Control of an analog-to-digital converter by a digital computer

Nelson, R.J.

Publisher's version / Version de l'éditeur:
https://doi.org/10.4224/21275319
Report (National Research Council of Canada. Radio and Electrical Engineering Division : ERB), 1967-04

ERB-756

UNCLASSIFIED

NATIONAL RESEARCH COUNCIL OF CANADA

RADIO AND ELECTRICAL ENGINEERING DIVISION

CONTROL OF AN ANALOG-TO-DIGITAL CONVERTER BY A DIGITAL COMPUTER

R. J. NELSON

OTTAWA

APRIL 1967

ABSTRACT

A detailed step-by-step description of one set of solutions to the engineering problems associated with connecting a piece of non-standard peripheral equipment to a small fast data processor is given. The peripheral unit is a twelve-bit-plus-sign analog-to-digital converter with a programed multiplexer. The data processor is a Systems Engineering Laboratories computer with a 24-bit word length and 1.75-μs cycle time. The software or 'programing' aspect of the interconnection is also discussed.

This report was prepared by a student assistant and describes the work done during his intersemester employment at the Radio and Electrical Engineering Division, National Research Council.

TABLE OF CONTENTS

Table of Contents (cont'd)

FIGURES

CONTROL OF AN ANALOG-TO-DIGITAL CONVERTER BY A DIGITAL COMPUTER

-R.J. Nelson -

INTRODUCTION

For a theory to be accepted, it must be proved experimentally or observationally. This usually involves numerous experiments, each with its own data. One of the scientist's biggest problems is sampling and analyzing quickly the data available. In many cases, the time available per sample is in the order of micro-seconds. Electronic samplers and digital computers are being widely used to handle the data.

This report describes hardware and software needed to digitize analog data and feed it into a computer for analysis. The analog signal is converted to a digital signal by an Adage analog-to-digital converter and the binary output is analyzed by an SEL 840A high-speed digital computer. The computer has full control over the converter; i.e., the computer can be programed (instructed) to sample for a specified number of samples at a predetermined rate. This project deals with the necessary hardware development and soft-ware development. The former refers to the electronic interface between the converter and the computer. This involves controls, storage registers, and level changers (to make converter and computer electronics compatible). Software involves programing the computer to command the converter with the instructions the operator wishes, and programing the computer to store the data from the converter and to control the converter.

LOGIC

A decimal number has the base 10 while a binary number has the base 2. Only 0 and 1 are used in a binary number and its digits correspond to $2^n \ldots 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0 .\ 2^0\ 2^1\ 2^2 \ldots$ For example, the binary number 1101 corresponds to the analog number $(1 \times 2^0) + (0 \times 2^1) + (1 \times 2^2) + (1 \times 2^3) = 13$. Likewise, in the decimal system, $125 = (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$ or corresponds to the binary number 1111101.

The 1 and 0 are called states and are represented by certain electrical voltage levels. The SEL 840A computer uses positive logic with voltage levels of +3 volts and 0 volts meaning a '1' state and '0' state, respectively. The adage analog-to-digital (A/D) converter uses positive logic with voltage levels +2.5 and -2.5 volts. Positive logic means that the higher voltage level refers to the '1' state while the lower voltage level refers to the '0' state.

Control circuits and storage buffers are added between the converter and computer. These are constructed using DEC Flip Chip* modules in which 0 volts corresponds to a '0' state and -3 volts to the '1' state (negative logic).

DEC LOGIC CIRCUITS

DEC logic can be driven by pulses (minimum duration 100 nanoseconds (ns)) and levels (a signal remaining constant for at least 400 ns). There are several basic hardware modules used in the control unit.

---

*Registered trademark of Digital Equipment Corporation, Maynard, Massachusetts.

INVERTER

An inverter complements a level, i.e., changes its state. If the inverter has multiple inputs, it is called a diode gate. In this form it is operating as a NAND gate for negative inputs and as a NOR gate for ground inputs. A NAND gate inverts the product of the input states while a NOR gate inverts the sum of the input states. (see Figs. 1A and 1B).

DIODE-CAPACITOR-DIODE (DCD) GATE

DCD gates are used to trigger flip-flops. They have a level input, a pulse input, and a pulse output (Fig. 1C). For the DCD gate to operate, the input level must be grounded for at least 400 ns and the pulse input held at -3 volts for 100 ns. If a positive pulse or a positive-going level change (-3 to 0 V) is applied to the pulse input terminal, a positive pulse (0 to 3 volts) of 150-ns duration appears on the output terminal.

FLIP-FLOPS

If two input diode gates are connected so that the output of each becomes an input of the other, a flip-flop results (Fig. 1E).

If the direct clear input is grounded for 100 ns or more, the flip-flop is set to a '1' state; i.e., terminal 1 is set at -3 volts, terminal 0 is at 0 volts. Grounding the direct set input for 100 ns sets the flip-flop into a '0' state. A flip-flop stays in a given state until an input representing the other state is applied.

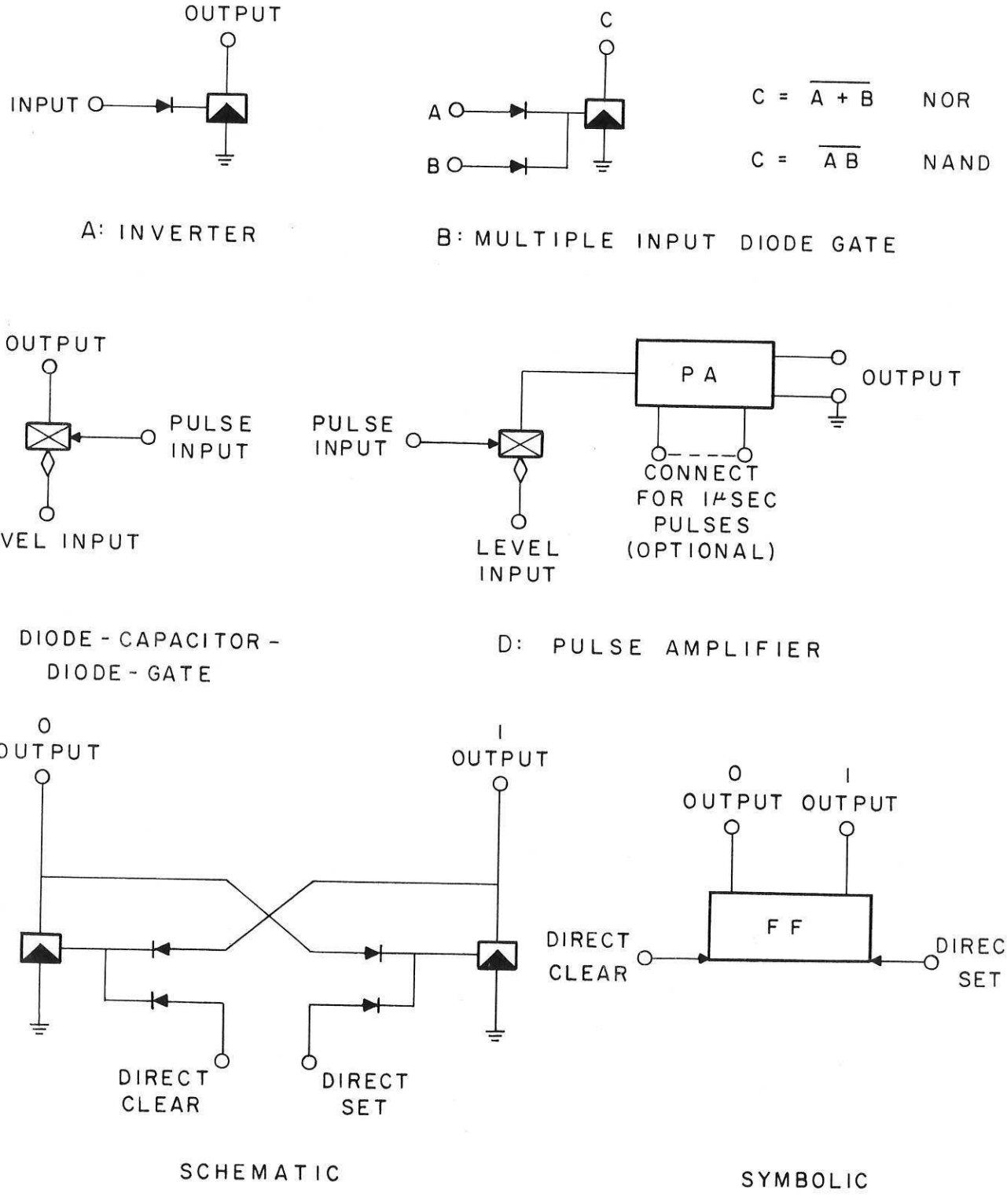


Fig. 1 Symbolic representation of DEC circuits

To increase the flexibility of flip-flops, DCD gates are added as inputs (Fig. 2A). The set and clear inputs function as before. If DCD gate 1 produces a positive pulse, the flip-flop is set in the '0' state. If DCD gate 2 produces a positive pulse, the flip-flop is set in the '1' state. If both DCD gates produce pulses, the flip-flop will complement its state.

PULSE AMPLIFIER

If a 100-ns pulse or a positive-going level change (rise time of 60-ns maximum) is fed into a pulse amplifier, a positive DEC pulse (-3 to 0 volts) appears on the output (Fig. 1D). The R603 pulse amplifier gives a 400-ns pulse.

