LA-UR - 82-2718

REVISION I: AUGUST 1983 (PAGES 3, 4, 6, 10, 25, 48, and 63)
REVISION II: MARCH 1985 (PAGES 5 AND 6)
REVISION III: JULY 1986 (PAGES 3, 4, 5, 6, AND 22)

TITLE A CAMAC PRIMER

AUTHOR(S) Peter Clout

SUBMITTED TO An in-house document to be handed out at a presymposium seminar to be conducted by Peter Clout (Dec 5), before the 1982 DECUS (Digital Equipment Computer Users System) U.S. Symposium to be held at Anaheim, California Dec 6-10, 1982

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FOREWORD

This Primer is an edited version of a set of notes produced for a CAMAC course given at Los Alamos in January-February, 1982. The intention of this Primer is to introduce CAMAC and to explain the basics of the system from a user's and CAMAC plug-in designer's point of view. An attempt has been made to pull together those aspects of CAMAC that are scattered amongst the various standards, for example, LAMs.

The Primer is not intended as a replacement for the actual standards, and those working with CAMAC should seriously consider obtaining copies of the standards of interest. The IEEE recently has published a book of the seven most used specifications, CAMAC Book 1982, see Table 1.1a. Tables and figures have been freely copied from the European copies of the standards, and I acknowledge the agreement and the encouragement of ESONE and its secretary, Dr. Horst Meyer, in this matter. I would like to acknowledge the help of Lou Costrell of NBS in supplying a copy of Fig. 1.1.

CAMAC is a well-supported, reliable computer input/output (I/O) system for data acquisition and control. I trust that this Primer will help users of CAMAC gain a greater understanding of the system and how to use it.

As with any system, words and diagrams can only go so far. In the end, one just has to use CAMAC to really understand it. This I encourage.

CONTENTS

A CAMAC PRIMER

by

Peter Clout

ABSTRACT

CAMAC has been in use for experimental physics for 12 years or more and has recently been gaining acceptance in other fields of research and industry. This tutorial covers the basics of CAMAC, including the primary standard, the parallel branch and the serial highway, auxiliary controllers in a CAMAC crate, and the software standards. Various systems aspects of CAMAC are also covered: plug-in design, CAMAC-computer coupler design, and Look-At-Me (LAM). This tutorial is intended as a supplement to the various CAMAC standards.

---

1. INTRODUCTION

1.1 History

In the '60s the complexity of high-energy physics and nuclear physics experiments convinced people that data-acquisition systems would need to be a bus structured rather than individual point-to-point connections. Figure 1.1 illustrates this problem as it appeared in 1963. The problem arose because the number of point-to-point connections tended to increase with the square of the complexity of the system; whereas, a bus system increased linearly with system complexity and was usually more flexible. Many Laboratories then started to develop and implement data-acquisition busses that all had the common feature of being mutually incompatible! The European Standards On Nuclear Electronics (ESONE) committee recognized this, and the need for a common system was agreed

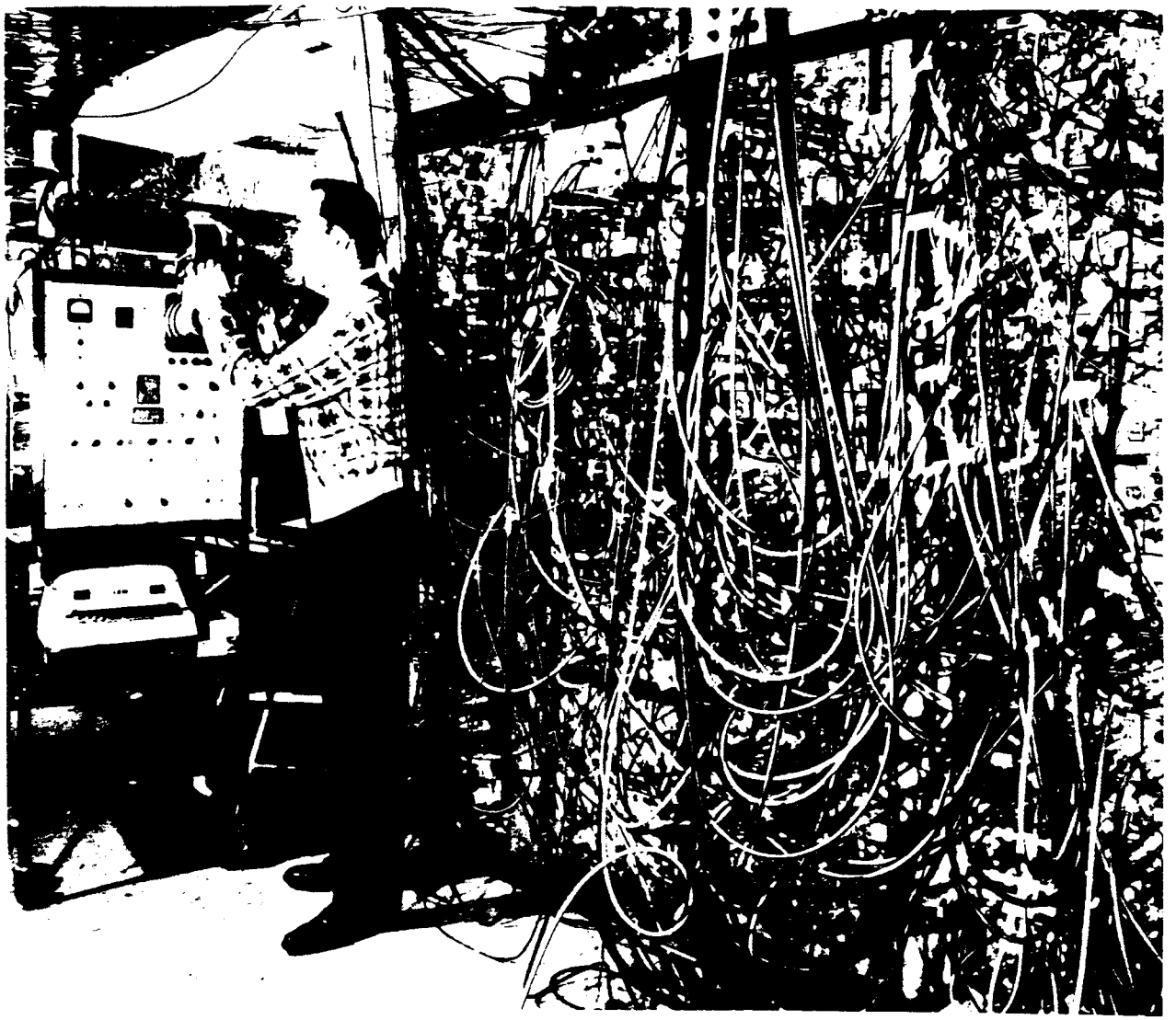


Fig. 1.1.
Lawrence Berkeley Laboratory data acquisition in 1963.

upon between the Laboratories. Between 1966 and 1969, the system (CAMAC) was defined, and the basic standard (EUR 4100) was published. This was endorsed by the National Instrumentation Method (NIM) committee in the US early in 1970. CAMAC was a name chosen to be palindromic, with no meaning attached. Later it acquired the phrase Computer Automated Measurement and Control. Subsequent to the first standard, many other standards and recommended practices have been published (Table 1.1a) as well as a large literature of application papers cited in the CAMAC bibliography, currently 1800 papers. (For an availability list see Table 1.1b.) In addition, the CAMAC product guide lists some 60

Table 1.1a

CAMAC SPECIFICATIONS AND SUPPLEMENTARY INFORMATION

Description	Publications by the Commission of the European Communities and the ESONE Committee or ECA	Corresponding Documents of Publications by Other Bodies			
		US Department of Energy and US NIM Committee	Published by IEEE, ANSI	Published by IEC	Published by CMEA
A Modular Instrumentation System for Data Handling	EUR 4100e (1972)(English)[a] EUR 4100f (1972)(French)[a] EUR 4100i (1972)(Italian)[a]	TID-25875[b]	ANSI/IEEE Std. 583[c] (1982)	IEC Publ. 482 IEC Publ. 516	4572-74 and 4573-74
Block Transfers in CAMAC Systems	EUR 4100 supp.	TID-26616[b]	ANSI/IEEE Std. 683[c] (1976)	IEC Publ. 677	
Organization of Multi-Crate Systems (Parallel Branch Highway)	EUR 4600e (English)[a] EUR 4600f (French)[a] EUR 4600i (Italian)[a]	TID-25876[b]	ANSI/IEEE Std. 596[c] (1982)	IEC Publ. 552	in preparation
Specifications of Amplitude Analogue Signals within a 50 Ω System	EUR 5100e (1974)	TID-26614			in preparation
Supplementary Information on CAMAC Instrumentation System	Supplement to Camac Bulletin Issue 6	TID-25877	Part of ANSI/IEEE Std. 583 and 596 (1982)		
CAMAC Serial Highway System and Serial Crate Controller Type L2	EUR 6100e[a]		ANSI/IEEE Std. 595[c] (1982)	IEC Publ. 640	in preparation
The Definition of IML A Language for Use in CAMAC Systems	ESONE/IML/01	TID-26615			
Real-Time Basic for CAMAC	ESONE/RTB/03		ANSI/IEEE Std. 726[c] (1982)	45 (CO) 221	
Recommendations for CAMAC Serial Highway Drivers and LAM Graders for the SCC-L2	ESONE/SD/02	DOE/EV-0006			
Multiple Controllers in a CAMAC Crate	EUR 6500e[a]	DOE/EV-0007	ANSI/IEEE Std. 675[c] (1979)	IEC Publ. 729	
Subroutines for CAMAC	ESONE/SR/01	DOE/EV-0016	ANSI/IEEE Std. 758 (1979)	IEC Publ. 713	
Definitions of CAMAC Terms used in ESONE Specifications	ESONE/GEN/01			IEC Publ. 678	
Revision of ESONE CAMAC Documents	ESONE/DOC/02	DOE/EV-0009[d]	IEEE Std. 583A[d]	45 (SEC) 228	
Compatible Extended Use of the CAMAC Dataway	ESONE/COMPEX				
CAMAC, Updated Specifications (including EUR 4100, EUR 4600, EUR 6100, EUR 6500, ESONE/GEN/01 and supplementary information)	EUR 8500e (1983)				
CAMAC Bibliography	ECA/GEN/01 (1979)				
CAMAC Bibliography	ECA/GEN/01 suppl. (1981)				
Recommendations for Analogue Signals for CAMAC Industrial Applications	ECA/ISG 81/1 (1981)				
Recommendations for the Industrial Application of CAMAC	ECA/ISG 81/2 (1981)				

Table 1.1a (Cont.)

OTHER DOCUMENTS

Description	Publications by the Commission of the European Communities and the ESONE Committee or ECA	Corresponding Documents of Publications by Other Bodies: US Department of Energy and US NIM Committee	Published by IEEE, ANSI	Published by IEC	Published by CMEA
Technique for the Assessment of Communication System for Process Control	EDISG/COM/01, ECA (1979)				
ıdustrial Real-Time Basic	EWICS TC2 81-8, CEC, DG III (1981)				
\MAC Product Guide, Hardware	ECA Publication (1981, new issue in preparation)				
\MAC Book 1982	EUR 8500 Vol. 1 & 2 Contains EUR 4100, EUR 4600, EUR 6100, EUR 6500, ESONA/RTB/03, ESONE/SR/01, ESONE/IML/01 EUR 4100/Supp., COMPEX		CAMAC Instrumentation and Interface Standards, Wiley/ IEEE, 1982 ISBN 0-471-89737 Lib. Congress 8185060, SH08482 IEEE Standards 583, 683, 596, 595, 726, 675, 758, and the definition of CAMAC terms. All documents have been updated.		
anguage Definition of CATY 1	UKCA/CATY 1/01 (1977)				
.anguage Definition of CATY 2	UKCA/CATY 2 (1979)				

[a]Documents to be used together with ESONE/DOC/2.
[b]No longer available and superseded by ANSI/IEEE Std. publications.
[c]Revised issues of corresponding documents in the US.
[d]Revisions for corresponding original documents in the US.

manufacturers in the CAMAC business in Europe. CAMAC is also manufactured in the Soviet block, People's Republic of China, and Japan, as well as in the US. Table 1.2 lists the major North American CAMAC manufacturers and Table 1.3 lists the major European manufacturers. Finally, CAMAC Associations as user organizations have been established in recent years in the UK, in Denmark, and in Poland.

Thus in the space of 12 years, CAMAC has grown from a single document, with little hardware developed, to a considerable, if specialized, industry. As can be seen in Table 1.1a, the specifications are adopted by most continents; indeed, 48 countries with CAMAC were counted in 1976.

Table 1.2

NORTH AMERICAN CAMAC MANUFACTURERS

AEON Systems, Inc.
1704 Moon NE
Albuquerque, NM 87112
(505) 298-0942

Bira Systems Inc.
2404 Comanche NE
Albuquerque, NM 87107
(505) 881-8887

Data Design Corporation
P. O. Box 611
Olney, MD 20832
(301) 774-5099

Data-Sud Systems/USA Inc.
5025 South Ash, Suite B-5
Tempe, AZ 85282
(602) 345-0940

DSP Technology Inc.
48500 Kato Road
Fremont, CA 94538
(415) 675-7555

GEC Canada Ltd
5112 Timbolea Blvd
Mississauga
Ontario, Canada L4W 255
(416) 624-8300

Interface Standards
45845-A Warm Springs Blvd.
Fremont, CA 94539
(415) 657-5100

Joerger Enterprises Inc.
166 Laurel Rd.
East Northport, NY 11731
(516) 757-6200

Jorway Corporation
27 Bond Street
Westbury, NY 11590
(516) 997-8120

Kinetic Systems Corporation
11 Maryknoll Drive
Lockport, IL 60441
(815) 838-0005

LeCroy Research Systems Corporation
700 South Main St.
Spring Valley, NY 10977
(914) 425-2000

Nova
1044 East La Cadena Drive, Suite 200
Riverside, CA 92501
(714) 781-7332

Phillips Scientific
13, Ackerman Ave.
Suffern, NY 10901-7197
(914) 357-9417

Table 1.3

MAJOR EUROPEAN CAMAC MANUFACTURERS

Borer Electronics AG
Postfach, 4501 Solothurn
Switzerland
065-311131

Camtec Electronics Ltd.
18, Melton Street
Leicester LE1 3NA
England
(0533) 537534

Creative Electronics Systems SA
70, Route du Pont-Butin
Case Postale 122
1213 Petit-Lancy 1
Switzerland
(022) 925745

Data-Sud Systems, SA
Mini Parc-Los Lavandes, Bat. 8
Rue Croix Verte Prolongee
Zolad BP 1067
34007 Montpellier
France

Dornier System
Vertrieb Electronik, ABT VCE
Postfach 1360
7990 Friedrichshafn
West Germany

Enertec Schlumberger
Branche Instrumentation Nucleaire
1 Parc des Tanneries
67680 Lingolsheim
France

Hytec Electronics, Ltd
Ladbroke House
Woodley
Reading RG5 4DX UK
(0734) 697973

INCAA BV
Postbox 211
Amsfoortseweg 15
7313 AE Apeldoorn
Netherlands
(055) 55 12 62

Kinetics Systems
3 Chemin de Tavernay
1218 Geneva
Switzerland
(022) 984445

Laben
Via Bassini 15
20133 Milan
Italy
(02) 2366551

LeCroy SA
Rue Cardinal-Journet 27
1217 Meyrin 1
Switzerland
(022) 823355

Nuclear Enterprises Ltd
Bath Road, Beenham
Reading RG7 5PR, UK
(073 521) 2121

Polon Nuclear Equipment Establish.
00-086 Warsaw
Bielanska 1
Poland

SEN Electronique
Case Postale 39
1211 Geneva 13
Switzerland
(022) 442940

Sension Scientific Ltd.
Denton Drive Industrial Estate
Northwich
Cheshire CW9 7LU, UK
(0606) 44321

Siemens AG
Bereich Mess-und Prozesstechnik
Postfach 21 1080
7500 Karlsruhe 21
West Germany

Karl Wehrmann AG
Spaldingstrasse 74
2000 Hamburg 1
West Germany
(040) 242475

Wenzel Elektronik
Schwedensteinstrasse 3
8000 Muenchen 82
West Germany

1.2 Why Use CAMAC?

Because

- CAMAC is a high-level hardware. That is, CAMAC is computer independent and thus can be used somewhat like high-level languages such as Fortran, BASIC, etc.
- CAMAC means that system building and maintenance resources are used effectively. This comes about because in choosing CAMAC much of the system design already exists, and thus this effort is saved. Maintenance resources are saved because, over several independent systems, only one stock of spares is needed, and engineers and technicians need be trained in only one system.
- CAMAC is modular. Thus systems can be built up and extended in easily understood units.
- CAMAC is an international standard. This results in a stable system over the coming years--very important for people investing in systems that will have a long life.
- CAMAC reduces overall system costs, assuming that a professional approach is used in building the system, and assuming that labor costs are realistic.
- CAMAC facilitates graceful upgrades. Consider a power station monitoring and control system. Power stations have a life of at least 30 to 40 years, whereas the life of a computer is substantially less. Using a computer independent input/output (I/O) system means that one can replace the computer without disturbing the I/O system and vice versa. Because in the past, computer-specific I/O systems have been used, the UK Central Electricity Generating Board still has germanium transistors custom made so that long-obsolete computers can be kept operational!

Having convinced ourselves of the need for CAMAC, now let us look at it in detail. Figure 1.2 shows the relationships between the hardware standards of CAMAC. From the foundation standard, EUR 4100/IEEE 583, springs two standards for connecting CAMAC crates together in a system: EUR 4600/IEEE 596 for the parallel branch and EUR 6100/IEEE 595 for the serial highway. This latter standard has a document giving a recommended practice for driving a serial highway: ESONE/SD/02 and DOE/EV-0006. Both highway systems can be extended to

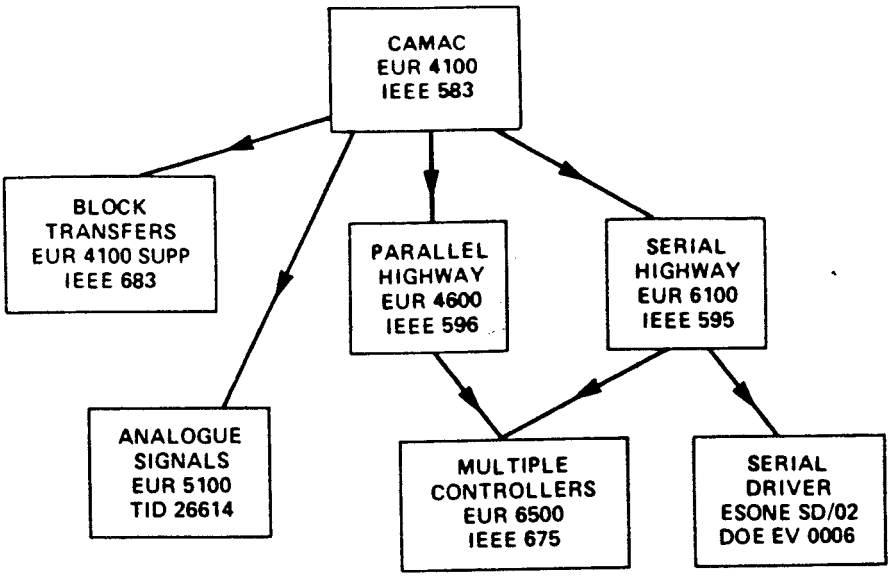


Fig. 1.2.
The CAMAC hardware standards and their relationships.

have multiple controllers in a crate: EUR 6500/IEEE 675. Finally there are standards for block transfers: EUR 4100 supp/IEEE 683 and analogue signals EUR 5100/TID-26614.

2. THE BASIC CAMAC STANDARD

2.1 What, Simply, is CAMAC?

CAMAC is a standard mechanism for connecting signals to a computer. In principal anything can be connected by CAMAC, including standard computer peripherals like disks, terminals, printers, etc. In practice it is mostly used for connecting experiments and other equipment to data-acquisition and control computers.

The basic standard (IEEE 583) defines a crate, dataway, and a plug-in. The crate houses the modules and provides them with convenient power. The dataway provides the means by which a controller, situated in the crate, is able to transfer data and control information to and from the plug-ins in the crate. With this standard, one can attach one or more crates of CAMAC plug-ins to a computer. The nature of the controller is entirely up to the designer, so

long as it operates on the CAMAC dataway according to the specification. Figure 2.1 shows this in a highly schematic form. Just as the interface between CAMAC and a computer is largely left to the designer, so is the design of the CAMAC module to connect signals, from and to the outside world, to CAMAC, and hence to the computer. The control station shown is such because it has access to the Look-at-Me (LAM) and to the station-addressed lines of the crate (Fig. 2.2). All other lines are bussed across the crate or are simply individual patch pins.

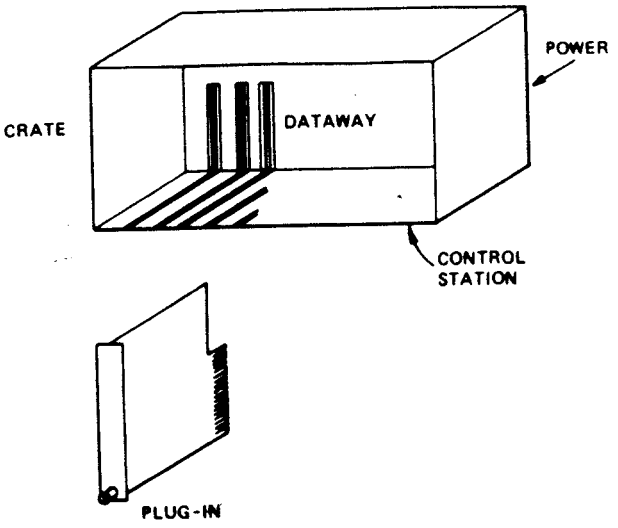


Fig. 2.1.
The basic CAMAC standard.

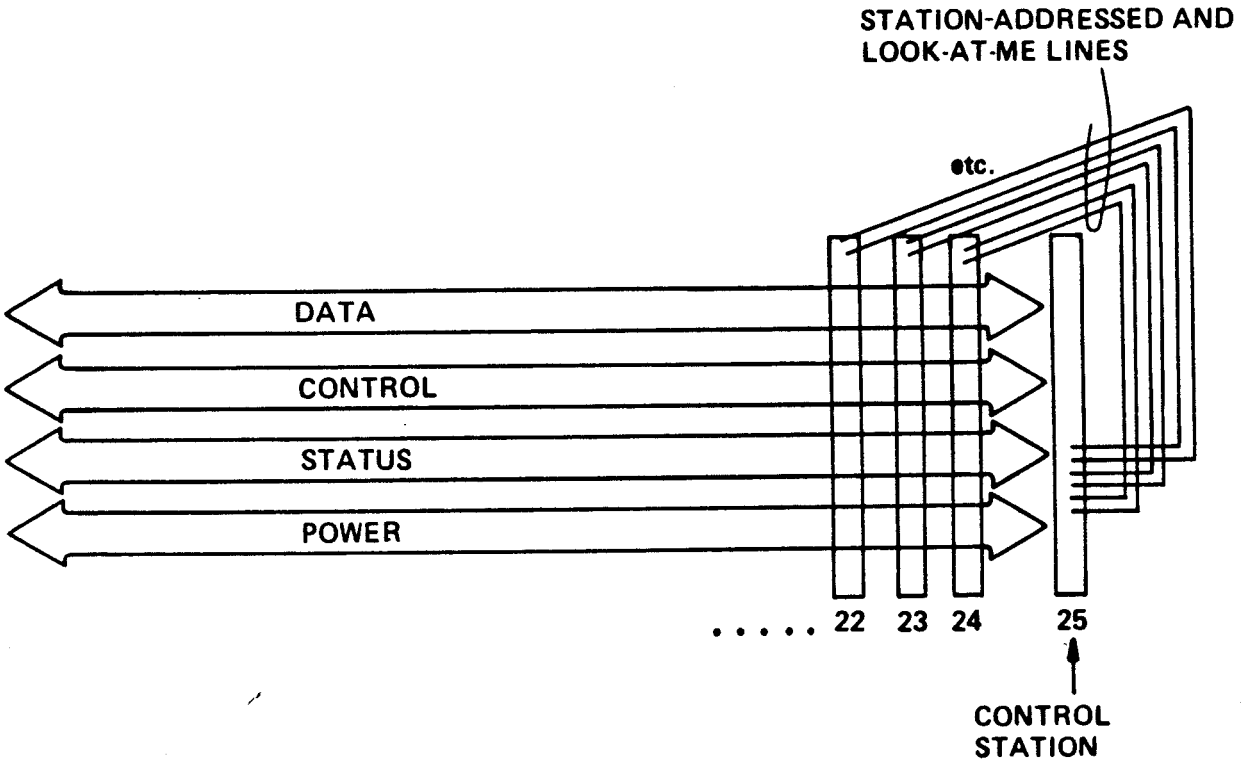


Fig. 2.2.
The dataway.

How does one connect a CAMAC crate to a computer? One solution (Fig. 2.3) is to buy or build a CAMAC controller that has the computer I/O bus plugged into its front panel. This controller is thus specific to that brand of computer and tends to be inflexible and a little expensive because of short production runs. The distance between the computer and the CAMAC crates is limited also by the computer I/O bus.

Another solution is to use a CAMAC Highway (Fig. 2.4) and build a computer to CAMAC-highway coupler. This allows the system to be easily expanded and will give other flexibility as we will see. The CAMAC crate controllers are now computer independent and thus less expensive. Table 2.1 compares the two defined CAMAC highways with the serial highway being used in bit or byte mode. These highways will be discussed in much more detail later.

Finally, can one have more than one controller in a CAMAC crate? All that is needed is

(1) access to the station-addressed and LAM lines, and

(2) arbitration between the controllers to resolve conflicts of access.

The former is achieved by the Auxiliary Controller Bus (ACB), a rear connection, and the latter by a request/grant chain established between front-panel connectors of each controller. This is described in Chapter 6, and shown schematically in Fig. 2.5.

In the remainder of this chapter we examine the basic standard, IEEE 583.

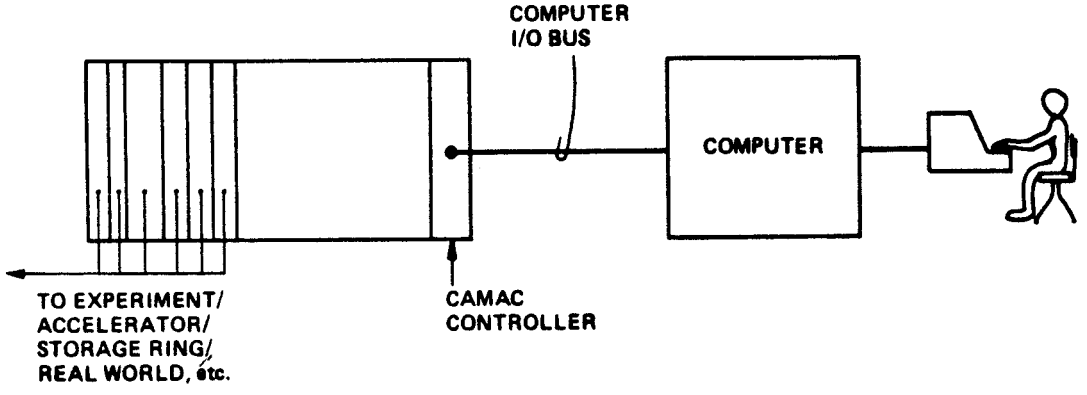


Fig. 2.3.
A single-crate system.

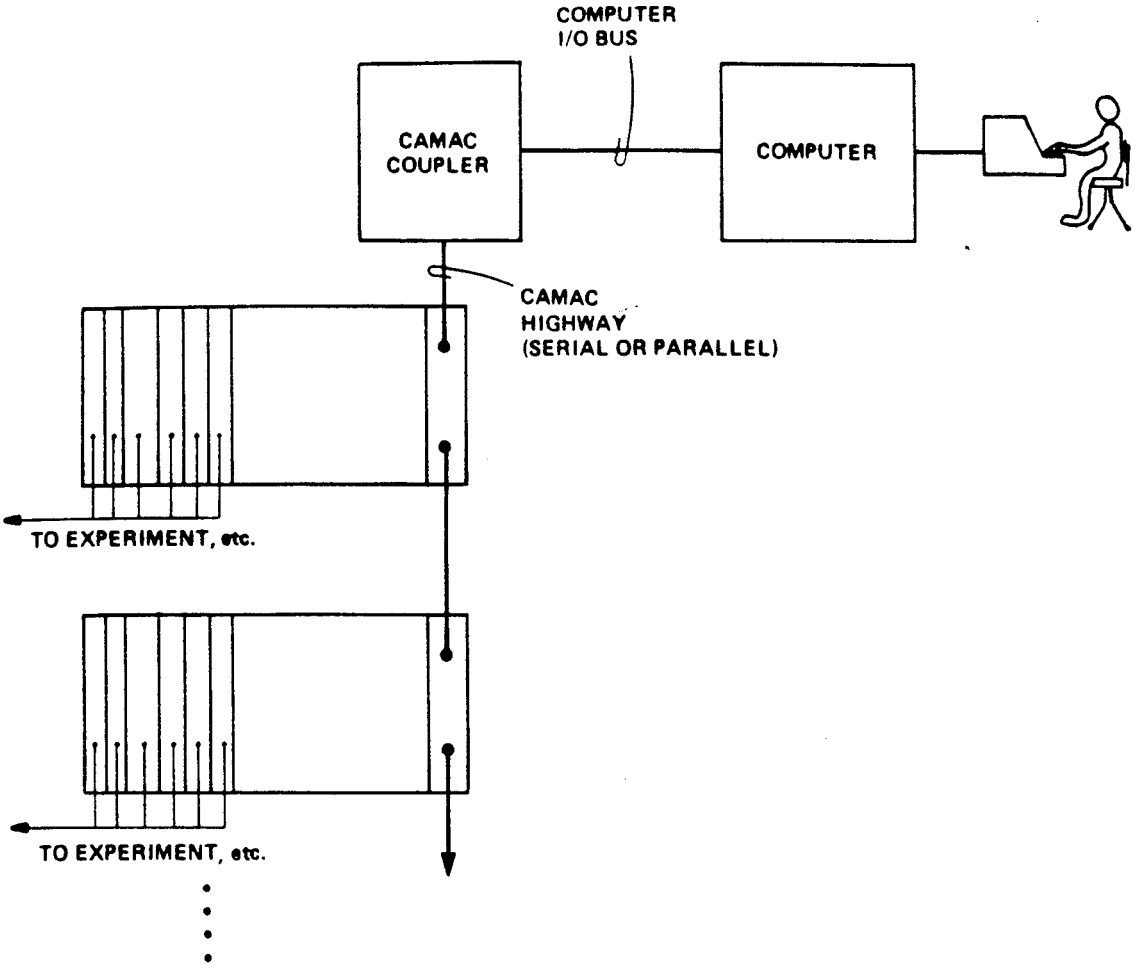


Fig. 2.4.
A multicrate CAMAC system.

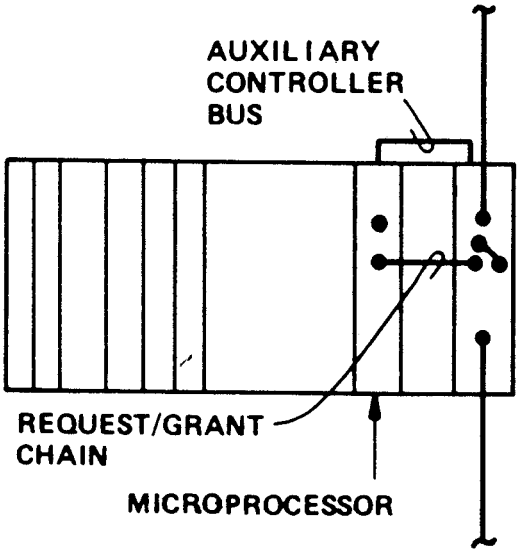


Fig. 2.5.
Multiple controllers.

Table 2.1

CAMAC HIGHWAYS

	Parallel Branch IEEE 596	Serial Highway IEEE 595	
		Bit	Byte
No. of wires	132	4	18
Communications	Handshake	Synchronous	Synchronous
Speed	$\geq$1.6 μs	$\geq$28 μs	$\geq$3.6 μs
No. of crates	1 to 7	1 to 62	1 to 62
Multicrate addressing?	Yes	No	No
Controller type	A1/A2	L2	L2
Multimodule addressing?	Yes	No	No
Multiple controllers?	Yes (with A2)	Yes	Yes
Error detection?	No	Yes	Yes

2.2 IEEE 583

This standard defines the mechanical and electrical properties of a crate and, to some extent, its power supply. Also defined are the mechanical properties of a module and controller and the electrical and logical properties of these sufficient for compatibility. The crate design also allows the insertion of normal NIM modules with a power adaptor. We need only consider the mechanical and electrical aspects but briefly; they are normally the concern of the manufacturer not the average user. Figures 2.6 to 2.13 indicate the mechanical aspects of a CAMAC crate and module. The main point is that the crate and module be very fully specified. Figure 2.6 shows that optionally UNC tapped holes can be provided to allow NIM modules to be inserted. Figure 2.12 defines the NIM-CAMAC adaptor necessary so that a NIM module can pick up power from the crate. Another point to note from Fig. 2.6 is that the number of stations (s) is not fixed but only specified to be $\leq$25. Figure 2.8 shows a side view cross section of the crate with the 86-way connector of the dataway. Above this connector is the free-access area. American usage of CAMAC has placed a

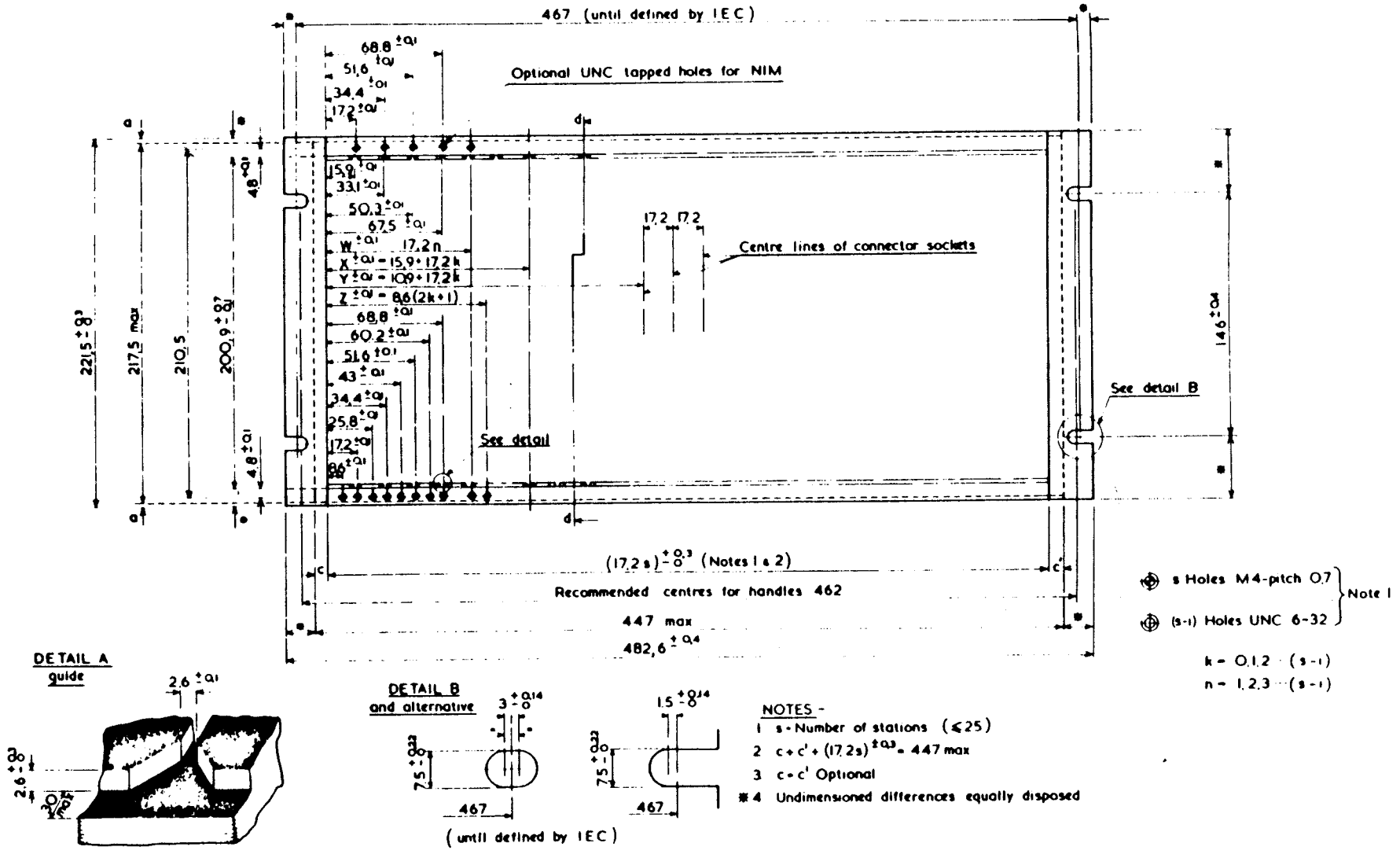
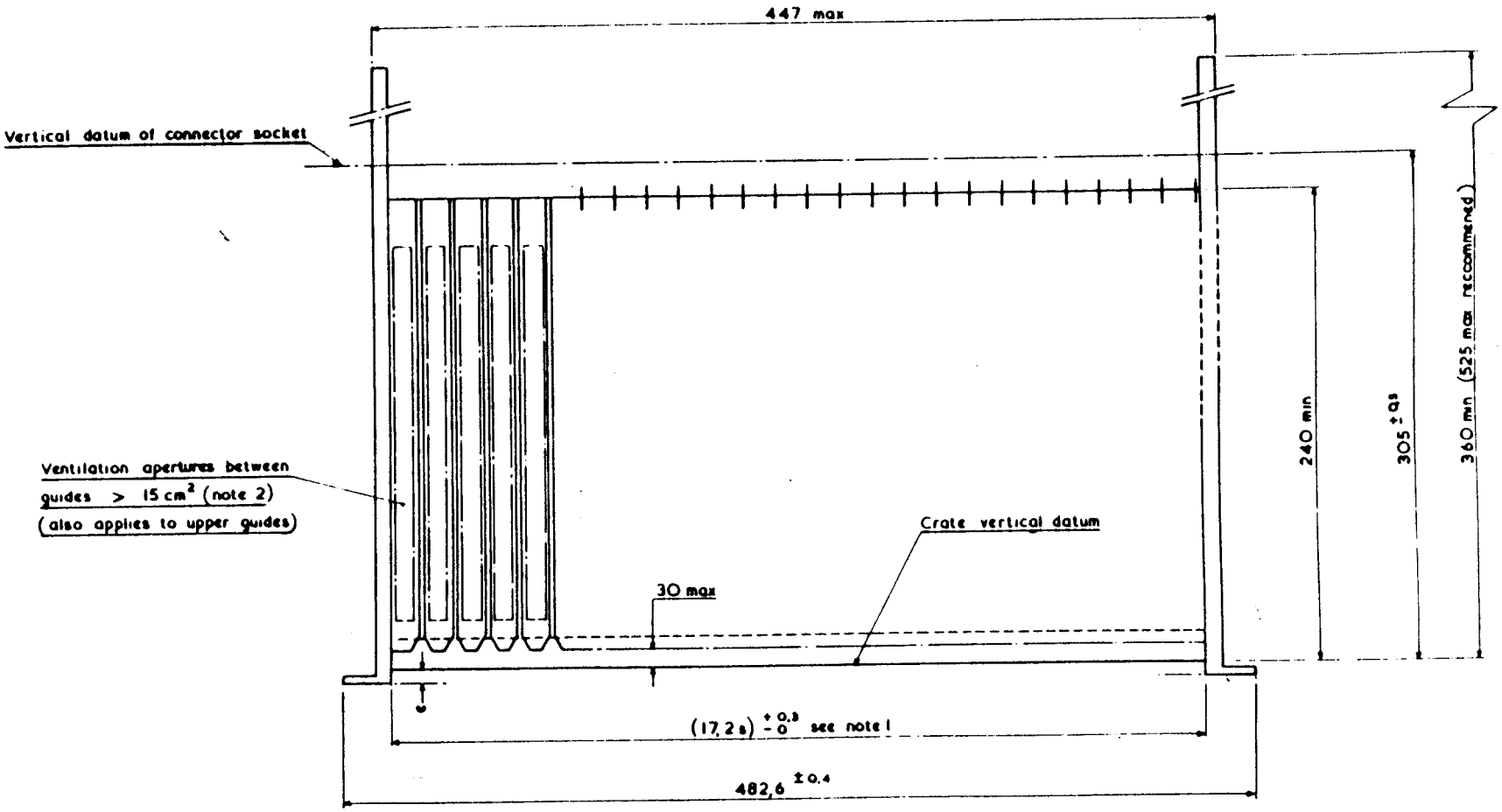


Fig. 2.6.
Unventilated Crate: front view.



