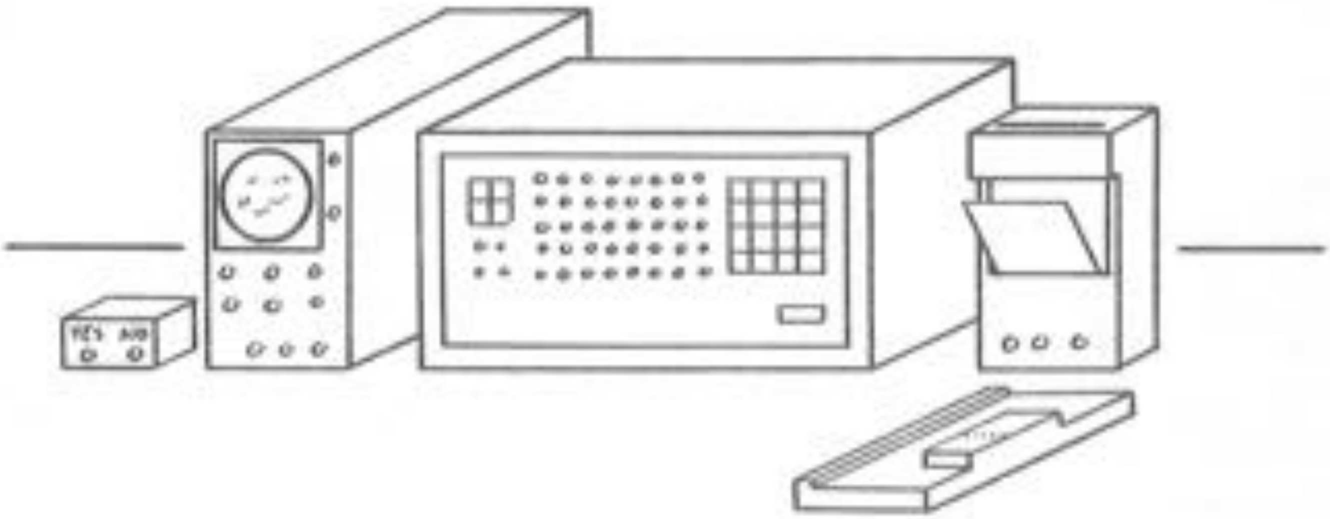
Title	Microprocessor Manual System 00
Author	Weisbecker, Joseph A.
Location	Hagley Museum and Library, from David Sarnoff Library Collection (Acc. 2464), Box 873, Folder 2
Published	RCA, ~1971
Contents	Section I - System Description — Section II - Order Code — Section III - Operating Procedures — Section IV Detailed Logic — Section V - Sample Programs

MICROPROCESSOR MANUAL

SYSTEM 00

BY

JOSEPH A. WEISBECKER



Section I - System Description

A. General

The basic, stand alone, micro-processor system comprises the following physical units as illustrated on the title page:

1. Micro-processor
2. Display
3. Card Reader and Punch
4. Switch box

This system satisfies a number of applications such as game playing and classroom use. It can be readily expanded via addition of peripheral devices and memory. Potential applications for expanded systems include testors, monitors, intelligent terminals, controllers, off line data processing devices, music synthesizers, etc.

All logic circuits utilize standard 7400 series TTL chips. (Refer to manufacturers' literature for circuit details.) The system as shown requires about 100 watts (including 60 watts for the display). System weight is about 30 pounds (including 10 for display).

A system block diagram is shown in Figure 1. The system comprises the following:

M - Memory (64K bytes maximum)

R - Register Array (16-16 bit registers)

F - Function unit (binary add, subtract, logical, shift)

C - Control

E - External interface

D - External device.

These functional units are connected via an 8 bit (byte) bus as shown. These units are described briefly below and in more detail in subsequent sections.

B. Basic Micro-processor

This physical unit contains the logic circuits for instruction interpretation and execution, power supplies, input-output interface circuits, 1024 bytes of memory, and an operator control panel. It also includes 16 switches for data and/or program entry in "Hex" format.

Memory capacity can be expanded to 64K bytes maximum. Word length is 8 bits. Two machine cycles of 1.6 μs minimum are required per instruction. Maximum instruction execution rate is 300K ops/sec. A direct memory access channel permits independent, asynchronous input/output up to 200K bytes/sec. An external program interrupt line is also provided.

Figure 2 shows the data flow paths within the micro-processor. (Detailed logic will be found in Section IV.) Register naming, and memory addressing conventions are described below.

RØ, R1 represent an addressable array of sixteen 16-bit R registers. RØ is the least significant R register byte and R1 is the most significant R register byte. X, P, and N are three 4-bit registers. The contents of X/P/N specify one of the 16 R registers. R(N) is used here to denote the R register specified by the 4 bits contained in the N register. RØ(N) specifies the low order 8 bits (byte) of the R register selected by N. R1(N) specifies the high order byte of an R register. The contents of a specified R register (2 bytes) can be transferred to A and C registers. The 16 bits in A are used to address M. The 16 bits in C can be incremented or decremented by "1" and written back into R.

M(R(N)) refers to a one byte memory location addressed by the contents of R(N). This indirect addressing system is basic to the simplicity and generality of the processor.

D is an 8 bit data register which has one bit right shift built in. F is an 8 bit logic network for performing binary add, subtract, logical and or, and exclusive or on two 8-bit operands. One of the operands is the bus byte and the other is the contents of D.

I is a 4 bit (digit) instruction register. 4 bit operation codes are placed in this register and decoded to control instruction execution.

Bytes can be read onto the bus from any of the registers, memory, or external interfaces. A bus byte can, in turn, be transferred to a register, memory, or external interface.

Most of the R registers are available for use as data registers, address registers, or program counters. However, if applications requiring external interrupts or cycle stealing are anticipated, R(Ø), R(1), and R(2) should be reserved for these functions. Detailed operation of the basic micro-processor is best described in terms of its instruction set. This description will be found in section II.

C. Display

The display provided comprises a conventional X-Y-Z scope. Two display options can be program selected. Either a 32x32 or 16x64 matrix of dots can be presented for viewing. These two 1024 dot (or bit) patterns require 128 bytes of memory reserved as a display area. In the 32x32 dot mode the memory bytes would be displayed as shown in figure 3. (Memory addresses are in hex format.) "1" bits are displayed as dots, "Ø" bits as no dots. Pictures, letters, or numbers can all be displayed as patterns of bits in a memory display area. Optional use of the 16x64 dot matrix option permits 4 lines of up to 10 characters per line in a 5x7 dot format.

Figure 4 provides a convenient worksheet on which to lay out 32x32 bit display patterns. Detailed display programming will be discussed in sections II and V.

The scope used was a heath model IO-17, 3" utility model. Only one minor modification is required. The input to the retrace blanking amplifier is disconnected from the internal sweep generator and brought out to a front panel jack. This amplifier is driven directly from 5 volt TTL logic to effect Z-axis modulation.

D. Card Reader

This unit was constructed to read one 3"x5" punched card at a time. The cards are manually dropped into a top slot and read photoelectrically while falling. The cards fall into a hopper and maintain their order when removed. This type of reader provides an extremely inexpensive means for program and parameter loading. It is particularly attractive for short programs.

The card format is illustrated in Figure 5. The card is divided into two tracks (A&B). Only one track at a time is read. The "A" track is read by dropping the card in the reader with the "A" arrow down as shown. The card is then turned around for the "B" track to be read.

Each side (or track) can contain up to 12 bytes. Each byte is written as two hex digits in the appropriate column of boxes in the center of the card. The "B" track of the card shown contains the punched codes for all 16 hex digits (Ø-F). Holes are punched at the intersections of the horizontal and vertical lines. The row of 5 hole positions opposite each written hex digit is punched with its binary representation (hole punched for "1", no hole for "Ø"). A parity bit (P) is then punched to make the sum of the holes (1's) in each row odd. Blank rows are ignored.

Cards are read one hex digit (hole row) at a time while falling. A parity error sets an error light on the reader.

A manual card punch is also provided. It comprises a card guide and chad catcher. Five holes are provided labeled P, 8, 4, 2, 1. The hex digit row line is aligned with a mark on the punch and a stylus is inserted into each hole in which "1" is desired.

E. Switch Box

An auxiliary "yes-no" switch box has been provided to facilitate simplified interaction with the system by unsophisticated users. These two switches can be sensed by the program.

Section II - Order Code

A. Processing Instructions

The operation of the micro-processor is best described in terms of its instruction set which is shown in the micro-instruction summary (figure 6).

A one byte instruction format is used. Two machine cycles are required per instruction. The first machine cycle causes an 8-bit instruction to be fetched from M and placed in the I and N registers [M(R(P))→I,N]. This is accomplished by gating the contents of P to select R. R(P) is then gated to A and C. While waiting for the M(R(P)) byte access time, C is incremented by 1 and replaces the original contents of R(P). The most significant digit (4 bits) of M(R(P)) is gated to I via the bus. The LSD of M(R(P)) is gated to N. At the end of the instruction fetch machine cycle, I and N contain the 8 bit instruction originally addressed by the program counter [R(P)] and the program counter has been incremented by 1 so that it now points to the next instruction byte in sequence.

At this point it should be noted that any of the 16R registers could be used as the program counter. Multiple program counters are facilitated.

The next machine cycle always executes the instruction contained in I and N. The execution of the data handling and branch instructions are described below (I2 denotes that the digit in I has the value of 2):

I1 - R(N)+1→R(N)
The 16 bits in the R register specified by the current digit in N is incremented.

I2 - R(N)-1→R(N)
The 16 bits of R(N) are decremented by 1.

I4 - M(R(N))→D,R(N)+1→R(N)
The M byte addressed by R(N) is read from M and placed in D. R(N) is incremented by 1.

I5 - D→M(R(N))
The byte in D is written to the M byte location addressed by R(N).

I8 - RØ(N)→D
The least significant byte of R(N) is placed in D.

I9 - R1(N)→D
The most significant byte of R(N) is placed in D.

IA - D→RØ(N)
The byte in D replaces the least significant byte of R(N).

IB - D→R1(N)
The byte in D replaces the most significant byte of R(N).

IC - DØ→RØØ(N)
The least significant 4 bits (digit) in D replaces the least significant digit of R(N).

ID - N→P
The 4 bit digit in N is placed in P. This effectively changes the current program counter and constitutes a branch.

IE - N→X
The 4 bit digit in N is placed in X.

IF - Perform function specified by digit in N:
- NØ - M(R(X))→D
- N1 - M(R(X)) "OR" D→D
- N2 - M(R(X)) "AND" D→D
- N3 - M(R(X)) "EXCL.OR" D→D
- N4 - (MR(X)) + D→D [BIN.ADD,FINAL CARRY→DF]
- N5 - M(R(X)) - D→D [BIN.SUBT.,FINAL CARRY→DF]
- N6 - SHIFT D RIGHT 1 BIT [LSB→DF]

Note that a flag bit (DF) is provided. This flag can be tested by the following branch instruction.

I3 - Conditional branch
N specifies the condition to be tested.
- NØ - unconditional branch
- N1 - byte in D not all zeros
- N2 - byte in D all zeros
- N3 - D flag (DF) equals 1
- N4 - external flag 1 set
- N5 - external flag 2 set
- N6 - external flag 3 set
- N7 - external flag 4 set

The last four tests will be discussed with I/O device instructions. If the condition specified by N exists the M byte following the I3 instruction is read from M and replaces the least significant byte of R(P). This permits direct branching within a 256 byte mini-page. If the specified test condition is not present the M byte following I3 is skipped and the next instruction in sequence will be fetched.

B. Input-Output Instructions

1. Yes-No Switches

The "yes" switch causes external flag 3 to be set. The "no" switch sets external flag 4. These flags are tested by the I3 instruction described above. An I6 instruction with N=3 resets both flags (3&4).

2. Hex Switch Panel

16 hex digit switches are provided (Ø-F). Bytes can be entered by two switch depressions per byte. The least significant digit is entered

first followed by the leftmost or most significant digit. Instructions are provided to permit several switch input modes.

The first switch mode comprises sensing a byte ready flag and subsequently executing an input instruction to write the input byte into a memory location. This mode of operation first requires selection of the switch panel. An I6 instruction with N=1 always causes selection of an input or output device. The byte contained in M(R(X)) specifies the device to be selected. If M(R(X)) contains ∅1 the hex switch panel is selected. (R(X)+1 is performed by this instruction.) Following selection of the switch panel it must be set to 1 of 2 possible modes of operation (programmed or direct). The switch panel remains selected until another I6 with N=1 and (MR(X))≠1 is executed.

The switch panel is set to the "programmed" mode by an I6 instruction with N=2 and M(R(X))=∅. (R(X)+1 is also performed.)

From this point on the switch panel interface logic is activated until reset by an I6 instruction with N=2 and M(R(X))≠∅1 (switch panel must be selected).

While active the switch logic causes external flag 1 to be set after each pair of digits is entered, indicating that a byte is ready to be stored in memory. An I3 instruction with N=4 should be used to test this condition periodically in the program. When the flag is set, an I6 instruction with N=8 will cause the input byte to be stored at M(R(X)). R(X) is unmodified. Note that the switch panel must be selected prior to testing its flag or attempting to store an input byte. When operating in this mode flag sensing and byte storage should be programmed to occur at a faster rate than anticipated manual digit entry. The byte storage instruction resets external flag 1.

The switch panel (after selection) can optionally be placed in the "direct" mode by executing an I6 instruction with N=2 and M(R(X))=Ø2. In this mode of operation each pair of switch entered digits will be automatically stored at M(R(Ø)). R(Ø) will be automatically incremented by 1 following storage of each input byte. Input byte storage will occur even if the seitch panel is no longer selected. R(Ø) should initially contain the memory address of the first byte to be stored.

When the switch panel is used the card reader should be off (or disconnected).

3. Card Reader

The card reader shares the switch panel interface circuits. When the card reader is "on" each hex digit punched on a card is treated as a digit entered via the switch panel. Selection and byte entry is programmed as described above for the switch panel.

A ready light on the card reader indicates that the program is ready to receive card data. In the "program" mode flag sensing and byte storage instructions must be programmed to occur at a higher rate than card digits. Assuming 1 ms/card digit should be safe with this reader.

An error light on the reader indicates a card digit parity error. Restarting the entire reader operation is required. Turning the reader off momentarily will reset the error indicator.

Either the card reader/switch panel can be used for initial program loading. This type of operation is discussed in section III.

4. Display

The display is always operated in the "direct" mode (cycle stealing) and also utilizes the program interrupt facility of the microprocessor.

To activate the display, the following program steps are required.

First the display must be selected by an I6 instruction with N=1 and M(R(X))=∅2 (R(X) is incremented by 1). Next the display is activated by an I6 instruction with N=2 and M(R(X))=∅1/∅2(R(X)+1 is performed). If M(R(X))=∅1 a 32x32 bit display results. If M(R(X))=∅2 the display format will be 16x32 bits.

As soon as the display is activated, the display interface circuits cause memory bytes to be read from memory as needed. All bits of each byte are displayed as shown in Figure 3. The direct memory access circuits are used by the display interface. Each byte is read automatically from M(R(∅)) and R(∅) incremented by one. The display steals 128x60 or 7680 machine cycles per second to maintain a 1024 bit refresh rate of 60 refresh cycles/sec. A machine cycle time of 1.6μs results in a total availability of over 600,000 machine cycles per second. Display cycle stealing results in degrading performance by less than 2%. (Degradation would be 7680/200,000 or less than 4% for a 5μs machine cycle.)

The 128 byte memory area to be displayed must be defined by R(∅). The display circuits cause a program interrupt 60 times per second (after displaying byte number 128). This program interrupt always causes an automatic transfer of control to the instruction addressed by R(1) (P is automatically set to 1). An interrupt routine addressed by R(1) should be provided which initializes R(∅) to the address of the first byte of a 128 byte memory display area.

Two additional actions occur when an interrupt occurs. The contents of X and P are placed in T, and X is set to 2. Instruction I7 is provided to facilitate returning to normal processing following interrupt. I7 with N=8 causes T to be stored at M(R(X)). This instruction permits the interrupted values of X&P to be saved. 17 with N=8 causes M(R(X)) to be placed in X&P effecting a return after interrupt. This instruction also causes R(X)+1 and an interrupt mask (IM) to be reset. IM is always set by an interrupt. IM inhibits interrupts and must be reset by a dummy 17 instruction when the display is initially activated.

Examples of interrupt programming provided in Section V will clarify the above.

Section III - Operating Procedures

The operator control panel is shown in Figure 7 (actual size). "X" indicates a light, "T" a toggle switch, and "M" a momentary contact push button. All square switches are momentary contact types.

A. Normal Operation

1. Power On

All toggle switches should be off. Pressing the power switch turns on the processor and interface circuits. If the display is to be used it should also be turned on. (Display, card reader, and yes-no switch box are connected via plugs at back of micro-processor.)

After power on, press SP (stop) and CL (clear). SP (stop) and ID (idle) lights should be on.

The SP switch turns off the processor clock. The ST switch turns the processor clock on. The CL switch sets ∅ in I, ∅ in N, and ∅∅∅∅ in R(∅). It also resets various control and interface circuits.

The SP light is off when the clock is running. The ID light is on whenever I contains ∅ (an idle instruction).

2. Initial Program Load

   a. Load Switch Off

   b. Clear (CL) - Load switch must be off

   c. Start (ST)

   d. Load Switch On.

Bytes can now be entered and automatically stored in memory starting at location ØØØØ. To enter bytes via the switch panel depress the desired hex digit switch for the least significant digit of the byte to be entered followed by the switch for the most significant digit.

Upon entering the most significant digit the full byte will be stored in memory. Bytes will be stored sequentially starting at memory location ØØØØ. The center top lights will display the 8 bits of each byte entered as they appear in memory. The card reader should be off when using the hex switch panel.

If desired, the card reader may be turned on and bytes can be entered sequentially from cards. If the card reader error light comes on, steps a-d should be repeated and loading restarted. Turn "load" off and press SP after completion of loading from switches/reader.

3. Load Memory Address Check

The address of the memory byte following the last byte entered can be checked as follows:

   a. Stop (SP)

   b. MN on

   c. Set bus select switches to "1ØØ". Lights Ø-7 will show the most significant byte of the last memory byte storage address plus one.

   d. Set bus select switches to "Ø1Ø". Lights Ø-7 will display least significant byte of memory address.

   e. MN off, bus select switches off (ØØØ).

4. Program Initiation

   a. All toggle switches off

   b. CL (clear)

   c. ST (start)

At this point the clock is running and an idle instruction is being executed. P=∅ so that R(∅) is the initial program counter. Pressing RS (resume) will cause the cycling idle instruction to be terminated and the instruction in memory location ∅∅∅1 to be fetched and executed. The program (starting at location ∅∅∅1) has now been initiated. Normal instruction fetching and execution will proceed until stopped by pressing the SP switch or an idle instruction is executed.

Note that an idle instruction in a program will cause the processor to idle only if no external devices are stealing memory cycles or causing interrupts.

The SP switch causes the clock to stop cleanly at the end of the current machine cycle. Subsequently pressing the ST switch will cause resumption of program execution with no error.

5. Reading Memory

Bytes can be read sequentially from memory at any time for checking purposes. The procedure is as follows:

   a. SP (stop)

   b. CL (clear)

   c. ST (Start)

   d. Read on

The byte lights ∅-7 will now be displaying the byte in memory location ∅∅∅∅. Each time the RS switch is pressed a memory address counter will be incremented by one, and the next memory byte displayed.

Note that in both the load and read modes R(ϕ) is used as the memory address counter. R(ϕ) is set to ϕϕϕϕ by the CL switch. It is possible to start a load/read operation at any memory location by presetting R(ϕ). This will be discussed under test procedures.

B. Test Procedures

1. Setting Registers

Any R register can be set to any value as follows:

a. SP, CL, MN on

b. set R select switches to number of register to be set. (ϕ-F)

c. R1/Rϕ on to select byte of register to be set

d. Set switches ϕ-7 to bit pattern to be placed in selected register byte

e. Press WR to write bit pattern into selected register.

f. Bit switches off, R1 and Rϕ off

g. Set bus SEL switches to "1ϕϕ" to display byte 1 of register just set or to "ϕ1ϕ" for byte ϕ.

The above procedure can be used to preset R(ϕ) to any starting address prior to load/read operations discussed previously. The previous procedures are modified so that following CL, R(ϕ) is set to the desired starting address.

2. Setting Memory Bytes

Any memory byte can be set as follows:

a. SP, CL, MN on

b. Set R(ϕ) equal to address of memory byte to be written

c. Set switches ϕ-7 to bit pattern to be placed in memory

d. Press WM to store bit pattern in memory

e. Bit switches off

f. Set bus select switches to "ØØ1" to display memory byte just stored.

3. Miscellaneous

The repeat switch permits repeated execution of a single machine cycle.

The MC switch permits one machine cycle to be executed per ST depression.

The win switch permits setting A bit switch pattern into I and N. The CM switch permits clearing memory. The memory clear procedure is:

a. SP, CL, MN on

b. Bit switch 4 on

c. Pressing win writes 4 into I and Ø into X.

d. MN off, RPT on.

e. Pressing ST causes repeated execution of an I4 instruction. This continuously sequences through all memory locations in a read mode. Briefly turning CM on will result in all memory locations being set to ØØ.

A variety of other manual operations are possible. The bit switches and lights are connected to the byte bus for flexibility. The detailed logic in Section IV should be examined to determine feasibility of any desired manual operation. The following briefly reviews the major functions of all switches and lights:

ST - Start Time Pulses (SP light off).

SP - Stop Time Pulses

RS - Resume execution following idle (causes R(∅)+1)

CL - Clear to idle state, stop TP's, set R(∅), N, P. to ∅

READ - M(R(∅))→ Bus, R(∅)+1 (if SP light off)

LOAD - Permits card/switch load

∅-7 Lights - Bus Status

∅-7 Switches - Set Bus to "1" (if MN)

MN - Enable Manual operations (MN light on)

R1 - Inhibit R∅ RD/WR

R∅ - Inhibit R1 RD/WR

PX - P&X → 7

WIN - Bus→I,N

WP - Bus→P

WR - Bus→R∅/R1 of selected R

WM - Bus→M(A)

R Select - Selector R→A

Bus Select:

∅∅∅ - E/∅-7 Switches→Bus

∅∅1 - M→Bus

∅1∅ - A∅→Bus

∅11 - T→Bus

1∅∅ - A1→Bus

1∅1 - N→Bus

11∅ - D→Bus

111 - "∅∅1∅∅∅∅1"→Bus

RPT	- Repeat Machine Cycle
MC	- One machine cycle at a time
CM	- Write M
SP Light	- TP's off
ID Light	- Idle
MN Light	- MN switch on
NB Light	- Ready for next input byte (when hex switch panel active)
Power	- AC on/off

Section IV - Detailed Logic

Logic-1 through logic-19 show the detailed logic design of the microprocessor system. Table I shows the number of chips used in the design. Three power supplies are provided as shown in Figure 8. All circuits were mounted on cards as shown in Figures 9-14. Figures 15-18 provide sample timing for selected logic.

Logic-1 and logic-2 illustrate the control panel switch wiring. The 16 hex switches are shown on logic-14 with their interface logic.

A word about conventions - Logic symbols have both a card location (SE1, XC6, etc.) and a chip type (7400, 74121, etc.). Pin numbers are also provided. Multiple wire buses are indicated by slash marks.

Logic-3 shows the memory. It comprises 32-256 bit fully decoded, static ram chips. Outputs, inputs, and addresses, are connected directly to form a 1024 byte memory bank. The least significant 8 bits of an address select a byte location within a chip. The next two bits are decoded to select one of 4 groups of 8 chips. Memory access must be less than 800ns to achieve a 1.6μs machine cycle. Operation of the memory merely comprises

comprises applying address levels. After the access time the addressed byte appears as DC levels on the memory output lines. Read is non destructive.

Writing is performed by applying address and write data followed by a write pulse (MWR).

Logic-4,5,6 show the details of the processor data flow paths as indicated in Figure 2.

Logic-9 shows the time pulse generator which controls timing of all operations. 8 time pulses are provided as shown by the timing diagram. Eight 100ns time pulses separated by 100ns would yield the minimum machine cycle of 1.6µs. In order to maintain the required display refresh cycle the time pulse oscillator should not be adjusted to cause machine cycles over 5µs.

A machine cycle comprises 8 time pulses. There are four different types of machine cycles. These four types are defined by the two bit register (YB4) shown in logic-10. The 4 possible states of this register activate 4 lines SØ, S1, S2, and S3. These levels define one of four types of machine cycles at any given time.

In normal operation only SØ and S1 types of machine cycles occur. SØ defines an instruction fetch machine cycle. SØ is combined with time pulses to cause a new instruction to be fetched from memory and placed in the I and N registers. S1 follows SØ and activates the instruction decoder (logic-10). One of 16 instruction lines (IØ-IF) is activated depending on the instruction code in I. This line is combined with the time pulses to cause the specified instruction function to be executed. Logic 11, 12, and 13 provide the proper data path control signals as a function of the type of machine cycle and instruction.

