TABLE OF CONTENTS

APPENDICES

Preface

This manual explains the operation and architecture of the New England Digital Corporation ABLE line of computers. Section I provides introductory material of a general nature while Section II explains the instruction set in detail. The Appendices provide information on basic IO equipment, interfacing, installation, and other reference material.

Section I -

Introduction

The ABLE series computers are high-speed general purpose computers whose design has emphasized simplicity and compatability with efficient structured language programming. They are 16-bit word length machines organized around 16 registers. All of the registers are usable as index registers with auto-increment capability. Two of the registers, R17(octal) and R16(octal), may be used as program counters. R17 is the normal program counter, while R16 is used while processing interrupts or calling subroutines.

Four of the registers, (R0-R3), are designated as accumulators. These registers have the additional capability of executing the complete set of arithmetic, logical and test operations. Byte manipulation instructions are also included in the logical operation set to facilitate character processing and other byte-oriented calculations.

The memory addressing is by word using a 16-bit memory address buss to enable direct addressing of 65,536 words. All memory addressing is indirect on a register; i.e. the register contents are used to point to the desired memory location. Relative addressing is provided for the transfer (jump, branch) instructions. Any memory location in the entire 65K space may be accessed within at most one Load instruction, eliminating the need for any paging or mapping complexity. The importance of this feature is supreme when contemplating programs that must run in different combinations of ROM and RAM. The

existance of so many registers makes the task of memory addressing much more tractable, whether it is being done by a human assembly language programmer or by a compiler.

There are three main busses in the machine. Data transfers are made by means of a 16-bit bi-directional data buss (BDB0-BDB15). Memory addressing is by means of a 16-bit memory address buss (MA0-MA15).

An 8-bit address/instruction buss serves to address all IO devices and to carry Instruction information to the register/ALU sections of the machine. All data transfer instruction information is encoded on this special buss, greatly facilitating multi-processor applications. Any intelligent agent may HALT the processor and take command of the complete buss system to execute any instruction of which the system is capable. Instead of a normal programmer's panel which depends on jamming data into an instruction register, the panel of an ABLE computer is really a processor in its own right called the HOP:Hand Operated Processor.

In addition to the major data, address, and instruction busses, there are several control lines which define execution timing sequences, direction of data transfers, and so forth. Among these are the READ and WRITE control lines which carry directionality information. A READ transfers data from a source to the processor; a WRITE transfers data from the processor to a destination.

Section II describes the ABLE instruction set in detail.

The ABLE Line Computers

The computers in the ABLE line are based on a single central processor configured on two 5" by 7" printed circuit boards. All ABLE computers use the same basic mechanical design, differing only in power supply capacity, slot availability, and "standard" equipment. The central processor, memory, and interface modules are constructed on 5" by 7" circuit boards, all of which are compatible with the different ABLE systems.

The ABLE/20, New England Digital's smallest OEM system, incorporates a central processor, 10-slot chassis, multi-function terminal interface and loader card, real time clock, and a single drive mini-floppy disk subsystem offering 89,600 bytes of on-line storage. A 12-amp power supply provides sufficient capacity for most OEM applications. 4,096 bytes of memory, expansible up to 32,768 bytes, are included in the ABLE/20.

The /20 is designed to be incorporated into OEM systems where compiled programs, either stored in ROM or on a diskette, are used. For dedicated control applications, an ABLE/20 can be configured without a diskette (using ROM for program storage) to realize a significant cost savings to the OEM.

The ABLE/40 system, the smallest complete software-development system, incorporates a 15 slot interface bin, 32,768 bytes of memory, a 20-amp switching power supply, plus a dual-drive Mini-floppy disk subsystem offering 179,200 bytes of on-line storage. High-level language software development is accommodated using New England Digital's Real Time XPL Operating System - a complete operating system including a Monitor, line-numbered file Editor, and an optimizing 3-pass XPL language compiler.

XPL is a high-level structured-programming language specifically designed to simplify the task of writing real-time computer programs. XPL is an extremely powerful language in that many logical and bit-manipulation functions can easily be performed. The XPL system gives the programmer sufficient control of the processor for the needs of complex system programming, but provides automatic control over many specific computer resources such as the registers, memory, and push down stack.

XPL has been designed to facilitate the use of modern techniques in structured programming. Capabilities such as procedure definitions, variable localization, and sophisticated if-then-else logical processing assist in the rapid development of easy to debug and straightforward to maintain software.

By use of the powerful READ and WRITE statements built into XPL, directly executed machine instructions can be incorporated into any XPL program. Although register allocation and memory location

assignments are best performed by the compiler itself, the in-line code capability provided in XPL can be used to override these features.

For highly specialized software applications, New England Digital's Assembly Language Operating System can be used to write programs using the ABLE Assembler.

The ABLE/40 can be expanded up to 114,688 bytes of memory. Additional disk drives, serial data lines, or other interface modules can be added to an ABLE/40 to customize the system for specialized applications.

The ABLE/60 computer is similar to the /40 except that full-sized diskettes (8") are used for the standard storage media. The dual-drive full-size diskette system features 630,784 bytes of on-line storage. Memory expansion up to 114,688 bytes is possible with the /60.

The ABLE/80 computer includes an expanded 21-slot rack-mounted bin plus 4 special purpose interface modules: a 16-line 12-bit analog to digital converter, a scientific timer offering 1 microsecond resolution, a 32-bit digital IO interface (TTL), plus a Hand Operated Processor.

Peripheral Devices

The ABLE line of computers supports a wide range of peripheral devices designed for scientific, data processing, and OEM applications. Each device is compatible across the entire ABLE line.

Processor Products

D1	Instruction Sequencer
D300	Register Module
MFC1	Multifunction Terminal Interface and Loader Card with 100Hz/1kHz RTC.

Storage Devices

D100	Mini-floppy Diskette System
D100	Full-size Diskette System
	Winchester Disk System - 10, 20, 40 megabyte
	Tape Drive

Output Devices

	Line Printer
	Plotter
D4050	Dual Serial Port

Memory Products

M8K	8192 word static semi-conductor memory
M2K	2048 word static semi-conductor memory

Scientific Interfaces

D12	16-channel analog-to-digital converter
D60	Dual 10-bit digital-to-analog converter
D16	Scientific timer
D34	32-bit digital IO module
DØ	Hand Operated Processor
D70	Oscilloscope Interface

Section II

Central Processor

The central processor is the control unit for the entire ABLE computer: it controls the sequencing of computer instructions, governs all input and output, and performs all arithmetic, logical, and data movement operations. The processor uses the 8-bit Address/Instruction buss in conjunction with the READ command line and the WRITE command line to fetch instructions from memory and perform the specified data movement or computation. During normal operation, the processor fetches and then executes instructions stored in sequential locations of memory. Sequential program flow is accomplished automatically by incrementing the program counter after each instruction fetch, using the built-in auto-increment feature. This sequential accessing of successive memory locations combines most effectively with the 8-way interleaving available on the 8K memory module to provide a significant increase in system throughput.

Sequential program flow is altered by the execution of a JUMP instruction. The ABLE processor provides 7 modes of conditional transfer to either an absolute or relative memory location. (See specific instruction descriptions beginning p. 2-12.)

The central processer is constructed on two 5" by 7" printed circuit boards. The first board, designated the D1 Sequencer, is the module that sequences the operation of the entire computer. When the HALT line is released, the Sequencer uses a READ command to read a 16-bit instruction from memory. This instruction then directs the Sequencer to perform a data movement, arithmetic computation, or conditional transfer of control. The formats for the different instructions will be presented shortly.

The rest of the central processer is constructed on a circuit board designated the D300 Register Module. The Register Module contains a high speed 16-bit register stack, a 16-bit ALU, and three status flags labelled C (for carry), Z (for zero) and M (for minus). By manipulating the Instruction/Address buss and the various control lines, the Sequencer can direct the Register Module to:

a) gate the contents of a register onto the Data Buss;

b) strobe the information on the Data Bus into a register;

c) perform an arithmetic computation such as ADD, SUBTRACT, or SHIFT using one operand stored in a register and the other operand from the Data Buss;

d) present the three status flags (C, Z, and M) to the processer for evaluation of a conditional transfer; OR

e) gate the contents of a register onto the Memory Address Buss to identify a specific memory location.

The C, Z, and M status flags are used to indicate that an arithmetic computation:

A) resulted in a number greater than 2^{16} (c),

B) produced a zero result (Z), OR

C) produced a negative result (M).

During the processing of a conditional transfer instruction, the Sequencer evaluates the condition of the 3 status flags. By comparing the 3 status flags with the condition code present in the conditional transfer instruction, the Sequencer can decide whether or not to transfer control to the specified memory location.

