UIUCDCS-R-73-559

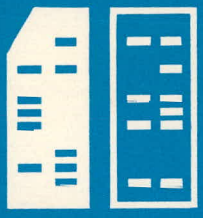
COO-1469-0217

SEMANTRIX: A Semantically Guided Digital Electronic Machine

by

Trevor Nigel Mudge

February 1973



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

DISCLAIMER

UIUCDCS-R-73-559

SEMANTRIX: A Semantically Guided Digital Electronic Machine*

by

Trevor Nigel Mudge

February 1973

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

* Supported in part by the Department of Computer Science and the Atomic Energy Commission under contract US AEC AT(11-1)1469, and submitted in partial fulfillment of the requirements of the Graduate College for the Degree of Master of Science in Computer Science.

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ACKNOWLEDGMENT

The author would like to take this opportunity to thank his advisor, Professor S. Ray, for his guidance and Professor W. J. Poppelbaum for the opportunity to work in his Research Group.

The work described in chapter 4 is largely the effort of Dennis Kodimer, as was the construction of the electromechanical system outlined in chapter 5. The author would like to thank him for a convivial collaboration.

Finally a word of thanks goes to the 2 typists who typed the report; Ms Barbara Bunting and Ms Janet van Weringh.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 An Overview of Semantrix

Semantrix is a digital machine, whose domain of activity (or world) is a rectangular plane or table top. (See Figure 1.1.1 for a diagram of the machine). Its activity in the plane can be instructed from either a teletype or a digital computer. Semantrix and a teletype can form a stand alone system. However, to make full use of the machine, a digital computer should be used as a controller. Semantrix can thus be viewed as a special piece of I/O equipment.

The machine's capabilities in its two dimensional world are:

1. The ability to compute the position of any one of up to 64 small cubes that can be placed in the rectangle.

   Each cube has a numerical label associated with it. It is thus possible to instruct the machine to compute the position of cube N($N \in \{0, \ldots, 63\}$) by just handing the number N to the machine. The reply is a 6 octit number. This represents a unique intersection on a quadruled grid which partitions the table top. The controller (human or electronic) can also use the label to create an associative memory which can be used to store information concerning a particular block (e.g. color, record of movements, etc.).

2. The ability to move any particular cube to a prescribed point on the table top. This is achieved electro-mechanically.

1.2 Applications to Cognitive Systems

An immediate application for Semantrix is to test the viability of certain cognitive maps, or world models (WM). The map would be stored

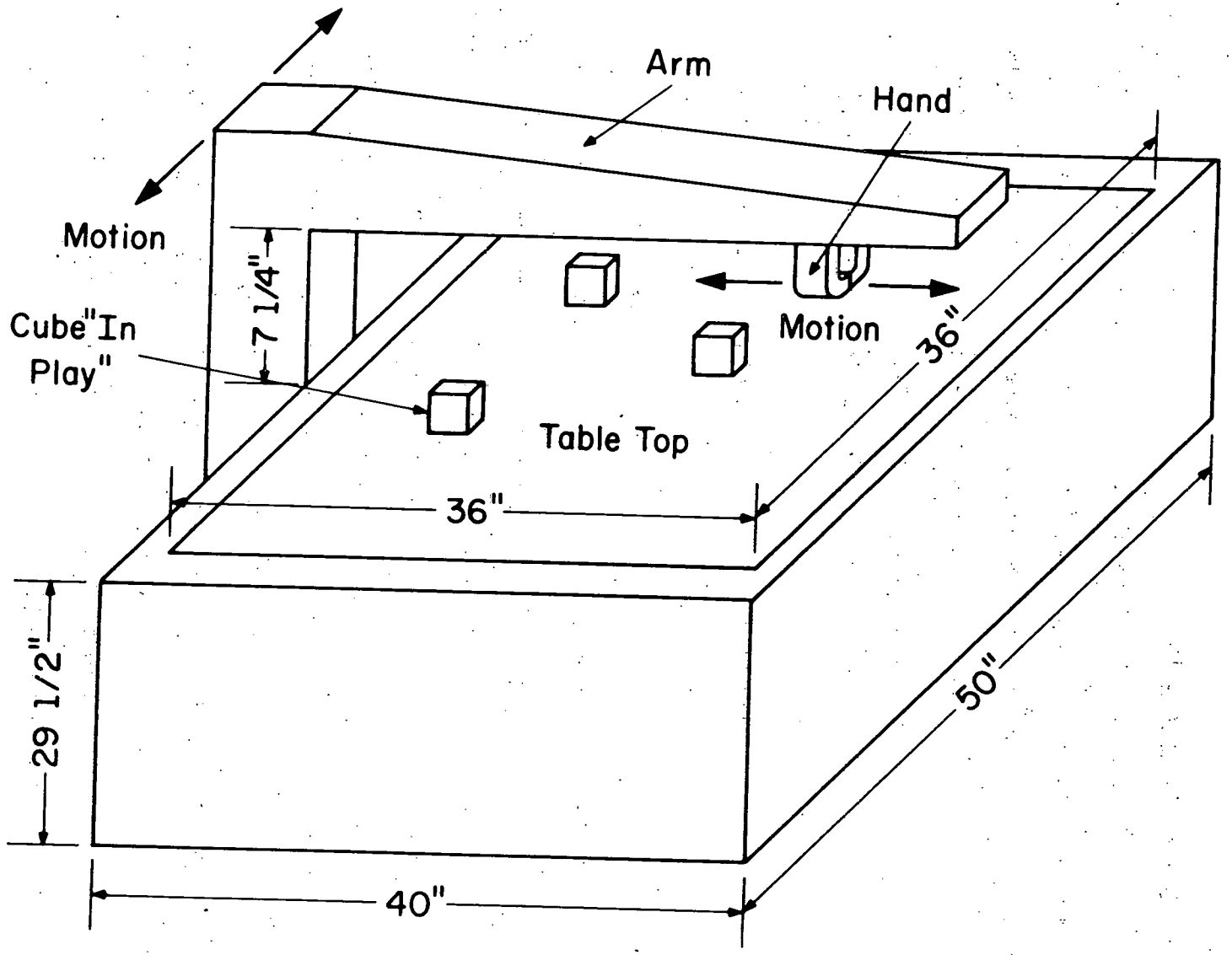


Figure 1.1.1 Diagram of Semantrix

in the digital computer which controls Semantrix. Thus by referring to the map the computer can direct Semantrix to synthesize a particular example of the general class of things which are supposedly modelled by the cognitive map.

For reasons that will be outlined in Section 1.3 an interesting problem is the synthesis of landscape pictures. To test a WM which attempts to model a landscape picture, colored cubes are used, and under the control of the computer Semantrix synthesizes a mosiac landscape picture.

Two basic attributes are associated with each cube. One is position on the table top. This can be computed by Semantrix. The second is color. This must be input to the computer prior to an attempted synthesis. It takes the form of a table of information mapping the cube number onto the colors. Typically it might be as follows:

Cube number	Color
(1)	Red
(2, ..., 25)	Green
(26, ..., 49)	Blue
(50, ..., 58)	White
(59, ..., 64)	Brown

1.2.1 Organization of WM for the Landscape Synthesis Problem

At its highest level the WM embodies relationship data, or context-interdependency data between a set of hypothesis which describe lower level regions of the model landscape. The context-interdependency data reflects constraints known to exist normally in the "real world". (See list 1 for examples.)

		RELATIONS		
H	H	INCLUDES	ADJACENT-ABOVE	ADJACENT-BESIDE
SKY	SUN	1	--	--
SKY	CLOUD	1	--	--
SKY	WATER	-1	.5	-.5
FIELD	HOUSE	.5	0	0
FIELD	ROAD	.5	.5	.5
ROCK	FIELD	-1	0	-.5

BINARY INTERDEPENDENCE RELATIONS of the form (H_1)(RELATION)(H_2).
Values from -1 to 1 range from "strongly denied" through "uncertain" to "strongly affirmed"; "-" means "value is redundant" since the relation is superceded by another (e. g. "includes" supercedes "adjacency" in some cases above).

List 1. Examples of Context Interdependency

A straightforward example of a strong constraint is that of "SKY includes SUN". Hence creating two regions of mosaic, one of which was labelled SUN and the other SKY, would be done such that the SUN was contained in the SKY.

An example of a weak constraint is "SKY adjacent-above WATER", meaning that if the hypothesis pair (SKY, WATER) have been created, then it is desirable to place SKY adjacent to and above WATER. However, this is not necessary, and it may not be given precedence if it causes conflict to occur in another step of the synthesis. At this point it may seem that the goal of the synthesis may be expressed in purely deterministic terms, that is to maximize the sum of the binary interdependency relations in the mosaic.

This is not quite the case, as it would imply an optimum landscape. Clearly this is in some sense unrealistic, as there are many scenes that could be called landscapes, many of which may even have joint membership in other categories of pictorial scenes. The goal of the synthesizer is better expressed in nondeterministic terms by saying that, after a large number of syntheses, it would be expected that some scenes would occur frequently, some less frequently and some of the possible 64! permutations of the 64 cubes, never at all.

At a lower level in the WM, information about all the possible regions, that can be hypothesized, exists. This information is in the form of a list of all the possible major regions thought to be found in a landscape and all the possible subregions that are thought to occur within those major regions. (See List 2).

	OPEN REGIONS	INCLUDED REGIONS
1.	SKY	SUN, CLOUD
2.	WATER	BOAT, ISLAND
3.	FIELD	HOUSE, TREE, ROAD
4.	ROCK	
5.	OVERCAST SKY	BLUE PATCHES
6.	ICE-SNOW	HOUSE, TREE, ROAD
7.	SAND	ROCK, HOUSE

List 2. Regions

At the lowest level in the WM the regions are described in terms of the cubes. That is the color of the cubes and the bounds on the number of cubes constituting any particular type of region.

By specifying the number of cubes in a region, the cubes positions on the quadruled rectangle are indirectly specified. Hence this model is based solely upon two physical attributes of the cubes. One is color, the other is position in the rectangle. Since there are only a finite number of

meaningful statements that can be generated about landscapes based on two attributes, more sophisticated models would be based on a greater number of physical attributes. Hence the building blocks in such a synthesis would also have to be more sophisticated, than the cubes use in Semantrix.

With this in mind a more sophisticated model can be constructed by incorporating inconceivable factors. E.g. Information concerning the climate that is to be associated with the synthesized scene could be incorporated. This would prejudice the synthesizing process, so that specifying an artic climate would increase the frequency of scenes having large white regions in them.

1.3 The WM Concept

Chapter 1 will be concluded with a brief introduction to the WM concept.

The WM concept is an attempt to incorporate into a cognitive system a prescribed data structure (called the cognitive map) that will enable the cognitive system to exhibit intelligent behaviour.

A general system theoretic model for a cognitive system is shown in Figure 1.3.1 The cognitive system partitions into two parts. A model of the stimuli's world (the WM) and a cognitive algorithm which interprets the stimulus under the control of the model. Together they are called a "cognitive memory", since they perform static data storage together with dynamic recognition of stimuli. The output of the cognitive memory after it has been excited by a particular stimulus is the interpretation. This may be characterized as a type of algorithmic association of features of the stimulus to features of the map. "Understanding" would be a bolder description.

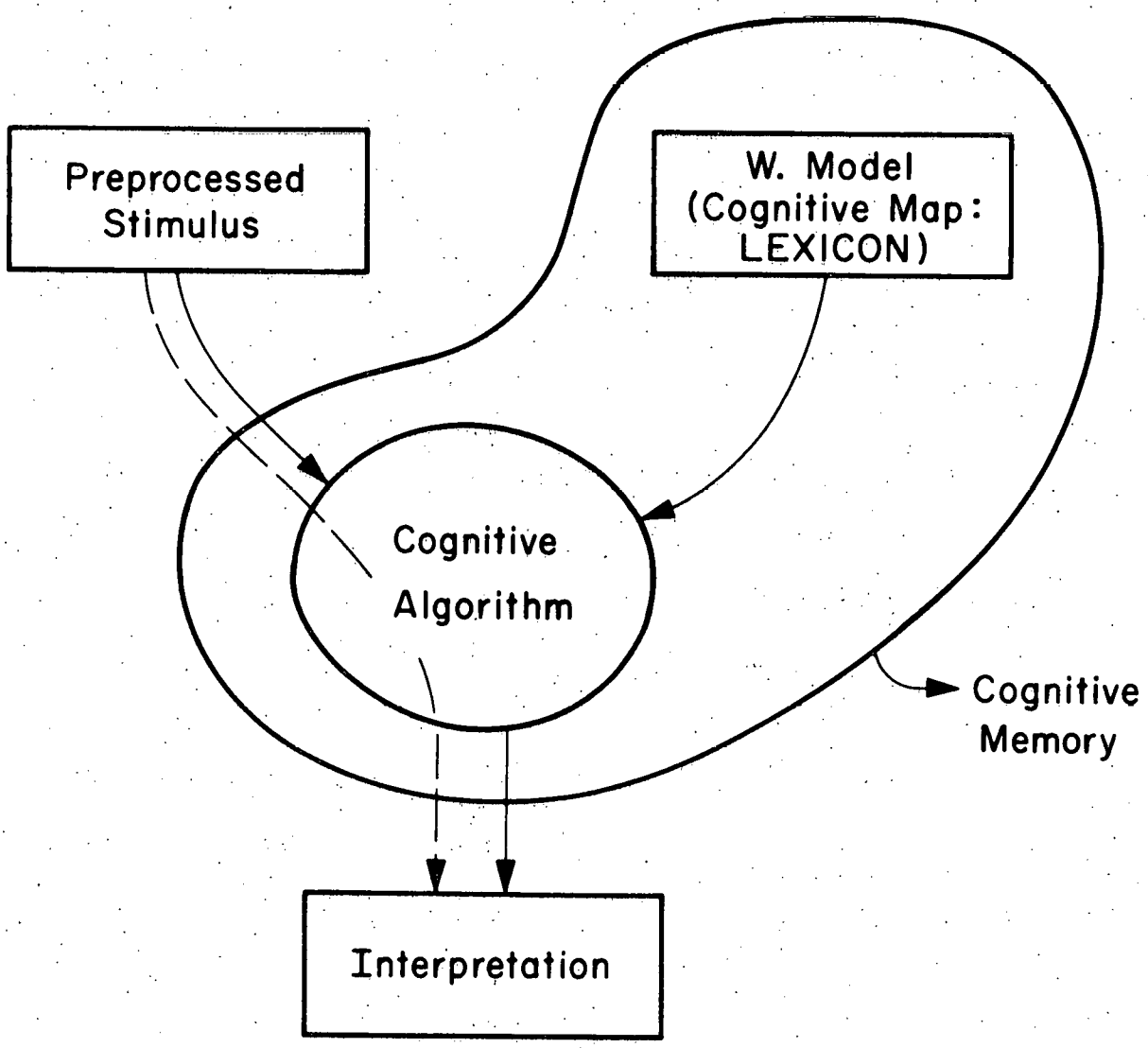


Figure 1.3.1 General Cognitive System

Semantrix is an attempt to test the fidelity of the type of prescribed data structure (i.e. a WM) that might be used in such a cognitive system.

In the past it was generally accepted that the WM could be generated by the now classical techniques of learning theory. These are a collection of statistical techniques for learning distributions from paradigms, that have long been known to people interested in signal processing classification and statistical estimation. In many cases it was found that the model learnt by such methods was in some sense adequate. However in many other cases this approach broke down.

As an example take the identification of a particular electronic network, say a flip-flop. The input to the system can be a circuit schematic. Under control of the WM the cognitive system must identify those drawings which represent flip-flops. Distinguishing circuit components using a WM which is learnt from a training set is quite feasible. Identifying configurations of these components which are flip-flops can only be accomplished if the WM incorporates some prescribed syntatic description of a flip-flop, which uses the circuit elements as terminal symbols. Another example where statistical models prove inadequate, is modelling natural scenes. The constituents - colored regions in the case of Semantrix - are easily discernable, but to be effective the WM must embody the semantic structure of such a class of scenes; in other words, reflect the constraints of our universe. The shortcomings of the statistical approach can be overcome, if the constraints can be identified and incorporated into a WM.

Work on natural scenes using Semantrix is an attempt to identify the semantic constraints and incorporate them into a WM; hence the acronym Semantrix.

The notion of a WM is relatively new to the study of artificial cognition. Two of the most recent papers to discuss this concept in depth are given in References 1 and 4.

2. A SYSTEMS DESCRIPTION OF SEMANTRIX

In this section a general description of Semantrix is presented, at a system theoretical level. The details of the implementation are deferred until later sections.

The system divides into three separate subsystems (see Figure 2.1).

1. The control logic.
2. The electromechanical hand-arm system for manipulating the cubes.
3. The cube locating system.

2.1 The Control Logic

The control logic interprets commands input through its single bilateral data channel. The design and operation of the control logic is discussed in detail in Section 3. Here it is sufficient to remark that the commands interpreted by the control logic can result in three types of response.

The first of these is the response to a command which is syntatically incorrect. This results in an error message being output along the bilateral data channel.

The second of these is the response to a command to move the hand-arm limb in one of its four independent motions. When a command to move the limb has been interpreted and then achieved, a completion message is output along the bilateral data channel.

The last type of response is a command to locate a specific cube (specified by number) on the quadruled grid that partitions the table top. A six octit number which uniquely specifies the grid square over which the cube's center rests, is output along the bilateral data channel.

2.2 The Hand-Arm Limb

The four independent motions of the hand-arm limb are shown in Figure 2.2.1. The hand can be moved to any one of 2^9 positions in the x-direction and any one of 2^9 positions in the y-direction. The rectangular table top is 36 inches square (see Figure 1.1.1), so this represents a linear precision of:

$$\frac{36}{2^9} = \frac{9}{128}" = \frac{5}{64}$$

That is, the motion of the hand conforms to a quadruled grid having a grid spacing of 5/64" in both x and y directions. To eliminate positional error the x and y motions are achieved by using a "torque proportional to error" closed loop servomechanism. The block diagram for this subsystem is shown in Figure 2.2.2. The detailed discussion of the subsystem is left until Section 4. However, a few remarks will be made in passing. First, the description of the servomechanism derives from the type of motor used. This generates a torque, proportional to the driving voltage. The driving voltage is a measure of the difference between the actual position of the hand and the desired position of the hand; it thus represents a measure of error from the hand's desired position. Hence the phrase "torque proportional to error".

Second, the input to the servomechanism is in digital form (the contents of a 9-bit register), which has to be converted to analog form to be compatible with the servomechanism. The conversion is done by a standard D/A converter, which introduces a possible $\pm 0.05\%$ of FS error. FS in this case corresponds to 80 inches. Therefore, the error is given by:

$$\pm \frac{5}{10000} \times 80 \doteq \frac{1}{2} \times \frac{5}{64}";$$

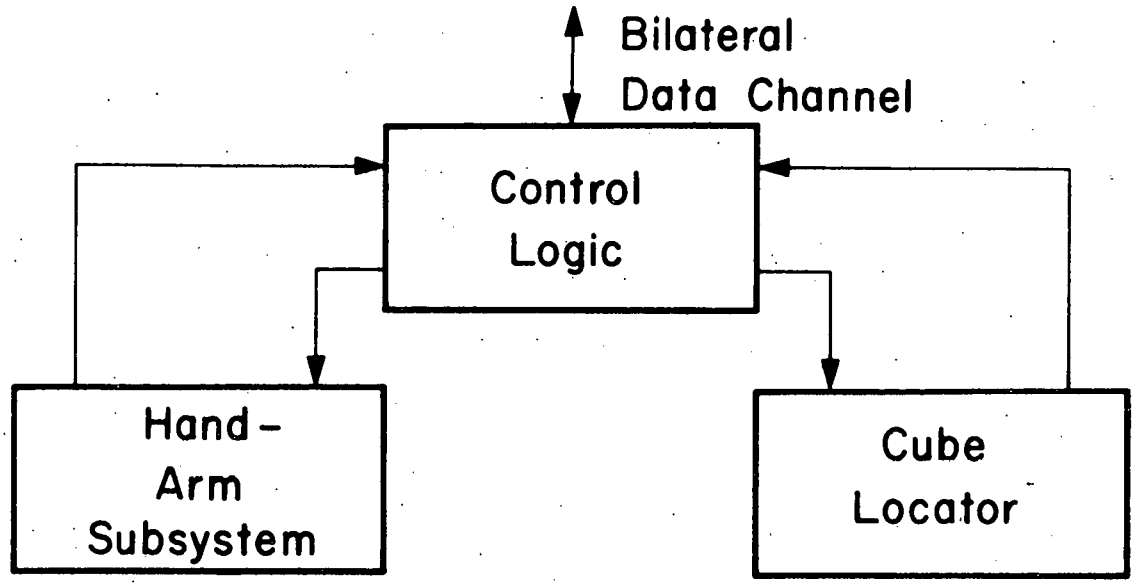


Figure 2.1 System Block Diagram

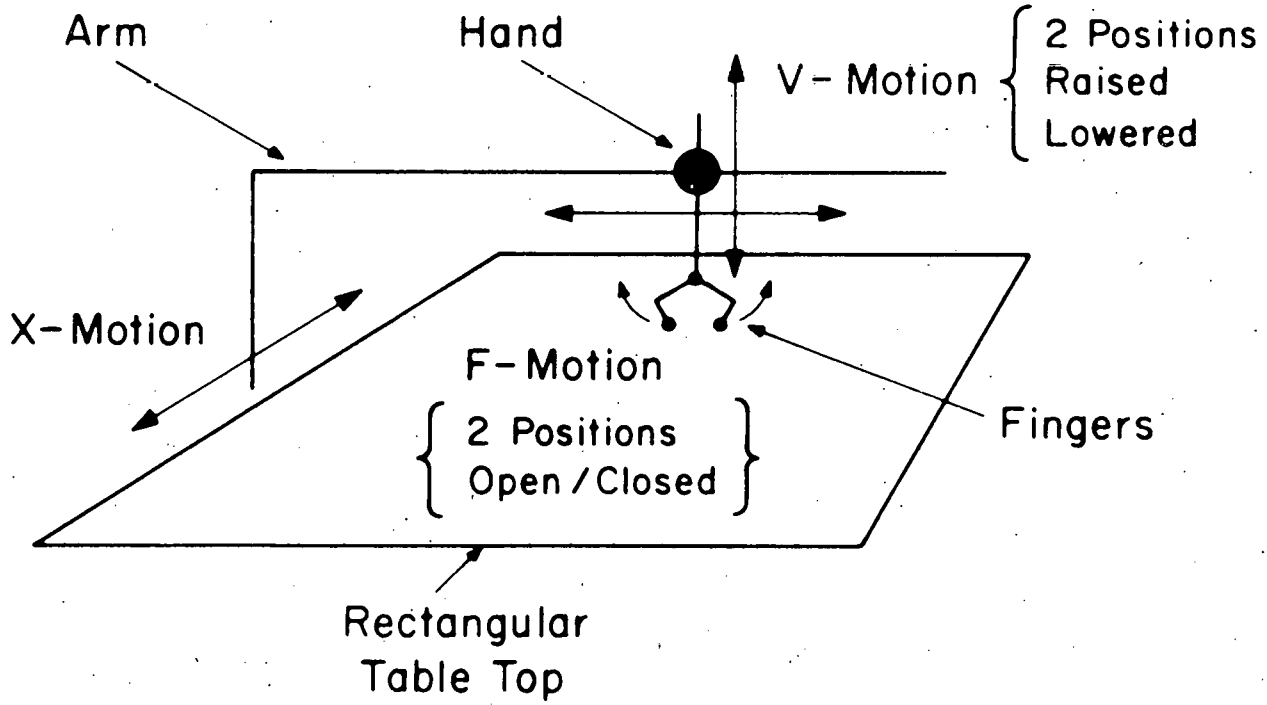


Figure 2.2.1 Motions of the Hand-Arm Limb

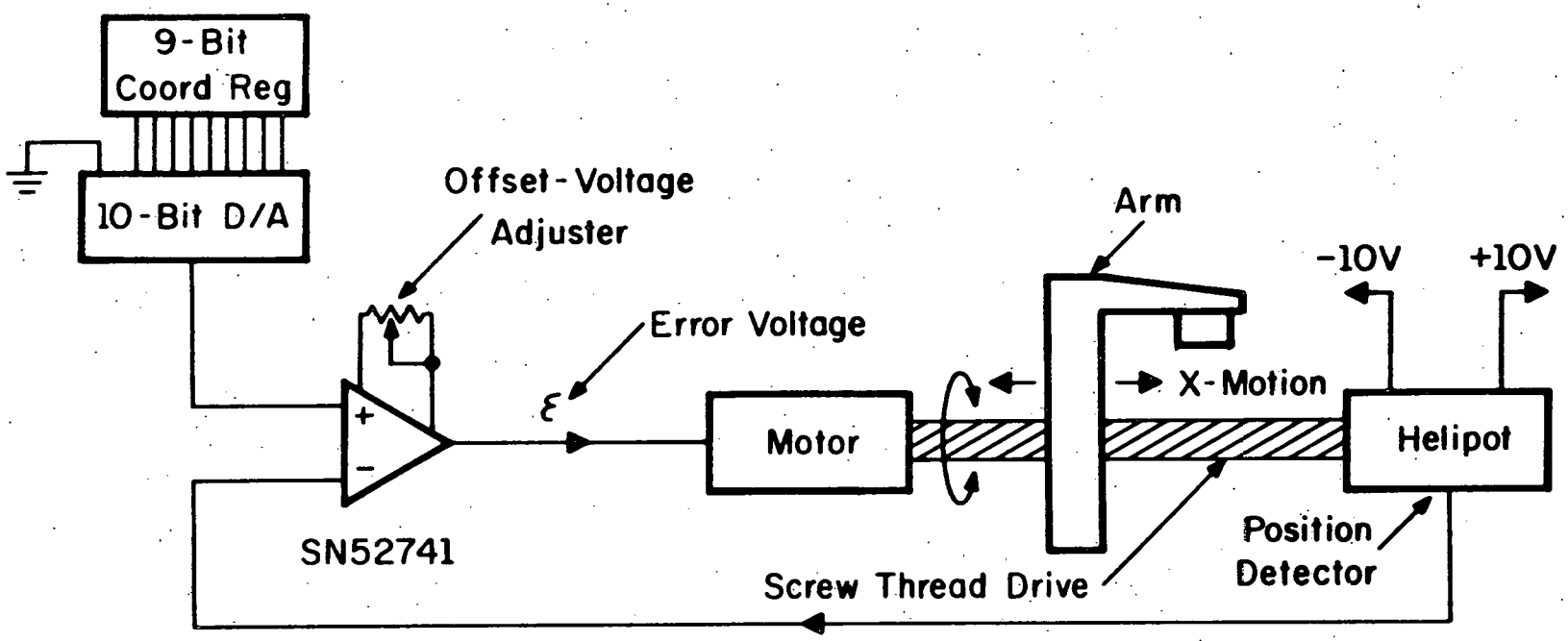


Figure 2.2.2 Block Diagram of the Servomechanism

that is, half the grid pitch. This is acceptable, as will be seen in later analyses. The other source of error in the x and y positions is introduced by the error detector (see Figure 2.2.2), which is realized by an operational amplifier. Here the null position is the main concern. For correct operation the offset voltage should be zero. This can be achieved by using an operational amplifier with an offset-voltage null capability.

The electromechanical hand has two more motions, the F-motion and the V-motion. Both of these are transition motions between two stable positions. The F-motion refers to the motion of the mechanical fingers on the hand (see Figure 2.2.1). These have two stable positions: "open" and "closed". This motion enables the hand to grasp the cubes on the table top. The V-motion refers to the vertical motion of the hand's subassembly, containing the fingers. The two stable positions are "raised" and "lowered". Once the cube has been grasped by the hand's fingers, this motion enables the hand to raise it clear of other cubes that might be lying on the table. By initiating motion in the x and y directions the cube may then be transported across the table top.

2.3 The Cube Locating System

The cube locating system is depicted in Figure 2.3.1. The 64 cubes available to Semantrix are uniquely identified by a 2 octit number on the range 00_8 - 77_8. To locate a particular cube its number is input to the control logic through the bilateral data channel, together with the appropriate instruction (see Section 3.1 for instruction formats). The instruction is interpreted by the control logic, and as a result the 2 octit number is handed to the locating system, together with a start signal. The start signal enables the power transmitter (see Figure 2.3.1 (a)), which begins radiating

electromagnetic energy from an inductor. The inductor is formed from a single loop of copper which runs beneath the perimeter of the table top. It also forms, together with some capacitors, the tank circuit of the power transmitter, the details of which are discussed in Section 4.

