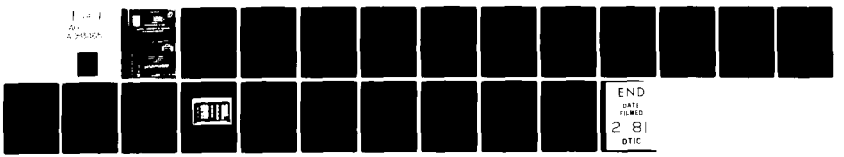
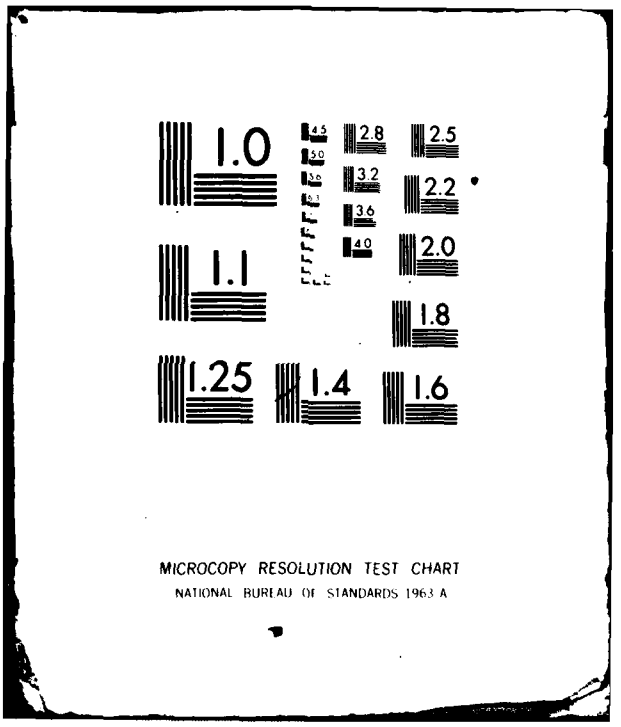
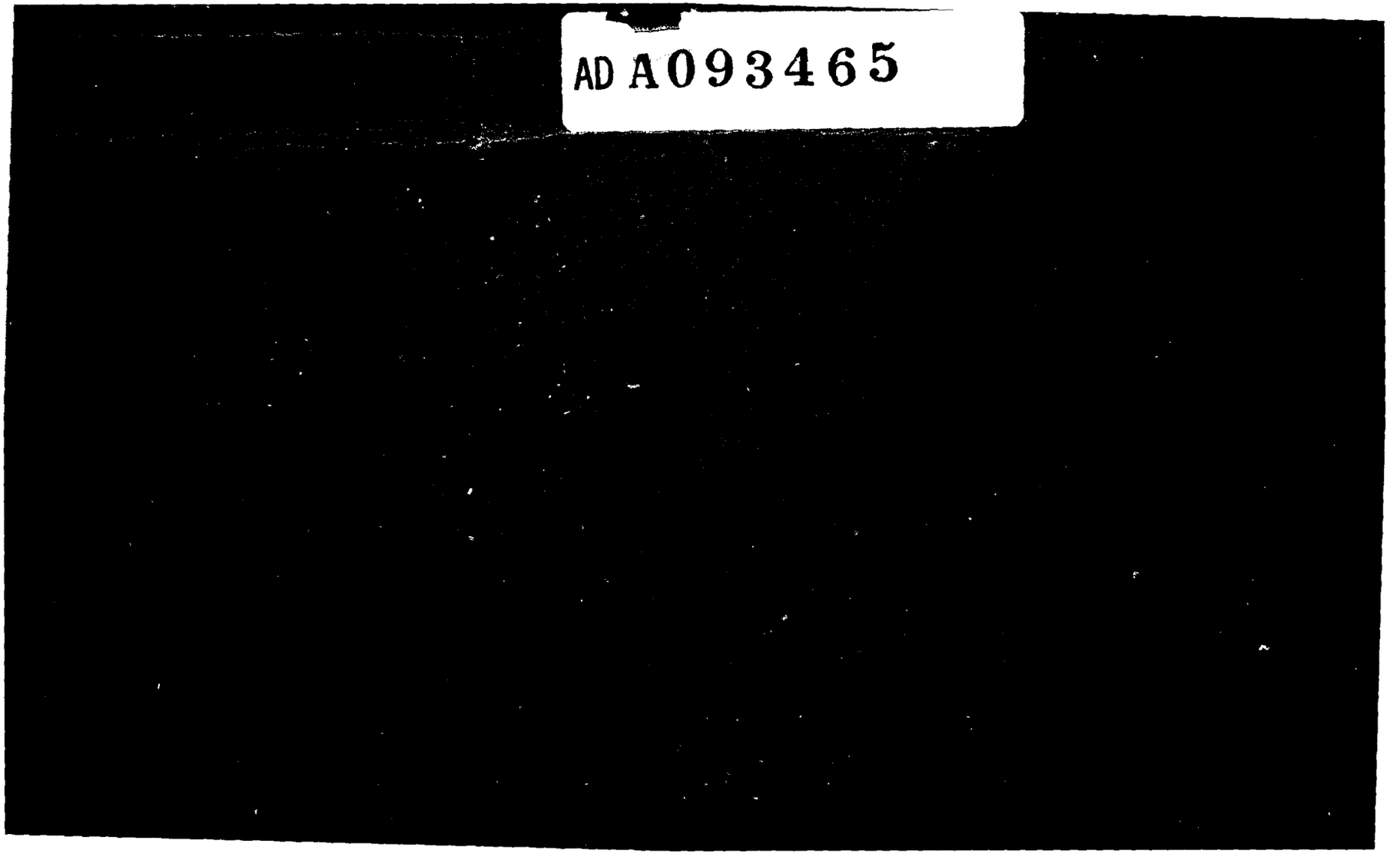
AD-A093 465 ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) F/G 17/2
THE USE OF THE IEEE-488 BUS AND THE MASCOT PHILOSOPHY IN AUTOMA--ETC(U)
SEP 80 M O'BRIEN
UNCLASSIFIED RSRE-MEMO-3285 DRIC-BR-76468 NL



END
DATE FILMED
2 81
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A



ROYAL SIGNALS AND RADAR ESTABLISHMENT

(9) Memorandum [redacted]

Title: (6) THE USE OF THE IEEE-488 BUS AND THE MASCOT PHILOSOPHY IN AUTOMATIC TEST EQUIPMENT.

Author: (10) M.O'Brien

Date: (11) Sep[redacted]80

(12) 23

(14) RSRE-MEMO-3285

SUMMARY

This memorandum describes the results of research undertaken to investigate the suitability of the IEEE-488 Bus as a standard ATE bus and the Modular Approach to Software Construction Operation and Test (MASCOT) as a standard software technique in Automatic Test Equipment.

R-1

409929

RSRE MEMORANDUM No 3285

THE USE OF THE IEEE-488 BUS AND THE MASCOT PHILOSOPHY IN AUTOMATIC TEST EQUIPMENT

M O'Brien

CONTENTS

1 INTRODUCTION

This memorandum describes the results of research work which was directed towards assessing the usefulness of the IEEE-488 Bus[1] and the Modular Approach to Software Construction and Test (MASCOT[2]) philosophy, in Automatic Test Equipment (ATE).

It was decided that to bring about the aims of the research work a simple experimental ATE would be constructed. This involved designing and building an Intel compatible IEEE-488 Bus controller and deriving a basic MASCOT system. Once the experimental ATE was operational then various tests would be performed to determine the value of the IEEE-488 Bus and MASCOT in ATE.