CLOCK

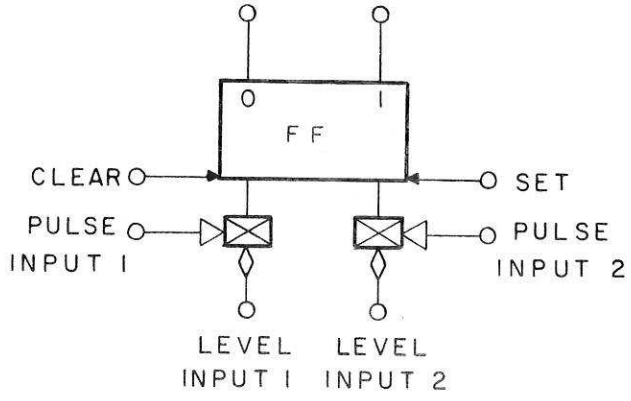
An R401 variable clock produces 100-ns pulses. Its frequency can be varied from 30 Hz to 2.0 MHz by adding resistors and capacitors. It is enabled by a -3 volt input and disabled by a ground input (see Fig. 2B).

DELAY

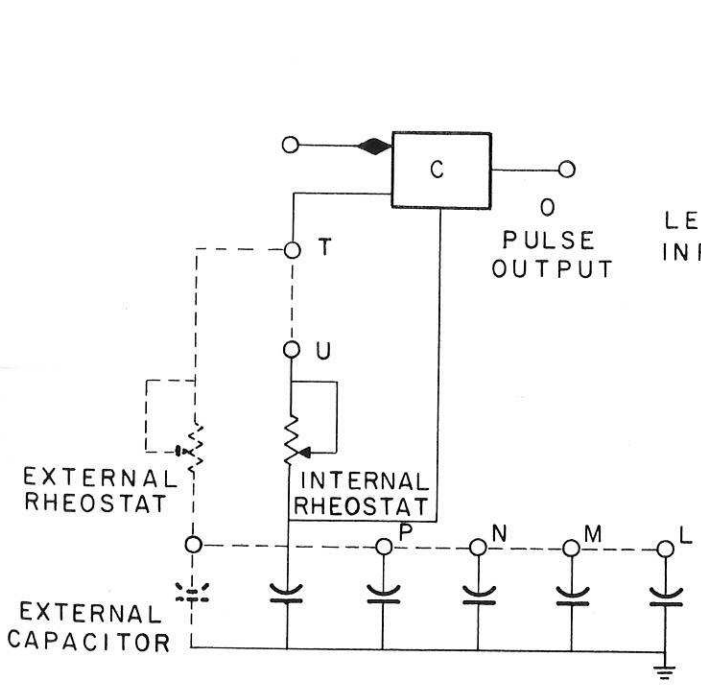
The R302 delay is triggered by a DCD gate. When it is triggered, its output changes from ground to -3 volts for an adjustable period of time, then returns to ground (see Fig. 2C).

LEVEL CHANGERS

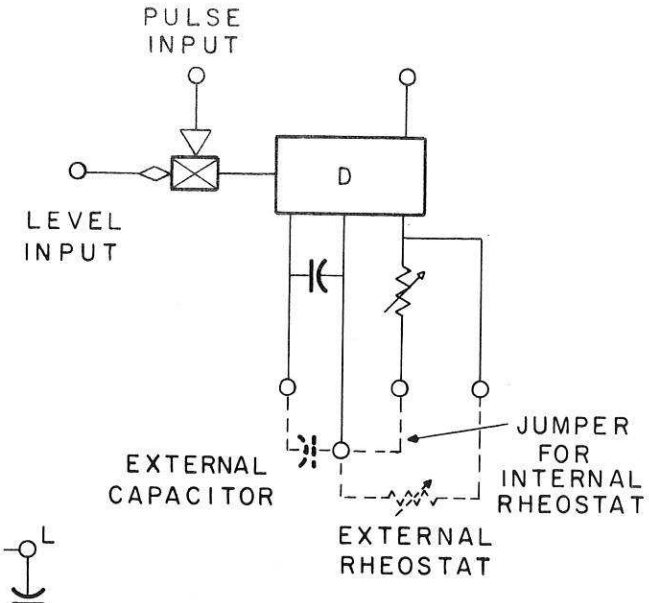
A level changer makes different logic systems compatible. For example, a W510 input converter changes Adage logic levels to DEC logic levels.



A: FLIP - FLOP WITH DCD GATES



B: R 401 VARIABLE CLOCK



C: DELAY

Fig. 2 Symbolic representation of DEC circuits

ADAGE ANALOG-TO-DIGITAL CONVERTER

The Adage A/D converter is a high-speed multiplexing converter with ten input channels. Sampling can be programed in a repetitive, random, or sequential mode. In the repetitive mode (specified by applying 2.5 volts on the repetitive line), the channel previously selected will continue to be sampled. In the random mode (specified by applying 2.5 volts on the random line), an externally selected channel is sampled. If both the random and repetitive lines are at -2.5 volts, then the sampling is done sequentially, starting with the next channel and going up to and including the channel selected by the end-of-sequence switch.