The energy is radiated for 1 mS, then the transmitter, automatically shuts off. The cubes contain a receiving coil which, in effect, forms a loosely coupled transformer with the loop of the transmitter. Capacitors in each cube store the energy they receive during the 1mS radiation period; then subsequent to this period, they discharge their stored energy at a predetermined time. The discharge is through another inductor which is wound on a cylindrical ferrite core (see Figure 2.3.1 (b)). This results in a pulse of electromagnetic flux coaxial with the ferrite core. The detailed design of the electronics in the cubes is discussed in Section 4. However, it should be noted that the cubes are passive; that is, they have no local power supply but derive all their operating power from the power transmitter.

The pulse of magnetic flux produced by each block is normal to the table top and is detected by a matrix of conductors which is just under the surface of the table top (see Figure 2.3.1 (c)). The construction of the matrix is shown in Figure 2.3.1 (d). There are two sets of 89 loops, which are etched onto opposite sides of a printed circuit board, at right angles to one another. The loops are open at one end so that a small potential difference is induced between the ends of any loop if the flux through the loop changes. This induced voltage is sensed by a threshold circuit (see Section 4 for details of the threshold circuits).

The table top is partitioned into a matrix of 89 x 89 0.4" square cells or quadruled grid, by the orthgonal loops. Hence, the flux

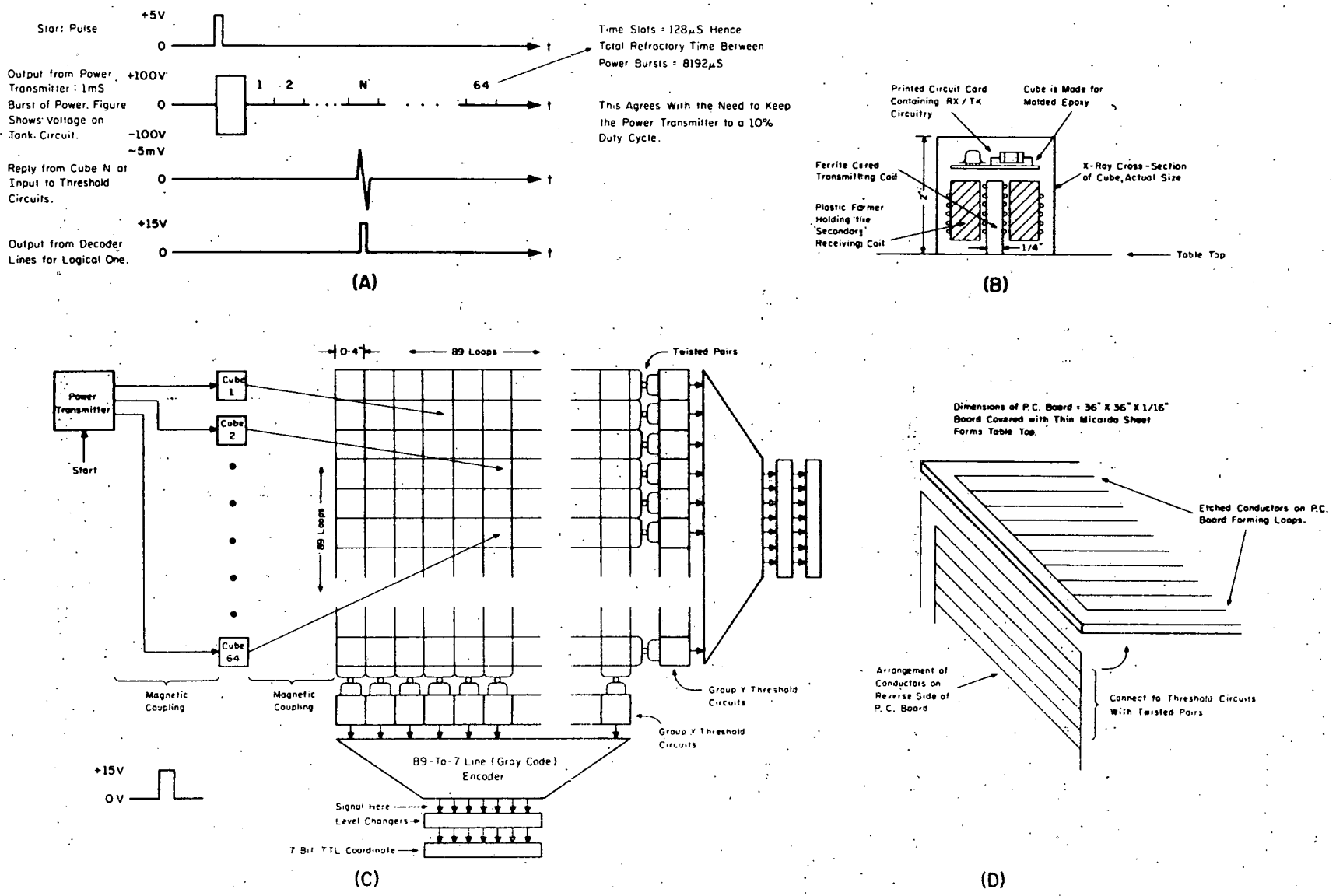


Figure 2.3.1 The Cube Locating System

change produced when each cube releases its stored energy will be sensed as having occurred in one of the 89 x 89 square cells (assuming of course, that the cube is on the table top). Two threshold circuits, one from the x group and one from the y group, are thus activated when each cube releases its stored energy. There are 89 threshold circuits in both the x and y group. The 89 output lines of each group are encoded into 7 bits of Gray code by a diode matrix. Thus the position on the quadruled grid of the center of each cube is encoded into two 7 bit Gray code numbers; one representing the x coordinate, the other the y coordinate.

In order to distinguish the cubes, the point in time when they release their stored energy is made unique. This time out, between the termination of power transmission and the release of the stored energy in a cube, is governed by an RC time constant. Each cube has a different time constant which is chosen so that its time out is related to its code number N ($0 < N < 63$) by the following equation:

$$T_{OUT} = (N \cdot 128 + 64)\mu S$$

(Recall that each cube has associated with it a unique two octit number on the range $00_8 - 77_8 = N_{10}$). The time period after the power transmission can be regarded as being divided into 64 time slots of 128μS each (see Figure 2.3.1 (a)). Cube N will then be expected to reply in the $(N + 1)^{-th}$ such time slot. As implemented, the instruction to locate a block will request the coordinates of a specific cube, N. In response to this the control logic enables the inputs to the buffer register, which receives the two 7 bit coordinates, during the N^{th} time slot only. Hence the coordinate buffers contain either two 7 bit Gray code numbers, representing the position of cube N on the table

top, or, if the cube is not on the table top, 0. The control logic output the contents of this buffer in a prescribed format (see Section 3) through the bilateral data channel.

Since there are 64 time slots the T_{OUT} for cube N = 63, the worst case, should be within $\pm 0.8\%$ of its normal value over the normal operating temperature range. T_{OUT} is a linear function of the RC time constant, so it is necessary to use resistors and capacitors with the above degree of temperature stability to form each cubes time out generator. This is not a very difficult or expensive specification to satisfy. In fact, this worst case analysis is only true for N = 63; the tolerance progressively loosens as N decreases. For N = 0, the tolerance is $\pm 100\%$.

Two more points should be mentioned before concluding the section. The first concerns the use of Gray code to encode the x and y positions.

It is possible that two adjacent loops in either the x or y direction may detect the same cube's reply. Hence two input lines on one of the 89-to-7 line encodes would be activated at once. The result would be as follows: consider two adjacent input lines that would normally encode as

$$A = a_6 a_5 a_4 a_3 a_2 a_1 a_0$$

and $$B = b_6 b_5 b_4 b_3 b_2 b_1 b_0$$

When they are activated simultaneously the resultant encoding, C, is given by the bit-wise logical OR of A and B.

$$C = c_6 c_5 c_4 c_3 c_2 c_1 c_0$$

where $$c_i = a_i \vee b_i \qquad i = 0, \ldots, 6$$

If A and B are in Gray code, then C = A, or C = B, since adjacent Gray codes differ in one bit only. Had A and B been in a normal binary sequences the above would not be true. Consider:

		Normal	Gray
	A =	0001000	0001100
	B =	0000111	0000100
then	C =	0001111	0001100

By using the Gray code approach the loss in precision of the cube's center is held at $\pm$0.4". The fingers of the hand are designed to accommodate this degree of uncertainly.

The second point to be mentioned concerns the precision of the hand's motion. To place cubes adjacent to one another requires high precision in the motion of the hand. Between centers two adjacent cubes are 2" apart. The hand moves in increments of 5/64," therefore, it is possible to place two cubes within 1/32" of each other. This tolerance also allows for the uncertainly in the hand's motion.

3. A LOGICAL DESCRIPTION OF SEMANTRIX

In this section the design and operation of the control logic is described.

Semantrix is intended for use in a logical system, such as the one shown in Figure 3.1. The system comprises:

1. Semantrix (S)
2. A teletype (T)
3. A data processor (D)

As was mentioned in the introduction, Semantrix can form a stand alone system with just the teletype. This is achieved, in a system such as the one shown in Figure 3.1, by moving the switch to the D position, thus putting the system into direct mode. However, this should only be regarded as a test mode, in which the logicality of Semantrix' control logic may be tested. In order to operate the machine in the type of experiments outlined in the introduction, the switch must be moved to position R, putting the system into remote mode. In this mode the data processor instructs the control logic of Semantrix, and the teletype is used to initiate the I/O subroutines in the data processor, which handle the data flow to and from Semantrix. Furthermore, the storage ability of the data processor provides a residence for any model or cognitive map. The complete assemblage forms a cognitive system, as described in the introduction.

The double pole single throw switch which connects Semantrix to the data processor or directly to the teletype also switches different clock generators into the clock bus of the control logic. In direct mode a 110Hz clock

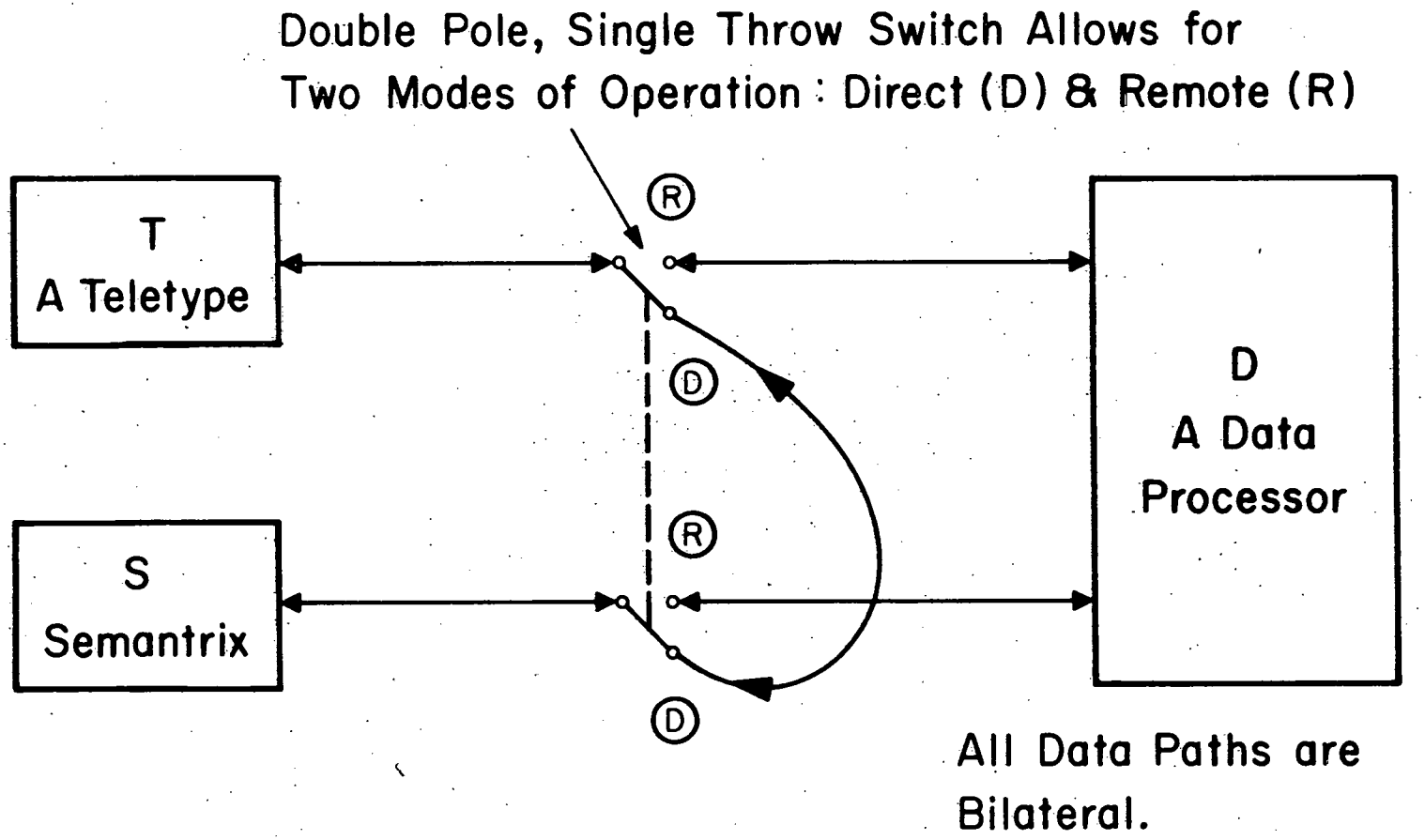


Figure 3.1 Semantrix as a System Components

drives the clock bus. This is compatible with the 110 Baud data rate of the teletype. In remote mode as fast a clock as is compatible with the data processor should be used.

3.1 The Instruction Set

Before the control logic is examined in detail, it is necessary to give the instruction set that is interpreted by Semantrix. All instructions enter the bilateral data channel of the control logic in serial form, each character being in standard teletype format (see Section 3.3 for details). Replies generated by Semantrix are output serially along the bilateral data channel, also with each character in standard teletype format. For the purpose of discussing the instruction set it is sufficient to consider that the instructions are input through a teletype, and that the replies are received by a teletype. When this is not in fact the case i.e. when Semantrix is connected to a data processor, the same format is preserved, but the data rate is considerably increased; the data processor is programmed to have the I/O characteristics of a teletype. The bilateral data channel is the normal type used to communicate with the ASR33 Teletype, and is sometimes referred to as a half-duplex channel.

There are four types of instruction that can be input into Semantrix, discounting incorrect ones.

3.1.1 Type 1 Instructions

These instructions are used to request the coordinates of any block.

Send: N mn Rt — The letter N followed by two octal digits m and n is typed in through the teletype, together with the non-printing character "carriage return" (Rt). Recall

that each of the 64 cubes are specified by a unique number mn which lies on the range 00_8 - 77_8. A reply is generated by Semantrix, which produces the following result at the teletype.

Receive: Lf $x_2x_1x_0$ $y_2y_1y_0$ Rt Lf|Lf E Rt Lf

The non-printing character "line feed" (Lf) is sent from Semantrix to the teletype followed by a six octit number which uniquely determines the cube's position on the quadruled grid. Recall that the grid partitions the table top into 89 x 89 squares; hence $x_2x_1x_0$ or $y_2y_1y_0$ lie in on the range 001_8 - 131_8. In the case where a cube is not on the table top $x_2x_1x_0$ and $y_2y_1y_0 = 000_8$. The non-printing characters Rt and Lf terminate the reply, causing the teletype's carriage to be returned to its left most position on a new line.

If there is a transmission error between the teletype and Semantrix, or if the instruction input through the teletype is syntatically incorrect a standard error message "Lf E Rt Lf" is generated by Semantrix and received by the teletype. The allowed formats that the instruction "N nm Rt" can have without being rejected by the control logic and causing an error message to be output is discussed in Section 3.2.

3.1.2 Type 2 Instructions

These instructions are used to move the hand-arm limb across the table top. It was seen in 2.2 that the x and y motions were subdivided into incremental motions of $1/2^9$ of FS. Hence to specify a point to which the hand should

move, the two 9 bit numbers must be used. For convenience octal notation is used, which means a six octit number must be input to Semantrix. The instruction has the following form:

Send: C $x_2x_1x_0$ $y_2y_1y_0$ Rt

A C is typed in followed by a six octit number; which is on the range: 000000_8 - 777777_8. A non-printing carriage return delimits the message. Two possible replies result.

Receive: LF E Rt LF|Lf Bell

The first of these denotes a syntax error as before. The second is nonprinting, and causes the teletype's bell to ring, indicating the instruction has been executed by the control logic.

3.1.3 Type 3 Instructions

These instructions are used to lower and raise the hand.

Send: L E Rt|R E Rt The first instruction lowers the hand, the second raises it.

The possible replies are the same as for type 2 instructions, and they have the same meaning.

3.1.4 Type 4 Instructions

These instructions are used to open and close the hand's fingers.

Send: E 0 Rt|E 1 Rt The first instructions opens the fingers, the second closes them.

Once again, the possible replies are the same as for Type 2 instructions, and they have the same meaning.

3.2 The Input Format

The 4 types of instructions can be input through a teletype in various types of format. With certain limitations the strings of characters which represent instructions can be interspersed with any sequence of blanks and other characters.

The following is an acceptable type 1 instruction:

$$* N * 3 * 2 * Rt$$

where * can be the null string or any string of teletype characters. The only restriction on the stars is that they do not contain another allowed input sequence. Formally, this limitation can be described in a recursive fashion. If $w_1 w_2 \ldots w_r w_{r+1} \ldots w_n Rt$ is an allowed input sequence and

$$_1W_r = w_1 w_2 \ldots w_r$$

$$_{r+1}W_n = w_{r+1} w_{r+2} \ldots w_n$$

Then the input sequence

$$w_1 w_2 \ldots w_r A_r w_{r+1} \ldots w_n Rt$$

is allowed iff the set of sequences given by

$$\{{}_1W_r^* \times A_r \times {}_{r+1}W_n^* \times Rt\} \rightarrow w_1 w_2 \ldots w_r A_r w_{r+1} \ldots w_n Rt$$

are not. Where the * operation denotes the set of all subsets with order preserved. E.g. if A = {a, b, c}

$$A^* = \{a, b, c, ab, ac, bc, abc\}$$

whereas if A = {b, a, c}

$$A^* = \{a, b, c, ba, bc, ac, bac\}$$

The X operation denotes the Cartesian product.

For example, if N329Rt were input it would be accepted and would be taken to be the same as N 32 Rt. But N 32 EO Rt would cause an error message to be printed out, because N 32 Rt and E 0 Rt are both allowable. The don't-care assignment of the stars allows an extended mnemonic facility and a flexible input format.

3.2.1 Extended Mnemonics

The following are examples of the extended mnemonic facility made available by the flexible input format.

Type	Essential	Extended
1.	N 36 Rt	NUM 36 Rt
2.	C 142 613 RT	COORD 142 613 Rt
3.	L E Rt|RE Rt	LOWER Rt|RAISE Rt
4.	E 0 Rt|El Rt	OPEN 0 Rt|CLOSE 1 Rt

3.3 The Control Logic

Having discussed the set of instructions together with their allowed formats, the organization of the control logic that interperets them can be described.

Figure 3.3.1 is a block diagram of the control logic, depicting the signal flow between the component parts of the logic. All logic is implemented in Texas Instruments SN74 series TTL. The design guides and a discussion of the circuit limitations are to be found in References 3 and 5.

The heart of the control logic is the 11 bit by 16 word sequential read only memory (SROM). In each of the 16 words an 11 bit character in standard teletype (TTY) format is stored. For details of the SROM contents see

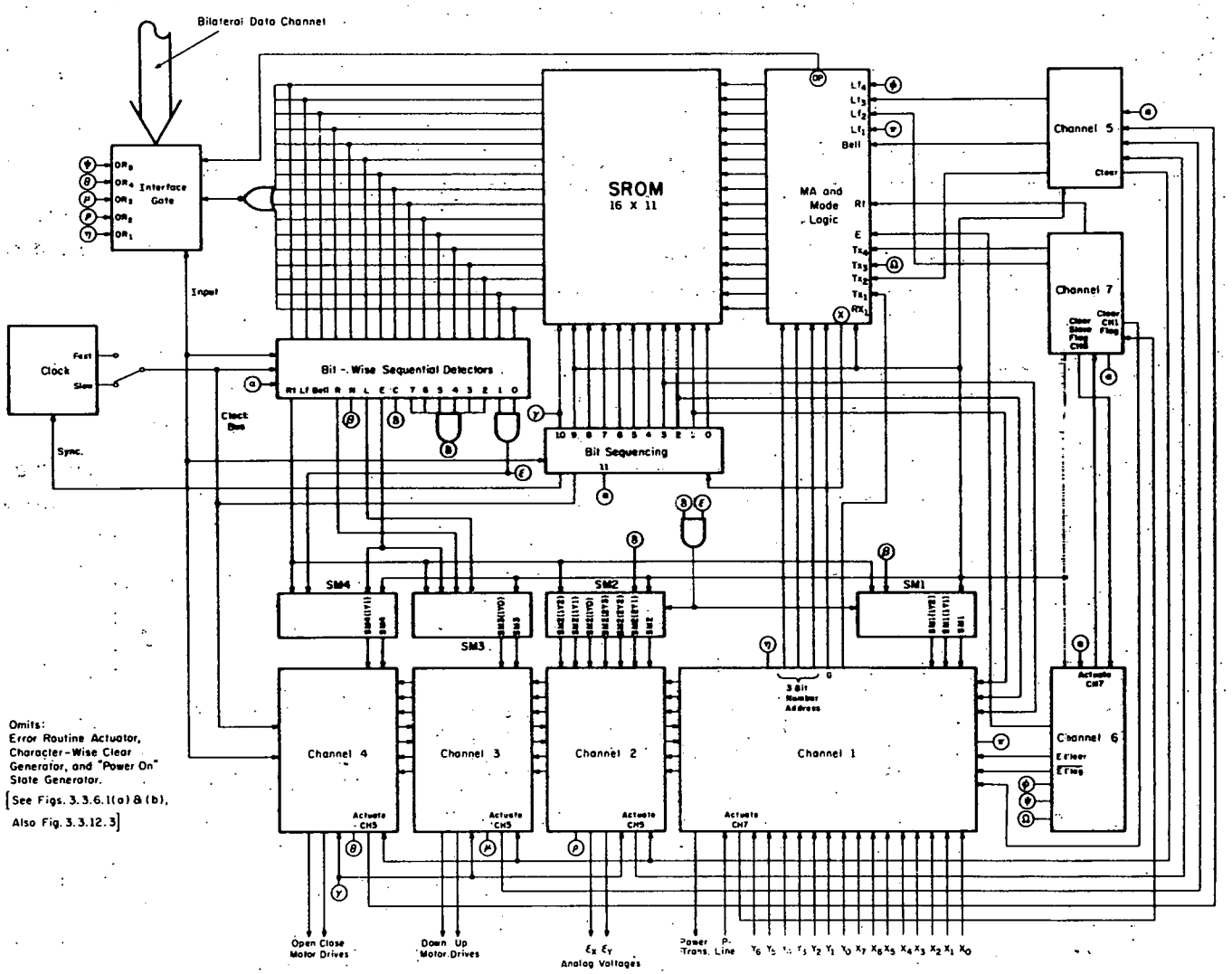


Figure 3.3.1 Block Diagram of the Control Logic

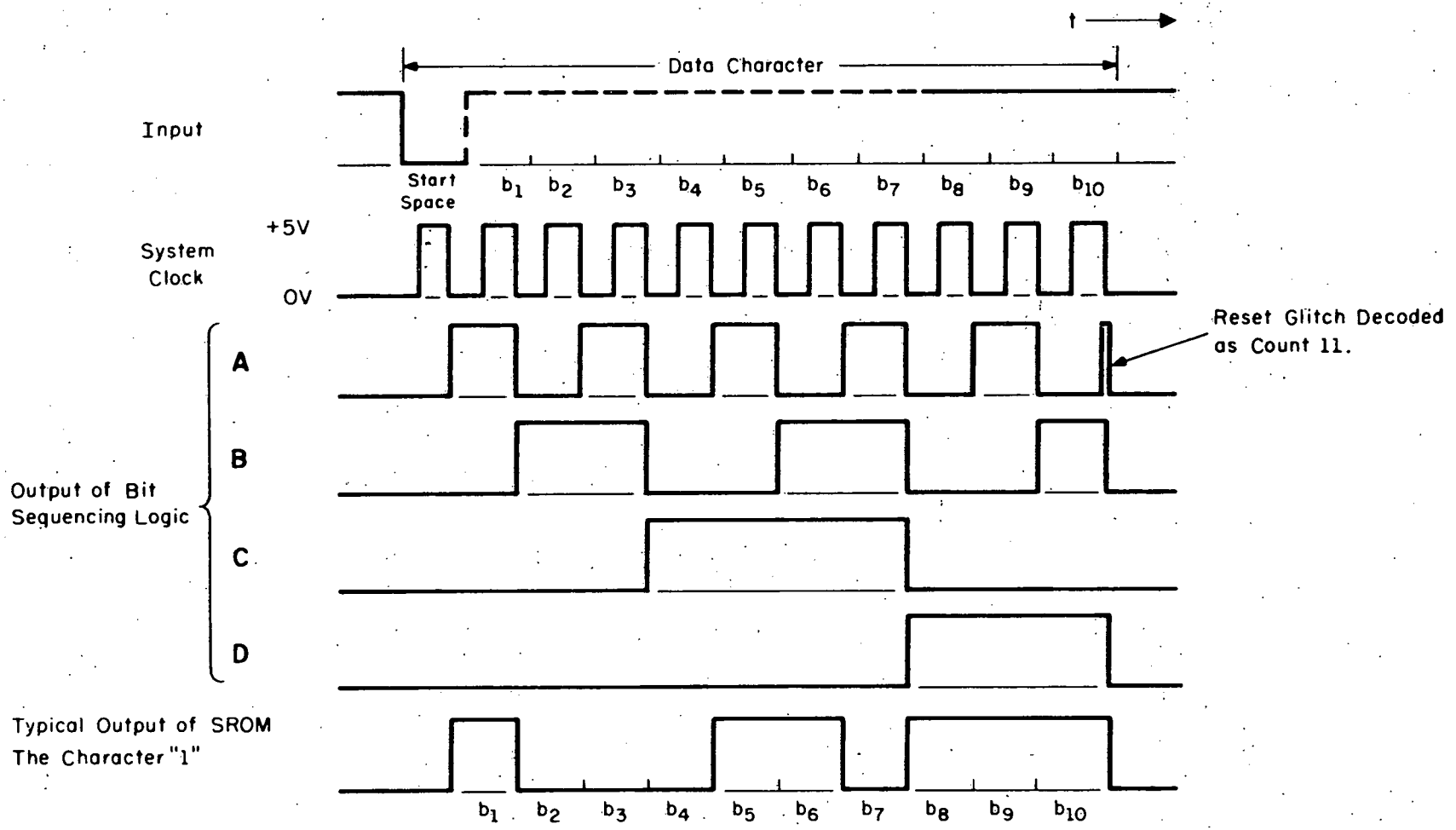


Figure 3.3.2 Waveforms from Bit Sequencing Logic

Appendix B. The standard TTY format can be described by the following bit string:

0XXXXXXX111

where the 7 X's are an ASCII character.

Each word in the memory is output in a bit-wise serial fashion, under the control of the bit sequencing circuitry, which is basically a modulo-11 counter and decoder. The counter counts clock pulses from the system clock. The waveforms are shown in Figure 3.3.2; note the glitch that occurs after the count 10 pulse from the decoder. This is used to reset this and other counters in the system.

The output of each individual memory location can be inhibited by suitable signals from the MA (memory address) and Mode logic. The Mode logic is in essence a flipflop which indicates whether or not the control logic is in the receive mode or the transmit mode. When the system is in the receive mode none of the memory outputs are inhibited and the action of the bit sequencing logic is to output all 16 characters from the SROM simultaneously.

This facility is used to identify input TTY characters. The falling edge that begins every TTY character simultaneously starts the bit sequencing circuitry and symchronizes the clock (see Figure 3.3.2). The character is broadcase to an array of 16 bit-wise character detectors where it is compared simultaneously to each of the 16 characters output from the SROM in a serial bit-wise manner. In this way input characters are classified.

In order to interpret input instructions, it is necessary to be able to identify certain strings of characters. This is achieved by using

4 character-wise detectors SM1, SM2, SM3 and SM4 (see Figure 3.3.1). There are 4 types of instructions (see Section 3.1); SM1 detects type 1 instructions, SM2 type 2, etc. In the instance of a string of characters not being identified by any of the 4 character-wise detectors, the machine is put into transmit mode and an error message is transmitted down the bilateral data channel. This corresponds to a message being syntatically incorrect.

