An 8085 Repeater Control System

Robert Glaser N3IC

March, 1980

Developed for the Baltimore Amateur Radio Club's Repeater

WA3KOK/RPT

(146.34/146.94 MHz)

Robert Glaser N3IC
3922 Algiers Road
Randallstown, Md 21133

An 8085 Repeater Control System

The feasibility and desirability of a microprocessor controlled repeater has been proven for large, intricate repeater systems. The 8080 repeater control system described in February, March, April, and May 1979 73 Magazine showed how a microprocessor based design could perform tasks which would be impossible in the practical sense without a microprocessor. The repeater system for which that design was initiated was more complex than the norm. Any reasonable control system for a repeater so complicated as WR3AFM would be expected to be of a moderate to large size, so the 8080 design which utilized 57 integrated circuits was not considered out of line for the intended use. Where does that design leave the smaller repeater operator who would like to modernize his control structure, but not at a cost of so much circuitry? We ran into this delemma with our club's secondary repeater; it would have been nice to have a neat micro-based design, but not if it entailed as much circuitry as the 8080 designed for our primary repeater. The result is this 8085 repeater control system.

The overriding goal for the new design was to make it as small as possible, while retaining software compatibility with the existing 8080 control system. It is a major task to write a complete, debugged control program, and the thought of rewriting the program for a different microprocessor language was not

relished. The Intel 8085 microprocessor has complete software compatibility with the 8080, and has a number of hardware advantages, so it was the natural choice. This 8085 control system consists of only eight integrated circuits, and fits easily on a 4.5 by 8.5 inch board. The power requirement is simply a five volt supply at considerably less than one ampere of current. This new system basically performs the same functions as the 8080 controller. The major function which has been removed is the autopatch.

The new implementation incorporates the old functions of touchtone(TM) blocking and unblocking, testing of touchtones, multiple identifications, programmable IDs, and extensive control outputs. In addition, the system has a CW "Bulletin Board" feature. Messages may be left for other stations to retrieve at their leisure. There is also an RTTY test feature which permits the repeater to identify in Baudot AFSK tones for equipment alignment. The microprocessor also generates a unique "bleep" to indicate the time-out timer reset. For a detailed description of the functions available to the general repeater users, see "Using the System," and for those functions available to control operators, see "Controlling the System."

The project has been a successful one, and there have been no hardware failures or software crashes as of five months after installation. Recently a duplicate of this system has been placed on the Anne Arundel Radio Club's 222.28/223.88 MHz repeater by K8JW/3. Reportedly that system is functioning properly also. In the following sections, I will describe

construction, hardware, and the hardware/software balance. A microprocessor and complete circuit description, and the software additions and changes to the old system will be given. Some potentially desirable modifications will then be discussed.

Construction

I am discussing construction before going into a detailed description of the controller because I wish to emphasize how simple the project really can be. There are two levels upon which this controller can be approached. For the ham willing to duplicate it with no modifications, an understanding of how it works is not mandatory. Understanding the sections, "Using the System", and "Controlling the System", and knowledge of the interface connections in Fig. 4 is sufficient for rote duplication. The detailed hardware and software analysis is for those hams who wish to modify the system according to their needs. For those of you in the first category, do not shy away from the project because there is a lot involved internally. Few people who utilize computers actually understand fully how they work.

The scope of this project is vastly different from that of the original 8080 control system. It consists of wiring eight integrated circuits on a vectorboard, and making the cabling harnesses to interface with your repeater. The use of relay switching for the audio assures that there will be no voltage or impedance problems with any repeater. The limited connection to the existing interface circuitry of your repeater should require

little or no additional circuitry over what was needed for ours. In short, to duplicate this project for another repeater, all that is really necessary is to understand the repeater; the "black box" approach can certainly apply here. The only exotic microprocessor knowledge needed is how to wire this small board of eight chips. Get a local computer type to program the EPROM from the program listing, or contact me for the current price of this service. The only changes required in the program for each individual repeater are choosing the codes currently specified as 123, 456, 789, and of course the call letters of the target repeater.

The controller board is assembled with wire-wrap techniques using the Vector Slit-N-Wrap tool. Discrete components are mounted using Vector T49 pins, which permit wrapping underneath the board while soldering the components on the top. Conventional wire wrap will work just as well, although daisy-chaining the bus wiring between the the three main chips is slightly easier with the prescribed tool. Power supply bypassing, not shown on the schematic, is accomodated by several .047 mfd capacitors added to various spots on the board between ground and +5 V.

Fig. 1 shows the layout of the board. The reset switch is any normally open, SPST switch. The VTT, SVTT, and beep delay trimpots are shown as oblong objects. The resistor package R consists of 13 6.8K resistors, with common pin 14. Beckman 899-1-R6.8K or Bourns 4114R-002-682S are suitable. The relays are 5-volt TTL compatible SPDT reed relays. The price for the three

microprocessor chips is about $90 at the time of writing. Construction time for the board is about four hours.

Connections are made to the rest of the repeater with standard DIP sockets and cable plugs. A five volt power supply must be provided. IC specifications call for an absolute maximum current draw of 700 ma., but you can expect about half of that.

Contact me for the price of a completely assembled and tested controller board.

Hardware Description

To illustrate where the control system fits in a typical repeater, consider the Baltimore Amateur Radio Club's 34/94 repeater (WA3KOK/RPT). Fig. 2 shows a block diagram of the transmit site of this repeater. 34/94 has multiple receivers in the same fashion as our main repeater, 07/67. Each of the remote sites consists of a 146.34 MHz receiver and 440 MHz link transmitter. The link receivers at the transmit site feed the voting selector, which makes the multiple receivers appear as a single receiver. This configuration is atypical in the amateur service, but for the purposes of describing the 8085 control system, the voting selector makes no difference, other than to point out that there are six select and six disable lines for the purposes of forcibly selecting or disabling individual receivers. All that matters from the controller's viewpoint is that there are twelve outputs from it to an external device. These lines may be used for control of any external device desired.

The audio from the voter goes through the controller, and

normally passes through unaffected to the repeater interface. The carrier operated switch (COS) becomes modified to incorporate the beeper, and feeds the interface circuitry. Audio from the voter also drives the touchtone decoder, which provides for the user functions. The touchtone decoder is separate from the controller. It must decode all twelve digits and provide a ground upon the presence of the proper tone pair. A valid touchtone (VTT) output must supply a ground whenever any of the twelve digits are recognized. The decoder should have no delay on the outputs; the controller supplies the desired delay to prevent voice falsing. If delay is present in the decoder, it will disturb the reception of Morse code via touchtones.

For convenience in testing, and for control at the repeater site, a dummy keyboard can be assembled which appears the same as the decoder to the controller. Fig. 3 shows how to connect a twelve button SPST keypad to emulate the decoder. An OR of all twelve buttons is provided by diodes to produce the VTT signal. The keyboard is not the matrix type. To use the keypad, unplug the decoder plug and plug in the keypad.

For control purposes, a separate control receiver is used to enter the system. The presence of its COS signal informs the control system that a priority signal from the control receiver is present, and the touchtone decoder switches from the voter output to the control receiver output.

The interface circuitry consists of the audio coupler, timers, and level shifter for keying the PTT line, all in a normal repeater installation. The audio and COS leads from the

voter, or the receiver, are removed from the interface circuitry and looped through the control system. The normal interface lines for connection of a standard CW identifier are also required. The audio input from the IDer, and a PTT input which permits the IDer to key the transmitter, are used by the controller. Additional control inputs for your system can be met by any of the general outputs of the controller.

The detailed interconnections are shown in Fig. 4. The VOTER output socket provides 16 general purpose control outputs. Of these, 14 are open collector, and 2 are TTL level (pins 8 and 16). In our system these are used to command the voting selector. The touchtone decoder plugs into the TTD socket on the controller. The rest of the connections are through the interface (INT) socket. The COS signal from the voter or receiver is open collector, and must provide a ground when a signal is present. The COS switch shown is not required, but makes it easier to test the controller when at the repeater site by fooling the controller into believing a signal is present. This is necessary to test the touchtone features with the local keypad; the COS signal tells the processor when you are done a sequence.

A ground on the PTT input to the interface circuitry should key the transmitter. The DIS input disables the repeater when grounded. The TIM input is not required, but in our system a ground on this line disables the three minute time-out timer. The SOD and SID lines are currently unused, but suitable program changes could make use of this output and input.

The audio gain on the touchtone decoder should be

adjusted for the level from the voter or receiver, and the audio level of the control receiver adjusted to that of the voter or receiver.

Hardware/Software Balance

In the original 8080 controller article I pointed out the advantages of finding the best split for these two methods. This implementation is considerably less complicated than the first, and most of the hardware external to the processor has disappeared because those functions have gone. Given the prior successful model, on this project I minimized the hardware by placing as much in software as possible. The new software functions are the beeper, and the automatic blocking defeat when controlling via the control receiver. Retained in hardware is the amount of time required to hold the first touchtone for recognition, and the amount of time after carrier drop before the beep occurs. These are adjusted by potentiometers.

The Microprocessor

Three ICs comprise the microprocessor section of the controller. Fig. 5 shows what these chips contain. The 8085 is the central processor unit (CPU). It has a built-in clock oscillator - all you do is add a crystal! The clock then runs the internal affairs of the CPU, and its output is also available for use by the other chips. The 8085 has two built-in input/output lines, but these are not used in the controller.

This CPU has several input pins called interrupt inputs.

When these inputs are activated, the processor stops executing its current program, and instead, executes a different program, called an interrupt service routine. When the interrupt routine is finished, the processor picks up where it left off in the original program. The use of interrupts makes programming simpler, because it becomes possible to make the microprocessor perform several different functions by writing independent programs for each function. Without interrupts, there can only be one main program which must perform all of the required tasks; moreover, it must constantly check to see which function it should be performing.

On the controller, two different interrupts are used. One is activated whenever a touchtone is sent. The processor stops whatever it is doing at that time, and sees what touchtone code is being sent. After performing whatever task the touchtone code required, if any, it returns to the main program. This is the same as in the original 8080 controller. A second interrupt is used to detect the carrier drop. Approximately 3/4 second after the repeater receiver squelches, the microprocessor is interrupted. Again, it stops executing its current program, and goes to a program which does nothing other than generate a beep to indicate the time-out timer reset. After beeping, the original program continues. It is possible to mask out individual interrupts when it is desired to prevent interruption of a current program.

The instructions which tell the CPU what to do are stored in the 8755 erasable, programmable, read only memory (EPROM). The

8755 contains 2048 bytes of memory, the same amount as the two 2708s in the original 8080 controller. This EPROM also has sixteen input/output lines. Each line may be programmed as either an input or an output, which makes the system quite flexible. In this design, they are all programmed as inputs. These inputs permit the processor to read the touchtone decoder and the COS signals from the repeater and control receivers.

The third element of this powerful trio is the 8155. This is a random access memory (RAM) which is used for temporary data storage, message storage, and the programmable ID in this controller. This is equivalent to the two 2112s used in the 8080 controller. The 8155 also has 22 input/output pins. These may be programmed as inputs or outputs in groups of 8, 8, and 6. In this design they are all set up as outputs. They go to TTL gates which convert the TTL level signals to open collector outputs which can give either an open or a ground. Open collector outputs are about the best general purpose output levels for interfacing to external devices where the exact voltage requirements vary, such as in repeater service.

The 8155 also has a programmable 14 bit counter/timer. In this application it is used as a programmable divider which can divide anywhere from 2 to 16383. This divide ratio can be changed at any time by the microprocessor. In this controller, the divider input is connected to the clock oscillator output from the 8085, which is crystal controlled. The clock output is half the crystal frequency, which in this case is 3579.545 kHz (the common color burst crystal). The output of the divider can

therefore be programmed to be from 109 Hz to 895 kHz with crystal controlled accuracy. This is the output which generates the ID tone, the conversational CW tone, and the "bleep". All of this without even having to use an external audio oscillator!

The three microprocessor chips communicate through the address, control, and data buses. The 8755 and 8155 peripherals are sophisticated enough that no additional interfacing logic is required other than one inverter. Only a single supply voltage is required to power them.

Circuit Description

The controller schematic is shown in Fig. 6. The touchtone decoder outputs go directly to the 8755 input ports, but are paralleled with pull-up resistors. Package R is a convenience and may be replaced with 13 individual resistors if desired. The COS and CRCOS signals have separate pull-up resistors and isolation diodes.