Normally the sequence SØ-S1-SØ-S1... is repeated causing alternating instruction fetch and execute cycles. The circuits shown in logic-7 permit this SØ-S1 sequence to be modified. External in/out request lines are provided for direct memory access by an external device. Activation of either line causes an S2 machine cycle to be inserted as follows, SØ-S1-SØ-S1-S2-SØ...

For an input request, S2 causes R(Ø) to address the memory and stores an input byte at the addressed location. R(Ø) is incremented by 1 so that a sequence of input bytes will be stored in sequential memory locations. For an output request, S2 fetches the byte at M(R(Ø)) and places it on the output bus. R(Ø) is again incremented by 1.

A program interrupt line is also provided (logic-7). Activating this line causes an S3 cycle to occur after the next S1 cycle. S3 causes S and P to be placed in T. P is then set to 1 and X to 2. Resumption of the normal SØ-S1 sequence will then result in execution of an interrupt routine specified by R(1).

Logic-8 shows the circuits used by the I6 instruction for selecting specific I/O devices. This logic is readily expanded to permit selection of additional devices.

Logic-14 and 15 illustrate the hex switch interface. A counter (VA1) continuously scans the 16 switches via a decoder (VCØ). Depression of a switch sets VD1, stopping the counter at its binary equivalent. The counter digit is placed in 4 bit register VAØ (logic-15) and switch scanning is resumed after appropriate bounce elimination delays (VEØ and VDØ).

A second switch depression again stops the counter at the switch count. This time, however, the processor is notified that an input byte is ready via either the EF1 bus or input request bus. After the byte is transferred to the processor, switch scanning is resumed.

Logic 16 shows the card reader interface. The card reader contains six photosensors (1, 2, 4, 8, P and C). Sensors 1, 2, 4, 8, and P provide outputs to the interface indicating the presence of data holes. The C photosensor monitors the center of the card in which holes are never punches. The C line indicates the presence of a card and is used to blank the hole sense line when no card is present. Top and bottom card notches insure integrity of the C sense line.

Each row of card holes read represents one hex digit. These are handled in the same manner as hex switch depressions. The four bit card digit is compared with the contents of the scanning register (VA1-Logic 14). A match generates a set X signal stopping VA1 at the hole pattern just read. This digit is transferred to register VAØ as before and scanning resumed. Byte accumulation, processor notification, and transfer are handled as before by the switch interface circuits.

The card reader logic also includes digit parity checking. The reader on switch disables bounce suppression delays in the hex switch logic. Switch entry should not be attempted with the reader switch on. Errors will result.

Logic-17 and 18 show the display interface logic. It is best understood by reference to the timing shown in Figures 17 and 18. A 12 bit counter is provided (AØ-A5, BØ-B5 in logic-18). Y and X ramp generators provide a display raster synchronized to 60$\mathcal{S}$. The counter triggering rate provides the timing shown. Bytes are requested from the processor at the times shown. All bytes for a 32 (or 64) bit line are requested and stored in ZA1 (logic-17) prior to each line sweep (X ramp). During the line sweep 4 bits at a time are fetched from AZ1 and the individual bits used to modulate the beam at the proper bit times. At the end of the 32nd line sweep a program interrupt request is generated.

The processor interrupt program is responsible for initializing the display address register R(ϕ).

Logic-19 shows the interface circuits for the auxiliary "yes-no" switch box. Pressing a switch sets the appropriate flag. The program is responsible for sensing and resetting the flags at appropriate times.

One additional feature is included in the system. This is an audio monitor circuit to which an external speaker may be connected via the monitor plug. Bit ϕ of input/output bytes set and reset the RC1 latch.

Section V - Sample Programs

The examples presented in this section were selected to clarify the operation of the system. Examples of input-output modes and interrupt processing are provided. All examples are in machine language. The coding form was designed for use with this order code and system. Each sheet provides space for 48 bytes or instructions. After coding, card numbers can be added and cards prepared directly from the coding form. Each 12 byte card trace is numbered separately.

The "S" column can be used to name specified memory locations (subroutine entry points, table base addresses, etc.). The "P" column facilitates keeping track of the register being used as the P counter for each section of code. The "X" column can be used to indicate which register has been assigned to X. The "M" column should contain consecutive memory addresses. The (M) column represents the contents of each memory location. This column represents the sequence of bytes to be placed in memory. The last two columns facilitate self documentation of programs.

Program SA-deduce is shown in Figure 19 and the following two coding sheets. Deduce illustrates the use of the switches and uses the 8 bus bit lights as output. The player first loads the program starting at M(ØØØØ) as described in Section III. He then chooses one of the 8 lights (Ø-7). Pressing Cl, ST, and RS in sequence initiates the program. First, the light pattern "1111ØØØØ" is displayed. The player responds by entering ØØ via the hex switch panel if his chosen light is off and "11" if it's on. The program displays two more patterns of lights, waiting for a player switch response following each pattern. After the last (third) player response the program deduces the chosen light, turns it on, and stops. A CL-ST-RS sequence permits the game to be repeated.

The program requires only 7Ø bytes. This is relatively good memory utilization since only 18 average (4 byte) 360 type machine instructions could be provided in the same memory space.

The program flow as shown in Figure 19 is straightforward. The section of code M1 partially initializes registers. Since R(Ø) will be subsequently required as a hex switch input byte storage pointer the program counter is changed to R(5) leaving R(Ø) free. Initialization is resumed at M2. The hex switches are selected and set to the direct mode. As discussed previously, the direct mode causes an automatic memory store cycle (at M(R(Ø))) to occur for each byte entered (2 hex digits).

The idle instruction (1∅) at M3 halts the program until an input byte storage cycle occurs. Following the stolen input byte storage cycle, the instruction following idle is fetched and executed. Since R(∅) is automatically incremented during each input byte storage cycle it must be decremented to point to the byte just entered. An idle instruction always causes M(R(N)) to be placed on the byte bus. The idle instruction here, causes M(R(1)) to be placed on the bus. The 8 bus lights therefore display the contents of M(R(1)) for the duration of the idle instruction. In this manner the 3 bit patterns at MA are displayed sequentially as required.

The storage of an input byte causes program resumption (after idle) and the newly entered byte at IA is tested. From the responses to the 3 light patterns the number of the chosen light is readily determined. This is converted to a single light pattern via table MT. The use of an IC type of instruction to perform this table look up is illustrated in program segment M7.

Deduce requires 6 card tracks and is completely contained on 3 cards.

It should be again noted that while programs are normally loaded starting at M(∅∅∅∅), normal initiation (CL-ST-RS sequence) always skips M(∅∅∅∅). Program execution therefore begins at M(∅∅∅1) leaving M(∅∅∅∅) free to be used in any manner desired.

The next sample program (S3-display) illustrates the use of the scope display and program interrupt. This program is shown in Figure 20 with detailed coding on the following two sheets. It requires 2∅8 bytes of memory, 128 of which are required as display refresh storage. The program itself requires only 8∅ bytes and can be contained on 3.5 cards.

128 bytes (1024 bits) of memory are displayed in a 32x32 bit format on the scope. M(Ø1ØØ-Ø17F) is used as the area of memory displayed. Figure 3 illustrates the layout of the 128 bytes on the display CRT.

The program is loaded starting at M(ØØØØ). It is initiated at M(ØØØ1) by a CL-ST-RS switch sequence. The display will be blank at this point. The operator may now cause any pattern of dots desired to be displayed. First two hex digits are entered (via hex switches/card reader) specifying a displayed byte address. The 2 digit addresses of the displayed bytes are shown in Figure 3. (Figure 4 provides a convient form for laying out desired 32x32 display patterns). A second byte is then entered representing the actual 8 bit pattern to be displayed at the previously entered address. This second set of 8 bits will immediately appear on the display screen. Repeating the above permits construction of any dot pattern on the display screen.

Code M1,M2 initializes registers and sets a 1 bit counter (K) to Ø. K specifies whether an input byte specifies a display address or an 8-bit display pattern. Display refresh requires R(Ø) for cycle stealing and R(1) and R(2) for program interrupt. M1 code therefore changes the program counter from R(Ø) to R(3) and resumes initialization at M2.

The C1 loop initially clears the memory area to be displayed. M3 indicates the use the I6 type of instruction to activate the display interface. From this point on, the display interface will automatically steal memory cycles to retrieve memory bytes for display refresh. R(Ø) is used for this purpose. Display generated program interrupts are also required to reset R(Ø) to the beginning of the memory display area after 128 bytes have been refreshed. This program interrupt will be ignored until the interrupt mask (IM) is reset. IM is always set when the machine is cleared. IM reset is performed by a dummy "7∅" instruction in M3. Note that P and X are set by this instruction so that careful use of this instruction is required.

After activation of the output display, the hex switch/reader input channel is selected. Note that, once initiated, output display operation is independent of external channel selection. The input channel is now set to the program mode and EF1 is monitored in loop M4 to determine when A input byte is available to be stored. The stored input byte is then used to modify R(A) or A displayed byte ~~A~~ M(R(S)) depending on the value of X. Input monitoring is then resumed until a new input byte.

Since the output display causes program interrupts an interrupt routine is provided (I1-I2-I3). This routine is automatically entered when an interrupt occurs. P is set to 1 and X to 2 by the interrupt. Instruction "78" in I1 stores X and P values of the interrupted program which are contained in T. Instruction "7∅" in I3 restores the original values of X and P thereby effecting return to the interrupted routine. This instruction also resets the interrupt mask which is set each time an interrupt occurs.

A side benefit resulting from the simple machine structure is the ability to store its state in two bytes (D, X, and P).

Because of the limited instruction set, extensive use of subroutines is anticipated. This is not illustrated in the above examples. The machine structure facilitates branch and link functions. One method is described below. For example, the following register conventions might be established:

R(3) - Main program counter (ØØAØ)

R(4) - Call routine pointer (Ø1Ø1)

R(5) - Subroutine program counter (Ø2XX)

If ( ) represents initial R(3), R(4), and R(5) contents with R(3) pointing to a subroutine call instruction, then the following illustrates one possible call and return sequence:

Main Program (P3)

M Location	Contents	Instruction	Comments
ØØAØ	D4	4→P	Jump to call routine
ØØA1	2Ø	---	Subroutine identifier
ØØA2	P1	---	Parameter 1
.	.	.	
.	.	.	
.	.	.	

Call Routine (P4)

M Location	Contents	Instruction	Comments
Ø1ØØ	D3	3→P	Return to main pro
Ø1Ø1	43	M(R(3))→D, R(3)+1	Put subroutine
Ø1Ø2	A5	D→RØ(5)	Identifier in R(5)
Ø1Ø3	D5	5→P	Jump to subroutine
Ø1Ø4	3Ø	ØØ→RØ(P)	Go to Ø1ØØ
Ø1Ø5	ØØ	---	---

Subroutine (P5)

M Location	Contents	Instruction	Comments
Ø22Ø	43	M(R(3))→D, R(3)+1	Get P1 from M (ØØA2)
.	.	.	.
.	.	.	.
.	.	.	.
Ø232	D4	4→P	Return to M (Ø1Ø4)

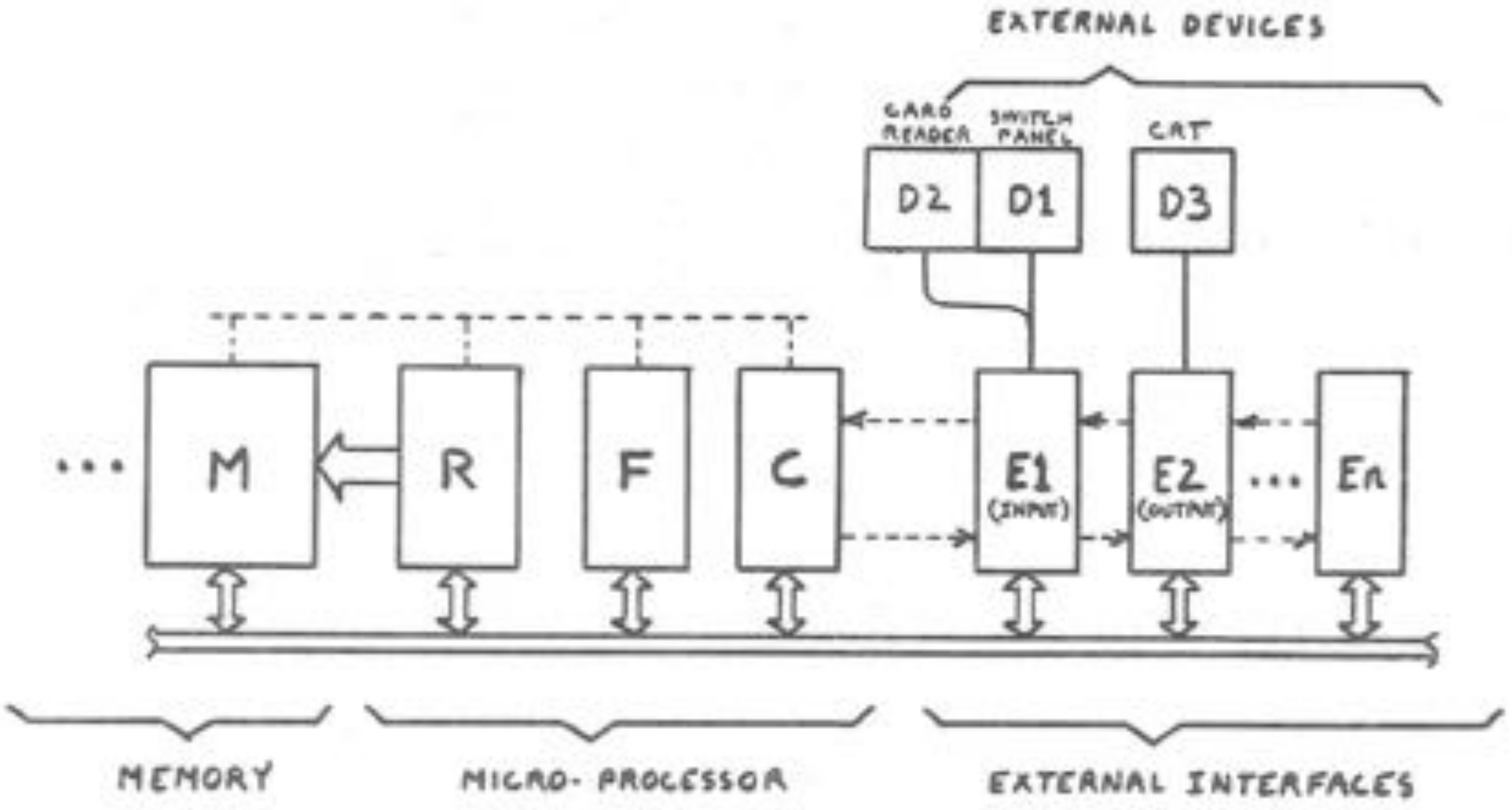
Using the above system requires only two program bytes to call a subroutine. These two bytes can then be followed by as many one byte subroutine parameters as required. It should be noted that the use of one byte instructions permits a calling routine of only six bytes. Other subroutine calling techniques could, of course, be used.

The above system also solves the problem of branching between 256 byte mini-pages. One subroutine can be a "go to" subroutine. A "go to" function would then be specified as follows:

DN, GT, XX, YY.

This four byte sequence would enter a call routine specified by R(N) which in turn selects the "go to" subroutine specified by "GT". The "go to" subroutine would place "XX" and "YY" into the original program counter and return to this "go to" address (via the call routine as above). In this manner a branch to any memory location would only require four bytes.

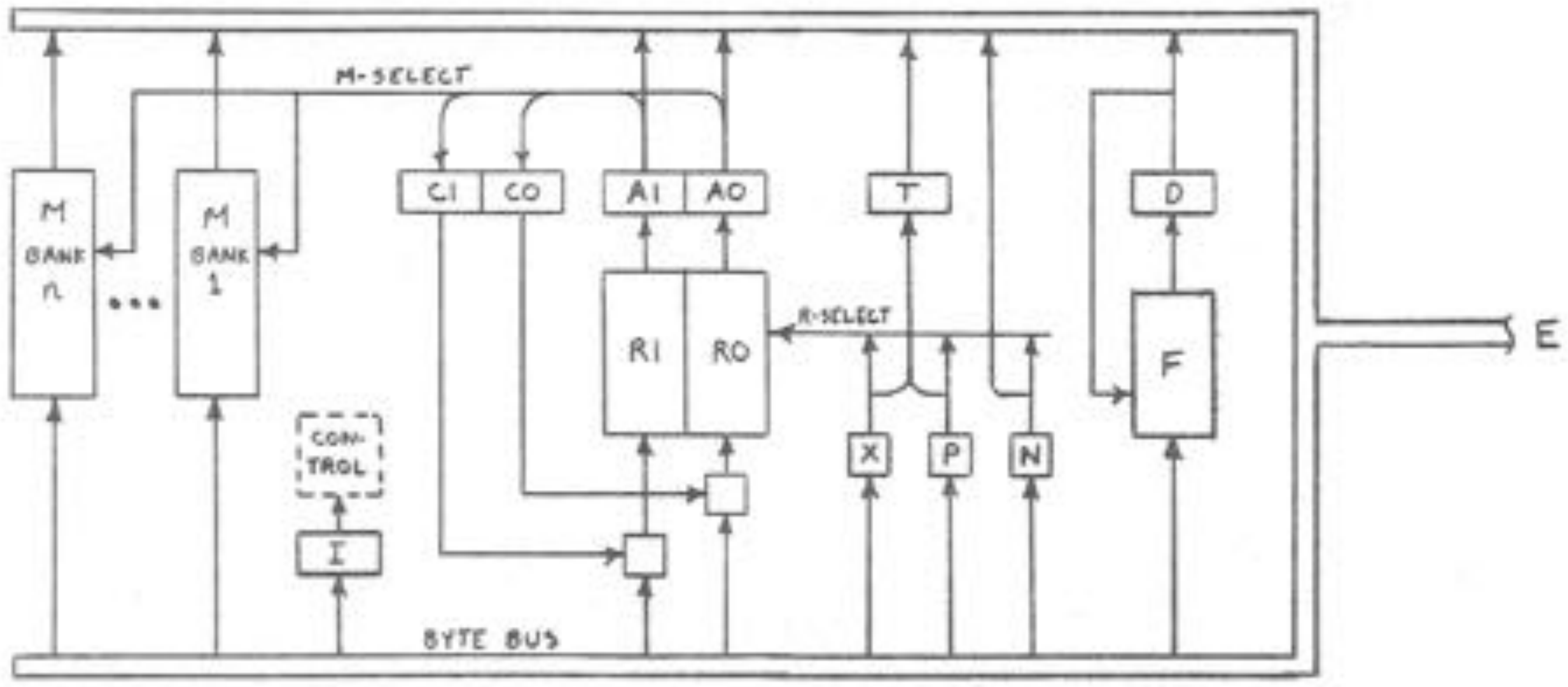
It is advisable to set up sub-routine calling conventions whenever the program exceeds 256 bytes.



SYSTEM BLOCK DIAGRAM

FIGURE 1

JAW



MICRO-PROCESSOR
BLOCK DIAGRAM
FIGURE 2

JAW

32X32 DISPLAY

	D1 D∅	D1 D∅	D1 D∅	D1 D∅
	7 6 5 4 3 2 1 ∅	7 6 5 4 3 2 1 ∅	7 6 5 4 3 2 1 ∅	7 6 5 4 3 2 1 ∅
		↓		
∅	∅3	∅2	∅1	∅∅
	∅7	∅6	∅5	∅4
	∅B	∅A	∅9	∅8
	∅F	∅E	∅D	∅C
1	13	12	11	1∅
	17	16	15	14
	1B	1A	19	18
	1F	1E	1D	1C
2	23	22	21	2∅
	27	26	25	24
→	2B	2A	29	28
	2F	2E	2D	2C
3	33	32	31	3∅
	37	36	35	34
	3B	3A	39	38
	3F	3E	3D	3C
4	43	42	41	4∅
	47	46	45	44
	4B	4A	49	48
	4F	4E	4D	4C
5	53	52	51	5∅
	57	56	55	54
	5B	5A	59	58
	5F	5E	5D	5C
6	63	62	61	6∅
	67	66	65	64
	6B	6A	69	68
	6F	6E	6D	6C
7	73	72	71	7∅
	77	76	75	74
	7B	7A	79	78
	7F	7E	7D	7C

EXAMPLE: ↓
→• = BIT "2" OF BYTE AT M LOCATION "2A"

FIGURE 3

JAW

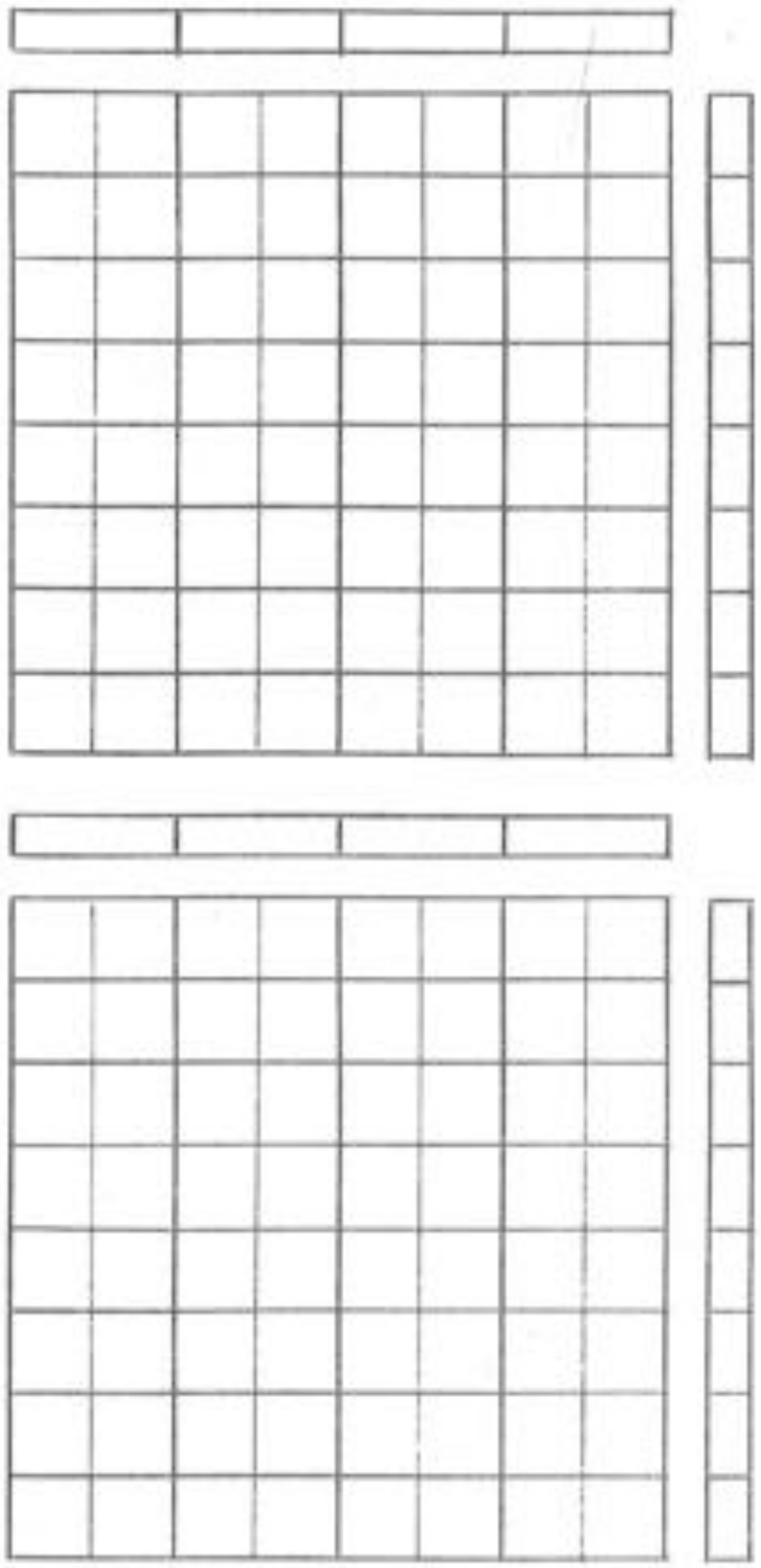
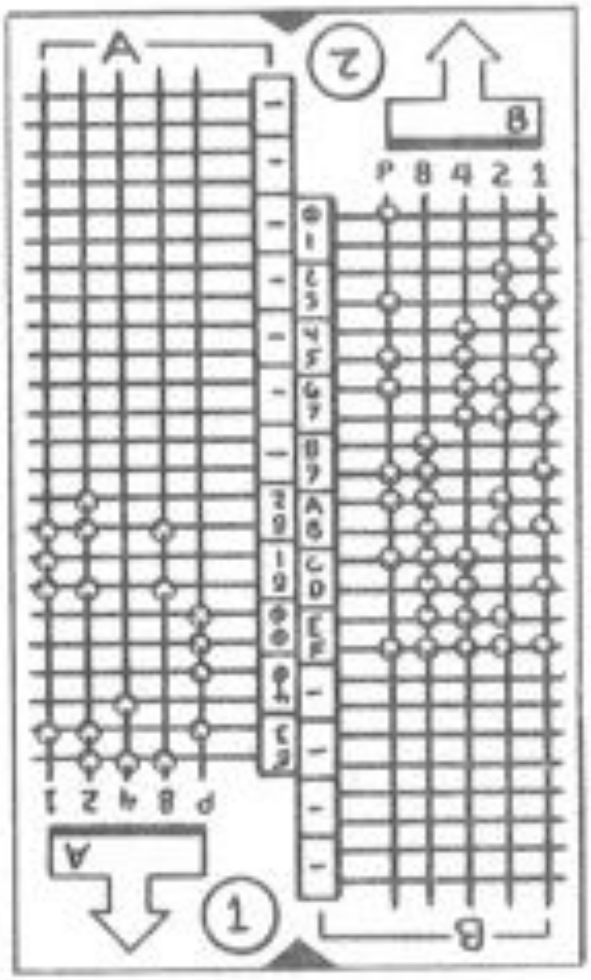