Once a character-wise detector has accepted an input message as an allowed instruction, certain action must be taken; e.g. move the limb, locate cube number N, etc. This is done in the following fashion. Each character-wise detector has slaved to it an interpretive channel which operates asynchronously to it (see Figure 3.3.1). When a message has been accepted by a character-wise detector, it sets the channel flag of the channel which is slaved to it and passes prescribed informtion to the channel for interpretation and eventual use at the output end of the channel (the output end might drive the limb's motors, etc.).

The channels also return completion signals, when the action required by the input message has been effected. When a channel flag is set it inhibits any input through the bilateral data channel by means of a set of OR gates in the interface circuitry. The completion signals reset the channel flags, place the machine in the transmit mode and initiate the output of a reply message along the bilateral data channel to the TTY or data processor.

The retransmission of TTY characters along the bilateral data channel also makes use of the SROM. Naturally the only characters that can be retransmitted are those contained the SROM. When a channel requires to

output a string of characters, it sends an appropriate signal to the MA and Mode logic, which sets the machine into the transmit mode, thus enabling the output path of the bilateral data channel. It simultaneously inhibits, through the memory address logic, all the outputs of the SROM except that of the first character in the required string and initiates the counting sequence in the bit sequencing logic. The first character of the reply message is thus output. The count 11 glitch generated by the bit sequencer is used to reset the mode of the machine to receive. The mode remains unchanged until the next asynchronous signal comes from the channel requesting the output of the next character in the reply message.

When the error routine is initiated due to a disallowed input string, all the channel flags must be reset together with the character-wise detectors, some of which may have been in the middle of accepting the input string. Furthermore, when one character-wise detector has accepted a string it must reset itself and the other three character-wise detectors, since the other detectors may be in a state of partial acceptance. The error routine generates its reply in the same fashion as the channels and is complete with a flag so it can be regarded as being pseudo-asynchronous.

Figure 3.3.3 shows the asynchronous flow or signals from the main logic to a channel. There are four channels which communicate with the main logic in an asynchronous fashion. These are: channel 1, which is responsible for cube location; channel 2, which controls the x-y motion of the hand; channel 3, which controls the up/down motion of the hand; and finally, channel 4, which controls the open/close action of the hand's fingers. All of them communicate with external devices which are time independent of the control

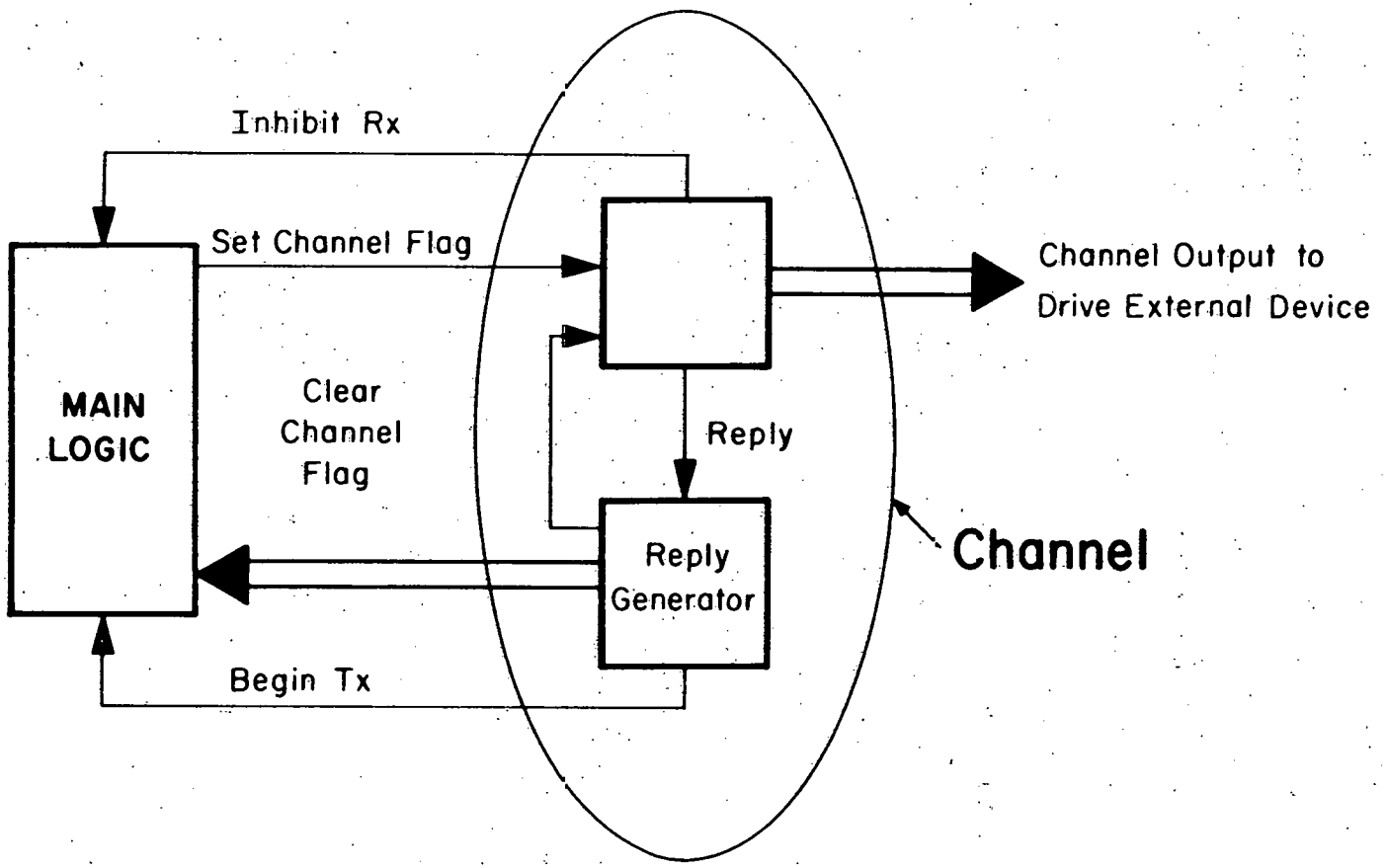


Figure 3.3.3 The Asynchronous Flow of Signals Between the Main Logic and a Channel

logic's clock. (E.g. channel 1 is dependent on the timing requirements of the power transmitter.)

The reply message is not generated directly by each of the channels 2 through 4, since it is the same for each, namely Lf Bell. This task is delegated to channel 5, which upon the receipt of an appropriate pulse from any one of channels 2 through 4 will cause the retransmission of the above non-printing characters. Channel 6 is the pseudoasynchronous channel which generates the error message Lf E Rt Lf. Once again the output of the characters Rt Lf is not done directly by channel 6 since this pair of characters also terminates the channel 1 output message. Hence it is more economical in terms of logic to delegate the retransmission of Rt Lf to another channel, in this case channel 7 which is activated by a suitable pulse from either channel 1 or channel 6 (see Figure 3.3.1).

Several points should be noted. First, the SROM has a dual role. In the receive mode it is used in the sequential bit-wise analysis of incoming TTY characters to the control logic. In the transmit mode it is used as a normal ROM to access data (in this case TTY characters) that is to be serially output. The second point to note is that the machine readily decomposes into a series of submachines. There is the SROM with the MA and Mode logic. Then there are the bit-wise analysers. Finally there are four submachines (SM1, SM2, SM3, and SM4) and their dependent channels, which perform the character-wise analysis and interpretation. This simple decomposition of the control logic makes debugging easier through replication of design techniques and also speeds the basic system design. The third point to notice is that the use of asynchronous channels permits the use of any speed of clock in the main logic. Each channel is slaved to the time of operation of the external device

that it controls, but since information is exchanged between the main logic and the channels asynchronously, the main logic is time independent of the external devices. It is, however, dependent upon the data rate used on the bilateral data channel. If a TTY is at the other end this is 110baud; if a computer is at the other end the data rate should be considerably higher. The last point to notice is that the system clock is synchronized by the falling edge which begins each input character. Hence the clock need only maintain its synchronism for 11 clock cycles (the number of bits in a TTY character). This relaxes the requirements on clock frequency drift. A drift of $\pm 2\%$ is acceptable.

To complete the description of the system logic, a detailed discussion of some of the various boxes shown in Figure 3.3.1 follows, beginning with the system clock. The logic convention used in logic diagrams is MIL-STD-806B.

3.3.1 The System Clock

The logic diagram of the clock board is shown in Figure 3.3.1.1. Two clocks are on the board. One operates at a frequency of 110Hz to be compatible with the TTY, the other at a frequency compatible with the data processor used to control Semantrix. The state of the flip-flop (FF) determines which clock drives the clock bus, and it is set by the Remote/Direct mode switch discussed in Section 3. SN7440 NAND buffers are used to drive the clock bus. The operation of the clock itself, which is built up from standard TTL SN7405's is quite and is dealt with in Appendix A.

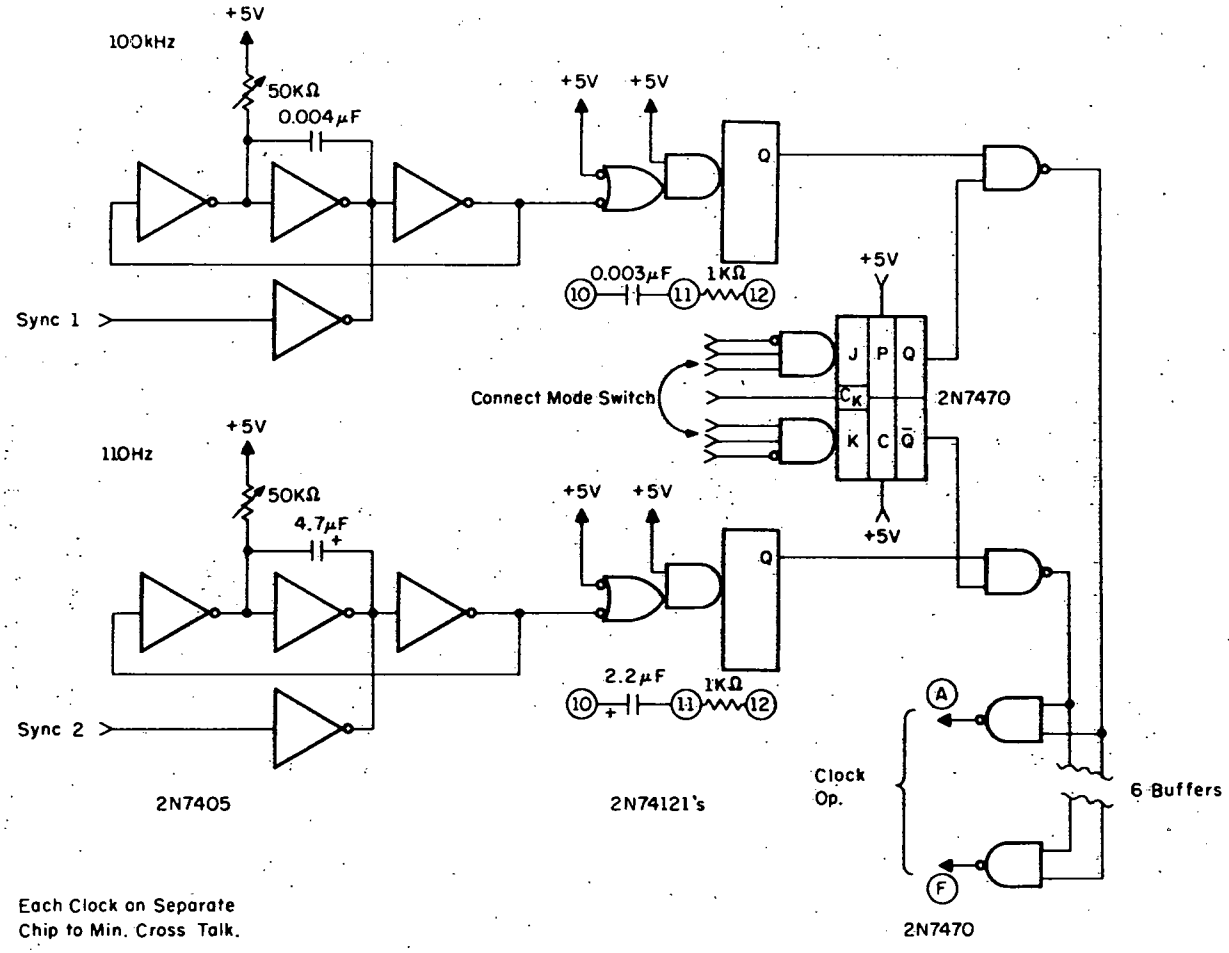


Figure 3.3.1.1 The System Clock

3.3.2 The Bit Sequencing Logic

This is diagrammed in Figure 3.3.2.1. A 4 stage ripple counter (the SN7493) is used to count the clock pulses, and a 4-to-16 line demultiplexor is used to decode the output of the counter.

Using a ripple counter can lead to hazards in any combinatorial logic driven by the counter, because of transition states that the counter can cycle through in going from one state to another. To illustrate this point consider the following state of the counters FF's:

D C B A

0 0 1 1

Upon receipt of a clock pulse to FF A the next state should be:

D C B A

0 1 0 0

In fact the following cycle occurs:

D C B A

0 0 1 1

↓

0 0 1 0 ↓ 0 0 0 0 ↓ } Transient, unstable states

0 1 0 0

In other words, unwanted output pulses occur on the count one and count zero lines of the decoder.

In the control logic six count lines are utilized: count 1, count 2, count 3, count 9, count 10, and count 11. It can be seem from the state transition diagram for the ripple counter in Figure 3.3.2.2, that 9, 10, and 11 never occur as unstable states; hence, no undesirable

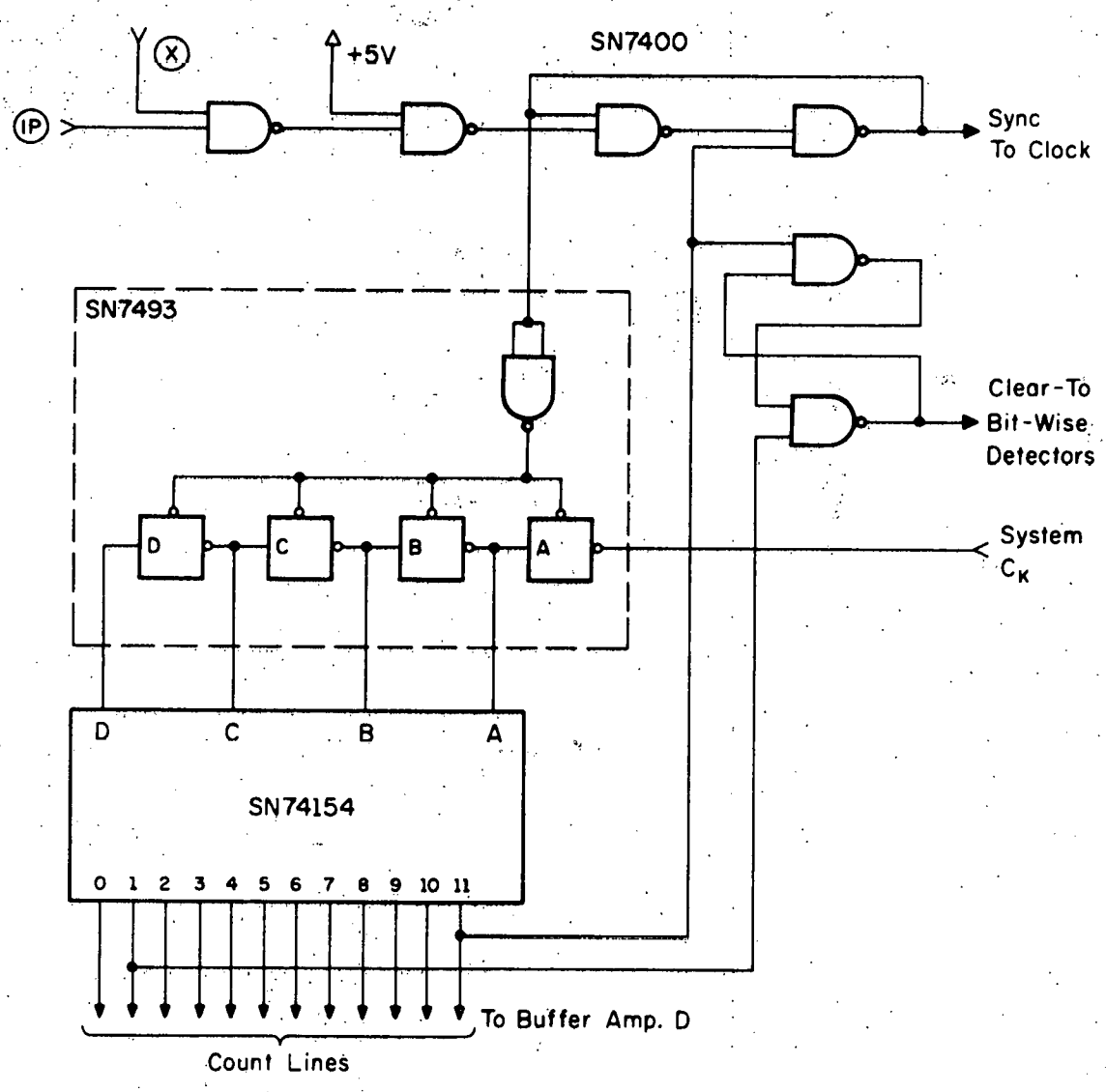


Figure 3.3.2.1 The Bit Sequencing Logic

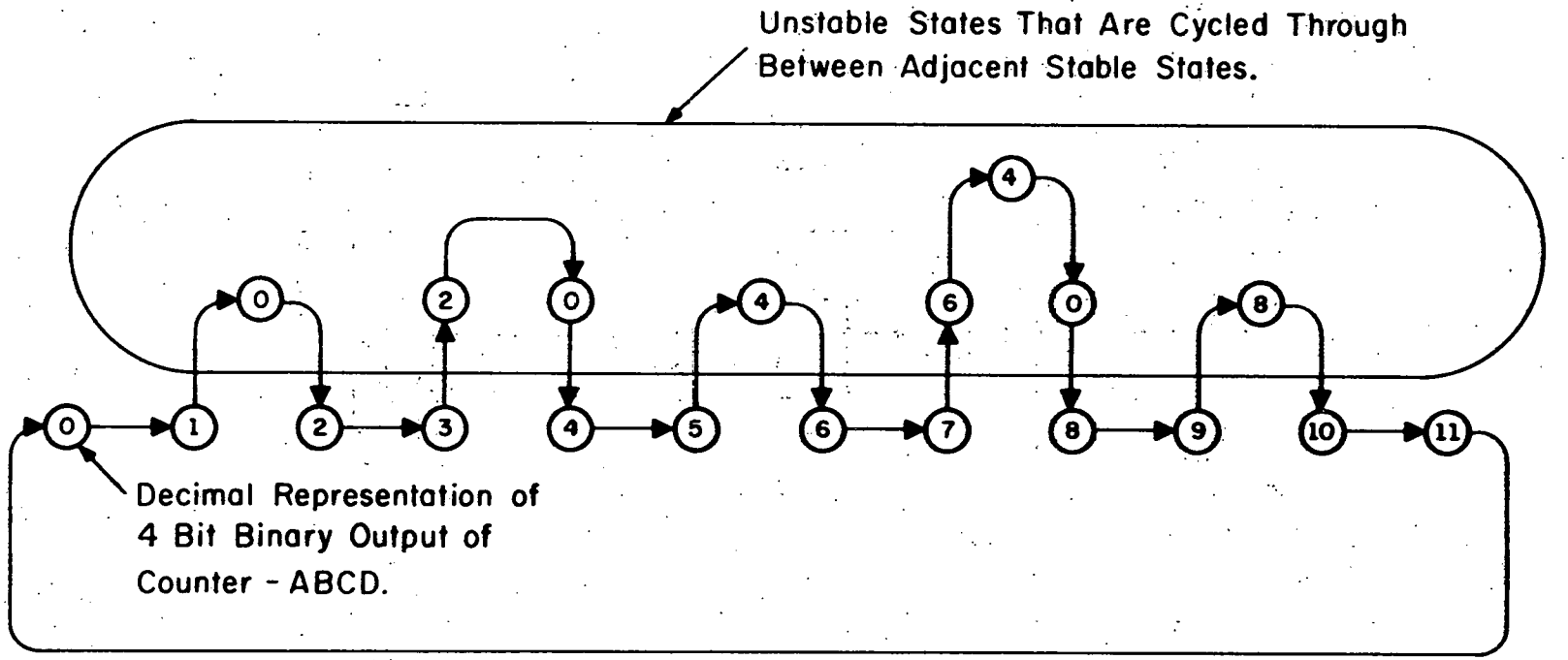


Figure 3.3.2.2 State Transition Diagram for the Ripple Counter

pulses occur on the count 9, 10 and 11 lines. The same can be said of 1 and 3, but not our count 2 in however, count 1, 2, 3 are all used. with the system clock as a strobe, so that occurrences of count 2 pulses due to unstable transition states are masked. The transition of particular interest is that between count 3 and count 4, when the count 2 state occurs in transition. Here the counter cycles through count 2 and count 0 before settling at the stable state, count 4 (see Figure 3.3.2.2). The FF's change state on the falling edge of the incoming clock pulses; hence, the unstable count 2 state above will not be concurrent with a clock pulse (unless the system clock is run at a rate near to the maximum speed of operation of the SN series logic). Hence its output is masked, since pulses on the count 2 line of the decoder are strobed by the clock at their point of application.

There is another reason for strobing the count 1, 2 and 3 outputs of the decoder, besides eliminating the effect of the count 2 unstable state, and that is connected with aligning the input of data to channels 1, 2, 3 and 4. It is discussed in Section 3.3.4 and Section 3.3.8.

3.3.3 The SROM

This is constructed from 16 SN74150 multiplexors (see Figure 3.3.3.1). The operation of the SN74150 multiplexor can be described by the Boolean equation:

$$\overline{W} = \overline{S}(\overline{A}\overline{B}\overline{C}\overline{D}\ E_0 + A\overline{B}\overline{C}\overline{D}\ E_1 + \overline{A}B\overline{C}\overline{D}\ E_2 + \ldots + ABCD\ E_{15})$$

An eleven bit TTY character is hard wired into $E_0E_1E_2 \ldots E_{10}$ and the bit sequencing is done by applying the modulo-11 counting sequence to the A, B, C and D inputs of each multiplexor. Bits E_{11} through E_{15} are not used so they are left floating. Each character may be accessed by starting the bit

sequencing counter and bringing the strobe line low. The 16 strobe lines are used in memory addressing, and are controlled by the MA and Mode logic.

To store the TTY character "1" shown in Figure 3.3.3.2 the following permanent connections are made:

E_0 = H E_4 = H E_8 = L

E_1 = L E_5 = L E_9 = L

E_2 = H E_6 = L E_{10} = L

E_3 = H E_7 = H $E_{11} \rightarrow E_{15}$ floating

Where H = high, or +5 volts.

L = low, or 0 volts.

The output is also shown in Figure 3.3.3.2 (with S = L).

Notice that the channel to the TTY normally requires a high input in the absence of a character transmission. This is achieved by the interface logic. The control logic, in its idle state, is in the receive mode, which inhibits any input to the interface logic. In the absence of an input the interface logic establishes the normally high output required by the TTY. This masks the fact that the inhibited input to the interface logic from the SROM is normally low in the idle state. The contents of the SROM are listed in Appendix B.

3.3.4 The Bit-Wise Sequential Detectors

They are shown in Figure 3.3.4.1. Their operation is straightforward. They are all cleared prior to each input by a clear pulse from the bit sequencing logic. The data character is broadcast to each detector in a bit serial fashion. Each detector compares the data character with

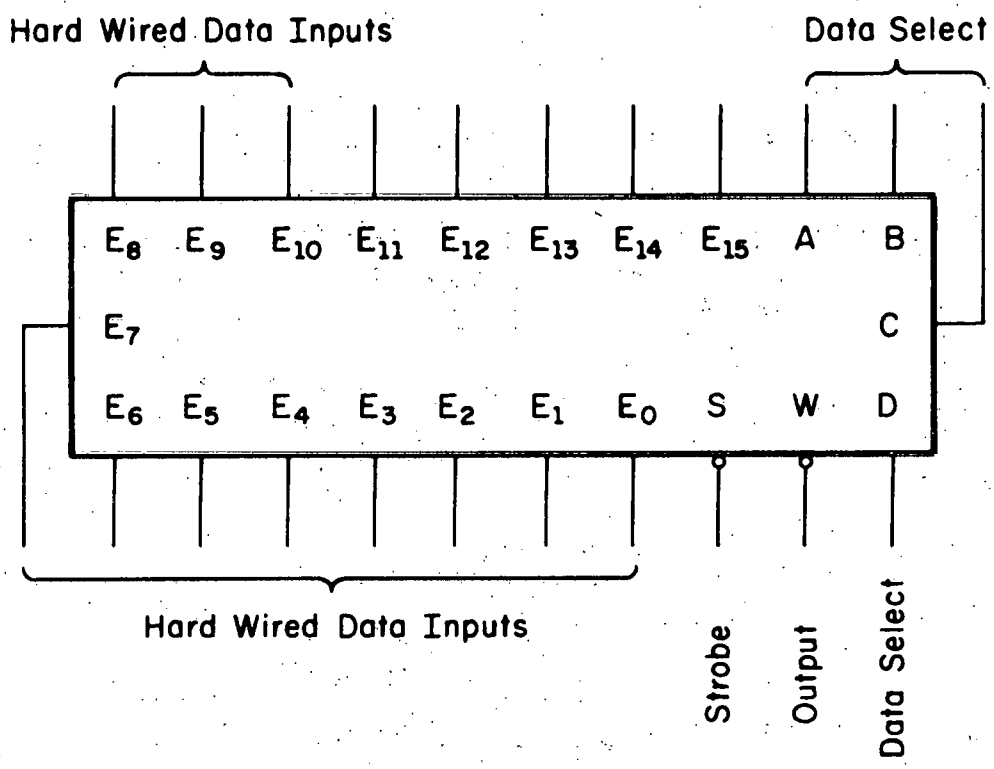


Figure 3.3.3.1 The SN74150 Multiplexor

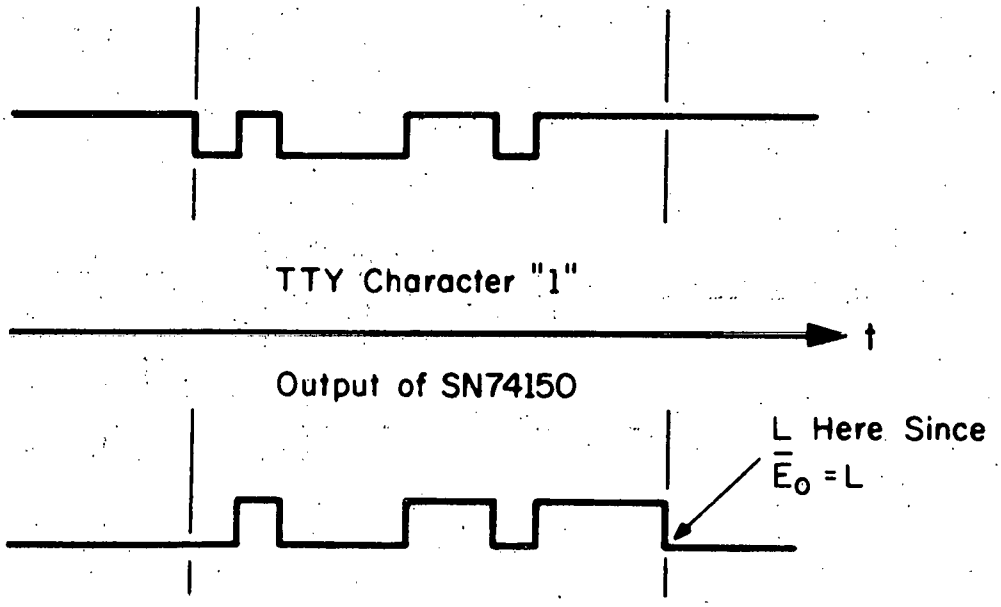


Figure 3.3.3.2 SROM Contents

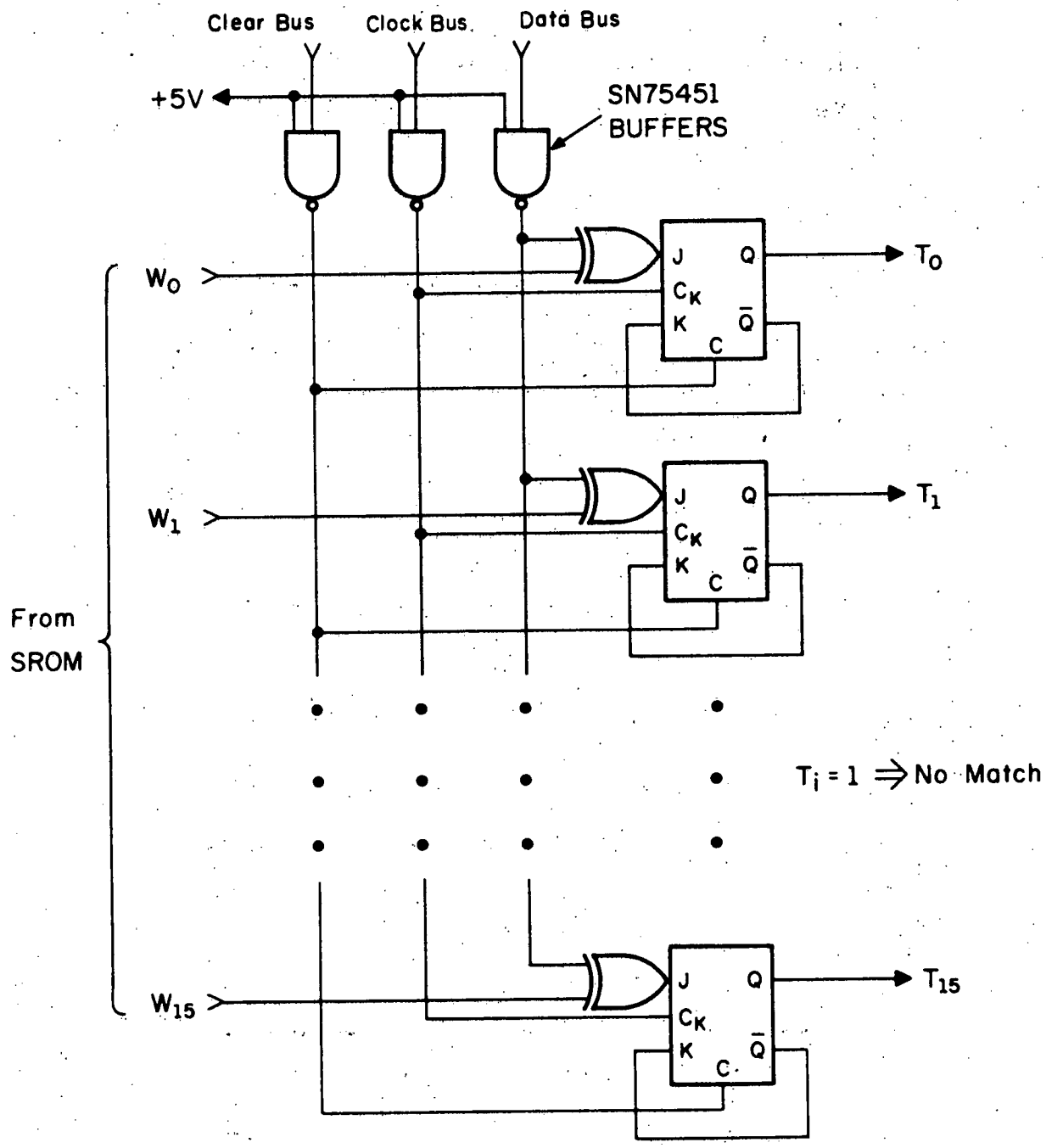


Figure 3.3.4.1 Bit-Wise Sequential Detectors

one from the SROM which is also input to the detector in a bit serial fashion. The comparison is achieved by the exclusive OR gate. A mismatch causes the FF to be set. The FF is strobed by the system clock to ensure alighment between the data bits and those from the SROM character. The point about alignment is important as will be seen from Figure 3.3.2. The time slots of the bits output from the SROM are displaced back in time with respect to those of the data character, due to the action of the bit sequencing logic. To compare bit b_i in the input data character with bit b_i in an SROM location, the comparison must be made during the clock pulse, as this is the only time the two slots overlap.

