MAE/N-TRAK Computer Communications

by J. Gurian and D. Simon

ABSTRACT

Hardware and software for implementing a communications network between track computers and the master controller of the MAE/N-TRAK system are described. The basic requirements for such a network are listed, as well as why such a network is desirable. The considerations that went into the design of the system are outlined.

42

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In order for the track computers to achieve their fullest potential in the MAE/N-TRAK system, they should be able to both receive instructions from the master control unit and send return information back to the master controller. The following scenario illustrates the use of such a system.

> An automatic box car loading/unloading dock is controlled by a track computer. Upon receipt of the command 'unload the 4th car on locomotive number 5' from the master, the track computer would begin its operation sequence. All trains would be passed directly through the block until locomotive number 5 entered. At this point, a message from the track computer would be sent to the master requesting that locomotive number 5 be set to speed level 2. At this slow speed, each car on the train could be identified using optosensors, and when car 4 was properly aligned, power would be locally killed. The car would now be unloaded. Upon completion of this task, power would be resumed to the track, and the track computer would request that locomotive number 5 be reset to its former speed.

In order for the system described to be implemented, several requirements are necessary. First, bidirectional communication must exist between the master and track computers. Second, the response time of the master to a request by the track computer must be sufficiently fast to allow proper operation of the accessory

The communications system developed should be easily used by the programmer of the track computer. In addition it should appear transparent to the user, aside from several interface buffers. The hardware needed should integrate easily into the system, and should not require any change in operating protocol or operational parameters. (ie. change in track voltage or modification of already existing FSK communications system)

Summary of System Requirements

1. The transmission of controller data to the locomotives should not be affected by the return information.
2. The service routine, required for operation of the communications system, should run in 1K of ROM and may use no more than 256 bytes (1 page) of RAM.
3. The system should only make use of VIA port B on the track computers, as port A should remain available for use by the user foreground routine.
4. It should communicate with the foreground routine through the following RAM locations:

   RDATA - data received from controller
   WDATAH - high byte of return data
   WDATAL - low byte of return data
   DFLAG - set to indicate data to send by track computer
5. It should work with an inverted track signal.
6. It should provide outputs to indicate when the data and power frames are present.

Because of the system requirements, amplitude modulation was chosen as a scheme to transmit data from the track computers to the master. It was found that by modulating the track waveform in a manner that was synchronized with the incoming FSK data, it was possible to recover both the frequency shift keyed signal, as well as the AM signal.

2. SYSTEM ORGANIZATION

2.1 STRUCTURE OF DATA

There are two types of data transmitted over the track. 'Controller data' is the frequency shift keyed information sent by the master to the track computers. 'Return data' is the amplitude modulated information which is sent from the track computers to the master

2.1.1 Controller Data

Controller data is organized in eight four bit 'nibbles.' In a data frame directed to a track computer, these nibbles contain the following information:

NIBBLE 0: Contains '0100', where '01' are the start bits, and the next two bits are always zero in an accessory frame they identify the message frame number in a locomotive frame).

NIBBLE 1: '0000' always zeroes in an accessory frame.

NIBBLE 2: '0000' always zeroes in an accessory frame.

NIBBLE 3: 'nnnn' - Block number; ID of receiving track computer.

NIBBLE 4: 'hhhh' - High order bits of task code.

NIBBLE 5: '1111' - Low order bits of task code.

NIBBLE 6: 'pppp' - Parity bits.

NIBBLE 7: '0100' - Stop bits - always the same.

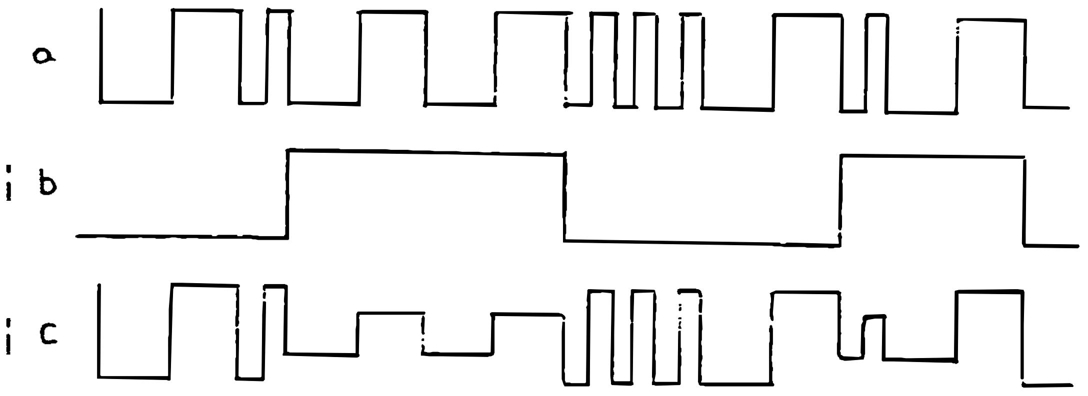
The least significant bit of each nibble is sent first, starting with nibble 0, and ending with nibble 7. (ie. the first four bits sent are 0010)

2.1.2

Return Data

The return information is sent by amplitude modulating the track waveform during the data frame (see Figure 1). The return information consists of 16 bits of data which are amplitude modulated in phase with the incoming FSK controller data. In this scheme, a binary 1 is represented by the attenuated signal level, while a binary 0 is represented by the normal signal level. Note that each bit of return data has the same period as two bits of controller data (see hardware section for more information).

The return data is not as of yet organized in any special configuration, however, once the system becomes more fully developed, a protocol can be added to the return data.



** Not Implemented

Figure 1: Structure and Synchronization of Return Data

a. Incoming controller data as it appears in data frame

b. Return data in phase with the incoming signal

c. Resulting track waveform after return data has been amplitude modulated on top of incoming signal

TRACK WAVEFORM STRUCTURE

The track waveform, as explained in previous chapters, consists of a repeating sequence of two power cycles, and data cycle. Our routine divides this sequence into five frames (see Figure 2). The current frame index is stored in location franum for synchronization purposes.

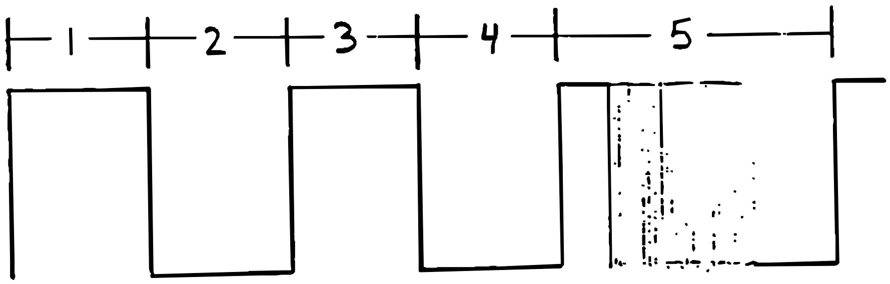
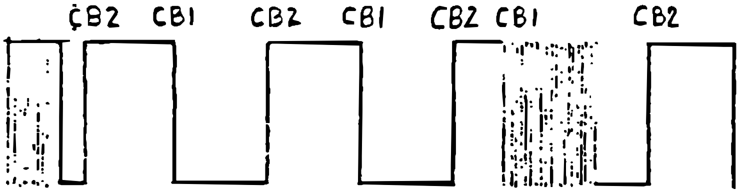


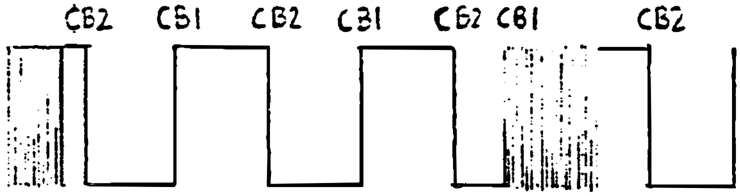
Figure 2: Five Frame Structure of Track Waveform

2.3 CB1 AND CB2 INTERRUPTS

In order to drive the communications routine, the track computer must be able to follow the transitions of the track waveform. This is done by interrupting the processor every time a transition occurs. It is also necessary to differentiate between negative and positive going transitions. our scheme, the CB1 and CB2 control ports on the 6522 VIA, provide a means to sync with the track transitions. The control ports are set to interrupt the processor on opposite transitions of the track waveform; i.e. if the CB1 line terrupts on the rising edge, the CB2 will interrupt on the falling edge (see figure 3). The CB1 and CB2 control register (PCR) is set so that a CB1 interrupt will always occur at the beginning of the high frequency portion of data frame regardless of the track polarity (see figure 3).



Interrupt configuration when track polarity is positive



Interrupt configuration when track polarity is negative

Figure 3: CB1 and CB2 Interrupt Structure

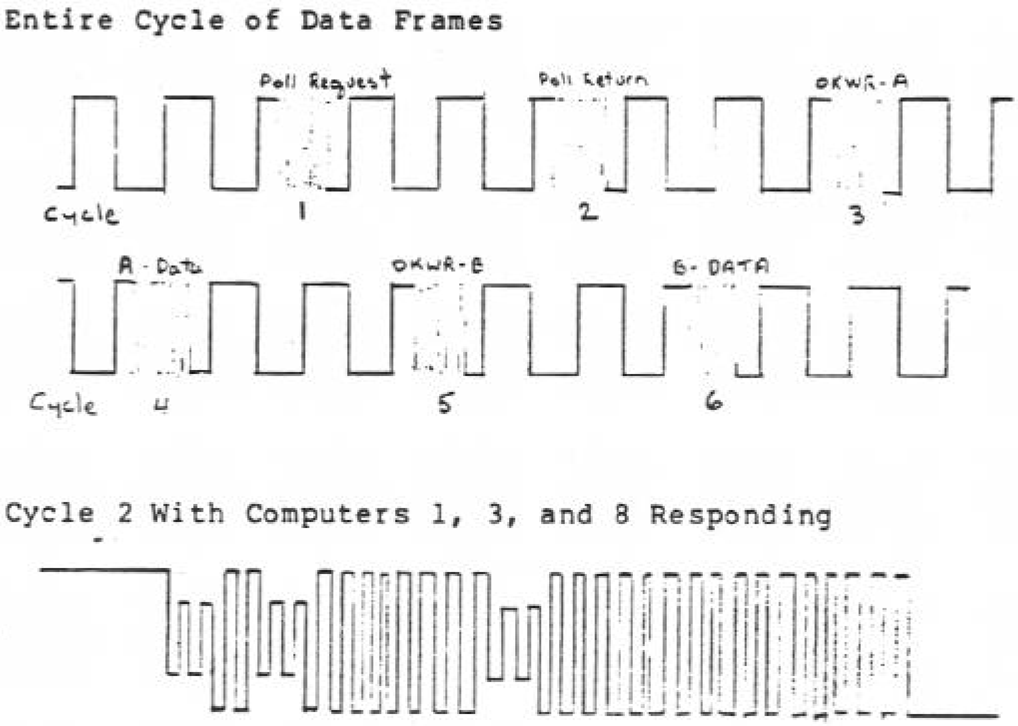
2.4 TRANSMISSION SEQUENCE

The system designed for sending return data consists of a polling sequence. The controller polls each of the track computers simultaneously to determine which, if any, of the computers have return data to transmit. It then grants one data frame of the track waveform to a single track computer for transmission of its sixteen bits of data.

For ease of understanding, the data frames will be numbered in the following discussion, starting with Cycle 1. The entire sequence is illustrated in Figure 4. The controller polls all track computers by sending a message with computer ID set to '00' during Cycle 1. This is a reserved computer ID; no computer is allowed to have an ID of zero. When the ISERV routines in all of the track computers read in this message with BLKID = 0, they interpret it as a 'poll request'. If a track computer has return data to send, it will indicate this to the controller during the next data frame (Cycle 2) by sending a 'one' in a bit position corrsesponding to its block ID. For example, if computer #6 has return data to send, it will wait until it receives a message with BLKID = 0 (Cycle 1). During the next data frame (cycle 2), it will send a 'one' during bit 6 and a 'zero' for all other bits. Since each computer has its own bit position, up to 16 computers may respond to the controller's poll. The master reads the return data and makes a list of all track computers that have requested to send data. The master then individually permits each computer to send its sixteen bits of information. It does this by sending a task code of '00'. This is a reserved task code and may not be used for any other purpose by the controller or the track computers. The computer that has received the '00' task code interprets this code as a 'Send Return Data' command and responds by sending return data during the next data cycle. This process is summarized as follows:

Cycle 1: Controller sends 'poll request' accessory data frame with block ID of zero.