The VTT signal from the decoder needs to be debounced before sending it to the processor. The .047 mfd capacitor and 1M trimpot set the time constant, and the 4049 CMOS inverters buffer the signal and square it up. A similar circuit provides the slow VTT (SVTT). This signal does not appear until the VTT has been present for a specified length of time. The 1M trimpot and 1 mfd capacitor set the delay, nominally .5 second. The uncomplemented SVTT goes to the RST 5.5 interrupt input on the CPU.

The remaining two inverters in the 4049 are used to generate the delay between carrier drop and the bleep. The COS is

delayed and fed to the RST 7.5 interrupt input. The 1M trimpot and 5 mfd capacitor set the delay, nominally .75 second. The 7406s #1 and #2 invert the outputs from the 8155 and provide open collector outputs. Four inverters in 7406 #3 do likewise. The fifth provides an open collector output for the delayed COS signal, to be sent to the repeater interface circuitry. The last inverter in this package is used to invert the A11 address line for proper address selection for the 8155 and 8755. If an 8755 is used, pin 5 should go to ground, and if an 8755A is used, pin 5 should go to +5 V. The timer output from the 8155 is low pass filtered to shape the square wave output closer to a sine wave before going to the repeater interface circuitry.

The control receiver touchtone (CRTT) relay is activated by the control receiver COS (CRCOS). Whenever a signal is present on the control receiver, CRCOS goes low, activating CRTT, which switches out the audio from the repeater receiver and switches in the audio from the control receiver as a source for the touchtone decoder input. One section of a 7403 open collector NAND gate is used to buffer PC1 of the 8155. This permits CRTT to be pulled in under program control, which removes all access to the decoder from the repeater receiver. There is no method in this design to inhibit access by the control receiver.

Repeat audio passes through the BLOCK relay to the repeater interface. When a touchtone signal is received, and PC0 of the 8155 is high, the BLOCK relay pulls in. This removes the repeat audio path, and grounds the transmitter audio input, muting the touchtones from repeating. PC0 only goes high when the

8085 has been interrupted by the SVTT signal, so there is no problem with falsing of the touchtone decoder accidentally muting voices. Upon a touchtone interrupt, if there is a signal from the control receiver, PC0 is not set high, since the decoder access is not from the repeater receiver in that case, and there is no need to mute any tones. The last section of the 7403 is used to buffer PC4.

The connector pinouts are summarized in Fig. 7. There are only two paths broken from normal repeater connections, so to place the repeater on the air with the controller, the interface plug can be plugged into the dummy socket which jumpers these paths.

Software Description

Much of the software is identical to that published in the 8080 controller article, so only the changes will be described. In order to fit all of the new features into the 2K EPROM, it was necessary to go over the program many times, streamlining it where possible to save memory space. As a result, some of the original routines are functionally identical, but are changed somewhat to be more efficient.

The new hardware configuration changed the input/output port structure. The port assignments are shown in Fig. 8. For each port, the mnemonics used in the program are given first, followed by the actual hexadecimal port address. DDR1 and DDR2 are the data direction registers for PORT1 and PORT2, and are set to zero to make these two ports inputs. Output port PORT0 has no

physical output lines and is used internally for function disabling. A copy of PORT0 is kept in memory as OUT0M, and this port is accessed from the output mode in the control codes as port 0. Similarly, output ports OPOR1, OPOR2, PORT3, TIML, TIMH, and CSR are copied in memory locations OUT1M, OUT2M, OUT3M, TIMLM, TIMHM, and CSRM, and are accessed in the control codes as ports 1, 2, 3, 4, 5, and 6. TIML, TIMH, and CSR are not normally used, but with a complete understanding of the 8155, it is possible to directly command the programmable divider through the control codes to synthesize any desired test tone on the repeater. Bits 7 and 8 on PORT3 are dummy bits and are used as internal switches.

The foreground program monitors the use of the repeater and produces an identification when necessary. It is the same as in the original article except that all references to the second repeater have been removed, and a few additions have been made. This main program is shown in Fig. 9. When in the master loop, the CW sending speed is set to 19 WPM. Likewise, the ID tone is set to 1 kHz. This is because the touchtone codes can change the sending speed and CW tones, and it is necessary to reset these back to normal when returning to the main loop. The speed for the Morse code copying routine is set to 13 WPM as the starting point. The interrupt structure is constantly updated to aid in error recovery. IDTM has been changed to maintain the three minute ID time interval despite the software changes and clock speed difference.

The delayed COS signal causes an RST 7.5 interrupt. This

interrupt service routine to generate the bleep tones is shown in Fig. 10. After saving some registers, the subroutine CHECK is called. CHECK determines if the stack pointer is in the proper area of RAM. If all is okay, the subroutine returns. If the stack pointer is not even in writable memory, a cold start is executed by jumping to the very beginning of the entire program. If the pointer is in RAM, but below its normal boundary, a warm start is executed by reloading the stack pointer and jumping to the main loop. If the beep is enabled, and a false interrupt has not occurred, the three beeps are generated. The DELAY routine used to time a dit for CW purposes also times the length of the beep notes. After the conclusion of the third note, the counter is stopped, and checked to see if it was in the process of counting when the RST 7.5 interrupt occurred. This can happen during an ID sequence. If it was, the counter is restored. The RST 7.5 interrupt input has a latching input, and this latch is cleared when the beep is finished in case another interrupt occurs during the beep time. Registers are restored, interrupts reenabled, and the interrupt sequence exits.

The PTTON (PTT on), PTTOF (PTT off), TONON (tone on), and TONOF (tone off) subroutines should be self-explanatory. The control routine (CNTRL) has the telephone number load jump removed, so single digit codes * or # now perform the same function, resetting the ID timer.

An RST 5.5 interrupt calls TTONE, the touchtone handling software. TTONE has been modified to remove the * knockdown code. It now also provides the blocking signal on PC0 only if the

control receiver is inactive. Otherwise TTONE is identical to that of the 8080 system.

The output routine (OUT) has three modifications: it only permits access to ports 0 through 6, it is changed to reflect the new hardware output port structure, and a # now exits the routine instead of a *. ROGER is identical except that the response "R" has been changed to "OK" for variety. The routine is still named ROGER, however. The deleted functions have been removed from the code table (CODTB) and new codes added.

The LIST routine is new and is shown in Fig. 11. This program goes through the stored messages, computes message numbers, and sends the number and call for each message on file. The message storage format is at the bottom of the figure. Starting at IDAD5, the fifth ID and all of the messages are stored in sequence. A zero byte indicates the end of the fifth ID and the beginning of the stored messages. Each message is stored with the call first, followed by a zero byte, the message, and a null. Any number of messages can be stored in this fashion, and after the end of the last message (message N), a final null ends the table. The function of the LIST routine is to skip ID5, give the number of each message, send the call, skip the message text, and continue until the end of the table. To aid with these searches, the subroutine PASTZ (go past the first zero) is used. Register pair DE is the memory pointer, and DE is incremented until it points one past the memory location where a null is located.

If the list function is enabled, LTONE (low tone) is

called first. This sets the pitch of the CW tone to 400 Hz. WCD waits for the carrier drop, and SPACE is called which sends a space (just a delay). DE is loaded with IDAD5, and PASTZ is called. Upon return, DE contains the address of the beginning of the first call in the message list. If the contents of this memory location are zero, there are no stored messages, so "NONE" is sent, and the program exits. Otherwise, at LIST1 BC is loaded with DIGAD, the address of conversions for sending the single digit numbers in CW. HL is loaded with the actual CW address for the current digit. BC is doubly incremented to point to the next digit, for future messages. The beep interrupt is inhibited, and interrupts enabled. CW is called, sending the message number, but it is possible to cause a touchtone interrupt during the CW. The call letters are then sent, interrupts disabled, and a space sent. PASTZ is called, advancing the pointer past the message text. If the message table is ended, the routine exits, otherwise it loops back to LIST2, repeating the sequence for the other messages. In this fashion message numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, and # can be assigned, in that order. It is unrealistic to expect more than four or five messages to be stored, since there is not enough room to hold them. If more RAM is added in the future (who knows, maybe Intel will come up with a new 8155 type with more RAM) the software can handle it.

The PLAY routine is shown in Fig. 12. Its function is to play the requested message. If enabled, LTONE, WCD, and SPACE are called, and "NR?" is sent in CW. INDIG is then called. The subroutine INDIG waits for a signal to appear, and counts the

number of touchtones sent during that transmission. The zero flag is set if only a single tone is received. After playing back the digit or digits requested, if a single digit was not received, "NONE" is sent at LIST3 and the program exits. Otherwise, the digit is saved in B and C, and at PLAY1 DE is initialized to IDAD5. PASTZ advances the pointer to the stored call, and if the table end has been reached, control goes to LIST3. If this path is taken, a message number has been requested which does not exist. Otherwise, DE is stored in TEMP, and PASTZ advances to the beginning of the message. The message number, in B, is decremented, and if it is now zero, the pointer is at the requested message. If it is not, the program loops to PLAY2, looping until either the message is found or the table ends. When found, the address is placed in HL and CW is called, sending the message. The question "ERASE?" is sent, and INDIG is called. If a single digit is received, control goes to PLAY3 where the message is erased. Otherwise, the query "SLOW?" is made. If the response is not a single digit, ROGER is called and the program exits. Otherwise, SLWCW is loaded into SPEED, informing the CW program to send at 10 WPM. A space is sent, and the digit request in B (previously destroyed) is reset to the original request saved in C. The program jumps to PLAY1, repeating the entire sequence at the slower CW speed.

At PLAY3, DE points to the null at the end of the message that was played. TEMP contains the starting address of that call/message pair. The double loop copies everything after the current message lower in memory, starting where the beginning of

the current message was, until the table end is found.

The STORE routine is shown in Fig. 13. If enabled, UNBLK is called, which unblocks the touchtones, permitting them to be repeated. This lets repeater listeners hear the touchtones to permit copying the CW. LTONE, WCD, and SPACE are called, and DE is loaded with IDAD5. At STOR1, PASTZ is called until the table end is found. "TO?" is sent, and the COPY subroutine is called. This receives the Morse code, converts characters into the proper format, and stores them in memory starting at the address pointed to by DE. The table end null is replaced by the first character received, extending the table. "MSG?" is sent, and DE is incremented to save the null for call/message separation. The message text is stored by calling COPY again, and the table end zero is appended. ROGER is called, and the routine exits.

The Morse decoding algorithm comes from "A Morse Code Handler" [Byte Magazine, October 1976, page 57]. The COPY routine is shown in Fig. 14. Upon entry, it loops until a signal arrives. The current estimate of the input speed is stored in BAUDI, and this is accessed through the HL register pair. The loop beginning at UPTIM measures the key up time, the period when no tones are being received. In this loop, if the signal dissappears a null is stored, indicating the message end, and the routine exits. The subroutine TICKI is used to measure time. TICKI delays an amount dependent upon the current value of BAUDI, and ideally, delays 1/8 of a dit time. At NTEOM (not end of message) E counts the number of TICKI calls made while the key is up. If this exceeds six dit times (48 TICKI delays) a space is assumed, and the space

code (80H) is stored at COPY3. If this amount of time has not elapsed, but 2.5 dit times have passed, the character is assumed to be completed, and the input character is left justified and stored. If at any time, the allocated message memory is exceeded, the pointer is decremented, causing overwriting of the last character and preventing excursion out of the specified space.

When a tone is received (indicating key down), the down time is measured at INSEN. If the down time is two dit times or greater, the element is assumed to be a dah, otherwise a dit. If a dit is measured as too short, or a dah as too long, BAUDI is modified to speed up or slow down the estimated CW speed. This way, the decoding algorithm tracks speed changes. At INPOK, the dit (0) or dah (1) is shifted into the current character register, D. Control then goes to COPY1 to measure the key up time.

The RTTY sending program is shown in Fig. 15. This is a rather simple-minded program, as no code conversions are made. Data stored at TTYMS (teletype message) is shifted out and converted into audio frequency shift keying tones with no regard to start bits or stop bits. These bits are handled by placing the correct data at TTYMS. The byte is shifted left, and when a zero is encountered, a space tone is sent, and when a one is found, a mark tone is sent.

If the function is enabled, the transmitter is keyed, and HL is initialized to point to the message. A null byte indicates the end of message, and when this is found the tone is disabled, the transmitter unkeyed, and the program exits. If the fetched

byte is nonzero, all eight bits are shifted left, and the mark or space tones are sent accordingly. A number of FFs are placed in the beginning of the teletype message to send a few seconds of mark tone leader.

