AN ABSTRACT OF THE THESIS OF

SALEH HASSAN IBRAHIM for the degree of Master of Science
in Nuclear Engineering presented on October 28, 1987.
Title: COMPUTER CONTROL OF THE AUTOMATIC GAMMA WELL COUNTING SYSTEM

Abstract Approved: Redacted for Privacy
STEPHEN E. BINNEY

This project implements the control of an AUTOMATIC GAMMA WELL counting system with an AIM-65 microcomputer. All states and control signals to and from the AIM-65 are obtained via three VIAs (Versatile Interface Adaptors). Motor controls were implemented using triacs, operational amplifiers and TTL logic devices while the RTC (Real Time Clock) utilizes a 32.768 kHz quartz precision crystal and battery backup.

The radiation detection system can handle solid or liquid phase gamma ray emitting samples. Samples may be prepared in 15mm X 110mm bottles or in 15mm X 125mm test tubes and placed in receptacles on the conveyor belt of the sample changer. Under software control, selected samples can be lowered into the well of the detector and counted. A hardcopy of the parameters used in setting up the experiment as well as the results may be obtained on a teletype.

Computer Control of the AUTOMATIC GAMMA WELL counting System

by

SALEH HASSAN IBRAHIM

A THESIS
submitted to
Oregon State University

in partial fulfillment of
the requirements for the
degree of
Master of Science
Completed October 28, 1987
Commencement June 1988

APPROVED:

Redacted for Privacy

Professor of Nuclear Engineering in charge of major

Redacted for Privacy

Head of Department of Nuclear Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented October 28, 1987

Typed by SALEH HASSAN IBRAHIM for SALEH HASSAN IBRAHIM

TABLE OF CONTENTS

CHAPTER I. INTRODUCTION 1
1.1 Purpose of equipment 1
1.2 Description of units 2
1.3 Specifications 5
1.4 Principles of operation 8

CHAPTER II. MICROCOMPUTER SELECTION 10
2.1 HP-IL interface 11
2.2 HP87/HP-IB interface 13
2.3 Intel 8748 microcomputer based design 14
2.4 AIM-65 microcomputer 16
2.5 6522 VIA 19

CHAPTER III. POWER SUPPLY AND PERIPHERALS SELECTION 22
3.1 Power supply and triac 22
3.2 Real time clock (RTC) 28
3.3 Quad counter/timer 33
3.4 Drive motors 34
3.5 Contact switches 35

CHAPTER IV. COMPUTER INTERFACE DESIGN FOR PERIPHERALS 36
4.1 Interfacing the real time clock (RTC) 36
4.2 Interfacing the ORTEC 874 timer/counter 42
4.3 Interfacing the motors and contact switches 53
4.4 Light emitting diodes (LED) 59
4.5 Hardware assembly and testing 60

CHAPTER V. SOFTWARE 62
5.1 Overview of 6502 assembler and the basic-E/65 62
5.2 Assembler routines 63
5.3 Basic programs and subroutines 65

CHAPTER VI. BIBLIOGRAPHY 67

APPENDICES
A. User's manual 69
B. Components layout and CABLEs 82
C. Program listings 92

LIST OF FIGURES

Figure Page

2.1 The mos technology 6502 microprocessor 18

3.1 System power transformer 23

3.2 Triac/low voltage power supply 24

3.3 Auxilliary power supply 25

3.4 Triac pellet structure, circuit symbol and ac volt-ampere characteristics 27

3.5 Block diagram and pin out specifications of the National Semiconductor MM58167A real time clock 30

3.6 Block diagram and pin out specifications of the National Semiconductor MM58174A real time clock 31

4.1 Real time clock interface circuit with the VIA2 (9F80-9F8F) 37

4.2 Ortec timer/counter interface circuit with a UART 43

4.3 LED interface and baud rate selector (A000-A00F) 47

4.4 Ortec timer/counter interface circuit with a software UART 50

4.5 Down-motor driver 54

4.6 Up-motor driver 57

4.7 Sample changer-motor driver 58

B.1 Component layout on interface card 82

B.2 Component layout description 83

B.3 Detailed layout of resistors, transistor and diode 84

B.4 Connections of CABLE A to VIA (A000-A00F) 85

B.5 Connections of CABLEs B and C to VIA.2 and VIA.3 repectively 86

B.6 Connections of CABLE D to the LEDs baud rate selector 87

B.7 Signals on interface card and auxilliary power supply card connectors 88

B.8 Wire wrap plugboard edge connector 89

B.9 Solder plugboard edge connector 90

B.10 Amphenol connector for 20 mA loop 91

C.1 MISC.PRN listing 92

C.2 PROJ.BAS listing 100

C.3 PROJ2.BAS listing 136

LIST OF TABLES

Table Page

1.1 Efficiency and Average Background Measurement 7

2.1 I/O location summary and ports of the VIA 20

4.1 Decoding logic for the real time clock (RTC) 39

4.2 Baud rate selection for the Ortec 874 48

COMPUTER CONTROL OF THE AUTOMATIC GAMMA WELL COUNTING SYSTEM

CHAPTER I

INTRODUCTION

1.1 PURPOSE OF EQUIPMENT

The Automatic Gamma Well counting systems as built by Nuclear Chicago are transistorized radiation detection systems for solid or liquid phase gamma ray emitting samples. The systems have the capability of counting up to 100 individual solid and/or liquid gamma emitting samples in a well-type scintillation detector chamber. The models produced range from 4216 through 4223. The models were classified into single-channel counting systems (4216 through 4219) and dual-channel counting systems (4220 through 4223). Some were further designed for 60 Hz power operation (4216.4218.4220 and 4222) and others for 50 Hz (4217.4219.4221 and 4223). Each of these models basically comprises an automatic sample changer, heavily shielded well-type scintillation detector, analyzer, scaler or combination analyzer/scaler, printing lister, logic assembly and the necessary low voltage and high voltage power supplies. Even though the scalers, analyzers, listers and logic assemblies have long since become obsolete, the heavy shielding and nearly 4-pi detection geometry provide excellent sensitivity and resolution. Auxiliary power receptacles are also provided on the modules for connection to external equipment (1).

1.2 DESCRIPTION OF UNITS

An older Gamma Well counting system at the Oregon State University Radiation Center has been upgraded (by the exclusion of some units that are now obsolete and the inclusion of modern electronic devices) into a more state of the art controlled system.

Only the units that are retained for this project will be described as most of the electronics that came with the system have been discarded in favor of the more modern NIM BIN-compatible electronic devices. The units that have been retained are the sample changer, detector assembly and the low and high voltage power supply. The automatic well changer available is the 1085 model while the scintillation well detector is of the type 972 and the counting system model is 4218.

1.2.1 SAMPLE CHANGER

The sample changer is comprised of a mechanically driven conveyor belt and elevator mechanism. The conveyor linked belt is readily accessible for the ease of loading and maintainance. The sample elevator mechanism "bottoms" all bottles (indexed at the elevator) in the crystal well regardless of the bottle length.

Samples may be prepared in the 15mm x 110mm bottles provided or in 15mm x 125mm test tubes. The bottles and the test tubes may be intermixed if desired. An adapter is provided for holding the various bottle sizes or the test tube size in the conveyor belt. Receptacles on the conveyor belt are numbered consecutively from 0 through 99, where 0 is equivalent to the 100th sample. The sample changer unit is equipped with seven contact switches for "sensing" various states of the system to effect complete controllability (1).

1.2.2 DETECTOR ASSEMBLY

The detector assembly consists of a preamplifier circuit, photomultiplier tube, and scintillation crystal. The thallium-activated sodium iodide [NaI(Tl)] crystal provides high sensitivity and resolution with nearly 4-pi detection geometry for small volume samples. It is optically coupled to the photo-cathode face of the photomultiplier tube and except for the polished photo-cathode surface is enclosed by a light reflecting shield on all sides. The photomultiplier tube is shielded by extensive mu-metal magnetic shielding. The shielding is of stepped, interlocking construction for ease of assembly and for maintaining shield integrity. Additional steel shielding is also employed between the detector assembly and the samples in the changer conveyor belt (1).

1.2.3 TRIAC/LOW VOLTAGE POWER SUPPLY

About the only piece of electronics that has been retained from the old system is the triac and low voltage power supply card. This card carries the triacs that are used for driving the three motors and also performs a full wave voltage rectification to provide the d.c. operating voltages of -15v, +15v and other transistorized derivatives (-3v, +6v). The latter will not be used in this project. The full details are given in Chapter IV.

1.3 SPECIFICATIONS

1.3.1 SAMPLE CHANGER

SAMPLE CAPACITY - 100

SAMPLE SIZE - 5 cubic centimeter recommended. 15mm x 110mm bottles or 15mm x 125mm test tubes.

SAMPLE ELEVATOR - positive rack-and-pinion.

DRIVE MECHANISM - heavy duty motors with reduction trains.

1.3.2 DETECTOR

CRYSTAL - Sodium iodide, thallium activated [NaI(Tl)]. Hermetically sealed 7.62 centimeter diameter x 7.62 centimeter thick.

Multiplier phototube - 10 stage with mu-metal and lead shields .

VOLTAGE PLATEAU - 150 volts long, with 5% slope per 100 volts (with Co-60).

SHIELDING - 8.89 to 12.7 centimeters of lead around sides of detector, 25.4 centimeters of lead above crystal, combination of steel and several centimeters of lead below crystal. Steel shielding between detector assembly and samples in conveyor belt.

RESOLUTION - Approximately 8% for cesium-137.

EFFICIENCY AND BACKGROUND - The efficiencies and average background measurements of the detector for Cesium-137 and Cobalt-60 standards are given in Table 1.1.

TABLE 1.1 DETECTOR EFFICIENCY AND AVERAGE BACKGROUND

Standard	Cesium-137				Cobalt-60			
Energy range	X4	X5	X1	X2	X6	X7	X3	X2
Efficiency %	-	-	22	39	-	-	18	50
Average Background cpm	-	-	30	360	-	-	20	360
Background variation (cpm/microcurie)	4(-4)	.14	-	-	.016	.5	-	-
	(+/- 20%)				(+/- 15%)			

X1 = 200 keV window at Cesium-137 photopeak

X2 = 100 keV to infinity

X3 = 500 keV window at Cobalt-60 photopeak

X4 = 611 to 711 keV

X5 = X2

X6 = 1.1 MeV to 1.3 MeV

X7 = 208 keV to infinity

HIGH VOLTAGE - +400 V to 2600 V .

1.4 PRINCIPLES OF OPERATION

Gamma ray emitting isotopes emit electromagnetic rays with an energy spectrum particular to that isotope. Hence each different isotope can be detected by its energy spectrum, by making use of a detector with a response proportional to the energy of these incident electromagnetic rays. Gamma rays produce scintillations through linear reactions in a thallium activated sodium iodide crystal. These reactions are Compton scattering, photoelectric effect and pair production. Any combination of these reactions which results in absorption of energy of the gamma rays will produce scintillations or "light flashes" in the crystal. The total magnitude of these scintillations is proportional to the gamma ray energy lost in the crystal. The scintillations are detected by a photomultiplier tube optically coupled to the crystal, and then converted to current pulses with amplitudes proportional to the energy of the incident gamma rays. The voltage pulses produced by the detector are amplified by the pre-amplifier and amplifier and are coupled to the single channel analyzer (SCA). The SCA discriminates pulses to be counted against undesired pulses.

CHAPTER II

MICROCOMPUTER SELECTION

In this age of computer revolution, there is a move to more dependence on automated and computer controlled systems. Considerable programmable instrumentation meeting these trends is being put on the market today. In keeping with these trends, engineers and scientists alike are adopting a philosophy of "designed-in expandability and modification" in their instrumentation and data acquisition systems. This inevitably means that most of the instrumentation used for research and development today has at its core a microcomputer system or a microprocessor based controller. Microcomputers certainly offer an excellent opportunity for expandability as additional interfaces can easily be added to the bus and software modified to meet additional requirements.

In the light of these developments, it is of paramount importance for the system engineer to carefully select a microcomputer or microprocessor-based system that will best suit his or her requirements. In addition certain universal characteristics such as high reliability, low power dissipation, small size, easy serviceability and low cost are desirable in any system (3). In this project iterations of selection and evaluation of microcomputers were necessary before a final computer system was adopted.

2.1 HP-IL INTERFACE

The first system considered was an HP-41CV type controller. The main idea behind this was to reconfigure the HP-41CV calculator as the main controller and interface it to the sample changer using the HP-IL loop. HP-IL, the Hewlett-Packard Interface Loop, is a digital communication system designed primarily for portable devices (4). Devices are connected in a circular loop structure with digital messages traveling from one device to the next around the loop in one direction only. HP-IL is a master/slave system. One of the devices on the loop is designated the loop controller and this device has the responsibility of transmitting all commands to the other devices on the loop. The HP 41-C(V.X) can act as an HP-IL controller when it is equipped with the appropriate plug-in module.

Much like the IEEE 488-Bus structure, each device on the HP-IL loop has an address and is designated as either a controller, talker or listener. A device may have one of the three capabilities or may include some combinations of the capabilities. Talkers often have listener capabilities and controllers almost always have both talker and listener capabilities as well. Functionally the HP-IL may be considered as a bit-serial version of the HP-IB (Hewlett-Packard's implementation of IEEE-488) (5).

Message on the loop is sent as a sequence of eleven bits. The electrical connection between one device and the next is a differential, voltage-mode, two-wire balanced line. Both wires float with respect to both devices' ground connections. One wire is reference and the voltage on the other wire is measured with respect to the reference. In this case devices' grounds need not be at the same potential. Bits are encoded using a three level, or bipolar code (5). The voltage difference between the two wires may be nominally -1.5v, 0v, or 1.5v. A logic one is encoded as a high pulse (+1.5v) followed by a low pulse (-1.5v). A logic zero is a low followed by a high. The nominal pulse width is one microsecond and each bit sequence is always followed by a minimum delay time (0v) of about two microseconds.

Each device on the HP-IL loop must completely implement all the HP-IL protocols (talker, listener or controller) for which it has been configured. This implies a separate HP-IL plug-in module for each device - an expensive undertaking. This together with the small memory capability of the HP 41C (without going to cassette tape or disc drives) made the HP-IL loop unsuitable for the project.

2.2 HP87/HP-IB INTERFACE

Recently the Oregon State University (OSU) Department of Nuclear Engineering acquired some HP87 microcomputers that were donated by Hewlett-Packard. It was therefore natural to consider the HP87 next as a possible choice for the microcontroller. The HP-IB interface connects the HP87 microcomputer to the Hewlett-Packard interface bus which conforms to the IEEE standard 488-1978. This is a parallel bus structure that allows the transfer of data and command messages over short distances (6). All devices on the bus must fully implement the IEEE 488 bus protocols. A preliminary design performed for this project accomplished this by using an 8748 single chip microcomputer to handle all local I/O and by using the 8291 GPIB (General Purpose Interface Bus) talker/listener and the 8293 GPIB transceiver to sit on the bus. The 8291 GPIB talker/listener is primarily designed to interface microcomputers to an IEEE 488 digital interface bus and it implements all the standard's interface except for the controller. The 8293 GPIB tranceiver is a high current, non-inverting buffer chip designed to interface the 8291 GPIB talker/listener or the 8292 GPIB controller to the IEEE standard 488-1978 instrumentation interface bus. Each GPIB interface would contain two 8293 bus transceivers (7).

In order to make the HP87 functional as a bus controller, it would require an I/O ROM plug-in module, in addition to a plotter/printer ROM and disc drives. Due to budget limitations, this option had to be dropped.

2.3 INTEL 8748 MICROCOMPUTER BASED DESIGN

Since it appeared that cost limitation was a primary problem, it was decided that designing and building a system from scratch was the only way out. This of course involves an extensive design and construction task. This method, however, allows the greatest versatility in microcomputer function selection as it can be completely tailored to the specific project at hand, with expandability in mind. An extensive review of microprocessors was conducted with emphasis on those for which development systems are available in the OSU Department of Electrical and Computer Engineering. The search was narrowed down to the following: 6802, 6502, 8085, 8086 and the MCS-48 family (8). From these the

8748 was selected.

The 8748 is a single chip 8-bit parallel microcomputer that contains 1K x 8 UV-erasable, user-programmable memory, a 64x8 RAM data memory, 27 I/O lines, and an 8-bit timer/counter in addition to on-board oscillator and clock circuits. The 8748 can be expanded using standard memories and MCS-80/MCS-85 peripherals (20). It is designed to be an efficient controller as well as an arithmetic processor. It exhibits extensive bit handling capabilities as well as facilities for both binary and BCD arithmetic.

The system was completely designed around the 8748 with 2716's as additional program memories and 8243 I/O expanders for handling the sample changer and elevator motor drive hardware. The 8155 256x8 RAM complete with timers was added. Two 8251A USARTs (Universal Synchronous, Asynchronous Receiver, Transmitter) were used for handling serial I/O. The first USART interfaces the ORTEC 874 Quad timer/counter via a TIL111 photoisolator which runs on the 20ma current loop (21). The second interfaces a teletype that uses either the 20ma current loop or the RS232 serial loop. SN75152 dual line receivers and SN74150 dual line drivers were incorporated to meet the EIA standard RS232-C for serial communication (24). Baud rate generators were designed for the transmit and receive clocks of the two USARTS using 74LS393, 74LS11 and CD-4024BE 7-stage counters (9,22,23). The baud rates are selectable on DIP switches from as low as 110 to 9600 baud with an error of 0.16%, which is well within the 6% timing variation standard required for the RS 232 interface (9). The primary clock for the baud rate generators is derived from the T0 output of the 8748 (T0=XTAL/3). For a 6MHz crystal, T0=2MHz.

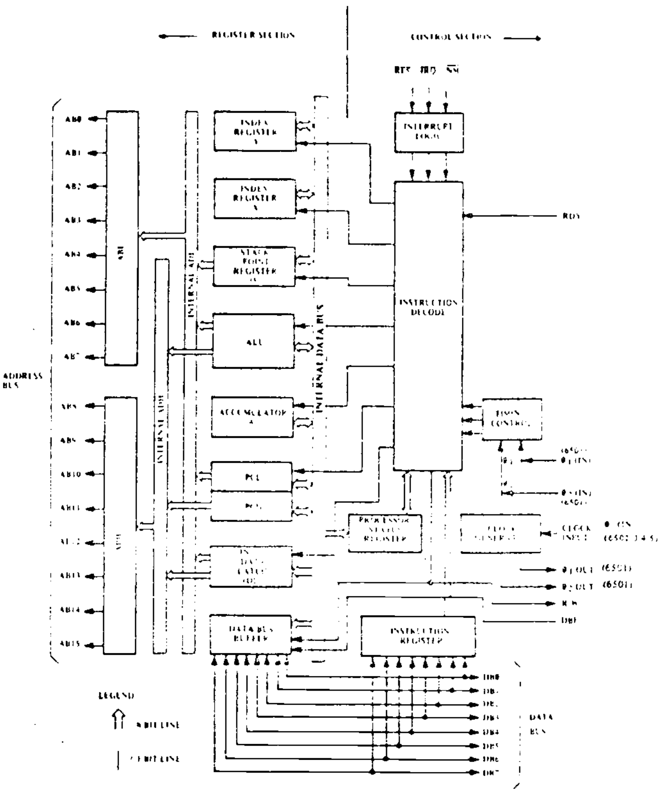
An MM58167A National Semiconductor real time clock (RTC) chip was also included in the design (2). The complete software for the system was written in 8748 assembler language, and part of it had already been installed on the HP64000 development system. At this stage an AIM-65 computer complete with disc drives, monitor and six parallel ports became available. Thus the designs and software developed to date were dropped and a fresh start was made with the AIM-65 system.

2.4 THE AIM-65 MICROCOMPUTER

The Rockwell R6500 Advanced Interactive microcomputer (AIM-65) is a general purpose microcomputer that can serve as a central processor or controller/monitor. The heart of the AIM-65 is an R6502 central processing unit (CPU) that operates at 1MHz to provide a minimum instruction execution time of two microseconds (10). The 6502 microprocessor architecture is shown in Figure 2.1.

A brief overview of the 6502 CPU is now presented. Most operations such as add, subtract and compare are done in the accumulator with the result usually remaining there. The X and Y index registers may be used as temporary data storage or to aid calculation of addresses via indexing or for counting (11). The program counter (PC) is used for holding the address of the next instruction to be executed while the stack pointer holds the low byte of the address in the next available cell in stack memory. The arithmetic logic unit (ALU) performs logical operations dictated by the content of the instruction latch and the data latch holds incoming/outgoing data values from/to the data bus at appropriate times.

Apart from the keyboard/display and the 6522 VIA (discussed later), the Forethought Products' (now Versa Logic) version of the AIM-65 microcomputer also comes equipped with floppy disk interface, video display generator and a memory mate expansion board. The STD FDI-1 is a versatile full function floppy disk interface board for the STD bus (12). It features the Motorola MC6843 floppy disk controller designed with MOS (N-channel, silicon gate) technology. It is 5 1/4" and 8" disk drive compatible; however, it comes equipped with



NOTE 1 CLOCK GENERATOR IS NOT INCLUDED ON MCS6501
2 ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE MCS650X PRODUCTS

MCS 6502 8-Bit Microprocessor

Manufacturer	MOS Technology
Device Technology	NMOS
Data word width	8 bits
Addressing range	65,536 words
Instruction width	8 bits
Instruction count	56
Clock frequency	20 kHz-2 MHz
Package	40-pin DIP
Power requirements	5 V at 140 mA

Programmer's Eye View

8 bits	Accumulator A
8 bits	Index register X
8 bits	Index register Y
16 bits	Program counter PC
8 bits	Stack pointer SP
8 bits	Status register

[illegible]

MS thesis: H. I. SALEH			
Title THE MOS TECHNOLOGY 6502 MICROPROCESSOR			
Size	Document Number		REV
A	Figure 2.1		01
Date:	November 17, 1987	Sheet	1 of 1

Figure 2.1 The mos technology 6502 microprocessor

a 5 1/4" disk drive. The VID 64/80 is a memory-mapped video display board for the STD bus (13). It has a flexible screen format, programmable character size, alphanumeric and graphic characters and a 2K on-board RAM. It utilizes the Motorola's MC6845 CRT controller (CRTC). The memory-mate expansion board has additional RAM, on-board I/O ports, PROM sockets and a tone generator (14).

2.5 6522 VIA

Most of the interface for this project utilizes the spare 6522 VIA (Versatile Interface Adaptors) on the AIM-65 and the two on the memory-mate expansion board. These are programmable I/O chips with 16 8-bit registers. The AIM-65 VIA occupies a 16 byte block of addresses from A000 through A00F. The memory-mate's first VIA (labeled IC57) occupies locations 9F80 through 9F8F while the second (IC58) sits at locations 9F90 through 9F9F. The VIA is partitioned into two 8-bit ports (port A, port B) with associated handshake lines for each port and two timers. The I/O location summary for the ports are given in Table 2.1.

TABLE 2.1 VIA I/O LOCATION SUMMARY

ADDRESS	FUNCTION
A000-A00F	6522 AIM-65's VIA
A000	port B data register
A001	port A data register
A002	DDRB-data direction register B
A003	DDRA-data direction register A
A004	T1L-for timer 1
A005	T1CH-for timer 1
A006	T1LL-for timer 1
A007	T1LH-for timer 1
A008	T2L-for timer 2
A009	T2CH-for timer 2
A00A	SR-shift register
A00B	ACR-auxiliary control register
A00C	PCR-peripheral control register
A00D	IFR-interrupt flag register
A00E	IER-interrupt enable register
A00F	port A data register (no handshake effect)

Exactly the same functions apply to IC57 (9F80-9F8F) and IC58 (9F90-9F9F). For the organization, processor interface, and peripheral interface of the VIA refer to reference (11).

CHAPTER III

POWER SUPPLY AND PERIPHERALS SELECTION

3.1 POWER SUPPLY AND TRIAC

The sample changer came equipped with its own transformer (Figure 3.1) and Triac/low voltage power supply card (Figure 3.2). Fullwave rectification is performed using four EM506 diodes and two offboard 1500mf 50VDC capacitors. The system initially designed for +/- 15v dc was found to give +/- 18v dc unloaded. A 1A fuse is used on the Triac/lv card for overload protection on the -15v dc line. Also incorporated on the Triac card are CDC 2N3638 PNP and CDQ10050 CDC PNP transistors to provide smaller voltages of -3v and +6v respectively. However, these voltages are not utilized in this project.

An auxiliary power supply card utilizing standard voltage regulators was designed to interface with the Triac/lv card (Figure 3.3). This parallel arrangement provides up to 3.5 amperes at -15v, -12v, +15v and +5v for the TTL logics, Real Time Clock, UART, OP AMP (operational amplifiers), LEDs (light emitting diodes)

