APPENDIX A

AIM 65 COMMAND DEFINITIONS

[ADDRESS] Hexadecimal address, one to four characters

[BYTE] Two-digit hexadecimal value from 00 to FF.

[DECIMAL NUMBER] A two-digit decimal number in the range 00 to 99.

[FILE NAME] A string of 1 to 5 characters.

[HEX OPERAND] The instruction operand.

Addressing Mode	Operand Format
Immediate	#HH
Zero Page	HH
Zero Page, X	HH, X or HHX
Zero Page, Y	HH, Y or HHY
Absolute	HHHH
Absolute, X	HHHH, X or HHHHX
Absolute, Y	HHHH, Y or HHHHY
Relative	HH or HHHH
(Indirect, X)	(HH,X) or (HHX) or (HH,X or (HHX
(Indirect), Y	(HH,Y or (HH,Y)
(Absolute Indirect)	(HHHH)

[INPUT DEVICE] RETURN or SPACE — AIM 65 Keyboard (S2 = KB) or TTY Keyboard (S2 = TTY)

- M — Memory
- T — Audio Tape, AIM 65 format
- K — Audio Tape, KIM-1 format
- L — TTY Paper Tape Reader
- U — User-defined input device

[MNEMONIC OPCODE] A three-letter mnemonic abbreviation.

[OUTPUT DEVICE] RETURN or SPACE — AIM 65 Display/Printer (S2 = KB) or TTY Printer (S2 = TTY)

- P — AIM 65 Printer
- X — Dummy
- T — Audio Tape, AIM 65 format
- K — Audio Tape, KIM-1 format
- L — TTY Paper Tape Punch
- U — User-defined output device

APPENDIX B

AIM 65 MONITOR COMMAND SUMMARY

MAJOR FUNCTION ENTRY COMMANDS

[RESET] — Enter and Initialize Monitor
ROCKWELL AIM 65

E — Enter and Initialize Editor
<E>

T — Re-enter Text Editor at Top of Text
<T>
TOP LINE OF TEXT

N — Enter Assembler
<N>

5 — Enter and Initialize BASIC Interpreter
<5>

6 — Re-enter BASIC Interpreter
<6>

INSTRUCTION ENTRY AND DISASSEMBLY COMMANDS

I — Enter Mnemonic Instruction Entry Mode
<I>
AAAA[*] = [ADDRESS]
AAAA XX [OPCODE][HEX OPERAND]
AAAA XX XX XX

K — Disassemble Memory
<K> * = [ADDRESS]
/[DECIMAL NUMBER]
AAAA XX OPCODE HEX OPERAND

DISPLAY/ALTER REGISTER COMMANDS

* — Alter Program Counter
<*> = [ADDRESS]

A — Alter Accumulator
<A> = [BYTE]

X — Alter X Register
<X> = [BYTE]

Y — Alter Y Register
<Y> = [BYTE]

P — Alter Processor Status
<P> = [BYTE]

S — Alter Stack Pointer
<S> = [BYTE]

R — Display Register Values
<R>

```
 ****  PS AA XX YY SS
 0200  00 00 01 02 FF
```

DISPLAY/ALTER MEMORY CONTENTS

M — Display Specified Memory Locations
<M> = [ADDRESS] XX XX XX XX

SPACE — Display Next 4 Memory Locations
< > AAAA XX XX XX XX

/ — Alter Current Memory Locations
</> AAAA XX XX XX XX

LOAD/DUMP MEMORY COMMANDS

L — Load Object Code into Memory
<L> IN = [INPUT DEVICE]

D — Dump Memory
<D>
FROM = [ADDRESS] TO = [ADDRESS]
OUT = [OUTPUT DEVICE]
MORE? [Y, N]

BREAKPOINT MANIPULATION COMMANDS

— Clear All Breakpoints
<#> OFF

4 — Toggle Breakpoint Enable
<4> OFF/ON

B — Set/Clear Breakpoint Address
<B> BRK/[0, 1, 2, 3] = [ADDRESS]

? — Display Breakpoint Addresses
<?>
AAAA AAAA AAAA AAAA

EXECUTION/TRACE CONTROL COMMANDS

G — Start Execution of User's Program
<G>/[DECIMAL NUMBER]

Z — Toggle Instruction Trace Mode
<Z> ON/OFF

V — Toggle Register Trace Mode
<V> ON/OFF

H — Trace Program Counter History
<H>
AAAA
⋮
AAAA

CONTROL PERIPHERAL DEVICES

CTRL PRINT — Toggle Printer On/Off
<CTRL><PRINT>

PRINT — Print Display Contents
<PRINT>

LF — Advance Printer Paper
<LF>

1 — Toggle Tape 1 Control On/Off
<1>

2 — Toggle Tape 2 Control On/Off
<2>

3 — Tape Verify Block Checksum
<3> IN = [T] F = [FILE NAME] T = [1, 2]

USER FUNCTION COMMANDS

F1 — Call User Function 1
<F1>

F2 — Call User Function 2
<F2>

F3 — Call User Function 3
<F3>

APPENDIX C

AIM 65 TEXT EDITOR COMMAND SUMMARY

ENTER AND EXIT EDITOR COMMANDS

E — Enter and Initialize Editor
<E>
EDITOR
FROM = [ADDRESS] TO = [ADDRESS]
IN = [INPUT DEVICE]
Note: Defaults are TO = $0200,
FROM = Last contiguous RAM, IN = Keyboard

Q — Exit the Text Editor and Return to Monitor
= <Q>

LINE ORIENTED COMMANDS

R — Read Lines into Text Buffer from Input Device
= <R>
IN = [INPUT DEVICE]

I — Insert One Line of Text Ahead of Active Line
= <I>
INSERTED TEXT LINE
ACTIVE LINE OF TEXT

K — Delete Current Line of Text
= <K>
DELETED LINE OF TEXT
ACTIVE LINE OF TEXT

U — Move the Text Pointer Up One Line
= <U>
PRIOR LINE OF TEXT

D — Move the Text Pointer Down One Line
= <D>
NEXT LINE OF TEXT

T — Move the Text Pointer to the Top of the Text
= <T>
TOP LINE OF TEXT

B — Move the Text Pointer to the Bottom of the Text
= <B>
BOTTOM LINE OF TEXT

L — List Lines of Text to Output Device
= <L>
/[DECIMAL NUMBER]

SPACE — Display the Active Line
= < >
ACTIVE LINE OF TEXT

STRING ORIENTED COMMANDS

F — Find a Character String
= <F>
[CHARACTER STRING]
LINE CONTAINING CHARACTER STRING

C — Change a Character String
= <C>
[OLD STRING]
LINE CONTAINING OLD STRING
TO = [NEW STRING]
SAME LINE, WITH NEW STRING

APPENDIX D

AIM 65 ASSEMBLER COMMAND SUMMARY

D.1 ASSEMBLER COMMAND SEQUENCE

```
<N>
ASSEMBLER
FROM = [ADDRESS] TO = [ADDRESS]
IN = [INPUT DEVICE]
LIST? [Y, N]
LIST-OUT = [OUTPUT DEVICE]
OBJ? [Y, N]  Note: N = Object code to Memory
OBJ-OUT = [OUTPUT DEVICE]  Note: Prompts only on Y response to OBJ?
PASS 1
 SYM TBL OVERFLOW  } Displayed only if Symbol Table overflows
 ASSEMBLER         }
PASS 2
= = AAAA LABEL                                  }
OBJECT CODE   MNEMONIC OPCODE                   } Displayed only if
SYMBOLIC OPERAND   ;COMMENT                     } LIST?Y, or LIST?N
                                                  and error detected
* * ERROR NN  Note: Error code displayed only on error
ERRORS =  MMMM  Decimal count of errors detected
```

D.2 ASSEMBLER EXPRESSIONS

ELEMENTS

Numeric constants — may be written in one of four bases.

Prefix Character	Base
(none)	10 (Decimal)
$	16 (Hexadecimal)
@	8 (Octal)
%	2 (Binary)

OPERATORS

Type	Operator	Operation
Arithmetic	+	Addition
Arithmetic	–	Subtraction
Special	>	High-Byte Selection
Special	<	Low-Byte Selection

Operators < and > truncate a two-byte value to its high or low byte, respectively.

D.3 ASSEMBLER DIRECTIVES

= — Assigns the value of an operand containing no forward references to either a symbol or the location counter.

{SYMBOL / *} = Operand

.BYTE — Assigns multiple ASCII strings or expressions to consecutive single byte memory locations in high-byte, low-byte order.

.BYT Expression, Expression, . . . Expression

.WORD — Assigns multiple expression operands to consecutive memory locations in low-byte, high-byte order.