2 SUMMARY OF THE EXPERIMENTAL ATE

As the aim of this work was to investigate the value of applying the IEEE-488 Bus, (for Bus details see Appendix A) and the MASCOT approach to ATE (for MASCOT details see Appendix B), it was clearly unrealistic to build a complete ATE system so a small experimental ATE to test a digital circuit was devised. The function of this simple ATE was to stimulate a digital Unit Under Test (UUT), collect the output responses, compare them with the responses expected from a good circuit and give either a 'pass' or 'fail' report concerning the UUT.

From the block diagram in Figure 1.1 it can be seen that the system consists of a microcomputer with an IEEE-488 controller interface, a pattern generator and a logic analyser. The input pins of the UUT are connected to the pattern generator. If required the pattern generator initially outputs an initialisation sequence, this is followed by a test pattern in parallel with a clock output. (The clock is used to inform the logic analyser that the data on its inputs are valid.) When the data have been acquired the logic analyser indicates this to the microcomputer via the IEEE-488 Bus. The acquired data is compared with previously collected data from a known working unit to give either a 'pass' or 'fail' report.

The programs were written in CORAL 66[3] and were developed on an Intel Microprocessor Development System (MDS). During development the MASCOT supervisor with trace facilities and a capability to initiate, stop and restart program segments, was used. After development the programs were reconfigured so that the test program would run on a target machine, in this case an Intel SBC 80/10 microcomputer system.

3 IMPLEMENTATION OF THE EXPERIMENTAL AUTOMATIC TEST EQUIPMENT

The implementation of the experimental ATE was broken down into 4 distinct phases. These were:

1 Design, construction and test of an IEEE-488 Bus controller interface compatible with the Intel Multibus.

2 Preparation of a library of basic software routines to test the controller interface.

3 Design and construction of an Intel 8085 based pattern generator.

4 MASCOT system design and test.

3.1 DESIGN, CONSTRUCTION AND TEST OF AN IEEE-488 BUS CONTROLLER INTERFACE

The IEEE-488 Bus controller interface was designed to operate with an Intel SBC 80/10 microcomputer as the target machine although much of the development was done using an Intel MDS. As both microcomputers use an Intel Multibus internal bus the IEEE-488 Bus controller interface was designed to interface with this internal bus. The IEEE-488 Bus controller which is described in more detail in Appendix A, was built, tested and from the prototype a printed circuit board layout was produced.

3.2 PREPARATION OF A LIBRARY OF SIMPLE SOFTWARE ROUTINES

A series of simple software routines was written to test the IEEE-488 Bus controller. These routines were written in Intel 8080 assembly language[4] and after assembly the object code modules were formatted by the ISIS[5] librarian facility and stored on disk.

Figure A5 contains a listing of the more important software routines needed to drive the IEEE-488 Bus controller. (Controller address selection is set by links on the board).

3.3 DESIGN AND CONSTRUCTION OF AN INTEL 8085 MICROPROCESSOR BASED PATTERN GENERATOR

A pseudo listener function was implemented using two 8-bit parallel input ports of an INTEL 8085 to monitor the state of the IEEE-488 Bus. (The device was a pseudo-listener because it did not create an active handshake.) The Pattern Generator was designed so that it responded to a trigger command after it had been addressed. On receipt of the trigger command a pattern, previously loaded into the 8085's EPROM, was used to stimulate the UUT.

3.4 MASCOT SYSTEM DESIGN AND TEST

An ACP diagram described the system and is reproduced in Figure 2. Activity LGASET sets the logic analyser, via the IEEE-488 bus to a state which was determined by interpreting the parameter received from the root activity AC1. Activity LGASET then triggered the pattern generator, and when a test was complete extracted data from the logic analyser. These data were then compared with the data stored in a pool LGADT which was obtained from a known good UUT and a result, good or faulty was given.

In order to check the flexibility of such a software scheme an additional IEEE-488 Bus compatible multi-function voltmeter was interfaced to the bus. Another activity DVMSET was conceived together with two additional channels to give bi-direction links to the activity AC1. Activity AC1 passed a parameter which was interpreted by DVMSET to obtain the range and function to be measured. Next DVMSET set the multi-function voltmeter to the required function and after a reading had been taken returned the measured value to activity AC1. Figure 1.2 shows the block diagram for the modified test system and Figure 2.2 is then related ACP diagram.

4 CONCLUSIONS

The limited research work undertaken indicated that the IEEE-488 Bus simplified many of the design and construction problems associated with conventional ATE systems. The task of expanding the number of devices connected to an IEEE-488 Bus structured ATE is relatively straightforward and by having more than one IEEE-488 Bus operated by a process controller, more than 15 instruments, the maximum for a single Bus, can be used. The research also showed that the IEEE-488 Bus could potentially be applied to limited digital testing if used with an intelligent instrument, such as a logic analyser. (It must however be appreciated that switching matrix problems were not addressed by this work.)

The IEEE-488 Bus runs at the rate dictated by the slowest device connected to it and so, although the maximum speed of the Bus is 1M Byte/second, the actual

rate will be much slower. As an example the logic analyser took over 3 minutes to dump the contents of its memories in ASCII form to the controller, a data transfer rate of only 5 bytes/second. (This was due to the internal processing performed by the logic analyser).

For large integrated digital test systems, the IEEE-488 Bus would create severe limitations due to its speed and the small number of parallel data lines.

The results of the work with the MASCOT approach demonstrated that some of the claimed advantages of this philosophy are indeed valid. For example, once the basic ACP diagram and the related software scheme are designed, the task may be divided and written by various programmers.

Originally it was thought that the MASCOT approach would use an excessive amount of the available memory and the development system (including the supervisor) did use 21K bytes out of the maximum of 64K bytes. However once development had been completed, removal of the supervisor saved 7K bytes of memory and deleting redundant code (used for diskette and other operations) saved another 5K of memory, so the system could actually be implemented in only 9K bytes. Out of these 9K bytes about 1K byte were required by the channels. If it is considered that in expanding the system, many previously written routines can be utilised, then by using MASCOT, even with a machine with limited memory address capability, such as an 8080 microprocessor, the overheads are not too great.

REFERENCES

1 IEEE Standard Digital Interface for Programmable Instruments. The Institute of Electrical and Electronic Engineers.

2 The Official MASCOT Handbook. MASCOT Suppliers Association.

3 Official Definition of CORAL 66. HMSO Publication.

4 Intel 8080/8085 Assembly Language Programming Manual. Intel Corporation.

5 ISIS-II Users Guide. Intel Corporation.

6 Intel Multibus Application Note. Intel Corporation.

7 8291 DATA Sheet. Intel Corporation.

8 8292 DATA Sheet. Intel Corporation.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the assistance given by members of T2 Division, T18 Section and D P Taylor in particular with this project.

