Apple II Original ROM Information

Source
http://members.buckeye-express.com/marksm/6502/
27 June 2004

The 6502 Firmware Page

This site is mostly about the firmware -- software in ROM -- that came with the original Apple II, not the II+, IIe, IIc, or IIgs. The original Apple II had 4K of RAM and 8K of ROM. The ROM contains software, such as the Monitor and Integer BASIC, appropriate for a SBC.

Red Book refers to the original Apple II Reference Manual dated 1978.
WOZPAK refers to the WOZPAK II, a publication by Call-A.P.P.L.E., an Apple II user group.
DDJ refers to Dr. Dobbs Journal, a computer magazine.
IA refers to Interface Age, a publication of the SCCS (Southern California Computer Society).
SYM and AIM refer to early 6502 single board computers.

Contents

Questions or comments? Email me at
paulrsm@buckeye-express.com

Updates

* 2000-09-01 -- Added AIM BASIC Language Reference Manual

```
+-------------------------------------------------------------------------
|  TOPIC -- Apple II -- Apple II ROM (12 KB binary)
+-------------------------------------------------------------------------

File ............. "a2rom.bin"
Fork ............. DATA
Size (bytes) ..... 12,288 (12KB) / $00003000
Created .......... Sunday, December 8, 2002 -- 8:47:53 PM
Modified ......... Sunday, December 8, 2002 -- 8:47:53 PM

D/000000: A9208D26 03AD57C0 AD53C0AD 50C0A900 [...&..W..S..P...]
D/000010: 851CAD26 03851BA0 00841AA5 1C911A20 [...&............]
D/000020: A2D0C8D0 F6E61BA5 1B291FD0 EE608D22 [.........)...`."]
D/000030: 038E2003 8C210348 29C08526 4A4A0526 [.....!.H)..&JJ.&]
D/000040: 85266885 270A0A0A 26270A26 270A6626 [.&h.'...&'.&'.f&]
D/000050: A527291F 0D260385 278AC000 F005A023 [.')..&..'......#]
D/000060: 6904C8E9 07B0FB8C 2503AABD EAD08530 [i.......%......0]
D/000070: 984AAD24 03851CB0 2960202E D0A51C51 [.J.$....)`.....Q]
D/000080: 26253051 26912660 1024A530 4AB00549 [&%0Q&.&`.$.0J..I]
D/000090: C0853060 881002A0 27A9C085 308C2503 [..0`....'...0.%.]
D/0000A0: A51C0AC9 C01006A5 1C497F85 1C60A530 [.........I...`.0]
D/0000B0: 0A498030 DCA981C8 C02890DF A000B0DB [.I.0.....(......]
D/0000C0: 18A55129 04F027A9 7F253031 26D01BEE [..Q)..'..%01&...]
D/0000D0: 2A03A97F 25301012 18A55129 04F00FB1 [*...%0....Q)....]
D/0000E0: 26451C25 30D003EE 2A035126 9126A551 [&E.%0...*.Q&.&.Q]
D/0000F0: 65532903 C9026AB0 8F303018 A5272CEA [eS)...j..00..',.]
D/000100: D1D02206 26B01A2C F3D0F005 691F38B0 [..".&..,....i.8.]
D/000110: 12692348 A52669B0 B00269F0 852668B0 [.i#H.&i...i..&h.]
D/000120: 02691F66 2669FC85 276018A5 2769042C [.i.f&i..'`..'i.,]
D/000130: EAD1D0F3 06269019 69E0182C 2ED1F013 [.....&..i..,....]
D/000140: A5266950 49F0F002 49F08526 AD260390 [.&iPI...I..&.&..]
D/000150: 0269E066 2690D048 A9008D20 038D2103 [.i.f&..H......!.]
D/000160: 8D220368 4838ED20 03488AED 21038553 [.".hH8...H..!..S]
D/000170: B00A6849 FF690148 A900E553 85518555 [..hI.i.H...S.Q.U]
D/000180: 68855085 54688D20 038E2103 9818ED22 [h.P.Th....!...."]
D/000190: 03900449 FF69FE85 528C2203 665338E5 [...I.i..R.".fS8.]
D/0001A0: 50AAA9FF E551851D AC2503B0 050A2088 [P....Q...%......]
D/0001B0: D038A554 65528554 A555E900 8555B126 [.8.TeR.T.U...U.&]
D/0001C0: 451C2530 51269126 E8D004E6 1DF06BA5 [E.%0Q&.&......k.]
D/0001D0: 53B0DA20 F9D018A5 54655085 54A55565 [S.......TeP.T.Ue]
D/0001E0: 5150D981 82848890 A0C01CFF FEFAF4EC [QP..............]
D/0001F0: E1D4C5B4 A18D7861 493118FF A5260AA5 [......xaI1...&..]
D/000200: 2729032A 05260A0A 0A8D2203 A5274A4A [').*.&....".'JJ]
D/000210: 29070D22 038D2203 AD25030A 6D25030A [).."..".%..m%..]
D/000220: AACAA530 297FE84A D0FC8D21 038A186D [...0)..J...!...m]
D/000230: 25039003 EE21038D 20036086 1A841BAA [%....!....`.....]
D/000240: 4A4A4A4A 85538A29 0FAABCEB D1845049 [JJJJ.S.)......PI]
D/000250: 0FAABCEC D1C88452 AC2503A2 008E2A03 [.......R.%....*.]
D/000260: A11A8551 A2808654 8655AE27 03A55438 [...Q...T.U.'..T8]
D/000270: 65508554 900420D8 D018A555 65528555 [eP.T.......UeR.U]
D/000280: 900320D9 D0CAD0E5 A5514A4A 4AD0D3E6 [.........QJJJ...]
D/000290: 1AD002E6 1BA11AD0 C960861A 841BAA4A [.........`.....J]
D/0002A0: 4A4A4A85 538A290F AABCEBD1 8450490F [JJJ.S.)......PI.]
D/0002B0: AABCECD1 C88452AC 2503A200 8E2A03A1 [......R.%....*..]
D/0002C0: 1A8551A2 80865486 55AE2703 A5543865 [..Q...T.U.'..T8e]
D/0002D0: 50855490 0420C0D0 18A55565 52855590 [P.T.......UeR.U.]
```

```
D/0002E0: 0320D9D0 CAD0E5A5 514A4A4A D0D3E61A [........QJJJ....]
D/0002F0: D002E61B A11AD0C9 602090D3 8D240320 [........`....$..]
D/000300: AFD34820 9AD36820 2ED0AE23 036020F9 [..H...h....#.`..]
D/000310: D24C7DD0 AD25034A 2090D320 75D0209A [.L}..%.J....u...]
D/000320: D38A4898 AA20AFD3 A8682064 D1AE2303 [..H......h.d..#.]
D/000330: 602090D3 4C10D020 F9D22051 D3203BD2 [`...L......Q..;.]
D/000340: AE230360 20F9D220 51D3209A D2AE2303 [.#.`....Q.....#.]
D/000350: 608E2303 A0322092 D38D2703 A0282092 [`.#..2....'..(..]
D/000360: D348AD28 03851AAD 2903851B A0202092 [.H.(....).......]
D/000370: D3F039A2 00C11AF0 02B0310A 9003E61B [..9.......1.....]
D/000380: 18A8B11A 651AAAC8 B11A6D29 03A86860 [....e.....m)..h`]
D/000390: A016B14A D01688B1 4A608E23 03A005B1 [...J....J`.#....]
D/0003A0: 4AAAC8B1 4AA8E018 E90190ED 4C68EEA0 [J...J.......Lh..]
D/0003B0: 0D2092D3 C9C0B0F4 608E2303 201EF120 [........`.#.....]
D/0003C0: FDFEA900 853C8D28 031865CE A8A90885 [.....<.(..e.....]
D/0003D0: 3D8D2903 65CFB025 C4CA48E5 CB68B01D [=.).e..%..H..h..]
D/0003E0: 843E853F C8D00269 01844A85 4B84CC85 [.>.?...i..J.K...]
D/0003F0: CD20FAFC A9032002 FFAE2303 604C6BE3 [..........#.`Lk.]
D/000400: 2089F6B0 3334F400 2089F618 4C006838 [....34......L.h8]
D/000410: 19CE00C9 3536213B 3CC93739 29D80346 [....56!;<.79)..F]
D/000420: 3A26E0D7 03384AA9 396AD302 2AD40202 [:&...8J.9j..*...]
D/000430: 07307600 A501A600 201BE5A9 AD20EDFD [.0v.............]
D/000440: A9BE20ED FDA517A6 16201BE5 208EFD20 [................]
D/000450: 8CF62B3C A23B0DD1 02C2004C 68EE004C [..+<.;.....Lh..L]
D/000460: 6BE3ECDC 02F419B0 001AC000 27D80363 [k...........'..c]
D/000470: E7673D25 3B211C2C A23C2BB6 03076BBD [.g=%;!.,.<+...k.]
D/000480: 07F5C72C 771B2800 1C67FC08 E547D902 [...,w.(..g...G..]
D/000490: 09DA02F5 F76705FC F747DB06 F71C5D00 [.....g...G....].]
D/0004A0: DC06F108 13FDFD06 0F1D2400 DD0609F0 [..........$.....]
D/0004B0: 06BA1D74 00BD0901 B03C01D1 2089F61C [...t.....<......]
D/0004C0: 4E00CC38 19CA0069 7C0020DF F02089F6 [N..8...i|.......]
D/0004D0: CC287C00 60A9DCA0 D44CB0D5 A434B900 [.(|.`....L...4..]
D/0004E0: 02C9AAD0 0CE634A2 07B53C95 02CA10F9 [......4...<.....]
D/0004F0: 60A002B1 3C990B00 8810F820 8EF8A62F [`...<........../]
D/000500: CAD00CA5 0B290DF0 142908D0 10850D20 [.....)...)......]
D/000510: 89F622D6 020626B1 0202A436 00A200B5 [.."...&....6....]
D/000520: 0B9142E8 20B4FCC6 2F10F490 C460A954 [..B...../....`.T]
D/000530: A0D54CB0 D586D838 A2FFB54D F5CB95CF [..L....8...M....]
D/000540: E8F0F720 1EF12054 D5A20120 2CF12054 [.......T....,..T]
D/000550: D5A6D860 20FAFCA9 1620C9FC 852E20FA [...`............]
D/000560: FCA02420 FDFCB0F9 20FDFCA0 3B20ECFC [..$.........;...]
D/000570: F00E452E 852E20BA FCA03490 F04C26FF [..E.......4..L&.]
D/000580: EAEAEAC1 3CF0EB48 202DFF20 92FDB13C [....<..H.-.....<]
D/000590: 20DAFDA9 A020EDFD A9A820ED FD6820DA [.............h..]
D/0005A0: FDA9A920 EDFDA98D 4CEDFDA9 8D4CEDFD [........L....L..]
D/0005B0: 8DF9038C FA03A94C 8DF80360 A9C3A0D5 [.......L...`....]
D/0005C0: 4CB0D5A9 0020D0D5 A9FF20D0 D54C3AFF [L............L:.]
D/0005D0: 850049FF 8501A53D 85078509 850BA000 [..I....=........]
D/0005E0: 84068408 840AA63E A5009108 C8D0FBE6 [.......>........]
D/0005F0: 09CAD0F6 A63EB106 C500F013 48A50720 [.....>......H...]
D/000600: DAFD9820 8AD6A500 208AD668 2092D6C8 [...........h....]
D/000610: D0E4E607 CAD0DFA6 3EA50191 0A840D84 [........>.......]
D/000620: 0CE60CA5 012045D6 A5002045 D6060C26 [......E....E...&]
D/000630: 0DA50DC5 3E90ECA5 00910AE6 0AD0DAE6 [....>...........]
D/000640: 0BCAD0D5 608502A5 0A450C85 08A50B45 [....`....E.....E]
D/000650: 0D8509A5 029108B1 0AC501F0 E748A50B [.............H..]
```

```
D/000660: 20DAFDA5 0A208AD6 A501910A 208AD668 [...............h]
D/000670: 4CCB02A5 0920DAFD A508208A D6A50220 [L...............]
D/000680: 8AD6202D FFA98D4C EDFD20DA FDA9A04C [...-...L.......L]
D/000690: EDFD840F 850E208A D6202DFF A500450E [..........-...E.]
D/0006A0: 850EA007 460E9023 A9A020ED FDA53DC9 [....F..#......=.]
D/0006B0: 50A9C469 0020EDFD A9AD20ED FD98D005 [P..i............]
D/0006C0: A9B120ED FDB9D3D6 20EDFD88 10D6A40F [................]
D/0006D0: 4C85D6B0 B9B8B7B6 B5B4B3B2 B1A00084 [L...............]
D/0006E0: 06840788 98D00EA0 1A200ED7 85068407 [................]
D/0006F0: A021200E D7850884 09A00820 0ED78502 [.!..............]
D/000700: 8403A011 200ED785 0484054C 08D4B14A [...........L...J]
D/000710: 48C8B14A A868604C 4ED7A401 AD30C0E6 [H..J.h`LN....0..]
D/000720: 02D005E6 03D00560 EA4C2CD7 88F0054C [.......`.L,....L]
D/000730: 32D7D0EB A400AD30 C0E602D0 05E603D0 [2......0........]
D/000740: 0560EA4C 46D788F0 D14C4CD7 D0EBADFF [.`.LF....LL.....]
D/000750: 020AA8B9 96D78500 ADFD024A F0044600 [...........J..F.]
D/000760: D0F9B996 D738E500 8501C8B9 96D76500 [.....8........e.]
D/000770: 8500A900 38EDFE02 8503A900 8502A501 [....8...........]
D/000780: D098EAEA 4C87D7E6 02D005E6 03D00560 [....L..........`]
D/000790: EA4C94D7 D0EC0000 F6F6E8E8 DBDBCFCF [.L..............]
D/0007A0: C3C3B8B8 AEAEA4A4 9B9B9292 8A8A8282 [................]
D/0007B0: 7B7B7474 6D6E6768 61625C5C 57575252 [{{ttmnghab\\WWRR]
D/0007C0: 4D4E4949 45454141 3D3E3A3A 36373334 [MNIIEEAA=>::6734]
D/0007D0: 30312E2E 2B2C2929 26272425 22232021 [01..+,))&'$%"#.!]
D/0007E0: 1E1F1D1D 1B1C1A1A 18191717 15161415 [................]
D/0007F0: 13141212 11111010 0F100E0F FFFFFFFF [................]
D/000800: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000810: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000820: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000830: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000840: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000850: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000860: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000870: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000880: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000890: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008A0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008B0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008C0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008D0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008E0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0008F0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000900: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000910: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000920: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000930: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000940: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000950: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000960: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000970: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000980: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000990: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0009A0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0009B0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0009C0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0009D0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
```

```
D/0009E0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/0009F0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000A90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AD0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000AF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000B90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BD0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000BF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000C90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CD0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000CF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
```

```
D/000D60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000D90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DD0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000DF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000E90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000EA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000EB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000EC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000ED0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000EE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000EF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F00: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F10: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F20: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F30: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F40: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F50: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F60: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F70: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F80: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000F90: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FA0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FB0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FC0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FD0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FE0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/000FF0: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF [................]
D/001000: 2000F04C B3E28533 4CEDFD60 8A2920F0 [...L...3L..`.)..]
D/001010: 23A9A085 E44CEDFD A920C524 B00CA98D [#....L.....$....]
D/001020: A00720ED FDA9A088 D0F8A000 B1E2E6E2 [................]
D/001030: D002E6E3 602015E7 2076E5A5 E2C5E6A5 [....`....v......]
D/001040: E3E5E7B0 EF206DE0 4C3BE0A5 CA85E2A5 [......m.L;......]
D/001050: CB85E3A5 4C85E6A5 4D85E7D0 DE2015E7 [....L...M.......]
D/001060: 206DE5A5 E485E2A5 E585E3B0 C786D8A9 [.m..............]
D/001070: A085FA20 2AE09885 E4202AE0 AA202AE0 [....*.....*...*.]
D/001080: 201BE520 18E084FA AA10180A 10E9A5E4 [................]
D/001090: D0032011 E08A20ED FDA92520 1AE0AA30 [..........%....0]
D/0010A0: F585E4C9 01D005A6 D84C8EFD 4884CEA2 [.........L..H...]
D/0010B0: ED86CFC9 519004C6 CFE95048 B1CEAA88 [....Q.....PH....]
D/0010C0: B1CE10FA E0C0B004 E00030F2 AA68E901 [..........0..h..]
D/0010D0: D0E924E4 300320F8 EFB1CE10 10AA293F [..$.0.........)?]
```

```
D/0010E0: 85E41869 A020EDFD 88E0C090 EC200CE0 [...i............]
D/0010F0: 68C95DF0 A4C928D0 8AF09E20 18E19550 [h.]...(........P]
D/001100: D5789011 A02B4CE0 E32034EE D55090F4 [.x...+L...4..P..]
D/001110: 20E4EF95 784C23E8 2034EEF0 E738E901 [....xL#..4...8..]
D/001120: 602018E1 955018F5 784C02E1 A014D0D6 [`....P..xL......]
D/001130: 2018E1E8 B55085DA 65CE48A8 B57885DB [.....P..e.H..x..]
D/001140: 65CF48C4 CAE5CBB0 E3A5DA69 FE85DAA9 [e.H........i....]
D/001150: FFA865DB 85DBC8B1 DAD9CC00 D00F98F0 [..e.............]
D/001160: F56891DA 99CC0088 10F7E860 EAA080D0 [.h.........`....]
D/001170: 95A90020 0AE7A002 9478200A E786D8AA [.........x......]
D/001180: E6332051 F3C6338A A6D89578 B55185CE [.3.Q..3....x.Q..]
D/001190: B57985CF E8E820BC E1B54ED5 76B015F6 [.y........N.v...]
D/0011A0: 4EA8B1CE B450C4E4 9004A083 D0C191DA [N....P..........]
D/0011B0: F65090E5 B4508A91 DA4C23F2 B55185DA [.P...P...L#..Q..]
D/0011C0: 38E90285 E4B57985 DBE90085 E5A000B1 [8.....y.........]
D/0011D0: E418E5DA 85E460B5 5385CEB5 7B85CFB5 [......`.S...{...]
D/0011E0: 5185DAB5 7985DBE8 E8E8A000 947894A0 [Q...y........x..]
D/0011F0: C89450B5 4DD57508 48B54FD5 77900768 [..P.M.u.H.O.w..h]
D/001200: 28B00256 5060A8B1 CE85E468 A828B0F3 [(..VP`.....h.(..]
D/001210: B1DAC5E4 D0EDF64F F64DB0D7 20D7E14C [.......O.M.....L]
D/001220: 36E72054 E206CE26 CF900D18 A5E665DA [6..T...&......e.]
D/001230: 85E6A5E7 65DB85E7 88F00906 E626E710 [....e........&..]
D/001240: E44C7EE7 A5E62008 E7A5E795 A006E590 [.L~.............]
D/001250: 284C6FE7 A95585E5 205BE2A5 CE85DAA5 [(Lo..U...[......]
D/001260: CF85DB20 15E784E6 84E7A5CF 1009CA06 [................]
D/001270: E5206FE7 2015E7A0 1060206C EEF0C5FF [..o......`.l....]
D/001280: E633A000 20CEE3C6 33602034 EE4A0820 [.3......3`.4.J..]
D/001290: 47F82034 EEA8B126 2890044A 4A4A4A29 [G..4...&(..JJJJ)]
D/0012A0: 0FA00020 08E794A0 8884D760 FFFFFFFF [...........`....]
D/0012B0: 20D3EF20 8EFD46D9 A9BE2006 E0A00084 [......F.........]
D/0012C0: FA24F810 0CA6F6A5 F7201BE5 A9A020ED [.$..............]
D/0012D0: FDA2FF9A 20CEE384 F18A85C8 A2202091 [................]
D/0012E0: E4A5C869 0085E0A9 00AA6902 85E1A1E0 [...i......i.....]
D/0012F0: 29F0C9B0 F0034C83 E8A002B1 E099CD00 [).....L.........]
D/001300: 88D0F820 8AE3A5F1 E5C8C904 F0A891E0 [................]
D/001310: A5CAF1E0 85E4A5CB E90085E5 A5E4C5CC [................]
D/001320: A5E5E5CD 9045A5CA F1E085E6 A5CBE900 [.....E..........]
D/001330: 85E7B1CA 91E6E6CA D002E6CB A5E2C5CA [................]
D/001340: A5E3E5CB B0E0B5E4 95CACA10 F9B1E0A8 [................]
D/001350: 88B1E091 E698D0F8 24F81009 B5F775F5 [........$.....u.]
D/001360: 95F7E8F0 F7107E00 000000A0 14D07120 [......~.......q.]
D/001370: 15E7A5E2 85E6A5E3 85E72075 E5A5E285 [...........u....]
D/001380: E4A5E385 E5D00E20 15E7206D E5A5E685 [...........m....]
D/001390: E2A5E785 E3A000A5 CAC5E4A5 CBE5E5B0 [................]
D/0013A0: 16A5E4D0 02C6E5C6 E4A5E6D0 02C6E7C6 [................]
D/0013B0: E6B1E491 E690E0A5 E685CAA5 E785CB60 [...............`]
D/0013C0: 20EDFDC8 B900EB30 F709804C EDFD98AA [.......0...L....]
D/0013D0: 2075FD8A A8A9DF99 0002A2FF 6060A006 [.u..........``..]
D/0013E0: 20D3EE24 D930034C B6E24C9A EB2A69A0 [...$.0.L..L..*i.]
D/0013F0: DD0002D0 53B1FE0A 300688B1 FE3029C8 [....S...0....0).]
D/001400: 86C89848 A200A1FE AA4A4940 11FEC9C0 [...H.....JI@....]
D/001410: 9001E8C8 D0F368A8 8A4CF8F2 E6F1A6F1 [......h..L......]
D/001420: F0BC9D00 0260A6C8 A9A0E8DD 0002B0FA [.....`..........]
D/001430: B1FE293F 4AD0B6BD 0002B006 693FC91A [..)?J.......i?..]
D/001440: 906F694F C90A9069 A6FDC8B1 FE29E0C9 [.oiO...i.....)..]
D/001450: 20F07AB5 A885C8B5 D185F188 B1FE0A10 [..z.............]
```

```
D/001460: FA88B038 0A3035B4 5884FFB4 80E810DA [...8.05.X.......]
D/001470: F0B3C97E B022CA10 04A00610 299480A4 [...~."......)...]
D/001480: FF9458A4 C894A8A4 F194D129 1FA8B997 [..X........)....]
D/001490: F10AA8A9 762A85FF D001C8C8 86FDB1FE [....v*..........]
D/0014A0: 3084D005 A00E4CE0 E3C903B0 C34AA6C8 [0.....L......J..]
D/0014B0: E8BD0002 9004C9A2 F00AC9DF F00686C8 [................]
D/0014C0: 201CE4C8 88A6FDB1 FE880A10 CFB45884 [..............X.]
D/0014D0: FFB480E8 B1FE299F D0ED85F2 85F39848 [......)........H]
D/0014E0: 86FDB4D0 84C918A9 0A85F9A2 00C8B900 [................]
D/0014F0: 02290F65 F2488A65 F3301CAA 68C6F9D0 [.).e.H.e.0..h...]
D/001500: F285F286 F3C4F1D0 DEA4C9C8 84F1201C [................]
D/001510: E468A8A5 F3B0A9A0 00108B85 F386F2A2 [.h..............]
D/001520: 0486C9A9 B085F9A5 F2DD63E5 A5F3FD68 [..........c....h]
D/001530: E5900D85 F3A5F2FD 63E585F2 E6F9D0E7 [........c.......]
D/001540: A5F9E8CA F00EC9B0 F00285C9 24C93004 [............$.0.]
D/001550: A5FAF00B 20EDFD24 F8100499 0002C8CA [.......$........]
D/001560: 10C16001 0A64E810 00000003 27A5CA85 [..`..d......'...]
D/001570: E6A5CB85 E7E8A5E7 85E5A5E6 85E4C54C [...............L]
D/001580: A5E5E54D B026A001 B1E4E5CE C8B1E4E5 [...M.&..........]
D/001590: CFB019A0 00A5E671 E485E690 03E6E718 [.......q........]
D/0015A0: C8A5CEF1 E4C8A5CF F1E4B0CA 6046F8A5 [............`F..]
D/0015B0: 4C85CAA5 4D85CBA5 4A85CCA5 4B85CDA9 [L...M...J...K...]
D/0015C0: 0085FB85 FC85FEA9 00851D60 A5D04C6B [...........`..Lk]
D/0015D0: E3A0FF84 D8C8B1E0 3006C940 D06885D8 [........0..@.h..]
D/0015E0: D1D0F0F1 B1D0C84A D0FAB1D0 48C8B1D0 [.......J....H...]
D/0015F0: A86885D0 84D1C5CC D0D7C4CD D0D3A000 [.h..............]
D/001600: C8B1E030 FB4940F0 F7986904 4865D0A8 [...0.I@...i.He..]
D/001610: A5D16900 48C4CAE5 CBB0B384 CC6885CD [..i.H........h..]
D/001620: 68A8A900 8891D088 91D088A5 CD91D088 [h...............]
D/001630: A5CC91D0 88A90091 D0883097 B1E0D0F7 [..........0.....]
D/001640: A54AA44B D0ACB1D0 C940B09A 959F9869 [.J.K.....@.....i]
D/001650: 034865D0 200AE720 FFE688D0 FA9865D1 [.He...........e.]
D/001660: 95786824 D8301DA8 A900200A E79578B1 [.xh$.0........x.]
D/001670: D0100FF6 78C8D0F7 09A90085 D485D5A2 [....x...........]
D/001680: 2048A000 B1E01018 0A30B520 FFE62008 [.H.......0......]
D/001690: E720FFE6 95A024D4 1001CA20 FFE6B0E6 [......$.........]
D/0016A0: C928D01F A5E0200A E7A5E195 7824D430 [.(..........x$.0]
D/0016B0: 0BA90120 0AE7A900 9578F678 20FFE630 [..........x.x...0]
D/0016C0: F9B0D324 D41006C9 04B0D046 D4A885D6 [...$.......F....]
D/0016D0: B980E929 550A85D7 68A8B980 E929AAC5 [...)U...h....)..]
D/0016E0: D7B00998 4820EBF3 A5D69095 B900EA85 [....H...........]
D/0016F0: CEB980EA 85CF20FC E64CD8E6 6CCE00E6 [.........L..l...]
D/001700: E0D002E6 E1B1E060 9477CA30 03955060 [.......`.w.0..P`]
D/001710: A0664CE0 E3A000B5 5085CEB5 A085CFB5 [.fL.....P.......]
D/001720: 78F00E85 CFB1CE48 C8B1CE85 CF6885CE [x......H.....h..]
D/001730: 88E86020 4AE72015 E7982008 E795A0C5 [..`.J...........]
D/001740: CED006C5 CFD002F6 50602082 E72059E7 [........P`....Y.]
D/001750: 2015E724 CF301BCA 602015E7 A5CFD004 [...$.0..`.......]
D/001760: A5CEF0F3 A9FF2008 E795A024 CF30E920 [...........$.0..]
D/001770: 15E79838 E5CE2008 E798E5CF 5023A000 [...8........P#..]
D/001780: 1090206F E72015E7 A5CE85DA A5CF85DB [...o............]
D/001790: 2015E718 A5CE65DA 2008E7A5 CF65DB70 [......e......e.p]
D/0017A0: DD95A060 2034EEA8 D0034CCB EE884CF4 [...`.4....L...L.]
D/0017B0: F3A52409 07A8C8D0 F5C8D0F5 B0F96000 [..$...........`.]
D/0017C0: 0020B1E7 2015E7A5 CF100AA9 AD20EDFD [................]
D/0017D0: 2072E750 EF8884D5 86CFA6CE 201BE5A6 [.r.P............]
```

```
D/0017E0: CF602015 E7A5CE85 F6A5CF85 F78884F8 [.`..............]
D/0017F0: C8A90A85 F484F560 2015E7A5 CEA4CF10 [.......`........]
D/001800: F22015E7 B55085DA B57885DB A5CE91DA [.....P...x......]
D/001810: C8A5CF4C 07F26068 6824D510 05208EFD [...L..`hh$......]
D/001820: 46D560A0 FF84D760 20CDEFF0 07A92585 [F.`....`......%.]
D/001830: D68884D4 E860A5CA A4CBD05A A041A5FC [.....`.....Z.A..]
D/001840: C910B05E A8E6FCA5 E0990001 A5E19910 [...^............]
D/001850: 01A5DC99 2001A5DD 99300120 15E7206D [.........0.....m]
D/001860: E59004A0 37D03BA5 E4A4E585 DC84DD18 [....7.;.........]
D/001870: 69039001 C8A2FF86 D99A85E0 84E1202E [i...............]
D/001880: F0A00020 79E624D9 104918A0 00A5DC71 [....y.$..I.....q]
D/001890: DCA4DD90 01C8C54C D0D1C44D D0CDA031 [.......L...M...1]
D/0018A0: 46D94CE0 E3A04AA5 FCF0F7C6 FCA8B91F [F.L...J.........]
D/0018B0: 0185DCB9 2F0185DD BEFF00B9 0F01A88A [..../...........]
D/0018C0: 4C75E8A0 6320C4E3 A001B1DC AAC8B1DC [Lu..c...........]
D/0018D0: 201BE54C B3E2C6FB A05BA5FB F0C4A8B5 [...L.....[......]
D/0018E0: 50D93F01 D0F0B578 D94F01D0 E9B95F01 [P.?....x.O...._.]
D/0018F0: 85DAB96F 0185DB20 15E7CA20 93E72001 [...o............]
D/001900: E8CAA4FB B9CF0195 9FB9BF01 A0002008 [................]
D/001910: E72082E7 2059E720 15E7A4FB A5CEF005 [.....Y..........]
D/001920: 596F0110 12B97F01 85DCB98F 0185DDBE [Yo..............]
D/001930: 9F01B9AF 01D087C6 FB60A054 A5FBC910 [.........`.T....]
D/001940: F09AE6FB A8B55099 4001B578 4C88F260 [......P.@..xL..`]
D/001950: 2015E7A4 FBA5CE99 BF01A5CF 99CF01A9 [................]
D/001960: 01995F01 A900996F 01A5DC99 7F01A5DD [.._....o........]
