MicroDaSys

POST OFFICE BOX 36051 • LOS ANGELES, CA 90036 U.S.A. • ~~(213) 935-4555~~

MD-690 A

OLD NUMBER rev f
NEW 731-0876

S-100 BUS COMPATIBLE 6802/6809 SINGLE BOARD COMPUTER

Assembly Instructions

NEVER USE ACID CORE SOLDER FOR ELECTRONICS WORK — RESIN ONLY!!

( ) Check that each part in the parts list at the end of these instructions has been included. If any parts are missing, notify MicroDaSys immediately. Be sure to include proof of purchase with all requests.

( ) Some of the parts in your kit are static sensitive, and can be destroyed by mishandling. These parts are packed in special conductive foam and should be left this way until needed. Do not work on a carpet prone to static electricity.

( ) Install all 6 of the 20 pin sockets on the board. If the pins are numbered or if there is a mold mark identifying pin one, insert the socket accordingly. Pin one is identified by the dot on the silk screen pattern of the board. As each socket is inserted, turn the board over and solder two diagonally opposite pins to hold the socket in place. DO NOT SOLDER THE REST OF THE PINS YET. When all of the sockets have been tacked in place, check that they are fully against the PC board. Now finish soldering the rest of the pins, doing one or two on the same socket at a time, so as not to overheat or damage them. DID YOU MISS ANY PINS?

( ) Similarly install the two 18 pin sockets.

( ) Similarly install all 3 of the 16 pin sockets.

( ) Similarly install all 14 of the 14 pin sockets.

( ) Similarly install the two 40 pin sockets.

( ) Similarly install the 24 pin socket for U19.

( ) Insert all of the resistors except R17 through the appropriate holes and bend the leads slightly outward to hold each in place. Check that they are all completely seated and then turn the board over and solder them. DO NOT OVERHEAT. Clip the leads off at the solder joint. (Nail clippers work well for this.)

( ) Insert the capacitors C1 - C20 in a manner similar to that used with the resistors. C1, C2, C11, C13, C14, C15, and C16 are *polarized*, and must be inserted with the plus and minus leads as indicated on the circuit board. Solder and clip.

( ) Insert potentiometer R9 and solder.

( ) Install connectors P2, P3, and P4 by cutting the Molex strips to the required length, bending the long side of the pins at a 90° angle and soldering the short pins through the board as in Figure 1.

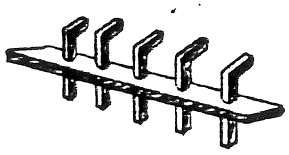


Figure 1

( ) Place the heat sink for U11 on the board, lining up the four holes. If heat sink compound is available it may be placed on the bottom of U11. Do not use too much. (Too much is worse than none at all.) Insert U11 through the holes in the heatsink and through the circuit board and secure with 6-32 screws from the bottom and nuts. Solder. DO NOT OVERHEAT. Clip.

( ) Install regulators U25 and U26 as shown on the board. DO NOT GET THEM MIXED UP! Regulators improperly installed or soldered may explode when power is applied. Use caution when first applying power. Secure the regulators with 6-32 screws from the bottom and nuts. Solder, being careful not to overheat.

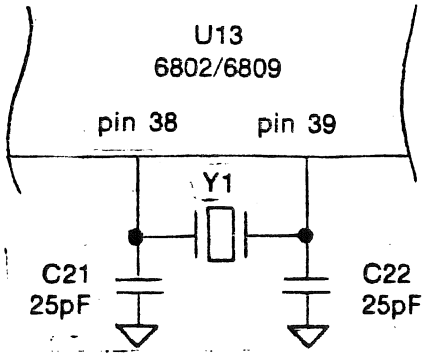
( ) Install U1 with the curved side toward the edge of the board. Solder, being careful not to overheat. Clip.

( ) Carefully install and solder crystal Y1. Clip.

( ) Install a small insulated wire jumper from pin 9 of U27 to pin 8 of U24.

OSCILLATOR DAMPING

To improve crystal oscillator response and prevent ringing, connect the two enclosed 25pF capacitators from either side of Y1 to ground, as shown. A convenient ground point is the upper left lead of C4 (with the edge connector toward the bottom of the board as viewed from the component side). Solder both capacitors to this point on the *rear* side of the board and insulate them with tape if necessary.



JUMPER SELECTION

( ) **J1** Cut the existing trace between the middle hole and the upper hole and then jumper the middle hole to the lower hole if your keyboard has a positive active strobe. If using the MicroDaSys keyboard, no alteration is needed.

( ) **J2** Connect the middle hole of each triplet to the lower hole of each triplet.

( ) **J4** For the 2 MHz processor and slower than 250 - 300 ns PROMs, jumper J4 to implement a wait state throughout the top 16K of memory.

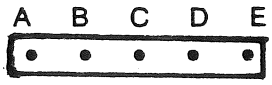
( ) **J5** If an external RAM card is being used from $AOOO to $AFFF, cut J5 and remove U17 and U18. Do not do this until checkout of the board is completed. RAM must be located at $AOOO for MONBUG to function.

( ) **J8** If U19 is a 2716, jumper the following pins:

A to B C to D

If U19 is a 2708, jumper the following pins:

B to C D to E



( ) **J9** Connect the middle hole to the right hole.

For 6802 processors:

( ) **J6** Connect.

( ) **J7** Connect the upper hole to one of the lower holes. The 6802 has 128 bytes of internal RAM, useful for the direct addressing mode of 6800 machine language. However, these bytes will interfere with an external RAM card located at $OOOO. It is therefore suggested that these 128 bytes be disabled and replaced by external RAM. This is accomplished by jumpering the middle hole to one of the lower holes. This should be done if an external RAM card is used at $OOOO.

( ) **J10** DO NOT CONNECT THE MIDDLE HOLE OF THE GROUP OF THREE. Jumper others as desired.

( ) **J11** Connect the middle hole to the left hole. The right hole may also be connected to the left hole as desired.

For 6809 processors:

( ) Install R17 (2.2K) on the MD690a board. Be sure you are using the **6809** monitor PROM for U19.

( ) **J7** BS (Interrupt Acknowledge) Jumper the middle hole of J7 to the upper hole only.

( ) **J11** DMA/BREQ (D.M.A. Request) If you do not plan to use this input (this is very likely) connect the middle hole of J11 to +5v at pin 20 of U31. Also jumper the left hole to the right hole. If you do plan to use it, it must be pulled up.

( ) **J6, J10** If you intend the use FIRQ (fast interrupt request) make the following change. Cut the trace that goes to the middle hole of the triplets at J10 **on the side toward U26** and jumper this hole to the upper hole of J6. This is the FIRQ, and as with IRQ and NMI it may be jumpered to any of the interrupt lines from the bus. These are the eight holes across the bottom of J10 and also pin 73 of the S100 bus.

ADDITIONAL OPTIONS

( ) For the 2MHz processor, use an 8 MHz crystal.

( ) To use the MIC input of the cassette instead of the AUX, change R3 to 1K.

6809 ADAPTER ASSEMBLY INSTRUCTIONS

Follow Carefully

Insert the 40 pin socket in the holes marked 6809 (toward the center of the board) FROM THE UN-PRINTED SIDE. *Is it the right spot?* If so, solder it in place.

Similarly install the 14 pin socket from the un-printed side and solder in place.

Install a wire on the unprinted side of the board between the two separate round holes, and solder.

The larger pins of the double-sided carrier will now be soldered to the printed side of the adapter socket board.

Install the carrier from the *printed* side of the board, in the holes marked 6802. *Do not press it all the way in.* Solder from the sides, *being careful not to bridge connections or traces.*

With the 14 pin socket to the right side, install the 74LS32 and the 6809 with pin one to the *upper left.*

TEST

( ) This completes assembly of your MD-690A. CHECK ALL SOLDER CONNECTIONS NOW. Finding the solder bridges now will save you weeks of troubleshooting later, and maybe even some parts.

( ) Do not plug the CPU card into the bus until you have verified the following voltages on the motherboard:

pins 1 and 51	+8 to +12 volts
pins 50 and 100	Ground
pin 2	+16 to +22 volts
pin 52	—16 to —22 volts

If these voltages do not check out STOP and determine why.

( ) Turn off power and discharge the capacitors in your power supply by touching an old screwdriver across their terminals. There will be a large spark. It is better to have it go through your screwdriver than the foil conductors of your board's edge connector.

( ) Plug the board in and CAUTIOUSLY APPLY POWER. Check the left hand pin of U26 (the pin toward U25) for +12 (+ or — .5) volts. Check the left hand pin of U25 for —5 (+ or — .5) volts. Check the upper pin of R11 for +5 (+ or — .5) volts. Check the upper pin of U1 for —12 (+ or — .5) volts. If any of these voltages are not correct, check your work and replace defective regulators. Problems are almost always due to bad solder connections.

( ) If the voltages are all correct, turn off power, DISCHARGE THE CAPACITORS, remove the board and insert the IC's. MAKE SURE YOU LINE UP PIN ONE AS INDICATED BY THE BOARD AND SOCKETS. Plugging IC's in backwards kills them. Do not plug in the keyboard yet.

( ) Insert the board. If possible insert a "known good" video board into the bus and set the address for hexadecimal $FOOO. Apply power. The processor should home up, clear the screen, and print an asterisk (*). If this occurs (25% chance) proceed directly to the MicroDaSys User's Guide. If this does not occur but everything looks OK (60% chance) proceed to troubleshooting suggestions. If one or more chips get too hot or you smell something burning, remove power immediately and check your soldering and chip placement/orientation.

NOTE: The MD-690A/B should not be used in a terminated motherboard. In particular, termination of the DIO-7 lines will impair the function of the board.

TROUBLESHOOTING — oscilloscope required

( ) Connect the keyboard. Try its reset.

( ) Is there a 1MHz clock on the E line of the processor? If not, check crystal and power to 6802 (or 6809). The crystal must be 'AT' cut fundamental series resonant.

( ) Are the read/write and vma lines of the 6802 (or 6809) active immediately after Reset? (Reset the processor from the keyboard.)

( ) Are all the address and data lines active? Check for shorts. If all the address lines seem to be counting, the processor cannot read the monitor program. Check the connections to the PROM. Is the chip selected on Reset?

( ) If these lines look OK at the 6802 (or 6809), check that the buffers are working.

( ) Is the RAM selected after Reset? If neither RAM nor PROM is selected check the decoder circuitry (74155 etc.)

( ) If only the keyboard doesn't work check for a strobe and check the PIA connections.

( ) As a last resort, check the pins of all IC's to see if they look reasonable. Use an ohmeter to check for shorts.

CASETTE CALIBRATION USING MONBUG

( ) A fully functional processor is assumed. Refer to the User's Manual and execute the following program, recording a 2-3 minute tape.

```
$A100     PO    LDAA #$FF       86 FF
                JSR CASOUT      BD E1 54
          P1    LDAA #$CA       86 CA
                JSR CASOUT      BD E1 54
                BRA P1          20 F9
```

Play back the tape running this program:

```
$A200     LO    LDX #SCNTOP     CE FO OO
          L1    JSR CASIN       BD E1 97
                STAA X          A7 OO
                INX             08
                CPX #SCNBOT     8C F4 OO
                BE L1           26 F5
                BRA LO          2O FO
```

Adjust R9 to the middle of the range for which the screen is filled with reverse field J's

( ) Some cassette recorders invert the phase of the playback data. This makes recovery of the valid data from your cassette impossible. AS A LAST RESORT, if it becomes apparent that your casette inverts phase, cut the connection between the middle hole and the right hole of J3 on the back of the board and jumper the middle hole to the left hole. If this does not help, phase inversion was not the problem. Remove the new jumper and re-jumper as before.