.WOR Expression, Expression, . . . Expression

.DBYTE — Assigns multiple expression operands to consecutive double byte (16 bits) memory locations.

.DBY Expression, Expression, . . . Expression

.PAGE — Generates a title under a dashed line.

.PAG {'NEW TITLE' / BLANK / ' '} (New Title) (No Change of Title) (Blanks Title)

.SKIP — Generates one blank line.

.SKI

.OPT — Controls assembly listings. All are optional and can be specified in any order or in separate statements.

.OPT {LIS / NOL}, {GEN / NOG}, {ERR / NOE}

.FILE — Last record in a multiple file source program (except the last file) which points to the continuation file.

.FIL File Name

.END — Last record in a single or multiple source file.

.END

D.4 ASSEMBLER ERROR CODES

01	Undefined Symbol
02	Label Previously Defined or Forward Reference to Page 0 Symbol
03	Illegal or Missing Opcode
04	Address Not Valid
05	Accumulator Mode Not Allowed
06	Forward Reference to Page Zero
07	Ran off End of Line
08	Label Does Not Begin with Alphabetic Character
09	Label Greater Than Six Characters
10	Label or Opcode Contains Non-Alphanumeric
11	Forward Reference in Equate
12	Invalid Index — Must Be X or Y
13	Invalid Expression
14	Undefined Assembler Directive
17	Relative Branch Out of Range
18	Illegal Operand Type for This Instruction
19	Out of Bounds on Indirect Addressing
20	A, X, Y, S and P are Reserved Labels
21	Program Counter Negative — Reset to 0

APPENDIX E
ASCII CHARACTER SET

ASCII CHARACTER SET (7-BIT CODE)

LSD \ MSD		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P		p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	–	=	M	]	m	}
E	1110	SO	RS	.	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

NOTES

1. [labeled F1 on AIM 65 keyboard.
   ] labeled F2 on AIM 65 keyboard.
   ↑ labeled F3 on AIM 65 keyboard.

2. ↑ in ASCII appears a ∧ on AIM 65 display and printer.

NUL — Null
SOH — Start of Heading
STX — Start of Text
ETX — End of Text
EOT — End of Transmission
ENQ — Enquiry
ACK — Acknowledge
BEL — Bell
BS — Backspace
HT — Horizontal Tabulation
LF — Line Feed
VT — Vertical Tabulation
FF — Form Feed
CR — Carriage Return
SO — Shift Out
SI — Shift In

DLE — Data Link Escape
DC — Device Control
NAK — Negative Acknowledge
SYN — Synchronous Idle
ETB — End of Transmission Block
CAN — Cancel
EM — End of Medium
SUB — Substitute
ESC — Escape
FS — File Separator
GS — Group Separator
RS — Record Separator
US — Unit Separator
SP — Space (Blank)
DEL — Delete

APPENDIX F
AIM 65 AUDIO TAPE FORMAT

The AIM 65 audio cassette tape format is designed to provide fast, reliable recording and reading of both object code and source code. Object code is recorded in binary form, exactly as it appears in memory. Recording object code in binary, is twice as fast as recording it in ASCII, since one byte contains two hexadecimal numbers in binary format, whereas one byte contains only one hexadecimal number in ASCII format.

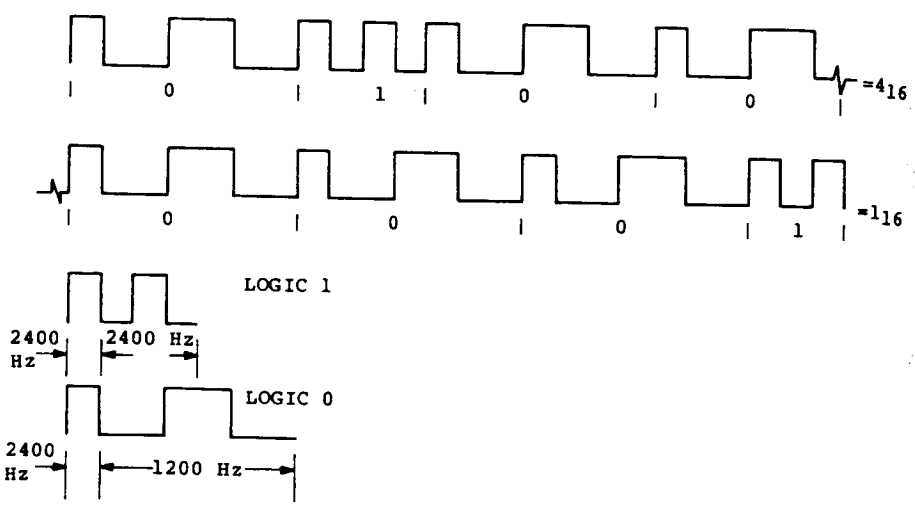
Source code is recorded in ASCII format, the form in which it appears in the Editor Text Buffer.

BIT LOGIC STATE DEFINITION

Each transmitted bit begins with a positive one-half cycle of 2400 Hz. tone. The following three half-cycles determine the logic state of the bit. Three half-cycles of 2400 Hz. equal a logic "1". Three half-cycles of 1200 Hz. tone equals a logic "0".

The following example shows eight bits of data. If this data represents one byte, the hexadecimal value of 41 equals two hexadecimal numbers, 4 and 1. If the data is in ASCII format, the character A is represented.

EXAMPLE:



F.1 BLOCKED FORMAT

The data is recorded in blocked format. One block contains 80 bytes of data and 36 bytes of synchronization, control, and block checksum information. The block format is:

SUBFIELD	SYN	#	BLK	OBJECT OR TEXT DATA	BLK CHKSUM
Value	1616...1616	23	HH	XXX...XXXX	HHHH
No. of bytes	(32)	(1)	(1)	(79)	(2)

SYN

The block begins with 32 bytes of Synchronous Idle (SYN) characters (ASCII 16). During read operations, the SYN pattern allows AIM 65 to sense the start of the block and synchronize to the incoming serial data stream.

SYN characters are also used to provide an interblock gap. The number of SYN characters is determined by the contents of address A409 (GAP). The default value of 08, established by a "cold" RESET, generates 32 SYN characters. This value provides a minimum gap for loading object code or source code into an empty Editor text buffer.

For assembling from tape or reading data into a partially-filled Editor text buffer,a larger gap size is required. This allows AIM 65 to stop the tape after a block has been read in order to process the data before reading another block. To lengthen the gap size additional SYN characters are required. The gap value in address A409 should be changed from 08 (32 SYN characters) to 80 (512 SYN characters). This number should be adequate for all audio cassette recorders, but a lower number may be suitable

for your recorder. That number can be determined by experimentation.

#

The character # (ASCII 23) denotes that the data on the tape is recorded in the AIM 65 format, as opposed to the KIM-1 format described in Appendix G.

BLK

The block count (BLK) defines the block number. This number starts with 00 on the first block and increments by one for each block recorded, in hexadecimal, to FF. If more than FF blocks are recorded, the number restarts at 00.

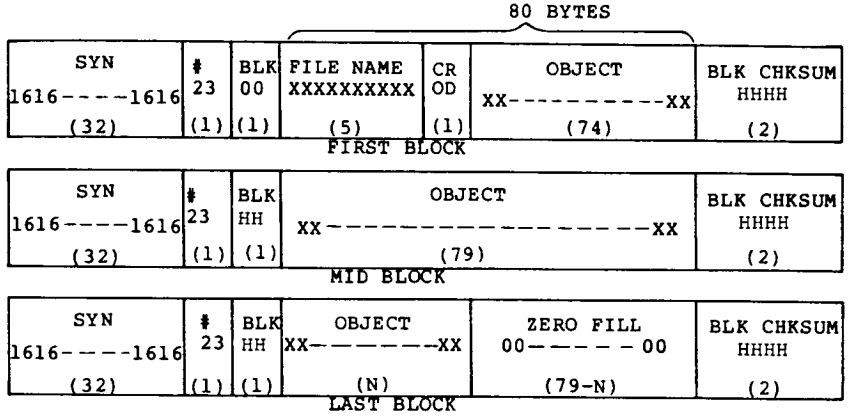
Data

The actual recorded data represents either source or object code. Within each data type are three unique data block types: First block, mid-blocks, and last block. See the object and text data record formats for detailed definition of the data format.

BLK CHKSUM

The block checksum (BLK CHKSUM) is the hexadecimal sum of the 80 data characters, truncated to four hexadecimal digits, (i.e., carry is ignored).

F.2 OBJECT DATA FILE FORMAT



FILE NAME

The file name (FILE NAME) consists of one to five ASCII characters that uniquely identify the file.