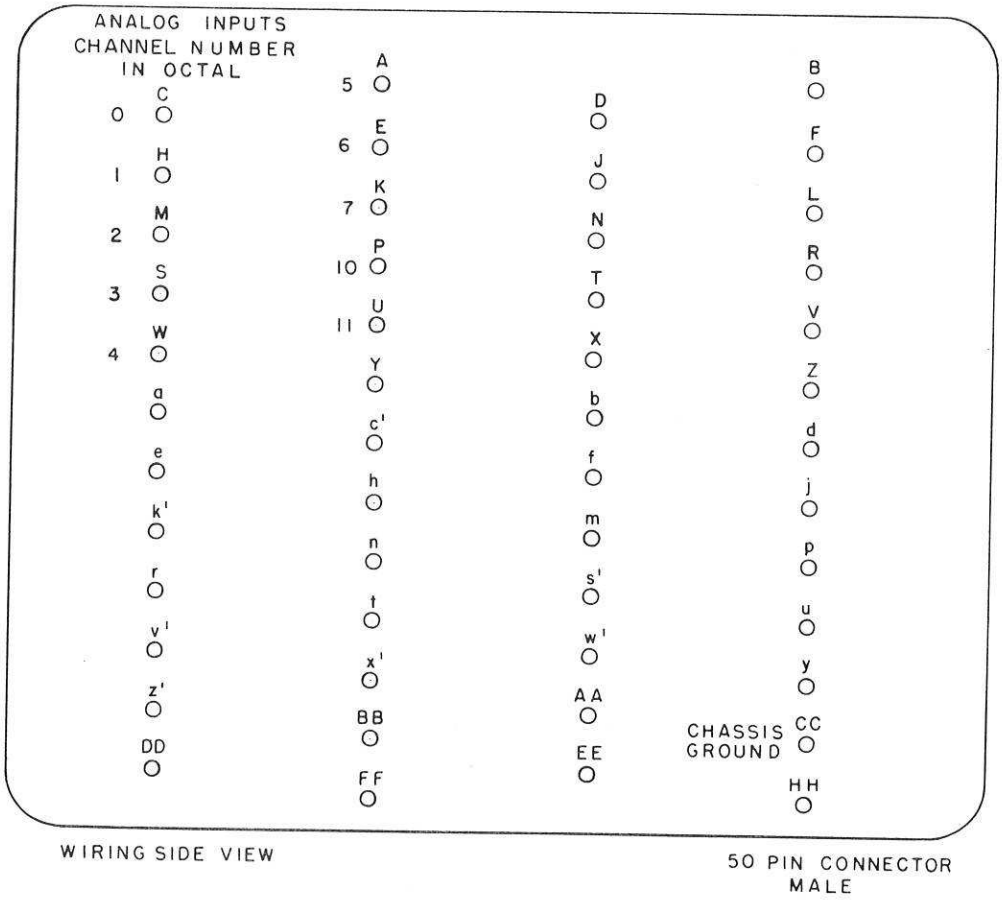


Fig. 3 Converter analog input plug J-50

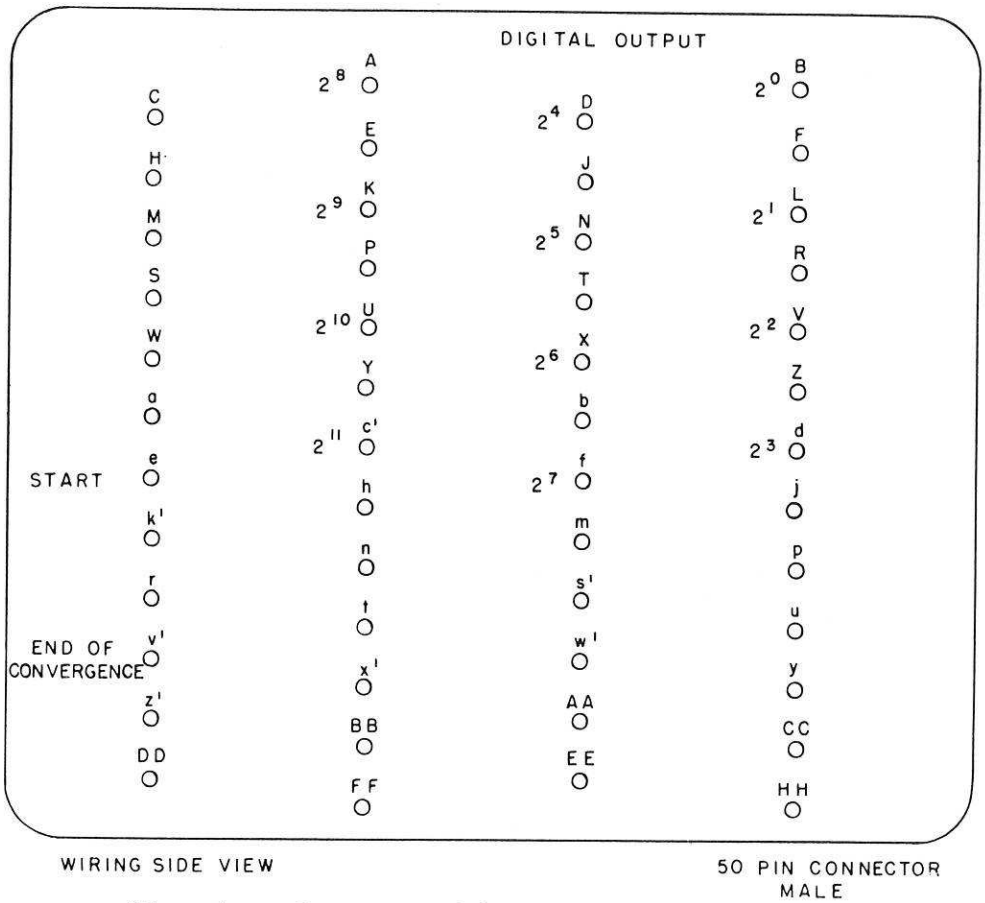


Fig. 4 Converter binary output plug J-53

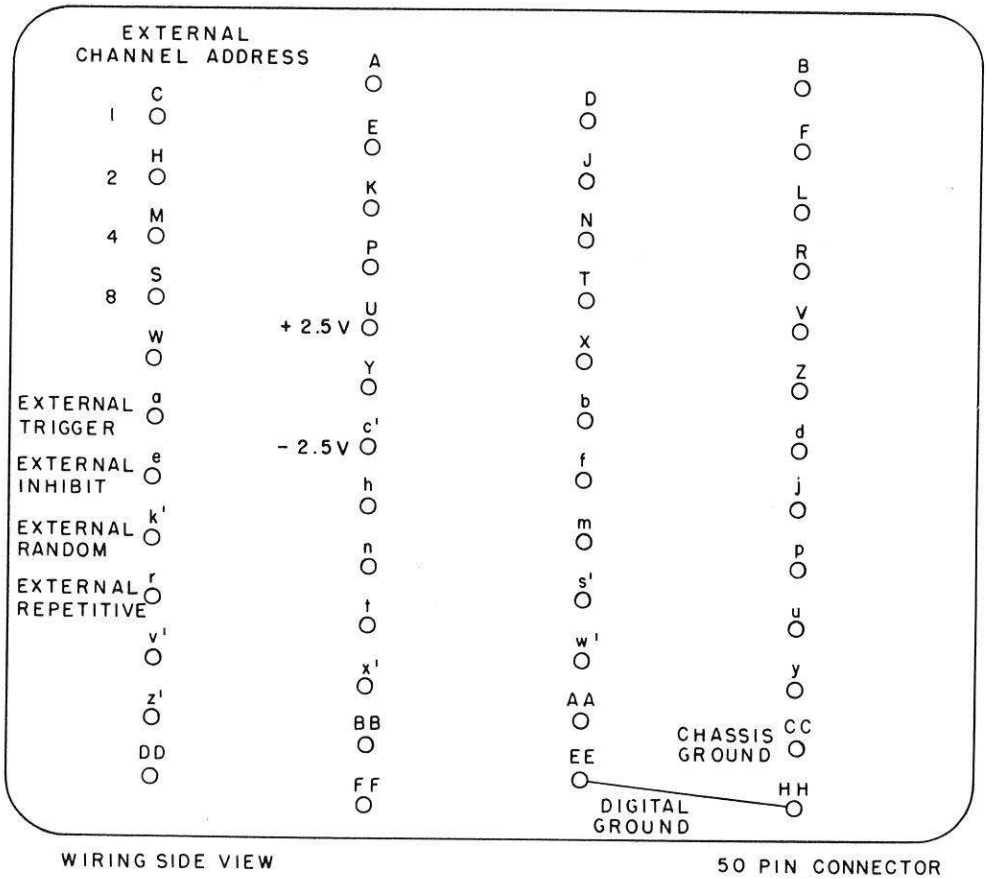


Fig. 5 Converter command input plug J-54

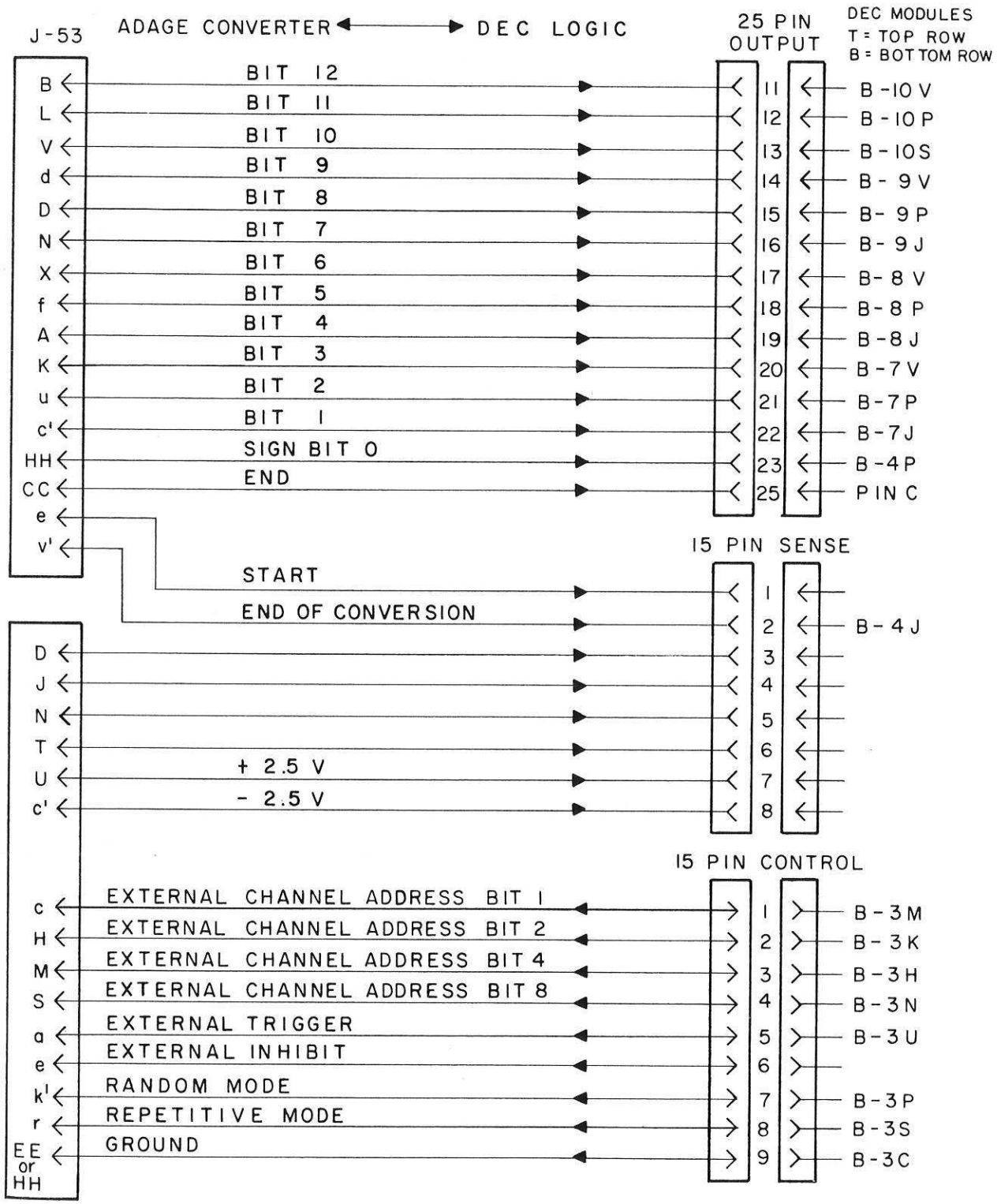


Fig. 6 Converter-DEC connections

After this channel is sampled, channel 0 is sampled and the sequence is repeated. The converter can be triggered from three separate sources. Single sampling in the sequential mode is possible with a manual push button on the converter while multiple sampling at 60 Hz in the repetitive mode is accomplished by selecting the line triggering switch. An external triggering source can be connected to the converter to control any of the three modes. The minimum time for a conversion is 50 μs. After the conversion is completed, an end-of conversion pulse is available as an output. The binary data appear on the output plug J-53 for approximately 26 μs. The wiring diagrams for the input/output plugs are shown in Figs. 3, 4, 5, 6, and 7.

A common ground for all shielding is available on the back of the converter and is called MECCA. Shielding should not be attached to any digital ground pins.

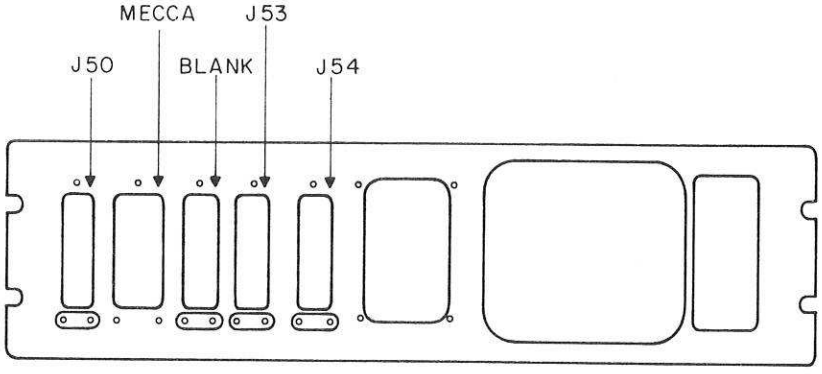


Fig. 7 Converter rear panel diagram

SEL 840A COMPUTER

The SEL 840A is a 24-bit digital computer with a memory cycle of 1.75 μs. It has an 8K memory which is expandable to 32K. A Fortran compiler and Mnembler assembler are available for program use. Input/output is handled by the unit input/output control (UI/OC) which communicates with the external data terminals by a common I/O bus. Peripheral units can communicate directly with memory or with A accumulator. Up to eight block transfer controls (BTC) can be added to provide transfer of blocks of data between the peripheral unit and the computer. A priority-interrupt system is also available. Two accumulators and three index registers are available for arithmetic and control work.