PARTS LIST : MD-690A

INTEGRATED CIRCUITS*

( )	U1	79L12	( )	U19	2708 (2716 optional)
( )	U2	74LS14	( )	U20	2716 (optional) 3 voltage type
( )	U3	4040	( )	U21	2716 (optional) "
( )	U4	74LS08	( )	U22	2716 (optional) "
( )	U5	74LS74	( )	U23	2716 (optional) "
( )	U6	74LS86		U24	74LS08
( )	U7	8T20	( )	U25	7905 or 320T-5
( )	U8	74LS121	( )	U26	7812 or 340T+12
( )	U9	74LS10	( )	U27	74LS00
( )	U10	74LS155	( )	U28	81LS98
( )	U11	LM340K or LM309K, +5	( )	U29	81LS95
( )	U12	6821	( )	U30	81LS95
( )	U13	6802 (6809 optional)	( )	U31	81LS95
( )	U14	74LS74	( )	U32	81LS95
( )	U15	1488	( )	U33	81LS95
( )	U16	1489	( )	U34	74LS02
( )	U17	4045 or 2114	( )	U35	74LS25
( )	U18	4045 or 2114			

*NOTE: 74XX parts may be substituted for 74LSXX parts.

RESISTORS

( )	R1	10K, BN,BK,OR	( )	R11	10K BN,BK,OR
( )	R2	100K BN,BK,YL	( )	R12	2.2K RD,RD,RD
( )	R3	10K BN,BK,OR	( )	R13	2.2K RD,RD,RD
( )	R4	220 ohms RD,RD,BN	( )	R14	2.2K RD,RD,RD
( )	R5	220 ohms RD,RD,BN	( )	R15	2.2K RD,RD,RD
( )	R6	22K RD,RD,OR	( )	R16	2.2K RD,RD,RD
( )	R7	220 ohms RD,RD,BN	( )	R17	2.2K RD,RD,RD
( )	R8	22K RD,RD,OR	( )	R18	2.2K RD,RD,RD
( )	R9	20K potentiometer	( )	R19	2.2K RD,RD,RD
( )	R10	8.2K GY,RD,RD	( )	R20	2.2K RD,RD,RD

BK=black
BN=brown
RD=red
OR=orange
YL=yellow
GN=green
BU=blue
VL=violet
GY=gray
WT=white

CAPACITORS

( )	C1	1 uF electrolytic	( )	C11	100 uF electrolytic
( )	C2	10 uF electrolytic	( )	C12	.01 uF disc (103)
( )	C3	.01 uF disc (103)	( )	C13	1 uF electrolytic
( )	C4	.01 uF disc (103)	( )	C14	1 uF electrolytic
( )	C5	DELETED	( )	C15	10 uF electrolytic
( )	C6	.1 uF disc (104)	( )	C16	1 uF electrolytic
( )	C7	.1 uF disc (104)	( )	C17	.01 uF disc (103)
( )	C8	.01 uF disc (103)	( )	C18	.01 uF disc (103)
( )	C9	.05 uF mylar (503)	( )	C19	.01 uF disc (103)
( )	C10	82 pF mica (82)	( )	C20	.01 uF disc (103)
			( )	C21	25pF disc (25)
			( )	C22	25pF disc (25)

CRYSTAL

( )	Y1	4 MHz

SOCKETS

( )	2	40 pin IC sockets
( )	1	24 pin IC socket
( )	6	20 pin IC sockets
( )	2	18 pin IC sockets
( )	3	16 pin IC sockets
( )	14	14 pin IC sockets

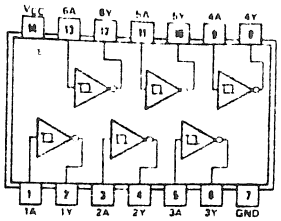
HARDWARE

( )	1	To-3 Heat Sink
( )	4	6-32 x 3/8 machine screws
( )	4	6-32 nuts

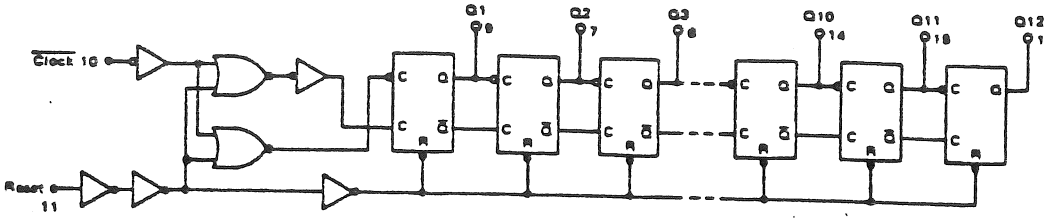
MISCELLANEOUS

( )	3	Molex strips
( )	1	Circuit Board

PINOUTS



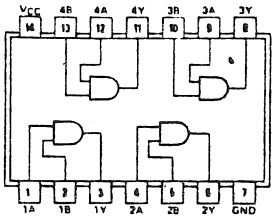
7414



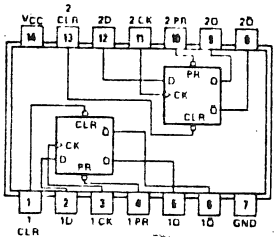
4040

Q4 = Pin 5 Q7 = Pin 4
Q5 = Pin 3 Q8 = Pin 13
Q6 = Pin 2 Q9 = Pin 12

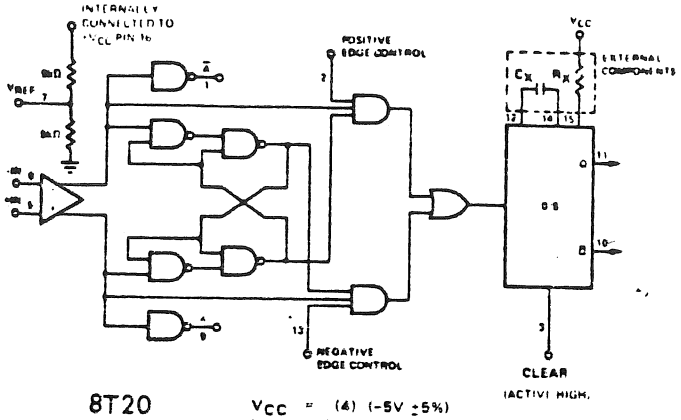
V_{DD} = Pin 16
V_{SS} = Pin 8



7408

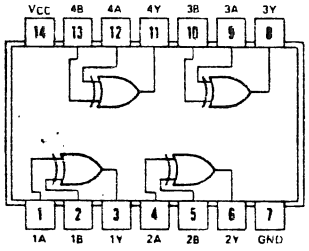


7474

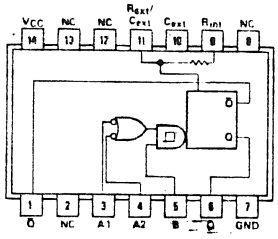


8T20

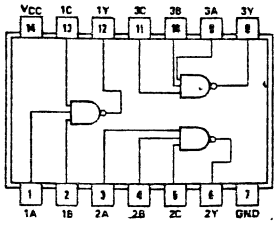
V_{CC} = (4) (−5V ±5%)
V_{CC} = (16) (+5V ±5%)
GND = (8)
( ) = Denotes Pin Numbers



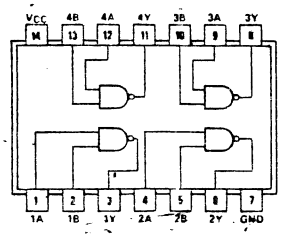
7486



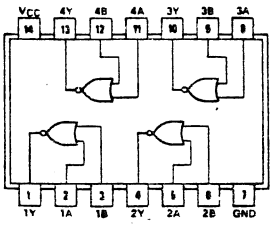
74121



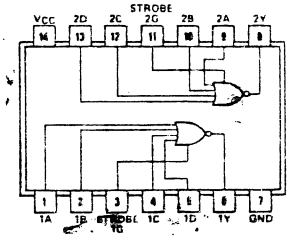
7410



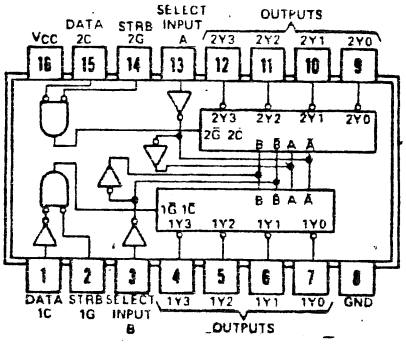
7400



7402

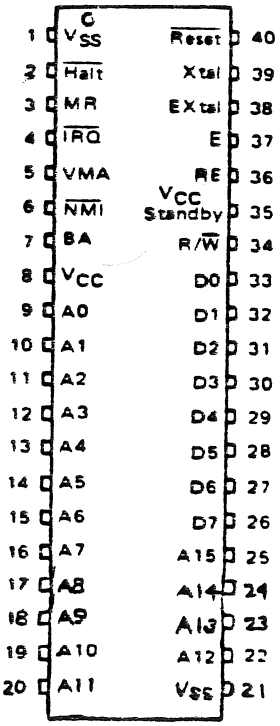


7425

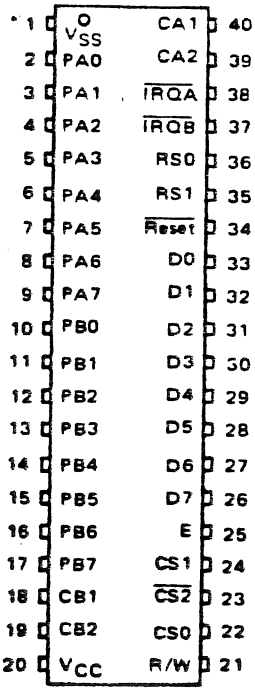


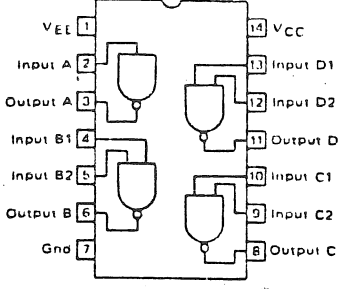
74155

6802

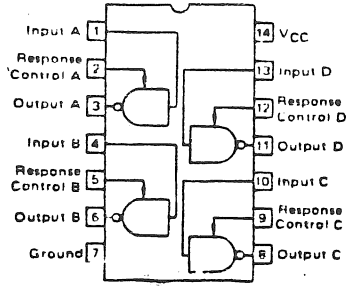


6821

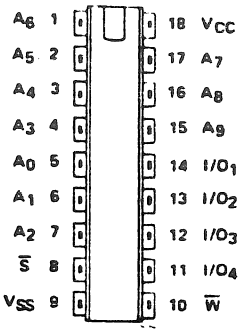




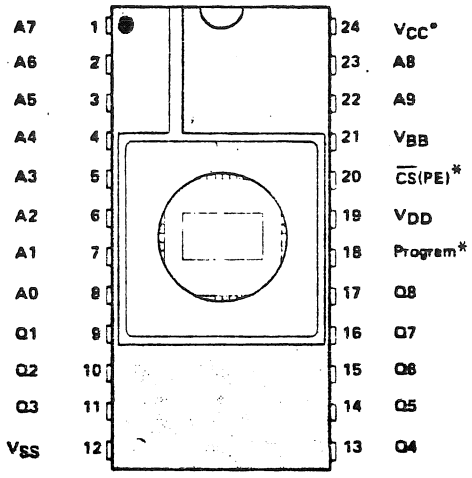
1488



1489



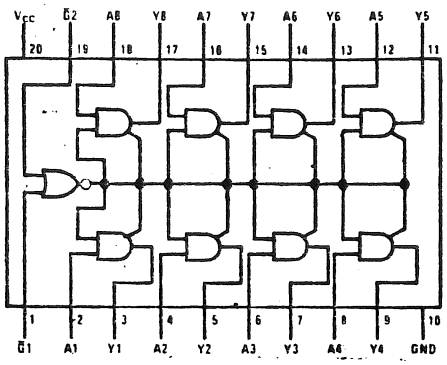
4045 or 2114



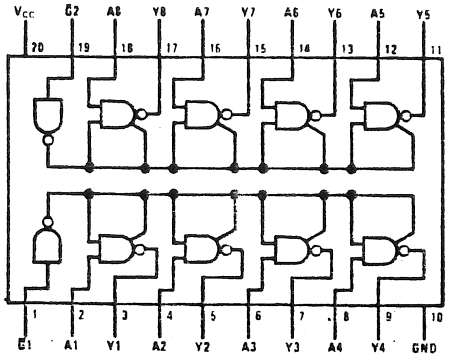
*For 2716 JL Only: Pin:

18	CS (Program)
20	A10
24	VCC (PE)

2708 / 2716



81LS95



81LS98

P4 RS-232 CONNECTOR

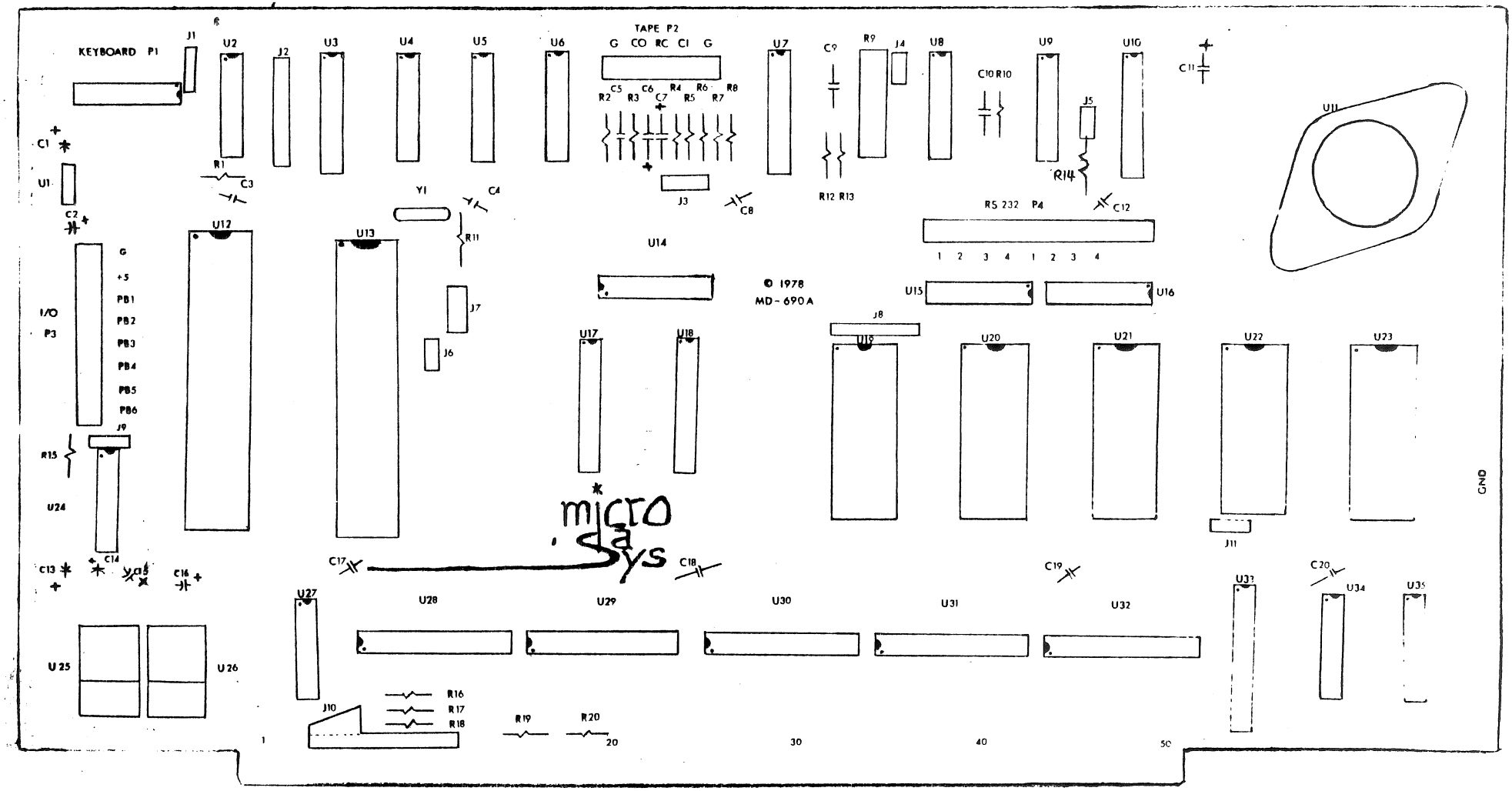
pin number	signal name	signal description
1	GND	Ground
2	OUTC	RS-232 version of input 1
3	OUTB	" 2
4	OUTD	" 3
5	OUTA	" 4
6	GND	Ground
7	INC	RS-232 version of output 1
8	INB	" 2
9	IND	" 3
10	INA	" 4

P1 KEYBOARD CONNECTOR

1	B0	data from keyboard (LSB)
2	B1	"
3	B2	"
4	B3	"
5	B4	"
6	B5	"
7	B6	"
8	B7	" (MSB)
9	STROBE (—)	Data Valid (pos or neg, jumper sel)
10	GND	Ground
11	GND	Ground
12	PRESET—	Reset (negative active)
13	—12 volts	—12 volts +/— .5 volts DC
14	+5 volts	+5 volts +/— .5 volts DC

THE MD-690A BUS

pin number	signal name	signal description
1	+8 volts	+7 to +12 volts DC
2	+16 volts	+15 to +22 volts DC
3	XRDY+	external device ready
4-11	VIO—7—	vectored interrupt lines 0—7
24	θ2	6800 *phase 1* clock
25	θ1	6800 *phase 2* clock
26	PHLDA+	processor acknowledge bus available
29	A5	address lines
30	A4	"
31	A3	"
32	A15	"
33	A12	"
34	A9	"
35	D01	data out of processor
36	D00	"
37	A10	address line
38	D04	data out of processor
39	D05	"
40	D06	"
41	D12	data in to processor
42	D13	"
43	D17	"
45	SOUT+	Output (write to $E800-$EFFF)
46	SINP+	Input (read from $E800-$EFFF)
47	SMEMR+	Memory Read
49	CLOCK—	vma • 02
50	GND	Ground
51	+8 volts	+7 to +12 volts DC
52	—16 volts	—16 to —22 volts DC
54	EXTCLR—	Clear External Devices
68	MWRITE+	Memory Write
72	PRDY+	external device ready
73	PINT—	master interrupt line
74	PHOLD—	Bus Request
75	PRESET—	Master Reset
76	PSYNC+	conditional θ1, status valid
77	PWR—	Output/Write, negative true
78	PDBIN+	Input/Read
79	A0	address line
80	A1	"
81	A2	"
82	A6	"
83	A7	"
84	A8	"
85	A13	"
86	A14	"
87	A11	"
88	D02	data out of processor
89	D03	"
90	D07	"
91	D14	data into processor
92	D15	"
93	D16	"
94	D11	"
95	D10	"
96	SINTA+	Interrupt Acknowledge (6809 only)
97	SWO—	Output/Write, negative active
99	POC—	Reset
100	GND	Ground



Parts Placement

REVISIONS					
EFF	AUTHORITY	ZONE	LTR	DESCRIPTION	DATE	APPROVED

U13
6802
(6809)

U19
2708(2716)
$E000/F000

(U20)
2716
$C000

(U21)
2716
$C800

(U22)
2716
$D000

(U23)
2716
$D800

U17
2114
$A000

U18
2114
$A400

U16
1488

U15
1489

RS-232

P1
KBD CONN

P2
CASSETTE CONN

P3
I/O CONN

P4
RS232 CONN

U12
6821
$F400

U14
8T28

U7
8T20

U3
4040

U8
'121

U1
79L12

U25
7905

U26
7812

U11
LM309K

U29
U30
U31
U32
8T95

U33
U34
U35
U36
U37
U38
U39
U40
U41
U42
U43
U44
U45
U46
U47
U48
U49
U50
U51
U52
U53
U54
U55
U56
U57
U58
U59
U60
U61
U62
U63
U64
U65
U66
U67
U68
U69
U70
U71
U72
U73
U74
U75
U76
U77
U78
U79
U80
U81
U82
U83
U84
U85
U86
U87
U88
U89
U90
U91
U92
U93
U94
U95
U96
U97
U98
U99
U100
U101
U102
U103
U104
U105
U106
U107
U108
U109
U110
U111
U112
U113
U114
U115
U116
U117
U118
U119
U120
U121
U122
U123
U124
U125
U126
U127
U128
U129
U130
U131
U132
U133
U134
U135
U136
U137
U138
U139
U140
U141
U142
U143
U144
U145
U146
U147
U148
U149
U150
U151
U152
U153
U154
U155
U156
U157
U158
U159
U160
U161
U162
U163
U164
U165
U166
U167
U168
U169
U170
U171
U172
U173
U174
U175
U176
U177
U178
U179
U180
U181
U182
U183
U184
U185
U186
U187
U188
U189
U190
U191
U192
U193
U194
U195
U196
U197
U198
U199
U200
U201
U202
U203
U204
U205
U206
U207
U208
U209
U210
U211
U212
U213
U214
U215
U216
U217
U218
U219
U220
U221
U222
U223
U224
U225
U226
U227
U228
U229
U230
U231
U232
U233
U234
U235
U236
U237
U238
U239
U240
U241
U242
U243
U244
U245
U246
U247
U248
U249
U250
U251
U252
U253
U254
U255
U256
U257
U258
U259
U260
U261
U262
U263
U264
U265
U266
U267
U268
U269
U270
U271
U272
U273
U274
U275
U276
U277
U278
U279
U280
U281
U282
U283
U284
U285
U286
U287
U288
U289
U290
U291
U292
U293
U294
U295
U296
U297
U298
U299
U300
U301
U302
U303
U304
U305
U306
U307
U308
U309
U310
U311
U312
U313
U314
U315
U316
U317
U318
U319
U320
U321
U322
U323
U324
U325
U326
U327
U328
U329
U330
U331
U332
U333
U334
U335
U336
U337
U338
U339
U340
U341
U342
U343
U344
U345
U346
U347
U348
U349
U350
U351
U352
U353
U354
U355
U356
U357
U358
U359
U360
U361
U362
U363
U364
U365
U366
U367
U368
U369
U370
U371
U372
U373
U374
U375
U376
U377
U378
U379
U380
U381
U382
U383
U384
U385
U386
U387
U388
U389
U390
U391
U392
U393
U394
U395
U396
U397
U398
U399
U400
U401
U402
U403
U404
U405
U406
U407
U408
U409
U410
U411
U412
U413
U414
U415
U416
U417
U418
U419
U420
U421
U422
U423
U424
U425
U426
U427
U428
U429
U430
U431
U432
U433
U434
U435
U436
U437
U438
U439
U440
U441
U442
U443
U444
U445
U446
U447
U448
U449
U450
U451
U452
U453
U454
U455
U456
U457
U458
U459
U460
U461
U462
U463
U464
U465
U466
U467
U468
U469
U470
U471
U472
U473
U474
U475
U476
U477
U478
U479
U480
U481
U482
U483
U484
U485
U486
U487
U488
U489
U490
U491
U492
U493
U494
U495
U496
U497
U498
U499
U500
U501
U502
U503
U504
U505
U506
U507
U508
U509
U510
U511
U512
U513
U514
U515
U516
U517
U518
U519
U520
U521
U522
U523
U524
U525
U526
U527
U528
U529
U530
U531
U532
U533
U534
U535
U536
U537
U538
U539
U540
U541
U542
U543
U544
U545
U546
U547
U548
U549
U550
U551
U552
U553
U554
U555
U556
U557
U558
U559
U560
U561
U562
U563
U564
U565
U566
U567
U568
U569
U570
U571
U572
U573
U574
U575
U576
U577
U578
U579
U580
U581
U582
U583
U584
U585
U586
U587
U588
U589
U590
U591
U592
U593
U594
U595
U596
U597
U598
U599
U600
U601
U602
U603
U604
U605
U606
U607
U608
U609
U610
U611
U612
U613
U614
U615
U616
U617
U618
U619
U620
U621
U622
U623
U624
U625
U626
U627
U628
U629
U630
U631
U632
U633
U634
U635
U636
U637
U638
U639
U640
U641
U642
U643
U644
U645
U646
U647
U648
U649
U650
U651
U652
U653
U654
U655
U656
U657
U658
U659
U660
U661
U662
U663
U664
U665
U666
U667
U668
U669
U670
U671
U672
U673
U674
U675
U676
U677
U678
U679
U680
U681
U682
U683
U684
U685
U686
U687
U688
U689
U690
U691
U692
U693
U694
U695
U696
U697
U698
U699
U700
U701
U702
U703
U704
U705
U706
U707
U708
U709
U710
U711
U712
U713
U714
U715
U716
U717
U718
U719
U720
U721
U722
U723
U724
U725
U726
U727
U728
U729
U730
U731
U732
U733
U734
U735
U736
U737
U738
U739
U740
U741
U742
U743
U744
U745
U746
U747
U748
U749
U750
U751
U752
U753
U754
U755
U756
U757
U758
U759
U760
U761
U762
U763
U764
U765
U766
U767
U768
U769
U770
U771
U772
U773
U774
U775
U776
U777
U778
U779
U780
U781
U782
U783
U784
U785
U786
U787
U788
U789
U790
U791
U792
U793
U794
U795
U796
U797
U798
U799
U800
U801
U802
U803
U804
U805
U806
U807
U808
U809
U810
U811
U812
U813
U814
U815
U816
U817
U818
U819
U820
U821
U822
U823
U824
U825
U826
U827
U828
U829
U830
U831
U832
U833
U834
U835
U836
U837
U838
U839
U840
U841
U842
U843
U844
U845
U846
U847
U848
U849
U850
U851
U852
U853
U854
U855
U856
U857
U858
U859
U860
U861
U862
U863
U864
U865
U866
U867
U868
U869
U870
U871
U872
U873
U874
U875
U876
U877
U878
U879
U880
U881
U882
U883
U884
U885
U886
U887
U888
U889
U890
U891
U892
U893
U894
U895
U896
U897
U898
U899
U900
U901
U902
U903
U904
U905
U906
U907
U908
U909
U910
U911
U912
U913
U914
U915
U916
U917
U918
U919
U920
U921
U922
U923
U924
U925
U926
U927
U928
U929
U930
U931
U932
U933
U934
U935
U936
U937
U938
U939
U940
U941
U942
U943
U944
U945
U946
U947
U948
U949
U950
U951
U952
U953
U954
U955
U956
U957
U958
U959
U960
U961
U962
U963
U964
U965
U966
U967
U968
U969
U970
U971
U972
U973
U974
U975
U976
U977
U978
U979
U980
U981
U982
U983
U984
U985
U986
U987
U988
U989
U990
U991
U992
U993
U994
U995
U996
U997
U998
U999
U1000