CR

The CR character (ASCII 0D) after the FILE NAME indicates that the data format is object rather than text.

Object Data Format

The object data consists of multiple object data records, each containing a starting address and up to 24 bytes of information. The object data is recorded in hexadecimal form. The object data includes both object instructions and data.

The object data record format is:

$$\text{Data Record: } ;N_1N_0A_3A_2A_1A_0\overbrace{D_1D_0}^{1}\overbrace{D_1D_0}^{2}\ldots\overbrace{D_1D_0}^{n}X_3X_2X_1X_0CR$$

$$\text{Last Record: } ;00C_3C_2C_1C_0X_3X_2X_1X_0CR$$

Where:

;	= Start of the record (ASCII 3B)
N_1N_0	= Number of data bytes in the record, in hexadecimal. The maximum number of bytes in one record is 18_{16} (24_{10}).
	= 00 for the last record.
$A_3A_2A_1A_0$	= Address of the first data byte in the record, in hexadecimal.
D_1D_0	= One 8-bit data byte = Two hexadecimal numbers.

$C_3C_2C_1C_0$	= Number of records in hexadecimal, including the data records and the last record.
$X_3X_2X_1X_0$	= Record checksum, in hexadecimal. This is the sum of all the characters in the object code record except the ; character and the record checksum. The checksum is truncated to four hexadecimal digits, i.e., carry is ignored.
CR	= Carriage Return (ASCII 0D) which indicates end of record.

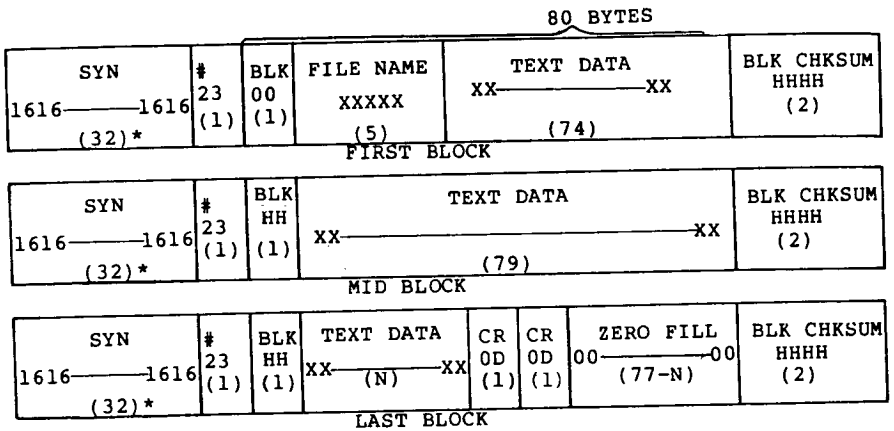
All files contain at least two object code records: the first record and the last record. The last record uniquely identifies the end of the file data.

Since each object code record contains a starting address, various portions of memory can be recorded in one file. Programs or program segments residing in different parts of memory may therefore be recorded on the same file. This simplifies subsequent memory loading procedures, as well as saving load setup time.

Zero Fill

After the last data record is recorded, the remaining data bytes are filled with hexadecimal zeros.

F.3 TEXT DATA FILE FORMAT



*The value shown corresponds to a gap size in $A409 (GAP) of $08.

FILE NAME

The file name (FILE NAME) consists of one to five ASCII characters that uniquely identify the file.

TEXT DATA

The text data consists of characters recorded from the Editor Text Buffer. The data is recorded in ASCII format as it exists in memory. The text data may be the source program for input into the assembler or any text information.

CR

The CR character (ASCII 0D) indicates end of a text record in the text buffer. CR will appear throughout the text buffer separated by no more than 60 characters. Two CR's in succession indicate end of the text file.

ZERO FILL

After the end-of-file indication, the remainder of the block is filled with hexadecimal zeros.

APPENDIX G
KIM-1 AUDIO TAPE FORMAT

Data is transmitted to the tape recorder in the form of serial ASCII encoded characters (seven data bits plus Parity bit). Object data retrieved from the memory is converted into this form by separating each byte into two half-bytes. The half bytes are then converted into their ASCII equivalents.

Each record transmitted begins with a leader of 100 "SYN" characters (ASCII 16) followed by a * character (ASCII 2A). During playback, this pattern allows AIM 65 to detect the start of a valid data record and synchronize to the serial data stream. Following the *, the record identification number (ID), and starting address low (SAL) and the starting address high (SAH) are transmitted. The data specified by the starting (SAL, SAH) and ending limits (EAL, EAH) is transmitted next followed by a "/" character (ASCII 2F) to indicate the end of the data portion of the record. Following the "/" two "CHECK-SUM" bytes are transmitted for comparison with a calculated checksum number during playback to further insure that a proper data retrieval has taken place. Two "EOT" characters (ASCII 04) mark the end of record transmission.

Each transmitted bit begins with a 3700 hertz tone and ends with a 2400 hertz tone. "Ones" have the high-to-low frequency transition at one-third of the bit period. "Zeros" have the transition at two-thirds of the period. During playback the phase locked loop function locks to, and tracks these two frequencies producing a logic "1" pulse of one-third

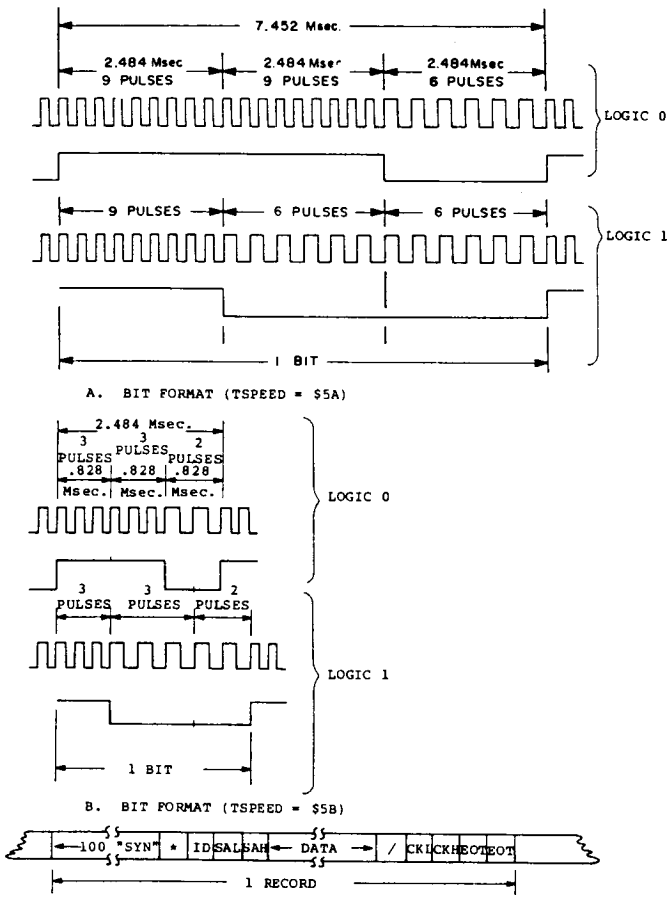


Figure G-1. KIM-1 Audio Tape Format

the bit period for a "One". A pulse two thirds the bit period is likewise produced for a "Zero". AIM 65 Monitor software converts two ASCII characters read from the tape into hexadecimal numbers and packs two numbers into one eight bit byte in memory.

APPENDIX H
PAPER TAPE FORMAT

The AIM 65 Load and Dump commands (Section 3.8) store and retrieve data in a format designed to ensure error free recovery. Each byte of data to be stored is converted to two half-bytes. The half-bytes (possible values are 0 to F) are translated into their ASCII equivalents and written out onto paper tape in this form.

Each output record begins with a ";" character (ASCII 3B) to mark the start of a valid record. The next byte transmitted (18_{16}) or (24_{10}) is the number of data bytes contained in the record. The record's starting address High (1 byte, 2 characters), starting address Low (1 byte, 2 characters), and data (24 bytes, 48 characters) follow. Each record is terminated by the record's checksum (2 bytes, 4 characters), a carriage return (ASCII 0D), line feed (ASCII 0A), and a "DEL" characters (ASCII FF).