Interrupt recognition is also handled by the D1 Sequencer. The Interrupt Request line, IRQ, is sensed by the Sequencer at the end of every instruction. When IRQ is true (i.e. held low by the interrupting device), the Sequencer disables further interrupts and begins to fetch instructions using R16 as the program counter instead of the normal R17. For proper operation, R16 must be loaded with a pointer to the interrupt routine before interrupts are enabled.

Further information on interrupt processing is found in the discussion of the Processor Status Word (PSW) and the Processer Control Word (PCW) later on in this section.

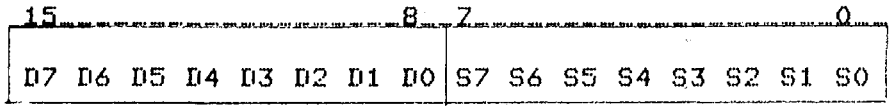
INSTRUCTION FORMAT

The Instruction Format of the ABLE series computers is somewhat unique in that each machine instruction is a full 16-bit word composed of two 8-bit addresses. The least significant half of the instruction word (bits 0-7) comprises the address of a SOURCE of data; the most significant half (bits 8-15) comprises the address of the DESTINATION for that data. That the instruction information is encoded as an address makes the operation of the computer "observable" from the outside world; and conversely allows the execution of machine operations under the control of intelligent agents other than the processor itself.

Normal instruction execution involves data transfers of the form of a READ from the SOURCE followed by a WRITE to the DESTINATION. A single machine instruction can read from any SOURCE and write to any DESTINATION. The operation of the instruction can be viewed as a general crossbar switch connecting all the modules of the system.

A unique feature of the instruction/address format is that the two 8-bit fields of the machine are INDEPENDENT; i.e. any code in either field does not modify the meaning of the code in the other field.

The instruction appears in the Instruction Register as follows, where Dn represents the nth bit of the DESTINATION ADDRESS and Sn represents the nth bit of the SOURCE ADDRESS.



The followings are the defined classes of SOURCES and DESTINATIONS and their binary codes.

	CODES (7 … 0)	CLASS
SOURCES	0 | IO DEV. ADDR.	IO SOURCE
	1 0 | S D_4 D_3 D_2 D_1 D_0	IMMEDIATE SOURCE
	1 1 0 | I | R_3 R_2 R_1 R_0	REGISTER SOURCE
	1 1 1 | I | R_3 R_2 R_1 R_0	MEMORY SOURCE

	CODES (15 … 8)	CLASS
DESTINATIONS	0 | IO DEV. ADDR.	IO DESTINATION
	1 0 0 | A_2 A_1 A_0 | R_1 R_0	ARITHMETIC/LOGICAL
	1 0 1 0 | T_1 T_0 | R_1 R_0	TEST
	1 0 1 1 0 0 | R_1 R_0	BYTE SWAP/OR
	1 0 1 1 0 1 | R_1 R_0	RIGHT SHIFT/ADD
	1 1 0 0 | R_3 R_2 R_1 R_0	LOAD REGISTER
	1 1 0 1 0 | C_2 C_1 C_0	COND. TRANSFER, ABSOLUTE
	1 1 0 1 1 | C_2 C_1 C_0	COND. TRANSFER, RELATIVE
	1 1 1 | I | R_3 R_2 R_1 R_0	MEMORY DESTINATION

The structure of the ABLE series computers is illustrated in simplified form in Fig.1. The three major busses of the system are the 16-bit Data Buss, the 16-bit Memory Address Buss and the 8-bit Instruction/Address Buss. These busses and several Control lines such as READ, WRITE, and SYNC serve to link all the modules of the system.

The Sequencer controls the READ and WRITE operations on all system modules by means of the Instruction/Address Buss and the Control lines.

The Register Module utilizes 16 registers to control the Memory Address Buss. The Register Module also contains the system ALU which performs arithmetic and logical operations as well as memory address arithmetic for pointer control and relative branching .

The Sequencer transfers 16-bit operands to and from the Register Module and IO devices using the Instruction/Address Buss for operation/module identification, the Control lines for directionality and timing, and the Data Buss for the actual transfer of the data. To transfer data to and from Memory, in addition to these busses the Sequencer utilizes an Indirect bit in the Instruction/Address to cause the Register Module to provide the proper Memory Address to the Memory Address Buss.

The Sequencer utilizes a basic three-step micro-cycle to implement a two-address instruction format. The first step

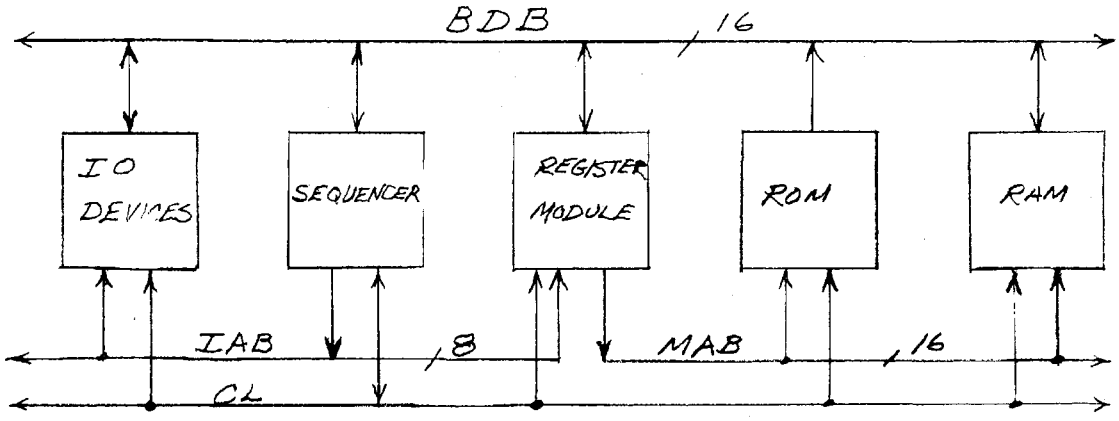


FIG. 1

STRUCTURE OF ABLE SERIES COMPUTERS

BDB	BIDIRECTIONAL DATA BUSS	16 BITS
CL	CONTROL LINES	
IAB	INSTRUCTION/ADDRESS BUSS	8 BITS
MAB	MEMORY ADDRESS BUSS	16 BITS