D/001970: 998F01A5 E0999F01 A5E199AF 01602015 [.............`..]
D/001980: 000000AB 03030303 03030303 03030303 [................]
D/001990: 03033F3F C0C03C3C 3C3C3C3C 3C300FC0 [..??..<<<<<<<0..]
D/0019A0: C3FF5500 ABAB0303 FFFF55FF FF55CFCF [..U.......U..U..]
D/0019B0: CFCFCFFF 55C6C6C6 55F0F0CF CF550155 [....U...U....U.U]
D/0019C0: FFFF5503 03030303 03030303 03030303 [..U.............]
D/0019D0: 03030303 03030303 03030303 0300AB03 [................]
D/0019E0: 57030303 03070303 03030303 03030303 [W...............]
D/0019F0: 0303AAFF 03030303 03030303 03030303 [................]
D/001A00: 17FFFF19 DF420AF2 EC876FAD B7E2F854 [.....B....o....T]
D/001A10: 4DC98582 2210334A 5B4E534A 49666D7A [M...".3J[NSJIfmz]
D/001A20: 71FF2309 5B16B6CB FFFFFBFF FF24F64E [q.#.[........$.N]
D/001A30: 59503BFF 23A36F36 23D71C22 1D8AAB23 [YP;.#.o6#.."...#]
D/001A40: FFFF2130 1E03C420 00C1BA39 40A0301E [..!0.......9@.0.]
D/001A50: A4D3B6BC AA3A0150 79D8D8A5 3CFF165B [.....:.Py...<..[]
D/001A60: 2803C41D 08004E00 3E00A6B0 00BCC657 [(.....N.>......W]
D/001A70: 8C0127FF 5D354B67 E0E17604 0571C91A [..'.]5Kg..v..q..]
D/001A80: E8FFFFE8 F0F1F3EF EFE3E3E5 E5E7E7EE [................]
D/001A90: F0F0E7E7 E2EFE7E7 F2F2F2E7 F2F2F2E2 [................]
D/001AA0: F3FFE8E1 E8E8EFEB FFFFE0FF FFEFEEEF [................]
D/001AB0: E7E7F3FF E8E7E7E7 E8E1E2EE F3E2E2E8 [................]
D/001AC0: FFFFE1E1 EFEEE7E8 EEE7F3FB FBEEE1EF [................]
D/001AD0: E7E8EFEF EBE9E8E9 F2E8E8E8 E8FFE8E8 [................]
D/001AE0: E8EEE7E8 EFEFEEEF EEEFEEEE EFEEEEEE [................]
D/001AF0: E1E8E8FF E0E0E0F1 F2F2F1F3 F3F1F3F4 [................]
D/001B00: BEB3B2B7 B637D4CF CFA0CCCF CE47D3D9 [.....7.......G..]
D/001B10: CED4C158 CDC5CDA0 C6D5CC4C D4CFCFA0 [...X.......L....]
D/001B20: CDC1CED9 A0D0C1D2 C5CE53D3 D4D2C9CE [..........S.....]
D/001B30: 47CECFA0 C5CE44C2 C1C4A0C2 D2C1CEC3 [G.....D.........]
D/001B40: 48B1B6A0 C7CFD3D5 C253C2C1 C4A0D2C5 [H........S......]
D/001B50: D4D5D24E B1B6A0C6 CFD253C2 C1C4A0CE [...N......S.....]
```

```
D/001B60: C5D854D3 D4CFD0D0 C5C4A0C1 D420AAAA [..T.............]
D/001B70: AA20A0C5 D2D20DBE B2B535D2 C1CEC745 [..........5....E]
D/001B80: C4C94DD3 D4D2A0CF D6C64CDC 0DD2C5D4 [..M.......L.....]
D/001B90: D9D0C5A0 CCC9CEC5 8D3F46D9 90034CC3 [.........?F...L.]
D/001BA0: E8A6CF9A A6CEA08D D002A099 20C4E386 [................]
D/001BB0: CEBA86CF 2066F384 F1A9FF85 C80A85D9 [.....f..........]
D/001BC0: A220A915 2091E4E6 D9A6CEA4 C80A85CE [................]
D/001BD0: C8B90002 C980F0D2 49B0C90A B0F0C8C8 [........I.......]
D/001BE0: 84C8B900 0248B9FF 01A00020 08E76895 [.....H........h.]
D/001BF0: A0A5CEC9 33D00320 6FE74C01 E8FFFFFF [....3...o.L.....]
D/001C00: 50204FC0 F4A1E4AF ADF2AFE4 AEA1F0A5 [P.O.............]
D/001C10: B4B3EFB4 EEA5A8B4 5C800040 608D608B [........\..@`.`.]
D/001C20: 7F1D207E 8C330000 6003BF12 4783AEA9 [...~.3..`...G...]
D/001C30: 6783B2B0 E5A3A1B2 B479B0B3 A469B0B3 [g........y...i..]
D/001C40: A4E5A3A1 B2B4AFAE 79B0B3A4 AFAE69B0 [........y.....i.]
D/001C50: B3A4AFAE F0AFB0F4 B3A9AC60 8C20B4B3 [...........`....]
D/001C60: A9AC0040 89C9479D 17689D0A 587B67A2 [...@..G..h..X{g.]
D/001C70: A1B4B667 B4A1078C 07AEA9AC B667B4A1 [...g.........g..]
D/001C80: 078C07AE A9ACA867 8C07B4AF ACB0679D [.......g......g.]
D/001C90: B2AFACAF A3678C07 A5ABAFB0 F4AEA9B2 [.....g..........]
D/001CA0: B07F0E27 B4AEA9B2 B07F0E28 B4AEA9B2 [...'.......(....]
D/001CB0: B06407A6 A967AFB4 AFA778B4 A5AC6B7F [.d...g....x...k.]
D/001CC0: 02ADA5B2 67A2B5B3 AFA7EEB2 B5B4A5B2 [....g...........]
D/001CD0: 7E8C39B4 B8A5AE67 B0A5B4B3 27AFB407 [~.9....g....'...]
D/001CE0: 9D19B2AF A67F0537 B4B5B0AE A97F0528 [.......7.......(]
D/001CF0: B4B5B0AE A97F052A B4B5B0AE A9E4AEA5 [.......*........]
D/001D00: 0047A2A1 B47F0D30 ADA9A47F 0D23ADA9 [.G.....0.....#..]
D/001D10: A467ACAC A1A3F2A7 F4B8A5B4 004DCC67 [.g...........M.g]
D/001D20: 8C688CDB 679B689B 508C638C 7F015107 [.h..g.h.P.c...Q.]
D/001D30: 88298480 C4195771 07881471 078C0788 [.)....Wq...q....]
D/001D40: AEB2A3B3 710888A3 B3A17108 88AEA5AC [....q.....q.....]
D/001D50: 68830868 9D087107 886075B4 AFAE758D [h..h..q..`u...u.]
D/001D60: 758B5107 8819B8A4 AEB2ECA4 B0F3A2A1 [u.Q.............]
D/001D70: EEA7B3E4 AEB2EBA5 A5B05107 883981C1 [..........Q..9..]
D/001D80: 4F7F0F2F 00510688 29C20C82 578C6A8C [O../.Q..)...W.j.]
D/001D90: 42AEA5A8 B460AEA5 A8B44F7E 1E358C27 [B....`....O~.5.']
D/001DA0: 51078809 8BFEE4AF ADF2AFE4 AEA1DCDE [Q...............]
D/001DB0: 9CDD9CDE DD9EC3DD CFCACDCB 00479AAD [.............G..]
D/001DC0: A5ADAFAC 679AADA5 ADA9A8EE A1AD608C [....g.........`.]
D/001DD0: 20AFB4B5 A1F2ACA3 F7A5AE60 8C20ACA5 [...........`....]
D/001DE0: A4EEB5B2 60AEB5B2 EEAFA3E5 B6A1B3E4 [....`...........]
D/001DF0: A1AFAC7A 7E9A2220 006003BF 6003BF1F [...z~."..`..`...]
D/001E00: 20B1E7E8 E8B54F85 DAB57785 DBB44E98 [......O...w...N.]
D/001E10: D576B009 B1DA20ED FDC84C0F EEA9FF85 [.v........L.....]
D/001E20: D560E8A9 00957895 A0B57738 F54F9550 [.`....x...w8.O.P]
D/001E30: 4C23E8FF 2015E7A5 CFD028A5 CE602034 [L#........(..`.4]
D/001E40: EEA4C8C9 30B021C0 28B01D4C 00F82034 [....0.!.(..L...4]
D/001E50: EE4C64F8 46F86020 B3F3C918 B00A8525 [.Ld.F.`........%]
D/001E60: 4C22FCA0 774CE0E3 A07BD0F9 2054E2A5 [L"..wL...{...T..]
D/001E70: DAD007A5 DBD0034C 7EE706CE 26CF26E6 [.......L~...&.&.]
D/001E80: 26E7A5E6 C5DAA5E7 E5DB900A 85E7A5E6 [&...............]
D/001E90: E5DA85E6 E6CE88D0 E160FFFF FFFFFFFF [.........`......]
D/001EA0: 2015E76C CE002034 EEC5C890 BB852C60 [...l...4......,`]
D/001EB0: 2034EEC9 30B0B1A4 C84C19F8 2034EEC5 [.4..0....L...4..]
D/001EC0: C890A585 2D602034 EEC928B0 9BA8A5C8 [....-`.4..(.....]
D/001ED0: 4C28F898 AAA06E20 C4E38AA8 20C4E3A0 [L(....n.........]
```

```
D/001EE0: 724C61F1 203FF206 CE26CF30 FAB0DCD0 [rLa..?...&.0....]
D/001EF0: 04C5CEB0 D6602015 E7B1CE94 9F4C08E7 [.....`.......L..]
D/001F00: 2034EEA5 CE85C860 2015E7A5 C891CE60 [.4.....`.......`]
D/001F10: 206CEEA5 CE85E6A5 CF85E74C 44E220E4 [.l.........LD...]
D/001F20: EE4C34E1 20E4EEB4 78B55069 FEB00188 [.L4.....x.Pi....]
D/001F30: 85DA84DB 1865CE95 509865CF 9578A000 [.....e..P.e..x..]
D/001F40: B550D1DA C8B578F1 DAB0804C 23E82015 [.P....x....L#...]
D/001F50: E7A54E20 08E7A54F D004C54E 6900297F [..N....O...Ni.).]
D/001F60: 854F95A0 A011A54F 0A186940 0A264E26 [.O.....O..i@.&N&]
D/001F70: 4F88D0F2 A5CE2008 E7A5CF95 A04C7AE2 [O............Lz.]
D/001F80: 2015E7A4 CEC44AA5 CFE54B90 1E844CA5 [......J...K...L.]
D/001F90: CF854D4C ADE52015 E7A4CEC4 4CA5CFE5 [..ML........L...]
D/001FA0: 4DB00884 4AA5CF85 4B90E84C CBEEFFFF [M...J...K..L....]
D/001FB0: FFFFFFFF FFFF2071 E14CBFEF 2003EEA9 [.......q.L......]
D/001FC0: FF85C8A9 808D0002 602036E7 E82036E7 [........`.6...6.]
D/001FD0: B55060A9 00854A85 4CA90885 4BA91085 [.P`...J.L...K...]
D/001FE0: 4D4CADE5 D578D001 184C02E1 20B7E54C [ML...x...L.....L]
D/001FF0: 36E820B7 E54C5BE8 E080D001 884C0CE0 [6....L[......L..]
D/002000: A00084A0 844A844C A908854B 854DE64D [.....J.L...K.M.M]
D/002010: B14C49FF 914CD14C D00849FF 914CD14C [.LI..L.L..I..L.L]
D/002020: F0EC4CAD E54C79F1 2032F04C BEE8A6E0 [..L..Ly..2.L....]
D/002030: A5E1AC00 C0C083D0 EC2C10C0 86508551 [.........,...P.Q]
D/002040: A5DC8578 A5DD8579 4CC3E8FF FF2015E7 [...x...yL.......]
D/002050: 86D8A2FE 38B5D095 E6B54EF5 D095DCE8 [....8.....N.....]
D/002060: D0F3904B CAB5CB95 E7F5DB95 E5E8F0F5 [...K............]
D/002070: 900AA5CC C5E4A5CD E5E59013 4C6BE3B1 [............Lk..]
D/002080: E691E4E6 E4D002E6 E5E6E6D0 02E6E7A5 [................]
D/002090: E6C54CA5 E7E54D90 E6A2FEB5 E6954EB5 [..L...M.......N.]
D/0020A0: CCF5DC95 CCE8D0F3 A6D860B1 4C91CEA5 [..........`.L...]
D/0020B0: CED002C6 CFC6CEA5 4CD002C6 4DC64CC5 [........L...M.L.]
D/0020C0: CAA54DE5 CB90E4B0 D02015E7 A4CEC0CA [..M.............]
D/0020D0: A5CFE5CB B0A6844A A5CF854B 4CB7E586 [.......J...KL...]
D/0020E0: D8201EF1 20FDFEA2 FF38B54D F5CF95DB [.........8.M....]
D/0020F0: E8F0F790 87A5CCC5 DAA5CDE5 DBB0D5A5 [................]
D/002100: CED004A5 CFF011A5 DA85CAA5 DB85CB20 [................]
D/002110: 2CF120FD FEA6D860 203AFF4C 15F1A0CE [,......`.:.L....]
D/002120: 843CC884 3EA00084 3D843F60 B5CA953C [.<..>...=.?`...<]
D/002130: B44C943E CA10F5A5 3ED002C6 3FC63E60 [.L.>....>...?.>`]
D/002140: 86D838A2 FFB54DF5 CB95CFE8 F0F7201E [..8...M.........]
D/002150: F120CDFE A201202C F1A91A20 CFFEA6D8 [.......,........]
D/002160: 6020C4E3 4C3AFFA5 FCD0034C A5E8C6FC [`...L:.....L....]
D/002170: 60A9FF85 A06046A0 6024A010 19A9A320 [`....`F.`$......]
D/002180: EDFDA001 B1DCAAC8 B1DC201B E5A9A04C [...............L]
D/002190: EDFDA5DC A4DD60C1 007FD1CC C7CFCEC5 [......`.........]
D/0021A0: 9A988D96 9593BFB2 32120FBC B0ACBE35 [........2......5]
D/0021B0: 0C613010 0BDDFBA0 0020C7E7 A9A04CED [.a0...........L.]
D/0021C0: FD000000 00000000 00A44AA5 4B48C4DA [..........J.KH..]
D/0021D0: E5DBB01C 6884D085 D1A0FFC8 B1D030FB [....h.........0.]
D/0021E0: C940F0F7 C8C8B1D0 4888B1D0 A868D0DD [.@......H....h..]
D/0021F0: 68A000B1 D030054A F008A9A4 20EDFDC8 [h....0.J........]
D/002200: D0F1A9BD 4CEDFD91 DAE8B59F F0304CD5 [....L........0L.]
D/002210: F3A03007 A5DCA4DD 207DF120 C9F1A6D8 [..0......}......]
D/002220: 4CB7F1E8 E8B59FF0 1F4CE0F3 3007A5DC [L........L..0...]
D/002230: A4DD207D F120C9F1 A6D84C09 F4E86020 [...}......L...`.]
D/002240: 15E7E6CE D002E6CF 60205BF2 D0152053 [........`.[....S]
D/002250: F2D01020 82E7206F E7500320 82E72059 [.......o.P.....Y]
```

```
D/002260: E756504C 36E720C9 EF154F10 0520C9EF [.VPL6.....O.....]
D/002270: 354F9550 10ED4CC9 EF2015E7 A4FBA5CE [5O.P..L.........]
D/002280: 995F01A5 CF4C66E9 99500188 3051B940 [._...Lf..P..0Q.@]
D/002290: 01D550D0 F6B95001 D578D0EF C6FBB941 [..P...P..x.....A]
D/0022A0: 01994001 B9510199 5001B9C1 0199C001 [..@..Q..P.......]
D/0022B0: B9D10199 D001B961 01996001 B9710199 [.......a..`..q..]
D/0022C0: 7001B981 01998001 B9910199 9001B9A1 [p...............]
D/0022D0: 0199A001 B9A10199 A001C8C4 FB90BF60 [...............`]
D/0022E0: E8A90048 B55038E9 0385CEB5 78E90085 [...H.P8.....x...]
D/0022F0: CF68A000 91CEE860 C985B003 4CC0E4A0 [.h.....`....L...]
D/002300: 024C48E4 E8A901D0 DAE8A578 85DCA579 [.LH........x...y]
D/002310: 85DDA550 A4514C75 E8A901D0 C6B550D5 [...P.QLu......P.]
D/002320: 7890034C 68EEA8B5 5185CEB5 7985CFB1 [x..Lh...Q...y...]
D/002330: CEA000E8 E82008E7 4C04F420 34EE86D8 [........L...4...]
D/002340: 2903AA20 1EFBA6D8 98A00020 08E794A0 [)...............]
D/002350: 602075FD 8A48BD00 02C983D0 034C03E0 [`.u..H.......L..]
D/002360: CA10F368 AA602080 E298AA20 54F38AA8 [...h.`......T...]
D/002370: 602015E7 A5CF1008 98CA2008 E794A060 [`..............`]
D/002380: 85D1A5CE 85D02015 E7A5CE85 D2A5CF85 [................]
D/002390: D3A90120 08E794A0 A5D0D004 C6D130DF [..............0.]
D/0023A0: C6D0A5D2 A0002008 E7A5D395 A02022E2 [.............".]
D/0023B0: 4C98F320 34EE1869 FF6020B1 E746D560 [L...4..i.`...F.`]
D/0023C0: 86D99A20 2EF04C83 E82034EE 86D82095 [......L...4.....]
D/0023D0: FEA6D860 FE24D910 E086D824 A04C12F2 [...`.$.....$.L..]
D/0023E0: 24D910D5 86D824A0 4C2CF2A0 004CFFE6 [$.....$.L,...L..]
D/0023F0: A8208EFD 9838E521 B0F68424 60000000 [.....8.!...$`...]
D/002400: FFFFFFFF 94A04C23 E8A000F0 0420EDFD [......L#........]
D/002410: C8B1DA30 F8A9FF85 D5602034 EE86D820 [...0.....`.4....]
D/002420: 8BFEA6D8 6018A202 B5F975F5 95F9CA10 [....`.....u.....]
D/002430: F76006F3 2037F424 F9100520 A4F4E6F3 [.`...7.$........]
D/002440: 38A20494 FBB5F7B4 F394F795 F3CAD0F3 [8...............]
D/002450: 60A98E85 F8A5F9C9 C0300CC6 F806FB26 [`........0.....&]
D/002460: FA26F9A5 F8D0EE60 20A4F420 7BF4A5F4 [.&.....`....{...]
D/002470: C5F8D0F7 2025F450 EA700590 C4A5F90A [.....%.P.p......]
D/002480: E6F8F075 A2FA76FF E8D0FB60 2032F465 [...u..v....`.2.e]
D/002490: F820E2F4 182084F4 90032025 F48810F5 [...........%....]
D/0024A0: 46F390BF 38A203A9 00F5F895 F8CAD0F7 [F...8...........]
D/0024B0: F0C52032 F4E5F820 E2F438A2 02B5F5F5 [...2......8.....]
D/0024C0: FC48CA10 F8A2FD68 900295F8 E8D0F826 [.H.....h.......&]
D/0024D0: FB26FA26 F906F726 F626F5B0 1C88D0DA [.&.&...&.&......]
D/0024E0: F0BE86FB 86FA86F9 B00D3004 686890B2 [..........0.hh..]
D/0024F0: 498085F8 A0176010 F74CF503 FFFFFFFF [I.....`..L......]
D/002500: E9814AD0 14A43FA6 3ED00188 CA8A18E5 [..J...?.>.......]
D/002510: 3A853E10 01C898E5 3BD06BA4 2FB93D00 [:.>.....;.k./.=.]
D/002520: 913A8810 F8201AFC 201AFC20 D0F82053 [.:.............S]
D/002530: F9843B85 3A4C95F5 20BEFFA4 3420A7FF [..;.:L......4...]
D/002540: 8434A017 88304BD9 CCFFD0F8 C015D0E8 [.4...0K.........]
D/002550: A531A000 C6342000 FE4C95F5 A53D208E [.1...4...L...=..]
D/002560: F8AABD00 FAC542D0 13BDC0F9 C543D00C [......B......C..]
D/002570: A544A42E C09DF088 C52EF09F C63DD0DC [.D...........=..]
D/002580: E644C635 F0D6A434 98AA204A F9A9DE20 [.D.5...4...J....]
D/002590: EDFD203A FFA9A185 332067FD 20C7FFAD [...:....3.g.....]
D/0025A0: 0002C9A0 F013C8C9 A4F09288 20A7FFC9 [................]
D/0025B0: 93D0D58A F0D22078 FEA90385 3D2034F6 [.......x....=.4.]
D/0025C0: 0AE9BEC9 C290C10A 0AA2040A 26422643 [............&B&C]
D/0025D0: CA10F8C6 3DF0F410 E4A20520 34F68434 [....=.......4..4]
```

```
D/0025E0: DDB4F9D0 132034F6 DDBAF9F0 0DBDBAF9 [......4.........]
D/0025F0: F007C9A4 F003A434 18882644 E003D00D [.......4..&D....]
D/002600: 20A7FFA5 3FF001E8 8635A203 88863DCA [....?....5....=.]
D/002610: 10C9A544 0A0A0535 C920B006 A635F002 [...D...5.....5..]
D/002620: 09808544 8434B900 02C9BBF0 04C98DD0 [...D.4..........]
D/002630: 804C5CF5 B90002C8 C9A0F0F8 60207DF4 [.L\.........`.}.]
D/002640: A5F81013 C98ED0F5 24F9100A A5FBF006 [........$.......]
D/002650: E6FAD002 E6F960A9 0085F985 FA60FFFF [......`......`..]
D/002660: FFFFFFFF FFFF4C92 F5845886 57855608 [......L...X.W.V.]
D/002670: 688559BA E8E8BD00 010A0A0A 0A60A458 [h.Y..........`.X]
D/002680: A657A559 48A55628 60204AFF 68851E68 [.W.YH.V(`.J.h..h]
D/002690: 851F2098 F64C92F6 E61ED002 E61FA9F7 [.....L..........]
D/0026A0: 48A000B1 1E290F0A AA4A511E F00B861D [H....)...JQ.....]
D/0026B0: 4A4A4AA8 B9E1F648 60E61ED0 02E61FBD [JJJ....H`.......]
D/0026C0: E4F648A5 1D4A6068 68203FFF 6C1E00B1 [..H..J`hh.?.l...]
D/0026D0: 1E950188 B11E9500 9838651E 851E9002 [.........8e.....]
D/0026E0: E61F6002 F9049D0D 9E25AF16 B247B951 [..`......%...G.Q]
D/0026F0: C02FC95B D285DD6E 0533E870 931EE765 [./.[...n.3.p...e]
D/002700: E7E7E710 CAB50085 00B50185 0160A500 [.............`..]
D/002710: 9500A501 950160A5 008100A0 00841DF6 [......`.........]
D/002720: 00D002F6 0160A100 8500A000 8401F0ED [.....`..........]
D/002730: A000F006 2066F7A1 00A82066 F7A10085 [.....f.....f....]
D/002740: 008401A0 00841D60 2026F7A1 0085014C [.......`.&.....L]
D/002750: 1FF72017 F7A50181 004C1FF7 2066F7A5 [.........L...f..]
D/002760: 0081004C 43F7B500 D002D601 D60060A0 [...LC.........`.]
D/002770: 0038A500 F5009900 00A501F5 01990100 [.8..............]
D/002780: 98690085 1D60A500 75008500 A5017501 [.i...`..u.....u.]
D/002790: A000F0E9 A51E2019 F7A51F20 19F718B0 [................]
D/0027A0: 0EB11E10 0188651E 851E9865 1F851F60 [......e....e...`]
D/0027B0: B0EC600A AAB50110 E8600AAA B50130E1 [..`......`....0.]
D/0027C0: 600AAAB5 001501F0 D8600AAA B5001501 [`........`......]
D/0027D0: D0CF600A AAB50035 0149FFF0 C4600AAA [..`....5.I...`..]
D/0027E0: B5003501 49FFD0B9 60A21820 66F7A100 [..5.I...`...f...]
D/0027F0: 851F2066 F7A10085 1E604CC7 F6F6FFFF [...f.....`L.....]
D/002800: 4A082047 F828A90F 900269E0 852EB126 [J..G.(....i....&]
D/002810: 4530252E 51269126 602000F8 C42CB011 [E0%.Q&.&`....,..]
D/002820: C8200EF8 90F66901 482000F8 68C52D90 [......i.H...h.-.]
D/002830: F560A02F D002A027 842DA027 A9008530 [.`./...'.-.'...0]
D/002840: 2028F888 10F66048 4A290309 04852768 [.(.....`HJ)....'h]
D/002850: 29189002 697F8526 0A0A0526 852660A5 [)...i..&...&.&`.]
D/002860: 30186903 290F8530 0A0A0A0A 05308530 [0.i.)..0.....0.0]
D/002870: 604A0820 47F8B126 2890044A 4A4A4A29 [`J..G..&(..JJJJ)]
D/002880: 0F60A63A A43B2096 FD2048F9 A13AA84A [.`.:.;....H..:.J]
D/002890: 90096AB0 10C9A2F0 0C29874A AABD62F9 [..j......).J..b.]
D/0028A0: 2079F8D0 04A080A9 00AABDA6 F9852E29 [.y.............)]
D/0028B0: 03852F98 298FAA98 A003E08A F00B4A90 [../.).........J.]
D/0028C0: 084A4A09 2088D0FA C888D0F2 60FFFFFF [.JJ.........`...]
D/0028D0: 2082F848 B13A20DA FDA20120 4AF9C42F [...H.:......J../]
D/0028E0: C890F1A2 03C00490 F268A8B9 C0F9852C [.........h.....,]
D/0028F0: B900FA85 2DA900A0 05062D26 2C2A88D0 [....-......-&,*..]
D/002900: F869BF20 EDFDCAD0 EC2048F9 A42FA206 [.i........H../..]
D/002910: E003F01C 062E900E BDB3F920 EDFDBDB9 [................]
D/002920: F9F00320 EDFDCAD0 E7608830 E720DAFD [.........`.0....]
D/002930: A52EC9E8 B13A90F2 2056F9AA E8D001C8 [.....:...V......]
D/002940: 9820DAFD 8A4CDAFD A203A9A0 20EDFDCA [.....L..........]
D/002950: D0F86038 A52FA43B AA100188 653A9001 [..`8./.;....e:..]
```

```
D/002960: C8600420 54300D80 04900322 54330D80 [.`..T0....."T3..]
D/002970: 04900420 54330D80 04900420 543B0D80 [....T3......T;..]
D/002980: 04900022 44330DC8 44001122 44330DC8 [..."D3..D.."D3..]
D/002990: 44A90122 44330D80 04900122 44330D80 [D.."D3....."D3..]
D/0029A0: 04902631 879A0021 81820000 594D9192 [..&1...!....YM..]
D/0029B0: 864A859D ACA9ACA3 A8A4D900 D8A4A400 [.J..............]
D/0029C0: 1C8A1C23 5D8B1BA1 9D8A1D23 9D8B1DA1 [...#]......#....]
D/0029D0: 002919AE 69A81923 24531B23 245319A1 [.)..i..#$S.#$S..]
D/0029E0: 001A5B5B A5692424 AEAEA8AD 29007C00 [..[[.i$$....).|.]
D/0029F0: 159C6D9C A5692953 84133411 A56923A0 [..m..i)S..4..i#.]
D/002A00: D8625A48 26629488 5444C854 6844E894 [.bZH&b..TD.ThD..]
D/002A10: 00B40884 74B4286E 74F4CC4A 72F2A48A [....t.(nt..Jr...]
D/002A20: 00AAA2A2 74747472 4468B232 B2002200 [....tttrDh.2..".]
D/002A30: 1A1A2626 727288C8 C4CA2648 4444A2C8 [..&&rr....&HDD..]
D/002A40: FFFFFF20 D0F86885 2C68852D A208BD10 [......h.,h.-....]
D/002A50: FB953CCA D0F8A13A F042A42F C920F059 [..<....:.B./...Y]
D/002A60: C960F045 C94CF05C C96CF059 C940F035 [.`.E.L.\.l.Y.@.5]
D/002A70: 291F4914 C904F002 B13A993C 008810F8 [).I......:.<....]
D/002A80: 203FFF4C 3C008545 68480A0A 0A30036C [.?.L<..EhH...0.l]
D/002A90: FE032820 4CFF6885 3A68853B 2082F820 [..(.L.h.:h.;....]
D/002AA0: DAFA4C65 FF186885 4868853A 68853BA5 [..Le..h.Hh.:h.;.]
D/002AB0: 2F2056F9 843B1890 14182054 F9AA9848 [/.V..;.....T...H]
D/002AC0: 8A48A002 18B13AAA 88B13A86 3B853AB0 [.H....:...:.;.:.]
D/002AD0: F3A52D48 A52C4820 8EFDA945 8540A900 [..-H.,H....E.@..]
D/002AE0: 8541A2FB A9A020ED FDBD1EFA 20EDFDA9 [.A..............]
D/002AF0: BD20EDFD B54A20DA FDE830E8 6018A001 [.....J....0.`...]
D/002B00: B13A2056 F9853A98 38B0A220 4AFF38B0 [.:.V..:.8...J.8.]
D/002B10: 9EEAEA4C 0BFB4CFD FAC1D8D9 D0D3AD70 [...L..L........p]
D/002B20: C0A000EA EABD64C0 1004C8D0 F88860A9 [......d.......`.]
D/002B30: 008548AD 56C0AD54 C0AD51C0 A900F00B [..H.V..T..Q.....]
D/002B40: AD50C0AD 53C02036 F8A91485 22A90085 [.P..S..6...."...]
D/002B50: 20A92885 21A91885 23A91785 254C22FC [..(.!...#...%L".]
D/002B60: 20A4FBA0 10A5504A 900C18A2 FEB55475 [......PJ......Tu]
D/002B70: 569554E8 D0F7A203 7650CA10 FB88D0E5 [V.T.....vP......]
D/002B80: 6020A4FB A0100650 26512652 265338A5 [`......P&Q&R&S8.]
D/002B90: 52E554AA A553E555 90068652 8553E650 [R.T..S.U...R.S.P]
D/002BA0: 88D0E360 A000842F A25420AF FBA250B5 [...`.../.T....P.]
D/002BB0: 01100D38 98F50095 0098F501 9501E62F [...8.........../]
D/002BC0: 60484A29 03090485 29682918 9002697F [`HJ)....)h)...i.]
D/002BD0: 85280A0A 05288528 60C987D0 12A94020 [.(...(.(`.....@.]
D/002BE0: A8FCA0C0 A90C20A8 FCAD30C0 88D0F560 [..........0....`]
D/002BF0: A4249128 E624A524 C521B066 60C9A0B0 [.$.(.$.$.!.f`...]
D/002C00: EFA810EC C98DF05A C98AF05A C988D0C9 [.......Z...Z....]
D/002C10: C62410E8 A5218524 C624A522 C525B00B [.$...!.$.$.".%..]
D/002C20: C625A525 20C1FB65 20852860 49C0F028 [.%.%...e..(`I..(]
D/002C30: 69FD90C0 F0DA69FD 902CF0DE 69FD905C [i.....i..,..i..\]
D/002C40: D0E9A424 A5254820 24FC209E FCA00068 [...$.%H.$......h]
D/002C50: 6900C523 90F0B0CA A5228525 A0008424 [i..#.....".%...$]
D/002C60: F0E4A900 8524E625 A525C523 90B6C625 [.....$.%.%.#...%]
D/002C70: A5224820 24FCA528 852AA529 852BA421 [."H.$..(.*.).+.!]
D/002C80: 88686901 C523B00D 482024FC B128912A [.hi..#..H.$..(.*]
D/002C90: 8810F930 E1A00020 9EFCB086 A424A9A0 [...0.........$..]
D/002CA0: 9128C8C4 2190F960 3848E901 D0FC68E9 [.(..!..`8H....h.]
D/002CB0: 01D0F660 E642D002 E643A53C C53EA53D [...`.B...C.<.>.=]
D/002CC0: E53FE63C D002E63D 60A04B20 DBFCD0F9 [.?.<...=`.K.....]
D/002CD0: 69FEB0F5 A02120DB FCC8C888 D0FD9005 [i....!..........]
```

```
D/002CE0: A03288D0 FDAC20C0 A02CCA60 A2084820 [.2.......,.`..H.]
D/002CF0: FAFC682A A03ACAD0 F56020FD FC88AD60 [..h*.:...`.....`]
D/002D00: C0452F10 F8452F85 2FC08060 A424B128 [.E/..E/./..`.$.(]
D/002D10: 48293F09 40912868 6C3800E6 4ED002E6 [H)?.@.(hl8..N...]
D/002D20: 4F2C00C0 10F59128 AD00C02C 10C06020 [O,.....(...,..`.]
D/002D30: 0CFD202C FC200CFD C99BF0F3 60A53248 [..., .........`.2H]
D/002D40: A9FF8532 BD000220 EDFD6885 32BD0002 [...2......h.2...]
D/002D50: C988F01D C998F00A E0F89003 203AFFE8 [.............:..]
D/002D60: D013A9DC 20EDFD20 8EFDA533 20EDFDA2 [...........3....]
D/002D70: 018AF0F3 CA2035FD C995D002 B128C9E0 [......5......(..]
D/002D80: 900229DF 9D0002C9 8DD0B220 9CFCA98D [..).............]
D/002D90: D05BA43D A63C208E FD2040F9 A000A9AD [.[.=.<....@.....]
D/002DA0: 4CEDFDA5 3C090785 3EA53D85 3FA53C29 [L...<...>.=.?.<)]
D/002DB0: 07D00320 92FDA9A0 20EDFDB1 3C20DAFD [............<...]
D/002DC0: 20BAFC90 E8604A90 EA4A4AA5 3E900249 [.....`J..JJ.>..I]
D/002DD0: FF653C48 A9BD20ED FD68484A 4A4A4A20 [.e<H.....hHJJJJ.]
D/002DE0: E5FD6829 0F09B0C9 BA900269 066C3600 [..h).......i.l6.]
D/002DF0: C9A09002 25328435 4820FDFB 68A43560 [....%2.5H...h.5`]
D/002E00: C634F09F CAD016C9 BAD0BB85 31A53E91 [.4..........1.>.]
D/002E10: 40E640D0 02E64160 A434B9FF 01853160 [@.@...A`.4....1`]
D/002E20: A201B53E 95429544 CA10F760 B13C9142 [...>.B.D...`.<.B]
D/002E30: 20B4FC90 F760B13C D142F01C 2092FDB1 [.....`.<.B......]
D/002E40: 3C20DAFD A9A020ED FDA9A820 EDFDB142 [<..............B]
D/002E50: 20DAFDA9 A920EDFD 20B4FC90 D9602075 [..............`.u]
D/002E60: FEA91448 20D0F820 53F9853A 843B6838 [...H....S..:.;h8]
D/002E70: E901D0EF 608AF007 B53C953A CA10F960 [....`....<.:...`]
D/002E80: A03FD002 A0FF8432 60A90085 3EA238A0 [.?.....2`...>.8.]
D/002E90: 1BD008A9 00853EA2 36A0F0A5 3E290FF0 [......>.6...>)..]
D/002EA0: 0609C0A0 00F002A9 FD940095 0160EAEA [.............`..]
D/002EB0: 4C00E04C 03E02075 FE203FFF 6C3A004C [L..L...u..?.l:.L]
D/002EC0: D7FAC634 2075FE4C 43FA4CF8 03A94020 [...4.u.LC.L...@.]
D/002ED0: C9FCA027 A200413C 48A13C20 EDFE20BA [...'..A<H.<.....]
D/002EE0: FCA01D68 90EEA022 20EDFEF0 4DA2100A [...h..."....M...]
D/002EF0: 20D6FCD0 FA602000 FE6868D0 6C20FAFC [.....`...hh.l...]
D/002F00: A91620C9 FC852E20 FAFCA024 20FDFCB0 [...........$....]
D/002F10: F920FDFC A03B20EC FC813C45 2E852E20 [.....;....<E....]
D/002F20: BAFCA035 90F020EC FCC52EF0 0DA9C520 [...5............]
D/002F30: EDFDA9D2 20EDFD20 EDFDA987 4CEDFDA5 [............L...]
D/002F40: 4848A545 A646A447 28608545 86468447 [HH.E.F.G(`.E.F.G]
D/002F50: 08688548 BA8649D8 602084FE 202FFB20 [.h.H..I.`..../..]
D/002F60: 93FE2089 FED8203A FFA9AA85 332067FD [.......:....3.g.]
D/002F70: 20C7FF20 A7FF8434 A0178830 E8D9CCFF [.......4...0....]
D/002F80: D0F820BE FFA4344C 73FFA203 0A0A0A0A [......4Ls.......]
D/002F90: 0A263E26 3FCA10F8 A531D006 B53F953D [.&>&?....1...?.=]
D/002FA0: 9541E8F0 F3D006A2 00863E86 3FB90002 [.A........>.?...]
D/002FB0: C849B0C9 0A90D369 88C9FAB0 CD60A9FE [.I.....i.....`..]
D/002FC0: 48B9E3FF 48A531A0 00843160 BCB2BEED [H...H.1...1`....]
D/002FD0: EFC4ECA9 BBA6A406 95070205 F000EB93 [................]
D/002FE0: A7C699B2 C9BEC135 8CC396AF 17172B1F [.......5......+.]
D/002FF0: 837F5DCC B5FC1717 F503FB03 59FF86FA [..].........Y...]
```