NOTES

1 s = No of stations for plug-in units

2 Ventilation apertures between guides to be as long as possible

3 Typically e = 3.5 ±0.6

Fig. 2.7.
Plan view of lower guides in crate.

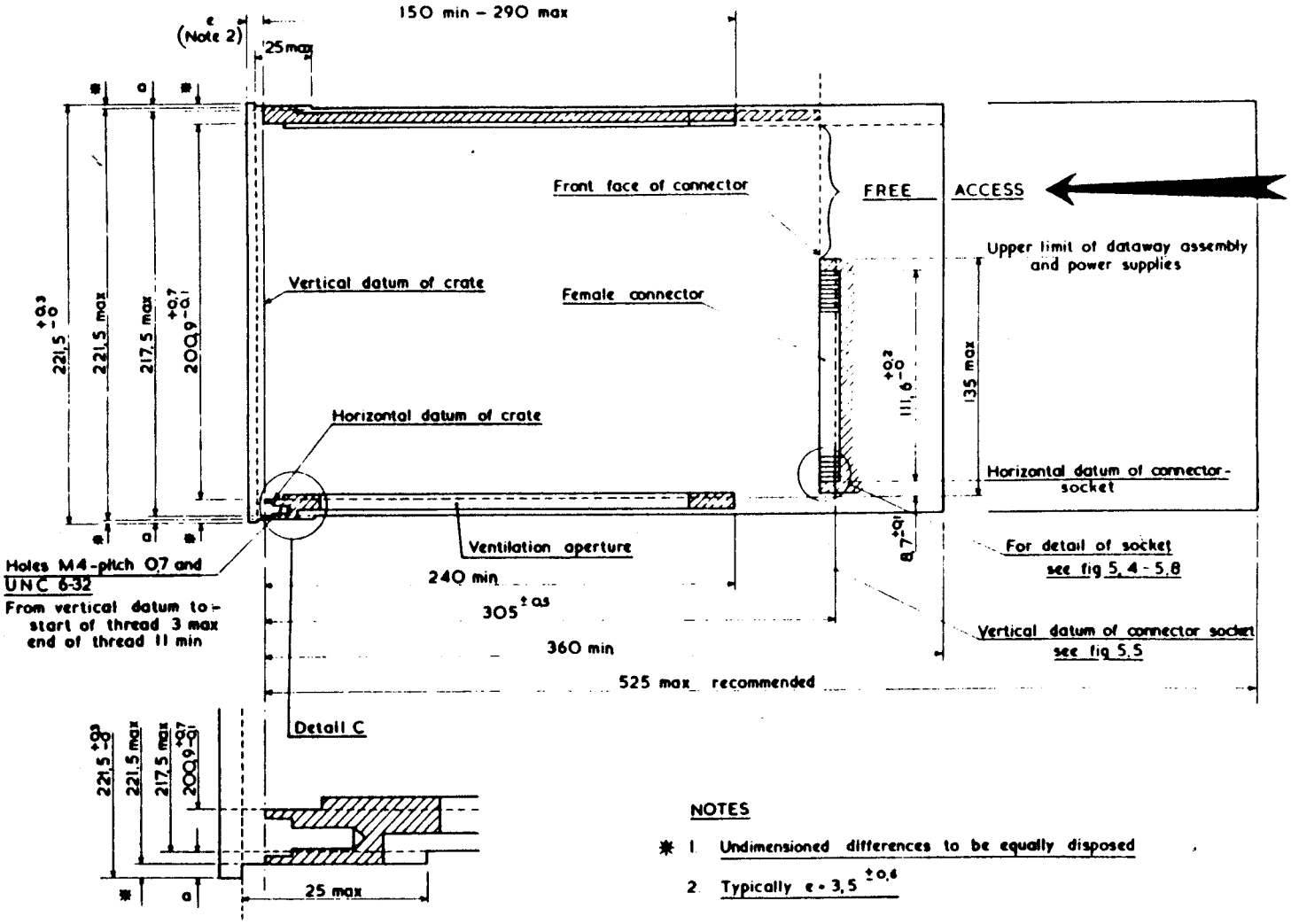
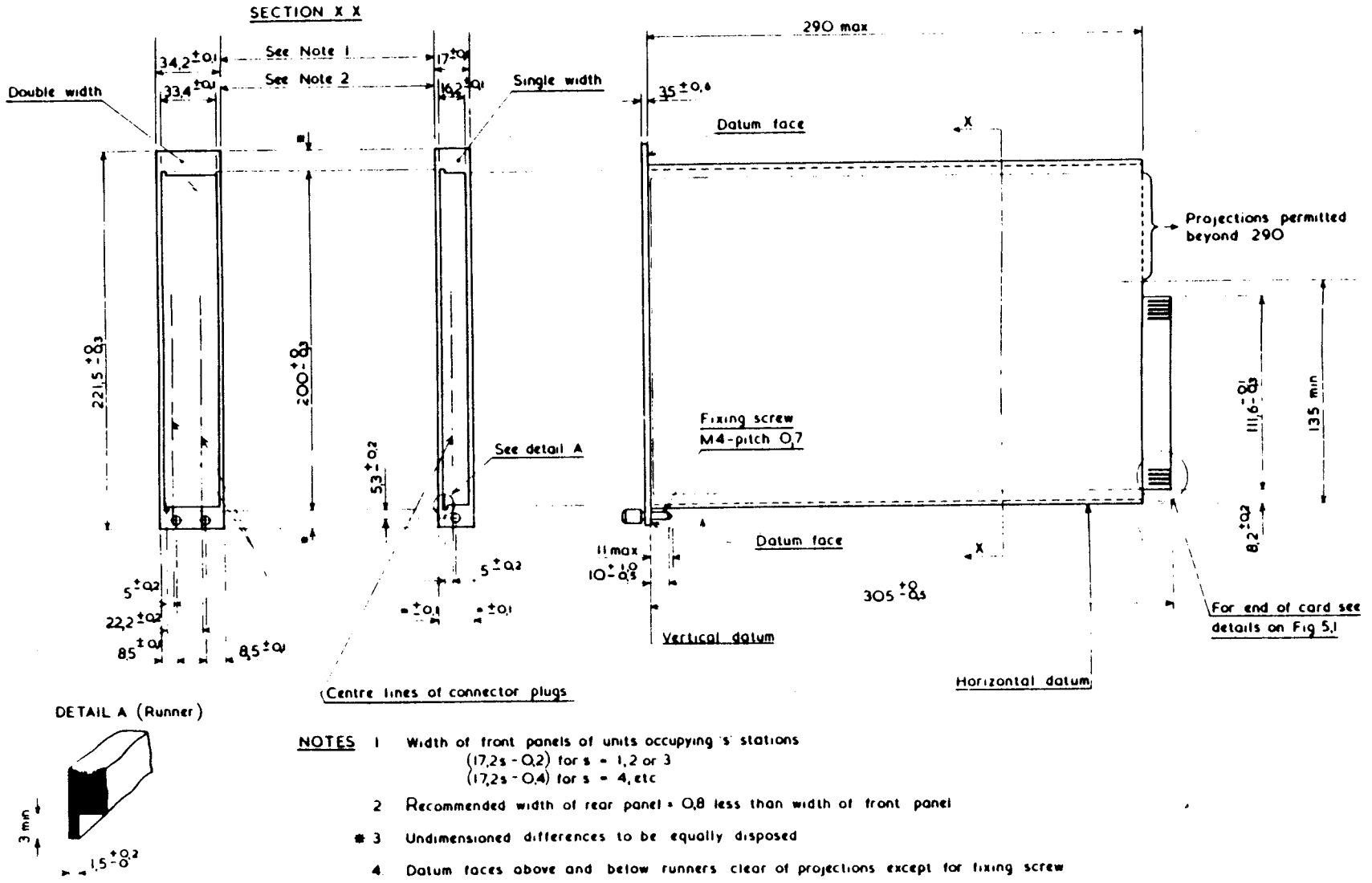


Fig. 2.8.
Crate side view: Section d-d.



NOTES
1. Width of front panels of units occupying 's' stations
   (17,2s - 0,2) for s = 1, 2 or 3
   (17,2s - 0,4) for s = 4, etc
2. Recommended width of rear panel = 0,8 less than width of front panel
* 3. Undimensioned differences to be equally disposed
4. Datum faces above and below runners clear of projections except for fixing screw

Fig. 2.9.
Plug-in unit: side and rear views.

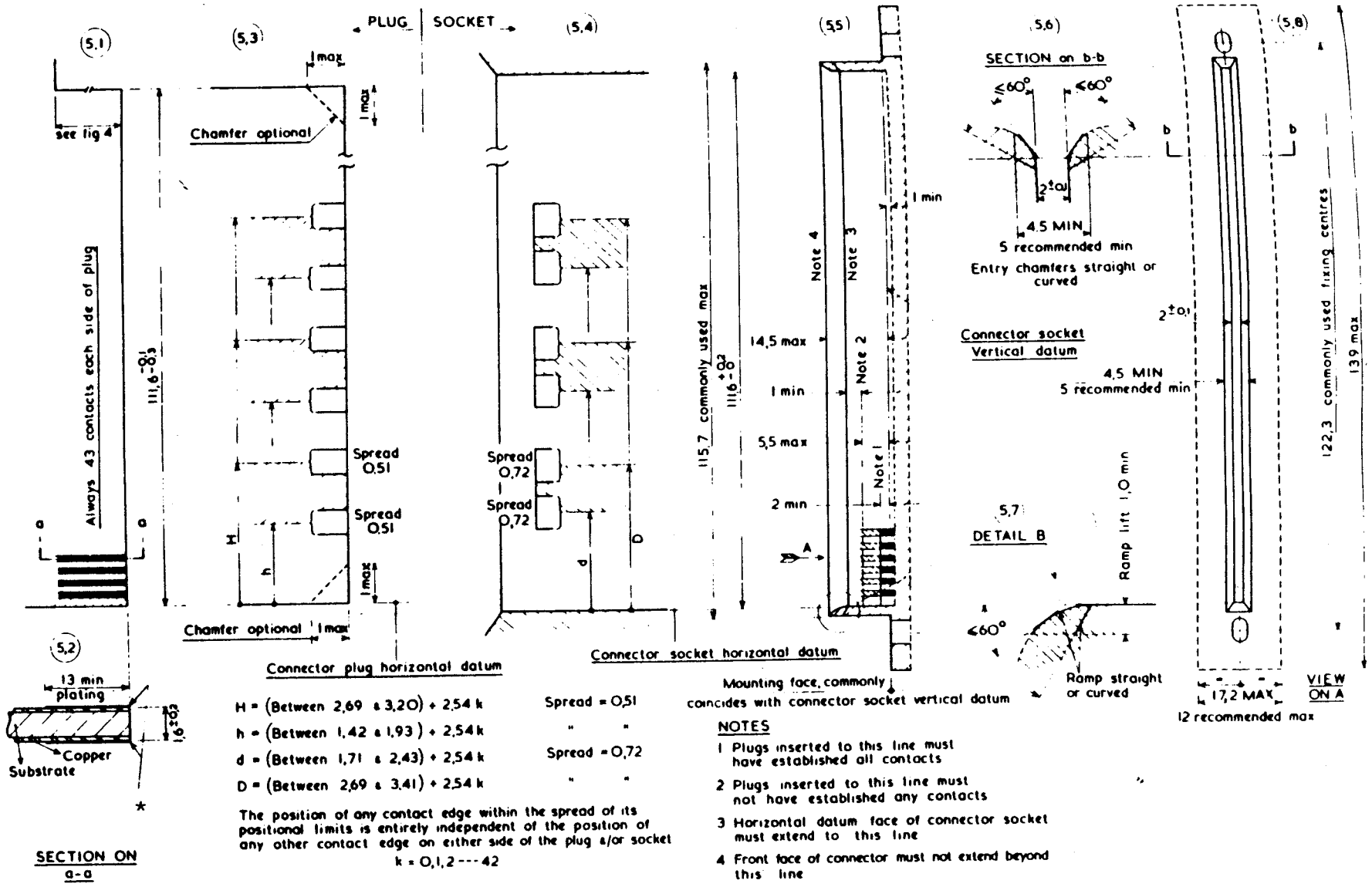
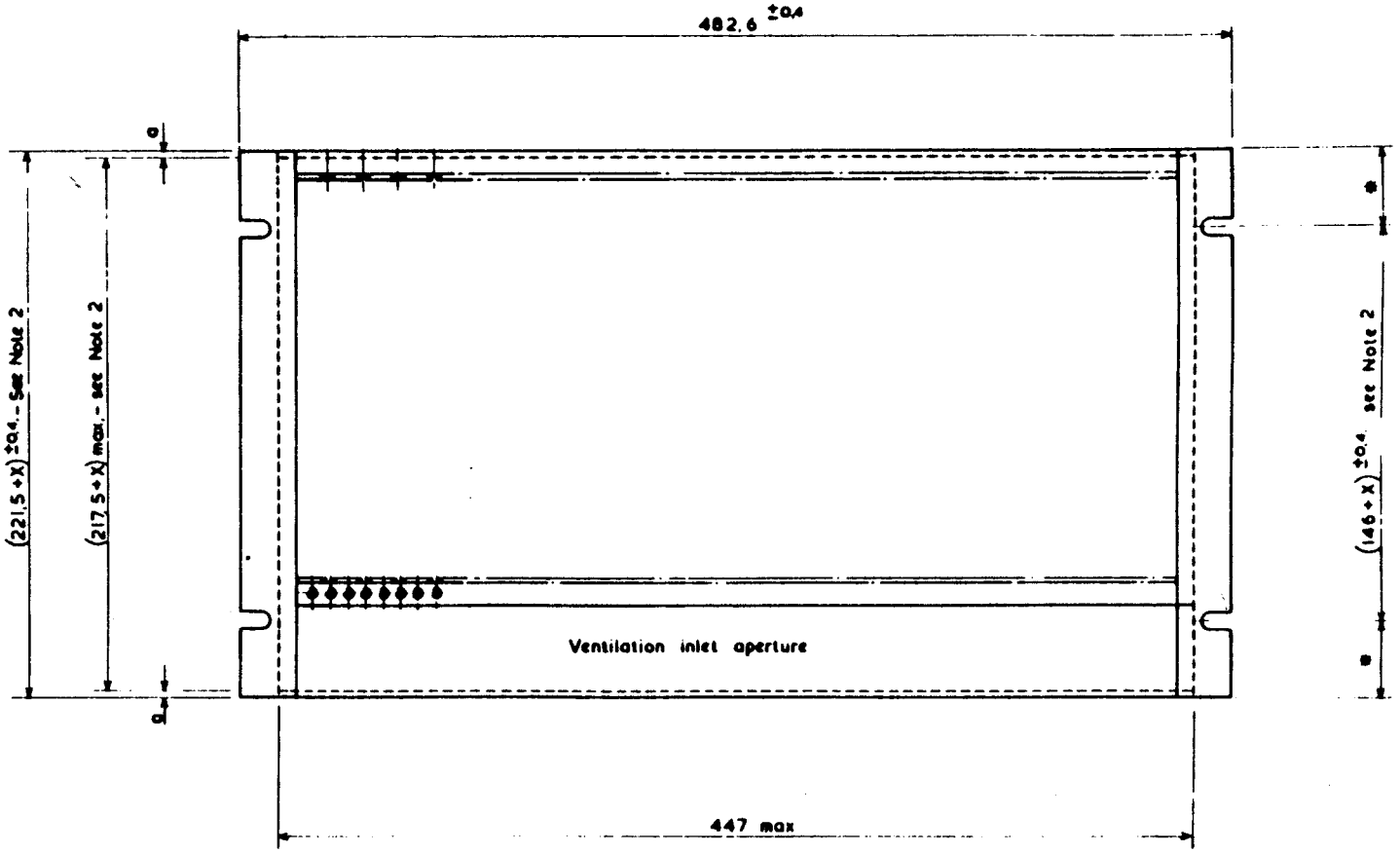


Fig. 2.10.

Dataway connector: 5.1 to 5.3 connector plug; 5.4 to 5.8 connector socket. The asterisk refers to the chamfer of 0.3 by 0.3 mm that is optional on the vertical edge of the dataway connector.



Notes: 1. For all details not shown see Fig. 2.6
2. X = 44, 45 L, where L = 1, 2, 3, etc.
* 3. Undimensioned differences to be equally disposed.

Fig. 2.11.
Ventilated crate: front view.

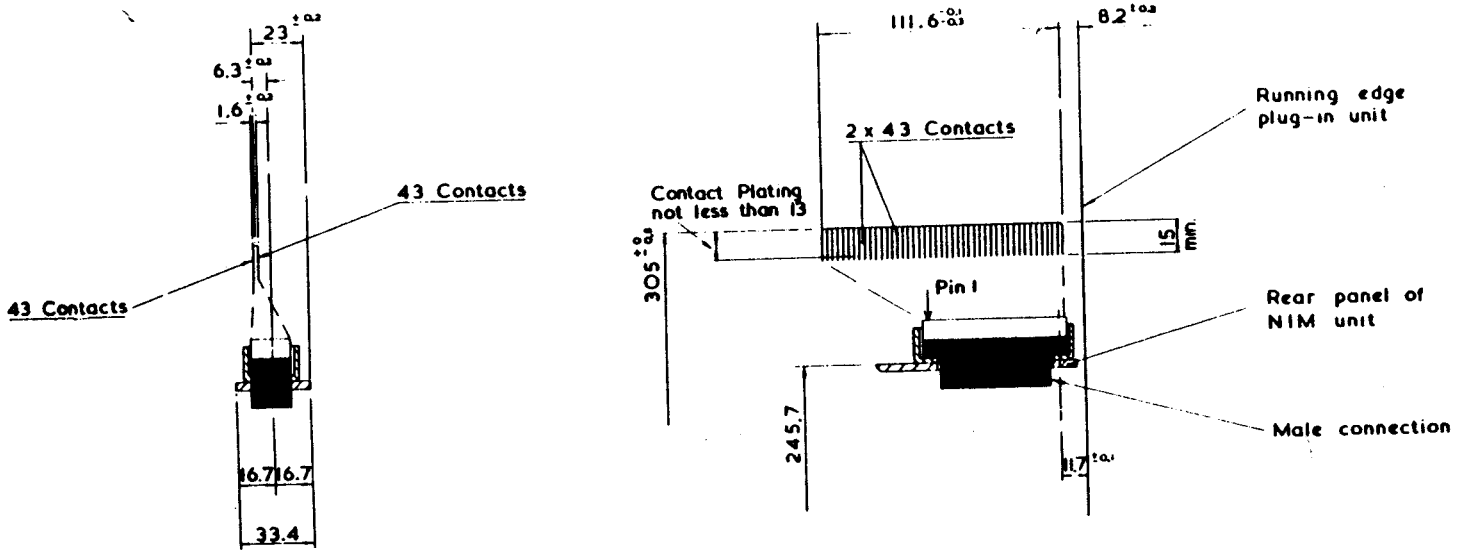


Fig. 2.12.
Adaptor for NIM units.

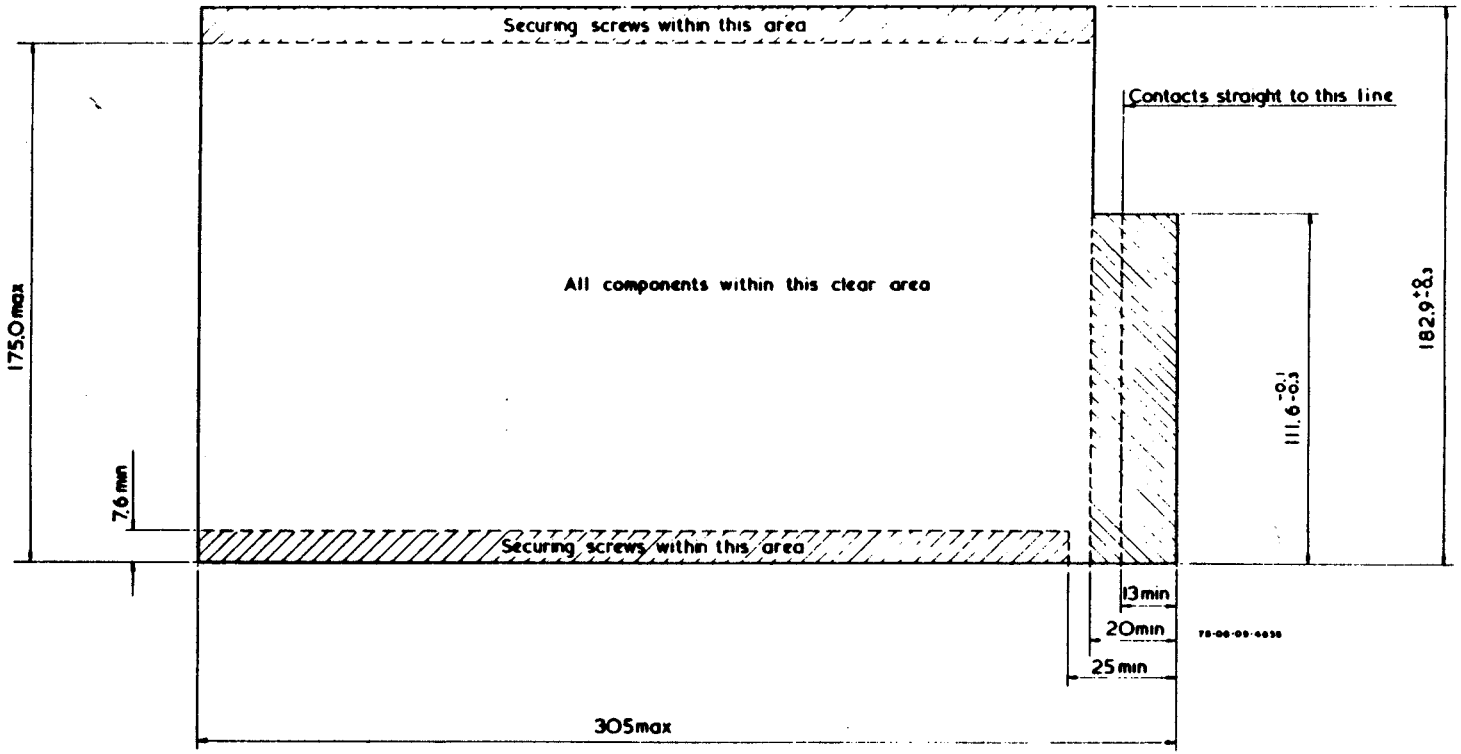


Fig. 2.13.
Typical printed wiring card.

36-way 0.1-in. pitch edge connector in this location as an I/O connector. This is, however, outside the standard and has not been implemented in Europe. It does enable I/O as well as dataway connections to be made automatically as the module is inserted.

2.3 The Dataway

Having disposed of the mechanical aspects of IEEE 583, let us now look at the electrical and logical aspects of the dataway and hence the modules. The first point to make is that the dataway is entirely passive; that is, it consists only of connectors, a printed circuit card, and/or wires. There are no resistors, capacitors, transistors, etc. The connectors are 86-way 0.1-in. pitch connectors, and up to 25 are used to make up the standard dataway. Table 2.2 lists the lines in a general way, Table 2.3 lists the voltage standards for logic signals on the dataway, and Table 2.4 lists the current standards and the pull-up locations for the various lines. The main point is that all current paths have been considered, so that the system will work in a typical environment. Also, modules other than controllers receiving signals can only attach one TTL gate input to a line for each width of the module--thus the lines usually need to be buffered at the module. This establishes the electrical standards for the dataway.

2.3.1 Use of the Dataway Lines. Table 2.5 lists the dataway usage briefly. A fuller description of each line follows.

2.3.2 Station Number, N. Each normal station is addressed by a signal on an individual station number line, N_i, which comes from a separate contact at the control station. Stations are numbered in decimal from the left-hand end as viewed from the front, beginning with Station 1. There is no restriction on the number of stations that can be addressed simultaneously, although multiple addressing is not commonly done. The serial crate controller Type L2 has no mechanism for multiple addressing.

2.3.3 Subaddress A8, A4, A2, A1. Different sections of the module are addressed by signals on the four bus lines, A. These signals are decoded in the module to select one of up to sixteen subaddresses, A(0) to A(15). The

Table 2.2

THE DATAWAY

24	READ lines (R1 - R24)	
24	WRITE lines (W1 - W24)	
5	FUNCTION lines (F1 - F16)	
4	SUB-ADDRESS lines (A1 - A8)	
1	STATION ADDRESSED line (N)	Individual lines
1	LOOK-AT-ME line (L)	Individual lines
1	RESPONSE line (Q)	Individual lines
1	COMMAND ACCEPTED line (X)	Individual lines
1	INHIBIT line (I)	Individual lines
1	CLEAR line (C)	Individual lines
2	STROBE lines (S1 & S2)	Individual lines
5	PATCH lines (P1 - P5, P1 & P2 bussed, P3 - P5 individual pins)	Individual lines
1	INITIALIZE line (Z)	Individual lines
1	BUSY line (B)	Individual lines
6	POWER lines (±6 V ±12 V ±24 V)	Individual lines
2	SUPPLEMENTARY POWER lines (±6 V, Y1 & Y2)	Individual lines
3	RESERVED POWER lines	Individual lines
1	CLEAN EARTH line	Individual lines
2	POWER RETURN lines	Individual lines

Table 2.3

THE LOGIC OF THE DATAWAY

	0 State (V)	1 State (V)
Accepted at input	+2.0 to +5.5	0 to 0.8
Generated at output	+3.5 to +5.5	0 to 0.5

Note:
- This is negative logic.
- Lines are wire-or'ed i.e. drivers cannot have active pull-ups as in standard TTL.
- The levels correspond to TTL.

Table 2.4

STANDARDS FOR SIGNAL CURRENTS THROUGH DATAWAY CONNECTORS AND FOR PULL-UP CURRENT SOURCES

<table>
<tr><td>Designation of Dataway Signal Line</td><td>N</td><td>L</td><td>Q, R, X</td><td>W, A, F, B, Z, C, I</td><td>S1, S2</td></tr>
<tr><td rowspan="2">Line in "1" state at +0.5 V
Minimum current sinking capability (current drawn from line) of each unit generating the signal.</td><td rowspan="2">6.4 mA</td><td colspan="2" rowspan="2">16 mA</td><td colspan="2">Controllers
1.6 (25-s) mA
36.8 mA typical</td></tr>
<tr><td>Other Units
9.6 + 1.6 (25-s) mA
48.0 mA typical</td><td>58 + 1.6 (25-s) mA
96.4 mA typical</td></tr>
<tr><td>Line in "1" state at +0.5 V
Maximum current fed into line by each unit receiving the signal.</td><td>3.2 mA each unit,
6.4 mA total
(Note 1).</td><td colspan="2">Unit with pull-up current source: 11.2 mA
Units without pull-up current source 1.6 mA each:
4.8 mA total (Note 1)</td><td colspan="2">1.6s mA</td></tr>
<tr><td>Line in "0" state at +3.5 V
Minimum pull-up capability (current fed into line) of the unit with pull-up current source.</td><td colspan="4">100 (25-s) µA
2.3 mA typical for controllers
2.4 mA typical for other units</td><td>9.9 mA</td></tr>
<tr><td>Line in "0" state at +3.5 V
Maximum current drawn from line by each unit without pull-up current source.</td><td>200 µA</td><td colspan="4">100s µA</td></tr>
<tr><td>Location of pull-up current</td><td>Unit generating the signal.</td><td>One unit receiving the signal.</td><td colspan="3">Controller</td></tr>
<tr><td>Pull-up current Ip, from positive potential
Line in "1" state at +0.5 V</td><td colspan="4">6 mA ≤ Ip ≤ 9.6 mA</td><td>38 mA ≤ Ip ≤ 58 mA</td></tr>
<tr><td>Pull-up current Ip, from positive potential
Line in "0" state at +3.5 V</td><td colspan="4">2.5 mA < Ip</td><td>10 mA < Ip</td></tr>
</table>

Where appropriate, the current passing through the dataway connector of a plug-in unit is defined as a function of the width of the unit (s stations). Values are given, as examples, for typical controllers (s = 2, control station and one normal station) and other units (s = 1).

NOTE 1: Although only the controller and one module are connected directly to each N and L line, additional units may be conntected via patch points or auxiliary connectors.

Table 2.5

STANDARD DATAWAY USAGE

Title	Designation	Contacts	Use at a Module
Command			
Station number	N	1	Selects the module (individual line from control station)
Subaddress	A1, 2, 4, 8	4	Selects a section of the module
Function	F1, 2, 4, 8, 16	5	Defines the function to be performed in the module
Timing			
Strobe 1	S1	1	Controls first phase of operation (dataway signals must not change)
Strobe 2	S2	1	Controls second phase (dataway signals may change)
Data			
Write	W1 to W24	24	Bring information to the module
Read	R1 to R24	24	Take information from the module
Status			
LAM	L	1	Indicates request for service (individual line to control station)
Busy	B	1	Indicates dataway operation in progress
Response	Q	1	Indicates status of feature selected by command
Command accepted	X	1	Indicates module is able to perform action required by command
Common Controls			Operate on all features connected to them, no command required
Initialize	Z	1	Sets module to a defined state (accompanied by S2 and B)
Inhibit	I	1	Disables features for duration of signal
Clear	C	1	Clears registers (accompanied by S2 and B)
Nonstandard Connections			
Free bus lines	P1, P2	2	For unspecified uses
Patch contacts	P3 to P5	3	For unspecified interconnections, no dataway lines
Mandatory Power Lines			Crate is wired for mandatory and additional lines
+24 V dc	+24	1	
+ 6 V dc	+6	1	
- 6 V dc	-6	1	
-24 V dc	-24	1	
0 V	0	2	Power return
Additional Power Lines			Lines are reserved for the following power supplies:
+12 V dc	+12	1	
-12 V dc	-12	1	
Clean Earth	E	1	Reference for circuits requiring clean earth
Supplementary (-6 V & +6 V)	Y1, Y2	2	
Reserved	Undesignated	3	
Total		86	

subaddress lines must be fully decoded by the module. In general, the use of subaddresses is the choice of the module designer, but a good design here can simplify the understanding of the module.

2.3.4 Function F16, F8, F4, F2, F1. These lines define the function to be performed by the module at the specified subaddress. These signals must be fully decoded in the module to give 32 separate functions, F(0) to F(31). Some functions are defined; some are unreserved; some are reserved. The functions F(0) to F(7) are all read functions; F(16) to F(23) are all write functions; the remaining functions are control or dataless functions. Detailed description of these functions comes later.

2.3.5 Strobe Signals S1 and S2. These bussed timing or strobe signals must be generated by the controller during each CAMAC cycle. Modules must not take irreversible action based on the other bussed lines on the dataway until the time of S1. This is to ensure that transients and differential settling times of the lines at the beginning of a dataway cycle do not cause ill effects. Actions concerned with the acceptance of Read, Write, Q and X lines must be initiated at the time of S1. These lines must not be changed during S1 time. S2 is to time follow-on action in the module, for example, clearing a register, and is also generated during unaddressed operations.

2.3.6 Write Lines W1 to W24. The controller generates data on these lines during each write operation. They must reach a steady state before S1 and must be maintained until the end of the operation, unless modified by S2. Strobe S1 must be used by the modules to strobe the data, unless there are strong technical reasons for choosing S2.

2.3.7 Read Lines R1 to R24. Data are set up on the read lines by the addressed module during a read operation. These lines must reach a steady state before S1 and must be maintained for the full duration of the dataway cycle, unless the state of the data source is changed by S2. The controller must initiate acceptance of this data at the time of S1 and must not take irreversible action before then.

2.3.8 Look-at-Me L. These lines, one from each station, are individual connections to separate contacts at the control station. Any module can indicate that it needs attention by generating a signal on its individual L line. Modules that occupy more than one station may indicate different demands by signals on the separate L lines of the stations it occupies. The L signal in a module may indicate demands for attention from one or more sources in a module. The further resolution of the actual source and the mandatory features for a module's LAM structure are discussed later.

When a module that is generating L=1 receives a command that may cause it to cease doing so, it must inhibit the L signal or the appropriate LAM request. This must be effective before S1 and must be maintained until the end of the dataway operation. This is often achieved by gating L with N so that whenever the module is addressed, the L=1 condition will be removed. The purpose is to remove a potential race condition where, if the L=1 is not removed until the end of S2, it might still seem to be true within the controller at the end of the cycle, even though the CAMAC cycle cleared the LAM. Note that this is an example of doing the right thing for the wrong reason! The first version of the CAMAC specification, published in 1969, required that L be gated with Busy in each module because people were concerned about noise on the dataway. This requirement avoided the race condition, but this was not recognized until the second version was published in 1972.

2.3.9 Busy B. Busy is used to interlock various aspects of the system that can compete for the dataway. A signal B=1 must be generated during each dataway cycle and during unaddressed commands, when Z or C are generated.

2.3.10 Response Q. Q is a single-bit status response from the addressed module; it may or may not be generated according to the design of the module. In test operations, Q=1 means true, present, etc. For read, write, and test LAM operations, Q must be stable by S1 time and must not change from that point to the end of the cycle. In all other operations it is permitted to change during the dataway cycle, but this may result in the Q response being missed. As with R lines, the controller must initiate action to accept the Q response at the time of strobe S1 and must not take irreversible action before then.

2.3.11 Command Accepted X. Whenever a module is addressed during a CAMAC operation, it must generate X=1 by S1 and must maintain it until the end of S2, if it recognizes the command as one it is equipped to perform--even though at that moment it may be unable to perform it. In this latter case, an X=1, Q=0 response is appropriate. The X=0 response then indicates a serious malfunction, for example, an absent module, a module lacking external connections, etc. In older modules designed before the X line was defined, the module may generate X=N.

2.3.12 Initialize Z. Z has absolute priority over all other signals. It consists of Z and S.2 asserted together (Z.S2). In response, all data and control registers must be set to defined initial state, usually zero. All LAM status registers must be reset and, if possible, all LAM requests must be disabled.

2.3.13 Inhibit I. This signal must inhibit any feature to which it is connected in the module. The signal may be gated or routed within the module. The inhibit line must be set to a logic 1 as a result of an initialized cycle (Z.S2). Thus all units able to generate I must respond to Z.S2 by generating and maintaining I=1 until specifically reset. The inhibit line is used in data-acquisition systems to, for example, inhibit scalers during no-signal periods. In practice I have not seen it used much because it is inflexible and because few modules have facilities in them to selectively enable, disable, or invert its effect. If a system is not working, quite often it is the inhibit line being set that is the cause. I suggest that, unless you plan to use inhibit, you cut the track in all modules to disconnect them from this line!

2.3.14 Clear C. The signal C.S2 must clear all registers to which it is connected. Busy and S2 must be generated with C. Gating C with S2 is a protection against noise. This signal may, as with I, be gated and routed within a module, although in practice this is not much done. While all crates come up with I=1 as a static condition leading to the problems mentioned above, C.S2 occurs by specific controller action for a short period only (Fig. 2.4); thus, it does not catch the user unawares. It is used to clear down data-acquisition modules in a single efficient operation, usually in event-based systems such as are common in high-energy physics.

2.3.15 Free Bus Lines P1 and P2. These contacts at each station are bussed together by the dataway. Any plug-in can generate or receive signals on these lines but must be provided with a means to disconnect or disable that access. Their use is entirely up to the system designer, but they have been used at CERN to hold up the dataway cycle and in the UK as DMA synchronization lines. As with all incompletely specified facilities, they must be used with care.

2.3.16 Patch Contacts P3 to P5. P3 to P5, at normal stations, and P1 to P7, at control stations, are not wired to any dataway lines. They are available for patch connections to other patch pins, to optional patch points, or to certain dataway lines to the 0V line or to external equipment. Their use must not be essential for the operation of the main features of general-purpose plug-ins. Their use does make the dataway a special-purpose one that can give trouble when systems are changed or when crates are exchanged. As before, I cannot recommend their use, especially as the actual wiring is difficult to examine in most crates because of the positioning of the power supply.

2.3.17 Power Lines and Grounds. The dataway must include lines for all power lines defined or reserved. Table 2.6 gives the specification for the power lines at the plug-in connector. In particular, note that each connector pad is rated for 3 A maximum; to provide more power, Y1 and Y2 can be assigned to -6 V and +6 V, respectively. However, current paths must be such that the return lines do not exceed 3 A per contact, making a total current to 0 V in a module, 6 A maximum. To provide the current sharing, it is recommended that two 10-mΩ resistors are put in series with the return pads, and that different sections of the board are powered separately from the two ±6-V power sources (Fig. 2.14). Modules that use Y1 and/or Y2 must be specifically marked as crates with this connection are nonstandard. The power supply and its mounting is quite outside the basic standard, although different institutions have established de-facto standards, notably CERN.

2.3.18 Cooling the Crate. Without forced cooling, the dissipation of a crate should not be above 200 W, 8 W per module. With forced cooling, the dissipation can be increased to 25 W per station. Because cool circuits mean more reliable circuits in general, good cooling is important.

Table 2.6

POWER LINE STANDARDS

Nominal Voltage on Power Line in Crate	Voltage Tolerance at Dataway Connectors	Maximum Current Loads: In the Plug-in per unit width) [See Notes (1) and (3)]	Maximum Current Loads: In the Crate [See Note (2)]	Comments
Mandatory				(1) The current carried by each contact of the dataway connector must not exceed 3 A
+24 V dc	±1.0%	1 A	6 A	(2) The total power dissipation in a crate without forced ventilation must not exceed 200 W
+ 6 V dc	±2.5%	2 A	25 A	(3) The power dissipation in each station must not exceed 8 W in general or 25 W under special circumstances
- 6 V dc	±2.5%	2 A	25 A	
-24 V dc	±1.0%	1 A	6 A	
0 V				
Additional (as required)				
+12 V dc	±1.0%			As specified in TID-20893 (Latest revision)
-12 V dc	±1.0%			

Comments:
(1) The current carried by each contact of the dataway connector must not exceed 3 A.
(2) The total power dissipation in a crate without forced ventilation must not exceed 200 W.
(3) The power dissipation in each station must not exceed 8 W in general or 25 W under special circumstances.