FIGURE 4

JAW



CARD FORMAT

FIGURE 5

JAW

BASIC MICROPROCESSOR- ORDER CODE SUMMARY

0 N	IDLE	D N	N → P
1 N	R(N)+1	E N	N → X
2 N	R(N)-1	F 0	M(R(X))→D
4 N	M(R(N))→D,R(N)+1	F 1	M(R(X))/D→D
5 N	D→M(R(N))	F 2	M(R(X))&D→D
8 N	R0(N)→D	F 3	M(R(X))⊕D→D
9 N	R1(N)→D	F 4	M(R(X))plus D→D
A N	D→R0(N)	F 5	M(R(X))minus D→D
B N	D→R1(N)	F 6	SHIFT D RIGHT ONE BIT
C N	DD →R0D(N)	F 7	SPARE
6 3	RESET EF3 & EF4,R(X)+1	6 8	INPUT BYTE → M(R(X))
7 8	T→M(R(X))	7 0	M(R(X))→XP,R(X)+1,RESET IM

3 0	Y	Y → R0(P) UNCONDITIONAL BRANCH
3 1	Y	Y → R0(P) IF D≠0
3 2	Y	Y → R0(P) IF D=0
3 3	Y	Y → R0(P) IF DF=1
3 4	Y	Y → R0(P) IF EF1=1 (INPUT BYTE READY)
3 5	Y	Y → R0(P) IF EF2=1
3 6	Y	Y → R0(P) IF EF3=1 (YES SWITCH)
3 7	Y	Y → R0(P) IF EF4=1 (NO SWITCH)

6 1	& M(R(X))=	0 1	SELECT INPUT (CARD/SWITCH),R(X)+1
6 2	& M(R(X))=	0 1	SET SELECT INPUT TO PROGRAM MODE,R(X)+1
6 2	& M(R(X))=	0 2	SET SELECT INPUT TO DIRECT MODE,R(X)+1
6 1	& M(P(X))=	0 2	SELECT OUTPUT (DISPLAY),R(X)+1
6 2	& M(R(X))=	0 1	SET SELECT OUTPUT TO 32x32 MODE,R(X)+1
6 2	& M(R(X))=	0 2	SET SELECT OUTPUT TO 16x64 MODE,R(X)+1

FIGURE 6

JRW

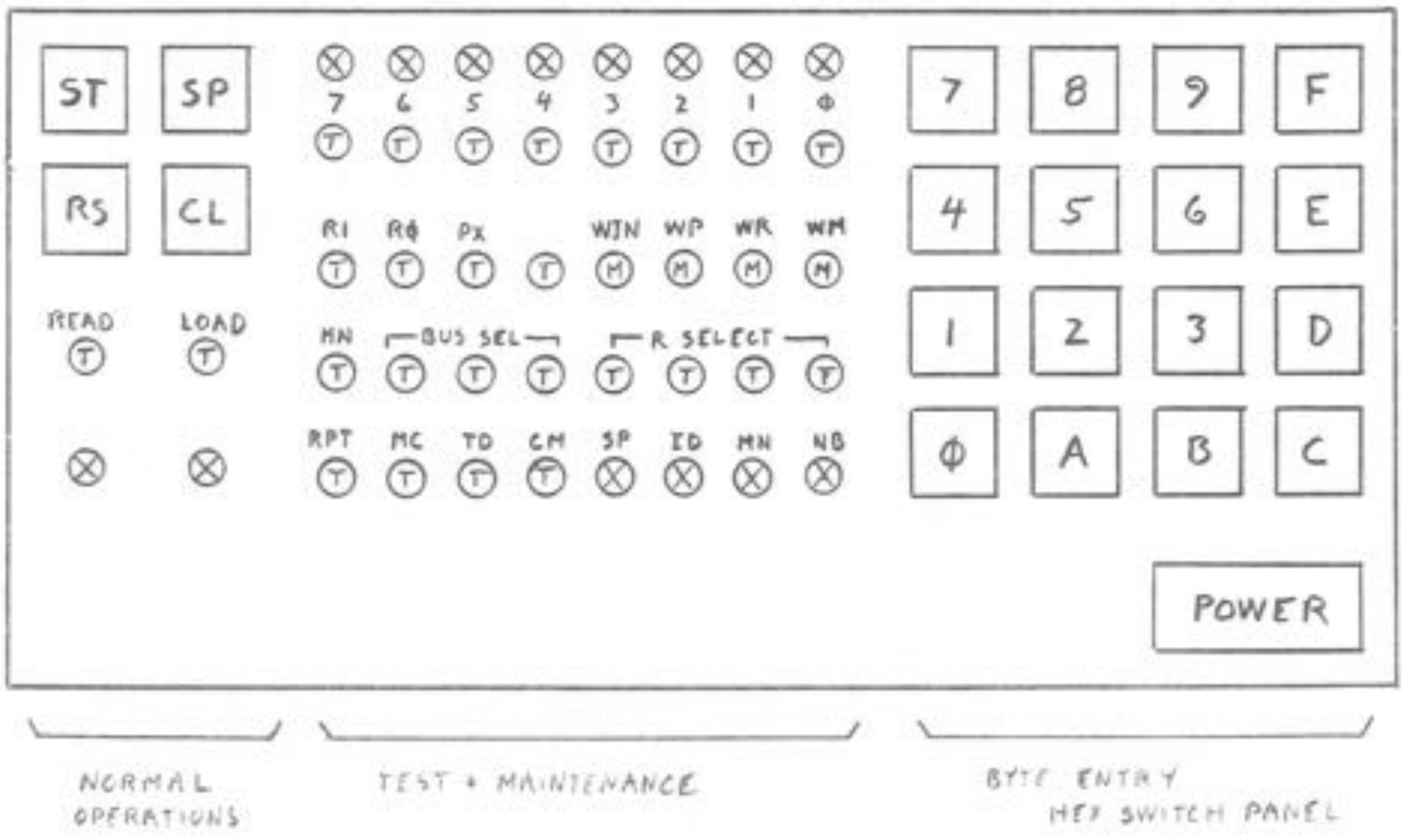


FIGURE 7 CONTROL PANEL

JAW

CHIP TYPE	M	P	IO	TOT
7400 4x2 NAND	—	24	12	36
7401 4x2 NAND (O.C.)	—	3	6	9
7402 4x2 NOR	—	6	4	10
7404 HEX INVERT	—	8	6	14
7408 4x2 AND	—	3	1	4
7410 3x3 NAND	—	7	5	12
7420 2x4 NAND	—	4	1	5
7427 3x3 NOR	—	2	—	2
7430 1x8 NAND	—	1	—	1
7474 2 D FF	—	2	4	6
7475 4 LATCH	—	9	—	9
7486 QUAD EXCLUSIVE OR	—	—	1	1
7489/8225 4x16 RAM	—	4	1	5
7493 4 BIT CTR.	—	1	4	5
7495 4 BIT L/R SHIFT	—	2	1	3
7496 5 BIT REG.	—	—	2	2
74121 ONE SHOT	—	3	10	13
74151 8 BIT SELECT	—	9	1	10
74154 4 TO 16 DECODE	1	1	1	2
74155/M4006 3→8 DECODE	—	2	2	4
74175 QUAD D FF	—	3	—	3
74180 PARITY CHECKER	—	—	1	1
74181 4 BIT ALU	—	2	—	2
74193 4 BIT +/- COUNTER	—	4	—	4
SIG. 2501 B. 256 BIT MEMORY	32	—	—	
	33	100	63	196

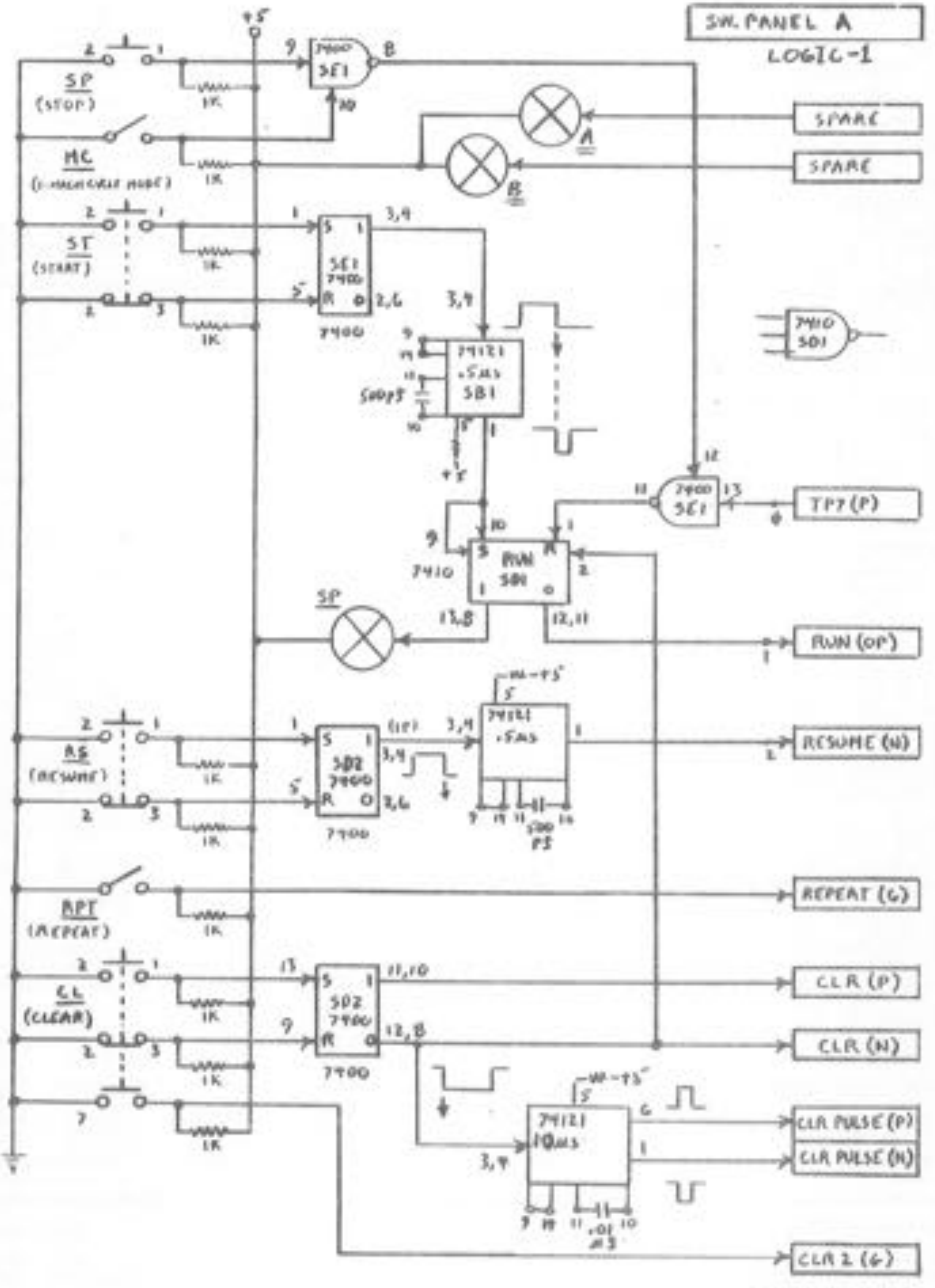
TABLE I

SYSTEM CHIP COUNT

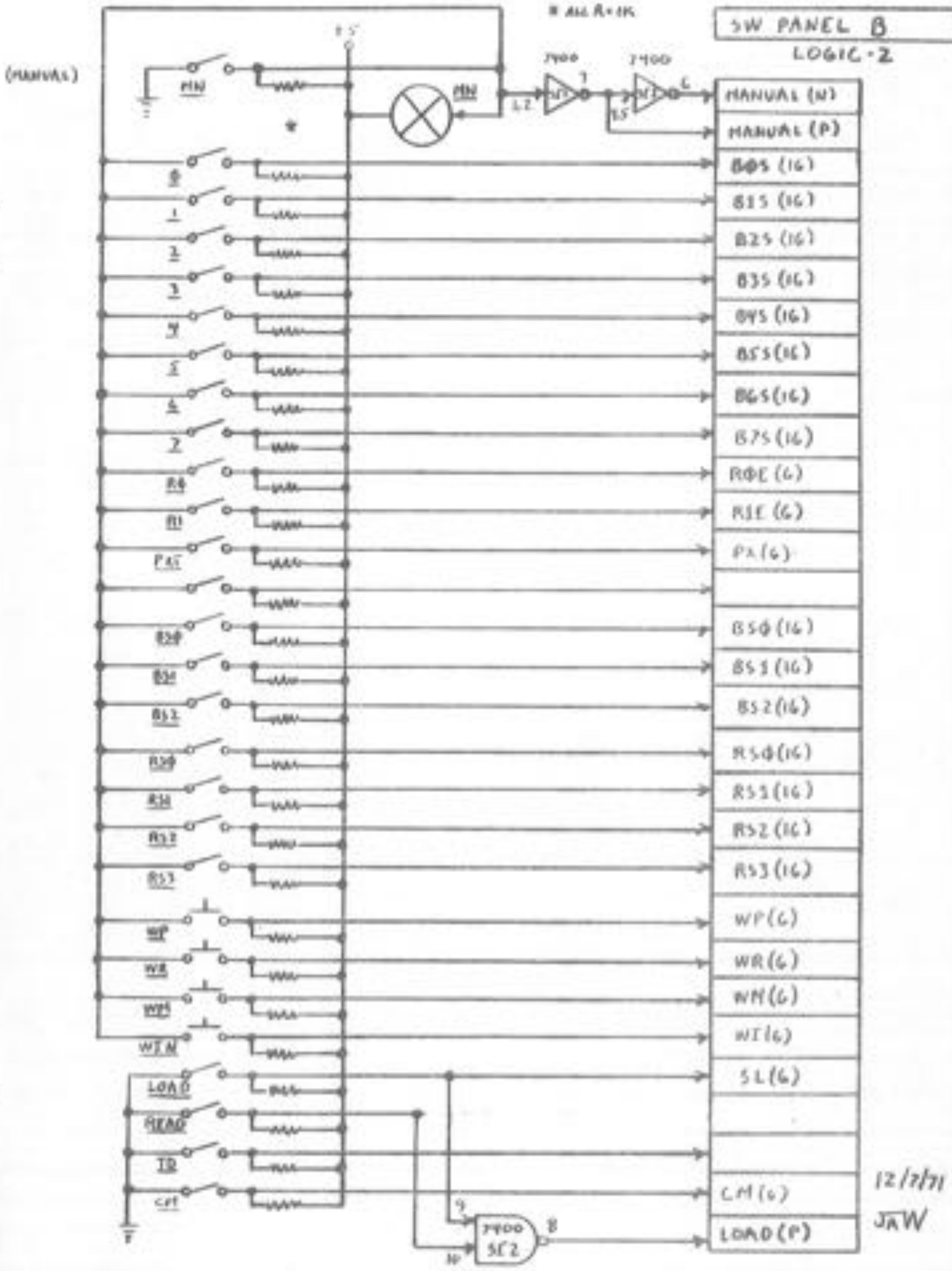
JAW 12/7/71

SW. PANEL A

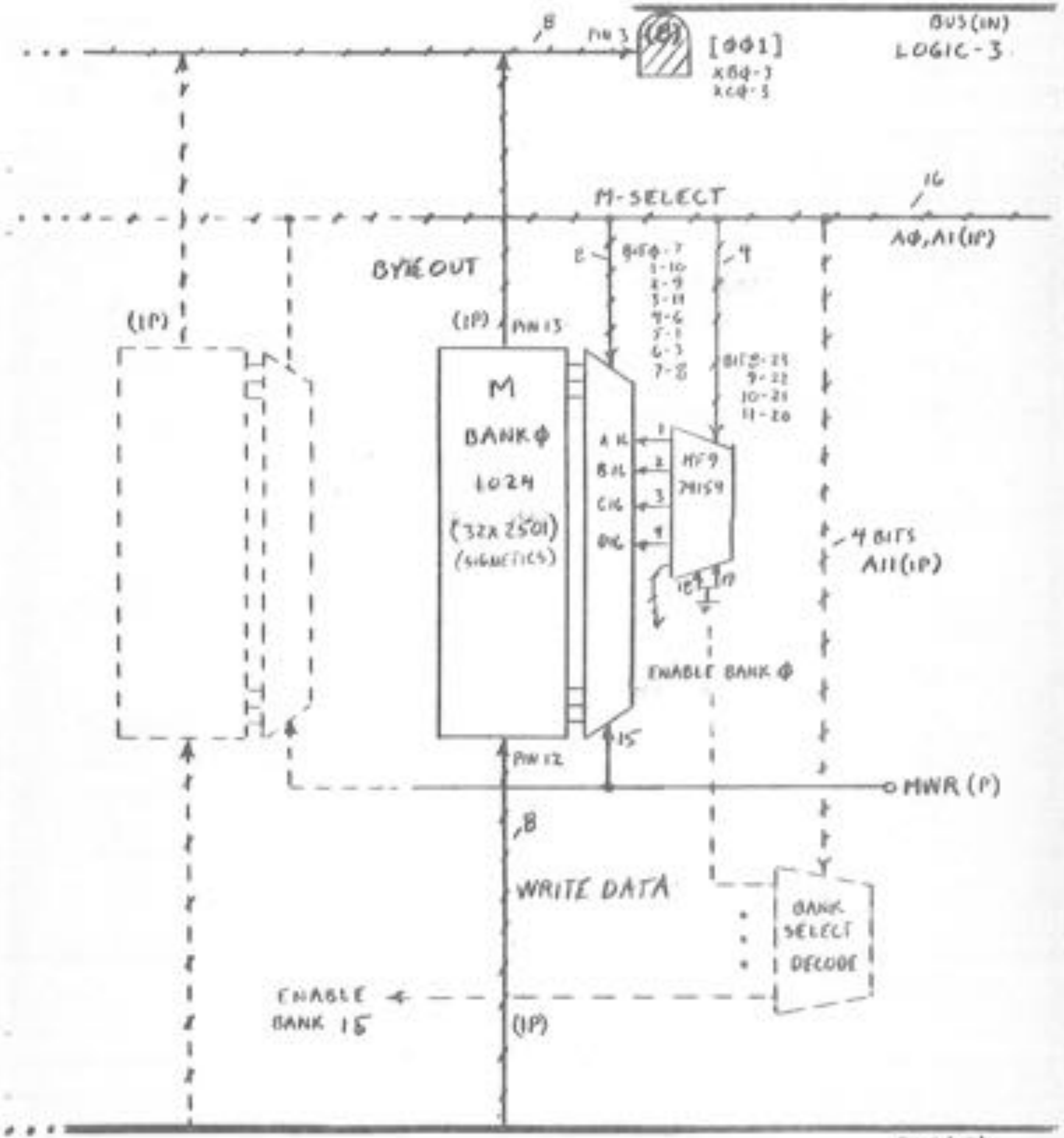
LOGIC-1



JAW 10/28/71


* ALL R = 1K
SW PANEL B
LOGIC - 2
(MANUAL)
+5
MN
MN
7400
7400
L2
1
B5
6
*
MANUAL (N)
MANUAL (P)
0
1
2
3
4
5
6
7
R0
R1
PXT
BS0
BS1
BS2
RS0
RS1
RS2
RS3
WP
WR
WM
WIN
LOAD
READ
TD
CM
B0S (16)
B1S (16)
B2S (16)
B3S (16)
B4S (16)
B5S (16)
B6S (16)
B7S (16)
R0E (6)
R1E (6)
PX (6)
BS0 (16)
BS1 (16)
BS2 (16)
RS0 (16)
RS1 (16)
RS2 (16)
RS3 (16)
WP (6)
WR (6)
WM (6)
WI (6)
SL (6)
CM (6)
9
10
7400
5E2
8
LOAD (P)
12/7/71
JRW