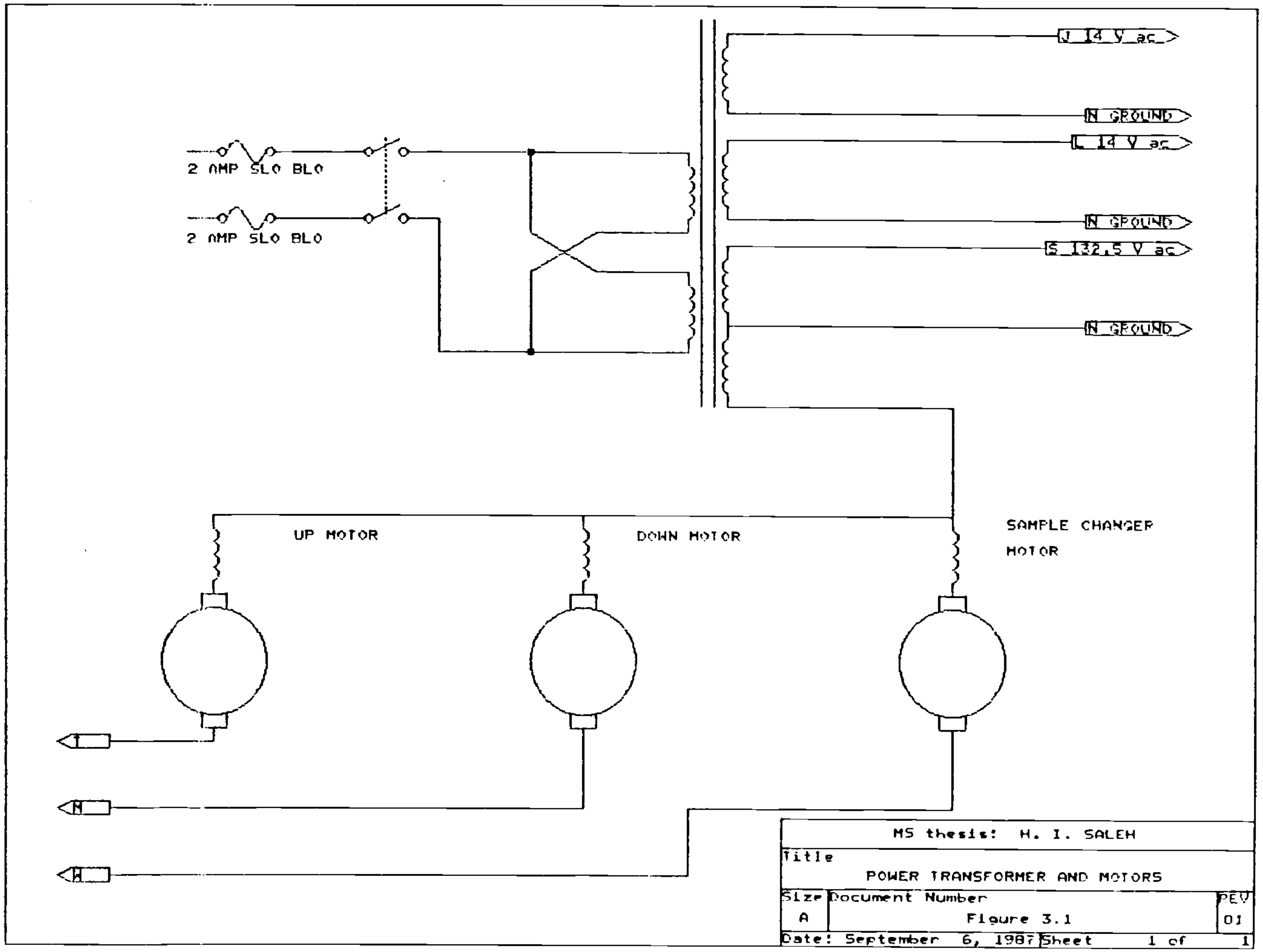


Figure 3.1 System power transformer

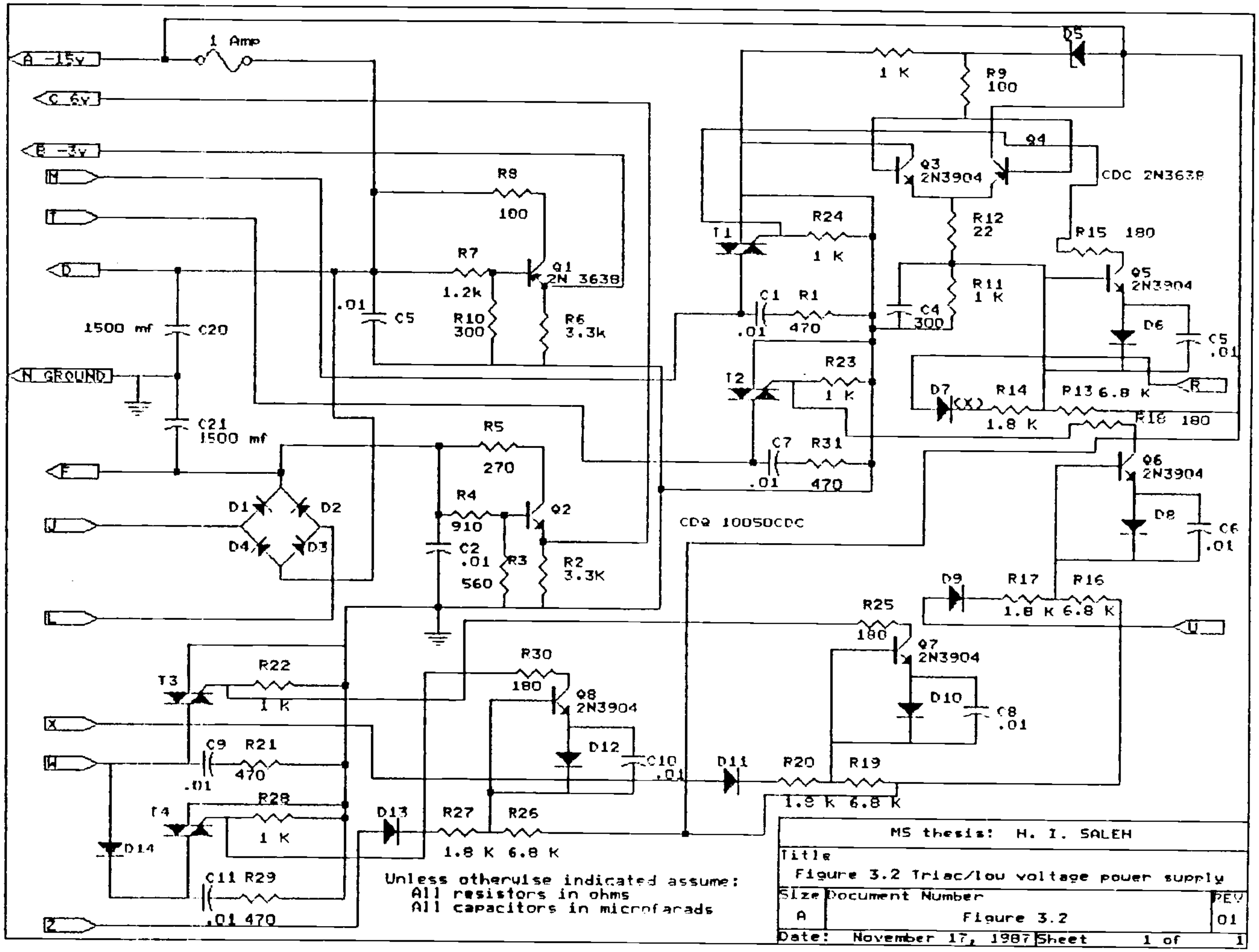


Figure 3.2 Triac/low voltage power supply

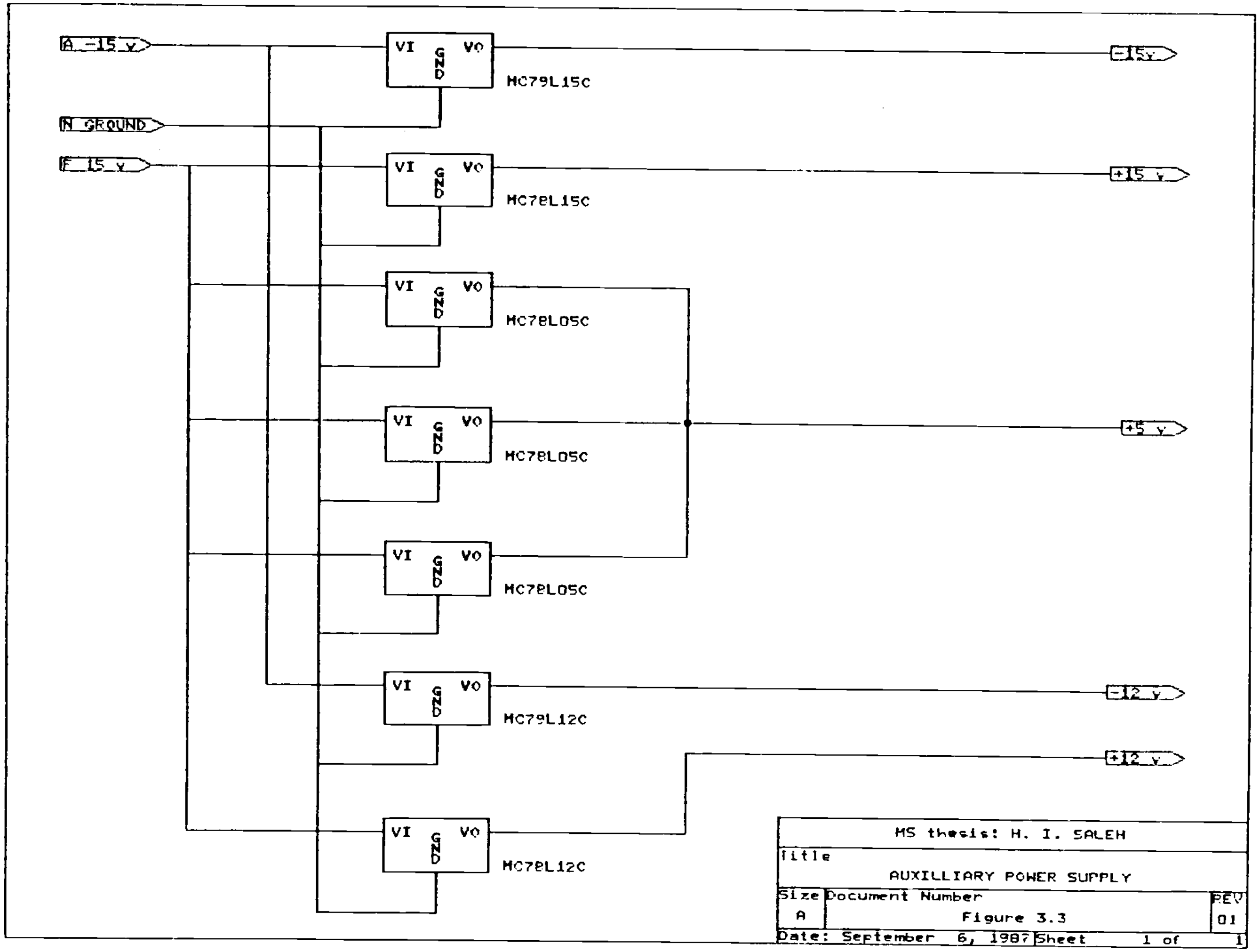


Figure 3.3 Auxilliary power supply

and other circuitry to interface the peripherals completely to the AIM-65 microcomputer.

40430 RCA CTR04 TRIACS are used for controlling the sample motors. Triacs are bi-directional triode thyristors with two main terminals and a gate (Figure 3.4). The main terminal 1 (MT1) is the reference point and the voltage at MT2 is reckoned either positive or negative with respect to MT1 (15). The four triggering modes for the TRIAC are:

1) MT2+, gate + ; I+ ; First quadrant, positive gate current and voltage,

2) MT2+, gate - ; I- ; First quadrant, negative gate current and voltage,

3) MT2-, gate + ; III+ ; Third quadrant, positive gate current and voltage,

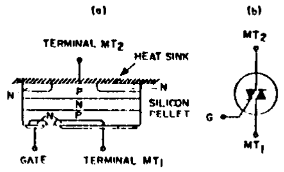
4) MT2-, gate - ; III- ; Third quadrant, negative gate current and voltage.

Triac sensitivity is greatest in I+ and III- modes, slightly lower in the I- mode and much less in the III+ mode. The design made by Nuclear Chicago (no documentation provided) utilizes negative gate current and voltage firing in the I- and III- modes.

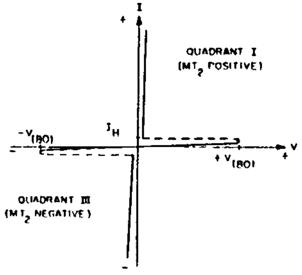
3.1.1 GATE FIRING

To fire the gates and trigger the Triacs, the 2N3904

THE TRIAC



THE TRIAC; (A) SIMPLIFIED PELLET STRUCTURE, (B) CIRCUIT SYMBOL



AC VOLT-AMPERE CHARACTERISTIC OF THE TRIAC

MS thesis: H. I. SALEH		
Title TRIAC STRUCTURE SYMBOL AND CHARACTERISTICS		
Size A	Document Number Figure 3.4	REV 01
Date: September 8, 1987	Sheet 1 of 1	

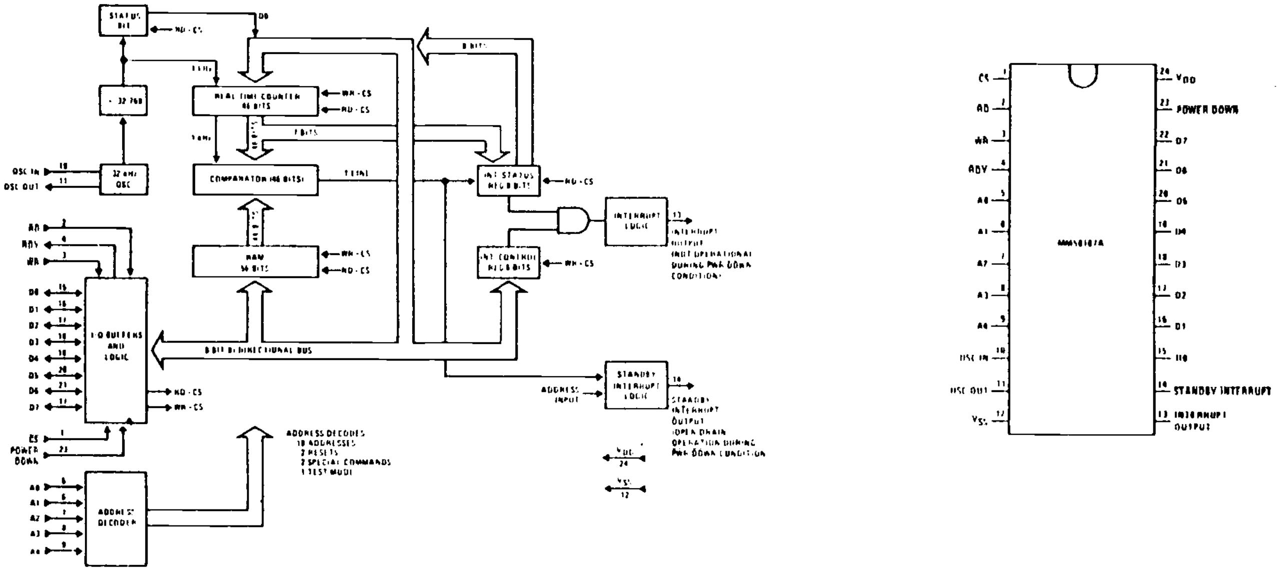
Figure 3.4 Triac pellet structure, circuit symbol and ac volt-ampere characteristics

NPN transistors must be turned on (Figure 3.2). As an illustration, consider the gating control for the Down motor Triac. The emitter of Q5 is at -11 volts. To turn off Q5 and hence the gate, the base voltage at Q5 should be -11 volts. The current through R13 is IR13=(-11-(-17.5))/6.8K = 0.96mA. The voltage drop across R14 is VR14=1.8K * 0.96=1.72 volts. The voltage at point x is VX= -11+1.72 = -9.28 volts. Allowing for 0.6v across D7, input voltage to D7 is Vin=-9.28+0.6 = -8.68 volts. Hence for Vin < -9 volts the Triac T1 is turned off. It will be triggered for voltages appreciably higher than -9V, e.g., 0 volts. The same applies to the other TRIACS.

3.2 REAL TIME CLOCK (RTC)

A real time clock is a hardware circuit or hardware/software combination that accurately records time with respect to an external observer (16). This is a very handy peripheral in a control applications environment where a microcomputer monitors a number of physical parameters and triggers a series of sequentially timed control outputs in response to certain changes in the parameters. This is of course in addition to making available timed and dated copies of data and parametric settings for data acquisition experiments. Such a real time clock is superior to software timing and heartbeat-interrupt with regards to ease, accuracy and nonvolatility. In these latter examples, the time of the day is kept in software and the clock works only when the computer is powered. It is therefore impossible to keep the clock running all the time without keeping power applied to the processor and part of the program memory, which can be an expensive undertaking. Hence the easiest solution is a separate hardware time-of-day real time clock interfaced to the processor but running independently. Such a clock should keep track of the time of the day to a resolution of milliseconds and should, with battery backup, never need to be reset. Additional features of such an RTC should include variable-rate processor interrupts and alarm clock interrupts.

These features have been made available by National Semiconductor Corporation's two CMOS (complementary metal-oxide semiconductor) LSI (large scale integrated circuits) devices, the MM58167A and MM58174A (2). These two devices are designed for direct connection to the control and data buses of most microprocessors. Figure 3.5 and Figure 3.6 show the block diagram of the MM58167A and the MM58174A, respectively.



MS thesis: H. I. SALEH

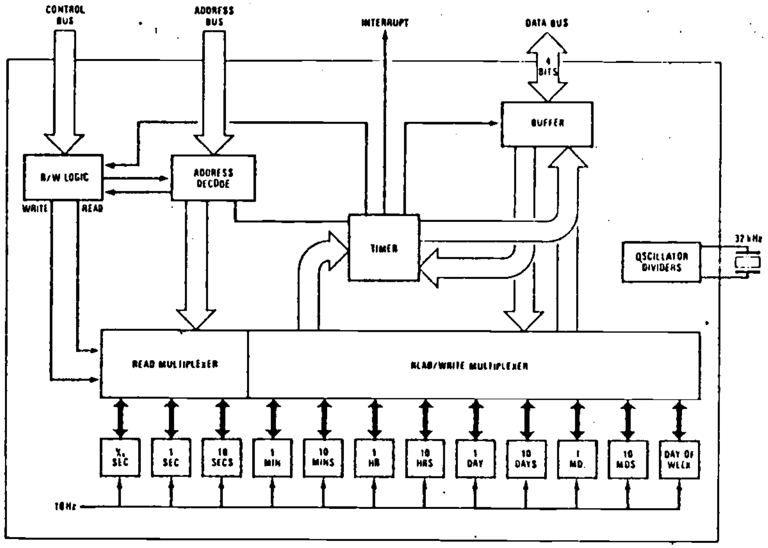
Title

BLOCK DIAGRAM AND PIN OUT OF MM58167A

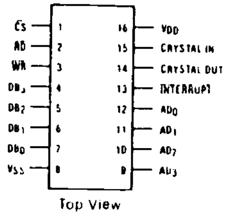
Size	Document Number	REV
A	Figure 3.5	

Date: September 8, 1987 Sheet 1 of 1

Figure 3.5 Block diagram and pin out specifications of the National Semiconductor MM58167A real time clock



Connection Diagram



reference (2)

MS thesis: H. I. SALEH		
Title BLOCK DIAGRAM AND PIN OUT OF MM58174A		
Size A	Document Number Figure 3.6	REV
Date: November 17, 1987	Sheet 1 of 1	

Figure 3.6 Block diagram and pin out specifications of the National Semiconductor MM58174A real time clock

3.2.1 RTC MM58167A

The MM58167A is packaged in a 24-pin DIP (dual-in-line package) and contains a 48-bit (14-digit) counter chain clocked from a 32.768Hz crystal-reference oscillator. The MM58167A can track and communicate to the processor the time in any increments from 1/10,000 seconds to months. It has 56 bits of on-chip RAM (random access read/write memory) that can be used to store any desired quantity of time or data while the system is powered-down provided it is supplied with backup power from a battery. It can therefore be used in the alarm clock mode to store a value to be compared to the real time counter (either in its entirety or against individual digits in the counter). When a match occurs between the storage latch and the counters a maskable interrupt line called the standby interrupt is set active low.

Another line called the interrupt output provides the heartbeat interrupt described earlier. This may be programmed to provide clock ticks at seven regular intervals (ten times per second (10Hz), once per second (1Hz), once a minute, once an hour, once a day, once a week, and once a month) and when a comparison match occurs between the storage latch and the real-time counter.

The MM58174A which is a 16-pin integrated circuit is less versatile than the MM58167A. It derives its timing from a 32,768Hz oscillator like the MM58167A, but only counts time intervals from 1/16 seconds through months. It also does not have a comparison match interrupt but does have a tick interrupt programmmable for intervals of 1/2 second, 5 seconds or 60 seconds. For this design project the more versatile MM58167A was adopted.

3.3 QUAD COUNTER/TIMER

In an automatic counting system it is a necessity to have modern and reliable counters that are not only NIM-standard (for compatibility to other nuclear instrumentation), but also have the capability of being computer programmable for remote data acquisition. The EG&G ORTEC 874 is one such general purpose counter. It provides three 8-decade counters and one 8-decade presettable counter with internal time base. The presettable counter portion may be used as a counter or as a timer for the other three counters (17).

The 874 comes in a double-width NIM module with remote control capabilities via the IEEE 488 bus or the 20mA current loop interface. For the purposes of this project, it was decided to use the serial 20mA current loop communucation standard for controlling the 874 from the AIM-65 microcomputer. The communications card of the 874 is built around the 8085 microprocessor that utilizes its SID (serial input data) and SOD (serial output data) pins at selectable baud rates for communicating over the 20mA current loop. The 874 is designated active as it supplies the 20mA current for the loop. The communications rate for this mode are 110, 300, 1200 and 2400 baud and are DIP selectable on switch S2. The use of a serial device with the 874 for remote data acquisition allows for control of individual or grouped counter functions from the keyboard. This may also provide a hard-copy of the printout of results (as in the case of a teletype) or parameters used in setting up the various functions of the instrument.

The serial mode of communication is good for up to several hundred feet at high baud rate or up to several thousand feet at lower baud rates.

3.4 DRIVE MOTORS

The sample changer system came equipped with three motors: one for driving the conveyor belt, and two for the elevator mechanism (one up and the other down). As there is virtually no documentation on these, it is presumed that they are induction type motors as they run from an ac power supply. The motors are controlled by

Triacs, the gates of which may be fired at the instance induction is desired. The triggering system for the gate circuits can be interfaced to a computer for remote and software control.

3.5 CONTACT SWITCHES

There are seven contact switches (CTS) mounted on the sample changer for sensing the various states of the system. CTS1 is used for group plug detection. CTS2 and CTS3 sense when the elevator is at the bottom (in the detector) and at the top (out of detector), respectively. CTS4 senses the vial position indexed at the elevator while CTS5 indicates when nonstandard vial detection data (NSV data) are valid. CTS6 is used for reset: it senses the #1 vial under the elevator when the conveyor belt is running. CTS7 is used for NSV detection.

These contact switches can be wired to produce TTL logic signals for each of the states that are sensed.

The design of the control and interface circuits for these peripherals is presented in Chapter IV.

CHAPTER IV

COMPUTER INTERFACE DESIGN FOR PERIPHERALS

In this chapter design considerations for the computer interface circuitry to the selected peripheral devices are presented. All the interfaces to the AIM-65 are done via the VIAs (versatile interface adapters) located at addresses A000 through A000F, 9F80 through 9F8F, and 9F90 through 9F9F.

4.1 INTERFACING THE REAL TIME CLOCK (RTC)

The application notes for the National Semiconductor's MM58167A microprocessor Real Time Clock are given in references (2, 16). The RTC is interfaced to the AIM-65 via the VIA2 located at addresses 9F80 through 9F9F (Figure 4.1). The data lines of the RTC (D0 through D7) are connected to port A of the VIA2 (PA0 through PA7) while the port B lines (PB0 through PB4) are connected to the five address lines A0 through A4 of the RTC. To read any register in the clock, interface circuitry must place signals on the RD and CS lines while the proper address appears on the address lines; similarly, to write data into the clock registers, the WR

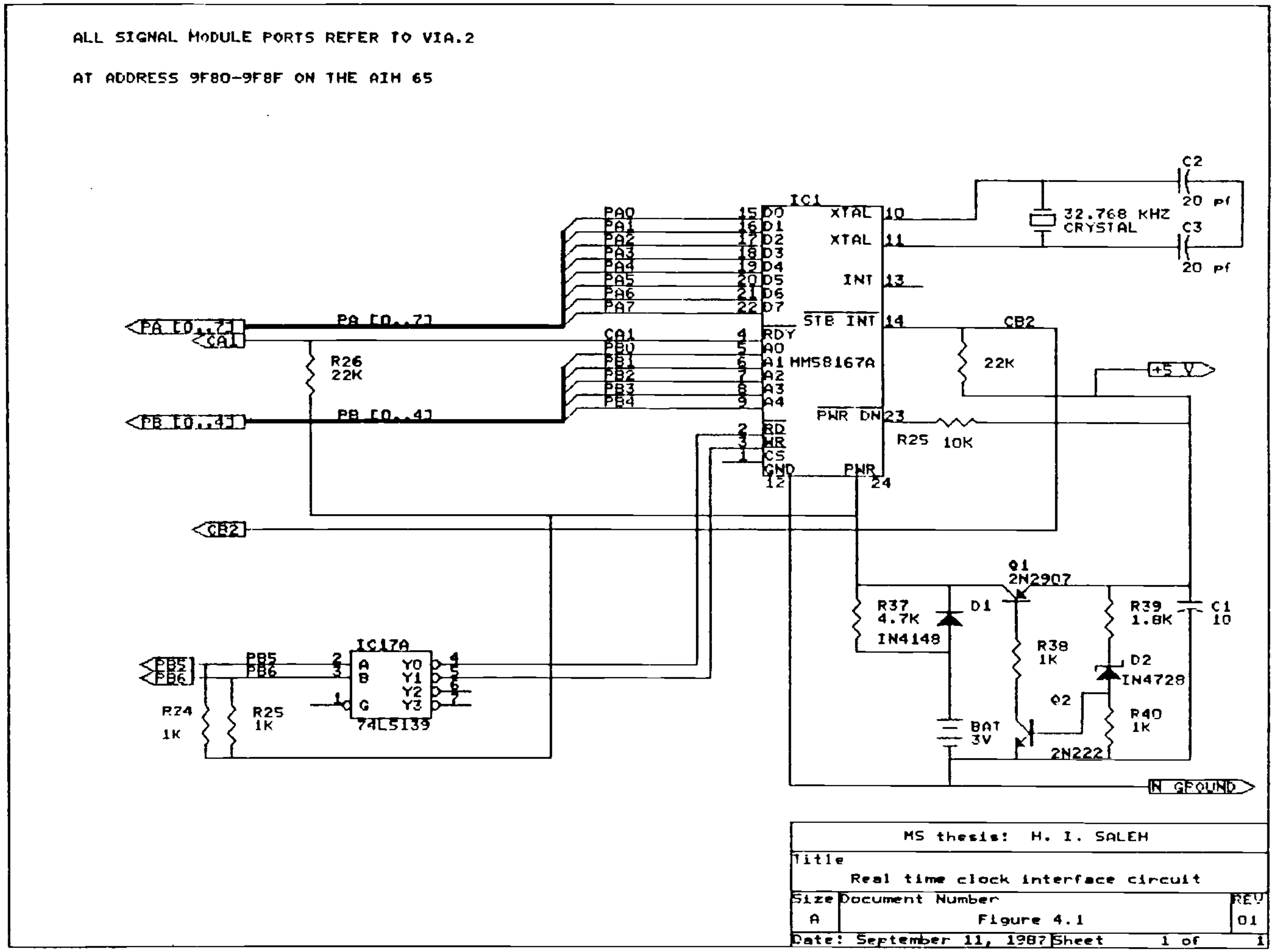


Figure 4.1 Real time clock interface circuit with the VIA2 (9F80-9F8F)

and CS lines must be enabled while the address appears on the address lines. The data bus provides the path for data in and out of the counters and the latches. The five address lines may be used to activate any of the 24 counter and memory functions of the RTC.

An SN74ALS139 (IC17) dual 2-line to 4-line decoder/demultiplexer is used to decode PB5 and PB6 to provide the RD, WR and CS signals and two unused tri-state lines for the RTC. This is illustrated in Table 4.1.

TABLE 4.1: DECODING LOGIC FOR THE RTC

		PB5	PB6	CS	RD	WR	
CS=PB5	(	0	0	0	0	1	
	(	0	1	0	1	0	
		1	0	1	1	1	) IDLE/TRI-STATES
IDLE=PB5		1	1	1	1	1	)

CS = Chip select (active low)

RD = Read data (active low)

WR = Write data (active low)

PB5 is connected to the CS line so that the RTC is enabled whenever a logic 0 is placed on PB5 by the AIM-65. Similarly the RTC is disabled or tri-stated whenever PB5 is strobed to a logic 1 state. Whenever the RTC is enabled and PB6 is strobed low, the RD function is enabled while the WR is enabled for PB6 held high. The Ready signal appears on RDY which is connected to the CA1 input of the VIA. This is an open drain output which will pull low and remain low at the start of each read or write cycle until valid data from a chip read appears on the bus or data on the bus is latched-in during a write.

By referring to the read and write cycle timing diagrams in the reference (2, 16), a typical read/write protocol may be formulated as follows:

READ CYCLE

1) Send valid address on PB0 through PB4
2) Send RD, CS low
3) Wait for RDY low-high transition
4) Read data

WRITE CYCLE

1) Send valid address
2) Put out data on port A
3) Wait for RDY low-high transition
4) Data latched

In these protocols, it should be noted that the RDY line is used for handshaking and the port A of the VIA is configured either as an input or an output depending on whether we are reading/writing data from/to the RTC.

The MM58167A has the ability to operate from battery power when the main power system is down. It will keep track of real time when supplied with power at voltages down to 2.2volts. In this mode only 20 microamperes of current is required, dissipating 44 microwatts of power which may be supplied from two standard 1.5volt batteries. Figure 4.1 shows the circuitry for operating the RTC on battery power when the computer and main power supply are turned off. The transistors Q1 and Q2 are used as voltage sensitive switches. When the system power is on at +5volts, the .6v at the anode of the 3v-zener diode D2 turns on Q2 which forces Q1 into conduction and power is supplied to the RTC. The diode D1 blocks any large current flow into the batteries although it receives a trickle through the 4.7k resistor R37. In this mode of operation, the RTC requires about 12mA of current. When the power is off and the +5v line drop to 0v, Q2 is off and Q1 opens to prevent the battery from sourcing current onto the system's power bus. Current flows from the battery through D1 into the RTC and the PWRDN (power down, pin 23) input senses the low voltage condition of the power bus and causes the clock to enter the powered-down operating mode. In the powered-down mode, the RTC's three-state I/O lines enter the high-impedance mode, effectively disconnected from the computer, and the current drawn from the power source is reduced from 12mA to 20 microamperes.

The power down circuitry, crystal and capacitors for the RTC are mounted on the system's regulated power supply card. R23 and R24 are 1K pull-up resistors that ensure that these lines float-high when the AIM-65 power is turned off to prevent spurious data from being written to the RTC.

4.2 INTERFACING THE ORTEC 874 TIMER/COUNTER

The 874 quad counter/timer is interfaced to the AIM-65 via the VIA located at addresses 9F90 through 9F9F on the memory mate expansion board. Initially the interfacing was performed with a UART and two TIL 111 optical isolators (Figure 4.2). Refer to reference (18) for the UART application notes. The UART converts parallel data to/from the computer to TTL serial, and the TIL 111 devices provide optical isolation between the computer/controller and the 874 counter/timer to take into account different instrument grounds.

The received data (pins 5 to 12) of the UART are

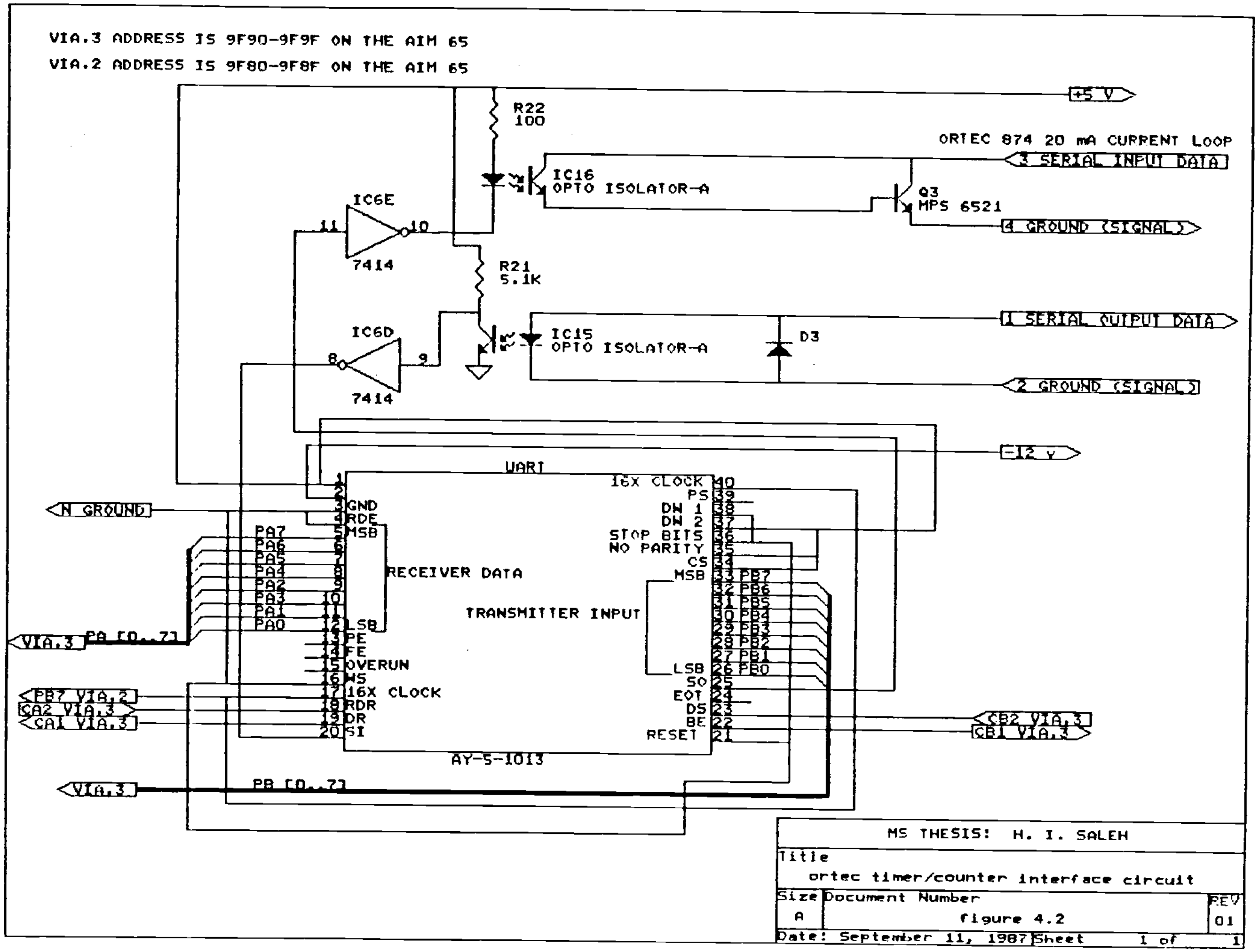


Figure 4.2 Ortec timer/counter interface circuit with a UART

connected to port A (PA0 through PA7) and the transmit data (pins 26 through 33) are connected to port B (PB0 through PB7) of the VIA. The PMOS version of the UART (AY-S-1013) which was used in this design requires two power supplies: a +5v at pin 1, and -12v at pin 2. Complete data handshake between the UART and the VIA was implemented via the control lines CA1, CA2, CB1, CB2 of the VIA and the received data available (pin 19), reset data available (pin 18), transmitter buffer empty (pin 22) and data strobe (pin 23) of the UART.