17. FOR 1MHZ MP JUMPER U24 PIN 5 TO 14.
16. INSTALL R17 FOR 6809 ONLY
15. U20-U23 ARE OPTIONAL.
14. D AND A' REFER TO UNBUFFERED LINES AT MP.
13. FOR MIC INPUT CHANGE R3 TO 1K.
12. 1MHZ MP: 4MHZ XTAL; 2MHZ MP: 8 MHZ
11. 6802: A-B-C; 6809: A-C, B to +5V U31 pin 20.
10. JUMPER AS DESIRED. 6802: DO NOT USE FIRQ
9. JUMPER MID-RIGHT.
8. FOR 2708: B-C, D-E; FOR 2716: A-B, C-D.
7. FOR 6802 A-D AND B-C TO DISABLE RAM; 6809: A-B
6. FOR 6802 MP JUMPER A TO B.
5. CUT FOR OFF-CARD RAM AT $A000.
4. CONNECT J4 FOR 2MHZ MP AND >300NS PROMS
3. IF RECORDER INVERTS PHAZE ALTER J3.
2. CASSETTE SPEED SELECT: AB-SLOW; BC-FAST.
1. ALTER J1 FOR POSITIVE ACTIVE KBD STROBE

NOTES

NEXT ASSY	USED ON
APPLICATION	

QTY REQD	FSCM NO	PART NO OR IDENTIFYING NO	NOMENCLATURE OR DESCRIPTION	ZONE	ITEM OR FIND NO

PARTS LIST

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES AND PER ANSI Y14.5

XXX	XX	ANGLES
±.010	±.03	±0°30'

MATERIAL

CONTRACT:		
DR Steve Ciarcia	DATE 10/25/78	micro sys
CHK		MD-690a
rev e	12/2/79	5100 COMPATIBLE 6800 CPU
APPD		

SIZE	FSCM NO	DWG NO
D	82577	78MDS6904

SCALE | SHEET

FORM NO. 11885-CS-CC REV. 7-76

NOTES ON CASSETTE INTERFACES: Your cassette interface should be the most reliable part of your computer. It's your link to stored programs. If it's not, you should find out why. A decent but inexpensive recorder is all that is required. Volume and tone adjustments are fairly critical, although once set they should not require additional tweaking. Use a good tape with no dropouts. We recommend SONY Low Noise tape as the best buy (under $2), although Maxell is probably the best tape available.

LOADING 300 BAUD TAPES THE FIRST TIME

Programs that come in the 300 baud Kansas City Standard cassette format may be read through your high speed 2400 baud cassette interface using the LD300 software enclosed. Enter the software at the address specified using MONBUG. Note that the contents of $A295 may be $4C or $4D, depending upon the polarity of you cassette recorder/player. No adjustment of your cassette interface should be neccessary. Load the program by executing LD300 while playing the 300 baud tape. It may take some experimentation to determine the correct contents of $A295.

Once the program is successfully loaded, **immediately** save it at 2400 baud as a backup. Next make sure that the input/output routines are correctly patched. MONBUG's keyboard input routine is INEEE ($E1AC) and its video output routine is OUTEEE ($E1D1). For cassette save and load, we suggest saving the entire memory image. After successfully patching the I/O pointers in the program and verifying its operation, save the final copy on cassette.

PROGRAM TO LOAD 300 BAUD KANSAS CITY TAPES THROUGH MICRODASYS 2400 BAUD MANCHESTER CASSETTE INTERFACE.

```
A200 8D 6E      LD300   BSR   CIN300    GET CHAR
A202 81 53              CMPA  #'S
A204 26 FA              BNE   LD300     NOT S
A206 8D 68              BSR   CIN300
A208 81 39              CMPA  #'9       E.O.F.
A20A 27 24              BEQ   C1        QUIT
A20C 81 31              CMPA  #'1
A20E 26 F0              BNE   LD300     NOT 1
A210 7F A0 4E           CLR   CKSM
A213 8D 2C              BSR   BYT300    GET BYTE
A215 80 02              SUBA  #2
A217 B7 A0 4F           STAA  BYTECT    BYTE COUNT
A21A 8D 17              BSR   BAD300    ADDRESS
A21C 8D 23      LD11    BSR   BYT300    DATA
A21E 7A A0 4F           DEC   BYTECT
A221 27 05              BEQ   LD15      LINE DONE?
A223 A7 00              STAA  X         STORE DATA
A225 08                 INX
A226 20 F4              BRA   LD11
A228 7C A0 4E   LD15    INC   CKSM
A22B 27 D3              BEQ   LD300     CHECKSUM OK
A22D BD E1 CF           JSR   WHAT      PRINT "?"
A230 7E E0 E3           JMP   CONTRL
A233 8D 0C      BAD300  BSR   BYT300    GET BYTE
A235 B7 A0 50           STAA  TXHI
A238 8D 07              BSR   BYT300    GET BYTE
A23A B7 A0 51           STAA  TXLO
A23D FE A0 50           LDX   TXHI      COMBINE
A240 39                 RTS
A241 8D 10      BYT300  BSR   INH300    GET HEX CHAR
```

```
A243 48                 ASLA
A244 48                 ASLA
A245 48                 ASLA
A246 48                 ASLA
A247 16                 TAB             MOVE TO MSB OF B
A248 8D 09              BSR   INH300    GET HEX CHAR
A24A 1B                 ABA             COMBINE
A24B 16                 TAB
A24C FB A0 4E           ADDB  CKSM      UPDATE CHECKSUM
A24F F7 A0 4E           STAB  CKSM
A252 39                 RTS
A253 8D 1B      INH300  BSR   CIN300    GET CHAR
A255 80 30              SUBA  #$30      CONVERT TO HEX
A257 2B 0F              BMI   C2        NOT HEX
A259 81 09              CMPA  #9
A25B 2F 0A              BLE   IN1H      DONE
A25D 81 11              CMPA  #$11
A25F 2B 07              BMI   C2        NOT HEX
A261 81 16              CMPA  #$16
A263 2E 03              BGT   C2        NOT HEX
A265 80 07              SUBA  #7
A267 39         IN1H    RTS
A268 86 58      C2      LDAA  #'X       PRINT "X"
A26A BD E1 D1           JSR   OUTEEE
A26D 20 C1              BRA   C1        QUIT

*300 BAUD INPUT ROUTINE BY J. RUSSELL LEMON

A270 37         CIN300  PSHB            SAVE
A271 FF A0 4A           STX   T1        SAVE
A274 C6 0C              LDAB  #$C
A276 BD E1 7B           JSR   LDTMP8    X=PIADA=$F400
A279 C6 08      KC1     LDAB  #8        BIT COUNT
A27B F7 A0 4C           STAB  TC
A27E 3E                 WAI             WAIT FOR CLOCK
A27F 69 02              ROL   2,X
A281 46                 RORA
A282 81 AA              CMPA  #$AA      LOOK FOR MARK
A284 26 F8              BNE   KC2
A286 C6 08      KC3     LDAB  #8        CYCLE COUNT
A288 F7 A0 60           STAB  T3
A28B 3E         KC4     WAI             WAIT FOR CLOCK
A28C 69 02              ROL   2,X
A28E 46                 RORA
A28F 7A A0 60           DEC   T3        8 CYCLES
A292 26 F7              BNE   KC4
A294 0C                 CLC

*THE FOLLOWING BYTE MAY BE $4C OR $4D,
DEPENDING UPON THE POLARITY OF THE CASSETTE

A295 4C (or 4D)         INCA  (or TSTA)
A296 26 01              BNE    *+1
A298 0D                 SEC
A299 76 A0 61           ROR   TT4
A29C 5F                 CLRB
A29D 7A A0 4C           DEC   T2
A2A0 26 E4              BNE   KC3
A2A2 B6 A0 61           LDAA  TT4
A2A5 84 7F              ANDA  #$7F      MASK MSB
A2A7 B7 F2 00           STAA  $F200     LOOK AT CHAR ON CRT
A2AA 7C A2 A7           INC   $A2A7     ADVANCE POINTER
A2AD 33                 PULB
A2AE FE A0 4A           LDX   T1
A2B1 39                 RTS
```

