). The RAM contains a signed 8-bit value used to modify the current value of the program pointer (see Fig. 13).

The program pointer is a programmable counter for the RAMs of the four modules: NAF, DTA, CMD, and BRA. Its value can be modified in different ways:

- In PGM mode it can be set (SPP) and examined (EPP). An INCA signal is generated when writing (WPP) or reading (RPP) a program in order to allow a DMA transfer.
- In RUN mode it is incremented whenever the ASM is in state 0, except if it just starts operation (after PGM mode); if a GOTO is performed (the data on the internal bus, coming from the DTA module, are loaded into the counter); or if a BRANCH is done (the signed 8-bit value of the BRA base is added to the current address and the result is loaded into the program pointer).

The counter is 16 bits wide, of which 11 bits are used to access the 2K of address space. Only the 2 MSBs of the remaining 5 bits are used: in PGM mode, they select the module which

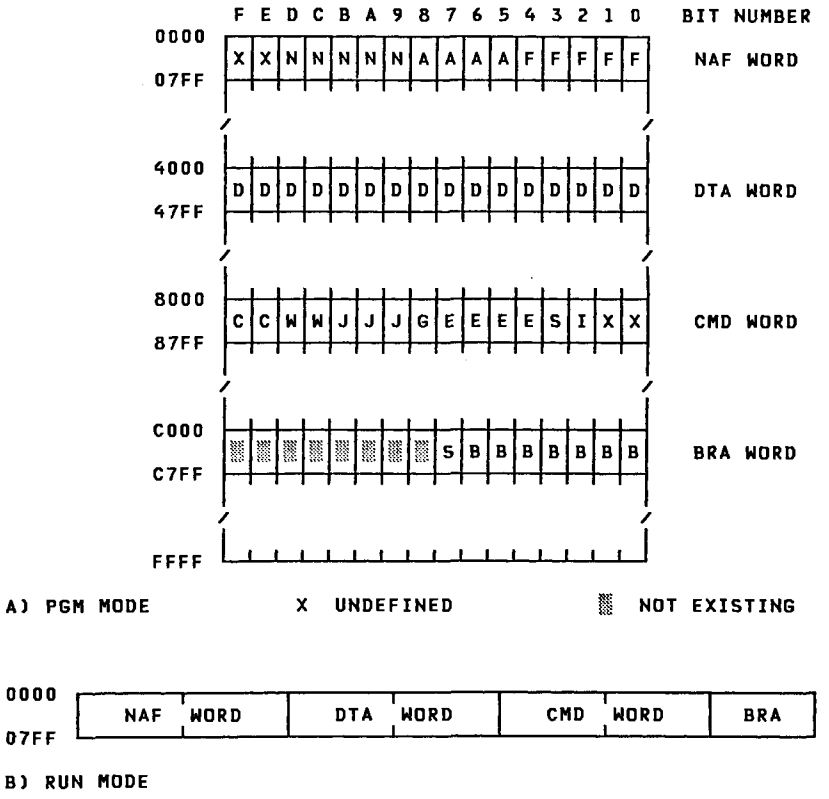

Fig. 13 Memory layout of the ICC

is being read or written into. Thus the ICC can be programmed in 4 DMA transfers, each preceded by setting the program pointer to the appropriate value (usually the top of the module address). In RUN mode, all the modules are selected simultaneously. A bus conflict cannot occur, since all the modules are isolated from the internal data bus (see Fig. 9). The only exception is the DTA module, which is allowed to write on this internal data bus.

The delay until the new address is available is independent of the way in which it is found, since the adder as well as the multiplexer (for GOTO/BRANCH) are always preparing before state 0. The new value has to be loaded only. Since a synchronous counter is used, the time for loading is the same as the time for incrementing, i.e. 10 ns.

3.6 The bus

The bus connects the FIFO and the ICC. It consists of a 2 × 25-pole Scotchflex cable not longer than 10 m. The transmitted signals are TTL standard. The allocation of the pins is shown in Fig. 11. The command and HSH lines are unidirectional, whereas the data lines are bidirectional. On the ICC side, all the lines are pulled up by 1.5 kΩ resistors in order to provide a defined state if the cable is pulled out or if the power in the FIFO crate is switched off. The data lines are driven and sensed by bus transceivers with strong fan-out and Schmitt trigger inputs reducing the noise sensitivity.

By using a modified CAMAC cycle when operating the ICC, virtually any bus length can be accommodated. However, special buffers (differential ECL drivers and/or optical couplers) and better cable (twisted and/or shielded) may then be necessary.

4. FUNCTIONAL DESCRIPTION OF THE MODULES[1)]

4.1 FIFO

When starting operation, the FIFO should be cleared by F(9) followed by F(0)A(0) (see Table 1). The first command clears the memory, whilst the second one clears the output latch. After this initialization, data written into the FIFO by the ICC are immediately available.

The readout type has to be adapted to the data structure in order to preserve it.

If the event consists of only one word, e.g. one ADC, no data structure at all is needed. Therefore, the FIFO can be read in the way that fits best: either word by word, or with fixed block length (e.g. 16384 words). For most computers, a block transfer will result in higher system throughput, compared to single word readout. If an ICC, however, is reading a FIFO, there is no difference.

If the event consists of several words, e.g. a few hundred ADCs, which have to be read in a fixed sequence, there are two possibilities: i) either the data are processed as mentioned above -- a fixed-length block transfer of several events at a time, resulting in the highest speed possible; or ii) data are structured for each event by using some characteristic ICC event formatting commands. This introduces some, often negligible, overhead in the event length, but makes it possible to recover from the situation if, for any reason, some data have been lost.

If the events have variable length, event formatting is necessary. This situation occurs when the data reduction features of the ICC are used.

Figure 12 shows a typical event structure. When the event length is read from the FIFO, no Q-response is given. This allows the readout computer to read events of variable length individually.

The command F(0)A(7) to prepare a FIFO dump allows the FIFO to be re-read provided that it is not full (in this case data read are immmediately over-written and therefore lost).

If enabled, the FIFO generates a CAMAC LAM when it is half full. Considering its size, this signal may come very late when running with a low event rate. Therefore, a TTL output is provided at the front panel: "EARLY" turns active when more than 512 words are in the FIFO, and becomes passive as soon as this is no longer true. Another output has the same characteristic, but it is active if the FIFO is at least half full (32768 words). Using these signals, a "weak" and a "strong" interrupt for the readout computer can be constructed. A veto for input data can be made as well. This prevents the readout of an event being interrupted because the FIFO is full.