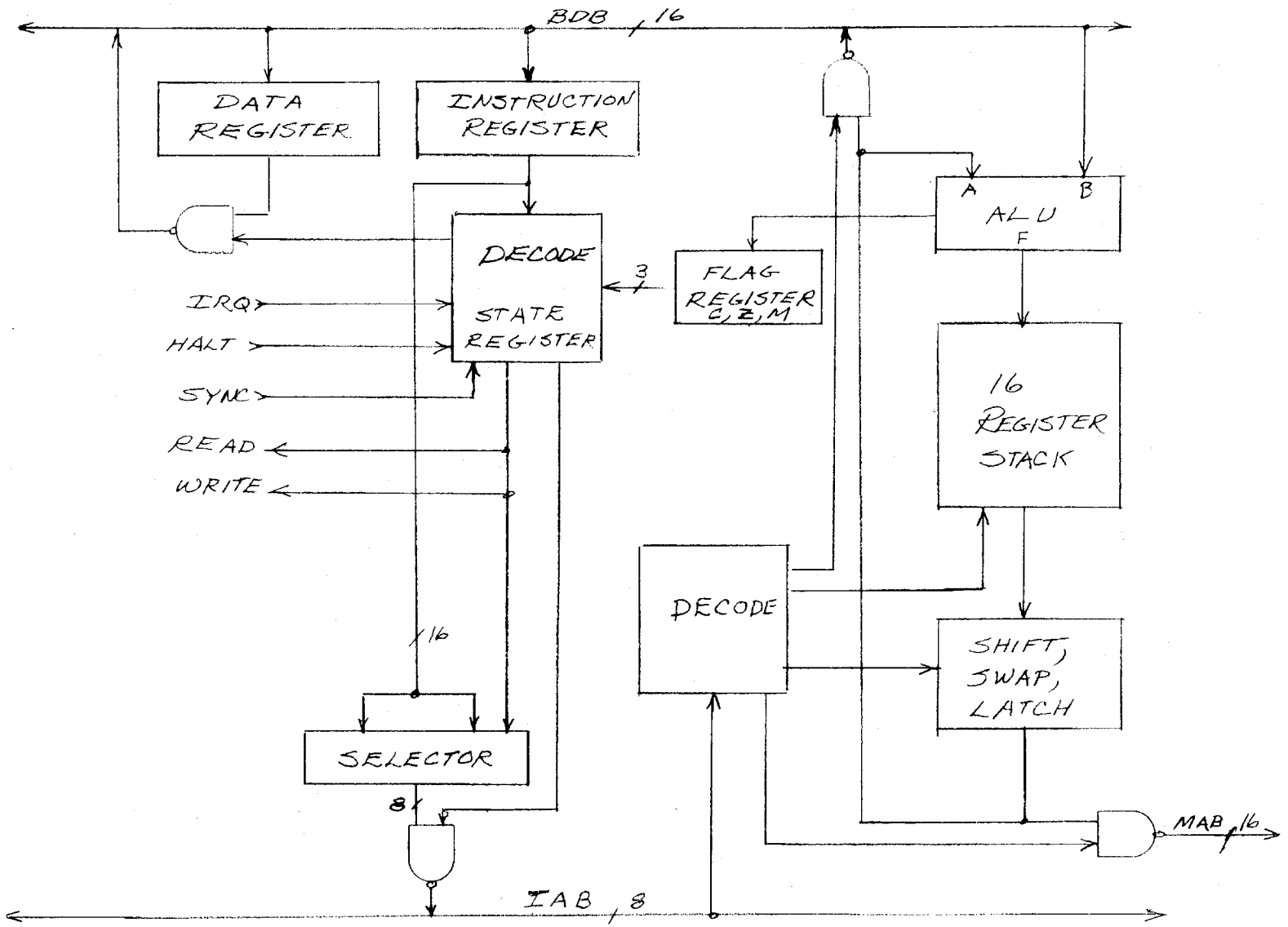


FIG. 2 STRUCTURE OF ABLE SERIES SEQUENCER & REGISTER MODULE

of the micro-cycle is to READ Memory at the location pointed to by the Program Counter and then increment the Program Counter. The data read by this step will consist of a machine instruction. This 16-bit word is stored in the Sequencer's Instruction Register and Data Register.(The reason for storing the instruction in the Data Register will become apparent when IMMEDIATE SOURCES are discussed.) The least- significant half of this instruction word consists of an Address for a SOURCE of data while the most-significant half consists of an Address of a DESTINATION for that data.

The second step of the micro-cycle is to perform a READ operation using the least-significant half of the instruction as the SOURCE ADDRESS. The 16-bit operand obtained as a result of this READ operation is stored in the Sequencer's Data Register.

The third step of the micro-cycle is to perform a WRITE operation using the most-significant half of the instruction word as the DESTINATION ADDRESS.The 16-bit data operand stored in the Data Register of the Sequencer is presented on the Data Buss to the addressed operation/device. At the end of the WRITE operation the micro-cycle is essentially complete. Data has been transferred from any SOURCE in the system to any DESTINATION in the system.

Before starting another micro-cycle, however, the Sequencer at the end of a WRITE operation interrogates the

HALT Control line and the Interrupt Request Control line. If the HALT Control line is being asserted at this time the Sequencer will answer by means of the Halt Acknowledge Control line; further operation of the Sequencer will cease until such time that the HALT Control line again becomes false. If the Interrupt Request Control line is asserted at this time and interrupts are enabled, the next instruction will be fetched using R16 octal as Program Counter instead of the usual R17 octal.

Otherwise the Sequencer enters the next micro-cycle and operation continues as described above.

Under certain circumstances the micro-cycle as described above is modified by omitting either the SOURCE READ or the DESTINATION WRITE or both. If the specified SOURCE was the 6-bit two's compliment type of IMMEDIATE SOURCE, the required data is already present in the Data Register at the end of the Instruction Fetch. The sign bit (bit 5) need only be extended through bit 15 and the data is ready to be written to the DESTINATION. This is the reason for copying the instruction into the Instruction Register and the Data Register. In this case the Sequencer will omit the SOURCE READ operation step of the micro-cycle and will proceed with the DESTINATION step if appropriate.

The omission of the DESTINATION step of the micro-cycle is also possible. The CONDITIONAL TRANSFER instructions

operate by either writing a 16-bit data operand into the current Program Counter or omitting that WRITE depending upon whether or not the condition specified is met. So in the case of a condition code which is not met the Sequencer will fetch the appropriate SOURCE operand but will proceed to the next Instruction Fetch, omitting the DESTINATION WRITE to the current Program Counter.

DETAILS OF SOURCES AND DESTINATIONS OF INSTRUCTIONS

IO SOURCE 0 - - - - - - -

The IO SOURCE allows the readins of data from up to 128 different devices. The encodins of the instruction set demands that IO devices have 8-bit addresses with bit 7 equal to 0. That is, all IO devices have addresses less than 200(octal). IO sources are used both for data and for status of the device. For example, device 50 is the data port of the system control terminal; device 51 is the address of the status resister of system control terminal.

IMMEDIATE SOURCE 1 0 S D4 D3 D2 D1 D0

THE IMMEDIATE SOURCE provides a compact and rapid means of providins small constants(statistically the most often used) within the instruction itself. That is, the execution cycle for an Immediate Source will be altered to omit the normal source read because the data is already at hand after the instruction fetch. The immediate source provides a 6-bit, two's-compliment operand; i.e. a number in the ranse of -32(decimal) to +31(decimal). The sisn bit, S, is extended from bit 5 throush bit 15.

REGISTER SOURCE 1 1 0 I R3 R2 R1 R0

The sixteen registers of the machine are read by means of the REGISTER SOURCE codes. The four least significant bits name the register to be read. The "I" bit controls auto-incrementation of the contents of the register AFTER reading. When the I bit is zero the register contents remain unchanged after a read. When the I bit is one the register contents are first read and then increased by one.This feature is useful in the implementation of loop counters using non-accumulator registers (i.e. registers 4-15 octal).

MEMORY SOURCE 1 1 1 I R3 R2 R1 R0

To read a location of memory a source field of this type is employed. A register of the machine must be pointing at the desired memory location. This register is named by the least significant four bits of the field; and the named register may be automatically post-incremented under control of the "I" bit , in a manner exactly similar to the register source. A very important feature of the general nature of the memory addressing and register architecture is the ability to use the PC (program counter) as a pointer to an operand of data immediately in line with the program.

IO DESTINATION 0 - - - - - - -

Data is written to true IO devices using the IO DESTINATION field. As in the case of IO sources device addresses of IO destinations are in the range of 0-177 octal.The IO destination is used both for data transfers and for transfer of control information to IO devices.

ARITHMETIC DESTINATIONS 1 0 0 A2 A1 A0 R1 R0

The members of the ARITHMETIC DESTINATION class of instruction share several common features. They may be used with any of the four accumulators R0-R3. Which accumulator is specified by the "R" bits , bits 9 and 8 of the full instruction.

When an ARITHMETIC DESTINATION instruction is used the states of the ZERO (Z), CARRY (C), and MINUS (M) flags at the end of execution are stored in the FLAG REGISTER. These flags are the means of providing alternative execution information in conjunction with the CONDITIONAL TRANSFER instructions.

The specification of which of the eight arithmetic/logical operations is to be performed is encoded using bits 12, 11, and 10; (A2,A1, and A0) of the full instruction word.

ARITHMETIC LOAD 1 0 0 0 0 0 R1 R0

The ARITHMETIC LOAD destination transfers the 16-bit operand read by the source-read operation into the specified accumulator. The ZERO flag will be SET (=1) if the operand loaded was zero and will be CLEARED if there were any ones in the operand. The MINUS flag will be SET if the most-significant bit of the operand was one; and CLEARED if the MSB was zero. The CARRY flag will be UNDEFINED.

The previous contents of the specified accumulator are lost.

AND 1 0 0 0 0 1 R1 R0

The AND instruction performs a bit-by-bit logical AND operation between the previous contents of the specified accumulator and the 16-bit operand read by the source-read operation. The accumulator specified is left containing the result of this AND.

The ZERO and MINUS flags are affected. The CARRY flag will be UNDEFINED.

ADD 1 0 0 0 1 0 R1 R0

The ADD instruction adds the 16-bit operand read by the source-read operation to the previous contents of the specified accumulator and stores the result in that accumulator. The ZERO, CARRY, and MINUS flags are all affected by ADD.

Since there is no specific left-shift instruction in the instruction set, this operation is performed by specifying the same accumulator in the source field as is specified in the ADD instruction; e.g. R1 to R1 ADD has the affect of shifting the contents of R1 left one place.

SUBTRACT 1 0 0 0 1 1 R1 R0

The SUBTRACT instruction subtracts the 16-bit operand read by the source-read operation from the previous contents of the specified accumulator and stores the difference in that accumulator. The ZERO,CARRY,and MINUS flags are all affected by SUBTRACT.

Note that after a SUBTRACT a CARRY may be defined as a NOT BORROW.