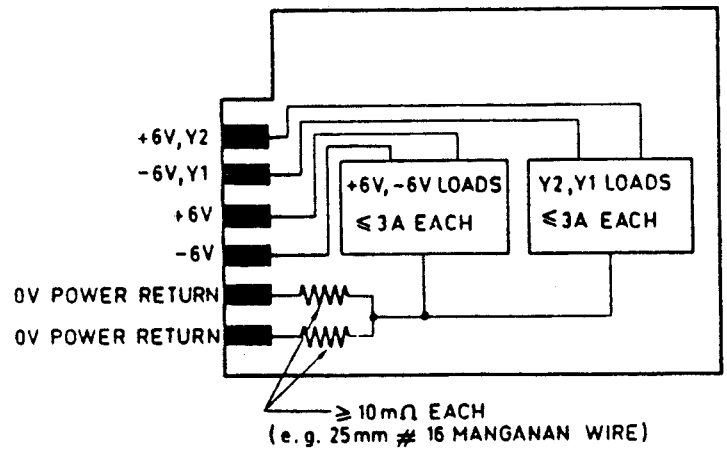


Fig. 2.14.
Typical current sharing with supplementary 6-V power.

2.3.19 Summary of the Dataway. Table 2.7 shows the pin assignments for a normal CAMAC station and serves as a summary of what has been covered above. Table 2.8 gives the same assignments for the control station, the extreme right-hand station. Here it will be seen that the R and W lines have been replaced by the individually wired L and N lines. Access to these lines makes this position priviledged and thus the control station. The only other difference is that the N and L lines from this station become P6 and P7. Because a controller also needs access to the R and W lines, it must, by definition, be at least two widths. To emphasize, no other plug-in in a CAMAC crate can initiate a dataway cycle without separate access to the controller in Station 25; neither can it respond to LAMs. Finally, Fig. 2.15 shows the wiring of a dataway schematically. The point to note here is that the bussing of the R and W lines ends at Station 24, and Station 25 has in their place wired the N and L lines.

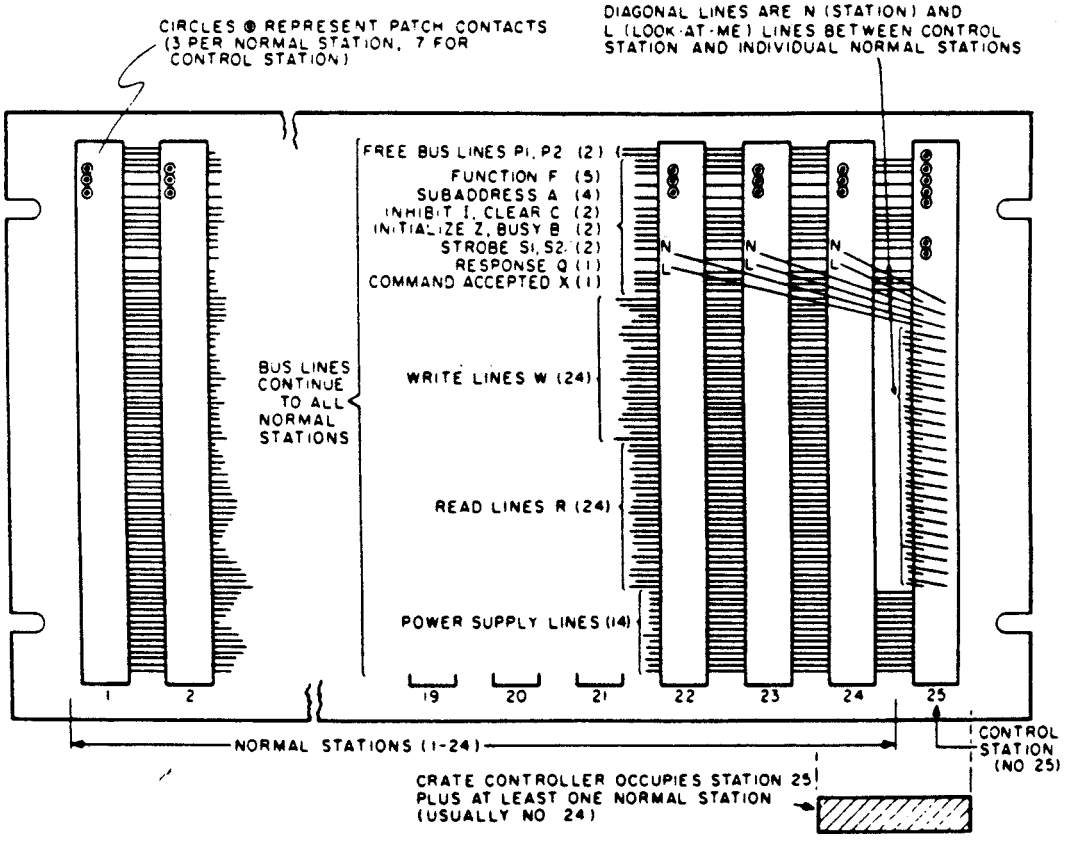


Fig. 2.15.
Dataway wiring, front view of a twenty-five station crate.

Table 2.7

CONTACT ALLOCATION AT A NORMAL STATION
(Viewed From Front of Crate)

Bus line	Free Bus line	P1	B	Busy	Bus line
Bus line	Free Bus line	P2	F16	Function	Bus line
Individual patch contact		P3	F8	Function	Bus line
Individual patch contact		P4	F4	Function	Bus line
Individual patch contact		P5	F2	Function	Bus line
Bus line	Command Accepted	X	F1	Function	Bus line
Bus line	Inhibit	I	A8	Subaddress	Bus line
Bus line	Clear	C	A4	Subaddress	Bus line
Individual line	Station Number	N	A2	Subaddress	Bus line
Individual line	LAM	L	A1	Subaddress	Bus line
Bus line	Strobe 1	S1	Z	Initialize	Bus line
Bus line	Strobe 2	S2	Q	Response	Bus line
Twenty-four Write Bus lines W1 = least significant bit W24 = most significant bit		W24	W23		
		W22	W21		
		W20	W19		
		W18	W17		
		W16	W15		
		W14	W13		
		W12	W11		
		W10	W9		
		W8	W7		
		W6	W5		
		W4	W3		
		W2	W1		
Twenty-four Read Bus lines R1 = least significant bit R24 = most significant bit		R24	R23		
		R22	R21		
		R20	R19		
		R18	R17		
		R16	R15		
		R14	R13		
		R12	R11		
		R10	R9		
		R8	R7		
		R6	R5		
		R4	R3		
		R2	R1		
Power Bus lines	-12 V dc	-12	-24	-24 V dc	
	Reserved (C)		-6	-6 V dc	
	Reserved (A)			Reserved (B)	
	Supplementary -6 V	Y1	E	Clean Earth	
	+12 V dc	+12	+24	+24 V dc	
	Supplementary +6 V	Y2	+6	+6 V dc	
	0 V (Power Return)	0	0	0 V (Power Return)	

Table 2.8

CONTACT ALLOCATION AT A CONTROL STATION
(Viewed From Front of Crate)

Individual patch contact		P1	B	Busy	Bus line
Individual patch contact		P2	F16	Function	Bus line
Individual patch contact		P3	F8	Function	Bus line
Individual patch contact		P4	F4	Function	Bus line
Individual patch contact		P5	F2	Function	Bus line
Bus line	Command Accepted	X	F1	Function	Bus line
Bus line	Inhibit	I	A8	Subaddress	Bus line
Bus line	Clear	C	A4	Subaddress	Bus line
Individual patch contact		P6	A2	Subaddress	Bus line
Individual patch contact		P7	A1	Subaddress	Bus line
Bus line	Strobe 1	S1	Z	Initialize	Bus line
Bus line	Strobe 2	S2	Q	Response	Bus line
Twenty-four individual LAM lines (L1 from Station 1, etc.)		L24	N24	Twenty-four individual Station Number lines, (N1 to Station 1, etc.	
		L23	N23		
		L22	N22		
		L21	N21		
		L20	N20		
		L19	N19		
		L18	N18		
		L17	N17		
		L16	N16		
		L15	N15		
		L14	N14		
		L13	N13		
		L12	N12		
		L11	N11		
		L10	N10		
		L9	N9		
		L8	N8		
		L7	N7		
		L6	N6		
		L5	N5		
		L4	N4		
		L3	N3		
		L2	N2		
		L1	N1		
Power Bus lines	-12 V dc	-12	-24	-24 V dc	
	Reserved (C)		-6	-6 V dc	
	Reserved (A)			Reserved (B)	
	Supplementary -6 V	Y1	E	Clean Earth	
	+12 V dc	+12	+24	+24 V dc	
	Supplementary +6 V	Y2	+6	+6 V dc	
	0 V (Power Return)	0	0	0 V (Power Return)	

2.4 The Timing of a Dataway Cycle

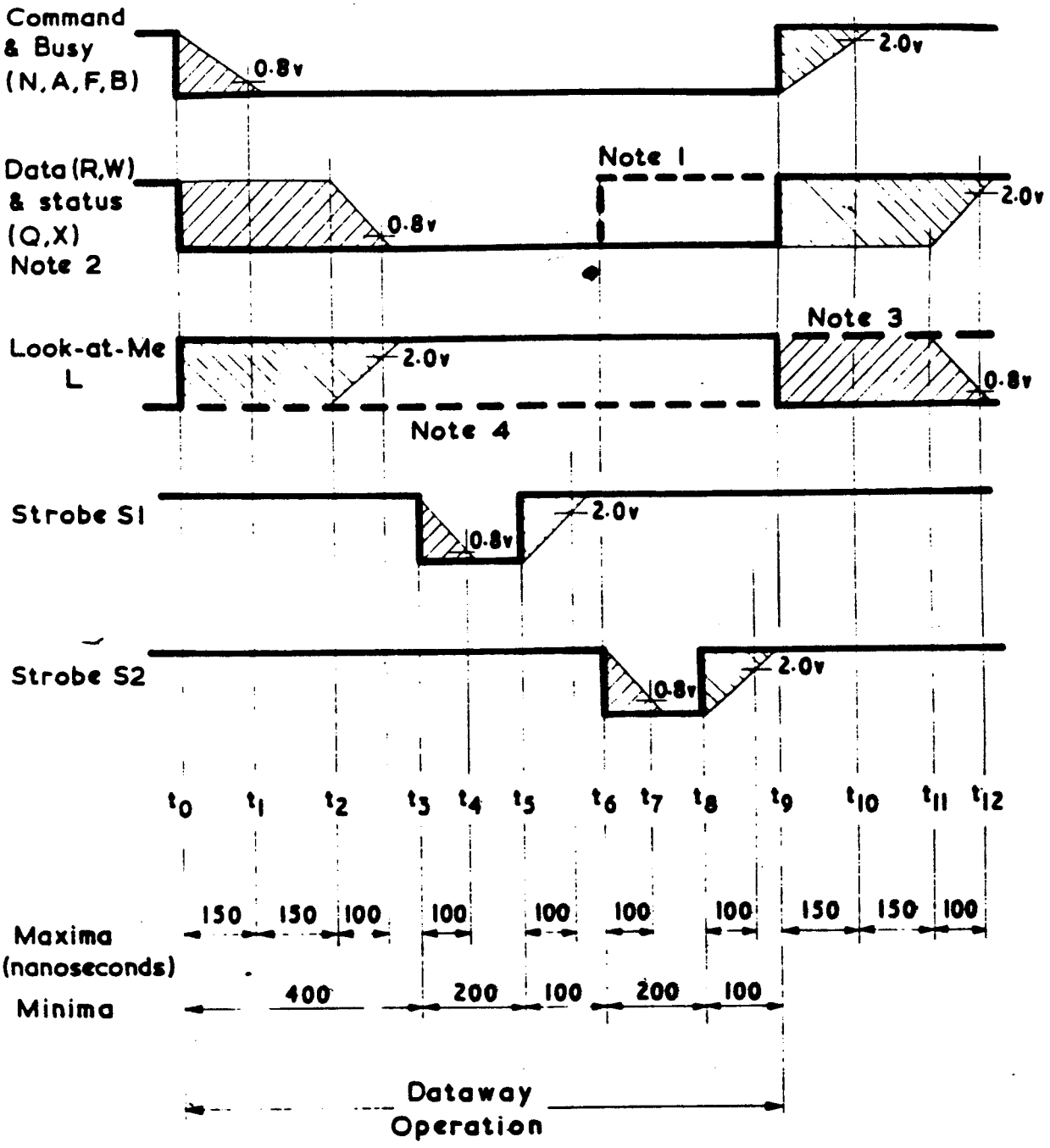
Figure 2.16 shows the timing of an addressed dataway cycle. The first step is that the controller asserts N, A, F, and B. When N is true at the module, not all of the A and F lines will yet be in their final state; therefore, at first the module might well decode other operations--recall that it must take no irrevocable action until the start of S1. After a maximum of 150 ns, these lines are stable; there is a further 150 ns for the controller or module, depending on the command, to establish R or W lines Q and X. There is then a further 100 ns minimum before the start of S1, which is 200 ns long. At S1, all lines are assumed settled, and the operation takes place; data and responses are clocked, etc. A minimum of 100 ns after the end of S1, S2 occurs. (Note the consideration of the degradation of these pulse shapes allowed for in the timing.) The end of the cycle is at a time at least 100 ns after S2 is removed by the controller. Thus, the minimum CAMAC cycle length is 1 μs, but no maximum is defined, there being five windows in the cycle before t_3, t_5, t_6, t_8, and t_9. Also shown in the figure is the behavior of the L line during the operation.

Figure 2.17 shows a Z or C operation on the dataway. The operation is reduced to a minimum of 750 ns by the removal of the need for the R, W, Q, and X lines being asserted and allowed to settle. Again, there is no maximum cycle, with windows before t_6, t_8, and t_9.

It can thus be seen that the absolute maximum bandwidth of a CAMAC crate is 3 Mbytes/second. If one recognizes that few modules use S2, 300 ns can be trimmed off, and the other timings also can be trimmed slightly to double the overall bandwidth. This is, however, outside the standard and for specialists only.

2.5 Addressing

We have already seen that a crate has 24 addressable stations and 23 usable ones. Each plug-in has up to 16 subaddresses available, so that the total address space of a crate is 24 x 16 = 384. For data, this is effectively doubled because data registers are divided into Group 1 and Group 2 registers. Group 1 registers contain primary data, whereas Group 2 registers are control registers. This separation of the address space is done to simplify rapid data



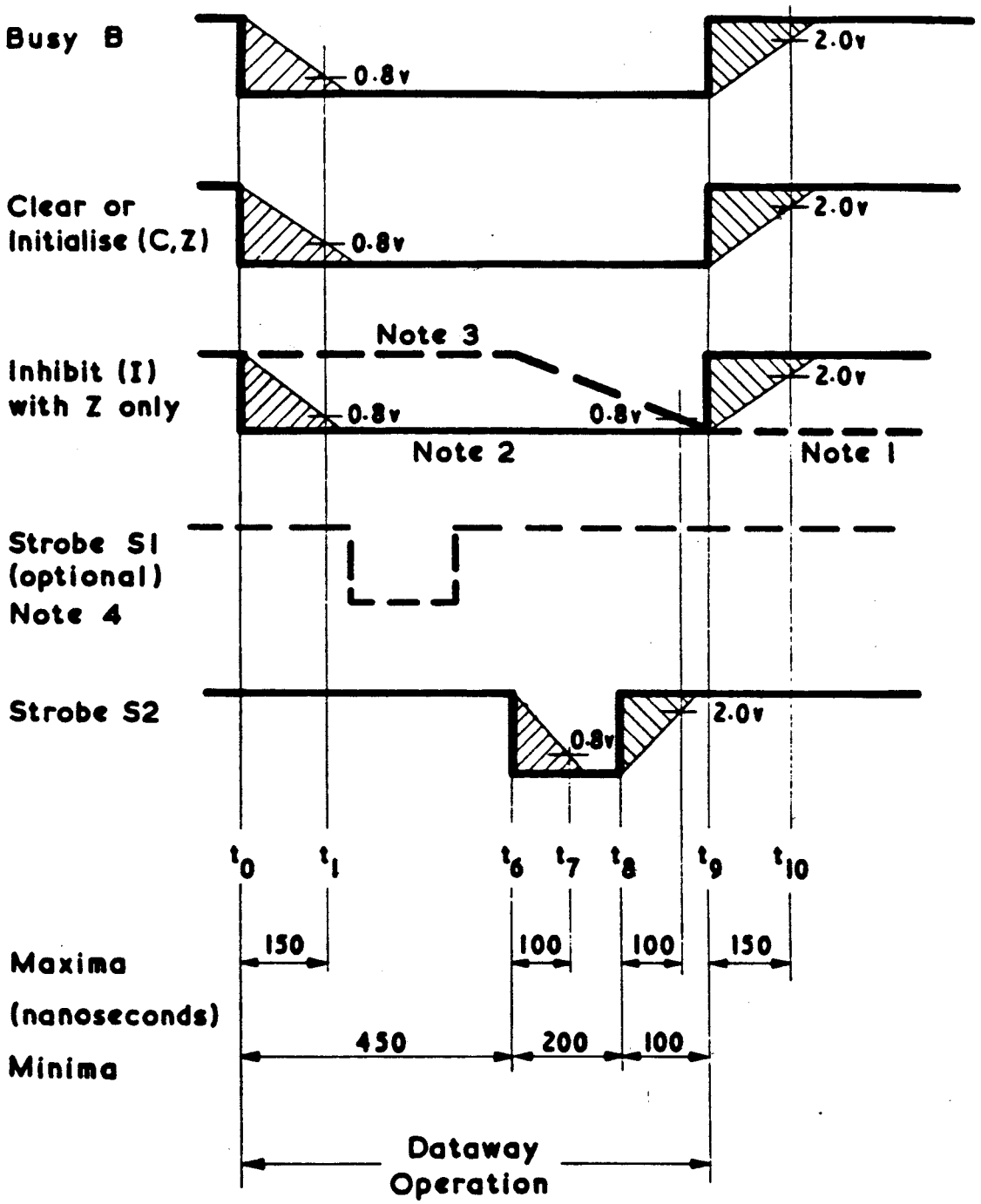
Note 1: Data & status may change in response to S2.

Note 2: During some operations Q may change at any time.

Note 3: LAM status may be reset during operation.

Note 4: L signal may be maintained during operation.

Fig. 2.16.
Timing of a dataway command operation.



Notes:
1. I preferably maintained.
2. I accompanying Z.
3. I generated in response to Z.S2.
4. Other times as in Fig. 2.16.

Fig. 2.17.
Timing of a dataway unaddressed operation.

acquisition. As we will see, the distinction is made by using different function codes.

2.6 The Function Codes

The 31 function codes are listed in Table 2.9, together with a brief description of their meaning. An overall point to note is that many functions have a defined meaning; this is a strong point in CAMAC's favor, because it puts a uniformity on module designs without restricting their function. A more modern approach is to have but two basic functions, read and write, and to deal in bits, read or written, to enable, test, mask, etc.

Selective Set functions cause the bits in the target register, identified by ones in the word transferred, to be set. Target register bits, corresponding to zeros in the transferred word, are unchanged; thus, a logical OR operation. Selective Clear works in a similar way except that the indicated bits are cleared rather than set.

Bit Set and Bit Clear work in a similar way except that only a single bit is operated on at a time. The position of this bit is identified by the lower 5 bits of the data word transferred, which contain the encoded bit number. We will see later the usefulness of this.

In general, the full meaning of these instructions is left to the module designer; although, as we will see, there are good and bad designs.

2.7 LAMs

LAMs and their handling will crop up again and again in this Primer. At this elementary level, we need only deal with their handling up to Station 25, the controller being, at this stage, undefined. Note that in all other respects, the CAMAC plug-in behaves in strict response to commands from the crate controller--the LAM line is the one exception to this. Except for the case mentioned above, a module can assert its L line when it pleases, quite independent of what the crate controller and its computer is doing at that moment. For all practical purposes, a LAM from a module is an interrupt to the computer. However, although a module is free to assert its LAM when it pleases, it can only remove its LAM as a result of specific CAMAC action. Therefore, how are LAMs handled in the module and what aspects of that handling are mandatory?

Table 2.9

THE FUNCTION CODES

+	Code F( )	Function	Use of R and W Lines	Function Signals F16	F8	F4	F2	F1	Code F( )
RD1	0	Read Group 1 register	Functions using the R lines	0	0	0	0	0	0
RD2	1	Read Group 2 register		0	0	0	0	1	1
RC1	2	Read and Clear Group 1 register		0	0	0	1	0	2
RCM	3	Read Complement of Group 1 register		0	0	0	1	1	3
	4	Nonstandard		0	0	1	0	0	4
	5	Reserved		0	0	1	0	1	5
	6	Nonstandard		0	0	1	1	0	6
	7	Reserved		0	0	1	1	1	7
TLM	8	Test LAM	Functions not using the R or W lines	0	1	0	0	0	8
CL1	9	Clear Group 1 register		0	1	0	0	1	9
CLM	10	Clear LAM		0	1	0	1	0	10
CL2	11	Clear Group 2 register		0	1	0	1	1	11
	12	Nonstandard		0	1	1	0	0	12
	13	Reserved		0	1	1	0	1	13
	14	Nonstandard		0	1	1	1	0	14
	15	Reserved		0	1	1	1	1	15
WT1	16	Overwrite Group 1 register	Functions using the W lines	1	0	0	0	0	16
WT2	17	Overwrite Group 2 register		1	0	0	0	1	17
SS1	18	Selective Set Group 1 register		1	0	0	1	0	18
SS2	19	Selective Set Group 2 register		1	0	0	1	1	19
BS2	20	Nonstandard--Bit Set Group 2 register[a]		1	0	1	0	0	20
SC1	21	Selective Clear Group 1 register		1	0	1	0	1	21
BC2	22	Nonstandard--Bit Clear Group 2 register[a]		1	0	1	1	0	22
SC2	23	Selective Clear Group 2 register		1	0	1	1	1	23
DIS	24	Disable	Functions not using the R or W lines	1	1	0	0	0	24
XEQ	25	Execute		1	1	0	0	1	25
ENB	26	Enable		1	1	0	1	0	26
TST	27	Test Status		1	1	0	1	1	27
	28	Nonstandard		1	1	1	0	0	28
	29	Reserved		1	1	1	0	1	29
	30	Nonstandard		1	1	1	1	0	30
	31	Reserved		1	1	1	1	1	31

[a]These do not form a part of 583 but derive from DOE/EV-0006.

+The mnemonics derive from TID-26615.

The minimum mandatory features are to be able to test or distinguish individual LAM sources in a module, as well as in the state of the overall LAM. This is either done with a TLM F(8) function or by reading a bit pattern from a Group 2 register. The function TLM is mandatory for the overall LAM. The other mandatory feature required is that Initialize must clear all LAMs and must clear LAM masks if they are implemented. A LAM mask is a register with as many bits as there are LAMs and where each bit is used to pass or to block the corresponding LAM. By convention, writing a 1 to a mask enables the LAM.

Figure 2.18 shows the ways of handling LAM sources in a module. The two main choices are by registers; thus, there is a particular data bit for each LAM, or by subaddress for each LAM. The first method works well when there are several LAM sources in a module; the latter works well when there are but few. The LAM status register is necessary to clock the LAM source, so that it only is cleared by specific CAMAC action. The LAM request is formed by the logical AND of the LAM status and the LAM mask registers, together with the overall LAM mask; then, the individual LAM requests are OR'd to form the overall request, internal L. Finally, the internal L is gated by N or other signal to gate off the L signal during CAMAC operations that might clear it.

If LAMs are handled within a module as registers, then IEEE 583 states that they should be Group 2 registers at the following subaddresses:

- LAM status register A(12)
- LAM mask register A(13)
- LAM request register A(14)

The important things to note on LAMs are that

- the hardware of the crate can only recognize one LAM per station; therefore, when a plug-in has more internal LAMs, it must provide the means to identify the particular LAM.
- LAMs can be masked both individually and in an overall way in the module.
- some LAM functions are mandatory for plug-ins that generate LAMs.

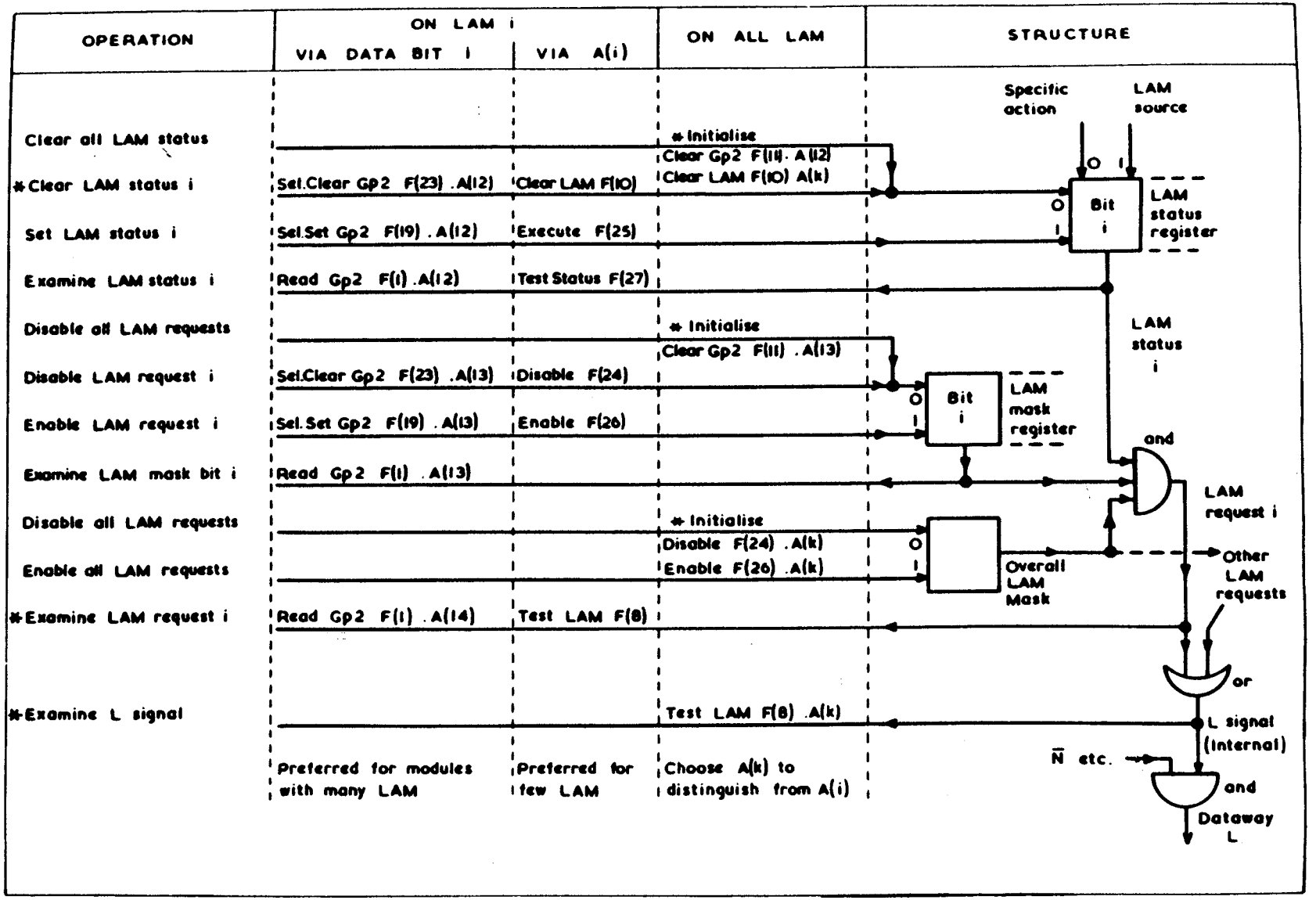


Fig. 2.18.
Some LAM structure options.

3. CAMAC SOFTWARE

3.1 The Software Problem

The essence of good software support is to make it as easy as possible for the user to express what he wants the computer to do. Good software is

- readable, and
- reasonably efficient.

The first point because software maintenance is a big expense in the life of a piece of code, and even the author will have trouble coming back to the code after a period. The second point only stresses "reasonably" because machines are cheaper than people and so, for most problems, ease of writing and speed in debugging are overriding considerations. I accept the fact that there will always be the extreme problems where the above order is reversed, and that "reasonably" can be deleted or even changed to "very" but I feel that people kid themselves that their position is the latter far more often than is really the case. One can buy a VAX for the average programmer's annual salary, plus overheads. Even so, I don't suggest sloppiness!

It follows that good software support (languages, editors, compilers, linkers, and debugging aids, if any,) is

- easy to use;
- encourages, if not imposes, readable code;
- encourages, if not imposes, correct code;
- encourages, if not imposes, structured code;
- simplifies debugging by allowing the user to do it in the terms of the language used; no hex or octal dumps as a minimum!

The general rule is that the easier a system is to use, the better the use will be made of it.

3.2 I/O Programming

Figure 3.1 shows a piece of code picked out of the waste basket in the IBM room at a high-energy physics Laboratory. It is the old familiar Fortran, and I have underlined the I/O statements. The point to note is that the user is not concerned about the mechanism by which the bytes or characters specified at label 182 actually get onto the printer--that will be different for each machine and operating system. All that is taken care of, by the compiler and

```
C --- READ IN CONTROL DATA
      READ(5,1)TITUL
    1 FORMAT(15A4)
      PRINT10,TITUL
   10 FORMAT(1H1,18A4)
      READ(5,2)IN
    2 FORMAT(10I5)
      CALL USER(1)
      IIN=IN(1)
      J=1
      NVT=0
  182 FORMAT(/'      MONTE CARLO DATA FROM NEW SET : DEVICE NO. ',I2)
C ---
C --- EVENT LOOP
 1000 READ(IIN,ERR=80,END=90)BF,WT,ERRWT,PP
      NRED=NRED+1
C --- IF I/O DEVICES HAVE RIGHT NUMBERS THEN SET UP ANGLE IN BF(20)
      INPUT=J
      IF(INPUT.LT.1)GO TO 472
      IF(INPUT.GT.6)GO TO 472
      BF(20)=SPECT(INPUT)
  472 CONTINUE
```

Fig. 3.1.
A typical Fortran program with I/O statements underlined.

the operating system. Hidden in there are interrupts, I/O instructions, and various other things that need not worry this physicist. The same comments apply to the READ statements. As an aside, it is also notable how unreadable this code is and how few comments there are.

Figure 3.2 shows all this schematically, with the jumble of machine-dependent facilities that, to first order, do not concern the user. Whatever machine is used, the theory is that the output will be the same because a machine independent high-level language was used. CAMAC is a machine independent high-level hardware, so Fig. 3.3 shows the new situation with CAMAC. Whereas from a high-level language the user had control only over the characters and their placing on the print-out, now the user also has intimate control of the CAMAC modules. This is because the CAMAC data formats, which are defined by the module designer, and the CAMAC commands and addresses are independent of the computer system used. Certainly, different systems will generate different code, but that will always be the case. However, an extra parameter is that the CAMAC code generated is a function not only of the computer

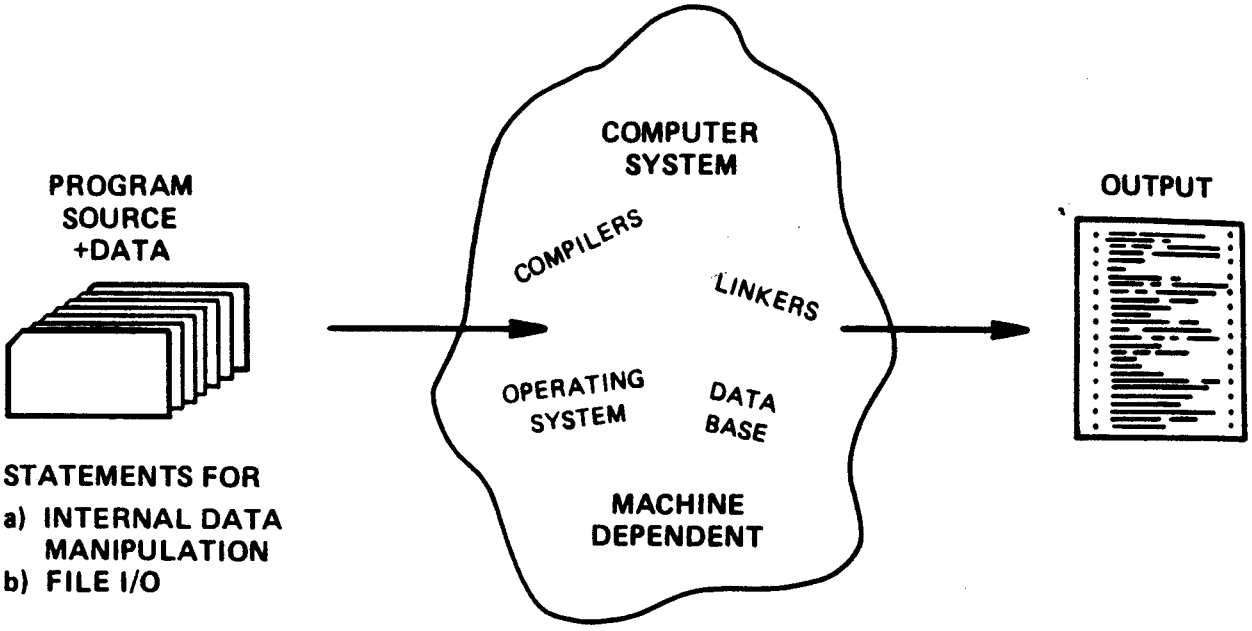


Fig. 3.2.
Conventional computing system.

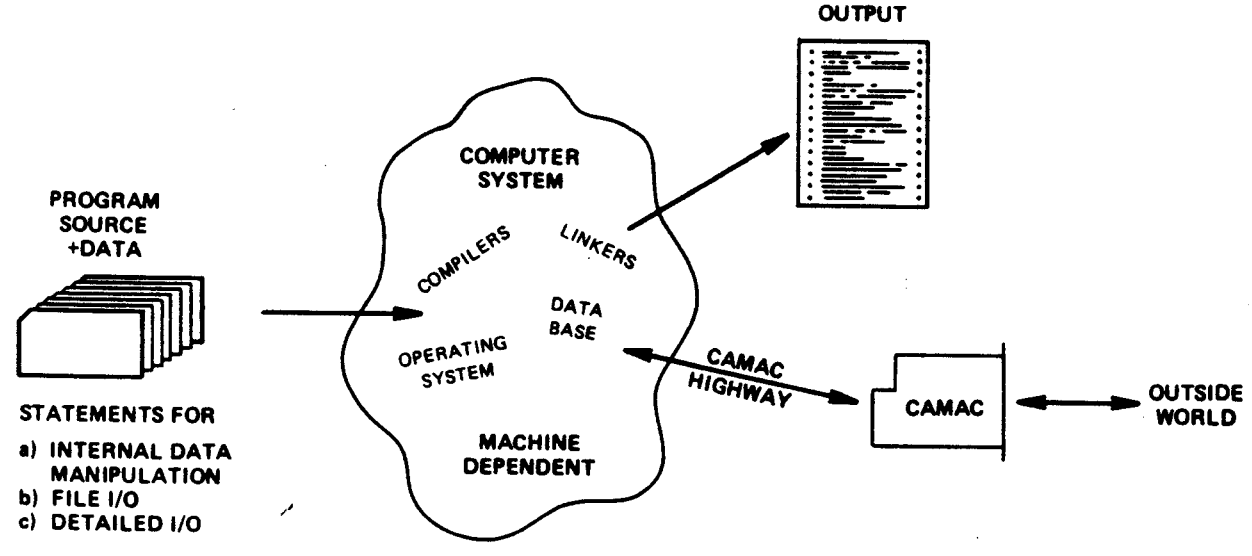


Fig. 3.3.
Real-time computing with CAMAC.

type but also of the CAMAC coupler. In the CAMAC coupler, one must also include the LAM handling system, of which more later.

3.3 Software Response to Interrupts

Interrupts are often difficult to grasp. There are two distinct kinds of interrupts, although at the hardware level they are the same. These are

- synchronous interrupts and
- asynchronous interrupts.

Synchronous interrupts are those that are expected to occur as a direct result of action on the part of the program and usually signal completion of some commanded operation. An example of this might be the completion of a disk I/O operation or a motor drive. Thus, each synchronous interrupt is only expected at certain time periods in the execution of a code.

Asynchronous interrupts are those that can occur at any time during the running of a code and usually indicate an event that resulted from action outside the computer system. Examples here are a keystroke at a computer terminal or an analogue or binary input changing.

The computer hardware will have hardware facilities that, on occurence of an interrupt, will

- ignore the interrupt until a defined point in executing the current instruction is reached--usually the end;
- save the address that contains the next instruction that will be obeyed;
- save the "status" of the processor;
- set a new processor status--this will usually set the processor priority to a level so that it cannot be interrupted further; and
- start program execution at a defined point, the interrupt response code.

This interrupt response code must then save any working registers it might use, respond to and clear the interrupt, restore working registers, and then cause the processor to restore the saved processor status and resume the interrupted program. This is shown schematically in Fig. 3.4. The interrupt response code might also need to change processor priorities, etc., depending on details of the computer and its operating system. In this way the interrupted program has no way of knowing that the interrupt occurred.

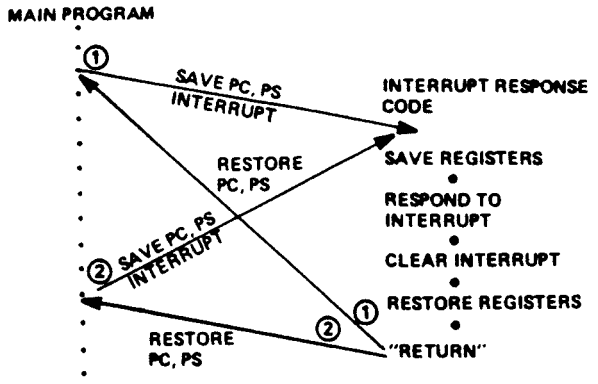


Fig. 3.4.
Flow of control on an interrupt.

Interrupt masks are registers of one or more bits that are used to enable or disable interrupts or groups of interrupts. They usually exist in the hardware attached to the computer I/O port. Processor priorities are part of the processor status that will disable or mask off all interrupts at an equal or lower priority.

Because interrupts can occur at any time, and a sequence of events is never repeatable, good code using interrupts is hard to write and often is harder to debug.

Having set the picture, let us look at how one might put CAMAC facilities in languages. But first, let us take a look at some aspects of CAMAC plug-in design.

3.4 CAMAC Plug-in Design Considerations

Keeping an eye on the requirements and limitations of software can make the difference between a poor, hard-to-use plug-in and a good, easy-to-use plug-in. The plug-in will be designed once but programmed many times; therefore, it is clearly cost-effective to take good care in the design phase. These considerations follow.

3.4.1 Data Formats. Although one can, with logical operations added, do anything with Fortran and other high-level languages, it considerably simplifies the software if twos-complement integer data is understood by the module. I once had to program a stepper-motor driver that took the number of steps to drive in a sign and magnitude integer, so that +1 was 0001 (hex) and -1 was 8001 (hex). Even in Assembler, this was not simple to generate from the twos-complement subtraction to determine how many steps to drive!

Some instruments naturally generate ASCII or BCD data and the engineer must then decide whether to convert this to binary in hardware or in software.

One guide is that the bigger and more powerful the computer system is, the more the hardware should do for it. Here it is a case of matching expense.

3.4.2 Register Packing. This is a bad aspect of some designs. Let us assume there are a few single bits of information and one or two small numbers to make available or to be written to the plug-in. Why not, for economy, pack them all up in one register so that only one CAMAC function need be used? I show this in Fig. 3.5. It looks nice and efficient but, given that you have just read that word, think of all the masking and shifting that need doing to separate out the components! Worst of all, the LINE number spans the boundary between byte 2 and 3--even more difficult for a 16-bit machine. As an exercise, write a program in Fortran or Assembler to separate out LINE number, SPEED, and COUNT into three separate variables or registers--how much faster to use four separate registers, with one dedicated to the status bits.