4.2 NAF

This module allows the user to execute a code whose standard CAMAC equivalent requires several FORTRAN statements.

Example 1: The test -- whether a given selected station generates a LAM, or whether any station does so, combined with the loop to wait for it -- and a subsequent operation, can be done in one single instruction:

```
N = 11  A = 0  F = 2 WTLAM READ
```

The machine tests whether station 11 gives a LAM, until this happens (or until a timeout signal is generated, see subsection 3.4.7); then, it reads data from its subaddress 0 into the FIFO and clears the register (CAMAC function 2).

Example 2: The sequence

WALAM HEADER

is very useful when waiting for data but when a master trigger has not been defined. The machine loops until any station generates a LAM. If this occurs, an event header is generated and written into the FIFO, the BSY output is activated, and the execution of the program is resumed.

Example 3: This example, being very similar to the first one, is particularly suitable when a FIFO has to be read (see Fig. 14). As soon as the primary FIFO is half full, the first word is taken out of it, but it is not written into the secondary FIFO, since the CAMAC control word is not READ. This can be useful because the secondary ICC generates an event header by itself, and this would otherwise only create unnecessary redundancy. The command is as follows:

N = 17 A = 0 F = 0 WALAM EXEC

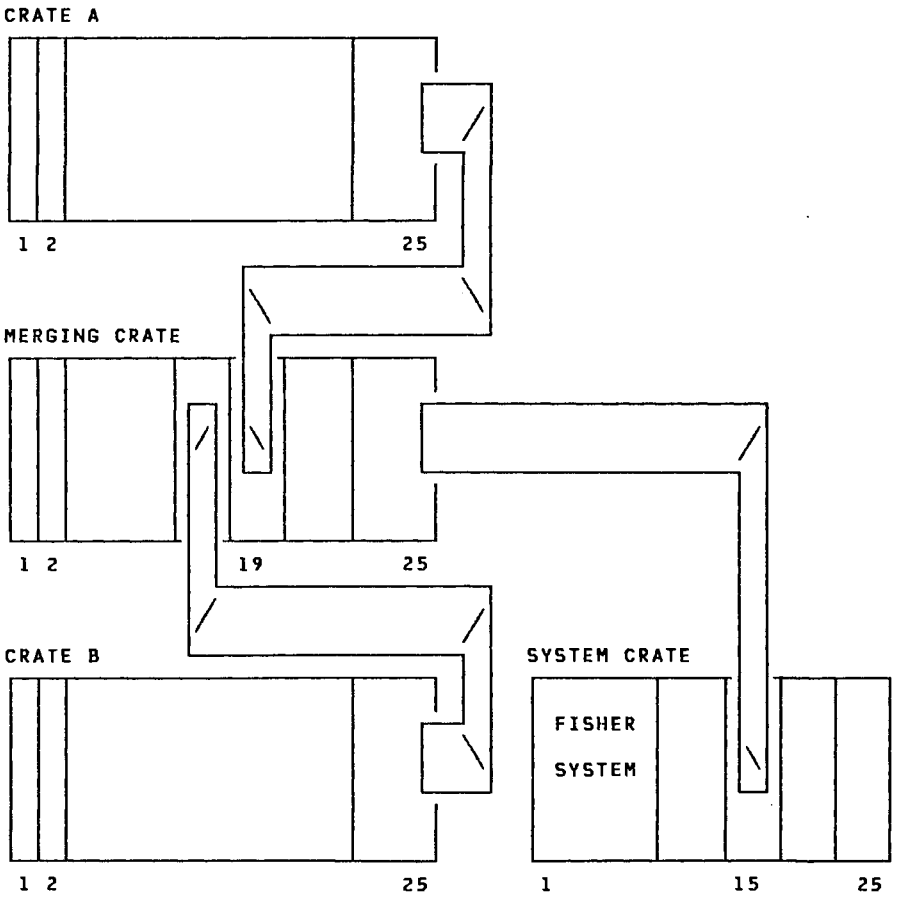

Fig. 14 The data-taking set-up

4.3 DTA

This module is used to subtract pedestals when data are read, to provide information to the CMD module for data reduction, to write data into CAMAC, and for GOTO commands.

Example 1: The three different uses of this module are:

```
LOOP:  PED = 53  N = 1  A = 0  F = 0 READ
       DATA = 1001  N = 22  A = 1  F = 16 WRITE
       GOTO LOOP
```

First, the number 53 is subtracted from data read at station 1, subaddress 0. If the result is negative, it is artificially set to zero. The result is written into the FIFO. Then, the number 1001 is written into station 22, subaddress 1. Finally, the program pointer is set to the value of the line number at the label LOOP. (See subsection 3.3.1.)

Example 2: data reduction:

```
PED = 0  N = 3  A = 0  F = 2 STORE
PED = EXTERN  N = 8 FOR A= 0 TO 15  F = 0 NEXTBIT IF BIT THEN READ
```

First, the data of station 3 is stored into the latch. At the same time it is written into the FIFO, and the latch counter is set to zero. No pedestal is subtracted from this item. Then, for station 8, the subaddresses are scanned together with the bits of the latched word. If the bit is set, the appropriate ADC will be read and the individual pedestal is subtracted (see subsection 3.3.2), otherwise the CAMAC cycle is skipped. This results in a time saving of the order of 1 μs per skipped word, and in space saving in the event. Since the pattern used to select the ADCs is also stored in the FIFO, it is easy to determine to which ADC the data belong when the off-line analysis is done.

Example 3: optimized data reduction:

```
       PED = 0  N = 3  A = 0  F = 2 STORE
       IF EMPTY THEN GOTO BLA
       PED = EXTERN  N = 8 FOR A = 0 TO 15  F = 0 NEXTBIT IF BIT THEN READ
BLA:   LENGTH
```

This code, which is almost identical to the one in Example 2, speeds up the readout by approximately 5 μs when no ADC has to be read. If any (one or more) ADC contains useful information, however, the execution is slowed down by 0.4 μs on the whole. After the ADC data the event length is written into the FIFO in this example.