THE MICRODASYS MONBUG MONITOR USER'S GUIDE

REVISION 2.3A
SEPTEMBER 25, 1978

********** INTRODUCTION **********

YOUR MD-690A MICROCOMPUTER COMES WITH A 1024 BYTE MONITOR PROGRAM CALLED MONBUG (COPYRIGHT 1978 BY MICRODASYS). THE MONITOR HAS TWO MAJOR USES IN A COMPUTER SYSTEM. FIRST, IT ALLOWS THE USER TO GAIN CONTROL OF THE PROCESSOR, BY FORMATTING THE I/O PORTS AND INITIALIZING ANY REQUIRED CONSTANTS. IN THIS ROLE, THE MONITOR RESPONDS TO THE USER'S COMMANDS AND ENABLES HIM TO ENTER AND RUN HIS OWN PROGRAMS. THE SECOND FUNCTION OF THE MONITOR IS TO PROVIDE USEFUL SUBROUTINES WHICH THE USER CAN CALL FROM HIS OWN PROGRAMS. THE MOST IMPORTANT OF THESE SUBROUTINES ARE THE INPUT AND OUTPUT ROUTINES WHICH ENABLE THE USER TO "TALK" WITH HIS COMPUTER. THE 6800 PROCESSOR HAS THE UNIQUE ADVANTAGE THAT MOST MANUFACTURERS OF 6800 COMPUTERS HAVE USED THE SAME MONITOR PROGRAM, MIKBUG (REG. TRADEMARK OF MOTOROLA, INC.). THIS MEANS THAT PROGRAMS WRITTEN FOR ONE 6800 SYSTEM WILL RUN ON OTHER SYSTEMS. BUT BECAUSE MIKBUG WAS WRITTEN TO INTERFACE WITH TELETYPES AND RELATIVELY SLOW SERIAL I/O TERMINALS, IT IS NOT VERY POWERFUL OR EFFICIENT. WE HAVE SOLVED THIS PROBLEM BY WRITING A MONITOR WHICH LOCATES MOST OF THE MIKBUG SUBROUTINES IN THE SAME LOCATION IN MEMORY, AND YET WHICH RUNS MUCH FASTER BECAUSE IT INTERFACES WITH INTERRUPT DRIVEN KEYBOARDS AND MEMPORY-MAPPED VIDEO DISPLAYS. THIS PROVIDES FOR POWERFUL TECHNIQUES LIKE ANIMATION, MULTI-TASKING AND MULTI-USER SYSTEMS. IN ADDITION, MONBUG ADDS MANY SUBROUTINES NOT FOUND IN MIKBUG. THE DISCUSSION THAT FOLLOWS IS IN THREE PARTS. THE FIRST PART IS A GENERAL DESCRIPTION OF MONBUG, INCLUDING A LISTING OF THE MEMORY LOCATIONS WHICH IT USES IN THE RAM (RANDOM ACCESS MEMORY) LOCATED FROM $A000 THROUGH $A3FF. SECOND IS A DESCRIPTION OF HOW TO USE THE MONITOR IN THE COMMAND MODE FROM THE TIME THAT POWER IS APPLIED. THIRD IS A DESCRIPTION OF THE SUBROUTINES WHICH MAY BE CALLED FROM USER PROGRAMS. THE SUBROUTINES WHICH ARE MIKBUG COMPATIBLE ARE INDICATED BY AN ASTERISK (*). AND FOURTH ARE SOME EXAMPLES OF THE USE OF THE MONITOR TO ENTER, SAVE AND RUN PROGRAMS. EXAMPLE PROGRAMS ARE INCLUDED. FOR FURTHER INFORMATION ON THE OPERATION OF THE 6800 MICROPROCESSOR IN PARTICULAR, OR MICROCOMPUTERS IN GENERAL, WE RECOMMEND THE FOLLOWING REFERENCES, AVAILABLE FROM MICRODASYS OR YOUR LOCAL COMPUTER STORE:

AN INTRODUCTION TO MICROCOMPUTING, VOLUME 0 - THE BEGINNER'S BOOK
BY ADAM OSBORNE, PUBLISHED BY ADAM OSBORNE & ASSOCIATES, 1977
SOFTCOVER, $7.50

USING THE 6800 MICROPROCESSOR
BY ELMER POE, PUBLISHED BY HOWARD W. SAMS & CO., 1978
SOFTCOVER, $6.95

M6800 MICROPROGRAMMING MANUAL
PUBLISHED BY MOTOROLA INC., 1975
SOFTCOVER, $3.00

AN INTRODUCTION TO MICROCOMPUTING, VOLUME 1 - BASIC CONCEPTS
BY ADAM OSBORNE, PUBLISHED BY ADAM OSBORNE & ASSOCIATES, 1976
SOFTCOVER, $7.50

********** ABOUT MONBUG **********

WHEN POWER IS FIRST APPLIED TO YOUR PROCESSOR BOARD, MONBUG INITIALIZES THE MICROPROCESSOR AND MEMORY, CLEARS THE VIDEO SCREEN AND PRINTS AN ASTERISK. THE WHITE BLOCK AFTER THE ASTERISK IS THE 'CURSOR'. IT TELLS YOU WHERE CHARACTERS WILL APPEAR WHEN YOU TYPE. THE CURSOR DOES NOT HAVE TO BE THERE, BUT THE MONITOR PUTS IT THERE FOR YOUR CONVENIENCE. AT THIS POINT THE MONITOR IS READY TO RESPOND TO FIVE COMMANDS. EACH COMMAND IS GIVEN BY TYPING A SINGLE CHARACTER, SOMETIMES FOLLOWED BY NUMBERS.
IF AN ERROR IS MADE IN THE FORMAT OF A COMMAND THE PROCESSOR WILL SIMPLY PRINT ANOTHER ASTERISK PROMPT.
AN IMPORTANT POINT TO REMEMBER IS THAT IF CONTROL OF THE PROCESSOR IS EVER LOST IT CAN ALWAYS BE REGAINED SIMPLY BY HITTING THE RESET KEY.
THE COMMANDS OF THE MONITOR WILL NOW BE DISCUSSED IN DETAIL.

NOTES:

$ = HEXADECIMAL NUMBER FOLLOWS (BASE SIXTEEN).
^ = CONTROL KEY PRESSED SIMULTANEOUSLY WITH THE FOLLOWING KEY.

MONBUG USES THE FOLLOWING AREAS OF MEMORY FOR TEMPORARY STORAGE:

NAME	LOCATION	USE
IOV	$A000,01	IRQ VECTOR FOR KEYBOARD SERVICING
BEGA	$A002,03	STARTING ADDRESS OF DUMPS, PUNCHES
ENDA	$A004,05	ENDING ADDRESS OF DUMPS, PUNCHES
NIO	$A006,07	NMI VECTOR
SP	$A008,09	STORAGE FOR USER'S STACK POINTER
CBUF	$A00A	KEYBOARD INPUT CHARACTERS RETURNED HERE
TEMP	$A00B	TEMPORARY VARIABLE STORAGE
XHI	$A00C	TEMPORARY STORAGE FOR X REGISTER'S MSB
XLO	$A00D	TEMPORARY STORAGE FOR X REGISTER'S LSB
NWSTRT	$A00E,0F	TEMPORARY 16 BIT STORAGE AND COUNTER
CURSOR	$A010,11	LOCATION OF CURSOR ALWAYS STORED HERE
TEMPB	$A012	TEMPORARY VARIABLE STORAGE
TEMPX	$A013,14	TEMPORARY STORAGE FOR X
XTEMP	$A015,16	TEMPORARY STORAGE FOR X
CASTMP	$A017	TEMPORARY STORAGE DURING PUNCH ROUTINE
ROLORG	$A018,19	WORKING REGISTERS FOR SCROLLING SCREEN
ROLDES	$A01A,1B	WORKING REGISTERS FOR SCROLLING SCREEN

THE STACK POINTER IS INITIALIZED TO $A042, AND THE STACK GROWS DOWNWARD FROM THIS POINT. IT IS THEREFORE UNADVISABLE TO USE THE MEMORY LOCATIONS FROM $A01C THROUGH $A042 FOR ANYTHING ELSE. ALSO, THE USER REGISTERS ARE STORED IMMEDIATELY ABOVE THE STACK POINTER FROM $A043 THROUGH $A049. THESE LOCATIONS SHOULD NOT BE USED UNLESS THE USER HAS CHANGED HIS STACK POINTER TO ANOTHER LOCATION. THE REMAINING 950 BYTES OF RAM FROM $A04A THROUGH $A3FF ARE AVAILABLE TO THE USER FOR ANY DESIRED PURPOSE.

********** COMMANDS **********

L -- LOAD A CASSETTE TAPE

WHEN L IS TYPED THE PROCESSOR WILL WAIT FOR A CASSETTE TAPE TO BE PLAYED FROM THE TAPE RECORDER THROUGH THE 2400 BAUD CASSETTE INTERFACE. IF THE PROPER FORMAT IS ENCOUNTERED ON THE TAPE, DATA WILL BE LOADED INTO MEMORY. THIS DATA COULD BE A PROGRAM WHICH THE USER COULD THEN EXECUTE. THIS IS HOW BASIC COULD BE LOADED. WHEN THE TAPE IS FINISHED THE PROCESSOR WILL NOT COME OUT OF THE LOAD MODE UNLESS SPECIAL CHARACTERS ($ECB9) HAVE BEEN RECORDED AT THE END. NORMALLY IT WILL SIT AND WAIT FOR THE NEXT TAPE. HOWEVER CONTROL CAN BE REGAINED BY HITTING RESET. REMEMBER-- NO MATTER WHAT HAPPENS, CONTROL WILL ALWAYS BE RETURNED TO THE MONITOR BY HITTING RESET. IF THERE IS AN ERROR ON THE TAPE A QUESTION MARK (?) WILL BE PRINTED ON THE SCREEN AND CONTROL WILL RETURN TO THE MONITOR. USING THE MICRODASYS 2400 BAUD CASSETTE INTERFACE AND MONBUG FORMATTED TAPE, IT REQUIRES ONLY 19 SECONDS TO LOAD 4096 (4K) BYTES OF MEMORY.

P -- RECORD (PUNCH) A FORMATTED TAPE

WHEN THE COMMAND P IS HIT TWO HEXADECIMAL ADDRESSES MUST BE INPUT. FOR EXAMPLE: P 004C 3B2E
A CASSETTE TAPE SHOULD BE RUNNING IN THE RECORD MODE BEFORE THE LAST DIGIT IS TYPED. ALL THE DATA WILL BE RECORDED FROM (IN THIS EXAMPLE) $004C TO $3B2E. EACH RECORD EXCEPT THE LAST WILL BE 70 BYTES LONG AND IN THE FOLLOWING FORMAT:

$FF (FOR PLAYBACK SYNCRONIZATION)
AABCCDDDDDD...DDDDDDDDDDDDE

WHERE:
- AA IS THE START CODE $EC9D (2 BYTES)
- B IS THE NUMBER OF BYTES IN THIS RECORD
- CC IS THE STARTING ADDRESS OF THIS RECORD
- D IS A DATA BYTE (UP TO 64 BYTES PER RECORD)
- E IS A CHECKSUM.

THIS DATA MAY BE RELOADED AT ANY TIME USING THE LOAD COMMAND.