EXCLUSIVE OR 1 0 0 1 0 0 R1 R0

The EXCLUSIVE OR instruction performs the EXCLUSIVE OR or "add without carry" on a bit-by bit basis between the 16-bit operand read by the source-read operation and the previous contents of the specified accumulator and stores the result in that accumulator. The ZERO and MINUS flags will be affected. The CARRY flag will be UNDEFINED.

LEFT ROTATE 1 0 0 1 0 1 R1 R0

The LEFT ROTATE instruction shifts every bit of the previous contents of the specified accumulator left one position . The MSB (bit 15) is rotated into the LSB (bit 0) position. The ZERO flag will be unchanged. The MINUS flag will be affected. the CARRY flag will be set to the value of bit 15 (MSB) prior to the execution of the LEFT ROTATE instruction.

The LEFT ROTATE instruction takes only a single operand and so the customary source field for the instruction would be an IMMEDIATE zero ; this will cause the source-read operation to be omitted with a savings of time resulting.

OR 1 0 0 1 1 0 R1 R0

The OR instruction performs the logical OR operation on a bit-by-bit basis between the 16-bit operand read by the source-read operation and the previous contents of the specified accumulator. It is sometimes referred to as the "set bit" instruction. The ZERO and MINUS flags will be affected and the CARRY flag will be UNDEFINED.

LOAD COMPLIMENT 1 0 0 1 1 1 R1 R0

The LOAD COMPLIMENT instruction loads the 16-bit operand read by the source-read operation into the specified accumulator and then performs the "one's compliment" operation on that operand. That is, each bit is simply changed to its opposite. The ZERO and MINUS flags will be affected and the CARRY flag will be UNDEFINED. The previous contents of the accumulator are lost.

TEST DESTINATIONS 1 0 1 0 T1 T0 R1 R0

The TEST destinations form a group of four instructions which behave with respect to the setting of the ZERO, CARRY and MINUS flags exactly as their corresponding ARITHMETIC destinations do. The difference is that the test instructions do NOT affect the CONTENTS of the accumulator specified. Thus they provide an important means of performing logical or arithmetic tests without storing accumulators when all accumulators are in use; and a means of testing various properties of the contents of an accumulator without requiring the restoring of the tested quantity.

The TEST operations may be performed on any accumulator as specified by the "R" bits. The actual test operation is specified by the "T" bits as follows:

T1	T0	OPERATION
0	0	LOAD
0	1	AND
1	0	ADD
1	1	SUBTRACT

BYTE-SWAP WITH OR 1 0 1 1 0 0 R1 R0

The BYTE_SWAP WITH OR instruction operates on any of the four accumulators as specified by the "R" bits. This instruction interchanges the most-significant and least sicnificant bytes of the previous contents of the specified accumulator and then persorms the losical OR of the swapped word with the 16-bit operand read by the source-read operation.The ZERO and MINUS flags are affected and the CARRY flag is undefined.

If the OR feature of this instruction is not needed the source field is set to IMMEDIATE zero and the instruction becomes the single-operand instruction BYTE-SWAP.

The OR feature of the BYTE-SWAP with OR instruction is useful when handling characters from terminals or modems in that 8-bit quantities may be positioned in 16-bit words in a single instruction

RIGHT-SHIFT with ADD 1 0 1 1 0 1 R1 R0

The RIGHT-SHIFT WITH ADD instruction operates on any of the four accumulators as specified by the "R" bits. This instruction shifts every bit of the previous contents of the specified accumulator one place to the right.A zero is shifted into the MSB (bit 15) and the previous LSB (bit 0) is stored in the CARRY flag.After the shift takes place the

16-bit operand read by the source-read operation is added to the shifted accumulator. The ZERO and MINUS flags are affected; the CARRY flag is equal to the previous LSB of the selected accumulator.

As in the BYTE-SWAP WITH OR instruction , if the ADD feature is not desired the instruction can be made into the single-operand SHIFT RIGHT instruction simply by choosing an IMMEDIATE zero for the source field.

The ADD feature of the RIGHT-SHIFT with ADD instruction is used to provide the ability to shift in either ones or zeroes as desired. This means that the RIGHT ROTATE instruction may be performed by means of branching on the condition of the carry flag since that flag represents the previously shifted LSB of the operand.

REGISTER DESTINATION 1 1 0 0 R3 R2 R1 R0

This form of destination allows the loading of the non-accumulator registers (R4-R17 octal). It also may be used to load an accumulator register (R0-R3) without changing the flags. In any case the register to be loaded is specified by the "R" bits.

The PROGRAM COUNTER (R17 octal normally or R16 octal) may be specified as the register sub-field in which event the instruction becomes an ABSOLUTE JUMP UNCONDITIONAL to the address specified by the 16-bit operand read by the source-read operation.

CONDITIONAL TRANSFERS, ABSOLUTE 1 1 0 1 0 C2 C1 C0

The ABSOLUTE CONDITIONAL TRANSFERS enable the programmer to alter the execution sequences of instructions depending on the results of ARITHMETIC or TEST instructions. Both the positive and negative conditions of the ZERO, CARRY, and MINUS flags are specifiable as conditions under which program execution is transferred to another address of memory. If the specified condition is not met the next instruction executed will be from the next sequential address after the CONDITIONAL instruction.

The specific condition under which program execution will be transferred is given by the "C" bits:

C2	C1	C0	CONDITION
0	0	0	UNCONDITIONAL TRANSFER
0	0	1	TRANSFER IF ZERO
0	1	0	TRANSFER IF CARRY
0	1	1	TRANSFER IF MINUS
1	0	0	TRANSFER NEVER
1	0	1	TRANSFER IF NOT ZERO
1	1	0	TRANSFER IF NOT CARRY
1	1	1	TRANSFER IF NOT MINUS (PLUS)

If the specified condition is met, the 16-bit operand read by the source-read operation is loaded into the program counter which is currently in use (be it R17 or R16 octal). Program execution will then commence at the location transferred. If the condition specified is not met the next sequential instruction will be executed; the processor will shortenthe machine micro-cycle by omitting the DESTINATION WRITE operation entirely.

CONDITIONAL TRANSFER, RELATIVE 1 1 0 1 1 C2 C1 C0

In the RELATIVE CONDITIONAL TRANSFER instruction, operation with respect to the invoked conditions is exactly the same as in the the ABSOLUTE CONDITIONAL TRANSFER instruction.The difference is that in the RELATIVE TRANSFER instruction the 16-bit operand read by the source-read operation is ADDED to the current program counter in the event that the specified condition is met. Assuming that the operand is expressed in two's compliment notation it is obvious that program control may be transferred either forward or backward with respect to the current location.

It should be noted that when the RELATIVE CONDITIONAL TRANSFER instruction is used, the program counter has already been incremented beyond the location of the conditional instruction itself.In the case of an operand fetch using the program counter as a memory pointer to an immediate data word, the program counter will have been incremented twice beyond the location of the conditional instruction itself.

MEMORY DESTINATION 1 1 1 I R3 R2 R1 R0

The memory destination instruction provides the ability to store 16-bit operands read by the source-read operation in a location of memory pointed to by any of the sixteen registers. In addition the contents of the named register may be post-incremented automatically under the control of the "I" bit (bit 12).

The register to be used in the store operation is specified by the four "R" bits of the destination field.

Note that a simple class of DMA device may be constructed which would use a register of the Register Module to access the Memory Address Buss. While this type of design would not utilize the full bandwidth of the memory channel it would have the advantage of simplicity.

SPECIAL SOURCES AND DESTINATIONS

Several processor related functions have been implemented as IO address functions to preserve the symmetry of the instruction set . These are the Hand Operated Processor (HOP), the Processor Status Word (PSW), and the Processor Control Word (PCW).

HOP Device Address 000 (octal)

The Hand Operated Processor allows an operator to observe and control the 16-bit Bi-Directional Data Buss, the 8-bit IO Address Buss, the Read and Write control lines,the Sync line and the Halt and Halt Acknowledge lines.

Data can be entered into a program by reading Device 000 octal; i.e. using an instruction Source field equal to 000 octal. Since the ABLE computers are asynchronous, execution will be suspended upon a Read of Device 000 until the operator returns a Sync signal from the HOP. Before signalling Sync the operator will have entered the data on the Data Buss using the switches.

Similarly a program may Write to device 000 octal and the execution will be suspended with the data visible in the data lights until the operator signals acknowledgement by sending Sync from the HOP.

When the processor is Halted by the HOP the Halt Acknowledge light will light and the operator may Read or Write data from/to any device on the system including registers, accumulators with arithmetic operations, and memory. This feature demonstrates two very important points about the system architecture.