The last record transmitted has zero data bytes (indicated by ;00). The starting address field is replaced by a four digit hexadecimal number representing the total number of data records contained in the transmission, followed by the records usual checksum digits. A "XOFF" character ends the transmission.

```
;180000FFEEDDCCBBAA00998877665544332211223344556677889 90AFC
;0000010001
```

APPENDIX I

R6500 INSTRUCTION SET

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			(IND, X)			(IND), Y			Z PAGE, X			ABS, X			ABS, Y			RELATIVE			INDIRECT			X PAGE, Y			PROCESSOR STATUS CODES			INSTRUCTIONS
MNEMONIC	OPERATION	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	7 6 5 4 3 2 1 0 N V - B D I Z C	MNEMONIC
ADC	A + M + C → A (1)(4)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										N V · · · · Z C	ADC
AND	A∧M → A (1)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										N · · · · · Z ·	AND
ASL	C ← [M] ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													N · · · · · Z C	ASL
BCC	BRANCH ON C = 0 (2)																															90	2	2									BCC
BCS	BRANCH ON C = 1 (2)																															B0	2	2									BCS
BEQ	BRANCH ON Z = 1 (2)																															F0	2	2									BEQ
BIT	A∧M				2C	4	3	24	3	2																															M7 M6 · · · Z ·	BIT
BMI	BRANCH ON N = 1 (2)																															30	2	2									BMI
BNE	BRANCH ON Z = 0 (2)																															D0	2	2									BNE
BPL	BRANCH ON N = 0 (2)																															10	2	2									BPL
BRK	BREAK													00	7	1																									· · 1 · · · ·	BRK
BVC	BRANCH ON V = 0 (2)																															50	2	2									BVC
BVS	BRANCH ON V = 1 (2)																															70	2	2									BVS
CLC	0 → C													18	2	1																									· · · · · · 0	CLC
CLD	0 → D													D8	2	1																									· · · · 0 · ·	CLD
CLI	0 → I													58	2	1																									· · · · · 0 · ·	CLI
CLV	0 → V													B8	2	1																									· 0 · · · · ·	CLV
CMP	A - M	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										N · · · · · Z C	CMP
CPX	X - M	E0	2	2	EC	4	3	E4	3	2																															N · · · · · Z C	CPX
CPY	Y - M	C0	2	2	CC	4	3	C4	3	2																															N · · · · · Z C	CPY
DEC	M - 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													N · · · · · Z ·	DEC
DEX	X - 1 → X													CA	2	1																									N · · · · · Z ·	DEX
DEY	Y - 1 → Y													88	2	1																									N · · · · · Z ·	DEY
EOR	A ∨ M → A (1)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										N · · · · · Z ·	EOR
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													N · · · · · Z ·	INC
INX	X + 1 → X													E8	2	1																									N · · · · · Z ·	INX
INY	Y + 1 → Y													C8	2	1																									N · · · · · Z ·	INY
JMP	JUMP TO NEW LOC				4C	3	3																												6C	5	3						JMP
JSR	JUMP SUB				20	6	3																																				JSR
LDA	M → A (1)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										N · · · · · Z ·	LDA
LDX	M → X (1)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3								86	4	2	N · · · · · Z ·	LDX
LDY	M → Y (1)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													N · · · · · Z ·	LDY
LSR	0 → [M] → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0 · · · · · Z C	LSR
NOP	NO OPERATION													EA	2	1																									· · · · · · ·	NOP
ORA	A ∨ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										N · · · · · Z ·	ORA
PHA	A → Ms S - 1 → S													48	3	1																									· · · · · · ·	PHA
PHP	P → Ms S - 1 → S													08	3	1																									· · · · · · ·	PHP
PLA	S + 1 → S Ms → A													68	4	1																									N · · · · · Z ·	PLA
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)	PLP
ROL	[M] ← [C]				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													N · · · · · Z C	ROL
ROR	[C] → [M]				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													N · · · · · Z C	ROR
RTI	RTRN INT													40	6	1																									(RESTORED)	RTI
RTS	RTRN SUB													60	6	1																									· · · · · · ·	RTS
SBC	A - M - C̄ → A (1)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										N V · · · · Z C (3)	SBC
SEC	1 → C													38	2	1																									· · · · · · 1	SEC
SED	1 → D													F8	2	1																									· · · · 1 · ·	SED
SEI	1 → I													78	2	1																									· · · 1 · · ·	SEI
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	99	5	3													· · · · · · ·	STA
STX	X → M				8E	4	3	86	3	2																																	· · · · · · ·	STX
STY	Y → M				8C	4	3	84	3	2													94	4	2																	· · · · · · ·	STY
TAX	A → X													AA	2	1																									N · · · · · Z ·	TAX
TAY	A → Y													A8	2	1																									N · · · · · Z ·	TAY
TSX	S → X													BA	2	1																									N · · · · · Z ·	TSX
TXA	X → A													8A	2	1																									N · · · · · Z ·	TXA
TXS	X → S													9A	2	1																									· · · · · · ·	TXS
TYA	Y → A													98	2	1																									N · · · · · Z ·	TYA

(1) ADD 1 to N IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO N IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO N IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT ⎺ BORROW
(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

- ADD
- SUBTRACT
∧ AND
∨ OR
∨ EXCLUSIVE OR

M MEMORY BIT 7
M6 MEMORY BIT 6
~ NO CYCLES
# NO BYTES

APPENDIX J
AIM 65 CONNECTOR SIGNALS

TERMINAL	SIGNAL	
1	-12V	
2	GND	*
3	+5	
4	+12V	
5	GND	*
6	+24V	

*Connected together on Master Module.

Figure J-1. Terminal Board TB1 (Power) Signals

PIN	SIGNAL	PIN	SIGNAL
22		Z	
21	CA2	Y	SERIAL INPUT
20	CA1	X	
19	CB2	W	TAPE 1A
18	CB1	V	TAPE 2A
17	PB6	U	TTY PTR
16	PB5	T	TTY KYBD
15	PB7	S	TTY PTR RTN (+)
14	PA0	R	TTY KYBD RTN (+)
13	PB4	P	AUDIO OUT HI
12	PB3	N	+12V
11	PB2	M	AUDIO OUT LO
10	PB1	L	AUDIO IN
9	PB0	K	
8	PA7	J	TAPE 2B
7	PA6	H	TAPE 2B RTN
6	PA5	F	TAPE 1B
5	PA4	E	TAPE 1B RTN
4	PA1	D	R/$\overline{W}$
3	PA2	C	ø2
2	PA3	B	
1	GND	A	+5V

Connector J1 Pin Assignments (Back View)

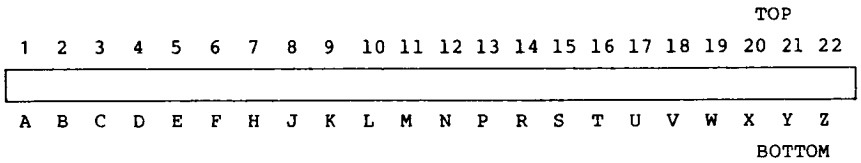


Figure J-2. Connector J1 (Application) Signals

PIN	SIGNAL
1	TE10
2	TE9
3	TE8
4	TE7
5	TE6
6	VTH
7	TE5
8	TE4
9	TE3
10	TE2
11	TE1
12	P2
13	P1
14	START
15	COMMON
16	GND
17	M+

Connector J2 Pin Assignments (Top View)

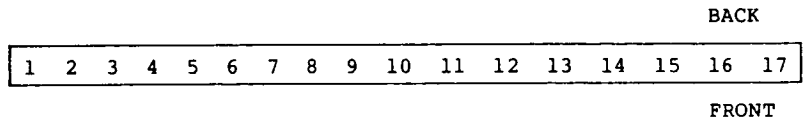


Figure J-3 Connector J3 (Printer) Signals

PIN	SIGNAL	PIN	SIGNAL
22	GND	Z	RAM R/$\overline{W}$
21	+5V	Y	$\overline{\phi 2}$
20	$\overline{CSA}$	X	TEST
19	$\overline{CS9}$	W	R/$\overline{W}$
18	$\overline{CS8}$	V	SYS R/$\overline{W}$
17	+12V	U	SYS ϕ2
16	-12V	T	A15
15	D0	S	A14
14	D1	R	A13
13	D2	P	A12
12	D3	N	A11
11	D4	M	A10
10	D5	L	A9
9	D6	K	A8
8	D7	J	A7
7	$\overline{RES}$	H	A6
6	$\overline{NMI}$	F	A5
5	S.O.	E	A4
4	$\overline{IRQ}$	D	A3
3	ϕ1	C	A2
2	RDY	B	A1
1	SYNC	A	A0

Connector J3 Pin Assignments (Back View)