DATA TERMINAL

The data terminal provides the link between the computer and a peripheral device. Figure 8 shows the wiring logic for a data terminal and the connections which must be made with a peripheral unit. The functions of these lines and the timing of commands will be discussed in computer-converter timing.

HARDWARE DEVELOPMENT

The electronic circuitry between the data terminal and the converter performs several tasks: it must provide trigger pulses, mode selection, and, if required, external channel address selection to the converter; it must store the binary output of the converter until the computer is ready for it; it must accept commands from the computer and furnish test results; further, it must make SEL and Adage logic levels compatible.

Because of the wide selection of modules available, Digital Equipment Corporation's Flip Chip modules are used to form the control hardware. Since their logic levels are different from either those of SEL or Adage, level changers will be required at both

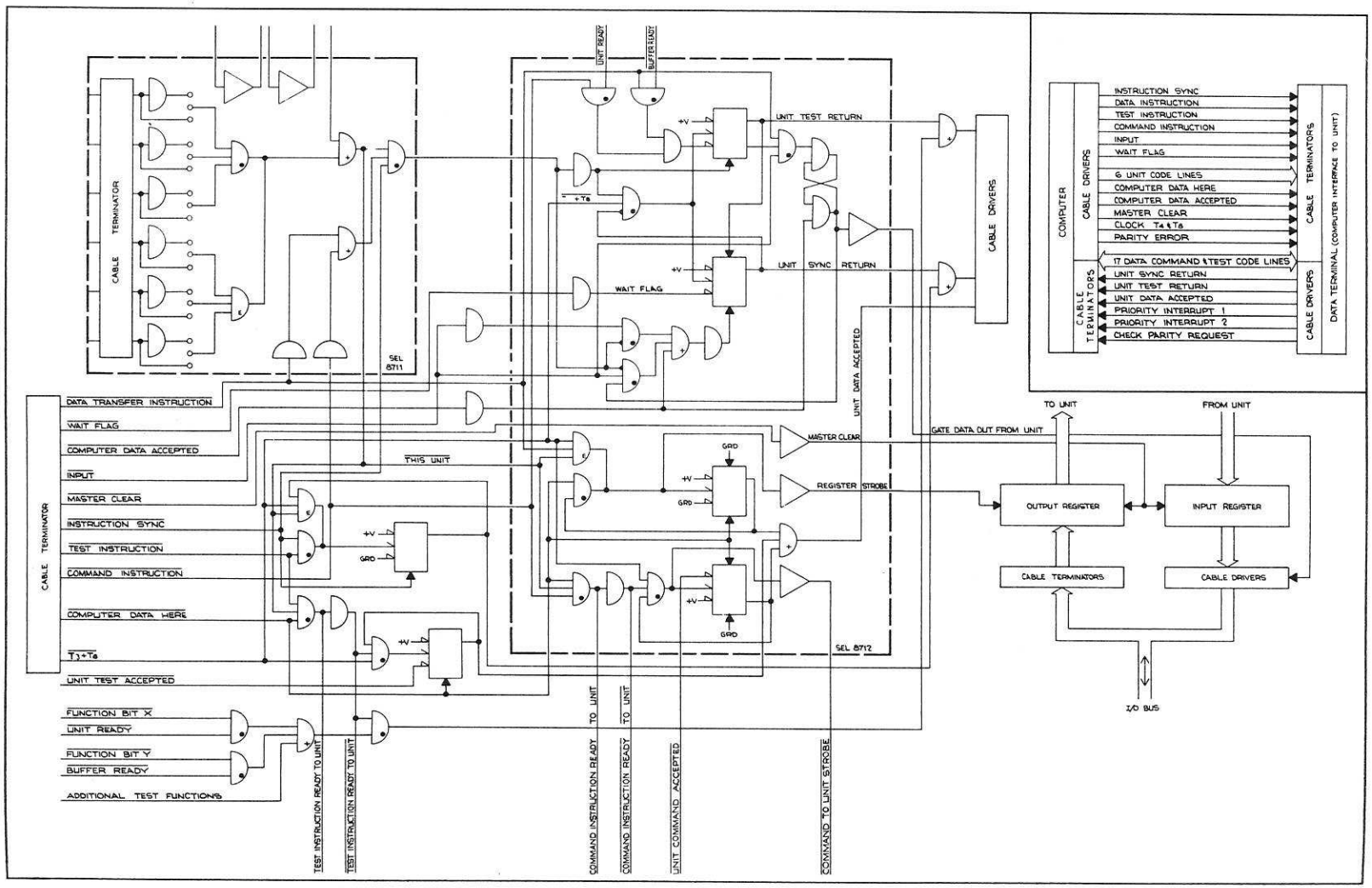


Fig. 8 SEL data terminal circuitry

ends. These modules are mounted on a mounting panel, type H900, with power supply.

TRIGGER TIMING

For flexible operation, the sampling rate of the analog input channels of the converter must be variable. In addition, continuous sampling and single sampling are needed. For continuous sampling, a variable clock (R401, 30 Hz to 2.0 MHz) is used. The clock is enabled by a -3-volt input and disabled by a ground input. It is controlled by a flip-flop (R201) and drives a pulse amplifier (W640) producing 400-ns pulses. (These long pulses are needed to activate diode gates.) Manual control of the clock is permitted by manual setting of the flip-flop. A single-pole double-throw centre-off switch is connected to the direct-set and direct-clear inputs. Computer control of the clock is possible using two of the DCD gates of the flip-flop. The computer commands are used to enable or disable the level inputs of the DCD gates while the command-to-unit strobe signal from the computer is used to trigger the gates.

Manual single sampling is done by momentarily setting a flip-flop (R204) with a single-pole double-throw switch connected to its direct-clear and direct-set inputs. The output of the flip-flop is combined (in an OR gate) with the computer-generated single-sample pulse, and the combined output drives a pulse amplifier (W640). The computer command bit specifying a single sample is combined (in an OR gate) with the command-to-unit strobe signal and the output becomes the computer-generated single-sample pulse. The output of the pulse amplifier is combined (in an OR gate) with the clock pulse output. The resulting pulse is converted to Adage levels and is used to trigger the A/D converter (Fig. 9). The DEC-to-Adage logic-level converters were designed and made at NRC.