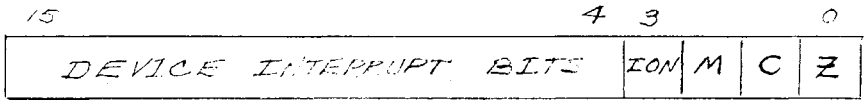
The first point is that operation of system devices and functions is OBSERVABLE on the busses. This is helpful when experimental interfaces are being integrated into the system and when problems occur.

The second point is that the HOP is heirarchally similar to a very general class of DMA device. When the processor is halted any intelligent agent may assume complete control of the entire system. This relationship enables a very natural system expandability by configuring multiprocessors under a similar controlling processor.

PSW Device Address 001 (octal)

The Processor Status Word is a special IO function implemented as Device 001 octal. When a Source field of 001 octal is included in a machine instruction the data acquired during the source-read operation will include the ZERO, CARRY, and MINUS flags; the state of the INTERRUPT CONTROL FLIP-FLOP; and up to twelve bits from potentially interrupting devices. A Read of Device 001 octal activates a back-plane line called Acknowledge (ACK). Interrupting devices strobe their identification bits onto the Data Buss upon receipt of the ACK signal.Thus the complete state of the machine is picked up on a single source-read operation.

The form of the Processor Status Word is as follows:



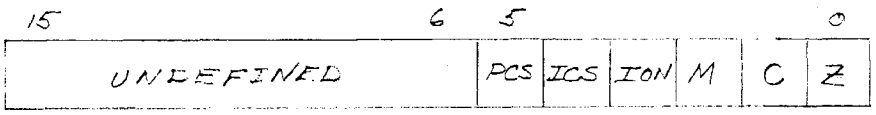
INTERRUPT ASSIGNMENTS FOR PSW

BIT#	FUNCTION/DEVICE
0	Z FLAG
1	C FLAG
2	M FLAG
3	ION bit
4	D43 Real Time Clock
5	D40 Modem Controller
6	D50 Control Terminal Interface
7	D16 Scientific Timer
8	D03 Real Time Clock
9	Undefined
10	
11	
12	
13	
14	User Definable
15	" "

PCW Device Address 001 (octal)

The Processor Control Word is implemented as Destination IO Device 001 octal. By writing data to the PCW on the destination-write operation the programmer can set or clear the ZERO, CARRY, and MINUS flags; enable or disable interrupts; and control which program counter (R17 or R16 octal) is to be used.

The format of the PCW is as follows:



The Z,C, and M bits of the PCW are copied into the flag register upon execution of the Write to Device 001 octal.

The ION bit is effective in setting or clearing the Interrupt Control Flip-Flop but ONLY when the ICS bit is a one. This allows use of the PCW destination under circumstances where the programmer may not actually know whether interrupts are supposed to be on or off. With the ICS bit equal to one, a one to the ION bit will enable interrupts and a zero will disable them.If the ICS bit is zero no change will take place in the condition of the ICFF.

The PCS bit is used to control which program counter is to be used. The use of the PCW destination requires at all times the knowledge of which PC is appropriate. There is no way to avoid this specification. When PCS is one the program counter will be R17 octal; when PCS is zero the program counter will be R16 octal.

The programmer should note well that invoking the use of R16 octal as program counter automatically disables interrupts. (A read of the PSW when using R16 octal as program counter will produce ION=0.)

The program counter is set to be R17 octal and interrupts are disabled under initial conditions; i.e. after the INIT control line has been activated.

Appendix II.

Interfacing Guide to the NEDCO

ABLE Series Computers and the EZbuss

Introduction

NEDCO has developed its ABLE Series computers around a simple and straightforward buss design called the EZbuss. The EZbuss allows simple interfaces to be designed to work efficiently with the Able Series computers. It is entirely TTL compatible; no voltages greater than +5VDC exist on the buss. Thus the longevity of the system is enhanced even in the presence of untested user-built interface devices.

Definition

EZbuss is defined for a 72-pin two-surface edge connector similar to Cinch-Jones 50-72C-20. The complete definition is shown in Table 1.

A) $\overline{BDB\emptyset}$ - $\overline{BDB15}$ the main 16 bit bi-directional data buss.

B) $\overline{ADR\emptyset}$ - $\overline{ADR7}$ 8 bit I/O address buss allows addressing many I/O modules by computer. See address map (Table 2) for available addresses.

C) $\overline{RD}$, $\overline{WR}$ command lines from computer give the directionality of the data transfer.

$\overline{RD}$: data from device to computer.

$\overline{WR}$: data from computer to device.

D) $\overline{SYN}$ SYNC line from devices to computer signals completion of task.

E) $\overline{IORST}$ I/O reset used to establish initial conditions of devices.

All of the above busses are negative-logic busses; i.e. a logical one corresponds to zero volts and a logical zero corresponds to +5 volts. Typical circuits for driving and receiving the busses will be shown in the examples section of this document.

System Timing

In order to use the busses the interface designer must understand a simple set of timing rules. These rules are given separately for the

Table 1 NEW ENGLAND DIGITAL Corp. EZbuss

Pin	Function	Slots	Pull-ups	Pin	Function	Slots	Pull-ups
72	+5 Volts	1-32		71	+5 Volts	1-32	
70	$\overline{PDS}$	1-32	390 ohms	69	1mHz.	1-32	on card
68	$\overline{WRC}$	1-32	390 ohms	67	$\overline{MAIS}$	1-32	390 ohms
66	DAISY with 65			65	DAISY with 66		
64	$\overline{z}$	9-10	390 ohms	63	$\overline{BDB\emptyset}$	1-32	330 ohms
62	$\overline{c}$	9-10	390 ohms	61	$\overline{BDB1}$	1-32	330 ohms
60	$\overline{MIN}$	9-10	390 ohms	59	$\overline{BDB2}$	1-32	330 ohms
58	MAE	1-32	390 ohms	57	$\overline{BDB3}$	1-32	330 ohms
56	special TTY			55	$\overline{BDB4}$	1-32	330 ohms
54	special TTY			53	$\overline{BDB5}$	1-32	330 ohms
52	$\overline{RST}$	1-32	390 ohms	51	$\overline{BDB6}$	1-32	330 ohms
50	$\overline{LD}$	1-32	390 ohms	49	$\overline{BDB7}$	1-32	330 ohms
48	GROUND	1-32		47	$\overline{BDB8}$	1-32	330 ohms
46	$\overline{MA\emptyset}$	1-12	330 ohms	45	$\overline{BDB9}$	1-32	330 ohms
44	$\overline{MA1}$	1-12	330 ohms	43	$\overline{BDB10}$	1-32	330 ohms
42	$\overline{MA2}$	1-12	330 ohms	41	$\overline{BDB11}$	1-32	330 ohms
40	$\overline{MA3}$	1-12	330 ohms	39	$\overline{BDB12}$	1-32	330 ohms
38	$\overline{MA4}$	1-12	330 ohms	37	$\overline{BDB13}$	1-32	330 ohms
36	$\overline{MA5}$	1-12	330 ohms	35	$\overline{BDB14}$	1-32	330 ohms
34	$\overline{MA6}$	1-12	330 ohms	33	$\overline{BDB15}$	1-32	330 ohms
32	$\overline{MA7}$	1-12	330 ohms	31	GROUND	1-32	
30	$\overline{MA8}$	1-12	330 ohms	29	$\overline{ADR7}$	1-32	3 1K's
28	$\overline{MA9}$	1-12	330 ohms	27	$\overline{ADR6}$	1-32	3 1K's
26	$\overline{MA10}$	1-12	330 ohms	25	$\overline{ADR5}$	1-32	3 1K's
24	$\overline{MA11}$	1-12	330 ohms	23	$\overline{ADR4}$	1-32	3 1K's
22	$\overline{MA12}$	1-12	330 ohms	21	$\overline{ADR3}$	1-32	3 1K's
20	$\overline{MA13}$	1-12	330 ohms	19	$\overline{ADR2}$	1-32	3 1K's
18	$\overline{MA14}$	1-12	330 ohms	17	$\overline{ADR1}$	1-32	3 1K's
16	$\overline{MA15}$	1-12	330 ohms	15	$\overline{ADR\emptyset}$	1-32	3 1K's
14	GROUND	1-32		13	$\overline{ACK}$	1-32	330 ohms
12	$\overline{MEM}$	1-12	390 ohms	11	$\overline{RD}$	1-32	2 270's
10	$\overline{RES}$	1-32	390 ohms	9	$\overline{WR}$	1-32	2 270's
8	$\overline{INIT}$ tied to 7			7	$\overline{IORST}$ tied to 8		390 ohms
6	$\overline{HALT}$	1-32	390 ohms	5	$\overline{SYN}$	1-32	3 390's
4	$\overline{HAK}$	1-32	390 ohms	3	$\overline{IRQ}$	1-32	390 ohms
2	GROUND	1-32		1	GROUND	1-32	

Read Operation and the Write Operation. (See also Figure 1.)