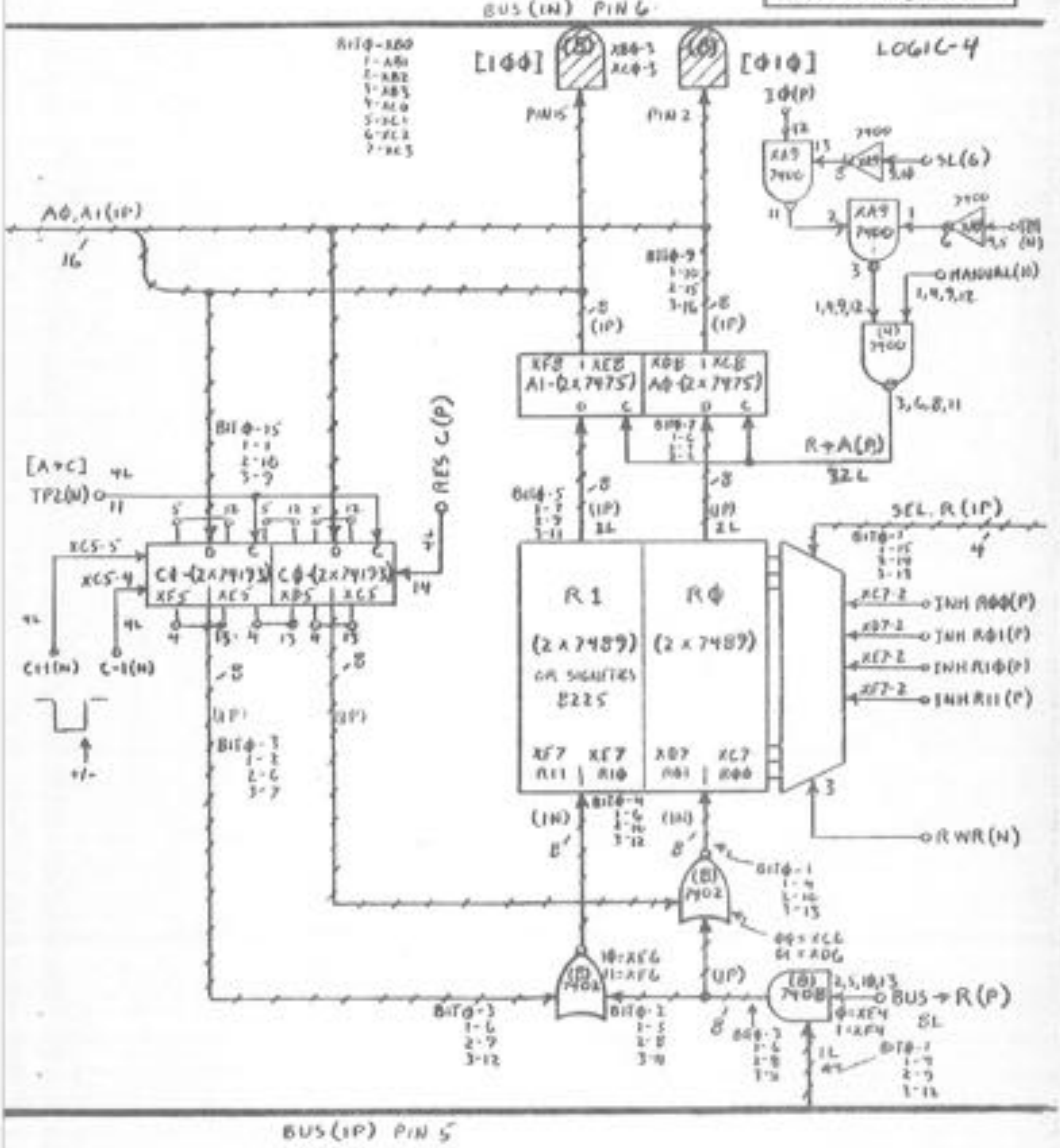
MAIN MEMORY



10/28/71
JAW

REGISTER STACK

LOGIC-4

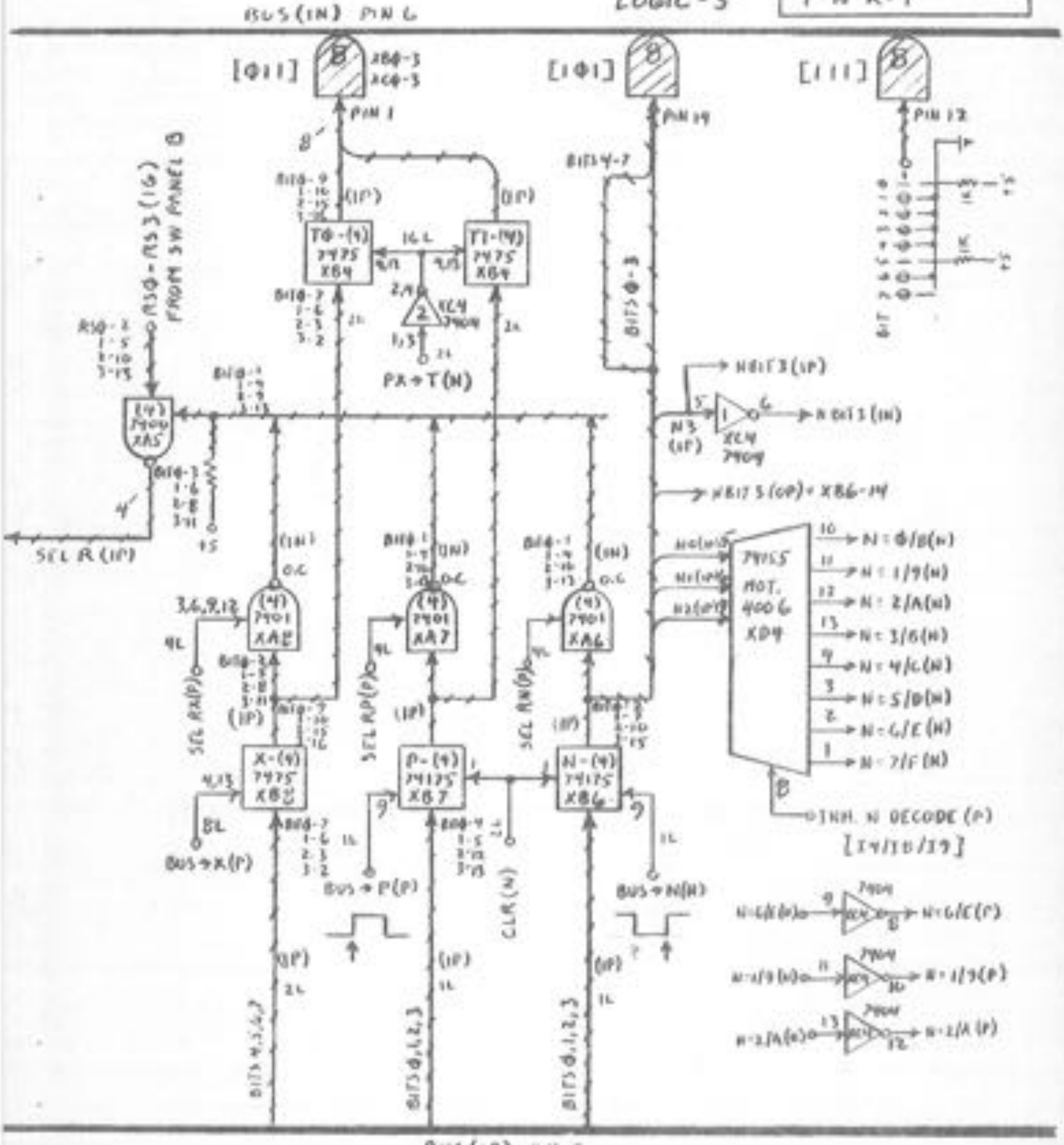


10/28/71

JAW

LOGIC-5

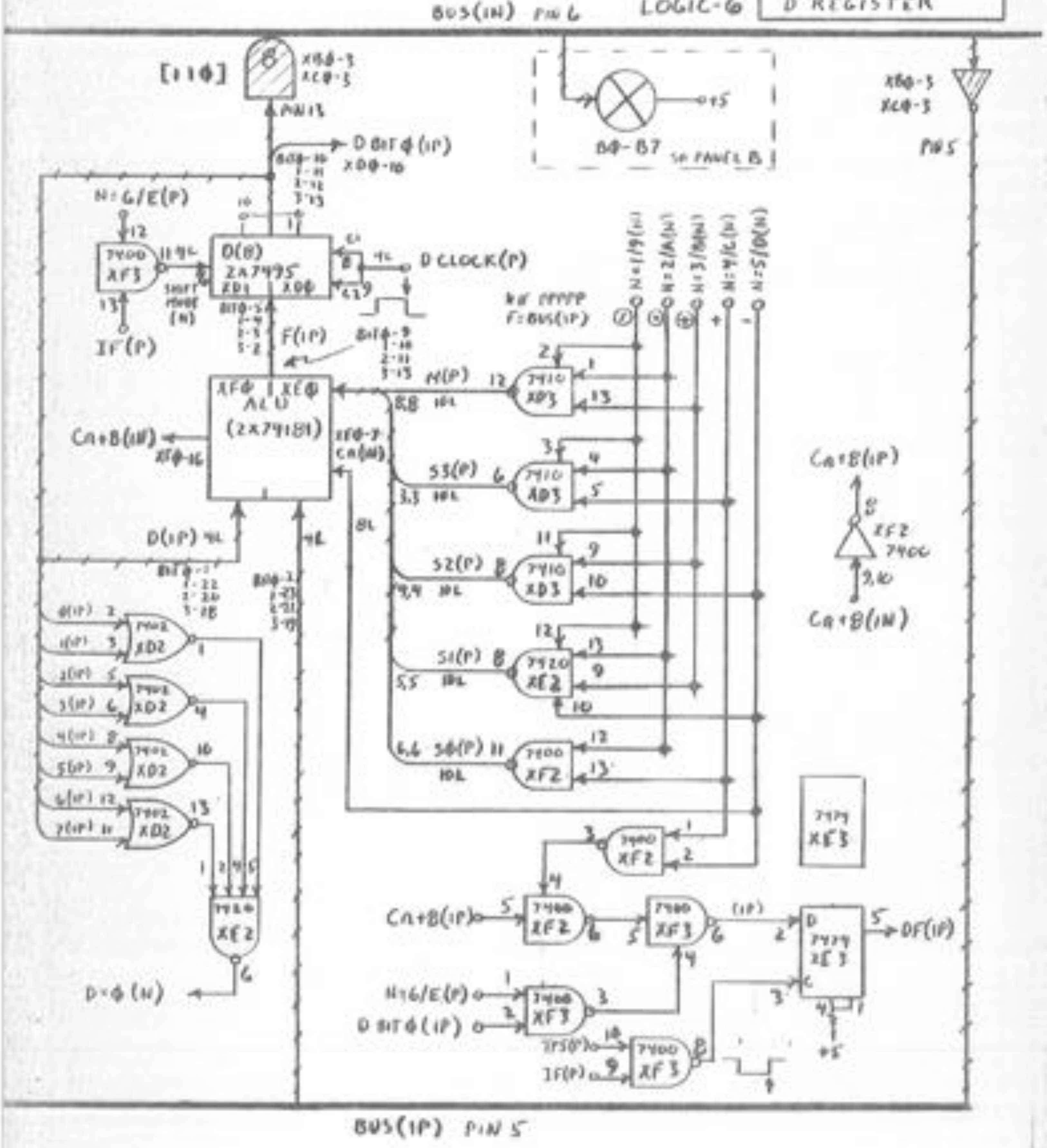
P-N-X-T



10/28/71

JAW

LOGIC-6 D REGISTER

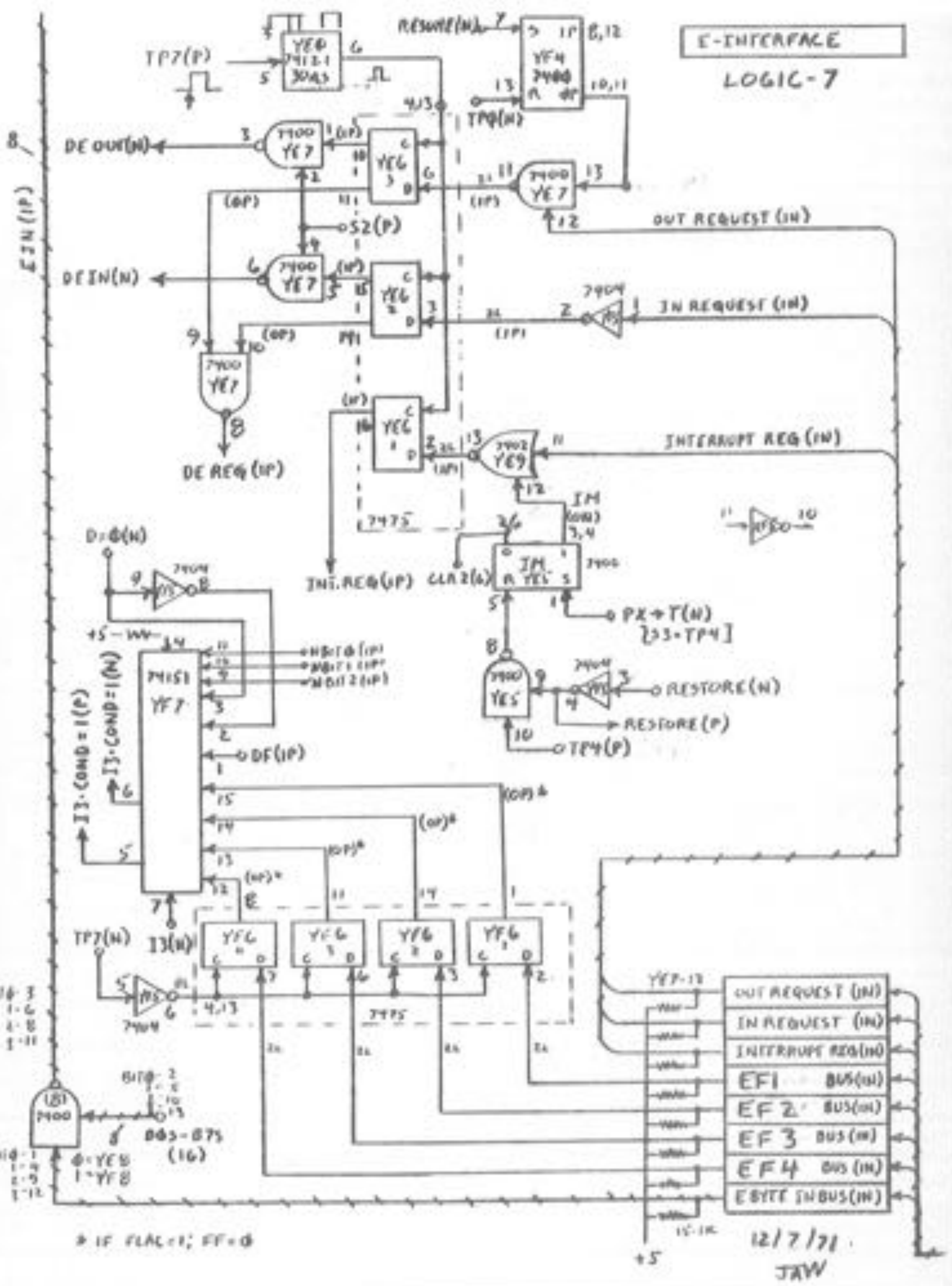


10/28/71

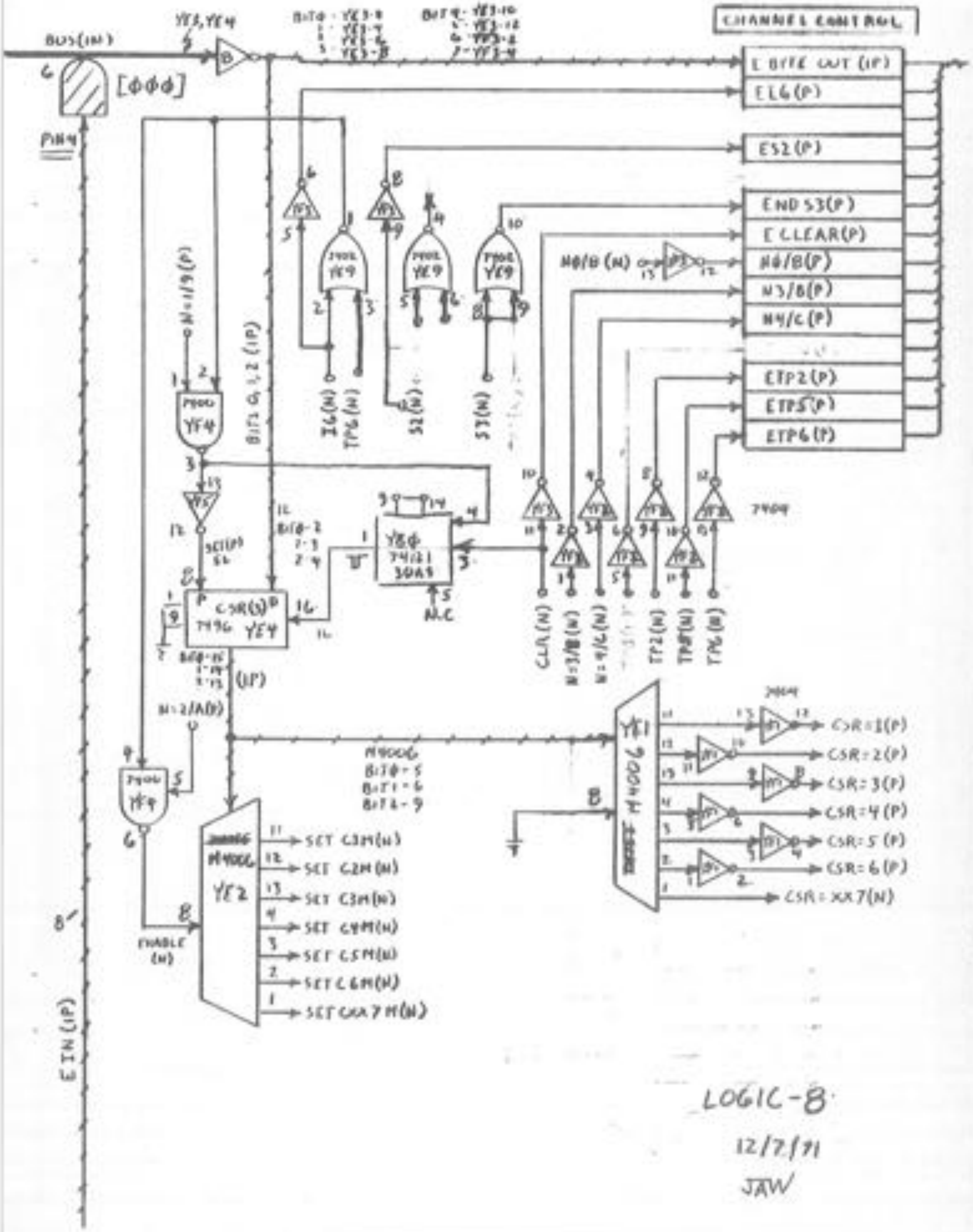
JAW

E-INTERFACE

LOGIC-7



12/7/71
JAW



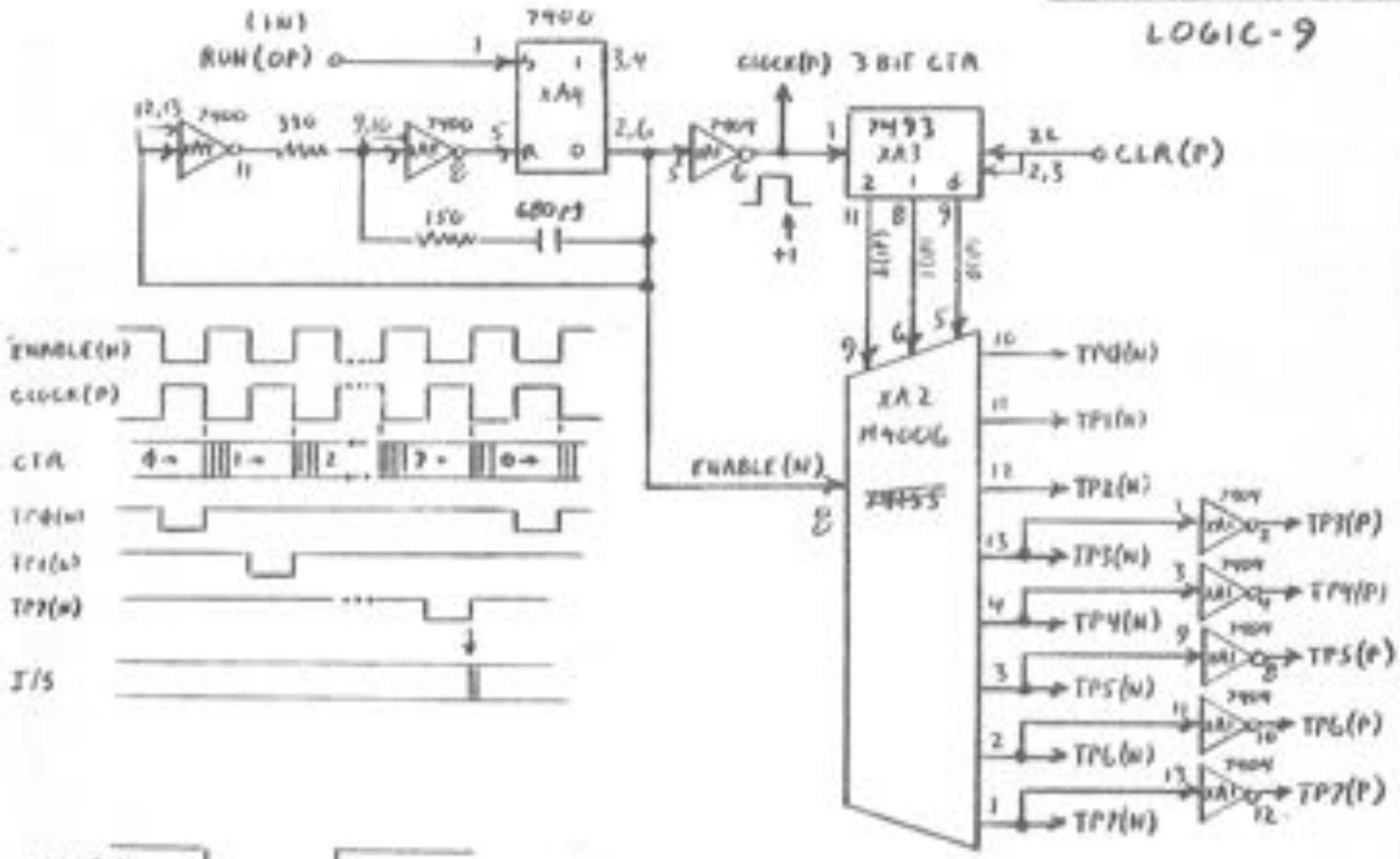
LOGIC-8

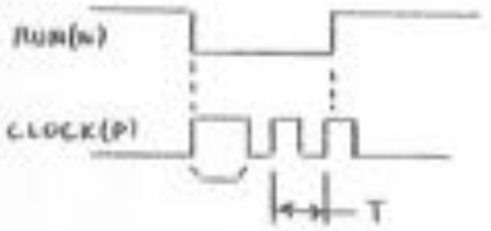
12/7/71

JAW

TP GENERATOR

LOGIC-9





FIRST PULSE ≈ 2 × WIDTH OF SUCCESSORS

C = 680pF FOR T ≈ 400ns

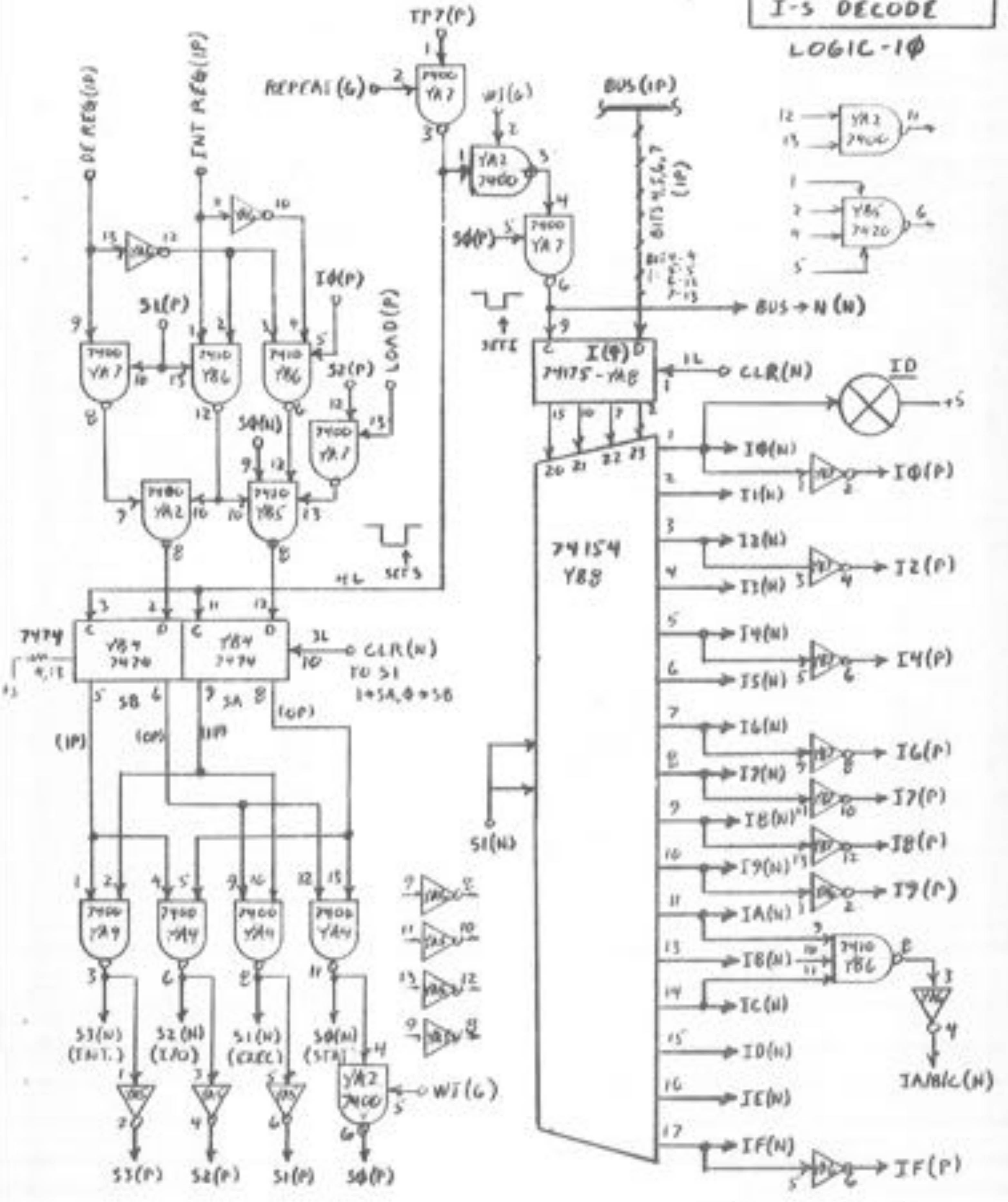
C = 240pF FOR T ≈ 200ns (5MC)