3.4.3 Interrupt Decoding. The two choices presented by IEEE 583 are not quite satisfactory. Clearly, testing for a LAM at each possible subaddress is neither fast nor elegant if there are more than a very few LAMs to sort out. In the register mode, a word is read in with a bit set for each LAM. The code to test-shift-count is not difficult to write, but stop to work out how long it might take on an LSI 11-23 given 16 LAMs possible, thus on average going eight times round the loop. I make that about 100 μs on an 11/23 and 240 μs on an LSI 11/2 writing in Assembler. Now, there are quite simple circuits, priority encoders, which will determine the most significant bit set; therefore, my suggestion for a LAM request register would be as shown in Fig. 3.6. The top byte now can be used as a direct indexed jump or computed GOTO to get to the appropriate service routine. Clearly, the idea of fixed LAM priority has been introduced here, and some thought will need to be given as to the exact placing of LAMs.

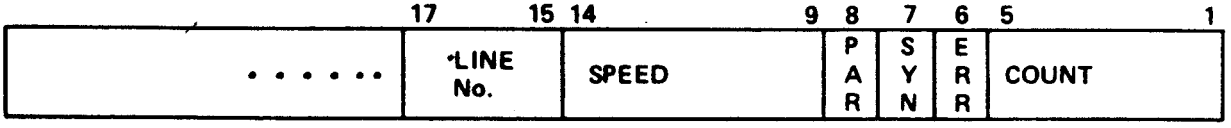


Fig. 3.5.
Register packing.

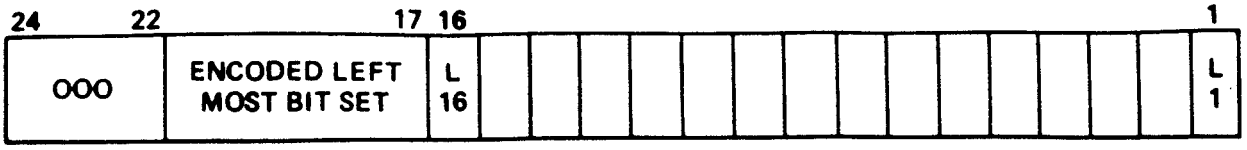


Fig. 3.6.
Suggested LAM request register.

3.4.4 Subaddress Use. A well thought-out plug-in design makes it far easier, and hence less expensive, to use. Here, not many designs follow the recommendation of IEEE 583 that features should be at separate subaddresses. One must first analyze the function of the module into separate features then do the detailed specification. An example might be a stepper-motor controller. A specification, somewhat abbreviated, might be as given below.

- Step Register A(0)--Can be written (WT1) and read (RD1). Contains twos complement of number of steps to be driven, negative equals counterclockwise. Action of writing a nonzero number starts motor.
- Speed Register A(1)--Can be written (WT1). This is a 12-bit register and the positive number written specifies speed in steps/second.
- Acceleration Register A(2)--Can be written (WT1). This 10-bit register defines number of milliseconds over which acceleration and deacceleration to/from defined speed take place.

Note that the subaddress is closely tied to the identified feature. There are 16 subaddresses, and there is no particular merit in cramming everything into just one of them!

3.4.5 Soft Links. This point relates to hardware, hardware maintenance, and software. Very often there are several quite distinct ways of using a module, and often these choices are made with wire links and solder on the board. That is fine except for two points:

- Almost always the software cannot check to see if the links are where they should be.
- When the plug-in is changed out for any reason, there is only a $(0.5)^N$ chance that the N links on the new board are correct.

These problems are overcome if the choices are not made with wire but with bits in a status register; then the software can set up the plug-in, and it

will not forget to do so. (Many hours have been wasted because of the lack of soft links!)

3.5 Software Standards for CAMAC

Software standards for CAMAC took a little while to emerge after the original CAMAC standard. Many approaches were taken in different institutions and the initial attempts at standardization were not good. This was because basically they were academic exercises rather than practical standards. The first attempt was published in the CAMAC Bulletin in November 1972 as a proposal for a CAMAC language. Fixing on the idea of a language turned out to be a mistake, and indeed the idea carried over to the first standard: IML, which stood for Intermediate Language. In fact, it was the definition of the semantics for adding CAMAC facilities to a language with an example of the syntax. The three standards I will discuss are all additions to existing languages.

In adding CAMAC to an existing language, one has three choices of overall approach:

- Add CAMAC to the language using its general syntax rules. This approach means writing a new compiler or modifying an existing one.
- Add CAMAC in a simpler manner so that a preprocessor to the compiler can take the CAMAC statements and expand them as necessary into standard language statements. This approach is usually used with assemblers.
- Add CAMAC calls using the standard subroutine or procedure call mechanism of the language. In this case, one adds procedures or subroutines to the run-time library.

3.5.1 Real-Time Basic For CAMAC. This is now in its third revision as it is kept in step with the standardization of Real-Time Basic. The intent is to produce a version of the language that requires a new compiler or interpreter. The approach is to declare "names" as PROCESS or CAMAC variables, so that in the body of the code the CAMAC registers can be easily and legibly referenced. The I/O data format also needs to be declared. Finally, CAMAC

LAMs can be defined as events to be waited on in WAIT statements. Examples of these declarations are

```
400 PROCESS INPUT WEIGHT "CAMAC (1,3,17,0) (F2) (B10)"
410 PROCESS OUTPUT PANEL "CAMAC (,,2,4)(C4)"
420 PRODIM MPX(4)
421 PROCESS INPUT MPX(1) "CAMAC (,,5,0)"
422 PROCESS INPUT MPX(2) "CAMAC (,,5,1)"
423 PROCESS INPUT MPX(3) "CAMAC (,,5,2)"
424 PROCESS OUTIN MPX(4) "CAMAC (,,7,0) (F1)"
430 PROCESS EVENT FULL "CAMAC WEIGHT GL3"
```

In the body of the code, I/O operations are then written like

```
600 IN FROM WEIGHT TO G
        or
810 OUT TO PANEL FROM H
```

Earlier versions of Real-Time Basic did not have explicit I/O operations in the code.

Also defined is a simple form of block transfer that allows transfers from/to a single CAMAC address into/from an array.

Control, or dataless operations, take place with a statement like

```
805 CONTROL MPX(4) ENB.
```

The defined function codes are

- ENB - function code 26,
- DIS - function code 24,
- CL1 - function code 9 (clear Group 1 register),
- CL2 - function code 11 (clear Group 2 register),
- CZ - activate the crate "Z" bus,
- ENCD - enable crate demands,
- DISCD - disable crate demands,
- CC - activate the crate "clear" bus,
- CLRCI - remove "crate inhibit", and
- SETCI - set "crate inhibit".

Note the lack of XEQ, and TST. Q and X can be tested with statements like

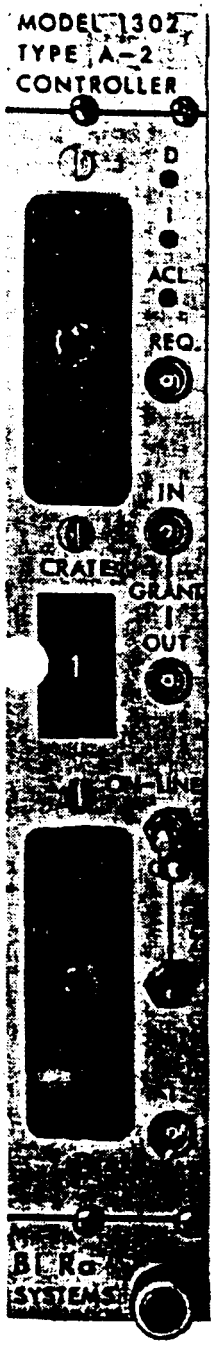
```
850 IF QCAM = 0 THEN 900.
```

Control of the LAM at the module or system level is achieved with the CONTROL statement and a new set of LAM-action keywords. These do unify the two ways

Bi Ra Systems, Inc.

2404 COMANCHE NORTHEAST • ALBUQUERQUE, NEW MEXICO 87107 • (505) 881-8887

MODEL 1302
TYPE A-2 CRATE CONTROLLER



FEATURES

- Conforms to IEEE Standards 596 and 675
- 7 Crates per Branch
- Full Parallel operation
- Two modes of arbitration, ACL, and Request/Grant

APPLICATIONS

- Crate Controller for Branch Highway
- Local Data Operation
- Allows operation of Auxiliary Controllers.

General Description

The Model 1302 is double-width crate controller that provides the interface between the CAMAC crates dataway and the Branch Highway. With respect to the Branch Highway crate controller, Type A-2 is interchangeable with Controller Type A-1 as defined in section A1 of the appendix of IEEE Std. 596. The Branch Highway cycle times will be different because of the priority arbitration logic of the Type A-2. In order to accomodate the use of Auxiliary Controllers, the Type A-2 differs from the Type A-1 by having an Auxiliary Controller Bus connector, and its associated features as described IEEE Std. 675. The Model 1302 conforms fully to both specifications.

This crate controller facilitates parallel data transfers between CAMAC modules in a crate and a CAMAC Branch Highway Driver (such as the Model 1201), at a computer. Up to seven crates can be on one branch highway. This system implement all of the functions of the CAMAC Dataway as well as multiple addressing of modules and crates. The crate can be switched off-line and in that state manual clear and initialize cycles can be initiated. Indicators are provided for the Crate address, Branch Demand, Dataway Inhibit, and ACL mode.

Functions Codes INTERNAL:

Command	BQ	Action
N28 F26 A8	ENB 0	Generates Dataway Z.
N28 F26 A9	ENB 0	Generates Dataway C.
N30 F0 A(0-7)	RD1 1	Reads the Graded-LAM connector.
N30 F16 A8	WT1 1	Writes the Station Number register.
N30 F24 A9	DIS 0	Disables the Dataway Inhibit.
N30 F24 A10	DIS 0	Disables the Branch Demand Output.
N30 F26 A9	ENB 0	Enables the Dataway Inhibit.
N30 F26 A10	ENB 0	Enables the Branch Demand Output.
N30 F27 A9	TST 0	Tests if the Dataway Inhibit is set.
N30 F27 A10	TST BD	Tests if the Branch Demand is enabled.
N30 F27 A11	TST D	Tests if Demands are present.

BX =1 for all valid addressed commands.

Command operations with N26 generate Dataway signals on all lines N1 through N23.

Command operations with N24 generate Dataway signals on N1 through N23 as per contents of SNR register. BX=1 is generated in response to all controller commands.

of referring to LAMs--the subaddress and the register methods. A statement example might be

951 CONTROL FULL ENL

The list of available keywords with their meanings are

- ENL - enable LAM, either F(26) at a subaddress or set the appropriate bit in the Group 2 mask register A(13), according to the declaration.

- DISL - disable LAM, either F(24) at a subaddress or clear the appropriate bit in the Group 2 mask register A(13), according to the declaration.

- TEST - test LAM request, either F(8) at a subaddress or test the bit position in the Group 2 register A(14). This test leaves QCAM=1 if the LAM is set, QCAM=0 if clear.

- CLRL - clear LAM, either F(10) at a subaddress or clear the bit in the group 2 register A(12).

- MENL - module enable LAM using the subaddress for overall control, see Sec. 2.3.

- MDISL - module disable LAM, overall control.

- MTEST - test LAM in the module, overall test.

- MCLRL - clear all LAMs in the module.

Finally, there are some bit string facilities as extensions to the standard and also some facilities concerning LAM graders that are programmable.

In summary, Real-Time Basic for CAMAC is an attempt to remove the I/O aspects to the declaration phase and to use simple I/O statements in the main body of code. This makes it a little inflexible for simple systems but, given the general deficiencies of BASIC, Real-Time Basic for CAMAC does provide a useful standard. Its status is that it was only recently approved in Europe by ESONE. It is awaiting approval in the US.

3.5.2 Subroutines for CAMAC. This document defines a set of subroutines to be provided as part of the run-time system of a language to give full CAMAC access. Like Real-Time Basic for CAMAC, one has to declare CAMAC registers. These are packed into a variable supplied in the call. This looks like, in Fortran,

```
CALL CDREG (ext,b,c,n,a)
```

The declaration can be unpacked with

```
CALL CGREG (ext,b,c,n,a)
```

Single CAMAC actions are defined with

```
CALL CFSA (f,ext,int,q)
```

or

```
CALL CSSA (f,ext,int,q)
```

Here int is an integer expression or integer array that contains the data to be transferred or receives the transferred data. These data are 24 bits. CSSA carries out the same operation but with the data word limited to the computer word length, usually 16 bits. The parameter q returns the Q response of the transfer.

By now it may be becoming clear that the routines all start with the letter C, and that the second letter gives the subroutine type. The remaining letters are a mnemonic of the routine function.

The defined meanings of the second letter are

"C" indicates that the subroutine performs a control function;

"D" indicates that the subroutine is a declaration of a CAMAC entity;

"F" indicates that the subroutine transfers full-length (24-bit) data words;

"G" indicates that the subroutine analyzes a named CAMAC entity into its address components;

"S" indicates that the subroutine transfers short (less than 24-bit) data words; and

"T" indicates that the subroutine tests the state of a signal or status indication.

There is a set of subroutines to set, clear, and/or test; initialize (Z), clear (C), and inhibit (I).

The problem that often arises with adding CAMAC to existing languages and compilers is that the language does not usually have facilities for asynchronous processes--for example interrupt response code. This is certainly the

case with Fortran. Subroutines for CAMAC allow for LAMs to be declared down to the subaddress or bit-position, as in Real-Time Basic for CAMAC. This is done in a similar way to CDREG with a call

CALL CDLAM (lam,b,c,n,m,inta)

The information is encoded in the variable "lam". The parameter m is a subaddress if zero or positive, a negative bit position if negative. The parameter inta is an implementation dependent array that contains other implementation-dependent information for the LAM declaration.

Once declared, the LAM can easily be enabled and disabled (CCLM), tested (CTLM), and cleared (CCLC) with simple calls.

Using the call,

CCLNK (lam,label)

the program can identify the service routine for a particular LAM to the system. From that point, if the LAM is enabled, whenever it occurs the main program will be stopped, registers will be saved, and execution will start at the label defined. This code will return control to the system when it has finished dealing with the interrupt. The method by which the interrupt-response code communicates with the main code is outside this standard and up to the systems implementer using the facilities of the operating system.

Of all the software standards, Subroutines for CAMAC has by far the most complete set of routines for block transfers. In fact, it covers all the Block Transfers defined in IEEE 683; however, some aspects of the calls are system dependent, which is an unfortunate aspect of the lack of standardization of operating-system interfaces. These will be covered in more detail later.

In summary, the Subroutines for a CAMAC document provide a good framework for adding CAMAC features to an existing language system. One should note that there is no need to implement more routines than are needed in a particular situation.

3.5.3 CATY. CATY is a language developed specifically for CAMAC programming. It developed from the need to provide CAMAC test engineers an easy facility for programming small loops to check out CAMAC plug-ins. For oscilloscope-triggering reasons, the CATY system compiles the code and then runs it rather than interpreting it line by line, resulting in much faster

execution. The syntax and semantics are taken from BASIC but the following major changes have been made:

- Data defaults to 24-bit unsigned integer.
- Only simple mathematical functions (+,-,*,/) are supported.
- Bit manipulation has been added (&,IOR,EOR).
- Shift operations are included (UP,DOWN).
- CAMAC statements have been added.
- String substitution for CAMAC addresses has been added.
- Interrupt routine facilities have been added.
- The ability to link Fortran subroutines with the CATY system and to call these routines has been added.
- Access to the Q and X response of the last operation has been provided.

In addition, some other changes have been made, as necessary, for data-acquisition systems, for example, access to time and date as well as general-stream I/O.

CAMAC address definitions look like

```
EXP1    = 1,1
DMA     = 1,1,3
SCALER  = EXP1,12
SCALER2 = 1,1,13,0
```

The address is branch, crate, station, subaddress. CAMAC operations in the Code then look like

```
WT1 SCALER,0,A  (= WT1 1,1,12,0,A)
RD1 SCALER2, B
CLM SCALER
```

where A and B are variables to be written from or read into. Note the structure that can be included in the address definitions. These definitions have two advantages: First, the code can easily be made to "talk" to a different module; second, the code can be far more readable.

One can use a "PEEP" statement in a CATY code for debugging. This statement will cause the running code to halt and print out all the single variables and their current value. With simple commands, one can then examine arrays and continue execution of the program or get back into command mode to make changes.

nterrupt routines look like

```
INTR  b,c,n
   .
   .
   .
EXIT
```

except that the ability to specify b,c,n assumes the ability of CATY to determine which station is interrupting--other systems information that CATY can determine might well be in this field, and it may then be up to the interrupt-response routine to determine which of a group of plug-ins caused the interrupt. The interrupt code between INTR and EXIT must, at least, clear the interrupt. The b,c,n may be replaced by a named CAMAC address, for example,

```
INTR SCALER
```

A feature needed for CAMAC-set-up DMA is access to variable or array addresses. This is done as follows:

```
WT1 DMA,2,@B
```

which will write the address of B to the DMA unit already defined. This puts a restriction on CATY that its internal storage of 24-bit data must be compatible with the DMA unit.

An example program is

```
10  MODULE = 1,1,3
20  LIGHTS = MODULE,1
30  SWITCHES = MODULE,0
40  RD1 SWITCHES,N
50  FOR I = 1 to N
60  WT1 LIGHTS,I
70  NEXT I
80  GO TO 40
```

Having run this program, one might get back into command mode and, to demonstrate interrupts, type

```
15  LAM = MODULE,0
90  INTR LAM
100 CLM LAM
110 PRINT "I GOT A LAM"
120 EXIT
```

Another legal program is

```
10  MODULE = 1,1,N,A
20  PRINT "ENTER MODULE STATION NUMBER",
30  INPUT N
40  FOR A = 0 TO 15
50  FOR F = 0 TO 31
60  FF MODULE,B
70  IF CAMX PRINT "LEGAL OPERATION F = ",F,"A=",A
80  NEXT F
90  NEXT A
100 GO TO 20
```

This demonstrates the ability to put variables in for CAMAC addresses and functions. To make the program more robust one might type

```
33 IF N<1 STOP
36 IF N>22 GO TO 20
```

In practice, CATY is excellent for plug-in testing and simple data-acquisition systems but starts to run out of steam as a language in which to express complex ideas or as a language to code programs that have to run extremely fast. Its primary merit is that it makes CAMAC programs extremely easy to write and test, most important for fault finding or for users to do their own data-acquisition programming. CATY has been standardized by the UK CAMAC Association at two levels: CATY1 and CATY2; these documents have been adopted by the European CAMAC Association. It is used for prototype and production module testing by all the UK CAMAC manufacturers and, indeed, some US manufacturers.

4. THE CAMAC PARALLEL BRANCH

4.1 Introduction

IEEE 596 defines a parallel branch used to connect between one and seven crates to a computer through a branch driver. This is shown schematically in Fig. 4.1. The basic standard defines the cable, connectors, signal standards, and the use of the signals so that different crate controllers and branch drivers can all work together, the very essence of a standard. As an appendix,

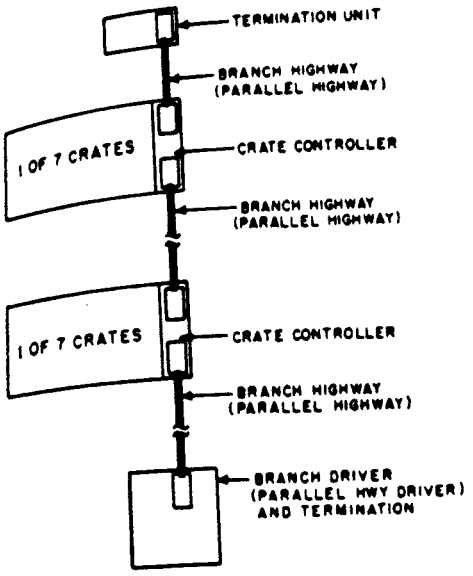


Fig. 4.1.
The CAMAC parallel branch.

the crate controller Type A-1 is defined and, indeed, until recently it was the only controller available for this branch. Now the definition has been slightly extended to incorporate the ACB and request-grant mechanism, thus defining the Type A-2 controller.

4.2 The Signal Lines

Table 4.1 lists the signal lines on the branch. Each signal has two wires, a signal path, and a return path; thus the cable is made up of 66 twisted pairs.

Table 4.1

SIGNAL LINES AT PARALLEL BRANCH PORTS

Title		Designation	Generated by	Signal Lines	Use
Command	Crate address	BCR1 - BCR7	Branch driver	7	Each line addresses one crate in the branch
	Station number	BN1, 2, 4, 8, 16	Branch driver	5	Binary coded station number
	Subaddress	BA1, 2, 4, 8	Branch driver	4	As on Dataway A lines
	Function	BF1, 2, 4, 8, 16	Branch driver	5	As on Dataway F lines
Data	Read/Write	BRW1 - BRW24	Branch driver W or Crate controller R, GL	24	For Read data, Write data, and Graded-L
Status	Response	BQ	Crate controller	1	As on Dataway Q line
	Command accepted	BX	Crate controller	1	As on Dataway X line
Timing	Timing A	BTA	Branch driver	1	Indicates presence of command, etc.
	Timing B	BTB1 - BTB7	Crate controller	7	Each line indicates presence of data, etc. from one crate controller
Demand handling	Branch demand	BD	Crate controller	1	Indicates presence of demand
	Graded-L request	BG	Branch driver	1	Requests Graded-L operation
Common control	Initialize	BZ	Branch driver	1	As on Dataway Z line
Reserved		BV6 and BV7		2	For future requirements
Free		BV1 - BV5		5	For nonstandard user requirements

NOTE: An individual return line is provided for each signal line. Two lines are provided for a connection to the shield, if any, of the parallel branch cable.

4.2.1 Crate Address BCR1 - BCR7. These seven signals are assigned one to each crate on the branch. Each crate controller must have a suitable switch to select its crate number and this setting must be visible on the front panel. Thus one can choose to address any or all crates at once, and we will see that facilities exist so that one can address any selection of plug-ins in a system at once.

4.2.2 Station Number BN1,2,4,8,16. These five lines address the station number of interest, and are decoded in the crate controller so that the particular N line(s) can be asserted. The 32 possible codes are assigned as shown in Table 4.2. Apart from normal station numbers, N(0) is reserved, N(24) addresses stations selected by a previous operation, N(26) addresses all normal stations, and N(28) addresses the crate controller. All these operations generate a dataway cycle. N(30) also addresses the crate controller but does not generate a cycle; thus, no other module in the crate can be aware that anything has happened.

4.2.3 Subaddress BA1,2,4,8 and Function BF1,2,4,8,16. These correspond directly to the dataway lines.

Table 4.2

STATION NUMBER CODES USED IN CRATE CONTROLLERS

N Code	Use	B, S1, and S2	Remarks
N(0)	Reserved		
N(1) - (23)	Address the corresponding normal station	yes	
N(24)	Address preselected normal stations	yes	Normal stations occupied by the controller need not be addressed
N(26)	Address all normal stations	yes	
N(28)	Address crate controller only	yes	
N(30)	Address crate controller only	no	No dataway operation
N(25, 27, 29, 31)	Reserved		

4.2.4 Read/Write BRW1 - BRW24. These are bidirectional lines, unlike the dataway lines, used to save cable and connector costs. In read operations they correspond to the dataway read lines, in write operations to the dataway write lines. Finally, they carry the Graded-L (or Graded LAM) information during a graded LAM read operation, of which more later.

4.2.5 Response BQ and Command Accepted BX. These are the same as the dataway signals.

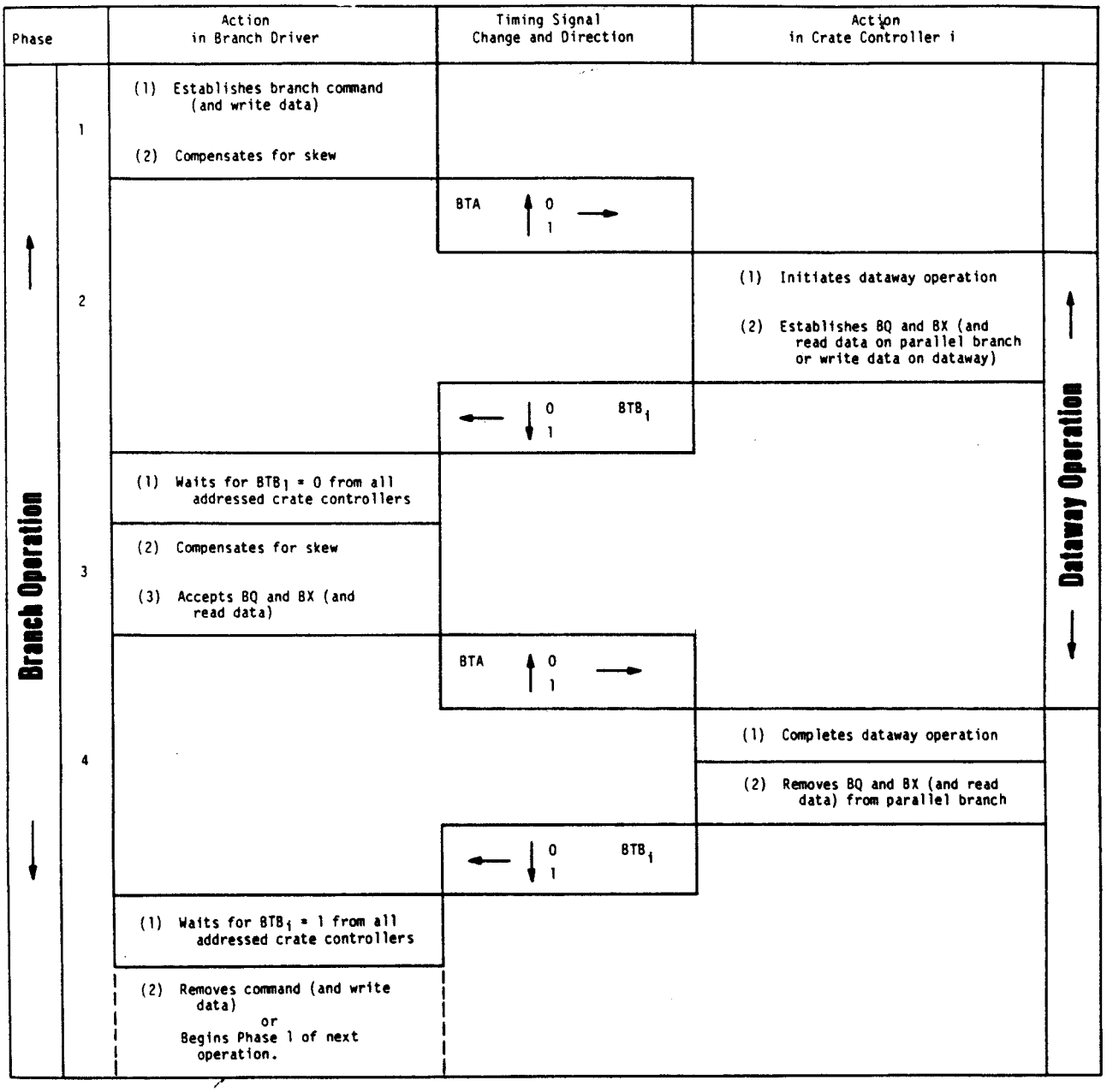
4.2.6 Timing A,BTA and B,BTB1 - BTB7. The branch timing signals control the timing of all branch operations. The branch driver initiates an operation by a signal on the BTA line and all addressed crates respond with signals on their individual BTB lines. Each crate that is on-line must generate a BTB=1 on its BTB line so that the branch driver can know which crates are on-line.

A branch operation is divided into four phases as shown in Table 4.3. In the first phase, the branch driver establishes all the lines appropriate and then waits to allow for skew. In the second phase, it sets BTA=1. All controllers then respond to the other lines on the branch; if that particular controller is addressed, it initiates the operation--usually a dataway cycle. It then, at S1, establishes BX, BQ, and BRW if it was a read operation and then sets its BTB from 1 to 0. The end of this phase is when the branch driver detects that all addressed controllers have set their BTB=0. Phase three consists of the branch driver waiting to allow for skew and then accepting the data set by the controller (BQ, BX, and BRW if a read). The final phase, four, closes down the operation with the branch driver setting BTA=0. When the controller sees this, it completes the dataway operation, removes its data from the branch and sets its BTB=1. The branch driver waits for all addressed BTBs=1 and then removes command and write data to close the cycle. Alternatively, at this point, a new cycle can be started.

4.2.7 Branch Demand BD. This single line is asserted by any crate that has a LAM waiting service. This signal can be as a result of any logical function of the L lines. There is a restriction on timing in that the propagation delay through the controller and any auxiliary LAM-handling modules must not exceed 400 ns. Also, the BD line-driver transition time must be in the range 100 ±50 ns to avoid noise effects as the BD line can be asserted at any time.

Table 4.3

SEQUENCE OF COMMAND MODE OPERATION



NOTES: (1) Throughout the operation BG = 0.
(2) Actions shown in brackets apply when required by the command.

4.2.8 Graded-L Request BG. The branch driver initiates Graded-L operations by generating BG along with BCR for each on-line crate. Each addressed crate controller generates a 24-bit Graded-L word on the BRW lines, and the branch driver then reads the OR of these words. The dataway L signals of each crate therefore need to be selected and assigned to the appropriate bit of the Graded-L word. The A-1 controller extends these facilities as we will see.

Table 4.4 indicates in a way similar to Table 4.3, the logical sequence of operations in a Graded-L operation.

4.2.9 Branch Initialize BZ. This signal has priority over all others on the branch, and crate controllers must generate an initialize cycle in the crate in response to this signal. No other signals are generated with it and, to avoid noise effects, the branch driver must maintain it for at least 10 μs and must not generate any branch operation for 5 μs after removing BZ. The crate controller assumes BZ is true after integrating it for 3 ±1 μs.

4.2.10 Dataway Clear and Inhibit. Crate controllers must have some facilities for setting and clearing these lines.

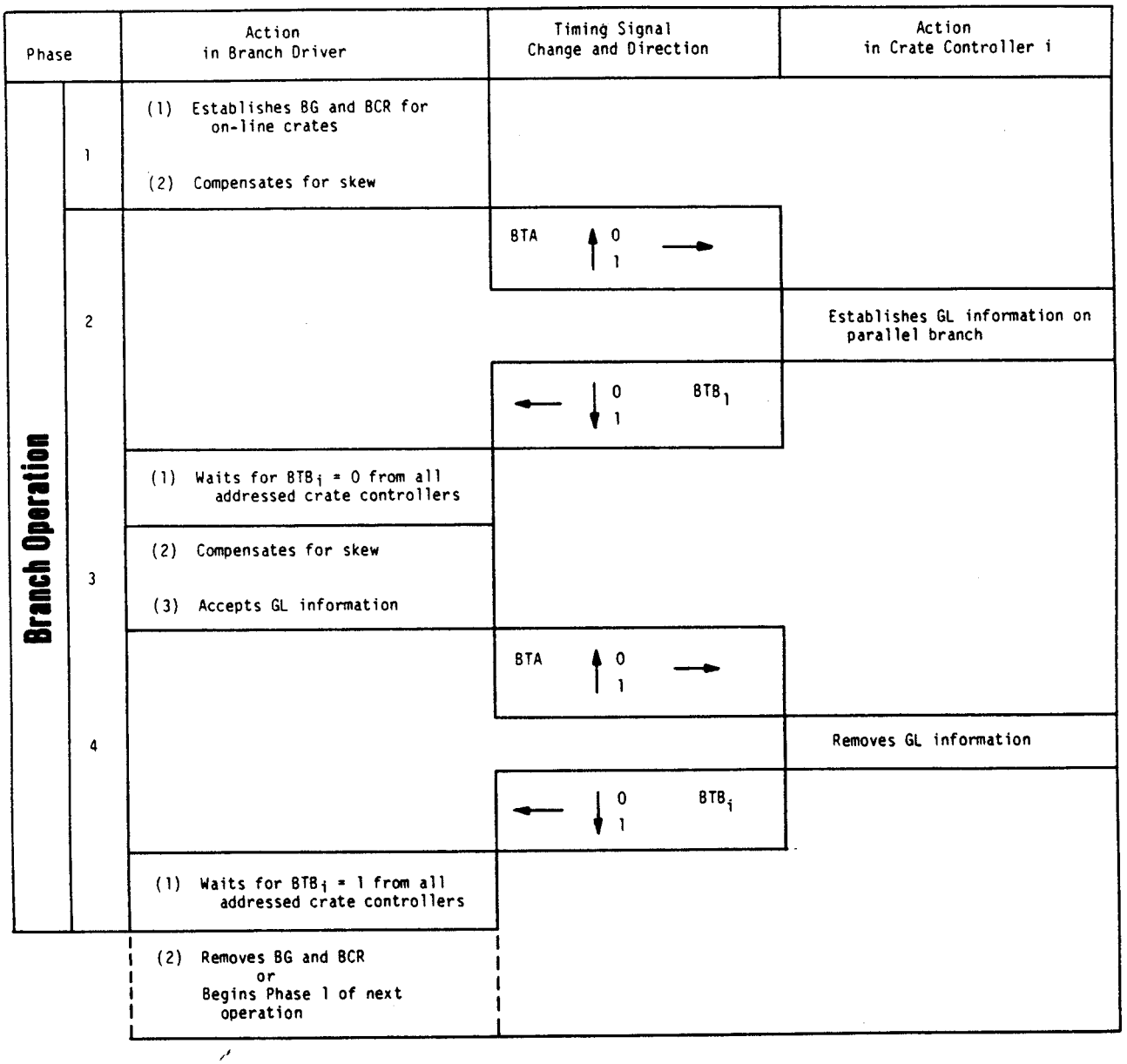
4.2.11 Reserved and Free Lines BV1 - BV7. BV6 and BV7 are reserved for future allocation, whereas BV1 - BV5 are available for use in systems. However, reliance on their use does introduce incompatibility into plug-ins controllers and branch drivers; thus, their use should be embarked on only with care.

4.2.12 Timing Summary. Figures 4.2 and 4.3 summarize the timing and relationships for branch read and branch write operations. The diagrams attempt to show the transmission delays along the branch. It will be noted that the branch operation is a handshake operation so that the length of the branch or delays in the crate controller will not jeopardize reliable operation of the branch. This is not the situation on the CAMAC crate, where a module either responds or the operation fails.

Looking at a branch read operation, Fig. 4.2, it starts with the branch driver asserting, as required by the particular operation, the BCR, BN, BA, and BF lines. After a deskew time, the asserting of BTA indicates to the addressed controllers that they must act on the state of these lines. The BTA signals

Table 4.4

SEQUENCE OF GRADED-L OPERATION



NOTES: (1) Throughout the operation BG = 1.
(2) Command signals BN, BA, and BF are ignored.

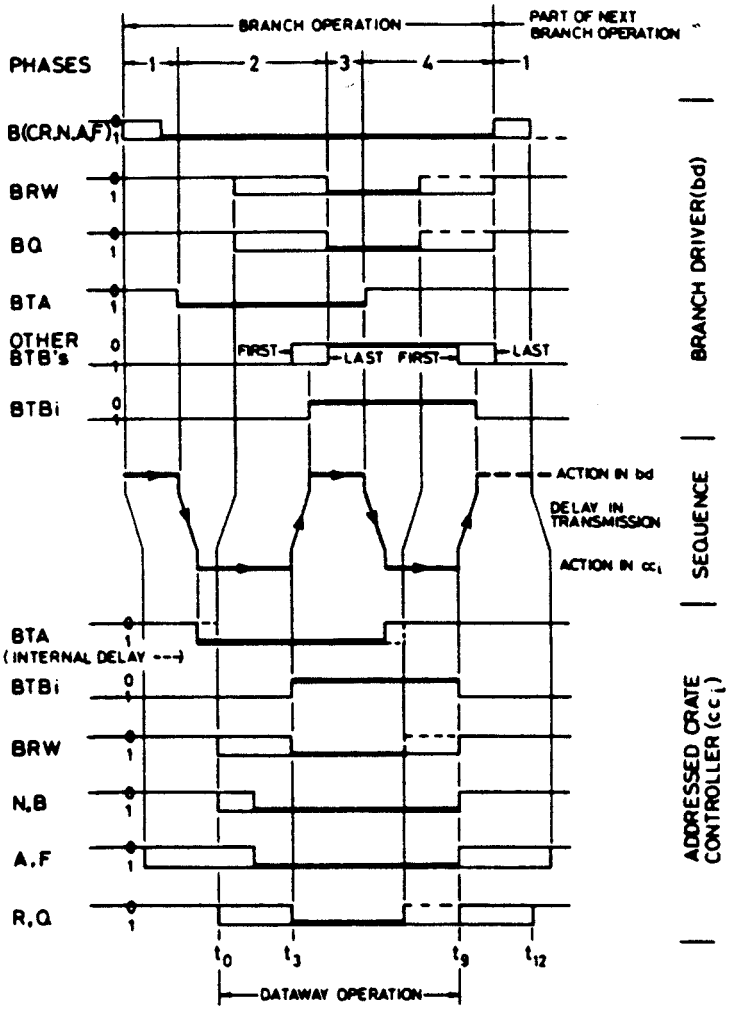


Fig. 4.2.
Timing of a branch-read operation.

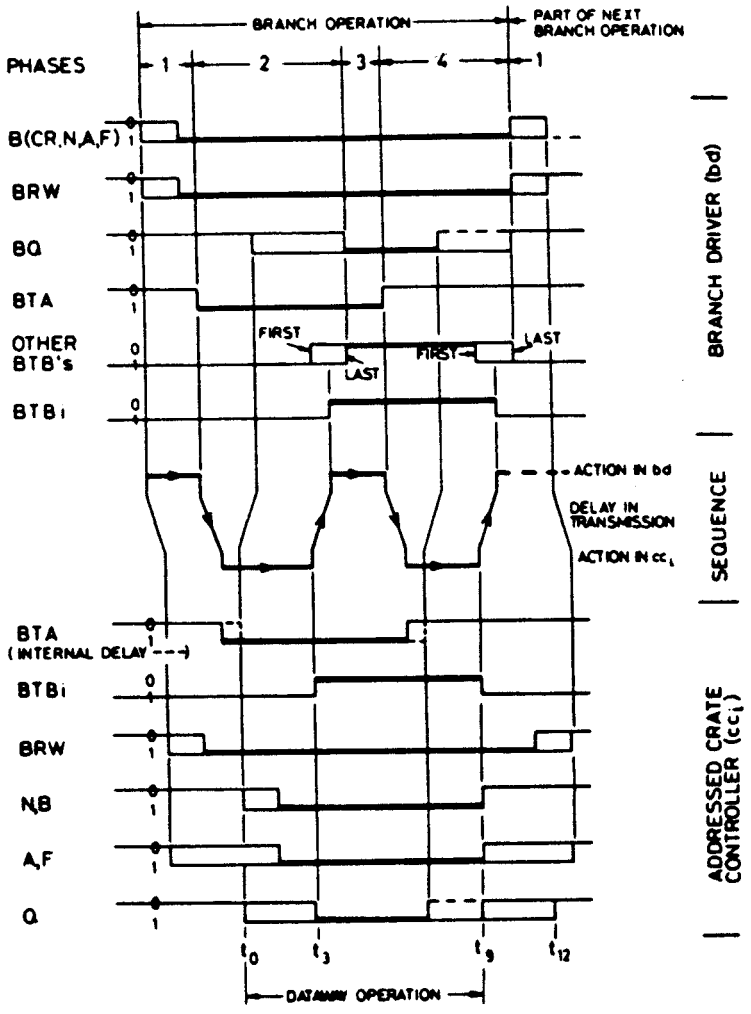


Fig. 4.3.
Timing of branch-write operation.

will be seen by the controllers after a transmission delay. After an internal delay to allow for decoding, the controller initiates a CAMAC cycle, and at the start of S1 the controller gates the R lines to the BRW lines and Q X to BQ and BX. At the same time, the controllers respond with their own BTB. When the branch driver has all the expected BTB responses, it clocks the BRW, Q and X lines and then sets BTA=0. The controller responds to this by completing the dataway cycles and then setting BTB back to a 1. When the driver sees all addressed crates' BTBs=1, the cycle is over. A typical parallel branch operation, including the CAMAC cycle, is 1.6 μs.