Modifications

Potentially the most useful function removed from this version is the autopatch. With the addition of the touchtone generator chip, controlled by one of the current output ports, it would be possible to add the autopatch function without much more hardware. The autopatch software could then be lifted from the 8080 article. The only problem is that there is not enough room in EPROM to fit both the autopatch and the message storage programs. However, to some extent, the routines in the original article and those in this one can be "mixed and matched" to suit individual requirements. Of course, by adding one more 8755, there would be ample room to hold software for all functions (and gain two more ports, too).

The RTTY routine is set up for 45.45 baud to accomodate the Baudot standard. Since ASCII has just been approved for the amateur service, it may be desired to make the RTTY test function an ASCII standard. All that needs to be done is to change the value of RTYSP to reflect the desired sending speed, and to work out the proper bit patterns at TTYMS to send the ASCII code for the message. A value of 249 for RTYSP in line #36 will give a 300 baud sending speed. The mark and space tones are defined in lines #43 and #44, and can be changed if another convention is

preferred.

A control routine which has been added to the 8080 controller, but not included in the 8085 controller due to lack of space, has been quite useful. It permits reading back the status of the output ports to determine what has been activated by a control operator. When in the status read mode, the processor waits for two-digit codes. For request XY, the processor will send either "0" or "1", indicating the status of port X, bit Y, using the same conventions as the output mode. To exit the mode, a * is sent. By leaving out some of the routines in the modified software, it would be possible to make room for the status read routine shown in Fig. 16. The address of READ should be placed into the confirm code table for access to this feature, or alternatively the single digit # code could be diverted to it.

Upon access, ROGER is called, informing the control operator that his code request has been accepted. LOAD is called, which gets three touchtone digits. Only the first two are used. If an invalid code is requested, it is ignored, the same as in the output mode. Port 0 is stored as 10, so it is necessary to adjust for this. The binary digit code for the requested bit is converted into a bit code which can test the desired memory location. A "0" or "1" is sent, and the routine loops back to READ7, continuing.

Acknowledgments

Thanks go to Robin Becker WA2NYE/3 and Carroll Van Ness K3HZU for their assistance on the day of installation on 34/94. Our repeater users must be complimented on how easily they picked up how to use the new functions. Their integrity has resulted in the functions being used intelligently instead of causing additional headaches. Quite naturally, for several weeks after installation, the message store facility was overloaded, and messages were accidentally destroyed by people who had not yet learned how to properly utilize it. After the break-in period, this rarely occurred.

Using the System

(The following text, after suitable modifications, is intended to be distributed to users of repeater systems using the 8085 control system. A brief description of the processor and instructions on the use of the functions is included.)

34/94 (WA3KOK/RPT) is now controlled by an 8085 microprocessor. There are five cw IDs; four are permanently stored, and one is remotely programmmable via a Touch Tone pad.

Each of the individual receive sites may be both disabled or forced on through the voting selector. It is possible for control stations to operate the system without interfering with repeater users, so when "OK"s are heard on 94 this means that someone is commanding it. Under such circumstances, be slow to pick up transmissions in case the control operator has any requests; if none are made, feel free to continue using the repeater and ignore the "OK"s.

The 8085 microcomputer board consists of 8 integrated circuits. The active microprocessor components are: 8085 CPU, 8755 EPROM-I/O, and 8155 RAM-I/O-Programmable counter. The input to the counter is the crystal controlled clock (oscillator contained in the 8085), and the CPU can program the divide ratio, producing finely variable, highly accurate audio frequency signals. This is the source of the ID tones, "bleep", and AFSK

tones. It has 2K of program ROM, 256 bytes of RAM storage, three 8-bit output ports, and two 8-bit input ports. The control program is 1500 lines long.

There is no autopatch on 94, and there are no plans to install one. The conventional "beep" previously heard on 94 (and quite commonly heard on a variety of repeaters) has been supplanted by a 3-pitch "bleep" which is activated .75 seconds after the incoming carrier drops, indicating the time-out timer reset. The "bleep" duration is 190 ms.

In addition to the control functions, there are seven codes available for general use which are accessable via 146.34 MHz. For each of the codes, it is necesary for the first digit to be held at least one full second. Subsequent digits may be as short as 50 ms., and must be sent within 3 seconds of the preceding digit.

The repeater ID note is precisely 1000 Hz. The ID speed is precisely 19 wpm. When responding to user and control codes, the repeater responds with a 400 Hz cw note. During the processing of functions, the "bleep" is disabled; upon completion, it is reenabled.

Here are the various codes:

3#3 - Disables the blocking of touchtones. Any tones sent after the 3#3 before the carrier is dropped will not be blocked from repeating. Normally, upon recognition of valid tones, the repeater mutes them. This is done to protect the ears of those of us who monitor often. It is done on a tone by tone basis to

facilitate diagnosis of problems, since you can hear a short blip for every digit and can tell how many tones were sent. For those hams with selective call decoders, it is necessary for the tones to pass unimpeded, which is the reason for this function. If only short tones are required, the selective call function need not be used, as the repeater does not initiate tone blocking until a valid tone of about one second is received. This is to prevent blocking of voices.

4#4 - Touchtone Test. Any digits sent after the 4#4 before the carrier is dropped will be sent in Morse to tell the user what the repeater decoded the digits as. Any sequence up to 24 digits can be accomodated.

5#5 - If preceded by a 4#4 test, will repeat what the 4#4 sent. In general, it will repeat the last touchtone sequence sent to the processor for any command.

The repeater also incorporates a message relay service, or "bulletin board". Through the use of the 6#6, 7#7, and 8#8 functions the messages may be stored and retrieved. These functions require careful use, so be confident that the sequence is understood before activation.

6#6 - Message List. A list of all messages currently on file will be sent. Normally this will consist of a list of call letters preceded by a number. OST and other addresses may also be

included. If no messages are on file, the repeater says, "NONE".

There is a maximum of 12 possible messages, one for each touchtone digit; however, this is irrelevant because there is not enough room to store more than three to five messages of moderate length. There is space for 145 letters including messages and the programmable ID.

7#7 - Message Playback. To play a message back, send 7#7 and drop carrier. The repeater will send "NR?". Send the single touchtone digit which corresponds to the number which preceded the message desired on the Message List. The repeater will repeat the number, and follow with the message text. After a brief pause, the repeater will send, "ERASE?". If you desire to remove the message from the machine, send any single touchtone digit. The repeater will then send "OK" and the function has ended. If you choose to retain the message, simply kerchunk the repeater. Do not talk during this period, or the message may be accidentally erased. After the kerchunk, the repeater will respond with "SLOW?". If you wish to hear the message text repeated at 10 wpm (it is normally 19), send any single touchtone digit. The entire sequence will then repeat at that speed. If you do not wish to hear the message repeated slowly, kerchunk the repeater. It will answer "OK" and you are done.

If a message number is requested for which there is no message, the repeater will repeat the requested number and send "NONE". If no request is made, the repeater will just say "NONE".

8#8 - Message Store. To store a message, send 8#8 and drop carrier. The repeater will respond, "TO?". Using any of the 12 Touch Tone digits, send in Morse code the call of the station for whom you are leaving the message. Drop carrier, and the repeater asks, "MSG?". Send the message text followed by DE your call. Drop carrier, and the response will be "OK".

To check what was stored, utilize the 6#6 and 7#7 codes. If you make mistakes and the message is unrecognizable, immediately play it and erase it. If nothing is sent when code is expected, still play it back, as a message number may have already been reserved.

The processor can copy cw reliably over moderate speed variations, and will conform to your sending speed. Initially, send 10-15 wpm and you may slowly vary your speed as desired. For a sending error, if a string of dits is immediately appended to the character, it will be erased and a space will be inserted in its place. The primary problem often encountered is incorrect spacing between characters. If the space is too short, the characters will be run together, and if too long, a space will be inserted between letters in a word. It is best to relax between words, as only one space will be placed between words anyway.

The processor does not simply record what you send and play it back, it decodes the characters and sends them back perfectly. The machine will always send back perfect code - that may be good, as it can clean up sloppy sending, or it may be bad,

for letters may be run together perfectly. Remember - it knows how to copy code, and it only copies what you send, not what you intend to send (most human operators can figure out what you mean when you send NNMA, but that is what was sent, not CQ.)

The originating station should be responsible for removing an unclaimed message after a reasonable period of time. The machine cannot hold many messages, and it is not wise to keep it cluttered with unclaimed traffic.

9#9 - RTTY Test. A message will be sent in Baudot via AFSK of 2295-mark/2125-space tones. The audio frequencies and the baud rate (45.45) are crystal controlled and are highly accurate. This function may be used to calibrate RTTY gear.

NOTE: The functions 3#3 through 9#9 are intended for use by anyone, club member or not. (Although we certainly encourage interested amateurs to join the group.) When performing any of these commands, be certain to identify your station first. The functions are there to be used, but not abused. This is somewhat of an experiment in the hope that our repeater users will use these functions wisely. We hope to be able to continue this free access. Should it become necessary, any of the functions may be disabled by remote control. Please do not force us to deactivate them.

Controlling the System

(The following text is intended to be distributed to control operators of repeater systems utilizing the 8085 control system. The necessary codes will have to be changed.)

This control system may be accessed either on the repeater input frequency or the UHF control frequency. The control frequency is to be used in all situations for control functions unless a dire need necessitates otherwise. Any control sequences done on the repeater input frequency require the carrier to be dropped between each code sequence. Transmission on the control frequency removes the repeater input signal from the touchtone decoder, so simply transmitting on control will foil any touchtone functions attempted on the repeater. This is often sufficient to solve the problem of "squirrels'" shenanigans.

For any 3-digit touchtone sequence, the first tone in the code must be held for about one second. The successive two may be brief bursts. To perform any control function, it is first necessary to get into the command mode. To do so, the command code must be sent followed by the confirmation code. After attaining the command mode, the repeater will expect a single digit code. When that digit is received, the repeater will acknowledge with an "OK". For all single digit codes with the exception of 8 and 0, the function will be immediately executed and the command mode will be left.

Single digit codes 1 through 7 select the identification

which the repeater will use. Codes 1 through 4 will select one of the prestored ID messages for use, exclusively. Code 5 selects the fifth ID, which is programmable. Code 6 will cause IDs 1 through 4 to be cycled, and code 7 will cause IDs 1 through 5 to be cycled. Code * or # will reset the ID timer; after this reset, the repeater will ID upon the next kerchunk.

Single digit code 9 is the "panic" code. A master reset will be issued, placing the repeater in normal operation. Upon reset, ID #1 is exclusively utilized. The programmable ID and all stored messages will be cleared. A power failure reset will have the same effect.

To program the fifth ID message, code 8 is used. After sending "OK", the repeater will wait for a touchtone sequence to load the message. The message is entered on a character by character basis. A dit is entered as 0, and a dah as 1. When the letter is completed, it is entered with a 2. When the entire message is loaded, a 3 exits the ID load mode and also the command mode, acknowledged with an "OK". To load the message, "BOO", send the following: 123, 456, 8, 1, 0, 0, 0, 2, 1, 1, 1, 2, 1, 1, 1, 2, 3. If a mistake is made before entering a letter with the 2, it may be deleted with a 4. Sending 4 wipes out any dits or dahs sent for the current character. If a character is entered with a 2 without any preceding dits or dahs (0s or 1s), a space will be entered. It is generally best to place a space at the beginning and at the end of the message. The leading space gives the repeater transmitter time to activate before the CW tones begin, and the trailing space makes it sound better. To

enter " BOO " with the leading and trailing spaces, send: 123, 456, 8, 2, 1, 0, 0, 0, 2, 1, 1, 1, 2, 1, 1, 1, 2, 2, 3.

Single digit 0 is the most powerful code. This places the system into the output mode. The output mode permits changing the output port levels. All ports are normally 0. For the open collector outputs, this corresponds to an open circuit. When a bit is changed to 1, the open collector outputs will supply a ground. The functions are grouped into several ports. To activate a specific bit, the port number and bit number must be given. The output level desired (0 or 1) must also be specified. When in output mode, the controller expects 3-digit sequences of the form XYZ, where X is the port nummber, Y is the bit number, and Z is the output level. Valid requests will be acknowledged with an "OK". Invalid requests are ignored. Any number of outputs can be made while in the output mode. To exit the output and command modes, send a #. The repeater will send "OK" to verify the exit.

Port 0 is a dummy port. This means that there is no physical port, but there are internal switches which this port affects. To disable the 8#8 function, send: 123, 456, 0, 081, #. To restore it, send: 123, 456, 0, 080, #. To turn the repeater off and disable the bleep, send: 123, 456, 0, 351, 381, #.