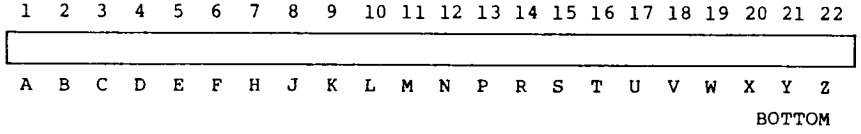
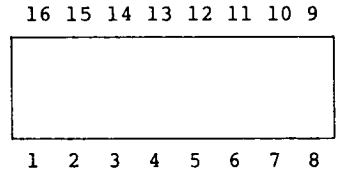


Figure J-4. Connector J3 (Expansion) Interface Signals

PIN	SIGNAL
1	KO1
2	KI1
3	KI8
4	KO7
5	KI7
6	KI2
7	KI4
8	KI3
9	KO8
10	KO2
11	KI5
12	KI6
13	KO6
14	KO5
15	KO4
16	KO3

Master Module J4 Pin Assignments (Top View):



Keyboard Module J1 Pin Assignments (Top View):

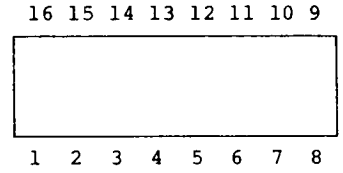


Figure J-5. Connector J4 (Keyboard) Signals

PIN	SIGNAL	PIN	SIGNAL
1	GND	20	A7
2	SYNC	21	A6
3	RDY	22	A5
4	ø1	23	A4
5	$\overline{IRQ}$	24	A3
6	S.O.	25	A2
7	$\overline{NMI}$	26	A1
8	$\overline{RES}$	27	A0
9	$\overline{CSAC}$	28	D0
10	SYS R/$\overline{W}$	29	D1
11	SYS ø2	30	D2
12	A15	31	D3
13	A14	32	D4
14	A13	33	D5
15	A12	34	D6
16	A11	35	D7
17	A10		
18	A9		
19	A8		

Connector J5 Pin Assignments (Top View):

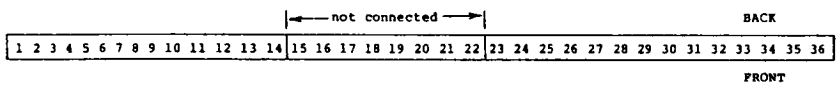


Figure J-6. Connector J5 (Display) Signals

APPENDIX K
AIM 65 USER SUB-MONITOR WITH 24-HOUR CLOCK

This appendix describes a typical user program - an AIM 65 User Sub-Monitor with 24-hour clock. This program features a Sub-Monitor, entered from the AIM 65 Monitor, which allows user-defined application subroutines to be called by typing certain user-defined keys.

Also included with the Sub-Monitor is a 24-hour clock that may be displayed continuously or displayed and printed upon command, along with a 20 character message.

An included I/O Monitor displays and prints the state of the 16 User R6522 Port A and Port B lines and the current time whenever a change is detected on any of the I/O lines. This program can easily be changed to display/print a message corresponding to the changed lines, e.g., 'PUMP MOTOR OFF' and the time of turn off.

The Sub-Monitor is listed in disassembly, source assembly and dump forms in Table K-1 to K-4. Although this program is shown as an example, it may be used as a model, or starter, for a similar user application. This program was initially prepared using the mnemonic entry capability (I command) so it reflects some operands in absolute and relative hexadecimal form rather than symbolic form. If the source code is loaded for input into the Assembler, all linkage should be symbolic to eliminate possible branching

errors. The program, as shown, is fairly long and should be loaded after you are familiar with AIM 65 operations.

The Sub-Monitor described will execute in the 1K RAM and requires addresses $0200-$03C9. In addition to the main program located in this area of RAM, a JMP instruction to address $0200 must be loaded in address $010C and the IRQ after Monitor Vector address $0368 must be loaded in address $A400.

K.1 SUB-MONITOR FUNCTIONS

The five functions programmed into the listed Sub-Monitor are:

KEY T: Allows the initial time to be entered in 24-hour format, HH:MM:SS (e.g., 15:01:15 = 1 minute 15 seconds after 3 p.m.). Enter all six digits and two colons. Automatic return to the Sub-Monitor occurs after entry of the last digit.

KEY M: Allows a 12-character message to be entered that will be displayed/printed with the time. If exactly 12 characters are entered (including spaces) the message will be displayed on the same line as the time. If less than 12 characters are entered (follow with a RETURN), the message will be displayed on a separate line, preceding the time.

KEY C: Causes continuous display of message and time. The time is updated at one second intervals. Return to the Sub-Monitor by pressing ESC.

KEY D: Display and print the message and time one time. Repeat as often as desired.

KEY I: Monitor the 16 User R6522 Port A and Port B I/O lines. Whenever any line changes state, the current state of all the lines is displayed and printed along with the time. Return to the Sub-Monitor by pressing ESC.

The Sub-Monitor is entered from the Monitor by typing F1. The Sub-Monitor prompt "%" is displayed upon entry from the Monitor and upon return from a commanded Sub-Monitor function.

When % is displayed, any of the programmed Sub-Monitor functions may be executed by typing the appropriate function key (T, M, D, C or I). To return to the AIM 65 Monitor, type ESC whenever % is displayed.

CAUTION

The Sub-Monitor 24-hour clock uses the User R6522 Timer 1 and the IRQ interrupt. Timer 1 and the IRQ interrupt processing routine will continue to run even after return to the AIM 65 monitor. This allows the

CAUTION (Cont.)

Sub-Monitor to be reentered and the current time to be displayed/printed at any time.

If you alter the memory locations used by the IRQ interrupt processing routine ($0368-$03C9) without first disabling the R6522 Timer 1 time-out generated IRQ interrupt, an error may occur. The easiest way to avoid this situation is to press RESET when you desire to run the Sub-Monitor 24-hour clock function.

Example:

```
<I>
%T
ENTER TIME: 12:30:00
%M
MESSAGE:AIM 65 TIME
%D
AIM 65 TIME 12:30:09
%D
AIM 65 TIME 12:30:14
%D
AIM 65 TIME 12:30:18
%M
MESSAGE:USER TIME
%D
USER TIME   12:30:25
%D
```


Example (Cont.)

```
USER TIME   12:30:39
12:30:41 A= 11111111
         B= 00001010
12:30:42 A= 11111111
         B= 11111111
12:30:55 A= 11111101
         B= 11111111
12:30:59 A= 11111111
         B= 11111111
12:31:01 A= 10111111
         B= 11111111
12:31:04 A= 11111111
         B= 11111111

%D
USER TIME   12:31:11
%
```

K.2 MNEMONIC ENTRY OF THE SUB-MONITOR

1. Enter the program as shown in Table K-1 using the I command.

2. Verify by running a disassembly listing using the K command.

3. Enter the Sub-Monitor by typing F1 and enter any function by typing T, M, D, C or I after % is displayed.

   If the Sub-Monitor or any of the commands do not operate correctly, compare the disassembly listing with Table K-1 and correct any detected errors.

4. Save the object code on audio cassette tape using the D command as shown in Table K-4.

Dump $A400 - $A401, $010C - $010E, and $0200 - $03C9 (unless changes have been made).

K.3 ASSEMBLY OF THE SUB-MONITOR

Assembly of the Sub-Monitor in single source file form requires the AIM 65 Assembler ROM, 4K RAM and one audio cassette recorder. With 4K RAM the source can be read into $03D0 - $0E6F then assembled for errors only and object code output to dummy device with the symbol table at $0E70 - $1000. Any detected errors during assembly can easily and quickly be corrected since the source code remains in RAM.

The assembly can also be performed in 1K RAM, by segmenting the source code into six files and assembling from tape. Any errors detected during assembly will require loading the source file into RAM, updating the source file and listing back to tape for a subsequent assembly.

The following procedure assumes 4K RAM is available:

1. Enter and initialize the Text Editor from $03D0 - $0E6F.

2. Type the source code from Table K-1 into the Text Buffer. List the source code to the printer to verify it with Table K-1. List the source also at selected checkpoints to audio cassette tape before assembly to save the source in case of accidental assembly symbol table overwrite or AIM 65 power turn-off.

3. Enter the Assembler and set the symbol table limits to $0E70 to $1000:

```
.CN
ASSEMBLER
FROM=0E70 TO=1000
```

4. Continue the assembly in an errors-only run with the object code output directed to dummy device:

```
IN=N
LIST?N
LIST-OUT=

OBJ?Y
OBJ-OUT=X

PASS 1

PASS 2

 ERRORS= 0000
```

If any errors are detected by the Assembler, re-enter the Text Editor using the T command and correct such errors. Then re-run the Assembler until no errors are detected as shown above.