Cycle 2: All track computers with return data pending send a 'one' during the bit of the return data that matches each computers' ID.


Entire Cycle of Data Frames

Cycle 2 With Computers 1, 3, and 8 Responding

Figure 4: Protocol For Transmission of Return Data

Cycle 3: Controller sends 'Send Return Data' message to computer A.

Cycle 4: Computer A sends return data to the controller.

Cycle 5: Controller sends 'Send Return Data' message to computer B.

Cycle 6: Computer B sends return data to controller.

.

.

.

And so on until all track computers have sent their data to the controller.

In order to initiate the transmission sequence, the user program running in the track computers load up locations WDATAL and WDATAH with the return data, and indicate to ISERV that there is data to be sent by setting DFLAG to hex 'FF'. ISERV looks at DFLAG to determine if there is any data to send, and resets DFLAG to '00' once the return data has been sent.

3. HARDWARE

3.1 SYSTEM OVERVIEW

In order for the block computer to send data, and the controller to receive it, two circuits are necessary. The first is a data transmission circuit to be used by the block computer, and the second is a data recovery circuit to be used by the master controller. The data transmission circuit was designed so that it could be controlled by the PB4 output of the 6522 VIA, and the data recovery circuit was designed so that it could easily be added to the master controller. An additional circuit is necessary in order to link the track waveform to the block computer. This is necessary so that the block computer can receive data from the master. The requirements for this circuit are that it electrically isolate the computer from the track, and provide a TTL input to the CB1 and CB2 control inputs.

3.2 CIRCUIT DESCRIPTIONS

3.2.1 Data Transmission Circuit

As explained previously, amplitude modulation is used to transmit return data from track computers to the master controller. The data transmission circuit is used by the track computer to switch a resistor in and out across the track (see Figure 5). This load will cause a power supply current surge which can be sensed at the master controller. The circuit consists of two back to back SCR's (the equivalent of a triac) which switch in a 10 ohm load when triggered.

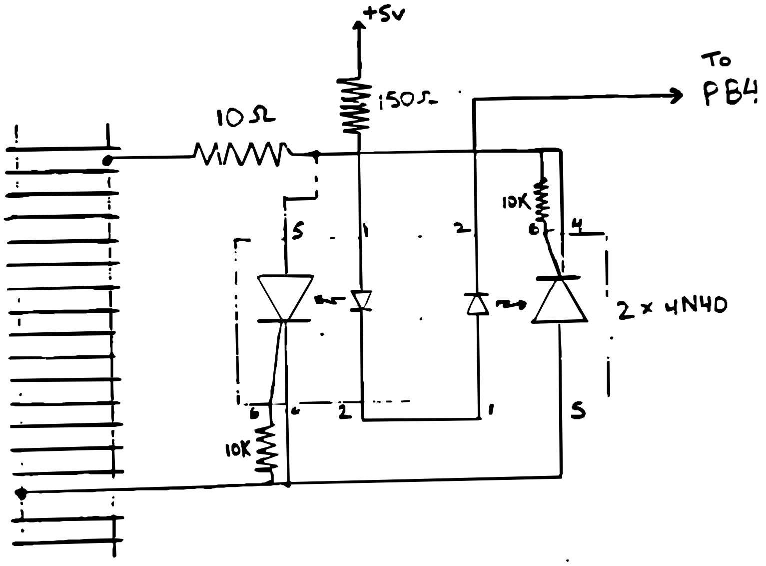


Figure 5: Data Transmission Circuit

Binary data can be sent by pulsing the SCR gates thus causing the load to short out the track. When the load is shorted across the track, the track waveform will be attenuated. When the load is open circuited, the waveform will be at its normal level. (see section 2.1.2) We have adopted the convention that a binary '1' is given by the shorted condition, and a binary '0' is given by the open circuit condition.

3.2.2 Data Recovery Circuit

The second circuit developed is used by the master controller to recover the signal transmitted by the track computers (see Figure 6). During the data cycle, the current being drawn from the track is constant. Only computers are drawing current, therefore the total current drawn will be a function of the number of computers connected. The voltage drop across resistor R will therefore also be constant. However, when the track computer connects the 10 ohm resistor across the track, there will be a current surge and a corresponding voltage increase across resistor R. With proper selection of R, transistors T1 and T2 will turn on only when the track is in the 'shorted' state. T1 will turn on for positive portions of the waveform, and T2 for negative portions. These transistors will trigger two 4N25 optocouplers, which will in turn, switch on and off T3. A 1K resistor is placed between the base of T3 and ground in order to help dissapate the saturation base charge, and speed operation of the transistor. The TTL output is taken from the collector of T3.

56

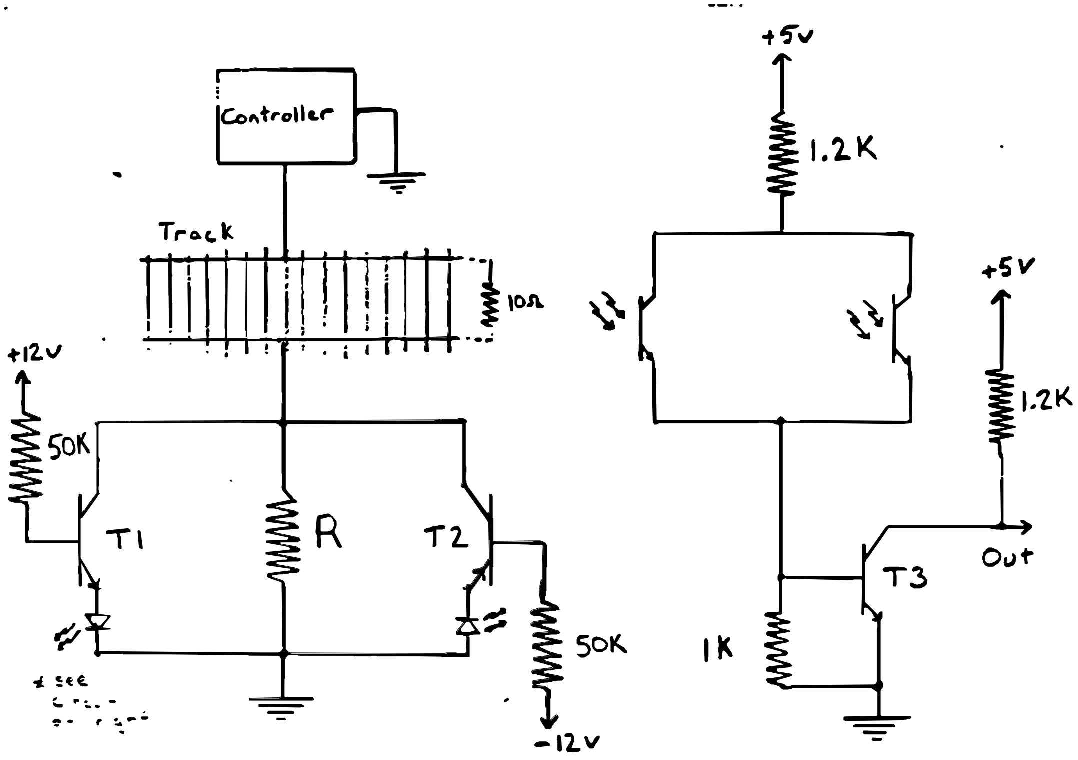


Figure 6: Data Recovery Circuit

The resistor R must be selected so that T1 and T2 do not trigger unless the 10 ohm resistor is switched across the track. T1 will turn on when the voltage across R is approximately 1.4 volts, and the trigger voltage for T2 is approximately -1.4 volts. Since the steady state data frame current will vary with the number of computers in use, we must vary R when the number of computers is changed in order to get the proper steady state voltage across R. For one computer it is found that a 1.3 ohm resistor can be utilized. As the number of computers increases, the resistance must be decreased indirectly with the steady state current draw. Note that all of the current used in the system will flow thru this resistance, and therefore it must be able to handle the power. We found that a 75 watt resistor was sufficient.

3.2.3 Block Computer Receiver Circuit

The final circuit uses a 4N~~25~~33 opto coupled transistor to isolate the track from the computer (see figure 7). This circuit provides the CB1 and CB2 control inputs with the necessary TTL compatible signals.

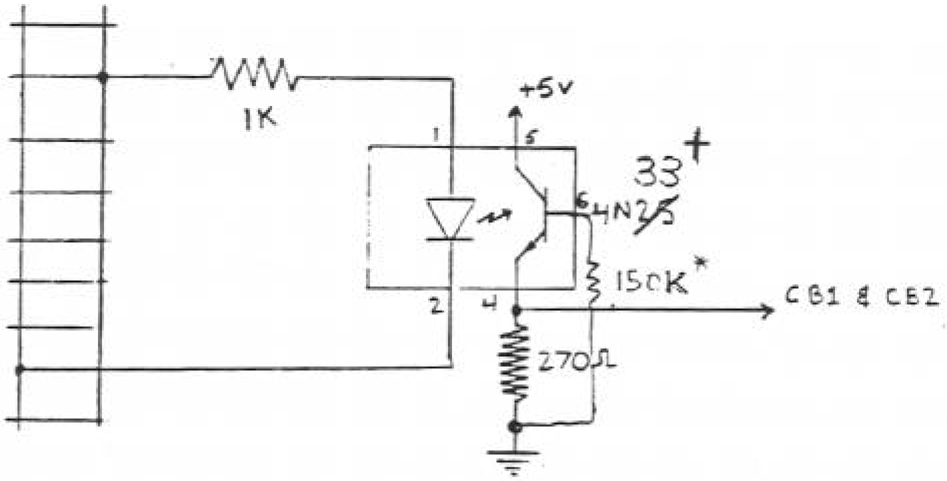


Figure 7: Block Computer Receiver Circuit

* 7-19-82 modification (added to improve freq. response)
† 7-19-82 modification (4N33 used because of higher gain)

4. SOFTWARE

4.1 SOFTWARE SYSTEM OVERVIEW

4.1.1 Software Requirements

The first requirement is that the transmit/receive loop (RDWR) must operate at speeds which allow the block computer to decode the track waveform. The shortest interval between track transitions occurs if a binary zero is being sent (75 microseconds). This means that the RDWR loop cycle time must be less than this value. In order to make this possible, two 32 byte buffers were used. The first of these buffers, WBUF, contains two bytes for each bit of return information sent. (this is necessary because each bit of return data has the same period as two bits of incoming data) The second buffer, TABLE, is used to store the incoming data as it arrives. Both of these buffers allow quick processing of the I/O, by simplifying the RDWR loop structure. Only one bit of each byte of TABLE is used for information storage, however the speed gained by using the extra memory made it worth the tradeoff. All of the other system requirements explained in section 1 were also incorporated into the software design.

4.1.2 Overall Program Logic

The processor is interrupted every time a CB1 or CB2 transition occurs. Present postion in the track waveform is determined, and if not in the data frame (frame #5), various housekeeping tasks are performed. After this processing, control is returned to the user routine. If position of the waveform is at the beginning of the data frame, the buffers mentioned in the above section, are prepared for the transmit/receive loop (RDWR). During the loop, I/O is performed; all of the data in WBUF is sent out to the tracks, and the incoming FSK bits are decoded and stored in TABLE. Directly following this loop, data processing takes place. The data is decoded and any valid commands are carried out. Control is then returned to the user, and the subroutine waits for the next interrupt.

(for a more specific description of software, see below)

4.2 LABEL DEFINITIONS

The following is a list of the variable names and labels used in the assembly routine. The list is divided into three types of variables; VIA address constants, interface variables which are used by the foreground program to communicate with the subroutine, and local variables which are used only by the subroutine and are invisible to the user.

4.2.1 Versatile Interface Adapter (VIA) Address Constants

VORB = A000 : Output Register B.

VDDRB = A002 : Data Direction Register B.

VT2L = A008 : Timer 2 Low Byte (Latch/Counter).