Always be certain to exit the command mode before terminating a control session. If the repeater bleeps or IDs, all is well, because these functions are inhibited during control sequences.

Control Codes:

123 - Initiate command mode

456 - Confirm command mode

Command Mode Single Digit Codes:

1 - ID #1: "DE WA3KOK/RPT BARC"
2 - ID #2: "DE WA3KOK/R BALTIMORE"
3 - ID #3: "73 DE WA3KOK/R"
4 - ID #4: "DE WA3KOK/R BALTO ARC"
5 - ID #5: Programmable
6 - Rotate IDs 1 through 4
7 - Rotate IDs 1 through 5
8 - Load ID #5:
0 - Dit
1 - Dah
2 - End character
3 - End message
4 - Delete current character
9 - Master reset:
Clears ID #5 and all messages
Selects ID #1
0 - Output mode (XYZ)
Port X, bit Y, level Z
Exit with #
*, # - Reset ID timer

Output Ports:

Port #0
1 - Spare internal switch
2 - Spare internal switch
3 - Disable 9#9
4 - Disable 4#4
5 - Disable 5#5
6 - Disable 6#6
7 - Disable 7#7
8 - Disable 8#8

Port #1
1 - Disable Rx 1
2 - Disable Rx 2
3 - Disable Rx 3
4 - Disable Rx 4
5 - Disable Rx 5
6 - Disable Rx 6
7 - Spare
8 - Spare

Port #2
1 - Select Rx 1
2 - Select Rx 2
3 - Select Rx 3
4 - Select Rx 4
5 - Select Rx 5
6 - Select Rx 6
7 - Spare
8 - Spare

Port #3
1 - Do not use (block)
2 - 146.34 touchtone disable
3 - Spare
4 - Disable time-out timer
5 - 146.94 transmitter off
6 - Do not use (PTT)
7 - Spare internal switch
8 - Disable bleep

Port #4 - Do not use (timer low)

Port #5 - Do not use (timer high)

Port #6 - Do not use (command register)

Examples: (underline means hold 1 second)