Brought to you by:

dtcdumpfile 1.0.0 (Apple Macintosh File Hex Dumper) Sunday, July 6, 1997

```
+------------------------------------------------------------------------
|  TOPIC -- Apple II -- Memory map of the Apple II ROMs
+------------------------------------------------------------------------
```

Memory map of the Apple II ROMs

* $F800-$FFFF

Monitor. Handles screen I/O and keyboard input. Also has a disassembler, memory dump, memory move, memory compare, step and trace functions, lo-res graphics routines, multiply and divide routines, and more. This monitor has the cleanest code of all the Apple II monitors. Every one after this had to patch the monitor to add functions while still remaining (mostly) compatible. Complete source code is in the manual.

· $F689-F7FC

Sweet-16 interpreter. Sweet-16 code has been benchmarked to be about half the size of pure 6502 code but 5-8 times slower. The renumber routine in the Programmer's Aid #1 is written in Sweet-16, where small size was much more important than speed. Complete source code is in the manual.

· $F500-F63C and $F666-F668

Mini-assembler. This lets you type in assembly code, one line at a time, and it will assemble the proper bytes. No labels or equates are supported--it is a MINI assembler. Complete source code is in the manual.

· $F425-F4FB and $F63D-F65D

Floating point routines. Woz's first plans for his 6502 BASIC included floating point, but he abandoned them when he realized he could finish faster by going integer only. He put these routines in the ROMs but they are not called from anywhere. Complete source code is in the manual.

· $E000-F424

Integer BASIC by Woz (Steve Wozniak, creator of the Apple II). "That BASIC, which we shipped with the first Apple II's, was never assembled--ever. There was one handwritten copy, all handwritten, all hand assembled." Woz, October 1984.

· $D800-DFFF

Empty ROM socket. There was at least one third party ROM add-on.

· $D000-D7FF

Programmer's Aid #1--missing from the original Apple II, this is a ROM add-on Apple sold that contains Integer BASIC utilities such as high-resolution graphics support, renumber, append, tape verify, music, and a RAM test. Complete source code is in the manual.

```
+----------------------------------------------------------------------------
|  TOPIC -- Apple II -- Summary of Monitor Commands
+----------------------------------------------------------------------------
```

Summary of Apple II Monitor Commands

Examining Memory.

*   {adrs}
Examines the value contained in one location.
*   {adrs1}.{adrs2}
Displays the values contained in all locations between {adrs1} and {adrs2}.
*   [RETURN]
Displays the values in up to eight locations following the last opened location.

Changing the Contents of Memory.

*   {adrs}:{val} {val} ...
Stores the values in consecutive memory locations starting at {adrs}.
*   :{val} {val}
Stores values in memory starting at the next changeable location.

Moving and Comparing.

*   {dest}<{start}.{end}M
Copies the values in the range {start}.{end} into the range beginning at {dest}. (M=move)
*   {dest}<{start}.{end}V
Compares the values in the range {start}.{end} to those in the range beginning at {dest}. (V=verify)

Saving and Loading via Cassette Tape.

*   {start}.{end}W
Writes the values in the memory range {start}.{end} onto tape, preceded by a ten-second leader.
*   {start}.{end}R
Reads values from tape, storing them in memory beginning at {start} and stopping at {end}. Prints "ERR" if an error occurs.

Running and Listing Programs.

*   {adrs}G
Transfers control to the machine language program beginning at {adrs}. (G=go)
*   {adrs}L
Disassembles and displays 20 instructions, starting at {adrs}. Subsequent L's will display 20 more instructions each. (L=list)

Miscellaneous.