M -- MEMORY/REGISTER EXAMINE AND CHANGE

WHEN M IS TYPED, FOLLOWED BY AN ADDRESS, THE SCREEN DISPLAYS THE VALUES THAT WILL BE USED FOR THE PROCESSOR'S REGISTERS WHEN THE USER'S PROGRAM IS EXECUTED. (THESE ARE SIMPLY THE CONTENTS OF MEMORY ABOVE THE USER'S STACK POINTER. AN RTI STATEMENT LOADS THESE INTO THE INTERNAL REGISTERS WHEN THE 'G' COMMAND IS HIT - SEE BELOW. THE USER'S STACK POINTER IS STORED IN $A008, BUT IS INITIALIZED TO THE SAME VALUE AS MONBUG'S STACK POINTER ($A042) EVERYTIME RESET IS HIT.) THE ORDER IN WHICH THE REGISTERS ARE DISPLAYED ARE AS FOLLOWS:

SP CC B A X PC

FOLLOWING THIS, 192 BYTES OF MEMORY ARE DISPLAYED STARTING AT THE ADDRESS INPUT FOLLOWING THE COMMAND. THE USER MAY THEN MOVE THE CURSOR AROUND THE SCREEN CHANGING ANY DATA HE WISHES, EVEN THE INTERNAL REGISTERS. (ADDRESSES MAY NOT BE CHANGED.) NOTE THAT A CARRIAGE RETURN CAUSES AN AUTOMATIC LINE FEED AND TAB, PREPARING FOR DATA ENTRY TO THE NEXT LINE. AT THIS POINT, ANY NEW DATA HAS NOT AUTOMATICALLY BEEN ENTERED INTO MEMORY. FIRST, A CONTROL C MUST BE HIT. THE CURSOR WILL THEN STOP AT ANY ERRORS. IF ERRORS ARE NOTICED BEFORE ^C IS HIT THEY MAY EITHER BE CORRECTED OR THE ENTIRE SCREEN MAY BE 'SCRATCHED' BY SIMPLY TYPING ^Z. BECAUSE OF THE WAY THAT THE MONITOR OPERATES, MEMORY CANNOT BE CHANGED WHILE LOCATIONS $A002 THROUGH $A00F ARE DISPLAYED. THESE LOCATIONS ARE USED BY THE CHANGE ROUTINE ITSELF.

G -- EXECUTE (GO TO) USER'S PROGRAM

WHEN G IS TYPED THE PROCESSOR LOADS ITS REGISTERS FROM THE USER STACK AND BEGINS EXECUTION AT THE LOCATION SPECIFIED BY THE PROGRAM COUNTER. BEFORE G IS HIT THE PROGRAM COUNTER SHOULD BE INITIALIZED USING THE CHANGE ROUTINE. ALTERNATELY, IF THE USER LEAVES THE STACK POINTER AT $A042, AS INITIALIZED BY MONBUG, THE PROGRAM COUNTER WILL ALWAYS CORRESPOND TO LOCATIONS $A048,49 AND THE DESIRED VALUE MAY BE ENTERED THERE DIRECTLY USING THE CHANGE FUNCTION. A PROGRAM ON CASSETTE TAPE CAN ALSO LOAD THE PROGRAM COUNTER (AND THE USER STACK POINTER, IF DESIRED) WITH THE APPROPRIATE VALUE. THEN HITTING G WILL AUTOMATICALLY EXECUTE THE PROGRAM.

H -- EXECUTE USER'S PROGRAM, DIRECTED

TYPING H FOLLOWED BY AN ADDRESS CAUSES PROGRAM EXECUTION TO BEGIN AT THAT ADDRESS. THE PROGRAM COUNTER IS AUTOMATICALLY STORED IN THE APPROPRIATE SPOT ABOVE THE USER'S STACK POINTER BEFORE THE RTI IS EXECUTED.

********** MEMORY MAP **********

THE FOLLOWING IS AN ITEMIZATION OF THE MEMORY USAGE FOR THE 64K ADDRESS SPACE OF THE MD-690A:

$0000-$9FFF	40K USER RAM
$A000-$A049	74 BYTES MONBUG RAM (SUPPLIED ON MD-690A)
$A04A-$A3FF	950 BYTES USER RAM (SUPPLIED ON MD-690A)
$A400-$BFFF	7K USER RAM
$C000-$DFFF	8K USER PROM SPACE (ON MD-690A) (MAY BE USED FOR RAM)
$E000-$E3FF	MONBUG (1K 2708 OR FIRST HALF OF 2K 2716)
$E400-$E7FF	TEXTED (OPTIONAL SECOND HALF OF 2716)
$E800-$EFFF	I/O SIGNALS GENERATED HERE INSTEAD OF MEMORY
$F000-$F3FF	MEMORY MAPPED VIDEO GRAPHICS BOARD
$F400-$F403	PIA (KBD, CASSETTE, SPARE I/O PORTS)
$F404-$F7FF	NOT USED
$F800-$FFF7	REPEAT OF MONBUG/TEXTED
$FFF8-$FFFF	INTERRUPT AND RESTART SERVICE VECTORS

********** PIA BIT USAGE **********

PA0-PA7 KEYBOARD DATA

CA1 KEYBOARD STROBE
CA2 READER CONTROL

PB0 CASSETTE OUTPUT DATA
PB1-PB6 AVAILABLE I/O
PB7 CASSETTE INPUT DATA

CB1 CASSETTE TRANSMIT CLOCK
CB2 CASSETTE RECEIVE CLOCK

********** SUBROUTINES **********

THE FOLLOWING IS A SUMMARY OF ALL USEFUL SUBROUTINES CONTAINED IN MONBUG. EXCEPT AS NOTED, THESE SUBROUTINES MAY BE CALLED FROM USER PROGRAMS WITH A JSR STATEMENT. UPON COMPLETION THEY WILL RETURN TO THE NEXT STATEMENT IN THE USER PROGRAM.

NOTES: UNLESS OTHERWISE SPECIFIED, ALL INPUTS ARE FROM AN INTERRUPT INTERRUPT DRIVEN KEYBOARD VIA INEEE AND ALL OUTPUTS ARE TO A MEMORY-MAPPED VIDEO BOARD AT $F000 VIA OUTEEE.
EOT = END OF TRANSMISSION CHARACTER ($04).
* = MIKBUG COMPATIBLE MEMORY LOCATION.
M(ADDR) = CONTENTES OF THE MEMORY LOCATION POINTED TO BY THE CONTENTS OF LOCATIONS ADDR AND ADDR+1.

IRQ($E000) *
NOT A SUBROUTINE. JUMP TO LOCATION POINTED TO BY INTERRUPT REQUEST VECTOR, IRQ($A000). DESTROYS X.

NMI($E005) *
NOT A SUBROUTINE. JUMP TO LOCATION POINTED TO BY THE NONMASKABLE INTERRUPT VECTOR, NMI($A006). DESTROYS X.

LOAD($E00A) *
LOADS A MONBUG FORMATTED CASSETTE TAPE TO ITS ORIGINAL LOCATION IN MEMORY. CASIN TURNS THE READER CONTROL ON AT THE START OF THE LOAD. THE PROCESSOR REMAINS IN THE LOAD MODE UNTIL IT ENCOUNTERS AN END OF FILE CODE ($ECB9) (OUTSIDE OF A RECORD, OF COURSE) OR IS RESET. THIS TURNS OFF THE READER CONTROL. DESTROYS A,B AND X.

BADDRS($E047) *
BUILD ADDRESS FROM 4 SUCESSIVE KEYBOARD ENTRIES AND STORE ADDRESS IN TEMPX,TEMPX+1($A013,14). PRINT THE ADDRESS FOLLOWED BY A SPACE ON THE SCREEN. IF EITHER CHARACTER IS NOT HEX, CONTROL RETURNS TO THE MONITOR. X IS LOADED WITH THE ADDRESS. DESTROYS A.

BYTE($E055) *
INPUT 2 HEX CHARACTERS FROM KEYBOARD AND PRINT ON SCREEN. THE FIRST HEX CHARACTER READ IS STORED IN THE MOST SIGNIFICANT HALF OF ACCUMULATOR A, THE SECOND HEX CHARACTER READ IN IS STORED IN THE LEAST SIGNIFICANT HALF OF ACCUMULATOR A. IF EITHER CHARACTER IS NOT HEX, CONTROL RETURNS TO THE MONITOR. DESTROYS B.

OUTHL($E067) *
PRINT THE MOST SIGNIFICANT HALF OF ACCUMULATOR A AS AN ASCII INTERPRETATION OF THE HEX DIGIT. DESTROYS A.

OUTHR($E06B) *
PRINTS THE LEAST SIGNIFICANT HALF OF ACCUMULATOR A AS AN ASCII INTERPRETATION OF THE HEX DIGIT. DESTROYS A.

OUTCH($E075) *
JUMPS TO OUTEEE($E1D1).

INCH($E078) *
JUMPS TO INEEE($E1AC).

PDATA2($E07B) *
PRINTS THE ASCII CONTENTS OF A, THEN PRINTS AN ASCII STRING STARTING AT THE MEMORY LOCATION POINTED TO BY THE INDEX REGISTER +1. EXITS WHEN AN EOT($04) IS ENCOUNTERED. ON EXITING X POINTS TO THE EOT. DESTROYS A.

PDATA1($E07E) *
PRINTS A STRING OF ASCII STARTING AT THE LOCATION POINTED TO BY THE INDEX REGISTER. EXITS WHEN AN EOT($04) IS ENCOUNTERED. ON EXITING X POINTS TO THE EOT. DESTROYS A.

PSTACK($E097)
PRINTS THE CONTENTS OF THE USER REGISTERS ABOVE THE STACK POINTER. THE REGISTERS ARE PRINTED IN THE FOLLOWING ORDER: CONDITION CODES, ACCUMULATOR B, ACCUMULATOR A, INDEX REGISTER, PROGRAM COUNTER AND STACK POINTER. DESTROYS A.

INHEX($E0AA) *
INPUTS A CHARACTER FROM THE KEYBOARD AND DISPLAYS IT ON THE SCREEN. ON EXITING, THE HEX EQUIVALENT OF THE CHARACTER INPUT IS IN THE LEAST SIGNIFICANT HALF OF ACCUMULATOR A. IF THE CHARACTER INPUT IS NOT HEX, CONTROL RETURNS TO THE MONITOR.

OUT2H($E0BF) *
OUTPUTS CONTENTS OF THE LOCATION POINTED TO BY THE INDEX REGISTERS AS TWO ASCII INTERPRETTED HEX CHARACTERS. INCREMENTS X. DESTROYS A.

OUT4HS($E0C8) *
OUTPUTS CONTENTS OF TWO BYTES POINTED TO BY X AND X+1 AS FOUR ASCII INTERPRETTED CHARACTERS, THEN OUTPUTS A SPACE. INCREMENTS X BY 2. DESTROYS A.

OUT2HS($E0CA) *
SAME AS OUT2H, THEN OUTPUTS A SPACE. INCREMENTS X. DESTROYS A.

OUTS($E0CC) *
OUTPUTS A SPACE TO THE SCREEN. DESTROYS A.

START($E0D0) *
NOT A SUBROUTINE. RESET EXECUTION BEGINS HERE. THIS IS THE START OF THE MONITOR. THE USER STACK POINTER IS INTITIALIZED TO $A042, THE PIA IS INITIALIZED, AND THE SCREEN IS CLEARED. THE KEYBOARD INTERRUPT SERVICE VECTOR IS INITIALIZED TO INPUT($E3E3). IF THERE IS A PROM AT LOCATION $C000 BEGINNING WITH $00 EXECUTION BEGINS THERE. OTHERWISE, THE MONITOR WAITS FOR A COMMAND.

PINIT($E0D6)
NOT A SUBROUTINE. SAME AS START EXCEPT THE USER'S STACK POINTER IS NOT ALTERED.

ERSCN($E0DF)
NOT A SUBROUTINE. SAME AS PINIT EXCEPT THE PIA IS NOT INITIALIZED.

CONTRL($E0E3) *
NOT A SUBROUTINE. SAME AS ERSCN EXCEPT THE SCREEN IS NOT ERASED.

PRMTST($E0E9)
NOT A SUBROUTINE. SAME AS CONTRL EXCEPT THE KEYBOARD SERVICE ROUTINE IS NOT INITIALIZED.

KNTRL($E0F2)
NOT A SUBROUTINE. SAME AS PRMTST EXCEPT NO CHECK IS MADE FOR A PROM AT $C000. THE MONITOR WAITS FOR A COMMAND.

CMNDIN($E0FE)
THE PROCESSOR GETS A COMMAND FROM THE KEYBOARD AND EXECUTES IT. IF AN UNDEFINED COMMAND IS INPUT, TWO ADDITIONAL ADDRESSES ARE INPUT. UPON RETURNING TO THE CALLING PROGRAM THIS DATA IS CONTAINED IN BEGA($A002,03) AND ENDA($A004,05).