5. After an error-free Assembler run is completed, list the source to audio cassette for intermediate saving.

6. Run the Assembler again but this time direct the source code to the display/printer and the object code output to memory:

```
(N)
ASSEMBLER
FROM=0E70 TO=1000
IN=M
LIST?Y
LIST-OUT=

OBJ?N
PASS 1

PASS 2
```

7. Execute the program by typing the commands as defined in Section K.1.

8. If the Sub-Monitor or any of the Sub-Monitor commands do not operate correctly, compare the source and/or the assembly listing with Tables K-1 and/or K-2 to find any errors. Correct any errors using the Text Editor, re-assemble, and checkout.

9. After the Sub-Monitor is correctly operating, permanently save the source and object programs on audio cassette tape. Also run a final assembly listing for future reference.

Table K-1. Sub-Monitor Source Listing

```
=(L)
/
OUT=
;*USER SUB-MONITOR*
;*CALLED BY F1 KEY*
;
;T=ENTER TIME
;   (IE: 14:34:51 -
;   24 HR. CLOCK)
;M=ENTER MESSAGE
;   (12 CHARS. TOTAL)
;D=DISPLAY TIME ONCE
;C=CONTINUOUS TIME
;I=I/O PORT MONITOR
;
;PRESS <ESC> KEY
;TO EXIT SUB-MONITOR
;
CLR =$EB44
CRLF =$E3F0
GETK2 =$EC82
OUTPUT =$E97A
RCHEK =$E807
RDRUB =$E95F
ROONEK =$ECEF
*=$92
SECSTO *=*+1
CNT *=*+1
STORA *=*+1
STORB *=*+1
*=$A000
ORB *=*+1
*=$A005
T1CH *=*+1
T1LL *=*+1
*=$A00B
ACR *=*+1
*=$A00E
IER *=*+1
ORA1 *=*+1
*=$A411
PRIFLG *=*+1
*=$010C
JMP UMON
*=$A400
 WOR INT
*=$0200
UMON JSR CRLF
LDA #$25 ;'%'
JSR OUTPUT ;PROMPT
JSR RCHEK ;<ESC>?
JSR RDRUB ;INPUT
CMP #$54 ;'T'
BNE *+5
JSR ENTIME
CMP #$4D ;'M'
BNE *+5
JSR ENTMES
CMP #$44 ;'D'
BNE *+5
JSR DISP
CMP #$43 ;'C'
BNE *+5
JSR CONT
CMP #$49 ;'I'
BNE *+5
JSR IOMON
JMP UMON
ENTIME JSR CRLF
LDX #00
MSGLP LDA TIMSG,X
JSR OUTPUT
INX
CPX #12
BNE MSGLP
LDX #00
TINLP JSR RDRUB
STA $A5,X
INX
CPX #08
BMI TINLP
LDA #$30
STA $98
LDA #$C0
STA IER
LDA #$40
STA ACR
LDA #$00; CALIBRATE
STA T1LL ; FOR
LDA #$F4 ; CORRECT
```

Table K-1. Sub-Monitor Source Listing (Cont)

```
STA TICH ; TIME
RTS
TIMSG .BYT 'ENTER
.BYT 'TIME: '
ENTMES JSR CRLF
LDX #00
MESLP LDA MESG,X
JSR OUTPUT
INX
CPX #08
BNE MESLP
LDX #00
MINLP JSR RDRUB
STA $99,X
INX
CPX #12
BMI MINLP
RTS
MESG .BYT 'MESSAGE:'
DISP JSR CRLF
LDX #00
DISPLA LDA $99,X
JSR OUTPUT
INX
CPX #20
BMI DISPLA
RTS
CONT LDA PRIFLG
STA $97
LDA #00 ;DISABLE
STA PRIFLG ;PRINTER
CONTLP JSR CLR
LDX #00
JSR DISPLA
LDA $AC
STA SECSTO
DISLP LDA $AC
CMP SECSTO
BEQ DISLP
JSR ESCHEK
BNE CONTLP
LDA $97
STA PRIFLG
RTS
ESCHEK JSR RDONEK
DEY
BMI NOESC
LDX #00
JSR GETK2
CMP #$1B ;<ESC>
BNE NOESC
LDA #00
RTS
NOESC LDA #$FF
RTS
IOMON JSR ESCHEK
BNE *+5
JMP UMON
TESTA LDA ORA1
CMP STORA
BEQ TESTB
STA STORA
JMP CHANGE
TESTB LDA ORB
CMP STORB
BEQ RETURN
STA STORB
CHANGE JSR CLR
LDX #00
STX CNT
CHLOOP LDA $A5,X
; TIME STORAGE LOC.
JSR OUTPUT
INX
CPX #08
BNE CHLOOP
LDA #$20 ;SPACE
JSR OUTPUT
LDA #$41 ; 'A'
JSR OUTPUT
LDA #$3D ; '='
JSR OUTPUT
LDA #$20 ; SPACE
LDX #00
STX CNT
SPLOOP JSR OUTPUT
INX
CPX #09
```


Table K-1. Sub-Monitor Source Listing (Cont)

```
BNE SPLOOP ;MORE SP
LDA #$42 ; 'B'
JSR OUTPUT
LDA #$3D ; '='
JSR OUTPUT
LDA #$20 ; SPACE
JSR OUTPUT
LDX STORB
JSR ORLOOP
RETURN JMP IOMON
ORLOOP TXA
ASL A
TAX
LDA #$30
ADC #00
JSR OUTPUT
INC CNT
LDA CNT
CMP #08
BNE ORLOOP
JSR CRLF
RTS
INT PHA
NOP
TXA
PHA
INC $98
LDX #$40
CPX $98
BNE CONTIN
LDA #$30
STA $98
INC $AC
LDX #$3A
CPX $AC
BNE TIMOK
STA $AC
INC $AB
LDX #$36
CPX $AB
BNE TIMOK
STA $AB
INC $A9
LDX #$3A
CPX $A9
BNE TIMOK
STA $A9
INC $A8
LDX #$36
CPX $A8
BNE TIMOK
STA $A8
INC $A6
LDX #$3A
CPX $A6
BNE NUDAY
STA $A6
INC $A5
JMP TIMOK
NUDAY LDX #$34
CPX $A6
BNE TIMOK
LDX #$32
CPX $A5
BNE TIMOK
STA $A5
STA $A6
TIMOK NOP
NOP
NOP
CONTIN PLA
TAX
NOP
LDA $A004
PLA
RTI
.END
```

Table K-2. Sub-Monitor Assembly Listing

```
<N>
ASSEMBLER
FROM=0E70 TO=1000
IN=M
LIST?Y
LIST-OUT=

OBJ?Y
OBJ-OUT=X

PASS 1

PASS 2

==0000
;*USER SUB-MONITOR*
;*CALLED BY F1 KEY*
;
;T=ENTER TIME
;   (IE: 14:34:51
;   24 HR. CLOCK)
;M=ENTER MESSAGE
;   (12 CHARS. TOTAL)
;D=DISPLAY TIME ONCE
;C=CONTINUOUS TIME
;I=I/O PORT MONITOR
;
;PRESS <ESC> KEY
;TO EXIT SUB-MONITOR
;
==0000 CLR
       =$E844
==0000 CRLF
       =$E9F0
==0000 GETK2
       =$EC82
==0000 OUTPUT
       =$E97A
==0000 RCHEK
       =$E907
==0000 RDRUB
       =$E95F
==0000 ROONEK
       =$ECEF
==0000
       *=$93
==0093 SECSTO
       *=*+1
==0094 CNT
       *=*+1
==0095 STORA
       *=*+1
==0096 STORB
       *=*+1
==0097
       *=$A000
==A000 ORB
       *=*+1
==A001
       *=$A005
==A005 T1CH
       *=*+1
==A006 T1LL
       *=*+1
==A007
       *=$A00B
==A00B ACR
       *=*+1
==A00C
       *=$A00E
==A00E IER
       *=*+1
==A00F ORA1
       *=*+1
==A010
       *=$A411
==A411 PRIFLG
       *=*+1
==A412
       *=$010C
4C0002 JMP UMON
==010F
       *=$A400
==A400
6803     WOR INT
==A402
       *=$0200
==0200 UMON
20F0E9 JSR CRLF
A925   LDA #$25
;'%'
207AE9 JSR OUTPUT
;PROMPT
2007E9 JSR RCHEK
;<ESC>?
205FE9 JSR RDRUB
;INPUT
```