4.2.1 RECEIVER HANDSHAKE PROTOCOL

The received data available flag goes to a logic 1 when an entire character has been received by the UART and transferred to the receiver holding register. This causes the CA1 flag in the interrupt flag register of the VIA to be set. When a software polling is conducted by the computer, the CA1 flag is sensed high. CA1 is then cleared, the received data (which is placed on the output lines pins 5 through 12 of the UART by holding the received data available pin 4 low) are then read on port A of the VIA. The read operation on port A also sends a one cycle low output pulse (1 microsecond) on the CA2 output line. This causes the reset data available line (pin 18) to go to a logic 0 momentarily which resets the received data available flag at pin 19. This completes the read operation on the UART.

4.2.2 TRANSMITTER HANDSHAKE PROTOCOL

This is configured similar to the receiver protocol described above. In this case the transmitter buffer empty flag at pin 22 goes to a logic 1 state when the UART is ready to receive data from the computer for transmission. If the buffer is full, the flag is held at a logic 0 (TBMT flag=0). A low data strobe at pin 23 will initiate transmission of a full ASCII character. So the computer senses TBMT flag=1 on CB1, writes data through the port B to the UART and initiates data transmission by the UART by sending a low strobe on CB2 with the write.

It should be noted that the peripheral control register of the VIA is software configured for the selected modes of operation of the control lines CA1, CA2, CB1 and CB2.

4.2.3 CLOCKS FOR THE UART

The receiver and transmitter clock lines for the UART (at pins 17 and 40, respectively) were tied and driven from PB7 of 9F9X (i.e. the VIA located at addresses 9F90 through 9F9F). The baud rate is selected on s1 and s2 of the DIP connected to PA3 and PA4 of A00X (Figure 4.3). The possible baud rates are indicated in Table 4.2.

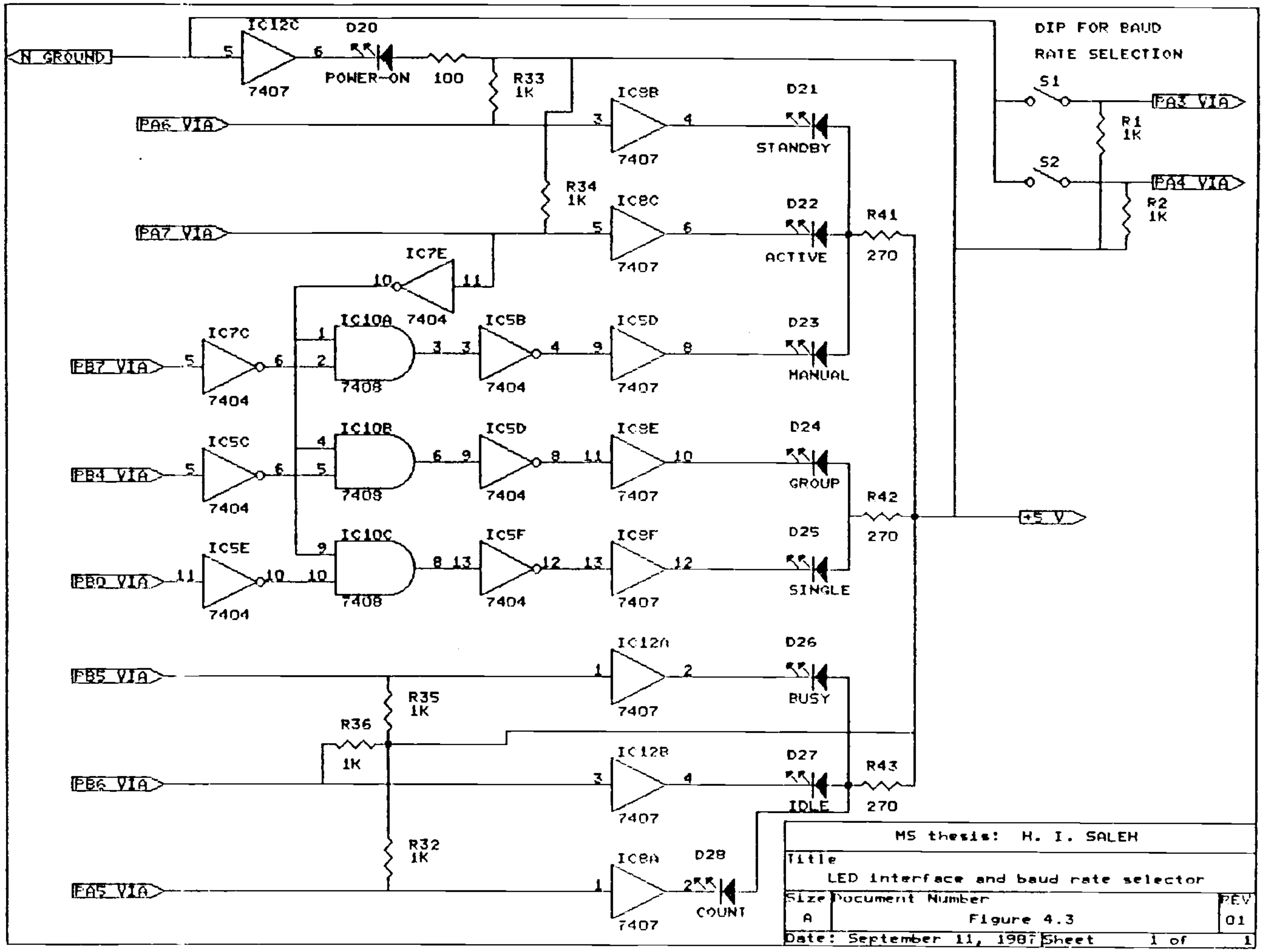


Figure 4.3 LED interface and baud rate selector (A000-A00F)

TABLE 4.2: BAUD RATE SELECTION

S2	S1	BAUD RATE	CLOCK=16 X BAUD RATE	PERIOD MICRO SEC	PULSE WIDTH =1/2 PERIOD	% ERROR
0	0	110	1760	568	284	.032
0	1	300	4800	208	104	.16
1	0	1200	19200	52	26	.16
1	1	2400	38400	26	13	.16

Following a power-on reset, the computer reads S1S2 to determine the user selected baud rate and loads the timer 1 counter and latch with indicated pulse widths minus the overhead count of 2. The auxiliary control register of the 9F8X VIA was software configured to set timer 1 in the free running mode with output on PB7 enabled. In this mode, the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. The timer then automatically transfers the contents of the latch into the counter and the cycle starts again. The result is a continuous series of square waves on PB7 whose frequency is not affected by variations in the processor interrupt response time.

4.2.4 SOFTWARE UART

Due to the I/O pin limitations encountered on the VIAs, the hardware UART was replaced by a software implementation. The details of these are given in Chapter V. Figure 4.4 shows the current interface circuitry for the ORTEC 874. The same character length as for the hardware UART was maintained (7 bit ASCII, no parity). The TTL serial output from the AIM-65 comes from PB5 and is sent to the 874 via the TIL isolator IC16. The output from the 874 goes to the AIM-65 via PB6. Figure 4.4 also shows the interfacing of the

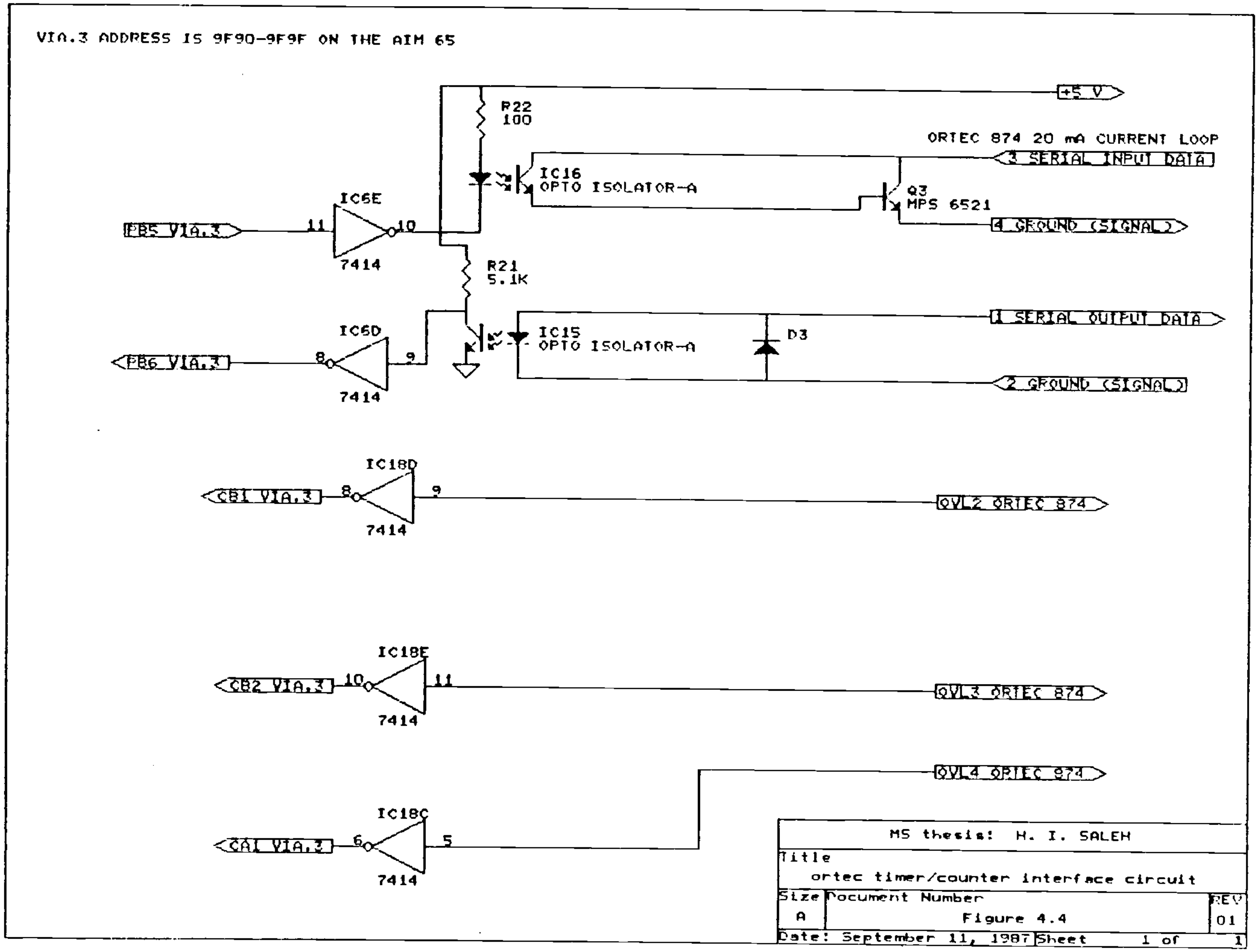


Figure 4.4 Ortec timer/counter interface circuit with a software UART

counter overflow signals from the 874 to the AIM-65 via CB1, CB2 and CA1 of the 9F9X VIA. These overflow signals are pulses of 500 nsec duration that have to be latched into the interrupt flag registers until the AIM-65 is ready to service them.

4.2.5 OPTICAL ISOLATION/20mA LOOP

The serial output of the 9F9X VIA at PB5 (Figure 4.4) is buffered by a 7414 Schmidt inverter (IC6) and connected to pin 2 of the TIL 111 (IC16). When the serial output goes high (logic 1), the 7414 output goes low. A forward current of about 32mA is sourced through the photo-diode, and the photo-transistor is turned on. The photo-transistor is connected to Q3 in a Darlington configuration; hence Q3 is forced into saturation (maximum conduction) and the 874 counter/timer sinks 20mA of current through the collector and emitter of Q3 to ground. When a logic zero appears on the serial output PB5, however, the output of IC6 is held high, hence the photo-diode, photo-transistor and Q3 are all turned off and no current flows from the 874. In this manner TTL logics 1 and 0 are translated into a current flow of 20mA and no current, respectively. The 874 then translates these current flows back to TTL logic levels and they are fed into the serial input data pin of the 8085 microprocessor on board the 874 communications card.

Next consider the sequence of events that occur when the 874 is sending rather than receiving data. As usual the 874 first translates the TTL output of its SOD pin into current. When the 8085 of the 874 sends a logic 1, 20mA of current is applied to TIL 111 (IC15) which turns on the photo-transistor. Current is then sourced through R21 to ground, the voltage input to the 7414 (IC6) drops to a logic 0, and its output goes to a logic 1 which is then applied to the serial input PB6 of the VIA. When a logic zero is sent from the 8085, no current flows from the 874, the photo-transistor is off and the input to the 7414 (IC6) is high while its output is at logic 0.

The planar diode D3 connected to TIL 111 (IC15) is used for reverse polarity protection.

4.2.6 INTERVAL TIMING

Since the serial poll status byte of the ORTEC 874 is only available from the IEEE 488 bus, a rear panel BNC connector is hooked-up to the interval line to provide a positive level signal (interval timing) through the duration of each counting session. This signal is interfaced to the AIM-65 through CB2 of the A00X VIA (Figure 4.3).

4.3 INTERFACING THE MOTORS AND CONTACT SWITCHES

Interfacing the motors involved designing circuits that can be used to fire the gates of the triacs as discussed in Section 3.1.1 under control of the AIM-65. It should be noted that the computer system can only provide TTL level signals; hence buffering and current amplification is required before these can be used to fire the gates of the triacs. The contact switches have to be debounced and their positions translated into TTL signals for input to the computer interfacing ports.

Most of the interfacing of the sample changer system is done with the VIA located at addresses A000 through A00F (Figure 4.3). This VIA is highly loaded as both inputs and outputs are mixed on port A which should normally be configured only as inputs.

First consider Figure 4.5. CTS2 (contact switch number 2) is debounced using IC2 (a 7474 D-type flip-flop) with its override/highest priority inputs (presets and clear) which are active low. When pin B2M of CTS2 is open, a logic 1 voltage level is applied to the preset input of IC2 at pin 4, and pin U17M is grounded. Hence the clear input at pin 1 is active and the output Q is cleared to a logic 0 while $\underline{Q}$ is set to a logic 1. The $\underline{Q}$ output also called CI1 (computer input 1) is connected to PA0 (Figure 4.3). This indicates to the

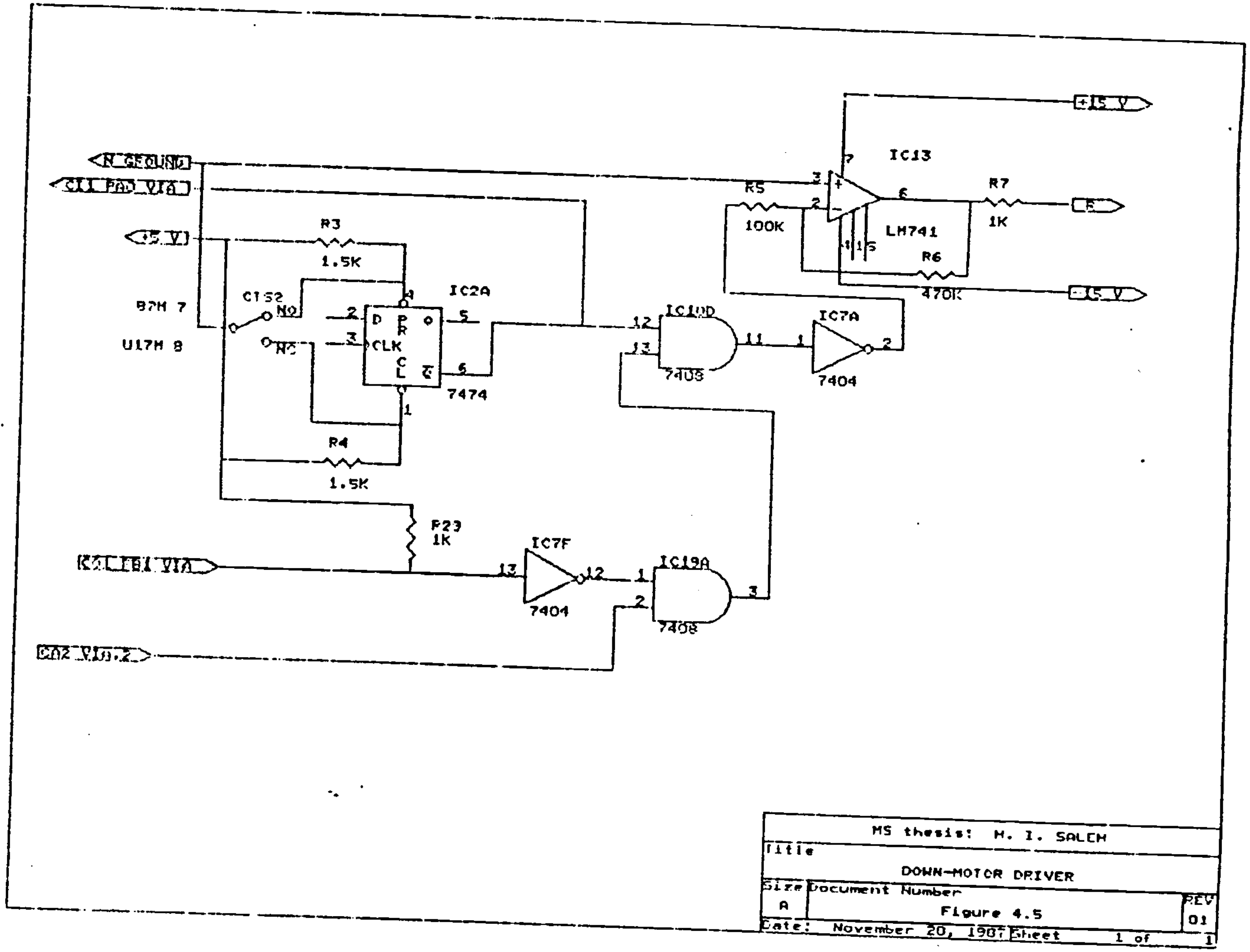


Figure 4.5 Down-motor driver

computer that the elevator is out of the detector and the down-motor may be enabled if desired. If a sample is to be lowered into the detector (under software control), the motor control line CA2 (of 9F8X) is held high, and CO1 (computer output 1) is held low. CO1 is inverted using IC7 and the two high signals are ANDed using IC19 and its output is further ANDed with CI1 at IC10. The high output is inverted using IC7 and applied to the inverting input of the operational amplifier (OP AMP, IC13). The op amp is set up to have a gain of R6/R5=4.7 but since the input is zero, the output is very close to zero except for the offset (25,26). As discussed in Section 3.1.1, a voltage level appreciably above -9v will fire the triac. Hence Q5 is turned on (Figure 3.2) and T1 is fired. The down-motor is then activated and continues to run until the sample is lowered completely into the detector at which time terminal B2M of CTS2 is closed, the preset input is active, Q goes high and $\underline{Q}$ goes low. When $\underline{Q}$/CI1 goes low, the output of IC10 is low, output of IC7 is high (close to 5v) and the output of IC13 is about -14v. This turns off Q5 and T1 (Figure 3.2) and the motor stops. R3 and R4 (Figure 4.3) limit the current sunk to ground CTS2 while R29 is a pull-up resistor which guarantees that CO1 floats high when the computer is turned off. R29 together with IC7 and IC19 form the crash-protect circuitry of the motor drive mechanism.

The operation of the up-motor circuit (Figure 4.6) is identical to the one discussed for the down-motor. In this case CTS3 is used for sensing the states of the elevator rod position. The sample changer motor drive circuitry shown in Figure 4.7 is similar to those of Figures 4.5 and 4.6 except that no contact switch is used directly to effect hardware control of the motor drive mechanism. In this case the first input of IC11 is permanently enabled-high.

Five contact switches: CTS1, CTS4, CTS5, CTS6 and CTS7 are used for control, reset, indexing and state detection of the vials on the conveyor as the sample changer motor is running. The positions of these switches are basically translated into TTL signals that are fed directly into the VIA inputs for software manipulation and control. CTS1 is used for group plug detection. A group plug is a unique plug used for indicating the beginning or end of a group of samples on the conveyor belt. A fixed set of parameters is used for analyzing samples in such a group. When a group plug is indexed at the elevator, the CTS1 prime terminal X20CL is grounded and the output of IC6 (7474 Schmidt invert) CI5 goes to a logic 1. This transition sets the CA2 flag of the VIA. A CI3 low to high transition sets the flag of the CA1 input line to reflect the position of CTS4. This

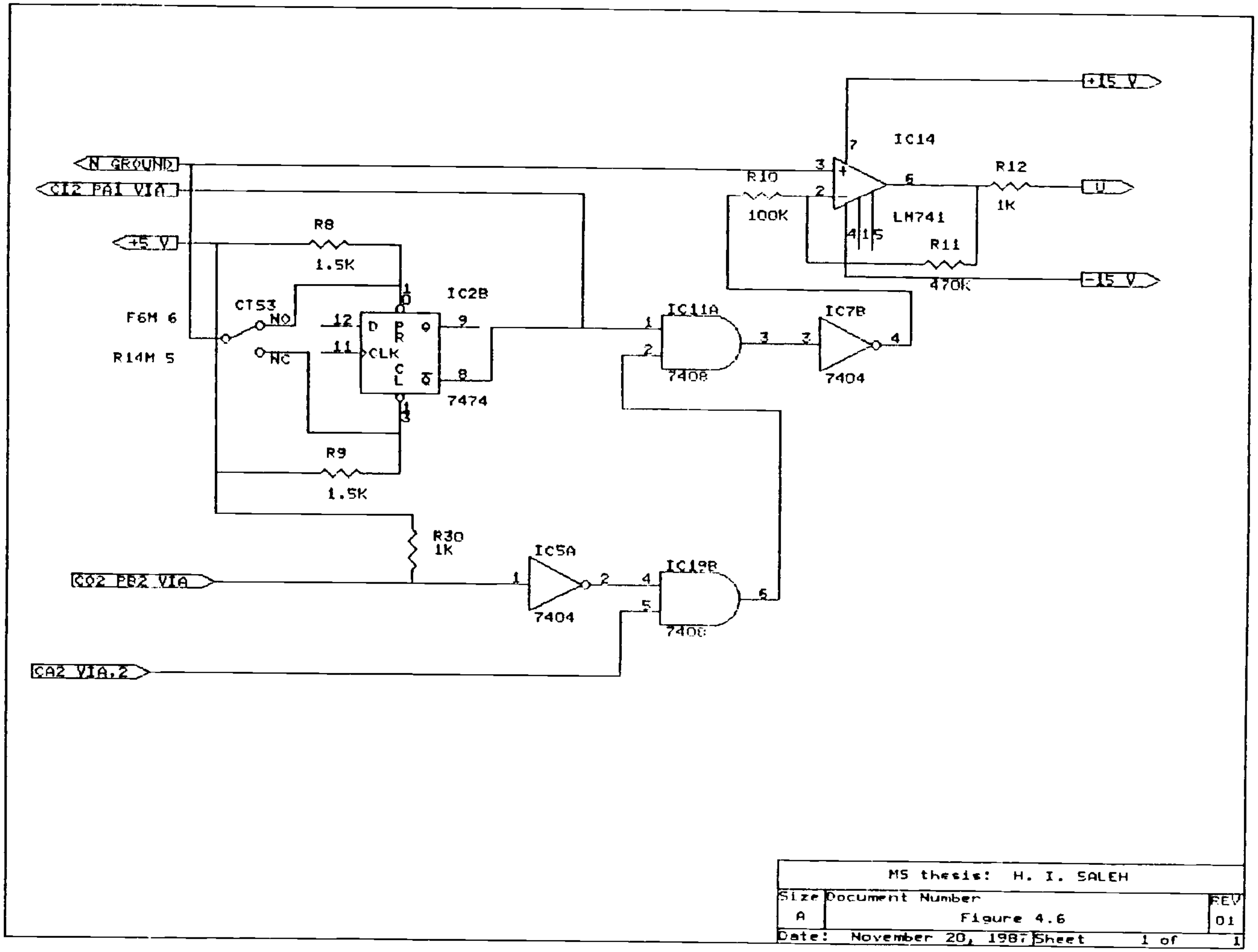


Figure 4.6 Up-motor driver

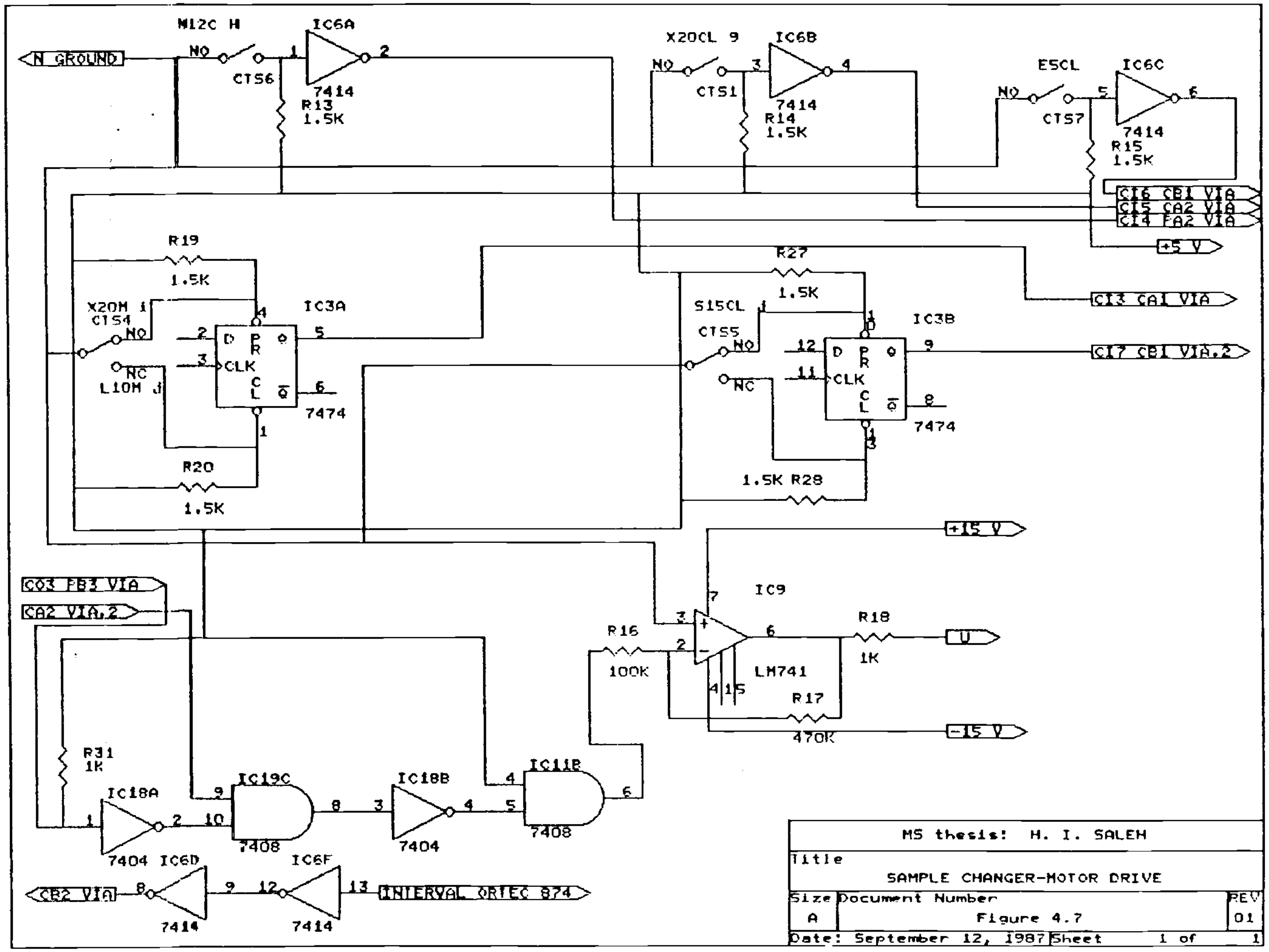


Figure 4.7 Sample changer-motor driver

is used for counting the number of vials indexed under the elevator. The CTS5/CI7 combination at 9F8X CB1 is used as an advanced warning signal to poll for the state of the next vial to be indexed at the elevator. CTS6/CI4 is used for reset. When vial position 1 is about to be indexed at the elevator, the CI4 input goes to a logic 1. This may be used in conjunction with CI3 to reset on vial position 1. The state detection software employs CTS7/CI6 as its hardware sensor. After the advance warning by CTS5/CI7, CI6 may be polled following a suitable delay to determine whether a standard vial or a nonstandard vial will be indexed at the elevator. Usually a nonstandard vial is signalled by CI6=0, in which case CI5 is utilized at index-time to determine whether the nonstandard vial is a group plug (CI5=1) or an empty vial position (CI5=0).

4.4 LIGHT EMITTING DIODES (LED)

The LEDs are interfaced on port B and also on PA5, PA6 and PA7 of port A (Figure 4.3). IC8 and IC12 (7407 open collector hex buffers) are utilized for driving the LEDs.

The LEDs are turned on by writing a logic 0 to the appropriate output lines on ports A and B. When the ACTIVE LED is turned on, MANUAL, GROUP and SINGLE are also enabled (but not turned on). These may then be turned on by writing a low to the corresponding output lines. The LEDs are powered by the +5v line via 270 ohm, 1 watt resistors R41, R42 and R43 and are expected to sink about 20mA of current when turned on. Three such resistors are used for the 9 LEDs; the resistors are soldered on the voltage regulator card. The pull-up resistors R33 through R36 are used to ensure that the LEDs are off when the computer power system is down. A power-on LED is also included; this is active whenever the power switch is toggled on.

4.5 HARDWARE ASSEMBLY AND TESTING

Many problems were encountered with the power supply primarily because a worst case analysis was not performed at the beginning of the design, and as a result the power requirement of the system was vastly underestimated. Poor and insufficient ground terminations also presented problems. The LEDs were overdriven and it took quite some time and component replacements before the puzzle of "the blinking LED" could be solved. Some of the computer port terminations on the three DB25 type sockets supplied with the AIM-65 were sheared and new terminations had to be made and each pin of all the six ports had to be checked individually with an oscilloscope (a tedious and lengthy procedure).