10/28/71

JAW

I-S DECODE

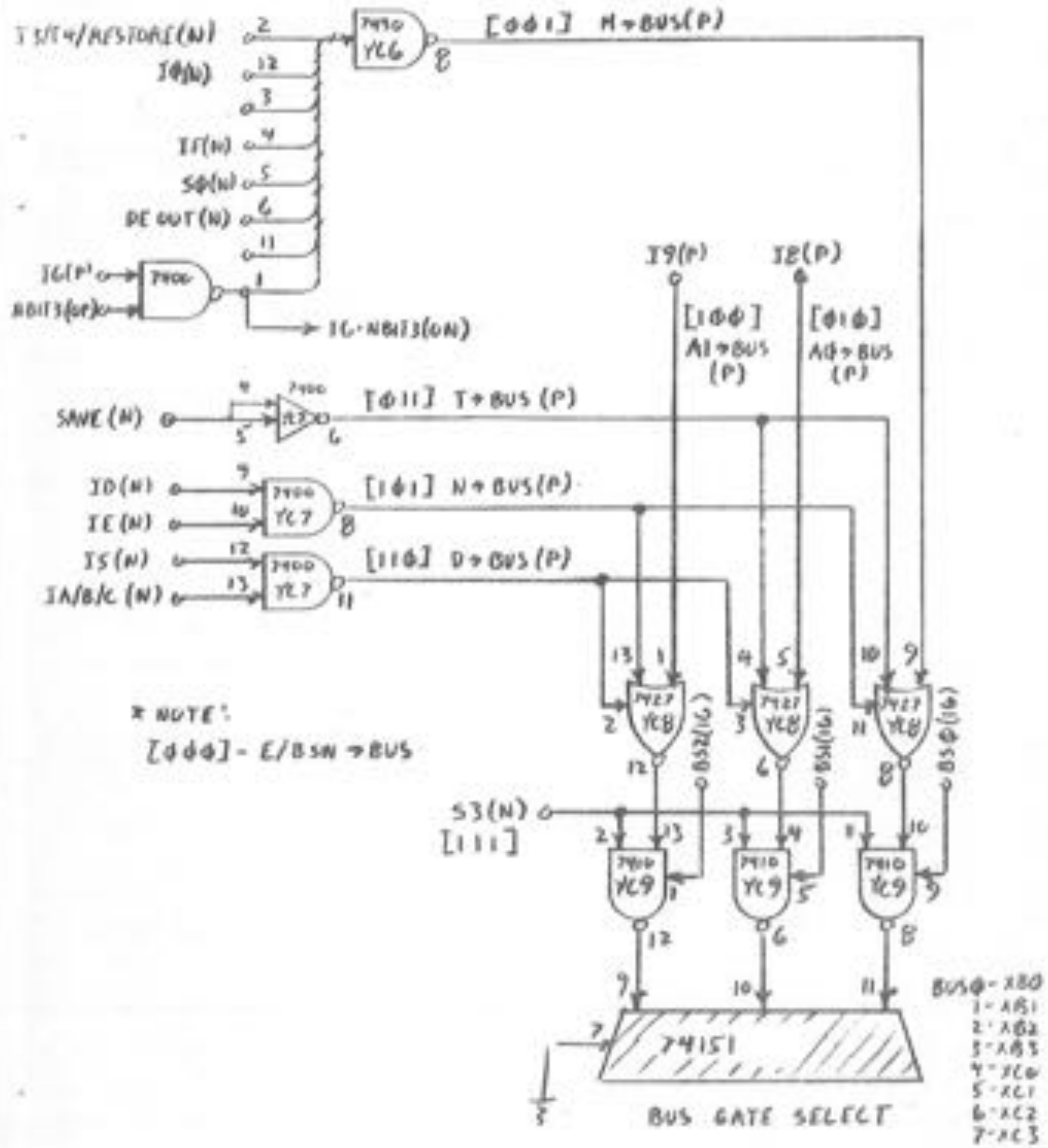
LOGIC-1Ø



10/28/71 JAW

BUS SELECT

LOGIC-11



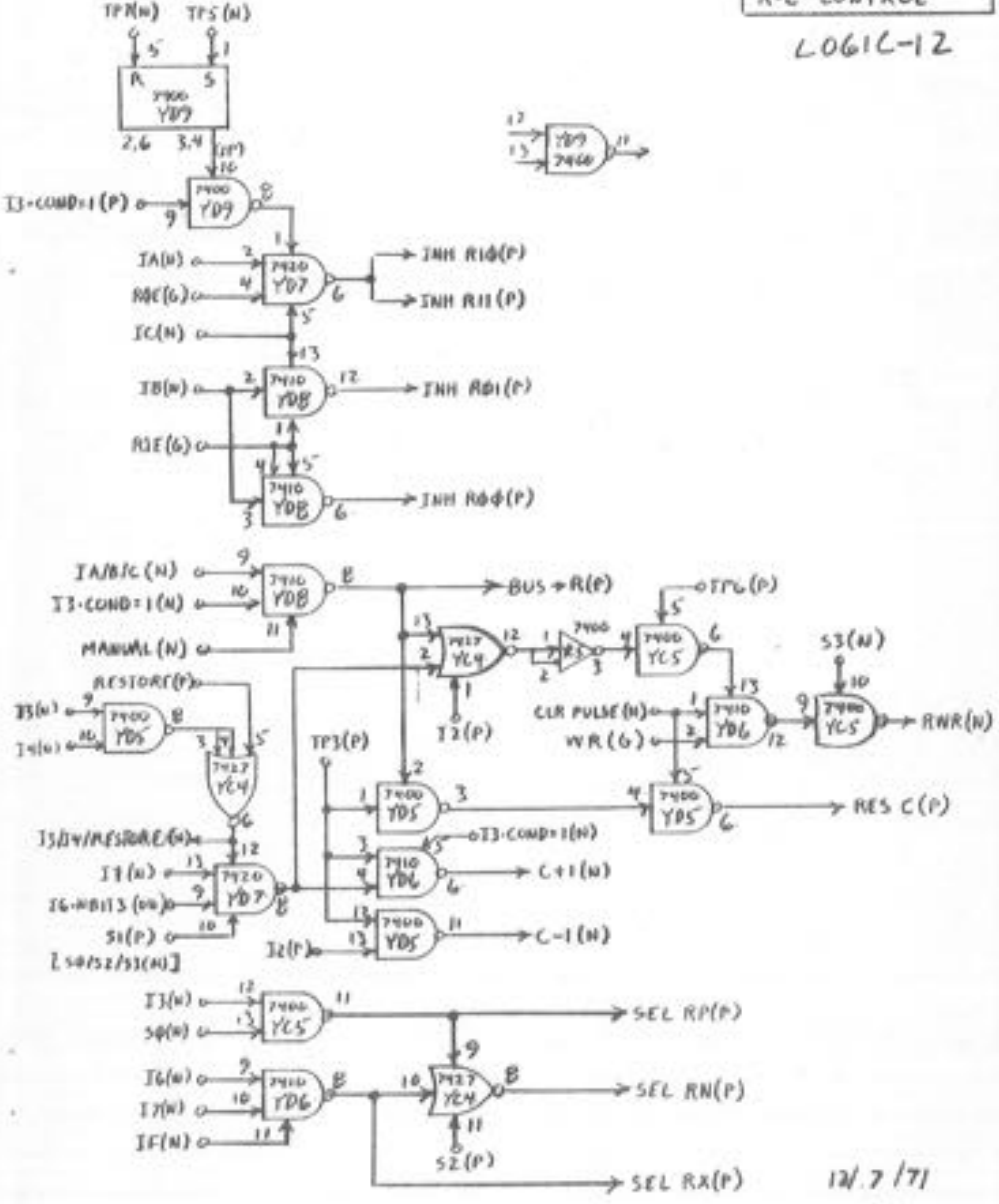
* NOTE:
[∅∅∅] - E/BSW → BUS

10/28/71

JAW

R-C CONTROL

LOGIC-12

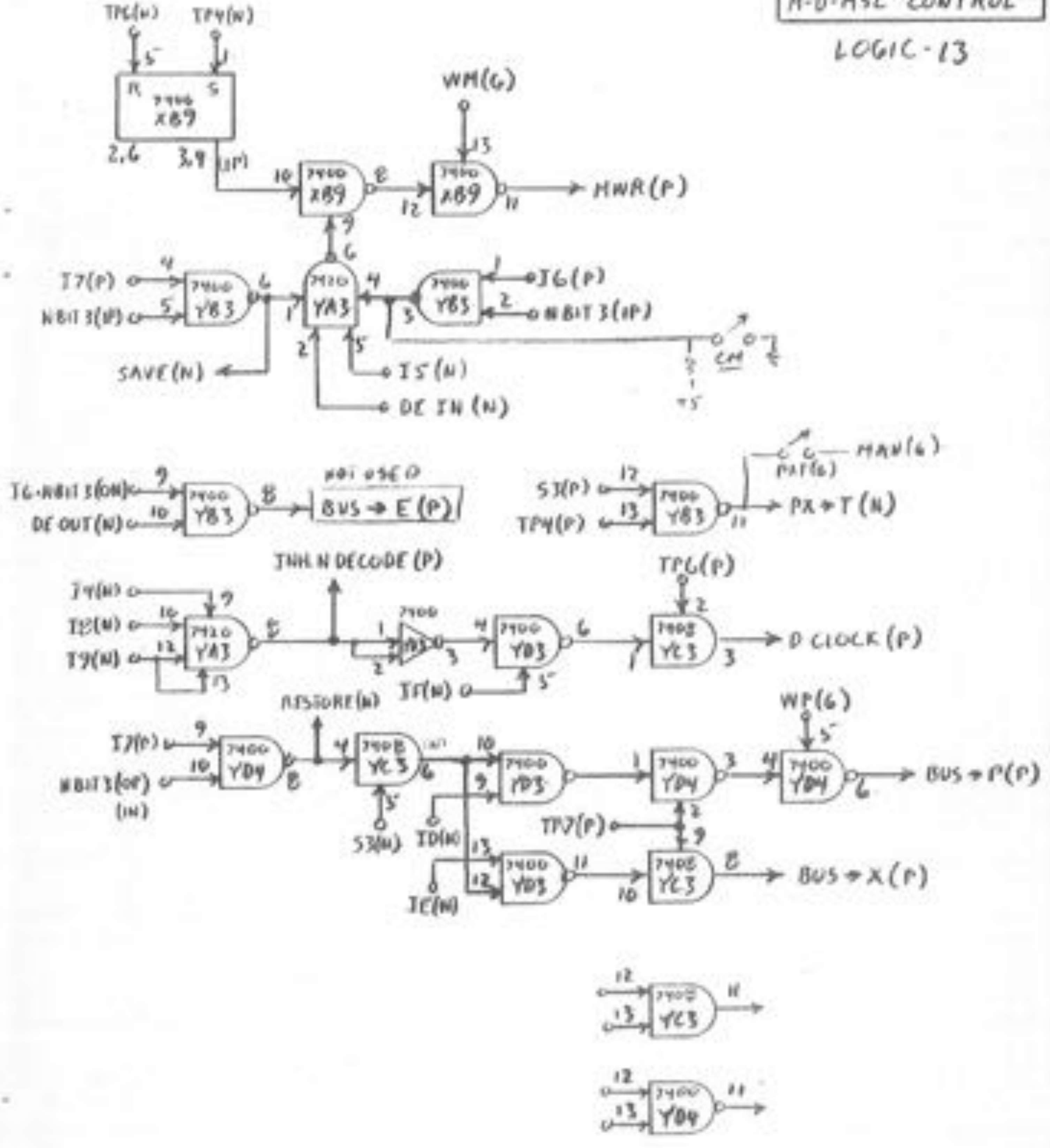


12/7/71

JAW

M-D-MSC CONTROL

LOGIC-13



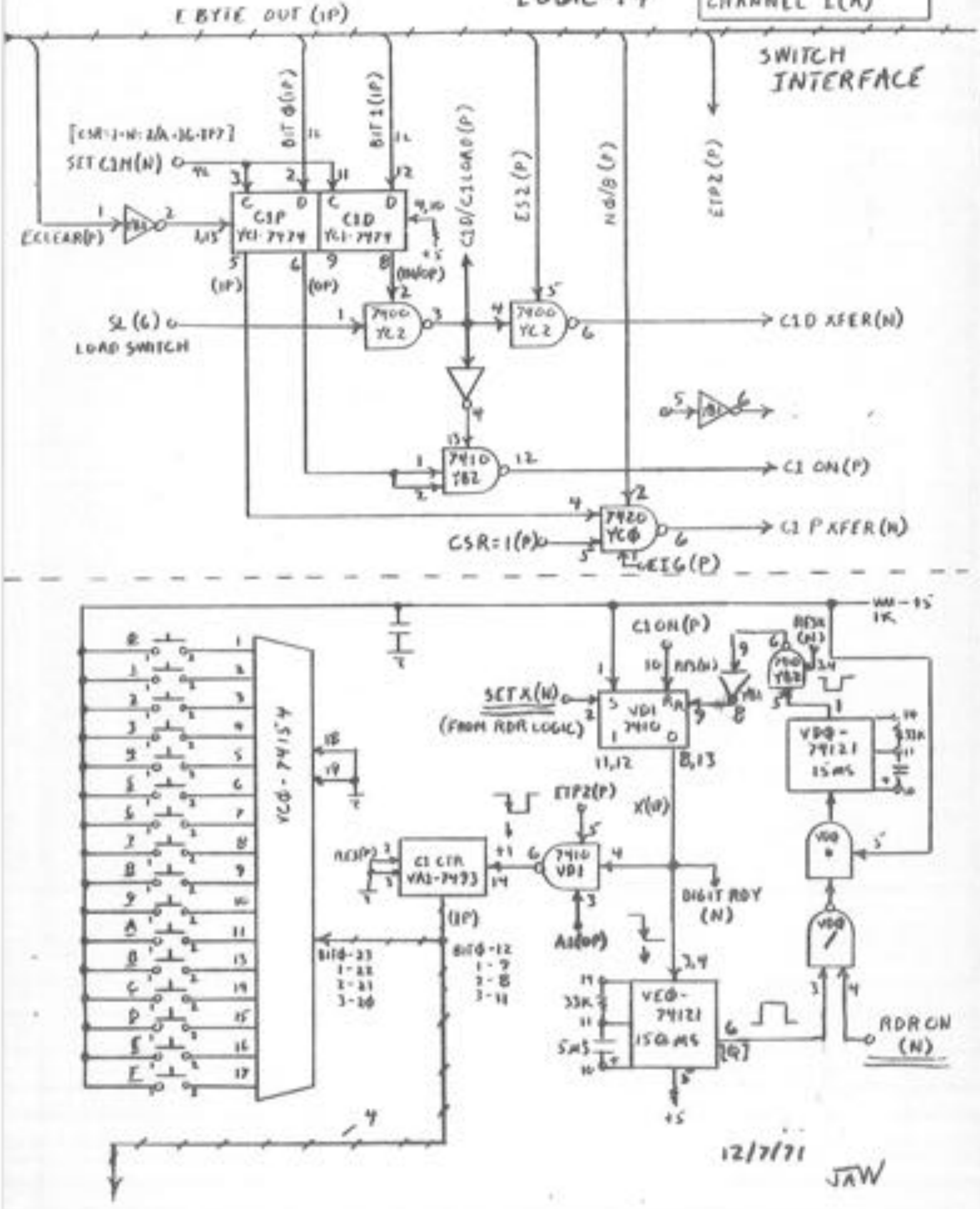
12/7/71

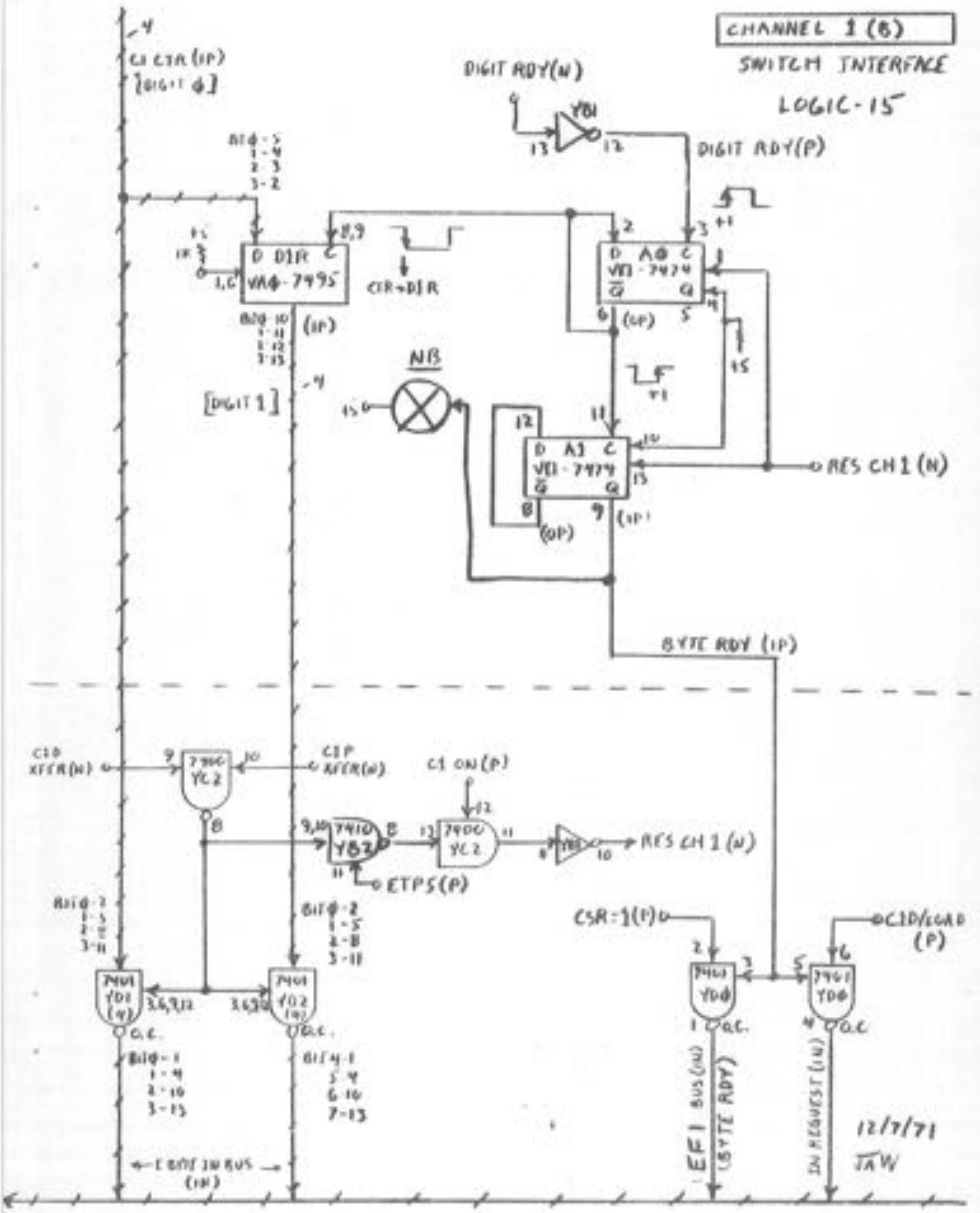
JAW

LOGIC-14

CHANNEL 1(A)

SWITCH INTERFACE



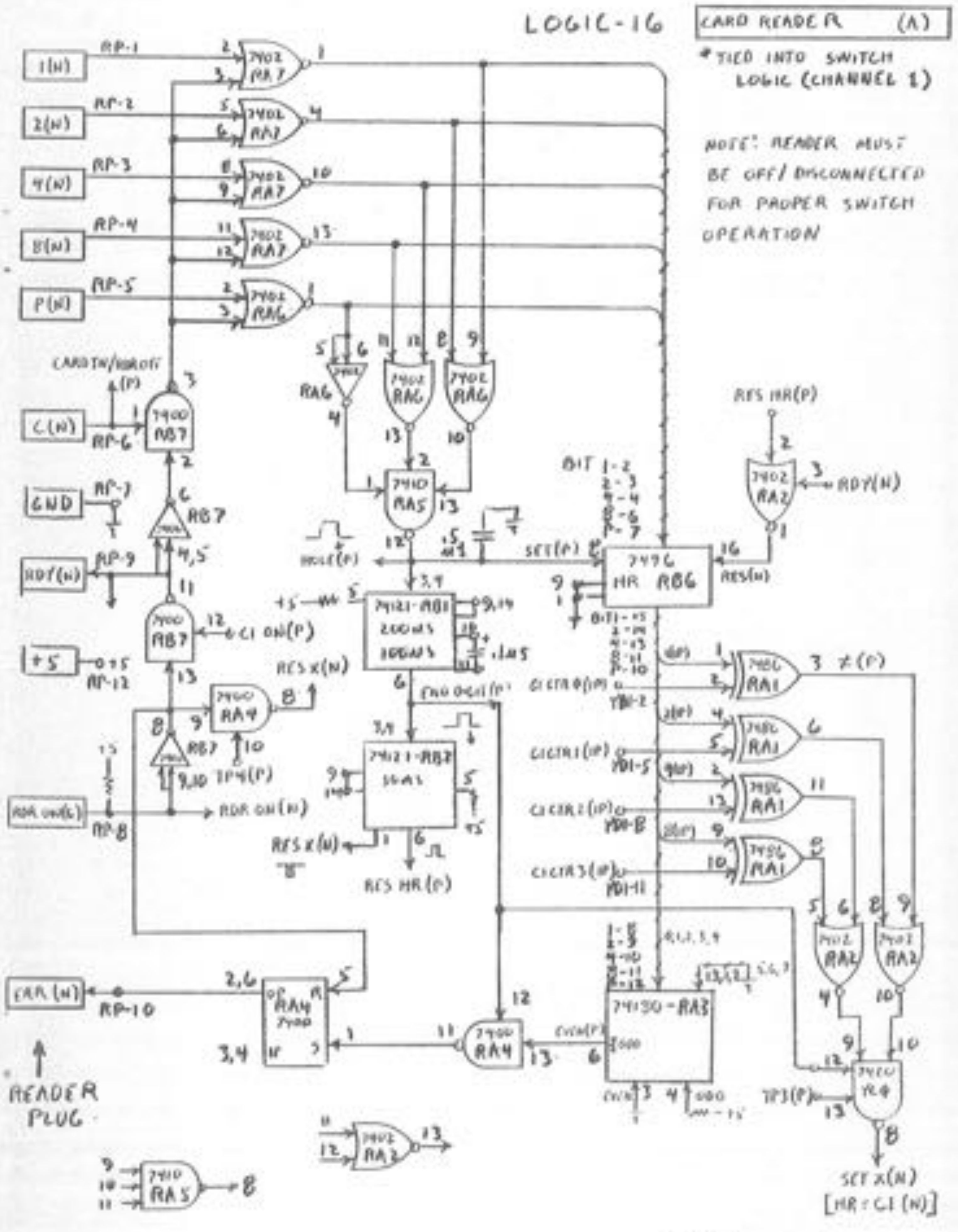

CHANNEL 1 (B)
SWITCH INTERFACE
LOGIC-15
DIGIT RDY(N)
YB1
13
12
DIGIT RDY(P)
C1 CTR (IP)
[DIGIT Φ]
BIΦ-5
1-4
2-3
3-2
D DIR C
VAΦ-7495
8,9
CTR→DIR
1,6
1K
+5
BIΦ-10
1-11
2-12
3-13
(IP)
[DIGIT 1]
D AΦ C
VE1-7474
Q̄ Q
2
3
+1
1
4
6
(OP)
5
+5
NB
+5
12
11
D A1 C
VE1-7474
Q̄ Q
10
13
8
(OP)
9
(IP)
RES CH 1 (N)
BYTE RDY (IP)
C1D XFER(N)
9
7400
YC2
10
C1P XFER(N)
8
C1 ON (P)
12
9,10
7410
YB2
8
13
7400
YC2
11
YB1
10
RES CH 1 (N)
ETPS(P)
BIΦ-2
1-5
2-8
3-11
BIΦ-2
1-5
2-8
3-11
7401
YD1
(4)
3,6,9,12
O.C.
7401
YD2
(4)
3,6,9,12
O.C.
BIΦ-1
1-4
2-10
3-13
BIΦ-1
5-4
6-10
7-13
← 8 BIT IN BUS → (IN)
CSR=1(P)
2
3
7401
YDΦ
1
O.C.
5
6
C1D/LOAD (P)
7401
YDΦ
4
O.C.
EF1 BUS (IN)
(BYTE RDY)
IN REQUEST (IN)
12/7/71
JAW

LOGIC-16

CARD READER (A)

* TIED INTO SWITCH LOGIC (CHANNEL 1)