BLOCK DIAGRAM OF EXPERIMENTAL DIGITAL TEST SYSTEM

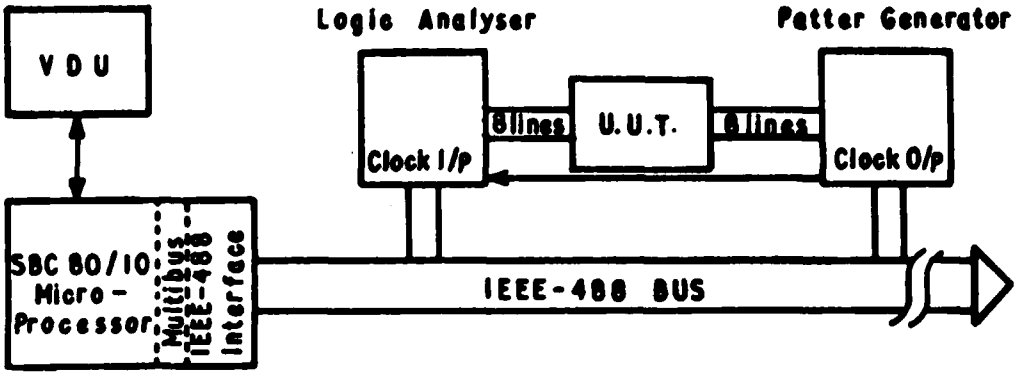


FIG. I.I. BLOCK DIAGRAM OF EXTENDED SYSTEM

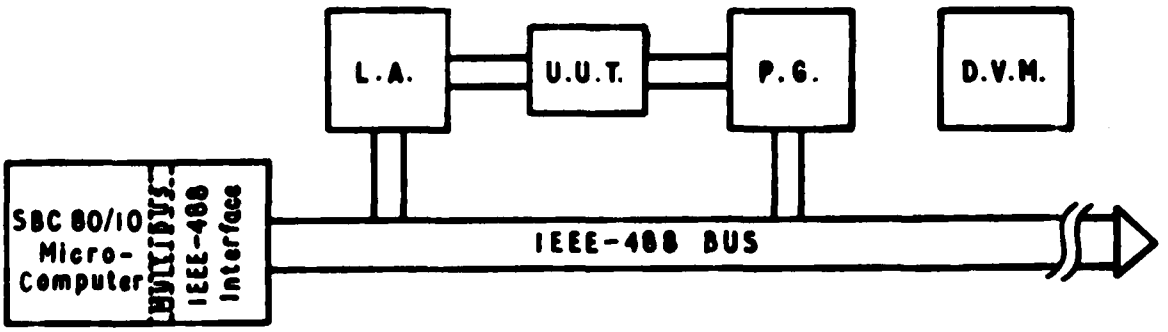


FIG. I.2.

MASCOT ACP DIAGRAM OF EXPERIMENT ATE

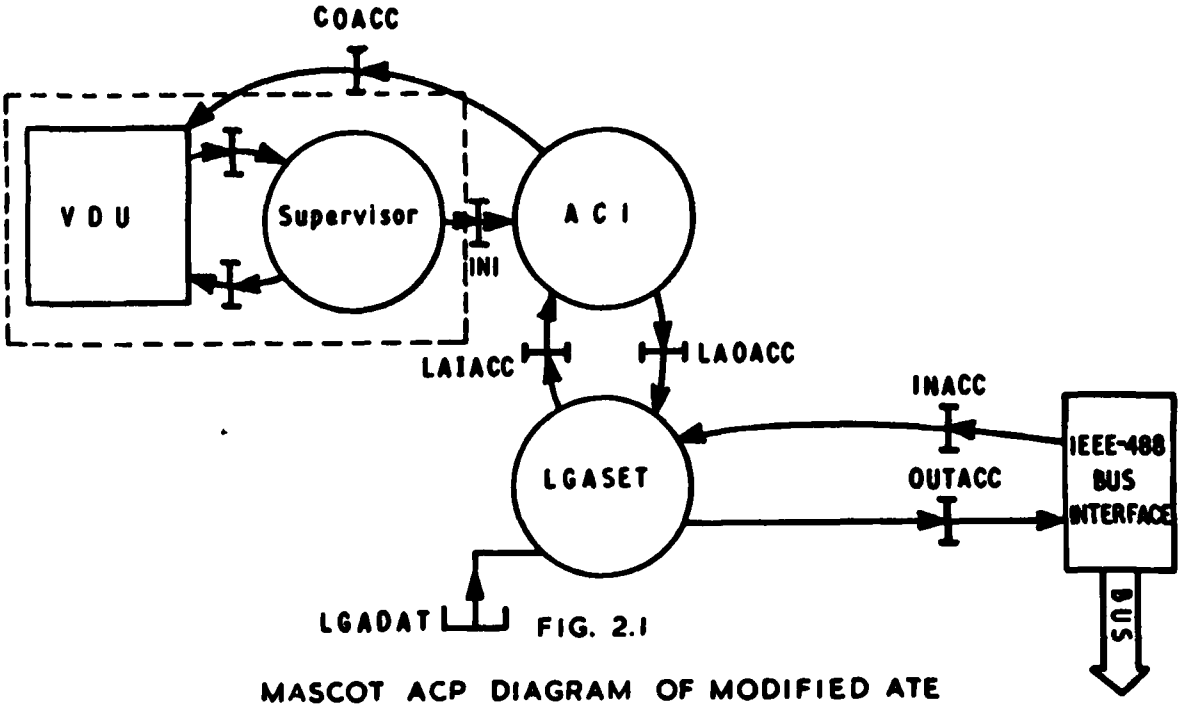


FIG. 2.1

MASCOT ACP DIAGRAM OF MODIFIED ATE

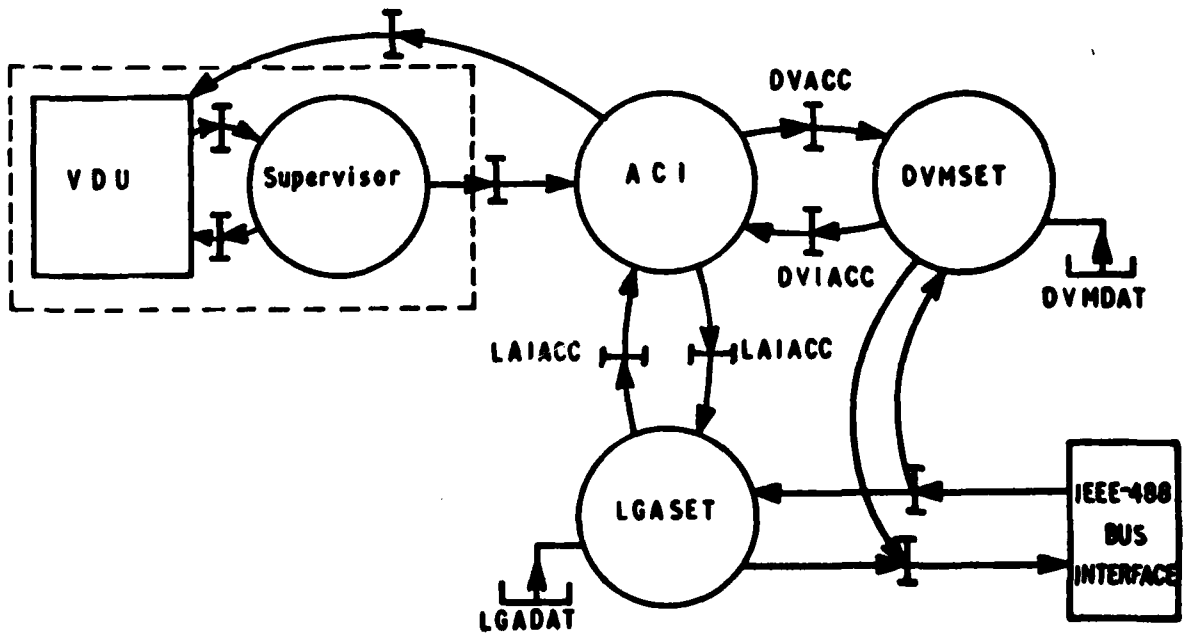


FIG. 2.2

APPENDIX 1

INTEL MULTIBUS COMPATIBLE IEEE-488 CONTROLLER INTERFACE

1 INTRODUCTION

As an IEEE-488 Bus controller was needed for the research work described in this memorandum, it was decided to construct an Intel Multibus Compatible IEEE-488 Bus Controller Interface that could be used in conjunction with an Intel SBC80/10 microcomputer or an Intel MDS.