A 10 centimeter long solder type plug board was used for the voltage regulators and an 11 centimeter long wire-wrap board was utilized for the interface. A very high density package resulted with the wire-wrap board carrying one 40-pin DIP, six 24-pin DIPs, twelve 14-pin DIPs, one 16-pin DIP, six 8-pin DIPs and two 6-pin DIPs (appendix B).

CHAPTER V

SOFTWARE

This chapter presents the software capabilities and limitations of the AIM-65 microcomputer, its adoption for controlling the external hardware systems and the dedicated assembler and basic codes written for a complete data acquisition system.

5.1 OVERVIEW OF 6502 ASSEMBLER AND THE BASIC-E/65

A complete listing of the 6502 microprocessor assembler mnemonics and opcodes is given in reference (19). It contains 56 legal opcodes with 6 additional directives available through the DOS/65 system. The assembler is compatible with the MOS Technology standards with respect to the operands, opcodes, labels and comments, but it does not furnish the same set of assembler directives as defined in either the cross assembler or the microcomputer family KIM assembler manual.

The Basic-E/65 is the Naval Postgraduate School Basic language (Basic-E) which is modified and implemented for the 6502 operating system (DOS/65). It is made of two subsystems: the compiler (COMPILE.COM) and the run-time interpreter (RUN.COM). The Basic-E/65 may be used interactively or with a printing terminal. It implements most features of the proposed ANSI standard BASIC with extensive string manipulation and file input/output capabilities. The DOS/65 is used to handle all input/output and disk file management. The source file is an ASCII text which is created and edited with the DOS/65 editor EDIT.COM

5.2 ASSEMBLER ROUTINES

The assembler subroutines are written to execute at a beginning address of $D000 (hexadecimal) in memory. The assembler source file MISC.ASM is assembled by the DOS/65 assembler (ASM.COM) to a hexadecimal code file MISC.KIM. The MISC.KIM file is a normal ASCII text file that may be edited using the DOS/65 editor (EDIT.COM). MISC.KIM is loaded into memory at run time from BASIC. The assembler also creates a listing file MISC.PRN showing the source code and the object code. MISC.PRN is listed in appendix C.

The assembler subroutines implement those tasks that cannot be done (or only done poorly) in BASIC. Among these are interrupts and critical timing, software UART and fast data access, and the alarm and beeper routines.

The interrupt vectoring routine located at $D032 points to the interrupt service routine by saving its low address byte $43 in $A400 and the high address byte $D0 in $A401. The current vial position is also reset to 1 (CVIAL=1). When the system is operating and samples are indexed at the elevator, interrupts are generated which vector the computer to the service routine located at $D043. The current vial position is incremented, appropriate flags are set or cleared and the computer resumes normal operation where it stopped on the interrupt. BASIC-E/65 has no interrupt handling capability and is about 500 times slower than the assembler.

The beeper and alarm routines are located from $D069 to $D0EA. These routines give beeps of different tones and pitch and also raise an alarm when called from BASIC. They are used for signaling different error conditions during system operation. If the teletype is used as the sole input/ouput device the BELL command may be used instead.

Subroutines RSTATE (at $D0ED) and NSTATE (at $D0F4) are used for critical timing on reset by saving the state of the vial and returning the information in the accumulator to BASIC. RUART (at $D111) and TUART (at $D147) implement a software UART for communicating with the ORTEC 874 quad counter/timer. The correct delay times for the baud rate are determined in BASIC and poked into T1LS (low byte) and T1HS (high byte). The receiver subroutine RUART returns a character received from the ORTEC 874 in the accumulator to BASIC; while the transmit routine TUART sends to the ORTEC 874 a character from BASIC poked into the TRANS storage. The selected baud rate is used for determining the time spent in the delay subroutines DELAY (at $D194) and DEHALF (at $D1A8). All characters transmitted or received are formatted in standard 7 bit ASCII with no parity.

Finally the routine RCOUNT (at $D1BD) is used for fast data access of the ORTEC 874 counters. The number of counters to be accessed is poked from BASIC into RBUFF and the data read from the selected counters are stored as contiguous bytes in memory beginning at location RBUFF+2. Up to 32 bytes can be read and stored corresponding to the 4 counters in the 874, each counter providing 8 bytes of data (8 decades). The data stored are then read in BASIC and stored on disk for further analysis.

5.3 BASIC PROGRAMS AND SUBROUTINES

Two BASIC programs PROJ.BAS and PROJ2.BAS implement the I/O data acquisition protocol and analysis. The I/O protocols handled include setting up the ports for the RTC, ORTEC 874, and motor controls, while the analysis primarily consists of obtaining the count rates for each of the selected counters in the desired energy range.

The two programs are rather easy to use as they are menu driven. First, PROJ is run to initiate the data acquisition for either the single or group plug mode. This program also allows the interrogation of the RTC which is essential for reading/setting time and date. After an exit is made from PROJ, PROJ2 should be run immediately to analyze the raw data and obtain a hardcopy of the output. PROJ2 is actually an extension of PROJ (the latter uses the I/O format setup by the former). The two were separated due to memory limitations. Detailed instructions for running these programs are given in appendix A. The listings of PROJ.BAS and PROJ2.BAS are given in appendix C.

VI. BIBLIOGRAPHY

1. "Operation and installation instructions for Automatic Gamma Well counting systems". Publication no. 713600, Nuclear-Chicago Corporation.

2. CMOS DATABOOK. National Semiconductor Corporation, 1984, p. 3-11 - 3-25.

3. Bruce A. Artwick. Microcomputer Interfacing, Prentice-Hall, 1980.

4. The HP-IL Interface Kit Technical Guide. HP 82166C, Hewlett-Packard, May 1983.

5. The HP-IL Interface Specification. 82166-90017, Hewlett-Packard, November 1982.

6. HP-IB Interface, Owner's Manual. 82937-90017, Hewlett-Packard, January 1982.

7. Intel Components Data Catalog, January 1982.

8. MCS-48 Family of Single Chip Microcomputers, User's Manual. Intel Corporation, September 1981.

9. Robert E. Turner. "Low-cost generator delivers all standard bit rates". Electronic Magazine: Designer's Casebook Number 5, Mc Graw-Hill Publications Co. 1982, p. 166-167

10. AIM 65 Microcomputer User's Guide. Rockwell International, December 1979.

11. R6500 Microcomputer System Hardware Manual. Rockwell International, August 1978.

12. Model STD FDI-1 Floppy Disk Interface for the STD Bus, Reference Manual. Forethought Products, 1982.

13. Model STD-VID 64/80 Video Display Generator for the STD Bus. Forethought Products, 1982.

14. Memory-Mate Expansion Board for the AIM-65 Computer, Operating Manual. Forethought Products, 1981.

15. SCR Manual, Fifth Edition. General Electric, 1972.

16. Steven A. Ciarcia. "Every one can know the real time," Byte Magazine, May 1982, p. 34-58.

17. Model 874 Quad Counter/Timer, Operating and Service Manual. EG&G ORTEC, June 1984.

18. David G. Larsen and Peter R. Rony. "The Bugbook IIA, Interfacing & Scientific Data Communications Experiments using the Universal Asynchronous Receiver/Transmitter (UART) and 20mA Current Loops", E & L Instruments Inc., 1975.

19. AIM-MATE DOS Operating Manual. Forethought Products, 1982.

20. MCS-80 User's Manual. Intel Corporation, October 1977.

21. OPTOELECTRONICS DATA BOOK. Texas Instruments Incorporated, 1983-84.

22. The TTL DATA BOOK. Texas Instruments Incorporated, Volume 1, 1984.

23. The TTL DATA BOOK. Texas Instruments Incorporated, Volume 3, 1984.

24. The Line Driver and Line Receiver DATA BOOK for Design Engineers. Texas Instruments Incorporated, 1981.

25. Linear Circuits DATA BOOK. Texas Instruments Incorporated, 1984.

26. Howard V. Malmstadt, Christie G. Enke and Stanley R. Crouch. Electronics and Instrumentation for Scientists, The Benjamin/Cummings Publishing Company, Inc., 1981.

APPENDICES

APPENDIX A

USER'S MANUAL

This appendix gives a brief overview of the integration of the system hardware and software, and a demonstration of system operation.

A.1 USER ENVIRONMENT

The three programs required to run the system are MISC, PROJ and PROJ2 and these are placed on the same floppy disk and may be run from drive A via either the AIM keyboard and display or from the teletype for a hardcopy of the output. It is preferred to run the programs from the teletype. The only problem with this is that there is no totally clean way to switch from the AIM keyboard to the TTY. The following procedure may be used to achieve an AIM 65 to TTY keyboard transfer.

A.2 AIM 65 TO TTY KEYBOARD TRANSFER

A - Power-up the AIM, disk drives, video monitor and the

TTY

B - Position the TTY control switch to local

C - Position the KB/TTY switch to TTY

D - Depress the AIM 65 RESET button

E - Type RUBOUT on the TTY (do not hit RETURN)

F - Position the TTY control switch to LINE and hit the RETURN key

The AIM 65 will respond by entering the monitor and printing

```
ROCKWELL AIM 65
<
```

The next keyboard entry should be made from the TTY keyboard.

A.3 BOOTING THE SYSTEM

Insert the AIM-MATE DOS diskette into drive A with the label on the diskette facing to the right. Load the boot routine beginning at the location $9800 by typing

```
*=9800
```

This would appear as

```
<*>=9800
```

Then type in the go command:

```
G
```

The drive should be active for 10-20 seconds and the question

```
HOW MANY DRIVES?
```

appears; then type 2

The system then indicates the default drive as A:

```
A>
```

The system can now be used to run the programs. But as a precaution, if you have not already done so, make a backup copy of the AIM-MATE DOS diskette which also contains MISC, PROJ and PROJ2. First, format a 5 1/4 " disk by typing:

```
MINI-FMT <RET>
```

and respond to the prompts in the program. Next, insert the formatted disk into drive B, the AIM-MATE DOS disk into drive A and type:

```
COPY ALL
```

This should make a backup copy of the diskette.

A.4 RUNNING PROJ

Once the system has been booted as described in section A.3, the TTY line printer should be adjusted to the top of a new page, the NIM-BIN power supply, detector HV power supply, and interface power supply should all be turned on. All these units may also be powered at the same instance as the AIM computer. Then type:

```
RUN PROJ
```

The program then prints the program title, date and time and comes up with the question:

```
PERFORM RESET WITH BATTERY CHANGE
RESET(Y/N)?
```

Reset should only be performed when the battery for the RTC on the interface is changed or when a system malfunction has occurred. If Y is typed in, the system will reset the current vial position indexed at the elevator to 1. Whichever response was selected the system will finally display:

```
vial position  = xx
NSV            = a
vial state     = b
```

where xx = current vial position indexed at the
elevator

a=0, b=0 => a sample vial is indexed

a=1, b=1 => a Group plug is indexed

a=1, b=2 => an empty vial is indexed

(NSV=nonstandard vial)

The following prompts then appears:

```
S(tandby  A(ctive  Q(uit ?
```

Entering S will enable you to interrogate the RTC for setting/reading time and date. This is always necessary when there is a battery change or a system malfunction. Typing A will enter you into the main program and the system will respond by displaying:

```
G(roup  S(ingle  M(anual  Q(uit ?
```

A.4.1 GROUP

This mode allows the handling of groups of vials to be run with the same parameters such as preset time/preset count, time base and counters to be used for the group. Up to 3 groups may be specified.

To use this mode first arrange the samples on the belt with a group plug to depict the beginning of a group and an empty vial or another group plug to signify its end. Up to 98 samples may be handled per group.

Type G to enter the group operating mode; the system will display:

```
enter number of groups < = 3  or Q(uit.
```

Type in the desired number of groups.

The program then initiates a loop to accept the group parameters by displaying:

```
GROUP # PARAMETERS
```

# is the current group whose parameters are to be input.

The following series of options are then displayed:

```
PC(preset.count  PT(preset.time  Q(uit.
```

This gives you the option of timing for a preset count (PC) or counting for a preset time (PT). Just enter the desired option.

time base S(sec M(min E(ext Q(uit.
Typing S will select a 0.1 sec time base in which case the Ortec 874 timer will count pulses derived from the internal, crystal-controlled, precision time base at 0.1 second intervals. When M is selected, the timer counts pulses at 1 minute intervals. External (E) should be selected for preset count (PC) measurements.

SELECT COUNTERS 1, 2, 3, 4, select=1 ?
For preset time measurements counter 1 is used as a timer for the other 3 counters and should not be selected. For this mode counters may be selected by typing:
0, 1, 1, 1
This selects counters 2, 3, and 4. Make sure that the selected counters are properly hooked up to the SCA, amplifier and pre-amp outputs. Note that a 1 in the counter position selects that counter; anything else disselects it. For preset count all 4 counters may be selected.

SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED
COUNTER DISPLAY (1-4) ?
This option allows the display of a selected counter on the ORTEC 874 during counting. The next 2 options are

only applicable for PC measurements.

```
MAX PRESETS:  COUNTER 1=9E+07; 2, 3, 4=1E+38
ENTER PRESETS FOR SELECTED COUNTERS
```

This options allow input of the desired preset counts for the selected counters with the preset for each counter for a separate line.

```
M ( 1 to 9 ) , N ( 0 to 6 ) or Q(uit ?
```

M and N determine the preset time period as M x 10**N in seconds or minutes depending on time base selection. The desired input is M,N.

After all the group parameters have been selected, the system initiates the counting process and indexes the vials for the selected number of groups into the detector one after the other. The data collected from the counters is stored on the disk file SCRATCH.BAS. After the data acquisition process is complete the system will respond:

```
G(roup  S(ingle  M(anual  Q(uit ?
```

The usual response here is to type Q to exit. G or S may initiate another counting sequence and destroy all the data saved in SCRATCH.BAS. The system then displays:

```
S(tandby  A(ctive  Q(uit ?
```

Type Q to exit. The system then exits PROJ by displaying:

```
BYEBYE.
```

A.4.2 SINGLE

If the initial response had been S (single operating mode) instead of G, the system will still invoke the same subroutine and request the same parameters as in Group, but instead of looking for a group plug, the vial position of the sample to be run will be requested:

```
sample number (1-100) or Q(uit ?
```

Normal data acquisition procedures will then be implemented for this sample and at completion the following prompt will be displayed:

```
repeat for another sample  Y(es  N(o  Q(uit ?
```

If the response here is Y, we will be asked whether we want to use the same parameters or enter a different set. If however the answer is N, the following will be displayed:

```
G(roup  S(ingle  M(anual  Q(uit ?
```

enter Q

```
S(tandby  A(ctive  Q(uit ?
enter Q
BYEBYE
```

A.4.3 MANUAL

If M is the initial response instead of either G or S, the manual mode would have been enabled. In this mode it is possible to index the next vial under the elevator and move the vial into and out of the detector but the control of the ORTEC 874 must be set to manual so that the front panel push buttons can be used to select the desired functions.

A.5 RUNNING PROJ2

At this stage all data collected from the counting system has been stored in SCRATCH.BAS, so all that is left is to run PROJ2 to perform the analysis and obtain a hardcopy of the results.

First position the line printer at the top of a new page and type:

```
RUN PROJ2
```

The following prompt will be displayed:

```
INPUT PROJECT NAME < 70 CHAR
?
```

Enter the name you wish to assign to this project and hit <RET>. Finally the output will be printed. Two sample runs for the group and single modes are given in the next section.

A.6 SAMPLE RUN

```
   RUN PROJ
 BASIC-E/65 INTERPRETER - VERSION 1.0-S

 PROJ.BAS    FRIDAY,OCTOBER   2 ,1987      8 :1 :13



 PERFORM RESET WITH BATTERY CHANGE
 RESET (Y/N)? N
 vial position = 55
 NSV=1
 VIAL STATE =2
 S(tandby   A(ctive   Q(uit? A
  G(roup  S(ingle   M(anual   Q(uit? S
 PC(preset.count  PT(preset.time  Q(uit? PT
  time base S(sec M(min E(ext Q(uit? S
 SELECT COUNTERS 1,2,3,4 ,SELECT=1? 0,1,0,0
 SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED
 COUNTER DISPLAY (1-4)? 2
  M ( 1 to 9 ) ,N ( 0 TO 6 ) or  Q(uit? 6,1
 sample number(1-100) or Q(uit? 59
 repeat for another sample Y(es N(o Q(uit? Y
 with same parameters Y(es N(o? Y
 sample number(1-100) or Q(uit? 60
 repeat for another sample Y(es N(o Q(uit? Y
 with same parameters Y(es N(o? Y
 sample number(1-100) or Q(uit? 61
 ERROR*** indexed vial position 61 is empty
 sample number(1-100) or Q(uit? Q
  G(roup  S(ingle   M(anual   Q(uit? Q
 S(tandby   A(ctive   Q(uit? Q
  BYEBYE


A>
```

```
  RUN PROJ2
BASIC-E/65 INTERPRETER - VERSION 1.0-S


INPUT PROJECT NAME < 70 CHAR
? SINGLE SAMPLE RUN




SINGLE SAMPLE RUN   PAGE 1



                                                          CHANNEL COUNTS
SG-POS#     START TIME   DELTA T,SEC        CH1         CH2          CH3          CH4
----------------------------------------------------------------------------------------
0   - 59    8 :2 :30     60              0          87          0            0
0   - 60    8 :4 :24     60              0          73          0            0

                                                       COUNTS PER SECOND
SG-POS#                                     CH1         CH2          CH3          CH4
----------------------------------------------------------------------------------------
0   - 59                                 0          1.45        0            0
0   - 60                                 0          1.21667     0            0

8 :5 :24




BYEBYE

A>
```

```
RUBOUT
  ROCKWELL AIM 65


<*>=9800

<G>/


FORETHOUGHT 33K DOS/65 V1.2
AIM-MATE VERS. REV 1.0
HOW MANY DRIVES?2
A>

  RUN PROJ
BASIC-E/65 INTERPRETER - VERSION 1.0-S


PROJ.BAS   THURSDAY,OCTOBER   1 ,1988      12 :24 :7



PERFORM RESET WITH BATTERY CHANGE
RESET (Y/N)? N
vial position = 37
NSV=1
VIAL STATE =2
S(tandby   A(ctive   Q(uit? S
ST(set.time SD(set.date RT(read.time RD(read.date  Q(uit? SD
S(et date   Q(uit? S
 enter YEAR? 1987
 enter MONTH (1-12)? 10
 enter day of the month (1-31)? 1
 enter day of the week (1-7)? 5
THE DATE IS THURSDAY,OCTOBER 1 ,1987
ST(set.time SD(set.date RT(read.time RD(read.date  Q(uit? Q
S(tandby   A(ctive   Q(uit? A
 G(roup  S(ingle   M(anual   Q(uit? G
enter number of groups <= 3 or Q(uit? 2
GROUP 1  PARAMETERS
PC(preset.count  PT(preset.time  Q(uit? PT
 time base S(sec M(min E(ext Q(uit? S
SELECT COUNTERS 1,2,3,4 ,SELECT=1? 0,1,0,0
SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED
COUNTER DISPLAY (1-4)? 2
 M ( 1 to 9 ) ,N ( 0 TO 6 ) or  Q(uit? 6,1
GROUP 2  PARAMETERS
PC(preset.count  PT(preset.time  Q(uit? PT
 time base S(sec M(min E(ext Q(uit? S
SELECT COUNTERS 1,2,3,4 ,SELECT=1? 0,1,0,0
SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED
COUNTER DISPLAY (1-4)? 2
 M ( 1 to 9 ) ,N ( 0 TO 6 ) or  Q(uit? 6,1
 G(roup  S(ingle   M(anual   Q(uit? Q
S(tandby   A(ctive   Q(uit? Q
 BYEBYE
```

```
  RUN PROJ2
BASIC-E/65 INTERPRETER - VERSION 1.0-S


INPUT PROJECT NAME < 70 CHAR
? GROUP SAMPLE RUN




GROUP SAMPLE RUN    PAGE 1




                                                     CHANNEL COUNTS
GP-POS#    START TIME  DELTA T,SEC      CH1              CH2         CH3         CH4
-----------------------------------------------------------------------------------------
1  - 42    12 :27 :11   60         0          93         0          0
1  - 43    12 :28 :48   60         0          89         0          0
1  - 44    12 :30 :21   60         0          78         0          0
2  - 46    12 :32 :0    60         0          85         0          0
2  - 47    12 :33 :33   60         0          104        0          0
2  - 48    12 :35 :8    60         0          75         0          0
2  - 49    12 :36 :42   60         0          80         0          0


                                                   COUNTS PER SECOND
GP-POS#                                      CH1              CH2         CH3         CH4
-----------------------------------------------------------------------------------------
1  - 42                                  0          1.55       0          0
1  - 43                                  0          1.48333    0          0
1  - 44                                  0          1.3        0          0
2  - 46                                  0          1.41667    0          0
2  - 47                                  0          1.73333    0          0
2  - 48                                  0          1.25       0          0
2  - 49                                  0          1.33333    0          0

12 :37 :42





BYEBYE

A>
```

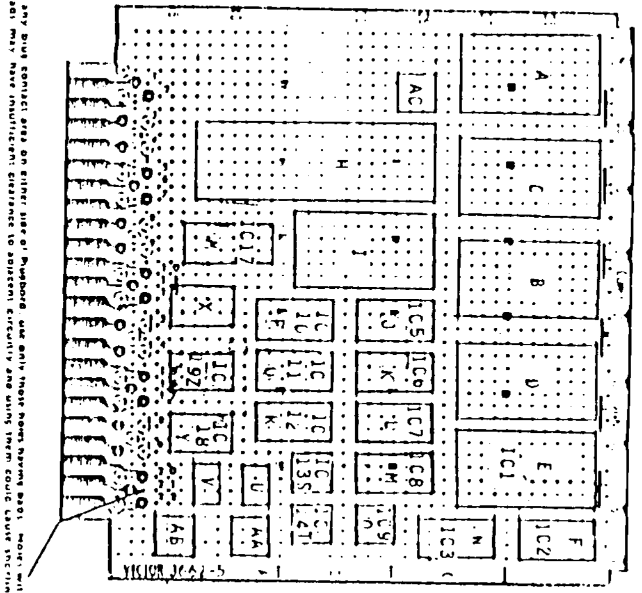
APPENDIX E. COMPONENTS LAYOUT AND CABLES

VECTOR ELECTRONIC CO., INC.
12460 GLADSTONE AVE.
SYLMAR, CALIF. 91342

VECTORBORD PLUGBORD
PATTERN .042" × 0.1" SPACED HOLES
LA9P2 LAYOUT PAPER Issue 75-9

3602 5 4.5" LONG CARD

COMPONENT SIDE



5 In any blue contact area on either side of Plugbord, use only those holes having pads. Holes without pads may have insufficient clearance to adjacent circuitry and using them could cause shorting.

4 Before pressing terminals into board, position (rotate) terminals to maximize the clearance between the widest part of the terminal and the nearest adjacent conductor.

2 Intended for use in non-hostile environments up to 200 volts RMS or 300 volts DC

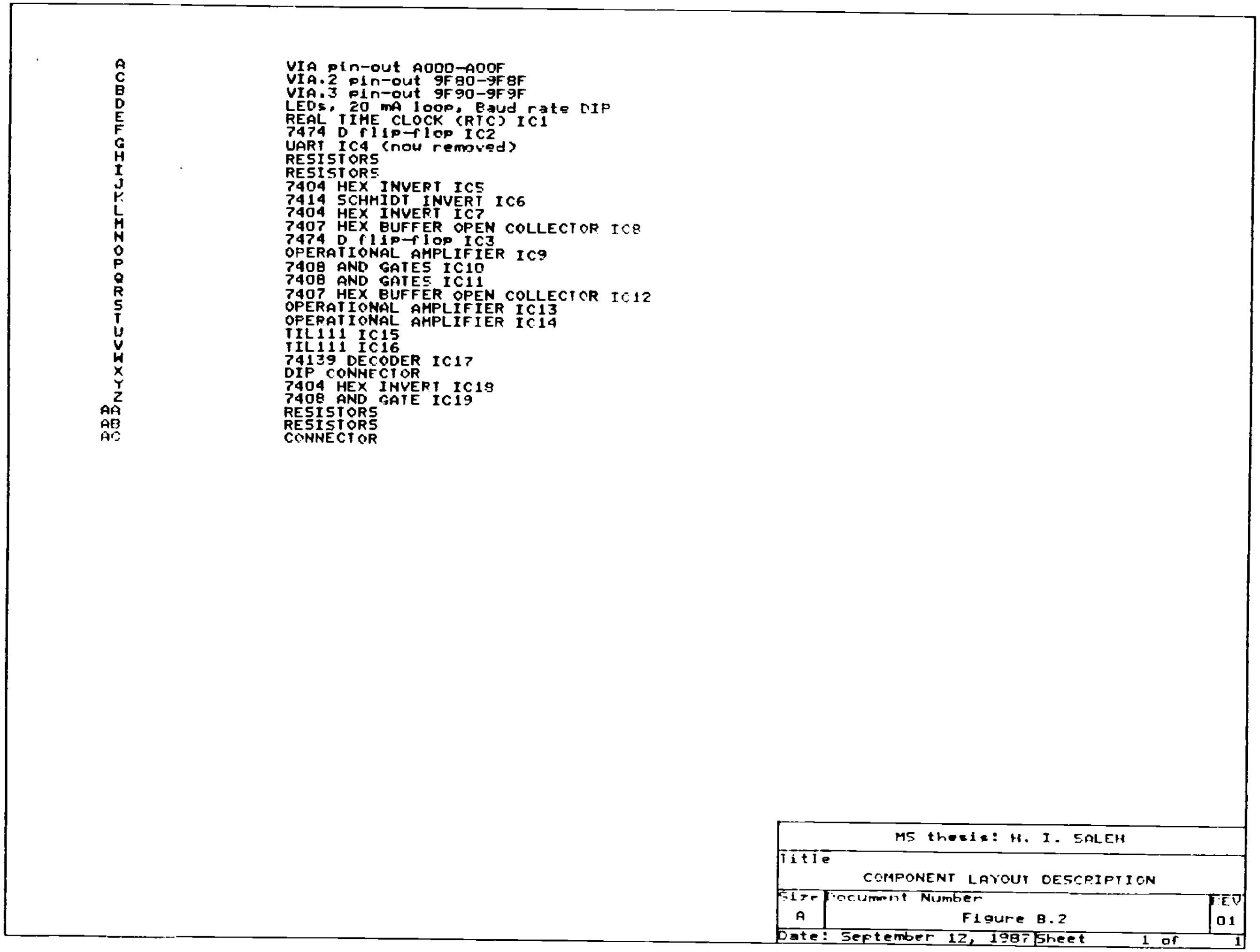
1 Broken circles above edge contacts indicate location of actual connector contact pads on opposite side of board.