3.3.5 The Character-Wise Sequential Detectors

These are designated SM1, SM2, SM3 and SM4 as was noted previously. Their implementation is shown in Figure 3.3.5.1, 2, 3 and 4. All four operate in essentially the same way. Basically, they contain a counter which counts the occurrence of certain characters appearing as input data, only at certain times. For example, in SM1, when the counter is in state 00, only the occurrence of a character N as input data will increment the counter. The occurrence of an N is indicated by the output T_{11} of the bit-wise detectors remaining low until the count 9 pulse occurs. In states 01 and 10 only the occurrence of number characters (0 through 7) will increment the counter. Finally in state 11 only the occurrence of a Rt character increments the counter. The occurrence of a Rt character in state 11 causes a output pulse on the SM1 line concurrent with the count 9 pulse of the sequencing logic. This signifies that SM1 has detected a type 1 instruction.

If characters are input out of sequence they are ignored by the character-wise detectors; if the characters are not contained in the SROM

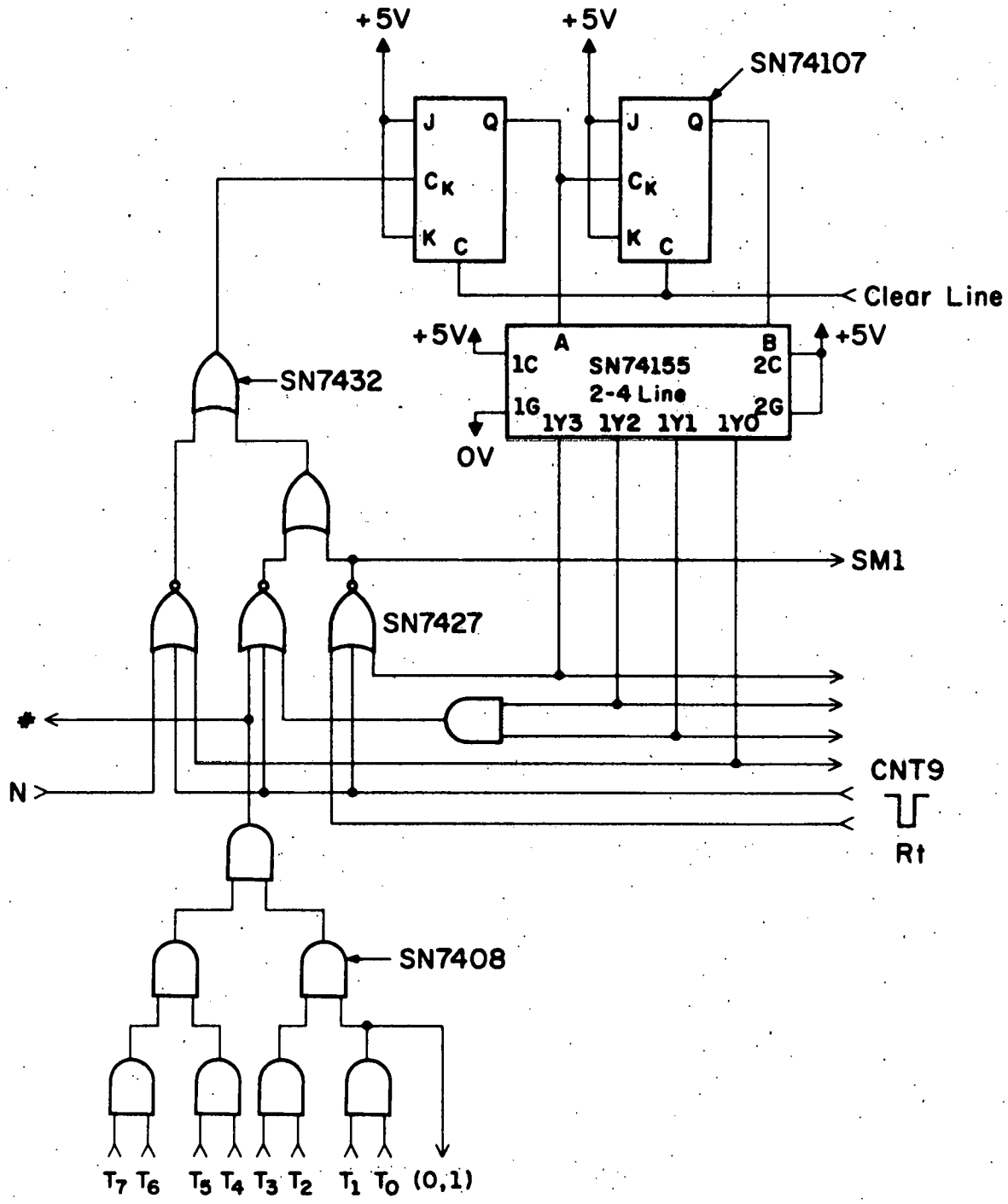


Figure 3.3.5.1 SM1

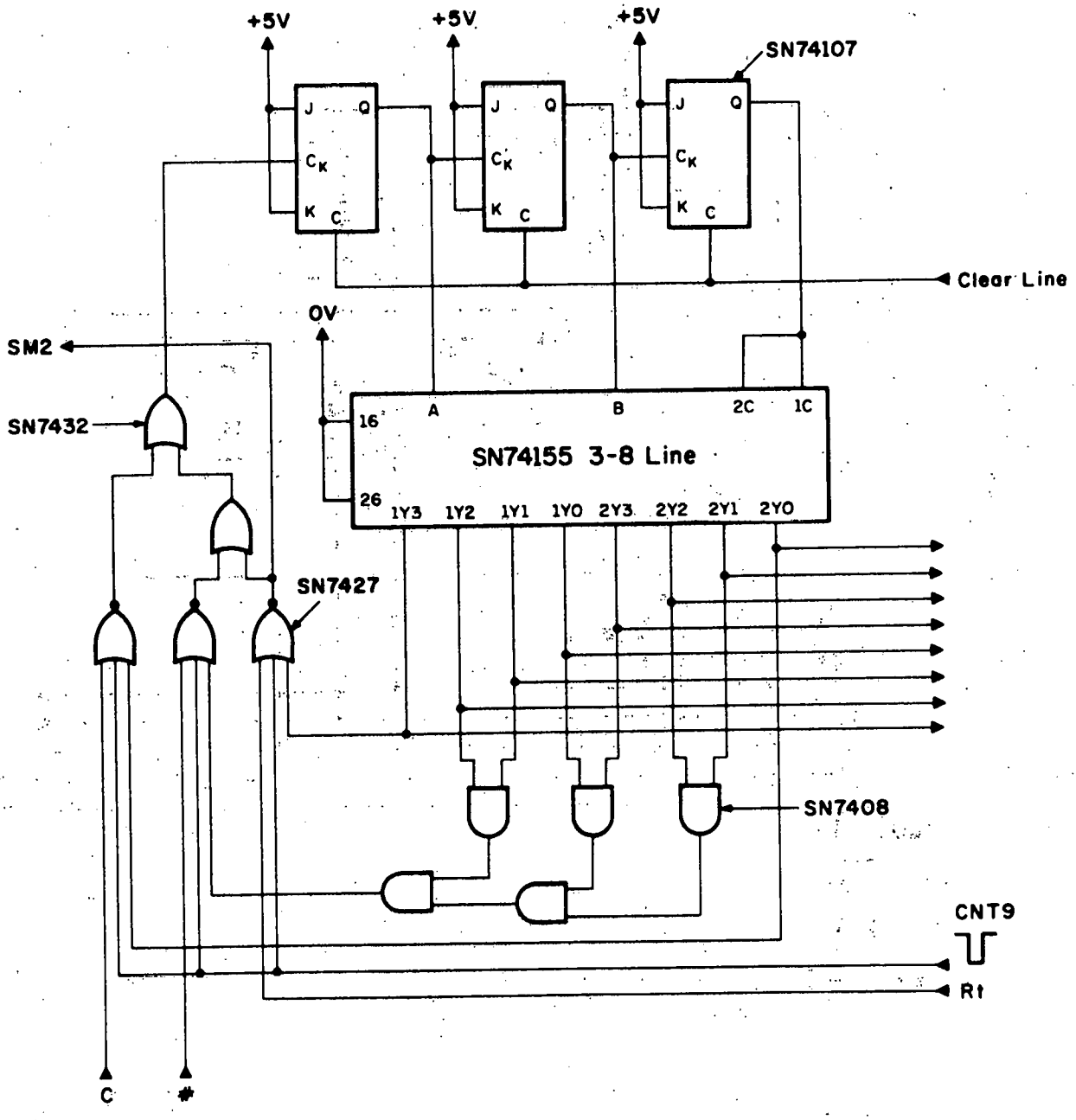


Figure 3.3.5.2 SM2

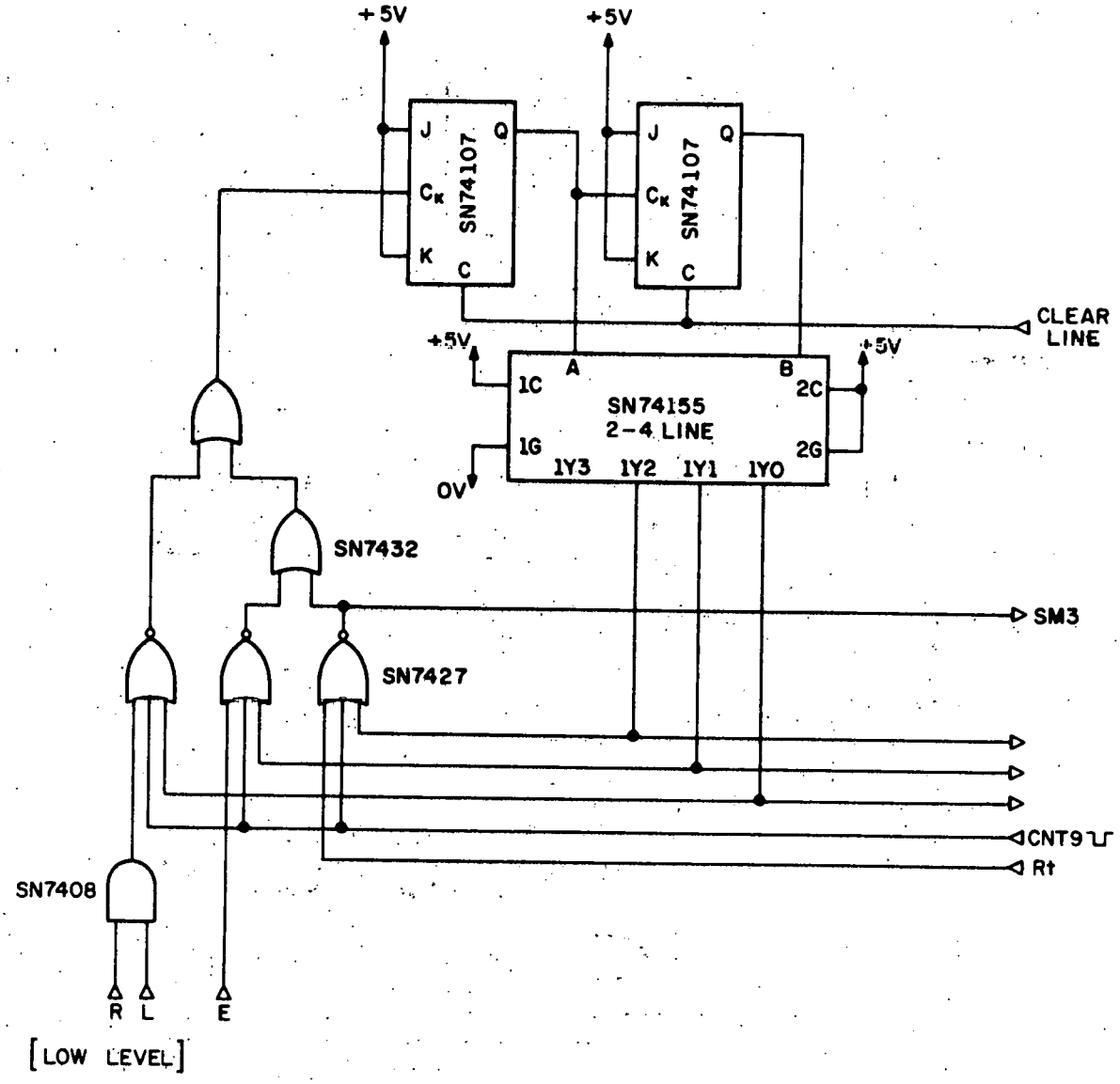


Figure 3.3.5.3 SM3

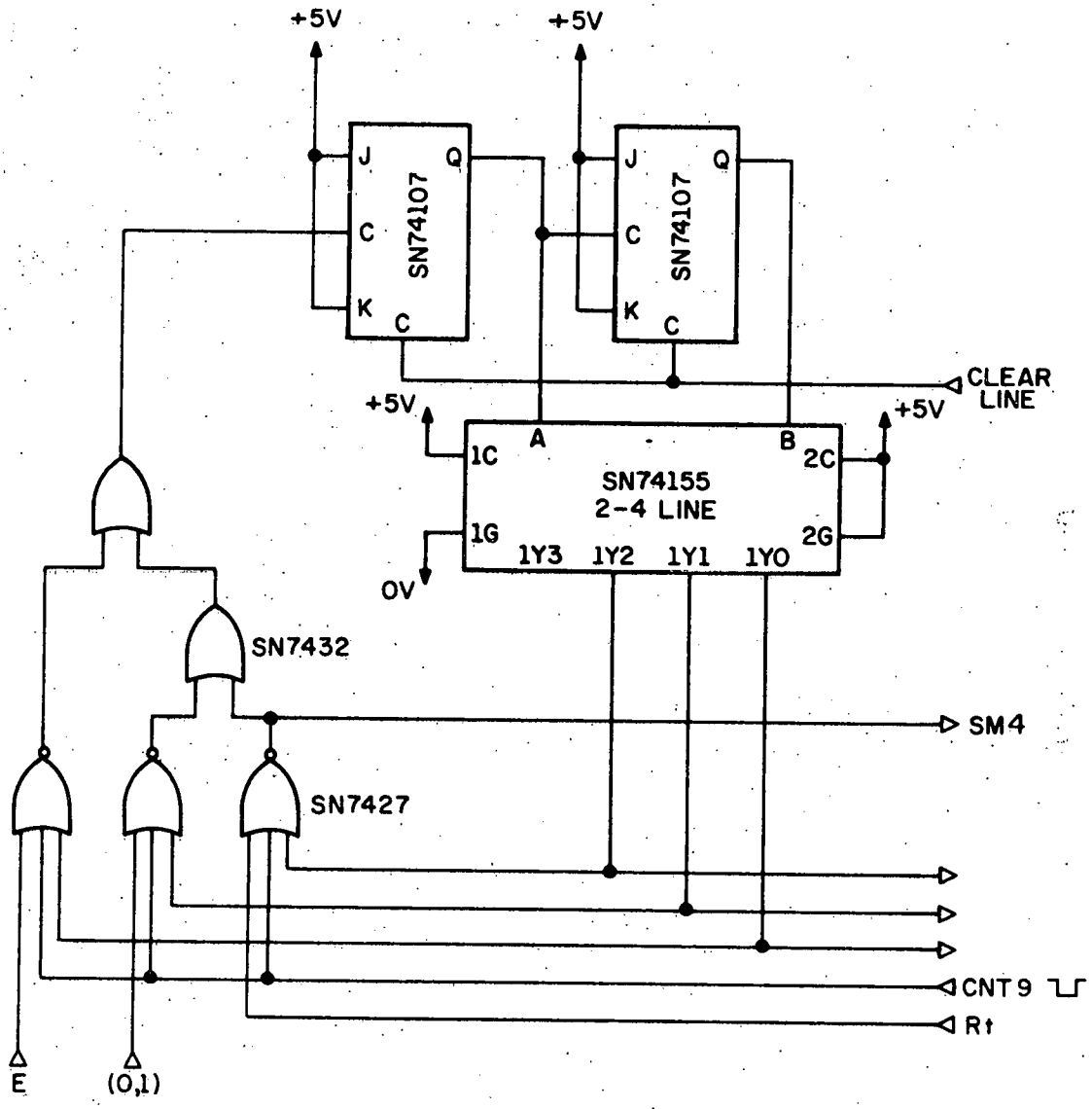


Figure 3.3.5.4 SM4

they are ignored by the bit-wise detectors. This allows the flexible input format described in Section 3.2. There is one exception and that is the nonprinting character Rt. It is assumed to terminate the input message whenever it occurs, and when it occurs the status of the character-wise detectors is examined to see if any one of them has accepted the input message. If none of them have, an error message is output.

Although the counters in the SM's are of a ripple type, unwanted transient cycles are not a problem, because the status of the detector is only interrogated during a count 9 pulse. There is more than enough time for the detector to reach a stable state between the occurrence of consecutive count 9 pulses.

3.3.6 The Error Routine Actuator

This is a combinatorial logic circuit which detects the occurrence of a syntax error in the output data message. As was seen in the previous section, the occurrence of a Rt character should cause the examination of the character-wise detectors. Hence, there should be an error situation detected if

$$\text{Rt} \wedge \text{Count } 9 \wedge (\overline{3}_1 \wedge \overline{7}_2 \wedge \overline{2}_3 \wedge \overline{2}_4) \qquad (1)$$

is true. Where M_n means the M-th state of the detector SMn. The above logical statement reads: detect an error situation if SM1 is not in state 11, and SM2 is not in state 111, and SM3 is not in state 10, and SM4 is not in state 10 when the 9th bit of the Rt character occurs.

A further condition which must cause an error situation to be detected is if two or more of SM1, SM2, SM3, and SM4 are found to have accepted

an input data message upon the occurrence of the Rt character. Let the presence of a pulse on the SM1 line, SM2 line, SM3 line, and SM4 line be denoted by w, x, y, and z respectively. Then the above condition can be express by the Boolean expression

$$\overline{\bar{w}\bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + \bar{w}x\bar{y}\bar{z} + w\bar{x}\bar{y}\bar{z}} \qquad (2)$$

Notice that the pulses will only occur concurrently with the 9th bit of a Rt character.

Thus the combinatorial logic circuit which actuates the error routine must output a pulse iff conditions 1 OR 2 are true. Figure 3.3.6.1(a) shows the logic that achieves this.

In Figure 3.3.6.1(b) is the logic necessary to clear the character-wise detectors after one of them has accepted an input message, or an error has been detected.

3.3.7 MA and Mode Logic

Before going on to discuss the channels that are slaved to the character-wise detector it is necessary to explain the operation of memory addressing as it applies to the SROM, and the action of the mode controller.

The MA and Mode Logic is shown in Figure 3.3.7.1. The S_i are the signals which inhibit the output of the SROM. The two SN7440 buffer gates form the mode FF. When the FF is in the receive mode point A is low; hence all the S_i are brought low. The outputs of the SROM are consequently enabled and the system is ready to perform bit-wise analysis upon any input data. The output line (OP) is also low, which means that transmissions from the control logic along the bilateral data channel are inhibited. (See Figure 3.3.7.2, the interface gating.)

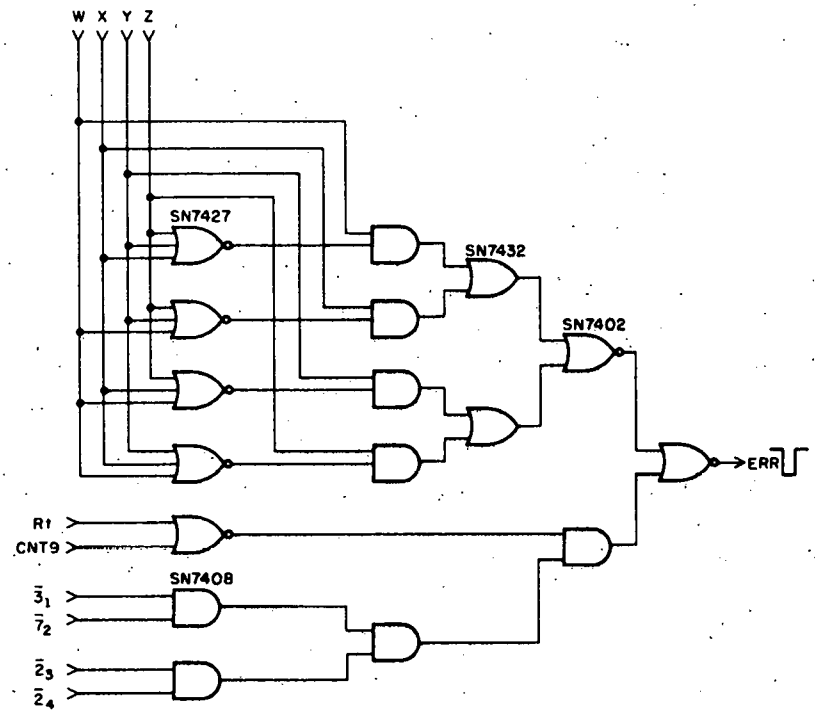


Figure 3.3.6.1 (a) Error Routine Actuator

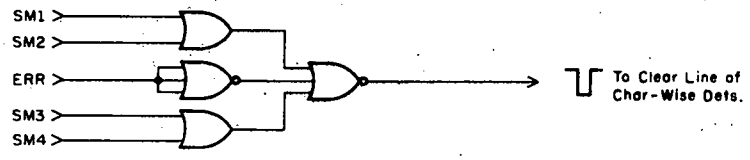


Figure 3.3.6.1 (b) Character-Wise Clear Generator

A negative pulse on any one of Tx_1 through Tx_6 sets the FF into transmit mode. The S_i are brought high, inhibiting the SROM outputs, and the line (OP) is brought high, enabling the output of the interface gating. The line x is brought low, which starts the clock be setting the sync FF in the bit sequencing logic (see Figure 3.3.2.1). A cycle of eleven clock pulses is generated. The mode FF is reset by the negative count 9 pulse.

During the cycle of eleven clock pulses a character may be transmitted along the bilateral data channel. The character is selected by bringing the appropriate address line low, when the mode FF is set to the transmit mode. For example, to transmit an E, line E (see Figure 3.3.7.1) is brought low. In the case of the numbers 0 through 7, their binary representation is input into a 3-to-8 line decoder (the SN74155) and the G line is brought low.

In order to inhibit inputs to the control logic which come through the bilateral data channel during the transmission phase, a series of OR gates are connected to the input line in the interface gating (see Figure 3.3.7.2). These are controlled by the channels. When a channel is generating a reply message to be output through the interface, it sets one of the OR_i high, which inhibits the input data from the bilateral data channel.