*   {adrs}S
Disassemble, display, and execute the instruction at {adrs}, and display the contents of the 6502's internal registers. Subsequent S's will display and execute successive instructions. (S=step)

```
*     {adrs}T
Step infinitely. The TRACE command stops only when it executes a BRK instruction
or when you press RESET. (T=trace)
*     Contrl-E
Displays the contents of the 6502's registers. (E=examine)
*     I
Set Inverse display mode.
*     N
Set Normal display mode. Also useful as a delimiter for putting multiple commands
on one line.
*     Control-B
Enter the language currently installed in the Apple's ROM (cold start at $E000).
*     Control-C
Reenter the language currently installed in the Apple's ROM (warm start at $E003).
*     {val1}+{val2}
Add the two values and print the result.
*     {val2}-{val1}
Subtract the second value from the first and print the result.
*     {slot} Control-P
Divert output to the device whose interface card in in slot number {slot}. If
{slot}=0, then route output to the Apple's screen.
*     {slot} Control-K
Accept input from the device whose interface card is in slot number {slot}. If
{slot}=0, then accept input from the Apple's keyboard.
*     Control-Y
Jump to the machine language subroutine at location $03F8. This lets you add your
own commands to the Monitor.

The Mini-Assembler.

*     F666G
Invoke the Mini-Assembler.
*     ${command}
Execute a Monitor command from the Mini-Assembler.
*     FF69G
Leave the Mini-Assembler.
```

```
+------------------------------------------------------------------------
|  TOPIC -- Apple II -- Red Book Monitor listing
+------------------------------------------------------------------------
```

```
**************************
*                        *
*        APPLE II        *
*     SYSTEM MONITOR     *
*                        *
*    COPYRIGHT 1977 BY   *
*   APPLE COMPUTER, INC. *
*                        *
*   ALL RIGHTS RESERVED  *
*                        *
*       S. WOZNIAK       *
*        A. BAUM         *
*                        *
**************************
                           ; TITLE "APPLE II SYSTEM MONITOR"
LOC0     EQU   $00
LOC1     EQU   $01
WNDLFT   EQU   $20
WNDWDTH  EQU   $21
WNDTOP   EQU   $22
WNDBTM   EQU   $23
CH       EQU   $24
CV       EQU   $25
GBASL    EQU   $26
GBASH    EQU   $27
BASL     EQU   $28
BASH     EQU   $29
BAS2L    EQU   $2A
BAS2H    EQU   $2B
H2       EQU   $2C
LMNEM    EQU   $2C
RTNL     EQU   $2C
V2       EQU   $2D
RMNEM    EQU   $2D
RTNH     EQU   $2D
MASK     EQU   $2E
CHKSUM   EQU   $2E
FORMAT   EQU   $2E
LASTIN   EQU   $2F
LENGTH   EQU   $2F
SIGN     EQU   $2F
COLOR    EQU   $30
MODE     EQU   $31
INVFLG   EQU   $32
PROMPT   EQU   $33
YSAV     EQU   $34
YSAV1    EQU   $35
CSWL     EQU   $36
CSWH     EQU   $37
KSWL     EQU   $38
KSWH     EQU   $39
PCL      EQU   $3A
PCH      EQU   $3B
XQT      EQU   $3C
A1L      EQU   $3C
A1H      EQU   $3D
```

```
                A2L      EQU   $3E
                A2H      EQU   $3F
                A3L      EQU   $40
                A3H      EQU   $41
                A4L      EQU   $42
                A4H      EQU   $43
                A5L      EQU   $44
                A5H      EQU   $45
                ACC      EQU   $45
                XREG     EQU   $46
                YREG     EQU   $47
                STATUS   EQU   $48
                SPNT     EQU   $49
                RNDL     EQU   $4E
                RNDH     EQU   $4F
                ACL      EQU   $50
                ACH      EQU   $51
                XTNDL    EQU   $52
                XTNDH    EQU   $53
                AUXL     EQU   $54
                AUXH     EQU   $55
                PICK     EQU   $95
                IN       EQU   $0200
                USRADR   EQU   $03F8
                NMI      EQU   $03FB
                IRQLOC   EQU   $03FE
                IOADR    EQU   $C000
                KBD      EQU   $C000
                KBDSTRB  EQU   $C010
                TAPEOUT  EQU   $C020
                SPKR     EQU   $C030
                TXTCLR   EQU   $C050
                TXTSET   EQU   $C051
                MIXCLR   EQU   $C052
                MIXSET   EQU   $C053
                LOWSCR   EQU   $C054
                HISCR    EQU   $C055
                LORES    EQU   $C056
                HIRES    EQU   $C057
                TAPEIN   EQU   $C060
                PADDL0   EQU   $C064
                PTRIG    EQU   $C070
                BASIC    EQU   $E000
                BASIC2   EQU   $E003
                         ORG   $F800        ;ROM START ADDRESS
F800: 4A        PLOT     LSR                ;Y-COORD/2
F801: 08                 PHP                ;SAVE LSB IN CARRY
F802: 20 47 F8           JSR   GBASCALC     ;CALC BASE ADR IN GBASL,H
F805: 28                 PLP                ;RESTORE LSB FROM CARRY
F806: A9 0F              LDA   #$0F         ;MASK $0F IF EVEN
F808: 90 02              BCC   RTMASK
F80A: 69 E0              ADC   #$E0         ;MASK $F0 IF ODD
F80C: 85 2E     RTMASK   STA   MASK
F80E: B1 26     PLOT1    LDA   (GBASL),Y    ;DATA
F810: 45 30              EOR   COLOR        ; EOR COLOR
F812: 25 2E              AND   MASK         ;  AND MASK
F814: 51 26              EOR   (GBASL),Y    ;   EOR DATA
F816: 91 26              STA   (GBASL),Y    ;    TO DATA
F818: 60                 RTS
F819: 20 00 F8  HLINE    JSR   PLOT         ;PLOT SQUARE
F81C: C4 2C     HLINE1   CPY   H2           ;DONE?
F81E: B0 11              BCS   RTS1         ; YES, RETURN
```

```
F820: C8                  INY                ; NO, INC INDEX (X-COORD)
F821: 20 0E F8            JSR   PLOT1        ;PLOT NEXT SQUARE
F824: 90 F6               BCC   HLINE1       ;ALWAYS TAKEN
F826: 69 01     VLINEZ    ADC   #$01         ;NEXT Y-COORD
F828: 48        VLINE     PHA                ; SAVE ON STACK
F829: 20 00 F8            JSR   PLOT         ; PLOT SQUARE
F82C: 68                  PLA
F82D: C5 2D               CMP   V2           ;DONE?
F82F: 90 F5               BCC   VLINEZ       ; NO, LOOP
F831: 60        RTS1      RTS
F832: A0 2F     CLRSCR    LDY   #$2F         ;MAX Y, FULL SCRN CLR
F834: D0 02               BNE   CLRSC2       ;ALWAYS TAKEN
F836: A0 27     CLRTOP    LDY   #$27         ;MAX Y, TOP SCREEN CLR
F838: 84 2D     CLRSC2    STY   V2           ;STORE AS BOTTOM COORD
                                             ; FOR VLINE CALLS
F83A: A0 27               LDY   #$27         ;RIGHTMOST X-COORD (COLUMN)
F83C: A9 00     CLRSC3    LDA   #$00         ;TOP COORD FOR VLINE CALLS
F83E: 85 30               STA   COLOR        ;CLEAR COLOR (BLACK)
F840: 20 28 F8            JSR   VLINE        ;DRAW VLINE
F843: 88                  DEY                ;NEXT LEFTMOST X-COORD
F844: 10 F6               BPL   CLRSC3       ;LOOP UNTIL DONE
F846: 60                  RTS
F847: 48        GBASCALC PHA                 ;FOR INPUT 000DEFGH
F848: 4A                  LSR
F849: 29 03               AND   #$03
F84B: 09 04               ORA   #$04         ;  GENERATE GBASH=000001FG
F84D: 85 27               STA   GBASH
F84F: 68                  PLA                ;  AND GBASL=HDEDE000
F850: 29 18               AND   #$18
F852: 90 02               BCC   GBCALC
F854: 69 7F               ADC   #$7F
F856: 85 26     GBCALC    STA   GBASL
F858: 0A                  ASL
F859: 0A                  ASL
F85A: 05 26               ORA   GBASL
F85C: 85 26               STA   GBASL
F85E: 60                  RTS
F85F: A5 30     NXTCOL    LDA   COLOR        ;INCREMENT COLOR BY 3
F861: 18                  CLC
F862: 69 03               ADC   #$03
F864: 29 0F     SETCOL    AND   #$0F         ;SETS COLOR=17*A MOD 16
F866: 85 30               STA   COLOR
F868: 0A                  ASL                ;BOTH HALF BYTES OF COLOR EQUAL
F869: 0A                  ASL
F86A: 0A                  ASL
F86B: 0A                  ASL
F86C: 05 30               ORA   COLOR
F86E: 85 30               STA   COLOR
F870: 60                  RTS
F871: 4A        SCRN      LSR                ;READ SCREEN Y-COORD/2
F872: 08                  PHP                ;SAVE LSB (CARRY)
F873: 20 47 F8            JSR   GBASCALC     ;CALC BASE ADDRESS
F876: B1 26               LDA   (GBASL),Y    ;GET BYTE
F878: 28                  PLP                ;RESTORE LSB FROM CARRY
F879: 90 04     SCRN2     BCC   RTMSKZ       ;IF EVEN, USE LO H
F87B: 4A                  LSR
F87C: 4A                  LSR
F87D: 4A                  LSR                ;SHIFT HIGH HALF BYTE DOWN
F87E: 4A                  LSR
F87F: 29 0F     RTMSKZ    AND   #$0F         ;MASK 4-BITS
F881: 60                  RTS
F882: A6 3A     INSDS1    LDX   PCL          ;PRINT PCL,H
```

```
F884: A4 3B             LDY   PCH
F886: 20 96 FD          JSR   PRYX2
F889: 20 48 F9          JSR   PRBLNK      ;FOLLOWED BY A BLANK
F88C: A1 3A             LDA   (PCL,X)     ;GET OP CODE
F88E: A8         INSDS2 TAY
F88F: 4A                LSR               ;EVEN/ODD TEST
F890: 90 09             BCC   IEVEN
F892: 6A                ROR               ;BIT 1 TEST
F893: B0 10             BCS   ERR         ;XXXXXX11 INVALID OP
F895: C9 A2             CMP   #$A2
F897: F0 0C             BEQ   ERR         ;OPCODE $89 INVALID
F899: 29 87             AND   #$87        ;MASK BITS
F89B: 4A         IEVEN  LSR               ;LSB INTO CARRY FOR L/R TEST
F89C: AA                TAX
F89D: BD 62 F9          LDA   FMT1,X      ;GET FORMAT INDEX BYTE
F8A0: 20 79 F8          JSR   SCRN2       ;R/L H-BYTE ON CARRY
F8A3: D0 04             BNE   GETFMT
F8A5: A0 80      ERR    LDY   #$80        ;SUBSTITUTE $80 FOR INVALID OPS
F8A7: A9 00             LDA   #$00        ;SET PRINT FORMAT INDEX TO 0
F8A9: AA         GETFMT TAX
F8AA: BD A6 F9          LDA   FMT2,X      ;INDEX INTO PRINT FORMAT TABLE
F8AD: 85 2E             STA   FORMAT      ;SAVE FOR ADR FIELD FORMATTING
F8AF: 29 03             AND   #$03        ;MASK FOR 2-BIT LENGTH
                                          ; (P=1 BYTE, 1=2 BYTE, 2=3 BYTE)
F8B1: 85 2F             STA   LENGTH
F8B3: 98                TYA               ;OPCODE
F8B4: 29 8F             AND   #$8F        ;MASK FOR 1XXX1010 TEST
F8B6: AA                TAX               ; SAVE IT
F8B7: 98                TYA               ;OPCODE TO A AGAIN
F8B8: A0 03             LDY   #$03
F8BA: E0 8A             CPX   #$8A
F8BC: F0 0B             BEQ   MNNDX3
F8BE: 4A        MNNDX1  LSR
F8BF: 90 08             BCC   MNNDX3      ;FORM INDEX INTO MNEMONIC TABLE
F8C1: 4A                LSR
F8C2: 4A        MNNDX2  LSR               ;1) 1XXX1010->00101XXX
F8C3: 09 20             ORA   #$20        ;2) XXXYYY01->00111XXX
F8C5: 88                DEY               ;3) XXXYYY10->00110XXX
F8C6: D0 FA             BNE   MNNDX2      ;4) XXXYY100->00100XXX
F8C8: C8                INY               ;5) XXXXX000->000XXXXX
F8C9: 88        MNNDX3  DEY
F8CA: D0 F2             BNE   MNNDX1
F8CC: 60                RTS
F8CD: FF FF FF          DFB   $FF,$FF,$FF
F8D0: 20 82 F8  INSTDSP JSR   INSDS1      ;GEN FMT, LEN BYTES
F8D3: 48                PHA               ;SAVE MNEMONIC TABLE INDEX
F8D4: B1 3A     PRNTOP  LDA   (PCL),Y
F8D6: 20 DA FD          JSR   PRBYTE
F8D9: A2 01             LDX   #$01        ;PRINT 2 BLANKS
F8DB: 20 4A F9  PRNTBL  JSR   PRBL2
F8DE: C4 2F             CPY   LENGTH      ;PRINT INST (1-3 BYTES)
F8E0: C8                INY               ;IN A 12 CHR FIELD
F8E1: 90 F1             BCC   PRNTOP
F8E3: A2 03             LDX   #$03        ;CHAR COUNT FOR MNEMONIC PRINT
F8E5: C0 04             CPY   #$04
F8E7: 90 F2             BCC   PRNTBL
F8E9: 68                PLA               ;RECOVER MNEMONIC INDEX
F8EA: A8                TAY
F8EB: B9 C0 F9          LDA   MNEML,Y
F8EE: 85 2C             STA   LMNEM       ;FETCH 3-CHAR MNEMONIC
F8F0: B9 00 FA          LDA   MNEMR,Y     ;  (PACKED IN 2-BYTES)
F8F3: 85 2D             STA   RMNEM
```

```
F8F5: A9 00     PRMN1    LDA   #$00
F8F7: A0 05              LDY   #$05
F8F9: 06 2D     PRMN2    ASL   RMNEM       ;SHIFT 5 BITS OF
F8FB: 26 2C              ROL   LMNEM       ;  CHARACTER INTO A
F8FD: 2A                 ROL               ;    (CLEARS CARRY)
F8FE: 88                 DEY
F8FF: D0 F8              BNE   PRMN2
F901: 69 BF              ADC   #$BF        ;ADD "?" OFFSET
F903: 20 ED FD           JSR   COUT        ;OUTPUT A CHAR OF MNEM
F906: CA                 DEX
F907: D0 EC              BNE   PRMN1
F909: 20 48 F9           JSR   PRBLNK      ;OUTPUT 3 BLANKS
F90C: A4 2F              LDY   LENGTH
F90E: A2 06              LDX   #$06        ;CNT FOR 6 FORMAT BITS
F910: E0 03     PRADR1   CPX   #$03
F912: F0 1C              BEQ   PRADR5      ;IF X=3 THEN ADDR.
F914: 06 2E     PRADR2   ASL   FORMAT
F916: 90 0E              BCC   PRADR3
F918: BD B3 F9           LDA   CHAR1-1,X
F91B: 20 ED FD           JSR   COUT
F91E: BD B9 F9           LDA   CHAR2-1,X
F921: F0 03              BEQ   PRADR3
F923: 20 ED FD           JSR   COUT
F926: CA        PRADR3   DEX
F927: D0 E7              BNE   PRADR1
F929: 60                 RTS
F92A: 88        PRADR4   DEY
F92B: 30 E7              BMI   PRADR2
F92D: 20 DA FD           JSR   PRBYTE
F930: A5 2E     PRADR5   LDA   FORMAT
F932: C9 E8              CMP   #$E8        ;HANDLE REL ADR MODE
F934: B1 3A              LDA   (PCL),Y     ;SPECIAL (PRINT TARGET,
F936: 90 F2              BCC   PRADR4      ;  NOT OFFSET)
F938: 20 56 F9  RELADR   JSR   PCADJ3
F93B: AA                 TAX               ;PCL,PCH+OFFSET+1 TO A,Y
F93C: E8                 INX
F93D: D0 01              BNE   PRNTYX      ;+1 TO Y,X
F93F: C8                 INY
F940: 98        PRNTYX   TYA
F941: 20 DA FD  PRNTAX   JSR   PRBYTE      ;OUTPUT TARGET ADR
F944: 8A        PRNTX    TXA               ;  OF BRANCH AND RETURN
F945: 4C DA FD           JMP   PRBYTE
F948: A2 03     PRBLNK   LDX   #$03        ;BLANK COUNT
F94A: A9 A0     PRBL2    LDA   #$A0        ;LOAD A SPACE
F94C: 20 ED FD  PRBL3    JSR   COUT        ;OUTPUT A BLANK
F94F: CA                 DEX
F950: D0 F8              BNE   PRBL2       ;LOOP UNTIL COUNT=0
F952: 60                 RTS
F953: 38        PCADJ    SEC               ;0=1-BYTE, 1=2-BYTE
F954: A5 2F     PCADJ2   LDA   LENGTH      ;  2=3-BYTE
F956: A4 3B     PCADJ3   LDY   PCH
F958: AA                 TAX               ;TEST DISPLACEMENT SIGN
F959: 10 01              BPL   PCADJ4      ;  (FOR REL BRANCH)
F95B: 88                 DEY               ;EXTEND NEG BY DEC PCH
F95C: 65 3A     PCADJ4   ADC   PCL
F95E: 90 01              BCC   RTS2        ;PCL+LENGTH(OR DISPL)+1 TO A
F960: C8                 INY               ;  CARRY INTO Y (PCH)
F961: 60        RTS2     RTS
                * FMT1 BYTES:    XXXXXXY0 INSTRS
                * IF Y=0         THEN LEFT HALF BYTE
                * IF Y=1         THEN RIGHT HALF BYTE
                *                   (X=INDEX)
```

```
F962: 04 20 54  FMT1     DFB    $04,$20,$54,$30,$0D
F965: 30 0D
F967: 80 04 90           DFB    $80,$04,$90,$03,$22
F96A: 03 22
F96C: 54 33 0D           DFB    $54,$33,$0D,$80,$04
F96F: 80 04
F971: 90 04 20           DFB    $90,$04,$20,$54,$33
F974: 54 33
F976: 0D 80 04           DFB    $0D,$80,$04,$90,$04
F979: 90 04
F97B: 20 54 3B           DFB    $20,$54,$3B,$0D,$80
F97E: 0D 80
F980: 04 90 00           DFB    $04,$90,$00,$22,$44
F983: 22 44
F985: 33 0D C8           DFB    $33,$0D,$C8,$44,$00
F988: 44 00
F98A: 11 22 44           DFB    $11,$22,$44,$33,$0D
F98D: 33 0D
F98F: C8 44 A9           DFB    $C8,$44,$A9,$01,$22
F992: 01 22
F994: 44 33 0D           DFB    $44,$33,$0D,$80,$04
F997: 80 04
F999: 90 01 22           DFB    $90,$01,$22,$44,$33
F99C: 44 33
F99E: 0D 80 04           DFB    $0D,$80,$04,$90
F9A1: 90
F9A2: 26 31 87           DFB    $26,$31,$87,$9A ;$ZZXXXY01 INSTR'S
F9A5: 9A
F9A6: 00        FMT2     DFB    $00         ;ERR
F9A7: 21                 DFB    $21         ;IMM
F9A8: 81                 DFB    $81         ;Z-PAGE
F9A9: 82                 DFB    $82         ;ABS
F9AA: 00                 DFB    $00         ;IMPLIED
F9AB: 00                 DFB    $00         ;ACCUMULATOR
F9AC: 59                 DFB    $59         ;(ZPAG,X)
F9AD: 4D                 DFB    $4D         ;(ZPAG),Y
F9AE: 91                 DFB    $91         ;ZPAG,X
F9AF: 92                 DFB    $92         ;ABS,X
F9B0: 86                 DFB    $86         ;ABS,Y
F9B1: 4A                 DFB    $4A         ;(ABS)
F9B2: 85                 DFB    $85         ;ZPAG,Y
F9B3: 9D                 DFB    $9D         ;RELATIVE
F9B4: AC A9 AC  CHAR1    ASC    ",),#($"
F9B7: A3 A8 A4
F9BA: D9 00 D8  CHAR2    DFB    $D9,$00,$D8,$A4,$A4,$00
F9BD: A4 A4 00
                *CHAR2: "Y",0,"X$$",0
                * MNEML IS OF FORM:
                *  (A) XXXXX000
                *  (B) XXXYY100
                *  (C) 1XXX1010
                *  (D) XXXYYY10
                *  (E) XXXYYY01
                *      (X=INDEX)
F9C0: 1C 8A 1C  MNEML    DFB    $1C,$8A,$1C,$23,$5D,$8B
F9C3: 23 5D 8B
F9C6: 1B A1 9D           DFB    $1B,$A1,$9D,$8A,$1D,$23
F9C9: 8A 1D 23
F9CC: 9D 8B 1D           DFB    $9D,$8B,$1D,$A1,$00,$29
F9CF: A1 00 29
F9D2: 19 AE 69           DFB    $19,$AE,$69,$A8,$19,$23
F9D5: A8 19 23
```

```
F9D8: 24 53 1B  347          DFB   $24,$53,$1B,$23,$24,$53
F9DB: 23 24 53
F9DE: 19 A1     348          DFB   $19,$A1    ;(A) FORMAT ABOVE
F9E0: 00 1A 5B  349          DFB   $00,$1A,$5B,$5B,$A5,$69
F9E3: 5B A5 69
F9E6: 24 24     350          DFB   $24,$24    ;(B) FORMAT
F9E8: AE AE A8  351          DFB   $AE,$AE,$A8,$AD,$29,$00
F9EB: AD 29 00
F9EE: 7C 00     352          DFB   $7C,$00    ;(C) FORMAT
F9F0: 15 9C 6D  353          DFB   $15,$9C,$6D,$9C,$A5,$69
F9F3: 9C A5 69
F9F6: 29 53     354          DFB   $29,$53    ;(D) FORMAT
F9F8: 84 13 34  355          DFB   $84,$13,$34,$11,$A5,$69
F9FB: 11 A5 69
F9FE: 23 A0     356          DFB   $23,$A0    ;(E) FORMAT
FA00: D8 62 5A  357  MNEMR   DFB   $D8,$62,$5A,$48,$26,$62
FA03: 48 26 62
FA06: 94 88 54  358          DFB   $94,$88,$54,$44,$C8,$54
FA09: 44 C8 54
FA0C: 68 44 E8  359          DFB   $68,$44,$E8,$94,$00,$B4
FA0F: 94 00 B4
FA12: 08 84 74  360          DFB   $08,$84,$74,$B4,$28,$6E
FA15: B4 28 6E
FA18: 74 F4 CC  361          DFB   $74,$F4,$CC,$4A,$72,$F2
FA1B: 4A 72 F2
FA1E: A4 8A     362          DFB   $A4,$8A    ;(A) FORMAT
FA20: 00 AA A2  363          DFB   $00,$AA,$A2,$A2,$74,$74
FA23: A2 74 74
FA26: 74 72     364          DFB   $74,$72    ;(B) FORMAT
FA28: 44 68 B2  365          DFB   $44,$68,$B2,$32,$B2,$00
FA2B: 32 B2 00
FA2E: 22 00     366          DFB   $22,$00    ;(C) FORMAT
FA30: 1A 1A 26  367          DFB   $1A,$1A,$26,$26,$72,$72
FA33: 26 72 72
FA36: 88 C8     368          DFB   $88,$C8    ;(D) FORMAT
FA38: C4 CA 26  369          DFB   $C4,$CA,$26,$48,$44,$44
FA3B: 48 44 44
FA3E: A2 C8     370          DFB   $A2,$C8    ;(E) FORMAT
FA40: FF FF FF  371          DFB   $FF,$FF,$FF
FA43: 20 D0 F8  372  STEP    JSR   INSTDSP    ;DISASSEMBLE ONE INST
FA46: 68        373          PLA              ;  AT (PCL,H)
FA47: 85 2C     374          STA   RTNL       ;ADJUST TO USER
FA49: 68        375          PLA              ;  STACK. SAVE
FA4A: 85 2D     376          STA   RTNH       ;  RTN ADR.
FA4C: A2 08     377          LDX   #$08
FA4E: BD 10 FB  378  XQINIT  LDA   INITBL-1,X ;INIT XEQ AREA
FA51: 95 3C     379          STA   XQT,X
FA53: CA        380          DEX
FA54: D0 F8     381          BNE   XQINIT
FA56: A1 3A     382          LDA   (PCL,X)    ;USER OPCODE BYTE
FA58: F0 42     383          BEQ   XBRK       ;SPECIAL IF BREAK
FA5A: A4 2F     384          LDY   LENGTH     ;LEN FROM DISASSEMBLY
FA5C: C9 20     385          CMP   #$20
FA5E: F0 59     386          BEQ   XJSR       ;HANDLE JSR, RTS, JMP,
FA60: C9 60     387          CMP   #$60       ;  JMP (), RTI SPECIAL
FA62: F0 45     388          BEQ   XRTS
FA64: C9 4C     389          CMP   #$4C
FA66: F0 5C     390          BEQ   XJMP
FA68: C9 6C     391          CMP   #$6C
FA6A: F0 59     392          BEQ   XJMPAT
FA6C: C9 40     393          CMP   #$40
FA6E: F0 35     394          BEQ   XRTI
```

```
FA70: 29 1F     395            AND   #$1F
FA72: 49 14     396            EOR   #$14
FA74: C9 04     397            CMP   #$04        ;COPY USER INST TO XEQ AREA
FA76: F0 02     398            BEQ   XQ2         ;  WITH TRAILING NOPS
FA78: B1 3A     399   XQ1      LDA   (PCL),Y     ;CHANGE REL BRANCH
FA7A: 99 3C 00  400   XQ2      STA   XQT,Y       ;  DISP TO 4 FOR
FA7D: 88        401            DEY               ;  JMP TO BRANCH OR
FA7E: 10 F8     402            BPL   XQ1         ;  NBRANCH FROM XEQ.
FA80: 20 3F FF  403            JSR   RESTORE     ;RESTORE USER REG CONTENTS.
FA83: 4C 3C 00  404            JMP   XQT         ;XEQ USER OP FROM RAM
FA86: 85 45     405   IRQ      STA   ACC         ;  (RETURN TO NBRANCH)
FA88: 68        406            PLA
FA89: 48        407            PHA               ;**IRQ HANDLER
FA8A: 0A        408            ASL
FA8B: 0A        409            ASL
FA8C: 0A        410            ASL
FA8D: 30 03     411            BMI   BREAK       ;TEST FOR BREAK
FA8F: 6C FE 03  412            JMP   (IRQLOC)    ;USER ROUTINE VECTOR IN RAM
FA92: 28        413   BREAK    PLP
FA93: 20 4C FF  414            JSR   SAV1        ;SAVE REG'S ON BREAK
FA96: 68        415            PLA               ;  INCLUDING PC
FA97: 85 3A     416            STA   PCL
FA99: 68        417            PLA
FA9A: 85 3B     418            STA   PCH
FA9C: 20 82 F8  419   XBRK     JSR   INSDS1      ;PRINT USER PC.
FA9F: 20 DA FA  420            JSR   RGDSP1      ;  AND REG'S
FAA2: 4C 65 FF  421            JMP   MON         ;GO TO MONITOR
FAA5: 18        422   XRTI     CLC
FAA6: 68        423            PLA               ;SIMULATE RTI BY EXPECTING
FAA7: 85 48     424            STA   STATUS      ;  STATUS FROM STACK, THEN RTS
FAA9: 68        425   XRTS     PLA               ;RTS SIMULATION
FAAA: 85 3A     426            STA   PCL         ;  EXTRACT PC FROM STACK
FAAC: 68        427            PLA               ;  AND UPDATE PC BY 1 (LEN=0)
FAAD: 85 3B     428   PCINC2   STA   PCH
FAAF: A5 2F     429   PCINC3   LDA   LENGTH      ;UPDATE PC BY LEN
FAB1: 20 56 F9  430            JSR   PCADJ3
FAB4: 84 3B     431            STY   PCH
FAB6: 18        432            CLC
FAB7: 90 14     433            BCC   NEWPCL
FAB9: 18        434   XJSR     CLC
FABA: 20 54 F9  435            JSR   PCADJ2      ;UPDATE PC AND PUSH
FABD: AA        436            TAX               ;  ONTO STACH FOR
FABE: 98        437            TYA               ;  JSR SIMULATE
FABF: 48        438            PHA
FAC0: 8A        439            TXA
FAC1: 48        440            PHA
FAC2: A0 02     441            LDY   #$02
FAC4: 18        442   XJMP     CLC
FAC5: B1 3A     443   XJMPAT   LDA   (PCL),Y
FAC7: AA        444            TAX               ;LOAD PC FOR JMP,
FAC8: 88        445            DEY               ;  (JMP) SIMULATE.
FAC9: B1 3A     446            LDA   (PCL),Y
FACB: 86 3B     447            STX   PCH
FACD: 85 3A     448   NEWPCL   STA   PCL
FACF: B0 F3     449            BCS   XJMP
FAD1: A5 2D     450   RTNJMP   LDA   RTNH
FAD3: 48        451            PHA
FAD4: A5 2C     452            LDA   RTNL
FAD6: 48        453            PHA
FAD7: 20 8E FD  454   REGDSP   JSR   CROUT       ;DISPLAY USER REG
FADA: A9 45     455   RGDSP1   LDA   #ACC        ;  CONTENTS WITH
FADC: 85 40     456            STA   A3L         ;  LABELS
```

```
FADE: A9 00             LDA   #ACC/256
FAE0: 85 41             STA   A3H
FAE2: A2 FB             LDX   #$FB
FAE4: A9 A0     RDSP1   LDA   #$A0
FAE6: 20 ED FD          JSR   COUT
FAE9: BD 1E FA          LDA   RTBL-$FB,X
FAEC: 20 ED FD          JSR   COUT
FAEF: A9 BD             LDA   #$BD
FAF1: 20 ED FD          JSR   COUT
FAF4: B5 4A             LDA   ACC+5,X
FAF6: 20 DA FD          JSR   PRBYTE
FAF9: E8                INX
FAFA: 30 E8             BMI   RDSP1
FAFC: 60                RTS
FAFD: 18        BRANCH  CLC               ;BRANCH TAKEN,
FAFE: A0 01             LDY   #$01        ;  ADD LEN+2 TO PC
FB00: B1 3A             LDA   (PCL),Y
FB02: 20 56 F9          JSR   PCADJ3
FB05: 85 3A             STA   PCL
FB07: 98                TYA
FB08: 38                SEC
FB09: B0 A2             BCS   PCINC2
FB0B: 20 4A FF  NBRNCH  JSR   SAVE        ;NORMAL RETURN AFTER
FB0E: 38                SEC               ;  XEQ USER OF
FB0F: B0 9E             BCS   PCINC3      ;GO UPDATE PC
FB11: EA        INITBL  NOP
FB12: EA                NOP               ;DUMMY FILL FOR
FB13: 4C 0B FB          JMP   NBRNCH      ;  XEQ AREA
FB16: 4C FD FA          JMP   BRANCH
FB19: C1        RTBL    DFB   $C1
FB1A: D8                DFB   $D8
FB1B: D9                DFB   $D9
FB1C: D0                DFB   $D0
FB1D: D3                DFB   $D3
FB1E: AD 70 C0  PREAD   LDA   PTRIG       ;TRIGGER PADDLES
FB21: A0 00             LDY   #$00        ;INIT COUNT
FB23: EA                NOP               ;COMPENSATE FOR 1ST COUNT
FB24: EA                NOP
FB25: BD 64 C0  PREAD2  LDA   PADDL0,X    ;COUNT Y-REG EVERY
FB28: 10 04             BPL   RTS2D       ;  12 USEC
FB2A: C8                INY
FB2B: D0 F8             BNE   PREAD2      ;  EXIT AT 255 MAX
FB2D: 88                DEY
FB2E: 60        RTS2D   RTS
FB2F: A9 00     INIT    LDA   #$00        ;CLR STATUS FOR DEBUG
FB31: 85 48             STA   STATUS      ;  SOFTWARE
FB33: AD 56 C0          LDA   LORES
FB36: AD 54 C0          LDA   LOWSCR      ;INIT VIDEO MODE
FB39: AD 51 C0  SETTXT  LDA   TXTSET      ;SET FOR TEXT MODE
FB3C: A9 00             LDA   #$00        ;  FULL SCREEN WINDOW
FB3E: F0 0B             BEQ   SETWND
FB40: AD 50 C0  SETGR   LDA   TXTCLR      ;SET FOR GRAPHICS MODE
FB43: AD 53 C0          LDA   MIXSET      ;  LOWER 4 LINES AS
FB46: 20 36 F8          JSR   CLRTOP      ;  TEXT WINDOW
FB49: A9 14             LDA   #$14
FB4B: 85 22     SETWND  STA   WNDTOP      ;SET FOR 40 COL WINDOW
FB4D: A9 00             LDA   #$00        ;  TOP IN A-REG,
FB4F: 85 20             STA   WNDLFT      ;  BTTM AT LINE 24
FB51: A9 28             LDA   #$28
FB53: 85 21             STA   WNDWDTH
FB55: A9 18             LDA   #$18
FB57: 85 23             STA   WNDBTM      ;  VTAB TO ROW 23
```

```
FB59: A9 17              LDA   #$17
FB5B: 85 25     TABV     STA   CV          ;VTABS TO ROW IN A-REG
FB5D: 4C 22 FC           JMP   VTAB
FB60: 20 A4 FB  MULPM    JSR   MD1         ;ABS VAL OF AC AUX
FB63: A0 10     MUL      LDY   #$10        ;INDEX FOR 16 BITS
FB65: A5 50     MUL2     LDA   ACL         ;ACX * AUX + XTND
FB67: 4A                 LSR               ; TO AC, XTND
FB68: 90 0C              BCC   MUL4        ;IF NO CARRY,
FB6A: 18                 CLC               ; NO PARTIAL PROD.
FB6B: A2 FE              LDX   #$FE
FB6D: B5 54     MUL3     LDA   XTNDL+2,X   ;ADD MPLCND (AUX)
FB6F: 75 56              ADC   AUXL+2,X    ; TO PARTIAL PROD
FB71: 95 54              STA   XTNDL+2,X   ; (XTND)
FB73: E8                 INX
FB74: D0 F7              BNE   MUL3
FB76: A2 03     MUL4     LDX   #$03
FB78: 76        MUL5     DFB   $76
FB79: 50                 DFB   $50
FB7A: CA                 DEX
FB7B: 10 FB              BPL   MUL5
FB7D: 88                 DEY
FB7E: D0 E5              BNE   MUL2
FB80: 60                 RTS
FB81: 20 A4 FB  DIVPM    JSR   MD1         ;ABS VAL OF AC, AUX.
FB84: A0 10     DIV      LDY   #$10        ;INDEX FOR 16 BITS
FB86: 06 50     DIV2     ASL   ACL
FB88: 26 51              ROL   ACH
FB8A: 26 52              ROL   XTNDL       ;XTND/AUX
FB8C: 26 53              ROL   XTNDH       ;  TO AC.
FB8E: 38                 SEC
FB8F: A5 52              LDA   XTNDL
FB91: E5 54              SBC   AUXL        ;MOD TO XTND.
FB93: AA                 TAX
FB94: A5 53              LDA   XTNDH
FB96: E5 55              SBC   AUXH
FB98: 90 06              BCC   DIV3
FB9A: 86 52              STX   XTNDL
FB9C: 85 53              STA   XTNDH
FB9E: E6 50              INC   ACL
FBA0: 88        DIV3     DEY
FBA1: D0 E3              BNE   DIV2
FBA3: 60                 RTS
FBA4: A0 00     MD1      LDY   #$00        ;ABS VAL OF AC, AUX
FBA6: 84 2F              STY   SIGN        ;  WITH RESULT SIGN
FBA8: A2 54              LDX   #AUXL       ;  IN LSB OF SIGN.
FBAA: 20 AF FB           JSR   MD3
FBAD: A2 50              LDX   #ACL
FBAF: B5 01     MD3      LDA   LOC1,X      ;X SPECIFIES AC OR AUX
FBB1: 10 0D              BPL   MDRTS
FBB3: 38                 SEC
FBB4: 98                 TYA
FBB5: F5 00              SBC   LOC0,X      ;COMPL SPECIFIED REG
FBB7: 95 00              STA   LOC0,X      ;  IF NEG.
FBB9: 98                 TYA
FBBA: F5 01              SBC   LOC1,X
FBBC: 95 01              STA   LOC1,X
FBBE: E6 2F              INC   SIGN
FBC0: 60        MDRTS    RTS
FBC1: 48        BASCALC  PHA               ;CALC BASE ADR IN BASL,H
FBC2: 4A                 LSR               ;  FOR GIVEN LINE NO
FBC3: 29 03              AND   #$03        ;  0<=LINE NO.<=$17
FBC5: 09 04              ORA   #$04        ;ARG=000ABCDE, GENERATE
```

```
FBC7: 85 29               STA   BASH       ;  BASH=000001CD
FBC9: 68                  PLA              ;  AND
FBCA: 29 18               AND   #$18       ;  BASL=EABAB000
FBCC: 90 02               BCC   BSCLC2
FBCE: 69 7F               ADC   #$7F
FBD0: 85 28      BSCLC2   STA   BASL
FBD2: 0A                  ASL
FBD3: 0A                  ASL
FBD4: 05 28               ORA   BASL
FBD6: 85 28               STA   BASL
FBD8: 60                  RTS
FBD9: C9 87      BELL1    CMP   #$87       ;BELL CHAR? (CNTRL-G)
FBDB: D0 12               BNE   RTS2B      ;  NO, RETURN
FBDD: A9 40               LDA   #$40       ;DELAY .01 SECONDS
FBDF: 20 A8 FC            JSR   WAIT
FBE2: A0 C0               LDY   #$C0
FBE4: A9 0C      BELL2    LDA   #$0C       ;TOGGLE SPEAKER AT
FBE6: 20 A8 FC            JSR   WAIT       ;  1 KHZ FOR .1 SEC.
FBE9: AD 30 C0            LDA   SPKR
FBEC: 88                  DEY
FBED: D0 F5               BNE   BELL2
FBEF: 60         RTS2B    RTS
FBF0: A4 24      STOADV   LDY   CH         ;CURSOR H INDEX TO Y-REG
FBF2: 91 28               STA   (BASL),Y   ;STORE CHAR IN LINE
FBF4: E6 24      ADVANCE  INC   CH         ;INCREMENT CURSOR H INDEX
FBF6: A5 24               LDA   CH         ;  (MOVE RIGHT)
FBF8: C5 21               CMP   WNDWDTH    ;BEYOND WINDOW WIDTH?
FBFA: B0 66               BCS   CR         ;  YES CR TO NEXT LINE
FBFC: 60         RTS3     RTS              ;  NO,RETURN
FBFD: C9 A0      VIDOUT   CMP   #$A0       ;CONTROL CHAR?
FBFF: B0 EF               BCS   STOADV     ;  NO,OUTPUT IT.
FC01: A8                  TAY              ;INVERSE VIDEO?
FC02: 10 EC               BPL   STOADV     ;  YES, OUTPUT IT.
FC04: C9 8D               CMP   #$8D       ;CR?
FC06: F0 5A               BEQ   CR         ;  YES.
FC08: C9 8A               CMP   #$8A       ;LINE FEED?
FC0A: F0 5A               BEQ   LF         ;  IF SO, DO IT.
FC0C: C9 88               CMP   #$88       ;BACK SPACE? (CNTRL-H)
FC0E: D0 C9               BNE   BELL1      ;  NO, CHECK FOR BELL.
FC10: C6 24      BS       DEC   CH         ;DECREMENT CURSOR H INDEX
FC12: 10 E8               BPL   RTS3       ;IF POS, OK. ELSE MOVE UP
FC14: A5 21               LDA   WNDWDTH    ;SET CH TO WNDWDTH-1
FC16: 85 24               STA   CH
FC18: C6 24               DEC   CH         ;(RIGHTMOST SCREEN POS)
FC1A: A5 22      UP       LDA   WNDTOP     ;CURSOR V INDEX
FC1C: C5 25               CMP   CV
FC1E: B0 0B               BCS   RTS4       ;IF TOP LINE THEN RETURN
FC20: C6 25               DEC   CV         ;DEC CURSOR V-INDEX
FC22: A5 25      VTAB     LDA   CV         ;GET CURSOR V-INDEX
FC24: 20 C1 FB   VTABZ    JSR   BASCALC    ;GENERATE BASE ADR
FC27: 65 20               ADC   WNDLFT     ;ADD WINDOW LEFT INDEX
FC29: 85 28               STA   BASL       ;TO BASL
FC2B: 60         RTS4     RTS
FC2C: 49 C0      ESC1     EOR   #$C0       ;ESC?
FC2E: F0 28               BEQ   HOME       ;  IF SO, DO HOME AND CLEAR
FC30: 69 FD               ADC   #$FD       ;ESC-A OR B CHECK
FC32: 90 C0               BCC   ADVANCE    ;  A, ADVANCE
FC34: F0 DA               BEQ   BS         ;  B, BACKSPACE
FC36: 69 FD               ADC   #$FD       ;ESC-C OR D CHECK
FC38: 90 2C               BCC   LF         ;  C, DOWN
FC3A: F0 DE               BEQ   UP         ;  D, GO UP
FC3C: 69 FD               ADC   #$FD       ;ESC-E OR F CHECK
```

```
FC3E: 90 5C            BCC   CLREOL      ;  E, CLEAR TO END OF LINE
FC40: D0 E9            BNE   RTS4        ;  NOT F, RETURN
FC42: A4 24    CLREOP  LDY   CH          ;CURSOR H TO Y INDEX
FC44: A5 25            LDA   CV          ;CURSOR V TO A-REGISTER
FC46: 48       CLEOP1  PHA               ;SAVE CURRENT LINE ON STK
FC47: 20 24 FC         JSR   VTABZ       ;CALC BASE ADDRESS
FC4A: 20 9E FC         JSR   CLEOLZ      ;CLEAR TO EOL, SET CARRY
FC4D: A0 00            LDY   #$00        ;CLEAR FROM H INDEX=0 FOR REST
FC4F: 68               PLA               ;INCREMENT CURRENT LINE
FC50: 69 00            ADC   #$00        ;(CARRY IS SET)
FC52: C5 23            CMP   WNDBTM      ;DONE TO BOTTOM OF WINDOW?
FC54: 90 F0            BCC   CLEOP1      ;  NO, KEEP CLEARING LINES
FC56: B0 CA            BCS   VTAB        ;  YES, TAB TO CURRENT LINE
FC58: A5 22    HOME    LDA   WNDTOP      ;INIT CURSOR V
FC5A: 85 25            STA   CV          ;  AND H-INDICES
FC5C: A0 00            LDY   #$00
FC5E: 84 24            STY   CH          ;THEN CLEAR TO END OF PAGE
FC60: F0 E4            BEQ   CLEOP1
FC62: A9 00    CR      LDA   #$00        ;CURSOR TO LEFT OF INDEX
FC64: 85 24            STA   CH          ;(RET CURSOR H=0)
FC66: E6 25    LF      INC   CV          ;INCR CURSOR V(DOWN 1 LINE)
FC68: A5 25            LDA   CV
FC6A: C5 23            CMP   WNDBTM      ;OFF SCREEN?
FC6C: 90 B6            BCC   VTABZ       ;  NO, SET BASE ADDR
FC6E: C6 25            DEC   CV          ;DECR CURSOR V (BACK TO BOTTOM)
FC70: A5 22    SCROLL  LDA   WNDTOP      ;START AT TOP OF SCRL WNDW
FC72: 48               PHA
FC73: 20 24 FC         JSR   VTABZ       ;GENERATE BASE ADR
FC76: A5 28    SCRL1   LDA   BASL        ;COPY BASL,H
FC78: 85 2A            STA   BAS2L       ;  TO BAS2L,H
FC7A: A5 29            LDA   BASH
FC7C: 85 2B            STA   BAS2H
FC7E: A4 21            LDY   WNDWDTH     ;INIT Y TO RIGHTMOST INDEX
FC80: 88               DEY               ;  OF SCROLLING WINDOW
FC81: 68               PLA
FC82: 69 01            ADC   #$01        ;INCR LINE NUMBER
FC84: C5 23            CMP   WNDBTM      ;DONE?
FC86: B0 0D            BCS   SCRL3       ;  YES, FINISH
FC88: 48               PHA
FC89: 20 24 FC         JSR   VTABZ       ;FORM BASL,H (BASE ADDR)
FC8C: B1 28    SCRL2   LDA   (BASL),Y    ;MOVE A CHR UP ON LINE
FC8E: 91 2A            STA   (BAS2L),Y
FC90: 88               DEY               ;NEXT CHAR OF LINE
FC91: 10 F9            BPL   SCRL2
FC93: 30 E1            BMI   SCRL1       ;NEXT LINE (ALWAYS TAKEN)
FC95: A0 00    SCRL3   LDY   #$00        ;CLEAR BOTTOM LINE
FC97: 20 9E FC         JSR   CLEOLZ      ;GET BASE ADDR FOR BOTTOM LINE
FC9A: B0 86            BCS   VTAB        ;CARRY IS SET
FC9C: A4 24    CLREOL  LDY   CH          ;CURSOR H INDEX