123 456 3 - Select ID #3

123 456 0 351 # - 146.94 off

123 456 0 111 # - Disable Rx 1

123 456 0 231 # - Select Rx 3

123 456 9 - Master reset

123 456 0 031 061 # - Disable 9#9, 6#6

123 456 0 381 # - Disable bleep

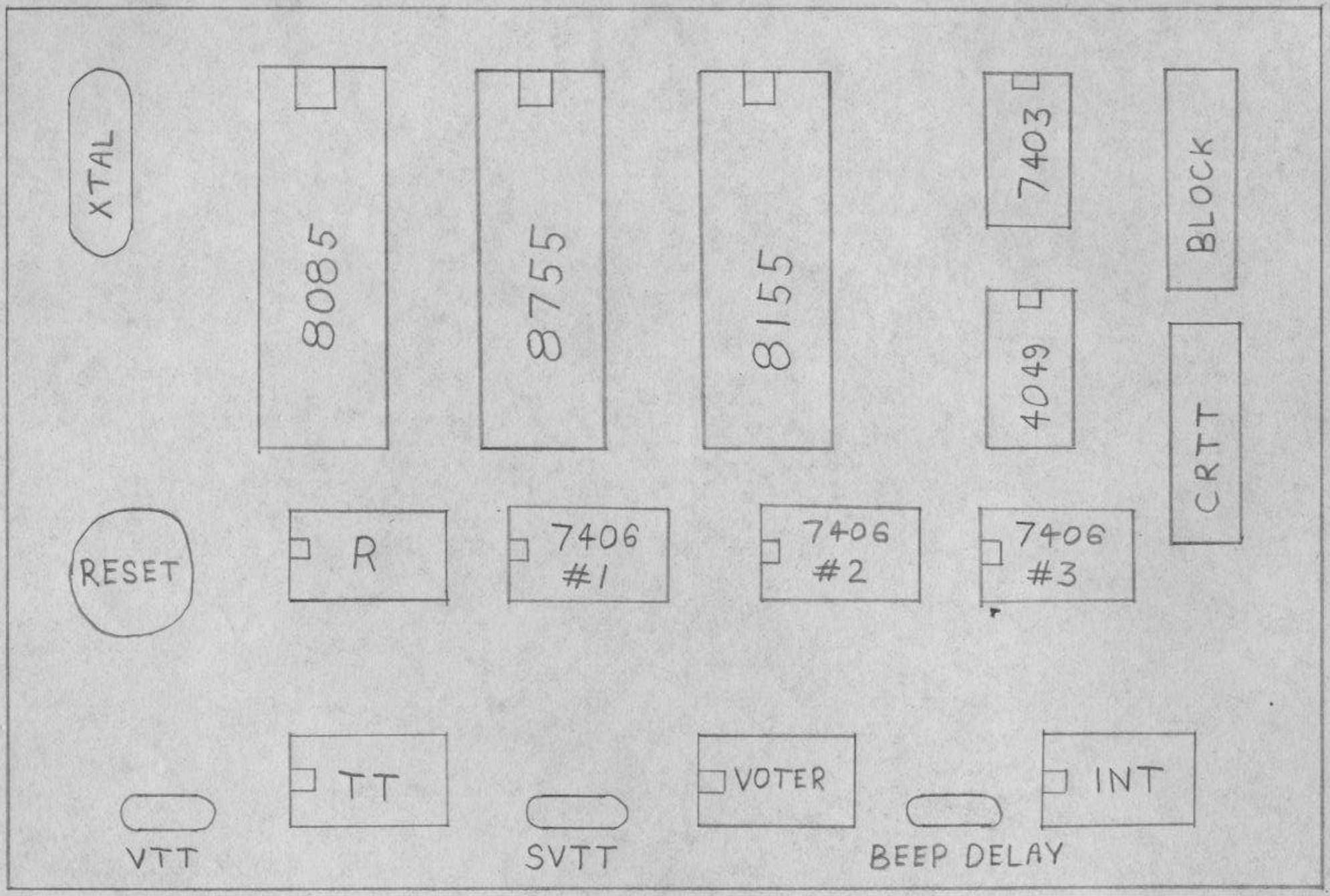


FIGURE 1 - BOARD LAYOUT

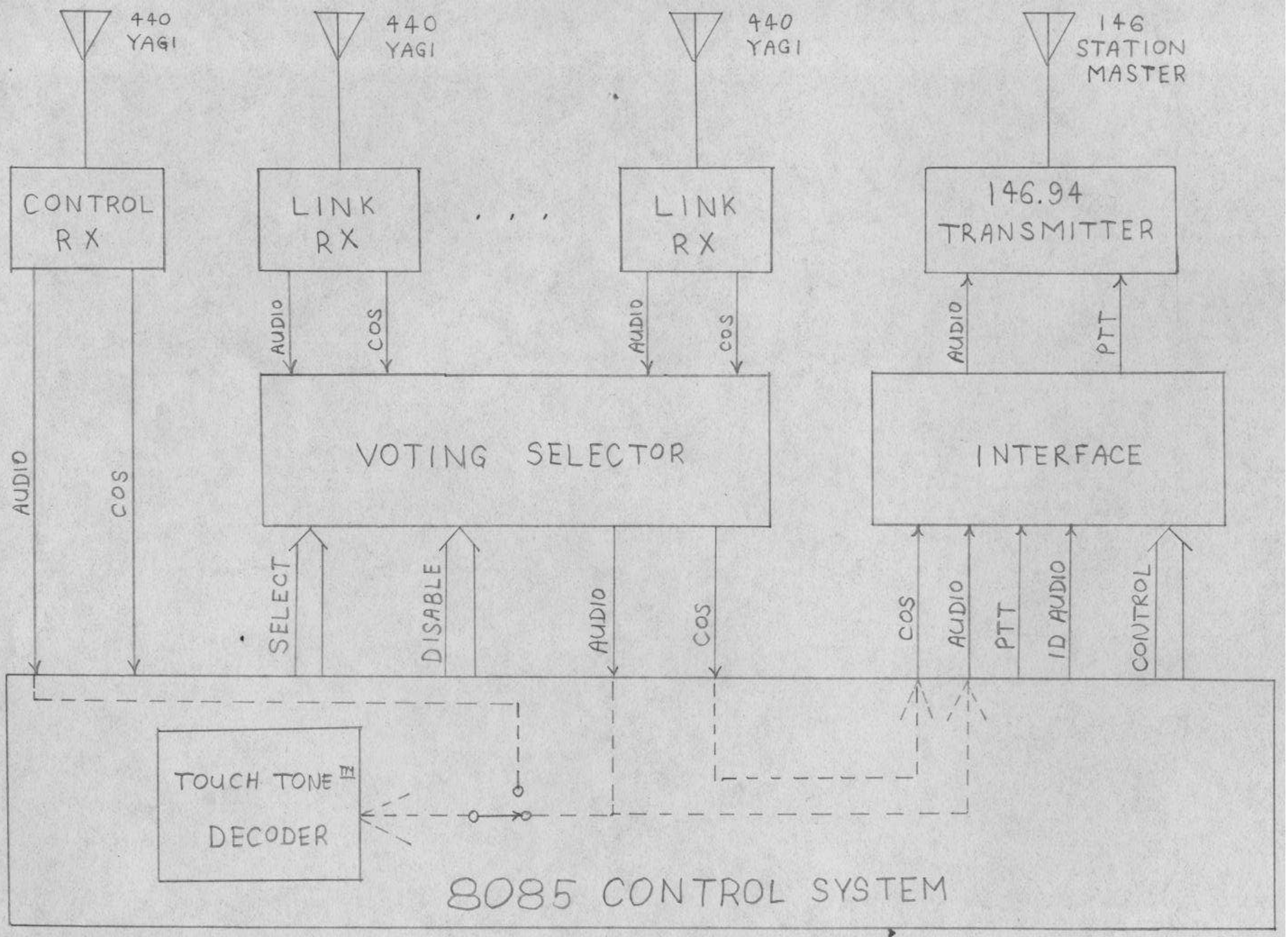


FIGURE 2 - REPEATER BLOCK DIAGRAM

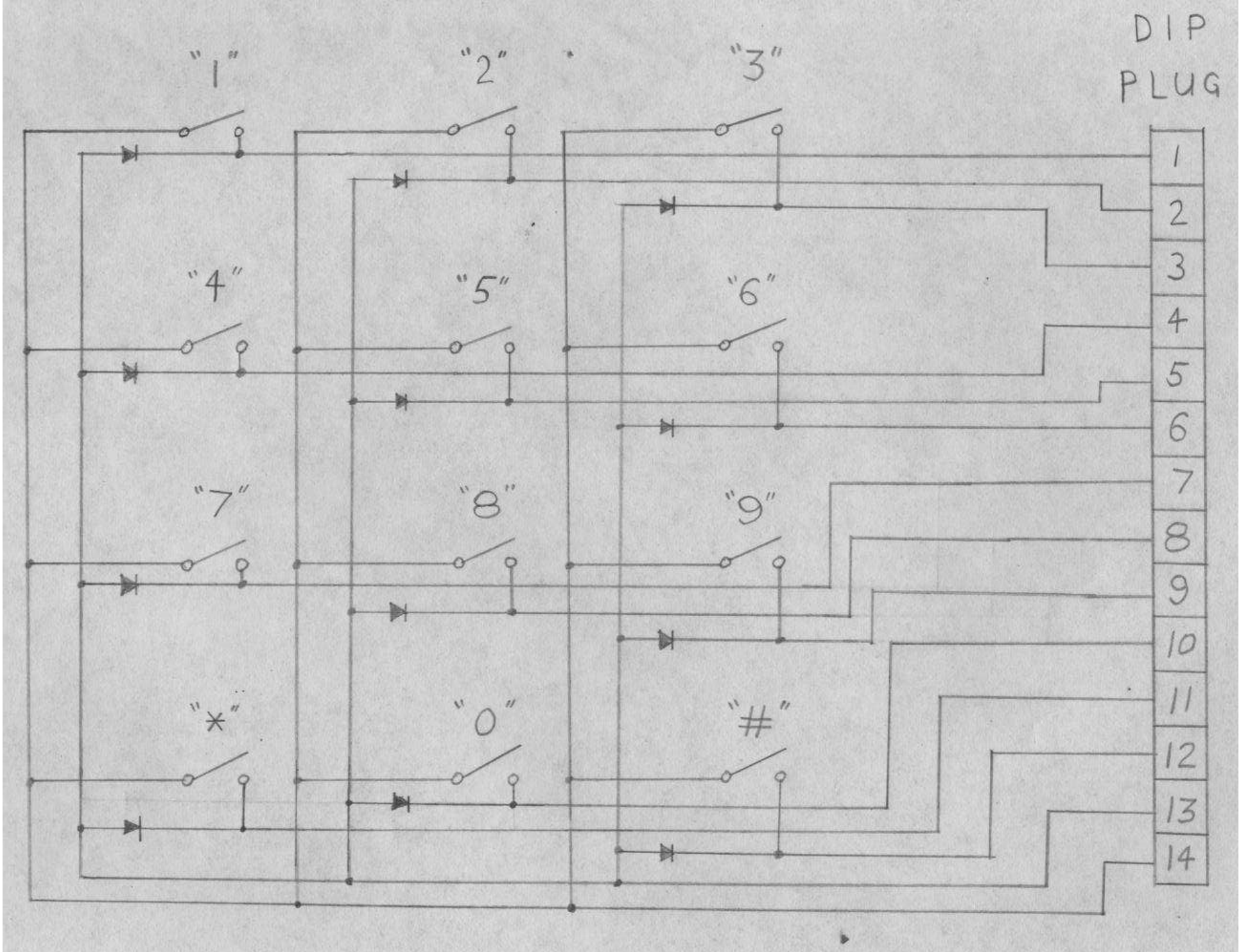


FIGURE 3 - TEST KEYPAD