Example 4: testing a specific bit:

```
RESTORE
NEXTBIT
NEXTBIT NOP
IF NOT BIT THEN SETFLAG
```

sets the counter to zero without changing the data latched. Then, the counter is incremented twice. If the selected bit (the third from right) is clear, the output FLAG is activated.

Table 5

Command word definition

CAMAC control:	C = 0:	READ
	C = 1:	WRITE
	C = 2:	EXEC
	C = 3:	(specified by E)
Wait condition:	W = 0:	(Continue)
	W = 1:	WMTR
	W = 2:	WALAM
	W = 3:	WTLAM
Jump condition:	J = 0:	(Continue)
	J = 1:	BIT
	J = 2:	EMPTY
	J = 3:	QRESP
	J = 4:	XRESP
	J = 5:	EXT1
	J = 6:	EXT2
	J = 7:	EXT3
Jump control:	G = 0:	(Skip command if condition true)
	G = 1:	ELSE [PP = PP + #BRA]
Extra function:	E = 0:	(Continue)
	E = 1:	GOTO [PP = #DTA]
	E = 2:	CAMCLR
	E = 3:	CAMINI
	E = 4:	STORE
	E = 5:	LOAD
	E = 6:	RESTORE
	E = 7:	NEXTBIT
	E = 8:	SETBUSY
	E = 9:	CLRBUSY
	E = 10:	SETFLAG
	E = 11:	CLRFLAG
	E = 12:	NUMBER (C = 3) OUTNUM (C = 1)
	E = 13:	LENGTH (C = 3) OUTLEN (C = 1)
	E = 14:	MARKER
	E = 15:	HEADER
Cycle length:	S = 0:	SHORT
	S = 1:	LONG
Jump inversion:	I = 0:	(Jump condition)
	I = 1:	NOT (Jump condition)

4.4 CMD

The CMD module is, strictly speaking, responsible for the interpretation of the compound statements described in Ref. 1. It contains information about data direction, whether or not the ICC has to wait and what it has to wait for, and whether the command has to be executed or not. It decodes the following structure:

... IF ... THEN ... (ELSE ...)

Instead of a table of possible allowed combinations of keywords as listed in Table 5, the ICCOLA manual (Ref. 1) should be consulted. A syntax diagram found there allows you to find these combinations yourself.

Just to demonstrate how complex possible commands can be, and that to make a list of them would be insane, the following (legal) example is given:

```
LONG, DATA = 12345  N = BORER  A = SUB  F = 16
NEXTBIT WMTR IF NOT QRESP THEN WRITE ELSE +123
```

LONG	the CAMAC cycle length is defined
DATA	to be written into CAMAC are defined
N	the station number
BORER	constant set previously (in the range 1-22)
A	the subaddress
SUB	constant set previously
F	the CAMAC function
NEXTBIT	after this command, the bit counter related to the latch is incremented
WMTR	if the following test is true, execution is halted until MASTER turns active.
IF	
NOT QRESP	is true, the
WRITE	instruction is executed.
ELSE	execution continues
+123	steps ahead without waiting for MASTER

4.5 BRA

The BRA module calculates the system address according to signals generated by the CMD module. Normally, the address is found by incrementing the program pointer. This is the default action, and it does not show up in the program. The program pointer can be modified by adding a signed 8-bit number if a set of instructions does not need to be executed. If a single instruction does not need to be carried out, it can be skipped. The program pointer can be loaded by a dedicated instruction in order to reach any statement in the program. This option takes data from the DTA module to accomplish the job.

The ICCOLA manual (Ref. 1) should be consulted for the syntax and the restrictions.

4.6 MEMORY

The memory layout is a specific feature of the ICC, since it allows it to be programmed in the same way as any 16-bit machine, whilst in RUN mode the control word has a length of 56 bits (see Fig. 13).

The meaning of the defined bits has been discussed in Section 3. The address is always given by the program pointer. In PGM mode, the two MSBs are used to select the module. In RUN mode, all the modules are selected at the same time.

4.7 BUS

Lines	1 .. 4	transfer commands
Name	CA .. CD	to operate and program the ICC;
Direction	to ICC	the logic is positive true.
Line	5	strobes the command
Name	CS	transferred to the ICC;
Direction	to ICC	the logic is active low.
Line	6	data have to be latched
Name	FFS	by the FIFO input latch
Direction	from ICC	on the rising edge.
Lines	7 .. 22	transfer program and data
Names	D_{15} .. D_0	to the ICC and back;
Direction	bidirectional	three-state logic, positive true.
Line	23	when low, indicates
Name	Q	that the last word of an event
Direction	from ICC	is being transferred (length).
Line	24	is open.
Line	25	when high, halts the ICC
Name	HSH	before the next command
Direction	to ICC	is performed (handshake).

All the lines are pulled up in the ICC by 1.5 kΩ resistors. The bidirectional data lines are driven by three-state logic.[See Figs. 8c) and 11c).]

5. FUNCTIONAL DESCRIPTION OF THE SYSTEM

5.1 The commands for the Intelligent Crate Controller (ICC)

For the timing, please refer to Fig. 7. The minimum cycle time when using a 10 m cable is given for each command, if applicable. The code number is applied to the CA ... CD bus lines. All data are 16 bit words.

5.1.1 Read program at pointer

Code: 3
Data: out of ICC
Mnemonic: RPP

The command transfers the data stored at the memory location pointed at by the program pointer. The pointer is auto-incremented when the command is terminated, either by changing the command

code, or by de-activating the CS. The command is performed in PGM mode only; it is ignored in RUN mode.

The response time is twice the cable delay plus 50 ns. The recovery time is the memory access time plus the counter delay: 130 ns + 40 ns = 170 ns. The minimum cycle time is 320 ns.

5.1.2 Examine program pointer

Code: 7
Data: out of ICC
Mnemonic: EPP

The command transfers the actual value of the program pointer. The two MSBs specify the selected memory page. The next three bits are not used. The eleven LSBs are the current address. They determine where the controller will start execution when switched to RUN mode. The command is performed in PGM mode only; it is ignored in RUN mode.

The response time is twice the cable delay plus 120 ns. The minimum cycle time is 220 ns.