1.1 THE IEEE-488 BUS

The IEEE-488 bus is known by a variety of names including the General Purpose Interface Bus (GPIB) and the Hewlett-Packard Interface Bus (HPIB). It was originally developed by Hewlett-Packard as an unambiguous method of interconnecting programmable instruments which are in close proximity, such as in a laboratory. When the standard was adopted by the IEEE it was changed slightly so as to be more rigorous and standard 488 was thus developed.

The bus allows up to fifteen instruments to be directly connected together using a standard connector and each instrument to be programmed independantly of the others. By using an IEEE-488 Bus controller, information can be exchanged between the controller and instrument or even between instruments. Information is transferred on an 8 bit parallel data bus under the sequential control of a three wire interlocked handshake. By this means, no step in the sequence can be initiated until the preceding step has been completed. The independent control at the instruments connected to the bus is achieved by using 5 lines, known as the management bus. Thus the IEEE-488 bus consists of 16 lines, the groupings being shown schematically in Figure A1.

2 IMPLEMENTATION OF IEEE-488 BUS CONTROLLER

The IEEE-488 Bus Controller Interface was designed to use two large scale integrated (LSI) circuits which implement all the talker-listener interface functions and all the controller functions specified in the IEEE standard.

The block diagram of the circuit is shown in Figure A2 and a brief functional description of each block is given below. The circuit diagram of the interface unit is shown in Figure A3. Figure A4 shows the layout of the prototype interface unit.

2.1 ADDRESS DECODERS

The address decoding is performed by 8205's, which are high speed Schottky bipolar decoders with a delay of under 18 ns together with Schottky TTL devices. The interface unit has 10 input and 10 output port address. The talker-listener element has 8 sequential addresses and the controller element has 2 addresses the starting positions can be configured anywhere in the address space.

2.2 MULTIBUS DATA TRANSCEIVERS

A small amount of logic was needed to control the direction of data flow through the transceivers and to disable the outputs when not driving the bus. Control is achieved by using the read and write lines on the multibus and the chip select output from the address decoders.

2.3 TRANSFER ACKNOWLEDGE

The transfer acknowledge block gives an output to the transfer acknowledge line on the multibus to inform the processor that the data have been accepted. The talker-listener LSI circuit responds rapidly to commands whereas the controller LSI circuit is very much slower, so the transfer acknowledge circuitry delays a transfer acknowledge signal for much longer when data are sent to the controller LSI circuit than when sent to the talker-listener LSI circuit.

2.4 TALKER-LISTENER AND CONTROLLER

The talker-listener functions are implemented by the 8291[7] which is a LSI circuit designed to operate all the interface functions specified by the IEEE-488 Standard except the controller functions. The device has 8 read registers and 8 write registers, the device being programmed and data sent onto the bus by writing to the appropriate write registers and the device status is checked and data extracted from the IEEE-488 bus by interrogating the appropriate read registers.

The controller functions are implemented using the 8292[8] which is a pre-programmed microcomputer. The device which has 2 input ports and 2 output ports is programmed by writing operation commands to the input ports and the device status is checked by writing utility commands to the input ports and then reading from the appropriate output port.

2.5 ADAPTER LOGIC

An IEEE-488 Bus transceiver circuit has been designed by Intel and connects directly to the 8291 and 8292 and is designated the 8293. However due to the non-availability of the 8293 transceivers other transceivers were substituted and some logic was needed to operate these devices.

2.6 TRANSCEIVERS

The transceivers used were Motorola type MC 3448. Each device contains four independent driver/receivers, has a high impedance output mode, an active pull up or open collector output option, and meets the specifications for a transceiver as defined in the IEEE-488 Standard.

3 OPERATING THE IEEE-488 BUS CONTROLLER INTERFACE

The IEEE-488 Controller Interface can be operated using any programming language which allows data to be put out to or taken in from a specified port, and has been operated by BASIC 80, CORAL 66 and 8080 Assembler Programs.

Originally the controller was tested using 8080 assembler routines and as the required response was obtained, the routines were included in an ISIS library. A comprehensive library of basic software routine was built and an

extract from this library is given in Figure A5, and a brief description of each of these routines is given below.

3.1 TALK AND LISTEN

This talk routine initialises the 8291 talker-listener into the talk only (TON) mode. The 8291 is informed of the external clock frequency, which must be in the range 1-8 MHz. The operation code for talker only is sent followed by an immediate execute power on (PON) message which release the 8291 from the initialised state. The listen routine is identical to the talk routine except that the operation code for listen only (LON) is substituted for the TON code.

3.2 TRIG

This routine causes a 'Group Execute Trigger' message to be output to the IEEE-488 Bus. This stimulates addressed instruments to operate in the manner decided by their respective programming.

3.3 SREM

This routine sends an operation code to the 8292 controller which upon receiving the message sets the Remote Enable Line (REN) on the IEEE-488 bus low. Due to the negative logic specified in the standard, a low means that line is active.

3.4 GIDL AND TCNTR

The Goto Idle (GIDL) Command instructs the 8292 controller to go to an idle state. As this happens the Attention Line (ATN) goes high-inactive and the controller LSI circuit idles.

The Take Control (TCNTR) commands instructs the 8292 controller to take control of the IEEE-488 bus and as the idle state is left, the ATN line drops and becomes active.

3.5 RESET

The reset routine resets both the 8291 and the 8292 state. The 8291 enters an initialise state and can be programmed before being released from this state by a PON local message. The 8292 sets all outputs high then resets the interrupts low and clears the registers. The effects caused are identical to those produced by reset on the external reset pin.

3.6 SRQIN AND EOIIN

If the Service Request Line (SRQ) or the End Or Identify Line (EOI) are inactive after calling the respective routine, then the A register will contain a zero. These are useful routines although the 8291 can be configured to produce an interrupt when EOI becomes active and the 8292 can be configured to produce an interrupt when SRQ becomes active.

These routines could be expanded and built into a library for ease of assembly language programming or they could be modified and included in Coral 66 procedures or macros as code inserts. The latter course was chosen as the system was expanded.