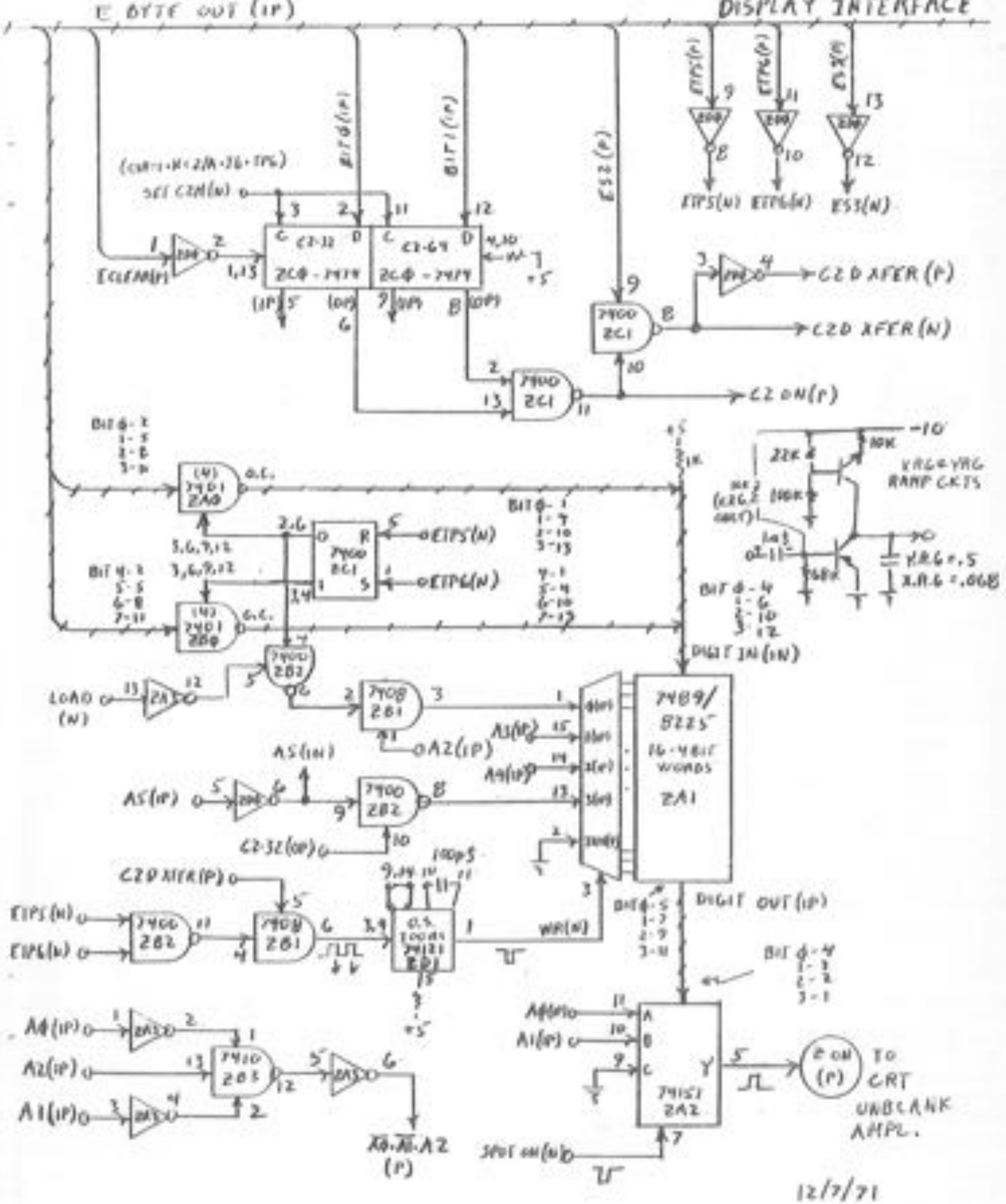
NOTE: READER MUST BE OFF/DISCONNECTED FOR PROPER SWITCH OPERATION



12/7/71 JAW

LOGIC-17

CHANNEL 2A - CRT DISPLAY INTERFACE

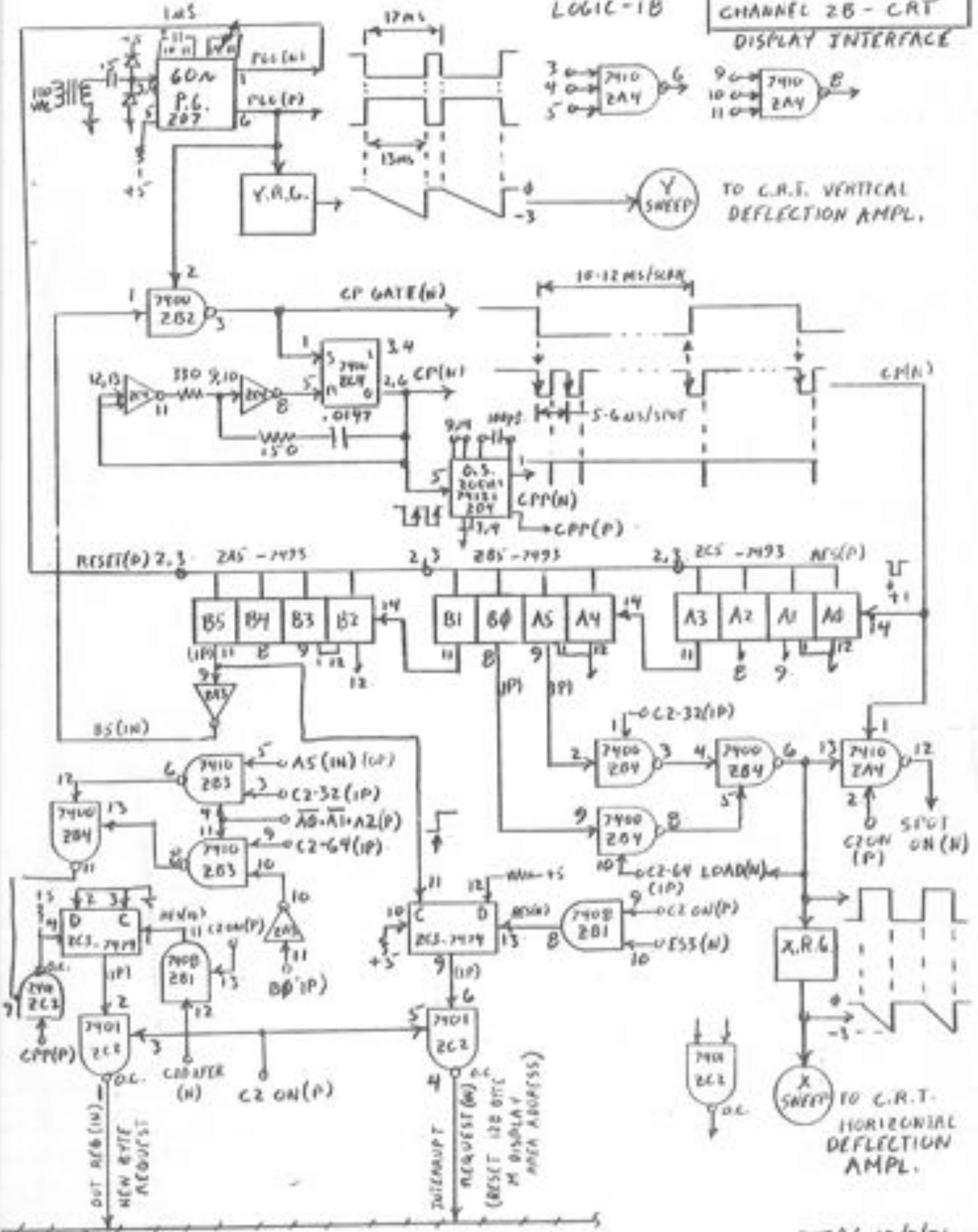


12/7/71

JAW

LOGIC-1B

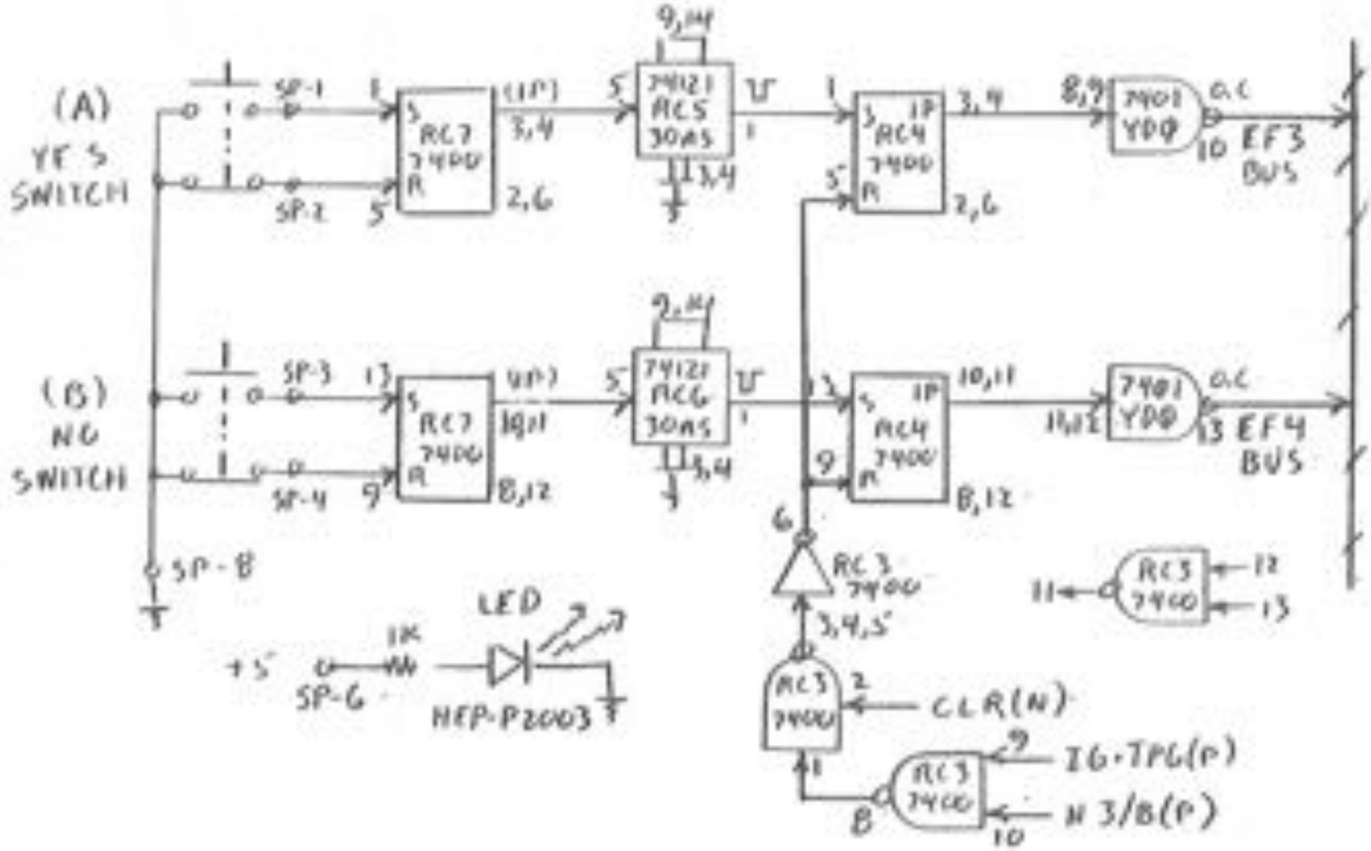
CHANNEL 2B - CRT DISPLAY INTERFACE



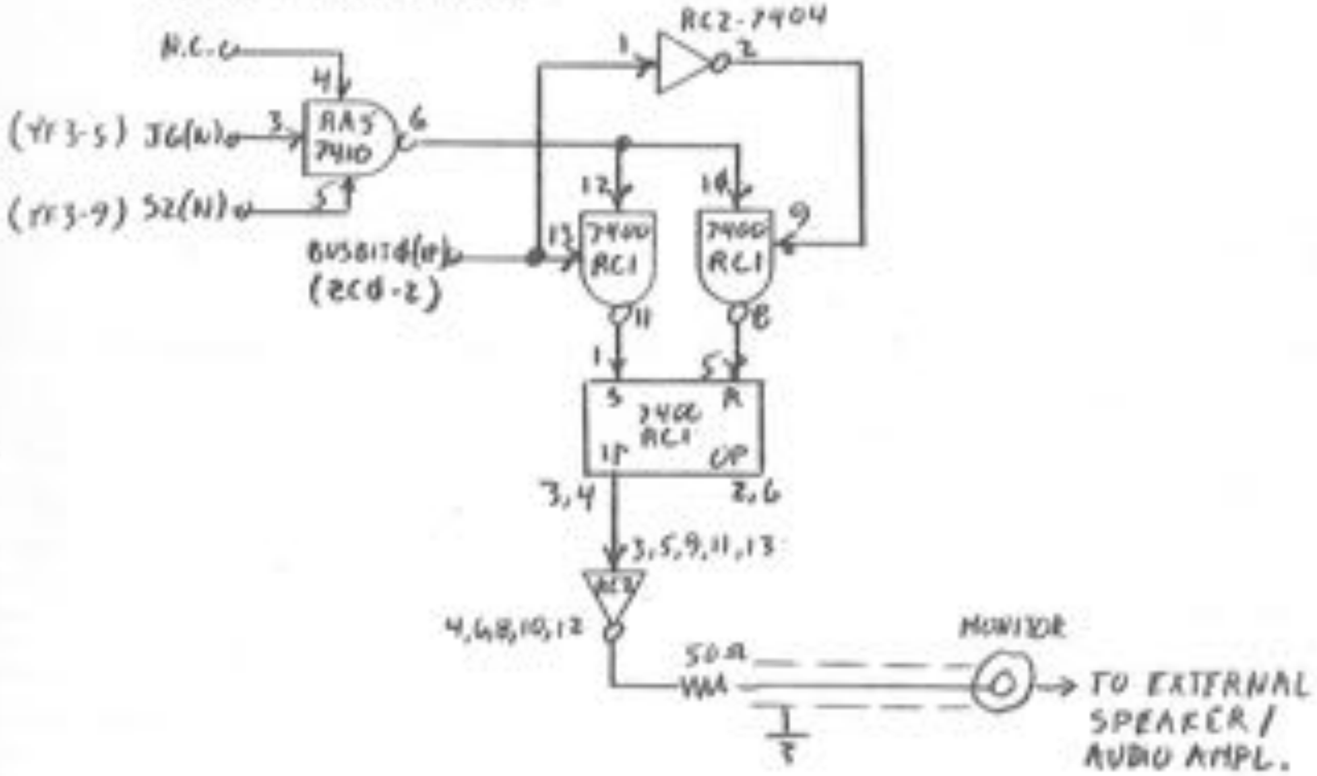
JAW - 12/7/71

LOGIC-19 MSC. I-O

YES-NO SWITCH BOX



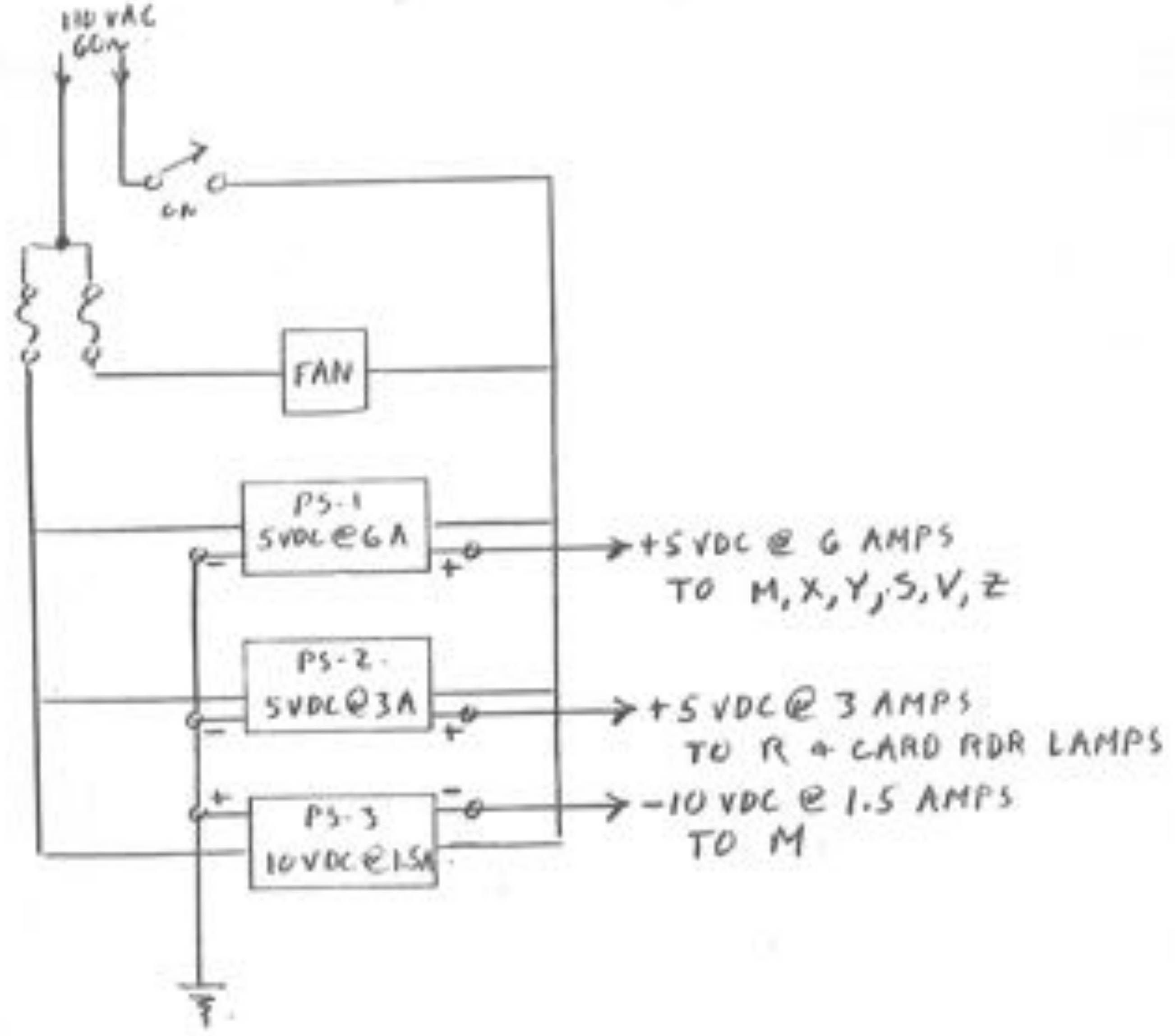
AUDIO MONITOR



JAW 12/7/71

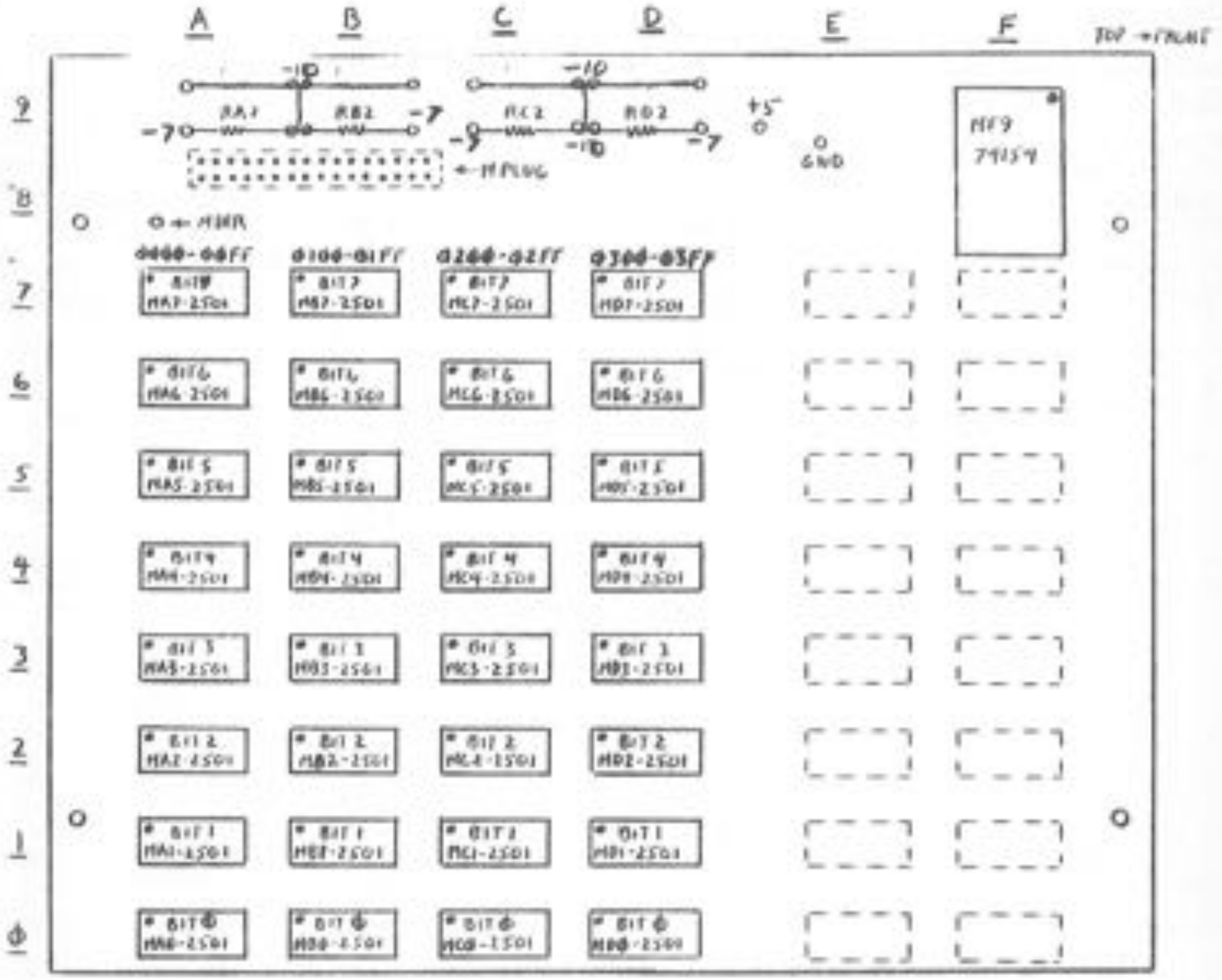
POWER

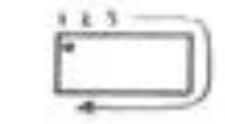
FIGURE 8



12/7/71
JAW.

M-CARD





M CARD (WIRING SIDE)

1024 BYTE M (+16 CHIP EXPANSION)

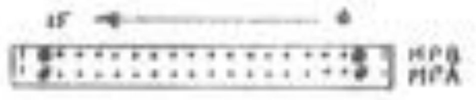
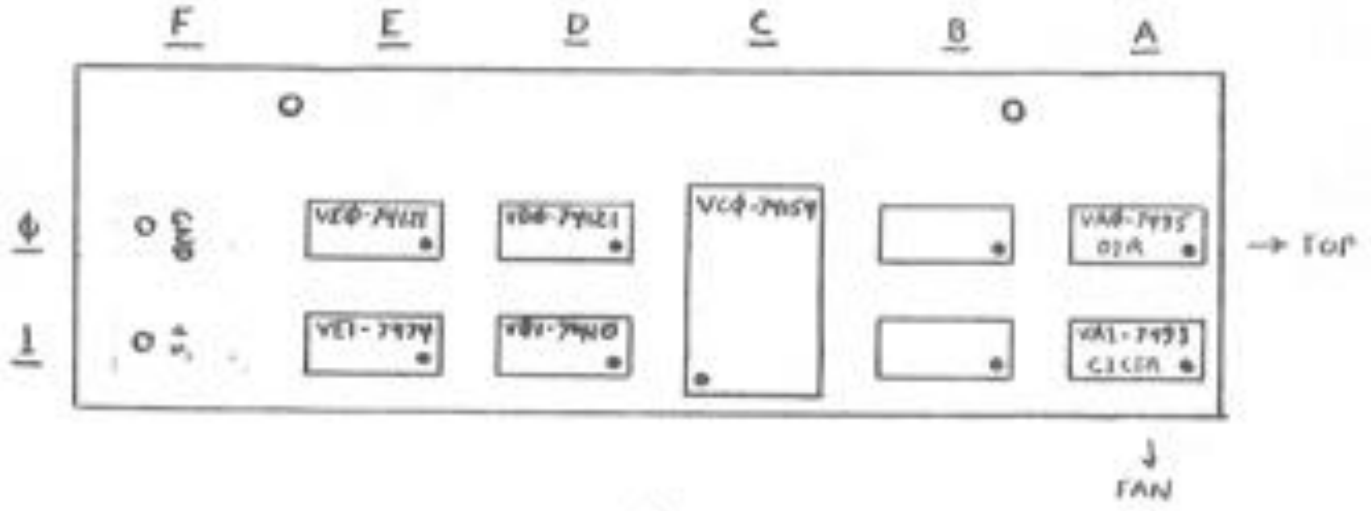


FIGURE 9

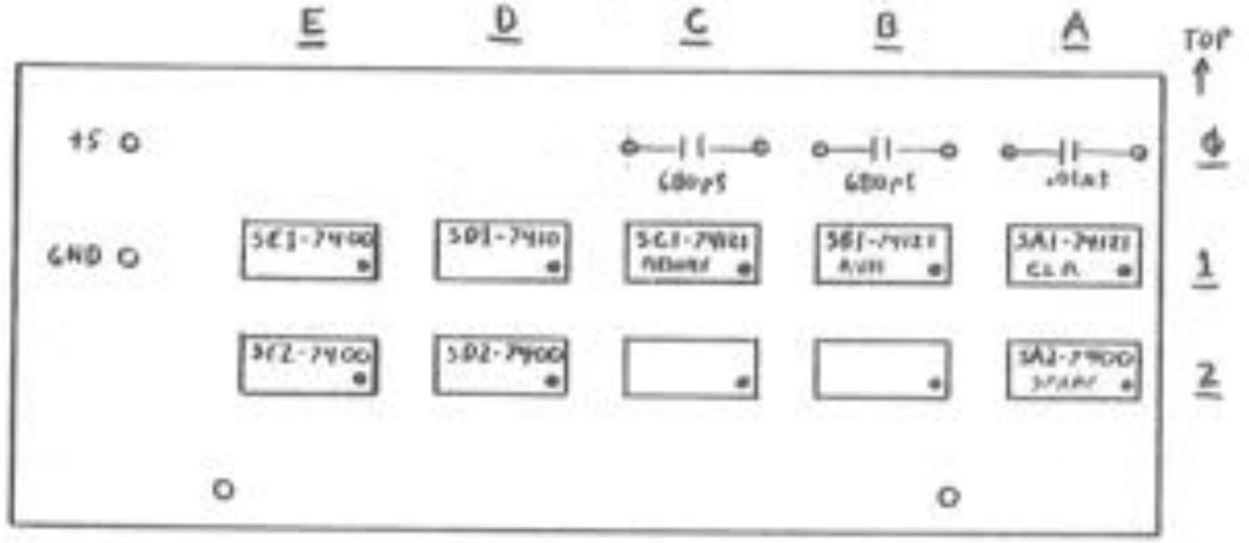
JAW 12/7/71

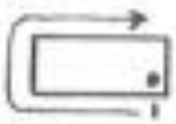
FIGURE 1Ø

S-V CARDS



CARD V (WIRING SIDE)
(PARTIAL HEX SWITCH INTERFACE)