FC9E: A9 A0    CLEOLZ  LDA   #$A0
FCA0: 91 28    CLEOL2  STA   (BASL),Y    ;STORE BLANKS FROM 'HERE'
FCA2: C8               INY               ;  TO END OF LINES (WNDWDTH)
FCA3: C4 21            CPY   WNDWDTH
FCA5: 90 F9            BCC   CLEOL2
FCA7: 60               RTS
FCA8: 38       WAIT    SEC
FCA9: 48       WAIT2   PHA
FCAA: E9 01    WAIT3   SBC   #$01
FCAC: D0 FC            BNE   WAIT3       ;1.0204 USEC
FCAE: 68               PLA               ;(13+27/2*A+5/2*A*A)
FCAF: E9 01            SBC   #$01
FCB1: D0 F6            BNE   WAIT2
```

```
FCB3: 60                      RTS
FCB4: E6 42         NXTA4     INC   A4L         ;INCR 2-BYTE A4
FCB6: D0 02                   BNE   NXTA1       ;  AND A1
FCB8: E6 43                   INC   A4H
FCBA: A5 3C         NXTA1     LDA   A1L         ;INCR 2-BYTE A1.
FCBC: C5 3E                   CMP   A2L
FCBE: A5 3D                   LDA   A1H         ;  AND COMPARE TO A2
FCC0: E5 3F                   SBC   A2H
FCC2: E6 3C                   INC   A1L         ;  (CARRY SET IF >=)
FCC4: D0 02                   BNE   RTS4B
FCC6: E6 3D                   INC   A1H
FCC8: 60            RTS4B     RTS
FCC9: A0 4B         HEADR     LDY   #$4B        ;WRITE A*256 'LONG 1'
FCCB: 20 DB FC                JSR   ZERDLY      ;  HALF CYCLES
FCCE: D0 F9                   BNE   HEADR       ;  (650 USEC EACH)
FCD0: 69 FE                   ADC   #$FE
FCD2: B0 F5                   BCS   HEADR       ;THEN A 'SHORT 0'
FCD4: A0 21                   LDY   #$21        ;  (400 USEC)
FCD6: 20 DB FC      WRBIT     JSR   ZERDLY      ;WRITE TWO HALF CYCLES
FCD9: C8                      INY               ;  OF 250 USEC ('0')
FCDA: C8                      INY               ;  OR 500 USEC ('0')
FCDB: 88            ZERDLY    DEY
FCDC: D0 FD                   BNE   ZERDLY
FCDE: 90 05                   BCC   WRTAPE      ;Y IS COUNT FOR
FCE0: A0 32                   LDY   #$32        ;  TIMING LOOP
FCE2: 88            ONEDLY    DEY
FCE3: D0 FD                   BNE   ONEDLY
FCE5: AC 20 C0      WRTAPE    LDY   TAPEOUT
FCE8: A0 2C                   LDY   #$2C
FCEA: CA                      DEX
FCEB: 60                      RTS
FCEC: A2 08         RDBYTE    LDX   #$08        ;8 BITS TO READ
FCEE: 48            RDBYT2    PHA               ;READ TWO TRANSITIONS
FCEF: 20 FA FC                JSR   RD2BIT      ;  (FIND EDGE)
FCF2: 68                      PLA
FCF3: 2A                      ROL               ;NEXT BIT
FCF4: A0 3A                   LDY   #$3A        ;COUNT FOR SAMPLES
FCF6: CA                      DEX
FCF7: D0 F5                   BNE   RDBYT2
FCF9: 60                      RTS
FCFA: 20 FD FC      RD2BIT    JSR   RDBIT
FCFD: 88            RDBIT     DEY               ;DECR Y UNTIL
FCFE: AD 60 C0                LDA   TAPEIN      ; TAPE TRANSITION
FD01: 45 2F                   EOR   LASTIN
FD03: 10 F8                   BPL   RDBIT
FD05: 45 2F                   EOR   LASTIN
FD07: 85 2F                   STA   LASTIN
FD09: C0 80                   CPY   #$80        ;SET CARRY ON Y
FD0B: 60                      RTS
FD0C: A4 24         RDKEY     LDY   CH
FD0E: B1 28                   LDA   (BASL),Y    ;SET SCREEN TO FLASH
FD10: 48                      PHA
FD11: 29 3F                   AND   #$3F
FD13: 09 40                   ORA   #$40
FD15: 91 28                   STA   (BASL),Y
FD17: 68                      PLA
FD18: 6C 38 00                JMP   (KSWL)      ;GO TO USER KEY-IN
FD1B: E6 4E         KEYIN     INC   RNDL
FD1D: D0 02                   BNE   KEYIN2      ;INCR RND NUMBER
FD1F: E6 4F                   INC   RNDH
FD21: 2C 00 C0      KEYIN2    BIT   KBD         ;KEY DOWN?
FD24: 10 F5                   BPL   KEYIN       ;  LOOP
```

```
FD26: 91 28               STA   (BASL),Y    ;REPLACE FLASHING SCREEN
FD28: AD 00 C0            LDA   KBD         ;GET KEYCODE
FD2B: 2C 10 C0            BIT   KBDSTRB     ;CLR KEY STROBE
FD2E: 60                  RTS
FD2F: 20 0C FD   ESC      JSR   RDKEY       ;GET KEYCODE
FD32: 20 2C FC            JSR   ESC1        ;  HANDLE ESC FUNC.
FD35: 20 0C FD   RDCHAR   JSR   RDKEY       ;READ KEY
FD38: C9 9B               CMP   #$9B        ;ESC?
FD3A: F0 F3               BEQ   ESC         ;  YES, DON'T RETURN
FD3C: 60                  RTS
FD3D: A5 32      NOTCR    LDA   INVFLG
FD3F: 48                  PHA
FD40: A9 FF               LDA   #$FF
FD42: 85 32               STA   INVFLG      ;ECHO USER LINE
FD44: BD 00 02            LDA   IN,X        ;  NON INVERSE
FD47: 20 ED FD            JSR   COUT
FD4A: 68                  PLA
FD4B: 85 32               STA   INVFLG
FD4D: BD 00 02            LDA   IN,X
FD50: C9 88               CMP   #$88        ;CHECK FOR EDIT KEYS
FD52: F0 1D               BEQ   BCKSPC      ;  BS, CTRL-X
FD54: C9 98               CMP   #$98
FD56: F0 0A               BEQ   CANCEL
FD58: E0 F8               CPX   #$F8        ;MARGIN?
FD5A: 90 03               BCC   NOTCR1
FD5C: 20 3A FF            JSR   BELL        ;  YES, SOUND BELL
FD5F: E8         NOTCR1   INX               ;ADVANCE INPUT INDEX
FD60: D0 13               BNE   NXTCHAR
FD62: A9 DC      CANCEL   LDA   #$DC        ;BACKSLASH AFTER CANCELLED LINE
FD64: 20 ED FD            JSR   COUT
FD67: 20 8E FD   GETLNZ   JSR   CROUT       ;OUTPUT CR
FD6A: A5 33      GETLN    LDA   PROMPT
FD6C: 20 ED FD            JSR   COUT        ;OUTPUT PROMPT CHAR
FD6F: A2 01               LDX   #$01        ;INIT INPUT INDEX
FD71: 8A         BCKSPC   TXA               ;  WILL BACKSPACE TO 0
FD72: F0 F3               BEQ   GETLNZ
FD74: CA                  DEX
FD75: 20 35 FD   NXTCHAR  JSR   RDCHAR
FD78: C9 95               CMP   #PICK       ;USE SCREEN CHAR
FD7A: D0 02               BNE   CAPTST      ;  FOR CTRL-U
FD7C: B1 28               LDA   (BASL),Y
FD7E: C9 E0      CAPTST   CMP   #$E0
FD80: 90 02               BCC   ADDINP      ;CONVERT TO CAPS
FD82: 29 DF               AND   #$DF
FD84: 9D 00 02   ADDINP   STA   IN,X        ;ADD TO INPUT BUF
FD87: C9 8D               CMP   #$8D
FD89: D0 B2               BNE   NOTCR
FD8B: 20 9C FC            JSR   CLREOL      ;CLR TO EOL IF CR
FD8E: A9 8D      CROUT    LDA   #$8D
FD90: D0 5B               BNE   COUT
FD92: A4 3D      PRA1     LDY   A1H         ;PRINT CR,A1 IN HEX
FD94: A6 3C               LDX   A1L
FD96: 20 8E FD   PRYX2    JSR   CROUT
FD99: 20 40 F9            JSR   PRNTYX
FD9C: A0 00               LDY   #$00
FD9E: A9 AD               LDA   #$AD        ;PRINT '-'
FDA0: 4C ED FD            JMP   COUT
FDA3: A5 3C      XAM8     LDA   A1L
FDA5: 09 07               ORA   #$07        ;SET TO FINISH AT
FDA7: 85 3E               STA   A2L         ;  MOD 8=7
FDA9: A5 3D               LDA   A1H
FDAB: 85 3F               STA   A2H
```

```
FDAD: A5 3C     MODSCHK  LDA   A1L
FDAF: 29 07              AND   #$07
FDB1: D0 03              BNE   DATAOUT
FDB3: 20 92 FD  XAM      JSR   PRA1
FDB6: A9 A0     DATAOUT  LDA   #$A0
FDB8: 20 ED FD           JSR   COUT        ;OUTPUT BLANK
FDBB: B1 3C              LDA   (A1L),Y
FDBD: 20 DA FD           JSR   PRBYTE      ;OUTPUT BYTE IN HEX
FDC0: 20 BA FC           JSR   NXTA1
FDC3: 90 E8              BCC   MODSCHK     ;CHECK IF TIME TO,
FDC5: 60        RTS4C    RTS               ;  PRINT ADDR
FDC6: 4A        XAMPM    LSR               ;DETERMINE IF MON
FDC7: 90 EA              BCC   XAM         ;  MODE IS XAM
FDC9: 4A                 LSR               ;  ADD, OR SUB
FDCA: 4A                 LSR
FDCB: A5 3E              LDA   A2L
FDCD: 90 02              BCC   ADD
FDCF: 49 FF              EOR   #$FF        ;SUB: FORM 2'S COMPLEMENT
FDD1: 65 3C     ADD      ADC   A1L
FDD3: 48                 PHA
FDD4: A9 BD              LDA   #$BD
FDD6: 20 ED FD           JSR   COUT        ;PRINT '=', THEN RESULT
FDD9: 68                 PLA
FDDA: 48        PRBYTE   PHA               ;PRINT BYTE AS 2 HEX
FDDB: 4A                 LSR               ;  DIGITS, DESTROYS A-REG
FDDC: 4A                 LSR
FDDD: 4A                 LSR
FDDE: 4A                 LSR
FDDF: 20 E5 FD           JSR   PRHEXZ
FDE2: 68                 PLA
FDE3: 29 0F     PRHEX    AND   #$0F        ;PRINT HEX DIG IN A-REG
FDE5: 09 B0     PRHEXZ   ORA   #$B0        ;  LSB'S
FDE7: C9 BA              CMP   #$BA
FDE9: 90 02              BCC   COUT
FDEB: 69 06              ADC   #$06
FDED: 6C 36 00  COUT     JMP   (CSWL)      ;VECTOR TO USER OUTPUT ROUTINE
FDF0: C9 A0     COUT1    CMP   #$A0
FDF2: 90 02              BCC   COUTZ       ;DON'T OUTPUT CTRL'S INVERSE
FDF4: 25 32              AND   INVFLG      ;MASK WITH INVERSE FLAG
FDF6: 84 35     COUTZ    STY   YSAV1       ;SAV Y-REG
FDF8: 48                 PHA               ;SAV A-REG
FDF9: 20 FD FB           JSR   VIDOUT      ;OUTPUT A-REG AS ASCII
FDFC: 68                 PLA               ;RESTORE A-REG
FDFD: A4 35              LDY   YSAV1       ;  AND Y-REG
FDFF: 60                 RTS               ;  THEN RETURN
FE00: C6 34     BL1      DEC   YSAV
FE02: F0 9F              BEQ   XAM8
FE04: CA        BLANK    DEX               ;BLANK TO MON
FE05: D0 16              BNE   SETMDZ      ;AFTER BLANK
FE07: C9 BA              CMP   #$BA        ;DATA STORE MODE?
FE09: D0 BB              BNE   XAMPM       ;  NO, XAM, ADD, OR SUB
FE0B: 85 31     STOR     STA   MODE        ;KEEP IN STORE MODE
FE0D: A5 3E              LDA   A2L
FE0F: 91 40              STA   (A3L),Y     ;STORE AS LOW BYTE AS (A3)
FE11: E6 40              INC   A3L
FE13: D0 02              BNE   RTS5        ;INCR A3, RETURN
FE15: E6 41              INC   A3H
FE17: 60        RTS5     RTS
FE18: A4 34     SETMODE  LDY   YSAV        ;SAVE CONVERTED ':', '+',
FE1A: B9 FF 01           LDA   IN-1,Y      ;  '-', '.' AS MODE.
FE1D: 85 31     SETMDZ   STA   MODE
FE1F: 60                 RTS
```

```
FE20: A2 01     LT       LDX   #$01
FE22: B5 3E     LT2      LDA   A2L,X       ;COPY A2 (2 BYTES) TO
FE24: 95 42              STA   A4L,X       ;  A4 AND A5
FE26: 95 44              STA   A5L,X
FE28: CA                 DEX
FE29: 10 F7              BPL   LT2
FE2B: 60                 RTS
FE2C: B1 3C     MOVE     LDA   (A1L),Y     ;MOVE (A1 TO A2) TO
FE2E: 91 42              STA   (A4L),Y     ;  (A4)
FE30: 20 B4 FC           JSR   NXTA4
FE33: 90 F7              BCC   MOVE
FE35: 60                 RTS
FE36: B1 3C     VFY      LDA   (A1L),Y     ;VERIFY (A1 TO A2) WITH
FE38: D1 42              CMP   (A4L),Y     ;  (A4)
FE3A: F0 1C              BEQ   VFYOK
FE3C: 20 92 FD           JSR   PRA1
FE3F: B1 3C              LDA   (A1L),Y
FE41: 20 DA FD           JSR   PRBYTE
FE44: A9 A0              LDA   #$A0
FE46: 20 ED FD           JSR   COUT
FE49: A9 A8              LDA   #$A8
FE4B: 20 ED FD           JSR   COUT
FE4E: B1 42              LDA   (A4L),Y
FE50: 20 DA FD           JSR   PRBYTE
FE53: A9 A9              LDA   #$A9
FE55: 20 ED FD           JSR   COUT
FE58: 20 B4 FC  VFYOK    JSR   NXTA4
FE5B: 90 D9              BCC   VFY
FE5D: 60                 RTS
FE5E: 20 75 FE  LIST     JSR   A1PC        ;MOVE A1 (2 BYTES) TO
FE61: A9 14              LDA   #$14        ;  PC IF SPEC'D AND
FE63: 48        LIST2    PHA               ;  DISEMBLE 20 INSTRS
FE64: 20 D0 F8           JSR   INSTDSP
FE67: 20 53 F9           JSR   PCADJ       ;ADJUST PC EACH INSTR
FE6A: 85 3A              STA   PCL
FE6C: 84 3B              STY   PCH
FE6E: 68                 PLA
FE6F: 38                 SEC
FE70: E9 01              SBC   #$01        ;NEXT OF 20 INSTRS
FE72: D0 EF              BNE   LIST2
FE74: 60                 RTS
FE75: 8A        A1PC     TXA               ;IF USER SPEC'D ADR
FE76: F0 07              BEQ   A1PCRTS     ;  COPY FROM A1 TO PC
FE78: B5 3C     A1PCLP   LDA   A1L,X
FE7A: 95 3A              STA   PCL,X
FE7C: CA                 DEX
FE7D: 10 F9              BPL   A1PCLP
FE7F: 60        A1PCRTS  RTS
FE80: A0 3F     SETINV   LDY   #$3F        ;SET FOR INVERSE VID
FE82: D0 02              BNE   SETIFLG     ; VIA COUT1
FE84: A0 FF     SETNORM  LDY   #$FF        ;SET FOR NORMAL VID
FE86: 84 32     SETIFLG  STY   INVFLG
FE88: 60                 RTS
FE89: A9 00     SETKBD   LDA   #$00        ;SIMULATE PORT #0 INPUT
FE8B: 85 3E     INPORT   STA   A2L         ;  SPECIFIED (KEYIN ROUTINE)
FE8D: A2 38     INPRT    LDX   #KSWL
FE8F: A0 1B              LDY   #KEYIN
FE91: D0 08              BNE   IOPRT
FE93: A9 00     SETVID   LDA   #$00        ;SIMULATE PORT #0 OUTPUT
FE95: 85 3E     OUTPORT  STA   A2L         ;  SPECIFIED (COUT1 ROUTINE)
FE97: A2 36     OUTPRT   LDX   #CSWL
FE99: A0 F0              LDY   #COUT1
```

```
FE9B: A5 3E     IOPRT    LDA   A2L         ;SET RAM IN/OUT VECTORS
FE9D: 29 0F              AND   #$0F
FE9F: F0 06              BEQ   IOPRT1
FEA1: 09 C0              ORA   #IOADR/256
FEA3: A0 00              LDY   #$00
FEA5: F0 02              BEQ   IOPRT2
FEA7: A9 FD     IOPRT1   LDA   #COUT1/256
FEA9: 94 00     IOPRT2   STY   LOC0,X
FEAB: 95 01              STA   LOC1,X
FEAD: 60                 RTS
FEAE: EA                 NOP
FEAF: EA                 NOP
FEB0: 4C 00 E0  XBASIC   JMP   BASIC       ;TO BASIC WITH SCRATCH
FEB3: 4C 03 E0  BASCONT  JMP   BASIC2      ;CONTINUE BASIC
FEB6: 20 75 FE  GO       JSR   A1PC        ;ADR TO PC IF SPEC'D
FEB9: 20 3F FF           JSR   RESTORE     ;RESTORE META REGS
FEBC: 6C 3A 00           JMP   (PCL)       ;GO TO USER SUBR
FEBF: 4C D7 FA  REGZ     JMP   REGDSP      ;TO REG DISPLAY
FEC2: C6 34     TRACE    DEC   YSAV
FEC4: 20 75 FE  STEPZ    JSR   A1PC        ;ADR TO PC IF SPEC'D
FEC7: 4C 43 FA           JMP   STEP        ;TAKE ONE STEP
FECA: 4C F8 03  USR      JMP   USRADR      ;TO USR SUBR AT USRADR
FECD: A9 40     WRITE    LDA   #$40
FECF: 20 C9 FC           JSR   HEADR       ;WRITE 10-SEC HEADER
FED2: A0 27              LDY   #$27
FED4: A2 00     WR1      LDX   #$00
FED6: 41 3C              EOR   (A1L,X)
FED8: 48                 PHA
FED9: A1 3C              LDA   (A1L,X)
FEDB: 20 ED FE           JSR   WRBYTE
FEDE: 20 BA FC           JSR   NXTA1
FEE1: A0 1D              LDY   #$1D
FEE3: 68                 PLA
FEE4: 90 EE              BCC   WR1
FEE6: A0 22              LDY   #$22
FEE8: 20 ED FE           JSR   WRBYTE
FEEB: F0 4D              BEQ   BELL
FEED: A2 10     WRBYTE   LDX   #$10
FEEF: 0A        WRBYT2   ASL
FEF0: 20 D6 FC           JSR   WRBIT
FEF3: D0 FA              BNE   WRBYT2
FEF5: 60                 RTS
FEF6: 20 00 FE  CRMON    JSR   BL1         ;HANDLE A CR AS BLANK
FEF9: 68                 PLA               ;  THEN POP STACK
FEFA: 68                 PLA               ;  AND RTN TO MON
FEFB: D0 6C              BNE   MONZ
FEFD: 20 FA FC  READ     JSR   RD2BIT      ;FIND TAPEIN EDGE
FF00: A9 16              LDA   #$16
FF02: 20 C9 FC           JSR   HEADR       ;DELAY 3.5 SECONDS
FF05: 85 2E              STA   CHKSUM      ;INIT CHKSUM=$FF
FF07: 20 FA FC           JSR   RD2BIT      ;FIND TAPEIN EDGE
FF0A: A0 24     RD2      LDY   #$24        ;LOOK FOR SYNC BIT
FF0C: 20 FD FC           JSR   RDBIT       ;  (SHORT 0)
FF0F: B0 F9              BCS   RD2         ;  LOOP UNTIL FOUND
FF11: 20 FD FC           JSR   RDBIT       ;SKIP SECOND SYNC H-CYCLE
FF14: A0 3B              LDY   #$3B        ;INDEX FOR 0/1 TEST
FF16: 20 EC FC  RD3      JSR   RDBYTE      ;READ A BYTE
FF19: 81 3C              STA   (A1L,X)     ;STORE AT (A1)
FF1B: 45 2E              EOR   CHKSUM
FF1D: 85 2E              STA   CHKSUM      ;UPDATE RUNNING CHKSUM
FF1F: 20 BA FC           JSR   NXTA1       ;INC A1, COMPARE TO A2
FF22: A0 35              LDY   #$35        ;COMPENSATE 0/1 INDEX
```

```
FF24: 90 F0     1015          BCC   RD3         ;LOOP UNTIL DONE
FF26: 20 EC FC  1016          JSR   RDBYTE      ;READ CHKSUM BYTE
FF29: C5 2E     1017          CMP   CHKSUM
FF2B: F0 0D     1018          BEQ   BELL        ;GOOD, SOUND BELL AND RETURN
FF2D: A9 C5     1019 PRERR    LDA   #$C5
FF2F: 20 ED FD  1020          JSR   COUT        ;PRINT "ERR", THEN BELL
FF32: A9 D2     1021          LDA   #$D2
FF34: 20 ED FD  1022          JSR   COUT
FF37: 20 ED FD  1023          JSR   COUT
FF3A: A9 87     1024 BELL     LDA   #$87        ;OUTPUT BELL AND RETURN
FF3C: 4C ED FD  1025          JMP   COUT
FF3F: A5 48     1026 RESTORE  LDA   STATUS      ;RESTORE 6502 REG CONTENTS
FF41: 48        1027          PHA               ;  USED BY DEBUG SOFTWARE
FF42: A5 45     1028          LDA   ACC
FF44: A6 46     1029 RESTR1   LDX   XREG
FF46: A4 47     1030          LDY   YREG
FF48: 28        1031          PLP
FF49: 60        1032          RTS
FF4A: 85 45     1033 SAVE     STA   ACC         ;SAVE 6502 REG CONTENTS
FF4C: 86 46     1034 SAV1     STX   XREG
FF4E: 84 47     1035          STY   YREG
FF50: 08        1036          PHP
FF51: 68        1037          PLA
FF52: 85 48     1038          STA   STATUS
FF54: BA        1039          TSX
FF55: 86 49     1040          STX   SPNT
FF57: D8        1041          CLD
FF58: 60        1042          RTS
FF59: 20 84 FE  1043 RESET    JSR   SETNORM     ;SET SCREEN MODE
FF5C: 20 2F FB  1044          JSR   INIT        ;  AND INIT KBD/SCREEN
FF5F: 20 93 FE  1045          JSR   SETVID      ;  AS I/O DEV'S
FF62: 20 89 FE  1046          JSR   SETKBD
FF65: D8        1047 MON      CLD               ;MUST SET HEX MODE!
FF66: 20 3A FF  1048          JSR   BELL
FF69: A9 AA     1049 MONZ     LDA   #$AA        ;'*' PROMPT FOR MON
FF6B: 85 33     1050          STA   PROMPT
FF6D: 20 67 FD  1051          JSR   GETLNZ      ;READ A LINE
FF70: 20 C7 FF  1052          JSR   ZMODE       ;CLEAR MON MODE, SCAN IDX
FF73: 20 A7 FF  1053 NXTITM   JSR   GETNUM      ;GET ITEM, NON-HEX
FF76: 84 34     1054          STY   YSAV        ;  CHAR IN A-REG
FF78: A0 17     1055          LDY   #$17        ;  X-REG=0 IF NO HEX INPUT
FF7A: 88        1056 CHRSRCH  DEY
FF7B: 30 E8     1057          BMI   MON         ;NOT FOUND, GO TO MON
FF7D: D9 CC FF  1058          CMP   CHRTBL,Y    ;FIND CMND CHAR IN TEL
FF80: D0 F8     1059          BNE   CHRSRCH
FF82: 20 BE FF  1060          JSR   TOSUB       ;FOUND, CALL CORRESPONDING
FF85: A4 34     1061          LDY   YSAV        ;  SUBROUTINE
FF87: 4C 73 FF  1062          JMP   NXTITM
FF8A: A2 03     1063 DIG      LDX   #$03
FF8C: 0A        1064          ASL
FF8D: 0A        1065          ASL               ;GOT HEX DIG,
FF8E: 0A        1066          ASL               ;  SHIFT INTO A2
FF8F: 0A        1067          ASL
FF90: 0A        1068 NXTBIT   ASL
FF91: 26 3E     1069          ROL   A2L
FF93: 26 3F     1070          ROL   A2H
FF95: CA        1071          DEX               ;LEAVE X=$FF IF DIG
FF96: 10 F8     1072          BPL   NXTBIT
FF98: A5 31     1073 NXTBAS   LDA   MODE
FF9A: D0 06     1074          BNE   NXTBS2      ;IF MODE IS ZERO
FF9C: B5 3F     1075          LDA   A2H,X       ; THEN COPY A2 TO
FF9E: 95 3D     1076          STA   A1H,X       ; A1 AND A3
```

```
FFA0: 95 41               STA  A3H,X
FFA2: E8         NXTBS2   INX
FFA3: F0 F3               BEQ  NXTBAS
FFA5: D0 06               BNE  NXTCHR
FFA7: A2 00      GETNUM   LDX  #$00        ;CLEAR A2
FFA9: 86 3E               STX  A2L
FFAB: 86 3F               STX  A2H
FFAD: B9 00 02   NXTCHR   LDA  IN,Y        ;GET CHAR
FFB0: C8                  INY
FFB1: 49 B0               EOR  #$B0
FFB3: C9 0A               CMP  #$0A
FFB5: 90 D3               BCC  DIG         ;IF HEX DIG, THEN
FFB7: 69 88               ADC  #$88
FFB9: C9 FA               CMP  #$FA
FFBB: B0 CD               BCS  DIG
FFBD: 60                  RTS
FFBE: A9 FE      TOSUB    LDA  #GO/256     ;PUSH HIGH-ORDER
FFC0: 48                  PHA              ;  SUBR ADR ON STK
FFC1: B9 E3 FF            LDA  SUBTBL,Y    ;PUSH LOW-ORDER
FFC4: 48                  PHA              ;  SUBR ADR ON STK
FFC5: A5 31               LDA  MODE
FFC7: A0 00      ZMODE    LDY  #$00        ;CLR MODE, OLD MODE
FFC9: 84 31               STY  MODE        ;  TO A-REG
FFCB: 60                  RTS              ; GO TO SUBR VIA RTS
FFCC: BC         CHRTBL   DFB  $BC         ;F("CTRL-C")
FFCD: B2                  DFB  $B2         ;F("CTRL-Y")
FFCE: BE                  DFB  $BE         ;F("CTRL-E")
FFCF: ED                  DFB  $ED         ;F("T")
FFD0: EF                  DFB  $EF         ;F("V")
FFD1: C4                  DFB  $C4         ;F("CTRL-K")
FFD2: EC                  DFB  $EC         ;F("S")
FFD3: A9                  DFB  $A9         ;F("CTRL-P")
FFD4: BB                  DFB  $BB         ;F("CTRL-B")
FFD5: A6                  DFB  $A6         ;F("-")
FFD6: A4                  DFB  $A4         ;F("+")
FFD7: 06                  DFB  $06         ;F("M") (F=EX-OR $B0+$89)
FFD8: 95                  DFB  $95         ;F("<")
FFD9: 07                  DFB  $07         ;F("N")
FFDA: 02                  DFB  $02         ;F("I")
FFDB: 05                  DFB  $05         ;F("L")
FFDC: F0                  DFB  $F0         ;F("W")
FFDD: 00                  DFB  $00         ;F("G")
FFDE: EB                  DFB  $EB         ;F("R")
FFDF: 93                  DFB  $93         ;F(":")
FFE0: A7                  DFB  $A7         ;F(".")
FFE1: C6                  DFB  $C6         ;F("CR")
FFE2: 99                  DFB  $99         ;F(BLANK)
FFE3: B2         SUBTBL   DFB  BASCONT-1
FFE4: C9                  DFB  USR-1
FFE5: BE                  DFB  REGZ-1
FFE6: C1                  DFB  TRACE-1
FFE7: 35                  DFB  VFY-1
FFE8: 8C                  DFB  INPRT-1
FFE9: C3                  DFB  STEPZ-1
FFEA: 96                  DFB  OUTPRT-1
FFEB: AF                  DFB  XBASIC-1
FFEC: 17                  DFB  SETMODE-1
FFED: 17                  DFB  SETMODE-1
FFEE: 2B                  DFB  MOVE-1
FFEF: 1F                  DFB  LT-1
FFF0: 83                  DFB  SETNORM-1
FFF1: 7F                  DFB  SETINV-1
```

```
FFF2: 5D                DFB   LIST-1
FFF3: CC                DFB   WRITE-1
FFF4: B5                DFB   GO-1
FFF5: FC                DFB   READ-1
FFF6: 17                DFB   SETMODE-1
FFF7: 17                DFB   SETMODE-1
FFF8: F5                DFB   CRMON-1
FFF9: 03                DFB   BLANK-1
FFFA: FB                DFB   NMI         ;NMI VECTOR
FFFB: 03                DFB   NMI/256
FFFC: 59                DFB   RESET       ;RESET VECTOR
FFFD: FF                DFB   RESET/256
FFFE: 86                DFB   IRQ         ;IRQ VECTOR
FFFF: FA                DFB   IRQ/256
                XQTNZ   EQU   $3C
```

```
+--------------------------------------------------------------------
|  TOPIC -- Apple II -- Red Book Sweet-16 listing
+--------------------------------------------------------------------
```

```
                 ***********************
                 *                     *
                 *   APPLE-II PSEUDO   *
                 * MACHINE INTERPRETER *
                 *                     *
                 *   COPYRIGHT 1977    *
                 * APPLE COMPUTER INC  *
                 *                     *
                 * ALL RIGHTS RESERVED *
                 *     S. WOZNIAK      *
                 *                     *
                 ***********************
                                                ; TITLE "SWEET16 INTERPRETER"
                 R0L      EQU   $0
                 R0H      EQU   $1
                 R14H     EQU   $1D
                 R15L     EQU   $1E
                 R15H     EQU   $1F
                 SW16PAG  EQU   $F7
                 SAVE     EQU   $FF4A
                 RESTORE  EQU   $FF3F
                          ORG   $F689
F689: 20 4A FF   SW16     JSR   SAVE            ;PRESERVE 6502 REG CONTENTS
F68C: 68                  PLA
F68D: 85 1E               STA   R15L            ;INIT SWEET16 PC
F68F: 68                  PLA                   ;FROM RETURN
F690: 85 1F               STA   R15H            ;  ADDRESS
F692: 20 98 F6   SW16B    JSR   SW16C           ;INTERPRET AND EXECUTE
F695: 4C 92 F6            JMP   SW16B           ;ONE SWEET16 INSTR.
F698: E6 1E      SW16C    INC   R15L
F69A: D0 02               BNE   SW16D           ;INCR SWEET16 PC FOR FETCH
F69C: E6 1F               INC   R15H
F69E: A9 F7      SW16D    LDA   #SW16PAG
F6A0: 48                  PHA                   ;PUSH ON STACK FOR RTS
F6A1: A0 00               LDY   #$0
F6A3: B1 1E               LDA   (R15L),Y        ;FETCH INSTR
F6A5: 29 0F               AND   #$F             ;MASK REG SPECIFICATION
F6A7: 0A                  ASL                   ;DOUBLE FOR TWO BYTE REGISTERS
F6A8: AA                  TAX                   ;TO X REG FOR INDEXING
F6A9: 4A                  LSR
F6AA: 51 1E               EOR   (R15L),Y        ;NOW HAVE OPCODE
F6AC: F0 0B               BEQ   TOBR            ;IF ZERO THEN NON-REG OP
F6AE: 86 1D               STX   R14H            ;INDICATE'PRIOR RESULT REG'
F6B0: 4A                  LSR
F6B1: 4A                  LSR                   ;OPCODE*2 TO LSB'S
F6B2: 4A                  LSR
F6B3: A8                  TAY                   ;TO Y REG FOR INDEXING
F6B4: B9 E1 F6            LDA   OPTBL-2,Y       ;LOW ORDER ADR BYTE
F6B7: 48                  PHA                   ;ONTO STACK
F6B8: 60                  RTS                   ;GOTO REG-OP ROUTINE
F6B9: E6 1E      TOBR     INC   R15L
F6BB: D0 02               BNE   TOBR2           ;INCR PC
F6BD: E6 1F               INC   R15H
F6BF: BD E4 F6   TOBR2    LDA   BRTBL,X         ;LOW ORDER ADR BYTE
F6C2: 48                  PHA                   ;ONTO STACK FOR NON-REG OP
F6C3: A5 1D               LDA   R14H            ;'PRIOR RESULT REG' INDEX
```

```
F6C5: 4A                  LSR                 ;PREPARE CARRY FOR BC, BNC.
F6C6: 60                  RTS                 ;GOTO NON-REG OP ROUTINE
F6C7: 68         RTNZ     PLA                 ;POP RETURN ADDRESS
F6C8: 68                  PLA
F6C9: 20 3F FF            JSR   RESTORE       ;RESTORE 6502 REG CONTENTS
F6CC: 6C 1E 00            JMP   (R15L)        ;RETURN TO 6502 CODE VIA PC
F6CF: B1 1E      SETZ     LDA   (R15L),Y      ;HIGH-ORDER BYTE OF CONSTANT
F6D1: 95 01               STA   R0H,X
F6D3: 88                  DEY
F6D4: B1 1E               LDA   (R15L),Y      ;LOW-ORDER BYTE OF CONSTANT
F6D6: 95 00               STA   R0L,X
F6D8: 98                  TYA                 ;Y-REG CONTAINS 1
F6D9: 38                  SEC
F6DA: 65 1E               ADC   R15L          ;ADD 2 TO PC
F6DC: 85 1E               STA   R15L
F6DE: 90 02               BCC   SET2
F6E0: E6 1F               INC   R15H
F6E2: 60         SET2     RTS
F6E3: 02         OPTBL    DFB   SET-1         ;1X
F6E4: F9         BRTBL    DFB   RTN-1         ;0
F6E5: 04                  DFB   LD-1          ;2X
F6E6: 9D                  DFB   BR-1          ;1
F6E7: 0D                  DFB   ST-1          ;3X
F6E8: 9E                  DFB   BNC-1         ;2
F6E9: 25                  DFB   LDAT-1        ;4X
F6EA: AF                  DFB   BC-1          ;3
F6EB: 16                  DFB   STAT-1        ;5X
F6EC: B2                  DFB   BP-1          ;4
F6ED: 47                  DFB   LDDAT-1       ;6X
F6EE: B9                  DFB   BM-1          ;5
F6EF: 51                  DFB   STDAT-1       ;7X
F6F0: C0                  DFB   BZ-1          ;6
F6F1: 2F                  DFB   POP-1         ;8X
F6F2: C9                  DFB   BNZ-1         ;7
F6F3: 5B                  DFB   STPAT-1       ;9X
F6F4: D2                  DFB   BM1-1         ;8
F6F5: 85                  DFB   ADD-1         ;AX
F6F6: DD                  DFB   BNM1-1        ;9
F6F7: 6E                  DFB   SUB-1         ;BX
F6F8: 05                  DFB   BK-1          ;A
F6F9: 33                  DFB   POPD-1        ;CX
F6FA: E8                  DFB   RS-1          ;B
F6FB: 70                  DFB   CPR-1         ;DX
F6FC: 93                  DFB   BS-1          ;C
F6FD: 1E                  DFB   INR-1         ;EX
F6FE: E7                  DFB   NUL-1         ;D
F6FF: 65                  DFB   DCR-1         ;FX
F700: E7                  DFB   NUL-1         ;E
F701: E7                  DFB   NUL-1         ;UNUSED
F702: E7                  DFB   NUL-1         ;F
F703: 10 CA      SET      BPL   SETZ          ;ALWAYS TAKEN
F705: B5 00      LD       LDA   R0L,X
                 BK       EQU   *-1
F707: 85 00               STA   R0L
F709: B5 01               LDA   R0H,X         ;MOVE RX TO R0
F70B: 85 01               STA   R0H
F70D: 60                  RTS
F70E: A5 00      ST       LDA   R0L
F710: 95 00               STA   R0L,X         ;MOVE R0 TO RX
F712: A5 01               LDA   R0H
F714: 95 01               STA   R0H,X
F716: 60                  RTS
```

```
F717: A5 00     119  STAT      LDA   R0L
F719: 81 00     120  STAT2     STA   (R0L,X)     ;STORE BYTE INDIRECT
F71B: A0 00     121            LDY   #$0
F71D: 84 1D     122  STAT3     STY   R14H        ;INDICATE R0 IS RESULT NEG
F71F: F6 00     123  INR       INC   R0L,X
F721: D0 02     124            BNE   INR2        ;INCR RX
F723: F6 01     125            INC   R0H,X
F725: 60        126  INR2      RTS
F726: A1 00     127  LDAT      LDA   (R0L,X)     ;LOAD INDIRECT (RX)
F728: 85 00     128            STA   R0L         ;TO R0
F72A: A0 00     129            LDY   #$0
F72C: 84 01     130            STY   R0H         ;ZERO HIGH-ORDER R0 BYTE
F72E: F0 ED     131            BEQ   STAT3       ;ALWAYS TAKEN
F730: A0 00     132  POP       LDY   #$0         ;HIGH ORDER BYTE = 0
F732: F0 06     133            BEQ   POP2        ;ALWAYS TAKEN
F734: 20 66 F7  134  POPD      JSR   DCR         ;DECR RX
F737: A1 00     135            LDA   (R0L,X)     ;POP HIGH ORDER BYTE @RX
F739: A8        136            TAY               ;SAVE IN Y-REG
F73A: 20 66 F7  137  POP2      JSR   DCR         ;DECR RX
F73D: A1 00     138            LDA   (R0L,X)     ;LOW-ORDER BYTE
F73F: 85 00     139            STA   R0L         ;TO R0
F741: 84 01     140            STY   R0H
F743: A0 00     141  POP3      LDY   #$0         ;INDICATE R0 AS LAST RESULT REG
F745: 84 1D     142            STY   R14H
F747: 60        143            RTS
F748: 20 26 F7  144  LDDAT     JSR   LDAT        ;LOW-ORDER BYTE TO R0, INCR RX
F74B: A1 00     145            LDA   (R0L,X)     ;HIGH-ORDER BYTE TO R0
F74D: 85 01     146            STA   R0H
F74F: 4C 1F F7  147            JMP   INR         ;INCR RX
F752: 20 17 F7  148  STDAT     JSR   STAT        ;STORE INDIRECT LOW-ORDER
F755: A5 01     149            LDA   R0H         ;BYTE AND INCR RX.  THEN
F757: 81 00     150            STA   (R0L,X)     ;STORE HIGH-ORDER BYTE.
F759: 4C 1F F7  151            JMP   INR         ;INCR RX AND RETURN
F75C: 20 66 F7  152  STPAT     JSR   DCR         ;DECR RX
F75F: A5 00     153            LDA   R0L
F761: 81 00     154            STA   (R0L,X)     ;STORE R0 LOW BYTE @RX
F763: 4C 43 F7  155            JMP   POP3        ;INDICATE R0 AS LAST RSLT REG
F766: B5 00     156  DCR       LDA   R0L,X
F768: D0 02     157            BNE   DCR2        ;DECR RX
F76A: D6 01     158            DEC   R0H,X
F76C: D6 00     159  DCR2      DEC   R0L,X
F76E: 60        160            RTS
F76F: A0 00     161  SUB       LDY   #$0         ;RESULT TO R0
F771: 38        162  CPR       SEC               ;NOTE Y-REG = 13*2 FOR CPR
F772: A5 00     163            LDA   R0L
F774: F5 00     164            SBC   R0L,X
F776: 99 00 00  165            STA   R0L,Y       ;R0-RX TO RY
F779: A5 01     166            LDA   R0H
F77B: F5 01     167            SBC   R0H,X
F77D: 99 01 00  168  SUB2      STA   R0H,Y
F780: 98        169            TYA               ;LAST RESULT REG*2
F781: 69 00     170            ADC   #$0         ;CARRY TO LSB
F783: 85 1D     171            STA   R14H
F785: 60        172            RTS
F786: A5 00     173  ADD       LDA   R0L
F788: 75 00     174            ADC   R0L,X
F78A: 85 00     175            STA   R0L         ;R0+RX TO R0
F78C: A5 01     176            LDA   R0H
F78E: 75 01     177            ADC   R0H,X
F790: A0 00     178            LDY   #$0         ;R0 FOR RESULT
F792: F0 E9     179            BEQ   SUB2        ;FINISH ADD
F794: A5 1E     180  BS        LDA   R15L        ;NOTE X-REG IS 12*2!
```

```
F796: 20 19 F7           JSR   STAT2      ;PUSH LOW PC BYTE VIA R12
F799: A5 1F              LDA   R15H
F79B: 20 19 F7           JSR   STAT2      ;PUSH HIGH-ORDER PC BYTE
F79E: 18           BR    CLC
F79F: B0 0E        BNC   BCS   BNC2       ;NO CARRY TEST
F7A1: B1 1E        BR1   LDA   (R15L),Y   ;DISPLACEMENT BYTE
F7A3: 10 01              BPL   BR2
F7A5: 88                 DEY
F7A6: 65 1E        BR2   ADC   R15L       ;ADD TO PC
F7A8: 85 1E              STA   R15L
F7AA: 98                 TYA
F7AB: 65 1F              ADC   R15H
F7AD: 85 1F              STA   R15H
F7AF: 60           BNC2  RTS
F7B0: B0 EC        BC    BCS   BR
F7B2: 60                 RTS
F7B3: 0A           BP    ASL              ;DOUBLE RESULT-REG INDEX
F7B4: AA                 TAX              ;TO X REG FOR INDEXING
F7B5: B5 01              LDA   R0H,X      ;TEST FOR PLUS
F7B7: 10 E8              BPL   BR1        ;BRANCH IF SO
F7B9: 60                 RTS
F7BA: 0A           BM    ASL              ;DOUBLE RESULT-REG INDEX
F7BB: AA                 TAX
F7BC: B5 01              LDA   R0H,X      ;TEST FOR MINUS
F7BE: 30 E1              BMI   BR1
F7C0: 60                 RTS
F7C1: 0A           BZ    ASL              ;DOUBLE RESULT-REG INDEX
F7C2: AA                 TAX
F7C3: B5 00              LDA   R0L,X      ;TEST FOR ZERO
F7C5: 15 01              ORA   R0H,X      ;(BOTH BYTES)
F7C7: F0 D8              BEQ   BR1        ;BRANCH IF SO
F7C9: 60                 RTS
F7CA: 0A           BNZ   ASL              ;DOUBLE RESULT-REG INDEX
F7CB: AA                 TAX
F7CC: B5 00              LDA   R0L,X      ;TEST FOR NON-ZERO
F7CE: 15 01              ORA   R0H,X      ;(BOTH BYTES)
F7D0: D0 CF              BNE   BR1        ;BRANCH IF SO
F7D2: 60                 RTS
F7D3: 0A           BM1   ASL              ;DOUBLE RESULT-REG INDEX
F7D4: AA                 TAX
F7D5: B5 00              LDA   R0L,X      ;CHECK BOTH BYTES
F7D7: 35 01              AND   R0H,X      ;FOR $FF (MINUS 1)
F7D9: 49 FF              EOR   #$FF
F7DB: F0 C4              BEQ   BR1        ;BRANCH IF SO
F7DD: 60                 RTS
F7DE: 0A           BNM1  ASL              ;DOUBLE RESULT-REG INDEX
F7DF: AA                 TAX
F7E0: B5 00              LDA   R0L,X
F7E2: 35 01              AND   R0H,X      ;CHECK BOTH BYTES FOR NO $FF
F7E4: 49 FF              EOR   #$FF
F7E6: D0 B9              BNE   BR1        ;BRANCH IF NOT MINUS 1
F7E8: 60           NUL   RTS
F7E9: A2 18        RS    LDX   #$18       ;12*2 FOR R12 AS STACK POINTER
F7EB: 20 66 F7           JSR   DCR        ;DECR STACK POINTER
F7EE: A1 00              LDA   (R0L,X)    ;POP HIGH RETURN ADDRESS TO PC
F7F0: 85 1F              STA   R15H
F7F2: 20 66 F7           JSR   DCR        ;SAME FOR LOW-ORDER BYTE
F7F5: A1 00              LDA   (R0L,X)
F7F7: 85 1E              STA   R15L
F7F9: 60                 RTS
F7FA: 4C C7 F6     RTN   JMP   RTNZ
```

```
+----------------------------------------------------------------------------
|  TOPIC -- Apple II -- WOZPAK Sweet-16 article by Steve Wozniak
+----------------------------------------------------------------------------
```