Figure 4.3 shows the branch cycle for a write operation; this differs from Fig. 4.2 in that only the BRW lines are set by the branch driver in Phase 1 of the operation.

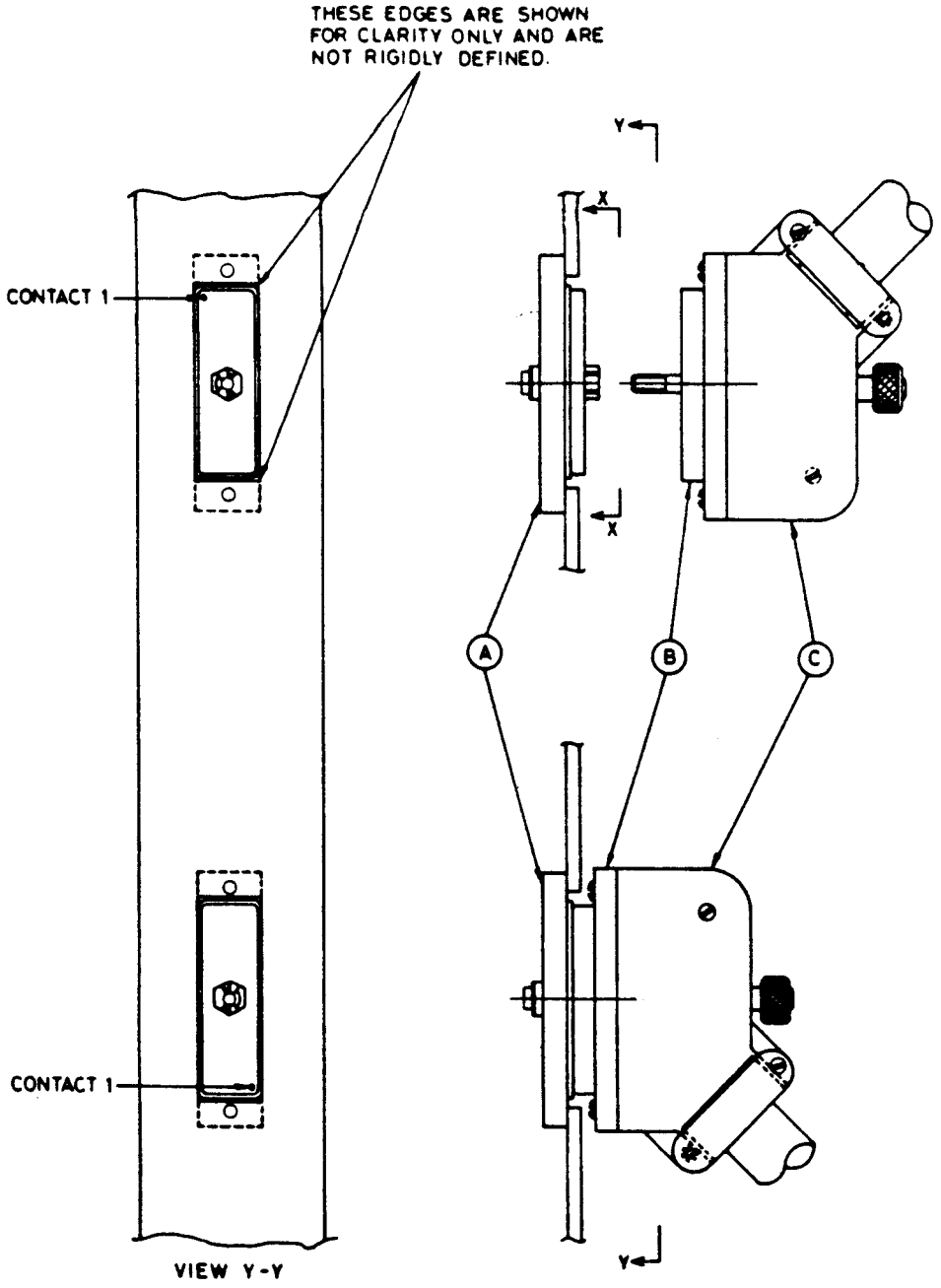
4.3 The Connectors

The connector chosen for the parallel branch was a 132-way Hughes connector shown in Fig. 4.4, together with the way sockets have to be mounted on the front panel of a crate controller. Being such high-density connectors, they and the cable are expensive and do at times give problems.

4.4 Electrical Signal Standards

All units attached to the parallel branch must conform to the signal standards. Because it is a bus structure, the outputs driving the branch must be wire OR'ed and this is usually an open-collector line-driver. If no unit is trying to set the line to a "1" that is, 0 V, the terminator will maintain it at ∿4 V. Table 4.5 defines the I/O circuits attached to the highway as well as the terminator. The BD, BTA, and BTB signals must be generated from sources that define the transition time, 100 ±50 ns. It is recommended that the branch be terminated at both ends. Finally, an off-line crate controller must not generate any signals on the branch; if it is not powered, it also should not hold any lines down to the "1" state.

The parallel branch is not driven in a balanced way; therefore, it does have a length limitation (∿40 to 100 m), but it is heavily dependent on the cable used and on the noise environment. This length restriction can be eased by using a converter of balanced drivers and receivers at each end of the long lengths of branch cable.



Note: 1. For information only (See Sec. 6).

A Fixed Connector--Socket body without jackscrew.
B Free Connector--Pin body with jackscrew.
C Hood--Recommended assembly with cable entry adjacent to Contact 1.

Fig. 4.4.
Parallel branch ports: arrangement of connectors on crate controllers.

Table 4.5

SIGNAL STANDARDS AT PARALLEL BRANCH PORTS

Condition at Parallel Branch Ports	Logic State	Absolute Limits	Recommended Values
Inputs			
(1) Voltage range accepted by unit	0	+2.4 V to +5.5 V	
	1	0 V to +1.2 V[a]	
(2) Maximum current supplied by unit	0	±0.3 mA	±0.3 mA[b,d]
	1	+1.6 mA (±0.3 mA for crate controller Type A-1)	
Outputs			
(3) Voltage range generated by unit	1	0 V to +5 V	0 V to +0.3 V
(4) Minimum current sinking capability[c]	1	127 mA	133 mA
Termination			
(5) Open-circuit voltage	0	+4.5 V maximum	+4.1 V preferred[d]
(6) Short-circuit current	1	50 mA maximum	
(7) Terminating impedance			100 Ω preferred[d]
Parallel Branch			
(8) Characteristic impedance		70 Ω minimum	100 Ω maximum[d]

[a]Higher than TTL voltage levels provide an increased noise margin taking into account cable losses and reflections due to mismatches.

[b]Low input currents result in smaller reflections. Receivers with high input impedance may feed current into the line or draw current from the line.

[c]The current sinking capability is given by

$$\frac{V_o - V_{out\ low}}{Z/2} + 8.I_{in\ low} = \begin{cases} 127\ \text{mA absolute minimum} \\ 133\ \text{mA recommended minimum} \end{cases}$$

where V_o = 4.5 V maximum open-circuit voltage

$V_{out\ low}$ = { 0.5 V absolute / 0.3 V recommended } maximum low state output voltage

Z = 70 Ω minimum characteristic impedance

$I_{in\ low}$ = 1.6 mA maximum low state input current.

[d]Recommended values refer to a set of design values for a preferred terminating circuit.

4.5 Crate Controller Type A-1

The A-1 crate controller is defined to work on the parallel branch but is not the only controller that can be used on a parallel branch. It is clearly required to operate within the standards IEEE 583 and IEEE 596. Figure 4.5 shows schematically the A-1 controller.

The front panel must have the following features and no others that might affect interchangeability.

(1) Two Hughes parallel branch connectors.

(2) An indication of the crate number. There must be no easy access to the switch, on or through the front panel.

(3) A means to set the controller off-line.

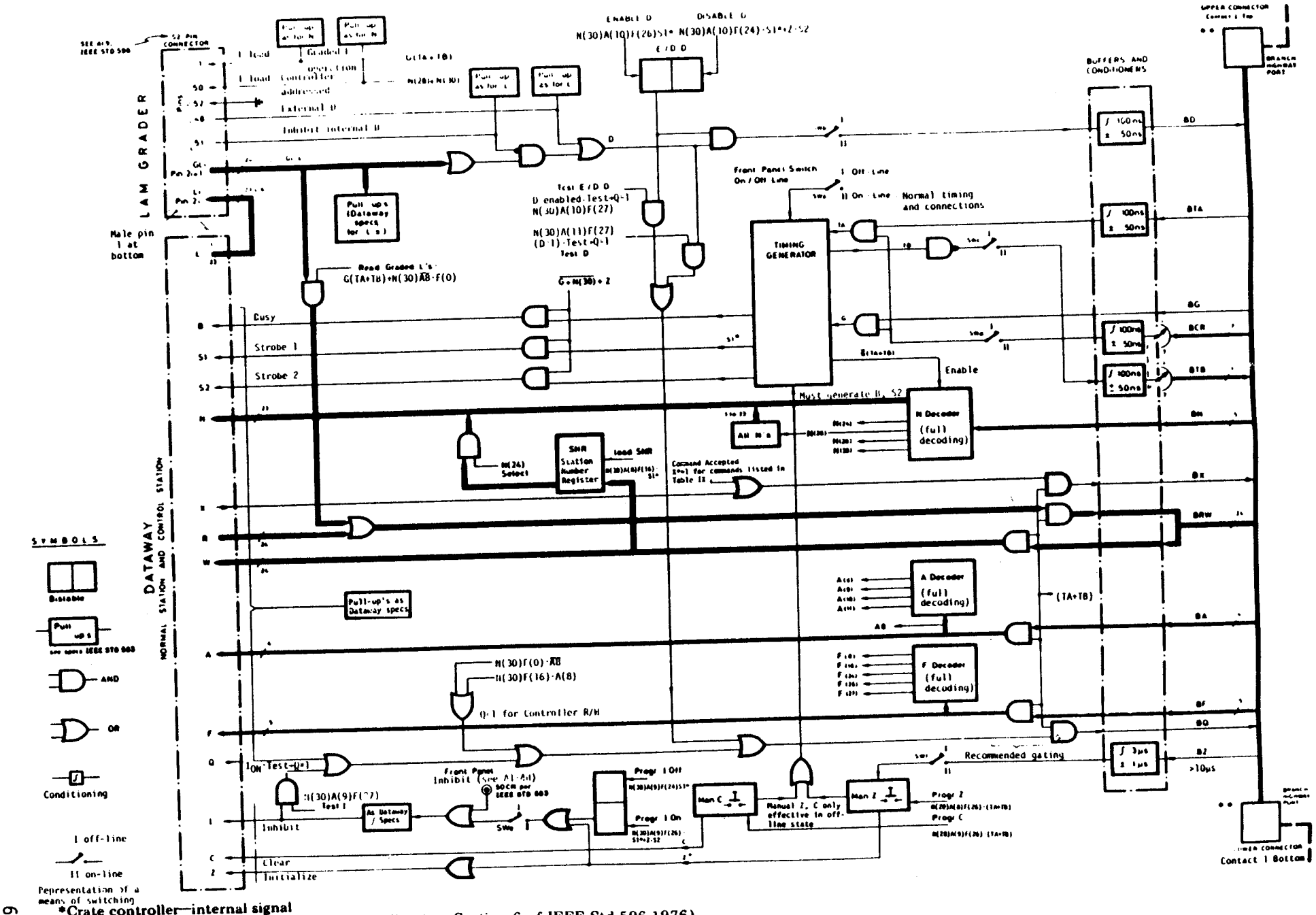
(4) A coaxial connector connected to the inhibit line.

(5) A way to manually initiate a dataway initialize or a dataway clear sequence. This is only effective in the off-line state, and this must be indicated.

4.5.1 Data Signals. When the crate controller is on-line and addressed, and the station number is not N(30), it must transmit the read lines to the BRW lines for a read command. During write operations and for stations other than N(30), it must transmit the BRW lines to the W lines of the crate. It is recommended that the crate controller gate the BRW lines to the R and W lines only when it is addressed, on-line, and for other than Station 30. It can be more selective in its gating if desired. For the A-2 controller, this gating is mandatory.

4.5.2 Command Signals. The A-1 controller should condition the BN, BA, and BF lines by integration or staticizing at time BTA 0→1 to protect the dataway command lines from the effects of noise on the branch. The controller must decode the BN lines and must assert the appropriate N line(s). When Station 26 is decoded, all N lines, N1 - N23, must be asserted. When Station 24 is decoded, the controller must assert the N lines 1 - 23 according to the contents of a 23-bit station number register (SNR). This is loaded by the command N(30), A(8), F(16), and BRW1 corresponds to N1. The SNR is not reset by the initalize sequence, Z.

Table 4.6 lists commands that must be implemented by the A-1 controller.



*Crate controller—internal signal

**132 contact connectors — fixed socket moulding (see Section 6 of IEEE Std 596-1976).

Notes:

(1) For contact allocation see IEEE Std 596-1976

(2) All 132 lines must be wired between connectors.

Fig. 4.5. CAMAC crate controller Type A-1. (Double-width unit).

Table 4.6

COMMANDS IMPLEMENTED BY CAMAC CRATE CONTROLLER TYPE A-1

Action	Command N	Command A	Command F	Response
Generate Dataway Z	28	8	26	BQ = 0
Generate Dataway C	28	9	26	BQ = 0
Read GL	30	0 to 7	0	BQ = 1
Load SNR	30	8	16	BQ = 1
Remove Dataway I	30	9	24	BQ = 0
Set Dataway I	30	9	26	BQ = 0
Test Dataway I	30	9	27	BQ = 1 if I = 1
Disable BD output	30	10	24	BQ = 0
Enable BD output	30	10	26	BQ = 0
Test BD output enabled	30	10	27	BQ = 1 if BD enabled
Test demands present	30	11	27	BQ = 1 if demands present

4.5.3 Demand Handling. The Type A-1 controller provides only minimal facilities for LAM handling. On the back of the controller is a 52-way double-density Cannon socket to which is wired directly the 23 L lines from stations other than those occupied by the controller. Any unit plugged into the socket can then sort, grade, or mask the L-lines at will and can present up to 24 graded LAMs back to this socket. The controller will gate these lines into the BRW lines when a graded-LAM operation occurs. It will also OR these lines to form an "internal-demand" signal that can be inhibited by a line from the rear 52-way connector, "inhibit internal D". The "external D" line at this connector is provided in case the LAM grader is to generate the "OR" to its own needs. In that case, it inhibits the internal demand by setting "inhibit internal D"=0. Finally, the controller makes available two signals, N(28) + N(30) and G(TA+TB). The first is so that the LAM grader can be addressed at N(28) or N(30), thus avoiding the need to know the station into which it is plugged. The LAM grader can differentiate between the two because with N(28) is generated B, S1, and S2 whereas with N(30) they are not. The second signal,

G(TA+TB) (TA is the BTA signal gated with "this controller addressed", likewise for TB), is to indicate to the LAM grader that the graded LAM information is needed. It then has 350 ns in which to present it to the GL lines.

Thus the system builder has full freedom to OR station LAMs from one or many crates onto a single GL line, or simply wire Ls to any selected GL. Note that there are only 24 separate GLs on a branch, although one can read them out crate by crate. The simplest LAM grader is simply a plug that wires the Ls of interest to a selected GL. The most complicated LAM graders provide a mask register for the LAMs and then the ability for software to patch these LAMs to any selected GL line. The idea of LAM grading arose out of high-energy physics, where one had but a small number of possible LAMs in a large system and they did not particularly relate to the module. In other systems, the LAM usually reflects the needs of the station that generated them, and so they should not be graded at all.

4.5.4 The EC 370. This is a LAMs grader designed by the Daresbury Laboratory. This grader provides a 23-bit LAM mask register, a 23-bit LAM request register, and a 5-bit encoded left-most LAM request register. The LAM mask register allows system software to mask off individual LAMs without knowing the particular design of the modules concerned. In response to the graded LAM operation, the EC 370 asserted the GL bit, corresponding to the crate in which it was if it had an unmasked LAM. Using this LAM grader, the branch graded-LAM operation would result in a 7-bit pattern of interrupting crates. Choosing one of these to read out the enclosed left-most LAM request register will then give, in two operations overall, the interrupting station, encoded. I use this module as a familiar example. This LAM grader served as the model for the serial LAM grader of which more later.

4.5.5 Summary. The parallel highway has the merit of speed and inexpensive controllers, but the disadvantage of expensive cable. It does provide a high-speed system for attaching up to seven CAMAC crates to a computer. If more crates are needed, then further branches must be implemented. Apart from the cable cost, there are limitations on the length of the branch and, because it is a handshake system, the longer the branch, the slower will be the cycles on that branch. Although LAM graders have not been even recommended for the parallel branch, in general the LAM graders implemented have emphasized patching of LAMs to GLs rather than rapidly finding the interrupting station.

5. THE CAMAC SERIAL HIGHWAY

5.1 Introduction

The CAMAC serial highway is a highway for connecting between 1 and 62 crates to a single controller. It consists of a unidirectional loop that begins and ends at the serial highway driver and passes through each serial crate controller. There is no requirement that equipment connected to the serial highway be CAMAC, so long as it conforms to the serial-highway protocol. The highway can be either bit or byte serial, and in addition to data lines, there is a clock line making, in all, two or nine twisted pairs in the highway.

The serial highway is defined primarily in terms of message formats, protocols, and signal standards at the defined I/O ports, the D-ports. These signal standards can be changed by external units to suit particular transmission requirements. The external units are termed undefined port adaptors or U-port adaptors.

Finally, we will look at the recommended serial crate controller Type L-2.

5.2 Serial Highway Messages

The serial highway message consists of bytes (8 bits) of information traveling around the serial loop. The message starts with the first byte with Bit 7 a zero and the last byte of the message has Bit 7 a one. Any space or other bytes pumped round the serial highway that do not form part of a message must have Bit 7 set to one. Thus, any device on the serial highway can break down the traffic into messages without knowing the format for the messages. Clearly, it has to know the format for messages to which it must respond, so that controllers on the serial highway know which messages are addressed to them. The first byte of any message is defined to contain the crate number. Every device on the highway has a unique crate number assigned, but the delimiter bit, Bit 7, means that it need not know the format of messages not addressed to it. Figure 5.1 shows the general format for a serial highway message. The header byte contains the address and is the first byte with Bit 7=0. In all bytes, Bit 8 is the parity bit and is set to maintain odd parity in the byte. This leaves 6 bits per byte for message or text. The last byte of the message has Bit 7=1; the parity bit is maintained, and the

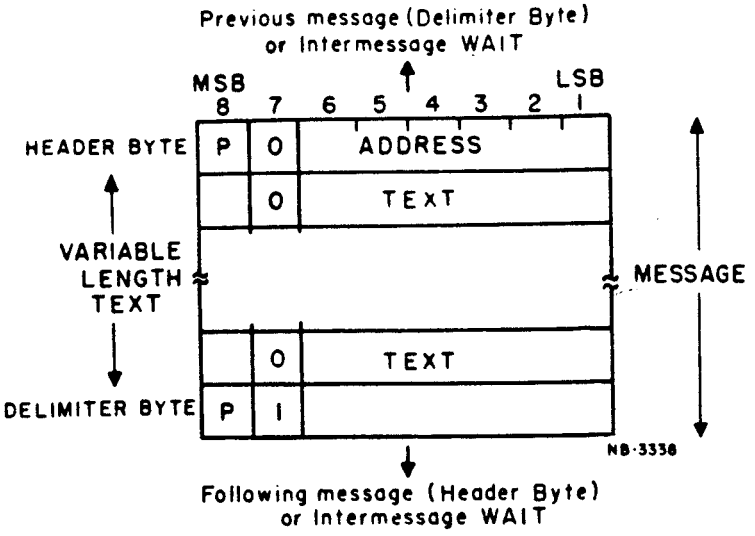


Fig. 5.1.
Basic message format.

remainder of the byte is set for longitudinal even parity. In this way there is excellent error detection on the transmitted and received messages in the system.

How are bytes actually transmitted on the wires of the serial highway? This transmission is different, depending on whether the mode is bit or byte serial. In byte-serial transmission, 8 bits are clocked by the clock, the ninth signal, so that byte synchronization is easily maintained. In bit serial mode, one signal is clock, the other is data. Thus, to be able to extract the bytes out of the stream of bits, some form of byte synchronization technique needs to be implemented. Two techniques are commonly used:

1. Transmit a known synchronization byte that will uniquely determine the byte frame. For example 11110000 will do this, 11001100 will not--there being two possible byte frames in such a stream. Once synchronization is achieved, it can be maintained by counting. Clearly the synchronization byte must be prohibited as the first byte of a message.
2. Provide synchronization on a per-byte basis by transmitting extra bits with each byte. This technique is used in the serial highway and is similar to that used in asynchronous terminals such as VT100s. (The difference is only that the terminal generates its own clock, knowing the speed. In the serial highway, the clock is transmitted so there is only a single speed adjustment in a system).

Figure 5.2 shows the bit stream of a bit serial system. Two extra bits are added to each byte: a start bit, which is zero, and a stop bit, which is a one. Further, if there is a gap between bytes, the signal line must remain at a one state. Thus using the synchronization byte 11100000, the 1-to-0 transition indicates the start of a byte on the line.

5.3 Electrical Characteristics of the Highway

The signal and clock standards are defined to be exactly those of the Electronic Industries Association Standard RS 422 and CCITT Standard X27. This standard defines a balanced driver and receiver connected by a twisted-pair cable of 100-Ω nominal impedance. Integrated circuits are available for transmitting and receiving these signals, much simplifying design and construction. The balanced transmitter is shown in Fig. 5.3. It will be seen that the two outputs are driven in opposition to each other in such a way that the average voltage on the lines is approximately zero regardless of signal transmitted, that is, balanced.

Table 5.1 shows the basic specification of an RS 422 balanced transmitter. In summary, it specifies that the average of the two outputs should be at least 2 V or 50% of the maximum single line voltage, whichever is greater. This average should not vary by more than 0.4 V as the output changes state, and the voltage between the lines should not vary in magnitude (it will reverse sign) by more than 0.7 V also. The remainder of the table is self-explanatory.

Table 5.2 indicates the basic specification for a balanced receiver. The interesting point, to me, is that the maximum common mode voltage, line to ground, is defined as 10 V, which is not too much for long lines. Clearly, some optical or other isolation would greatly improve reliability.

5.4 Timing

Within the standard, as we have already seen, there is a system clock that is transmitted to all devices on the highway on a separate clock-twisted pair. If the highway is converted to an undefined communication channel for whatever reason, the clock and data can be encoded onto the same channel and separated again at the receiving end to re-establish the D-port. Five MHz is the maximum instantaneous clock rate that is allowed in the serial highway system, and any

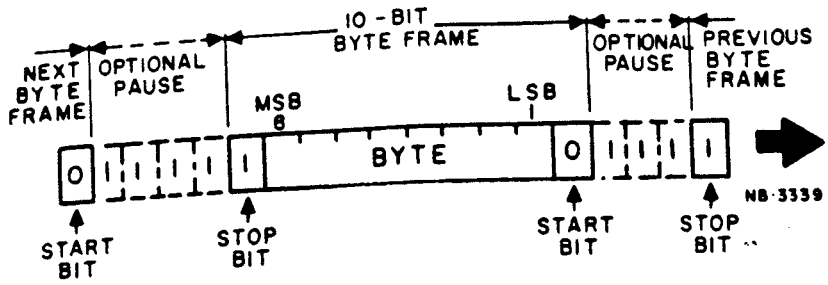
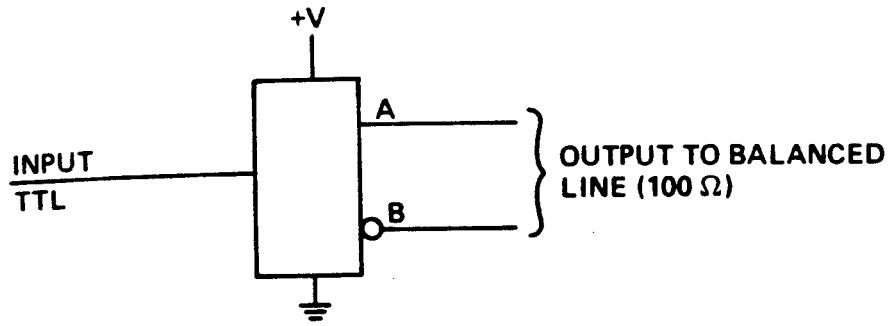


Fig. 5.2.
Bit-serial byte-frame.



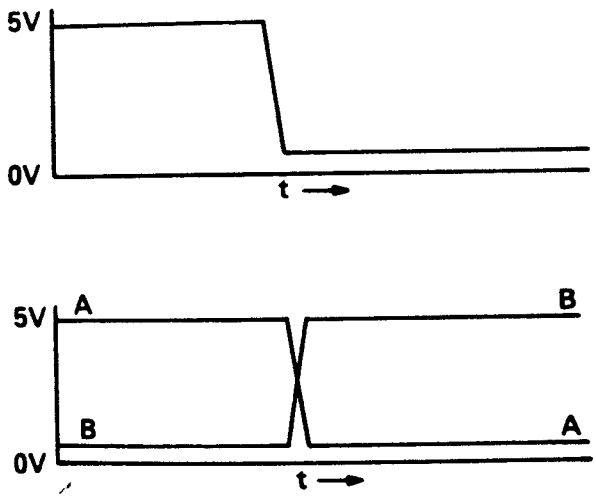


Fig. 5.3.
An RS422 balanced driver and its operation.

Table 5.1

SUMMARY OF CHARACTERISTICS OF BALANCED TRANSMITTER

Output resistance line-to-line	100 Ω or less
Magnitude of open circuit voltage, line-to-line [V_o]	6 V or less
Magnitude of open circuit voltage, line-to-ground [V_{oa}, V_{ob}]	6 V or less
Magnitude of output voltage, terminated in 100 Ω, line-to-line [V_t]	Not less than 2 V or 50% V_o, whichever is greater
Magnitude of offset voltage [V_{os}] (Note 3)	3 V or less
Magnitude of difference in [V_t] for two logic states	Less than 0.4 V
Magnitude of difference in [V_{os}] for two logic states	Less than 0.4 V
Magnitude of short-circuit current, line-to-ground	150 mA or less

NOTES: (1) The maximum values of the 10 to 90% rise and fall times of signals generated by the transmitter when applied to a 100 Ω resistive load are as follows (where T_{min} is the bit or byte period, as defined in Section 8.3):
- for clock signals, less than 20 ns or 0.05 T_{min}, whichever is the greater;
- for data signals, less than 0.1 T_{min}.

(2) Where a magnitude is defined, the parameter may be positive or negative.

(3) The offset voltage is measured between the center point of a 100 Ω test load consisting of two resistors, 50 Ω ± 1% each, and the generator circuit ground.

Table 5.2

SUMMARY OF CHARACTERISTICS OF BALANCED RECEIVER

Input resistance line-to-line [R_t] (Note 2)	100 Ω ± 10%
Input impedance line-to-ground, with 100 Ω termination removed	≥4000 Ω
Magnitude of input voltage, line-to-line, at which receiver must operate correctly [V_i]	≥0.2 V ≤6.0 V
Magnitude of common mode voltage at which receiver must operate correctly [V_{cm}] (Note 3)	≤7.0 V
Maximum magnitude of input voltage, line-to-ground	≤10.0 V
Magnitude of input voltage, line-to-line, without damaging the receiver (the termination of 100 Ω may be removed for this test).	≤12.0 V

NOTES: (1) Where a magnitude is defined, the parameter may be positive or negative.
(2) In EIA RS-422 the use of a cable termination at the balanced receiver is optional, depending on the specific environment in which the receiver is used, but is here specified as a mandatory feature for the balanced receiver used at the SH D-port.
(3) The common mode voltage is defined as the algebraic mean of the two voltages line-to-ground at the receiver input terminals.

device on the system must operate at any frequency up to a stated maximum, which may be less than 5 MHz. This means that the standard sets only an upper limit on the clock frequency, which may well be lower in any system and also may vary at will so long as the maximum system frequency is not exceeded. This frequency is the frequency of the slowest device on a particular system or 5 MHz, whichever is less. In a particular system, other considerations may restrict this frequency freedom more than the standard does.

Each time any controller receives a byte at its input, it must transmit one and only one byte from its output. At times this will not be the same byte. Thus, because bytes cannot be initiated by a controller, we will see that it is important that the serial highway driver keeps generating bytes even though it has no transactions in operation.

Figure 5.4 defines, in the top half, the timing relationships at a D-port. This shows that the data must be clocked in a period around the 0→1 clock transition and can be between 10% of the minimum clock cycle time before, up to 20% after the transition. Thus it follows that the 1→0 transition defines the start of a new clock period as in the lower diagram, when, for a short

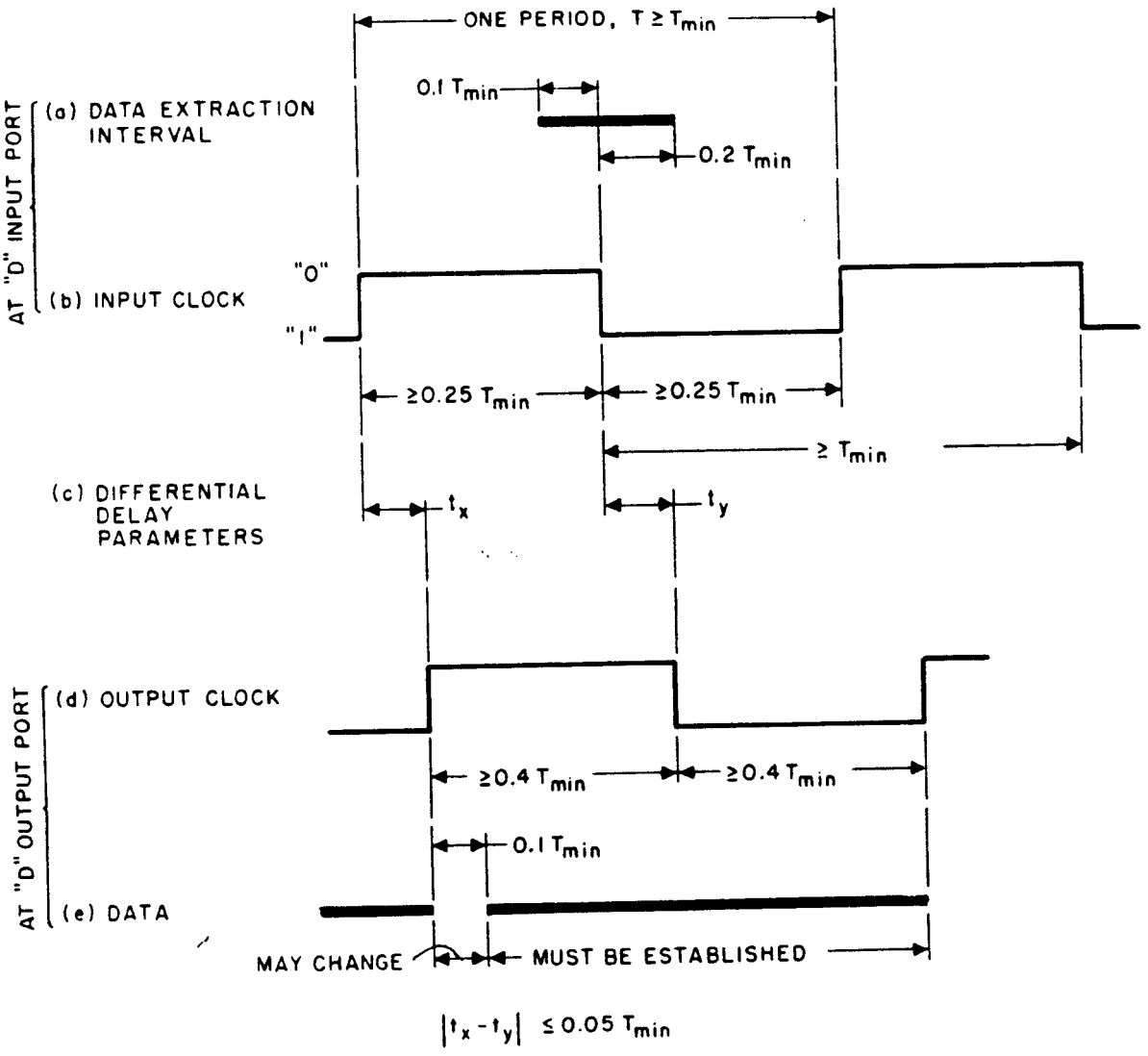


Fig. 5.4.
Timing of clock and data signals at D-ports.

time, the data lines can change. It will also be seen that the clock mark-space ratio is defined at the driver port to be no worse than 6:4, whereas the receiver port must be able to cope with a 3:1 ratio. Also, the differential delay parameters between input and output must not exceed 5% of the minimum clock cycle time. The remainder of the degradation is an allowance for the transmission-line effects. All this is necessary to ensure that everything on the highway will work reliably.

5.5 Data at the D-port

Pins are defined at the D-port for 1 bit-serial signal twisted pair or 8 byte-serial signal twisted pairs, as well as a clock-twisted pair. These pins are shown assigned in Table 5.3. Also shown are two free-use bus pairs, which are simply connected through the controller without any internal connection, a ground, bypass, and loop collapse control, of which more later.

Figure 5.5 shows the use of the free bus lines to turn a serial highway around and back to the driver through the same cable. With only two bus lines, this is limited to bit serial systems and it raises another restriction.

The signal is regenerated in each controller in the system when connected in a loop, so that the maximum loop length can be somewhat greater than the maximum single hop possible between RS 422 transmitters and receivers, given the particular cable and environment. If Bus 1 and 2 are used, then the total cable length is now limited to the maximum signal hop, because the signal is not regenerated on Bus 1 and 2.

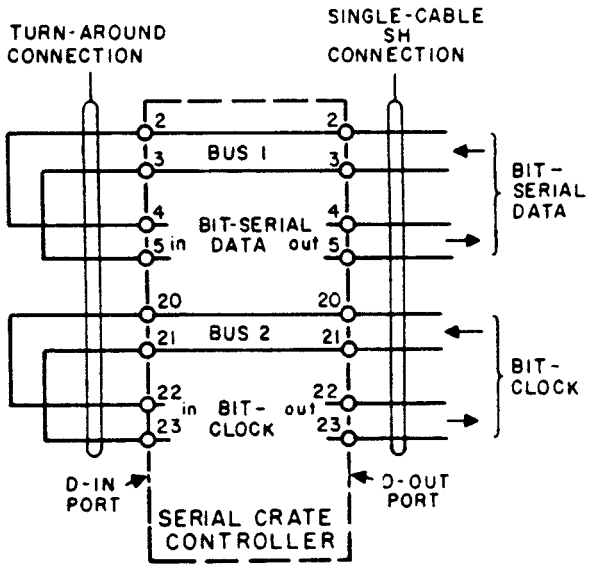


Fig. 5.5
Example of the use of BUS 1 and BUS 2 contacts at D-ports.

5.6 The Bytes of the Serial Highway

Figure 5.6 shows a breakdown of the types of bytes on the serial highway. These will be dealt with one by

Table 5.3

CONTACT ASSIGNMENTS FOR D-PORT CONNECTORS

D-Input Connector		Contact	D-Output Connector	
Circuit ground (Earth)		1	Circuit ground (Earth)	
Bus 1 Free use		2	Bus 1 Free use	
		3		
Bit serial data	in	4	Bit serial data	out
or Byte serial LSB	in	5	or Byte serial LSB	out
Byte serial Bit 2	in	6	Byte serial Bit 2	out
		7		
Byte serial Bit 3	in	8	Byte serial Bit 3	out
		9		
Byte serial Bit 4	in	10	Byte serial Bit 4	out
		11		
Byte serial Bit 5	in	12	Byte serial Bit 5	out
		13		
Byte serial Bit 6	in	14	Byte serial Bit 6	out
		15		
Byte serial Bit 7	in	16	Byte serial Bit 7	out
		17		
Byte serial MSB	in	18	Byte serial MSB	out
		19		
Bus 2 Free use		20	Bus 2 Free use	
		21		
Bit/byte clock	in	22	Bit/byte clock	out
		23		
Bypass control		24	Bypass control	
Reserved for control signal		25	Loop-collapse control	

NOTE: (1) Each balanced-signal input or output occupies a pair of contacts.
(2) The even-numbered contact is Terminal A, carrying Signal.
(3) The odd-numbered contact is Terminal B, carrying Signal.

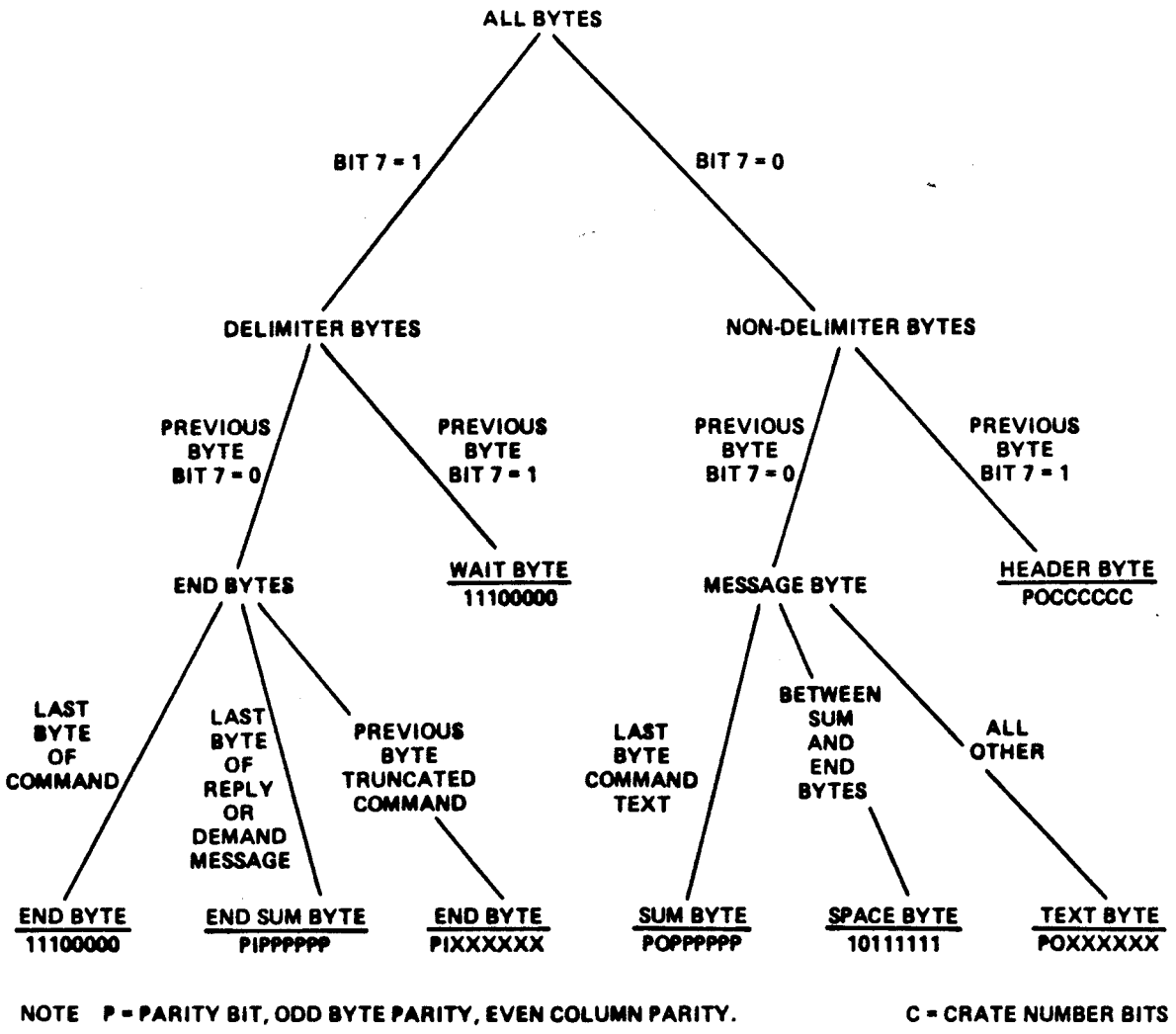


Fig. 5.6.
The bytes of the serial highway.

one below, but sufficient to note here is that the primary information is contained in the two underlined bytes on the far right, the header byte and the text byte. All the remaining bytes are associated with synchronization and error detection.