3.3.8 Channel One

This channel is responsible for generating a specified cube's coordinates. Figure 3.3.8.1 shows the N register which accepts the two octit number specifying the cube to be located. The bits b_1, b_2 and b_3 of characters two and three of type 1 instructions are shifted into the N register. This is achieved by shifting the two 4 bit shift registers with

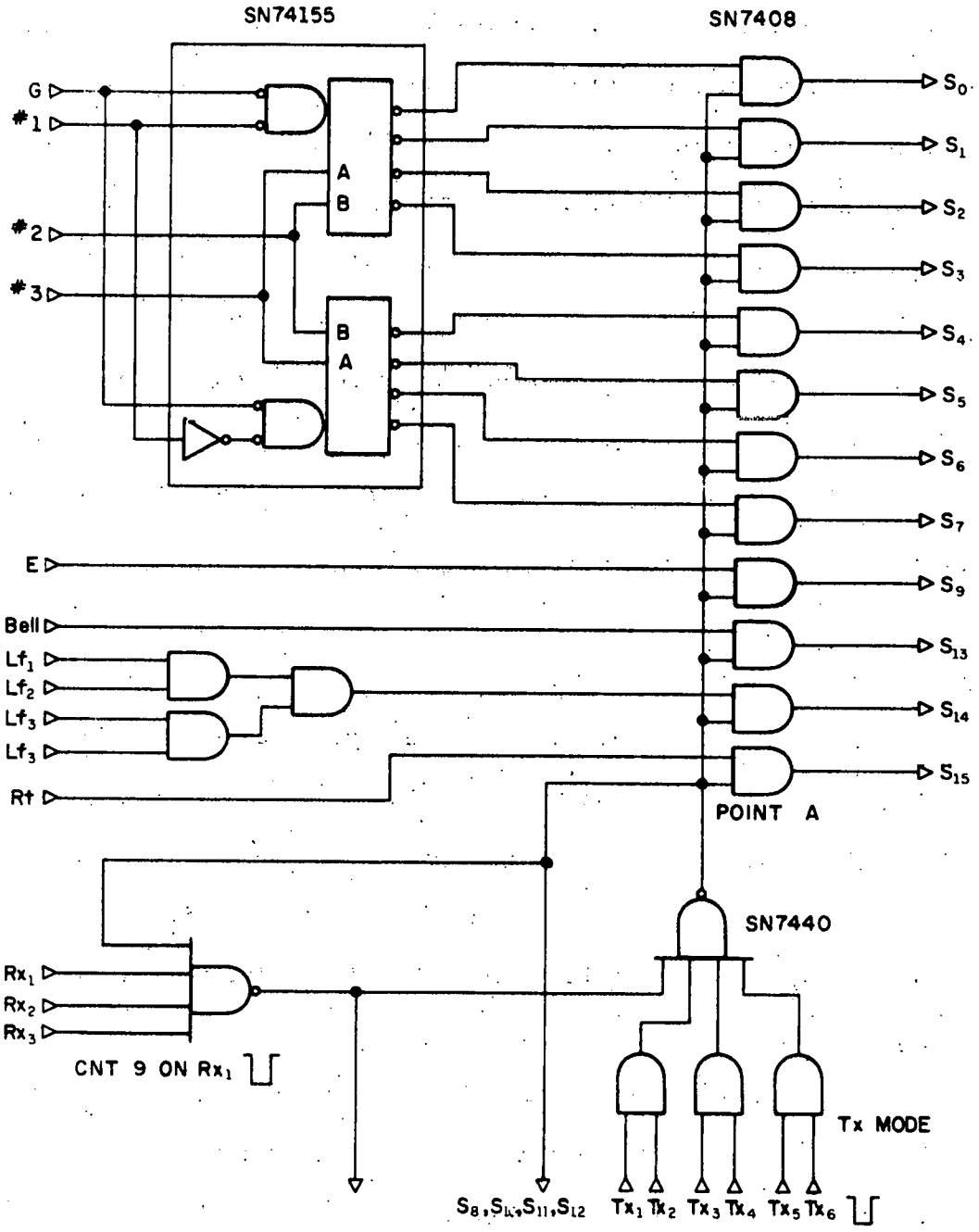


Figure 3.3.7.1 MA and Mode Logic

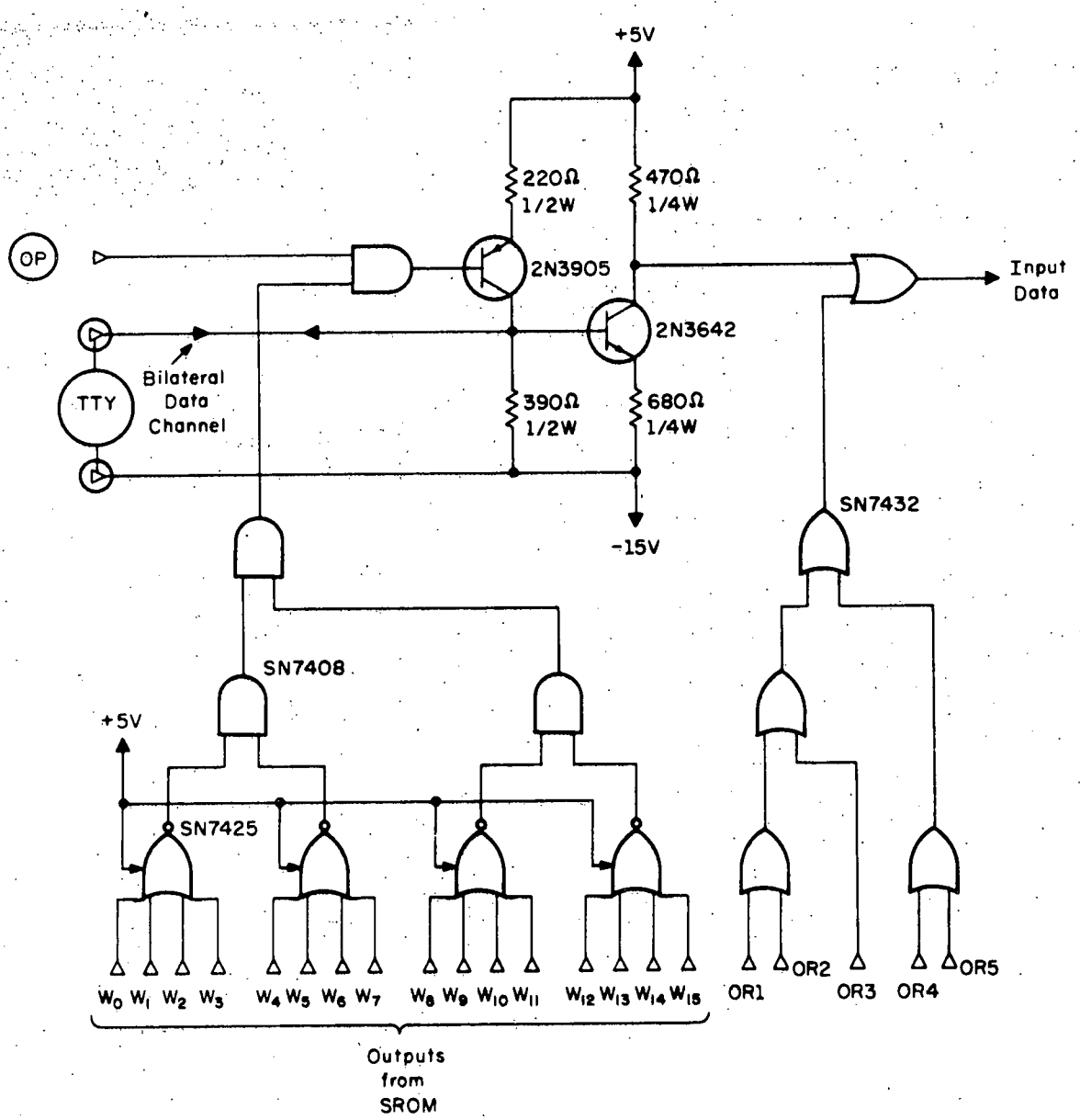


Figure 3.3.7.2 Interface Gating

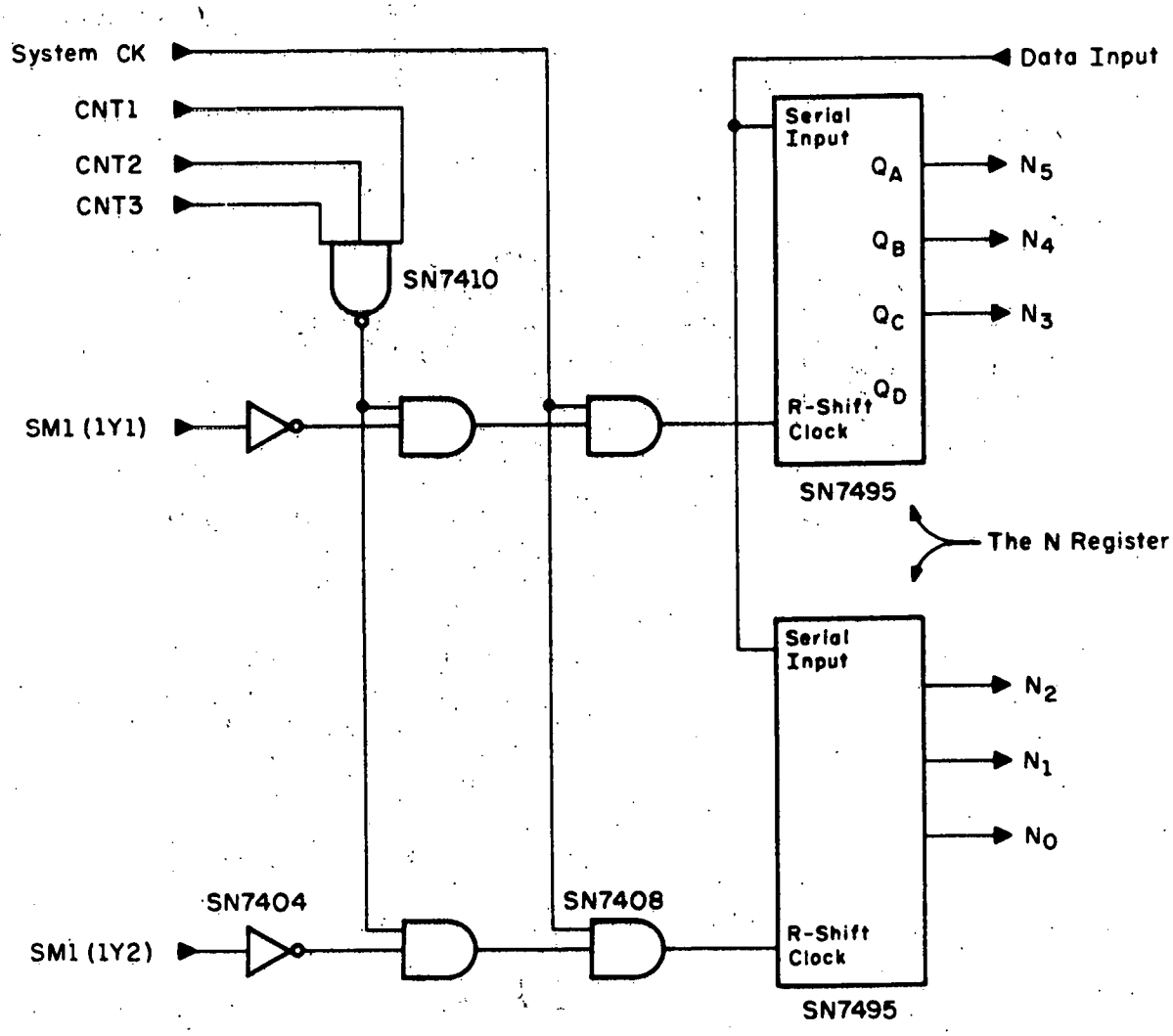


Figure 3.3.8.1 The N Register

the count 1, count 2 and count 3 pulses from the bit sequencing board. The condition that these bits come first from the second character and then from the third input character is achieved first by enabling the set of shift pulses to one of the 4 bit shift registers with the SM1 (1Y1) logic level, which is low only when the second character is input, and they by enabling the set of shift pulses to the other 4 bit shift register with the SM1 (1Y2) logic level, which is low only when the third character is input (see SM1, Section 3.3.5).

The system clock is used to strobe the input data to the N register; this is to achieve the correct alignment in time. It will be seen from Figure 3.3.2 that the time slots generated by the bit sequencing logic are displaced back in time with respect to those of the input data characters. Only during the clock pulses do they align.

Upon the receipt of a Type 1 instruction the N register will contain the six bit representation of the two octit number specifying the cube. This is so because the bits b_1, b_2 and b_3 of the TTY numeric characters 0 through 7 are the binary encoding of those characters (b_1 = LSB).

The channel flag is set by the SM1 pulse if a Type 1 instruction is detected and the power transmitter is pulsed for 1mS. The logic that accomplishes this is shown in Figure 3.3.8.2. Setting the flag also inhibits the input through the interface gating, by bringing line OR_1 high. If the instruction is not Type 1 the channel flag will not be set and thus, the contents of the N register will be ignored.

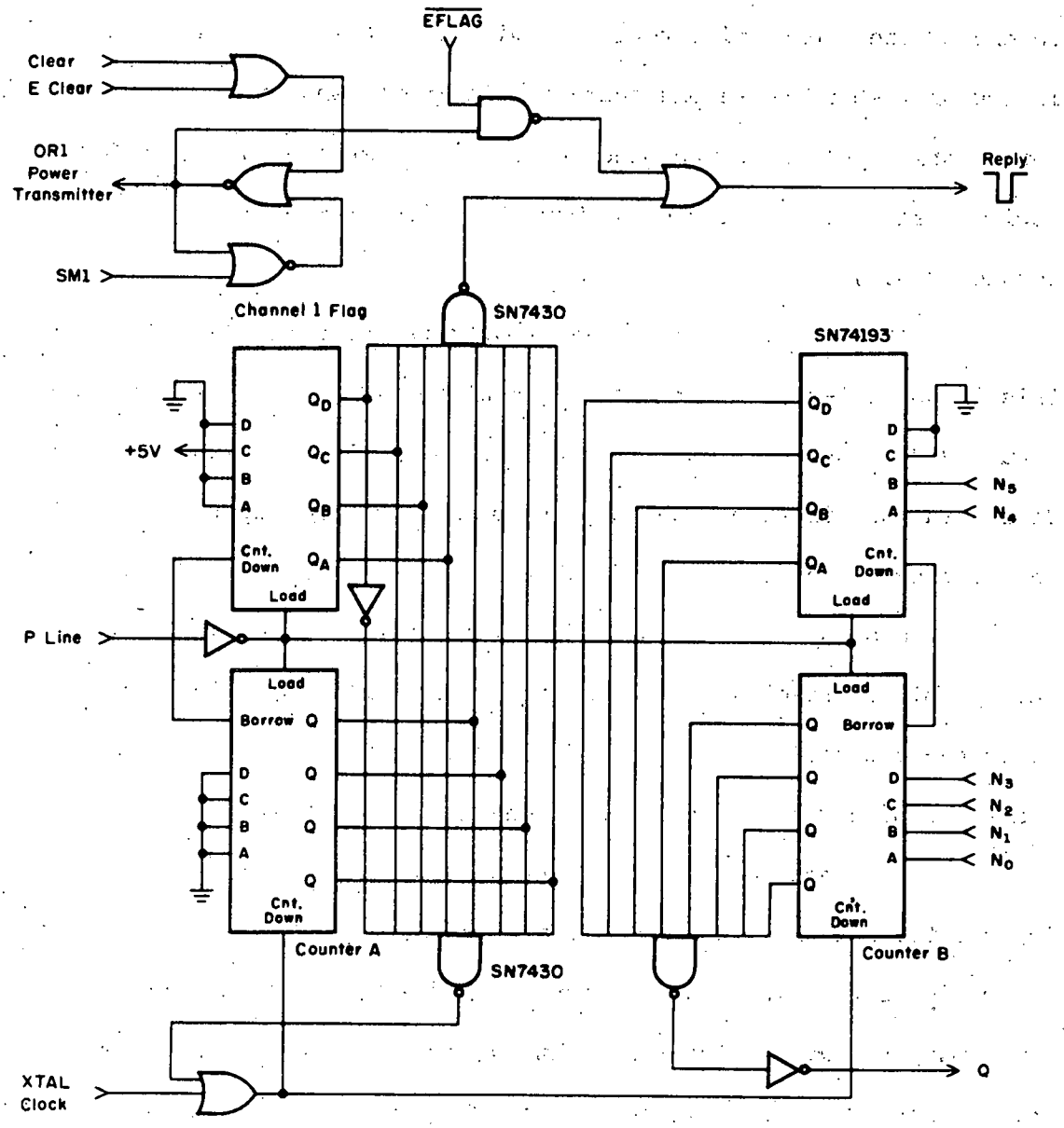


Figure 3.3.8.2 Channel One Logic

When the power transmitter has completed its transmission, the signal on the P line causes counters A and B to be loaded. Counter A is loaded with 1000000_2 and counter B with $(N_5N_4N_3N_2N_1N_0)_2$, the binary code representing the cube that is to be detected. As soon as the counters are loaded the OR gate controlling the crystal controlled clock signal is enabled and the counters begin to count down. When counter B reaches two's complement one (i.e. 11111111_2), the line Q is brought high for one period of the crystal controlled clock. This enables the coordinate buffers during the $(N_5N_4N_3N_2N_1N_0 + 1)^{-th}$ time slot after the cessation of the power transmission. A time slot is one period of the crystal controlled clock. If the N^{-th} cube is in play, it will reply during this time slot; hence, its coordinates will be entered into the coordinate buffer. If the N^{-th} cube is not in play, the coordinate buffer will contain all zeroes.

A time slot of 128μs was intended (see Figure 2.3.1); thus the crystal controlled clock must have a frequency of:

$$\frac{1000}{128}\ \text{kHz} \simeq 7.5\ \text{kHz}$$

In the Section 2.3 an analysis on the tolerance of the RC time constant used in the cubes was carried out. A bound of $\pm 0.8\%$ on RC was established. This analysis assumed the time slots were equal. This is not true. However, by using a crystal controlled clock the worst case cumulative error after 63 time slots can be kept as low as

$$\pm 64 \times 10^{-6}\ \text{time slots/°C}$$

without any difficulty. Over a 40°C operating range this represents

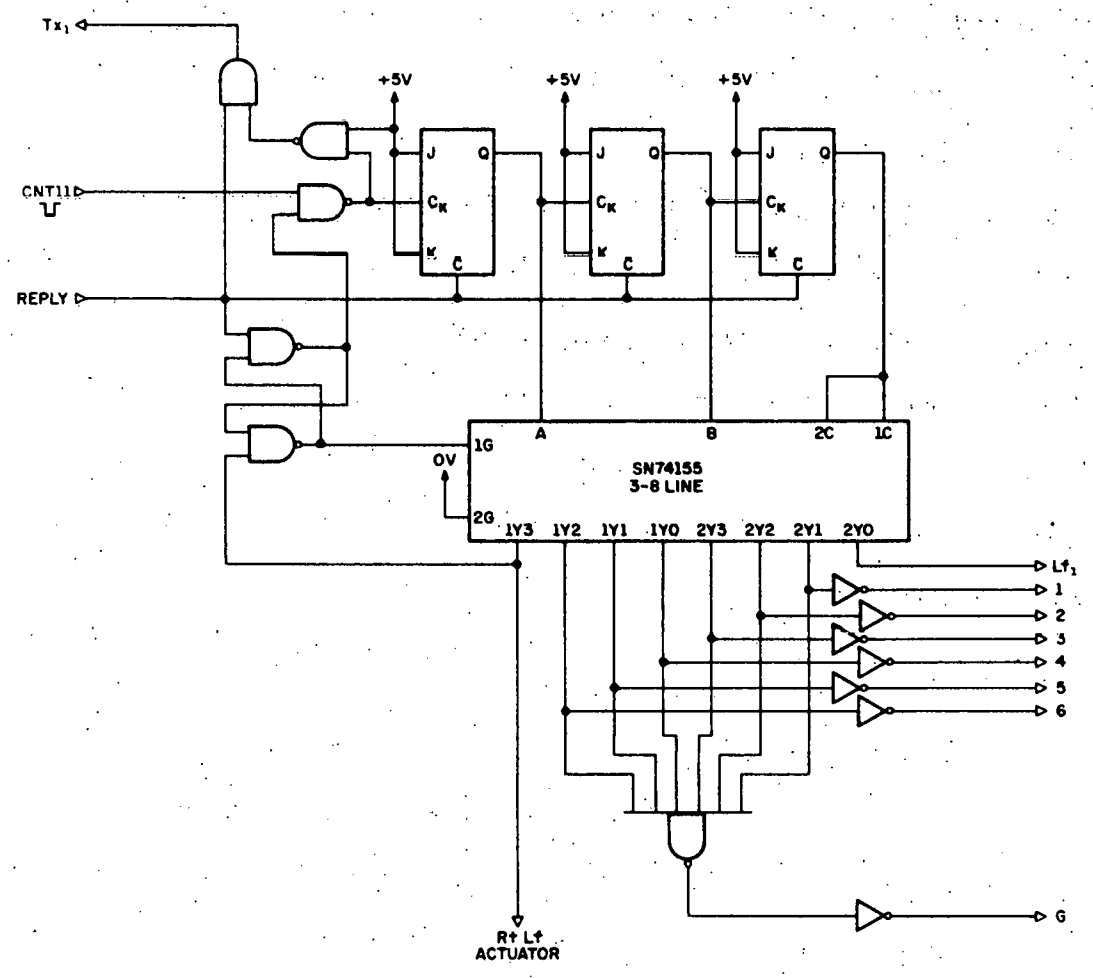


Figure 3.3.8.3 Channel One Reply

$$\pm 2.5 \times 10^{-4} \text{ time slots}$$

i.e. $\pm 0.0025\%$

This is negligible in comparison to $\pm 0.8\%$ and may be ignored in the analysis to tolerance RC.

When counter A reaches two's complement one a reply signal is generated which initiates the transmission of the reply message down the bilateral data channel. One time period later, when A reaches two's complement two, the cyrstal controlled crystal clock's signal to counters A and B is inhibited and counting ceases.

The logic to generate channel one's reply message is shown in Figure 3.3.8.3. The reply pulse clears the 3 bit ripple counter and enables the 3-8 line decoder by bringing the decoder's strobe, 1G, low. A negative pulse is simultaneously sent down the Tx_1 line. This sets the MA and Mode Logic to transmit mode, which starts the system clock on an eleven pulse cycle. Since the memory address line Lf_1 is initially low, the non-printing character Lf is output. The mode FF is reset by the count 9 pulse, and the count 11 glitch which occurs immediately after the Lf transmission increments the ripple counter, and sends a negative pulse down the Tx_1 line to set the mode FF to transmit mode once more. This time line 1 is brought low, and a 3 bit number in the coordinate buffer (see Figure 3.3.8.4) is input to the MA and Mode logic's 3-8 line decoder (see Figure 3.3.7.1). This results in the TTY numeric corresponding to the 3 bit number being output. This action is repeated five times so that all the six octits which uniquely describe the cube's position on the table top are output. The 3-8 line decoder in the MA and Mode Logic will not address the TTY numerics in the SROM unless it is

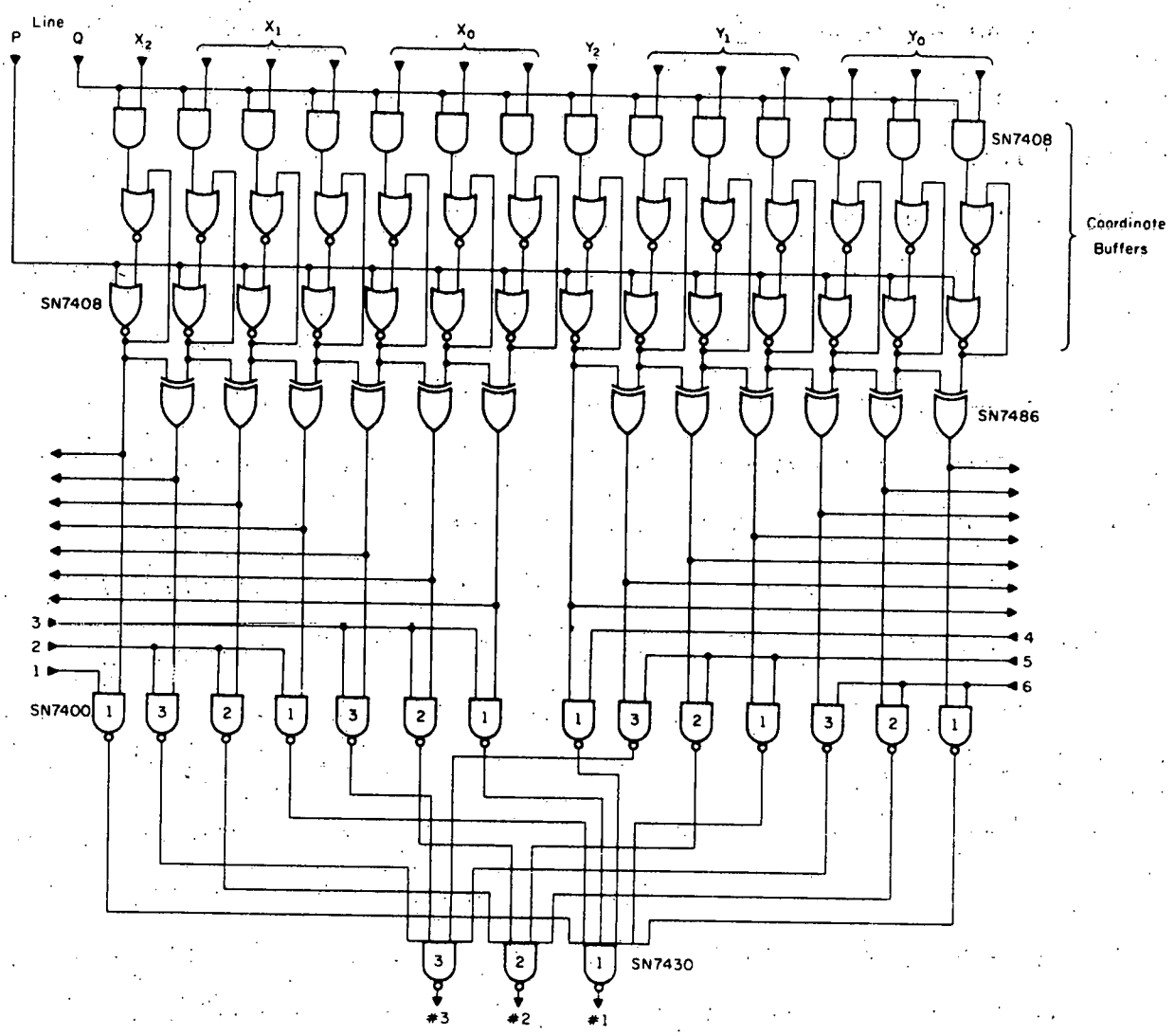


Figure 3.3.8.4 Gating Between Buffers and MA Logic

enabled by bringing line G low. During the transmission of the six octits this line is held low by the NAND decoder in the channel one reply logic.

Figure 3.3.8.4 shows the gating between the coordinate buffers and the MA and Mode Logic. The buffers contain the coordinates in Gray code (see Section 2.3). This is converted into binary by the exclusive-OR gates shown in Figure 3.3.8.4.

After outputting a cube's coordinates the channel one reply logic sets channel seven to go. Channel seven outputs the two non-printing characters Rl Lf which completes the operation of channel one. Channel seven is also responsible for clearing the channel one flag after it has transmitted Rt Lf.

The transition cycles that occur in the ripple counter used in the reply logic are not critical. They will cause incorrect addressing of characters in the SROM, but only for a brief period during the time it takes to output a character's first bit b_0. For all TTY characters $b_0 = 0$; hence, incorrect addressing during bit b_0 is unimportant.

3.3.9 Channel Two

This channel is responsible for moving the hand-arm limb to the coordinates requested by the Type 2 instructions. Figure 3.3.9.1 shows the six 4 bit shift registers that accepts bits b_1, b_2 and b_3 of each numeric in the six octit number of the Type 2 instructions. The six octit number specifies the coordinates to which the hand-arm must be moved.

The correct entry of bits b_1, b_2 and b_3 into the shift registers is achieved by shifting the 4 bit shift registers with the count 1, count 2

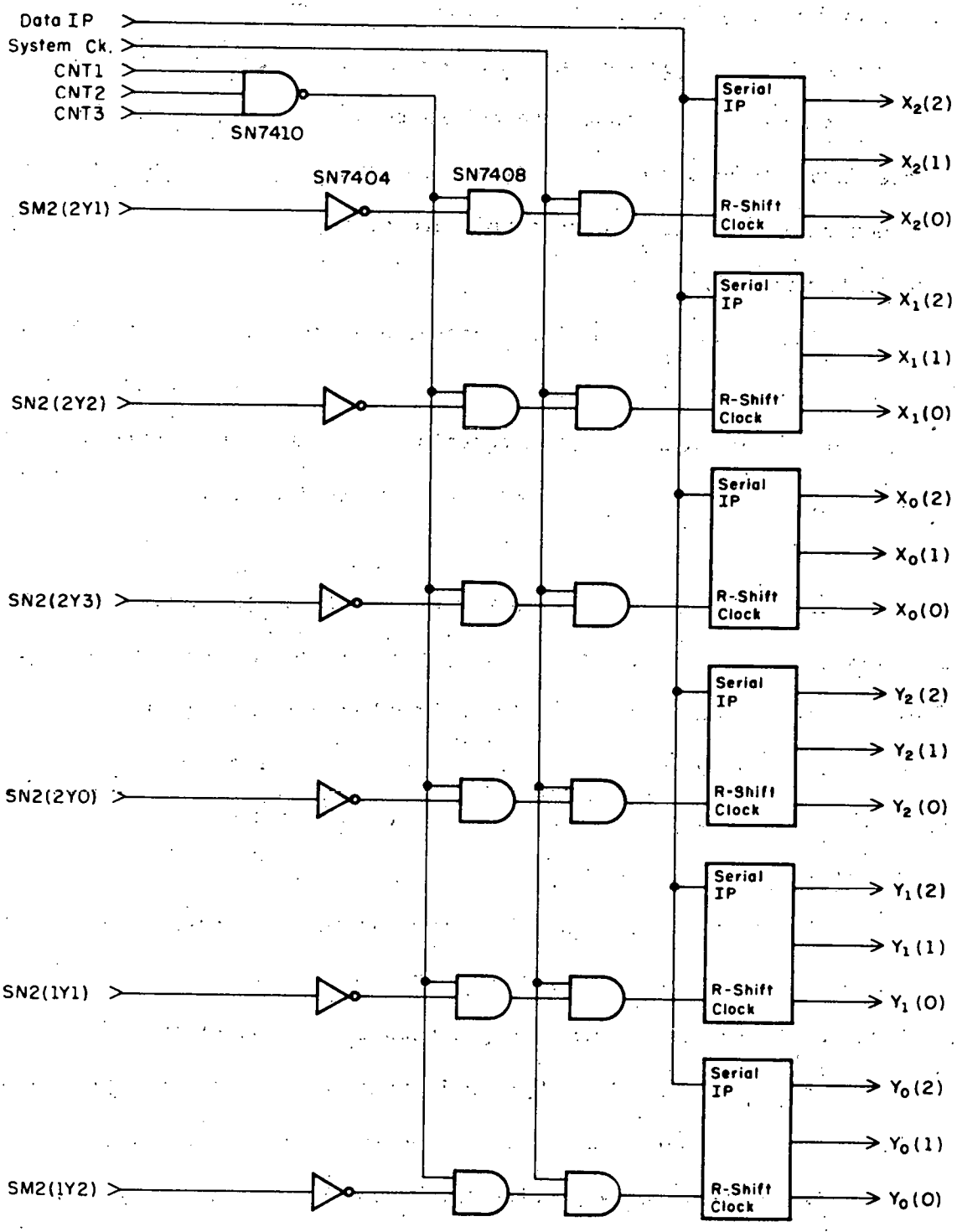


Figure 3.3.9.1 The 4 Bit Shift Registers

and count 3 pulses from the bit sequencing board. The condition that these bits come first from the second character, then from the third character, etc., is achieved first by enabling the set of shifting pulses to the first of the 4 bit shift registers with the SM2(2Y1) logic level, which is low only when the second character is input, and then by enabling the set of shift pulses to the second 4 bit shift register with the SM2(2Y2) logic level, which is low only when the third character is input, etc. (see Figure 3.3.9.1). As in channel one the system clock is used as a data alignment strobe. Should the input characters prove to be in an input sequence that is not a Type 2 instruction, the channel two flag will not be set; hence the incorrect data in the shift register will not be used.