Read Operation

When the $\overline{RD}$ line goes true the interface may assume that the $\overline{ADR\emptyset}$ - $\overline{ADR7}$ address busses have been valid for a period greater than 50 nanoseconds. When the device is able to produce valid data it may gate that data onto $\overline{BDB\emptyset}$ - $\overline{BDB15}$ any time after having decoded its (address and $\overline{RD}$).

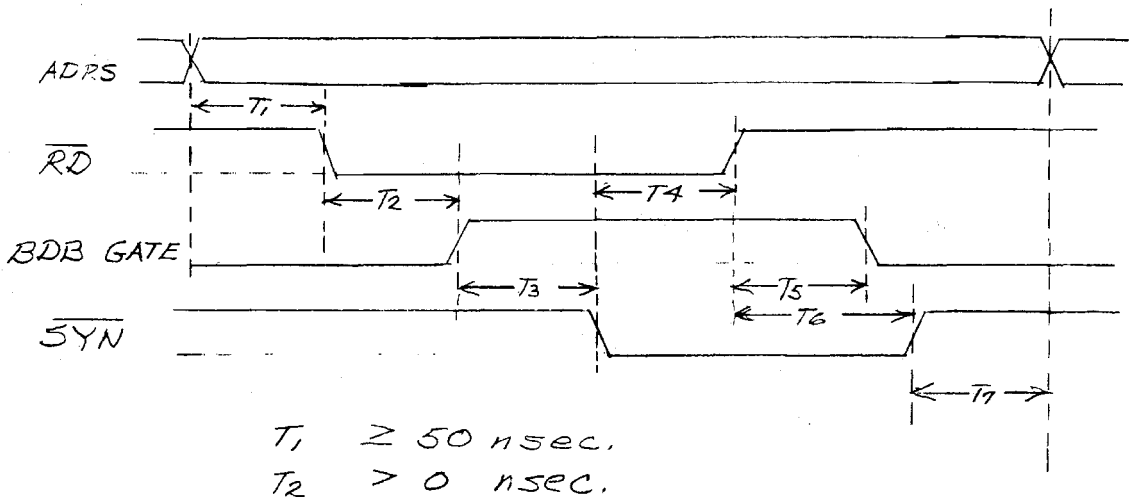
The interface must wait a minimum of 150 nanoseconds after gating (BDB GATE) valid data onto the data buss and then signal SYNC or $\overline{SYN}$. It must remain in this state; i.e. with data gated and SYNC active; until $\overline{RD}$ goes false. It should then remove the data and SYNC. Removal of SYNC may not precede removal of data. Whether data may be allowed to change while being gated must be decided on a case-by-case basis. The address lines will not change until SYNC has been removed.

Write Operation

The $\overline{WR}$ line will go true not earlier than 50 nanoseconds after the data and address have become valid. When the device decodes (address and $\overline{WR}$) it may use the data at any time. When the device has no further use for the data it may signal SYNC. It must retain SYNC until $\overline{WR}$ becomes false. It then must release SYNC. The address lines will not change until after SYNC has gone; but the data may change any time after SYNC is made true.

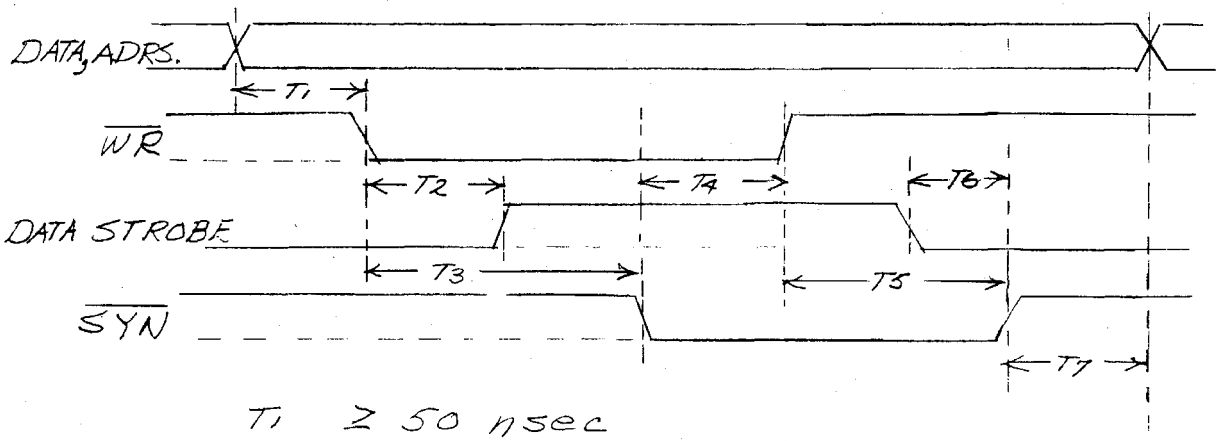
FIG. 1 SYSTEM TIMING, ABLE SERIES EZBUSS

A) READ OPERATION



$T_1 \geq 50$ nsec.
$T_2 > 0$ nsec.
$T_3 \geq 150$ nsec.
$T_4 \sim 150$ nsec.
$T_5 > 0$ nsec.
$T_6 \geq T_5$
$T_7 > 0$ nsec.

B) WRITE OPERATION



$T_1 \geq 50$ nsec
$T_2 > 0$ nsec
$T_3 > T_2$
$T_4 \sim 150$ nsec.
$T_5 > 0$ nsec.
T_6 MUST BE REMOVED BEFORE SYNC REMOVED.
$T_7 > 0$ nsec.

Design of a Simple Interface

Every interface must have an address decoding circuit. This circuit will in general provide an output which will signal that the computer is trying to read or write a given device number. (The choice of device number(s) for user constructed interfaces should be made with reference to Table 2.)

Some examples of address decoding circuits are given in Figure 2. Notice that in general only one low-power Schottky load need be presented to every buss line. Also notice that the $\overline{RD}$ and $\overline{WR}$ control lines effectively deskew the address lines.

If the interface is to respond to the $\overline{RD}$ command the decoded ($\overline{RD}$ and DEV#) signal may be used to gate the data onto the BDB buss. Recommended buss drivers are 7438, 74367, 74368 or the 74LS240 series. After gating the data onto the buss the interface must wait a minimum of 150 nanoseconds and then signal SYNC. This delay is to allow the busses, drivers and receivers to settle. A circuit which might be added to the decoder of Figure 2A to read four bits of data from device "140" is shown in Figure 3.

If the device is to respond to the WR command, it may construct an appropriate strobe signal from the decoded ($\overline{WR}$ and DEV#) signal. It then waits a few hundred nanoseconds and signals SYNC. A circuit to copy four bits of data from $\overline{BDB\emptyset}$ - $\overline{BDB3}$ is shown in Figure 4. Note that the $\overline{SYN}$ buss must be driven by 7438.

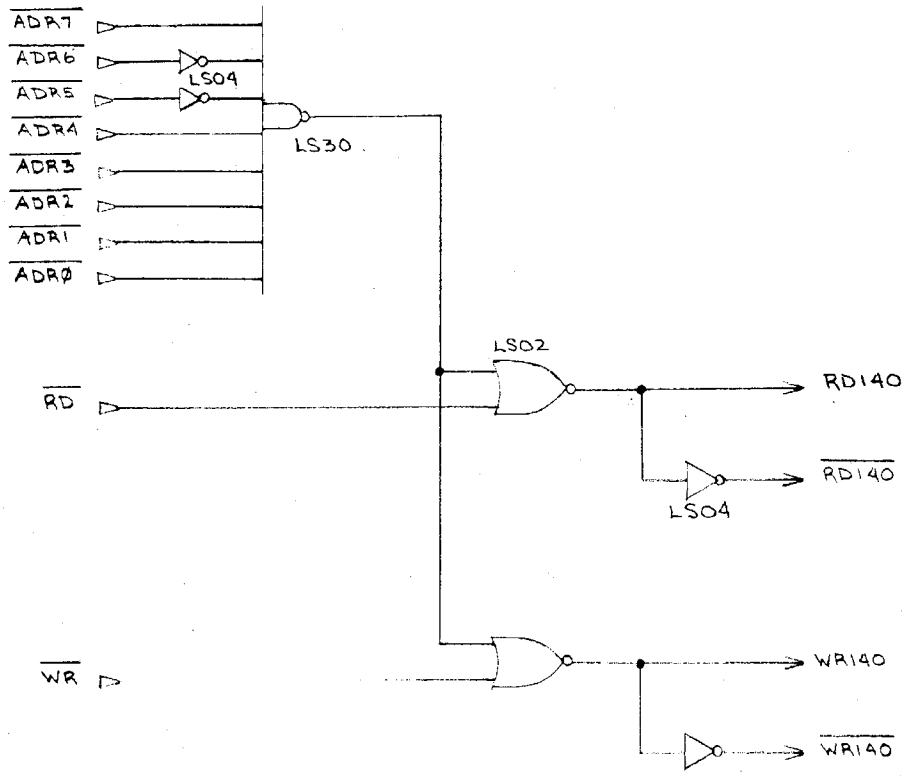


Figure 2A

Circuit to decode READ ("140") and WRITE ("140") in both Negative and Positive Logic.