PUNCH($E12E)
A MONBUG FORMATTED 2400 BAUD TAPE IS RECORDED CONTAINING DATA STARTING AT M(BEGA) AND ENDING AT M(ENDA). THE READER CONTROL WILL BE ON DURING THE PUNCH. DESTROYS A,B AND X.

CASOUT($E154)
TURNS THE READER CONTROL ON (DOES NOT TURN IT OFF LATER). OUTPUTS THE CONTENTS OF ACCUMULATOR A TO THE 2400 BAUD CASSETTE INTERFACE. THE PIA OUTPUT IS SERIAL DIGITAL DATA PRECEDED BY A LOW START BIT AND FOLLOWED BY A HIGH STOP BIT. THE CASSETTE INTERFACE THEN ENCODES IT IN MANCHESTER FORMAT. BECAUSE SOFTWARE IS USED TO SHIFT OUT THE DATA, THE EXECUTION TIME OF THIS SUBROUTINE IS LONGER THAN MOST, APPROXIMATELY 10/2400 SECONDS, OR THE LENGTH OF TIME NEEDED TO WAIT FOR 10 CASSETTE CLOCK CYCLES. DESTROYS A.

LDTMP8($E17B)
SAVES X IN XHI($A00C), LOADS X WITH PIADA($F400), TURNS THE READER CONTROL ON, ENABLES THE TX OR RX CLOCK INTERRUPT AS DETERMINED BY B, SETS UP ACCESS TO B DATA REGISTER OF PIA, LOADS B AND TEMP($A00B) WITH THE BIT COUNTER $08. CHANGES THE INTERRUPT SERVICE ROUTINE TO INTRET($E1C9).

CASIN($E197)
INPUTS ONE CHARACTER FROM CASSETTE TAPE TO ACCUMULATOR A. THE ROUTINE WILL WAIT FOR A VALID START BIT (LOGIC 0). THE CASSETTE INTERFACE IS ASSUMED TO BE PROPERLY CALIBRATED.

INEEE($E1AC) *
DISPLAYS CURSOR ON SCREEN AND WAITS FOR INTERRUPT. IF THE INTERRUPT VECTOR AT $A000 HAS NOT BEEN CHANGED, WHEN THE INTERRUPT OCCURS DATA IS READ FROM THE KEYBOARD, THE PARITY BIT IS MASKED AND THE DATA IS PRINTED. THE INPUT DATA IS RETURNED IN ACCUMULATOR A AND CBUF($A00A).

XSAVE($E1C2)
X IS STORED IN XTEMP($A015) AND THEN LOADED FROM THE CURSOR HOLDING REGISTER ($A010).

LDC($E1C5)
X IS LOADED FROM THE CURSOR HOLDING REGISTER.

INTRET($E1C9)
NOT A SUBROUTINE. UPON ARRIVAL OF CASSETTE TX OR RC CLOCK CLEARS FLAG IN PIA STATUS BY READING PIA AND RETURNS.

WHAT($E1CF)
PRINTS A '?' ON THE SCREEN AT THE CURSOR LOCATION INDICATED BY $A010. DESTROYS A.

OUTEEE($E1D1) *
DETERMINES IF THE CONTENTS OF A IS ONE OF THE FOLLOWING COMMANDS:

CURSOR LEFT (BACKSPACE)	^H	$08
CURSOR RIGHT	^L	$0C
CURSOR UP	^K	$0B
CURSOR DOWN (LINE FEED)	^J	$0A
CARRIAGE RETURN	^M	$0D
HORIZONTAL TAB	^I	$09
HOME UP	^^	$1E
ERASE TO END OF LINE	^N	$0E
ERASE TO END OF SCREEN	^E	$05

IF ONE OF THESE COMMANDS IS DETECTED, THE APPROPRIATE ACTION IS PERFORMED AND THE CURSOR REGISTER ($A010) IS UPDATED ACCORDINGLY. OTHERWISE, IF THE CONTENTS OF A IS NOT A CONTROL CHARACTER, IT IS PRINTED ON THE SCREEN AT THE LOCATION POINTED TO BY THE CURSOR REGISTER AND THE CURSOR REGISTER IS INCREMENTED. THE CURSOR CANNOT BE MOVED OFF THE TOP OF THE SCREEN, AND IF IT MOVES OFF THE BOTTOM, THE SCREEN WILL BE SCROLLED UP ONE LINE. DESTROYS A.

OUTABP($E256)
THE CURSOR IS MOVED TO THE NEXT MULTIPLE OF 8 LOCATION ACROSS THE SCREEN. DESTROYS A. LOADS X FROM XHI ($A00C).

INCLINP($E27F)
MOVES THE LOCATION POINTED TO BY THE CURSOR REGISTER DOWN ONE LINE. DESTROYS A. LOADS X FROM XHI ($A00C).

CLRSCN($E2A4)
ERASES SCREEN. DESTROYS A.

CHANGE($E2AA)
DISPLAYS THE USER REGISTERS ON TOP OF THE STACK AND 192 BYTES OF DATA. DATA IS DISPLAYED IN A TWELVE LINE BY 16 BYTE FORMAT PRECEDED BY THE ADDRESS. THE DATA BEGINS WITH THE ADDRESS POINTED TO BY BEGA ($A002). ALL DATA ON THE SCREEN MAY BE ALTERED. TO ENTER THE NEW DATA TYPE CONTROL C (^C). THE CURSOR WILL STOP AT ANY ERRORS. TO ESCAPE TO THE MONITOR WITHOUT THE NEW DATA BEING ENTERED, TYPE ^Z. DESTROYS A,B AND X.

HTEST($E361)
THE CONTENTS OF ACCUMULATOR A ARE CONVERTED FROM AN ASCII REPRESENTATION OF A HEXADECIMAL NUMBER TO THE HEXADECIMAL EQUIVALENT. THIS IS RETURNED IN THE LEAST SIGNIFICANT HALF OF A. IF A IS A VALID HEX EQUIVALENT THE CARRY IS CLEARED. OTHERWISE IT IS SET.

MCL1($E379)
NOT A SUBROUTINE. AN ASCII CHARACTER STRING CONSISTING OF HOME UP($), ERASE TO END OF SCREEN($05), EOT($04).

MCL($E37C)
NOT A SUBROUTINE. AN ASCII CHARACTER STRING CONSISTING OF CARRIAGE RETURN($0D), LINE FEED($0A), ERASE TO END OF LINE($0C), ASTERISK($2A), EOT($04).

DUMP($E3AD)
OUTPUTS DATA TO THE SCREEN. THE BEGINNING ADDRESS OF THE DATA IS THE CONTENTS OF $A002-3 AND THE ENDING ADDRESS IS THE CONTENTS OF $A004-5 MINUS ONE. THE DISPLAY FORMAT IS AN ADDRESS FOLLOWED BY 16 BYTES OF DATA. DESTROYS A. ON EXITING, X POINTS TO THE NEXT DATA LOCATION.

EDGE($E3C2)
OUTPUTS A CARRIAGE RETURN, LINE FEED, ERASE TO END OF LINE, ASTERISK, AND FOUR HEX CHARACTERS REPRESENTING THE CONTENTS OF THE INDEX REGISTER. THEN TABS OVER ONE COLUMN. DESTROYS A.

INCHAR($E3DA)
LOADS ACCUMULATOR A WITH THE DATA FROM THE A SIDE OF THE PIA LOCATED AT $F400, MASKS THE MSB AND STORES IN CBUF($A00A).

INPUT($E3E3)
NOT A SUBROUTINE. INTERRUPT SERVICE ROUTINE FOR THE KEYBOARD. READS THE KEYBOARD USING INCHAR ($E3DA). STORES B AT THE LOCATION POINTED TO BY THE CURSOR HOLDING REGISTER ($A010). (B IS PRESUMABLY THE CHARACTER THAT WAS REPLACED BY CBLOCK ($7F) DURING EXECUTION OF INEEE). OUTPUT THE CHARACTER TO THE SCREEN. THE LAST STATEMENT IS RTI. DESTROYS A,B,X. THE INPUT CHARACTER IS RETURNED IN CBUF($A00A).

TIMER0($E3EE)
GENERATES A DELAY OF .1 SEC TIMES THE NUMBER IN ACCUMULATOR B. (FOR A 2MHZ PROCESSOR THE DELAY IS .05 SEC TIMES B.) DESTROYS B AND X.

********** USING MONBUG **********

IF YOU'VE JUST TURNED YOUR SYSTEM ON, THE MONITOR SHOULD HAVE PRINTED AN ASTERISK PROMPT AND BE WAITING FOR A COMMAND. IF IT'S NOT, IT'S EITHER BROKEN OR YOU'LL HAVE TO HIT RESET. ONCE YOU'VE GOT THE PROMPT CHARACIER IT'S EASY TO TELL YOUR PROCESSOR WHAT TO DO. LET'S TRY TO RUN A PROGRAM. HERE IS AN EXAMPLE OF WHAT AN ASSEMBLY LANGUAGE LISTING LOOKS LIKE:

```
00001                          NAM    SEARCH
00002 A050                     ORG    $A050
00003                          *SEARCH - A PROGRAM FOR SEARCHING MEMORY LOCATIONS
00004                          *FOR A PARTICULAR BYTE.  EXECUTION BEGINS AT $A050
00005                          *
00006                          *COPYRIGHT 1978 BY MICRODASYS

00008 0004          EOT    EQU    4
00009 0000          MEM1   EQU    0
00010 0002          MEM2   EQU    2
00011 0004          DATA   EQU    4
00012 E047          BADDRS EQU    $E047
00013 E055          BYTE   EQU    $E055
00014 E07E          PDATA1 EQU    $E07E
00015 E0E3          CONTRL EQU    $E0E3
00016 E3C2          EDGE   EQU    $E3C2

00018 A050 CE A086  TOP    LDX    #STRNG1   LOAD X WITH POINTER TO 1ST STRING
00019 A053 BD E07E         JSR    PDATA1    PRINT STRING
00020 A056 BD E055         JSR    BYTE      INPUT BYTE TO BE SEARCHED FOR
00021 A059 97 04           STA A  DATA      TEMPORARYILY SAVE DATA
00022 A05B CE A091         LDX    #STRNG2    LOAD X WITH POINTER TO 2ND STRING
00023 A05E BD E07E         JSR    PDATA1    PRINT STRING
00024 A061 BD E047         JSR    BADDRS    INPUT STARTING ADDRESS OF SEARCH
00025 A064 DF 00           STX    MEM1      STORE STARTING ADDRESS IN MEMORY
00026 A066 CE A099         LDX    #STRNG3    LOAD X WITH POINTER TO 3RD STRING
00027 A069 BD E07E         JSR    PDATA1    PRINT STRING
00028 A06C BD E047         JSR    BADDRS    INPUT ENDING ADDRESS
00029 A06F DF 02           STX    MEM2      STORE ENDING ADDRESS IN MEMORY
00030 A071 DE 00           LDX    MEM1      LOAD X WITH STARTING ADDRESS
00031 A073 A6 00    SRCH   LDA A  X         LOAD A WITH DATA AT THAT ADDRESS
00032 A075 91 04           CMP A  DATA      COMPARE A WITH THE SEARCH DATA
00033 A077 26 03           BNE    NOPE      IF NOT FOUND DON'T PRINT ADDRESS
00034 A079 BD E3C2         JSR    EDGE      PRINT ADDRESS WHERE DATA FOUND
00035 A07C 9C 02    NOPE   CPX    MEM2      LAST ADDRESS REACHED?
00036 A07E 27 03           BEQ    DONE      YES, WE'RE FINISHED
00037 A080 08              INX              NO, LOOK AT NEXT ADDRESS
00038 A081 20 F0           BRA    SRCH
00039 A083 7E E0E3  DONE   JMP    CONTRL
```

```
00041 A086 53       STRNG1 FCC     +SRCH FOR? +
      A087 52
      A088 43
      A089 48
      A08A 20
      A08B 46
      A08C 4F
      A08D 52
      A08E 3F
      A08F 20
00042 A090 04              FCB     EOT
00043 A091 20       STRNG2 FCC     + FROM? +
      A092 46
      A093 52
      A094 4F
      A095 4D
      A096 3F
      A097 20
00044 A098 04              FCB     EOT
00045 A099 54       STRNG3 FCC     +TO? +
      A09A 4F
      A09B 3F
      A09C 20
00046 A09D 04              FCB     EOT

00048                      END


SYMBOL TABLE


EOT     0004  MEM1    0000  MEM2    0002  DATA    0004  BADDRS E047
BYTE    E055  PDATA1 E07E  CONTRL E0E3  EDGE    E3C2  TOP     A050
SRCH    A073  NOPE    A07C  DONE    A083  STRNG1 A086  STRNG2 A091
STRNG3 A099
```