5.6.1 Delimiter Bytes. These are defined to have Bit 7=1 and conversely nondelimiter bytes will have 7=0.

5.6.2 END Byte. This delimiter byte is generated by the serial driver to terminate a complete command message, in which case 11100000 must be generated. It is also generated by the addressed controller to terminate a truncated command message. Bit 8 of an END byte must conserve odd byte parity. All unaddressed crate controllers must retransmit received END bytes. The addressed controller must either retransmit the received END byte of the command message or replace it by an ENDSUM byte.

5.6.3 WAIT Byte. This delimiter byte is generated both by the serial-highway driver and by the addressed controllers. The serial driver may generate WAIT bytes between successive command messages. The addressed controller replaces certain command sequence input bytes with WAIT bytes. These bytes are those between the END byte of the truncated command message and the header byte of the reply message. Also, WAIT bytes are substituted following the ENDSUM byte of the reply message up to and including the END byte of the command sequence, if such a gap in time exists. WAIT bytes must have the bit pattern 11100000. This is the same pattern as the END byte, but the context is different as Fig. 5.6 shows. The END byte is always preceeded by a nondelimiter byte; the WAIT byte is always preceded by a delimiter byte, sometimes an END byte. This pattern of bits has been defined to aid in establishing byte synchronization. Together with start and stop bits, the pattern is 1111000000; that is, there is a single 1→0 transition in the byte and one 0→1 transition that uniquely indicates the beginning of the start bit.

5.6.4 ENDSUM Byte. This delimiter byte is generated by the addressed controller to terminate each reply or demand message. The column-parity field (Bits 1 - 6) conserves even column parity from the header byte up to and including this ENDSUM byte. The delimiter bit is at Logic 1 and Bit 8 conserves odd byte parity.

5.6.5 SUM Byte. This nondelimiter byte is generated in the serial driver in the command message. It follows the last TEXT byte and serves, in Bits 1 - 6, to conserve even column parity from the header byte up to and including this SUM byte. The delimiter bit is at Logic 0 and Bit 8 maintains odd byte parity.

5.6.6 SPACE Byte. This nondelimiter byte is generated in a sequence by the serial driver between the SUM and END byte of a command sequence. These bytes provide the time to carry out the CAMAC (or other) operation and the space for the reply message, together termed the reply space. Clearly, the space required is a function of whether the highway is operating in bit or byte mode and its clock speed. Both of these are determined at the serial driver (and for the mode at the controllers as well); thus, it has all the information necessary to insert sufficient SPACE bytes. The delimiter bit must be Logic 0 and Bit 8 must conserve odd byte parity. It is recommended that the SPACE byte be 10111111. However, the addressed controller that is expecting SPACE bytes in the reply space must accept any nondelimiter byte as a substitute for a SPACE byte. These substitutes need not conserve odd parity.

5.6.7 HEADER Byte. This is the first byte of any message that has Bit 7=0, Bit 8 conserves odd byte parity, and Bits 1-6 define the addressed controller (or crate) or the originating controller in the case of demand or reply messages.

5.6.8 TEXT Byte. These nondelimiter bytes are the body of the message between a HEADER byte and either a SUM or ENDSUM byte.

5.7 Serial Messages

Having looked at all the kinds of bytes that can be on the serial highway, how are these used and why are so many different kinds needed? There are four defined types of messages to be found on the serial highway: command, truncated command, reply, and demand. Only one kind is generated by the serial driver, the command message. In normal operation, the serial driver never receives this message, but instead receives a truncated command message and a reply message from the addressed controller. Finally, as a result of an unmasked LAM in a controller, that controller can insert a demand message into the byte stream flowing through it.

5.7.1 The Command-Reply Sequence. A command message is sent to the particular controller in the format shown in Fig. 5.7. Laid out as a message in this way, it shows the function of the delimiter bit. The first byte is the

crate address in the header byte. This byte will already be being retransmitted by the controller when it recognizes that it is being addressed. The serial highway specification recommends that the addressed controller transmit an end byte followed by wait bytes in place of the remaining bytes of the command message. This is, however, mandatory on the L-2 crate controller. The MI field is to aid message identification and Fig. 5.8 shows that, for a command message, these 2 bits are zero. The first 4 bytes of the message define the crate, sub-address, function, and station number. For write operations, there follows 4 data bytes and then the SUM byte. If the byte and column parity are in order, and the crate is on-line (unless addressed to an internal feature of the controller), and the controller is not in the bypassed state (see later) and the MI fields are correct, then the operation is carried out and a reply sent in the space provided by the SPACE bytes.

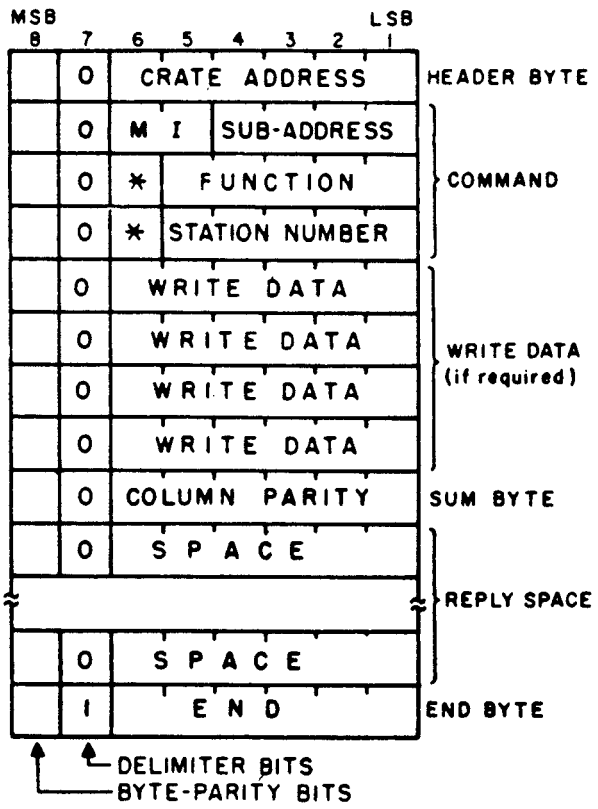


Fig. 5.7.
Command message: field assignments.

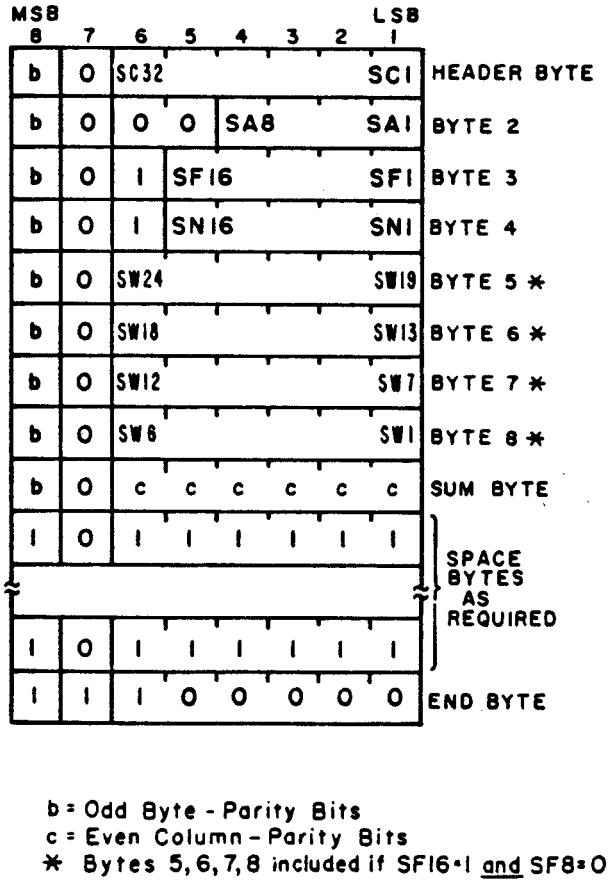


Fig. 5.8.
Command message: bit assignments.

Figures 5.9 and 5.10 show the bit and field assignments of the truncated command message. Note that this message has no column parity.

Figures 5.11 and 5.12 show the bit and field assignments for the reply message. Note that this contains the addressed crate number as the first byte; indeed, this is a repeat of the command message header byte by definition. The second byte contains the MI bits set to 01, and the status of the operation is in the remainder of the byte. SQ and SX are the CAMAC response lines, except that if the controller is off-line or bypassed, the operation is not carried out, and the status in the reply message is set to SX=0 and SQ=0 if the crate is off-line and SQ=1 if it is bypassed.

The ERR bit will be covered in more detail later, but basically it indicates that there was some form of transmission error detected by the addressed

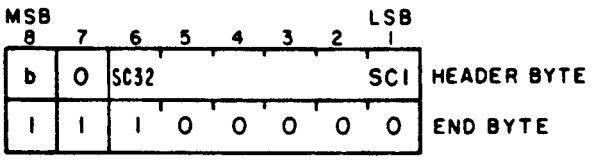


Fig. 5.9.
Truncated command message: bit assignments.

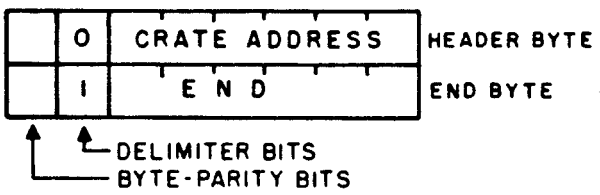


Fig. 5.10.
Truncated command message: field assignments.

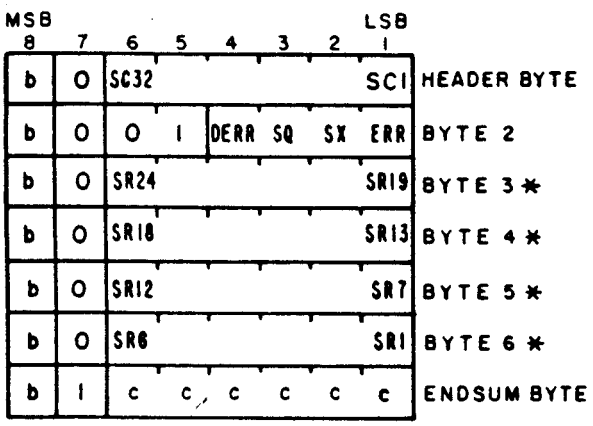


Fig. 5.11.
Reply message: bit assignments.

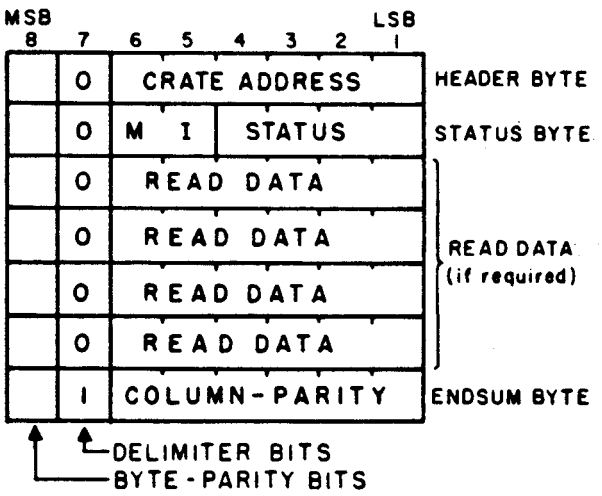


Fig. 5.12.
Reply message: field assignments.

controller. The DERR, Delayed Error, is the error bit of the previous reply message generated by the controller, the use of which we will see later. Naturally, the reply message for read operations will also contain the data read; finally, an ENDSUM byte delimits the message and provides column parity.

Figures 5.13 and 5.14 show the bit and field assignments of the demand message. The header byte identifies the crate reporting the LAM, whereas Byte 2 has the MI field, in this case only Bit 6, set to a 1 to identify this as a demand message. In both the command and reply MI field, Bit 6=0. The remaining 5 bits of Byte 2 contain information to identify the individual CAMAC plug-in. The message is completed with an ENDSUM byte to provide the column parity and to delimit the message.

Now let us look at how these messages work together around the serial highway. Figure 5.15 shows a command-reply sequence for a CAMAC control operation. On the left is the sequence of bytes pumped out by the serial highway driver. Each controller on the system examines the first nondelimiter byte, following one or more delimiter bytes, to see if its controller address is the header byte. If it is not, the controller remembers that a message is now passing through until it sees the next delimiter byte. If the address does match, the controller completes the retransmission of the header byte and then reads in the rest of the message, checking parity, both row and column, as it does. In place of the incoming message, first an END byte is transmitted and then WAIT bytes. This is shown in the block on the right-hand side of the figure. Once the SUM byte has been received and checked, the controller can go ahead and perform the CAMAC operation. AT t_3 of the CAMAC cycle (see Fig. 2.16), the controller can clock the result of the CAMAC cycle, in this

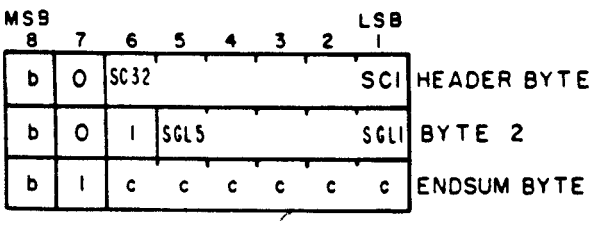


Fig. 5.13.
Demand message: bit assignments.

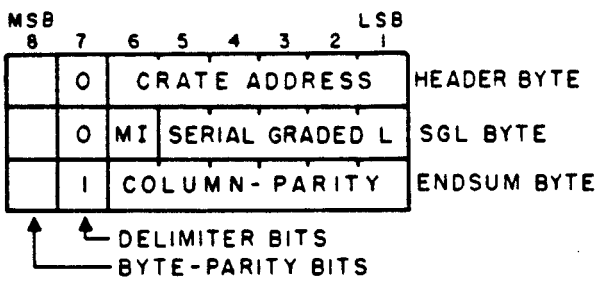


Fig. 5.14.
Demand message: field assignments.

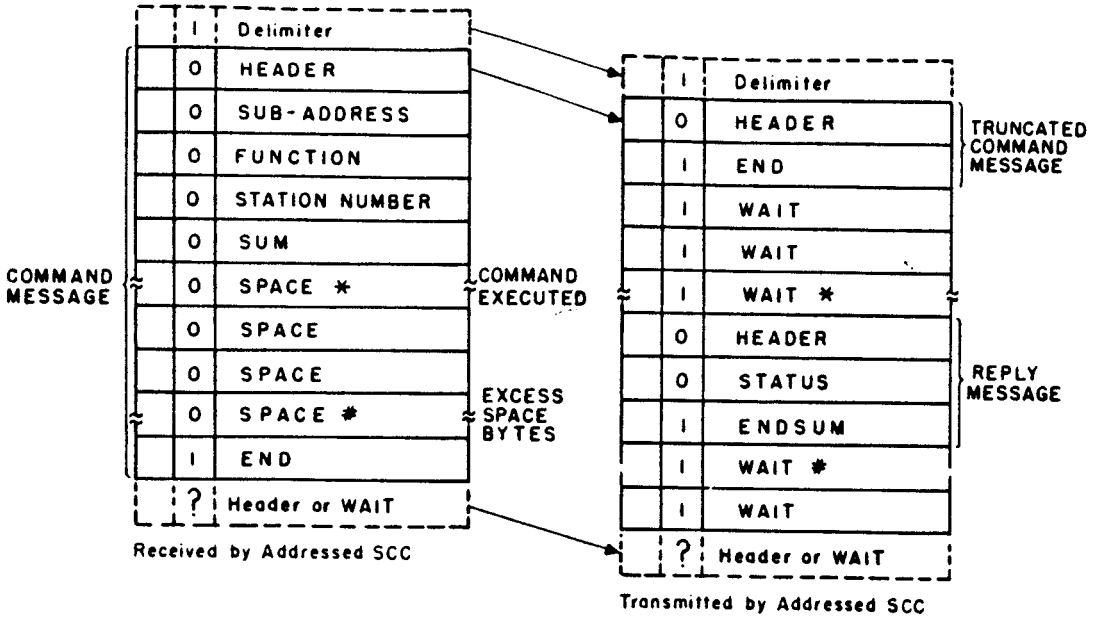


Fig. 5.15.
Command-Reply sequence: control operation.

case Q and X, and can start to transmit the reply message. Finally, excess SPACE bytes and the END byte are replaced with WAIT bytes by the addressed controller. The serial highway driver then sees back the truncated command message, which it can ignore, followed by some WAIT bytes and then the reply message. When the ENDSUM byte of the reply message is received by the serial driver, the transaction is complete.

Figure 5.16 shows the command-reply sequence for a read operation. The difference here is that, as data is involved, the message sequence is 4 bytes longer--in the reply message in this case. In all other respects, this operation is the same as in the previous example.

One can see here the advantage of standard CAMAC function codes in that the serial highway driver need only examine 2 bits of the function code to determine the type of command to send out and the type of reply to expect. This is shown in Table 5.4, which shows the length of command and reply messages for the three types of CAMAC operation and also shows the values of the relevant function bits. The final column shows the minimum command-reply transaction; however, this typically expands for the faster serial highways, as more bytes are needed to allow time for the CAMAC operation in the crate. The

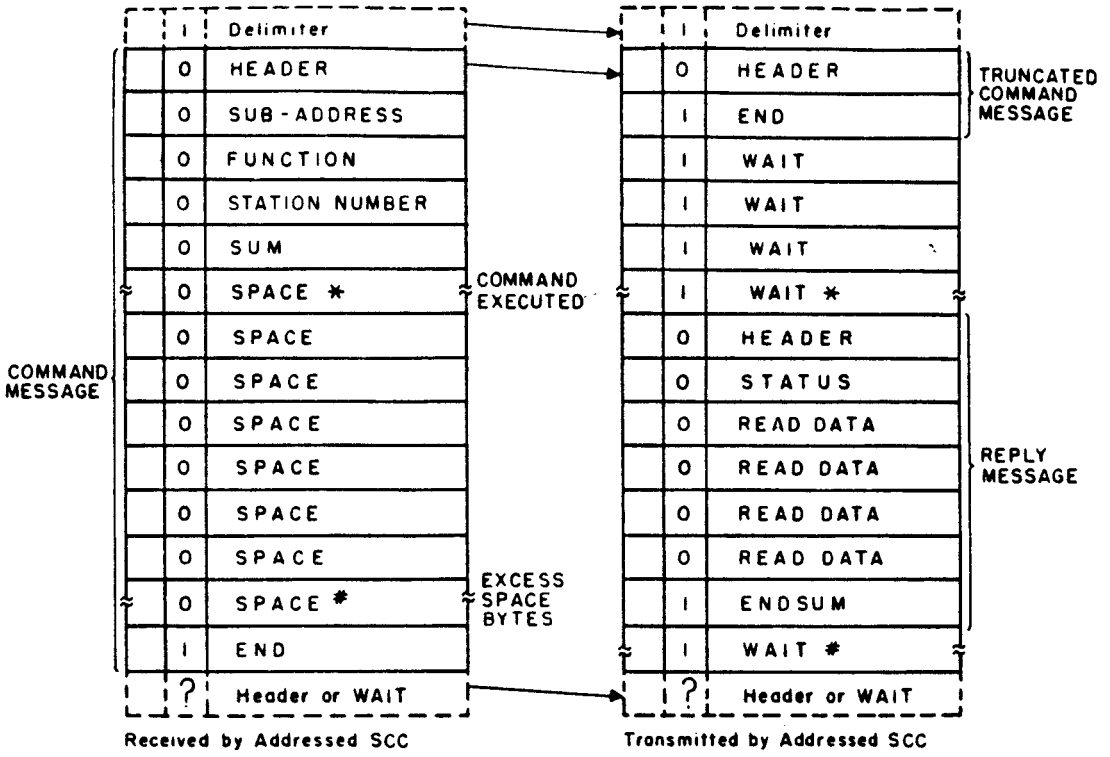


Fig. 5.16.
Command-Reply sequence: read operation.

Table 5.4

LENGTH OF COMMAND-REPLY TRANSACTIONS

	Function Field		Number of Bytes		
Operation	F16	F8	COMMAND from Header to SUM Inclusive	REPLY from Header to ENDSUM Inclusive	COMMAND-REPLY Transaction
Read	0	0	5	7	12[a]
	0	1			
Control			5	3	8[a]
	1	1			
Write	1	0	9	3	12[a]

[a]Minimum length, assuming that Reply Header is transmitted by SCC as first SPACE byte is received, and ENDSUM is transmitted as END byte is received.

number inserted is a setting in the branch driver and is usually given in the instructions for the driver's operation. Table 5.5 summarizes the use of the MI field in identifying the type of message. In the demand message, MI can be either 1 or 0 because, as for this message, it forms part of the SGL field.

Table 5.5

CONTENTS OF MESSAGE IDENTIFICATION FIELD

MESSAGE	MI-field M2	MI-field M1
Command	0	0
Reply	0	1
Demand	1	-

5.7.2 Demand-Message Generation. In the command-reply transaction, the addressed controller responds to the orders of the serial highway driver; therefore, the driver knows exactly the kind of messages to expect and when. Demand messages are generated quite asynchronously of the operation of the driver and are inserted into the byte stream by the controller. If the current byte going through is a WAIT byte, the controller can replace that with the first byte of a demand message. However, the controller cannot know if the next 2 bytes also will be WAIT bytes; if they are not, how does the controller transmit 3 or 4 bytes in a space intended for only 2?

First, let us look at the conditions that need to be satisfied before the controller can initiate a demand message:

- Demand-message generation must be enabled, signified by the appropriate bit in the controller.
- A demand is present that has either appeared since the last demand message was transmitted by the controller or was present when the controller changed to the demand-enabled stage.
- The controller is able to accept 3 incoming bytes while generating a demand message.
- The previous byte transmitted at the output port was a delimiter byte.

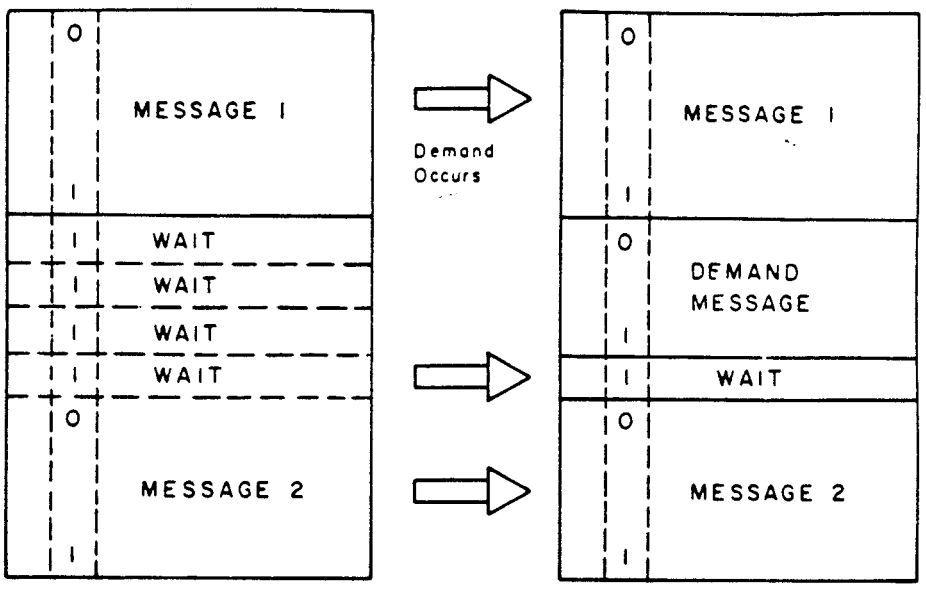
The first requirement is quite clear. The second means in practice that the Demand Message Initiate Signal presented at the SGL-encoder socket (see later) on the rear of the controller should be a Logical 1 with a 0→1 transition indicating the appearance of a new demand.

The third point indicates how a controller inserts a demand message into the byte stream without the risk of corrupting another message. In practice, each controller has a 3-byte delay register built-in in which it assembles the demand message. As soon as it fulfills the above conditions, it switches in the 3-byte delay, thus transmitting the demand message and storing up incoming bytes. Figure 5.17 shows before and after patterns for two cases. In the first case, WAIT bytes happen to be in the message stream just as the demand message is transmitted, so that the following message is not displaced at all in the message stream. In the second case, the above conditions are met by the delimiter byte, which is the last byte of Message 1. In that case, the 3-byte delay, with its demand message, is switched in after this delimiter byte, so that the second message is delayed by those 3 bytes. This shows how a demand message can exactly fit between existing messages on the highway. The second thing shown by this example is that the 3-byte delay is switched out of the serial loop as soon as the controller has transmitted a delimiter byte, and the 3 bytes in the delay registers are all WAIT bytes.

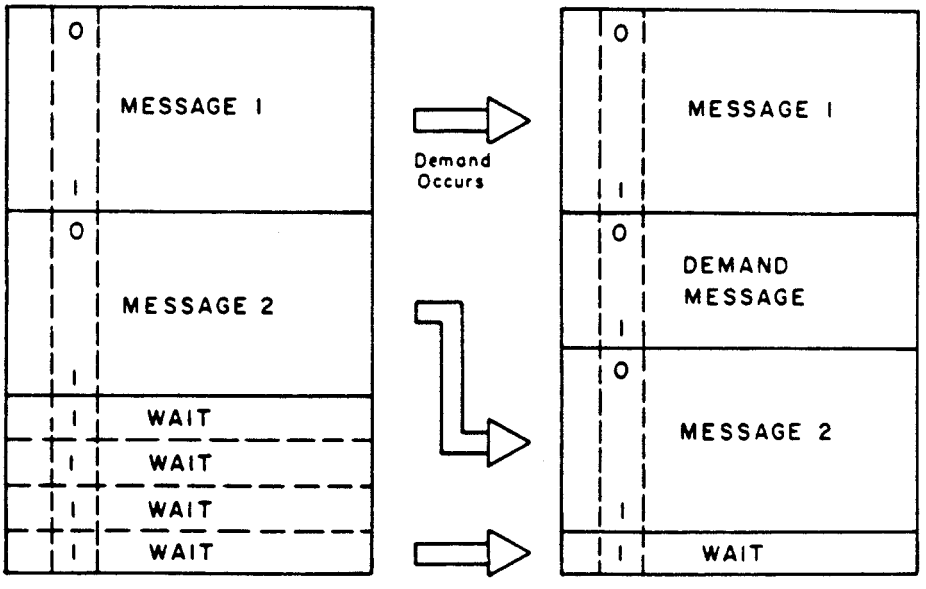
This need to clear the 3-byte delays for demand messages illustrates the importance of the serial highway driver pumping around WAIT bytes when it has no CAMAC operations to perform. This is so that demand messages can be sent, and the delays can be switched back out.

A more complicated illustration of command-reply messages and demands in a three-crate system is shown in Fig. 5.18. The figure illustrates what bytes and messages are passed on each of the four sections of the serial highway. In essence the serial driver sends out a command message with more than the minimum reply space. This message is followed by WAIT bytes. Just after the message has passed through Serial Crate Controller (SCC) 1, a LAM arises in that crate and, because the previous byte transmitted was a delimiter byte, it is immediately inserted into the message stream. The cross-hatched columns indicate when each SCC can insert a demand message. As soon as the demand message is transmitted, there are 3 WAIT bytes in the 3-byte delay of SCC-1; therefore, it is immediately switched out of circuit.

SCC-2 receives the command and strips off all but the header byte. It then inserts WAIT bytes until its reply is ready. After the ENDSUM byte of the reply, it replaces the remaining SPACE bytes of the incoming command message with WAIT bytes. A LAM then arises in SCC-2, where it can be transmitted immediately by SCC-2s switching in a 3-byte delay. However, once the demand



a) Demand Message directly displaces WAIT Bytes

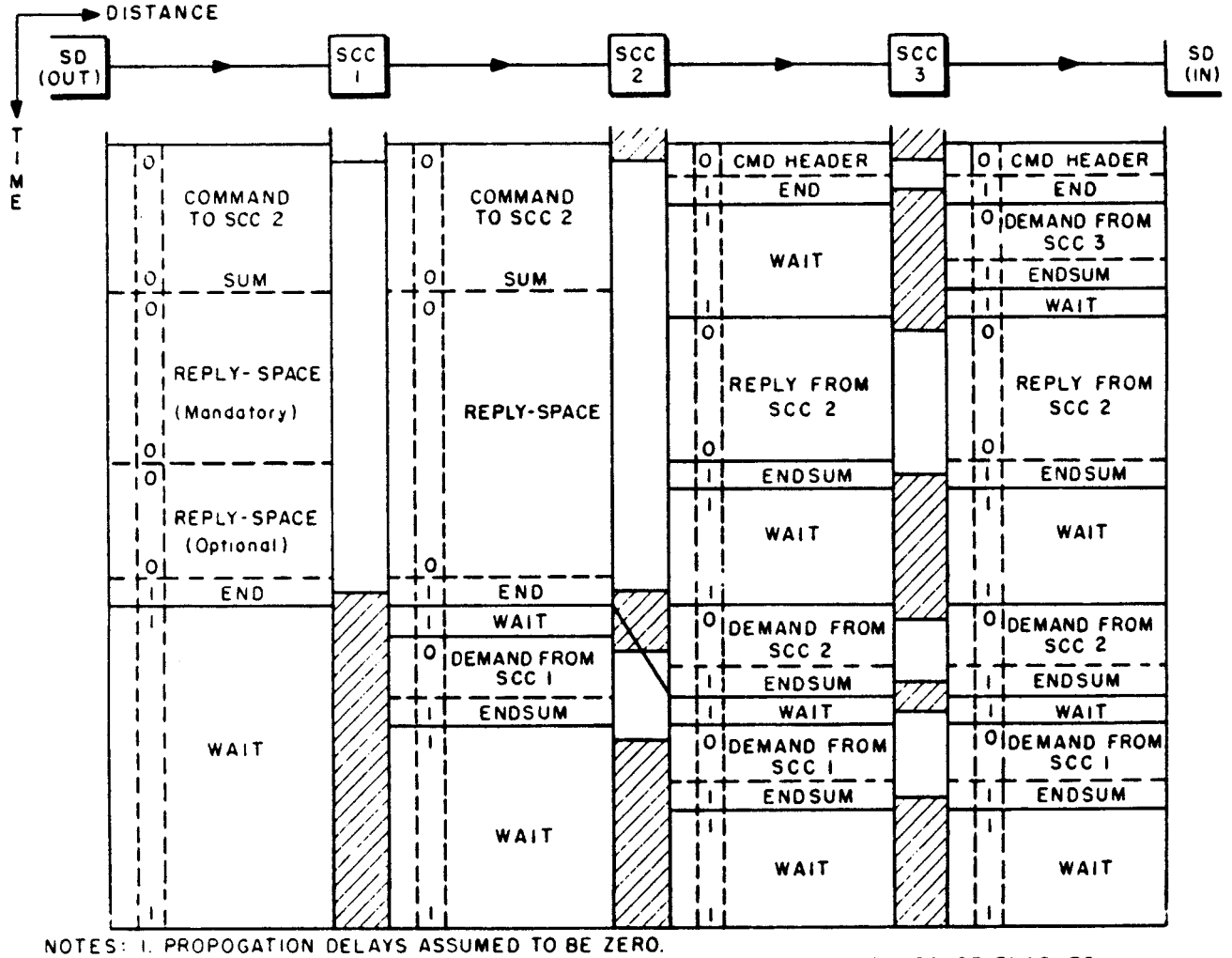


b) Demand Message delays incoming message

Fig. 5.17.
Demand message generation.

message is transmitted, the 3-byte delay contains 1 WAIT byte and the first 2 bytes of SCC-1s demand message. It is not until SCC-2 has just transmitted the ENDSUM byte that the delay circuit contains 3 WAIT bytes and satisfies the condition for it to be switched out of circuit.

Finally, Fig. 5.18 illustrates how SCC-3 can insert a demand message in the space that was formally the command message to SCC-2. As for the first demand message, as soon as the message is transmitted, the 3-byte delay contains 3 WAIT bytes and so can be switched out of circuit.



NOTES: 1. PROPOGATION DELAYS ASSUMED TO BE ZERO.
2. CROSS-HATCHED AREAS INDICATE SCC DEMAND AND COMMAND RESPONSE ENABLED.
3. DEMAND FROM SCC 1 IS DELAYED BY 3-BYTE DELAY IN SCC 2.

Fig. 5.18
Example of message sequence in a loop having three SCCs, showing demand messages in three contexts. (See Sec. 5).

Thus the serial driver receives a demand message from SCC-3 as it is still transmitting the command message to SCC-2, there being a maximum of a 3-byte delay in the serial loop. It is easy to see how the transmitting and receiving section of a serial driver need be only loosely interconnected ciruits! However, the construction of a serial driver is outside the scope of this work.

5.8 Bypass and Loop Collapse

The standard requires that the status register in any serial controller have two bits to control two outputs of the D-port. Bit 12 of the status register is the bypass bit; its action, when set, causes the serial highway to bypass this controller. In fact the controller still receives command messages, but it can no longer send replies. The controller is automatically bypassed when the crate is powered off. It is required that if the controller receives a command to set the bit to a "1" it must be so set after a reply message has been generated. If it is commanded to set it to a "0" then the reply message must be delayed for 100 ms ± 10%. The serial driver has to generate sufficient reply space for this delay, which is to allow relays to settle.

The other requirement is that the crate must have the bypass bit set on power-up and it must remain set until specifically overwritten. Thus part of the initialization process for a crate must be to unbypass it. When in the bypassed state, the controller must not respond to any command other than those that include clearing the bypassed bit of the status register. A bypassed controller receiving a command that does not include setting Bit 12 to a "0" must generate a reply message appropriate to the command with SX=0 SQ=1 in the status field. If the bypass is by an external device, such a message will not be transmitted, but rather the complete command message will be propagated back to the serial highway driver.

Each controller must provide a loop-collapse signal associated with Bit 11 of the status register, according to Fig. 5.19. This bit, if set to a "1" by command, is intended to collapse the serial loop past this controller so that this controller is still connected to the serial driver. A delay of 10 ms ± 10% is required before the reply message to a command "set this bit to a 1" is sent. A command "set the bit to a 0" has a reply sent before the bit is actually set to "0". As before, the serial driver has to generate a sufficient reply space for the delay.

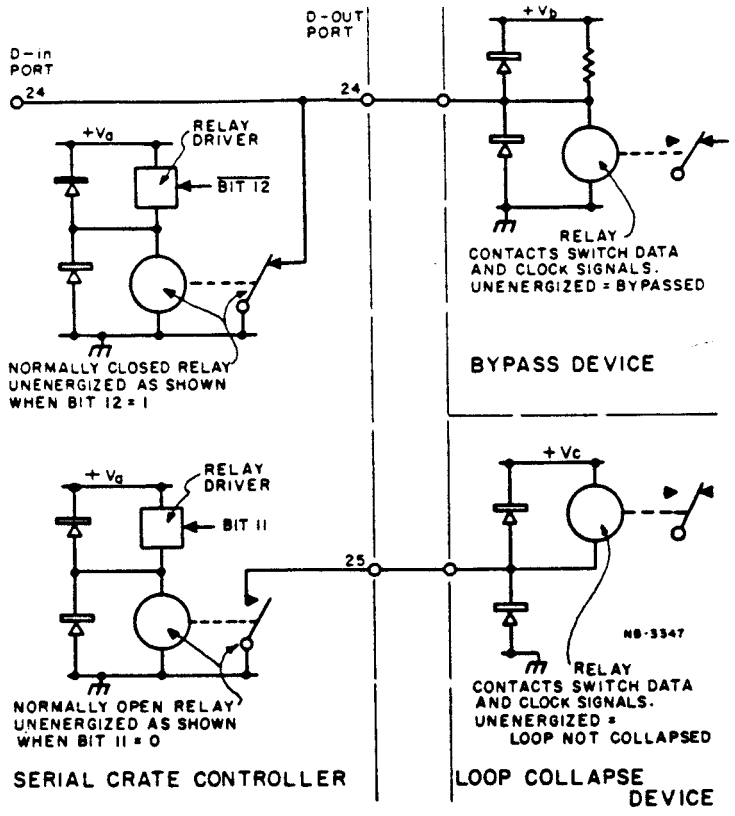


Fig. 5.19.
Examples of circuits for control signal sources and receivers. (Bit 11 and Bit 12 are signals from the status register of SCC.)

This loop-collapse facility is to allow systems to automatically reconfigure themselves in case of a problem. The bypass facility is for powered-down crates on a highway; in bit-serial systems, the necessary relay is contained in the L-2 controller.

Figure 5.20, by way of illustration, shows a bypass circuit that might be used external to a controller. The relay contacts are shown in the bypassed state and the termination resistor is switched in when unbypassed to terminate the input to the balanced receiver/driver.

Figure 5.21 shows an example of a loop-collapse circuit. This works somewhat in a similar way to the bypass control except that here the controller output is either switched to the next controller, to the normal state, or to the return line--in which case downstream controllers become disconnected.

Table 5.6 shows the electrical standards for the D-port control signals. This indicates the currents and voltage levels that the external devices must accept as Logic 1s or 0s.

Schemes are also available for having back-up highways, but in Laboratory systems these are seldom justified.

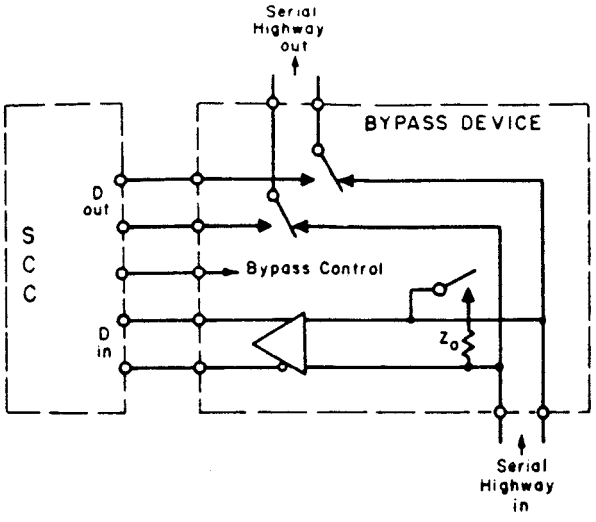


Fig. 5.20.
Example of bypass switching for one D-port signal. In a bypass device for bit-serial D-port signals, this is duplicated for data and clock signals.