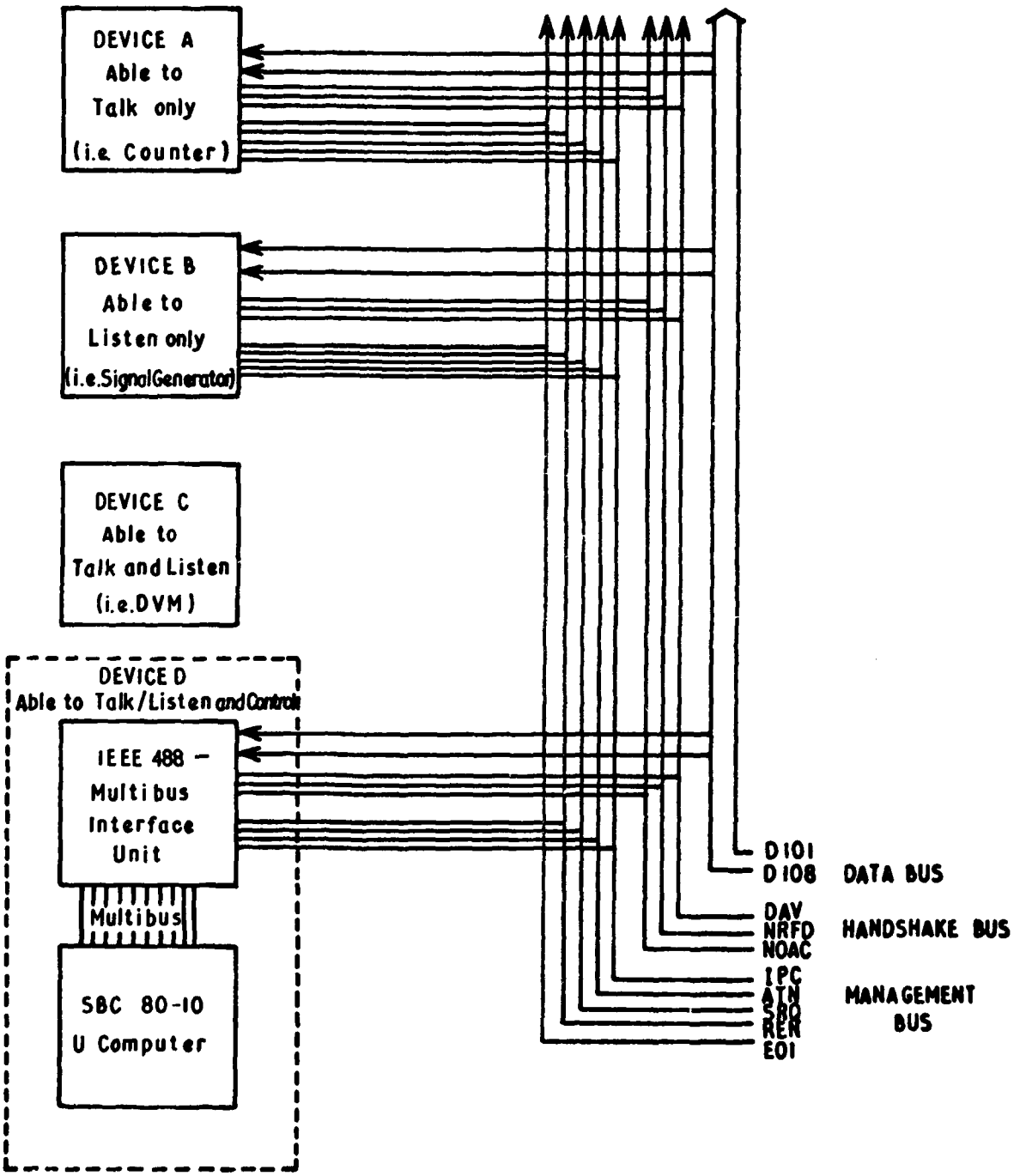


FIG.A1.INTERFACE CAPABILITIES AND BUS STRUCTURE

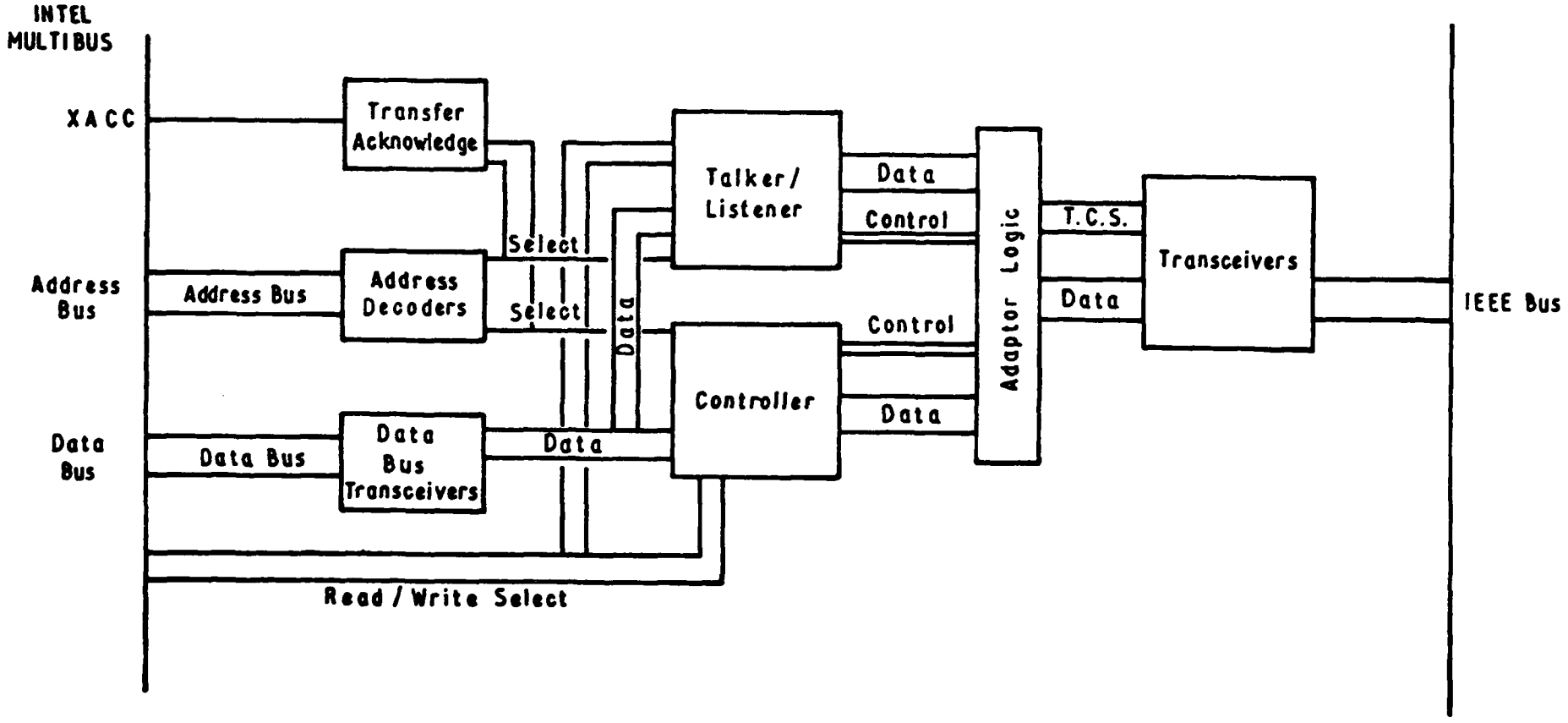
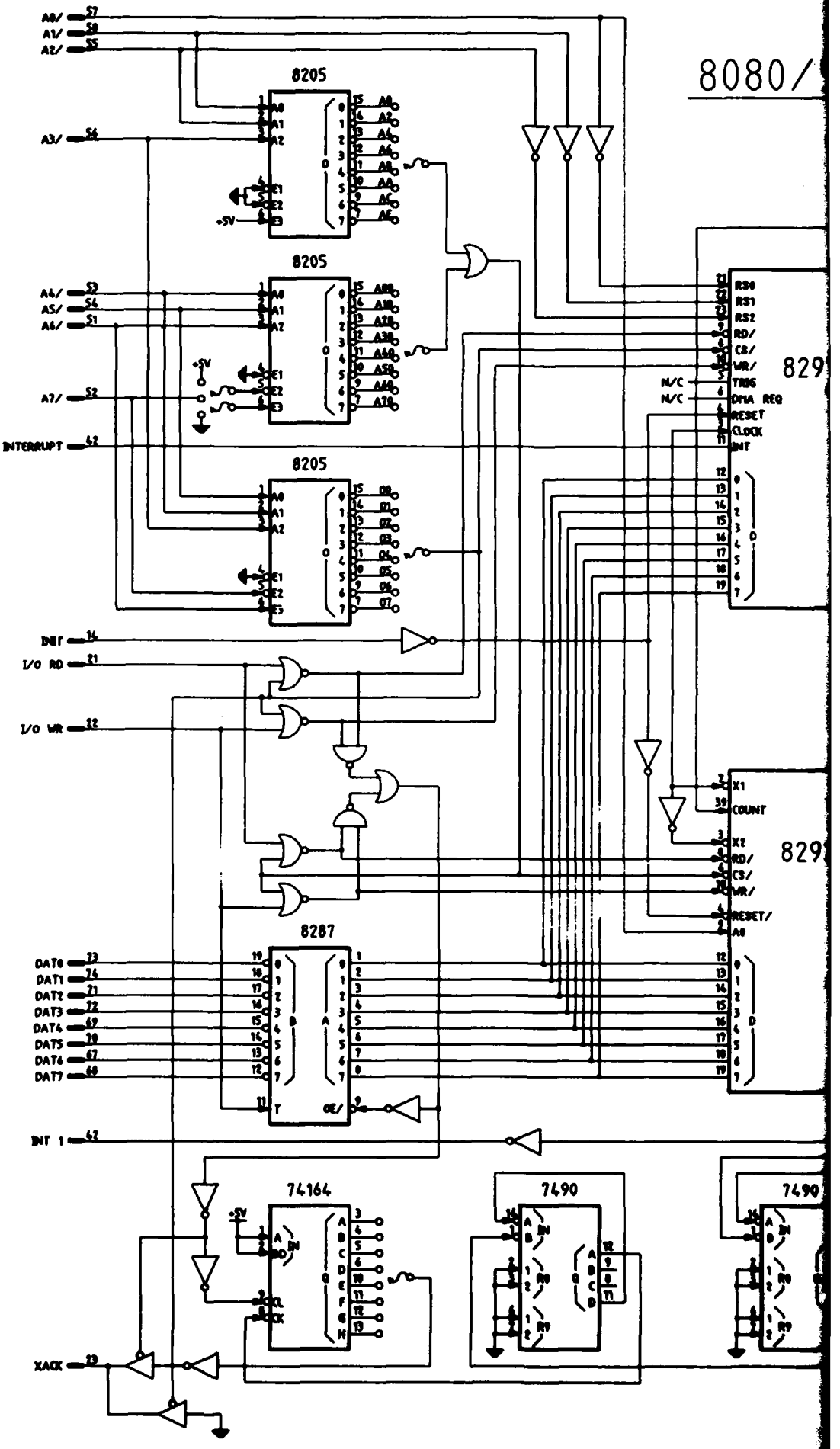