SWEET 16: A Pseudo 16 Bit Microprocessor

by Steve Wozniak

Description:

While writing APPLE BASIC for a 6502 microprocessor, I repeatedly encountered a variant of MURPHY'S LAW. Briefly stated, any routine operating on 16-bit data will require at least twice the code that it should. Programs making extensive use of 16-bit pointers (such as compilers, editors, and assemblers) are included in this category. In my case, even the addition of a few double-byte instructions to the 6502 would have only slightly alleviated the problem. What I really needed was a 6502/RCA 1800 hybrid - an abundance of 16-bit registers and excellent pointer capability. My solution was to implement a non-existant (meta) 16-bit processor in software, interpreter style, which I call SWEET 16.

SWEET 16 is based on sixteen 16-bit registers (R0-15), which are actually 32 memory locations. R0 doubles as the SWEET 16 accumulator (ACC), R15 as the program counter (PC), and R14 as the status register. R13 holds compare instruction results and R12 is the subroutine return stack pointer if SWEET 16 subroutines are used. All other SWEET 16 registers are at the user's unrestricted disposal.

SWEET 16 instructions fall into register and non-register categories. The register ops specify one of the sixteen registers to be used as either a data element or a pointer to data in memory, depending on the specific instruction. For example INR R5 uses R5 as data and ST @R7 uses R7 as a pointer to data in memory. Except for the SET instruction, register ops take one byte of code each. The non-register ops are primarily 6502 style branches with the second byte specifying a +/-127 byte displacement relative to the address of the following instruction. Providing that the prior register op result meets a specified branch condition, the displacement is added to the SWEET 16 PC, effecting a branch.

SWEET 16 is intended as a 6502 enhancement package, not a stand alone processor. A 6502 program switches to SWEET 16 mode with a subroutine call and subsequent code is interpreted as SWEET 16 instructions. The nonregister op RTN returns the user program to 6502 mode after restoring the internal register contents (A, X, Y, P, and S). The following example illustrates how to use SWEET 16.

```
300  B9 00 02           LDA   IN,Y     ;get a char
303  C9 CD              CMP   #"M"     ;"M" for move
305  D0 09              BNE   NOMOVE   ;No. Skip move
307  20 89 F6           JSR   SW16     ;Yes, call SWEET 16
```

```
30A  41         MLOOP   LD    @R1      ;R1 holds source
30B  52                 ST    @R2      ;R2 holds dest. addr.
30C  F3                 DCR   R3       ;Decr. length
30D  07 FB              BNZ   MLOOP    ;Loop until done
30F  00                 RTN            ;Return to 6502 mode.
310  C9 C5      NOMOVE  CMP   #"E"     ;"E" char?
312  D0 13              BEQ   EXIT     ;Yes, exit
314  C8                 INY            ;No, cont.
```

NOTE: Registers A, X, Y, P, and S are not disturbed by SWEET 16.

Instruction Descriptions:

The SWEET 16 opcode listing is short and uncomplicated. Excepting relative branch displacements, hand assembly is trivial. All register opcodes are formed by combining two Hex digits, one for the opcode and one to specify a register. For example, opcodes 15 and 45 both specify register R5 while codes 23, 27, and 29 are all ST ops. Most register ops are assigned in complementary pairs to facilitate remembering them. Therefore, LD ans ST are opcodes 2N and 3N respectively, while LD @ and ST @ are codes 4N and 5N.

Opcodes 0 to C (Hex) are assigned to the thirteen non-register ops. Except for RTN (opcode 0), BK (0A), and RS (0B), the non register ops are 6502 style branches. The second byte of a branch instruction contains a +/-127 byte displacement value (in two's complement form) relative to the address of the instruction immediately following the branch.

If a specified branch condition is met by the prior register op result, the displacement is added to the PC effecting a branch. Except for the BR (Branch always) and BS (Branch to a Subroutine), the branch opcodes are assigned in complementary pairs, rendering them easily remembered for hand coding. For example, Branch if Plus and Branch if Minus are opcodes 4 and 5 while Branch if Zero and Branch if NonZero are opcodes 6 and 7.

SWEET 16 Opcode Summary:

Register OPS-

```
     1n        SET       Rn     Constant  (Set)
     2n        LD        Rn     (Load)
     3n        ST        Rn     (Store)
     4n        LD        @Rn    (Load Indirect)
     5n        ST        @Rn    (Store Indirect)
     6n        LDD       @Rn    (Load Double Indirect)
     7n        STD       @Rn    (Store Double Indirect)
     8n        POP       @Rn    (Pop Indirect)
     9n        STP       @Rn    (Store POP Indirect)
     An        ADD       Rn     (Add)
     Bn        SUB       Rn     (Sub)
     Cn        POPD      @Rn    (Pop Double Indirect)
     Dn        CPR       Rn     (Compare)
```

```
     En         INR       Rn     (Increment)
     Fn         DCR       Rn     (Decrement)

Non-register OPS-

     00         RTN              (Return to 6502 mode)
     01         BR        ea     (Branch always)
     02         BNC       ea     (Branch if No Carry)
     03         BC        ea     (Branch if Carry)
     04         BP        ea     (Branch if Plus)
     05         BM        ea     (Branch if Minus)
     06         BZ        ea     (Branch if Zero)
     07         BNZ       ea     (Branch if NonZero)
     08         BM1       ea     (Branch if Minus 1)
     09         BNM1      ea     (Branch if Not Minus 1)
     0A         BK               (Break)
     0B         RS               (Return from Subroutine)
     0C         BS        ea     (Branch to Subroutine)
     0D                          (Unassigned)
     0E                          (Unassigned)
     0F                          (Unassigned)

Register Instructions:
----------------------

SET:

     SET Rn,Constant     [ 1n Low High ]

     The 2-byte constant is loaded into Rn (n=0 to F, Hex) and
     branch conditions set accordingly. The carry is cleared.

     EXAMPLE:

     15 34 A0   SET  R5   $A034     ;R5 now contains $A034

LOAD:

     LD Rn               [ 2n ]

     The ACC (R0) is loaded from Rn and branch conditions set
     according to the data transferred. The carry is cleared and
     contents of Rn are not disturbed.

     EXAMPLE:

     15 34 A0   SET  R5   $A034
     25         LD   R5             ;ACC now contains $A034

STORE:

     ST Rn               [ 3n ]

     The ACC is stored into Rn and branch conditions set according
     to the data transferred. The carry is cleared and the ACC
     contents are not disturbed.
```

EXAMPLE:

```
25          LD   R5              ;Copy the contents
36          ST   R6              ;of R5 to R6
```

LOAD INDIRECT:

```
LD @Rn              [ 4n ]
```

The low-order ACC byte is loaded from the memory location whose address resides in Rn and the high-order ACC byte is cleared. Branch conditions reflect the final ACC contents which will always be positive and never minus 1. The carry is cleared. After the transfer, Rn is incremented by 1.