VT2H = A009 : Timer 2 High Byte (Counter).

VACR = A00B : Accessory Control Register.

VPCR = A00C : Peripheral Control Register.

VIFR = A00D : Interrupt Flag Register.

VIER = A00E : Interrupt Enable Register.

4.2.2 Interface Variables

FRANUM = 3FA : Current track waveform frame number. Varies between 1 and 5, but since ISERV is the only routine running during frame 5, the user program only sees FRANUM cycles between 1 and 4. FRANUM can be used by the user program as a simple two-bit timer.

BLKID = 3FC : Block identification number of the track computer. The ID may be any number between 1 and 15.

RDATA = 3FB : Most recent controller data sent to this track computer. The data is stored in RDATA by ISERV if it determines that the data is valid for the computer named in BLKID.

WDATAH = 3FE : High byte of return data (set by user program)

WDATAL = 3FD : Low byte of return data (set by user program)

DFLAG = 3FF : Flag that signals to ISERV whether or not there is valid data to be sent in WDATAH and WDATAL. when DFLAG = FF, ISERV will attempt to send the data in WDATAH and WDATAL, and will reset DFLAG to 00 once the data has been successfully sent. Otherwise, it will ignore the return data. It is expected that a user will load WDATAH and WDATAL with return data, and then set DFLAG to FF when the data is ready to be transmitted.

4.2.3 Local Variables

OKWR = 3F9 : Flag that signals whether or not the controller is ready to accept return data from this track computer. FF = OK to write data; 00 = controller not ready.

INTMPU = 3F8 : Flag that signals whether or not the track computer may transmit a 'Request to Send' signal. Initially set to 00, ISERV sets INTMPU to FF when it receives a 'poll request' signal from the processor.

POS = 3F7 : Contains the byte to be output to Output Register B after a CB1 interrupt has been received in the power cycle. If positive polarity is present, bit 6 will be '1' and bit 5 will be '0'. The situation is reversed for negative polarity. In all cases bit 7 will b '0' to indicate the power portion of the waveform is present.

NEG = 3F6 : Contains the byte to be output to Output Register B after a CB2 interrupt has been received during a power cycle. If positive polarity is present, bit 6 will be '0' and bit 5 will be '1'. The situation is reversed for a negative polarity waveform. POS and NEG are set up so that PB6 will be '1' during positive cycles of the waveform, and PB5 will be '1' during the negative cycles of the waveform.

PLRTY = 3F5 : Contains FF for positive track polarity, and 00 for negative track polarity.

LCV = 3F4 : Loop variable.

MASK = 3F3 : Contains '11101111'; used to mask out a carry after a four-bit addition. Used by the PARITY routine.

CNT = 3F2 : Loop variable.

TIME = 3F1 : Equal to a period of time that is midway between the period of a 'zero' controller bit (nominally 75 usecs.) and a 'one' controller bit (nominally 150 usecs.). This has a nominal value of 112, but is dependent upon the accuracy of the track computer's clock. Used by ISERV to time incoming controller bits.

WBUF = 3D0 : Contains 32 bytes of data to be written to the controller. Two bytes of WBUF are needed for each bit of return data. A byte of WBUF contains the data to be written directly to ORB. The contents of each byte of WBUF is as follows:

Bit 7 : '1' - Indicates that a data frame is present.

Bit 6 : '0' - Positive power frame not present.

Bit 5 : '0' - Negative power frame not present.

Bit 4 : Return data to be sent. Contains a '0' if corresponding bit of return data is '0' and a '1' if the return data is a '1'.

Bits 3-0 : Depend on LOCCON routine (Bar-code routine).

FTAB = 3AD : Eight-byte-long table containing the eight nibbles of controller data. Used to process the incoming controller data.

TABLE = 38B : Thirty-two byte long table which contains the first 32 bits of the controller data. Each byte of TABLE contains the contents of the IFR when the trailing edge of the corresponding bit of controller data was detected. Since timer 2 is set to the value of TIME on the leading edge of each bit of controller data, timer 2 will time out if a '1' is sent, but not if a '0' is sent. Bit 5 of a byte in TABLE will therefore contain a '0' if a zero was sent or a '1' if a one was sent.

4.3 SOFTWARE ROUTINE DESCRIPTIONS

The following descriptions explain each of the communications subroutine sections. The descriptions can be read in conjunction with the subroutine flowcharts (appendix A) and the assembly code (appendix B).

4.3.1 INIT

INIT, the initialization routine for ISERV, must be called by the user program to initialize the service routine. INIT performs the following functions:

1. Its sets all pins of port B for output except for PB2; this is used by the bar-code routine LOCCON.
2. It sets the IER to interrupt the track computer ONLY during CB1 and CB2 interrupts.
3. It calls PWRPOL to determine the polarity of the track waveform and set appropriate variables.
4. It calls PANIC to synchronize itself with the track waveform.
5. It sets the variable TIME to a value which is halfway between the period of a controller '0' and '1' bit.

INIT then returns to the user program. INIT should be called on powerup as soon as the user program has initialized the system stack.

4.3.2 PWRPOL

PWRPOL determines the polarity of the track waveform by looking for a 10 millisecond interval between CB2 interrupts that is present in an inverted signal, but not in a normal signal (see Figure 8). PWRPOL first assumes that positive polarity is present. Therefore, the PCR is set so a rising waveform edge will trigger a CB2 interrupt, and a falling edge will trigger a CB1 interrupt. POS and NEG are also set to reflect positive polarity. One hundred successive intervals between CB2 interrupts are tested; if no 10 msec. interval is found after 100 interrupts, positive polarity is assumed for the remainder of ISERV's operation. To allow for some inaccuracy in track computer clocks, PWRPOL actually tests for any interval in the range 7.9 msecs < interval < 12.2 msecs.

If the 10 msec. interval is found, PWRPOL adjusts POS and NEG to reflect this. It also sets the PCR so a CB1 interrupt will be triggered on a rising edge, while a CB2 interrupt will be triggered on a falling edge.

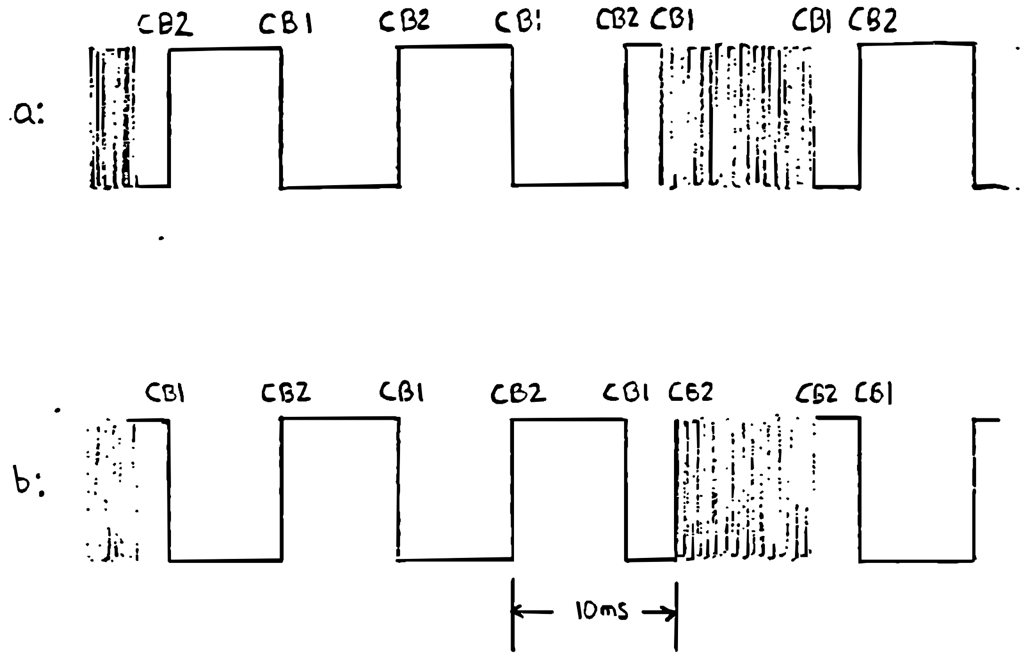


Figure 8: Positive and Negative Polarity Track Signals

a. Normal positive track waveform

b. Negative track waveform before PWRPOL has corrected the CB1 and CB2 interrupts

Note the 10 msec period that occurs only when the polarity is negative

4.3.3 PARITY

PARITY performs modulo four additions on the eight nibbles of the controller data. If the data is valid, the nibbles should add to '1111'. The result is left in the A-register. The calling routine should expect a '0F' to be returned if the data is valid.

As PARITY adds each successive nibble to the A-register, it checks bit 4 to determine if a four-bit carry has occured. If it has, the carry bit is set; otherwise, the carry bit is cleared. Bit 4 is cleared in any case.

4.3.4 PANIC

PANIC synchronizes ISERV with the track waveform. It does this by waiting for the first CB1 interrupt after the high-frequency data portion of the track waveform. This occurs at the start of frame 2; FRANUM is set accordingly.

Initially, WTHF is called to locate the high-frequency portion of the waveform. Next, a loop is entered which sets T2 to 256 microseconds, and times the interval between successive CB1 interrupts. The routine will loop until T2 has timed out in the interval between CB1 interrupts. When this happens, the routine has located frame #2.

4.3.5 PANIC2

The routine PANIC (above) is arranged as a subroutine. Some routines are not set up to call PANIC as a subroutine; rather, they branch to PANIC and require PANIC to branch to the RETURN routine of ISERV (not the same as executing an RTS). Thus, PANIC2 was written. PANIC2 merely calls PANIC, and branches to RETURN afterwards.

4.3.6 WTHF

WTHF is used to locate the high-frequency data portion of the track waveform. It does so by waiting for the first interval between CB1 interrupts that is less than 256 microseconds.

4.3.7 WTCB1, WTCB2, WT2

WTCB1 and WTCB2 wait for either a CB1 or CB2 transition to occur. WT2 waits for timer 2 to time out. All three work by looping until the proper bit in the IFR is set.

4.3.8 ISERV

Although ISERV as a rule refers to the entire interrupt service routine, the small routine actually labeled ISERV in the assembly listings performs the following functions:

1. It disables further interrupts from causing a hardware interrupt at the 6502 MPU.
2. It saves the A, X, and Y registers on the system stack.
3. It determines the present location in the waveform by refering to franum, and branches to the proper internal routine.

The section beginning with the label RETURN restores the registers and executes an RTI.

4.3.9 VCB1

VCB1 first increments FRANUM and checks to see whether the frame number is valid; if not, PANIC2 is called. It checks to make sure that T2 has timed out, which must always be the case (except in the data frame). As before, PANIC2 is called if anything is amiss. If things seem to be in order, T2 is reset to 12.288 milliseconds, and ORB is loaded with NEG to reflect a positive or negative power cycle (depending on the track polarity, of course). Finally, if FRANUM = 2, LOCCON is called to service the bar-code readers.

4.3.10 VCB2

VCB2 performs necessary processing prior to the actual reading and writing of data. It initially performs frame-number checking (similar to that done by VCB1) to determine if the system is still in phase with the track waveform. If the data frame is not present, control will be returned to the user program. If the data frame is present (FRANUM = 5), WBUF is prepared for data transmission by performing the following tests:

1. If DFLAG is cleared, the buffer is zeroed and no more processing takes place.
2. If DFLAG is set, INTMPU is next checked to see if a 'Request to Send' signal may be transmitted. If so, the bit in WBUF that corresponds to BLKID is set on;

all others are turned off. Example: computer 4 would have bits 8 and 9 set to '1'.

3. If INTMPU is not set, OKWR is checked. If this flag is set, the return data is copied from WDATAL and WDATAH and is formatted into WBUF. On the other hand, if the OKWR flag is clear, WBUF is zeroed.

After WBUF has been set up, control is passed to the routine RDWR.

4.3.11 RDWR

RDWR performs all of the actual transmission and reception of data. It sets T2 to TIME at the start of each bit of controller data, then waits for the CB1 interrupt marking the end of that bit (and the start of the next bit). When the CB1 flag is present in the IFR, the IFR is stored in TABLE and the proper byte from WBUF is output to ORB. The T2 flag in the IFR indicates whether a '0' or '1' was received. Since all WBUF data formatting is done beforehand, and all controller data decoding is done afterwards, the inner read/write loop is very efficient, and makes possible the decoding of high-frequency signals.