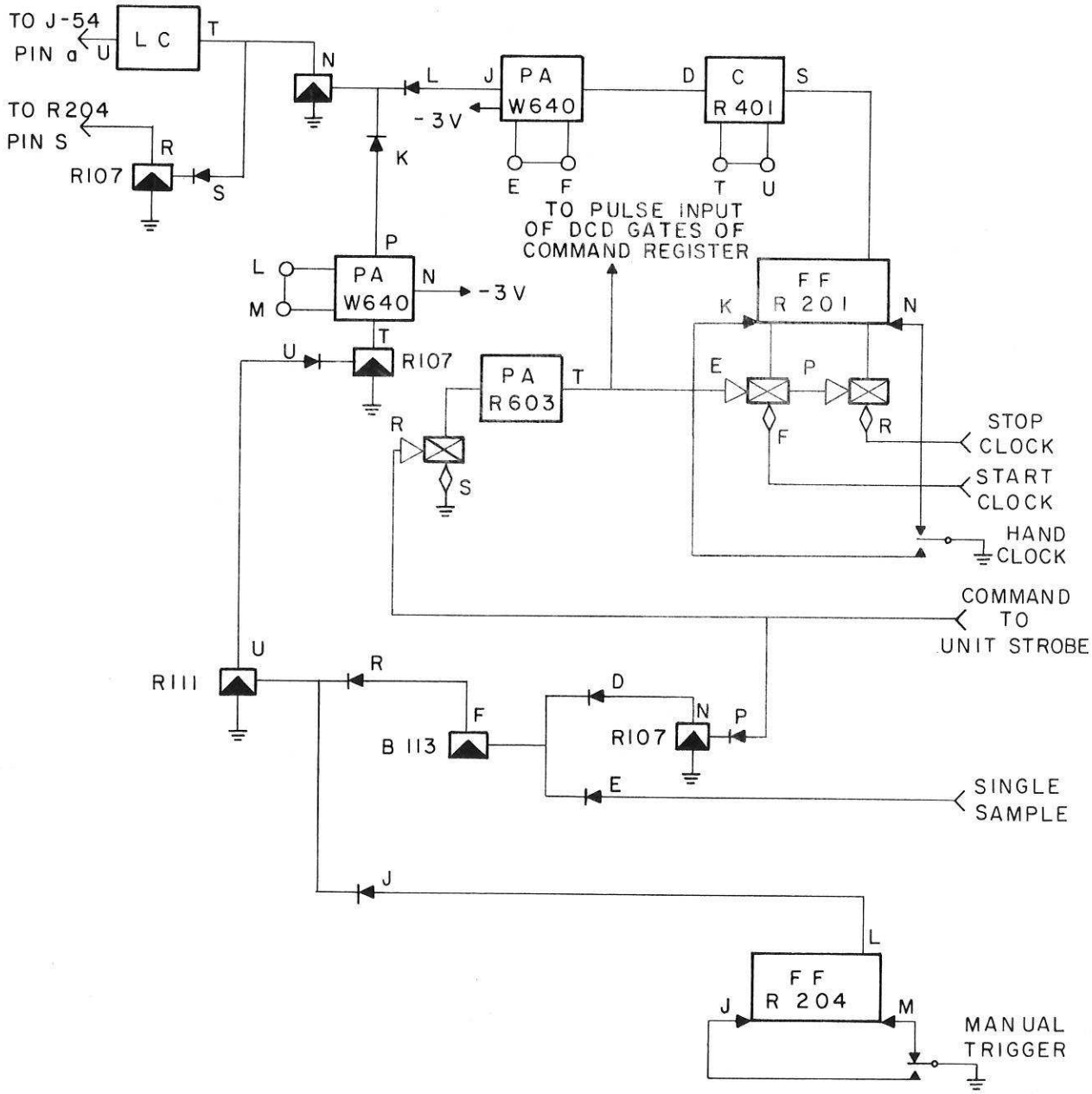


Fig. 9 Trigger timing

CLOCK SPEED

For purposes of sampling an analog input, sampling rates between 5 kHz and 20 kHz are needed. The R401 variable clock has rates between 30 Hz and 20 MHz. A multiposition switch is used to select the desired speeds. For testing purposes, the clock has two speeds, fast and slow. The multiposition switch is also used to set and clear a flip-flop depending on its position. The flip-flop is set for fast and cleared for slow clock speeds. Its output can be checked by the computer with a test-external-unit instruction.

EXTERNAL ADDRESS AND MODE SELECTION

The six bits for the external address and mode selection (inputs to the converter) are provided by the computer in the second word of the command instructions. After being changed to DEC levels, these signals are used to enable or disable the level inputs on the DCD gates of the command register (R203 flip-flops). The flip-flops are cleared using the command-instruction ready-to-unit signal from the computer. The DCD gates are pulsed with the command-to-unit strobe pulse. The outputs of the flip-flops are converted to Adage levels and sent to the converter (Fig. 10).

BINARY OUTPUTS

The 13-bit (12 data bits + sign) binary output from the Adage converter is changed to DEC levels by positive input level changers (W510). These DEC levels are used as level inputs of the DCD gates of the output register (R203 flip-flops). The end-of-conversion (ECC) pulse from the converter is changed to a DEC level pulse and used to clear the output register. It is then delayed for 1.5 μs (using an R302 delay), and is used to generate a pulse which loads the DCD gates on the output register. The outputs from this register are converted to SEL logic levels (using a W603) and go to the data terminal of the computer.

The ECC pulse that clears the output register also clears two control flip-flops (R204). One is set

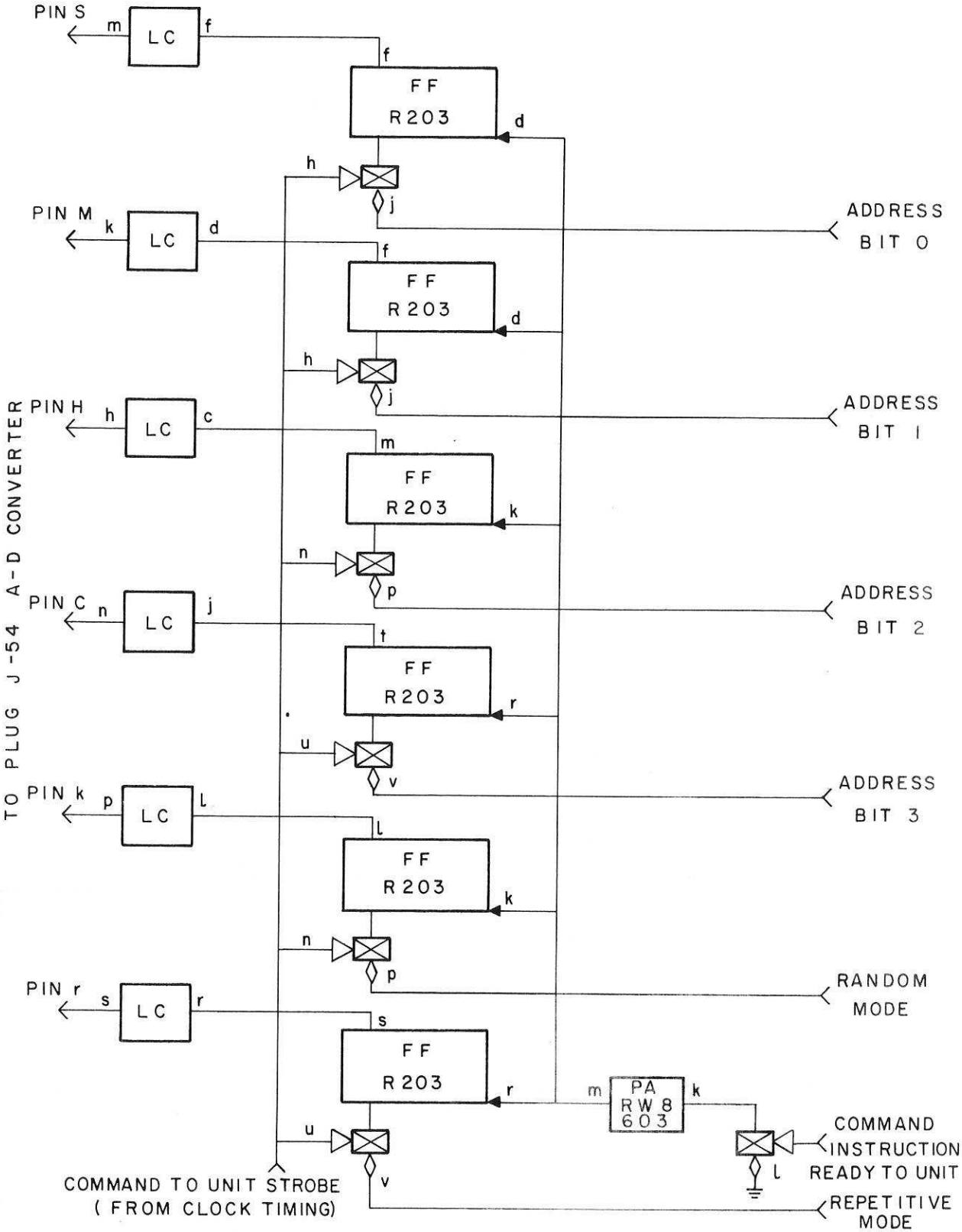


Fig. 10 Command registers

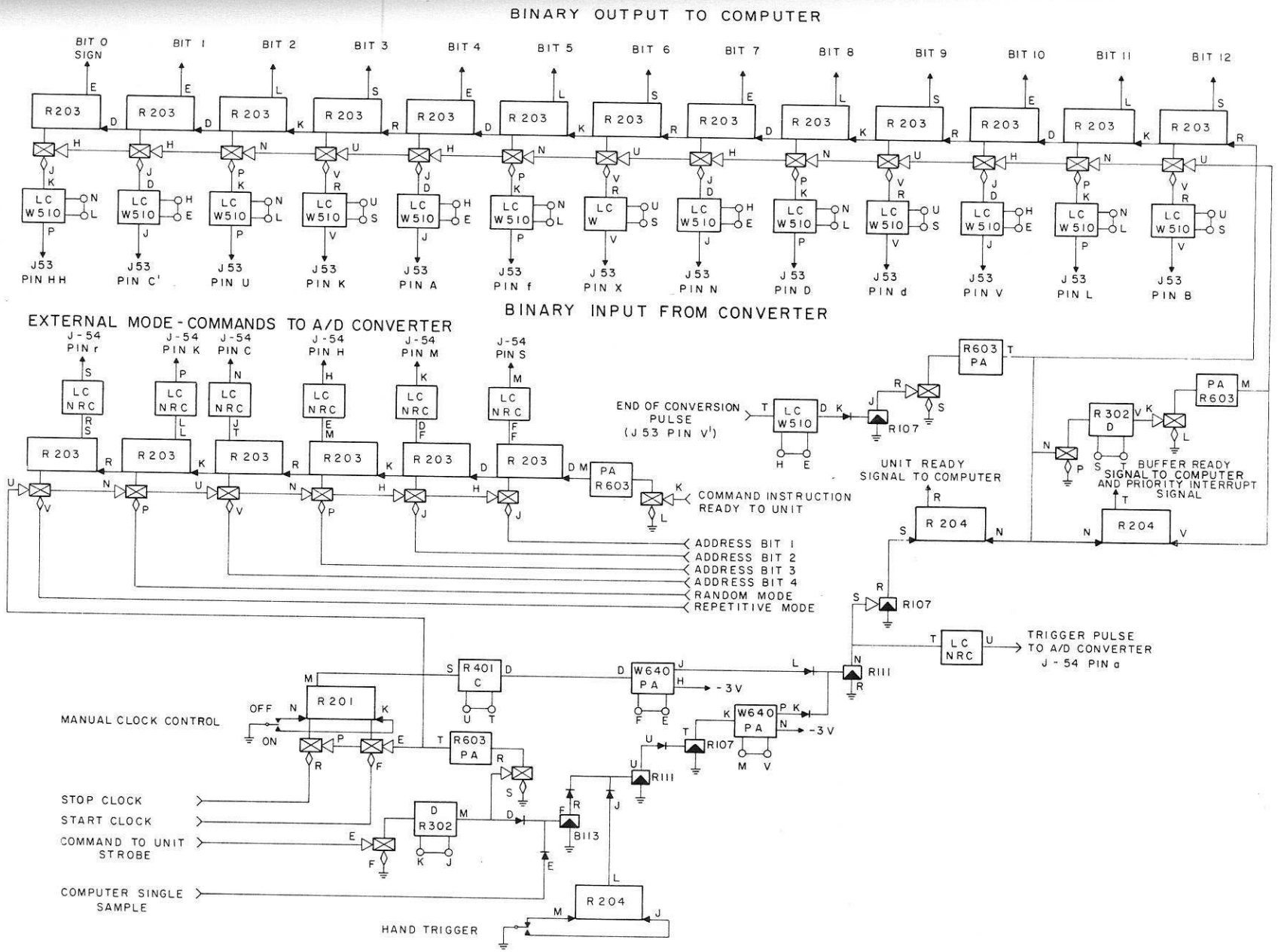


Fig. 11 Electronic interface

by the clock pulse and its output is used as the unit-ready signal for the computer. The other is set by the delayed ECC pulse and its output becomes the buffer-ready signal for the computer. The ECC pulse also generates a priority-interrupt signal going to the computer.