NOTES

X, Y, Z = Horizontal Zone Position DIPs
A, B, C = Vertical Zone Position DIPs
= No. 1 DIP Position

Zone Letters A,B,C,etc., on Y axis and X,Y,Z on X axis mark position for 14- or 16-pin DIPs

Figure B.1 Component layout on Interface card



A	VIA pin-out A000-A00F
C	VIA.2 pin-out 9F80-9F8F
B	VIA.3 pin-out 9F90-9F9F
D	LEDs, 20 mA loop, Baud rate DIP
E	REAL TIME CLOCK (RTC) IC1
F	7474 D flip-flop IC2
G	UART IC4 (now removed)
H	RESISTORS
I	RESISTORS
J	7404 HEX INVERT IC5
K	7414 SCHMIDT INVERT IC6
L	7404 HEX INVERT IC7
M	7407 HEX BUFFER OPEN COLLECTOR IC8
N	7474 D flip-flop IC3
O	OPERATIONAL AMPLIFIER IC9
P	7408 AND GATES IC10
Q	7408 AND GATES IC11
R	7407 HEX BUFFER OPEN COLLECTOR IC12
S	OPERATIONAL AMPLIFIER IC13
T	OPERATIONAL AMPLIFIER IC14
U	TIL111 IC15
V	TIL111 IC16
W	74139 DECODER IC17
X	DIP CONNECTOR
Y	7404 HEX INVERT IC18
Z	7408 AND GATE IC19
AA	RESISTORS
AB	RESISTORS
AC	CONNECTOR

MS thesis: H. I. SALEH		
Title COMPONENT LAYOUT DESCRIPTION		
Size A	Document Number Figure B.2	REV 01
Date: September 12, 1987	Sheet 1 of 1	

Figure B.2 Component layout description

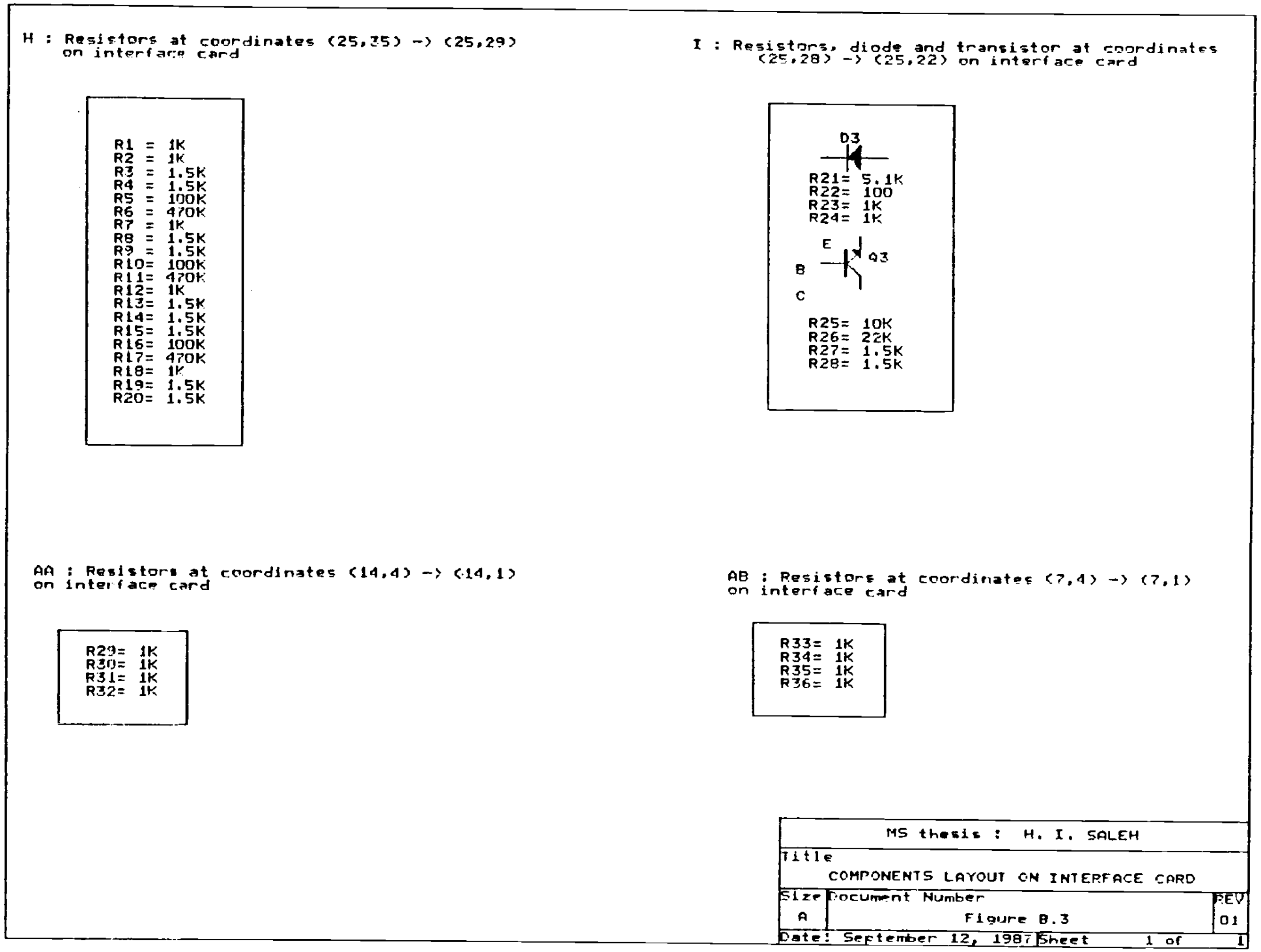
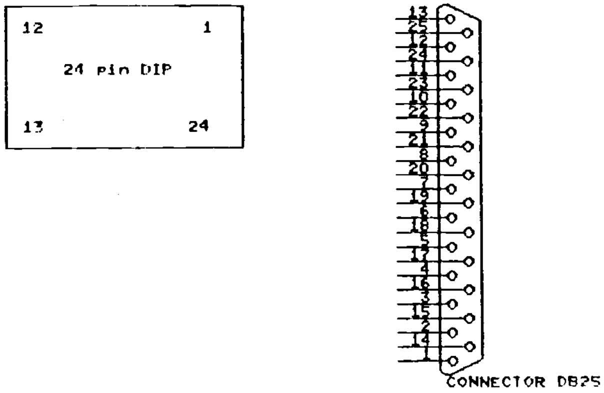


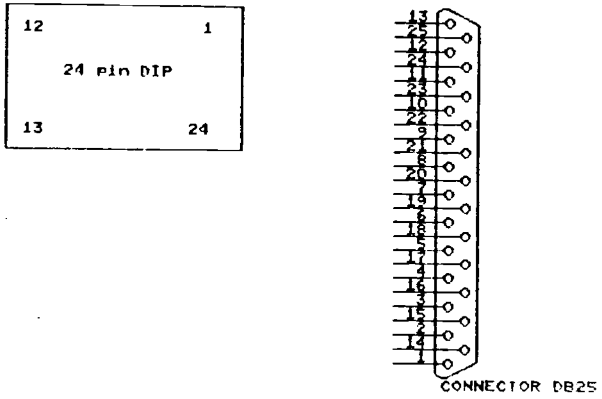
Figure B.3 Detailed layout of resistors, transistor and diode



DIP pin-out	DB25P	SIGNAL
1	24	CA1
2	22	CB1
3	20	PA0
4	18	PB1
5	16	PA5
6	14	PA3
7	12	unused
8	10	CB2
9	8	PB7
10	5	PB0
11	3	PA4
12	6	PB2
13	2	PA2
14	4	PA6
15	7	PB4
16	9	PB6
17	11	CA2
18	13	unused
19	15	PA1
20	17	PA7
21	19	PB3
22	21	PB5
23	23	unused
24	25	unused

MS thesis : H. I. SALEH
Title
CABLE A
Size A
Document Number
Figure B.4
REV 01
Date: September 12, 1987 Sheet 1 of 1

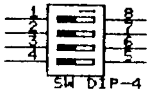
Figure B.4 Connections of CABLE A to VIA (A000-A00F)



DIP pin-out	DB25P	SIGNAL
1	24	PB3
2	22	PB7
3	20	CB2
4	18	CA2
5	16	PA5
6	14	PA1
7	12	PB2
8	10	PB6
9	8	GND
10	5	GND
11	3	PA4
12	1	PA0
13	2	PA2
14	4	PA6
15	7	GND/+5V
16	9	GND
17	11	PB4
18	13	PB0
19	15	PA3
20	17	PA7
21	19	CA1
22	21	CB1
23	23	PB5
24	25	PB1

MS thesis : H. I. SALEH
Title
CABLES B AND C
Size A
Document Number Figure B.5
REV 01
Date: September 12, 1987
Sheet 1 of 1

Figure B.5 Connections of CABLES B and C to VIA.2 and VIA.3 repectively



DIP PIN-OUT	CONNECTED TO	FUNCTION
1	SW DIP-4 7	BAUD RATE SELECTION
2	SW DIP-4 5	
3	SW DIP-4 3	
4	SW DIP-4 1	
5	AMPHENOL L	SERIAL INPUT DATA
6	AMPHENOL B	SERIAL OUTPUT DATA
7		unused
8		unused
9	CATHODE D28	COUNT
10	CATHODE D21	STANDBY
11	CATHODE D22	ACTIVE
12	CATHODE D23	MANUAL
13	CATHODE D20	POWER-ON
14	ALL ANODES	+5V
15	CATHODE D25	SINGLE
16	CATHODE D24	GROUP
17	CATHODE D26	BUSY
18	CATHODE D27	IDLE
19	AMPHENOL D	GROUND (SIGNAL)
20	AMPHENOL M	GROUND (SIGNAL)
21	SW DIP-4 2	BAUD RATE SELECTION
22	SW DIP-4 4	
23	SW DIP-4 6	
24	SW DIP-4 8	

MS thesis: H. I. SALEH
Title
CABLE D
Size A
Document Number Figure B.6
REV 01
Date: September 12, 1987
Sheet 1 of 1

Figure B.6 Connections of CABLE D to the LEDs baud rate selector

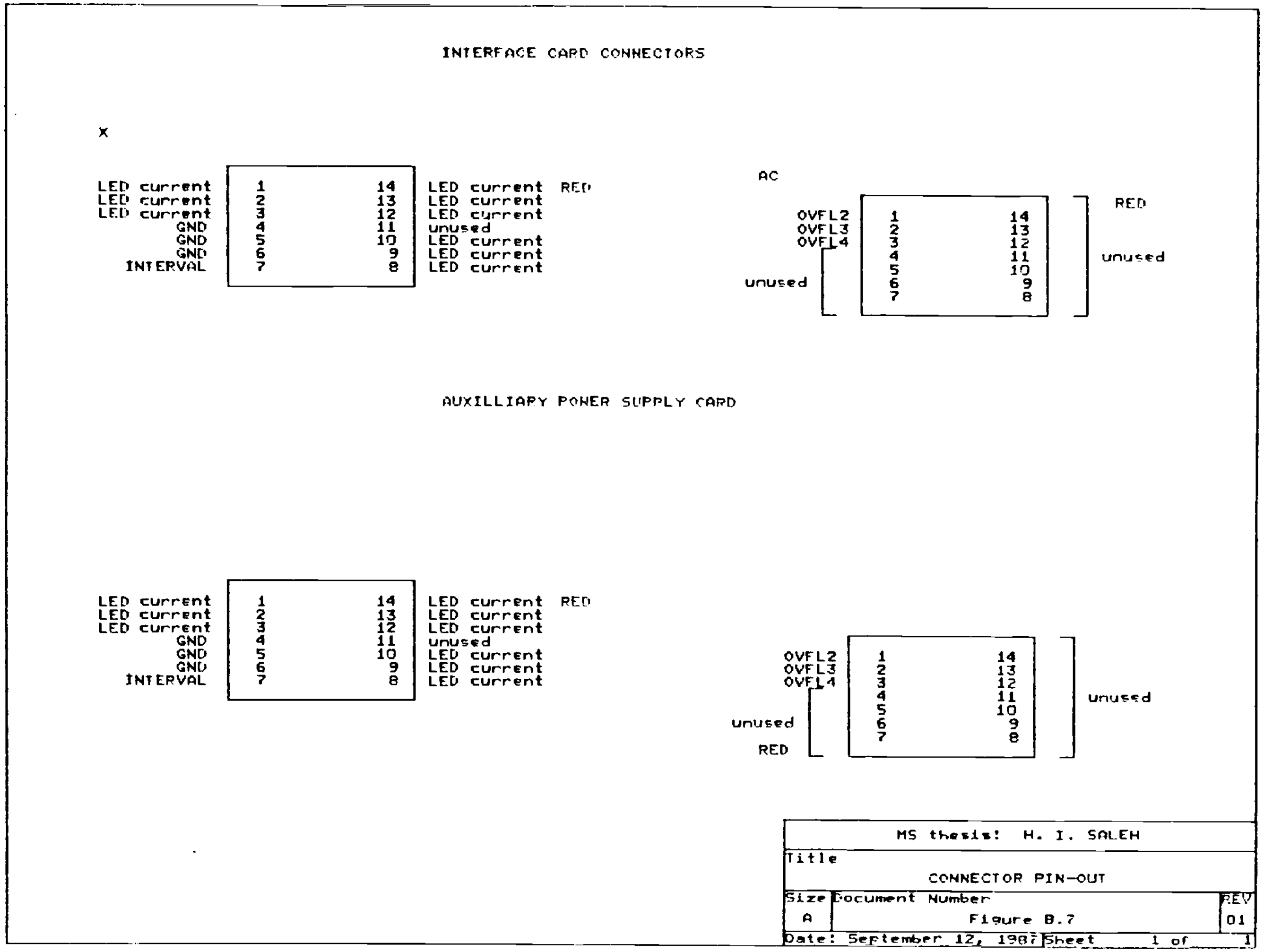


Figure B.7 Signals on interface card and auxilliary power supply card connectors

PIN	NAME	I/O
1	CRYSTAL	INPUT
2	CRYSTAL	INPUT
3	PWR	INPUT
4	X20CL 9 NO	INPUT
5	E5CL NO	INPUT
6	X20T	OUTPUT TO TRIAC
7	U17M 8 NC	INPUT
8	B2M 7 NO	INPUT
9	R14T	OUTPUT
10	R14M 5 NC	INPUT
11	F6M 6 NO	INPUT
12	U17T	OUTPUT
13	N12CL h NC	INPUT
14	L10M J NC	INPUT
15	X20M i NO	INPUT
16	INTERVAL	INPUT
17	CTS5 pin 3 C NC	INPUT
18	+15V	INPUT
19	GROUND	I/O
20	+5V	INPUT
21	-12V	INPUT
22	-15V	INPUT

MS thesis: H. I. SALEH
Title
WIRE WRAP PLUGBOARD EDGE CONNECTOR
Size A
Document Number Figure B.8
REV 01
Date: September 12, 1987
Sheet 1 of 1

Figure B.8 Wire wrap plugboard edge connector

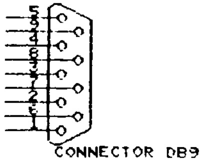
PIN	NAME	I/O
1	A1T (-15V)	INPUT
2	N12T GROUND	INPUT
3		
4	F6T (+15V)	INPUT
5	CRYSTAL	OUTPUT
6	CRYSTAL	OUTPUT
7	BATTERY +	INPUT
8	BATTERY -	INPUT
9	PWR	OUTPUT
10		
11	S15CL 9 NO	INPUT
12	OVFL2	OUTPUT
13	OVFL3	OUTPUT
14	OVFL4	OUTPUT
15		
16		
17	+12V	OUTPUT
18	+15V	OUTPUT
19	GROUND	OUTPUT
20	+5V	OUTPUT
21	-12V	OUTPUT
22	-15V	OUTPUT

MS thesis: H. I. SALEH
Title
SOLDER PLUGBOARD EDGE CONNECTOR
Size A
Document Number Figure B.9
REV 01
Date: September 12, 1987
Sheet 1 of 1

Figure B.9 Solder plugboard edge connector

AMPHENOL CONNECTOR

PIN	WIRE COLOR
A	BLACK + WHITE
B	BROWN
D	RED
E	LIGHT ORANGE
L	GREEN
H	BLUE
N	PINK (VIOLET)
P	GRAY
K	YELLOW



20 mA LOOP CONNECTION

AMPHENOL PIN	DB9 PIN	DESCRIPTION
B	1	SERIAL DATA OUTPUT
D	2	GROUND (SIGNAL)
L	3	SERIAL INPUT DATA
H	4	GROUND (SIGNAL)

MS thesis: H. I. SALEH		
Title AMPHENOL CONNECTOR FOR 20 mA LOOP		
Size A	Document Number Figure B.10	REV 01
Date: September 12, 1987	Sheet 1 of 1	

Figure B.10 Amphenol connector for 20 mA loop

APPENDIX C. PROGRAM LISTINGS

```
 TYPE MISC.PRN
0000              ;        THIS PROGRAM IS CALLED MISC.ASM
0000              ;        VERSION I.1    , 8/16/86
0000              ;
0000              ;        DOS ASSEMBLY PROGRAMS FOR INTERRUPT SERVICE
0000              ;        AND BEEPER ROUTINES
0000              ;
0000              ;        STARTING ADDRESS FOR ASSEMBLY LANGUAGE PROGRAMS
0000              ;        IS AT D000
0000              ;
0000              ;
0000              ;        INITIALIZATION OF MAINTAINANCE REGISTERS
0000              ;
0000              ;
0000                       START   =$D000
0000              ;
0000              ;
0000              ;
0000                       CVIALP  =START           ; STORAGE FOR CURRENT VIAL
0000                                                ; POSITION
0000                       BEEP    =START+1         ; BEEPER STORAGE
0000                       BEEP1   =START+2         ; BEEPER STORAGE
0000                       BEEP2   =START+3         ; BEEPER STORAGE
0000                       FREQQ   =START+4         ; BEEPER FREQUENCY REGISTER
0000                       CA1     =START+5         ; CA1 FLAG STORAGE
0000                       T1LS    =START+6         ; TIMER 1 LOW LATCH STORAGE
0000                       T1HS    =START+7         ; TIMER 1 HIGH LATCH STORAGE
0000                       TRANS   =START+8         ; TRANSMITTER STORAGE
0000                       RESS    =START+9         ; RECEIVER STORAGE
0000                       TBUFF   =START+10        ; TRANSMIT BUFFER
0000                       RBUFF   =START+11        ; COUNTER STORAGE
0000              ;
0000              ;
0000              ;
0000                       *       =START+50
D032              ;
D032              ;
D032              ;
D032 A0 43        SIRQ     LDY     #$43
D034 8C 00 A4              STY     $A400
D037 A9 D0                 LDA     #$D0             ; POINT TO INTERRUPT
D039 8D 01 A4              STA     $A401            ; SERVICE ROUTINE
D03C A9 01                 LDA     #01
D03E 8D 00 D0              STA     CVIALP           ; INITIALIZE VIAL POSITION
D041 58                    CLI                      ; ENABLE INTERRUPTS
D042 60                    RTS                      ; RETURN TO BASIC
```

Figure C.1 MISC.PRN listing

```
D043            ;
D043            ;
D043            ;
D043                                         ; INTERRUPT SERVICE ROUTINE
D043                                         ; BEGINS HERE
D043            ;       THIS ROUTINE IS USED TO UPDATE THE CURRENT VIAL
D043            ;       POSITION AS THE SAMPLE CHANGER MOTOR MOVES AND
D043            ;       INDEXES A VIAL AT THE ELEVATOR.
D043            ;
D043            ;       PC AND W ARE AUTOMATICALLY SAVED ON STACK AND
D043            ;       INTERRUPTS DISABLED BY AN IRQ VECTORING .
D043            ;       PC AND W WILL BE RESTORED AND INTERRUPTS
D043            ;       RE-ENABLED WITH AN RTI.
D043            ;
D043            ;       BUT THE REGISTERS NEED TO BE SAVED .
D043            ;
D043            ;
D043            ;
D043 48         SERV    PHA                  ; SAVE ACCUMULATOR ON STACK
D044 8A                 TXA
D045 48                 PHA                  ; SAVE X INDEX REGISTER
D046 98                 TYA
D047 48                 PHA                  ; SAVE Y INDEX REGISTER
D048 A9 01              LDA     #01          ;
D04A 8D 05 D0           STA     CA1          ; SAVE CA1 FLAG
D04D A9 02              LDA     #02          ;
D04F 8D 0D A0           STA     $A00D        ; CLEAR CA1 INTERRUPT
D052                                         ; FLAG IN IFR
D052 AE 00 D0           LDX     CVIALP       ; FETCH VIAL POSITION
D055 E8                 INX                  ; UPDATE VIAL POSITION
D056 8E 00 D0           STX     CVIALP       ; SAVE VIAL POSITION
D059 8A                 TXA                  ; TRANSFER POSITION TO ACCUMU.
D05A C9 65              CMP     #101         ; 101TH POSITION REACHED ?
D05C D0 05              BNE     quit         ; IF NO QUIT
D05E A9 01              LDA     #01          ;
D060 8D 00 D0           STA     CVIALP       ; IF YES RESET VIAL POSITION
D063 68         quit    PLA                  ; RESTORE Y INDEX REGISTER
D064 A8                 TAY
D065 68                 PLA                  ; RESTORE X INDEX REGISTER
D066 AA                 TAX
D067 68                 PLA                  ; RESTORE ACCUMULATOR
D068 40                 RTI                  ; RESTORE PC , PSW AND ENABLE
D069                                         ; INTERRUPTS , ,RETURN TO BASIC
D069            ;
D069            ;
D069            ;
```

Figure C.1 MISC.PRN (continued.)

```
D069                    SPKR     =$9FBD
D069                    KEYCK    =$ECEF
D069              ;
D069              ;
D069              ;     this sub is for end of line beep  or continual
D069              ;     beep until operator presses keyboard (beep2)
D069              ;
D069              ;
D069 20 8E D0     beepa JSR      beepb                   ;do a beep
D06C 4C 8D D0           JMP      don                     ;leave
D06F              ;
D06F              ;     this sub beeps until someone presses a key
D06F              ;
D06F 20 8E D0     beepc JSR      beepb                   ;do a beep
D072 A9 03              LDA      #$03                    ;length of beeps
D074 8D 02 D0           STA      BEEP1
D077 8D 03 D0           STA      BEEP2
D07A 20 EF EC     wait  JSR      KEYCK                   ;key pressed ?
D07D 98                 TYA
D07E D0 0D              BNE      don
D080 CE 02 D0           DEC      BEEP1
D083 D0 F5              BNE      wait
D085 CE 03 D0           DEC      BEEP2
D088 D0 F0              BNE      wait
D08A 4C 6F D0           JMP      beepc                   ;do another beep
D08D 60           don   RTS                              ;return
D08E              ;
D08E              ;
D08E              ;     the actual beeper routine is called for single
D08E              ;     tone.  pitch can be changed is desired
D08E              ;
D08E              ;
D08E A9 FF        beepb LDA      #$FF                    ;tone length
D090 8D 02 D0           STA      BEEP1
D093 A2 2C        A8    LDX      #$2C                    ;pitch value
D095 CA           A6    DEX
D096 D0 FD              BNE      A6
D098 8D BD 9F           STA      SPKR                    ;toggle speaker
D09B CE 02 D0           DEC      BEEP1
D09E D0 F3              BNE      A8
D0A0 60                 RTS                              ;return
D0A1              ;
D0A1              ;
D0A1              ;     for interesting effects try beepd
D0A1              ;     changes frequency with each cycle
D0A1              ;     press reset button to stop
```

Figure C.1 MISC.PRN (continued.)

```
D0A1            ;
D0A1            ;
D0A1 A2 00      beepd   LDX     #$00
D0A3 CA         A9      DEX
D0A4 D0 FD              BNE     A9
D0A6 8D BD 9F           STA     SPKR                    ;toggle speaker
D0A9 CE A2 D0           DEC     beepd+1                 ;lower pitch
D0AC 4C A1 D0           JMP     beepd
D0AF            ;
D0AF            ;       try and inc instruction (ee) in place of dec
D0AF            ;       for different tones
D0AF            ;
D0AF            ;
D0AF            ;       this is the alarm routine used to set someones
D0AF            ;       attention.  it continues to make noise unitl
D0AF            ;       the keyboard is pressed any key
D0AF            ;
D0AF            ;
D0AF            ;
D0AF            ;
D0AF 20 B3 D0   alarm   JSR     alarma                  ;do an alarm
D0B2 60                 RTS                             ;return
D0B3 A9 03      alarma  LDA     #$03                    ;tone length
D0B5 8D 02 D0           STA     BEEP1
D0B8 8D 03 D0           STA     BEEP2
D0BB 20 EF EC           JSR     KEYCK                   ;key pressed ?
D0BE 98                 TYA
D0BF F0 01              BEQ     A10             ; POINT TO INTERRUPT
D0C1 60                 RTS                             ;return
D0C2 CE 02 D0   A10     DEC     BEEP1
D0C5 D0 1A              BNE     tone
D0C7 CE 03 D0           DEC     BEEP2
D0CA D0 15              BNE     tone
D0CC A9 24              LDA     #$24                    ;hi pitch
D0CE CD 04 D0           CMP     freqq
D0D1 F0 06              BEQ     A11
D0D3 8D 04 D0           STA     freqq
D0D6 4C B3 D0           JMP     alarma
D0D9 A9 2B      A11     LDA     #$2B                    ;low pitch
D0DB 8D 04 D0           STA     freqq
D0DE 4C B3 D0           JMP     alarma                  ;do it
D0E1 AE 04 D0   tone    LDX     freqq                   ;set tone freq
D0E4 CA         A12     DEX
D0E5 D0 FD              BNE     A12
D0E7 8D BD 9F           STA     SPKR
D0EA 4C C2 D0           JMP     A10
D0ED            ;
```

Figure C.1 MISC.PRN (continued.)

```
D0ED                ;
D0ED                ;
D0ED                ;       THIS SUBROUTINE SAVES THE RESET VIAL STATE
D0ED                ;       IMMEDIATELY BEFORE RESET AND RETURNS THE INFORMATION
D0ED                ;       BACK TO BASIC.THE LOW BYTE OF THE STATE IS RETURNED
D0ED                ;       IN THE ACCUMULATOR AND THE HIGH BYTE IN THE Y INDEX
D0ED                ;       REGISTER.NOTE THAT THE HIGH BYTE IS ALWAYS ZERO.
D0ED                ;
D0ED                ;
D0ED                        IFR2    =$9F8D
D0ED                        IFR     =$A00D
D0ED                        PORTA   =$A001
D0ED                ;
D0ED A9 04          RSTATE  LDA     #04                ;  FETCH DATA
D0EF 2C 01 A0               BIT     PORTA              ;  MASK FETCH BIT 2
D0F2 F0 F9                  BEQ     RSTATE             ;  WAIT HERE UNTIL CI4.PA2=1
D0F4 A9 FF          NSTATE  LDA     #$FF
D0F6 8D 0D A0               STA     IFR
D0F9 8D 8D 9F               STA     IFR2
D0FC A9 10                  LDA     #$10
D0FE 2C 8D 9F       CI7     BIT     IFR2               ;  FETCH CI7.CB1
D101 F0 FB                  BEQ     CI7                ;  LOOP HERE UNTIL CI7.CB1=1
D103 8D 8D 9F               STA     IFR2               ;  CLEAR FLAGS
D106 A2 30                  LDX     #$30               ;
D108 CA             L1      DEX                        ;  DELAY LOOP
D109 D0 FD                  BNE     L1                 ;
D10B AD 0D A0               LDA     IFR                ;  FETCH STATE
D10E A0 00                  LDY     #00                ;  CLEAR HIGH BYTE OF STATE TO 0
D110 60                     RTS                        ;  RETURN TO BASIC
D111                ;
D111                ;
D111                ;
D111                ;       THESE SUBROUTINES IMPLEMENT A SOFTWARE UART TO COMMUNICATE
D111                ;       WITH THE ORTEC 874 QUAD TIMER/COUNTER.THE CORRECT DELAY
D111                ;       TIMES FOR THE BAUD RATE ARE DETERMINED IN BASIC AND POKED
D111                ;       INTO T1LS AND T1HS.
D111                ;               THE RECEIVER SUBROUTINE RUART RETURNSTHE CHARACTER
D111                ;       IN THE ACCUMULATOR TO BASIC;WHILE THE TRANSMIT SUBROUTINE
D111                ;       TUART SENDS TO THE 874 A CHARACTER FROM BASIC STROBED
D111                ;       INTO THE TRANS STORAGE.
D111                ;
D111                ;
D111                        PORTB3  =$9F90
D111                ;
D111 8A             RUART   TXA                        ; SAVE X
D112 48                     PHA                        ;
```

Figure C.1 MISC.PRN (continued.)

```
D113 A2 07              LDX     #$07            ; SET UP FOR 8 BIT COUNT
D115 8E 09 D0           STX     RESS            ;
D118 2C 90 9F   RU1     BIT     PORTB3          ; A^M , PB6 -> V
D11B 70 FB              BVS     RU1             ; WAIT FOR START BIT
D11D 20 94 D1           JSR     DELAY           ; DELAY 1 BIT
D120 20 A8 D1           JSR     DEHALF          ; DELAY 1/2 BIT TIME
D123 AD 90 9F   RU2     LDA     PORTB3          ; GET 8 BITS
D126 29 40              AND     #$40            ; MASK OFF OTHER BITS,ONLY PB6
D128 4E 09 D0           LSR     RESS            ; SHIFT RIGHT CHARACTER
D12B 0D 09 D0           ORA     RESS            ;
D12E 8D 09 D0           STA     RESS            ;
D131 20 94 D1           JSR     DELAY           ; DELAY 1 BIT
D134 CA                 DEX                     ;
D135 D0 EC              BNE     RU2             ; GET NEXT BIT
D137 20 94 D1           JSR     DELAY           ; DO NOT CARE FOR PARITY
D13A 20 A8 D1           JSR     DEHALF          ; UNTIL WE GET BACK TO ONE AGAIN
D13D 68                 PLA                     ; RESTORE X
D13E AA                 TAX                     ;
D13F A0 00              LDY     #00             ; CLEAR Y
D141 AD 09 D0           LDA     RESS            ;
D144 29 7F              AND     #$7F            ; CLEAR PARITY BIT
D146 60                 RTS                     ; BACK TO BASIC
D147            ;
D147            ;
D147            ;
D147 48         TUART   PHA                     ; SAVE ACCUMULATOR
D148 8A                 TXA                     ; SAVE X
D149 48                 PHA                     ;
D14A 20 94 D1           JSR     DELAY           ;STOP BIT FROM LAST CHAR
D14D AD 90 9F           LDA     PORTB3          ;
D150 29 DF              AND     #$DF            ;
D152 8D 90 9F           STA     PORTB3          ; START BIT PB5=0
D155 8D 0A D0           STA     TBUFF           ; SAVE THIS PATTERN
D158 20 94 D1           JSR     DELAY           ;
D15B A2 08              LDX     #$08            ; 8 BITS
D15D 2E 08 D0           ROL     TRANS           ; GET FIRST LSB INTO BIT 5
D160 2E 08 D0           ROL     TRANS           ;
D163 2E 08 D0           ROL     TRANS           ;
D166 2E 08 D0           ROL     TRANS           ;
D169 2E 08 D0           ROL     TRANS           ;
D16C 2E 08 D0           ROL     TRANS           ;
D16F 6E 08 D0   TU1     ROR     TRANS           ;
D172 AD 08 D0           LDA     TRANS
D175 29 20              AND     #$20            ; GET ONLY BIT 5 FOR PB5
D177 0D 0A D0           ORA     TBUFF           ; PUT BIT INTO PATTERN
D17A 8D 90 9F           STA     PORTB3          ; NOW TO ORTEC 874
```

Figure C.1 MISC.PRN (continued.)

```
D17D 08                      PHP               ; PRESERVE CARRY FOR ROTATE
D17E 20 94 D1                JSR      DELAY    ;
D181 28                      PLP               ; RESTORE CARRY
D182 CA                      DEX               ;
D183 D0 EA                   BNE      TU1      ;
D185 A9 20                   LDA      #$20     ; STOP BIT
D187 0D 0A D0                ORA      TBUFF    ;
D18A 8D 90 9F                STA      PORTB3   ;
D18D 20 94 D1                JSR      DELAY    ; STOP BIT
D190 68                      PLA               ; RESTORE X
D191 AA                      TAX               ;
D192 68                      PLA               ; RESTORE ACCUMULATOR
D193 60                      RTS               ; BACK TO BASIC
D194                 ;
D194                 ;
D194                 ;       DELAY 1 BIT TIMES AS GIVEN BY BAUD RATE
D194                         PORTB2   =$9F80
D194                         T1L      =PORTB2+4
D194                         T1H      =PORTB2+5
D194 AD 06 D0        DELAY   LDA      T1LS     ; START TIMER T1
D197 8D 84 9F                STA      T1L      ;
D19A AD 07 D0                LDA      T1HS     ;
D19D 8D 85 9F        DE1     STA      T1H      ;
D1A0 AD 8D 9F        DE2     LDA      IFR2     ; GET INT FLAG FOR T1
D1A3 29 40                   AND      #$40     ;
D1A5 F0 F9                   BEQ      DE2      ; TIME OUT
D1A7 60                      RTS
D1A8                 ;
D1A8                 ;
D1A8                 ;
D1A8                 ;       DELAY HALF BIT TIME
D1A8                 ;       TOTAL TIME DIVIDED BY 2
D1A8 AD 07 D0        DEHALF  LDA      T1HS     ;
D1AB 4A                      LSR      A        ; LSB TO CARRY
D1AC AD 06 D0                LDA      T1LS     ;
D1AF 6A                      ROR      A        ; SHIFT WITH CARRY
D1B0 8D 84 9F                STA      T1L      ;
D1B3 AD 07 D0                LDA      T1HS     ;
D1B6 4A                      LSR      A        ;
D1B7 8D 85 9F                STA      T1H      ;
D1BA 4C A0 D1                JMP      DE2      ;
D1BD                 ;
D1BD                 ;
D1BD                 ;
D1BD                 ;       THIS SUBROUTINE READS ORTEC COUNTERS AND STORES
D1BD                 ;       THE VALUES AS 32 CONTIGUOUS BYTES IN MEMORY
D1BD                 ;
```