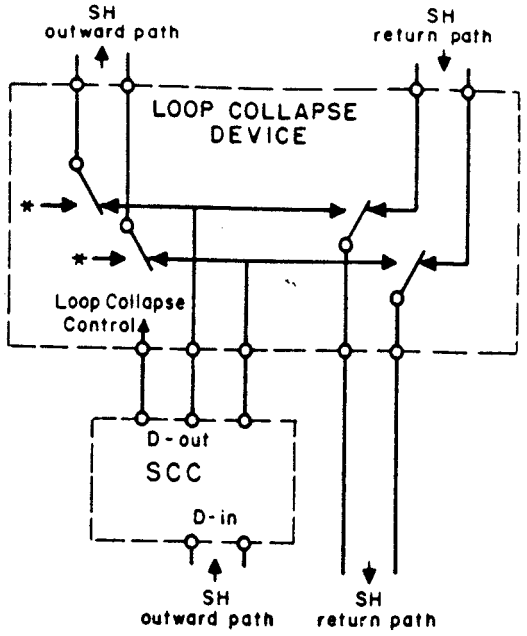


Fig. 5.21.
Example of loop-collapse switching for one D-port signal. In a loop-collapse device for bit-serial D-port signals, this is duplicated for data and clock signals. The clock-signal output to the disconnected part of the SH Loop is held in a fixed-logic state by the conditions at *-*.

Table 5.6

STANDARDS FOR CONTROL SIGNALS AT D-PORTS

Logic State	State of Control Line	Current Drawn from Control Line by the Source	Receiver Must Respond Correctly[a] to Control Signal in the Range:
0	Free	Magnitude of current not more than 100 μA for Control Line between 0 and +25 V	+10 to +24 V
1	Grounded	Minimum current sinking capability 115 mA for Control Line at +0.5 V	0 to +3 V

[a]The receiving device must respond within 80 mS.

5.9 The Controller Status Register

Table 5.7 lists the bits of a controller status register. Basically it can be seen that whereas in the A-1 controller for the parallel highway, the various features of the controller are controlled by individual commands, in a serial controller they are controlled by a single register. Bit 16 indicates that a LAM is present in this controller and that, usually, a demand message has been sent. Bit 10 allows a LAM to be set in the crate from the serial highway, a useful feature for checking out systems. Bits 4, 5, and 6 are for error-recovery procedures. The remaining bits of the status register are self-explanatory or have been covered already. Table 5.8 defines the state of the status register on power-up.

5.10 Controller Commands

These are listed in Table 5.9. Four operations are defined to operate on the status register, and this table also indicates two other registers that are available in the controller. The Data Field register contains the data of the last successful read operation performed by this controller. This register is needed for error recovery when the data is changed by the read operation. In that case it is no longer available in the module read, so that this register needs to be accessed to recover.

The second register is the LAM-pattern register, which makes available the pattern of interrupting stations in the crate. The state of L1 is indicated on Bit SR1 and so on. This register is unaffected by the state of the demand-enable bit in the controller status register. In fact, this register is not mandatory and, if it is not implemented, SX=0 must be returned by the controller in response to an attempt to read it.

If the off-line switch on the controller is set, then the controller must not respond to commands to the status register. If the dataway is off-line either because the front panel switch is set (see Bit 14 of the status register) or because Bit 13 of the status register is set, the controller must reply to any dataway commands, N(1) - N(23), with SX=SQ=0.

Table 5.7

ASSIGNMENT OF STATUS REGISTER BITS

Status Register Bit	Write Operations: Logic "1" into Register controls	Read Operations: Logic "1" from Register indicates	Comment
1	Generate Z	Always 0	Automatic reset to logic "0"
2	Generate C	Always 0	
3	Set I=1	I_{out}=1	
4	-	DERR=1	Previous Reply Status
5	-	DSX=1	
6	-	DSQ=1	
7	-	Dataway I=1	
8	(Reserved)	(Always 0)[a]	Reserved
9	Enable Demands	Demands Enabled	
10	Set Internal Demand L24	Internal Demand L24=1	
11	Collapse Loop	Loop Collapsed	Indicates Control Signal
12	Apply Bypass	Always 0	
13	Dataway off-line	Dataway off-line	
14	-	DOF Switch "off-line"	
15	(Reserved)	(Always 0)[a]	Reserved
16	-	Selected LAM Present	
17	(Reserved)	(Always 0)[a]	Reserved
18	(Reserved)	(Always 0)[a]	Reserved
19	(Reserved)	(Always 0)[a]	Reserved
20	(Reserved)	(Always 0)[a]	Reserved
21	As required	As required	Free-use
22	As required	As required	Free-use
23	As required	As required	Free-use
24	As required	As required	Free-use

[a]Applies while the bit has Reserved Status.

Table 5.8

INITIAL STATE OF STATUS REGISTER BITS AFTER POWER-UP

Status Register Bit	State After Power-up	Condition
3	1	Inhibit set (I=1)
9	0	Demands disabled
10	0	Internal demand L24=0
11	0	Loop not collapsed
12	1	SCC bypassed
13	1	Dataway off-line

Table 5.9

COMMANDS IMPLEMENTED BY SCC

	Command			Response	
Operation	SN	SA	SF	SQ	SX
Status register					
Read	30	0	1	1	1
Write	30	0	17	1	1
Selective set	30	0	19	1	1
Selective clear	30	0	23	1	1
Re-read data field	30	1	0	DSQ	1
Read LAM-pattern	30	12	1	1	1

5.11 LAMs and the Serial Controller

The standard specifies that if a facility is provided on a serial crate controller for an external device to sort or grade LAMs, then a standard SGL connector should be provided. This connecter is a double-density 52-way Cannon connector. It is designed so that only simple connections are necessary so as to be able to cause demand messages to be sent by the controller. With no connections, no demand messages will be sent. Figure 5.22 shows schematically the connections at the SGL connector and the associated logic. Connections between Pin 50 and Pin 23 will cause the logical OR of the crate L lines to start a timer. The output of the timer on Pin 19 is then routed to Pin 21, where it will cause a demand message to be sent. Figure 5.23 shows the timing of this. With demands enabled (top line) a LAM occurs, which causes the SUM signal on Pin 50 to go low, true (Line 3). Connecting the SUM signal directly to the STIM signal on Pin 23 causes TIMO (Pin 19) to go low, and hence DMI (Pin 21) to go low, causing a demand message to be sent. At this negative edge of DMI, Internal Repeat is high; thus the 5 bits of the SGL field of the demand message are taken from the SGL-encoder Pins 3, 5, 7, 9, and 11. On a simple connector, these signals can be wired to five separate LAM signals from the even-numbered Pins 2 to 48, giving a 5-bit LAM pattern. In this case, any other LAM occurring in the crate will cause the SGL field to equal 0.

What happens now if the LAM is serviced and cleared before the time-out? This is represented by the dashed waveforms in Fig. 5.23a. In this case SUM and hence STIM are removed. As a result, both TIMO and DMI go high and the demand cycle is over.

But what then is the function of the internal timer? It is specified that the controller must have a choice of time-outs between 1 ms and 10 s, and the time-out period must be unaffected by dataway cycles that might temporarily remove the LAM. Let us look at what happens if the SUM signal does not go away inside the time-out period. Figure 5.23a shows that at the next negative-going edge of TIMO, the internal repeat signal now is true. Returning to Fig. 5.22: at the top of the figure, this signal is OR'ed with an external repeat signal and then applied to OR gates on the 5 SGL lines. The effect will be that the 5 SGL bits transmitted this time, as a result of the negative transition of DMI, will be all "ones". This message, with the SGL field = 11111, is a hung-demand message and, at the highway driver, should be taken to indicate that

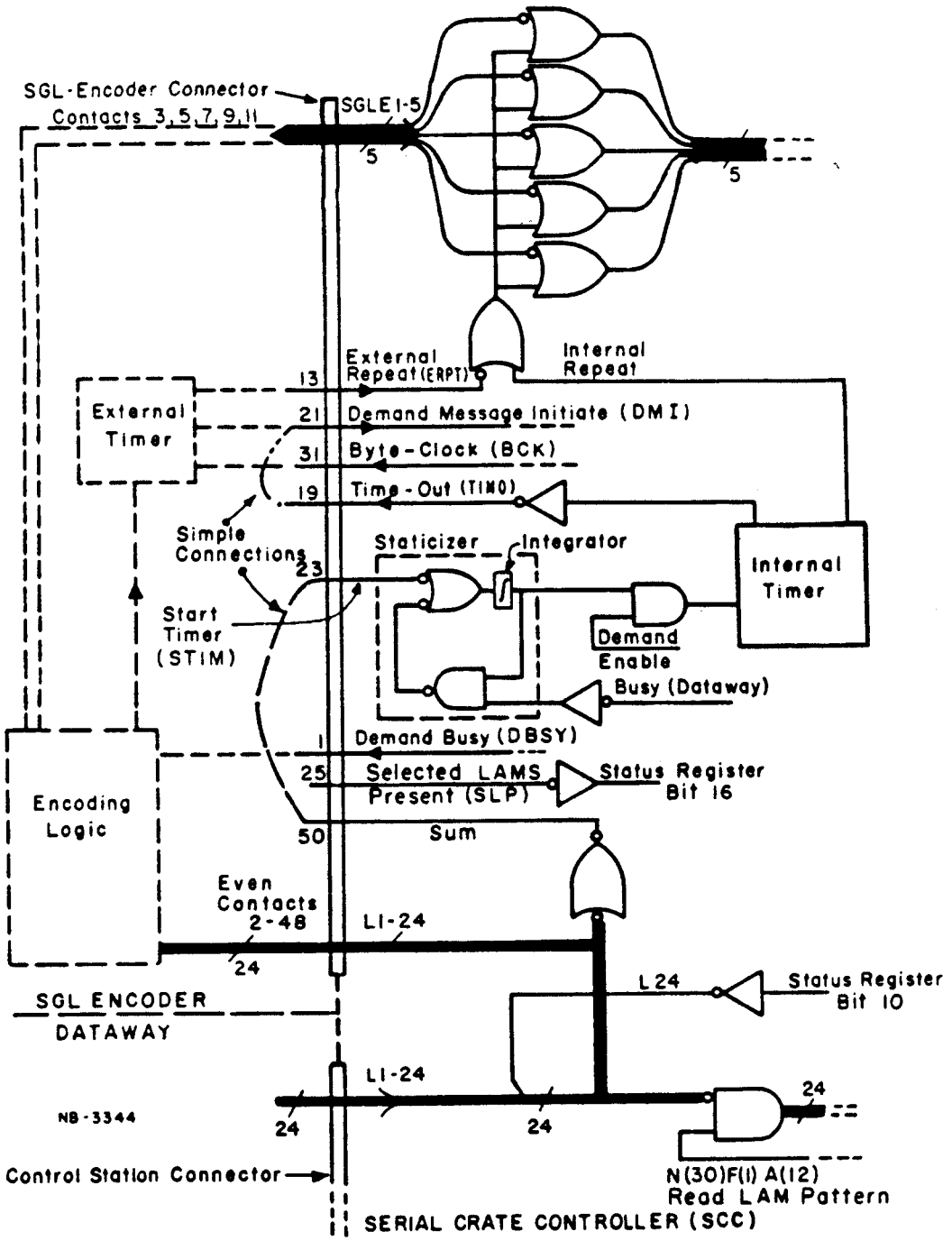
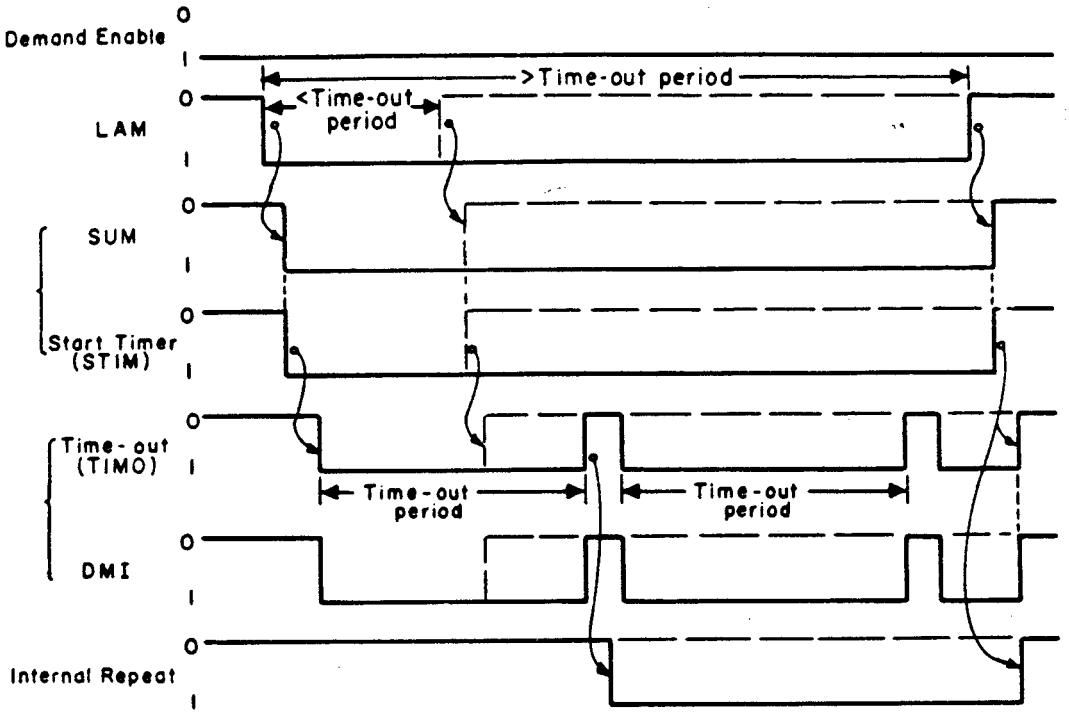
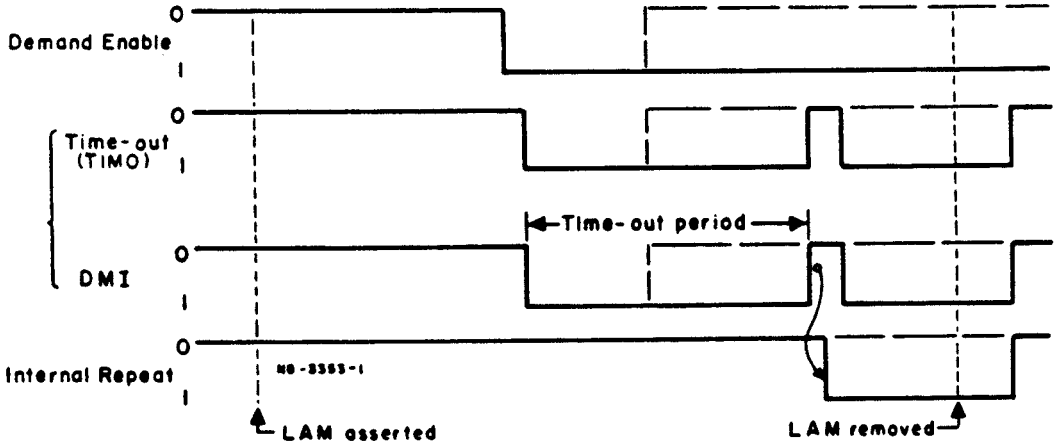


Fig. 5.22.
Example of associated parts of SCC and SGL encoder.



a) Shown for duration of LAM less-than and greater-than Time-out period



b) Similar to "a" except for Demand Enable after LAM assertion.

Fig. 5.23.
Relationship between signals at SGL encoder concerned with demand message generation.

possibly there is a demand in the crate that has not been responded to for whatever reason, including possibly a corrupted demand message.

A hung demand also can occur if a simple SGL-encoder connector is used and a second demand occurs before the first is cleared.

As will be seen from Fig. 5.22, all the signals exist for an external module to take over all the functions of deciding when to generate demand and hung-demand messages. The byte clock is provided so that the time-out can be in terms of the byte clock.

Figure 5.23b shows the time-out sequence when demands are disabled at the time the LAM first occurs. The start of the timeout sequence is delayed until demands are enabled, and then everything follows on as before, but delayed.

Figure 5.24 shows the deviation of the byte clock at the SGL encoder from the clock signal at the D-port connector. Figure 5.24a merely shows that it might be delayed by the logic when the serial highway is in byte serial mode; whereas, when the serial highway is in bit serial mode, the byte clock is a positive going pulse during the stop bit of each byte at the D-port.

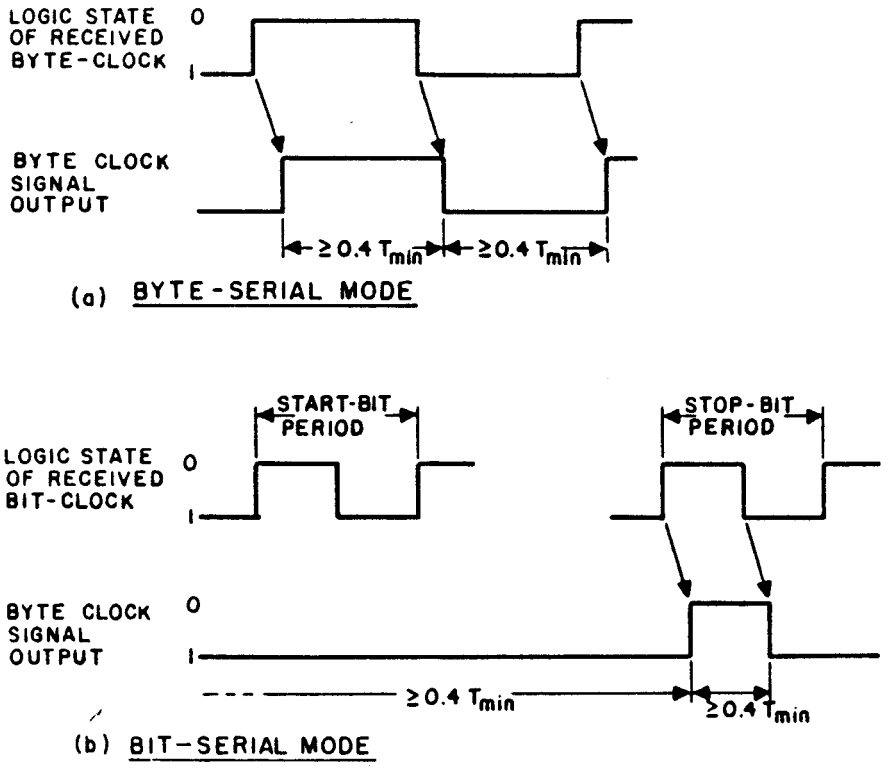


Fig. 5.24.
Relationship between byte clock signal at SGL-encoder connector and received Bit/Byte clock signals.

5.12 Errors on the Serial Highway

Figure 5.25 reminds us of the error detection scheme for all serial highway messages. Basically, it has odd byte parity and even column parity for Bits 1 - 6. Thus, should one or more bits be corrupted in a message there is an extremely high chance that the error is detected. But, how should the serial crate controller respond to a corrupted message? Table 5.10 indicates the controller responses in the reply message. As we see, the ERR bit of the status field indicates a parity error or an error in the MI bits of the command message, that is, MI not equal 0.

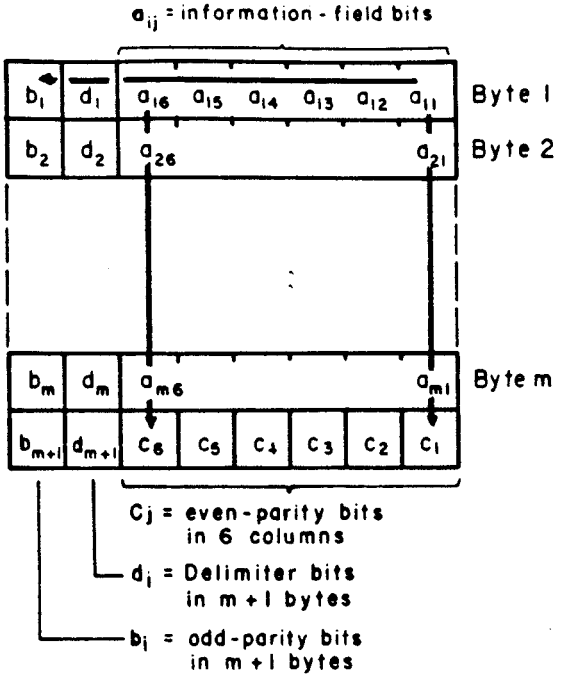


Fig. 5.25.
Geometric error detection as applied to the serial highway.

Table 5.10

ERROR INDICATIONS IN REPLY MESSAGE

	Reply Message			
	Status Field			
Execution of Command	ERR	SX	SQ	Length (bytes)
Successful	0	1	Q[a]	3 or 7
Unsuccessful				
Dataway off-line[b]	0	0	0	3 or 7
Command not accepted	0	0	0	3 or 7
Bypassed	0	0	1	3 or 7
Not executed (Parity or MI error)	1	0	0	3

[a]Response from the addressed feature of the module or controller.
[b]These can be distinguished by reading the contents of the status register. Bit 13=1 indicates dataway off-line.

The problem arises when error recovery is tried. When a corrupt reply message is received, was the highway traffic only corrupted between the addressed crate and the serial driver, or was it corrupted on both sides of the addressed crate? In this case, for a read operation, one either reads the data out of the controller or repeats the operation. The serial driver needs to read the controller status register to determine if the last command received was received successfully--the DERR bit indicates this. The whole subject of error recovery is covered thoroughly in ESONE SD/02, DOE EV-0006. Happily, one can buy serial drivers that take care of errors.

Figure 5.26 shows the flow diagram for the logic of a serial crate controller. The quiescent state of the controller is spinning in the top box looking for a header. If one is found that is not addressed to this controller ($\overline{DL} \cdot \overline{CA}$), it is simply passed. If a demand arises in the crate and the last byte was a delimiter (DL • Demand Present), a demand is sent. Finally, if a header is found that is addressed to this crate ($\overline{DL} \cdot CA$), then the receive, execute, and reply sequence is gone through. The final logic is to replace excess reply space with WAIT bytes.

Figure 5.27 extends this diagram to add all error conditions. Most of the extra lines are added because of the detection either of a framing error (bit serial only) or a delimiter byte when one is not expected. The numbers by the boxes refer to sections in Chapter 16 of the Standard, and I refer you there for more detail.

For the serial crate controller L-2, Fig. 5.28 shows a schematic logic diagram. The serial highway comes in at top left and is retransmitted top right. The input side is on the left of the two connectors, which represent the SGL-encoder connector at the top and the dataway connectors at the bottom. Going down the left side one first sees the serial-parallel converter, byte synchronization etc., and then the various testing of bytes for parity and various standard bytes. Following that is the 3-byte demand-message delay and then the various registers that clock in the various parts of the data in a command message.

In the middle of the figure are the LAM circuits, the status register, and dataway control circuits. On the right hand side are the circuits that generate the reply and demand messages. I leave it to the reader to work through the figure in detail.

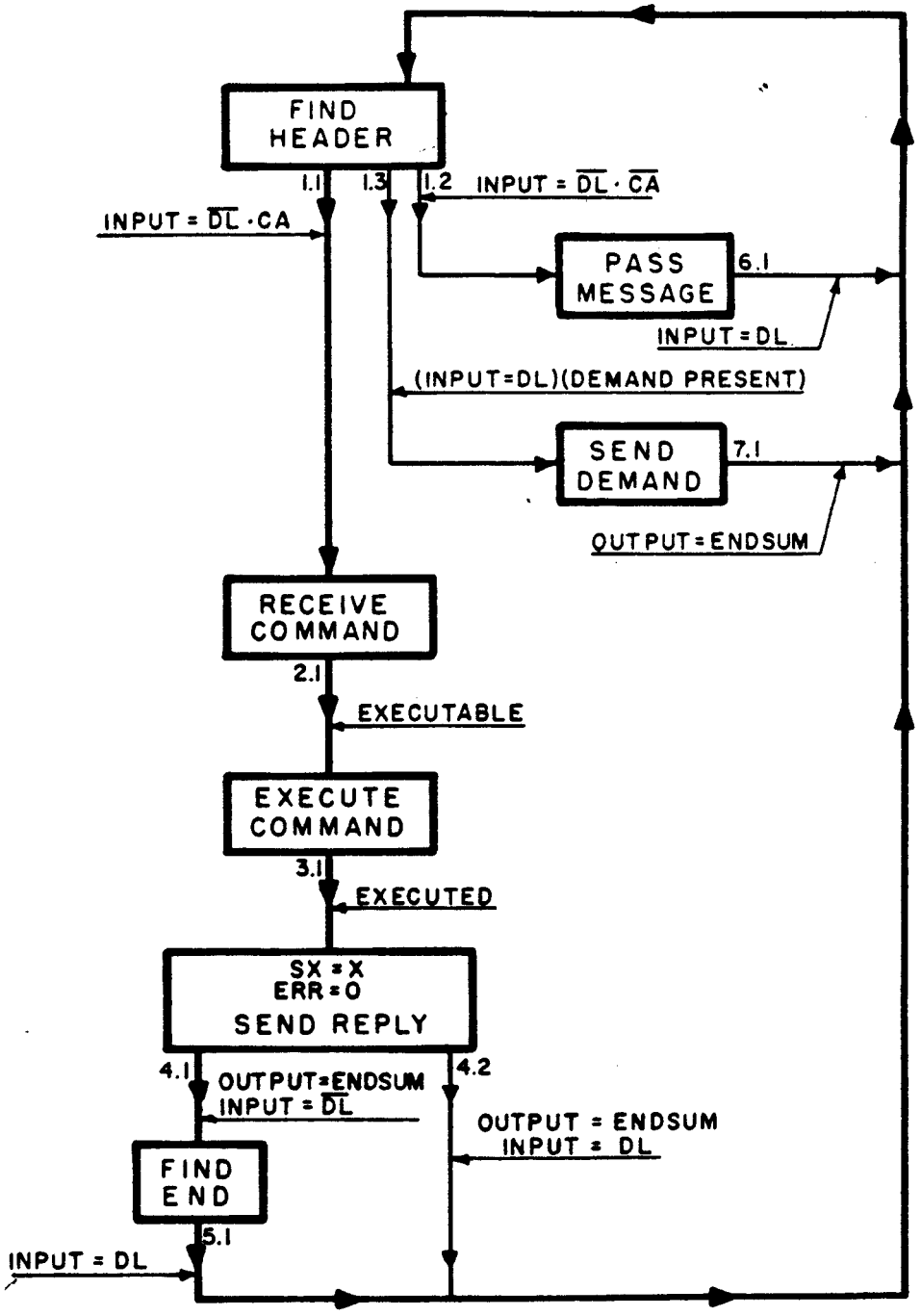


Fig. 5.26.
Major-state sequence in SCC--omitting all error conditions.

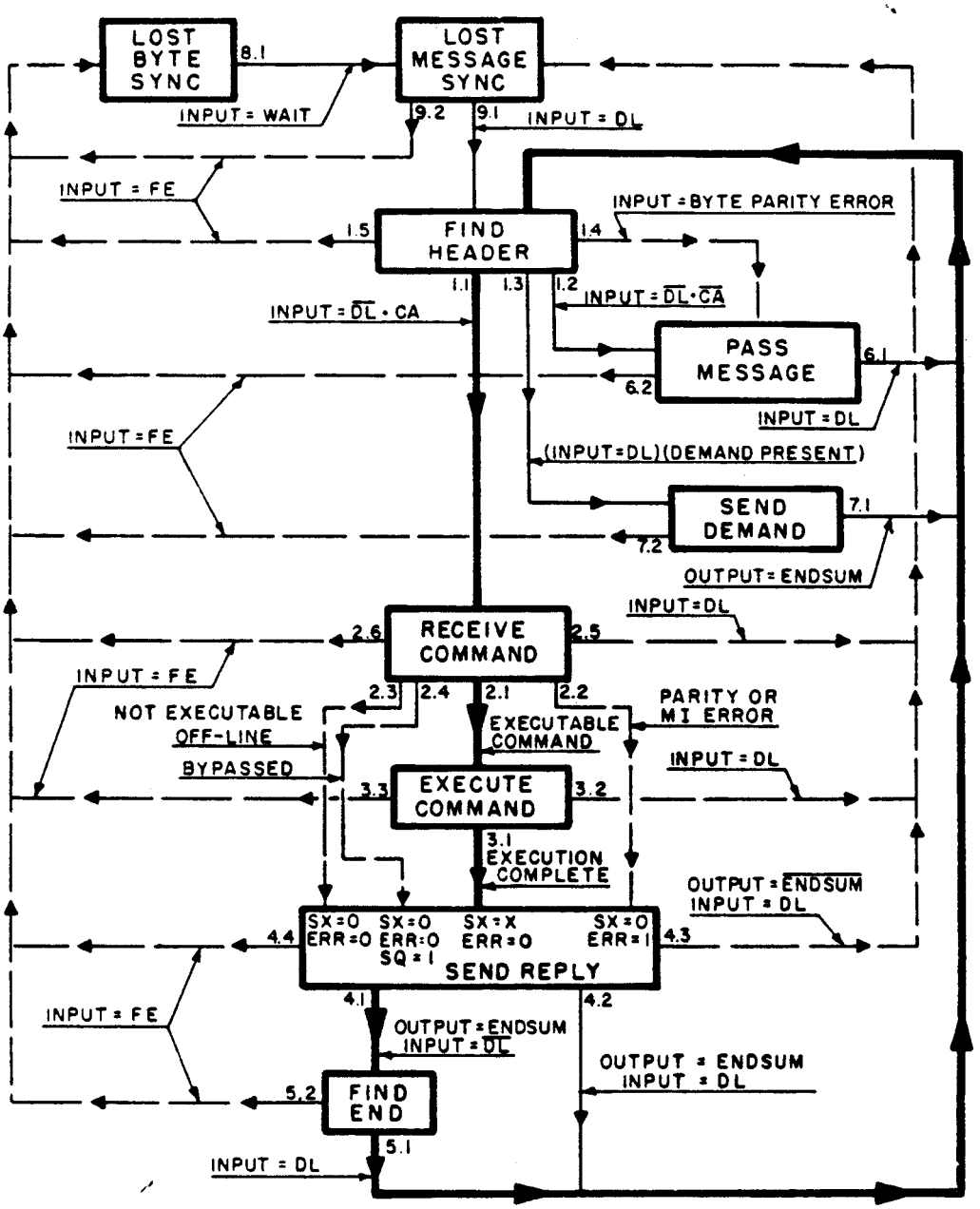


Fig. 5.27.
Major-state sequence in SCC.

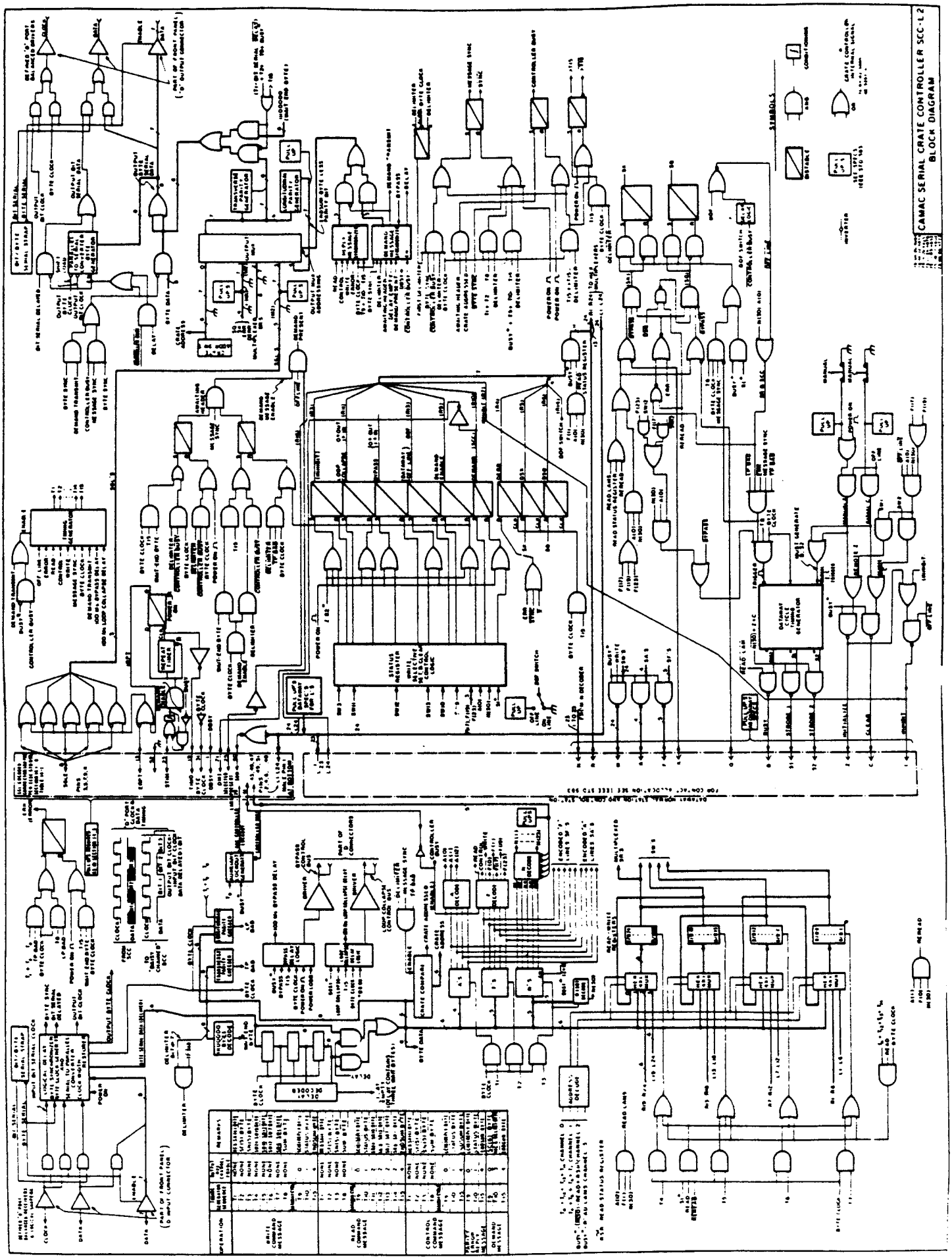


Fig. 5.28. Serial crate controller SCC-L2: block diagram.

5.13 Summary

Although the serial highway is a complex system, it does provide for more flexibility and solidity than the parallel highway. Happily, the majority of the complexity is taken care of in standard software and hardware.

6. AUXILIARY CONTROLLER IN A CAMAC CRATE

6.1 Introduction

The basic CAMAC standards make the assumption that there is but a single controller of any CAMAC crate. At the time CAMAC was defined, CAMAC modules were seen as boards containing but hard logic; therefore this was a quite reasonable restriction, and it resulted in the dataway design that made the right-hand station of the CAMAC crate the control station. Since that time, the integrated circuit datasheets have changed somewhat, and now it seems to be quite reasonable that any CAMAC module might want to operate the dataway. How can these modules be given control of the crate via the right-hand station, and how can they arbitrate for use of the crate in time? As we will see, the necessary hooks were inserted into the serial crate controller Type L-2 and a new parallel crate controller Type A-2 was defined.

6.2 Auxiliary Controllers with a Serial Crate Controller

Table 6.1 lists all the pins at the SGL connector on a serial crate controller. I have marked seven signals that are provided over and above those needed for a straight SGL connector. These signals allow one or more auxiliary controllers to operate in a CAMAC crate without mutual interference on the dataway. Because the serial crate controller must have access to the dataway immediately after the SUM byte of a command message has been received, the Auxiliary Controller Lockout signal (ACL) is asserted by the controller as soon as a header byte addressed to the controller is received, and it is removed at the completion of the dataway cycle, or when the operation is abandoned. In a 5 Mbyte/s serial system, that gives 4 bytes before the start of a CAMAC cycle or 800 ns. Thus an auxiliary controller must abandon a CAMAC cycle if it has

Table 6.1

CONTACT ASSIGMENTS AT SGL-ENCODER CONNECTOR

Contact	Signal	Direction[a]	Contact	Signal	Direction[a]
1	Demand Busy (DBSY)	Out	2	L1	Out
3	Graded-L SGLE1	In	4	L2	Out
5	Graded-L SGLE2	In	6	L3	Out
7	Graded-L SGLE3	In	8	L4	Out
9	Graded-L SGLE4	In			
11	Graded-L SGLE5	In	10	L5	Out
13	External Repeat (ERPT)	In	12	L6	Out
15	(Reserved)		14	L7	Out
17	Request Inhibit Out[b]		16	L8	Out
19	Time-out (TIMO)	Out	18	L9	Out
21	Demand Message Initiate (DMI)	In	22	L11	Out
23	Start Timer (STIM)	In	24	L12	Out
25	Selected-LAM Present (SLP)	In	26	L13	Out
27	(Reserved)		28	L14	Out
29	Auxiliary Controller Lockout (ACL)	Out	30	L15	Out
31	Byte Clock	Out	32	L16	Out
33	Free Use	In or out	34	L17	Out
35	Free Use	In or out	36	L18	Out
37	Free Use	In or out	38	L19	Out
39	Free Use	In or out	40	L20	Out
41	Controller Busy (CBY)	Out	42	L21	Out
43	Station Number N1	In	44	L22	Out
45	Station Number N2	In	46	L23	Out
47	Station Number N4	In	48	L24	In or Out
49	Station Number N8	In	50	Sum	Out
51	Station Number N16	In	52	0v	-

[a]"Out" indicates signal generated by SCC. "In" indicates signal received by SCC.
[b]The crate controller needs only a pull-up on the request inhibit (Contact 17) in order to be compatible with the ACB.

not yet asserted S1, but it has time to complete the cycle if it has already passed the start of S1 (See Fig. 2.16).

The remaining five lines are the encoded N lines, and these instruct the controller to assert the N line defined.

The SGL connector can thus form a bus that is connected to a serial LAM grader and various auxiliary controllers in the crate.

Finally, for completeness, Table 6.2 shows the signal standards and pull-up current sources for the SGL-encoder connector.

6.3 The Auxiliary Controller Bus (ACB)

The serial crate controller SGL encoder will allow one other auxiliary controller to use the CAMAC crate. It would do this by simply going ahead with a CAMAC cycle, so long as the ACL signal were not asserted. If it were asserted before the auxiliary controller reached S1 of its cycle, the logic would simply abandon the cycle and try again when ACL was removed. Remember that IEEE 583/EUR 4100 requires that no module should take irrevocable action before S1 is asserted. With multiple auxiliary controllers in a crate, one needs a mechanism for these controllers to arbitrate between themselves as to which one has access to the crate. Figure 6.1 shows the ACB schematically. It will be noticed that there is a new signal that is not on the SGL-encoder connector: Request (RQ). The lines specific to the SGL-encoder function, however, are absent from the ACB.

Let us now look at how the bus works. First, every auxiliary controller has access to the L lines, called AL lines in this figure. Thus auxiliary controllers can respond to LAMs. Also present are the Encoded N (EN) lines so that auxiliary controllers can assert any one N line in the crate at a time. However, they cannot address the internal registers of the serial crate controller because these are not accessed via a dataway cycle. This is a weakness because on powering up a crate, the auxiliary controller has no way of communicating with the serial driver and its computer until that driver initializes the crate and enables demands. A similar situation exists with the A-2 controller, of which more later.