FIG. A2 BLOCK DIAGRAM OF IEEE-488 MULTIBUS INTERFACE


8080/
8205
8205
8205
8287
74164
7490
7490
A0/
A1/
A2/
A3/
A4/
A5/
A6/
A7/
INTERRUPT
INT
I/O RD
I/O WR
DAT0
DAT1
DAT2
DAT3
DAT4
DAT5
DAT6
DAT7
INT 1
XACK
+5V
RS0
RS1
RS2
RD/
CS/
WR/
TRIG
N/C
N/C
DMA REQ
RESET
CLOCK
INT
X1
COUNT
X2
RD/
CS/
WR/
RESET/
A0
829
829

8080/5 IEEE BUS CONVERTER

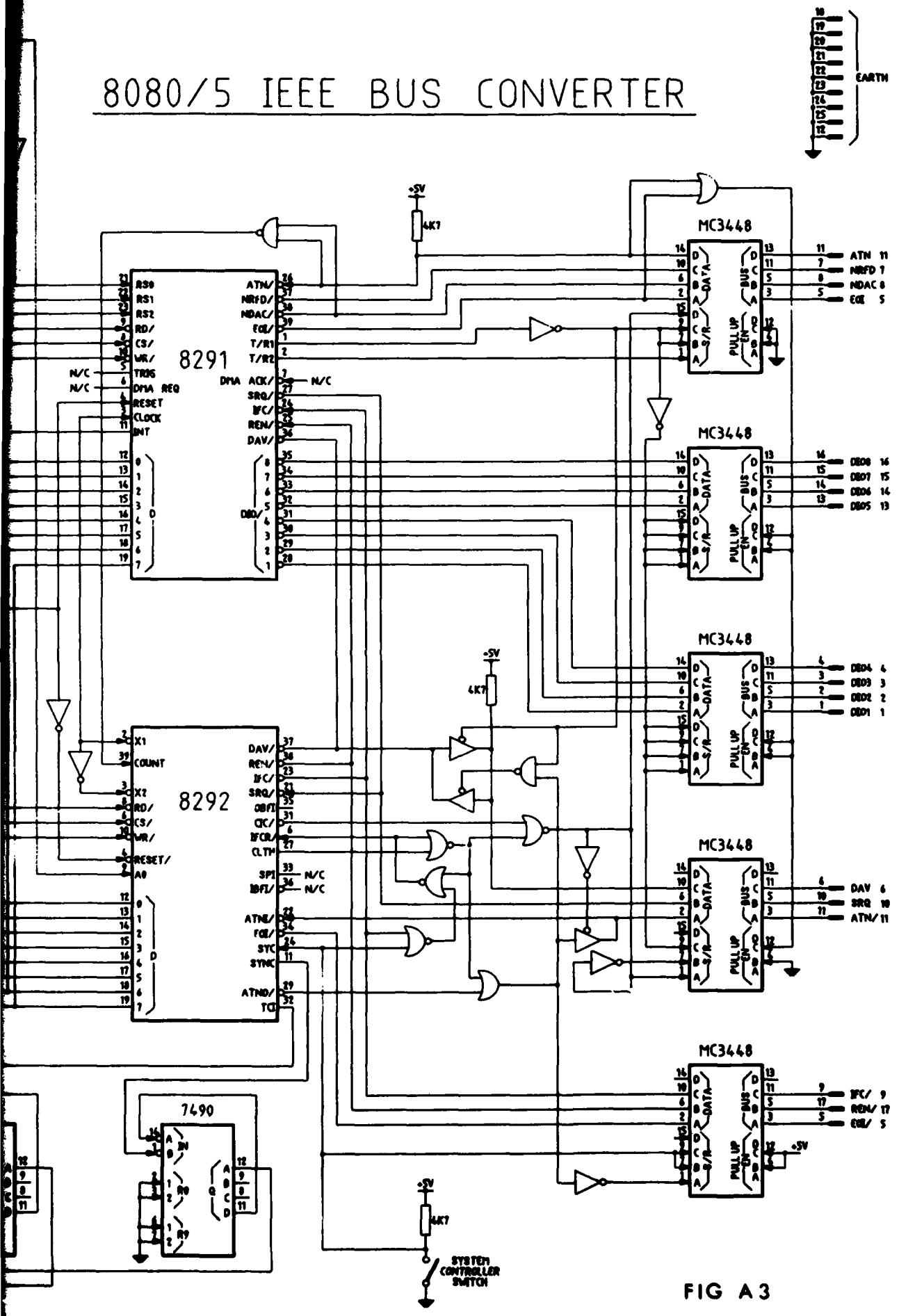


FIG A3

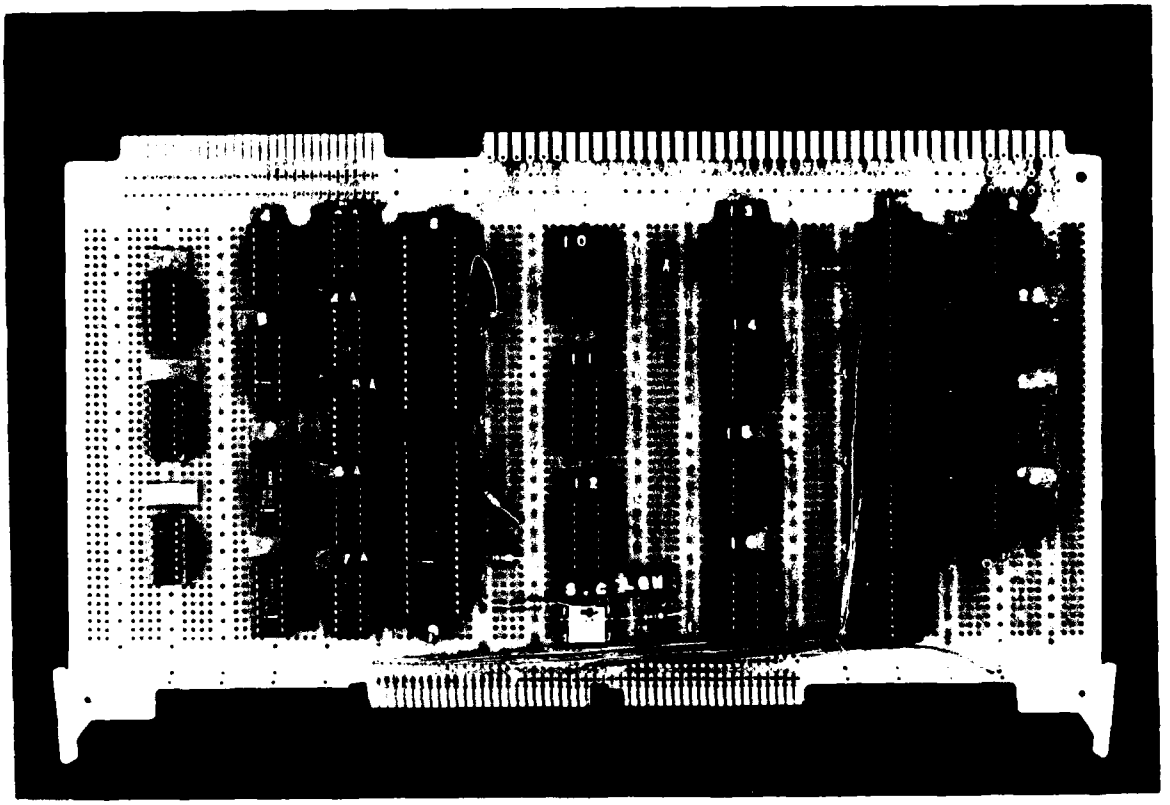


Fig A4

FIGURE A5

ASSEMBLY LANGUAGE ROUTINES FOR IEEE-488 CONTROLLER INTERFACE

```
MOBN01.002

OBRIEN.SRC - ASSEMBLER ROUTINES FOR IEEE-488 CONTROLLER INTERFACE
;*************************************************
;*       THIS PROGRAM CONTAINS THE ASSEMBLER ROUTINES     *
;*        NEEDED TO SET THE IEEE-488 INTERFACE.           *
;*        THESE COULD BE INSERTED INTO CONAL 66 CODE      *
;*        SEGMENTS OR USED IN AN ISIS LIBRARY.            *
;*************************************************

NAME MOB
TL0     EQU     0XXH            ;WHERE XX IS THE LOWEST PORT ADDRESS OF
TL1     EQU     TL0H + 1        ;     THE 8291 AS DECIDED BY THE LINK
TL2     EQU     TL0H + 2        ;      POSITIONS
TL3     EQU     TL0H + 3        ;
TL4     EQU     TL0H + 4        ;
TL5     EQU     TL0H + 5        ;
TL6     EQU     TL0H + 6        ;
TL7     EQU     TL0H + 7        ;
CLOCK   EQU     0010FFFFH       ;FFFF IS BINARY REP. OF CLOCK FREQUENCY
CON0    EQU     0YYH            ;WHERE YY IS THE ADDRESS OF THE 8292
CON1    EQU     CON0 + 1        ;          LOWEST PORT

PUBLICS TALKER,LISTEN,TRIG,SREM,GIDL,RESET,TCNTR,SRQIN,EOIIN

CSEG

TALKER:                         ;SET 8291 AS A TALKER
        MVI A,CLOCK             ;CLOCK FREQUENCY REPRESENTATION (1-8 MHZ)
        OUT TL5                 ;SEND TO AUX MODE REGISTER
        MVI A,10000000B         ;ENABLE TALK ONLY MODE (TON)
        OUT TL4                 ;ADDRESS REGISTER
        MVI A,0                 ;IMMEDIATE EXECUTE POWER ON (PON)
        OUT TL5                 ;AUX MODE REGISTER
        RET                     ;RETURN TO CALLING PROCEDURE

LISTEN:                         ;SET 8291 AS A LISTENER
        MVI A,21H               ;SET CLOCK
        OUT TL5                 ;
        MVI A,40H               ;ENABLE LISTEN ONLY MODE (LON)
        OUT TL4                 ;
        MVI A,0                 ;CON
        OUT TL5                 ;
        RET                     ;RETURN