EXAMPLE

```
15 34 A0    SET  R5   $A034
45          LD   @R5            ;ACC is loaded from memory
                                ;location $A034
                                ;R5 is incr to $A035
```

STORE INDIRECT:

```
ST @Rn              [ 5n ]
```

The low-order ACC byte is stored into the memory location whose address resides in Rn. Branch conditions reflect the 2-byte ACC contents. The carry is cleared. After the transfer Rn is incremented by 1.

EXAMPLE:

```
15 34 A0    SET  R5   $A034     ;Load pointers R5, R6 with
16 22 90    SET  R6   $9022     ;$A034 and $9022
45          LD   @R5            ;Move byte from $A034 to $9022
56          ST   @R6            ;Both ptrs are incremented
```

LOAD DOUBLE-BYTE INDIRECT:

```
LDD @Rn             [ 6n ]
```

The low order ACC byte is loaded from memory location whose address resides in Rn, and Rn is then incremented by 1. The high order ACC byte is loaded from the memory location whose address resides in the incremented Rn, and Rn is again incremented by 1. Branch conditions reflect the final ACC contents. The carry is cleared.

EXAMPLE:

```
15 34 A0    SET  R5   $A034     ;The low-order ACC byte is loaded
65          LDD  @R6            ;from $A034, high-order from
                                ;$A035, R5 is incr to $A036
```

STORE DOUBLE-BYTE INDIRECT:

```
STD @Rn            [ 7n ]
```

The low-order ACC byte is stored into memory location whose address resides in Rn, and Rn is the incremented by 1. The high-order ACC byte is stored into the memory location whose address resides in the incremented Rn, and Rn is again incremented by 1. Branch conditions reflect the ACC contents which are not disturbed. The carry is cleared.

EXAMPLE:

```
15 34 A0   SET  R5   $A034     ;Load pointers R5, R6
16 22 90   SET  R6   $9022     ;with $A034 and $9022
65         LDD  @R5            ;Move double byte from
76         STD  @R6            ;$A034-35 to $9022-23.
                               ;Both pointers incremented by 2.
```

POP INDIRECT:

```
POP @Rn             [ 8n ]
```

The low-order ACC byte is loaded from the memory location whose address resides in Rn after Rn is decremented by 1, and the high order ACC byte is cleared. Branch conditions reflect the final 2-byte ACC contents which will always be positive and never minus one. The carry is cleared. Because Rn is decremented prior to loading the ACC, single byte stacks may be implemented with the ST @Rn and POP @Rn ops (Rn is the stack pointer).

EXAMPLE:

```
15 34 A0   SET  R5   $A034     ;Init stack pointer
10 04 00   SET  R0   4         ;Load 4 into ACC
55         ST   @R5            ;Push 4 onto stack
10 05 00   SET  R0   5         ;Load 5 into ACC
55         ST   @R5            ;Push 5 onto stack
10 06 00   SET  R0   6         ;Load 6 into ACC
55         ST   @R5            ;Push 6 onto stack
85         POP  @R5            ;Pop 6 off stack into ACC
85         POP  @R5            ;Pop 5 off stack
85         POP  @R5            ;Pop 4 off stack
```

STORE POP INDIRECT:

```
STP @Rn             [ 9n ]
```

The low-order ACC byte is stored into the memory location whose address resides in Rn after Rn is decremented by 1. Branch conditions will reflect the 2-byte ACC contents which are not modified. STP @Rn and POP @Rn are used together to move data blocks beginning at the greatest address and working down. Additionally, single-byte stacks may be implemented with the STP @Rn ops.

EXAMPLE:

```
14 34 A0   SET  R4   $A034     ;Init pointers
15 22 90   SET  R5   $9022
84         POP  @R4            ;Move byte from
95         STP  @R5            ;$A033 to $9021
84         POP  @R4            ;Move byte from
95         STP  @R5            ;$A032 to $9020
```

ADD:

ADD Rn [An]

The contents of Rn are added to the contents of ACC (R0), and the low-order 16 bits of the sum restored in ACC. the 17th sum bit becomes the carry and the other branch conditions reflect the final ACC contents.

EXAMPLE:

```
10 34 76   SET  R0   $7634     ;Init R0 (ACC) and R1
11 27 42   SET  R1   $4227
A1         ADD  R1             ;Add R1 (sum=B85B, C clear)
A0         ADD  R0             ;Double ACC (R0) to $70B6
                               ;with carry set.
```

SUBTRACT:

SUB Rn [Bn]

The contents of Rn are subtracted from the ACC contents by performing a two's complement addition:

ACC = ACC + Rn + 1

The low order 16 bits of the subtraction are restored in the ACC, the 17th sum bit becomes the carry and other branch conditions reflect the final ACC contents. If the 16-bit unsigned ACC contents are greater than or equal to the 16-bit unsigned Rn contents, then the carry is set, otherwise it is cleared. Rn is not disturbed.

EXAMPLE:

```
10 34 76   SET  R0   $7634     ;Init R0 (ACC)
11 27 42   SET  R1   $4227     ;and R1
B1         SUB  R1             ;subtract R1
                               ;(diff=$340D with c set)
B0         SUB  R0             ;clears ACC. (R0)
```

POP DOUBLE-BYTE INDIRECT:

POPD @Rn [Cn]

Rn is decremented by 1 and the high-order ACC byte is loaded from the memory location whose address now resides in Rn. Rn is again decremented by 1 and the low-order ACC byte is loaded from the corresponding memory location. Branch conditions reflect the final ACC contents. The carry is cleared. Because Rn is decremented prior to loading each of the ACC halves, double-byte stacks may be implemented with the STD @Rn and POPD @Rn ops (Rn is the stack pointer).

EXAMPLE:

```
15 34 A0   SET  R5   $A034     ;Init stack pointer
10 12 AA   SET  R0   $AA12     ;Load $AA12 into ACC
75         STD  @R5            ;Push $AA12 onto stack
10 34 BB   SET  R0   $BB34     ;Load $BB34 into ACC
75         STD  @R5            ;Push $BB34 onto stack
C5         POPD @R5            ;Pop $BB34 off stack
C5         POPD @R5            ;Pop $AA12 off stack
```

COMPARE:

CPR Rn [Dn]

The ACC (R0) contents are compared to Rn by performing the 16 bit binary subtraction ACC-Rn and storing the low order 16 difference bits in R13 for subsequent branch tests. If the 16 bit unsigned ACC contents are greater than or equal to the 16 bit unsigned Rn contents, then the carry is set, otherwise it is cleared. No other registers, including ACC and Rn, are disturbed.

EXAMPLE:

```
15 34 A0           SET  R5   $A034     ;Pointer to memory
16 BF A0           SET  R6   $A0BF     ;Limit address
B0         LOOP1   SUB  R0             ;Zero data
75                 STD  @R5            ;clear 2 locations
                                       ;increment R5 by 2
25                 LD   R5             ;Compare pointer R5
D6                 CPR  R6             ;to limit R6
02 FA              BNC  LOOP1          ;loop if C clear
```

INCREMENT:

INR Rn [En]

The contents of Rn are incremented by 1. The carry is cleared and other branch conditions reflect the incremented value.

EXAMPLE:

```
15 34 A0   SET  R5   $A034     ;(Pointer)
B0         SUB  R0             ;Zero to R0
55         ST   @R5            ;Clr Location $A034
E5         INR  R5             ;Incr R5 to $A036
55         ST   @R5            ;Clrs location $A036
                               ;(not $A035)
```

```
DECREMENT:

     DCR Rn              [ Fn ]

     The contents of Rn are decremented by 1. The carry is cleared
     and other branch conditions reflect the decremented value.

     EXAMPLE:  (Clear 9 bytes beginning at location A034)

     15 34 A0            SET  R5   $A034     ;Init pointer
     14 09 00            SET  R4   9         ;Init counter
     B0                  SUB  R0             ;Zero ACC
     55          LOOP2   ST   @R5            ;Clear a mem byte
     F4                  DCR  R4             ;Decrement count
     07 FC               BNZ  LOOP2          ;Loop until Zero

Non-Register Instructions:
--------------------------

RETURN TO 6502 MODE:

     RTN 00

     Control is returned to the 6502 and program execution continues
     at the location immediately following the RTN instruction. the
     6502 registers and status conditions are restored to their
     original contents (prior to entering SWEET 16 mode).

BRANCH ALWAYS:

     BR ea               [ 01 d ]

     An effective address (ea) is calculated by adding the signed
     displacement byte (d) to the PC. The PC contains the address
     of the instruction immediately following the BR, or the address
     of the BR op plus 2. The displacement is a signed two's
     complement value from -128 to +127. Branch conditions are not
     changed.

     NOTE: The effective address calculation is identical to that
     for 6502 relative branches. The Hex add & Subtract features of
     the APPLE ][ monitor may be used to calculate displacements.

     d = $80  ea = PC + 2 - 128
     d = $81  ea = PC + 2 - 127

     d = $FF  ea = PC + 2 - 1
     d = $00  ea = PC + 2 + 0
     d = $01  ea = PC + 2 + 1

     d = $7E  ea = PC + 2 + 126
     d = $7F  ea = PC + 2 + 127

     EXAMPLE:
```

```
$300:  01 50  BR $352
```

BRANCH IF NO CARRY:

```
BNC ea              [ 02 d ]
```

A branch to the effective address is taken only is the carry is clear, otherwise execution resumes as normal with the next instruction. Branch conditions are not changed.

BRANCH IF CARRY SET:

```
BC ea               [ 03 d ]
```

A branch is effected only if the carry is set. Branch conditions are not changed.

BRANCH IF PLUS:

```
BP ea               [ 04 d ]
```

A branch is effected only if the prior 'result' (or most recently transferred dat) was positive. Branch conditions are not changed.

EXAMPLE: (Clear mem from A034 to A03F)

```
15 34 A0           SET  R5   $A034     ;Init pointer
14 3F A0           SET  R4   $A03F     ;Init limit
B0        LOOP3    SUB  R0
55                 ST   @R5            ;Clear mem byte
                                       ;Increment R5
24                 LD   R4             ;Compare limit
D5                 CPR  R5             ;to pointer
04 FA              BP   LOOP3          ;Loop until done
```

BRANCH IF MINUS:

```
BM ea               [ 05 d ]
```

A branch is effected only if prior 'result' was minus (negative, MSB = 1). Branch conditions are not changed.

BRANCH IF ZERO:

```
BZ ea               [ 06 d ]
```

A Branch is effected only if the prior 'result' was zero. Branch conditions are not changed.

BRANCH IF NONZERO

```
BNZ ea              [ 07 d ]
```

A branch is effected only if the priot 'result' was non-zero Branch conditions are not changed.

BRANCH IF MINUS ONE

```
BM1 ea              [ 08 d ]
```

A branch is effected only if the prior 'result' was minus one ($FFFF Hex). Branch conditions are not changed.

BRANCH IF NOT MINUS ONE

```
BNM1 ea             [ 09 d ]
```

A branch effected only if the prior 'result' was not minus 1. Branch conditions are not changed.

BREAK:

```
BK                  [ 0A ]
```

A 6502 BRK (break) instruction is executed. SWEET 16 may be re-entered non destructively at SW16d after correcting the stack pointer to its value prior to executing the BRK.

RETURN FROM SWEET 16 SUBROUTINE:

```
RS                  [ 0B ]
```

RS terminates execution of a SWEET 16 subroutine and returns to the SWEET 16 calling program which resumes execution (in SWEET 16 mode). R12, which is the SWEET 16 subroutine return stack pointer, is decremented twice. Branch conditions are not changed.

BRANCH TO SWEET 16 SUBROUTINE:

```
BS ea               [ 0c d ]
```

A branch to the effective address (PC + 2 + d) is taken and execution is resumed in SWEET 16 mode. The current PC is pushed onto a SWEET 16 subroutine return address stack whose pointer is R12, and R12 is incremented by 2. The carry is cleared and branch conditions set to indicate the current ACC contents.

EXAMPLE: (Calling a 'memory move' subroutine to move A034-A03B
         to 3000-3007)

```
15 34 A0          SET  R5   $A034     ;Init pointer 1
14 3B A0          SET  R4   $A03B     ;Init limit 1
16 00 30          SET  R6   $3000     ;Init pointer 2
0C 15             BS   MOVE           ;Call move subroutine
45         MOVE   LD   @R5            ;Move one
56                ST   @R6            ;byte
24                LD   R4
D5                CPR  R5             ;Test if done
04 FA             BP   MOVE
0B                RS                  ;Return
```

Theory of Operation:

SWEET 16 execution mode begins with a subroutine call to SW16. All 6502 registers are saved at this time, to be restored when a SWEET 16 RTN instruction returns control to the 6502. If you can tolerate indefinate 6502 register contents upon exit, approximately 30 usec may be saved by entering at SW16 + 3. Because this might cause an inadvertant switch from Hex to Decimal mode, it is advisable to enter at SW16 the first time through.

After saving the 6502 registers, SWEET 16 initializes its PC (R15) with the subroutine return address off the 6502 stack. SWEET 16's PC points to the location preceding the next instruction to be executed. Following the subroutine call are 1-,2-, and 3-byte SWEET 16 instructions, stored in ascending memory like 6502 instructions. the main loop at SW16B repeatedly calls the 'execute instruction' routine to execute it.

Subroutine SW16C increments the PC (R15) and fetches the next opcode, which is either a register op of the form OP REG with OP between 1 and 15 or a non-register op of the form 0 OP with OP between 0 and 13. Assuming a register op, the register specification is doubled to account for the 3 byte SWEET 16 registers and placed in the X-reg for indexing. Then the instruction type is determined. Register ops place the doubled register specification in the high order byte of R14 indicating the 'prior result register' to subsequent branch instructions. Non-register ops treat the register specifcation (right-hand half-byte) as their opcode, increment the SWEET 16 PC to point at the displacement byte of branch instructions, load the A-reg with the 'prior result register' index for branch condition testing, and clear the Y-reg.

When is an RTS really a JSR?

Each instruction type has a corresponding subroutine. The subroutine entry points are stored in a table which is directly indexed into by the opcode. By assigning all the entries to a common page, only a single byte to address need be stored per routine. The 6502 indirect jump might have been used as follows to transfer control to the appropriate subroutine.

```
     LDA     #ADRH       ;High-order byte.
     STA     IND+1
     LDA     OPTBL,X     ;Low-order byte.
     STA     IND
     JMP     (IND)
```

To save code, the subroutine entry address (minus 1) is pushed onto the stack, high-order byte first. A 6502 RTS (return from subroutine) is used to pop the address off the stack and into the 6502 PC (after incrementing by 1). The net result is that the desired subroutine is reached by executing a subroutine return instruction!

Opcode Subroutines:

The register op routines make use of the 6502 'zero page indexed by X' and 'indexed by X direct' addressing modes to access the specified registers and indirect data. The 'result' of most register ops is left in the specified register and can be sensed by subsequent branch instructions, since the register specification is saved in the high-order byte of R14. This specification is changed to indicate R0 (ACC) for ADD and SUB instructions and R13 for the CPR (compare) instruction.

Normally the high-order R14 byte holds the 'prior result register' index times 2 to account for the 2-byte SWEET 16 registers and the LSB is zero. If ADD, SUB, or CPR instructions generate carries, then this index is incremented, setting the LSB.

The SET instruction increments the PC twice, picking up data bytes in the specified register. In accordance with 6502 convention, the low-order data byte precedes the high-order byte.

Most SWEET 16 non-register ops are relative branches. The corresponding subroutines determine whether or not the 'prior result' meets the specified branch condition and if so, update the SWEET 16 PC by adding the displacement value (-128 to +127 bytes).

The RTN op restores the 6502 register contents, pops the subroutine return stack and jumps indirect through the SWEET 16 PC. This transfers control to the 6502 at the instruction immediately following the RTN instruction.

The BK op actually executes a 6502 break instruction (BRK), transferring control to the interrupt handler.

Any number of subroutine levels may be implemented within SWEET 16 code via the BS (Branch to Subroutine) and RS (Return from Subroutine) instructions. The user must initialize and otherwise not disturb R12 if the SWEET 16 subroutine capability is used since it is utilized as the automatic return stack pointer.

Memory Allocation:

The only storage that must be allocated for SWEET 16 variables are 32 consecutive locations in page zero for the SWEET 16 registers, four locations to save the 6502 register contents, and a few levels of the 6502 subroutine return address stack. if you don't need to preserve the 6502 register contents, delete the SAVE and RESTORE subroutines and the corresponding subroutine calls. This will free the four page zero locations ASAV, XSAV, YSAV, and PSAV.

User Modifications:

You may wish to add some of your own instructions to this implementation of SWEET 16. If you use the unassigned opcodes $0E and $0F, remember that SWEET 16 treats these as 2-byte instructions. You may wish to handle the break instruction as a SWEET 16 call, saving two bytes of code each time you transfer into SWEET 16 mode. Or you may wish to use the SWEET 16 BK (break) op as a 'CHAROUT' call in the interrupt handler. You can perform absolute jumps within SWEET 16 by loading the ACC (R0) with the address you wish to jump to (minus 1) and executing a ST R15 instruction.

```
+----------------------------------------------------------------------------
|  TOPIC -- Apple II -- WOZPAK Sweet-16 article by Dick Sedgewick
+----------------------------------------------------------------------------
```

SWEET 16 - INTRODUCTION

by Dick Sedgewick

Sweet 16 is probably the least used and least understood seed in the Apple][.

In exactly the same sense that Integer and Applesoft Basics are languages, SWEET 16 is a language. Compared to the Basics, however, it would be classed as low level with a strong likeness to conventional 6502 Assembly language.

To use SWEET 16, you must learn the language - and to quote "WOZ", "The opcode list is short and uncomplicated". "WOZ" (Steve Wozniak), of course is Mr. Apple, and the creator of SWEET 16.

SWEET 16 is ROM based in every Apple][from $F689 to $F7FC. It has it's own set of opcodes and instruction sets, and uses the SAVE and RESTORE routines from the Apple Monitor to preserve the 6502 registers when in use, allowing SWEET 16 to be used as a subroutine.

It uses the first 32 locations on zero page to set up its 16 double byte registers, and is therefore not compatible with Applesoft Basic without some additional efforts.

The original article, "SWEET 16: The 6502 Dream Machine", first appeared in Byte Magazine, November 1977 and later in the original "WOZ PAK". The article is included here and again as test material to help understand the use and implementation of SWEET 16.

Examples of the use of SWEET 16 are found in the Programmer's Aid #1, in the Renumber, Append, and Relocate programs. The Programmer's Aid Operating Manual contains complete source assembly listings, indexed on page 65.

The demonstration program is written to be introductory and simple, consisting of three parts:

1. Integer Basic Program
2. Machine Language Subroutine
3. SWEET 16 Subroutine

The task of the program will be to move data. Parameters of the move will be entered in the Integer Basic Program.

The "CALL 768" ($300) at line 120, enters a 6502 machine language subroutine having the single purpose of entering SWEET 16 and subsequently returning to BASIC (addresses $300,

$301, $302, and $312 respectively). The SWEET 16 subroutine of course performs the move, and is entered at Hex locations $303 to $311 (see listing Number 3).

After the move, the screen will display three lines of data, each 8 bytes long, and await entry of a new set of parameters. The three lines of data displayed on the screen are as follows:

Line 1: The first 8 bytes of data starting at $800, which is the fixed source data to be moved (in this case, the string A$).

Line 2: The first 8 bytes of data starting at the hex address entered as the destination of the move (high order byte only).

Line 3: The first 8 bytes of data starting at $0000 (the first four SWEET 16 registers).

The display of 8 bytes of data was chosen to simplify the illustration of what goes on.

Integer Basic has its own way of recording the string A$. Because the name chosen for the string "A$" is stored in 2 bytes, a total of five housekeeping bytes precede the data entered as A$, leaving only three additional bytes available for display. Integer Basic also adds a housekeeping byte at the end of a string, known as the "string terminator".

Consequently, for convenience purposes of the display, and to see the string terminator as the 8th byte, the string data entered via the keyboard should be limited to two characters, and will appear as the 6th and 7th bytes. Additionally, parameters to be entered include the number of bytes to be moved. A useful range for this demonstration would be 1-8 inclusive, but of course 1-255 will work.

Finally, the starting address of the destination of the move must be entered. Again, for simplicity, only the high-order byte is entered, and the program allows a choice between Decimal 9 and high-order byte of program pointer 1, to avoid unnecessary problems (in this demonstration enter a decimal number between 9 and 144 for a 48K APPLE).

The 8 bytes of data displayed starting at $00 will enable one to observe the condition of the SWEET 16 registers after a move has been accomplished, and thereby understand how the SWEET 16 program works.

From the article "SWEET 16: A 6502 Dream Machine", remember that SWEET 16 can establish 16 double byte registers starting at $00. This means that SWEET 16 can use the first 32 addresses on zero page.

The "events" occurring in this demonstration program can be studied in the first four SWEET 16 registers. Therefore, the 8 byte display starting at $0000 is large enough for this purpose.

These four registers are established as R0, R1, R2, R3:

```
R0        $0000     &   0001       -SWEET 16 accumulator
R1        $0002     &   0003       -Source address
R2        $0004     &   0005       -Destination address
R3        $0006     &   0007       -Number of bytes to move
 .
 .
 .
R14       $001C     &   001D       -Prior result register
R15       $001E     &   001F       -SWEET 16 Program counter
```

Additionally, an examination of registers R14 and R15 will extend and understanding of SWEET 16, as fully explained in the "WOZ" text. Notice that the high order byte of R14, (located at $1D) contains $06, and is the doubled register specification (3X2=$06). R15, the SWEET 16 program counter contains the address of the next operation as it did for each step during execution of the program, which was $0312 when execution ended and the 6502 code resumed.

To try a sample run, enter the Integer Basic program as shown in Listing #1. Of course, REM statements can be omitted, and line 10 is only helpful if the machine code is to be stored on disk. Listing #2 must also be entered starting at $300.

NOTE: A 6502 disassembly does not look like listing #3, but the SOURCEROR disassembler would create a correct disassembly.

```
     Enter "RUN" and hit RETURN
     Enter "12" and hit RETURN (A$ - A$ string data)
     Enter "18" and hit RETURN (high-order byte of destination)
```

The display should appear as follows:

```
     $0800-C1 40 00 10 08 B1 B2 1E  (SOURCE)
     $0A00-C1 40 00 10 08 B1 B2 1E  (Dest.)
     $0000-1E 00 08 08 08 0A 00 00  (SWEET 16)
```

NOTE: The 8 bytes stored at $0A00 are identical to the 8 bytes starting at $0800, indicating that an accurate move of 8 bytes length has been made. They are moved one byte at a time starting with token C1 and ending with token 1E. If moving less than 8 bytes, the data following the moved data would be whatever existed at those locations before the move.

The bytes have the following significance:

A Token$

```
     C1     40   00    10    08     B1     B2        1E
     ---------  ----   --------     ---------        --
```

```
       |       |        |          |     |      String
      VN      DSP      NVA        DATA   DATA   Terminator
```

The SWEET 16 registers are as shown:

```
     low   high      low   high      low   high      low   high
$0000 1E    00        08    08        08    0A        00    00
     ----------      ----------      ----------      ----------
         |               |               |               |
      register        register        register        register
         R0              R1              R2              R3
       (acc)          (source)         (dest)         (#bytes)
```

The low order byte of R0, the SWEET 16 accumulator, has $1E in it, the last byte moved (the 8th).

The low order byte of the source register R1 started as $00 and was incremented eight times, once for each byte of moved data.

The high order byte of the destination register R2 contains $0A, which was entered at 10 (the variable) and poked into the SWEET 16 code. The low-order byte of R2 was incremented exactly like R1.

Finally, register R3, the register that stores the number of bytes to be moved, has been poked to 8 (the variable B) and decremented eight times as each byte got moved, ending up $0000.

By entering character strings and varying the number of bytes to be moved, the SWEET 16 registers can be observed and the contents predicted.

Working with this demonstration program, and study of the text material will enable you to write SWEET 16 programs that perform additional 16 bit manipulations. The unassigned opcodes mentioned in the "WOZ Dream Machine" article should present a most interesting opportunity to "play".

SWEET 16 as a language - or tool - opens a new direction to Apple][owners without spending a dime, and it's been there all the time.

"Apple-ites" who desire to learn machine language programming, can use SWEET 16 as a starting point. With this text material to use, and less opcodes to learn, a user can quickly be effective.