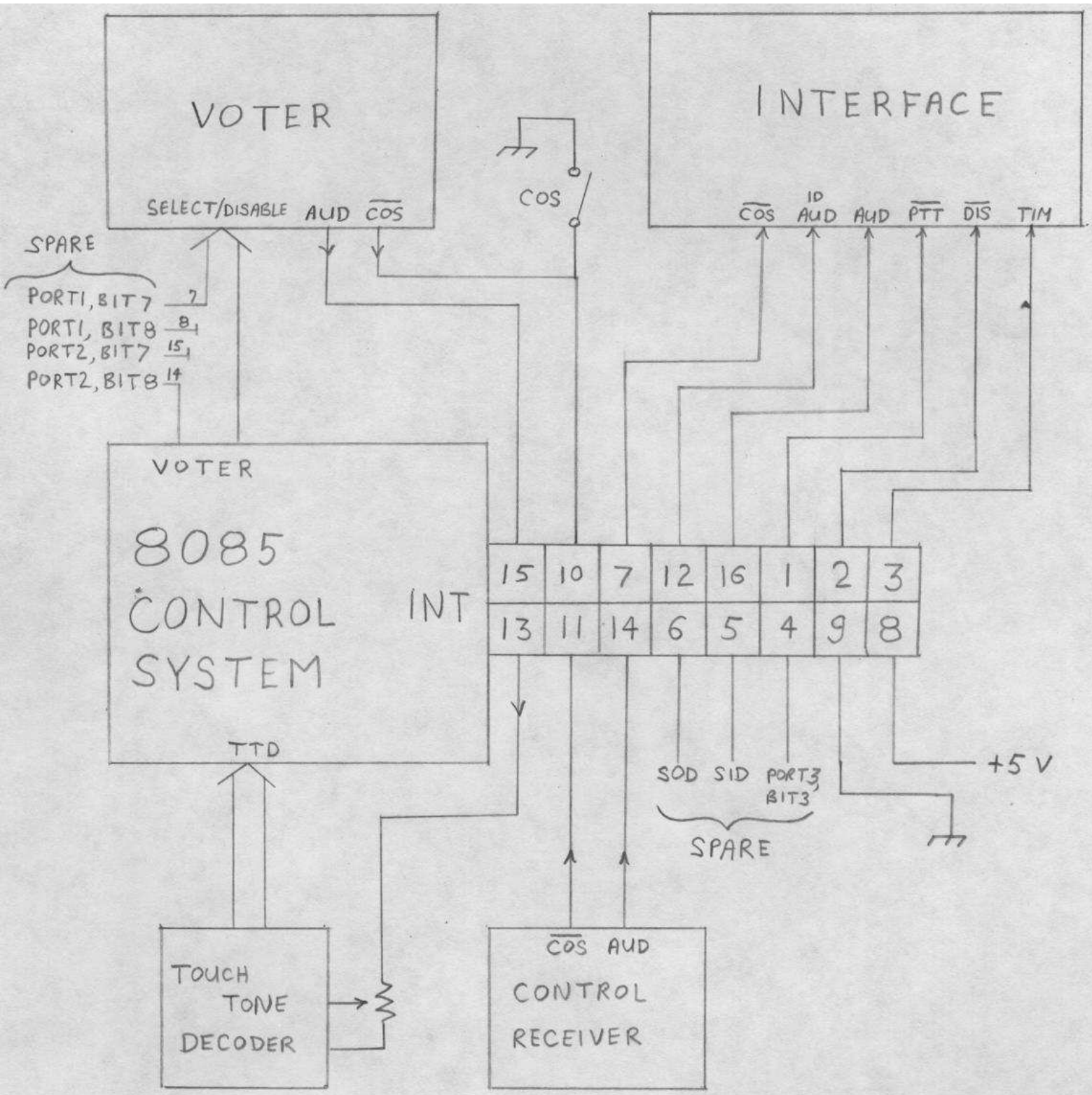


FIGURE 4 - INTERCONNECTIONS

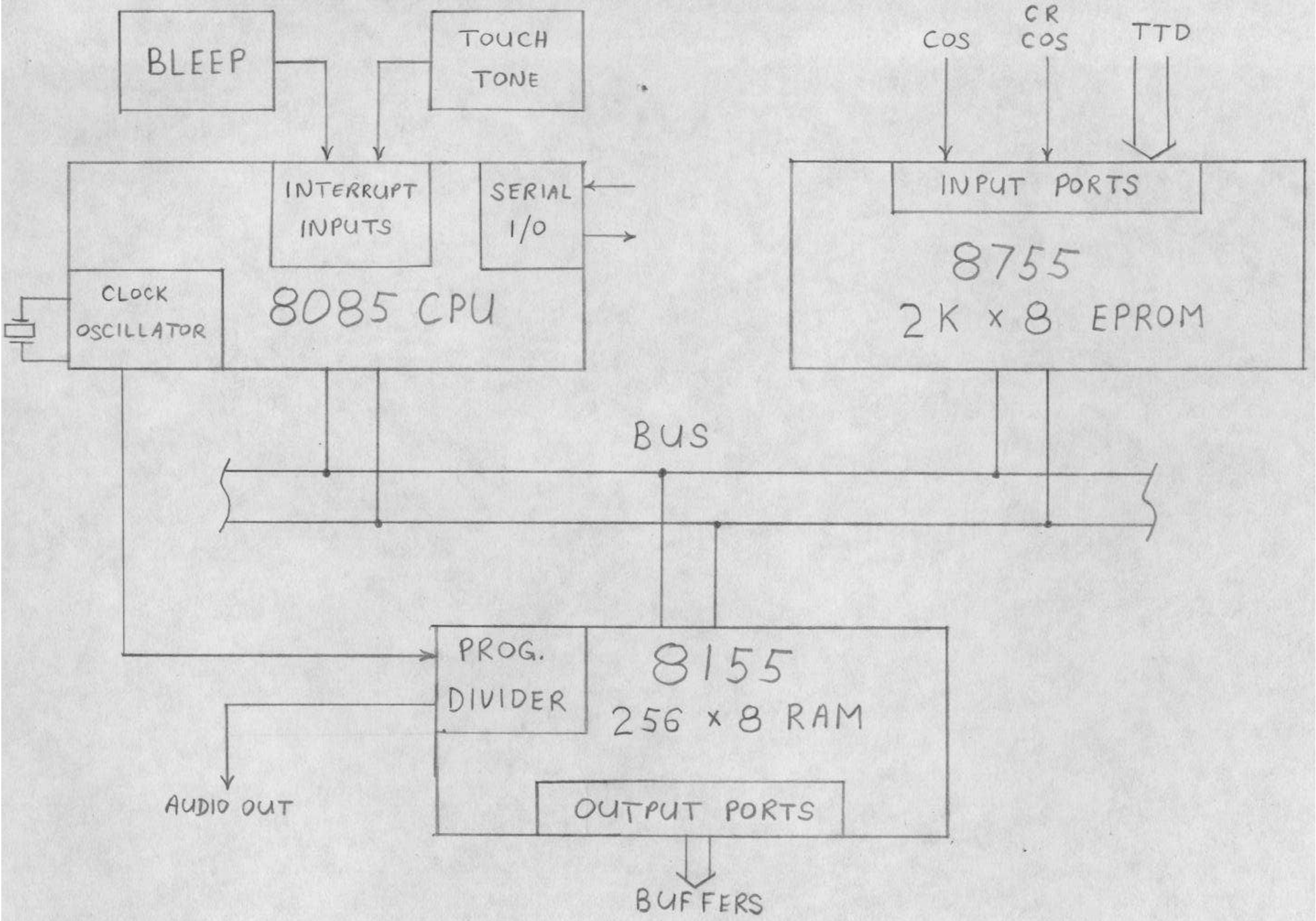


FIGURE 5 - MICROPROCESSOR CHIPS

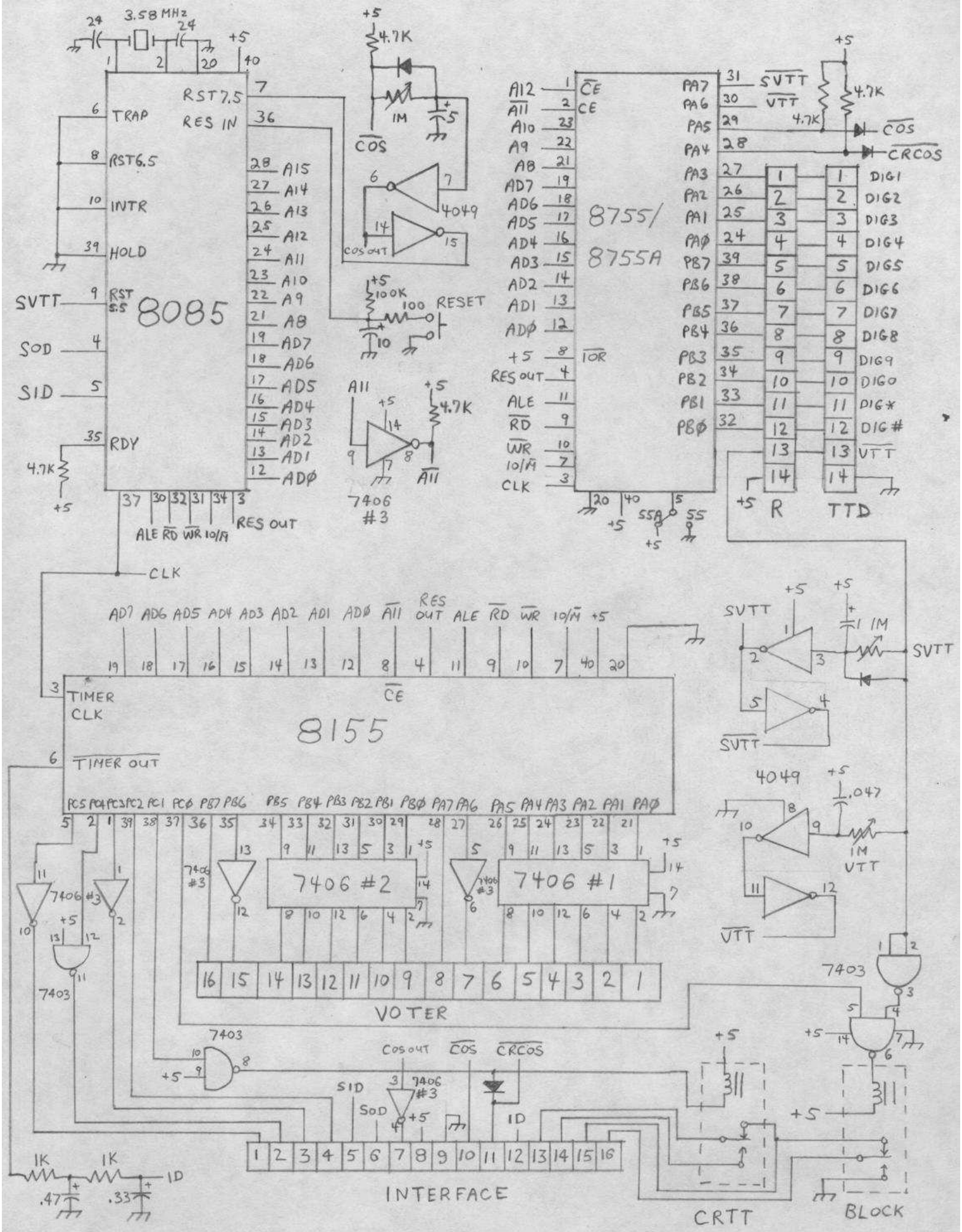


FIGURE 6 - CONTROLLER SCHEMATIC

Touchtone Decoder Connector (TTD):
1 - digit 1
2 - digit 2
3 - digit 3
4 - digit 4
5 - digit 5
6 - digit 6
7 - digit 7
8 - digit 8
9 - digit 9
10 - digit 0
11 - digit *
12 - digit #
13 - valid touchtone (VTT)
14 - common (ground)