Table K-2. Sub-Monitor Assembly Listing (Cont)

```
C954   CMP #$54
;'T'
==0210
D003   BNE *+5
203402 JSR ENTIME
C94D   CMP #$4D
;'M'
D003   BNE *+5
207502 JSR ENTMES
C944   CMP #$44
;'D'
D003   BNE *+5
==0220
209A02 JSR DISP
C943   CMP #$43
;'C'
D003   BNE *+5
20AA02 JSR CONT
C949   CMP #$49
;'I'
D003   BNE *+5
20E602 JSR IOMON
==0231
4C0002 JMP UMON
==0234 ENTIME
20F0E9 JSR CRLF
A200   LDX #00
==0239 MSGLP
BD6902 LDA TIMSG,X
207AE9 JSR OUTPUT
E8     INX
E00C   CPX #12
D0F5   BNE MSGLP
A200   LDX #00
==0246 TINLP
205FE9 JSR RDRUB
95A5   STA $A5,X
E8     INX
E008   CPX #08
10F6   BMI TINLP
A930   LDA #$30
8598   STA $98
A9C0   LDA #$C0
==0256
8D0EA0 STA IER
A940   LDA #$40
8D0BA0 STA ACR
A900   LDA #$00
; CALIBRATE
8D06A0 STA T1LL
; FOR
A9F4   LDA #$F4
; CORRECT
8D05A0 STA T1CH
; TIME
==0268
60     RTS
==0269 TIMSG
454E   .BYT 'ENTER '
5449   .BYT 'TIME: '
==0275 ENTMES
20F0E9 JSR CRLF
A200   LDX #00
==027A MESLP
BD9202 LDA MESG,X
207AE9 JSR OUTPUT
E8     INX
E008   CPX #08
D0F5   BNE MESLP
A200   LDX #00
==0287 MINLP
205FE9 JSR RDRUB
9599   STA $99,X
E8     INX
E00C   CPX #12
10F6   BMI MINLP
60     RTS
==0292 MESG
4D45   .BYT 'MESSAGE
;'
==029A DISP
20F0E9 JSR CRLF
A200   LDX #00
==029F DISPLA
B599   LDA $99,X
207AE9 JSR OUTPUT
E8     INX
E014   CPX #20
10F6   BMI DISPLA
```

Table K-2. Sub-Monitor Assembly Listing (Cont)

```
60      RTS
==02AA CONT
AD11A4 LDA PRIFLG
8597   STA $97
A900   LDA #00
;DISABLE
8D11A4 STA PRIFLG
;PRINTER
==02B4 CONTLP
2044EB JSR CLR
A200   LDX #00
209F02 JSR DISPLA
A5AC   LDA $AC
8592   STA SECSTO
==02C0 DISLP
A5AC   LDA $AC
C592   CMP SECSTO
F0FA   BEQ DISLP
20D102 JSR ESCHEK
D0E9   BNE CONTLP
A597   LDA $97
8D11A4 STA PRIFLG
==02D0
60      RTS
==02D1 ESCHEK
20EFEC JSR ROONEK
88      DEY
300C   BMI NOESC
A200   LDX #00
2082EC JSR GETK2
C91B   CMP #$1B
;<ESC>
D003   BNE NOESC
A900   LDA #00
==02E2
60      RTS
==02E3 NOESC
A9FF   LDA #$FF
60      RTS
==02E6 IOMON
20D102 JSR ESCHEK
D003   BNE *+5
4C0002 JMP UMON
==02EE TESTA
AD0FA0 LDA ORA1
C595   CMP STORA
F005   BEQ TESTB
8595   STA STORA
4C0303 JMP CHANGE
==02FA TESTB
AD00A0 LDA ORB
C596   CMP STORB
F04E   BEQ RETURN
8596   STA STORB
==0303 CHANGE
2044EB JSR CLR
A200   LDX #00
8694   STX CNT
==030A CHLOOP
B5A5   LDA $A5,X
; TIME STORAGE LOC.
207AE9 JSR OUTPUT
E8      INX
E003   CPX #08
D0F6   BNE CHLOOP
A920   LDA #$20
;SPACE
207AE9 JSR OUTPUT
A941   LDA #$41
; 'A'
==031B
207AE9 JSR OUTPUT
A93D   LDA #$3D
; '='
207AE9 JSR OUTPUT
A920   LDA #$20
; SPACE
207AE9 JSR OUTPUT
A695   LDX STORA
205203 JSR ORLOOP
==032D
A920   LDA #$20
; SPACE
A200   LDX #00
8694   STX CNT
==0333 SPLOOP
207AE9 JSR OUTPUT
E8      INX
E009   CPX #09
D0F8   BNE SPLOOP
;MORE SP
A942   LDA #$42
```


Table K-2. Sub-Monitor Assembly Listing (Cont)

```
; 'B'
207AE9 JSR OUTPUT
A93D   LDA #$3D
; '='
207AE9 JSR OUTPUT
==0345
A920   LDA #$20
; SPACE
207AE9 JSR OUTPUT
A696   LDX STORB
205203 JSR ORLOOP
==034F RETURN
4CE602 JMP IOMON
==0352 ORLOOP
8A      TXA
0A      ASL A
AA      TAX
A900   LDA #$00
6900   ADC #00
207AE9 JSR OUTPUT
E694   INC CNT
A594   LDA CNT
C908   CMP #08
==0362
D0EE   BNE ORLOOP
20F0E9 JSR CRLF
60      RTS
==0368 INT
48      PHA
EA      NOP
8A      TXA
48      PHA
E698   INC $98
A240   LDX #$40
E498   CPX $98
D04E   BNE CONTIN
A900   LDA #$00
8598   STA $98
==0378
E6AC   INC $AC
A23A   LDX #$3A
E4AC   CPX $AC
D03F   BNE TIMOK
85AC   STA $AC
E6AB   INC $AB
A236   LDX #$36
E4AB   CPX $AB
==0388
D025   BNE TIMOK
85AB   STA $AB
E6A9   INC $A9
A23A   LDX #$3A
E4A9   CPX $A9
D02B   BNE TIMOK
85A9   STA $A9
E6A8   INC $A8
==0398
A236   LDX #$36
E4A8   CPX $A8
D021   BNE TIMOK
85A8   STA $A8
E6A6   INC $A6
A23A   LDX #$3A
E4A6   CPX $A6
D007   BNE NUDAY
==03A8
85A6   STA $A6
E6A5   INC $A5
4C8F03 JMP TIMOK
==03AF NUDAY
A234   LDX #$34
E4A6   CPX $A6
D00A   BNE TIMOK
A232   LDX #$32
E4A5   CPX $A5
D004   BNE TIMOK
85A5   STA $A5
85A6   STA $A6
==03BF TIMOK
EA      NOP
EA      NOP
EA      NOP
==03C2 CONTIN
68      PLA
AA      TAX
EA      NOP
AD04A0 LDA $A004
68      PLA
40      RTI
        .END
 ERRORS= 0000
```

Table K-3. Sub-Monitor Disassembly Listing

```
<M>=A400 68 03 7B E0

<K>*=010C
/01
 010C 4C JMP 0200

<K>*=0200
/
 0200 20 JSR E9F0
 0203 A9 LDA #25
 0205 20 JSR E97A
 0208 20 JSR E907
 020B 20 JSR E95F
 020E C9 CMP #54
 0210 D0 BNE 0215
 0212 20 JSR 0234
 0215 C9 CMP #4D
 0217 D0 BNE 021C
 0219 20 JSR 0275
 021C C9 CMP #44
 021E D0 BNE 0223
 0220 20 JSR 029A
 0223 C9 CMP #43
 0225 D0 BNE 022A
 0227 20 JSR 02AA
 022A C9 CMP #49
 022C D0 BNE 0231
 022E 20 JSR 02E6
 0231 4C JMP 0200
 0234 20 JSR E9F0
 0237 A2 LDX #00
 0239 BD LDA 0263,X
 023C 20 JSR E97A
 023F E8 INX
 0240 E0 CPX #0C
 0242 D0 BNE 0239
 0244 A2 LDX #00
 0246 20 JSR E95F
 0249 95 STA A5,X
 024B E8 INX
 024C E0 CPX #08
 024E 30 BMI 0246
 0250 A9 LDA #30
 0252 85 STA 98
 0254 A9 LDA #C0
 0256 8D STA A00E
 0259 A9 LDA #40
 025B 8D STA A00B
 025E A9 LDA #00
 0260 8D STA A006
 0263 A9 LDA #F4
 0265 8D STA A005
 0268 60 RTS
 0269 45 EOR 4E
 026B 54 ???
 026C 45 EOR 52
 026E 20 JSR 4954
 0271 4D EOR 3A45
 0274 20 JSR F020
 0277 E9 SBC #A2
 0279 00 BRK
 027A BD LDA 0292,X
 027D 20 JSR E97A
 0280 E8 INX
 0281 E0 CPX #08
 0283 D0 BNE 027A
 0285 A2 LDX #00
 0287 20 JSR E95F
 028A 95 STA 99,X
 028C E8 INX
 028D E0 CPX #0C
 028F 30 BMI 0287
 0291 60 RTS
 0292 4D EOR 5345
 0295 53 ???
 0296 41 EOR (47,X)
 0298 45 EOR 3A
 029A 20 JSR E9F0
 029D A2 LDX #00
 029F B5 LDA 99,X
 02A1 20 JSR E97A
 02A4 E8 INX
 02A5 E0 CPX #14
 02A7 30 BMI 029F
 02A9 60 RTS
 02AA AD LDA A411
 02AD 85 STA 97
 02AF A9 LDA #00
 02B1 8D STA A411
 02B4 20 JSR EB44
 02B7 A2 LDX #00
 02B9 20 JSR 029F
 02BC A5 LDA AC
```