Upon the receipt of a Type 2 instruction the six 4 bit shift registers in Figure 3.3.9.1 will contain the 18 bit representation of the six octit number specifying the point on the table top over which the hand-arm should position itself. (Each shift register contains 3 of the 18 bits in its three least significant bit positions).

The channel flag is set by the SM2 pulse if a Type 2 instruction is detected and a positive pulse SM2' is generated by the logic shown in Figure 3.3.9.2. This transfers the 18 bit coordinate description from the shift registers to the transfer buffer shown in Figure 3.3.9.3. The transfer buffer can be regarded as two 9 bit buffers, one containing the x-coordinate and the other the y-coordinate. Each of these, after D/A conversion, form the input to the hand servomechanism and to the arm servomechanism, as depicted in Figure 2.2.2.

When both the error voltages driving the servomechanisms are within one Ge diode drop of ground, the one-shot in the logic of Figure 3.3.9.2 is

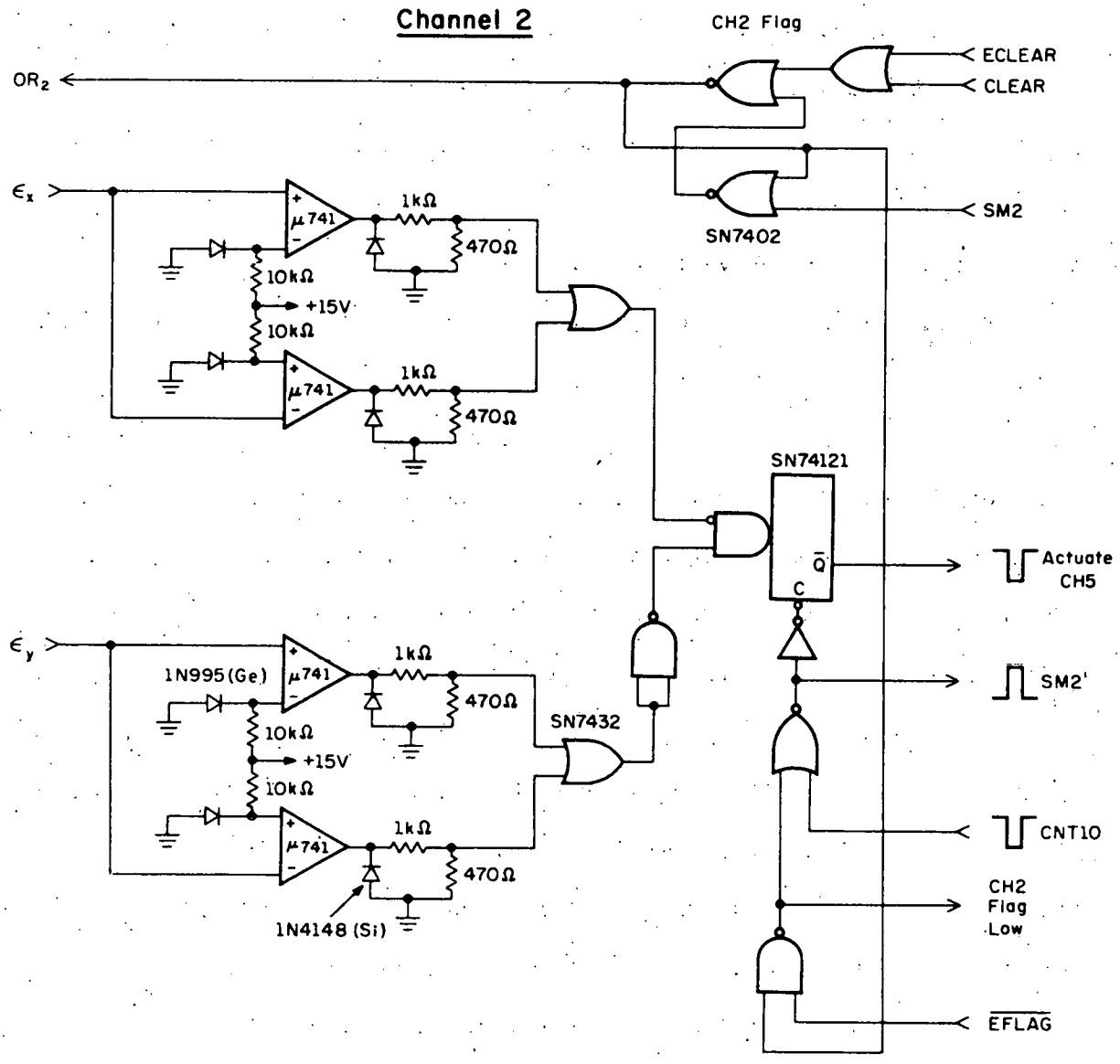


Figure 3.3.9.2 Channel Two

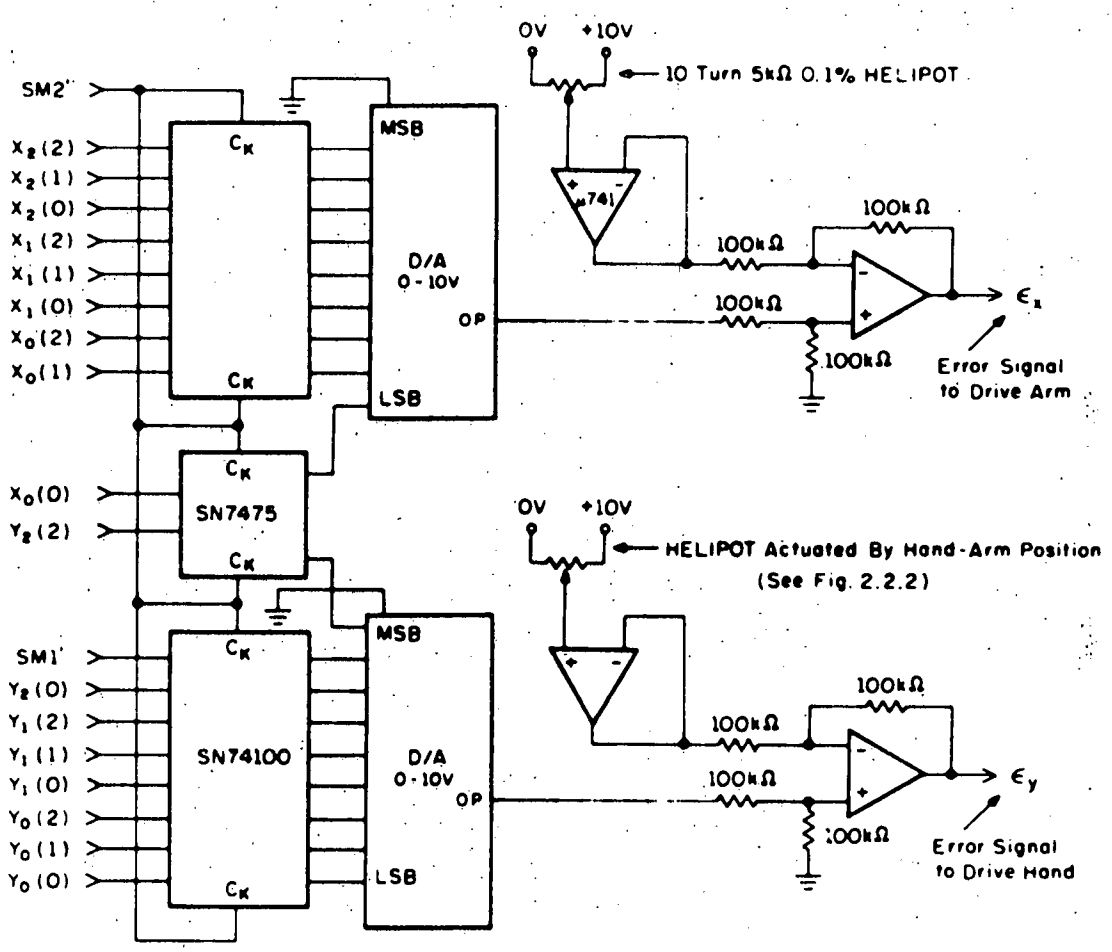


Figure 3.3.9.3 The Transfer Buffer and Servomechanism Drivers

triggered and channel five (transmit Lf Bell) is actuated, indicating the completion of operations for channel two.

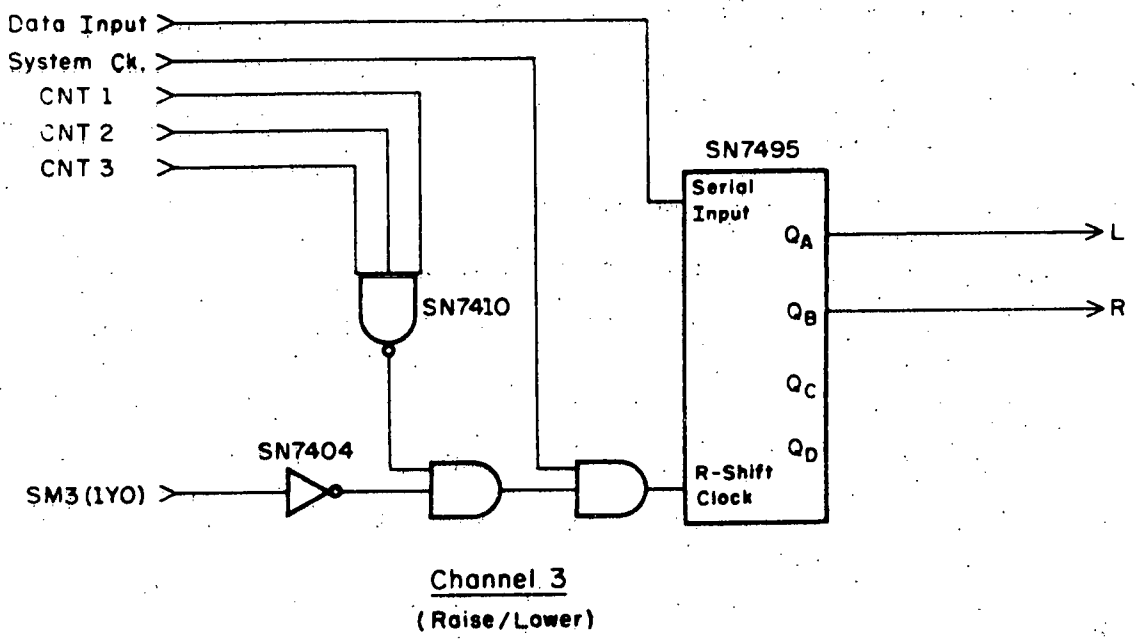
3.3.10 Channels Three and Four

The operation of channels three and four are logically similar. Channel three is responsible for the raise/lower motion of the hand, and channel four controls the open/close action.

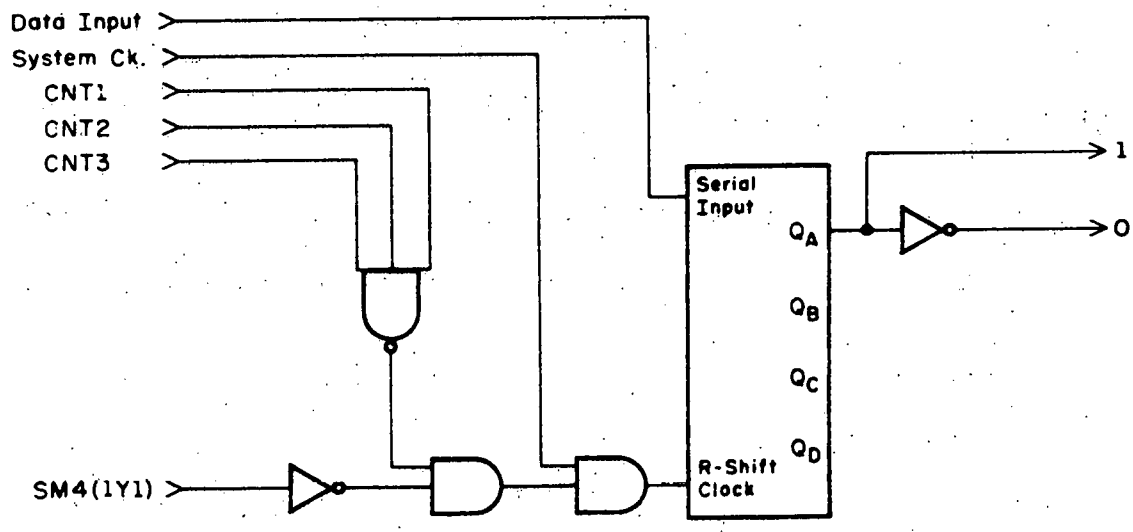
Channel 3 is shown in Figures 3.3.10.1 and 3.3.10.2. The bits b_1, b_2 and b_3 of the first accepted character are input to the 4 bit shift register. This is achieved by shifting the shift register with the count 1, count 2 and count 3 pulses from the bit sequencing board. As in channels one and two the system clock is used as a data alignment strobe. The condition that these bits come from the first input character is assured by enabling the whole set of shifting pulses with the SM3(1Y0) logic level which is low only when the first character is input (see SM3, Section 3.3.5). Should the first input character prove to be in a sequence that is not a Type 3 instruction the channel three flag will not be set (it is set by the SM3 pulse; see Figure 3.3.10.2); hence the incorrect data in the shift register will not be used. The data in the register indicates whether a raise or lower action is to be carried out. An L indicates lower and an R a raise action. The TTY characters for L and R begin as follows:

	b_0	b_1	b_2	b_3
L:	0	0	0	1
R:	0	0	1	0

These three bits are shifted into the register, with b_3 occupying the low order position. Hence a 1 bit in Q_A indicates L and a 1 bit in Q_B indicates R (see Figure 3.3.10.1).



Channel 3
(Raise/Lower)



Channel 4
(Open /Close)

Figure 3.3.10.1 Channels Three and Four

The remainder of the channel logic (see Figure 3.3.10.2) effects the action required and generates the reply message. The channel flag is set by SM3, bringing the OR_3 line high, which inhibits inputs through the interface. The count 10 pulse sets either the UP FF or the DOWN FF depending on whether an R or an L has been received. The outputs of these FF's are compared with a FF which indicates the current status of the hand (i.e. either raised or lowered). This FF is set by status micro-switches which detect the hand's position. The comparison between the desired state, as indicated by the UP or DOWN FF's, and the actual state is achieved by two AND gates, whose outputs control the raise/lower motor in the hand. When the desired state has been reached, channel 5 (output Lf Bell) is set to go, and the UP and DOWN FF's are cleared.

Channel 4 is shown in Figures 3.3.10.1 and 3.3.10.3. The operation is identical to channel 3. The input which determines whether to open or close the fingers is the second character of a Type 4 instruction. Hence, in channel 4 the shifting is only enabled when the SM4(1Y1) logic level is low (see SM4, Section 3.3.5). Receiving a "1" character indicates the fingers should be closed, and a "0" character indicates they should be opened. These two characters differ in the b_1 bits; hence, the simple decoder.

3.3.11 Channel Five

This channel causes the output of the two non-printing characters Lf Bell. It is actuated by a negative pulse from either channel 2, channel 3 or channel 4. The logic diagram for the channel is shown in Figure 3.3.11.1.

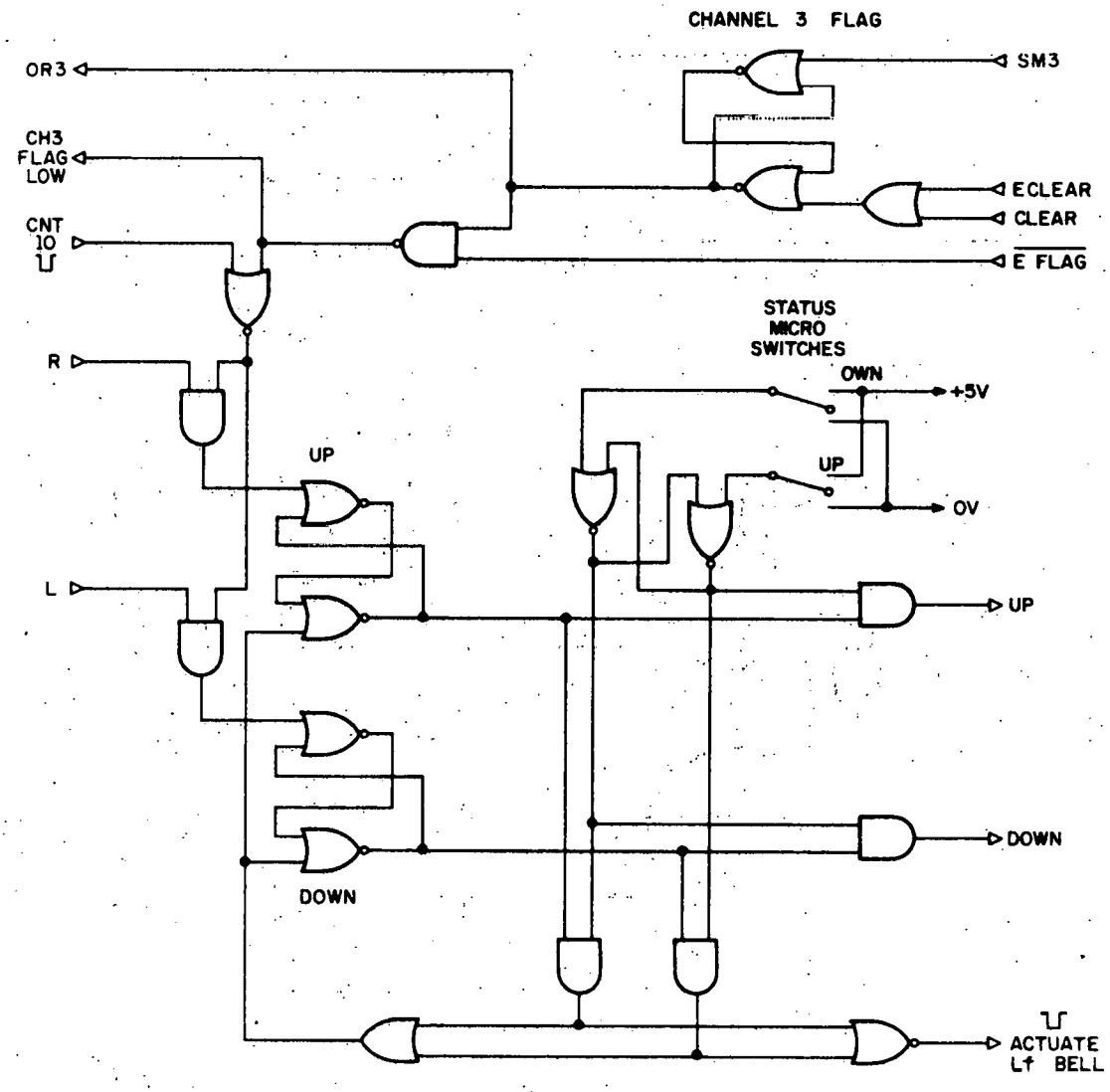


Figure 3.3.10.2 Channel Three Raise/Lower Actuator

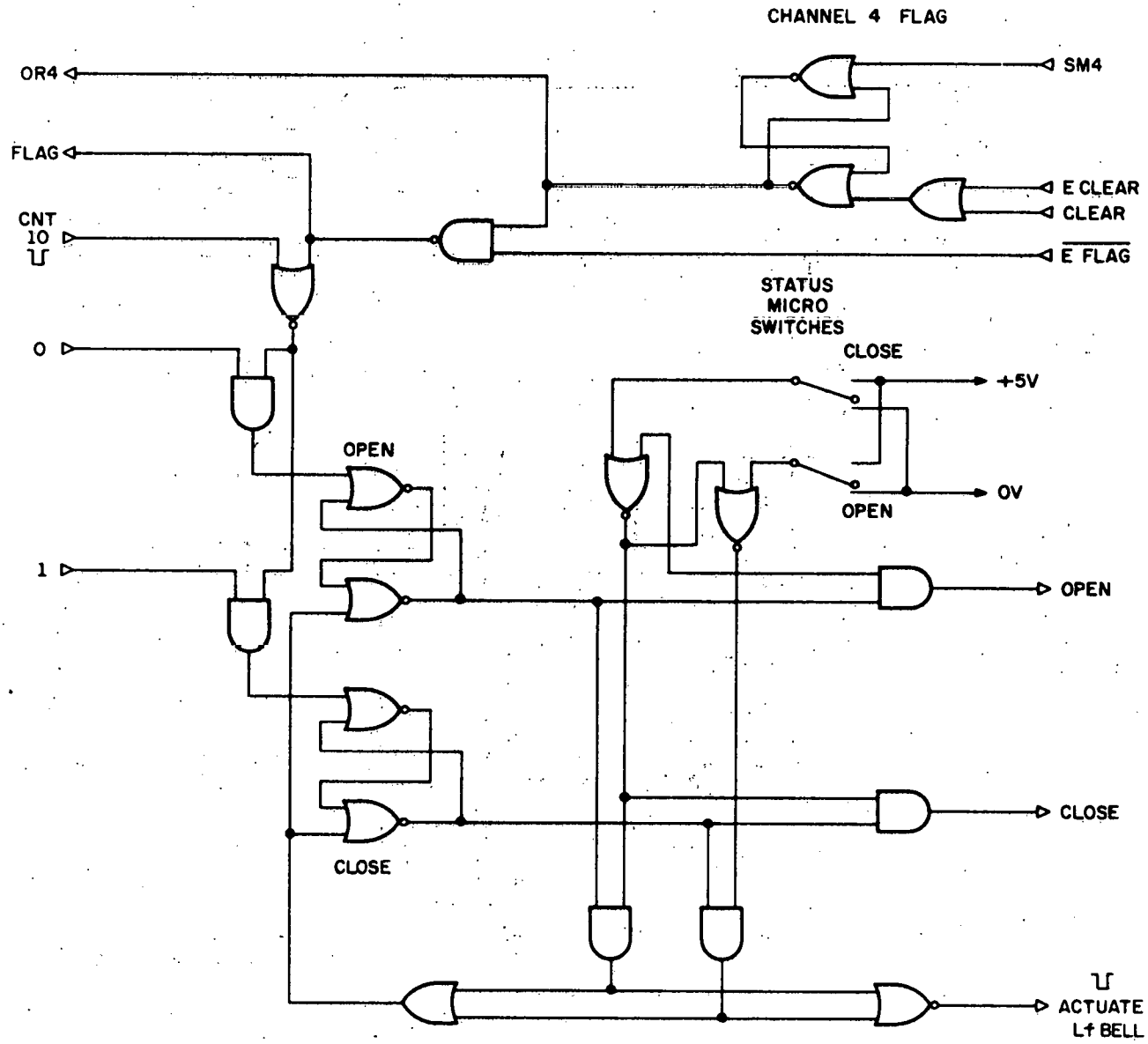


Figure 3.3.10.3 Channel Four Open/Close Actuator

The channel FF is set by the actuating pulse from channel 2, 3 or 4. The mode FF in the MA and Mode logic is set to transmit mode by a negative pulse on the Tx_2 line. The Lf_3 line is brought low and a Lf character is output. The count 11 glitch toggles the JKFF and resets the mode FF to transmit. This time the Bell line is brought low and a Bell character is output. The count 9 pulse that occurs when the sequencing logic outputs the Bell character from the SROM resets the channel FF and clears the flags for channel 2, 3 and 4. The transmission is complete.

3.3.12 Channels Six and Seven

Channel 6 is responsible for outputting the first part of the error message, Lf E, and then actuating channel 7 which is responsible for outputting the last part, Rt Lf. Channel 7 is also actuated by channel 1, which requires Rt Lf to be output to terminate its transmission.

The logic diagrams for channel 6 and 7 are shown in Figure 3.3.12.1 and 3.3.12.2. Their operation is similar to channel 5. However, channel 6 employs a slave flag which is not reset until channel 7 has completed its operations. This maintains line OR_5 high so that input through the interface gating is inhibited during the transmission of the Rt Lf characters by channel 7. None of the other OR_i will be high at this time, because channel 6 resets the channel flags for channels 1, 2, 3 and 4, by outputting a positive pulse on the ECLEAR line. This is done at the same time that it actuates channel 7. The four channel flags above control the levels of OR_1, OR_2, OR_3 and OR_4.

In the case when channel 1 actuates channel 7, OR_1 is held by channel 1 so that input through the interface gating is inhibited.