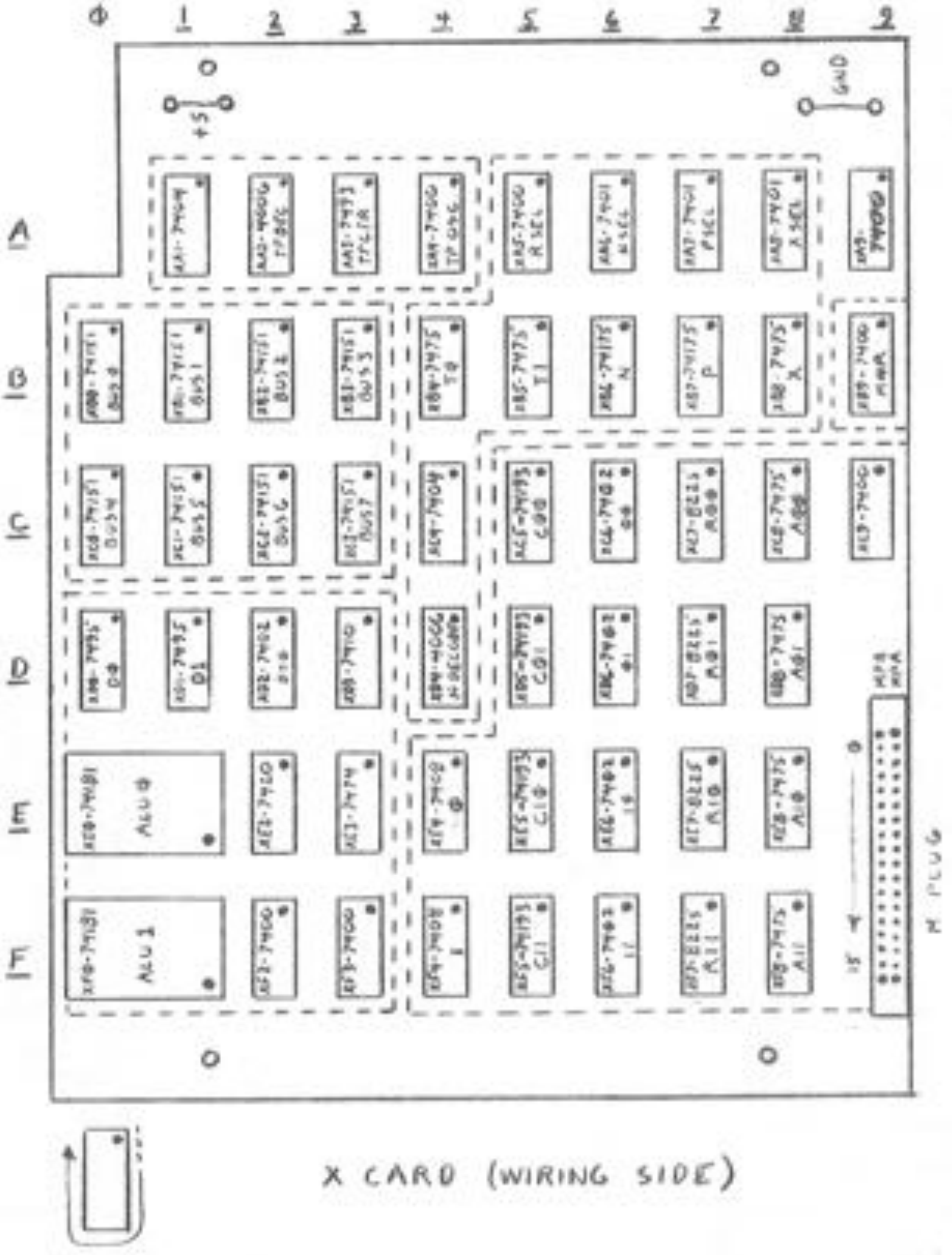




CARD S (WIRING SIDE)
(PARTIAL CONTROL SWITCH INTERFACE)

12/7/71
JAW

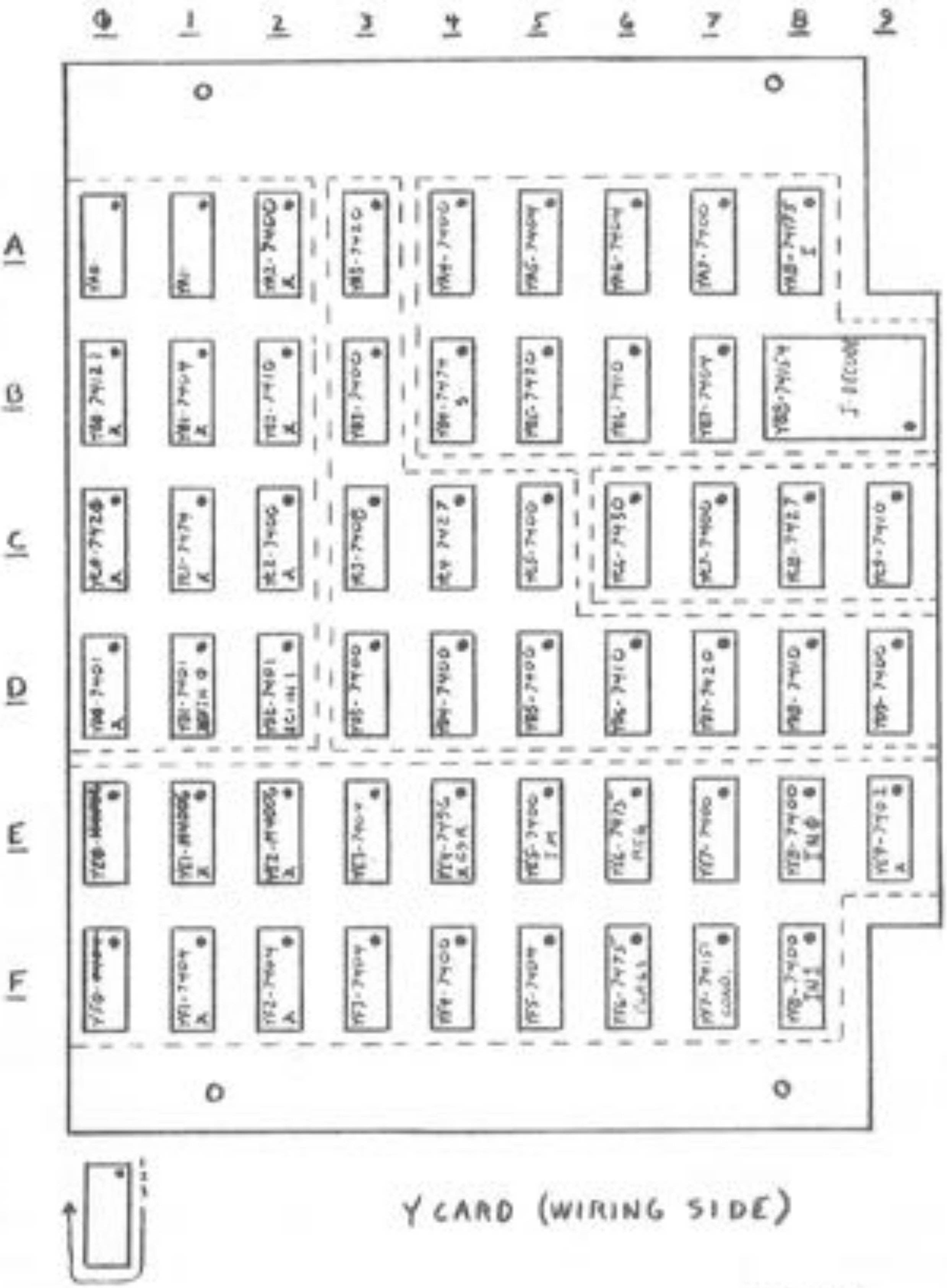
FIGURE 11 X CARD



X CARD (WIRING SIDE)

12/7/71
JAW

FIGURE 12 | Y CARD

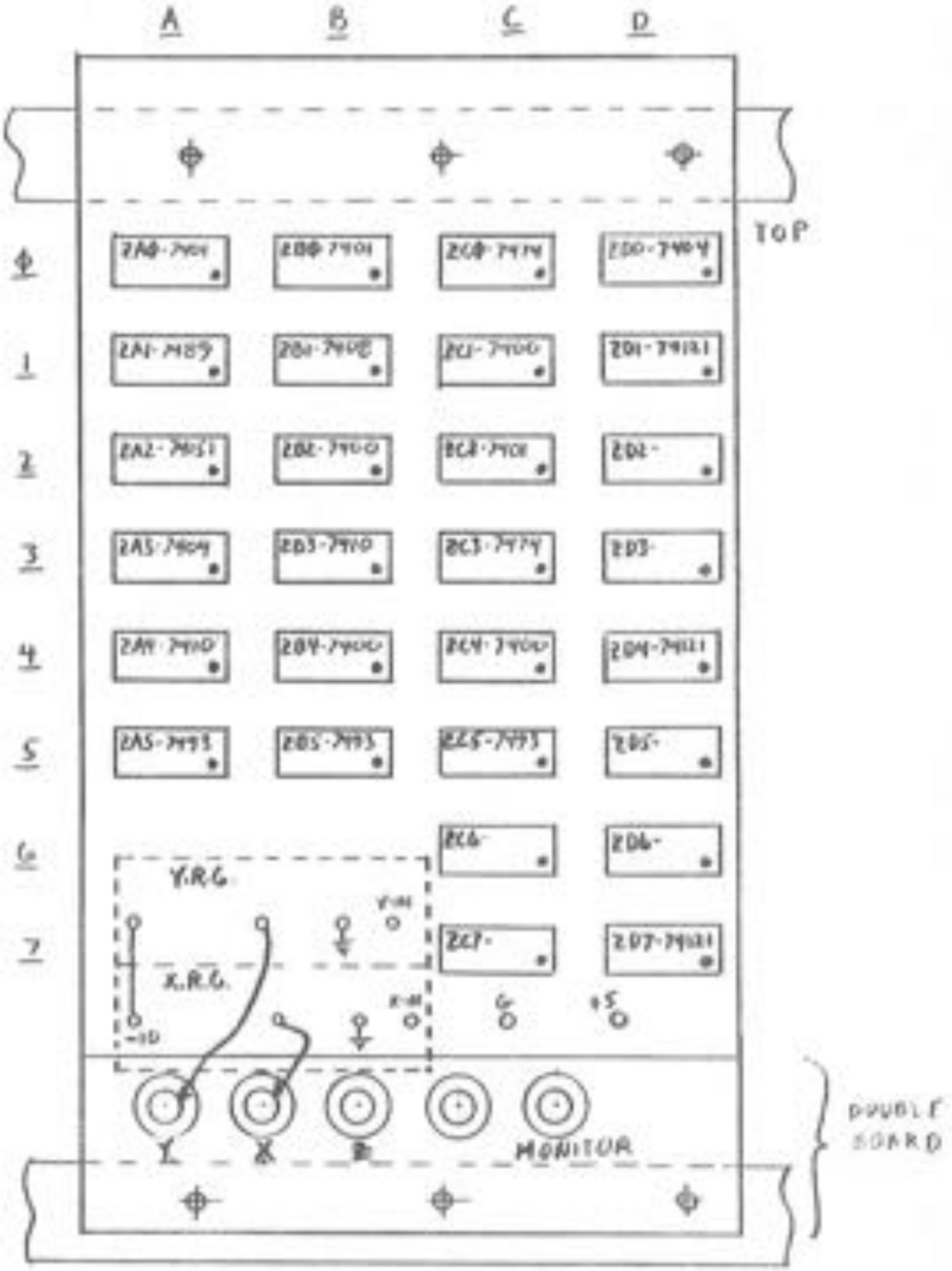


Y CARD (WIRING SIDE)

12/7/71
JAW

Z CARD

FIGURE 13



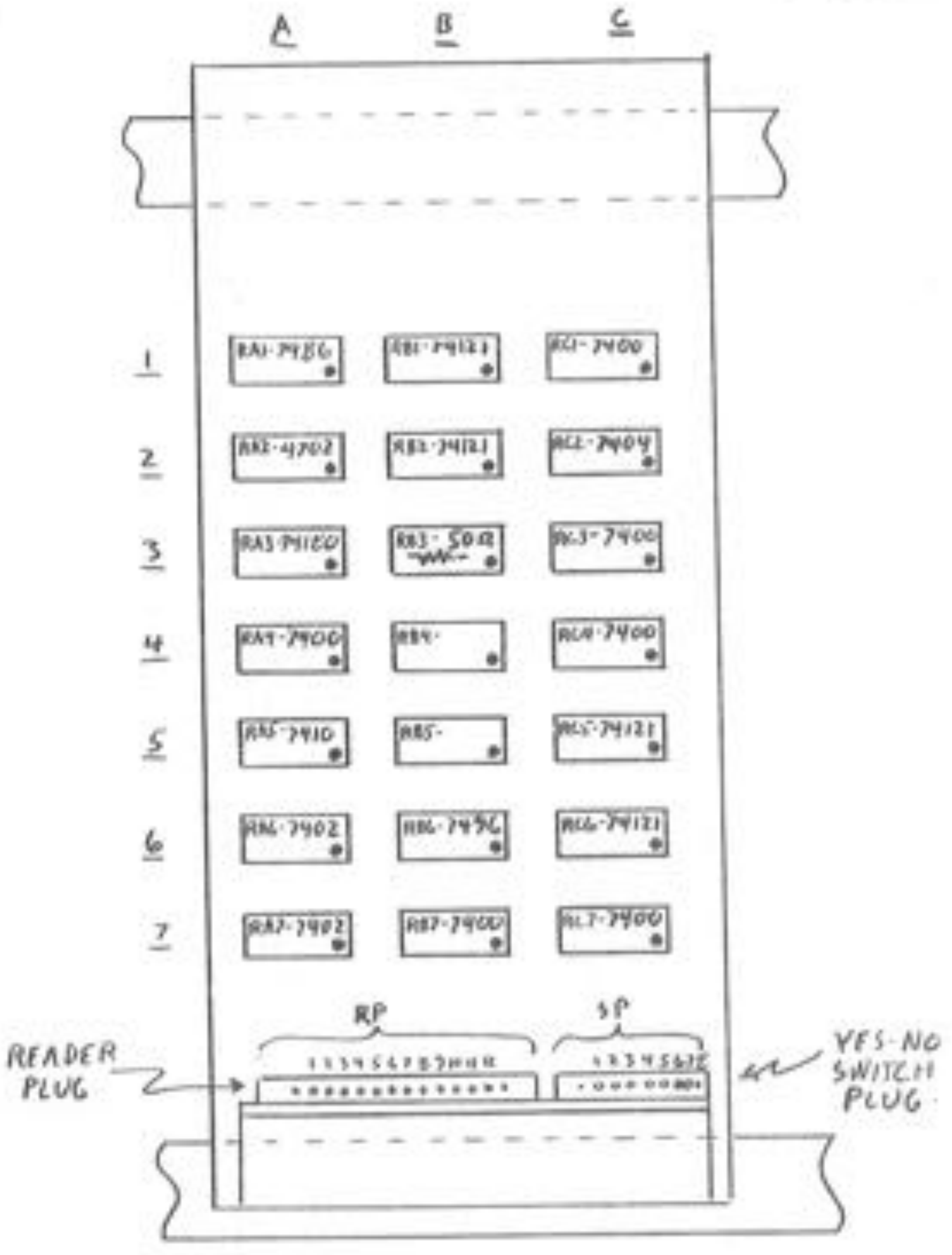
CARD Z (WIRING SIDE)

CRT DISPLAY INTERFACE

JAW

R CARD

FIGURE 14.



CARD R (WIRING SIDE)