After all of the 32 bits have been received, the IFR data stored in TABLE is reconstructed into eight, four bit nibbles. The data is discarded if any parity errors are detected (using the PARITY routine), if an accessory frame was not received, or if the message frame BLKID does not match the computer's BLKID.

Two special cases are possible:

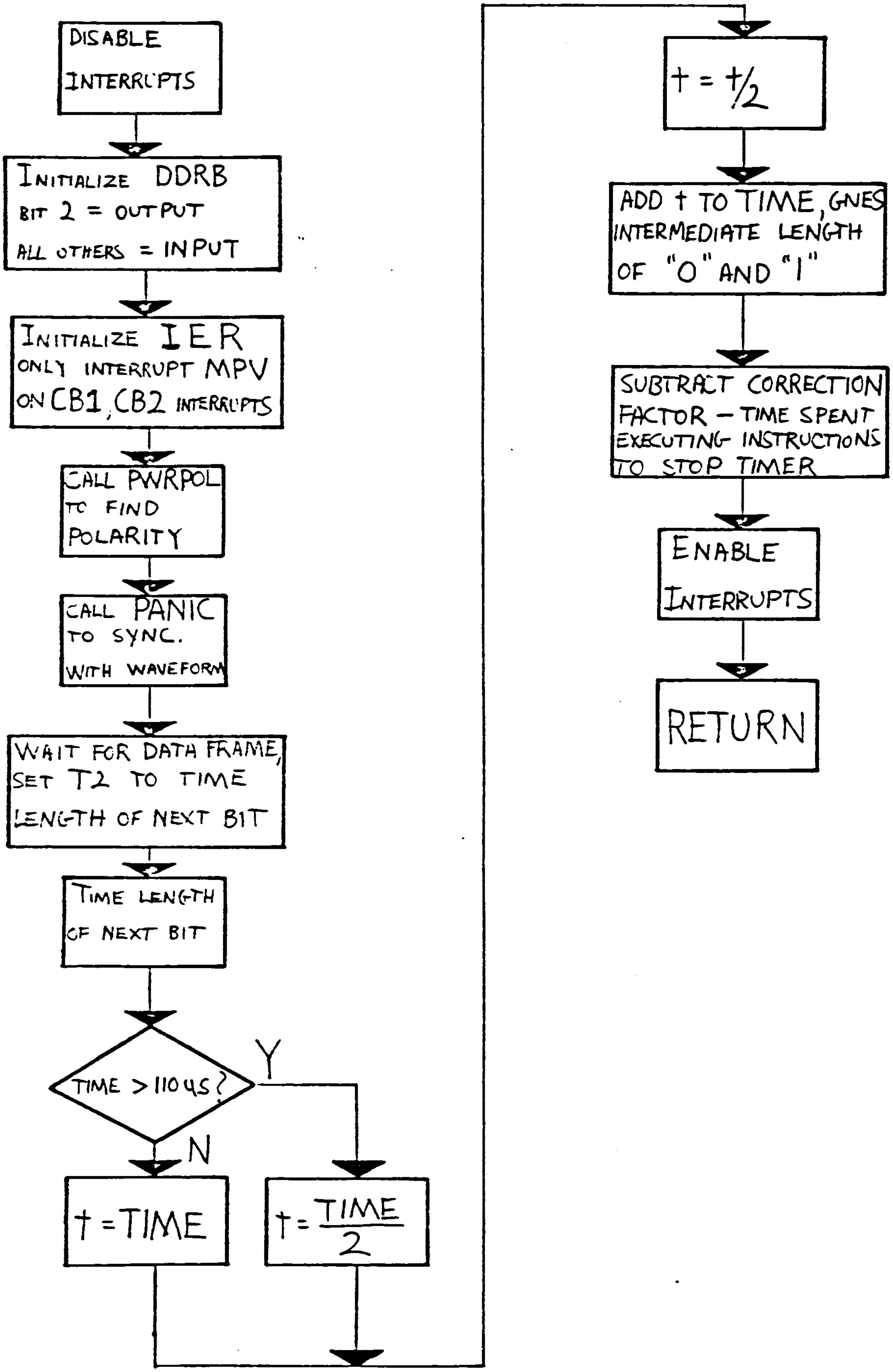
1. If the message frame BLKID = '00', this is not a message frame, but a 'poll request' from the controller. In this case INTMPU is set to 'FF'
2. If the task code (nibbles 4 and 5) is equal to zero, this is a 'Send Data' command from the controller. OKWR is set to 'FF', and data will be sent out during the next data frame.

If no special cases are present, and the received BLKID agrees with the computer BLKID, the task code is stored at RBUF. RDWR then waits for the power cycle to arrive, and sets FRANUM = 1 upon its arrival. Control is then returned to the user program through the RETURN routine.

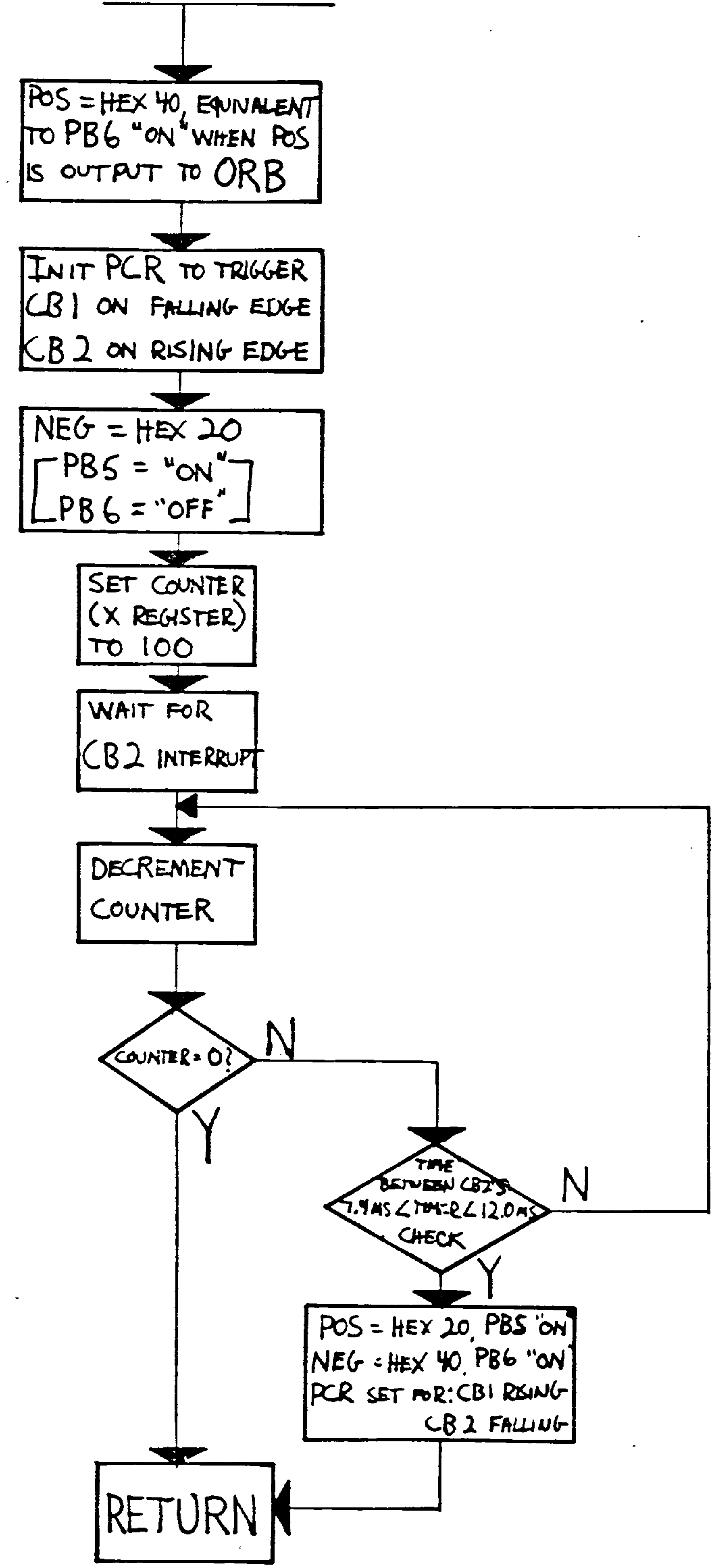
Appendix A

ISERV FLOWCHARTS

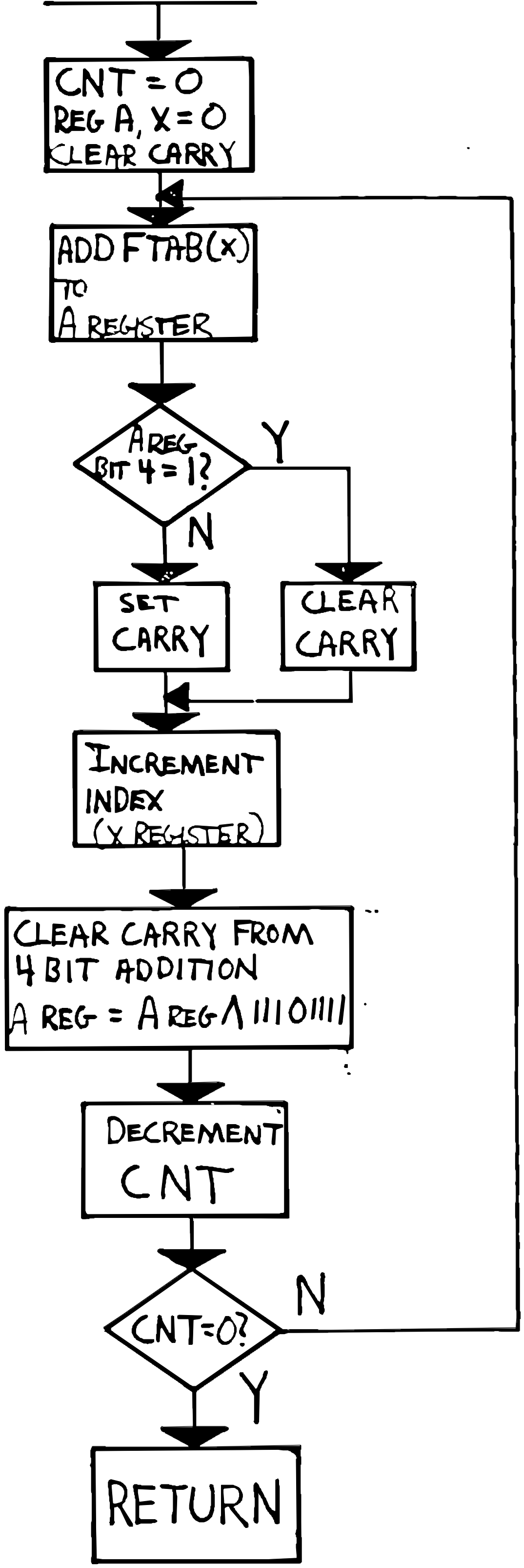
INIT

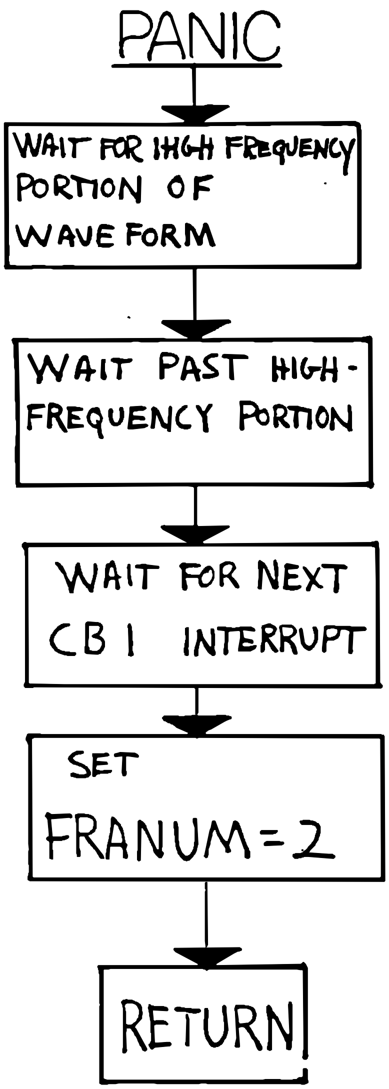

DISABLE INTERRUPTS
INITIALIZE DDRB
BIT 2 = OUTPUT
ALL OTHERS = INPUT
INITIALIZE IER
ONLY INTERRUPT MPV ON CB1, CB2 INTERRUPTS
CALL PWRPOL TO FIND POLARITY
CALL PANIC TO SYNC. WITH WAVEFORM
WAIT FOR DATA FRAME, SET T2 TO TIME LENGTH OF NEXT BIT
TIME LENGTH OF NEXT BIT
TIME > 110μs?
Y
N
t = TIME
t = TIME/2
t = t/2
ADD t TO TIME, GIVES INTERMEDIATE LENGTH OF "0" AND "1"
SUBTRACT CORRECTION FACTOR – TIME SPENT EXECUTING INSTRUCTIONS TO STOP TIMER
ENABLE INTERRUPTS
RETURN