TRIG:                           ;GROUP EXECUTE TRIGGER (GET)
        MVI A,00000100B         ;AUXILIARY COMMAND FOR GET
        OUT TL5                 ;AUX MODE REGISTER
        RET                     ;

SREM:                           ;SET INTERFACE TO REMOTE CONTROL
        MVI A,0F8H              ;8292 SREM CODE
        OUT CON1                ;UPPER PORT ON 8292
        RET                     ;

GIDL:                           ;GO TO IDLE,SETS THE ATN LINE FALSE
        MVI A,0F1H              ;8292 GIDL CODE
        OUT CON1                ;COMMAND FIELD PORT ON 8292
        RET                     ;
```


FIG A5

```
MOBNO1.002    ORRIEN.SRC   ASSEMBLER ROUTINES FOR IEEE-488 CONTROLLER INTERFA

TCNTR:                     ;TAKE CONTROL.SETS THE ATN LINE ACTIVE
        MVI A,0FAH         ;8292 UTILITY COMMAND FOR TCNTR
        OUT CON1           ;COMMAND FIELD
        RET                ;

RESET:                     ;RESETS THE 8291 & 8292
        MVI A,02H          ;8291 RESET CODE
        OUT TL5            ;AUX MODE REGISTER
        MVI A,0F2H         ;RST UTILITY COMMAND
        OUT CON1           ;COMMAND FIELD.UPPER PORT
        RET                ;

SRQIN:                     ;SRQ LINE ACTIVE IF NOT ZERO IN A REG(NO FLAG)
        MVI A,0E7H         ;RDST UTILITY COMMAND.READ GPIB BUS STATUS REGISTER
        OUT CON1           ;COMMAND FIELD PORT
        HLT                ;8292 IS SLOW TO RESPOND.USE TCI INTERRUPT ELSE LONG
        IN CONO            ;READ INTO LOWER PORT AFTER UTILITY COMMAND      ;WAIT
        ANI 01H            ;SRQ LINE SETS LSB (D0).THIS REMOVES OTHER BITS
        RET                ;

EOIIN:                     ;EOI LINE ACTIVE IF A REG NOT ZERO
        MVI A,0E7H         ;READ GPIB BUS STATUS REGISTER
        OUT CON1           ;COMMAND PORT
        HLT                ;AS ABOVE 8292 SLOW.EITHER WAIT OR USE TCI INTERRUPT
        IN CO              ;
        ANI 20H            ;EOI ACTIVE SETS BIT D5
        RET                ;

END
```

FIG A5(cont)

APPENDIX B

THE MASCOT APPROACH TO SOFTWARE CONSTRUCTION

MASCOT (A Modular Approach to Software Construction Operation and Test) is a design method which can be used with many programming languages or even assembly language. With the MASCOT technique a large system is broken down into smaller modules of which there are three main types, Activities, Channels and Pools.