So, let us assume that one of the auxiliary controllers needs to generate a dataway cycle. It then pulls down the request line to a Logic "1". It will be seen that in each controller, the request line on the bus is also brought

Table 6.2

SGL-ENCODER CONNECTOR: SIGNAL STANDARDS AND PULL-UP CURRENT SOURCES FOR ALL SIGNALS OTHER THAN CODED-N

	Signals from SCC	Signals to SCC
SIGNAL STANDARDS AT CONNECTOR		
Line in "1" state at +0.5 V		
Minimum current sinking capability (current drawn from line by unit generating the signal)	3.2 mA[a] L signals 6.4 mA[a] other signals (from line by SCC)	16 mA[a] (from line by encoder)
Line in "1" state at +0.5 V		
Maximum load current (current fed into line by unit receiving the signal)	3.2 mA[a] each unit (max 6.4 mA[a]) (into line by encoder)	3.2 mA (into line by SCC)
Line in "0" state at +3.5 V		
Minimum pull-up capability (current fed into line by the SCC)	2.3 mA (for L-signals this is common to encoder and dataway)	200 μA
Line in "0" state at +3.5 V		
Maximum current drawn from line by the encoder	200 μA	200 μA
PULL-UP CURRENT SOURCES IN SCC		
Internal pull-up current source (Ip) at +0.5 V	6.0 mA $\leq$ Ip $\leq$ 9.6 mA[a]	0.8 mA $\leq$ Ip $\leq$ 1.6 mA
Internal pull-up current source (Ip) at +3.5 V	2.5 mA $\leq$ Ip[a]	300 μA $\leq$ Ip
PULL-UP CURRENT SOURCES IN SGL ENCODER		(OUTPUTS ONLY)
Internal pull-up current source (Ip) at +0.5 V		6.0 mA $\leq$ Ip $\leq$ 9.6 mA[a]
Internal pull-up current source (Ip) at +3.5 V		2.5 mA $\leq$ Ip[a]

[a]Values derived from EUR 4100e/TID 25875.

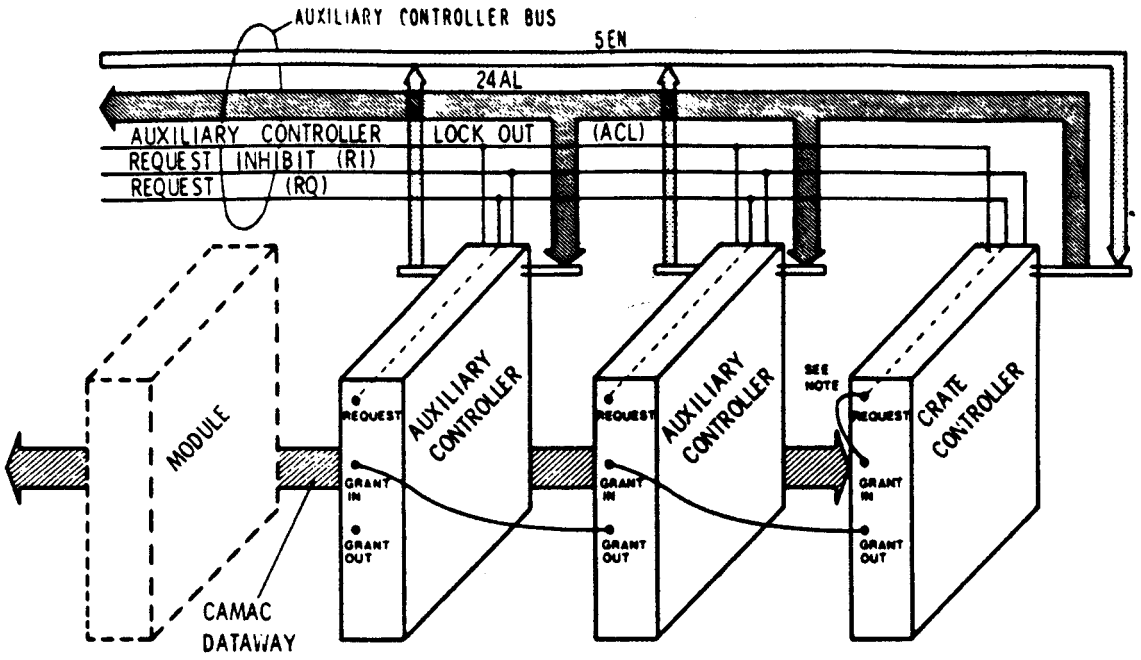


Fig. 6.1.
Multiple controllers in a CAMAC crate.

forward to the front panel. On one controller, the one at the highest priority, this request line is connected to Grant In and this is daisy-chained from Grant Out to next Grant In and so on. The controller that has the request assumes control of the CAMAC crate and does not pass on the Grant In to Grant Out when it receives Grant In. It then asserts Request Inhibit and performs the dataway cycle while being ready to abort the cycle if ACL goes true before S1. Any other controller removes any request from the RQ line while Request Inhibit or ACL is true. It must do this in less than 50 ns. So now we can have as many auxiliary controllers in a crate as we can fit in or afford, and they will not destructively interfere with each other.

Figure 6.2 shows this timing for the case when ACL does not interfere with the cycle. Note that the auxiliary controller CAMAC cycle is lengthened over that defined in IEEE 583/EUR 4100, with a minimum time between the start of the cycle and S1 or 500 ns. This is to allow for the delay introduced by the N-decoder, defined to be less than 100 ns. Thus a minimum auxiliary controller dataway cycle is 1·1 μs. Figure 6.3 illustrates what happens if the crate

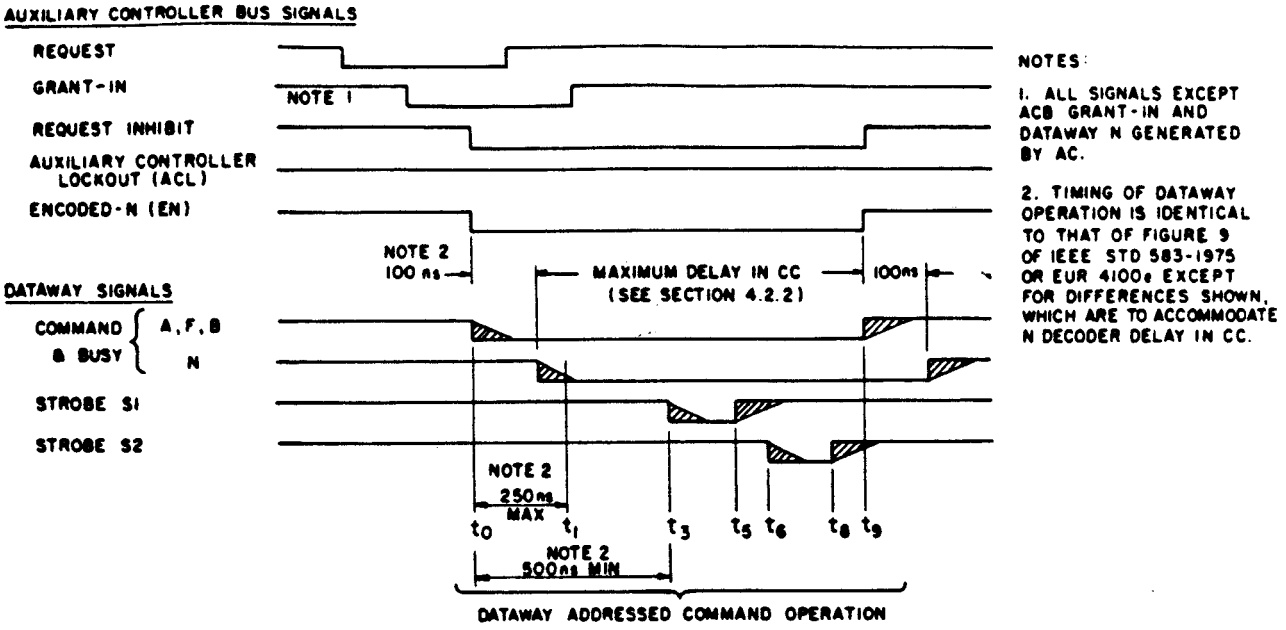


Fig. 6.2.
Sequence of signals for an auxilliary controller to gain control of the crate for an addressed command operation.

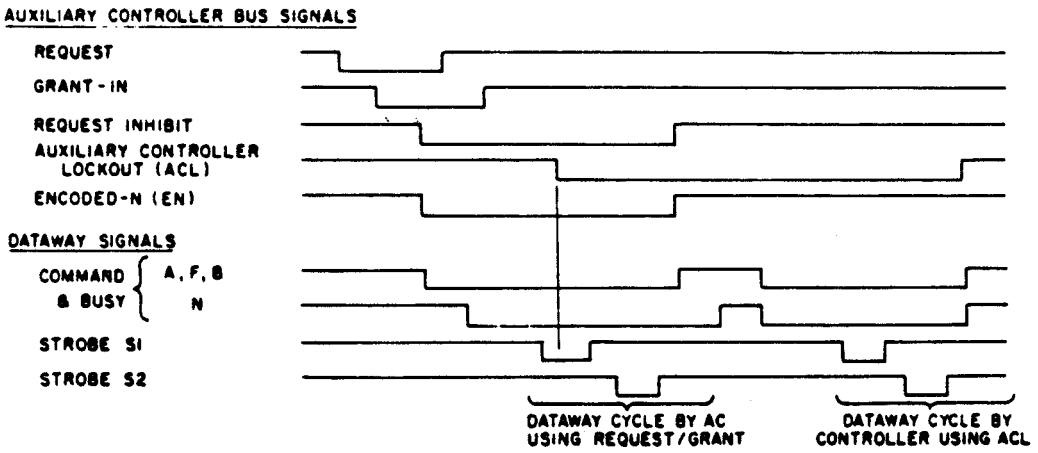


Fig. 6.3.
Example of a sequence in which auxiliary controller lockout (ACL) signal is generated too late to cause aborting of dataway cycle started by auxiliary controller using request grant.

controller asserts ACL after the auxiliary controller asserts S1. In this case the auxiliary controller completes the cycle, and the crate controller starts its dataway cycle after that of the auxiliary controller, by definition.

Figure 6.4 illustrates what happens if ACL is asserted before S1 of the auxiliary controller cycle. Here the cycle is aborted so as not to destructively interfere with the crate controller's cycle.

Clearly, only one controller in the crate can assume control of the crate using the ACL signal. A controller generating ACL must not start its dataway cycle until 200 ns has elapsed, and it has received Request Inhibit = "0". An exception to this rule is the serial crate controller that has a minimum delay of 800 ns between ACL and the dataway cycle and does not take into account Request Inhibit.

6.4 The ACB Connector and Signal Standards

The ACB connector is a 3M ribbon-cable connector with 40 contacts. When used with a serial crate controller, one needs a converter cable assembly that is available from manufacturers.

Table 6.3 gives the current signal standards and pull-up current sources for the ACB. The main difference from the dataway specifications is that the pull-up currents are less, and each module receiving signals can have more than one TTL load (1.6 mA each) on the line, depending on the line. The last line specifies the location of the pull-up. When auxiliary controllers are used with serial crate controllers, one should check for the request inhibit pull-up in the controller because older controllers did not have them, and thus a pull-up needs to be added.

Table 6.4 summarizes the pinning of the ACB connector.

IEEE 583/EUR 4100 specified the signal standards for the Q, R, and X lines on the dataway, assuming that only one controller received these. Thus the signal standards for auxiliary controllers on these lines are very tight and allow a maximum of 12 auxiliary controllers in a crate--less if other modules receive these signals. Table 6.5 gives the signal standards for these dataway lines.

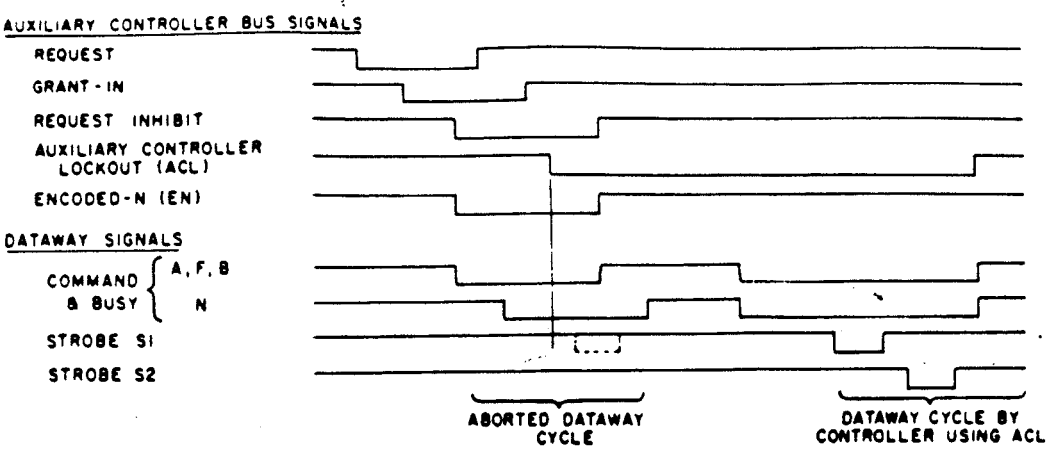


Fig. 6.4.
Example of sequence in which a dataway cycle started by auxiliary controller using Request/Grant, is caused to abort by ACL signal generated by another controller.

Table 6.3

CURRENT SIGNAL STANDARDS AND PULL-UP CURRENT SOURCES FOR THE AUXILIARY CONTROLLER BUS CONNECTOR AND ASSOCIATED FRONT PANEL CONNECTORS

Signal Standards at Connector	Auxiliary Control Lockout, Request Inhibit	Auxiliary LAM	Request Grant-In/Out	Encoded-N Condition Free
Line at "1" state at 0.5 V				
Minimum current sinking capability (current drawn from line by unit generating the signal)	For CC 6.4 mA; For AC 16.0 mA	3.2 mA	16.0 mA	16.0 mA
Line at "1" state at 0.5 V				
Maximum load current (current fed into line by unit receiving the signal)	0.4 mA per unit (6.4 mA maximum)	0.4 mA per unit (3.2 mA maximum)	12.8 mA	11.2 mA
Line at "0" state at 3.5 V				
(maximum current drawn from line by CC without sinking pull-up)	100 μA	100 μA	100 μA	100 μA
Line at "0" state at 3.5 V				
(minimum current fed into line by CC with sinking pull-up)	2.5 mA	2.5 mA	2.5 mA	2.5 mA
Location of pull-up for current, Ip, at 0.5 V:				
6 mA $\leq$ Ip $\leq$ 9.6 mA	CC	CC	Grant-in	CC

Table 6.4

CONTACT ASSIGNMENTS ON AUXILIARY CONTROLLER BUS CONNECTOR[a]

Contact	Usage	Contact	Usage
1	Ground (OV)	2	Encoded-N EN1
3	Encoded-N EN2	4	Encoded-N EN4
5	Encoded-N EN8	6	Encoded-N EN16
7	Ground (OV)	8	ACL
9	Ground (OV)	10	Conditionally Free
11	Ground (OV)	12	Request RQ
13	Ground (OV)	14	Request Inhibit RI
15	Ground (OV)	16	AL1
17	AL2	18	AL3
19	AL4	20	AL5
21	AL6	22	AL7
23	AL8	24	AL9
25	AL10	26	AL11
27	AL12	28	AL13
29	AL14	30	AL15
31	AL16	32	AL17
33	AL18	34	AL19
35	AL20	36	AL21
37	AL22	38	A123
39	AL24	40	Ground (OV)

[a]Contact 2 is across from Contact 1, Contact 4 is across from Contact 3, etc.

Table 6.5

SIGNAL STANDARDS FOR Q, R, AND X AT THE
AUXILIARY CONTROLLER DATAWAY CONNECTOR

Condition at Dataway Connection	Absolute Limit
Line in "1" state at +0.5 V (maximum current fed into line by auxiliary controller receiving signal)	0.4 mA
Line in "0" state at +3.5 V (maximum current drawn from line by each auxiliary controller)	100 μA

6.5 Type A-2 Parallel-Branch Crate Controller

The A-2 crate controller is a modification to the A-1 crate controller for the parallel highway, IEEE 596/EUR 4600. It operates in exactly the same way on the parallel branch and in that sense is quite interchangeable with the A-1 controller. The difference is that, in addition to the LAM-grader connector on the rear, there is an ACB connector. The A-2 controller has an internal switch to determine if it will use the ACL or request/grant mechanism for crate arbitration. Additions to the front panel are an indication of the state of this switch and the request, grant-in and grant-out connectors (Lemos). Thus an A-2 controller cannot assume control of the crate but must use one of the ACB mechanisms to avoid conflicts. An additional requirement is that the BRW lines must only be connected to the crate R and W lines when the A-2 controller is on-line, addressed, and in control of the crate. Similar gating restrictions apply to the other branch and crate lines: BA and A, BF and F, BQ and Q, BX and X.

Clear and Initialize (C&Z) operations likewise can be generated only when the A-2 controller is in control of the crate, and thus there are circumstances when auxiliary controller activity could prevent the controller from generating an initialize signal in the crate in response to a branch initialize signal (BZ).

The A-2 controller initiates the gaining control of the crate when its BCR signal goes to "1" and does not release control until BCR="0"; for efficient block transfers, the branch driver should maintain BCR="1".
Thus it will be seen that IEEE 675/EUR 6500 extends the CAMAC crate from a single controller crate to a multiple controller crate for both serial and parallel systems.

7. SYSTEM ASPECTS OF LAMs

7.1 Introduction

So far we have dealt with the handling of LAMs at the module level and within the type A-1 and L-2 controllers. This section looks at the LAM handling needs, once one extends beyond a single program owning the CAMAC system attached to the computer. The problem arises because the LAM structure of CAMAC does not allow the individual LAMs to be handed round to the interested parties by hardware, even if the operating system did. Thus, let us first look at the needs of any system's LAM handler and then look at some examples.

7.2 System Needs in LAM Handling

First, we must make some assumptions. We assume that we are running at least a multitasking system, if not a multiprogramming system. The difference between these is that, in the former, all the tasks contribute to the same overall job to be done; in the latter, the programs are, in general, completely independent of each other in that respect. The second assumption is that we are not able to operate on a module without knowing what it is. The third assumption is that we do not wish to mask off further LAMs for longer than is absolutely necessary. Finally, we assume that the tasks that handle the LAMs are not able to run at any priority that would mask out any computer interrupts.

The system code that has to field LAMs thus needs to

1. Accept requests from tasks that they be notified when a LAM (or group of LAMs) occurs.
2. Enable that LAM through the hardware and software.

3. When a LAM occurs, identify it, and notify the interested task.
4. Mask off the individual LAM so that it does not cause further interrupts until the interested task clears the LAM in the module.
5. Accept a request from the interested task to re-enable the LAM.

Because the system needs to mask individual lines in a module-design independent to achieve Point 4 above, it needs a mask register in each "crate controller." This indicates the need for a LAM grader in each crate, be it serial or parallel, to provide these facilities. Without a LAM grader, the system will need to disable all LAMs from the crate with the LAM and rely on the interested task saying either "re-enable the LAM" or "I've cleared the LAM but do not re-enable it for me." Let us hope here that the task remembers to reply and does not abort between being notified of the LAM and replying!

What can be involved in enabling a LAM through the system? Figure 7.1 shows a CAMAC system connected to a computer. Indicated are all points where software might have to enable the LAM or set a bit. The two points in the CAMAC plug-in in question are clearly the responsibility of the code that is specific to that module, but the remaining six possibilities are best taken care of by some system code, indeed often must be taken care of by such code.

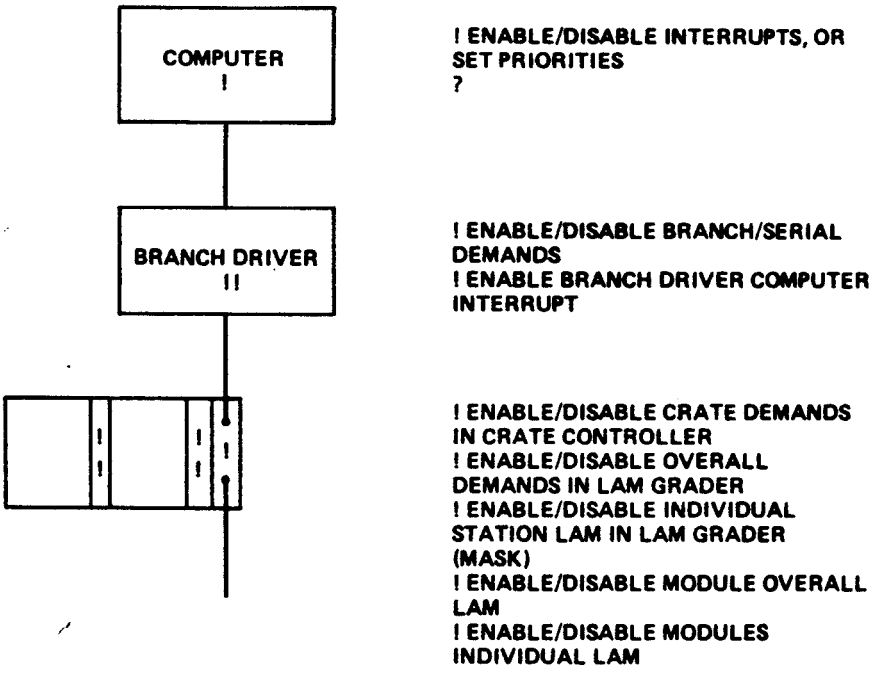


Fig. 7.1.
The path of a LAM through the system.

Although this enabling process does look complicated and involved, in practice (once it is fought through and coded) it proves to be no problem from then on. It is simply a question of being aware of the problem and checking the hardware manuals.

7.3 Priorities for LAMs

One thing that CAMAC does not provide inherently in the hardware is the ability to have LAMs at different priorities. In general, LAMs are either enabled or not enabled. The only priority that does exist is in the LAM sorting once an interrupt has occurred. This can be either hardware or software, but if two LAMs occur almost simultaneously, then clearly one is chosen for service first. Once the choice is made, and the service of that LAM is started, how responsive can we make the system to a new LAM that we deem to be of higher priority?

The shortest response time possible is the time that it takes to carry out the minimum service of the previous LAM. This service can be the minimum necessary to clear the LAM in the hardware and queue the further processing of the LAM by code that can run with interrupts enabled. Thus the initial response time is determined by the minimum-service time; the full response time is set by the scheduling of the processing of queued interrupts. This latter can easily include the priorities needed, depending on the computer type and operating system used.

In practice, all this involves overhead both in software and hardware; with the present low price of processors, it is cheaper to design a system with sufficient spare capacity so that priority problems do not arise. Certainly, one should look most carefully at any aspect where there is an absolute crisis time for a response! Can this be removed onto hardware or a small dedicated processor? In summary, a true priority structure for interrupts is only of use when the system has a severe real-time crisis, and the corresponding interrupt routine must be started the moment the hardware interrupt occurs.

7.4 Timing

Timing of interrupt response is very dependent on the architecture of the computer system used. Some systems leave all the saving of registers, etc.,

to the software so that there can easily be 20-100 μs of overhead before the first instruction related to the interrupt is executed; PDP-11s come in this class. Other systems have instructions purposely designed to make this overhead smaller, but still software is involved. Here the overhead can be reduced easily to ∿10 μs.

Finally, the most responsive computer design will have a complete set of registers for each interrupt level so that the hardware switches to interrupt processing in approximately a microsecond.

However, this is not the whole story--the winner of a race is not judged by who leaves the starting line first! In some systems (where, for example, the processor is simply histogramming data on interrupts) this kind of architecture is most important, In other systems (where each interrupt requires, on average, rather more processing) this speed of context switch may well be dwarfed by the different processing speed of another processor or even by a better operating system. This latter point arises because at some point the interrupt code needs to communicate with the rest of the system, and the formalization of this can easily involve times of 0.2 to 4 ms! But this is getting a little far from CAMAC. The important point to make is that no part of a system should be looked at in isolation. Rather it should be considered in the context, both of the rest of the system and the requirements on the system.

7.5 Graded LAMs or Station Numbers?

As we have already seen, the facilities exist in the CAMAC system to patch LAMs to so-called Graded LAMs such that the interrupting station number is lost unless the patch table is referred to. In some special systems, this patching can greatly speed up processing. Remember, one can also logically "OR" LAMs. This facility has most use in event-based data-acquisition systems, where the LAM is a logical LAM that triggers the data-acquisition sequence. In most other systems, a LAM from a particular station triggers action on that station and others; therefore, having the hardware present a station number rather than a Graded LAM is far more appropriate. In this case, if some LAM "OR"ing is required, it can be simply achieved in software. Thus, although facilities exist that can grade LAMs, one should think carefully before invoking them.

8. THE CAMAC-COMPUTER COUPLER

8.1 Introduction

In this last section I want to examine how one should interface a CAMAC system to a computer and to ask the question if there are good and bad interfaces. Finally we will look at the various DMA modes defined in the "Block Transfers for CAMAC."

8.2 The Address Space Problem

Few computers provide an I/O address space of greater than 12 bits or 4096. Is this sufficient for CAMAC? Figure 8.1 shows the address space requirements, and it will be seen that, without specifying a branch number at all, at least 20 bits of address space are required. Clearly this does not fit, and so a solution must be found. The problem arises not because CAMAC is a device to be interfaced to an I/O system, but it is itself an I/O system.

This problem is solved by having one or more registers in the CAMAC coupler that store the extra bits, leaving only a few to be defined by the I/O address accessed. In some systems it might be preferable to access CAMAC completely through registers; it really depends on the I/O structure of the computer concerned and on the application for which the coupler is designed, but more of that later.

8.3 The Data Problem

Although technology is changing rapidly, it is still mostly 16-bit computers by which CAMAC is driven. When CAMAC was first specified, this trend was not at all clear, and there existed a number of 24-bit machines. However, the data width of CAMAC is well-tuned to the precision with which one can make physical measurements. Although most modules in fact do not use the upper 8 bits of the data path, for those that do a register can be used in the computer coupler for the high 8 bits to overcome the deficiency.

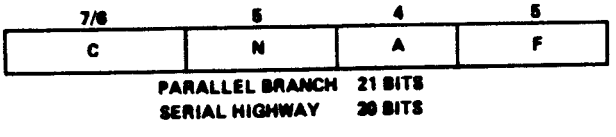


Fig. 8.1.
CAMAC address space requirements.

So, let us see what a CAMAC write operation might look like. Figure 8.2 shows such an operation. This assumes a 16-bit computer and also assumes that one only uses registers and not address space to address CAMAC. Thus, this coupler will take only 4 I/O addresses on the computer's I/O bus. The actual order of the operations can differ, but in Fig. 8.2 the first I/O operation writes the extra address bits to a register in the coupler. The second operation then writes the 8 most significant data bits, the third operation the remaining data bits, and last the remainder of the CAMAC address is written. This last write can trigger the CAMAC cycle on the branch and the resultant Q and X can be read back from a status register, which is not shown.

Clearly, if 16-bit data is to be written, the writing of the data high register can be omitted. Also, if the extra address bits do not change between successive operations, the writing of that register need be done but once.

Where a computer uses a particular part of its memory address space to access the I/O system, then the write data low operation can be done not to a single address, but to one of a range of addresses, the exact address determining the final CAMAC address bits. Thus, using this technique, the full CAMAC write operation is reduced to three I/O transfers. This is the technique which is often used on PDP-11s and LSI-11s.

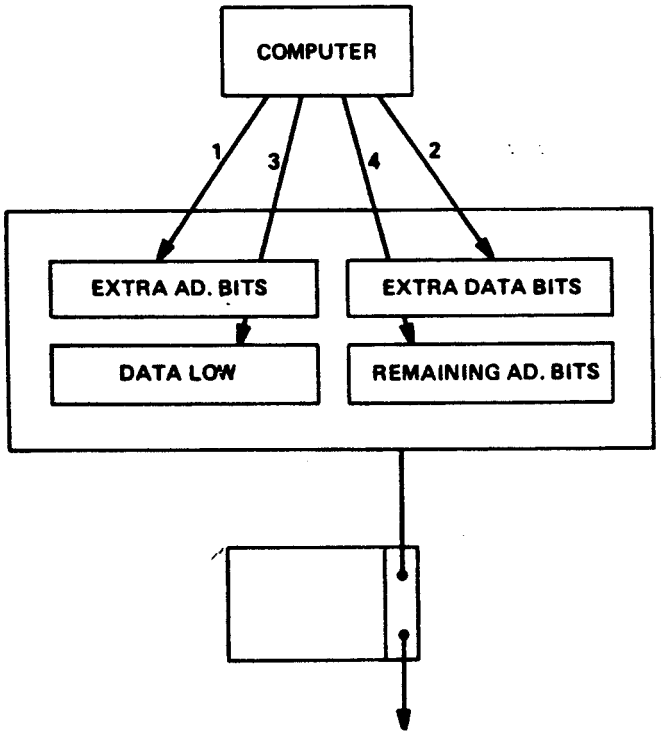


Fig. 8.2.
The all-register CAMAC-computer coupler.

8.4 Solutions to the Address Space Problem

As was pointed out above, often one does not need to write the extra CAMAC address bits, thus saving time and trouble. However, one needs to choose which of the CAMAC address bits are stored in this register to minimize the need to rewrite it. This choice of bits is very dependent on the application, but two main choices do arise.

Once an event occurs in a high-energy physics experiment, the software needs to do the same read function, RD1, at many different subaddresses of many different modules in several crates. Figure 8.3 shows the high-energy physicists' view of the CAMAC address, which reflects the fact that when he is most interested in efficiency the function never varies, the crate number varies but slowly, the station number faster, and the subaddress fastest of all. Thus, the designer of a CAMAC coupler destined for high-energy physics use would choose the bits for the extra address register, starting from the left of Fig. 8.3. In this way it is quite possible that this register is set up only once and then quite forgotten.

The other view of the CAMAC address is shown in Fig. 8.4. This view reflects the fact that most other systems address one module at a time and then carry out several operations on that one module. Thus, such systems are at an advantage if the crate and station is defined by a register, and the subaddress and function are defined either by another register or by the I/O address used. At this point the actual order of the subaddress and function is not too important.

There are some systems that have several of the same kind of CAMAC modules as a multichannel system. In this case, consider the advantage of the crate and station being completely defined only by the extra address register. This need then be set up once, and then common code can access the module pointed to without having to concern itself about the actual CAMAC slot being addressed!

Fig. 8.3.
The high-energy physicists' view of the CAMAC address.

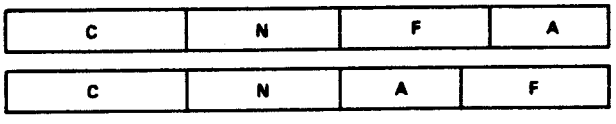


Fig. 8.4.
The other views of the CAMAC address.

8.5 The Interrupt Problem

This problem is not concerned with getting the interrupt to the computer so much as what to do when an interrupt occurs during a CAMAC operation. You will recall that the software needs to save registers so that they can be later restored and the interrupted program resumed without any error. As we have seen, a CAMAC operation takes more than one I/O instruction, so what happens if the interrupt occurs between these instructions and the interrupt code itself uses the CAMAC interface?

There are a number of solutions

1. Avoid the problem by shutting off interrupts during CAMAC cycles.
2. Use a computer-CAMAC coupler that has all registers as read/write registers so that the interrupt response code can save and restore these registers as well.
3. Inhibit just CAMAC interrupts during a CAMAC operation, which can be done in hardware by the computer-CAMAC coupler.

Before we examine these in more detail, let us review the I/O operations of a CAMAC write cycle again.

a. Write extra address bits
b. Write data high
c. Write data low
d. Write remaining address bits
e. Read Q and/or X response.

Any scheme that we choose must take account of the fact that any or all of steps a, b, and e can be absent in any particular operation.

Solution 1 above avoids the problem entirely but does add overhead to each CAMAC operation. Also to be considered is that, in some operations, Q or X will be tested; therefore, the instruction to re-enable interrupts must be placed after the reading of these responses.

Solution 2 above allows interrupts to occur at any time during a CAMAC operation, but it does add overhead to each interrupt because the registers must be read and restored. In the best circumstances, where c and d are combined into a single I/O operation and where Q and X are packed with data high, this means two registers must be saved and restored.

Solution 3 will avoid all software overhead but fails to provide the solution to three problems. It works by the computer-CAMAC coupler recognizing the

first and last I/O operation of a CAMAC sequence. For some computers, it is possible that an interrupt occurs at a point such that the coupler cannot stop it going through to the computer in time. In this case, the interrupt routine needs to test for this condition, and to return directly if it occurs. The next problem is to determine what is the last operation. Once again the standardized function codes help, but will the software test for Q? If it does, then the CAMAC operation is not over until this is done, but if Q is not tested, then the operation is complete. To overcome this, the interrupt routine must save and restore the Q and X responses.

However, the hardware solution does not overcome the problem in a multitasking system where the processor is switched between different tasks as a result of a non-CAMAC interrupt, from the clock for example. In this case, if a commercial operating system is used, then there is no solution except to either add a CAMAC driver to the system or modify the system such that CAMAC coupler registers are saved and restored along with the other registers so treated by the operating system. Neither solution is a trivial task.

One way to avoid these problems would be to provide several sets of registers in the CAMAC coupler and to allocate each set to a single task. As far as I know this has not ever been done.

8.6 Block Transfers

Block transfers are hardware-generated transfers involving more than one CAMAC cycle. They are designed to transfer medium to large amounts of data between the CAMAC system and the computer memory, and to do it considerably faster than can be done by software alone. A large number of different block transfers are defined; they have many uses, but typically only one or two types of block transfer might be used in a single system.

Three orthogonal considerations describe a block transfer mode. These are

1. CAMAC Address Sequencing. One can either transfer from or to a single CAMAC address, from a given array of addresses, or scan consecutive CAMAC addresses.
2. Synchronizing Source. The synchronizing source determines when the next operation will take place. The possibilities are controller synchronized (that is, as fast as one can go), Q response synchronized, LAM synchronized, and pseudo-LAM synchronized (that is, synchronized

by a signal connected directly between the module and the block transfer controller).

3. Operation Termination. The block transfer can be terminated by a channel word count, a final address reached, Q=0 response (with this data either being valid or invalid), on a LAM signal, or on a pseudo LAM signal.

All possible combinations add up to rather a large number of different modes, but happily they boil down to but a few common block transfer modes.

Table 8.1 lists all these possibilities and the single letters that represent them. As an example, a mode that transfers data between the computer memory and a single CAMAC address, at full speed and for a number of words set by a word count, would be UCC mode. In what follows I propose to cover some of these modes but by no means all combinations; I count 96 combinations!

8.6.1 Stop Mode (UCS and UCW). This is a uni-address block transfer, controller synchronized and terminated with a Q response. In this mode, the CAMAC module must generate a Q=1 for each valid transfer and a Q=0 for any further operations after the end-of-block condition has been encountered. Thus, transfers continue at full controller speed until a Q=0 response is received. If this last transfer contains valid data, the mode is UCW mode otherwise the mode is UCS mode. The block transfer controller will then have recorded the number of data words transferred and will usually have been set with a maximum count to protect the memory of the computer.

8.6.2 Address Scan Mode (ACA). The object of this mode is to access successive subaddresses through a system, reading out each one in turn. This mode typically is used in a high-energy physics or in other event-based experiments. The algorithm followed is to perform the operation at the start address and, if a Q=1 is received, increment the subaddress for the next operation. If Q=0 is received, then the subaddress is set to zero and the station number is increased. Q=0 indicates that there is no data at that subaddress, either because the register does not exist or because it has no valid data. Clearly, if A(15) is read, it is assumed to be the last subaddress; the subaddress is then set back to zero and the station number incremented.

A restriction of this mode is that the data must be ready to be read out because the Q response is already used to control the address incrementation.

Table 8.1

BLOCK TRANSFER MODE DESCRIPTOR

First Letter	CAMAC Address Sequencing
U	Uniaddress
M	Multiaddress (calculated or given array)
A	Address scan
E	Extended address scan

Second Letter	Synchronizing Source
C	Controller
Q	Q-response
L	LAM signal
D	Pseudo LAM (direct module-controller connection)

Third Letter	Operation Termination
C	Channel word count
A	Terminal address reached
S	Q=0 on last valid transfer +1
W	Q=0 on last valid transfer
L	LAM signal
D	Pseudo LAM signal (direction module-controller connection)

It is also quite possible to have an X=0 response because of a missing module, and a module with an error of some kind can also disturb the operation of this block transfer. Thus, systems that use the address-scan mode need to be designed with care and, indeed, should have known data words read out at certain points as a check.

The scan is terminated when the final address is reached. Some hardware has the ability to reset the station number to 1 and to increment the crate number when a predetermined station is reached.

8.6.3 Repeat Mode (UQC). This is a uniaddress block transfer, Q-response synchronized, controller terminated. This mode can be used when the module cannot accept or give data at the full CAMAC rate. The controller continually reads or writes data from or to the module, and the module returns Q=1 for successful transfers and Q=0 for all others. The controller will continually tie up the CAMAC system until the block transfer is complete. Thus, it is not recommended for slow CAMAC modules unless system considerations say that the CAMAC system has no other task at this time. Care must be taken because module design provides some difficulties in dealing with the CAMAC cycle that occurs just as the data becomes ready.

8.6.4 Classic DMA (UCC). This mode is at a single CAMAC address at full controller speed for a predetermined number of transfers. As such, it represents a classic DMA I/O operation to read or write a block of data.

8.6.5 LAM Synchronized Stop Mode (ULS and ULW). In this mode, the LAM from the addressed module synchronizes each transfer from or to the module, and the module signals the end of the block transfer with a Q=0 response. As before, ULS mode signals Q=0 on the first invalid transfer, ULW on the last valid transfer. Thus, for ULS mode, the module needs to generate an "end of block" LAM and then return the Q=0 when the controller responds with a CAMAC transfer. As before, reaching a predetermined number of transfers also will terminate the block transfer.

This is a good mode to use when reading or writing to a slow device, such as a terminal; the CAMAC system is only taken by valid transfers because of the use of the LAM from the module to signal either "data ready" or "ready for data."

8.6.6 Direct Synchronized Stop Mode (UDS and UDW). These modes are exactly similar to the preceding modes except that a direct synchronization signal is wired between the module and the DMA controller. One difference is that by avoiding the LAM handling mechanism, these two modes can, in some systems, be somewhat faster.

8.6.7 Multidevice Action Mode (MCA). This mode allows block transfers to a random array of CAMAC registers. This array might be set up as an array of

addresses for the controller or as a regular pattern defined so that the controller can calculate the next address. One advantage of this mode is that no special module features are required.

8.7 Requirements on Module Design

Clearly, for some of the above modes to work, the Q response has to be defined and engineered correctly. In particular, for many transfer modes Q=1 means data accepted or data valid. In other modes, Q=0 can mean data valid but it is the last valid transfer.

As regards the LAM handling in a module, it is convenient for DMA operations if the LAM, or particular internal LAMs, can be made available to the DMA controller directly. One scheme is to be able to connect the relevant LAMs to the bussed patch lines P4 and P5. If the DMA controller is then sitting in the same crate, it can take the synchronization signals directly from these lines. Another solution is to bring these signals to a front panel connector from where it can be connected directly to the DMA controller.

As can be seen when designing modules, thought has to be given to their suitability for use with a DMA controller. A further point to make is that DMA synchronization is a weak point in the CAMAC specifications, particularly at the highway and branch levels. A lot can be done, but this area would benefit from more standardization.

9. CONCLUSION

Although it is now 15 years since the design of CAMAC was started, CAMAC is still in the growth side of its life cycle. The reasons for this are mainly the large investment that has been made in CAMAC designs and systems by manufacturers and users. Thus it has continued to adapt to changes in both technology and demand. These facts have given new users confidence, both that it is a working standard and that it is a lasting one. This last point is most important because it provides confidence that one will be able to extend the system over the years ahead.

I trust that this Primer has given a basic grounding in CAMAC and has indicated some of the flexibility within the CAMAC specifications. There are many more interesting adaptions of CAMAC to individual systems, and insights into these must be gained by reading the literature of the subject (see the CAMAC Bibliography Chapter 1) and by actually working with CAMAC. With imagination most systems can be built from commercially available equipment.

*U.S. GOVERNMENT PRINTING OFFICE: 1986-0-676-034/40033