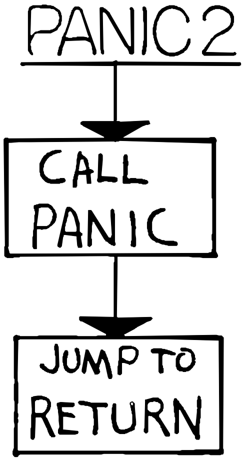
PWRPOL


POS = HEX 40, EQUIVALENT TO PB6 "ON" WHEN POS IS OUTPUT TO ORB
INIT PCR TO TRIGGER CB1 ON FALLING EDGE CB2 ON RISING EDGE
NEG = HEX 20
PB5 = "ON"
PB6 = "OFF"
SET COUNTER (X REGISTER) TO 100
WAIT FOR CB2 INTERRUPT
DECREMENT COUNTER
COUNTER = 0?
N
Y
TIME BETWEEN CB2'S
7.9MS < TIMER < 12.0MS
CHECK
N
Y
POS = HEX 20, PB5 "ON"
NEG = HEX 40, PB6 "ON"
PCR SET FOR: CB1 RISING
CB2 FALLING
RETURN

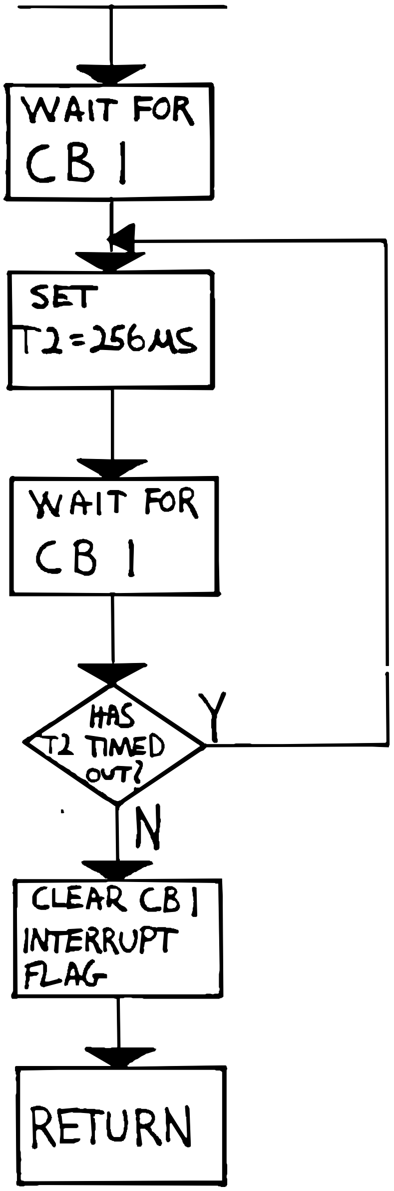
PARITY


CNT = 0
REG A, X = 0
CLEAR CARRY
ADD FTAB(X)
TO
A REGISTER
A REG
BIT 4 = 1?
Y
N
SET
CARRY
CLEAR
CARRY
INCREMENT
INDEX
(X REGISTER)
CLEAR CARRY FROM
4 BIT ADDITION
A REG = A REG ∧ 11101111
DECREMENT
CNT
CNT=0?
N
Y
RETURN

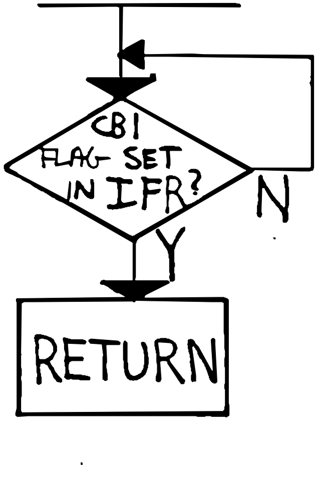

PANIC
WAIT FOR HIGH FREQUENCY PORTION OF WAVE FORM
WAIT PAST HIGH-FREQUENCY PORTION
WAIT FOR NEXT CB1 INTERRUPT
SET FRANUM=2
RETURN