CARD READER INTERFACE

JAW

I1, I2, ID, IE TIMING

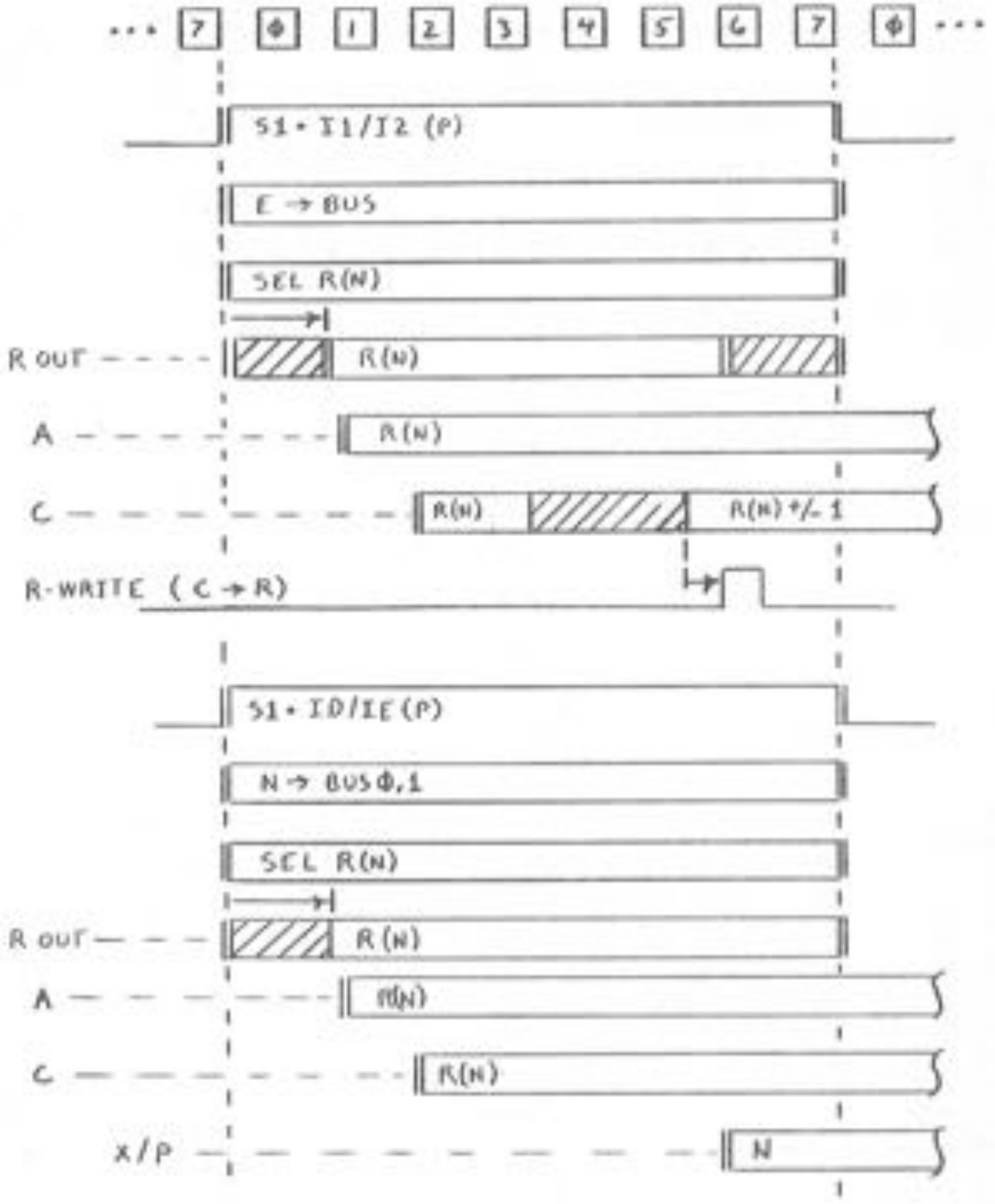


FIGURE 15

JAW

I3 TIMING

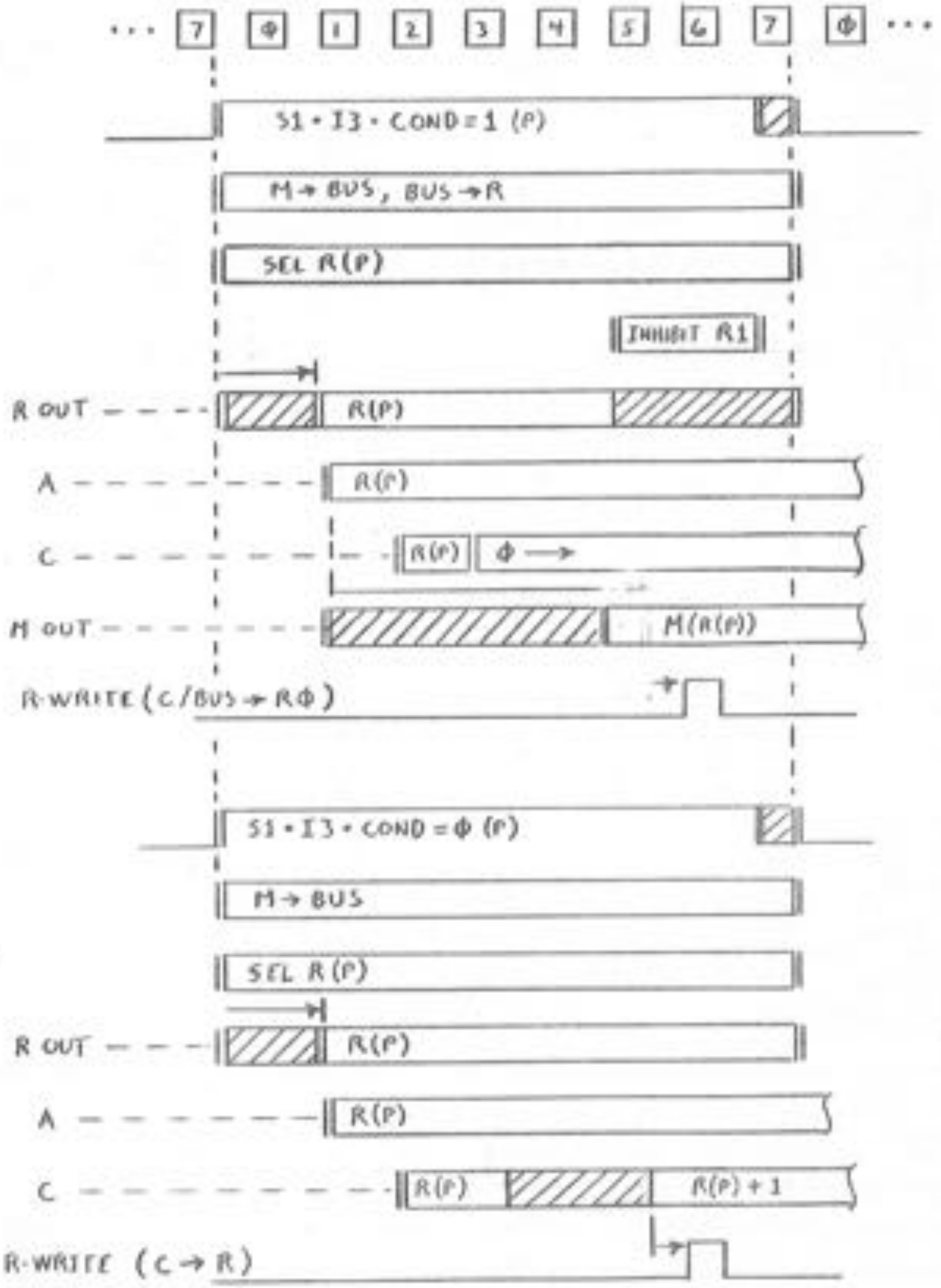


FIGURE 16

JAW

CHANNEL 2 (CRT) TIMING A

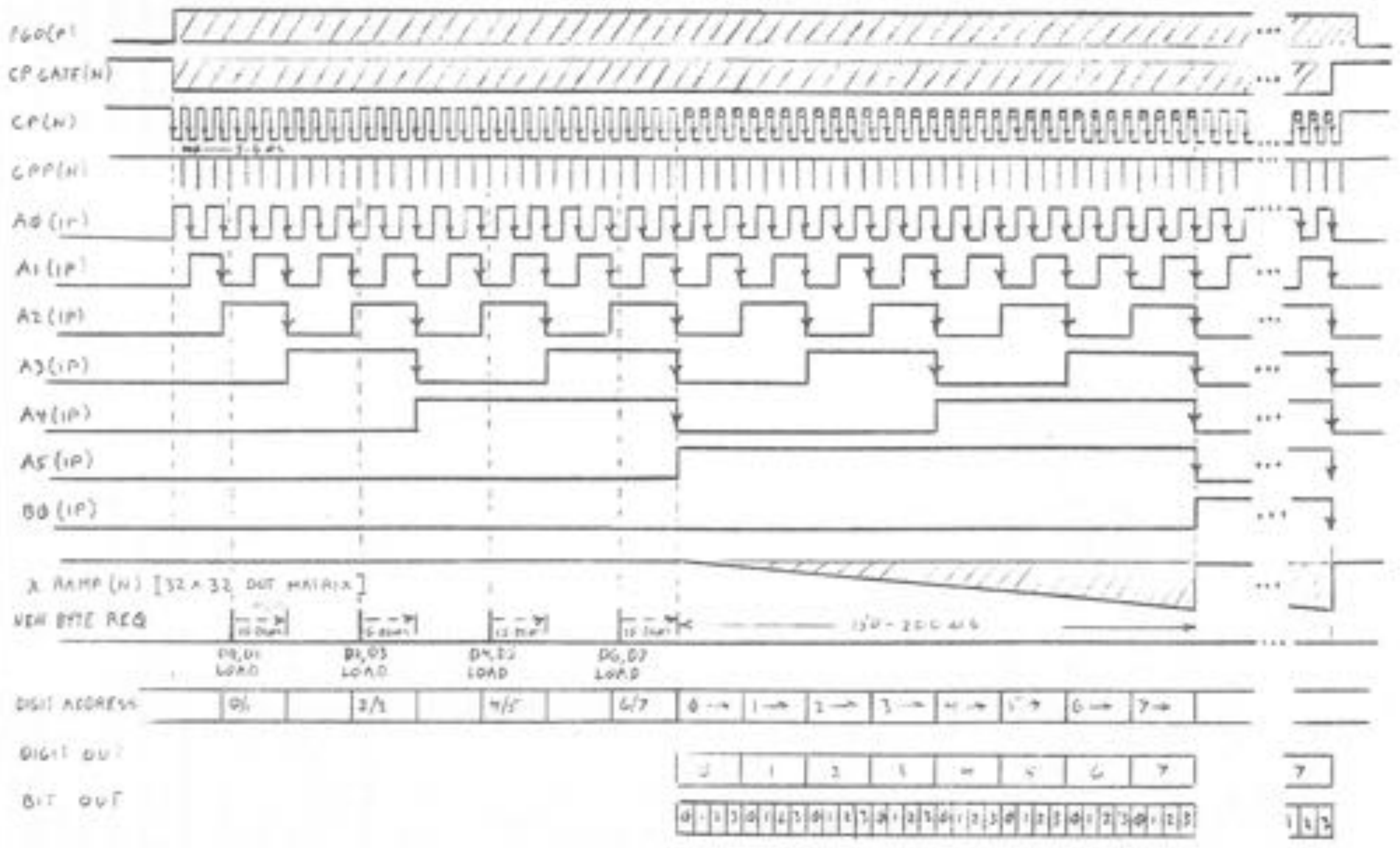


FIGURE 17

JAW

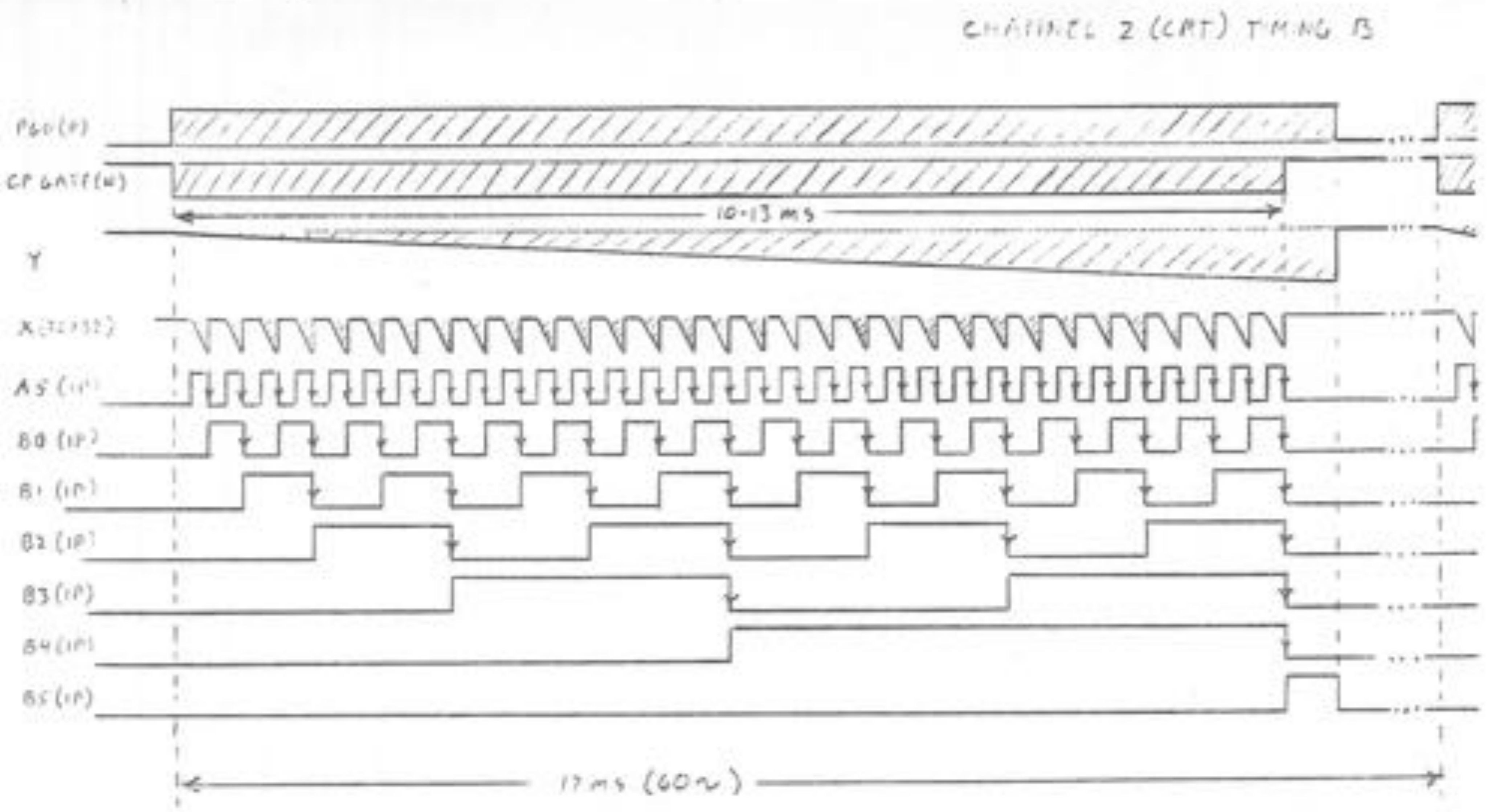
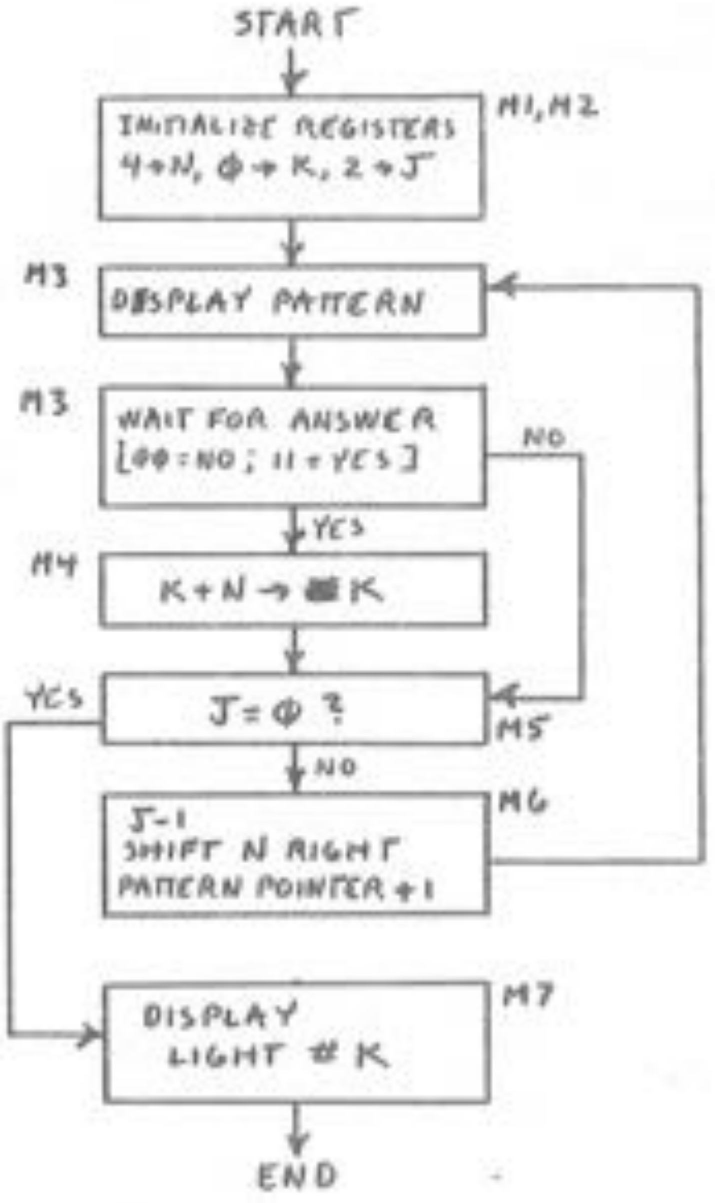

CHANNEL 2 (CRT) TIMING B
P60 (P)
CP GATE (N)
10-13 MS
Y
X (32X32)
A5 (IP)
B0 (IP)
B1 (IP)
B2 (IP)
B3 (IP)
B4 (IP)
B5 (IP)
17 MS (60~)

FIGURE 18

SA-SAMPLE PROGRAM A - DEDUCE



R(Φ) = INPUT ANSWER BYTE ADDRESS
R(1) = PATTERN POINTER
M(R(2)) = K
RΦ(3) = J
RΦ(4) = N
R(5) = PROGRAM COUNTER

FIGURE 19

JAW 12/7/71

NO. 5A PROGRAM DEDUCE DATE 12/7/71 PAGE 1/2

	S	P	X	M	(m)	INSTRUCTION	comments
card-1	IA	—	—	ΦΦΦΦ	ΦΦ	NOT EXECUTED	INPUT ANSWER BYTE STORAGE
	M1	Φ	—	ΦΦΦ1	9Φ	R1(Φ)→D	"ΦΦ"→D
		Φ	—	2	B1	D→R1(1)	
		Φ	—	3	B2	D→R1(2)	
		Φ	—	4	B5	D→R1(5)	
		Φ	—	5	4Φ	M(R(Φ))→D, R(Φ)+1	
		Φ	—	6	Φ2	"Φ2"	
		Φ	—	7	A3	D→RΦ(3)	2→J
		Φ	—	8	4Φ	M(R(Φ))→D, R(Φ)+1	
		Φ	—	9	18	"18"	
		Φ	—	A	A5	D→RΦ(5)	M2→R(5)
		Φ	—	B	D5	5→P	GO TO M2
card-2		—	—	ΦΦΦC	ΦΦ	——	K
	MA	—	—	ΦΦΦD	FΦ	"1111 ΦΦΦΦ"	
		—	—	F	CC	"11ΦΦ 11 ΦΦ"	PATTERN TABLE
		—	—	F	AA	"1Φ1Φ1Φ1Φ"	
	MT	—	—	ΦΦ1Φ	Φ1		
		—	—	1	Φ2		
		—	—	2	Φ4		BINARY TO
		—	—	3	ΦB		DECIMAL
		—	—	4	1Φ		CONVERSION
		—	—	5	2Φ		TABLE
		—	—	6	4Φ		
		—	—	7	8Φ		
card-3	M2	5	—	ΦΦ18	45	M(R(5))→D, R(5)+1	
		5	—	9	Φ4	"Φ4"	
		5	—	A	A4	D→RΦ(4)	4→N
		5	—	B	45	M(R(5))→D, R(5)+1	
		5	—	C	ΦD	"ΦD"	
		5	—	D	A1	D→RΦ(1)	MA→PATTERN POINTER
		5	—	E	95	R1(5)→D	
		5	—	F	AΦ	D→RΦ(Φ)	IA→R(Φ)
		5	—	ΦΦ2Φ	45	M(R(5))→D, R(5)+1	
		5	—	1	ΦC	"ΦC"	
		5	—	2	A2	D→RΦ(2)	K ADDRESS→R(2)
		5	—	3	E5	5→X	
card-4		5	5	4	61	SELECT HEX SWITCHES	
		5	5	5	Φ1	"Φ1"	
		5	5	6	62	SET DIRECT MODE	
		5	5	7	Φ2	"Φ2"	
	M3	5	—	ΦΦ28	Φ1	IDLE	M(R(1))→LIGHTS
		5	—	9	2Φ	R(Φ)-1	RESET R(Φ) TO IA AFTER INPUT BYTE
		5	—	A	4Φ	M(R(Φ))→D, R(Φ)+1	
		5	—	B	2Φ	R(Φ)-1	
		5	—	C	32	D=Φ?	
		5	—	D	32	"32" YES	→CHOSEN LIGHT OFF; SKIP TO M5
	M4	5	—	ΦΦ2E	E2	2→X	
		5	2	F	B4	RΦ(4)→D	N→D

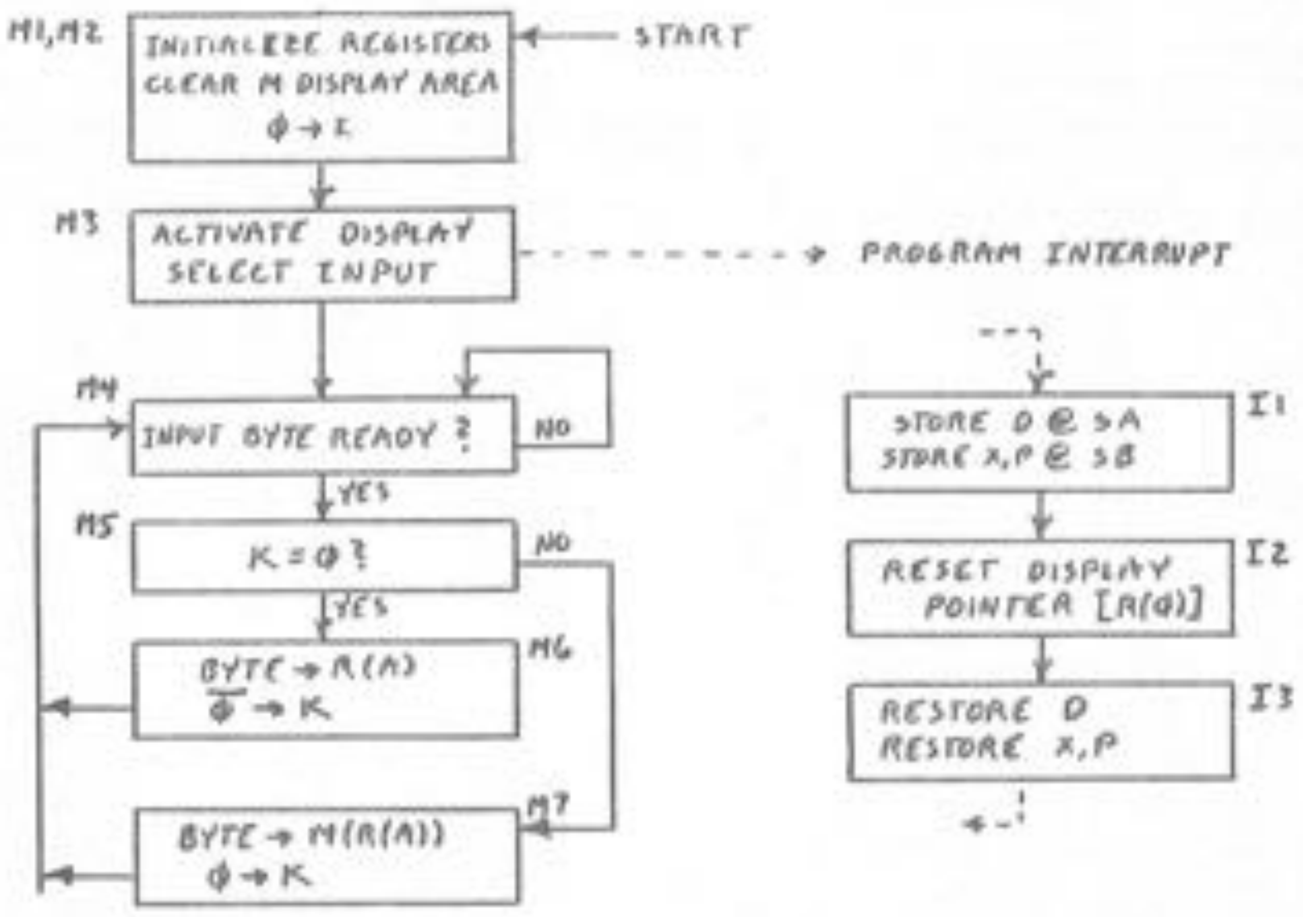
JAW

NO. 5A PROGRAM DEDUCE DATE 12/7/71 PAGE 2/2

	S	P	X	M	(m)	INSTRUCTION	comments
card-5		5	2	ΦΦ1Φ	F4	M(R(X))+D→D	K+N
		5	2	1	52	D→M(R(X))	K+N→K
	M5	5	–	ΦΦ32	83	RΦ(3)→D	
		5	–	3	32	D=Φ?	
		5	–	4	3C	"3C" YES	→ J=Φ → SKIP TO M7
	M6	5	–	ΦΦ35	23	R(3)-1	J-1
		5	–	6	11	R(1)+1	PATTERN POINTER +1.
		5	–	7	84	RΦ(4)→D	
		5	–	8	F6	SHIFT RIGHT	
		5	–	9	A4	D→RΦ(4)	SHIFT N
		5	–	A	3Φ	BRANCH	
		5	–	B	28	"28"	RETURN TO M3
card-6	M7	5	–	ΦΦ3C	11	R(1)+1	MT→R1
		5	–	D	42	M(R(2))→D, R(2)+1	
		5	–	E	C1	DΦ→RΦΦ(1)	K→R1
		5	–	F	22	R(2)-1	
		5	–	ΦΦ4Φ	95	R1(5)→D	
		5	–	1	52	D→M(R(2))	RESET K TO ΦΦ
		5	–	2	E5	5→X	
		5	5	3	62	TURN SWITCHES OFF	
		5	5	4	ΦΦ	Φ Φ	
	END	5	5	ΦΦ45	Φ1	IDLE	CHOSEN LIGHT ON
card							
card							

JAW

SB - SAMPLE PROGRAM B - DISPLAY



M(ϕ1ϕϕ) - ϕ17F) = M DISPLAY AREA (128 BYTES)
R(ϕ) = DISPLAY POINTER
R(1) = INTERRUPT PROGRAM COUNTER
R(2) = INTERRUPT STORAGE POINTER
R(3) = MAIN PROGRAM COUNTER
M(R(4)) = INPUT BYTE STORAGE (TEMPORARY)
Rϕ(5) = K
R(A) = DISPLAY MODIFICATION POINTER

FIGURE 20

JAN 12/7/71

NO. 5B PROGRAM DISPLAY DATE 12/7/71 PAGE 1/2

	S	P	X	M	(m)	INSTRUCTION	comments
card-1	BS	–	–	ΦΦΦΦ	ΦΦ	——	INPUT BYTE STORAGE
	M1	Φ	–	ΦΦΦ1	9Φ	R1(Φ) → D	
		Φ	–	2	B1	D → R1(1)	
		Φ	–	3	B2	D → R1(2)	
		Φ	–	4	B3	D → R1(3)	
		Φ	–	5	B4	D → R1(4)	
		Φ	–	6	A4	D → RΦ(4)	BS → R(4)
		Φ	–	7	A5	D → RΦ(5)	Φ → K
		Φ	–	8	4Φ	M(R(Φ)) → D, R(Φ)+1	
		Φ	–	9	1C	"M2"	
		Φ	–	A	A3	D → RΦ(3)	M2 → R(3)
		Φ	–	B	D3	3 → P	GO TO M2
card-2	SB	–	–	ΦΦΦC	ΦΦ	——	X,P SAVE LOCATION
	SA	–	–	ΦΦΦD	ΦΦ	——	D SAVE LOCATION
	I3	1	2	ΦΦΦE	FΦ	M(R(X)) → D	SA → D
		1	2	F	22	R(2) – 1	
		1	2	ΦΦ1Φ	7Φ	M(R(X)) → XP, R(X)+1	SB → X,P (RETURN), RESET IM
	I1	1	2	ΦΦ11	52	D → M(R(2))	SAVE D @ SA
		1	2	2	22	R(2) – 1	
		1	2	3	78	T → M(R(X))	SAVE X,P @ SB
	I2	1	2	ΦΦ14	91	R1(1) → D	
		1	2	5	AΦ	D → RΦ(Φ)	
		1	2	6	41	M(R(1)) → D, R(1)+1	
		1	2	7	Φ1	"Φ1"	
card-3		1	2	8	BΦ	D → R1(Φ)	Φ1ΦΦ → R1
		1	2	9	12	R(2)+1	
		1	2	A	3Φ	BRANCH	
		1	2	B	ΦE	"I3"	GO TO I3
	M2	3	–	ΦΦ1C	43	M(R(3)) → D, R(3)+1	
		3	–	D	11	"I1"	
		3	–	E	A1	D → RΦ(1)	I1 → R(1)
		3	–	F	43	M(R(3)) → D, R(3)+1	
		3	–	ΦΦ2Φ	ΦD	"SA"	
		3	–	1	A2	D → RΦ(2)	SA → R(2)
		3	–	2	43	M(R(3)) → D, R(3)+1	
		3	–	3	Φ1	"Φ1"	
card-4		3	–	4	BA	D → R1(A)	
		3	–	5	43	M(R(3)) → D, R(3)+1	
		3	–	6	7F	"7F"	
		3	–	7	AA	D → RΦ(A)	Φ17F → R(A)
	C1	3	–	ΦΦ28	93	R1(3) → D	
		3	–	9	5A	D → M(R(A))	ΦΦ → M(R(A))
		3	–	A	8A	RΦ(A) → D	
		3	–	B	32	D = Φ ?	
		3	–	C	3Φ	"M3" YES	→ SKIP TO M3 (END CLEAR)
		3	–	D	2A	R(A) – 1	
		3	–	E	3Φ	BRANCH	
		3	–	F	28	"C1" ——	→ REPEAT FROM C1

JAW

NO. 5B   PROGRAM DISPLAY   DATE 12/7/71   PAGE 2/2

	S	P	X	M	(m)	INSTRUCTION	comments
card-5	M3	3	–	ΦΦ3Φ	E3	3 → X	
		3	3	1	61	SELECT DISPLAY	
		3	3	2	Φ2	"Φ2"	
		3	3	3	62	SET DISPLAY = 32×32	
		3	3	4	Φ1	"Φ1"	
		3	3	5	7Φ	3→P, 3→X, R(X)+1	RESET IM
		3	3	6	33	"33"	
		3	3	7	61	SELECT INPUT	
		3	3	8	Φ1	"Φ1"	
		3	3	9	62	SET IN = PROGRAM MODE	
		3	3	A	Φ1	"Φ1"	
	M4	3	–	ΦΦ3B	34	EF1=1?	TEST FOR INPUT BYTE
card-6		3	–	C	3F	"M5"	→ BYTE READY → GO TO M5
		3	–	D	3Φ	BRANCH	
		3	–	E	3B	"M4"	→ REPEAT FROM M4
	M5	3	–	ΦΦ3F	E4	4 → X	
		3	4	ΦΦ4Φ	6B	BYTE → M(R(X))	INPUT BYTE → 5B
		3	4	1	85	RΦ(5) → D	
		3	4	2	32	D=Φ?	
		3	4	3	4A	"M6"	YES → K=Φ - GO TO M6
	M7	3	4	ΦΦ44	FΦ	M(R(X)) → D	
		3	4	5	5A	D → M(R(A))	BYTE → M(R(A))
		3	4	6	93	R1(3) → D	
		3	4	7	A5	D → RΦ(5)	Φ → K
card-7		3	4	8	3Φ	BRANCH	
		3	4	9	3B	"M4"	RETURN TO M4
	M6	3	4	ΦΦ4A	FΦ	M(R(X)) → D	
		3	4	B	AA	D → RΦ(A)	
		3	4	C	83	RΦ(3) → D	
		3	4	D	A5	D → RΦ(5)	$\overline{\Phi}$ → K
		3	4	E	3Φ	BRANCH	
		3	4	F	3B	"M4"	RETURN TO M4
card							

JaW