THE PROGRAM IS WELL COMMENTED AND MAKES EXTENSIVE USE OF THE MONBUG SUBROUTINES JUST DESCRIBED. A RUN OF THE PROGRAM MIGHT LOOK LIKE THIS:

```
*H A050
SRCH FOR? 3B FROM? 1C57 TO? 29BF

1C66
1E7D
208A
25CC

*
```

THE H COMMAND AND THE ADDRESS A050 WERE TYPED BY THE USER TO TELL THE COMPUTER WHERE TO BEGIN EXECUTION OF THE PROGRAM. THE PROGRAM DIRECTS THE PROCESSOR TO ASK US WHAT TO SEARCH FOR AND OVER WHAT RANGE. THE USER TYPES APPROPRIATE RESPONSES TO THESE QUESTIONS AND THE PROCESSOR INFORMS US THAT IT HAS FOUND THE DATA WE ASKED FOR FOUR TIMES IN THE RANGE SPECIFIED BY PRINTING THE ADDRESSES WHERE THE DATA WAS FOUND. WITH ITS TASK COMPLETE, THE PROGRAM RETURNS CONTROL TO THE MONITOR. EXAMINE THE PROGRAM. SEE IF YOU CAN FIGURE OUT WHAT EACH STEP DOES. NOTICE HOW MUCH OF THE PROGRAM IS THE "OVERHEAD" OF ASKING THE USER QUESTIONS. WHICH LINES ACTUALLY PERFORM THE SEARCH?

LET'S ENTER THE PROGRAM AND RUN IT. THE ACTUAL DATA IS CONTAINED AFTER THE ADDRESS "FIELD" AND BEFORE THE LABLE FIELD. FOR EXAMPLE, IN THE FIRST LINE OF CODE:

```
 00018 A050 CE A086 TOP    LDX    #STRNG1   LOAD X WITH POINTER TO 1ST STRING
```

THE ADDRESS FIELD TELLS US THAT THE PROGRAM STARTS AT $A050. THE DATA AT THAT ADDRESS AND ALSO AT $A051 AND $A052 IS $CE,$A0,$86. ENTER THIS DATA USING THE MEMORY CHANGE COMMAND. TYPE:

```
*M A050
```

SINCE WE WANT TO START THE PROGRAM AT $A050. UNDER THE REGISTERS, THE FIRST LINE OF MEMORY SHOULD LOOK SOMETHING LIKE THIS:

```
*A050   XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
```

WHERE X COULD BE ANY HEXADECIMAL NUMBER. BY TYPING SPACES OR MOVING THE CURSOR AROUND CHANGE THE FIRST THREE ENTRIES TO:

```
*A050 CE A0 86 XX XX XX XX XX XX XX XX XX XX XX XX XX
```

NOW GO AHEAD AND ENTER THE REST OF THE PROGRAM. NOTICE THAT SOME LINES MAY ONLY CONTAIN ONE OR TWO DATA BYTES. JUST ENTER THEM ALL ONE AFTER THE OTHER. WHEN YOU FINISH THE SCREEN SHOULD LOOK LIKE THIS:

```
*A050 CE A0 86 BD E0 7E BD E0 55 97 04 CE A0 91 BD E0
*A060 7E BD E0 47 DF 00 CE A0 99 BD E0 7E BD E0 47 DF
*A070 02 DE 00 A6 00 91 04 26 03 BD E3 C2 9C 02 27 03
*A080 08 20 F0 7E E0 E3 53 52 43 48 20 46 4F 52 3F 20
*A090 04 20 46 52 4F 4D 3F 20 04 54 4F 3F 20 04 XX XX
```

ONCE AGAIN, X REPRESENTS ANY HEXADECIMAL NUMBER. CHECK YOUR ENTRIES. IF THE PROGRAM IS WRONG IT MAY DESTROY ITSELF WHEN YOU TRY TO RUN IT. NOW HIT CONTROL AND C AT THE SAME TIME. THIS TRANSFERS THE DATA WE JUST ENTERED TO MEMORY. IF THERE ARE ANY NON-HEXADECIMAL NUMBERS THE CURSOR WILL STOP BEFORE THEM TO ALLOW US TO CORRECT THEM. JUST TO BE ON THE SAFE SIDE, LETS RECORD IT ON TAPE. INSERT A CASSETTE AND START IT RECORDING. NOW TYPE:

```
*P A050 A09D
```

THIS RECORDS THE DATA FROM $A050 THROUGH $A09D ON TAPE. WHEN THE PROMPT COMES BACK IT IS DONE. THIS SMALL AMOUNT OF DATA ONLY TAKES AN INSTANT TO RECORD. STOP THE TAPE AND PUT IT AWAY FOR FUTURE USE. NOW LETS TRY TO RUN THE PROGRAM. TYPE:

```
*H A050
```

DOES IT WORK? CHECK IT BY TELLING IT TO SEARCH ITSELF FOR SOMETHING YOU KNOW IS THERE. DOES IT TELL YOU THERE IS A $4D AT $A095?

```
SCH FOR? 4D FROM? A050 TO? A09D

*A095

*
```

YOU CAN RUN THE PROGRAM AS MANY TIMES AS YOU LIKE BY SIMPLY RESPONDING TO THE PROMPT WITH H A050. OR, TO RUN THE PROGRAM OVER AND OVER, CHANGE THE JUMP AT THE END SO THAT INSTEAD OF GOING TO THE MONITOR IT STARTS OVER. THIS MEANS CHANGE $A084,85 FROM $E0E3 TO $A050. DO THIS USING THE MEMORY CHANGE ROUTINE:

```
*M A084
```

AND ENTER:

```
*A084 A0 50 XX XX XX XX XX ...
```

NOW WHEN YOU RUN THE PROGRAM IT WILL NEVER COME BACK TO THE MONITOR. TO GET OUT OF IT YOU MUST HIT RESET.

WHEN YOU TURN THE POWER OFF YOU LOSE ALL OF THE DATA STORED IN RAM, INCLUDING THE PROGRAM YOU JUST ENTERED.
THE NEXT TIME YOU WANT TO RUN THE SEARCH ROUTINE YOU CAN RELOAD IT USING THE L COMMAND. PLAY THE SECTION OF THE TAPE CONTAINING THE PROGRAM AND TYPE L. WHEN THE TAPE IS SAFELY PAST THE END OF THE PROGRAM HIT RESET AND YOU'RE READY TO GO. JUST TYPE:

```
*H A050
```

IF YOU'D LIKE THE TAPE TO AUTOMATICALLY COME OUT OF THE LOAD MODE YOU'LL HAVE TO RECORD THE SPECIAL CHARACTERS $ECB9 AFTER THE PROGRAM. YOU CAN DO THIS BY ENTERING THIS PROGRAM:

```
00001                          NAM     EOF
00002 A100                     ORG     $A100

00004                          *PROGRAM TO PUNCH END OF FILE CHARACTERS
00005                          *TO CASSETTE TAPE

00007 E0E3           CONTRL    EQU     $E0E3
00008 E154           CASOUT    EQU     $E154

00010 A100 86 EC     BEGIN     LDA A   #$EC
00011 A102 BD E154             JSR     CASOUT
00012 A105 86 B9               LDA A   #$B9
00013 A107 BD E154             JSR     CASOUT
00014 A10A 86 35               LDAA    #$35    TURN READER CONTROL OFF
00015 A10C B7 F401             STAA    $F401   PIASA
00016 A10F 7E E0E3             JMP     CONTRL

00018                          END
```

```
SYMBOL TABLE

CONTRL E0E3  CASOUT E156  BEGIN  A100
```

START RECORDING ON THE TAPE AFTER THE SEARCH PROGRAM AND EXECUTE THE END OF FILE PROGRAM BY TYPING:

```
*H A100
```

THAT'S IT. YOU CAN PUT THE EOF CHARACTERS AT THE END OF ANY FILE OR FILES THAT YOU HAVE ON CASSETTE AND THEY WILL CAUSE THE LOAD ROUTINE TO RETURN TO THE MONITOR AUTOMATICALLY.

AND NOW, ONE LAST PROGRAM TO DEMONSTRATE THE GRAPHICS CAPABILITY OF THE VIDEO SCREEN:

```
00001                          NAM     GRAPH
00002 A200                     ORG     $A200

00004 E1AC           INEEE     EQU     $E1AC

00006 A200 CE F200   FIRST     LDX     #$F200   HALFWAY DOWN SCREEN
00007 A203 BD E1AC   SECOND    JSR     INEEE    GET A CHARACTER FROM KEYBOARD
00008 A206 8A 80               ORA A   #$80     SET THE EIGHTH BIT (GRAPHICS)
00009 A208 A7 00               STA A   X        PUT IT ON THE SCREEN
00010 A20A 08                  INX             ADVANCE POINTER TO NEXT
00011 A20B 20 F6               BRA     SECOND   SCREEN LOCATION

00013                          END
```

```
SYMBOL TABLE

INEEE  E1AC  FIRST  A200  SECOND A203
```

RUN THE PROGRAM BY TYPING:

*H A200

NOW TYPE ON THE KEYBOARD. EACH CHARACTER IS DISPLAYED BY THE OUTPUT ROUTINE AT THE TOP OF THE SCREEN. THESE CHARACTERS ALL HAVE THE MOST SIGNIFICANT BIT CLEARED. FOR EXAMPLE AN ASCII 'A' IS BINARY

01000001.

THIS PROGRAM STORES THE SAME CHARACTER WITH THE MSB SET IN THE MIDDLE OF THE SCREEN. AN 'A' BECOMES

11000001.

THIS PRODUCES EITHER GRAPHICS OR REVERSE FIELD VIDEO ON THE SCREEN DEPENDING ON THE MODE OF THE VIDEO BOARD.
TO ESCAPE FROM THE GRAPHICS PROGRAM HIT RESET.

********** SUMMARY **********

THE INFORMATION PRESENTED IN THIS GUIDE IS OF TWO TYPES: BASIC INFORMATION TO GET YOU STARTED AND ADVANCED INFORMATION ABOUT THE INTERNAL OPERATION OF THE MONITOR. IT WILL ENABLE YOU TO USE YOUR MICROCOMPUTER AND BEGIN LEARNING ABOUT ITS OPERATION AND PROGRAMMING. SOON YOU'LL BE WRITING PROGRAMS FAR MORE COMPLEX THAN YOU'VE DREAMED OF. REMEMBER - THE MORE YOU USE YOUR MICROCOMPUTER THE MORE YOU'LL LEARN ABOUT IT. THE CAPABILITIES OF A COMPUTER ARE ENDLESS. QUITE LITERALLY, THEY DEPEND ONLY ON YOUR PROGRAMMING IMAGINATION.

THE WORLD OF THE FUTURE IS MICROPROCESSORS, AND YOU'VE JUST TAKEN YOUR FIRST STEP INTO IT -- TODAY.