PANIC2
CALL PANIC
JUMP TO RETURN

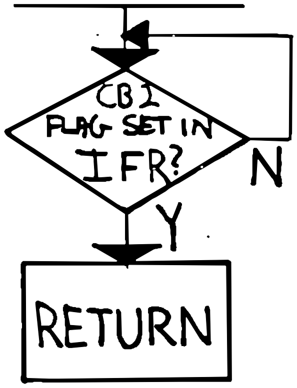
WTHF



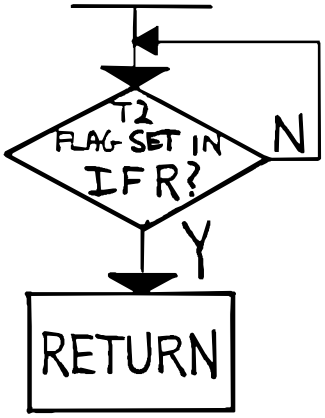
WTCB1

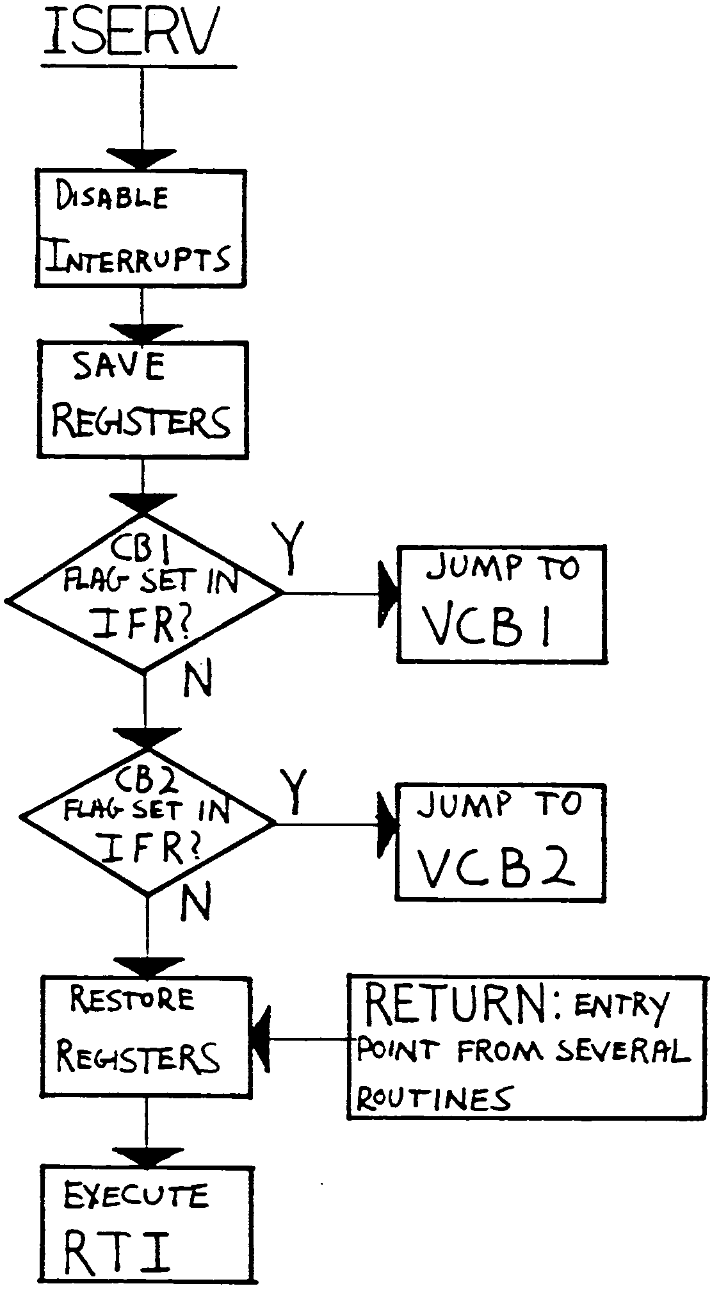


WTCB2

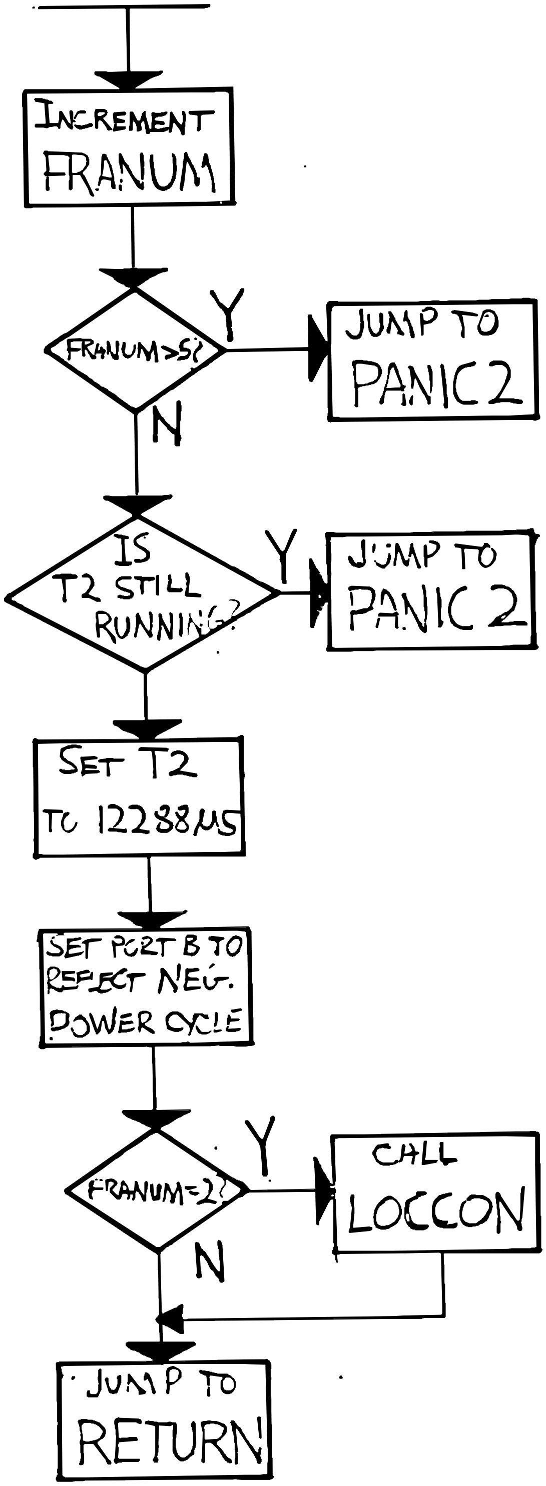


WT2

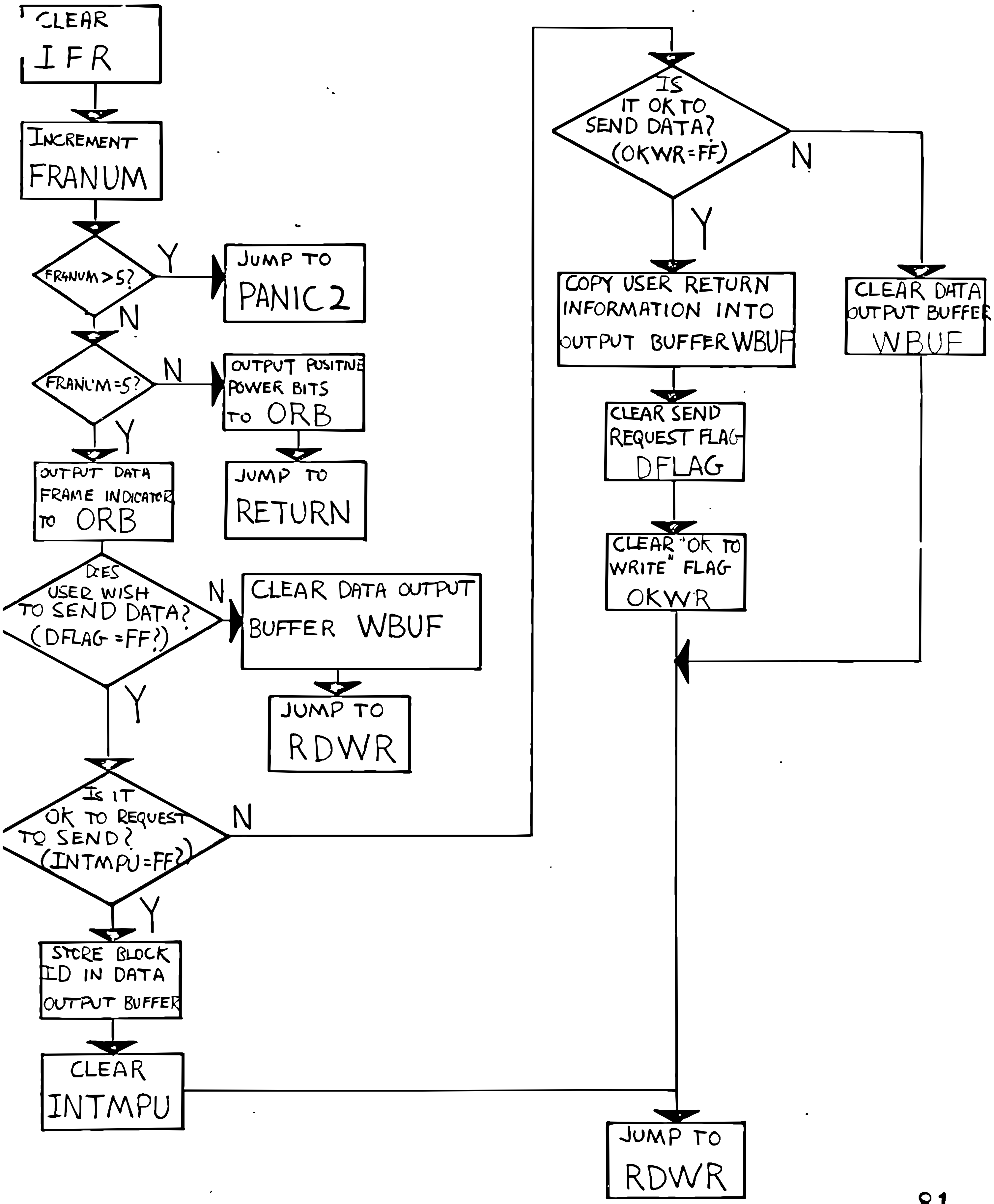


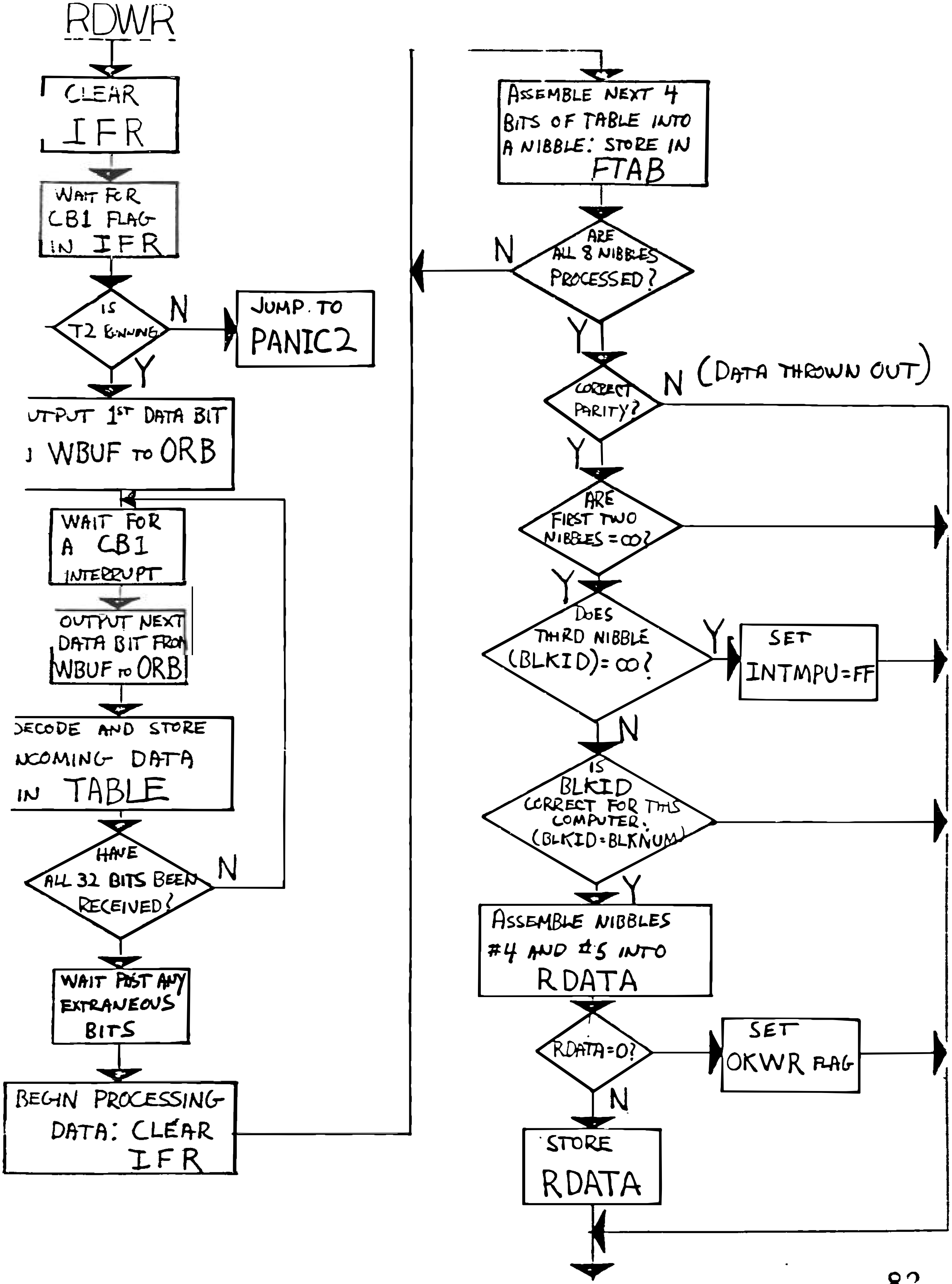

ISERV
DISABLE INTERRUPTS
SAVE REGISTERS
CB1 FLAG SET IN IFR?
Y
JUMP TO VCB1
N
CB2 FLAG SET IN IFR?
Y
JUMP TO VCB2
N
RESTORE REGISTERS
RETURN: ENTRY POINT FROM SEVERAL ROUTINES
EXECUTE RTI

VCB1

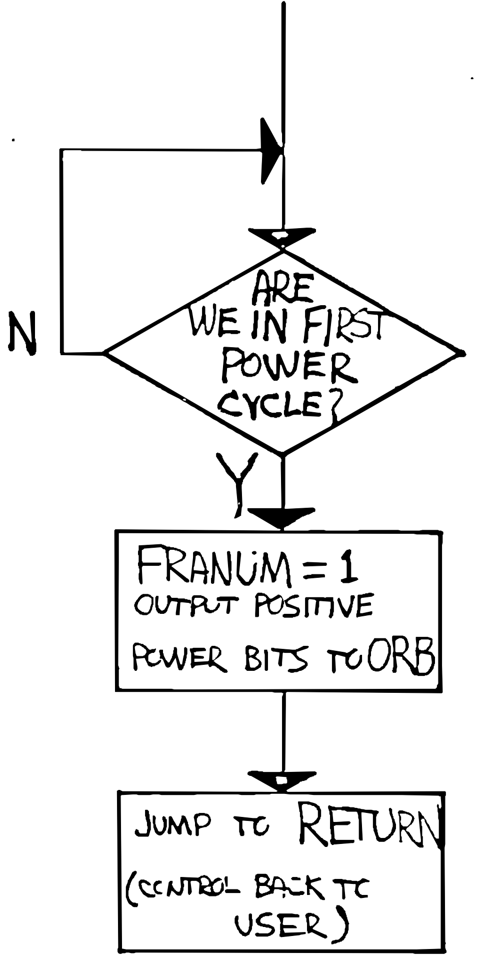

INCREMENT FRANUM
FRANUM>5?
Y
JUMP TO PANIC2
N
IS T2 STILL RUNNING?
Y
JUMP TO PANIC 2
SET T2 TO 12288μS
SET PORT B TO REFLECT NEG. POWER CYCLE
FRANUM=2?
Y
CALL LOCCON
N
JUMP TO RETURN

VCB2


CLEAR IFR
INCREMENT FRANUM
FRANUM>5?
Y
JUMP TO PANIC2
N
FRANUM=5?
N
OUTPUT POSITIVE POWER BITS TO ORB
JUMP TO RETURN
Y
OUTPUT DATA FRAME INDICATOR TO ORB
DOES USER WISH TO SEND DATA? (DFLAG=FF?)
N
CLEAR DATA OUTPUT BUFFER WBUF
JUMP TO RDWR
Y
IS IT OK TO REQUEST TO SEND? (INTMPU=FF?)
N
Y
STORE BLOCK ID IN DATA OUTPUT BUFFER
CLEAR INTMPU
IS IT OK TO SEND DATA? (OKWR=FF)
N
Y
COPY USER RETURN INFORMATION INTO OUTPUT BUFFER WBUF
CLEAR SEND REQUEST FLAG DFLAG
CLEAR "OK TO WRITE" FLAG OKWR
CLEAR DATA OUTPUT BUFFER WBUF
JUMP TO RDWR


RDWR
CLEAR IFR
WAIT FOR CB1 FLAG IN IFR
IS T2 RUNNING
N
JUMP TO PANIC2
Y
UTPUT 1ST DATA BIT
WBUF TO ORB
WAIT FOR A CB1 INTERRUPT
OUTPUT NEXT DATA BIT FROM WBUF TO ORB
ECODE AND STORE
NCOMING DATA
IN TABLE
HAVE ALL 32 BITS BEEN RECEIVED?
N
WAIT PAST ANY EXTRANEOUS BITS
BEGIN PROCESSING DATA: CLEAR IFR
ASSEMBLE NEXT 4 BITS OF TABLE INTO A NIBBLE: STORE IN FTAB
ARE ALL 8 NIBBLES PROCESSED?
N
Y
CORRECT PARITY?
N (DATA THROWN OUT)
Y
ARE FIRST TWO NIBBLES = 00?
Y
DOES THIRD NIBBLE (BLKID) = 00?
Y
SET INTMPU=FF
N
IS BLKID CORRECT FOR THIS COMPUTER. (BLKID=BLKNUM)
Y
ASSEMBLE NIBBLES #4 AND #5 INTO RDATA
RDATA=0?
SET OKWR FLAG
N
STORE RDATA

RDWR (CONTINUED)