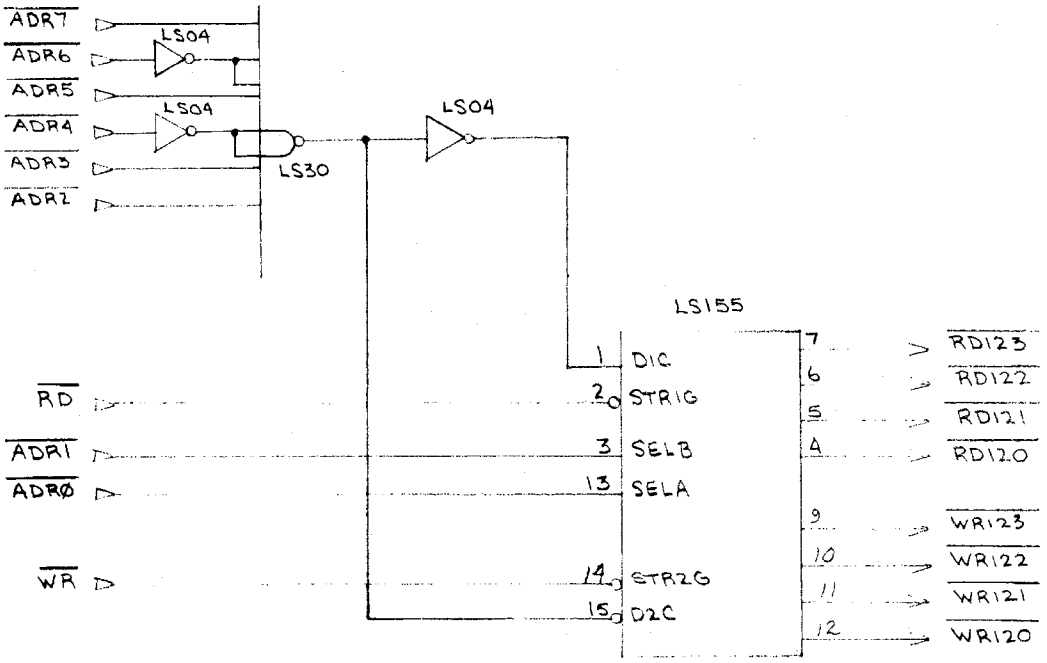


Figure 2B

Circuit to decode READ and WRITE of devices "120", "121", "122" and "123".

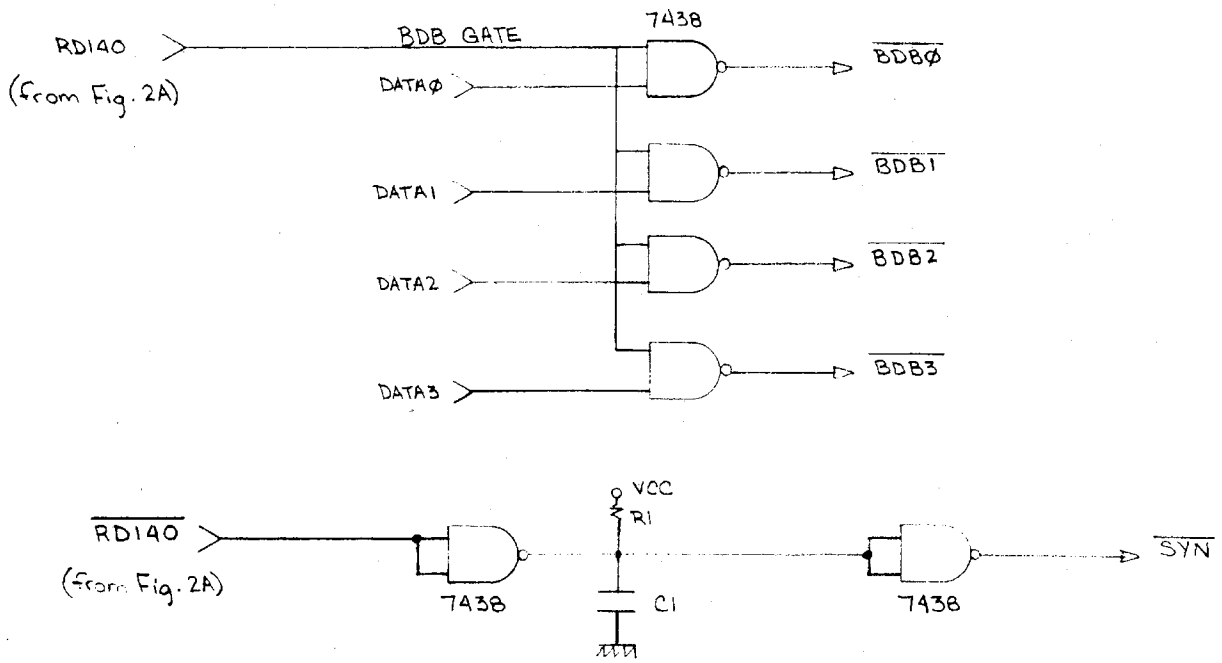


Figure 3

Circuit to GATE 4 Bits of Data onto
BDB Buss and Signal SYNC.

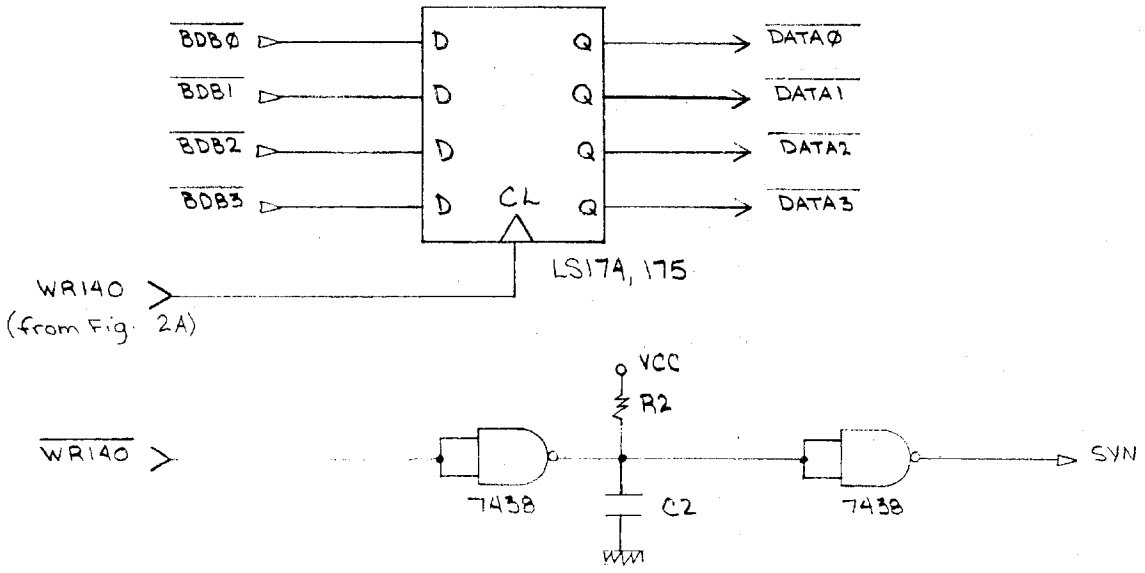


Figure 4

Circuit to Copy 4 Bits of Data from
BDB Buss and Signal SYNC.

Mechanical Details

Interfaces may be constructed using either:

a) Vector 3719-1 6.5" x 4.5" or

b) Vector 3719-4 9.6" x 4.5"

These are blank cards with holes every 0.1" in a square grid; they have 72 edge fingers spaced 0.1", 36 on each side.

These cards use a receptacle similar to TRW Cinch 50-72C-30 for conventional wire-wrap or for insertion into a PC backplane.

Receptacles are mounted with pins 1 and 2 on bottom; pins 71 and 72 are on top.