Figure C.1 MISC.PRN (continued.)

```
D1BD A2 01       RCOUNT  LDX      #01              ; INDEX MEMORY FOR STORAGE
D1BF A9 08       RC1     LDA      #08              ; # BYTES PER COUNTER
D1C1 8D 0C D0            STA      RBUFF+1          ; SAVE HERE
D1C4 E8          RC2     INX                       ; VECTOR MEMORY
D1C5 20 11 D1            JSR      RUART            ; READ BYTE
D1C8 9D 0B D0            STA      RBUFF,X          ; STORE IN MEMORY
D1CB CE 0C D0            DEC      RBUFF+1          ; # BYTES UNREAD PER COUNTER
D1CE D0 F4               BNE      RC2              ; READ ALL BYTES
D1D0 CE 0B D0            DEC      RBUFF            ; POINT TO NEXT COUNTER
D1D3 F0 06               BEQ      RC3              ; QUIT IF DONE
D1D5 20 11 D1            JSR      RUART            ; FETCH COMMA
D1D8 4C BF D1            JMP      RC1              ; REPEAT FOR NEXT COUNTER
D1DB 60          RC3     RTS                       ; BACK TO BASIC
D1DC             ;
D1DC             ;
D1DC             ;
D1DC
```

```
A10     D0C2     A11     D0D9     A12     D0E4     A6      D095
A8      D093     A9      D0A3     ALARM   D0AF     ALARMA  D0B3
BEEP    D001     BEEP1   D002     BEEP2   D003     BEEPA   D069
BEEPB   D08E     BEEPC   D06F     BEEPD   D0A1     CA1     D005
CI7     D0FE     CVIALP  D000     DE1     D19D     DE2     D1A0
DEHALF  D1A8     DELAY   D194     DON     D08D     FREQQ   D004
IFR     A00D     IFR2    9F8D     KEYCK   ECEF     L1      D108
NSTATE  D0F4     PORTA   A001     PORTB2  9F80     PORTB3  9F90
QUIT    D063     RBUFF   D00B     RC1     D1BF     RC2     D1C4
RC3     D1DB     RCOUNT  D1BD     RESS    D009     RSTATE  D0ED
RU1     D118     RU2     D123     RUART   D111     SERV    D043
SIRQ    D032     SPKR    9FBD     START   D000     T1H     9F85
T1HS    D007     T1L     9F84     T1LS    D006     TBUFF   D00A
TONE    D0E1     TRANS   D008     TU1     D16F     TUART   D147
WAIT    D07A

A>
```

Figure C.1 MISC.PRN (continued.)

```
  TYPE PROJ.BAS
REMARK THIS PROGRAM IS CALLED PROJ.BAS
REM
REMARK  VERSION I.1 8/16/86
REM
REMARK CHARACTER INITIALIZATION
        CR$  =CHR$(13)  REM CARRIAGE RETURN
        LF$  =CHR$(10)  REM LINE FEED
        BEEP$=CHR$(7)   REM WARNING BELL
REM SETTING UP REGISTERS AND DATA PORTS
        PORTB=40960     REM $A000
        PORTA=PORTB+1   REM PORT A DATA REG
        DDRB=PORTB+2    REM B DATA DIRECTION REG
        DDRA=PORTB+3    REM A DATA DIRECTION REG
        TIMEL1=PORTB+4  REM TIMER 1 WRITE LATCH LOW
        TIMEH1=PORTB+5  REM TIMER 1 READ COUNTER LOW
        TIML11=PORTB+6  REM
        TIMH11=PORTB+7  REM
        TIMEL2=PORTB+8  REM
        TIMEH2=PORTB+9  REM
        SR    =PORTB+10 REM SHIFT REGISTER
        ACR   =PORTB+11 REM AUXILIARY CONTROL REGISTER
        PCR   =PORTB+12 REM PERIPHERAL CONTROL REGISTER
        IFR   =PORTB+13 REM INTERRUPT FLAG REGISTER
        IER   =PORTB+14 REM INTERRUPT ENABLE REGISTER
        ORA   =PORTB+15 REM PORTA WITHOUT HANDSHAKE
        PORTB2=40832    REM $9F80
        PORTA2=PORTB2+1 REM PORT A DATA REG
        DDRB2=PORTB2+2  REM B DATA DIRECTION REG
        DDRA2=PORTB2+3  REM A DATA DIRECTION REG
        TIMEL12=PORTB2+4        REM TIMER 1 WRITE LATCH LOW
        TIMEH12=PORTB2+5        REM TIMER 1 READ COUNTER LOW
        TIML112=PORTB2+6        REM
        TIMH112=PORTB2+7        REM
        TIMEL22=PORTB2+8        REM
        TIMEH22=PORTB2+9        REM
        SR2    =PORTB2+10       REM SHIFT REGISTER
        ACR2   =PORTB2+11       REM AUXILIAR2Y CONTROL REGISTER
        PCR2   =PORTB2+12       REM PERIPHERAL CONTROL REGISTER
        IFR2   =PORTB2+13       REM INTERRUPT FLAG REGISTER
        IER2   =PORTB2+14       REM INTERRUPT ENABLE REGISTER
        ORA2   =PORTB2+15       REM PORTA WITHOUT HANDSHAKE
        PORTB3=40848    REM $9F90
        PORTA3=PORTB3+1 REM PORT A DATA REG
        DDRB3=PORTB3+2  REM B DATA DIRECTION REG
        DDRA3=PORTB3+3  REM A DATA DIRECTION REG
```

Figure C.2 PROJ.BAS listing

```
TIMEL13=PORTB3+4        REM TIMER 1 WRITE LATCH LOW
TIMEH13=PORTB3+5        REM TIMER 1 READ COUNTER LOW
TIML113=PORTB3+6        REM
TIMH113=PORTB3+7        REM
TIMEL23=PORTB3+8        REM
TIMEH23=PORTB3+9        REM
SR3    =PORTB3+10       REM SHIFT REGISTER
ACR3   =PORTB3+11       REM AUXILIARY CONTROL REGISTER
PCR3   =PORTB3+12       REM PERIPHERAL CONTROL REGISTER
IFR3   =PORTB3+13       REM INTERRUPT FLAG REGISTER
IER3   =PORTB3+14       REM INTERRUPT ENABLE REGISTER
ORA3   =PORTB3+15       REM PORTA WITHOUT HANDSHAKE
REM
REM
REM
REM REAL TIME CLOCK ADDRESS CONFIGURATION ON PORT B2
RCTHS   =0      REM COUNTER - TEN THOUSANDTH SECOND
RCTS    =1      REM COUNTER - HUNDREDTH & TENTH SEC.
RCSEC   =2      REM COUNTER - SECONDS
RCMIN   =3      REM COUNTER - MINUTES
RCHR    =4      REM COUNTER - HOURS
RCDOW   =5      REM COUNTER - DAY OF THE WEEK
RCDOM   =6      REM COUNTER - DAY OF THE MONTH
RCM     =7      REM COUNTER - MONTHS
RRTHS   =8      REM RAM - TEN THOUSANDTH SECOND
RRTS    =9      REM RAM - HUNDREDTH & TENTH SEC.
RRSEC   =10     REM RAM - SECONDS
RRMIN   =11     REM RAM - MINUTES
RRHR    =12     REM RAM - HOURS
RRDOW   =13     REM RAM - DAY OF THE WEEK
RRDOM   =14     REM RAM - DAY OF THE MONTH
RRM     =15     REM RAM - MONTHS
RISR    =16     REM INTERRUPT STATUS REGISTER
RICR    =17     REM INTERRUPT CONTROL REGISTER
RCR     =18     REM COUNTER RESET REGISTER
RRR     =19     REM RAM RESET REGISTER
RSB     =20     REM STATUS BIT REGISTER
RGC     =21     REM GO COMMAND REGISTER
RSTDI   =22     REM STANDBY INTERRUPT REGISTER
RTM     =31     REM TEST MODE REGISTER
REM
REM
REM
POKE PCR2,12    REM $0C HOLD CA2 LOW,DISABLE
                REM MOTOR DRIVES
POKE DDRA,255   REM $FF SETTUP PORTA AS OUTPUT
```

Figure C.2 PROJ.BAS (continued.)

```
POKE PORTA,224  REM $E0 TURN OFF ALL LEDS
POKE DDRB,255   REM $FF SETUP PORT B AS OUTPUTS
POKE PORTB,255  REM $FF GATE OFF ALL MOTORS
POKE PCR,53     REM $35 SET CA1,CA2,CB1 AS POSITIVE
                REM EDGE TRIGGERED INPUTS FOR CI3,
                REM CI5,CI6.FLAGS CLEARED BY PEEK
                REM OF PORTA
                REM ALSO SET CB2 AS NEGATIVE EDGE
                REM TRIGGERED INPUT FOR ORTEC INTERVAL
                REM COUNTER.CLEARED BY WRITING A 1 INTO
                REM BIT 3 OF IFR i.e. 08
POKE PORTB,223  REM $DF TURN ON BUSY LED , WRITE
                REM HI TO CO1,CO2,CO3
REM
REM
REM SETTING UP CLOCK OUTPUT FOR THE UART
POKE DDRA,224   REM $E0 SETTUP BITS 0-4 AS INPUTS
data=PEEK(PORTA)
S1.S2.PA3.PA4=data AND 24       REM $18 GET DIP SWITCH
                                REM S1S2 FOR BAUD RATE
                                REM CONTROL
IF S1.S2.PA3.PA4=0 THEN period.low=119
IF S1.S2.PA3.PA4=0 THEN period.high=35
                                REM S2S1=$00,T=9091
IF S1.S2.PA3.PA4=8 THEN period.low=249
IF S1.S2.PA3.PA4=8 THEN period.high=12
                                REM S2S1=$01,T=3333
IF S1.S2.PA3.PA4=16 THEN period.low=53
IF S1.S2.PA3.PA4=16 THEN period.high=3
                                REM S2S1=$10,T=833
IF S1.S2.PA3.PA4=24 THEN period.low=149
IF S1.S2.PA3.PA4=24 THEN period.high=1
                                REM S2S1=$11,T=417
REM AN OVERHEAD COUNT OF 12 HAS BEEN SUBTRACTED FROM
REM THE PERIODS IN MICROSECONDS.
POKE ACR2,0             REM $00 SET TIMER T1 FOR ONE SHOT
                        REM  MODE WITH OUTPUT
                        REM ON PB7.2 DISABLED
REM
REM
REM
REM CONFIGURING VIA.2 FOR INTERFACING THE REAL TIME
REM CLOCK RTC
POKE DDRA2,255  REM $FF SETUP PORT A2 AS OUTPUTS FOR NOW
POKE DDRB2,255  REM $FF SETTUP PORT B2 AS OUTPUTS
POKE PCR2,63    REM $3F SET CA1,CB1 AS POSITIVE EDGE TRIGGERED
```

Figure C.2 PROJ.BAS (continued.)

```
                         REM INPUTS FOR RDY.CA1,CI7.CB1
                         REM AND CA2 OUTPUT HIGH MODE TO
                         REM ENABLE MOTOR DRIVES,CB2 NEGATIVE EDGE TRIGGERED
                         REM INPUT FOR STB.INT
        POKE PORTB2,96   REM $60 TRI-STATE THE RTC
        REM
        REM
        REM
        REM CONFIGURE VIA.3 FOR SOFTWARE THE UART.
        REM THE UART IS SETUP FOR A 7 BIT ASCII CODE WITH
        REM NO PARITY i.e. 1 START BIT,1 STOP BIT,7 DATA BITS
        REM 0 PARITY BIT.
        POKE DDRB3,191   REM $BF BIT 5 IS OUTPUT:TRANSMIT DATA
                         REM BIT 6 IS INPUT:RECEIVED DATA
        POKE PCR3,32     REM $20 CA1,CB1,CB2 NEGATIVE  EDGE TRIGGERED INPUTS,
                         REM CLEAR CB1 BY WRITING 1 INTO BIT3 OF IFR3
        REM
        REM
        GOTO 21
5       REM SUBROUTINE RESET
        REM RESET UP/DOWN MOTORS AND VIAL POSITION
        REM
        REM
        LED2=223         REM $DF BUSY LED ON CODE
        GOSUB 30         REM ELEVATOR OUT OF DETECTOR
        NSV=0            REM INITIALIZE CURRENT
                         REM VIAL STATE TO STANDARD VIAL
        POKE PORTB,215   REM $D7 ENERGIZE CO3 ,START SAMPLE
                         REM CHANGER MOTOR,BUSY LED ON
        data1=CALL(RSTATE)       REM FETCH RESET STATE
        PRINT "3"
        data=data1
        GOTO 20
15      data=PEEK(IFR)
20      CI3.CA1=data AND 2       REM GET STATE OF CI3
        IF CI3.CA1=0 THEN 15     REM WAIT FOR CI3=1
        POKE PORTB,223   REM $DF STOP SAMPLE CHANGER MOTOR
                         REM IF CI3 IS HIGH i.e. VIAL
                         REM POSITION RESET TO ONE
        POKE IFR,2       REM CLEAR CA1 FLAG
        GOSUB 5280       REM UPDATE CURRENT VIAL STATE
        POKE CA1,1       REM SET CA1 FLAG
        GOSUB 5310       REM DETERMINES GROUP PLUG INDEX
        PRINT "NSV= ";NSV :
        RETURN
        REM
```

Figure C.2 PROJ.BAS (continued.)

```
         REM
         REM
21       REM CONTINUE
         REM OPEN A BLOCKED SCRATCH PAD FILE ON DISK WITH
         REM 20 BYTES PER RECORD
         B$      ='SCRATCH.BAS'
         FILE B$(20)
         REM
         REM NOW STORE STRING DATA ON FILE
         PRINT #1,1962;'SUNDAY'
         PRINT #1,1963;'MONDAY'
         PRINT #1,1964;'TUESDAY'
         PRINT #1,1965;'WEDNESDAY"
         PRINT #1,1966;'THURSDAY'
         PRINT #1,1967;'FRIDAY'
         PRINT #1,1968;'SATURDAY'
         PRINT #1,1969;'JANUARY'
         PRINT #1,1970;'FEBRUARY'
         PRINT #1,1971;'MARCH'
         PRINT #1,1972;'APRIL'
         PRINT #1,1973;'MAY'
         PRINT #1,1974;'JUNE'
         PRINT #1,1975;'JULY"
         PRINT #1,1976;'AUGUST'
         PRINT #1,1977;'SEPTEMBER'
         PRINT #1,1978;'OCTOBER"
         PRINT #1,1979;'NOVEMBER'
         PRINT #1,1980;'DECEMBER'
         PRINT #1,1985;'EOF.PROTECT'
         REM
         REM
         REM
         REM ASSEMBLY LANGUAGE PROGRAMS ARE BEING LOADED
         REM HERE WITH STARTING ADDRESSES AT D000 HEXADECIMAL
         REM
         REM
         DIM HEX(70)
         FOR X=0 TO 15
                 READ I
                 HEX(I)=X
         NEXT X
         DATA 48,49,50,51,52,53,54,55,56,57,65,66,67,68,69,70
                 A$='MISC.KIM'
                 FILE A$
                 IF END #2 THEN 26
22       READ #2;REC$
```

Figure C.2 PROJ.BAS (continued.)

```
                INDX=2
                GOSUB 24
                LENGTH=BYTE
                GOSUB 24
                ADDR=BYTE*256
                GOSUB 24
                ADDR=ADDR+BYTE
        FOR K=ADDR TO ADDR+LENGTH-1
                GOSUB 24
                POKE K,BYTE
        NEXT K
                GOTO 22
24      REM READ DATA CODE
                BYTE=HEX(ASC(MID$(REC$,INDX,1)))*16
                BYTE=HEX(ASC(MID$(REC$,INDX+1,1)))+BYTE
                INDX=INDX+2
        RETURN
26      REM END OF DATA INPUT
                CLOSE 2
        BEEPA=53353     REM $D069 DOES A BEEP
        BEEPC=53359     REM $D06F BEEP UNTIL KEY IS PRESSED
        BEEPD=53409     REM $D0A1 CHANGE FREQUENCY WITH EACH
                        REM CYCLE . STOP WITH RESET
        ALARM=53423     REM $D0AF ALARM UNTIL KEY IS PRESSED
        vector.int=53298        REM $D032 SETUP INTERRUPT ADDRESS
        CVIALP=53248            REM $D000
        T1LS=CVIALP+6
        T1HS=CVIALP+7
        TRANS=CVIALP+8
        RBUFF=CVIALP+11
        RCOUNT=53693            REM $D1BD
        RUART=53521             REM $D111
        TUART=53575             REM $D147
        CA1=53253               REM $D005
        RSTATE=53485            REM $D0ED RESET STATE
        NSTATE=53492            REM $D0F4 NEXT STATE
        X=CALL(vector.int)      REM RESET VIAL COUNTER,SET INTERRUPT VECTORS
        POKE T1LS,period.low    REM SAVE COUNTER LOW LATCH
        POKE T1HS,period.high   REM SAVE HIGH LATCH
        GOTO 29
        REM
        REM
        REM
28      REM SUBROUTINE POWER-UP RETRIEVAL
        REM RETRIEVE CURRENT VIAL STATE AND POSITION FROM RTC RAM
        POKE DDRA2,0            REM SETTUP PORTA2 AS INPUT
```

Figure C.2 PROJ.BAS (continued.)

```
        POKE PORTB2,RRDOM       REM SELECT STORAGE RAM FOR VIAL
                                REM STATE AND UNIT POSITION
        GOSUB 7050              REM READ PROTOCOL
        GOSUB 5650              REM CONVERT DATA TO HEX
        state.vial=INT(HEX.NUM/10)
        IF state.vial >= 1 THEN NSV=1
        PLOW=HEX.NUM-state.vial*10
        POKE PORTB2,RRTHS       REM SELECT STORAGE RAM FOR TENTHS
                                REM POSITION
        GOSUB 7050              REM READ PROTOCOL
        GOSUB 5650              REM CONVERT DATA TO HEX
        PHIGH=HEX.NUM
        PVIAL=PLOW+PHIGH        REM FORM VIAL POSITION
        POKE CVIALP,PVIAL       REM SAVE IN POWER-UP STORAGE
        RETURN
        REM
        REM
        REM
29      REM CONTINUE
        LINE=0
        GOSUB 8000              REM READ DATE
        DW=DAY.OF.WEEK
        GOSUB 7000              REM READ TIME
        READ #1,1962+DW;DDAY$
        READ #1,1969+MONTH;SMONTH$
        PRINT "PROJ.BAS";"   ";DDAY$;",";SMONTH$;\
              "   ";DAY;",";YEAR;"      ";HOURS;":";MINUTES;\
              ":";SECONDS
        PRINT LF$;LF$;LF$;CR$
        PRINT "PERFORM RESET WITH BATTERY CHANGE"
200     INPUT "RESET (Y/N)";state$
        IF state$="Y" THEN GOSUB 5 ELSE GOTO 300
        X=1
        GOTO 500
300     IF state$="N" THEN GOSUB 28 ELSE GOTO 400
        X=0
        GOTO 500
400     XX=CALL(BEEPA)          REM BEEP FOR ILLEGAL INPUT
        GOTO 200
500     IF X=0 THEN 600
        GOSUB 5770              REM UPDATE CURRENT VIAL STATE
600     REM CONTINUE
        POKE CA1,0              REM CLEARS CA1 FLAG
        POKE IER,130            REM $82 ENABLE INTERRUPT  ON ACTIVE
                                REM TRANSITION OF CA1 FLAG
        data=PEEK(CVIALP)
```

Figure C.2 PROJ.BAS (continued.)

```
        PRINT "vial position = ";data
        PRINT "NSV=";NSV
        PRINT "VIAL STATE =";state.vial
        GOTO 60
        REM
        REM
        REM
30      REM ELEVATOR OUT OF DETECTOR SUBROUTINE (EOOD)
        data=PEEK(PORTA)
        IF data AND 2 THEN GOTO 40 ELSE GOTO 50
40      POKE PORTB,219 AND LED2 REM $DB START UP-MOTOR WHEN CI2 IS HI
        GOTO 30
50      POKE PORTB,223 AND LED2 REM $DF DEENERGIZE CO2,STOP UP-MOTOR
                                REM WHEN CI2 LOW
        RETURN
        REM
        REM
        REM
60      REM YEAR UPDATE CODE
        POKE DDRA2,0            REM SETTUP PORTA2 AS INPUTS
        POKE PORTB2,RRM         REM SELECT MONTH RAM
                                REM SEND CS,RD LOW
        GOSUB 7050              REM READ PROTOCOL
        GOSUB 5650              REM CONVERT TO HEX
        HE1=HEX.NUM
        GOSUB 8030              REM READ MONTH
        GOSUB 5650              REM CONVERT TO HEX
        HE2=HEX.NUM
        HED=DECIMAL.NUM
        IF HE2 <> 1 THEN GOTO 62 ELSE GOTO 66
62      IF HE2 <> HE1 THEN GOTO 64 ELSE GOTO 68
64      POKE DDRA2,255          REM SETTUP PORTA2 AS OUTPUTS
        POKE PORTA2,HED         REM DATA OUT TO RAM
        POKE PORTB2,RRM+64      REM SELECT DAY OF MONTH RAM
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
66      IF HE1 = 1 THEN 68
        POKE DDRA2,255          REM SETTUP PORTA2 AS OUTPUTS
        POKE PORTA2,1
        POKE PORTB2,RRM+64      REM SELECT MONTH RAM
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        GOSUB 8040              REM READ YEAR.HIGH
        GOSUB 8050              REM READ YEAR.LOW
        GOSUB 5650              REM CONVERT TO HEX
        YEAR.LOW=HEX.NUM
```

Figure C.2 PROJ.BAS (continued.)

```
        YEAR=YEAR.HIGH*100 + YEAR.LOW
        YR=YEAR+1                      REM INCREMENT YEAR
        YEAR.HIGH=INT(YR/100)
        YEAR.LOW=YR-(YEAR.HIGH*100)
        HEX.NUM=YEAR.HIGH
        GOSUB 5660                     REM ENCODE IN DECIMAL
        YEAR.HIGH=DECIMAL.NUM
        HEX.NUM=YEAR.LOW
        GOSUB 5660                     REM ENCODE IN DECIMAL
        YEAR.LOW=DECIMAL.NUM
        POKE DDRA2,255                 REM SETTUP PORTA2 AS OUTPUTS
        POKE PORTA2,YEAR.HIGH
        POKE PORTB2,RRMIN+64           REM SELECT YEAR.HIGH STORAGE RAM
                                       REM SEND CS,WR LOW
        GOSUB 7210                     REM WRITE PROTOCOL
        POKE PORTA2,YEAR.LOW
        POKE PORTB2,RRTS+64            REM SELECT YEAR.LOW RAM
                                       REM SEND CS,WR LOW
        GOSUB 7210
68      REM CONTINUE
        REM
        REM
        REM
        REM PRINT MENU
        POKE PORTB,191  REM $BF LITE IDLE LED
        INPUT "S(tandby   A(ctive   Q(uit";state$
        IF state$="S" THEN GOSUB 1000 ELSE GOTO 70
        GOTO 68
70      IF state$="A" THEN  GOSUB 5000 ELSE GOTO 80
        GOTO 68
80      IF state$="Q" THEN GOTO 10000 ELSE GOTO 100
100     X=CALL(BEEPA)   REM BEEP FOR ILLEGAL INPUT
        GOTO 68         REM WAIT FOR CORRECT INPUT
        REM
        REM
        REM
1000    REM SUBROUTINE STANDBY INTEROGATES RTC
        REM READ AND SETS : TIME AND DATE
        POKE PORTA,191  REM $BF LITE STANDBY LED
        POKE PORTB,191  REM $BF LITE IDLE LED
1010    INPUT "ST(set.time SD(set.date RT(read.time RD(read.date  Q(uit";state$
        POKE PORTB,223  REM $DF LITE BUSY LED
        IF state$="Q" THEN GOTO 1020 ELSE GOTO 1030
1020    quit.flag=1
        GOTO 1120                     REM RETURN
1030    IF state$="ST" THEN GOTO 1040 ELSE GOTO 1050
```

Figure C.2 PROJ.BAS (continued.)

```
1040     GOSUB 7080                  REM SET TIME
         IF quit.flag=1 THEN  1120         REM RETURN
         GOTO 1010
1050     IF state$='SD' THEN GOTO 1060 ELSE GOTO 1070
1060     GOSUB 8070                  REM SET DATE
         IF quit.flag=1 THEN  1120         REM RETURN
         GOTO 1010
1070     IF state$='RT' THEN GOTO 1080 ELSE GOTO 1090
1080     GOSUB 7000                  REM READ TIME
         PRINT 'THE TIME IS ';HOURS;':';MINUTES;':';SECONDS
         GOTO 1010
1090     IF state$='RD' THEN GOTO 1100 ELSE GOTO 1110
1100     GOSUB 8000                  REM READ DATE
         DW=DAY.OF.WEEK
         READ #1,1962+DW;DDAY$
         READ #1,1969+MONTH;SMONTH$
         PRINT 'THE DATE IS ';DDAY$;',';SMONTH$;' ';DAY;',';YEAR
         GOTO 1010
1110     X=CALL(BEEPA)    REM BEEP FOR ILLEGAL INPUT
         GOTO 1010
1120     REM CONTINUE
         POKE PORTA,255   REM $FF TURN OFF STANDBY LED
         POKE PORTB,191   REM $BF LITE IDLE LED
         RETURN
         REM
         REM
5000     REM SUBROUTINE ACTIVE CONTROLS ALL FUNCTIONS
         POKE PORTA,127             REM $7F LITE ACTIVE LED
         POKE PORTB,191             REM $BF LITE BUSY LED
5010     INPUT ' G(roup  S(ingle   M(anual   Q(uit ';state$
         POKE PORTB,223             REM $DF LITE BUSY LED
         IF state$='G' THEN GOSUB 5050 ELSE GOTO 5020
         GOSUB 9400       REM RESET ORTEC 874 TO DEFAULT STATES
         GOTO 5010
5020     IF state$='S' THEN GOSUB 6000 ELSE GOTO 5025
         GOSUB 9400       REM RESET ORTEC 874 TO DEFAULT STATES
         SFLAG=1
         IGROUP=0 : GN=0
         GOTO 5010
5025     IF state$='M' THEN GOSUB 6500 ELSE GOTO 5030
         GOTO 5010
5030     IF state$='Q' THEN  5040
         X=CALL(BEEPA)              REM BEEP FOR ILLEGAL INPUT
         GOTO 5010
5040     REM CONTINUE
         POKE PORTA,255             REM $FF TURN OFF ACTIVE LED
```

Figure C.2 PROJ.BAS (continued.)

```
        POKE PORTB,191          REM $BF LITE IDLE LED
        RETURN
        REM
        REM
        REM
5050    REM SUBROUTINE GROUP CONTROLS THE GROUP PLUG
        REM OPERATING MODE
        SFLAG=0
        POKE PORTB,175          REM $AF LITE GROUP,IDLE LED
        quit.flag=0
5060    INPUT "enter number of groups <= 3 or Q(uit";group.number$
        POKE PORTB,207          REM $CF LITE GROUP,BUSY LED
        IF group.number$="Q" THEN  5080
        GN=VAL(group.number$)
        IF GN < 1 OR GN > 3 THEN X=CALL(BEEPA) ELSE GOTO 5070
                                REM BEEP FOR ILLEGAL INPUT
        GOTO 5060
5070    REM CONTINUE
        PRINT #1,1983;GN
        FOR IG=0 TO GN-1
        PRINT #1,1913+IG;0      REM SAMPLE.IN.GROUP
        PRINT #1,1916+IG;0      REM BEGIN.GROUP.SAMPLE
        NEXT IG
        FOR IGROUP = 0 TO GN-1
        PRINT "GROUP";" ";IGROUP+1;" ";"PARAMETERS"
        LED1=190                REM $BE GROUP,IDLE LED LITE CODE
        LED2=222                REM $DE GROUP,BUSY LED LITE CODE
        GOSUB 5090              REM OPERATING SYSTEM
        IF quit.flag=1 THEN  5080
        NEXT IGROUP
        FOR IGROUP=0 TO GN-1
        BRANCH=0
        READ #1,1904+IGROUP;preset$
        IF preset$="PT" THEN GOSUB 9030 ELSE GOSUB 9020
        REM LOAD COUNTER COMMANDS
        GOSUB 5420              REM MOTOR DRIVE
        NEXT IGROUP
        quit.flag=0
5080    REM CONTINUE
        POKE PORTB,223          REM $DF LITE BUSY LED
        RETURN
        REM
        REM
        REM
5090    REM SUBROUTINE OPERATING SYSTEM
        REM DETERMINES PRESET COUNT OR PRESET TIME MODES
```