Appendix B

ISERV ASSEMBLY LISTING

```
*         LABEL DECLARATIONS
*
          FRANUM = $3FA        CURRENT WAVEFORM FRAME NUMBER
          BLKID  = $3FC        TRACK COMPUTER BLOCK ID NUMBER
          RDATA  = $3FB        CONTROLLER DATA SENT TO THIS COMPUTER
          WDATAH = $3FE        HIGH BYTE OF RETURN DATA
          WDATAL = $3FD        LOW BYTE OF RETURN DATA
          DFLAG  = $3FF        00=NO RETURN DATA TO SEND
*                              FF=RETURN DATA READY TO SEND
          OKWR   = $3F9        00=CONTROLLER NOT READY TO ACCEPT RETURN I
*                              FF=CONTROLLER READY TO ACCEPT RETURN DATA
          INTMPU = $3F8        FF=MPU HAS ISSUED POLL REQUEST
          POS    = $3F7        OUTPUT DURING POSITIVE POWER CYCLE
          NEG    = $3F6        OUTPUT DURING NEGATIVE POWER CYCLE
          PLRTY  = $3F5        00=NEGATIVE POLARITY, FF=POSITIVE POLARIT
          LCV    = $3F4        LOCAL VARIABLE
          MASK   = $3F3        LOCAL CONSTANT
          CNT    = $3F2        LOCAL VARIABLE
          TIME   = $3F1        MEDIAN PERIOD OF A '0' AND '1' BIT OF
*                              CONTROLLER DATA
          WBUF   = $3D0        OUTPUT TABLE - DATA TO BE WRITTEN TO PORT
          FTAB   = $3AD        RECONSTRUCTED 'NIBBLES' OF CONTROLLER DAT
          TABLE  = $38B        INPUT TABLE - IFR CONTENTS FOR EACH INPUT
*
*         VERSATILE INTERFACE ADAPTER (VIA) ADDRESSES
*
          VDDRB   = $A002      DATA DIRECTION REGISTER B
          VORB    = $A000      OUTPUT REGISTER B
          VT2L    = $A008      TIMER 2 LOW BYTE
          VT2H    = $A009      TIMER 2 HIGH BYTE
          VACR    = $A00B      AUXILARY CONTROL REGISTER
          VPCR    = $A00C      PERIPHERAL CONTROL REGISTER
          VIFR    = $A00C      INTERRUPT FLAG REGISTER
          VIER    = $A00E      INTERRUPT ENABLE REGISTER
*
*         PROCEDURE: INIT
*         FUNCTION : INITIALIZE VARIABLES, SYNCHRONIZE WITH TRACK WAVEF
*
INIT      SEI                  DISABLE ALL MPU INTERRUPTS
          LDA #$FB             SET DDRB FOR ALL OUTPUTS EXCEPT PB2
          STA VDDRB            STORE AWAY
          LDA #0
          STA VACR             INITIALIZE ACR
          STA INTMPU           INITIALIZE INTMPU
```

```
         STA OKWR                LIKEWISE
         LDA #7F
         STA VIER                CLEAR ALL INTERRUPTS
         LDA #98
         STA VIER                ENABLE CB1, CB2 INTERRUPTS
         JSR PWRPOL              DETERMINE POLARITY OF INPUT SIGNAL
         JSR PANIC               SYNCHRONIZE WITH TRACK WAVEFORM
*
*        FIND THE PERIOD OF A '0' OR '1', AND COMPUTE AN INTERMEDIATE VALUE
*
         LDA VORB                CLEAR INTERRUPT FLAG IN IFR
         JSR WTHF                WAIT FOR HIGH-FREQUENCY PORTION OF WAVEFORM
         LDA #255
         STA V2TL                SET TIMER FOR 255 USECS.
         JSR WTCB1               ALIGN TO A CB1 INTERRUPT
         LDA #0
         STA VT2H                START TIMER 2
         LDA VORB                CLEAR CB1 INTERRUPT IN IFR
         JSR WTCB1               AND WAIT FOR THE NEXT CB1 INTERRUPT
         LDA VT2L                LOAD COUNTER TO FIND TIME BETWEEN CB1 INTERRUPTS
         STA TIME                AND SAVE
         SEC                     SET CARRY IN PREPARATION FOR ARITHMETIC
         LDA #255                LOAD ORIGINAL 255 USECS.
         SBC TIME                SUBTRACT REMAINDER OF T2 COUNTER
         SBC #2                  ADJUST FOR TIMING ERRORS TO GIVE TIME OF BIT
         CMP #110                COMPARE TO MEDIAN VALUE: IS THIS A '0' OR A '1'?
         BMI SHORT               IF <110, WE HAVE A '0'
         LSR A                   OTHERWISE, WE HAVE A '1', DIVIDE BY 2 TO NORMALIZE
SHORT    STA TIME                SAVE PERIOD OF A '0'
         CLC                     CLEAR CARRY
         LSR A                   DIVIDE BY 2
         CLC                     CLEAR CARRY
         ADC TIME                MEDIAN VALUE = 1.5 X PERIOD OF '0'
*                                             = .75 X PERIOD OF '1'
         SBC #38                 SUBTRACT CONSTANT DUE TO TIMING ERRORS FROM
*                                EXECUTION OF INSTRUCTIONS
         STA TIME                STORE AWAY
         CLI                     ENABLE INTERRUPTS
         RTS                     RETURN TO USER ROUTINE
*
*        ROUTINE : ISERV
*        FUNCTION : INTERRUPTS SERVICE ROUTINE, CALLS LOWER-LEVEL ROUTINES
*
ISERV    SEI                     DISABLE INTERRUPTS
*        PUSH REGISTERS ON SYSTEM STACK
         PHA
         TXA
         PHA
         TYA
         PHA
         LDA #$10
         BIT VIFR                TEST FOR CB1 INTERRUPT
         BNE VCB1                BRANCH IF FOUND
         LDA #$08
```

```
         BIT VIFR            CHECK FOR CB2 INTERRUPT
         BEQ RETURN          SKIP IF NOT FOUND
         JMP VCB2            OTHERWISE BRANCH
*        RETURN - RESTORE STACK AND LEAVE
RETURN   PLA
         TAY
         PLA
         TAX
         PLA
         RTI
*
*        ROUTINE : VCB1
*        FUNCTION : HANDLE CB1 INTERRUPTS
*
VCB1     INC FRANUM          INCREMENT FRAME NUMBER
         LDA FRANUM
         CMP #6              IS FRAME NUMBER > 5?
         BMI LESS            LESS THAN 6, SO OK
         JMP PANIC2          OTHERWISE, OUT OF SYNC - PANIC!
LESS     LDA #$20
         BIT VIFR            HAS T2 TIMED OUT YET?
         BNE L1              YES, WE'RE OK
         JMP PANIC2          NO, WE'RE OUT OF SYNC
         LDA #0
         STA VT2L
         LDA #$30
         STA VT2H            SET T2 = 12288 USECS.
         LDA NEG
         ORA #1              MAKE SURE PB0 IS SET
         STA VORB            OUTPUT POWER CYCLE, POS OR NEG DEPENDING ON POLARITY
         LDA FRANUM
         CMP #2              FRANUM = 2?
         BNE L2              IF NOT, CONTINUE
         JSR LOCCON          TAKE CARE OF BAR CODE READER(S)
L2       JSR RETURN          LEAVE
*
*        ROUTINE : PWRPOL
*        FUNCTION : DETERMINE POLARITY OF TRACK SIGNAL
*
*        INITIALLY, ASSUME POSITIVE POLARITY. SET POS TO OUTPUT
*        A '1' ON PB6 AND '0' ON PB5 DURING POSITIVE POWER CYCLE, AND
*        VICE-VERSA FOR NEG. OBJECT IS TO LOOK FOR A 10 MSEC PERIOD BETWEEN
*        CB2 INTERRUPTS. IF THIS IS FOUND, WE HAVE NEGATIVE POLARITY
*
PWRPOL   LDA #$40
         STA POS
         STA VPCR
         LDA #$FF
         STA PLRTY SET PLRTY = POSITIVE
         LDA #$20
         STA NEG
         LDA #0
         STA VT2L            SET LOW T2 BYTE TO ZERO
*
```