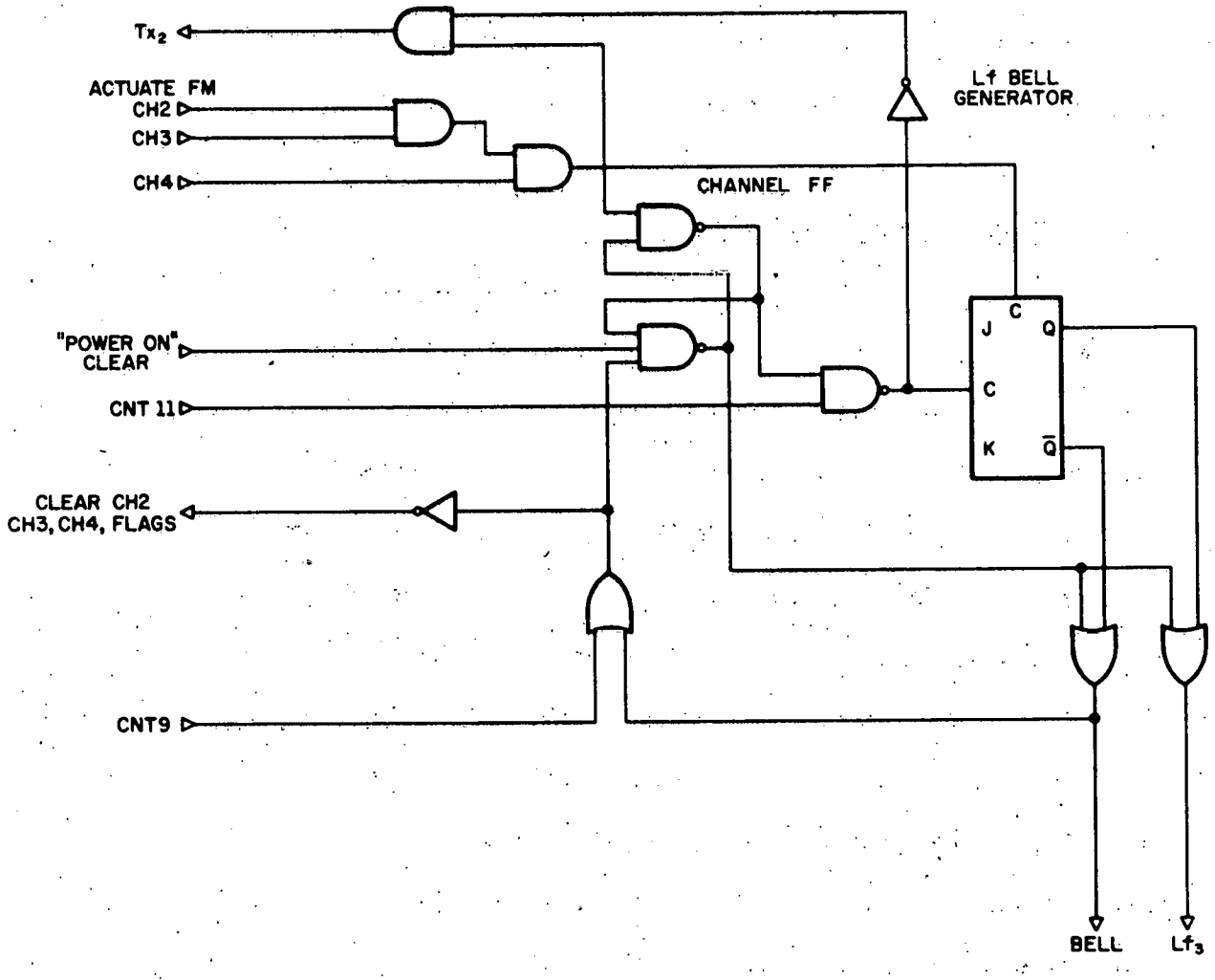


Figure 3.3.11.1 Channel Five

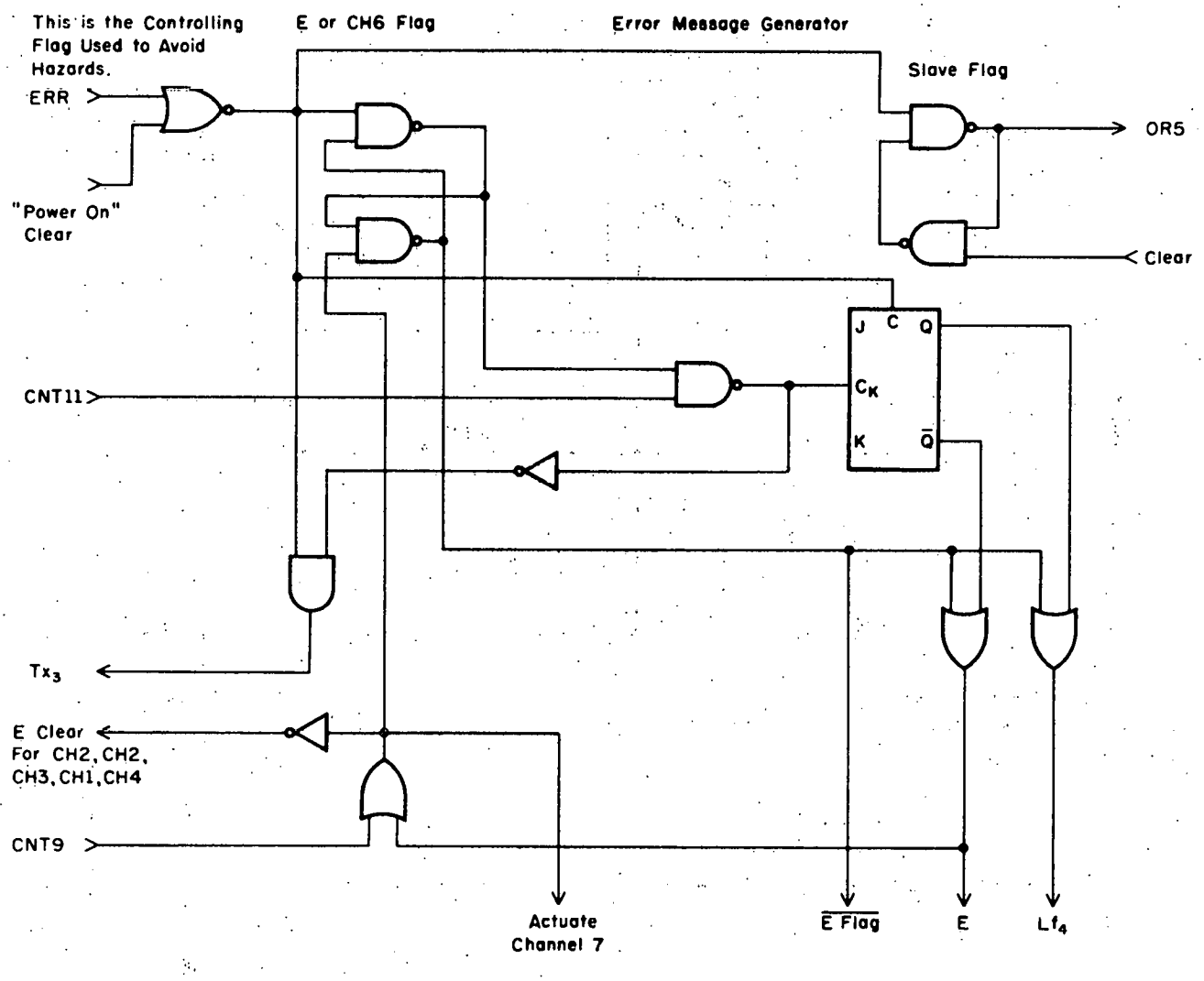


Figure 3.3.12.1 Channel 6

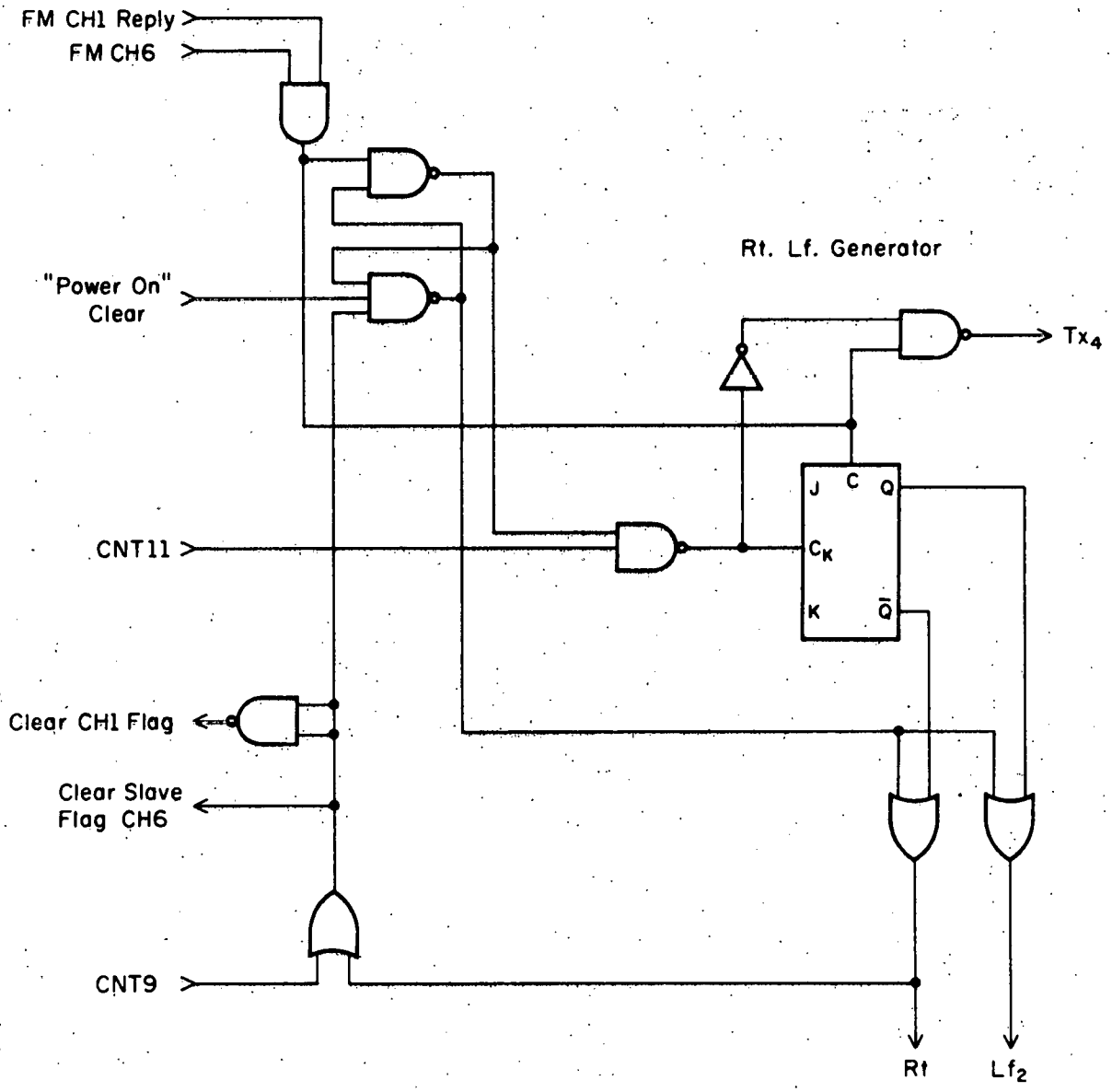


Figure 3.3.12.2 Channel Seven

Notice the $\overline{\text{EFLAG}}$ line in channel 6. When this is low the actions of all the four major channels are inhibited until the error message is transmitted, after which the ECLEAR line is pulse positive to reset the flags of channel 1, 2, 3 and 4. This is useful during a cold start. If the channel 6 flag can be forced to set when a "power on" situation occurs and the channel 5 flag and channel 7 flag can be forced to reset when a "power on" situation occurs, a cold start would always be characterized by the flags of channels 1, 2, 3 and 4 being cleared followed by an error message being output.

The generation of a "power on" state is achieved by using the circuit shown in Figure 3.3.12.3.

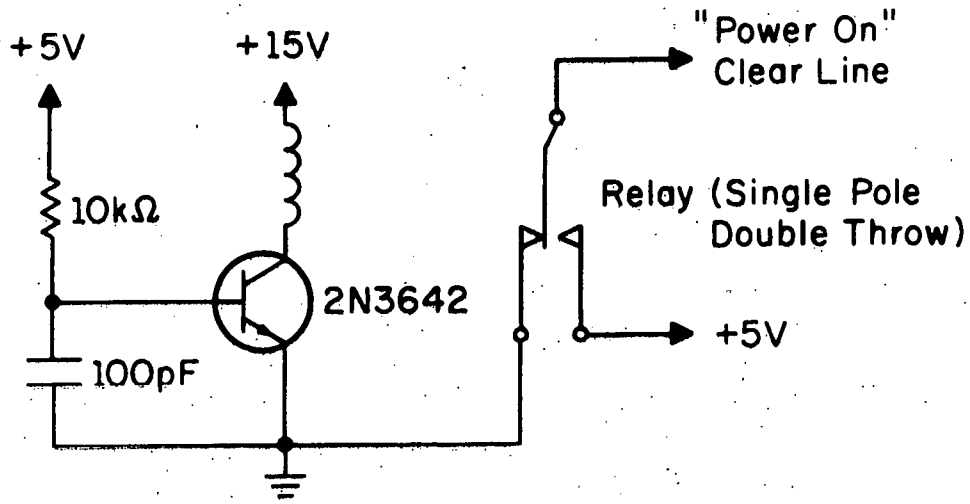


Figure 3.3.12.3 "Power On" State Generator

4. SPECIAL CIRCUITS: THEIR DESIGN AND OPERATION

In this section the design and operation of three types of special circuits used in Semantrix are discussed. They represent the only non-digital circuitry in the machine, and are all employed in cube location. In order of presentation they are:

1. The threshold circuits.
2. The cube's receiver/transmitter.
3. The power transmitter.

4.1 The Threshold Circuits

There are two arrays of 89 threshold circuits in Semantrix. One array detects the x-coordinate, the other the y-coordinate (see Figure 2.3.1). Each circuit detects a differential voltage induced across the loop that it services. The circuit schematic is shown in Figure 4.1.1.

A differential voltage pulse of a few mV at the inputs appears at the output amplified 1000x. However, an a.c. path between points A and B enables the output pulse to regenerate, since it is fed into the + input of the op-amp, resulting in further amplification until the op-amp saturates. The diode clips the negative going part of the regenerating pulse, giving a "square-up" 0V-15v pulse on the output. The diode also performs the thresholding action. For regeneration to occur the differential voltage pulse, amplified 1000x, must forward bias the diode. This can only be accomplished if the amplified pulse is more positive than the d.c. voltage at B. A bias network can be adjusted to time the threshold bias on all the circuits simultaneously (see Figure 4.1.2). It should be adjusted to reject the ambient noise input pulses which are less than 5mV.

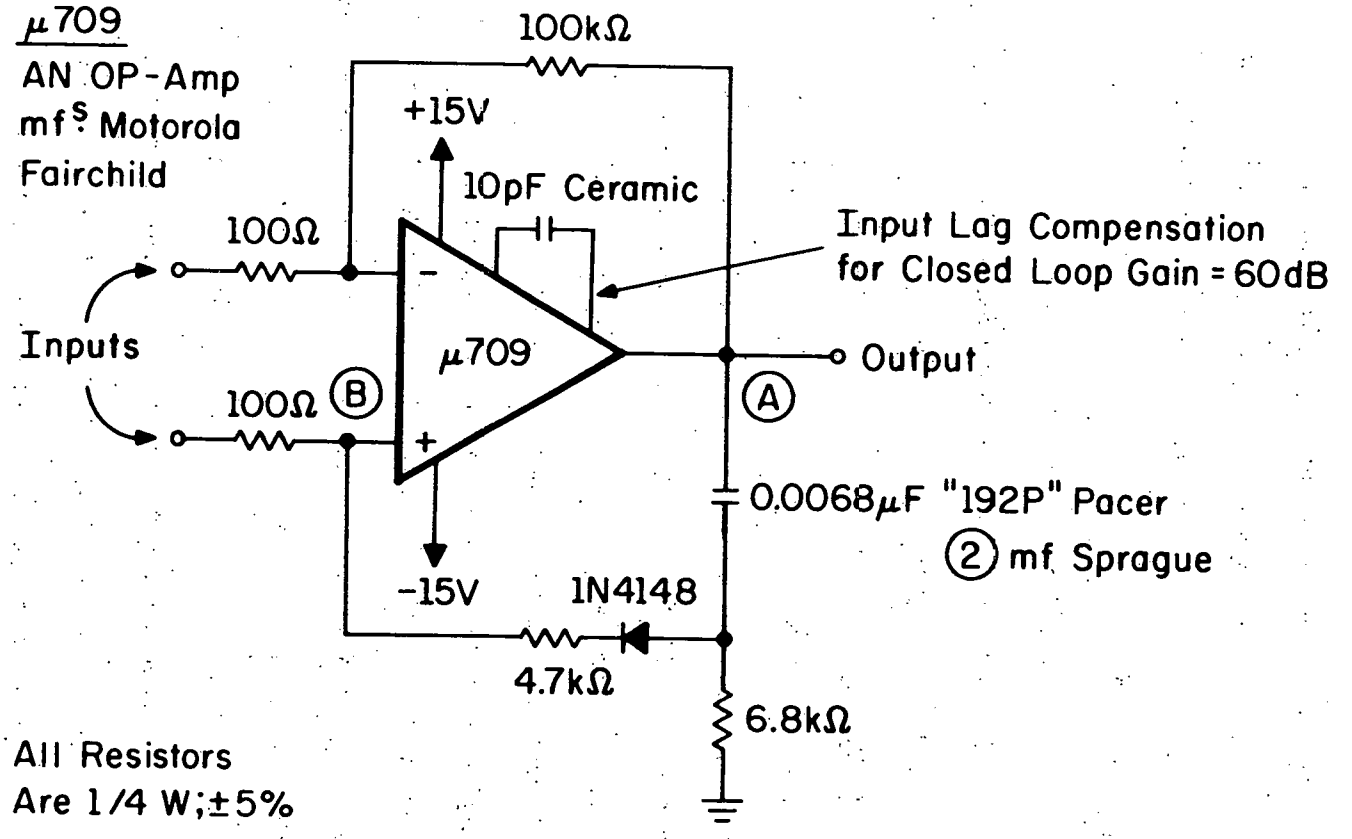


Figure 4.1.1 Schematic of a Threshold Circuit

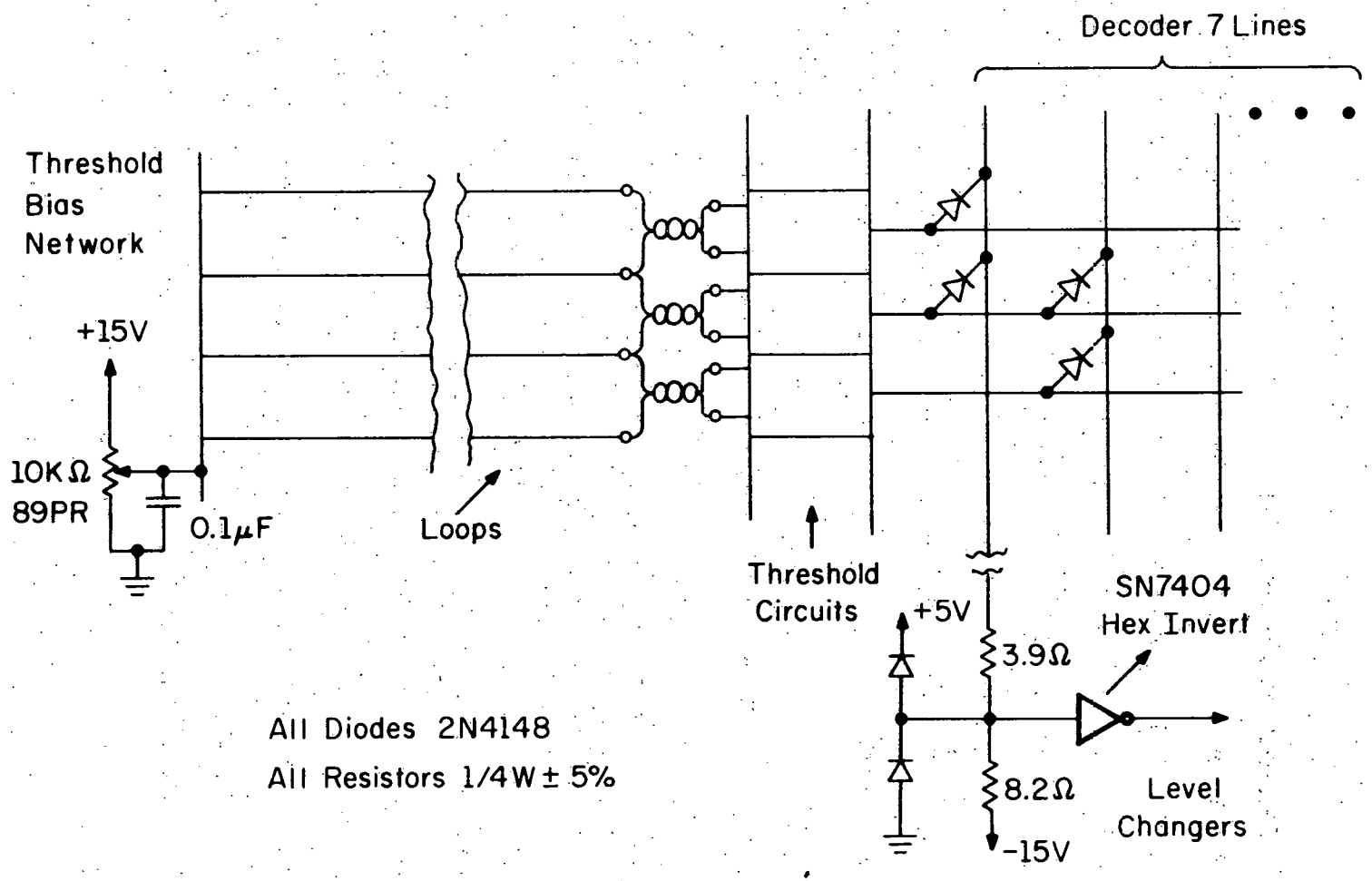


Figure 4.1.2 Arrangement of the Detectors and Decoders

Figure 4.1.2 also shows the level changers that convert the 15v pulse to one suitable for driving TTL logic loads.

4.2 The Cubes' Receiver/Transmitter

A cube receives the energy from the power transmitter, through its receiving coil. This energy is stored electrically in two 0.047μF capacitors (see Figure 4.2.1). The cube's transmitter operates by utilizing this stored energy. The manner in which energy can be obtained from the store is regulated by a 1N962B Zener diode.

The cube transmits a pulse of magnetic energy by releasing the charge stored in the two 0.047μF capacitors through a coil L_2 (the transmitting coil). The release of the charge is controlled by two transistors, Q2 and Q3, which function together like an SCR. When the point A is brought near enough to ground Q2 turns on, which turns on Q3. Turning Q3 on causes Q2 to be turned on harder i.e. a regenerative process is established. This gives a very rapid turn-on, which enhances the di/dt through the transmitting coil, and hence the differential voltage input to the threshold detectors. The SCR formed by Q2 and Q3 is fired when Q1 turns on. This is achieved in a controlled fashion by the RC charging network. Figure 4.22 shows the exponential charging ramp for the RC network. The RC charging network does not begin charging positive until the power transmitter has stopped. This is due to the action of the three diodes connecting points B and C. These form a short circuit when the induced voltage at C swings negative, holding one side of the capacitor at A to -(V-3d) volts until the power transmitter stops

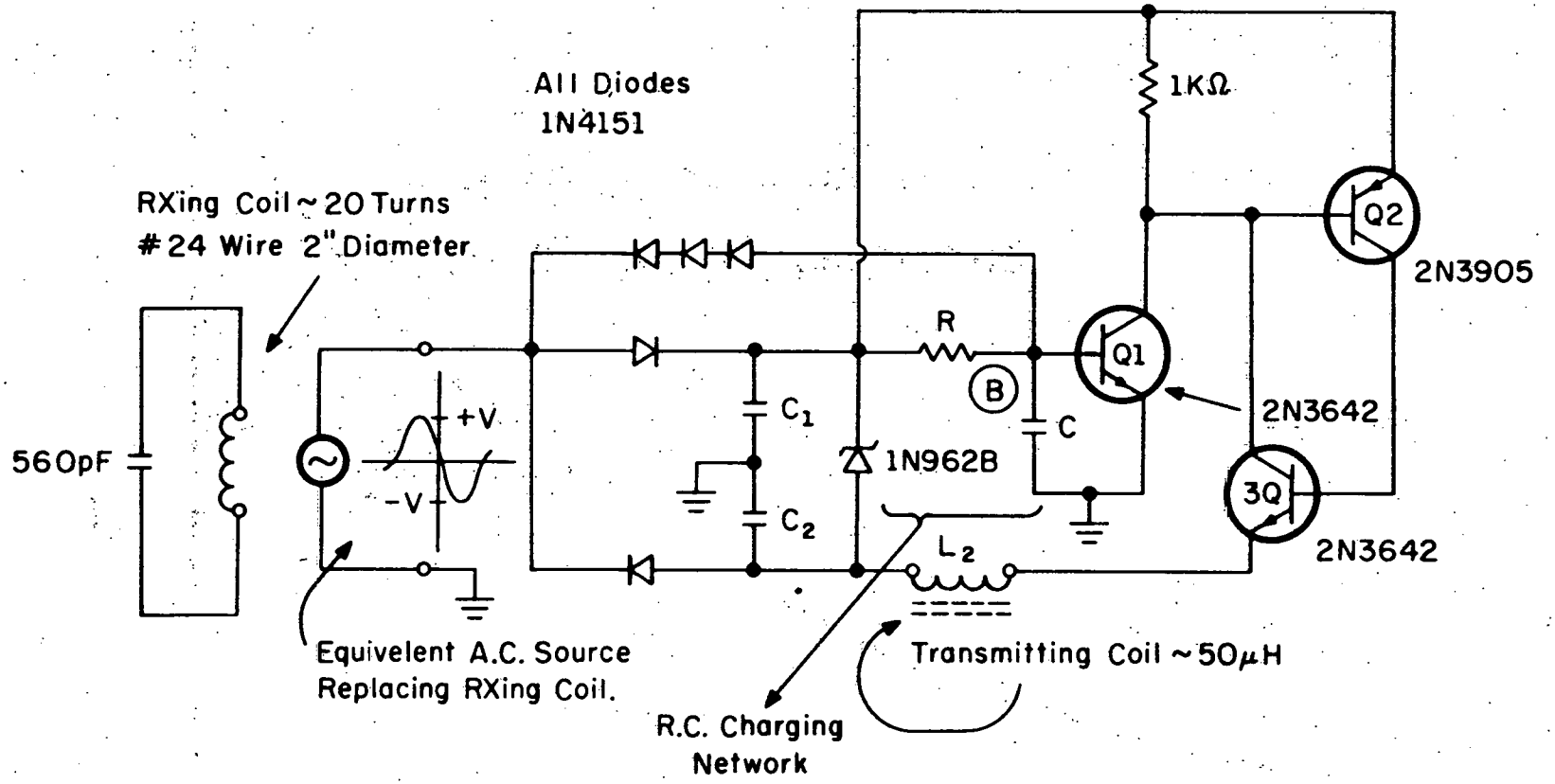


Figure 4.2.1 The Cube's Rx/Tx Circuitry

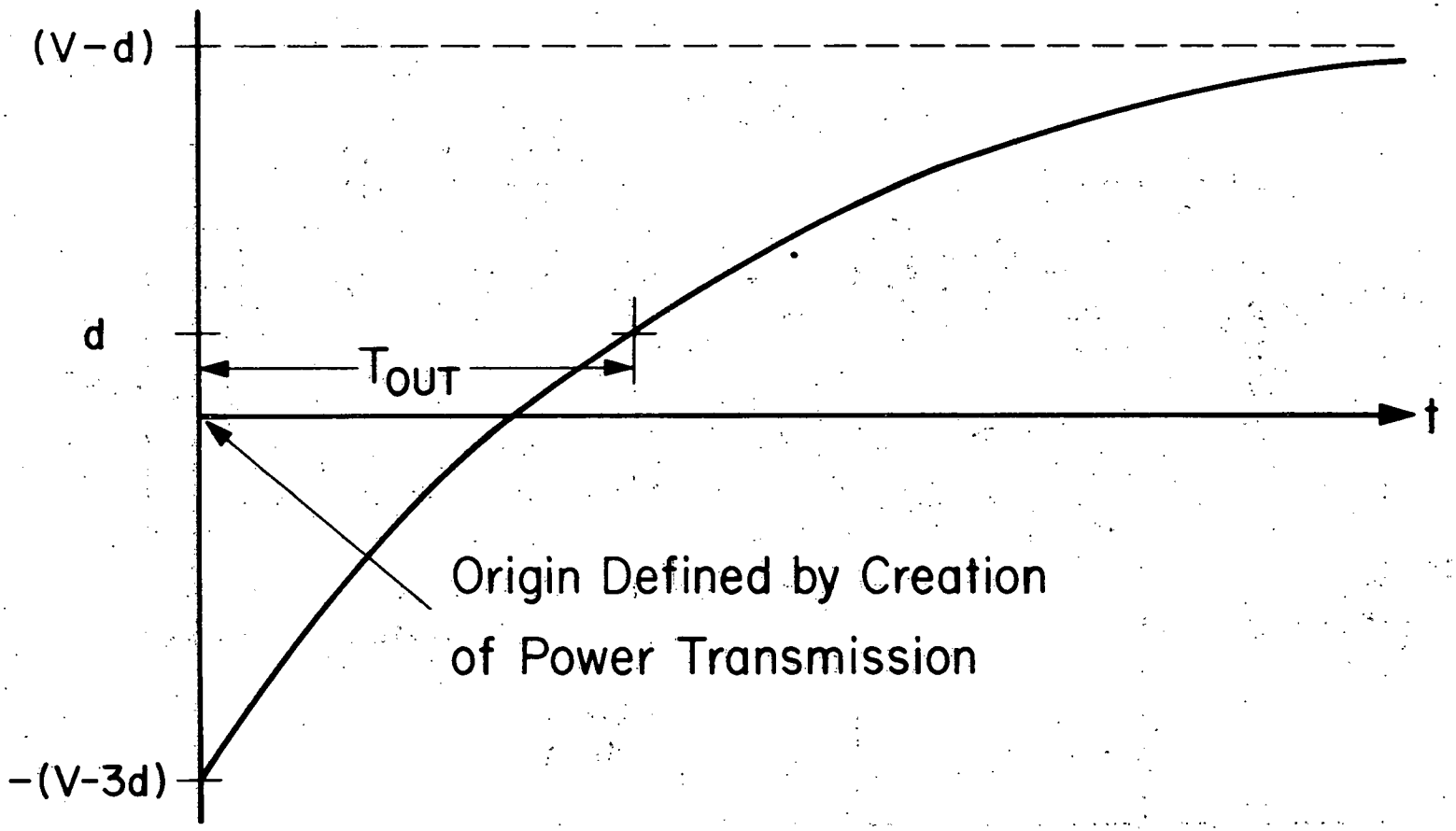


Figure 4.2.2 Voltage vs Time at Point B

and the inducted voltage at C goes to ground. The three diodes are then reversed biased and point B is essentially disconnected from point C. It can then start charging positive.