Figure C.2 PROJ.BAS (continued.)

```
        REM OF OPERATION AND ACCEPTS TIME BASE PARAMETERS
        REM M AND N
        REM GATV
        PRINT #1,1928;1
        PRINT #1,1929;2
        PRINT #1,1930;4
        PRINT #1,1931;8
        POKE PORTB,LED1           REM $## LITE MODE,IDLE LED
5100    INPUT "PC(preset.count  PT(preset.time  Q(uit";preset$
        PRINT #1,1904+IGROUP;preset$
        POKE PORTB,LED2           REM $## LITE MODE,BUSY LED
        IF preset$ <> "Q" THEN GOTO 5120 ELSE GOTO 5110
5110    quit.flag=1
        GOTO 5270
5120    IF preset$="PT" THEN GOSUB 5480 ELSE GOTO 5130
        GOTO 5140
5130    IF preset$="PC" THEN GOSUB 5560 ELSE GOTO 5150
        GOTO 5265
5140    IF quit.flag=1 THEN GOTO 5270 ELSE GOTO 5160
5150    X=CALL(BEEPA)             REM BEEP FOR ILLEGAL INPUT
        GOTO 5100
5160    INPUT "SELECT COUNTERS 1,2,3,4 ,SELECT=1";CC0,\
        CC1,CC2,CC3
        PRINT #1,1950+IGROUP;CC0
        PRINT #1,1953+IGROUP;CC1
        PRINT #1,1956+IGROUP;CC2
        PRINT #1,1959+IGROUP;CC3
        PRINT "SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED"
5163    INPUT "COUNTER DISPLAY (1-4)";DP :
        IF DP < 1 OR DP > 4 THEN X=CALL(BEEPA) ELSE GOTO 5167
        GOTO 5163
5167    PRINT #1,1922+IGROUP;DP
5170    POKE PORTB,LED1           REM $## LITE MODE,IDLE LED
        INPUT " M ( 1 to 9 ) ,N ( 0 TO 6 ) or  Q(uit";MS$,NS$
        POKE PORTB,LED2           REM $## LITE MODE,BUSY LED
        IF MS$ <> "Q" THEN GOTO 5190 ELSE GOTO 5180
5180    quit.flag=1
        GOTO 5270                 REM RETURN
5190    M=VAL(MS$)                REM CONVERT TO INTEGER
        PRINT #1,1910+IGROUP;M
        IF M < 1 OR M > 9 THEN GOTO 5200 ELSE GOTO 5210
5200    X=CALL(BEEPA)             REM BEEP FOR ILLEGAL INPUT
        GOTO 5170
5210    REM COMTINUE
        IF NS$ <> "Q" THEN GOTO 5240 ELSE GOTO 5230
5230    quit.flag=1
```

Figure C.2 PROJ.BAS (continued.)

```
        GOTO 5270
5240    N=VAL(NS$)                  REM CONVERT TO INTEGER
        IF N < 0 OR N > 6 THEN GOTO 5250 ELSE GOTO 5260
5250    X=CALL(BEEPA)               REM BEEP FOR ILLEGAL INPUT
        GOTO 5170
5260    REM CONTINUE
        N=N+1
        PRINT #1,1907+IGROUP;N
5265    GOSUB 9000          REM DETERMINES MASK BIT SEQUENCE FOR
                            REM COUNTERS TO BE MASKED
        quit.flag=0
5270    REM CONTINUE
        POKE PORTB,LED1 REM $## LITE MODE,IDLE LED
        RETURN
        REM
        REM
        REM
5280    REM SUBROUTINE NONSTANDARD.VIAL (NSV)
        REM DETECTS EITHER AN EMPTY VIAL POSITION
        REM OR A GROUP PLUG
        REM AN EMPTY VIAL POSITION IS USED AS AN END
        REM OF GROUP INDICATOR
        CI6.CB1=data1 AND 16            REM GET STATE OF CI6
        IF CI6.CB1 =0 THEN NSV=1 ELSE NSV=0
        POKE IFR,16                     REM CLEAR CB1 FLAG
        RETURN
        REM
        REM
        REM
5290    REM SUBROUTINE NEXT
        REM FETCH NEXT SAMPLE
5300    CI3.CA1=PEEK(CA1)
        IF  CI3.CA1 =0 THEN  5300
        POKE PORTB,LED2             REM $## LITE MODE,BUSY LED
                                    REM WRITE CO3 HI,STOP
                                    REM SAMPLE CHANGER MOTOR
        RETURN
        REM
        REM
        REM
5310    REM SUBROUTINE GROUP.INDEX
        REM DETERMINES WHETHER A GROUP PLUG IS INDEXED
        REM AT THE ELEVATOR
        BOGP=0                      REM INITIALIZE BEGINNING OF
                                    REM GROUP PLUG FLAG TO 0
        EOGP=0                      REM INITIALIZE END OF GROUP
```

Figure C.2 PROJ.BAS (continued.)

```
                                     REM FLAG TO 0
        IF NSV=0 THEN 5317
5315    data=PEEK(CA1)               REM FETCH FLAG CA1
        IF data=0 THEN 5315          REM WAIT UNTIL CA1=1
        data=PEEK(IFR)
        CI5.CA2=data AND 1           REM GET STATE OF CI5
        IF CI5.CA2 =1 THEN BOGP=1 ELSE EOGP=1
        POKE IFR,1        REM  CLEAR CA2 FLAG
5317    RETURN
        REM
        REM
        REM
5320    REM SUBROUTINE BEGIN.GROUP
        REM INDEXES THE FIRST GROUP FLUG AT THE ELEVATOR
        IF BOGP=1 THEN  5360
        POKE PORTB,LED  REM $## LITE MODE,BUSY LED,CO3 LOW
                        REM START SAMPLE MOTOR
5340    data1=CALL(NSTATE)
        POKE CA1,0      REM CLEAR CA1 FLAG
        GOSUB 5280      REM DETECT NON STANDARD VIAL
        GOSUB 5310      REM CHECK GROUP FLUG INDEX
        IF BOGP=0 THEN 5340
5360    REM CONTINUE
        POKE CA1,0      REM CLEAR CA1 FLAG
        GOSUB 5770      REM UPDATE CURRENT VIAL STATE
        RETURN
        REM
        REM
        REM
5370    REM SUBROUTINE INDEX
        REM INDEXES A SAMPLE AT THE ELEVATOR
        POKE PORTB,LED1 REM $## LITE MODE,BUSY LED,CO3 HI
                        REM RUN SAMPLE CHANGER MOTOR
        data1=CALL(NSTATE)
        GOSUB 5280      REM NON.STANDARD VIAL DETECTION
5390    GOSUB 5290      REM FETCH NEXT SAMPLE
        GOSUB 5310      REM CHECKS TO SEE IF A GROUP FLUG
                        REM IS ALREADY INDEXED
5400    REM CONTINUE
        POKE CA1,0      REM CLEAR CA1 FLAG
        GOSUB 5770      REM UPDATE CURRENT VIAL STATE
        RETURN
        REM
        REM
        REM
5410    REM SUBROUTINE LOWER.SAMPLE
```

Figure C.2 PROJ.BAS listing

```
        REM LOWERS A SAMPLE INTO THE DETECTOR
        POKE PORTB,LED1 REM $## LITE MODE,BUSY LED,CO1 LOW
                        REM LOWER SAMPLE INTO DETECTOR
        data=PEEK(PORTA)
        CI1.PAO=data AND 1
        IF CI1.PAO =1 THEN  5410
        POKE PORTB,LED2 REM $## LITE MODE,BUSY LED,CO1 HI
                        REM STOP DOWN.MOTOR
        RETURN
        REM
        REM
        REM
5420    REM SUBROUTINE MOTOR.DRIVE
        REM HANDLES ALL THE MOTOR DRIVE PROTOCOLS FOR THE
        REM GROUP PLUG MODE OF OPERATION
        GOSUB 8000              REM READ DATE
        IF BRANCH=2 THEN 5430
        BRANCH=1
        LED2=222
        GOSUB 30        REM ELEVATOR OUT OF DETECTOR
        LED=214         REM $D6 LITE GROUP,BUSY LED,CO3 LOW CODE
        GOSUB 5320      REM GROUP PLUG INDEXED AT ELEVATOR
5430    LED1=214                REM CO3 LOW CODE
        LED2=222                REM CO3 HI CODE
        GOSUB 5370      REM INDEX VIAL AFTER GROUP PLUG
        IF BOGP=1 OR EOGP=1 THEN   5470
        READ #1,1913+IGROUP;SAMPLE.IN.GROUP
        SAMPLE.IN.GROUP=SAMPLE.IN.GROUP+1
        PRINT #1,1913+IGROUP;SAMPLE.IN.GROUP
        IF BRANCH=1 THEN GOTO 5440 ELSE GOTO 5450
        REM THESE STATEMENTS ARE TO BE EXECUTED ONLY ONCE PER
        REM GROUP
5440    BEGIN.GROUP.SAMPLE=PEEK(CVIALP)
        PRINT #1,1916+IGROUP;BEGIN.GROUP.SAMPLE
        BRANCH=2
5450    REM CONTINUE
        LED1=220        REM $DC LITE GROUP,BUSY LED CO1 HI CODE
        LED2=222        REM LITE GROUP,BUSY LED CO1 LOW CODE
5460    GOSUB 5410      REM LOWER SAMPLE INTO DETECTOR
        GOSUB 9250      REM START COUNTER
        GOSUB 7000      REM READ START TIME
        POKE PORTA,95   REM $5F LITE ACTIVE,COUNT LED
        REM SAVE START TIME IN ARRAYS
        READ #1,1913+IGROUP;SAMPLE.IN.GROUP
        JINDEX=SAMPLE.IN.GROUP-1
        SEC=HOURS*3600+MINUTES*60+SECONDS
```

Figure C.2 PROJ.BAS (continued.)

```
        PRINT #1,1201+200*IGROUP+JINDEX;SEC
        X=100*IGROUP+JINDEX
        FOR K=0 TO 3
        XX=X+300*K
        PRINT #1,XX+1;0
        NEXT K
        READ #1,1904+IGROUP;preset$
        IF preset$="PT" THEN GOSUB 9260 ELSE GOSUB 9170
        REM ORTEC COUNTER TIME-OUT
        POKE PORTA,127  REM $7F LITE ACTIVE LED
        READ #1,1904+IGROUP;preset$
        IF preset$="PC" THEN 5465
        GOSUB 7000      REM READ STOP TIME
        REM SAVE STOP TIME IN ARRAYS
        SEC=HOURS*3600+MINUTES*60+SECONDS
        PRINT #1,1301+200*IGROUP+JINDEX;SEC
        REM READ COUNTERS
        GOSUB 9290      REM READ COUNTERS
5465    REM CONTINUE
        GOSUB 30        REM ELEVATOR OUT OF DETECTOR
        GOTO 5430       REM LOOP HERE UNTIL EOGP=1
5470    REM CONTINUE
        RETURN
        REM
        REM
        REM
5480    REM SUBROUTINE PRESET.TIME
        REM ACCEPTS THE PRESET TIME BASE FOR THE SYSTEM
        REM EITHER .1 SEC , .1 MIN , OR EXTERNAL
        REM PRESET TIME PERIOD = M * 10 ^ N * TIME BASE SELECTION
5490    INPUT " time base S(sec M(min E(ext Q(uit";time.base$
        PRINT #1,1901+IGROUP;time.base$
        IF time.base$ <> "Q" THEN GOTO 5510 ELSE GOTO 5500
5500    quit.flag=1
        GOTO 5540
5510    IF time.base$ <> "S" AND time.base$ <> "M" \
        AND time.base$ <> "E" THEN GOTO 5520 ELSE GOTO 5530
5520    X=CALL(BEEPA)   REM BEEP FOR ILLEGAL INPUT
        GOTO 5490
5530    quit.flag=0
5540    RETURN
        REM
        REM
        REM
5560    REM SUBROUTINE PRESET.COUNT
        time.base$="E"  REM EXTERNAL TIME BASE SELECTED
```

Figure C.2 PROG.BAS (continued.)

```
                       REM FOR PRESET COUNT
        PRINT #1,1901+IGROUP;time.base$
        INPUT "SELECT COUNTERS 1,2,3,4 ,SELECT=1";CC0,\
              CC1,CC2,CC3
        PRINT #1,1950+IGROUP;CC0
        PRINT #1,1953+IGROUP;CC1
        PRINT #1,1956+IGROUP;CC2
        PRINT #1,1959+IGROUP;CC3
        PRINT "SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED"
5563    INPUT "COUNTER DISPLAY(1-4)";DP:
        IF DP < 1 OR DP > 4 THEN X=CALL(BEEPA) ELSE GOTO 5567
        GOTO 5563
5567    PRINT #1,1922+IGROUP;DP
        PRINT "MAX PRESETS : COUNTER 1=9 E+07;2,3,4=1 E+38"
        PRINT "ENTER PRESETS FOR SELECTED COUNTERS" :
        FOR ICOUNT=0 TO 3
5570    REM CONTINUE
        READ #1,1950+3*ICOUNT+IGROUP;CC
        IF CC=1 THEN INPUT PCNT
        PRINT #1,1938+4*IGROUP+ICOUNT;PCNT
        READ #1,1938+4*IGROUP;PCNT0
        IF PCNT < 0 OR PCNT \
           > 1E+38 OR PCNT0 > 9E+07 THEN 5580
        GOTO 5590
5580    X=CALL(BEEPA)
        GOTO 5570
5590    REM CONTINUE
        NEXT ICOUNT
        quit.flag=0
        RETURN
        REM
        REM
        REM
5650    REM SUBROUTINE DECIMAL.TO.HEX
        REM ASSIGN WEIGHTS
        W1=65535
        W2=4096
        W3=256
        W4=16
        A=INT(DECIMAL.NUM/W1)
        NUM.NEW=DECIMAL.NUM-(W1*A)
        B=INT(NUM.NEW/W2)
        NUM.NEW=NUM.NEW-(W2*B)
        C=INT(NUM.NEW/W3)
        NUM.NEW=NUM.NEW-(W3*C)
        D=INT(NUM.NEW/W4)
```

Figure C.2 PROJ.BAS (continued.)

```
        E=NUM.NEW-(W4*D)
        HEX.NUM=A*(10^4)+B*(10^3)+C*(10^2)+D*10+E
        RETURN
        REM
        REM
        REM
5660    REM SUBROUTINE HEX.TO.DECIMAL
        V=INT(HEX.NUM/(10^4))
        NUM.NEW=HEX.NUM-(V*(10^4))
        W=INT(NUM.NEW/(10^3))
        NUM.NEW=NUM.NEW-(W*(10^3))
        X=INT(NUM.NEW/(10^2))
        NUM.NEW=NUM.NEW-(X*(10^2))
        Y=INT(NUM.NEW/10)
        Z=NUM.NEW-(Y*10)
        DECIMAL.NUM=V*65535+W*4096+X*256+Y*16+Z
        RETURN
        REM
        REM
        REM
5770    REM SUBROUTINE UPDATE.CURRENT
        REM UPDATE STATE OF CURRENT VIAL INDEXED
        IF NSV=0 THEN GOTO 5780 ELSE GOTO 5790
5780    state.vial=0    REM SET STANDARD VIAL STATE
        GOTO 5820
5790    IF BOGP=1 THEN GOTO 5800 ELSE GOTO 5810
5800    state.vial=1    REM SET GROUP PLUG STATE
        GOTO 5820
5810    state.vial=2    REM SET EMPTY VIAL STATE
5820    RETURN
        REM
        REM
        REM
5830    REM SUBROUTINE ERROR
        REM RINGS ALARM AND PRINTS ERROR MESSAGE IF A
        REM NON STANDARD VIAL IS INDEXED AT THE ELEVATOR
        REM IN THE SINGLE STEP MODE
        IF state.vial=1 THEN GOTO 5840 ELSE GOTO 5850
5840    FOR J=0 TO 2
        X=CALL(BEEPA)
        NEXT J
        PRINT 'ERROR*** indexed vial position';' ';SN;\
        'contains a group plug'
        ERROR=1
        GOTO 5880                REM RETURN
5850    IF state.vial=2 THEN GOTO 5860 ELSE GOTO 5870
```

Figure C.2 PROJ.BAS (continued.)

```
5860    FOR J=0 TO 2
        X=CALL(BEEPA)
        NEXT J
        PRINT 'ERROR*** indexed vial position';' ';SN;\
        'is empty'
        ERROR=1
        GOTO 5880               REM RETURN
5870    ERROR=0
5880    RETURN
        REM
        REM
        REM
6000    REM SUBROUTINE SINGLE CONTROLS THE SINGLE STEP
        REM OPERATING MODE
        POKE PORTB,222          REM $DE LITE SINGLE,BUSY LED
        IGROUP=0 : GN=1 : SFLAG=1
        PRINT #1,1983;GN
        JSINGLE=0
6010    LED1=190                REM $BE LITE SINGLE,IDLE LED CODE
        LED2=222                REM $DE SINGLE,BUSY LED CODE
        quit.flag=0
        GOSUB 5090              REM CALL OPERATING SYSTEM
        IF quit.flag=1 THEN 6190        REM RETURN
        READ #1,1904;preset$
        IF preset$='PT' THEN GOSUB 9030 ELSE GOSUB 9020
        REM LOAD COUNTER COMMANDS
        IF quit.flag=1 THEN  6190       REM RETURN
6020    REM CONTINUE.
        POKE PORTB,190  REM $BE LITE SINGLE,IDLE LED
        INPUT 'sample number(1-100) or Q(uit';sample.number$
        POKE PORTB,222          REM $DE LITE SINGLE,BUSY LED
        IF sample.number$='Q' THEN GOTO 6030 ELSE GOTO 6040
6030    quit.flag=1
        GOTO 6190               REM RETURN
6040    SN=VAL(sample.number$)
        IF SN < 1 OR SN > 100 THEN GOTO 6050 ELSE GOTO 6060
6050    X=CALL(BEEPA)           REM BEEP FOR ILLEGAL INPUT
        GOTO 6020
6060    VIALP=PEEK(CVIALP)      REM OBTAIN CURRENT VIAL POSITION
        IF SN=VIALP THEN GOTO 6100 ELSE GOTO 6070
6070    GOSUB 30                REM ELEVATOR OUT OF DETECTOR
        IF SN > VIALP THEN GOTO 6090 ELSE GOTO 6080
6080    FOR J=VIALP TO 99+SN
        LED1=214                REM $D6 SINGLE,BUSY C03 LOW CODE
        LED2=222                REM $DE SINGLE,BUSY C03 HI CODE
        GOSUB 5370              REM INDEX NEXT VIAL
```

Figure C.2 PROJ.BAS (continued.)

```
        NEXT J
        GOTO 6110
6090    FOR I=VIALP TO SN-1
        LED1=214                REM $D6 SINGLE,BUSY C03 LOW CODE
        LED2=222                REM $DE SINGLE,BUSY C03 HI CODE
        GOSUB 5370              REM INDEX NEXT VIAL
        NEXT I
6100    REM CONTINUE
6110    GOSUB 5830      REM TRAP NON STANDARD VIALS
        IF ERROR=1 THEN  6020
        GOSUB 8000              REM READ DATE
        LED1=220                REM $DC SINGLE,BUSY LED ,C01 LOW CODE
        LED2=222                REM $DE SINGLE,BUSY LED ,C01 HI CODE
        GOSUB 5410      REM LOWER SAMPLE INTO THE DETECTOR
        JSINGLE=JSINGLE+1
        FOR K=0 TO 3
        PRINT #1,JSINGLE+300*K;0
        NEXT K
        PRINT #1,1801+JSINGLE-1;SN
        GOSUB 9250              REM START COUNTER
        GOSUB 7000      REM READ START TIME
        POKE PORTA,95   REM $5F LITE ACTIVE , COUNT LED
        REM SAVE START TIME ON SCRATCH FILE
        SEC=HOURS*3600+MINUTES*60+SECONDS
        PRINT #1,1201+JSINGLE-1;SEC
        READ #1,1904;preset$
        IF preset$="PT" THEN GOSUB 9260 ELSE GOSUB 9170
                        REM COUNTER TIME-OUT
        POKE PORTA,127          REM $7F LITE ACTIVE LED,
                                REM OFF COUNT LED
        IF preset$="PC" THEN 6115
        GOSUB 7000              REM READ STOP TIME
        REM SAVE STOP TIME IN ARRAYS
        SEC=HOURS*3600+MINUTES*60+SECONDS
        PRINT #1,1301+JSINGLE-1;SEC
        REM READ COUNTERS
        GOSUB 9290              REM READ COUNTERS
6115    REM CONTINUE
        GOSUB 30        REM ELEVATOR OUT OF DETECTOR
6120    INPUT "repeat for another sample Y(es N(o Q(uit";state$
        IF state$="Q" THEN GOTO 6130 ELSE GOTO 6140
6130    quit.flag=1
        GOTO 6190               REM RETURN
6140    IF state$="Y" THEN GOTO 6150 ELSE GOTO 6160
6150    INPUT "with same parameters Y(es N(o";para$
        IF para$="Y" THEN  6020
```

Figure C.2 PROJ.BAS (continued.)

```
        IF para$='N' THEN  6010
        X=CALL(BEEPA)   REM BEEP FOR ILLEGAL INPUT
        GOTO 6150
6160    IF state$='N' THEN GOTO 6180 ELSE GOTO 6170
6170    X=CALL(BEEPA)   REM BEEP FOR ILLEGAL INPUT
        GOTO 6120
6180    quit.flag=0
6190    REM CONTINUE
        PRINT #1,1913;JSINGLE
        POKE PORTB,223          REM $DF LITE IDLE LED
        RETURN
        REM
        REM
        REM
        REM
6500    REM SUBROUTINE MANUAL
        REM ALL ORTEC TIMER/COUNTER COMMANDS ARE ENTERED
        REM BY THE OPERATOR VIA FRONT PANEL BUTTONS.
        REM THIS SUBROUTINE ALLOWS UP/DOWN CONTROL OF THE
        REM ELEVATOR MOTORS AND VIAL INDEXING.
        PRINT '874 ORTEC TIMER/COUNTER MUST BE IN LOCAL MODE'
6510    POKE PORTB,63           REM $3F LITE MANUAL,IDLE LED
        INPUT'U(p motor D(own motor N(ext sample G(roup Q(uit';state$
        POKE PORTB,95           REM $5F LITE MANUAL,BUSY LED
        IF state$='U' THEN GOTO 6520 ELSE GOTO 6530
6520    LED2=95
        GOSUB 30                REM ELEVATOR OUT OF DETECTOR
        GOTO 6510
6530    IF state$='D' THEN GOTO 6540 ELSE GOTO 6550
6540    REM CONTINUE
        SN=PEEK(CVIALP) REM FETCH CURRENT INDEXED POSITION
        GOSUB 5830      REM TRAP NON STANDARD VIALS
        IF state.vial <> 0 THEN 6510
        LED1=93         REM $5D LITE MANUAL,BUSY LED C01 LOW CODE
        LED2=95         REM $5F LITE MANUAL,BUSY LED C01 HI CODE
        GOSUB 5410      REM LOWER SAMPLE INTO DETECTOR
        GOTO 6510
6550    IF state$='N' THEN GOTO 6560 ELSE GOTO 6570
6560    REM CONTINUE
        LED2=95
        GOSUB 30
        LED1=87         REM $57 LITE MANUAL,BUSY LED C03 LOW CODE
        LED2=95         REM $5F C03 HI
        GOSUB 5370      REM INDEX NEXT SAMPLE
        SN=PEEK(CVIALP)
        GOSUB 5830      REM TRAP NON STANDARD VIALS
```

Figure C.2 PROJ.BAS (continued.)

```
        GOTO 6510
6570    IF state$='G' THEN GOTO 6580 ELSE GOTO 6590
6580    REM CONTINUE
        LED2=95
        GOSUB 30
        LED=87              REM $57 CO3 LOW CODE
        GOSUB 5320          REM INDEXES A GROUP PLUG
        LED1=87             REM $57 CO3 LOW CIDE
        LED2=95             REM $5F CO3 HI CODE
        GOSUB 5370          REM INDEXES VIAL AFTER GROUP PLUG
        SN=PEEK(CVIALP)
        GOSUB 5830          REM TRAP NON STANDARD VIALS
        GOTO 6510
6590    IF state$='Q' THEN GOTO 6610 ELSE GOTO 6600
6600    X=CALL(BEEPA)       REM BEEP FOR ILLEGAL INPUT
        GOTO 6510
6610    REM CONTINUE
        POKE PORTB,223      REM $DF LITE IDLE LED
        RETURN
        REM
        REM
        REM
7000    REM READ TIME
        REM READS HOURS , MINUTES , SECONDS
7010    GOSUB 7020          REM READ HOUR
        GOSUB 5650          REM CONVERT TO HEX
        HOURS=HEX.NUM
        GOSUB 7030          REM READ MINUTES
        DE1=DECIMAL.NUM
        GOSUB 7030          REM READ MINUTES AGAIN
        IF DECIMAL.NUM >= DE1 THEN 7015
        GOTO 7010
7015    GOSUB 5650          REM CONVERT TO HEX
        MINUTES=HEX.NUM
        GOSUB 7040          REM READ SECONDS
        DE1=DECIMAL.NUM
        GOSUB 7040          REM READ SECONDS AGAIN
        IF DECIMAL.NUM >= DE1 THEN 7017
        GOTO 7010
7017    GOSUB 5650          REM CONVERT TO HEX
        SECONDS=HEX.NUM
        RETURN
        REM
        REM
        REM
7020    REM SUBROUTINE READ HOUR
```

Figure C.2 PROJ.BAS (continued.)

```
        POKE PORTB2,RCHR          REM SELECT HR COUNTER CS,RD LOW
        GOSUB 7050                REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
7030    REM SUBROUTINE READ MINUTES
        POKE PORTB2,RCMIN         REM SELECT MINUTE COUNTER SEND
                                  REM CS,RD LOW
        GOSUB 7050                REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
7040    REM SUBROUTINE READ SECONDS
        POKE PORTB2,RCSEC         REM SELECT SECOND COUNTER SEND
                                  REM CS,RD LOW
        GOSUB 7050                REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
7050    REM SUBROUTINE READ PROTOCOL
        POKE DDRA2,0    REM SETTUP PORTA2 AS INPUTS
7060    data=PEEK(IFR2)
        RDY.CA1=data AND 2        REM GET STATE OF RDY
        IF  RDY.CA1 =0 THEN  7060         REM WAIT FOR RDY ^
        DECIMAL.NUM=PEEK(PORTA2)          REM READ DATA CLEAR CA1 FLAG
        POKE PORTB2,96            REM $60 TRI-STATE THE RTC
        RETURN
        REM
        REM
        REM
        REM
        REM
        REM
7080    REM SUBROUTINE SET TIME
7090    INPUT " S(et time   Q(uit";state$
        IF state$="Q" THEN GOTO 7100 ELSE GOTO 7110
7100    quit.flag=1
        GOTO 7200                 REM RETURN
7110    IF state$="S" THEN GOTO 7130 ELSE GOTO 7120
7120    X=CALL(BEEPA)             REM BEEP FOR ILLEGAL INPUT
        GOTO 7090
7130    REM CONTINUE
        POKE DDRA2,255  REM SETTUP PORTA2 AS OUTPUTS
```

Figure C.2 PROJ.BAS (continued.)

```
        POKE PORTA2,28  REM $1C RESET TIME COUNTER CODE
        POKE PORTB2,RCR+64      REM SELECT THE COUNTER RESET REGISTER
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        INPUT " enter HOURS ";HH
        IF HH < 0 OR HH > 24 THEN GOTO 7140 ELSE GOTO 7150
7140    X=CALL(BEEPA)           REM BEEP FOR ILLEGAL INPUT
        GOTO 7130
7150    INPUT " enter MINUTES " ;MM
        IF MM < 0 OR MM > 59 THEN GOTO 7160 ELSE GOTO 7170
7160    X=CALL(BEEPA)           REM BEEP FOR ILLEGAL INPUT
        GOTO 7150
7170    INPUT " enter SECONDS ";SS
        IF SS < 0 OR SS > 59 THEN GOTO 7180 ELSE GOTO 7190
7180    X=CALL(BEEPA)
        GOTO 7170
7190    REM ENCODE IN DECIMAL
        HEX.NUM=HH
        GOSUB 5660      REM HEX TO DECIMAL ROUTINE
        HH=DECIMAL.NUM
        HEX.NUM=MM
        GOSUB 5660
        MM=DECIMAL.NUM
        HEX.NUM=SS
        GOSUB 5660
        SS=DECIMAL.NUM
        REM WRITE TO GO REGISTER FOR PRECISE STARTING OF CLOCK
        REM DATA WRITTEN IS IGNORED
        POKE PORTA2,0   REM DUMMY DATA IGNORED
        POKE PORTB2,RGC+64      REM SELECT GO REGISTER,SEND
                                REM CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        POKE PORTA2,SS  REM DATA OUT TO SECONDS REGISTER
        POKE PORTB2,RCSEC+64    REM SELECT SECONDS COUNTER
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        POKE PORTA2,MM  REM DATA OUT TO MINUTES REGISTER
        POKE PORTB2,RCMIN+64    REM SELECT MINUTES COUNTER
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        POKE PORTA2,HH  REM DATA OUT TO HOURS REGISTER
        POKE PORTB2,RCHR+64     REM SELECT HOUR COUNTER
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        GOSUB 7000              REM READ TIME
        PRINT "THE TIME IS ";HOURS;":";MINUTES;":";SECONDS
```

Figure C.2 PROJ.BAS (continued.)

```
        quit.flag=0
7200    RETURN
        REM
        REM
        REM
7210    REM SUBROUTINE RTC WRITE PROTOCOL
7220    data=PEEK(IFR2)
        RDY.CA1=data AND 2       REM GET STATE OF RDY
        IF  RDY.CA1 =0 THEN  7220        REM WAIT FOR RDY
        POKE IFR2,2     REM CLEAR CA1 FLAG
        POKE PORTB2,96           REM TRI-STATE THE RTC
        RETURN
        REM
        REM
        REM
8000    REM READ DATE
        REM READS DAY OF THE WEEK,DAY,MONTH AND YEAR
        POKE DDRA2,0    REM SETTUP PORTA2 AS INPUTS
8010    GOSUB 8060      REM READ DAY OF THE WEEK
        DE1=DECIMAL.NUM
        GOSUB 8060      REM READ DAY OF THE WEEK AGAIN
        IF DECIMAL.NUM >= DE1 THEN 8015
        GOTO 8010
8015    DAY.OF.WEEK=DECIMAL.NUM-1
        GOSUB 8020      REM READ DAY OF THE MONTH
        GOSUB 5650      REM CONVERT TO HEX
        DAY=HEX.NUM
        GOSUB 8030      REM READ MONTH
        GOSUB 5650      REM CONVERT TO HEX
        MONTH=HEX.NUM-1
        GOSUB 8040      REM READ YEAR.HIGH
        GOSUB 8050      REM READ YEAR.LOW
        GOSUB 5650      REM CONVERT TO HEX
        YEAR.LOW=HEX.NUM
        YEAR=(YEAR.HIGH*100)+YEAR.LOW
        RETURN
        REM
        REM
        REM
8020    REM SUBROUTINE READ DAY OF MONTH
        POKE PORTB2,RCDOM        REM SELECT DAY OF MONTH COUNTER
                                 REM SEND CS,RD LOW
        GOSUB 7050               REM IMPLEMENT READ PROTOCOL
        RETURN
        REM
        REM
```