5.1.3 Examine event number

Code: 4
Data: out of ICC
Mnemonic: EEN

The command transfers the actual value of the event counter. This is a 16-bit word. The command is performed in PGM mode only; it is ignored in RUN mode. The response time is twice the cable delay plus 120 ns. The mininum cycle time is 220 ns.

5.1.4 Write program at pointer

Code: 5
Data: into ICC
Mnemonic: WPP

The command stores data transmitted into the memory location pointed at by the program pointer. The pointer is auto-incremented when the command is terminated, either by changing the command code, or by deactivating the CS. The command is performed in PGM mode only; it is ignored in RUN mode.

The time required is the cable delay plus 50 ns. The minimum cycle time is 270 ns.

5.1.5 Set program pointer

Code: 6
Data: into ICC
Mnemonic: SPP

The command stores data transmitted into the program pointer. The two MSBs specify the selected memory page. The next three bits are not used. The eleven LSBs are the current address. They determine where the controller will start execution when switched to RUN mode. The command is performed in PGM mode only; it is ignored in RUN mode.

The time required is the cable delay plus 120 ns. The mininum cycle time is 170 ns.

5.1.6 Set event number

Code: 8
Data: into ICC
Mnemonic: SEN

The command stores data transmitted into the event counter. This command is performed in PGM mode only; it is ignored in RUN mode.

The time required is the cable delay plus 120 ns. The minimum cycle time is 170 ns.

5.1.7 Initialize slave crate

Code: C
Data: none
Mnemonic: ISC

The command performs an initialization (Z) function in the slave crate. It is performed in PGM mode only; it is ignored in RUN mode. The ICC state is not affected.

The time required is the cable delay plus 50 ns. The recovery time is 1 μs. The minimum cycle time is 1.1 μs.

5.1.8 Clear slave crate

Code: D
Data: none
Mnemonic: CSC

The command performs a clear (C) function in the slave crate. It is performed in PGM mode only; it is ignored in RUN mode. The ICC state is not affected.

The time required is the cable delay plus 50 ns. The recovery time is 1 μs. Mininum cycle time is 1.1 μs.

5.1.9 Enter program mode

Code: F
Data: none
Mnemonic: PGM

The command sets machine status to PGM mode. This command is performed in any mode, but is irrelevant in PGM mode.

The time required is the cable delay plus 50 ns.

5.1.10 Enter run mode

Code: E
Data: none
Mnemonic: RUN

The command sets machine status to PGM mode. This command is performed in PGM mode only; it is ignored in RUN mode.

The time required is the cable delay plus 50 ns. The recovery time is three internal clock cycles.

Note: The command lines must not change while the CS is active.

5.2 Test for contact

In the following discussion, the mnemonic notation is chosen for its clarity (see Table 1). Assuming that CAMAC has already been booked, the first steps when using the ICC should be to verify if there is contact at all. This is most conveniently done by

```
ERR = .FALSE.
PGM                                 * Enter PGM mode
OLDPP = EPP                         * Save the old Program Pointer
OLDEN = EEN                         * Save the old Event Number
SPP = 43690                         * Set Program Pointer to 43690
SEN = 21845                         * Set Event Number to 21845
DATA = EPP                          * Read Program Pointer
IF (DATA.NE.43690) ERR = .TRUE.     * Test if Program Pointer is OK
DATA = EEN                          * Read Event Number
IF (DATA.NE.21845) ERR = .TRUE.     * Test if Event Number is OK
SPP = OLDPP                         * Restore old Program Pointer
SEN = OLDEN                         * Restore old Event Number
```

Sample program: Test for contact to ICC

since many features of the ICC are tested and because the state of it is well known afterwards (mode, Program Pointer, Event Number). It is left to the user to decide what action has to be taken if an error should occur. It takes some experience, too, to decide about the impact of a specific error (e.g. Is power switched on? This is the most common error source!).

5.3 Programming sequence of the ICC

```
 1)  PGM              * Enter Program Mode

 2)  SPP #0000        * Set Program Pointer to 0000
 3)  WPP #(NAF)       * Write Program at Pointer
 4)  repeat (3) until the NAF portion is transferred

 5)  SPP #4000        * Set Program Pointer to 4000
 6)  WPP #(DTA)       * Write Program at Pointer
 7)  repeat (6) until the DTA portion is transferred
 8)  SPP #8000        * Set Program Pointer to 8000
 9)  WPP #(CMD)       * Write Program at Pointer
10)  repeat (9) until the CMD portion is transferred

11)  SPP #C000        * Set Program Pointer to C000
12)  WPP #(BRA)       * Write Program at Pointer
13)  repeat (12) until the BRA portion is transferred

14)  SPP #0000        * Set Program Pointer to 0000
(15) SEN #nnnn        * Set Event Number to nnnn)
16)  RUN              * Enter Run Mode
```

Programming is most efficiently done in address auto-increment mode. This represents a series of four DMA transfers. Since maximum programming speed is around 3 MHz, the total programming time for 4 × 2048 words might be as low as 3 ms. Assuming one transfer every 3 ms (slowest mode on VAX), the programming would take 25 s.

For verification purposes, it is recommended that the program pointer is read after every portion and compared with the actual program length.

A total verification pass can be done by copying the sequence above but replacing the WPP commands by RPPs and comparing data read with the original file. The reading, of course, takes exactly the same time as the programming.

Note: The program pointer is incremented after every RPP and WPP. Therefore, a so-called Read-Modify-Write cycle is not possible. The program pointer must rather be set before every command.

6. HOW THE SYSTEM IS USED IN THE PS182 EXPERIMENT

The instrumentation of the PS182 experiment requires, amongst many NIM crates, some CAMAC crates, of which two must usually be read. For reference, they shall be called crate 'A' and crate 'B'. As shown in Fig. 14, a third crate -- the 'Merging Crate' -- is used in order to merge the data streams, so that the host computer needs only to read one buffer.

The two devices described in this report are shown in Fig. 15.