Voter Connector:
1 - disable receiver 1
2 - disable receiver 2
3 - disable receiver 3
4 - disable receiver 4
5 - disable receiver 5
6 - disable receiver 6
7 - spare output - port 1, bit 7 (open collector)
8 - spare output - port 1, bit 8 (TTL level)
9 - select receiver 1
10 - select receiver 2
11 - select receiver 3
12 - select receiver 4
13 - select receiver 5
14 - select receiver 6
15 - spare output - port 2, bit 7 (open collector)
16 - spare output - port 2, bit 8 (TTL level)

Interface Connector (INT):
1 - PTT input on interface (ground keys transmitter)
2 - repeater disable (ground turns transmitter off)
3 - timer (ground disables time-out timer)
4 - spare output - port 3, bit 3 (TTL level)
5 - spare input (SID, requires additional programming)
6 - spare output (SOD, requires additional programming)
7 - COS input on interface (ground indicates receiver unsquelch)
8 - +5 volt external power supply (700 ma. maximum)
9 - ground
10 - COS output from voter or receiver (ground indicates unsquelch)
11 - COS output from control receiver (ground indicates unsquelch)
12 - ID audio input on interface (level adjust in interface required)
13 - touchtone decoder audio input
14 - control receiver audio output
15 - audio output from voter or receiver
16 - repeat audio input on interface

Dummy Interface Socket (to place repeater on air without controller):
7 - 10 (COS)
15 - 16 (audio)

Figure 7 - Connector Pinouts

INPUT:

PORT1: 00H

BIT 7							BIT Ø
$\overline{SVTT}$	$\overline{VTT}$	$\overline{COS}$	$\overline{CRCOS}$	DIG 1	DIG 2	DIG 3	DIG 4

PORT2: 01H

DIG 5	DIG 6	DIG 7	DIG 8	DIG 9	DIG Ø	DIG *	DIG #

OUTPUT:

DDR1: 02H FOR PORT1 - ALL 0 FOR INPUT

DDR2: 03H FOR PORT2 - ALL 0 FOR INPUT

PORTØ: DUMMY (OUTØM) (Ø)

DIS 8#8	DIS 7#7	DIS 6#6	DIS 5#5	DIS 4#4	DIS 9#9	SPARE	SPARE

OPOR1: 09H (OUT1M) (1)

DISABLE

SPARE	SPARE	RX 6	RX 5	RX 4	RX 3	RX 2	RX 1

OPOR2: 0AH (OUT2M) (2)

SELECT

SPARE	SPARE	RX 6	RX 5	RX 4	RX 3	RX 2	RX 1

PORT3: 0BH (OUT3M) (3)

DIS BEEP	SPARE	PTT	RPT OFF	DIS TIMER	SPARE	DIS TT	BLOCK

DUMMY

TIML: 0CH (TIMLM) (4)

TIMH: 0DH (TIMHM) (5)

CSR: 08H (CSRM) (6)

FIGURE 8 - PORT ASSIGNMENTS

FIGURE 9 - FOREGROUND PROGRAM

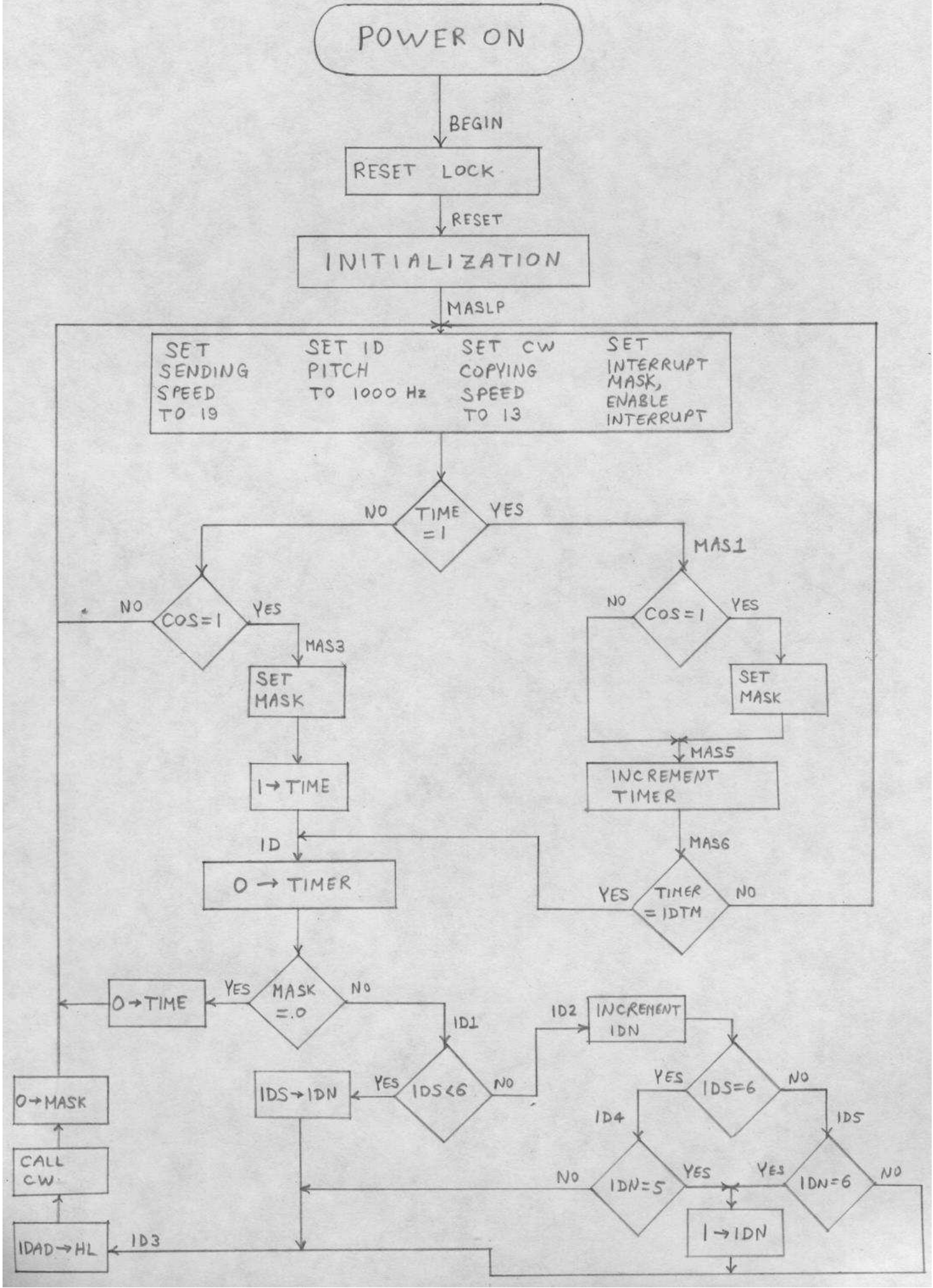


FIGURE 10 - BEEP INTERRUPT PROGRAM

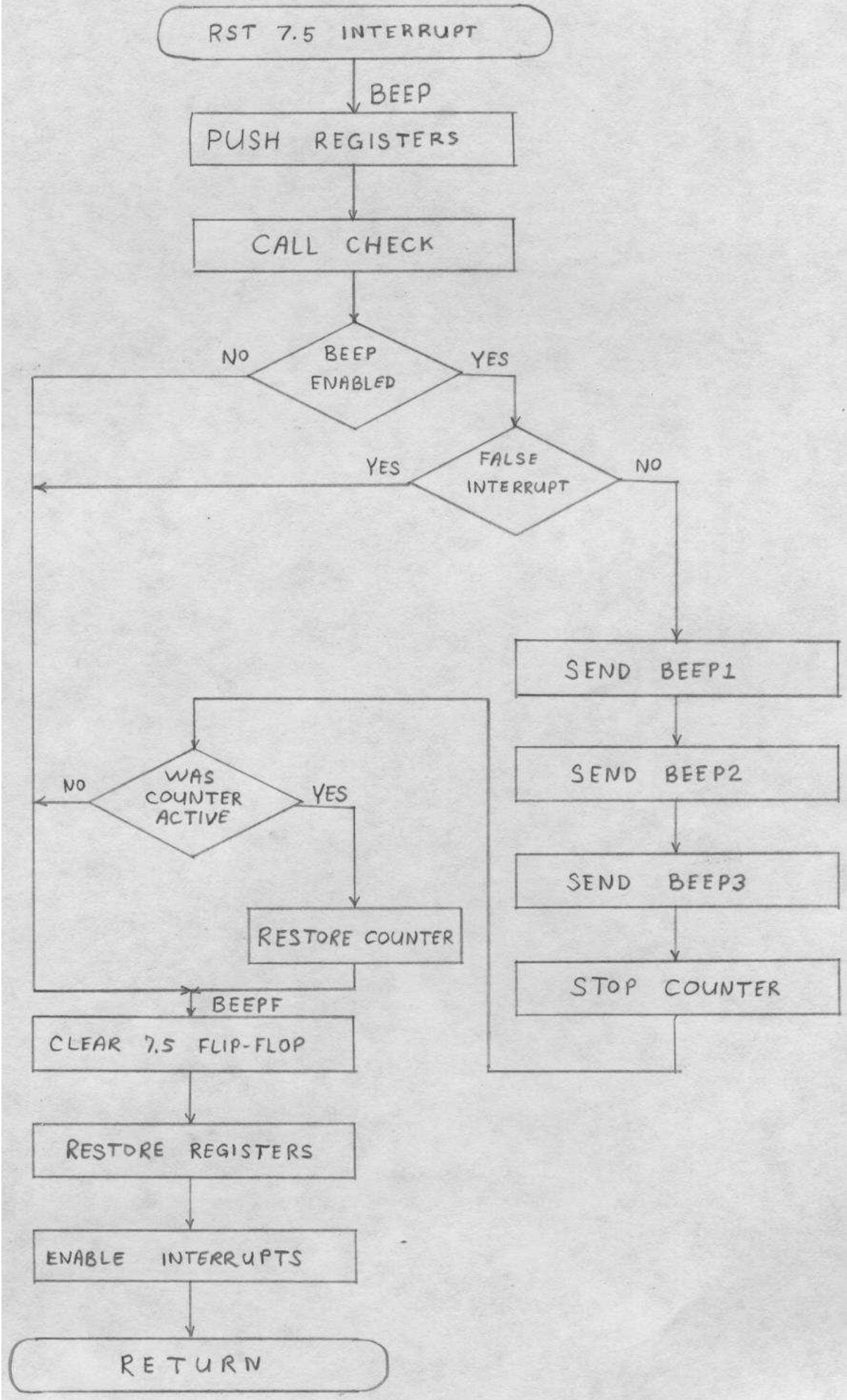
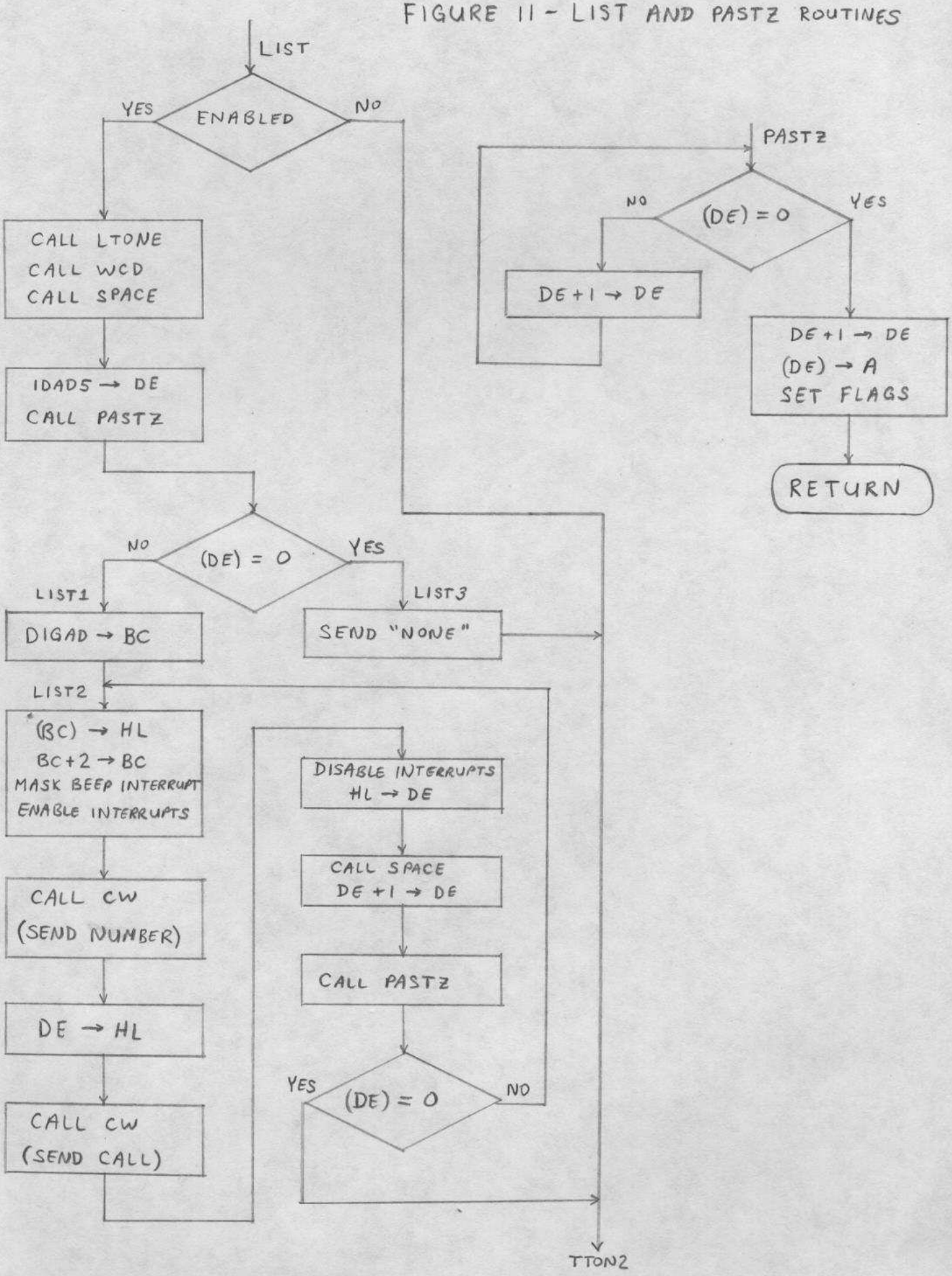