The complete wiring diagram for the converter/computer interface is illustrated in Fig. 11. Figure 12 shows the module locations.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
INPUTS	R 203	R 203	R 203	R 203	R 201	R 111	R 040	W 604	R 204	R 107	R 302	R 203	W 050	W 005	CONTROL OUTPUTS

TOP

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	BINARY INPUT 4	W 99	W 501	B 113	R 603	W 510	W 550	W 551	W 551	R 203	R 203	R 203	W 050	W 005	BINARY OUTPUT

BOTTOM

Fig. 12 DEC module mounting (wiring side view)

OUTPUT TIMING

Figure 13 shows the timing arrangements with the ECC pulse, converter output, and DEC register output. The converter holds the digital output on its data lines for a minimum of 26 μs for each conversion. When output registers are added, the digital output is available to the computer for a minimum of 49.5 μs for each conversion.

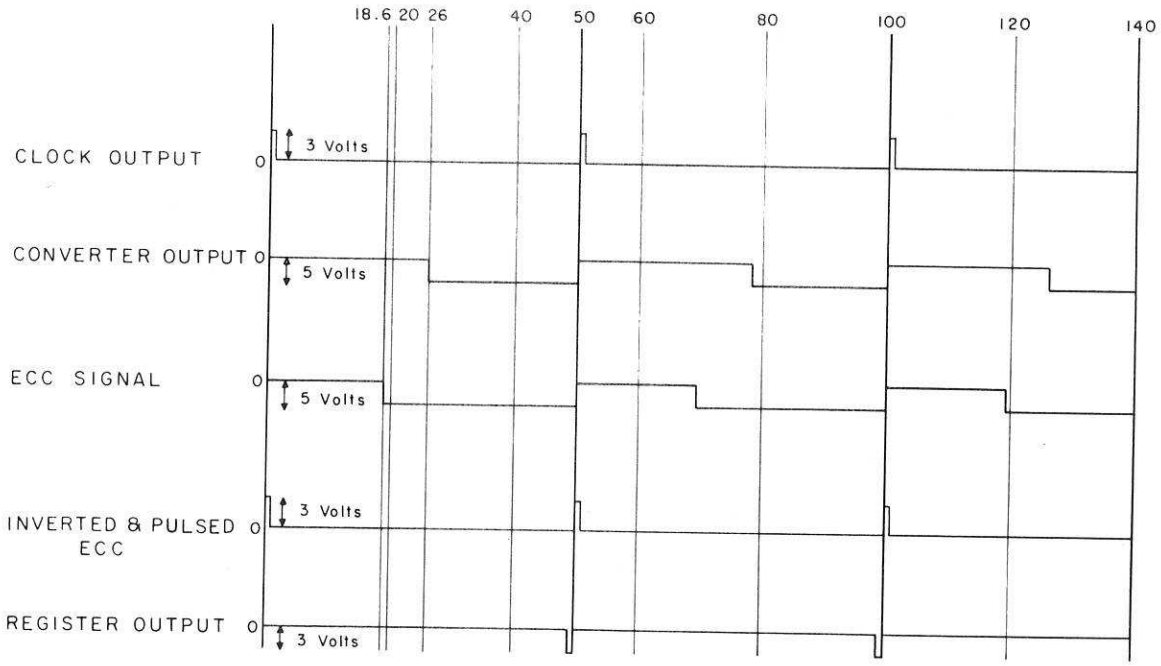


Fig. 13 Output register timing

INPUT/OUTPUT INSTRUCTIONS AND TIMING

There are four instructions the computer can give an external device: command-external-unit (CEU) instruction orders the unit to carry out an instruction; test-external-unit (TEU) tests the state of a unit; accumulator-input (AIP) and memory-input (MIP) instructions cause data to be transferred from the unit to the computer; while accumulator-output (AOP) and memory-output (MOP) instructions cause the transfer of data to the external device from the computer. CEU, AIP/MIP, AOP/MOP instructions are in two modes: wait and skip. If a wait flag is used, the computer waits until the unit answers ready, then the instruction is transferred. If the skip mode is used (no wait flag), the unit is only tested once. If it is ready, the next instruction is executed; if it is not ready, the next instruction is skipped. An external device is always ready to receive a test instruction. Direct and indirect addressing is possible.

Command, test, and data-transfer instructions consist of two words. With command and test instructions, the first word selects the unit concerned while the second word is the command code (or instruction) if in the direct mode, or is the address of the code if in the indirect mode. With data-to-memory instruction, the first word is similar to that of a test instruction, but the second word contains the storage address of the data, if in the direct mode. If the indirect mode is used, the second word is the address of the memory location containing the address of the storage location.

A general description of each command will be given, followed by a more detailed study of the electronics. If this is not sufficient, a technical description is available in "SEL 840A Computer Input/Output Interface Reference Manual".

COMMAND INSTRUCTIONS

A command instruction orders a selected unit to execute a given command. The instruction consists of two words. When the unit signals that it is ready to receive a command, the instruction code is put on the output data lines for the unit to read.

DETAILED DESCRIPTION OF CEU INSTRUCTION

The first word gives the type of command - CEU - and the unit it is for. At the data terminal, the unit code signal is combined (in an AND gate) with a T3 or T8 timing pulse to produce the this-unit signal. The operation code bits identify the instruction as a command. If the peripheral unit is prepared to accept the command, it signals unit-ready. When the data terminal receives the unit-ready signal, it puts the second word of the CEU instruction - the command contents - on the data bus and sends the unit a command instruction ready-to-unit signal. This signal is used to clear the command registers of the unit (in DEC hardware interface). On the next T3 or T8 timing pulse, a command-to-unit strobe signal is sent which is used to load the command registers. It is also sent back to the data terminal as unit-command-accepted to indicate to the computer that the registers are loaded. Because the data must be on the level input of the DCD gates for 400 ns before they are stable, the command-to-unit strobe signal is delayed for 500 ns before it is used to load the flip-flops (Fig. 14).

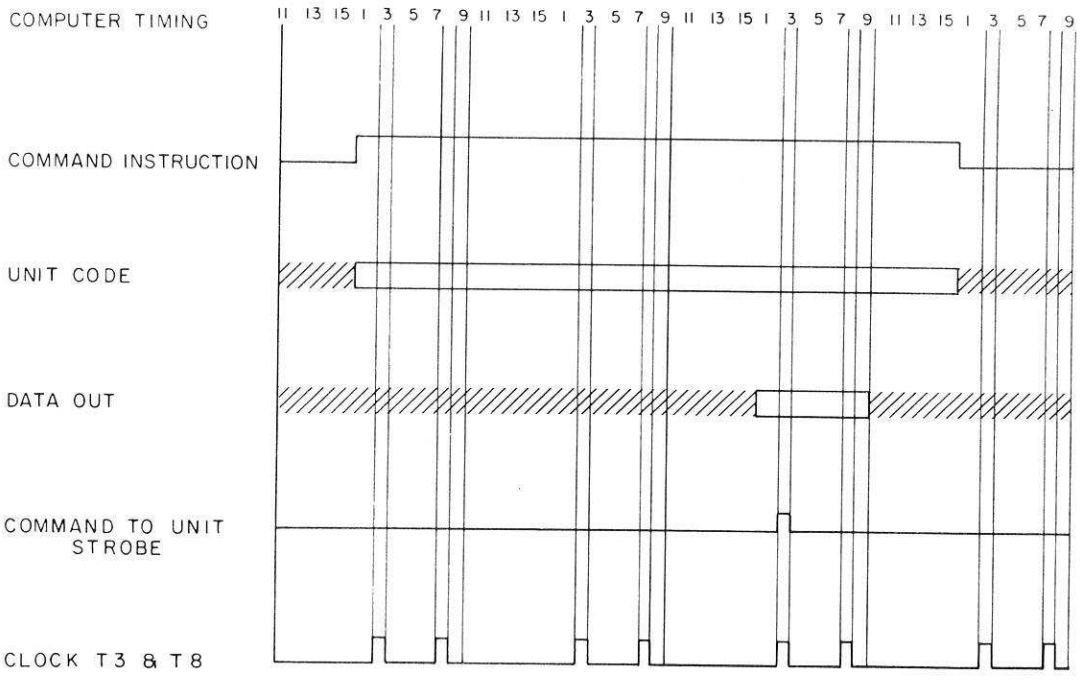


Fig. 14 CEU instruction timing

TEST INSTRUCTIONS

A test instruction tests the state of a selected unit with the unit answering either yes or no to the test. The instruction consists of two words: the first selects the unit while the second gives the test code for the desired condition. The decoded test code, unit selected signal, and the answer of the unit are all combined (in an AND gate) either before or in the data terminal. This signal is combined (in an OR gate) with the lines of other tested functions and the resulting signal is sent to the computer. If this signal indicates that the unit answers to the test, the program counter is advanced by two, while if the unit does not answer, the next instruction is executed.