Table K-3. Sub-Monitor Disassembly Listing (Cont)

```
 02BE 85 STA 93
 02C0 A5 LDA AC
 02C2 C5 CMP 93
 02C4 F0 BEQ 02C0
 02C6 20 JSR 02D1
 02C9 D0 BNE 02B4
 02CB A5 LDA 97
 02CD 8D STA A411
 02D0 60 RTS
 02D1 20 JSR ECEF
 02D4 88 DEY
 02D5 30 BMI 02E3
 02D7 A2 LDX #00
 02D9 20 JSR EC82
 02DC C9 CMP #1B
 02DE D0 BNE 02E3
 02E0 A9 LDA #00
 02E2 60 RTS
 02E3 A9 LDA #FF
 02E5 60 RTS
 02E6 20 JSR 02D1
 02E9 D0 BNE 02EE
 02EB 4C JMP 0200
 02EE AD LDA A00F
 02F1 C5 CMP 95
 02F3 F0 BEQ 02FA
 02F5 85 STA 95
 02F7 4C JMP 0303
 02FA AD LDA A000
 02FD C5 CMP 96
 02FF F0 BEQ 034F
 0301 85 STA 96
 0303 20 JSR EB44
 0306 A2 LDX #00
 0308 86 STX 94
 030A B5 LDA A5,X
 030C 20 JSR E97A
 030F E8 INX
 0310 E0 CPX #08
 0312 D0 BNE 030A
 0314 A9 LDA #20
 0316 20 JSR E97A
 0319 A9 LDA #41
 031B 20 JSR E97A
 031E A9 LDA #3D
 0320 20 JSR E97A
 0323 A9 LDA #20
 0325 20 JSR E97A
 0328 A6 LDX 95
 032A 20 JSR 0352
 032D A9 LDA #20
 032F A2 LDX #00
 0331 86 STX 94
 0333 20 JSR E97A
 0336 E8 INX
 0337 E0 CPX #09
 0339 D0 BNE 0333
 033B A9 LDA #42
 033D 20 JSR E97A
 0340 A9 LDA #3D
 0342 20 JSR E97A
 0345 A9 LDA #20
 0347 20 JSR E97A
 034A A6 LDX 96
 034C 20 JSR 0352
 034F 4C JMP 02E6
 0352 8A TXA
 0353 0A ASL  A
 0354 AA TAX
 0355 A9 LDA #30
 0357 69 ADC #00
 0359 20 JSR E97A
 035C E6 INC 94
 035E A5 LDA 94
 0360 C9 CMP #08
 0362 D0 BNE 0352
 0364 20 JSR E9F0
 0367 60 RTS
 0368 48 PHA
 0369 EA NOP
 036A 8A TXA
 036B 48 PHA
 036C E6 INC 98
 036E A2 LDX #40
 0370 E4 CPX 98
 0372 D0 BNE 03C2
 0374 A9 LDA #30
 0376 85 STA 98
 0378 E6 INC AC
 037A A2 LDX #3A
 037C E4 CPX AC
 037E D0 BNE 01BF
```

Table K-3. Sub-Monitor Disassembly Listing (Cont)

```
0380 85 STA AC
0382 E6 INC AB
0384 A2 LDX #36
0386 E4 CPX AB
0388 D0 BNE 03BF
038A 85 STA AB
038C E6 INC A9
038E A2 LDX #3A
0390 E4 CPX A9
0392 D0 BNE 03BF
0394 85 STA A9
0396 E6 INC A8
0398 A2 LDX #36
039A E4 CPX A8
039C D0 BNE 03BF
039E 85 STA A8
03A0 E6 INC A6
03A2 A2 LDX #3A
03A4 E4 CPX A6
03A6 D0 BNE 03AF
03A8 85 STA A6
03AA E6 INC A5
03AC 4C JMP 03BF
03AF A2 LDX #34
03B1 E4 CPX A6
03B3 D0 BNE 03BF
03B5 A2 LDX #32
03B7 E4 CPX A5
03B9 D0 BNE 03BF
03BB 85 STA A5
03BD 85 STA A6
03BF EA NOP
03C0 EA NOP
03C1 EA NOP
03C2 68 PLA
03C3 AA TAX
03C4 EA NOP
03C5 AD LDA A004
03C8 68 PLA
03C9 40 RTI
```


Table K-4. Sub-Monitor Dump Listing

```
CD
FROM=A400 TO=A401
OUT=

:02A40068030111
MORE?Y
FROM=010C TO=010E

:0301 0C4C0002005E
MORE?Y
FROM=0200 TO=03C9

:18020020F0E9A925207
AE92007E9205FE9C954D
00320340 2C94D000B03
:1802180 7207502C944D
00320 9A02C943D00320A
A02C949D00320E608FE
:180220024C000220F2E
9A200BD6302207AE9E3E
00C0F5A200205F0A9A
:180248E995A5E3E0033
0F6A9C08598A9C08D0EA
0A9408D0BA0A9000CDF
:1802608D06A0A9F43D0
5A060454E5445520544
94D453A2020F0E909FC
:180278A200BD9202207
AE9E3E003D0F5A200205
FE99599E8E00C300CD9
:180290F6604D455D534
14745 3A20F0E9A200B59
9207AE9E3E014300B87
:1802A8F660AD11A4859
7A9008D11A42044E8A20
0209F02A5AC85930B9C
:1802C0A5ACC593F0FA2
0D102D0E9A5978D11A46
020EFEC883 00CA20E58
:1802D8002082ECC918D
```

```
003A90060A9FF6020D10
2D0034C0002AD0F0A18
:1802F0A0C595F005859
54C0303AD00A0C596F04
E85962044EBA2000C57
:1803088694B5A5207AE
3E85003D0F6A920207AE
9A941207AE9A9ED0D4F
:180320207AE9A920207
AE9A695205203A920A20
08694207AE9E8E0088A
:180338 09D0F8A942207
AE3A93D207AE9A920207
AE9A696205204C0B44
:180350E6028A0AAAA93
0E900207AE9E594A594C
908D0EE20F0E9600CF1
:18036848EA8A48E593A
240E498D04EA9208593E
6ACA23AE4ACD03F0E84
:18038085ACE6ABA276E
4ABD0 585ABE6A9A23AE
4A9D02B85A9E5A80FA8
:180398A236E4A8D0218
5A8E5A6A23AE4AED0078
5A6E6A54CBF0 A20E64
:1803B034E4A5D00AA23
2E4A5D00485A585A6EAE
AEA68AAEAAD04A00EF4
:0203C868400175
MORE?N:0000170017
```

APPENDIX L
ERROR MESSAGES AND CODES

L.1 MONITOR/EDITOR ERROR MESSAGES

MEM FAIL HHHH - Memory failed to read stored data at address $HHHH.

ERROR BLK = HH - Block checksum failed during reading of audio tape at block number $HH.

ERROR HHHH - Record checksum failed during reading of audio tape at record beginning address $HHHH.

L.2 ASSEMBLER ERROR CODES

01	Undefined Symbol
02	Label Previously Defined or Forward Reference to Page 0 Symbol
03	Illegal or Missing Opcode
04	Address Not Valid
05	Accumulator Mode Not Allowed
06	Forward Reference to Page Zero
07	Ran off End of Line
08	Label Does Not Begin with Alphabetic Character
09	Label Greater Than Six Characters
10	Label or Opcode Contains Non-Alphanumeric
11	Forward Reference in Equate
12	Invalid Index — Must Be X or Y
13	Invalid Expression
14	Undefined Assembler Directive
15	Invalid Page 0 Operand
17	Relative Branch Out of Range
18	Illegal Operand Type for This Instruction
19	Out of Bounds on Indirect Addressing
20	A, X, Y, S and P are Reserved Labels
21	Program Counter Negative — Reset to 0