Another device, which has not been described so far, is placed in the Merging Crate, in addition to the two FIFOs: this is a Watchdog. It is used to generate macro events containing several physical events and some padding words at the end in order to get long

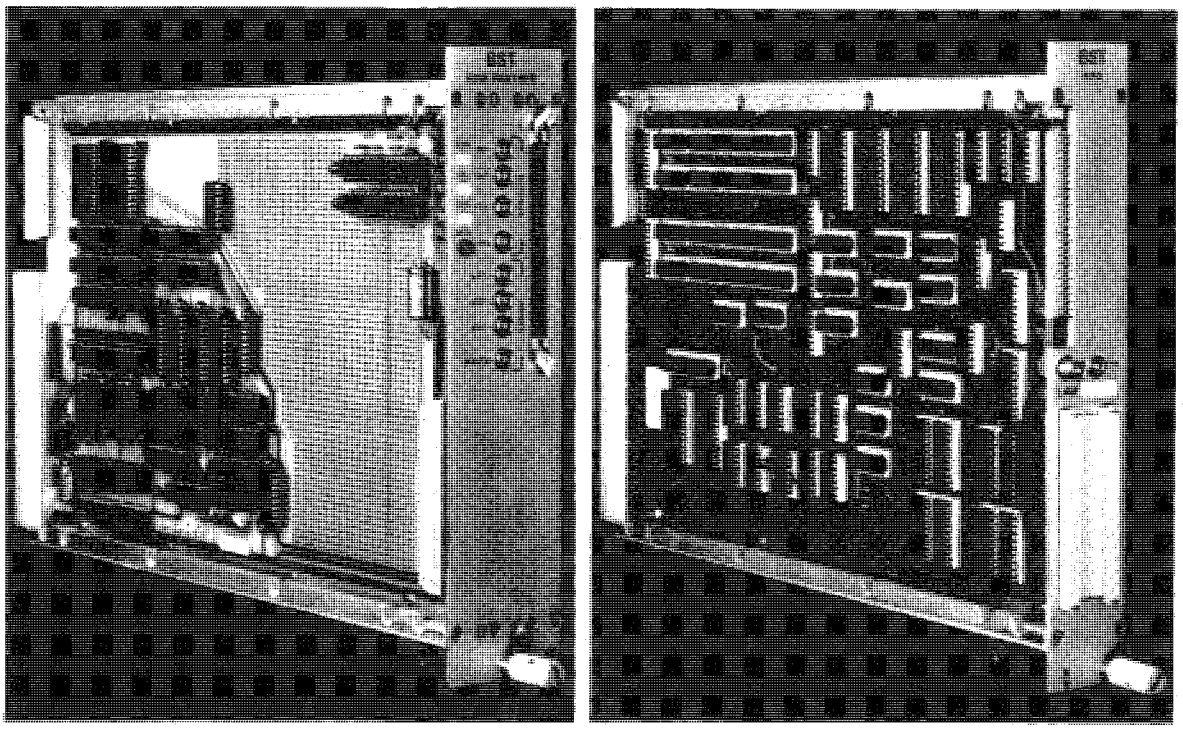

Fig. 15 The crate controller and the FIFO

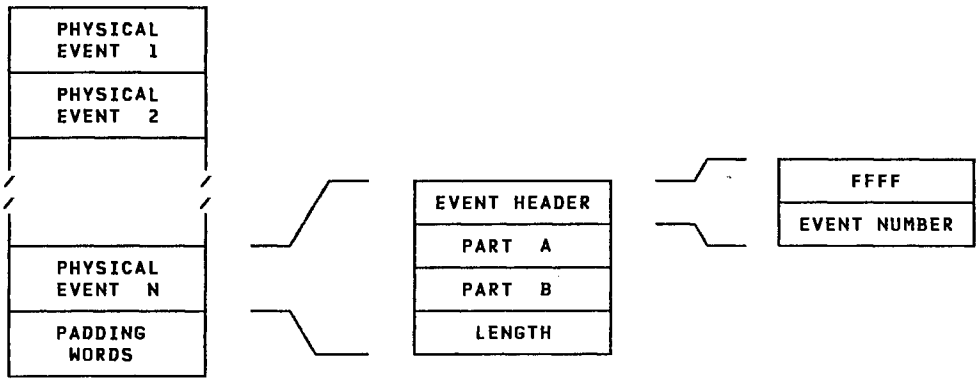

Fig. 16 The event structure presented to the host computer

data blocks of constant length. It is described in detail elsewhere[5]. Figure 16 shows the format of data delivered by the Merging Pink Panther.

For Crate B, two methods of readout are chosen, namely 'Data-taking' and 'Calibration'. A pattern unit is used to indicate the current run type to the Pink Panther, which will read corresponding subgroups of ADCs. Other bits in the same pattern unit are used to tell it which subgroups have to be read out. For Crate A, three operational modes are defined and fed into its Pink Panther via the front-panel connectors EXT1 ... EXT3 (see subsection 3.4.7): i) generate a dummy event; ii) read the crate with data reduction; and iii) read all the ADCs.

The first mode is used for incrementing the event number if its detector is providing no useful information at all; the second mode is standard for a 'Physics Run'; whilst the third mode may be used to perform a 'Pedestal Run'.

In 'Physics Run' mode, a pattern unit bit is associated with each ADC. If the bit is set, the appropriate ADC will be read and its pedestal may be subtracted if desired, without introducing any additional delay; otherwise the operation is skipped. During the time a standard CAMAC cycle lasts, three bits can be tested.

Thus the BST-Pink Panther concept offers much flexibility when data reduction and formatting are required.

7. ARE THERE OTHER POSSIBILITIES?

Of course, the ICC is not alone in the field of programmable CAMAC controllers, so it may be interesting to see where the others can be found.

There exist buffers for ADC readout which can be connected to one another in order to increase the size. However, these buffers have very little space: only 256 words. They are not very intelligent -- essentially they read only one ADC. Therefore, the normal assumption will be that a microprocessor or a bit-slice processor has to be used. It is well known that the microprocessors offer flexibility; their function depends on the programming which allows the user to do virtually whatever he likes. Program development is no problem, since powerful tools such as symbolic debuggers and versatile programming languages such as Pascal and Forth do exist. In the last few years, the computing power of the microprocessors has grown tremendously: an 8-bit CPU of 1978 with 2.5 MHz has to be compared with a 32-bit CPU of 1983 with 12 MHz. The latter microprocessor is at least 20 times more powerful!

If the flexibility of a microprocessor is not sufficient, or if the CPU size has to be adapted to the problem, the choice is to use bit-slice processors and microsequencers. A beautiful example can be found in Ref. 6, where an IBM computer has been emulated for a few percent of the price, but still having half the speed of the original! Therefore many designers have decided to adapt a microsequencer, sometimes without bit-slice processor, to the problem of taking data out of CAMAC.