FIGURE 11 - LIST AND PASTZ ROUTINES



MESSAGE STORAGE FORMAT:

IDADS: (IDS)(00)(CALL 1)(00)(MSG 1)(00)(CALL 2)(00)(MSG 2)(00) · · · (CALL N)(00)(MSG N)(00)(00)

FIGURE 12 - PLAY AND INDIG ROUTINES

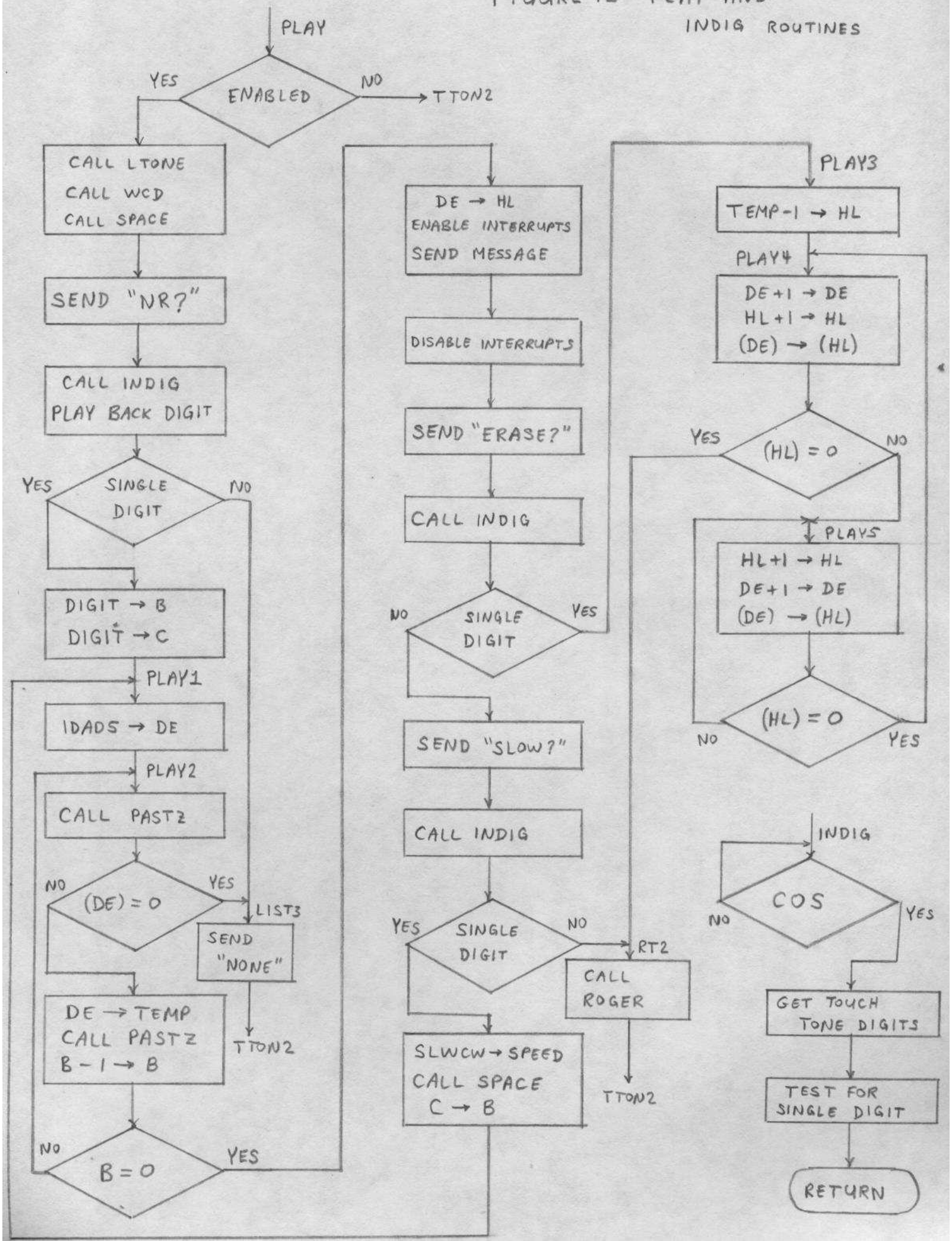


FIGURE 13 - STORE ROUTINE

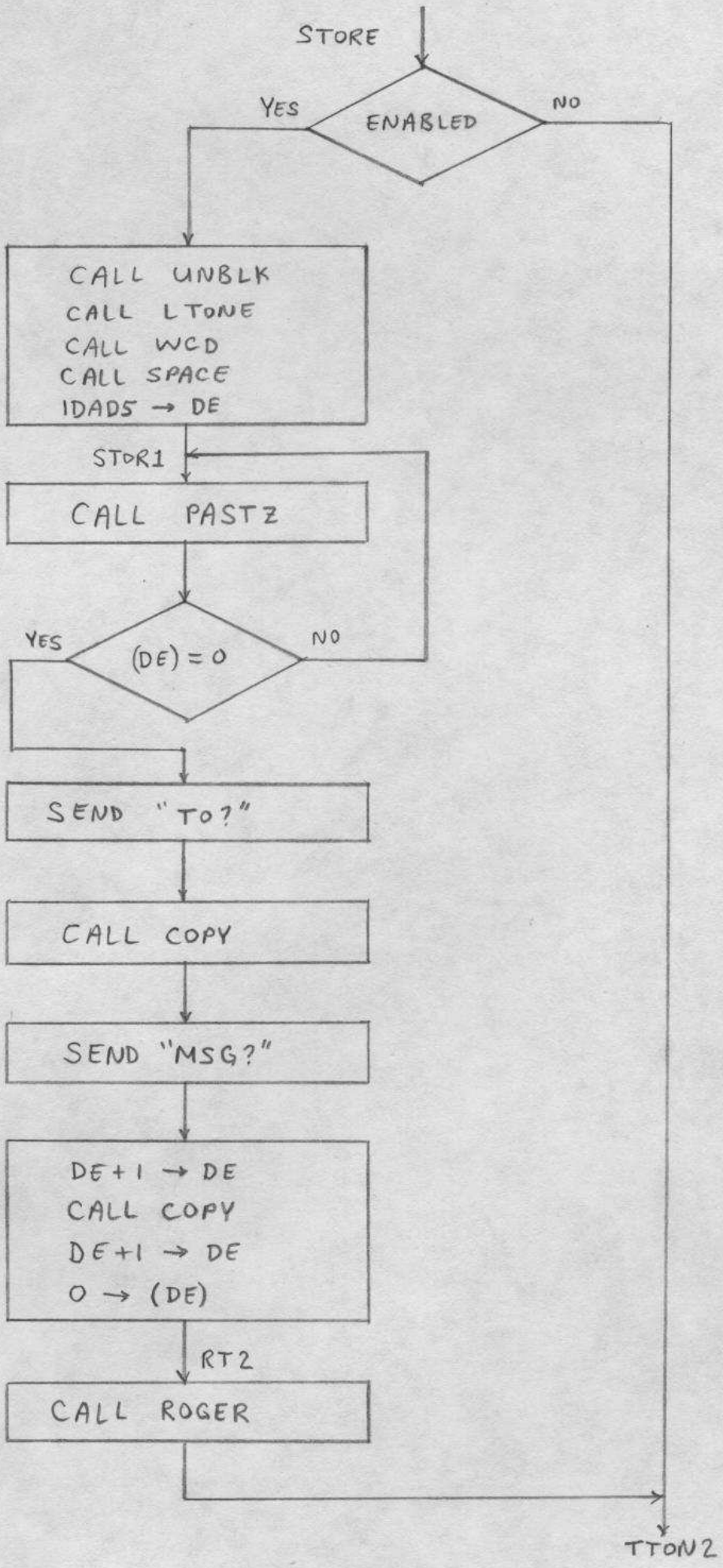


FIGURE 14 - COPY AND TICK1 ROUTINES

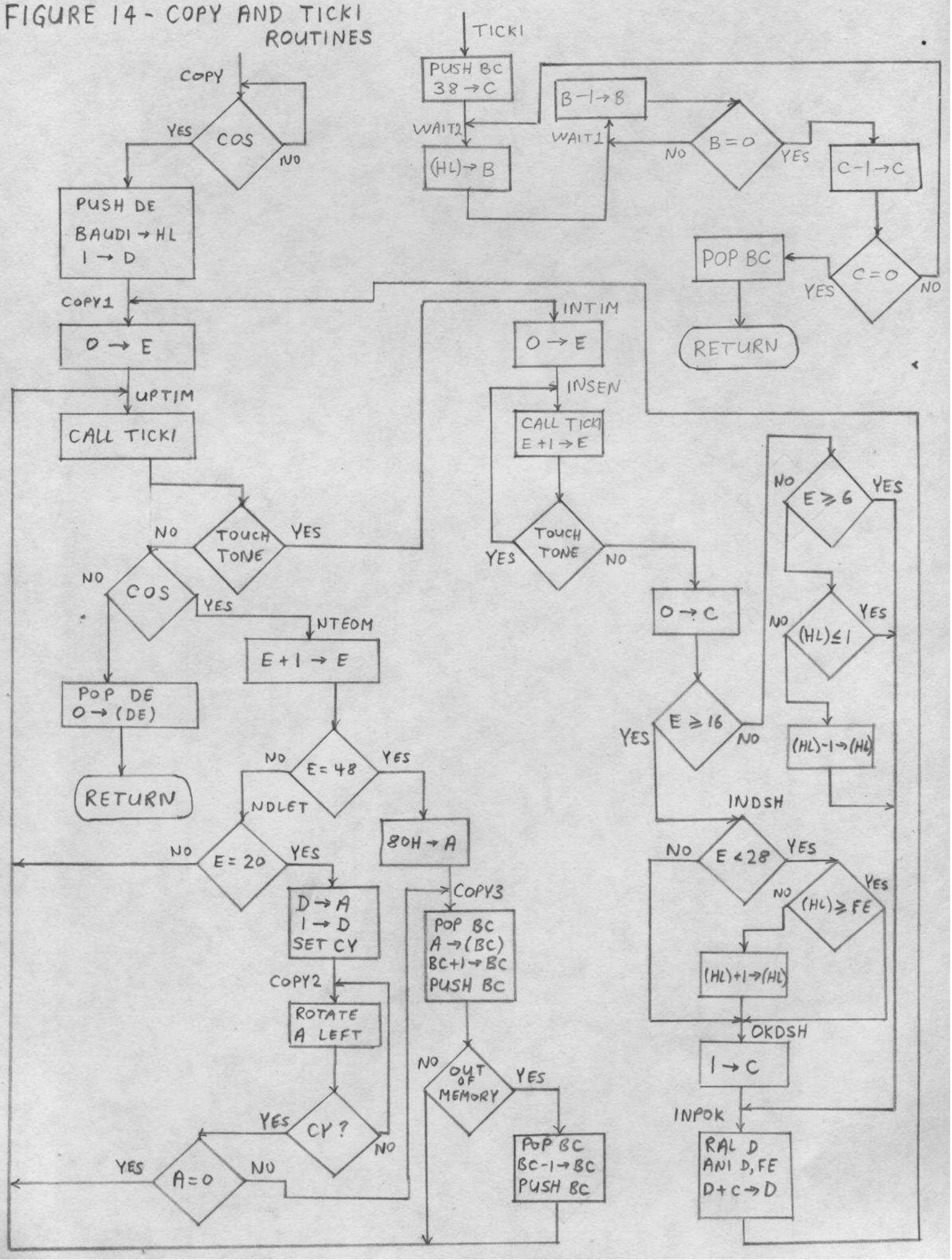


FIGURE 15 - RTTY ROUTINE

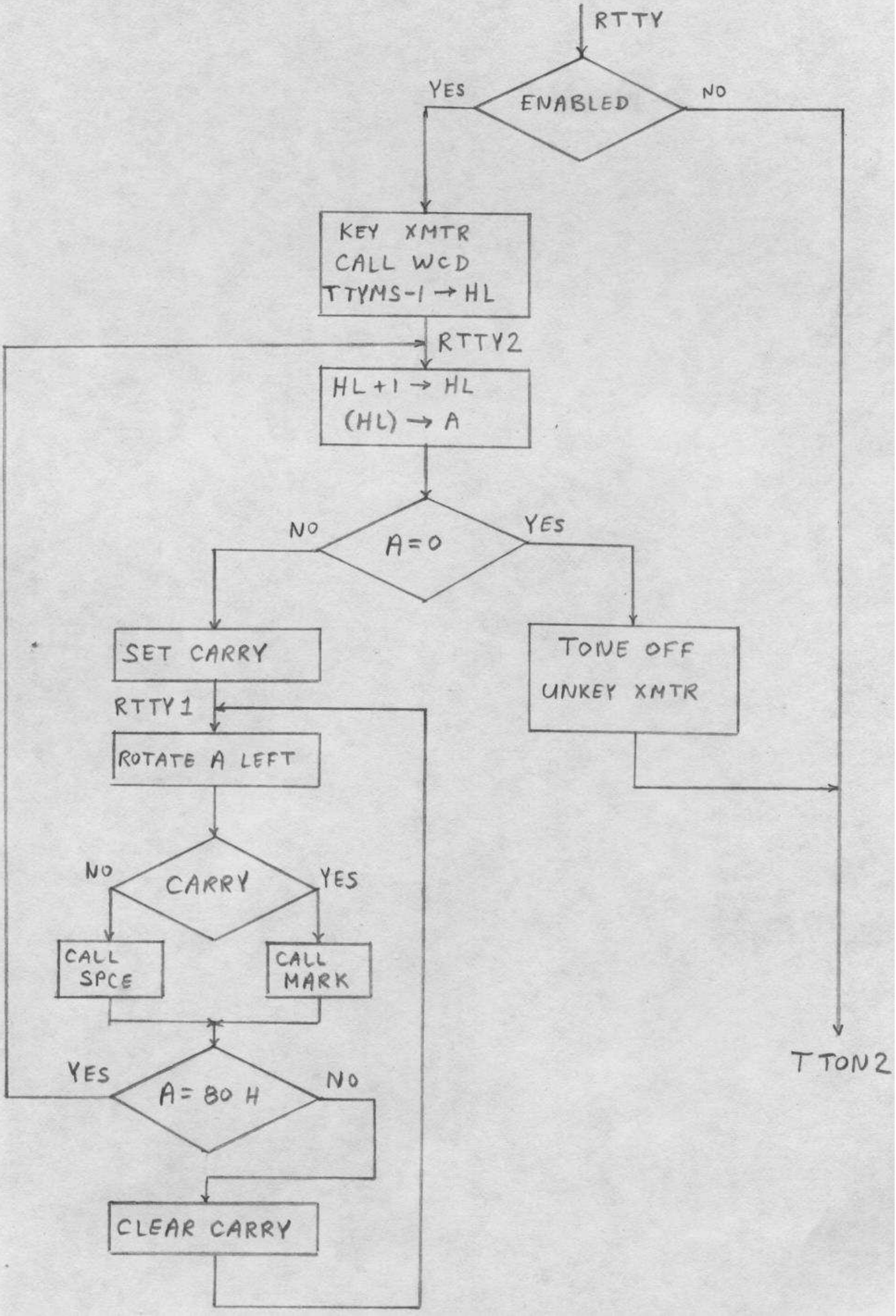
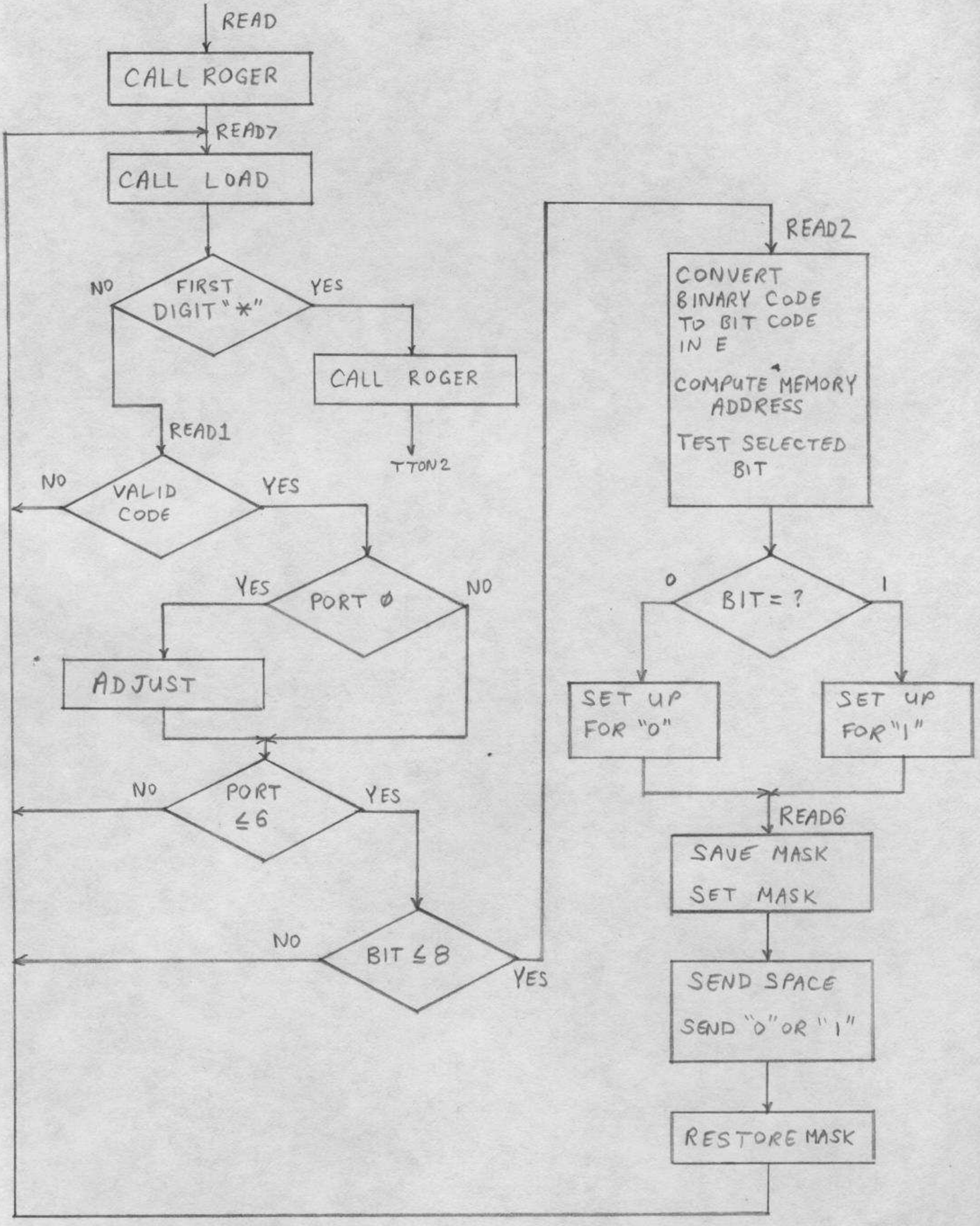


FIGURE 16 - STATUS READ ROUTINE



```
0001 0000                       ;REPEATER CONTROL SYSTEM MONITOR PROGRAM
0002 0000                       ;
0003 0000                       ;FOR USE WITH AN 8085 CONTROLLER
0004 0000                       ;WITH I/O PORTS
0005 0000                       ;AND THE NECESSARY EXTERNAL HARDWARE
0006 0000                       ;INCLUDING A TOUCH TONE (R) DECODER
0007 0000                       ;
0008 0000                       ;
0009 0000                       ;
0010 0000                       ;VERSION 85-1.7
0011 0000                       ;
0012 0000                       ;
0013 0000                       ;NOVEMBER 1977, ROBERT GLASER N3IC
0014 0000                       ;
0015 0000                       ;85- SERIES FOR 8085/8755/8155 SYSTEM
0016 0000                       ;AND INCLUDES MESSAGE STORAGE FUNCTIONS
0017 0000                       ;
0018 0000                       ;MODIFIED NOVEMBER 1979
0019 0000                       ;
0020 0000                       ;
0021 0000                       PSW:    EQU     6
0022 0000                       SP:     EQU     6
0023 0000                       SIM:    EQU     30H
0024 0000                       PORT1:  EQU     0
0025 0000                       PORT2:  EQU     1
0026 0000                       PORT3:  EQU     0BH
0027 0000                       OPOR1:  EQU     9
0028 0000                       OPOR2:  EQU     0AH
0029 0000                       DDR1:   EQU     2
0030 0000                       DDR2:   EQU     3
0031 0000                       CSR:    EQU     8
0032 0000                       TIML:   EQU     0CH
0033 0000                       TIMH:   EQU     0DH
0034 0000                       CWSPD:  EQU     4711    ;19 WPM
0035 0000                       SLWCW:  EQU     8951    ;10 WPM
0036 0000                       RTYSP:  EQU     1641    ;45.45 BAUD
0037 0000                       SPSET:  EQU     40
0038 0000                       IDTM0:  EQU     0
0039 0000                       IDTM1:  EQU     0
0040 0000                       IDTM2:  EQU     24
0041 0000                       CWTON:  EQU     1790+16384  ;1000 HZ
0042 0000                       CWLTN:  EQU     4474+16384  ;400 HZ
0043 0000                       MRKTN:  EQU     780+16384  ;2295 HZ
0044 0000                       SPCTN:  EQU     842+16384  ;2125 HZ
0045 0000                       BEEP1:  EQU     4068+16384  ;440 HZ
0046 0000                       BEEP2:  EQU     2712+16384  ;660 HZ
0047 0000                       BEEP3:  EQU     2034+16384  ;880 HZ
0048 0000                       ;
0049 0000                       ;
0050 0000                       ;
0051 0000                               ORG     0
0052 0000                       ;INITIALIZATION PROCEDURE
0053 0000                       ;
0054 0000                       ;
```