Figure C.2 PROJ.BAS (continued.)

```
        REM
8030    REM SUBROUTINE READ MONTH
        POKE PORTB2,RCM         REM SELECT MONTH COUNTER
                                REM SEND CS,RD LOW
        GOSUB 7050              REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
8040    REM SUBROUTINE READ YEAR.HIGH
        YEAR.HIGH=19
        RETURN
        REM
        REM
        REM
8050    REM SUBROUTINE READ YEAR.LOW
        POKE PORTB2,RRTS        REM SELECT YEAR.LOW RAM
                                REM SEND CS,RD LOW
        GOSUB 7050              REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
8060    REM SUBROUTINE READ DAY OF THE WEEK
        POKE PORTB2,RCDOW       REM SELECT DAY OF WEEK COUNTER
                                REM SEND CS,RD LOW
        GOSUB 7050              REM READ PROTOCOL
        RETURN
        REM
        REM
        REM
8070    REM SUBROUTINE SET DATE
8080    INPUT "S(et date   Q(uit";state$
        IF state$="Q" THEN GOTO 8090 ELSE GOTO 8100
8090    quit.flag=1
        GOTO 8210               REM RETURN
8100    IF state$="S" THEN GOTO 8120 ELSE GOTO 8110
8110    X=CALL(BEEPA)           REM BEEP FOR ILLEGAL INPUT
        GOTO 8080
8120    REM CONTINUE
        POKE DDRA2,255          REM SETTUP PORTA2 AS OUTPUTS
        POKE PORTA2,10          REM $0A RESET YEAR RAM CODE
        POKE PORTB2,RRR+64      REM SELECT RAM RESET REGISTER
                                REM SEND CS,WR LOW
        GOSUB 7210              REM WRITE PROTOCOL
        INPUT " enter YEAR ";YR
```

Figure C.2 PROJ.BAS (continued.)

```
        IF YR < 1986 OR YR > 5999 THEN GOTO 8130 ELSE GOTO 8140
8130    X=CALL(BEEPA)            REM BEEP FOR ILLEGAL INPUT
        GOTO 8120
8140    INPUT " enter MONTH (1-12) ";MM
        IF MM < 1 OR MM > 12 THEN GOTO 8150 ELSE GOTO 8160
8150    X=CALL(BEEPA)            REM BEEP FOR ILLEGAL INPUT
        GOTO 8140
8160    INPUT " enter day of the month (1-31) ";DD
        IF DD < 1 OR DD > 31 THEN GOTO 8170 ELSE GOTO 8180
8170    X=CALL(BEEPA)            REM BEEP FOR ILLEGAL INPUT
        GOTO 8160
8180    INPUT " enter day of the week (1-7) ";DW
        IF DW < 1 OR DW > 7 THEN GOTO 8190 ELSE GOTO 8200
8190    X=CALL(BEEPA)            REM BEEP FOR ILLEGAL INPUT
        GOTO 8180
8200    REM ENCODE DATA IN DECIMAL
        HEX.NUM=DD
        GOSUB 5660               REM CONVERT TO DECIMAL
        DD=DECIMAL.NUM
        HEX.NUM=MM
        GOSUB 5660
        MM=DECIMAL.NUM
        YEAR.HIGH=INT(YR/100)
        YEAR.LOW=YR-(YEAR.HIGH*100)
        HEX.NUM=YEAR.LOW
        GOSUB 5660
        YEAR.LOW=DECIMAL.NUM
        POKE PORTA2,DW           REM DAY OF WEEK OUT TO COUNTER
        POKE PORTB2,RCDOW+64     REM SEND CS,WR LOW
                                 REM SELECT DAY OF WEEK COUNTER
        GOSUB 7210               REM WRITE PROTOCOL
        POKE PORTA2,DD
        POKE PORTB2,RCDOM+64     REM SELECT DAY OF MONTH COUNTER
                                 REM SEND CS,WR LOW
        GOSUB 7210               REM WRITE PROTOCOL
        POKE PORTA2,MM           REM MONTHS OUT TO COUNTER
        POKE PORTB2,RCM+64       REM SELECT MONTH COUNTER
                                 REM SEND CS,WR LOW
        GOSUB 7210               REM WRITE PROTOCOL
        POKE PORTA2,YEAR.LOW     REM TENTHS YEAR OUT TO RAM
        POKE PORTB2,RRTS+64      REM SELECT TENTHS YEAR RAM
                                 REM SEND CS,WR LOW
        GOSUB 7210               REM WRITE PROTOCOL
        GOSUB 8000               REM READ DATE
        DW=DAY.OF.WEEK
        READ #1,1962+DW;DDAY$
```

Figure C.2 PROJ.BAS (continued.)

```
        READ #1,1969+MONTH;SMONTH$
        PRINT "THE DATE IS ";DDAY$;",";\
        SMONTH$;" ";DAY;",";YEAR
        quit.flag=0
8210    RETURN
        REM
        REM
        REM
9000    REM ORTEC COUNTER OPERATIONS
        REM THIS ROUTINE DETERMINES THE MASK BIT SEQUENCE
        REM FOR COUNTERS TO BE MASKED
        FOR IG=0 TO GN-1
        PRINT #1,1932+IGROUP;0          REM MASK
        PRINT #1,1925+IGROUP;0          REM COUNTERS
        NEXT IG
        FOR ICOUNT=0 TO 3
        READ #1,1950+3*ICOUNT+IGROUP;CC
        IF CC=1 THEN GOTO 9005 ELSE GOTO 9007
9005    READ #1,1932+IGROUP;MASK
        READ #1,1925+IGROUP;COUNTERS
        READ #1,1928+ICOUNT;GATV
        MASK=MASK+GATV
        COUNTERS=COUNTERS+1
        PRINT #1,1932+IGROUP;MASK
        PRINT #1,1925+IGROUP;COUNTERS
        NTOP=NTOP+1
9007    NEXT ICOUNT
        READ #1,1932+IGROUP;MASK
        IF MASK <= 9 THEN MASK.BIT$=CHR$(MASK+48)
        IF MASK > 9 THEN MASK.BIT$=CHR$(MASK+55)
        PRINT #1,1919+IGROUP;MASK.BIT$
        RETURN
        REM
        REM
        REM
9020    REM SUBROUTINE "PC" LOAD COUNTER
        PRINT #1,1910+IGROUP;9
        PRINT #1,1907+IGROUP;7
        GOSUB 9030
        RETURN
        REM
        REM
        REM
9030    REM SUBROUTINE LOAD COUNTER
        REM LOADS ALL BIT SEQUENCES REQUIRED FOR COUNTER
        REM OPERATION.ALL DATA INPUTS ARE TERMINATED BY
```

Figure C.2 PROJ.BAS (continued.)

```
        REM A CARRIAGE RETURN (CR) AND LINEFEED (LF)
        FOR J=1 TO 2
        GOSUB 9210       REM SEND 'L'
        POKE TRANS,ASC('M')
        X=CALL(TUART)
        READ #1,1910+IGROUP;M
        POKE TRANS,ASC(CHR$(M+48))        REM FORMULA
        X=CALL(TUART)
        GOSUB 9160       REM SEND CR,LF
        NEXT J
        GOSUB 9210       REM SEND 'L'
        POKE TRANS,ASC('N')
        X=CALL(TUART)
        READ #1,1907+IGROUP;N
        POKE TRANS,ASC(CHR$(N+48))        REM  FORMULA
        X=CALL(TUART)
        GOSUB 9160       REM SEND CR,LF
        REM SELECT COUNTER WHOSE DATA IS TO BE DISPLAYED
        GOSUB 9230       REM SEND 'LD'
        READ #1,1922+IGROUP;DP
        POKE TRANS,ASC(CHR$(DP+48))      REM DISPLAY COUNTER DATA
        X=CALL(TUART)
        GOSUB 9160       REM SEND CR,LF
        GOSUB 9040       REM RESET R.MASK
        GOTO 9060
        REM
        REM
        REM
9040    REM SUBROUTINE RESET R.MASK
        GOSUB 9210       REM SEND 'L'
        GOSUB 9245       REM SEND 'R'
        READ #1,1919+IGROUP;MASK.BIT$
        POKE TRANS,ASC(MASK.BIT$)         REM WITH BIT
                         REM SEQUENCES FOR COUNTERS TO BE MASKED
        X=CALL(TUART)
        GOSUB 9160       REM SEND CR,LF
        RETURN
        REM
        REM
        REM
9050    REM SUBROUTINE SET R.MASK
        GOSUB 9210       REM SEND 'L'
        GOSUB 9245       REM SEND 'R'
        POKE TRANS,ASC(CHR$(48))          REM MASK ALL COUNTERS
        X=CALL(TUART)
        GOSUB 9160       REM SEND CR,LF
```

Figure C.2 PROJ.BAS (continued.)

```
         RETURN
         REM
         REM
         REM
 9060    REM CONTINUE
         GOSUB 9210       REM SEND "L"
         GOSUB 9215       REM SEND "G"
                          REM LOAD THE GATE MASK REGISTER
                                  REM WITH BIT SEQUENCES FOR
                                  REM COUNTERS TO BE MASKED.
         READ #1,1919+IGROUP;MASK.BIT$
         POKE TRANS,ASC(MASK.BIT$)
         X=CALL(TUART)
         GOSUB 9160       REM SEND CR,LF
         GOSUB 9210       REM SEND "L"
         POKE TRANS,ASC("T")      REM SELECT TIME BASE 0=.1SEC
         X=CALL(TUART)
         REM 1 = MINUTES 2 = EXTERNAL
         READ #1,1901+IGROUP;time.base$
         IF time.base$="S" THEN GOTO 9120 ELSE GOTO 9130
 9120    POKE TRANS,ASC(CHR$(48))
         GOTO 9153
 9130    IF time.base$="M" THEN GOTO 9140 ELSE GOTO 9150
 9140    POKE TRANS,ASC(CHR$(49))
         GOTO 9153
 9150    POKE TRANS,ASC(CHR$(50))
 9153    X=CALL(TUART)
         GOSUB 9160       REM SEND LF,CR EXECUTE COMMANDS
 9155    REM CONTINUE
         RETURN
         REM
         REM
         REM
 9160    REM SUBROUTINE EXECUTE COMMANDS
         POKE TRANS,ASC(CR$)
         X=CALL(TUART)
         POKE TRANS,ASC(LF$)      REM SEND CARRIAGE RETURN,LINE
                                  REM FEED PAIR AND EXECUTE COMMANDS
         X=CALL(TUART)
         RETURN
         REM
         REM
         REM
 9170    REM SUBROUTINE "PC" TIMEOUT
         REM READ ALL COUNTERS NOT MASKED
         GOSUB 9290               REM READ ALL COUNTERS
```

Figure C.2 PROJ.BAS (continued.)

```
         FOR ICOUNT=0 TO 3
         READ #1,1950+3*ICOUNT+IGROUP;CC
         READ #1,1928+ICOUNT;GATV
         IF CC=1 THEN GOTO 9180 ELSE GOTO 9190
9180     OFGATE=GATV
         IF SFLAG=0 THEN JSI=JINDEX+1 ELSE JSI=JSINGLE
         GOSUB 9360                REM CHECK FOR COUNTER OVERFLOW
         X=300*ICOUNT+100*IGROUP+JSI
         READ #1,X;COUNTER.VALUES
         READ #1,1938+4*IGROUP+ICOUNT;PCNT
         IF COUNTER.VALUES >= PCNT THEN \
          GOTO 9185 ELSE GOTO 9190
9185      GOSUB 9350               REM GATE-OFF COUNTER
         GOSUB 7000                REM READ STOP TIME
         SEC=HOURS*3600+MINUTES*60+SECONDS
         PRINT #1,1301+200*IGROUP+ICOUNT;SEC
         NSTOP=NSTOP+1
         IF NSTOP=NTOP THEN GOTO 9200 ELSE GOTO 9170
9190     NEXT ICOUNT
9200     RETURN
         REM
         REM
         REM
9210     REM SUBROUTINE SEND 'L'
         POKE TRANS,ASC('L')
         X=CALL(TUART)
         RETURN
         REM
         REM
         REM
9215     REM SUBROUTINE SEND 'G'
         POKE TRANS,ASC('G')
         X=CALL(TUART)
         RETURN
         REM
         REM
         REM
9220     REM SUBROUTINE SEND 'H'
         POKE TRANS,ASC('H')
         X=CALL(TUART)
         RETURN
         REM
         REM
         REM
9230     REM SUBROUTINE SEND LD
         POKE TRANS,ASC('L')
```

Figure C.2 PROJ.BAS (continued.)

```
        X=CALL(TUART)
        POKE TRANS,ASC('D')
        X=CALL(TUART)
        RETURN
        REM
        REM
        REM
9240    REM SUBROUTINE SEND 'C'
        POKE TRANS,ASC('C')
        X=CALL(TUART)
        RETURN
        REM
        REM
        REM
9245    REM SUBROUTINE SEND 'R'
        POKE TRANS,ASC('R')
        X=CALL(TUART)
        RETURN
        REM
        REM
        REM
9250    REM SUBROUTINE START COUNTER
        GOSUB 9240      REM SEND 'C'
        GOSUB 9220                REM SEND 'H' COUNTER HALT
        GOSUB 9160      REM SEND CR,LF
        POKE IFR,8      REM CLEAR CB2 FLAG
        GOSUB 9040      REM RESET R.MASK
        GOSUB 9240      REM SEND 'C'
        GOSUB 9240      REM SEND 'C'
        GOSUB 9160      REM SEND CR,LF
        GOSUB 9050      REM SET R.MASK
        GOSUB 9240      REM SEND 'C'
        POKE TRANS,ASC('S')       REM SEND 'CS' COUNTER START
        X=CALL(TUART)
        GOSUB 9160      REM SEND LF,CR EXECUTE COMMANDS
        RETURN
        REM
        REM
        REM
9260    REM SUBROUTINE COUNTER TIME-OUT
        REM LOOP HERE UNTIL COUNTING IS STOPPED
9270    data=PEEK(IFR)
        INTERVAL.CB2=data AND 8
        GOSUB 9360                        REM OVERFLOW DETECTION
        IF  INTERVAL.CB2 =0 THEN  9270    REM LOOP HERE UNTIL
                                          REM COUNTING ENDS
```

Figure C.2 PROJ.BAS (continued.)

```
        GOSUB 9240          REM SEND 'C'
        GOSUB 9220          REM SEND 'H'
        GOSUB 9160          REM SEND CR,LF
        POKE IFR,8          REM CLEAR CB2 FLAG BY WRITING LOGIC
                            REM 1 TO CB2 BIT IN THE IFR
        RETURN
        REM
        REM
        REM
9280    REM SUBROUTINE READ COUNTER
        REM READ 8 DECADES OF COUNTER VALUES
        CVALUE=0
        X=RBUFF+2+8*(K-1)
        FOR I=0 TO 7
        XX=X+I
        data=PEEK(XX)                          REM FETCH ASCII DATA
        data=VAL(CHR$(data))    REM CONVERT TO DECIMAL
        CVALUE=CVALUE+data*(10^(7-I))
        NEXT I
        RETURN
        REM
        REM
        REM
9290    REM SUBROUTINE READ ALL COUNTERS
        GOSUB 9050          REM MASK ALL COUNTERS TO PREVENT RESET
        READ #1,1925+IGROUP;COUNTERS
        POKE RBUFF,4
        GOSUB 9240          REM SEND 'C'
        GOSUB 9245          REM SEND 'R' : 'CR'
        GOSUB 9160          REM SEND CR,LF EXECUTE COMMAND
        XX=CALL(RCOUNT)
        FOR K=1 TO 4
        GOSUB 9280          REM FETCH COUNTER DATA
        READ #1,1950+3*ICOUNT+IGROUP;CC
        ICOUNT=K-1
        IF SFLAG=0 THEN GOTO 9294 ELSE GOTO 9297
9294    X1=300*ICOUNT+100*IGROUP+JINDEX+1
        GOTO 9298
9297    X1=300*ICOUNT+100*IGROUP+JSINGLE
9298    REM CONTINUE
        IF CC=1 THEN GOTO 9300 ELSE GOTO 9320
9300    PRINT #1,X1;CVALUE
9320    REM CONTINUE
9340    NEXT K
        RETURN
        REM
```

Figure C.2 PROJ.BAS (continued.)

```
        REM
        REM
9350    REM SUBROUTINE GATEOFF
        REM INHIBITS COUNTING IN A COUNTER SELECTED BY
        REM THE GATE OFF PARAMETER OFGATE
        READ #1,1932+IGROUP;MASK
        MASKG=MASK-OFGATE
        IF MASKG < 9 THEN GATEM$=CHR$(MASKG+48)
        IF MASKG > 9 THEN GATEM$=CHR$(MASKG+55)
        GOSUB 9210              REM SEND 'L'
        GOSUB 9215              REM SEBD 'G'
        POKE TRANS,ASC(GATEM$)
        X=CALL(TUART)
        GOSUB 9160              REM EXECUTE 'LG' COMMAND
        RETURN
        REM
        REM
        REM
9360    REM SUBROUTINE OVERFLOW
        REM DETECTS OVERFLOW FLAGS FROM ORTEC COUNTERS 2,3 & 4
        REM AND UPDATES THE COUNTER STORAGE RESPECTIVELY
        REM OVFL
        PRINT #1,1935;16
        PRINT #1,1936;8
        PRINT #1,1937;2
        REM SET MASKS FOR OVFL2.CB1,OVFL3.CB2,& OVFL4.CA1 RESPECTIVELY
        FOR JCONT=1 TO 3
        READ #1,1950+3*JCONT+IGROUP;CC
        IF CC=1 THEN GOTO 9370 ELSE GOTO 9380
9370    IF SFLAG=0 THEN JSI=JINDEX ELSE JSI=JSINGLE
        OVF=PEEK(IFR3)          REM FETCH STATE
        READ #1,1935+JCONT-1;OVFL
        OVF=OVF AND OVFL        REM MASK FOR THIS OVERFLOW
        IF OVF=1 THEN GOTO 9375 ELSE GOTO 9380
9375    X=300*JCONT+100*IGROUP+JSI
        READ #1,X;COUNTER.VALUES
        COUNTER.VALUES=COUNTER.VALUES+99999999
        PRINT #1,X;COUNTER.VALUES
        READ #1,1935+JCONT-1;OVFL
        POKE IFR3,OVFL  REM CLEAR OVERFLOW FLAG
9380    REM CONTINUE
        NEXT JCONT
        RETURN
        REM
        REM
        REM
```

Figure C.2 PROJ.BAS (continued.)

```
9400    REM SUBROUTINE ORTEC 874 DEFAULT STATE
        GOSUB 9240      REM SEND 'C'
        GOSUB 9220      REM SEND 'H' :COUNTER HALT
        GOSUB 9160      REM SEND CR,LF
        GOSUB 9040      REM RESET R.MASK
        GOSUB 9210      REM SEND 'L'
        GOSUB 9245      REM SEND 'R'
        POKE TRANS,ASC('F')
        X=CALL(TUART)
        GOSUB 9160              REM SEND CR,LF
        GOSUB 9240              REM SEND 'C'
        GOSUB 9240              REM SEND 'C' : 'CC'=COUNTER CLEAR
        GOSUB 9160              REM SEND CR,LF
        FOR I=1 TO 2
        GOSUB 9210              REM SEND 'L'
        POKE TRANS,ASC('M')
        X=CALL(TUART)
        POKE TRANS,ASC(CHR$(48))
        X=CALL(TUART)
        GOSUB 9160              REM SEND CR,LF
        NEXT I
        GOSUB 9210              REM SEND 'L'
        POKE TRANS,ASC('N')
        X=CALL(TUART)
        POKE TRANS,ASC(CHR$(48))
        X=CALL(TUART)
        GOSUB 9160              REM SEND CR,LF
        GOSUB 9230              REM SEND 'LD'
        POKE TRANS,ASC(CHR$(49))
        X=CALL(TUART)
        GOSUB 9160              REM SEND CR,LF
        GOSUB 9210              REM SEND 'L'
        POKE TRANS,ASC('T')
        X=CALL(TUART)
        POKE TRANS,ASC(CHR$(48))
        X=CALL(TUART)
        GOSUB 9160              REM SEND CR,LF
        RETURN
        REM
        REM
        REM
10000   REM CONTINUE
        PRINT #1,1981;SFLAG
        PRINT #1,1982;LINE
        REM PERFORM POWER-DOWN STORAGE
        data=PEEK(CVIALP)       REM FETCH VIAL POSITION
```

Figure C.2 PROJ.BAS (continued.)

```
PHIGH=INT(data/10)
PLOW=data-PHIGH*10
POKE DDRA2,255          REM SETTUP PORTA2 AS OUTPUT
HEX.NUM=state.vial*10 + PLOW
GOSUB 5660              REM ENCODE IN DECIMAL
POKE PORTA2,DECIMAL.NUM
POKE PORTB2,RRDOM+64    REM SELECT STORAGE RAM
                        REM SEND CS,WR LOW
GOSUB 7210              REM WRITE PROTOCOL
HEX.NUM=PHIGH*10
GOSUB 5660              REM DECIMAL ENCODE
POKE PORTA2,DECIMAL.NUM
POKE PORTB2,RRTHS+64    REM SELECT STORAGE RAM
                        REM SEND CS,WR LOW
GOSUB 7210              REM WRITE PROTOCOL
REM
POKE PORTA,224          REM TURN OFF ALL LEDS
POKE PORTB,255          REM AND MOTORS
POKE PCR2,29            REM $1D DISABLE MOTOR DRIVES
CLOSE 1
PRINT " BYEBYE "
STOP
END

A>
```

Figure C.2 PROJ.BAS (continued.)

```
  TYPE PROJ2.BAS
        REMARK THIS PROGRAM IS CALLED PROJ2.BAS
        REM VERSION I.1 1/16/87
        REM
        CR$     =CHR$(13)         REM CARRIAGE RETURN
        LF$     =CHR$(10)         REM LINE FEED
        GOTO 2070
        REM
        REM
505     REM SUBROUTINE SETTUP
        LINE=4 : PAGE=0
        BEEPA=53353     REM $D069  DOES A BEEP
510     PRINT 'INPUT PROJECT NAME < 70 CHAR' : GOSUB 3416
        INPUT PNAME$ : GOSUB 3416
        IF LEN(PNAME$) > 70 THEN GOTO 520 ELSE GOTO 530
520     X=CALL(BEEPA)             REM BEEP FOR ILLEGAL INPUT
        GOTO 510
530     REM CONTINUE
        RETURN
        REM
        REM
        REM
        REM
        REM
2000    REM SUBROUTINE COMPUTE DELTA T
        READ #1,1910+J;M
        READ #1,1907+J;N
        N=N-1
        READ #1,1904;preset$
        READ #1,1901+J;time.base$
        IF preset$='PT' THEN GOTO 2010 ELSE GOTO 2020
2010    REM PRESET TIME ANALYSIS
        GOSUB 2060                REM COMPUTE TIME
        DELTA.TIME=M * 10 ^ ( N )
        IF time.base$='M' THEN DELTA.TIME=DELTA.TIME*60
        GOTO 2030
2020    REM PRESET COUNT ANALYSIS
        GOSUB 2060                REM COMPUTE TIME
2030    RETURN
        REM
        REM
        REM
2040    REM SUBROUTINE COMPUTE COUNTS PER SECOND
        GOSUB 2000                REM COMPUTE DELTA T
        REM SCRATCH PAD FILE VECTOR
```

Figure C.3 PROJ2.BAS listing

```
        X=100*J+KI
        READ #1,1+X;COUNTER.VALUES
        CPS0=COUNTER.VALUES/DELTA.TIME
        READ #1,301+X;COUNTER.VALUES
        CPS1=COUNTER.VALUES/DELTA.TIME
        READ #1,601+X;COUNTER.VALUES
        CPS2=COUNTER.VALUES/DELTA.TIME
        READ #1,901+X;COUNTER.VALUES
        CPS3=COUNTER.VALUES/DELTA.TIME
        RETURN
        REM
        REM
        REM
2050    REM SUBROUTINE COMPUTE TIME
        READ #1,XA;SEC
        HR=INT( SEC/3600 )
        MIN1=SEC-HR*3600
        MIN=INT ( MIN1 /60 )
        SECS=MIN1-MIN*60
        RETURN
        REM
        REM
        REM
2060    REM SUBROUTINE DEL.T
        GOSUB 2050
        SEC1=SEC
        XA=XA+100
        GOSUB 2050
        DELTA.TIME=SEC-SEC1
        RETURN
        REM
        REM
        REM
2070    REM MAIN PROGRAM STARTS HERE
        B$= 'SCRATCH.BAS'
        FILE B$(20)
        READ #1,1981;SFLAG
        READ #1,1983;GN
        IF SFLAG=1 THEN READ #1,1913;JSINGLE
        GOSUB 505               REM SETTUP I/O PARAMETERS
        REM STRING NAME DEFINITIONS
        CC$     ='CHANNEL COUNTS'
        CPS$    ='COUNTS PER SECOND'
        GS$     ='GP-POS#'
        IF SFLAG=1 THEN GS$='SG-POS#'
        ST$     ='START TIME'
```

Figure C.3 PROJ2.BAS (continued.)

```
        DT$      ="DELTA T,SEC"
        DASH$    ="----------------------------------------"
        DASH$    =DASH$+DASH$
3215    REM PRINTER OUTPUT
        FOR J=1 TO 5
        PRINT LF$
        NEXT J : PAGE=1          REM PERFORM FORM FEED
        LINE=LINE+5
        PRINT PNAME$;"   ";"PAGE";" ";PAGE : GOSUB 3416
        FOR J=1 TO 3
        PRINT LF$ : GOSUB 3416  REM SKIP 3 LINES
        NEXT J
        PRINT TAB(48);CC$ : GOSUB 3416
        PRINT TAB(2);GS$;TAB(12);ST$;TAB(24);DT$;TAB(40);\
                "CH1";TAB(51);"CH2";TAB(62);"CH3";TAB(73);\
                "CH4" : GOSUB 3416
        PRINT DASH$ : GOSUB 3416
        FOR J=0 TO GN-1
        IF SFLAG=0 THEN GOTO 3360 ELSE GOTO 3370
3360    REM SAMPLE.IN.GROUP
        READ #1,1913+J;IMAX
        REM BEGIN.GROUP.SAMPLE
        READ #1,1916+J;K
        JP1=J+1
        GOTO 3380
3370    IMAX=JSINGLE
        K=0 : JP1=0
3380    REM CONTINUE
        FOR I=0 TO IMAX-1
        KPI=K+I : KI=KPI
        IF KPI=100 THEN KPI=0
        IF SFLAG=1 THEN READ #1,1801+KPI;SINGLE.POSITION
        IF SFLAG=0 THEN KI=I
        X=100*J+KI
        READ #1,1+X;X0
        READ #1,301+X;X1
        READ #1,601+X;X2
        READ #1,901+X;X3
        IF SFLAG=1 THEN KPI=SINGLE.POSITION
        XA=1201+200*J+KI         REM START TIME VECTOR
        GOSUB 2000               REM COMPUTE DELTA T
        XA=XA-100
        GOSUB 2050
        TIME$=STR$(HR)+":"+STR$(MIN)+":"+STR$(SECS)
        GS1$=STR$(JP1)+" - "+STR$(KPI)
        PRINT TAB(2);GS1$;TAB(12);\
```

Figure C.3 PROJ2.BAS (continued.)

```
        TIME$;TAB(24);DELTA.TIME;TAB(36);X0;\
        TAB(47);X1;TAB(58);\
        X2;TAB(69);\
        X3 : GOSUB 3416
        GOTO 3517
        REM
        REM
        REM
3416    REM SUBROUTINE PAGING
        LINE=LINE+1
        IF LINE >= 63 THEN GOTO 3430 ELSE GOTO 3440
3430    REM CONTINUE
        FOR JJ=1 TO 4
        PRINT LF$
        NEXT JJ
        LINE=1 : PAGE=PAGE+1
        PRINT FNAME$;" ";"PAGE";" ";PAGE
        FOR JJ=1 TO 3
        PRINT LF$ : LINE=LINE+1
        NEXT JJ
3440    REM CONTINUE
        RETURN
        REM
        REM
        REM
3517    REM CONTINUE
        NEXT I
        NEXT J
        PRINT LF$ : GOSUB 3416
        PRINT TAB(46);CPS$ : GOSUB 3416
        PRINT TAB(2);GS$;TAB(40);"CH1";TAB(51);"CH2";\
              TAB(62);"CH3";TAB(73);"CH4" : GOSUB 3416
        PRINT DASH$ : GOSUB 3416
        FOR J=0 TO GN-1
        IF SFLAG=0 THEN GOTO 3690 ELSE GOTO 3700
3690    REM SAMPLE.IN.GROUP
        READ #1,1913+J;IMAX
        REM BEGIN.GROUP.SAMPLE
        READ #1,1916+J;K
        JP1=J+1 : GOTO 3710
3700    IMAX=JSINGLE
        K=0 : JP1=0
3710    REM CONTINUE
        FOR I=0 TO IMAX-1
        KPI=K+I : KI=KPI
        IF KPI=100 THEN KPI=0
```

Figure C.3 PROJ2.BAS (continued.)

```
        IF SFLAG=1 THEN READ #1,1801+KPI;SINGLE.POSITION
        IF SFLAG=1 THEN KPI=SINGLE.POSITION
        IF SFLAG=0 THEN KI=I
        XA=1201+200*J+KI          REM START TIME VECTOR
        GOSUB 2040                REM COMPUTE COUNTS PER SECONDS
        GS1$=STR$(JP1)+" - "+STR$(KPI)
        PRINT TAB(2);GS1$;TAB(36);\
              CPS0;TAB(47);CPS1;TAB(58);CPS2;TAB(69);CPS3
        GOSUB 3416
        NEXT I
        NEXT J
        PRINT LF$ : GOSUB 3416
        J=GN-1 : I=IMAX-1 : KI=K+I
        IF SFLAG=0 THEN KI=I
        XA=1301+200*J+KI          REM STOP TIME VECTOR
        GOSUB 2050                REM READ TIME
        HOURS=HR : MINUTES=MIN : SECONDS=SECS
        PRINT HOURS;":";MINUTES;":";SECONDS : GOSUB 3416
        FOR JJ=1 TO 5
        PRINT LF$
        NEXT JJ
4000    REM CONTINUE
        CLOSE 1
        PRINT "BYEBYE"
        STOP
        END

A>
```

Figure C.3 PROJ2.BAS (continued.)