But CAMAC is not the natural type of peripheral device for a processor. Usually, peripherals are memory-mapped and read/written as any memory location, including DMA transfers. In CAMAC, the symbolic memory locations are not dense (e.g. 12-channel ADCs, 8-channel TDCs), so a CAMAC crate is full of exceptions. What is worse, the devices have very different access times. The processor has to wait for a TDC's stop signal plus an eventual conversion. Some ADCs have their result within 0.1 µs, whilst other ADCs take as long as 100 µs.

Somehow, it is just a waste to take a micro (of any kind) to read CAMAC. A dedicated processor, not caring about the order in which stations are read (because every station is individually addressed), can collect the data at least as efficiently. If, in addition, a pedestal can be subtracted from data read without additional delay while operating with maximum CAMAC speed, and if an understandable programming language is provided, then it should be hard to argue against this machine. But it is even more powerful: the ICC delivers formatted events which are ready to be written to tape. The on-line computer can concentrate on monitoring, and DMA transfers from one single station (the FIFO), which holds up to 65536 words of data and can be read independently of the ICC's operation.

But this is not all. The ICC can be used to build tree-like structures, since an ICC can be programmed to program an ICC. There is no fundamental limitation to this. Up to now we have used only a very small tree[7], but this does not prevent larger ones from being built.

It should be noted that only the front-end ICCs need to be synchronized, i.e. they must get the same trigger at the same time, otherwise the events may be mixed, and that their BSY signal is combined (by using logical fan-in/fan-outs) to provide a common "Data-Taking Busy" signal for inhibiting the trigger as long as readout is in progress.

Time out can be given if a CAMAC module fails to send a LAM. The experiences of the PS182 group are noted in Ref. 8 also.

APPENDIX

INTRODUCTION INTO ALGORITHMIC STATE MACHINES

1. GENERAL INTRODUCTION

An Algorithmic State Machine (ASM) is a device which, in most general terms, controls a set of output lines according to its actual state, and determines the next state by using some algorithms on the actual state and some input variables. There are many types of ASM, and many devices usually not referred to as ASMs can be represented as such: an S/R flip-flop, an edge-triggered JK flip-flop, a counter, a traffic-light controller, are examples of increasing complexity.

Instead of writing a full introduction about this vast topic, I refer the reader to a complete overview, for example Ref. 9.

2. MULTIPLEXER-BASED ASM

Of the many possible implementations (Gate, Multiplexer, ROM) the fastest approach was chosen, namely the one based on a multiplexer. It has two major advantages over the alternatives: i) the system speed is almost as high as when using a gate-implemented ASM, but modifications are very easily done; ii) the programming is almost as easy as when using a ROM (Read Only Memory), but the multiplexer has a far superior speed. Figure A.1 shows the basic design.

Depending on the actual condition of the state latch, the multiplexer selects the number of the next state. This is loaded into the latch by a clock pulse. The maximum frequency can be calculated by adding the delay time of the latch to the delay of the multiplexer. Taking always the worst case figures, 9.2 ns for the latch and 35 ns for the multiplexer add up to 44 ns for the cycle, corresponding to 22.7 MHz safe operating frequency. Meanwhile, the state number is decoded by a demultiplexer with a delay of 9 ns maximum. This output is used to control the device where the ASM is built in.

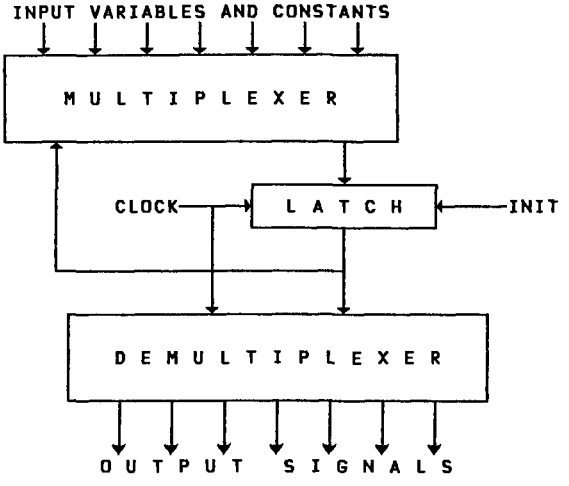

Fig. A.1 Multiplexer-based ASM

Whilst speed is the strongest point of this design, the weakest point is the very large number of inputs. If the state latch has n bits, resulting in 2^n possible states, there are $n \cdot 2^n$ input variables!

The FIFO and the ICC each have such an ASM built in which has 4 state bits involving 16 possible states, and, accordingly, 64 inputs variables.

3. THE FIFO CASE

The FIFO ASM has to manage input data, output data, and the FIFO memory. The states are numbered from 0 to F and have the following meaning:

EMPTY	0
FULL	2
IDLE	1,3
WRITE CYCLE	9-D
READ CYCLE	4-8

The latch can be cleared by an external signal which causes the ASM to be in state 0. The signal is called INIT and is automatically generated when power is switched on.

The latch can be preset to F (all bits set). The next state is then number 2. This causes the ASM to consider the FIFO as being full, so the signal is called PFD.

If, by accident, the state should be 7, C or E, the ASM will recover within 2 clock cycles to a legal state. No hang-up can occur.

If the FIFO is empty, only a Write request will be recognized. If it is full, a Read request is needed in order to exit the current state. If either request is accepted, the appropriate cycle is performed. The complementary request may come during the execution of the other one. Figure A.2 shows the State Chart and its program, Fig. A.3 its timing.

The State Chart has point symmetry, and in principle it is doing the same things on symmetrical states.

In state 4/9, the counter needed is loaded into the address latch. This address is going to the FIFO memory. At the end of this state, the counter is incremented in order to point to the next address to be read/written.

In state 5/A, time is spent for memory access and for the counter to increment.

In state 6/B, the difference between the counters is calculated and made available for the decision in state 8/D. At the end of state 6, data coming from memory are latched into the output latch.

In state 8/D, a comparison is made to see whether the counters are equal or not. If, in state 8 (after a read), they are equal, the FIFO must be empty. On the other hand, if the counters are equal in state D, the FIFO is full.