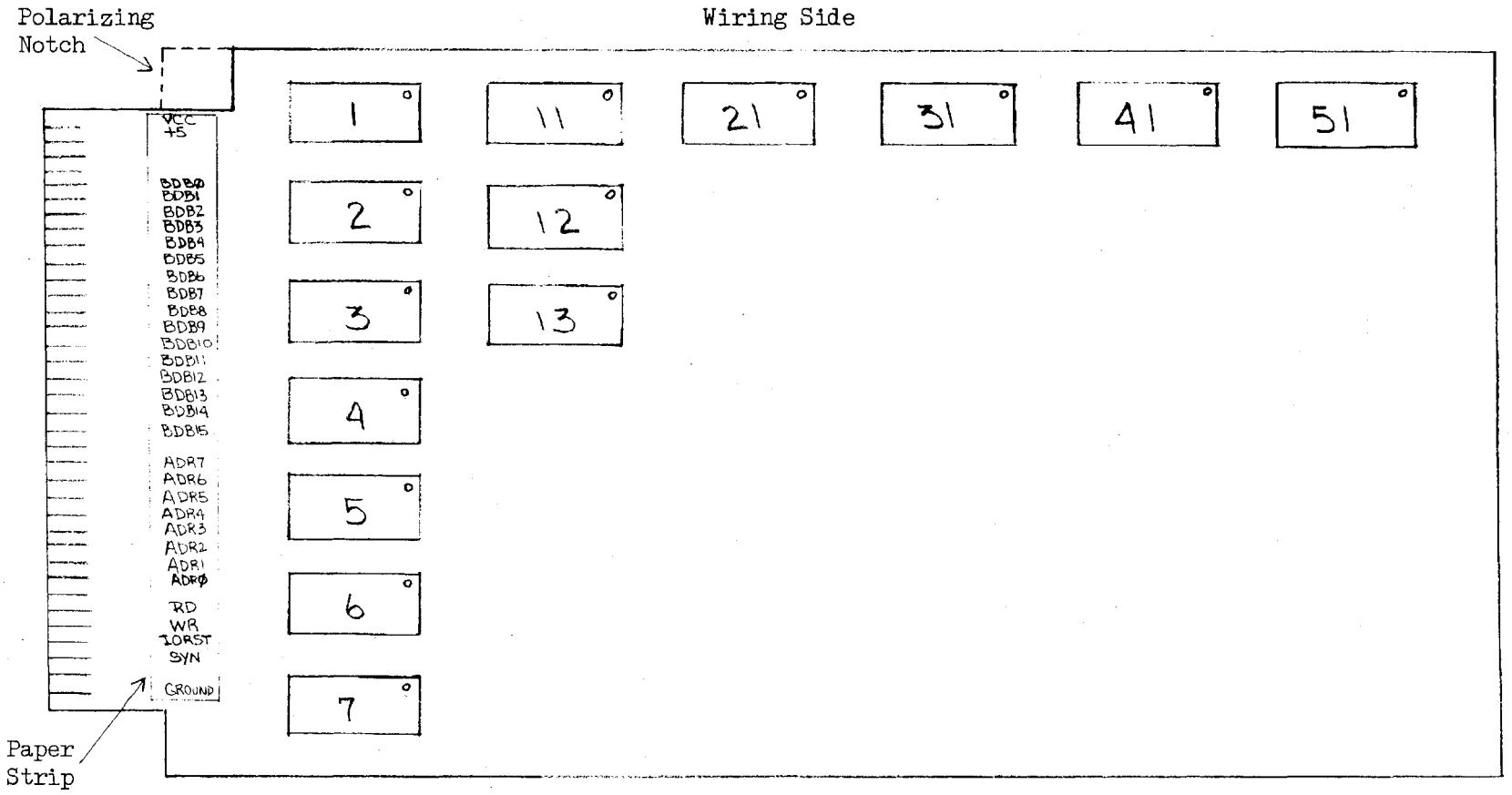
Cards are constructed with:

a) COMPONENTS on the EVEN numbered side.

b) WIRING on the ODD numbered side.

Note that Table 1 is shown as a REAR VIEW of the connector. Also carefully note that the Vector Card must be polarized by cutting a square (approximately 0.4" x 0.4") relief out of the top edge of the card near the contact fingers. (Figure 5) If these measures are implemented, finished cards will always plug into the Bellvue Rack with components on the right and wiring on the left (as viewed from front of rack). This uniformity is a great aid to achieving interface longevity.

Note also the numbering scheme (Figure 5) employed for the IC positions on the card. The IC's along the top of the card are numbered 1, 11, 21, 31 etc. starting nearest the contact fingers. The IC's in the vertical column nearest the contact fingers are numbered 1, 2, 3, etc. starting at the top.



Socket Numbers applied using gummed labels; red dots used to mark Vcc pin.

Connector Assignments Labelled.

Polarization notch - upper left corner.

Figure 5

Appendix III.

TIMING INFORMATION

for the

NEDCO ABLE Series COMPUTERS

ABLE SYSTEM TIMING

The ABLE Series computers are ASYNCHRONOUS machines. That is, there is no specific system clock. The SEQUENCER begins a data transfer by asserting a Device Address on the system Instruction/Address Buss. After waiting approximately 150 nsec. (for the Model ABLE /80) the Sequencer asserts a Command, either READ or WRITE. The system stays in this state until the addressed device or operation answers by means of the system SYNC Control line that the transfer has been completed. At this time the Sequencer removes the command and waits for the device to remove its SYNC signal. Only after the addressed device has has taken away its SYNC does the Address asserted by the Sequencer change and a new transfer begin. Thus the time taken by a given transfer is dependant upon the module addressed.

In addition, all ABLE Series devices which can be said to have a state of "busy" or "not ready" are constructed to withold SYNC until such time that they become ready or free in the event that they are prematurely accessed by the Sequencer.

This mode of operation enables one to write very simple programs for handling such devices in the event that Status Checking or Interrupt Driven programs are not appropriate. Also such operation of certain devices in sampling programs provides a useful technique of avoidance of any sampling

"Jitter" due to the asynchrony of the computer program with the sampled process. See for example the discussion of the Terminal (D50) and Scientific Timer (D16) programming in Appendix I.

It is due directly to this asynchronous feature that the ABLE Series computers can operate effectively with memory of any speed which is appropriate for a given system. There is no problem in utilizing the fastest available Schottky bipolar memory or the many types of slower MOS memory , even in the same system for the memory operation is asynchronous.

The standard M8K Memory Module is composed of MOS 2114 type chips having an access time of 450 nsec.This chip, however, has an enable time of only 100 nsec. (approx.). The asynchronous operation of the machine allows the construction of an 8Kx16-bit memory matrix in which 8 separate 16-bit words having sequential locations are accessed at once. This access takes 450 nsec. However, once accessed the data is available by selecting the desired chips in only 100 nsec. Whether the current address was accessed previously is signalled by a New/Old flag driven by a comparison of the current address and the previous stored address.

This technique is called 8-way interleaving. Each 8K card has its own interleaving hardware which is actuated only if that card has been addressed. Thus in a system with several

8K cards the effective interleaving can be higher than 8-ways. This can be used advantageously in problems which involve the moving of large vectors in memory by having the vector boundaries coincide with 8K boundaries of the memory space. XPL allows the partitioning of data storage areas to accomodate this type of operation.

With the 450 nsec. 2114 type chip the memory performance figures are as follows:

New access read.....950 nsce.

Old access read.....550 nsec.

Write new or old....950 nsec.

The timing of Resister Module operations is approximately 250 nsec. regardless of the type of operation being performed.

Timings to be observed by user-constructed modules are given in Appendix II. The IO module timings of standard devices follow these rules.

Appendix IV.

NEDCO ABLE SERIES

DEVICE ADDRESSES

DEVICE ADDRESSES USED IN STANDARD MODULES .

The followins are device addresses assisned or to be assisned for use in NEDCO ABLE Series computers.

Dev. ADDRESS	Function
000	HOP
001	PSW,FCW
002	Reserved
003	Real Time Clock
004-011	Reserved
012,013	A/D Converter
014,015	Aux. A/D Converter
016-033	Reserved
034,035	Disital IO Module
036,037	Aux. Disital IO
040,041	Modem Controller,Status
042-047	Aux. Serial Devices
050,051	System Control Term.,Status
052-057	Aux. Serial Devices
060,061	Dual D/A Converter
062-067	Aux. D/A Converters
070,071	Scope Interface
072,073	Aux. Scope Interfaces
074-077	Reserved
100-104	Floppy Disk Controller
105-107	Tape Drive
110-117	Disk Drive
120-127	FREE USER SPACE
130-133	Reserved
134-137	Disital IO
140-157	Aux. Serial Devices
160-167	Reserved
170-177	PROM Prosrammers, System Test Modules, etc.