DETAILED DESCRIPTION OF TEU INSTRUCTIONS

Test instructions are similar to command instructions in their form. The unit-code signal of the first word again is combined with a T3 or T8 timing pulse to produce the this-unit signal. The type of instruction bits produce a test-instruction signal. A unit is always ready to be tested so no waiting is required. The second word or data word is put on the data line and a test-instruction ready-to-unit signal is sent. This signal is used to load the test registers. The data are compared in AND gates with the state of the unit being tested and the resulting signal goes to the additional-test-function line of the data terminal. The test-instruction ready-to-unit signal is also sent back to the data terminal as unit-test-accepted (Fig. 15).

DATA TRANSFER INSTRUCTIONS

A data-transfer instruction removes data from the output registers of the unit and transfers them either to A accumulator or to memory. Accumulator input instructions are single words giving the unit number and the type of instruction. Memory input instructions are made from two words; the first is similar to an AIP, while the second gives the address where the data are to be stored.

The detailed descriptions are very similar to those for CEU instructions.

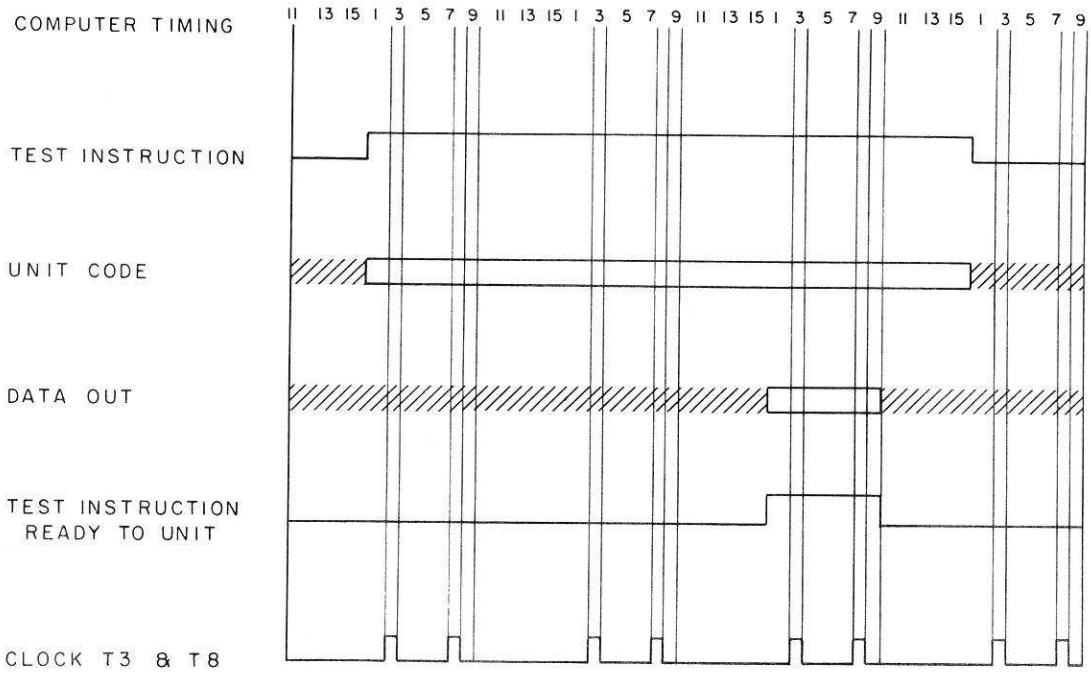


Fig. 15 TEU instruction timing

SOFTWARE

Software describes the orders that instruct the computer in the use of its hardware. This includes a program which outlines what must be done and a translator which changes the program to machine language. The SEL 840A has two types of program translators: a Fortran IV compiling system and a symbolic assembler (Mnembler).

FORTRAN IV COMPILER

A compiler changes a program written in a mathematical language to machine language. Machine language is the working code of a digital computer. Since Fortran is not used in programing the A/D converter, it is not discussed here. However, a comprehensive description is available in Fortran IV Reference Manual for the SEL 810, SEL 840 General Purpose Computers*

SYMBOLIC ASSEMBLER

A program written in assembly language is translated into machine language by the assembler. Each statement results in one operation in contrast to a Fortran statement which may result in several operations. For example, the assembler statement LAA '645 tells the computer to put the number contained in memory location 645 (octal) into A accumulator. For a list of Mnembler instructions,the SEL 840A Reference Manual* should be consulted.

If the least significant bit of the A/D converter output is dropped, the remaining 12 bits can be double packed into the 24-bit memory locations of the computer. This means storing two pieces of data in one location. However, this procedure is considerably slower than single packing.

The computer can control the converter in two modes. In one, the computer is idle when not required by the converter; i.e., there is no background program. In the other, a background program is being run. In both, the data are brought in using either a block transfer control or a priority-interrupt system. However, in the latter mode, any registers and accumulators used in the interrupt routine must be stored at the beginning and restored at the end.

---

*Systems Engineering Laboratories, Inc., Fort Lauderdale, Florida.

Several programs are needed to allow the converter to be operated at various sampling rates, owing to the time differences in operating in the different modes.

In all the programs, data are loaded from the DEC registers into two storage blocks in memory. As one block is being filled, the other is being emptied on to magnetic tape by an external method (usually a BTC). Storing the contents of index register 0 into memory location READ notifies the tape handler that a block of data is ready to be transferred.

The second word of a command or test instruction is made up as follows with a zero in a bit indicating the opposite situation from a one:

Bit 0: a one initializes BTC

1: left as zero
2: left as zero
3: left as zero
} used for interrupts

4: not used

5: not used

6: a one if clock is being tested for fast speed

7: one indicates the presence of address bit zero

8-15: not used

16: one for address bit 1

17: one for address bit 2

18: one for address bit 3

Bit 19: one for random mode
20: one for repetitive mode
} If both are 0, then sequential mode results.

21: one stops clock
22: one starts clock
23: one causes single sample
} Inverse case not true

Program A:

This is the fastest routine for removing data from the converter. The computer is dedicated to the converter, (i.e., no background program) and a block transfer control is used for moving the data from the DEC output register to memory. There are two equal-sized storage blocks in memory called ST1 and ST2.

```
        LIX    NO,1     load index register 1 with value in NO
        LAA    WC       load A accumulator with word count
        STA    '61      store word count in location 61
        LAA    ST1      load A accumulator with ST1
        STA    '60      store in location 60
        LAA    ROUT     load priority interrupt first instruction
*                       in A
        STA    '102     store in location '102
        PIE    2,0      enable interrupt
        CEU    33,W
        DATA   '00000002          initialize BTC, start clock
*                       in sequential mode
```

```
FAST    ***   **
        IIB   HOLD,1    increment index register 1 and if
*                       not 0, branch to HOLD
        LAA   ST1       load A accumulator with ST1
        STA   '60       store in location 60
        LIX   NO,1      reload index register 1 with NO
        STI   READ,0    signal to magnetic tape handler
        PIR   FAST      reset interrupt and return
HOLD    LAA   ST2       load A accumulator with ST2
        STA   '60       store in location 60
        STI   READ,0    signal to magnetic tape handler
        PIR   FAST      reset interrupt and return
* The following is needed to define variables used:
WC      DATA  number of words per storage block
ST1     BSS   WC        this reserves a block of storage for ST1
ST2     BSS   WC        this reserves a block of storage for ST2
NO      DATA  -2        puts -2 in location NO
READ    DAC   0
ROUT    SPB   FAST      causes branch to interrupt routine when
*                       interrupt signal is received
```

The interrupt routine is needed to switch blocks when one is full (or to re-initialize BTC). This happens only when a storage block has "WC" pieces of data in it. The time required for storing a data word and re-initializing the BTC is 22.75 μs. If the interrupt routine is bypassed, as in the middle of a block, only 1.75 μs are required for storing a data word.

Program B:

Again the computer is dedicated to the converter. However, a priority interrupt is being used instead of the BTC for transferring data. The storage blocks are successive. With this program, the converter is stopped after a given number of blocks (NOVA) has been filled.

```
LIX    NOVA,3     load index register 3 with value in NOVA
LIX    COUN,2     load index register 2 with value in COUN
LIX    SIZE,1     load index register 1 with value in SIZE
LBA    SIZE       load B accumulator  with value in SIZE
LAA    RT         load interrupt branch into memory
STA    '102       location 102 (octal)
PIE    2,0        enable interrupt
CEU    33,W
DATA '00000002    start clock, sequential mode
```

```
FASS    ***   **
        MIP*  33,W      input data from unit
        DAC   ST,2      store data in location ST + value of
*                       index register 2
        IIB   NEXT,2    increment index register 2 and branch
*                       if not 0
        LIX   COUN,2    reload index register 2
        BRU   END
NEXT    IIB   NO,1
END     TBI   ,1        load value of B accumulator in index
*                       register 1
        STI   READ,0    enable tape unit
        IIB   NO,3      check number of blocks stored
        CEU   33,W      if necessary number has been reached
        DATA  '00000004     stop conversions
NO      PIR   FASS
```

*The following is needed to define variables used:

```
NOVA    DATA  -number of blocks of data wanted
SIZE    DATA  -number of samples per block
COUN    DATA  -2 times SIZE
```

```
ST      BES   COUN
READ    DAC   0
RT      SPB   FASS
```

The minimum time for storing data in the middle of a block is 19.25 μs. If a block is full, 29.75 μs are required to store the data and change storage locations.