If a slower memory is used, the state 7/C can be inserted by cutting some jumpers to take account of this. The system clock speed can be modified as well, just by exchanging two cheap ICs (see Table 3).

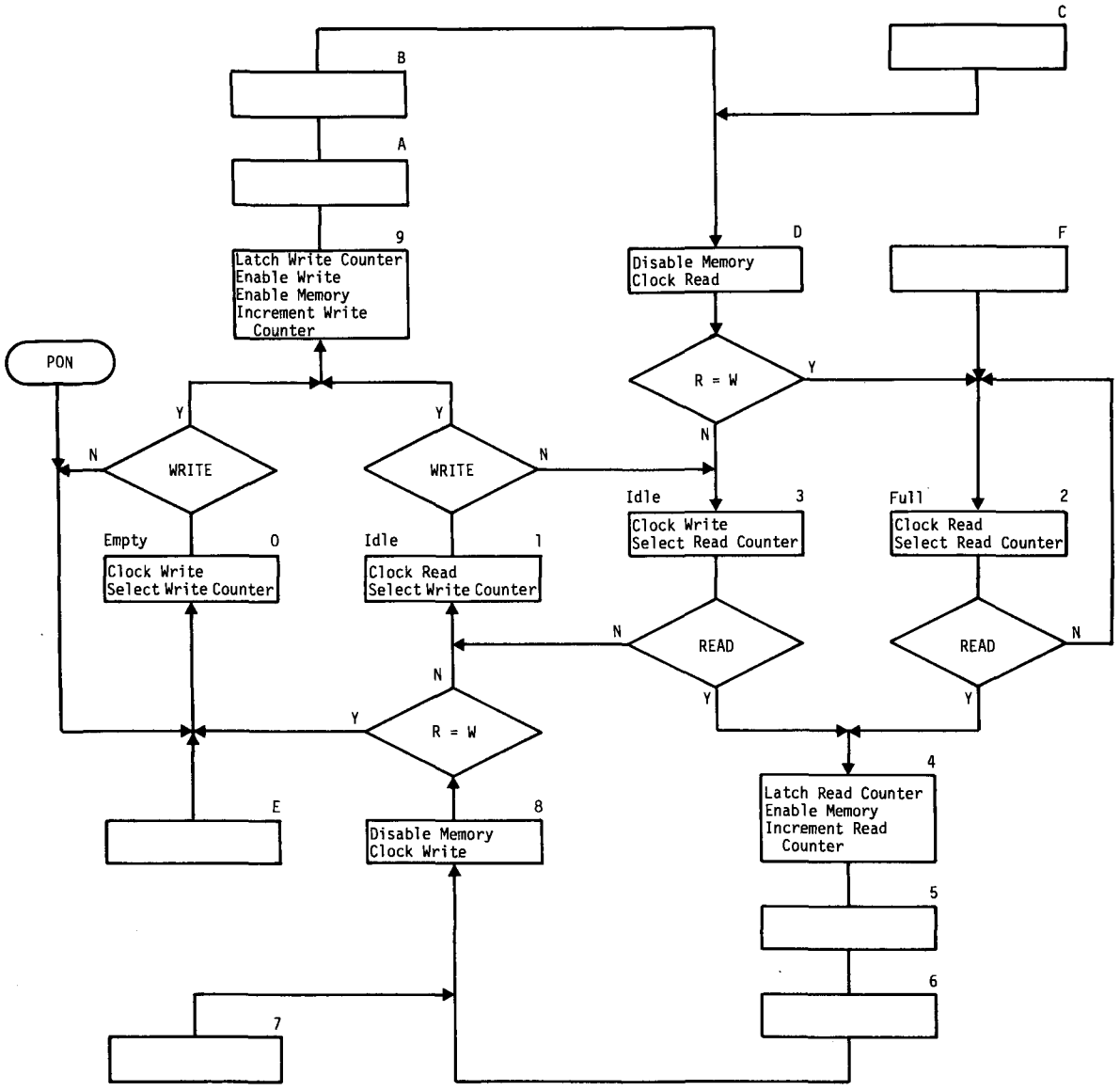

Fig. A.2 FIFO ASM chart

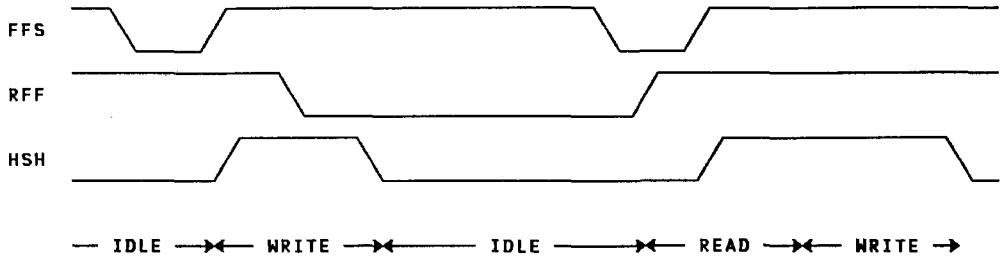

Fig. A.3 FIFO timing diagram

4. THE ICC CASE

The ASM generates the signals for the ICC to fetch a new command and decides how much of it has to be executed. The states are numbered from 0 to F. There is one input line which forces the ASM to enter state F when active; the line is $\overline{\text{RUN}}$/PGM. Two input lines