An Activity is a process which is separately scheduled and conceived to run simultaneously with other Activities and performs a certain processing or control function. Activities may only transfer data externally, to another Activity or a physical peripheral device, by explicitly defined Intercommunication Data Areas (IDA's) these being Channels or Pools.

A Channel is a unidirectional data path used to transfer data between activities or between activities and peripheral devices. A channel consists of an input interface and by an activity that produces data to be transferred to another Activity and an output interface used by an activity that receives data which is consumed in the process. A channel also has a data area used for storing data which has been received but has not been stimulated to output.

The third type of module, the other IDA is known as a Pool and is a store of data for reference purposes. A pool may be read by one or more activities depending upon requirements. It is possible to modify data in a pool by means of a destructive overwriting operation that destroys all the previous data.

In order to schematically illustrate a MASCOT system a MASCOT ACP diagram is drawn, using the symbols that have been assigned to the various module types. In ACP diagrams a fourth symbol is often encountered, this being the representation of a system data source or sink, a peripheral device. The symbols used in MASCOT ACP diagrams are shown in Figure B1.

Intel 8080 MASCOT developed by System Designers Ltd was used for this project. 8080 MASCOT is a simple yet effective realisation of MASCOT which implements on the essential features so as to keep the software compact. 8080 MASCOT is a 'frozen' system as the system is constructed before run time using standard ISIS programs.

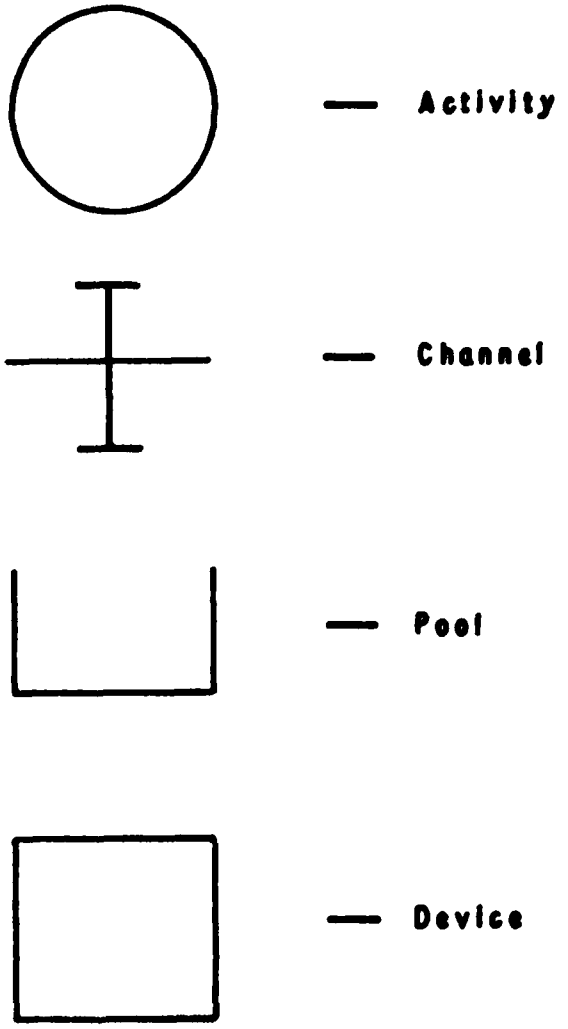


FIG. B1 MASCOT ACP DIAGRAM MODULE SYMBOLS

APPENDIX C

MASCOT MODULE DESCRIPTION

The MASCOT modules used in the complete test system are all described below except for those shown within the dotted lines. These modules make up the supervisor as it was supplied by System Designers Limited and is only a tool for debugging the system and was later removed to reduce the memory allocation required by the program.

1 ACTIVITY AC1

This is the main activity and is dependent upon the test being made but independent of the interface specification of the instruments used. In order to measure a certain value or to set a generator to a particular range and level, activity AC1 passes various parameters to the activity which controls the required instrument and, if applicable, receives a value back from that activity. Activity AC1 also displays after completion of the test the result obtained.

2 ACTIVITY LGASET

Activity LGASET is used to control the Logic Analyser and the Pattern Generator. Its purpose is to test one of several different simple digital circuits. The activity LGASET receives a parameter from AC1 indicating which digital unit is to be tested. Upon receipt of this the activity LGASET checks the IEEE-488 Bus status and at an appropriate time sets the Logic Analyser to the required state for a test. LGASET then addresses the Pattern Generator and triggers both instruments. The IEEE-488 Bus is monitored and when the test has been completed the data from the Logic Analyser is extracted. The data thus obtained is then compared with that for the relevant known good unit, the relevant sequence being decided by the value of the parameter passed from AC1. If the two streams are identical then a 'pass' message is transferred to AC1 otherwise a 'fail' message is sent.

3 ACTIVITY DVMSET

This activity is used to control the Multi-function Voltmeter and is independent of the test to be taken. The activity receives a parameter from AC1 which is interpreted to give the range and function required. The activity DVMSET then checks the IEEE-488 Bus status and then addresses the DVM and sends a suitable device dependent message so as to set it to the required state. After a measurement has been taken the DVMSET addresses the DVM to talk and reads from it a data string. This data string is checked to see if an error occurred and if not converts the value to floating point format and transfers it to activity AC1.

4 CHANNELS INI AND COACC

INI is a channel that allows the supervisor to initiate the activity AC1 and COACC is a channel from AC1 to the output device so that AC1 can display the results of the tests. Having manual intervention by the supervisor enables the facilities to be more fully used.

5 CHANNELS LAOACC AND LAIACC

These channels were used to allow communication between the activities AC1 and LGASET. Two channels are required due to the unidirectional nature of

MASCOT channels. The direction of the arrow on the MASCOT ACP diagram shows that the LAOACC is used to transfer information from activity AC1 to activity LGASET. LAIACC is used to transfer information in the reverse direction.

6 CHANNELS DVOACC AND DVIACC

These two channels perform a similar task to LAOACC and LAIACC in that they allow communication between the main activity AC1 and an instrument dependent activity, in this case the Multi-Function Voltmeter service activity.

7 CHANNELS OUTACC AND INACC

These two channels are used to allow communication between either of the activities LGASET and DVMSET and a peripheral device, in this case the IEEE-488 Bus Controller Interface. All access to and from the Bus is via these channels.

8 POOLS LGADAT AND DVMDAT

These modules, Pools, are used for data storage. The pool LGADAT contains the codes needed to set the Logic Analyser to the required state and also the streams of data obtained from various known good units. The pool DVMDAT contains the codes needed to set the Voltmeter. Obviously the codes needed to set the instruments depend upon the specification of the instruments.