Program C:

A background program is being processed when the interrupt routine takes control. Any index registers and accumulators used must be saved at the beginning and restored at the end. Data is stored in two separate blocks STO1 and STO2.

```
        LAA   SIZE
        STA   NUM
        LAA   RG
        STA   '102
        PIE   2,0
        CEU   33,W
        DATA  '00000002   start clock in sequential mode
```

```
NODE    ***     **
        STI     SAUX,1      save index register 1
        LIX     NUM,1       load index register 1 with number in NUM
        MIP*    33,W        data input into location STO1
LOC1    DAC     STO1,1      plus index flag
        IIB     RETU,1      advance sample count
        IMS     NOVA,1      advance block count, if zero skip one
*                           instruction
        BRU     NOW         unconditional branch to NOW
        CEU     33,W
        DATA    '00000004       stop clock
        BRU     RETU
NOW     IAM     LOC1    ┐
                        │   interchange contents of locations
        IAM     LOC2    │
                        │   LOC1 and LOC2
        IAM     LOC1    ┘
        LIX     SIZE,1      reset sample count
        STI     READ,0      enable tape unit
RETU    STI     NUM,1       store contents of index register 1 in
*                           memory location NUM
        LIX     SAUX,1      restore index register 1
        PIR     NODE        return to program
```

*The following list defines the above variables:

```
SAUX    DAC    0
NUM     DAC    0
LOC2    DAC    STO2,1
RO      SPB    NODED
NOVA    DATA   -number of blocks needed
SIZE    DATA   -number of samples per block
STO1    BES    SIZE
STO2    BES    SIZE
READ    DAC    0
```

The time required to input data if no transferring of blocks is required is 31.5 μs. If a transfer of blocks is required, 57.75 μs are needed.

Program D:

If a slow sampling rate (greater than 60 μs) is used and 11-bit words are allowed, data words can be double packed into the 24-bit memory locations. This routine also permits sampling to be triggered by a 60-cycle line feed or by single sampling triggered by the computer or by the operator. Data are stored in the manner outlined in Program C.

```
        LAA     SIZE
        STA     NUM         store size of blocks
        LAA     ONE
        STA     DUM
        LAA     ROS
        STA     '102
        PIE     2,0
        CEU     33,W
        DATA    '00000002   start clock in sequential mode

SLOW    ***     **
        STI     SAUX,1      save index register 1
        LIX     NUM,1
        STA     TEE         save A accumulator
        STB     BEE         save B accumulator
        LAA     DUM         beginning of double packing routine
        LBA     HOLD
        BAN     STEL        test sign of A accumulator
        NEG
        STA     DUM
        AIP     33,W
        TAB                 put contents of A accumulator into
*                           B accumulator
```

```
        STB   HOLD
        LAA   TEE       restore A accumulator
        LBA   BEE       restore B accumulator
        BRU   RETU
STEL    NEG
        STA   DUM
        AIP   33,W
        LSL   12        left shift logical A accumulator 12 places
        FRL   12        full left rotate logical 12 places
        LAA   TEE
        STB*  LOC1      store B accumulator indirectly in
*       LBA   BEE       location LOC1
        IIB   RETU,1
        IMS   NOVA      block count advanced
        BRU   NOW
        CEU   33,W
        DATA  '00000004     stop clock
        BRU   RETU
NOW     IAM   LOC1
        IAM   LOC2      interchange contents of locations
        IAM   LOC1      LOC1 and LOC2
```

```
        LIX   SIZE,1
        STI   READ,0
RETU    STI   NUM,1
        LIX   SAUX,1
        PIR   SLOW      return to program
```

*Variables to be defined are:

```
SAUX    DAC   0
NUM     DAC   0
ROS     SPB   SLOW      store place and branch to interrupt
*                       routine when interrupt occurs
HOLD    DAC   0
DUM     DAC   0
BEE     DAC   0
TEE     DAC   0
LOC2    DAC   STO2,1
LOC1    DAC   STO1,1
NOVA    DATA  -number of blocks needed
SIZE    DATA  -number of samples per block
STO1    BES   SIZE
STO2    BES   SIZE
ONE     DAC   1
READ    DAC   0
```

Double packing increases the execution time of the interrupt procedure by 31.5 μs. This means time for storing in the middle of a block and at the end of a block is 63 μs and 89 μs respectively.

Summary of Software:

Programs B, C, and D are similar in that they use a priority interrupt to input data. By using tests of the state of the clock and of the presence or absence of a background program, they can be combined into one program. The interrupt routine will have three entry points, one for each situation.

```
        LAA   TEST
        BAP   BACK      test for background program, skip
*                       if negative
        LIX   NOVA,3
        LIX   COUN,2
        LIX   SIZE,1
        LBA   SIZE
        LAA   RT
        STA   '102
        LAA   FT
        STA   FIN
```

```
        PIE   2,0
        CEU   33,W
        DATA  '00000002     start clock and convert in sequential
*                           mode
BACK    LAA   SIZE
        STA   NUM
        TEU   33            test clock speed, skip if fast
        DATA  '00400000
        BRU   LONG
        LAA   RG
        STA   '102
        LAA   FG
        STA   FIN
        PIE   2,0
        CEU   33,W
        DATA  '00000002     start clock in sequential mode
LONG    LAA   ONE
        STA   DUM
        LAA   ROS
        STA   '102
        LAA   FOS
```

```
        STA   FIN
        PIE   2,0
        CEU   33,W
        DATA  '00000002     start clock in sequential mode
FASS    ***   **
        MIP*  33,W
        DAC   ST,2
        IIB   NEXT,2
        LIX   COUN,2
        BRU    END
NEXT    IIB   NO,1
END     TBI   1
        IIB   NO,3
        CEU   33,W
        IOC   '00000004     stop clock
FIN     DAC   0             return to background program
NODE    ***   **
        STI   SAUX,1
        LIX   NUM,1
        MIP*  33,W
```

```
LOC1    DAC    STO1,1
        BRU    LATE
SLOW    ***    **
        STI    SAUX,1
        LIX    NUM,1
        STA    TEE
        STB    TEE
        LAA    DUM        begin double packing
        LBA    HOLD
        BAN    STEL
        NEG
        STA    DUM
        AIP    33,W
        TAB    12
        STB    HOLD
        LAA    TEE
        LBA    BEE
        BRU    RETU
STEL    NEG
        STA    DUM
        AIP    33,W
```

```
        LSL   12
        FRL   12
        STB*  LOC1
        LAA   TEE
        LBA   BEE
LATE    IIB   RETU,1
        IMS   NOVA
        BRU   NOW
        CEU   33,W
        IOC   '00000004     stop clock
        BRU   RETU
NOW     IAM   LOC1
        IAM   LOC2
        IAM   LOC1
        LIX   SIZE,1
        STI   READ,0
RETU    STI   NUM,1
        LIX   SAUX,1
FIN     DAC   0
```

Variables used and their meanings are listed below:

```
NOVA    DATA   -number of blocks of data wanted
SIZE    DATA   -number of samples per block
COUN    DATA   -2 x SIZE
ST      BES    COUN
STO1    BES    SIZE
STO2    BES    SIZE
LOC2    DAC    STO2,1
LOC1    DAC    STO1,1
SAUX    DAC    0
NUM     DAC    0
DUM     DAC    0
READ    DAC    0
BEE     DAC    0
TEE     DAC    0
HOLD    DAC    0
ONE     DAC    1
TEST    DAC    +ve if background program
               -ve if no background program
BT      SPB    FASS
FT      PIR    FASS
```

RG	SPB	NODE
FG	PIR	NODE
ROS	SPB	SLOW
FOS	PIR	SLOW

This program combines the features of programs B, C, and D. Program A has been left separate since it uses a Block Transfer Control to input data. As a BTC is needed to empty the storage locations, program A demands two Block Transfer Controls. The SEL 840A ordered is equipped with only one BTC so program A at present is inoperative. The mode of the computer is indicated to the program by the operator. If a background program is present, he loads a positive number or zero into memory location test. If the computer is dedicated, he loads a negative number into test.

The various tests done in the combined program may be testing of sense switches, flip-flops, or may be testing of a command typed in. As the computer has not arrived, these programs have not been tested.

SUMMARY

The hardware has been checked out and works satisfactorily. Sampling rates as high as 20 kHz have been tried successfully. As the computer has only just arrived, it has not been interfaced with the DEC hardware yet. No problems are anticipated. The software has not been tested but appears to be correct. All that remains to be completed is the interfacing of the computer with the control logic and the testing of the software.

ACKNOWLEDGMENTS

I wish to thank Mr. J.K. Pulfer for his invaluable assistance and Dr. M. Wein and Mr. N. Burtnyk for their ideas and suggestions.

REFERENCES

(1) Digital Logic Seminar, 1965, Digital Equipment Corporation, Maynard, Massachusetts.

(2) Fortran IV Reference Manual for the SEL-810, SEL-840 General Purpose Computers, 1966, Systems Engineering Laboratories, Inc., Fort Lauderdale, Florida.

(3) Instruction Manual for VMX10BIVS12-ABISA3 Multiplexer/ Analog-to-Digital Converter, Volumes I and II, 1966, Adage, Inc., Boston, Massachusetts.

(4) SEL 840A Computer Input/Output Interface Reference Manual, 1966, Systems Engineering Laboratories, Inc., Fort Lauderdale, Florida.

(5) Reference Manual for the SEL 840A General Purpose Digital Computer, 1966, Systems Engineering Laboratories, Inc., Fort Lauderdale, Florida.

(6) The Digital Logic Handbook, 1966-67, Digital Equipment Corporation, Maynard, Massachusetts.