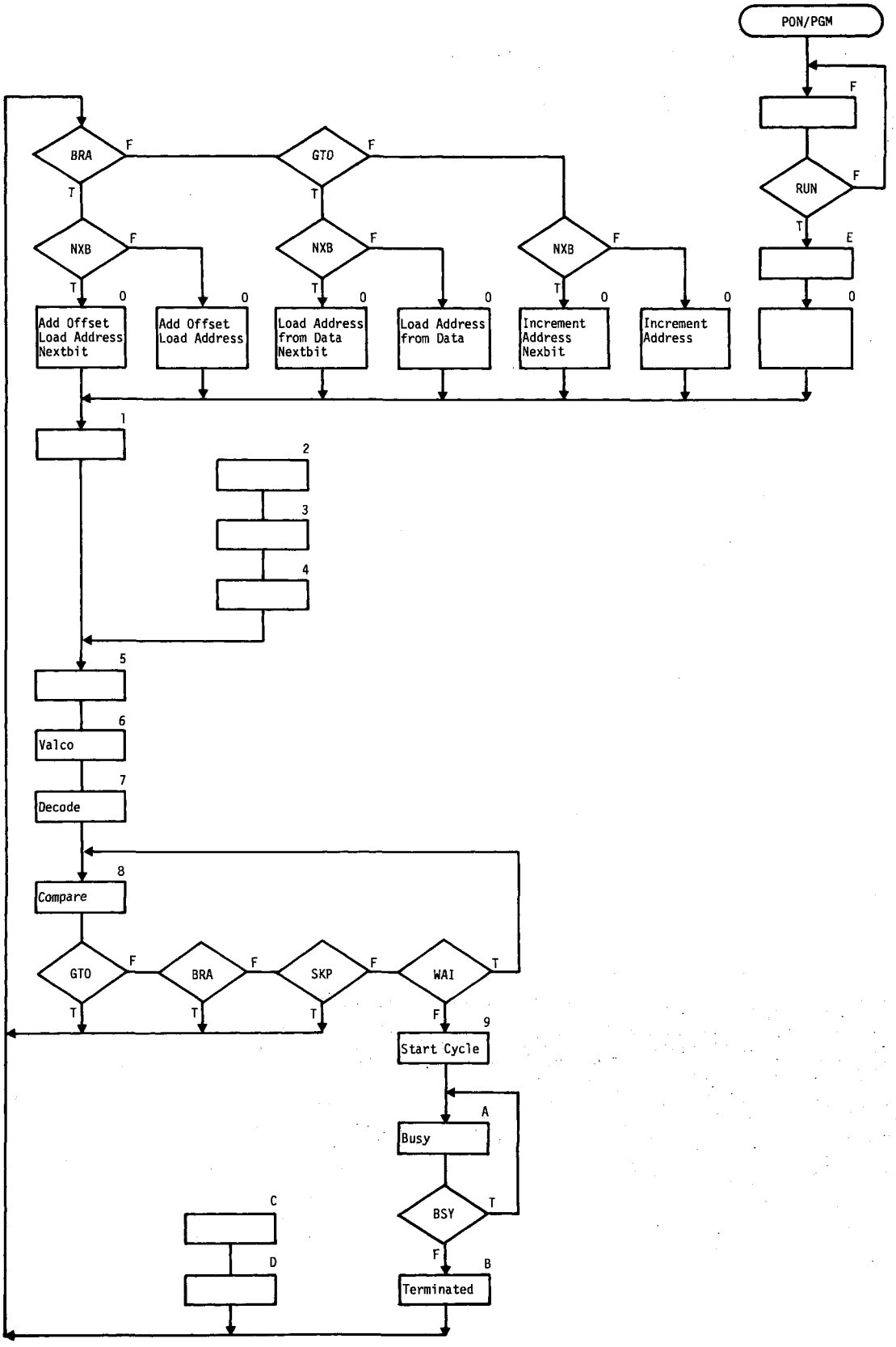

Fig. A.4 ICC ASM chart

are used to abort execution of the command, before any action is taken; they are called SKIP and GOBRA (GOTO or Branch). One input is used to halt the ASM before the command is executed; it is called HSH, and is active if the FIFO is either full, or is busy writing, or is not powered, or if contact is lost; or if the ICC is waiting for a LAM or a master trigger. In all these cases the ASM is looping in state 8. In state 9 the command is executed and one of three possible cycles started (see subsection 3.2.2). As long as this cycle is active, the last input line to the ASM is active: BUSY. Figure A.4 shows the ICC ASM State Chart with the corresponding program.

The output lines are distributed over the three boards of the ICC; namely the states 0, 6, 7, 8, 9, and B, and, in addition, the signal VALCO. These lines are used individually by the elements of the ICC.

The timing of the output lines is shown in Fig. A.5.

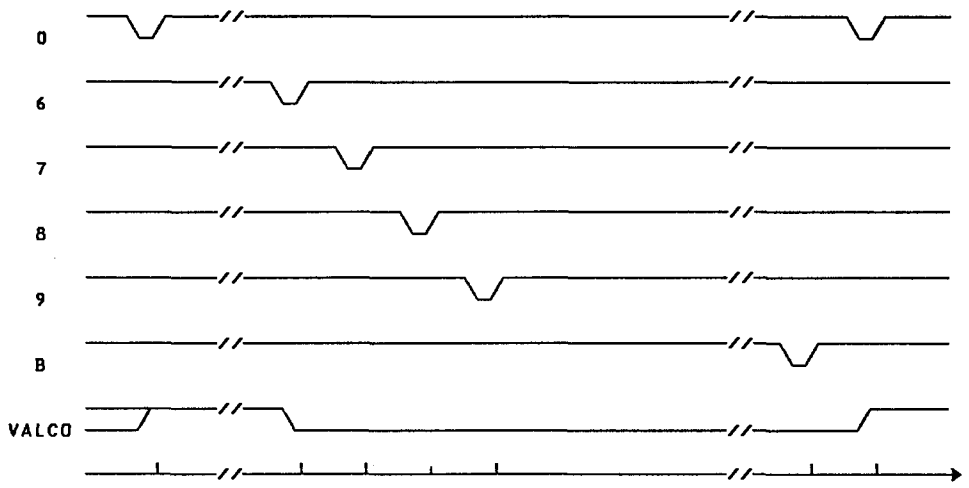

Fig. A.5 ICC timing diagram

REFERENCES

1) D. Tröster and Ch. Buess, ICCOLA user's manual, to be published as Blue Note.

2) EURATOM Report EUR 4100e (1980).

3) EURATOM Report EUR 4600e (1980).

4) EURATOM Report 6500e (1978).

5) D. Tröster, BST - Watchdog user's manual, to be published as Blue Note.

6) P.F. Kunz, Proc. CERN School of Computing, Jadwisin (Poland), 1978 (CERN 78-13, CERN, Geneva, 1978), p. 25.

7) Ch. Findeisen, BST-Pink Panther Primer, to be published as Blue Note.

8) D. Tröster, thesis, to be published

9) C.A. Wiatrowski and C.H. House, Logic circuits and microcomputer systems, International student edition (McGraw-Hill, Inc., New York, 1980).