was DEY in Lecky VII
Corrected in MAE-N/TRAK V1.2 28Feb 1983

```
         LOOP THROUGH 100 TIMES - IF 10 MSEC PERIOD IS NOT FOUND BY THEN,
         WE HAVE POSITIVE POLARITY
*
         LDX #100           SET COUNTER TO 100
         LDA VORB           CLEAR IFR
         JSR WTCB2          WAIT FOR CB2 INTERRUPT
TSTLP    DEX   DECREMENT LOOP COUNTER
         BEQ TSTEND         BRANCH TO TSTEND IF FINISHED
         LDA #$30
         STA VT2H           SET T2 TO 12288 USECS.
         LDA VORB           CLEAR IFR
 STA TIME JSR WTCB2         WAIT FOR NEXT CB2
         LDA VT2H           STORE TIME BETWEEN CB2 INTERRUPTS
         LDA VIFR
         BIT T2TEST         SEE IF T2 HAS TIMED OUT (T > 12288 USEC?)
         BNE TSTLP          > 12288 MSEC, BACK TO LOOP
         LDA TIME
         CMP #$11           TIME < 8000?
         BPL TSTLP          YES, GO BACK INTO LOOP
*
*        HERE, WE HAVE NEGATIVE POLARITY : 8000<TIME<12288 USEC
*
         LDA #0
         STA PLRTY          PLRTY = NEGATIVE
         LDA #$20
         STA POS            RESET POSITIVE
         LDA #$10
*
*        SET CB1 TO TRIGGER ON RISING EDGE
*        SET CB2 TO TRIGGER ON TRAILING EDGE
*
         STA VPCR
         LDA #$40
         STA NEG            RESET NEGATIVE
TSTEND   RTS
*
*        PROCEDURE : PANIC
*        FUNCTION : SYNCHRONIZE ISERV WITH TRACK WAVEFORM
*
PANIC    JSR WTHF           WAIT FOR DATA PORTION OF WAVEFORM
         LDA #0                                  deleted in VII & V1.2
         STA V2TL           ZERO LOW BYTE OF T2   accidently?
PLOOP    LDA #$01
         STA VT2H           SET T2 TO 256 USECS.
         JSR WTCB1          WAIT FOR NEXT CB1
         LDA VORB           CLEAR IFR OF CB2 INTERRUPT
         LDA VIFR
         BIT T2TEST         HAS T2 TIMED OUT?
         BEQ PLOOP          IF NOT, GO BACK
*
         T2 HAS TIMED OUT - WE HAVE FOUND AN INTERVAL GREATER THAN
         256 USECS. THIS MUST BE START OF POWER CYCLE.

         LDA #2
```

```
         STA FRANUM           FRAME NUMBER = 2, FIRST CB2 INTERRUPT AFTER
                              DATA CYCLE
         RTS                  WE'RE DONE
*
*        ROUTINE : PANIC2
*        FUNCTION : CALLS SUBROUTINE PANIC, BRANCHES TO 'RETURN'
*
         JSR PANIC            PANIC!
         JMP RETURN           AND SCRAM
*
*        PROCEDURE : PARITY
*        FUNCTION : PERFORMS FOUR-BIT ADDITION OF CONTROLLER NIBBLES
*        RETURNS : '0F' IN A-REGISTER IF NO PARITY ERRORS ARE PRESENT
*
PARITY   LDA #8
         STA CNT              INIT LOOP COUNTER
         CLC
         LDA #0
         TAX                  INIT A,X REGISTERS
ALOOP    ADC FTAB,X           ADD UP NIBBLES
         BIT AMASK            WAS THERE A CARRY INTO BIT 4?
         BEQ A1               IF SO, SKIP
         SEC                  OTHERWISE, SET CARRY
         JMP A2               AND SKIP
         CLC                  NO CARRY INTO BIT 4, SO CLEAR CARRY
         INX                  INCREMENT FTAB POINTER
         AND #$EF             CLEAR BIT 4
         DEC CNT              DECREMENT LOOP COUNTER
         BNE ALOOP            CONTINUE UNTIL FINISHED
         RTS                  THAT'S ALL, FOLKS!
AMASK    .BYT $10             MASK FOR CHECKING BIT 4
*
*        PROCEDURE : WTCB1
*        FUNCTION : SUSPEND PROCESSING UNTIL NEXT CB1 INTERRUPT OCCURS
*
WTCB1    LDA #$10             LOAD A WITH IFR FLAG FOR CB1
ABC      BIT VIFR             SEE IF INTERRUPT HAS OCCURED YET
         BEQ ABC              BRANCH BACK IF IT HASN'T
         RTS                  ELSE LEAVE
*
*        PROCEDURE : WTCB2
*        FUNCTION : WAIT FOR A CB2 INTERRUPT
*
         LDA #$08             LOAD A WITH IFR BIT FOR CB2
XYZ      BIT VIFR             CHECK FOR INTERRUPT
         BEQ XYZ              LOOP UNTIL WE'VE FOUND IT
         RTS                  YOU'VE MADE IT THIS FAR - NOT BAD!
*
*        PROCEDURE : WT2
*        FUNCTION : WAIT FOR TIMER 2 TIMEOUT

         LDA #$20             LOAD A WITH T2 IFR BIT
CBA      BIT VIFR             CHECK FOR TIMEOUT
         BEQ CBA              RUN BACK FOR MORE UNTIL WE TIMEOUT
```

```
        RTS                     NOPE, NOTHING TRICKY HERE - WAIT TILL LATER
*
*       PROCEDURE : WTHF
*       FUNCTION : WAIT FOR THE HIGH-FREQUENCY PORTION OF THE WAVEFORM
*
WTHF    LDA VORB                CLEAR CB INTERRUPTS IN IFR
        JSR WTCB1               WAIT FOR CB1
        LDA #0
        STA VT2L                ZERO T2 LOW BYTE
HFLOOP  LDA #$01
        STA VT2H                SET T2 TO 256 USECS.
        LDA VORB                CLEAR CB1 INTERRUPT
        JSR WTCB1               WAIT FOR NEXT CB1 INTERRUPT
        LDA #$20
        BIT VIFR                CHECK FOR T2 TIMEOUT
        BNE HFLOOP              IF TIMEOUT, INTERVAL IS TOO LONG, GO FOR MORE
*
*           OTHERWISE, INTERVAL IS LESS THAN 256 USECS, AND WE HAVE
*           FOUND THE HIGH FREQUENCY PORTION
*
        LDA VORB                CLEAR CB INTERRUPTS
        RTS                     AND CRUISE
*
*       ROUTINE : VCB2
*       FUNCTION : HANDLES CB2 INTERRUPTS

  B2    INC FRANUM              INCREMENT FRANUM
        LDA #5
        CMP FRANUM              FRAME NUMBER = 5?
        BMI PANIC2              PANIC IF FRAME > 5
        BEQ CONT1               PREPARE TO ENTER DATA FRAME IF FRAME = 5
        LDA POS                 OTHERWISE, OUTPUT POWER CYCLE
        ORA #1                  SET BIT 0 ON
        STA VORB                AND OUTPUT
        JMP RETURN              AND GO ON YOUR MERRY WAY
CONT1   LDA #$81
        STA VORB                SET PB7,PB1 ON - SIGNAL DATA FRAME TO OUTSIDE WORLD
        LDA DFLAG               GET DFLAG
        BNE CONT2               BRANCH IF THERE IS RETURN DATA TO WRITE
        JSR ZERO                ZERO OUT WBUF
        JMP RDWR                GO READ/WRITE
CONT2   LDA INTMPU              CHECK TO SEE IF WE CAN 'INTERRUPT' CONTROLLER
        CMP #$FF
        BNE CONT3               BRANCH IF CONTROLLER HAS SENT POLL REQUEST
        JSR ZERO                OTHERWISE, ZERO WBUF
        LDA BLKID               GET BLOCK # OF THIS COMPUTER
        ASL A                   MULTIPLY BY 2
        TAX                     TRANSFER TO INDEX REGISTER
        LDA #$90                LOAD A WITH PB7, PB4 TURNED ON
        STA WBUF,X              SET WBUF TO REFLECT BLKID
        STA WBUF+1,X            SET SECOND BYTE OF RETURN DATA BUFFER
        LDA #0
        STA INTMPU              ZERO OUT INTMPU
        JMP RDWR                AND GO TO RDWR
```

```
CONT3     LDA OKWR             SEE IF IT'S OK TO SEND RETURN DATA
          CMP #$FF
          BEQ CONT4            BRANCH IF WE CAN SEND DATA
          JSR ZERO             OTHERWISE, ZERO OUT WBUF
          JMP RDWR             AND BEGIN READIN' AND WRITIN'
CONT4     LDX #30
JHG0      ROR WDATAH           ROTATE 2 BYTES OF DATA
          ROR WDATAL           AND ROTATE LOW BIT INTO CARRY LATCH
          BCC JHG1             BRANCH IF LOW BIT IS '1'
          LDA #$90             LOAD A WITH 'DATA FRAME PRESENT, OUTPUT = 1'
          JMP JHG2
JHG1      LDA #$80             LOAD A WITH 'DATA FRAME PRESENT, OUTPUT = 0'
JHG2      STA WBUF,X           STORE PORT B CONTENTS IN OUTPUT BUFFER
          STA WBUF+1,X         REMEMBER, TWO BYTES PER BIT
          DEX                  DECREMENT X BY TWO
          DEX
          BPL JHG0             CONTINUE FOR 16 BITS (32 BYTES IN BUFFER)
          LDX #32              RESET INDEX TO 32
          LDA #$80
          STA WBUF,X           SET BIT 17 TO ZERO (BIT NOT A DATA BIT)
          LDA #0
          STA DFLAG            RESET DFLAG AFTER DATA IS SENT
          STA OKWR             RESET THIS WHILE WE'RE AT IT
          JMP RDWR             GO TO READ/WRITE ROUTINE
*
*         PROCEDURE : ZERO
*         FUNCTION : ZERO OUT THE OUTPUT BUFFER 'WBUF'
*
ZERO      LDA #$80
          LDX #0
LP2       STA WBUF,X           STORE 'DATA FRAME PRESENT' IN PORT B BUFFER
          INX                  INCREMENT BUFFER INDEX
          CPX #33              COMPARE TO 33
          BNE LP2              SCOOT BACK IF WE'RE NOT DONE
          RTS                  AND LEAVE WHEN WE ARE
*
*         ROUTINE : RDWR
*         FUNCTION : READ CONTROLLER DATA IN WHILE SIMULTANEOUSLY WRITING
*               RETURN DATA OUT ON PB4
*
RDWR      LDA VORB             CLEAR IFR OF ANY CB1 OR CB2 INTERRUPTS
          LDX #0
          JSR WTCB1            WAIT FOR FIRST CB1 (I.E. START OF FIRST BIT)
          LDA #$20
          BIT VIFR             CHECK FOR T2 TIMEOUT
          BEQ NEXTONE          CONTINUE IF NO T2 STILL RUNNING
          JMP PANIC2           IF T2 IS NOT RUNNING AT START OF DATA FRAME....
NEXTONE   LDA TIME             GET PERIOD INTERMEDIATE BETWEEN A '0' AND '1'
          STA VT2L             SET T2 TO MEDIAN VALUE
          LDA #0
          STA V2TH             ZERO HIGH BYTE OF T2 AND START CLOCK
          LDA WBUF             GET FIRST BYTE OF OUTPUT BUFFER
          ORA #1               TURN ON PB0
          STA VORB             AND OUTPUT
```

```
*
*        START OF LOOP THAT ACTUALLY READS AND WRITES DATA
*
RDLOOP   JSR WTCB1             WAIT FOR START OF NEXT BIT
         LDA WBUF+1,X          GET NEXT BYTE IN BUFFER
         ORA #1                NOT FORGETTING TO ZAP PB0
         STA VORB              OUTPUT RETURN DATA
         LDA VIFR              LOAD IFR, STORE IN TABLE
         STA TABLE,X
         INX                   INCREMENT TABLE/BUFFER INDEX
         LDA #0
         STA VT2H              RESTART TIMER 2
         CPX #32               SEE IF WE'VE READ IN 32 BITS
         BNE RDLOOP            AND GO BACK IF WE HAVEN'T
         LDA #64
         STA VT2L              SET TIMER 2 TO WAIT PAST ANY EXTRA BITS
         LDA #1
         STA VT2H              START T2 FOR ABOUT 200 MICROSECONDS
         JSR WT2               WAIT FOR THE TIMER TO FINISH
*
*        BEGIN PROCESSING OF DATA. DETERMINE IF A ZERO OR ONE WAS SENT BY
*        EXAMINING IFR TABLE THAT CONTAINS TIMING INFO ON EACH BIT
*
         LDA VORB              CLEAR PORT B INTERRUPTS
         LDX #0
         LDY #0                CLEAR X,Y REGS
         STY FTAB              ZERO OUT FIRST NIBBLE OF CONTROLLER DATA TABLE
OUTLOP   LDA #4
         STA LCV               SET COUNTER TO FOUR BITS IN A NIBBLE
         LDA #1
         STA MASK              SET MASK TO LEAST SIGNIFICANT BIT
INLOOP   LDA TABLE,X           GET INTERRUPT FLAG PERTAINING TO BIT X
         BIT T2TEST            SEE IF T2 HAD TIMED OUT
         BEQ GZERO             IF IT HADN'T, WE HAVE A ZERO FOR THIS BIT
         LDA FTAB,Y            OTHERWISE, GET RECONSTRUCTED NIBBLE
         ORA MASK              AND INSERT THIS BIT
         STA FTAB,Y            AND RESET MEMORY
GZERO    INX                   INCREMENT 'TABLE' INDEX
         ASL MASK              SET MASK TO TURN ON NEXT BIT
         DEC LCV               DECREMENT COUNTER
         BNE INLOOP            GO TO INNER LOOP IF NOT DONE WITH THIS NIBBLE
         INY                   INCREMENT FTAB INDEX
         LDA #0
         STA FTAB,Y            INIT CONTENTS OF THIS NIBBLE TO '0000'
         CPY #8                SEE IF WE'VE PROCESSED ALL EIGHT NIBBLES
         BNE OUTLOP            IF NOT, FINISH UP
         JSR PARITY            CHECK PARITY OF RECONSTRUCTED CONTROLLER DATA
         CMP #$0F              DOES IT ADD TO '1111'?
         BNE CHCB2             EXIT IF IT IS BAD DATA
*
*        CHECK TO SEE IF THIS DATA IS ACCESSORY FRAME DATA MEANT FOR THIS
         COMPUTER, OR IF ANY CONTROL DATA WAS SENT
*
         LDA FTAB+1
```

```
        ORA FTAB+2              OR TOGETHER NIBBLE 1 AND NIBBLE 2
        BNE CHCB2               SHOULD BE ZERO FOR ACCESSORY FRAME, SCRAM OTHERWI
        LDA FTAB+3              LOAD UP BLKID
        BNE SKIP                IF '0000', THIS IS A POLL REQUEST
        LDA #$FF
        STA INTMPU              SO, SIGNAL ISERV FOR THE NEXT TIME AROUND
        JMP CHCB2               AND EXIT
SKIP    CMP BLKID               CHECK FOR PROPER BLOCK ID
        BNE CHCB2               AND LEAVE IF THIS ISN'T OUR MESSAGE
        LDA FTAB+5              GET FOUR BITS WORTH OF TASK CODE
        ASL A                   SHIFT INTO UPPER FOUR BITS OF BYTE
        ASL A
        ASL A
        ASL A
        ORA FTAB+4              AND ADD THE REMAINING FOUR BITS
        BNE SKIP2               IF CODE IS NOT 'SEND' COMMAND, EXIT
        LDA #$FF                CODE IS SEND COMMAND, SO SIGNAL
        STA OKWR                ISERV TO SEND DATA NEXT TIME AROUND
        JMP CHCB2               AND RETURN
SKIP2   STA RDATA               STORE TASK CODE IN RDATA
CHCB2   LDA #$08
        BIT VIFR                CHECK TO SEE IF WE HAVE GOT A CB2 INTERRUPT
        BEQ CHCB2               AND LOOP UNTIL WE HAVE (MEANS START OF POWER CYCL
        LDA #1
        STA FRANUM              RESET FRAME NUMBER TO THE START
        LDA POS                 SIGNAL WHICH POWER CYCLE WE'RE IN
        ORA #1                  TURN ON PB0
        STA VORB                OUTPUT TO PORT B
        JMP RETURN              AND RETURN FROM INTERRUPT ROUTINE
T2TEST  .BYT $20                T2 INTERRUPT FLAG IN IFR
```