The voltage V is the peak voltage induced in the receiving coil of a cube and is not necessarily the same for each cube, nor is it the same for a particular cube at different points on the table top. The forward bias voltage drop of a pn-junction is denoted by d volts, and this is taken to be the same for all pn-junctions. In general d is a function, and in particular d = d(Temp.).

Since the RC charging network does not begin charging positive until the power transmitter has stopped, the time out to firing for every cube begins at the same instant. Furthermore, since each cube is identified by the time slot it replies in (see Figure 2.3.1(a)), cubes can be distinguished simply by choosing a unique value of the time constant RC for each cube.

The time out to firing, T_{OUT} can be computed as follows (see Figure 4.2.1):

Initial voltage at B due to the induced a.c. signal at C

$= -(V-3d)$

Final voltage at B = Voltage due to charged stored on the

$0.047\mu F$ capacitor C_1

$= V-d$

Q1 turns on when its base = d volts.

Hence the equation for the time out to fire is given by:

$$(2V-4d)(1-e^{-t/RC}) - (V-3d) = d$$

$$(2V-4d)(1-e^{-t/RC}) = V-2d$$

$$2(1-e^{-t/RC}) = 1$$

$$e^{-t/RC} = \frac{1}{2}$$

$$T_{OUT} = RC \text{ In } 2$$

Two points of interest arise from the above calculation. The first is that by using three diodes to connect points B and C the diode drops cancel out, eliminating temperature effects. The second is that the value of T_{OUT} is independent of V a highly inconsistant quantity.

4.3 The Power Transmitter

The circuit schematic for the power transmitter is shown in Figure 4.3.1. The circuit is basically a Colpitts oscillator, which is switched on for a predetermined period of time by a monstable one-shot. The inductor in the tank circuit of the oscillator is formed from a 1/2" x 1" copper bar, which has been made into a 40" square loop. This is laid round the edge of the table top flush with the surface. The magnetic field generated by the oscillating current in the loop is the medium of communication with the cubes. This field is perpendicular to the table top.

The Colpitts oscillator is formed from transistor Q6 and the tank circuit. The inductance of the loop is about 25μH, which gives the oscillator an operating frequency of about 480kHz. The voltage swing on the collector of Q6 approaches twice the power supply i.e. 200 volts; hence Q6 must have a high V_{CBO}. Furthermore, because the tank circuit has a very low Q, large bursts of current must be input through Q6 during each cycle, to sustain oscillations. Therefore, Q6 must have a high I_{CMAX}. The 2N4240 device used for Q6 satisfies these requirements, having V_{CBO} = 300 volts and I_{CMAX} = 6A.

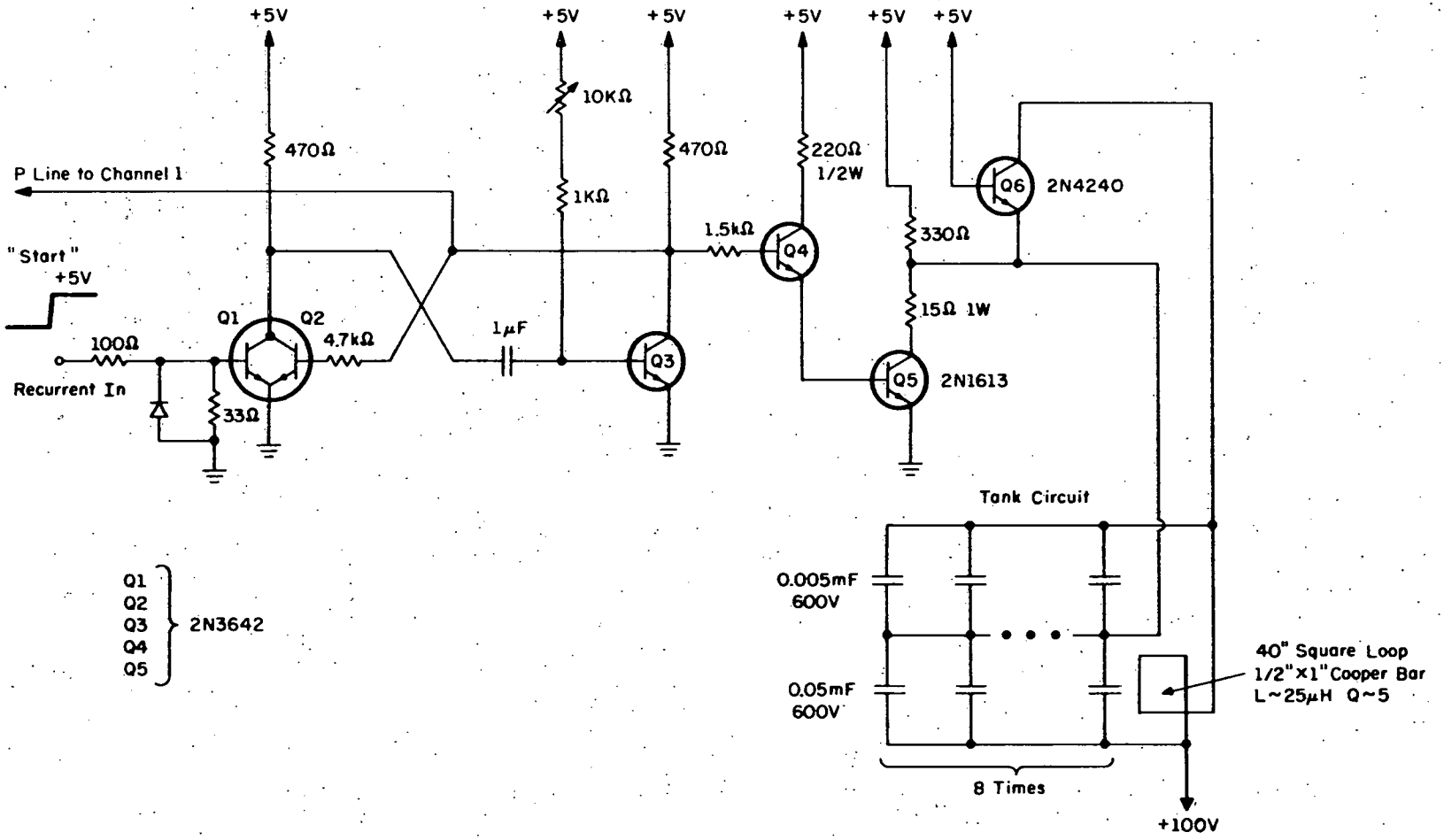


Figure 4.3.1 The Power Transmitter

Transistor Q5 provides a constant current source to bias Q6. It also provides a means to turn off the transmitter. By bringing the base of Q4 to ground, Q4 and Q5 are turned off. The emitter of Q6 is then returned to +5 volts through the 330Ω resistor. This means Q6 is turned off since V_{BE} = 0 volts. Turning off Q6 is turned off since V_{BE} = 0 volts. Turning off Q6 breaks the closed loop of the oscillator and the oscillations cease. (Quite rapidly too, since Q is so low).

To commence transmission to a 5 volt pulse in input at the start terminal. This turns on Q1 and trips the one-shot formed by Q2 and Q3. The on time of the one-shot determines the duration of the oscillations. The on time is given by:

$$t_{ON} = RC \ln 1.7$$

Now $C = 1\mu F$

Therefore $t_{ON} = R \times 0.53$ mS

Where R is in kilo-ohms.

The duration of oscillation that is required is about 1mS. Hence a resistance of about 2kΩ is required. This can be got by adjusting the 10kΩ variable resistor.

The power dissipation of Q6 demands that the oscillator be kept to a 10% duty cycle; thus it is important that noise on the +5 volt power line caused by pick-up from the oscillator or anything else does not retrigger the one-shot. For this reason three capacitors decouple the +5 volt supply on the board containing the oscillator. One is a 500mF electrolytic, which copes with low frequency noise. Due to its construction, at high frequency

it no longer behaves as a capacitor, but as an inductor. For this reason a 2.2mF ceramic capacitor is also used. Finally, it too has a high frequency safeguard which is a 200pF ceramic capacitor.

5. THE ELECTROMECHANICAL LIMB

In this section the design of the electromechanical hand-arm limb is discussed. Although this subsystem was not the responsibility of the author a brief description has been included for completeness.

The major part of the limb design is connected with generating the x-y motion.

5.1 Generating the X-Y Motion

Both of these motions use a torque proportional to error servo-mechanism. An ideal torque proportional to error servo is diagrammed in Figure 5.1.1. The error is defined as the difference between the required output (i.e. input θ_i) and actual output θ_0. The system inertia is J referred to the output and the system is damped by means of a viscous damping torque F per unit angular velocity.

Now the torque from the motor is

$$T_m = K\varepsilon$$

Retarding torque due to damping

$$T_f = F\frac{d\theta_0}{dt}$$

Resultant torque T to produce acceleration is given by

$$T = T_m - T_f$$

$$= K(\theta_i - \theta_0) - F\frac{d\theta_0}{dt}$$

But by Newton's second law

$$T = J\frac{d^2\theta_0}{dt^2}$$

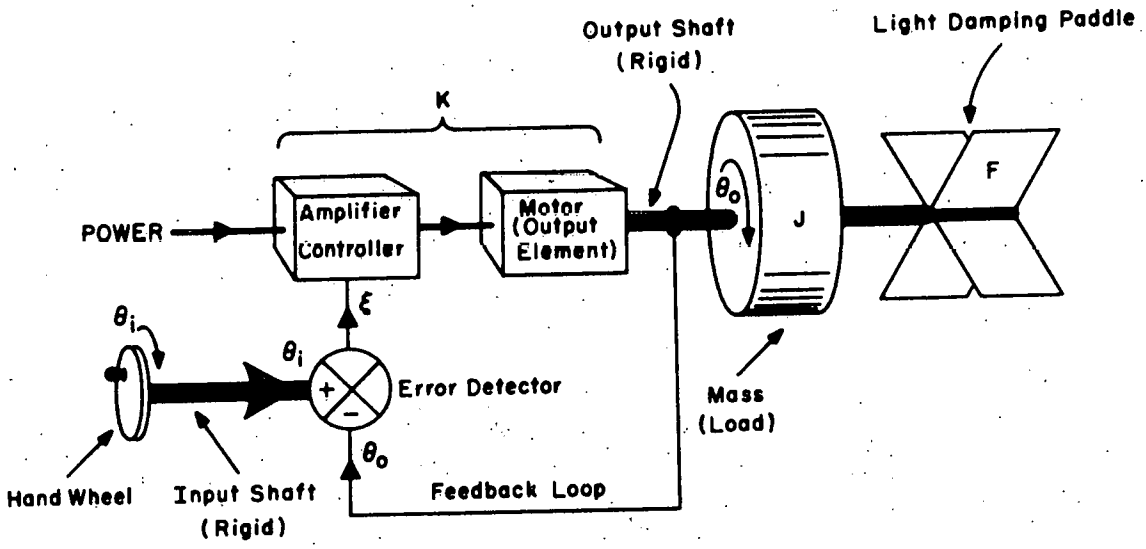


Figure 5.1.1. Proportional to Error Servo

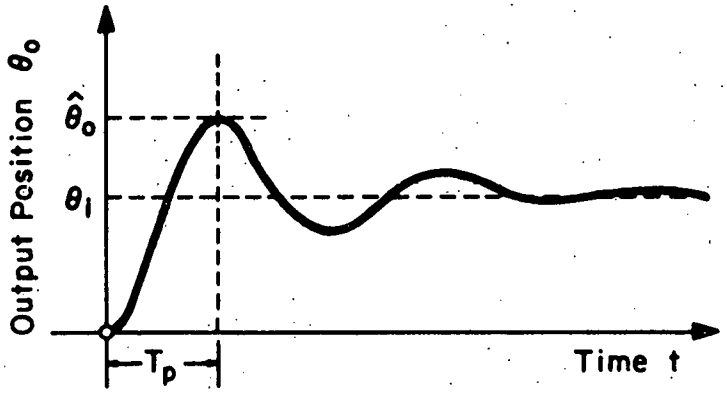


Figure 5.1.2 Response to Step Input

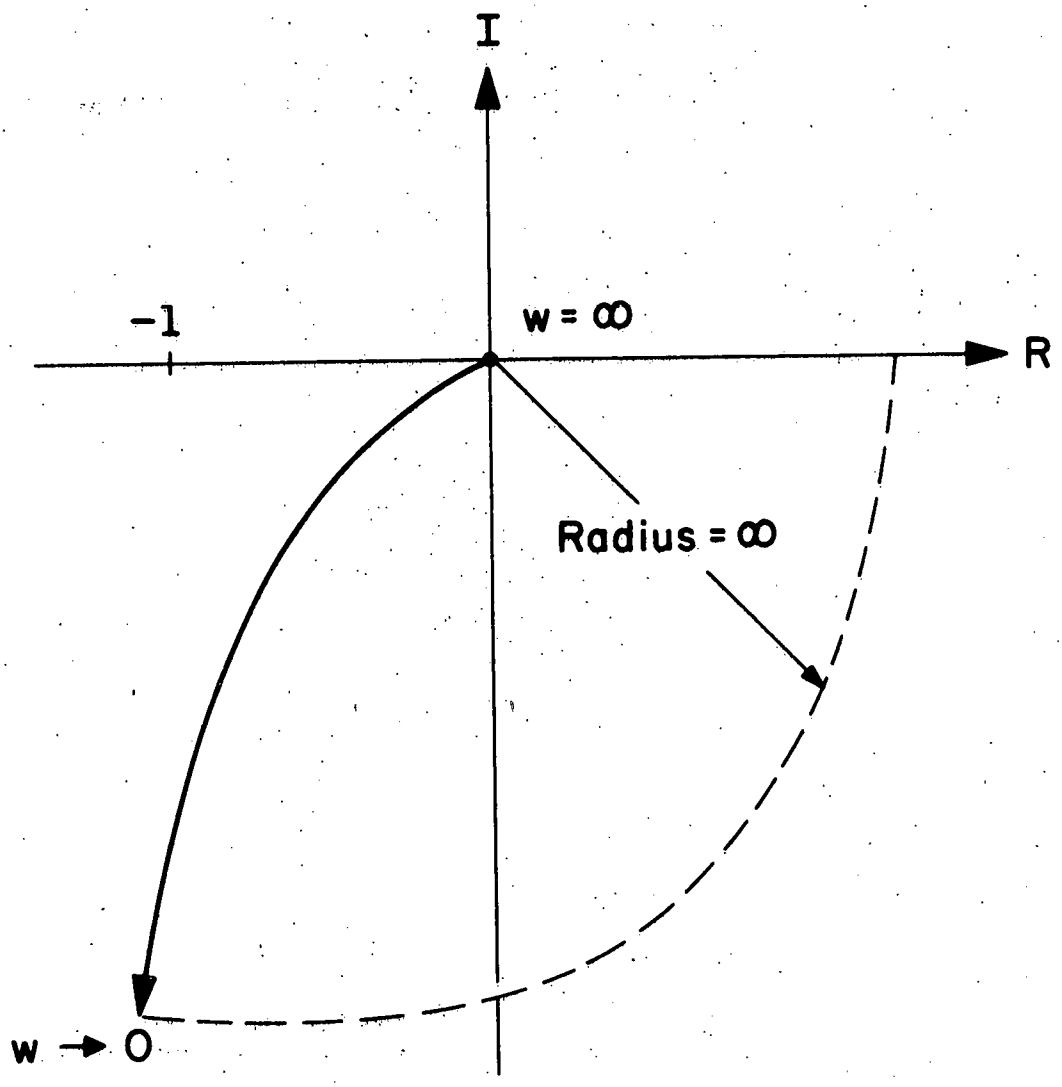


Figure 5.1.3 The Nyguist Plot for the Limbs Servomechanisms

Therefore $$J\frac{d^2\theta_0}{dt^2} + F\frac{d\theta_0}{dt} + K\theta_0 = K\theta_i$$

Which is the differential equation describing the ideal system.

In so far as this ideal analysis models the actual system, the main point of interest is the response of the system to a step input. There are three possible types of response:

1. Underdamped.
2. Critically damped.
3. Overdamped.

The solution to the system differential equation in case 1 is:

$$\theta_0 = \theta_i\left[1 - e^{-\alpha t}\sqrt{\left(1 + \left[\frac{\alpha}{\omega_r}\right]^2\right)}\,\sin\left\{\omega_r t + \tan^{-1}\frac{\omega_r}{\alpha}\right\}\right]$$

And is depicted in Figure 5.1.2.

$$\alpha = \frac{F}{2J}$$

And $$\omega_r = \sqrt{\frac{K}{J} - \frac{F^2}{4J^2}}$$

Two confliciting constraints arise at this point. On one hand, it is desirable to have an underdamped (or critically damped) system, so that the response time of the system is not unduly large. This requires ω_r to be real; that is

$$2\sqrt{JK} \geq F$$

On the other hand, to minimize the settling time, α should be large; that is,

$$F >> J$$

The actual system was shown in Figure 2.2.2. To identify the quantities J and F entails considerable measurement and calculation, and for a two off system, is not worthwhile. It is sufficient to know that increasing

F increases the settling time, and increasing J increases the underdamping. K is given in manufacturers specifications. Both motors used were high performance d.c. motors manufactured by Micro Switch. The limb was powered by a model 6VM1-1 and the hand, which moves on the limb, by a model 3VM1-1.

The open loop transfer function for the system is given by:

$$\frac{\theta_0}{\epsilon} = H(j\omega) = \frac{K^1}{j\omega(1 + j\omega T)}$$

Where

$$K^1 = K/F$$

$$T = J/F$$

A plot of this on an Argand diagram gives the Nyquist plot. This is shown in figure 5.1.3. Since the plot never encloses the (-1, 0) point, no matter what the values of T and K are, the system is unconditionally stable - a very desirable feature. For a complete discussion of stability and servomechanism see Reference 2.

5.2 The Hand's Motions

As well as the y-motion along the arm the hand has two other motions associated with it, as was seen when the instruction set was discussed in Section 3.1. These are raise/lower and open/close.

The raise/lower motion is effected by a small d.c. motor running on open loop. Sets of control switches turn the motor off when the raised position or the lowered position has been reached. The directions of the motor are also determined by these switches. (See Section 3.3.10 for a discussion of these switches).

The open/close motion of the hand's fingers is achieved in a similar fashion. The hand has two opposed fingers. These pick up the cubes by grasping a small protruding length of dowel which is to be found on the top of each cube.

6. CONCLUSION

This report describes the design of a special purpose piece of computer peripheral equipment called Semantrix. The emphasis of this report has been on the system logic, this being the major responsibility of the author.

LIST OF REFERENCES

1. Ambler, A. P., Barrow, H. G. and Burstall, R. M.: Some Techniques for Recognizing Structure in Pictures. International Conference on Frontiers of Pattern Recognition, Honolulu, Hawaii, January 1971.

2. Atkinson, P.: Feedback Control Theory for Engineers. Heineman Educational Books. London, 1968.

3. Morris, R. L. and Miller, J. R.: Designing with TTL Integrated Circuits. Texas Instruments Electronics Series, McGraw-Hill. 1971.

4. Preparata, F. and Ray, S.: An Approach to Artifical Nonsymbolic Cognition. Information Sciences, Vol. 4 (1972), pp. 65-82.

5. The Integrated Circuits Catalog. Texas Instruments Inc. Dallas, Texas.

APPENDIX A

The Clock Operation

Figure A.1 shows the clock circuitry. The hex inverters are SN7405's.

The ratio of on to off time is almost unity, as will be shown below, and is unaffected by the value of the capacitor, which can be choosen to generate frequencies from below 1 Hz.

Figure A.1 shows the central part of the circuit relevant to understanding how it works. Q_0 is the output transistor of the leftmost inverter, and Q_4 is the input transistor of the rightmost inverter.

Suppose that node b has just gone low, so that c is high and Q_0 is in saturation. Node a is then clamped at V_{CEsat} ($\simeq$ 0.2V) and the voltage at node b rises exponentially towards $V_{cc}-V_{BE}$ with time constant CR_4. When node b reaches the threshold voltage V_t ($\simeq$ 1.4V) required to switch Q_4, node c will go low, turning off Q_0. The current in R_1, which formerly flowed via the emitter of Q_1 into Q_0, can now flow only into the base of Q_2 and the left side of C. Thus, Q_2 and Q_3 turn on, causing the voltage at node a to jump to V_t. This step is transmitted through C to node b, causing it to rise to $2V_t-V_{CEsat}$. Q_1, Q_2 and Q_3 now behave like an operational amplifier. Q_2 and Q_3 are on, but not saturated. Aside from a small base current into Q_2, most of the current in R_1 flows via the emitter of Q_1 and C into Q_3. Should this current tend to increase, thereby lowering the voltage at node a, the base drive at Q_2 will be reduced, tending to turn Q_2 and Q_3 off, which in turn will tend to block the passage of current into Q_3 from C and pull the voltage at node a up again. A similar argument applies if the voltage at node a should tend to rise. This voltage, therefore, is

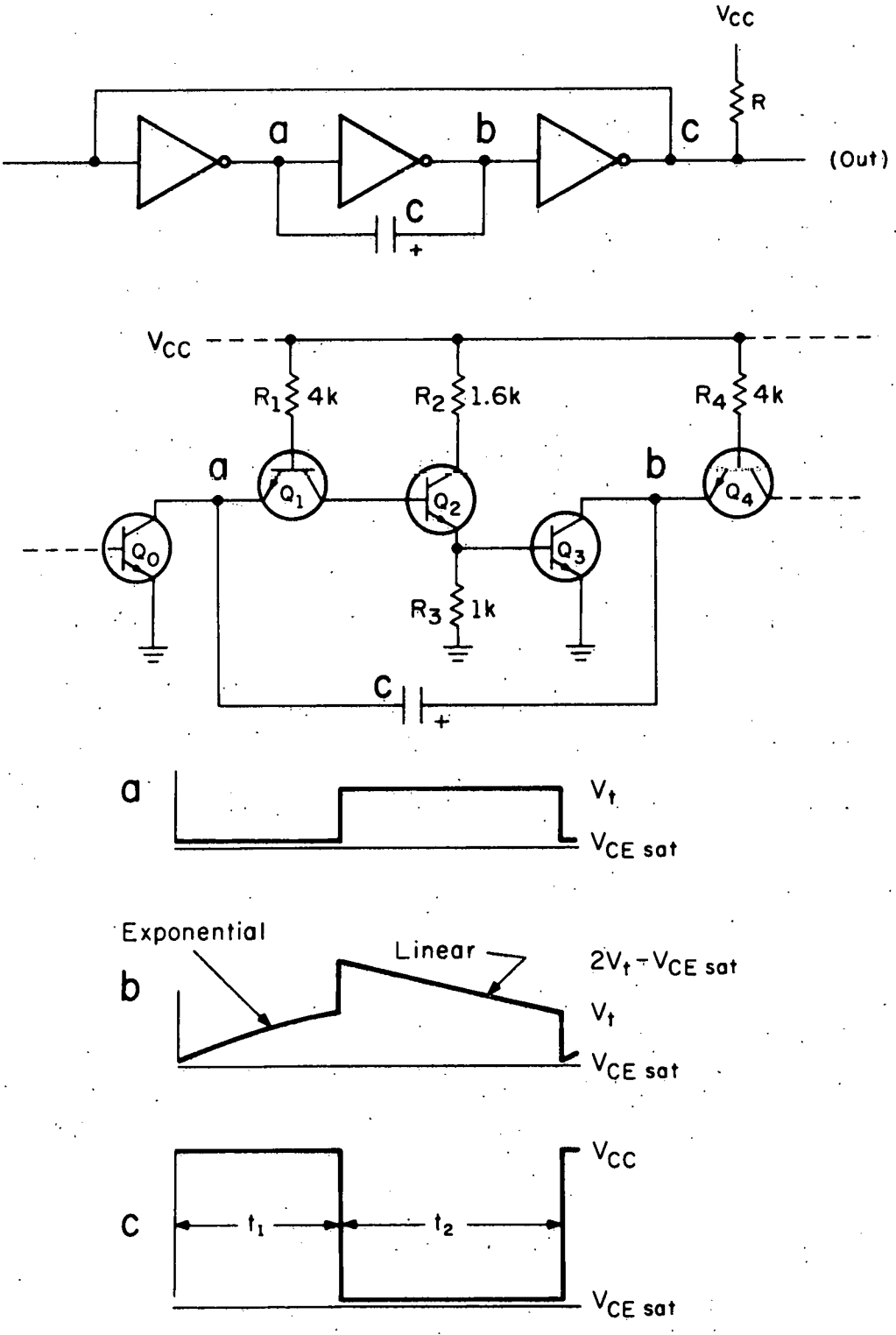


Figure A.1 The Clock Circuitry

clamped at V_t (by negative feedback through C). Hence C receives a constant current from R_1, which causes the voltage at node b to fall linearly. When it reaches V_t, Q_4 switches back to its former state, node c goes high, Q_0 saturates, snapping the voltage at node a and, via C, node b back down to V_{CEsat}, at which point the cycle is complete. The waveforms at the three nodes are shown in Figure A.1.

From the above reasoning one may readily obtain the on and off times t_1 and t_2, if one neglects the switching times of the individual transistors and the base current into Q_2 when it is on.

$$t_1 = CR_4 \ell n \frac{V_{cc} - V_{BE} - V_{CEsat}}{V_{cc} - V_{BE} - V_t}$$

$$t_2 = CR_1 \frac{V_t - V_{CEsat}}{V_{cc} - V_{BE} - V_t}$$

Nominal values are $R_1 = R_4$, $V_{cc} = 5V$, $V_{BE} = 0.8V$, $V_{CEsat} = 0.2V$ and $V_t = 1.4V$. These yield the ratio:

$$t_1/t_2 \simeq 0.83 \simeq 5/6$$

If equal on and off times are needed, t_2 may be reduced, without affecting t_1, by connecting a suitable resistor between node a and V_{cc}. This should be about 26k when $R_1 = R_4 = 4k$.

APPENDIX B

The Contents of the SROM

TTY CHARACTER	E_8	E_7	E_6	E_5	E_4	E_3	E_2	E_1	
0	1	0	1	1	0	0	0	0	$E_0 = 0$
1	1	0	1	1	0	0	0	1	$E_9 = E_{10} = 1$
2	1	0	1	1	0	0	1	0	E_{11}-- E_{15} leave
3	1	0	1	1	0	0	1	1	floating.
4	1	0	1	1	0	1	0	0	1 corresponds to
5	1	0	1	1	0	1	0	1	ground.
6	1	0	1	1	0	1	1	0	0 corresponds to
7	1	0	1	1	0	1	1	1	a connection to
C	1	1	0	0	0	0	1	1	+5 volts.
E	1	1	0	0	0	1	0	1	
L	1	1	0	0	1	1	0	0	
N	1	1	0	0	1	1	1	0	
R	1	1	0	1	0	0	1	0	
Bell	1	0	0	0	0	1	1	1	} Non-printing
Lf	1	0	0	0	1	0	1	0	
Rt	1	0	0	0	1	1	0	1	

The ROM is fabricated by placing 4 SN74150's per printed circuit board. A total of 4 such boards are thus required to make up the memory. The boards are all made with an identical layout, which is programmed so that each of the SN74150's can be wired to store any TTY character required. The

E_i of each multiplexor can be connected to +5 volts or 0 volts by soldering a small jumper pin into the appropriate hole.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-73-559	2.	3. Recipient's Accession No.
4. Title and Subtitle SEMANTRIX: A Semantically Guided Digital Electronic Machine			5. Report Date February 1973 6.
7. Author(s)			8. Performing Organization Rept. No. UIUCDCS-R-73-559
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. COO-1469-0217 11. Contract/Grant No. US AEC AT(11-1)1469
12. Sponsoring Organization Name and Address US AEC Chicago Operations Office 9800 South Cass Avenue Argonne, Illinois 60439			13. Type of Report & Period Covered Thesis Research 14.

15. Supplementary Notes

16. Abstracts

Semantrix is a digital machine, whose domain of activity (or world) is a rectangular plane or table top. (See Figure 1.1.1 for a diagram of the machine). Its activity in the plane can be instructed from either a teletype or a digital computer. Semantrix and a teletype can form a stand alone system. However, to make full use of the machine, a digital computer should be used as a controller. Semantrix can thus be viewed as a special piece of I/O equipment.

17. Key Words and Document Analysis. 17a. Descriptors

cognitive system
special purpose digital controller
semantrix
two dimensional position locating system

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited distribution	UNCLASSIFIED	105
	20. Security Class (This Page) UNCLASSIFIED	22. Price