Listing #1

```
>List
     10     PRINT "[D]BLOAD SWEET":  REM CTRL D
     20     CALL - 936: DIM A $ (10)
     30     INPUT "ENTER STRING A $ " , A $
```

```
40      INPUT "ENTER # BYTES " , B
50      IF NOT B THEN 40 : REM AT LEAST 1
60      POKE 778 , B : REM POKE LENGTH
70      INPUT "ENTER DESTINATION " , A
80      IF A > PEEK (203) - 1 THEN 70
90      IF A < PEEK (205) + 1 THEN 70
100     POKE 776 , A : REM POKE DESTINATION
110     M = 8 : GOSUB 160 : REM DISPLAY
120     CALL 768 : REM GOTO $0300
130     M = A : GOSUB 160 : REM DISPLAY
140     M = O : GOSUB 160 : REM DISPLAY
150     PRINT : PRINT : GOTO 30
160     POKE 60 , 0 : POKE 61 , M
170     CALL -605 : RETURN : REM XAM8 IN MONITOR
```

Listing #2

```
300:20 89 F6 11 00 08 12 00 00 13 00 00 41 52
    F3 07 FB 00 60
```

Listing #3

SWEET 16

```
$300  20  89  F6   JSR    $F689
$303  11  00  08   SET    R1  source address
$306  12  00  00   SET    R2  destination address
                    A
$309  13  00  00   SET    R3  length
                    B
$30C  41           LD     @R1
$30D  52           ST     @R2
$30E  F3           DCR    R3
$30F  07           BNZ    $30C
$311  00           RTN
$312  60           RTS
```

Data will be poked from the Integer Basic program:

```
"A"       from Line 100
"B"       from Line 60
```

```
+---------------------------------------------------------------------------
|  TOPIC -- Apple II -- Red Book Mini-Assembler listing
+---------------------------------------------------------------------------

                 ***********************
                 *                     *
                 *       APPLE-II      *
                 *    MINI-ASSEMBLER   *
                 *                     *
                 *  COPYRIGHT 1977 BY  *
                 * APPLE COMPUTER INC. *
                 *                     *
                 * ALL RIGHTS RESERVED *
                 *                     *
                 *      S. WOZNIAK     *
                 *       A. BAUM       *
                 ***********************
                                                ; TITLE "APPLE-II MINI-ASSEMBLER"
                 FORMAT   EQU   $2E
                 LENGTH   EQU   $2F
                 MODE     EQU   $31
                 PROMPT   EQU   $33
                 YSAV     EQU   $34
                 L        EQU   $35
                 PCL      EQU   $3A
                 PCH      EQU   $3B
                 A1H      EQU   $3D
                 A2L      EQU   $3E
                 A2H      EQU   $3F
                 A4L      EQU   $42
                 A4H      EQU   $43
                 FMT      EQU   $44
                 IN       EQU   $200
                 INSDS2   EQU   $F88E
                 INSTDSP  EQU   $F8D0
                 PRBL2    EQU   $F94A
                 PCADJ    EQU   $F953
                 CHAR1    EQU   $F9B4
                 CHAR2    EQU   $F9BA
                 MNEML    EQU   $F9C0
                 MNEMR    EQU   $FA00
                 CURSUP   EQU   $FC1A
                 GETLNZ   EQU   $FD67
                 COUT     EQU   $FDED
                 BL1      EQU   $FE00
                 A1PCLP   EQU   $FE78
                 BELL     EQU   $FF3A
                 GETNUM   EQU   $FFA7
                 TOSUB    EQU   $FFBE
                 ZMODE    EQU   $FFC7
                 CHRTBL   EQU   $FFCC
                          ORG   $F500
F500: E9 81      REL      SBC   #$81        ;IS FMT COMPATIBLE
F502: 4A                  LSR               ;WITH RELATIVE MODE?
F503: D0 14               BNE   ERR3        ;  NO.
F505: A4 3F               LDY   A2H
F507: A6 3E               LDX   A2L         ;DOUBLE DECREMENT
F509: D0 01               BNE   REL2
F50B: 88                  DEY
F50C: CA         REL2     DEX
```

```
F50D: 8A                     TXA
F50E: 18                     CLC
F50F: E5 3A                  SBC   PCL        ;FORM ADDR-PC-2
F511: 85 3E                  STA   A2L
F513: 10 01                  BPL   REL3
F515: C8                     INY
F516: 98           REL3      TYA
F517: E5 3B                  SBC   PCH
F519: D0 6B        ERR3      BNE   ERR        ;ERROR IF >1-BYTE BRANCH
F51B: A4 2F        FINDOP    LDY   LENGTH
F51D: B9 3D 00     FNDOP2    LDA   A1H,Y      ;MOVE INST TO (PC)
F520: 91 3A                  STA   (PCL),Y
F522: 88                     DEY
F523: 10 F8                  BPL   FNDOP2
F525: 20 1A FC               JSR   CURSUP
F528: 20 1A FC               JSR   CURSUP     ;RESTORE CURSOR
F52B: 20 D0 F8               JSR   INSTDSP    ;TYPE FORMATTED LINE
F52E: 20 53 F9               JSR   PCADJ      ;UPDATE PC
F531: 84 3B                  STY   PCH
F533: 85 3A                  STA   PCL
F535: 4C 95 F5               JMP   NXTLINE    ;GET NEXT LINE
F538: 20 BE FF     FAKEMON3  JSR   TOSUB      ;GO TO DELIM HANDLER
F53B: A4 34                  LDY   YSAV       ;RESTORE Y-INDEX
F53D: 20 A7 FF     FAKEMON   JSR   GETNUM     ;READ PARAM
F540: 84 34                  STY   YSAV       ;SAVE Y-INDEX
F542: A0 17                  LDY   #$17       ;INIT DELIMITER INDEX
F544: 88           FAKEMON2  DEY              ;CHECK NEXT DELIM
F545: 30 4B                  BMI   RESETZ     ;ERR IF UNRECOGNIZED DELIM
F547: D9 CC FF               CMP   CHRTBL,Y   ;COMPARE WITH DELIM TABLE
F54A: D0 F8                  BNE   FAKEMON2   ;NO MATCH
F54C: C0 15                  CPY   #$15       ;MATCH, IS IT CR?
F54E: D0 E8                  BNE   FAKEMON3   ;NO, HANDLE IT IN MONITOR
F550: A5 31                  LDA   MODE
F552: A0 00                  LDY   #$0
F554: C6 34                  DEC   YSAV
F556: 20 00 FE               JSR   BL1        ;HANDLE CR OUTSIDE MONITOR
F559: 4C 95 F5               JMP   NXTLINE
F55C: A5 3D        TRYNEXT   LDA   A1H        ;GET TRIAL OPCODE
F55E: 20 8E F8               JSR   INSDS2     ;GET FMT+LENGTH FOR OPCODE
F561: AA                     TAX
F562: BD 00 FA               LDA   MNEMR,X    ;GET LOWER MNEMONIC BYTE
F565: C5 42                  CMP   A4L        ;MATCH?
F567: D0 13                  BNE   NEXTOP     ;NO, TRY NEXT OPCODE.
F569: BD C0 F9               LDA   MNEML,X    ;GET UPPER MNEMONIC BYTE
F56C: C5 43                  CMP   A4H        ;MATCH?
F56E: D0 0C                  BNE   NEXTOP     ;NO, TRY NEXT OPCODE
F570: A5 44                  LDA   FMT
F572: A4 2E                  LDY   FORMAT     ;GET TRIAL FORMAT
F574: C0 9D                  CPY   #$9D       ;TRIAL FORMAT RELATIVE?
F576: F0 88                  BEQ   REL        ;YES.
F578: C5 2E        NREL      CMP   FORMAT     ;SAME FORMAT?
F57A: F0 9F                  BEQ   FINDOP     ;YES.
F57C: C6 3D        NEXTOP    DEC   A1H        ;NO, TRY NEXT OPCODE
F57E: D0 DC                  BNE   TRYNEXT
F580: E6 44                  INC   FMT        ;NO MORE, TRY WITH LEN=2
F582: C6 35                  DEC   L          ;WAS L=2 ALREADY?
F584: F0 D6                  BEQ   TRYNEXT    ;NO.
F586: A4 34        ERR       LDY   YSAV       ;YES, UNRECOGNIZED INST.
F588: 98           ERR2      TYA
F589: AA                     TAX
F58A: 20 4A F9               JSR   PRBL2      ;PRINT ^ UNDER LAST READ
F58D: A9 DE                  LDA   #$DE       ;CHAR TO INDICATE ERROR
```

```
F58F: 20 ED FD               JSR   COUT        ;POSITION.
F592: 20 3A FF     RESETZ    JSR   BELL
F595: A9 A1        NXTLINE   LDA   #$A1        ;'!'
F597: 85 33                  STA   PROMPT      ;INITIALIZE PROMPT
F599: 20 67 FD               JSR   GETLNZ      ;GET LINE.
F59C: 20 C7 FF               JSR   ZMODE       ;INIT SCREEN STUFF
F59F: AD 00 02               LDA   IN          ;GET CHAR
F5A2: C9 A0                  CMP   #$A0        ;ASCII BLANK?
F5A4: F0 13                  BEQ   SPACE       ;YES
F5A6: C8                     INY
F5A7: C9 A4                  CMP   #$A4        ;ASCII '$' IN COL 1?
F5A9: F0 92                  BEQ   FAKEMON     ;YES, SIMULATE MONITOR
F5AB: 88                     DEY               ;NO, BACKUP A CHAR
F5AC: 20 A7 FF               JSR   GETNUM      ;GET A NUMBER
F5AF: C9 93                  CMP   #$93        ;':' TERMINATOR?
F5B1: D0 D5        ERR4      BNE   ERR2        ;NO, ERR.
F5B3: 8A                     TXA
F5B4: F0 D2                  BEQ   ERR2        ;NO ADR PRECEDING COLON.
F5B6: 20 78 FE               JSR   A1PCLP      ;MOVE ADR TO PCL, PCH.
F5B9: A9 03        SPACE     LDA   #$3         ;COUNT OF CHARS IN MNEMONIC
F5BB: 85 3D                  STA   A1H
F5BD: 20 34 F6     NXTMN     JSR   GETNSP      ;GET FIRST MNEM CHAR.
F5C0: 0A           NXTM      ASL
F5C1: E9 BE                  SBC   #$BE        ;SUBTRACT OFFSET
F5C3: C9 C2                  CMP   #$C2        ;LEGAL CHAR?
F5C5: 90 C1                  BCC   ERR2        ;NO.
F5C7: 0A                     ASL               ;COMPRESS-LEFT JUSTIFY
F5C8: 0A                     ASL
F5C9: A2 04                  LDX   #$4
F5CB: 0A           NXTM2     ASL               ;DO 5 TRIPLE WORD SHIFTS
F5CC: 26 42                  ROL   A4L
F5CE: 26 43                  ROL   A4H
F5D0: CA                     DEX
F5D1: 10 F8                  BPL   NXTM2
F5D3: C6 3D                  DEC   A1H         ;DONE WITH 3 CHARS?
F5D5: F0 F4                  BEQ   NXTM2       ;YES, BUT DO 1 MORE SHIFT
F5D7: 10 E4                  BPL   NXTMN       ;NO
F5D9: A2 05        FORM1     LDX   #$5         ;5 CHARS IN ADDR MODE
F5DB: 20 34 F6     FORM2     JSR   GETNSP      ;GET FIRST CHAR OF ADDR
F5DE: 84 34                  STY   YSAV
F5E0: DD B4 F9               CMP   CHAR1,X     ;FIRST CHAR MATCH PATTERN?
F5E3: D0 13                  BNE   FORM3       ;NO
F5E5: 20 34 F6               JSR   GETNSP      ;YES, GET SECOND CHAR
F5E8: DD BA F9               CMP   CHAR2,X     ;MATCHES SECOND HALF?
F5EB: F0 0D                  BEQ   FORM5       ;YES.
F5ED: BD BA F9               LDA   CHAR2,X     ;NO, IS SECOND HALF ZERO?
F5F0: F0 07                  BEQ   FORM4       ;YES.
F5F2: C9 A4                  CMP   #$A4        ;NO,SECOND HALF OPTIONAL?
F5F4: F0 03                  BEQ   FORM4       ;YES.
F5F6: A4 34                  LDY   YSAV
F5F8: 18           FORM3     CLC               ;CLEAR BIT-NO MATCH
F5F9: 88           FORM4     DEY               ;BACK UP 1 CHAR
F5FA: 26 44        FORM5     ROL   FMT         ;FORM FORMAT BYTE
F5FC: E0 03                  CPX   #$3         ;TIME TO CHECK FOR ADDR.
F5FE: D0 0D                  BNE   FORM7       ;NO
F600: 20 A7 FF               JSR   GETNUM      ;YES
F603: A5 3F                  LDA   A2H
F605: F0 01                  BEQ   FORM6       ;HIGH-ORDER BYTE ZERO
F607: E8                     INX               ;NO, INCR FOR 2-BYTE
F608: 86 35        FORM6     STX   L           ;STORE LENGTH
F60A: A2 03                  LDX   #$3         ;RELOAD FORMAT INDEX
F60C: 88                     DEY               ;BACKUP A CHAR
```

```
F60D: 86 3D     FORM7    STX   A1H        ;SAVE INDEX
F60F: CA                 DEX              ;DONE WITH FORMAT CHECK?
F610: 10 C9              BPL   FORM2      ;NO.
F612: A5 44              LDA   FMT        ;YES, PUT LENGTH
F614: 0A                 ASL              ;IN LOW BITS
F615: 0A                 ASL
F616: 05 35              ORA   L
F618: C9 20              CMP   #$20
F61A: B0 06              BCS   FORM8      ;ADD "$" IF NONZERO LENGTH
F61C: A6 35              LDX   L          ;AND DON'T ALREADY HAVE IT
F61E: F0 02              BEQ   FORM8
F620: 09 80              ORA   #$80
F622: 85 44     FORM8    STA   FMT
F624: 84 34              STY   YSAV
F626: B9 00 02           LDA   IN,Y       ;GET NEXT NONBLANK
F629: C9 BB              CMP   #$BB       ;';' START OF COMMENT?
F62B: F0 04              BEQ   FORM9      ;YES
F62D: C9 8D              CMP   #$8D       ;CARRIAGE RETURN?
F62F: D0 80              BNE   ERR4       ;NO, ERR.
F631: 4C 5C F5  FORM9    JMP   TRYNEXT
F634: B9 00 02  GETNSP   LDA   IN,Y
F637: C8                 INY
F638: C9 A0              CMP   #$A0       ;GET NEXT NON BLANK CHAR
F63A: F0 F8              BEQ   GETNSP
F63C: 60                 RTS
                         ORG   $F666
F666: 4C 92 F5  MINIASM  JMP   RESETZ
```

```
+------------------------------------------------------------------------------
|  TOPIC -- Apple II -- Red Book Floating point listing
+------------------------------------------------------------------------------
```

Apple II Reference Manual (Red Book), January 1978, pages 94-95.

```
                ***********************
                *                     *
                *  APPLE-II FLOATING  *
                *   POINT ROUTINES    *
                *                     *
                *  COPYRIGHT 1977 BY  *
                * APPLE COMPUTER INC. *
                *                     *
                * ALL RIGHTS RESERVED *
                *                     *
                *     S. WOZNIAK      *
                *                     *
                ***********************
                 TITLE "FLOATING POINT ROUTINES"
                SIGN      EPZ  $F3
                X2        EPZ  $F4
                M2        EPZ  $F5
                X1        EPZ  $F8
                M1        EPZ  $F9
                E         EPZ  $FC
                OVLOC     EQU  $3F5
                          ORG  $F425
F425: 18        ADD       CLC           CLEAR CARRY
F426: A2 02               LDX  #$2      INDEX FOR 3-BYTE ADD.
F428: B5 F9     ADD1      LDA  M1,X
F42A: 75 F5               ADC  M2,X     ADD A BYTE OF MANT2 TO MANT1
F42C: 95 F9               STA  M1,X
F42E: CA                  DEX           INDEX TO NEXT MORE SIGNIF. BYTE.
F42F: 10 F7               BPL  ADD1     LOOP UNTIL DONE.
F431: 60                  RTS           RETURN
F432: 06 F3     MD1       ASL  SIGN     CLEAR LSB OF SIGN.
F434: 20 37 F4            JSR  ABSWAP   ABS VAL OF M1, THEN SWAP WITH M2
F437: 24 F9     ABSWAP    BIT  M1       MANT1 NEGATIVE?
F439: 10 05               BPL  ABSWAP1  NO, SWAP WITH MANT2 AND RETURN.
F43B: 20 A4 F4            JSR  FCOMPL   YES, COMPLEMENT IT.
F43E: E6 F3               INC  SIGN     INCR SIGN, COMPLEMENTING LSB.
F440: 38        ABSWAP1   SEC           SET CARRY FOR RETURN TO MUL/DIV.
F441: A2 04     SWAP      LDX  #$4      INDEX FOR 4 BYTE SWAP.
F443: 94 FB     SWAP1     STY  E-1,X
F445: B5 F7               LDA  X1-1,X   SWAP A BYTE OF EXP/MANT1 WITH
F447: B4 F3               LDY  X2-1,X   EXP/MANT2 AND LEAVE A COPY OF
F449: 94 F7               STY  X1-1,X   MANT1 IN E (3 BYTES).  E+3 USED
F44B: 95 F3               STA  X2-1,X
F44D: CA                  DEX           ADVANCE INDEX TO NEXT BYTE
F44E: D0 F3               BNE  SWAP1    LOOP UNTIL DONE.
F450: 60                  RTS           RETURN
F451: A9 8E     FLOAT     LDA  #$8E     INIT EXP1 TO 14,
F453: 85 F8               STA  X1       THEN NORMALIZE TO FLOAT.
F455: A5 F9     NORM1     LDA  M1       HIGH-ORDER MANT1 BYTE.
F457: C9 C0               CMP  #$C0     UPPER TWO BITS UNEQUAL?
F459: 30 0C               BMI  RTS1     YES, RETURN WITH MANT1 NORMALIZED
F45B: C6 F8               DEC  X1       DECREMENT EXP1.
F45D: 06 FB               ASL  M1+2
F45F: 26 FA               ROL  M1+1     SHIFT MANT1 (3 BYTES) LEFT.
```

```
F461: 26 F9              ROL  M1
F463: A5 F8     NORM     LDA  X1        EXP1 ZERO?
F465: D0 EE              BNE  NORM1     NO, CONTINUE NORMALIZING.
F467: 60        RTS1     RTS            RETURN.
F468: 20 A4 F4  FSUB     JSR  FCOMPL    CMPL MANT1,CLEARS CARRY UNLESS 0
F46B: 20 7B F4  SWPALGN  JSR  ALGNSWP   RIGHT SHIFT MANT1 OR SWAP WITH
F46E: A5 F4     FADD     LDA  X2
F470: C5 F8              CMP  X1        COMPARE EXP1 WITH EXP2.
F472: D0 F7              BNE  SWPALGN   IF #,SWAP ADDENDS OR ALIGN MANTS.
F474: 20 25 F4           JSR  ADD       ADD ALIGNED MANTISSAS.
F477: 50 EA     ADDEND   BVC  NORM      NO OVERFLOW, NORMALIZE RESULT.
F479: 70 05              BVS  RTLOG     OV: SHIFT M1 RIGHT, CARRY INTO SIGN
F47B: 90 C4     ALGNSWP  BCC  SWAP      SWAP IF CARRY CLEAR,
                *       ELSE SHIFT RIGHT ARITH.
F47D: A5 F9     RTAR     LDA  M1        SIGN OF MANT1 INTO CARRY FOR
F47F: 0A                 ASL            RIGHT ARITH SHIFT.
F480: E6 F8     RTLOG    INC  X1        INCR X1 TO ADJUST FOR RIGHT SHIFT
F482: F0 75              BEQ  OVFL      EXP1 OUT OF RANGE.
F484: A2 FA     RTLOG1   LDX  #$FA      INDEX FOR 6:BYTE RIGHT SHIFT.
F486: 76 FF     ROR1     ROR  E+3,X
F488: E8                 INX            NEXT BYTE OF SHIFT.
F489: D0 FB              BNE  ROR1      LOOP UNTIL DONE.
F48B: 60                 RTS            RETURN.
F48C: 20 32 F4  FMUL     JSR  MD1       ABS VAL OF MANT1, MANT2
F48F: 65 F8              ADC  X1        ADD EXP1 TO EXP2 FOR PRODUCT EXP
F491: 20 E2 F4           JSR  MD2       CHECK PROD. EXP AND PREP. FOR MUL
F494: 18                 CLC            CLEAR CARRY FOR FIRST BIT.
F495: 20 84 F4  MUL1     JSR  RTLOG1    M1 AND E RIGHT (PROD AND MPLIER)
F498: 90 03              BCC  MUL2      IF CARRY CLEAR, SKIP PARTIAL PROD
F49A: 20 25 F4           JSR  ADD       ADD MULTIPLICAND TO PRODUCT.
F49D: 88        MUL2     DEY            NEXT MUL ITERATION.
F49E: 10 F5              BPL  MUL1      LOOP UNTIL DONE.
F4A0: 46 F3     MDEND    LSR  SIGN      TEST SIGN LSB.
F4A2: 90 BF     NORMX    BCC  NORM      IF EVEN,NORMALIZE PROD,ELSE COMP
F4A4: 38        FCOMPL   SEC            SET CARRY FOR SUBTRACT.
F4A5: A2 03              LDX  #$3       INDEX FOR 3 BYTE SUBTRACT.
F4A7: A9 00     COMPL1   LDA  #$0       CLEAR A.
F4A9: F5 F8              SBC  X1,X      SUBTRACT BYTE OF EXP1.
F4AB: 95 F8              STA  X1,X      RESTORE IT.
F4AD: CA                 DEX            NEXT MORE SIGNIFICANT BYTE.
F4AE: D0 F7              BNE  COMPL1    LOOP UNTIL DONE.
F4B0: F0 C5              BEQ  ADDEND    NORMALIZE (OR SHIFT RT IF OVFL).
F4B2: 20 32 F4  FDIV     JSR  MD1       TAKE ABS VAL OF MANT1, MANT2.
F4B5: E5 F8              SBC  X1        SUBTRACT EXP1 FROM EXP2.
F4B7: 20 E2 F4           JSR  MD2       SAVE AS QUOTIENT EXP.
F4BA: 38        DIV1     SEC            SET CARRY FOR SUBTRACT.
F4BB: A2 02              LDX  #$2       INDEX FOR 3-BYTE SUBTRACTION.
F4BD: B5 F5     DIV2     LDA  M2,X
F4BF: F5 FC              SBC  E,X       SUBTRACT A BYTE OF E FROM MANT2.
F4C1: 48                 PHA            SAVE ON STACK.
F4C2: CA                 DEX            NEXT MORE SIGNIFICANT BYTE.
F4C3: 10 F8              BPL  DIV2      LOOP UNTIL DONE.
F4C5: A2 FD              LDX  #$FD      INDEX FOR 3-BYTE CONDITIONAL MOVE
F4C7: 68        DIV3     PLA            PULL BYTE OF DIFFERENCE OFF STACK
F4C8: 90 02              BCC  DIV4      IF M2<E THEN DON'T RESTORE M2.
F4CA: 95 F8              STA  M2+3,X
F4CC: E8        DIV4     INX            NEXT LESS SIGNIFICANT BYTE.
F4CD: D0 F8              BNE  DIV3      LOOP UNTIL DONE.
F4CF: 26 FB              ROL  M1+2
F4D1: 26 FA              ROL  M1+1      ROLL QUOTIENT LEFT, CARRY INTO LSB
F4D3: 26 F9              ROL  M1
F4D5: 06 F7              ASL  M2+2
```

```
F4D7: 26 F6                ROL  M2+1     SHIFT DIVIDEND LEFT
F4D9: 26 F5                ROL  M2
F4DB: B0 1C                BCS  OVFL     OVFL IS DUE TO UNNORMED DIVISOR
F4DD: 88                   DEY           NEXT DIVIDE ITERATION.
F4DE: D0 DA                BNE  DIV1     LOOP UNTIL DONE 23 ITERATIONS.
F4E0: F0 BE                BEQ  MDEND    NORM. QUOTIENT AND CORRECT SIGN.
F4E2: 86 FB     MD2        STX  M1+2
F4E4: 86 FA                STX  M1+1     CLEAR MANT1 (3 BYTES) FOR MUL/DIV.
F4E6: 86 F9                STX  M1
F4E8: B0 0D                BCS  OVCHK    IF CALC. SET CARRY,CHECK FOR OVFL
F4EA: 30 04                BMI  MD3      IF NEG THEN NO UNDERFLOW.
F4EC: 68                   PLA           POP ONE RETURN LEVEL.
F4ED: 68                   PLA
F4EE: 90 B2                BCC  NORMX    CLEAR X1 AND RETURN.
F4F0: 49 80     MD3        EOR  #$80     COMPLEMENT SIGN BIT OF EXPONENT.
F4F2: 85 F8                STA  X1       STORE IT.
F4F4: A0 17                LDY  #$17     COUNT 24 MUL/23 DIV ITERATIONS.
F4F6: 60                   RTS           RETURN.
F4F7: 10 F7     OVCHK      BPL  MD3      IF POSITIVE EXP THEN NO OVFL.
F4F9: 4C F5 03  OVFL       JMP  OVLOC
                           ORG  $F63D
F63D: 20 7D F4  FIX1       JSR  RTAR
F640: A5 F8     FIX        LDA  X1
F642: 10 13                BPL  UNDFL
F644: C9 8E                CMP  #$8E
F646: D0 F5                BNE  FIX1
F648: 24 F9                BIT  M1
F64A: 10 0A                BPL  FIXRTS
F64C: A5 FB                LDA  M1+2
F64E: F0 06                BEQ  FIXRTS
F650: E6 FA                INC  M1+1
F652: D0 02                BNE  FIXRTS
F654: E6 F9                INC  M1
F656: 60        FIXRTS     RTS
F657: A9 00     UNDFL      LDA  #$0
F659: 85 F9                STA  M1
F65B: 85 FA                STA  M1+1
F65D: 60                   RTS
```

```
+------------------------------------------------------------------------
|  TOPIC -- Apple II -- WOZPAK Floating point routines description
+------------------------------------------------------------------------
```

Wozpak][, November 1979, pages 109-115.

FLOATING POINT PACKAGE

The mantissa-exponent, or 'floating point' numerical representation is widely used by computers to express values with a wide dynamic range. With floating point representation, the number 7.5 x 10^22 requires no more memory to store than the number 75 does. We have allowed for binary floating point arithmetic on the APPLE][computer by providing a useful subroutine package in ROM, which performs the common arithmetic functions. Maximum precision is retained by these routines and overflow conditions such as 'divide by zero' are trapped for the user. The 4-byte floating point number representation is compatible with future APPLE products such as floating point BASIC.

A small amount of memory in Page Zero is dedicated to the floating point workspace, including the two floating-point accumulators, FP1 and FP2. After placing operands in these accumulators, the user calls subroutines in the ROM which perform the desired arithmetic operations, leaving results in FP1. Should an overflow condition occur, a jump to location $3F5 is executed, allowing a user routine to take appropriate action.

```
        FLOATING POINT REPRESENTATION

      ______   ______   ______   ______
     |      | |      | |      | |      |
     |      | |  HI  | |      | | LOW  |
     |______| |______| |______| |______|

    Exponent       Signed Mantissa
```

1. Mantissa

The floating point mantissa is stored in two's complement representation with the sign at the most significant bit (MSB) position of the high-order mantissa byte. The mantissa provides 24 bits of precision, including sign, and can represent 24-bit integers precisely. Extending precision is simply a matter of adding bytes at the low order end of the mantissa.

Except for magnitudes less than 2^-128 (which lose precision) mantissa are normalized by the floating point routines to retain maximum precision. That is, the numbers are adjusted so that the upper two high-order mantissa bits are unequal.

```
      HIGH-ORDER MANTISSA BYTE
     01.XXXXXX  Positive mantissa.
     10.XXXXXX  Negative mantissa.
     00.XXXXXX  Unnormalized mantissa.
     11.XXXXXX  Exponent = -128.
```

2. Exponent.

The exponent is a binary scaling factor (power of two) which is applied to the mantissa. Ranging from -128 to +127, the exponent is stored in standard two's complement representation except for the sign bit which is complemented. This representation allows direct comparison of exponents, since they are stored in increasing numerical sequence. The most negative exponent, corresponding to the smallest magnItude, -128, is stored as $00 ($ means hexidecimal) and the most positive, +127, is stored as $FF (all ones).

```
     EXPONENT      STORED AS

      +127      11111111  ($FF)

        +3      10000011  ($83)
        +2      10000010  ($82)
        +1      10000001  ($81)
         0      10000000  ($80)
        -1      01111111  ($7F)
        -2      01111110  ($7E)
        -3      01111101  ($7D)

      -128      00000000  ($00)
```

The smallest magnitude which can be represented is 2^-150.

```
      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
     |  0  |  |  0  |  |  0  |  |  1  |
     |_____|  |_____|  |_____|  |_____|

               HIGH                LOW
       EXP            MANTISSA
```

The largest positive magnitude which can be represented is +2^128-1.

```
      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
     | $7F |  | $7F |  | $FF |  | $FF |
     |_____|  |_____|  |_____|  |_____|

       EXP            MANTISSA
```

FLOATING POINT REPRESENTATION EXAMPLES

DECIMAL NUMBER	HEX EXPONENT	HEX MANTISSA
+ 3	81	60 00 00
+ 4	82	40 00 00
+ 5	82	50 00 00
+ 7	82	70 00 00
+12	83	60 00 00
+15	83	78 00 00
+17	84	44 00 00
+20	84	50 00 00
+60	85	78 00 00

```
     - 3         81      A0 00 00
     - 4         81      80 00 00
     - 5         82      B0 00 00
     - 7         82      90 00 00
     -12         83      A0 00 00
     -15         83      88 00 00
     -17         84      BC 00 00
     -20         84      B0 00 00
     -60         85      88 00 00
```

FLOATING POINT SUBROUTINE DESCRIPTIONS

FCOMPL subroutine (address $F4A4)

Purpose: FCOMPL is used to negate floating point numbers.

Entry: A normalized or unnormalized value is in FP1 (floating point accumulator 1).

Uses: NORM, RTLOG.

Exit: The value in FP1 is negated and then normalized to retain precision. The 3-byte FP1 extension, E, may also be altered but FP2 and SIGN are not disturbed. The 6502 A-REG is altered and the X-REG is cleared. The Y-REG is not disturbed.

Caution: Attempting to negate -2^128 will result in an overflow since +2^128 is not representable, and a jump to location $3F5 will be executed, with the following contents in FP1.

```
       _____    _____    _____    _____
      |     |  |     |  |     |  |     |
FP1:  |  0  |  | $80 |  |  0  |  |  0  |
      |_____|  |_____|  |_____|  |_____|

        X1       M1
```

Example: Prior to calling FCOMPL, FP1 contains +15.

```
       _____    _____    _____    _____
      |     |  |     |  |     |  |     |
FP1:  | $83 |  | $78 |  |  0  |  |  0  |   (+15)
      |_____|  |_____|  |_____|  |_____|

        X1       M1
```

After calling FCOMPL as a subroutine, FP1 contains -15.

```
       _____    _____    _____    _____
      |     |  |     |  |     |  |     |
FP1:  | $83 |  | $88 |  |  0  |  |  0  |   (+15)
      |_____|  |_____|  |_____|  |_____|

        X1       M1
```

```
FADD subroutine (address $F46E)

Purpose: To add two numbers in floating point form.

Entry: The two addends are in FP1 and FP2 respectively.  For maximum
precision, both should be normalized.

Uses: SWPALGN, ADD, NORM, RTLOG.

Exit: The normalized sum is left in FP1.  FP2 contains the addend of
greatest magnitude.  E is altered but sign is not.  The A-REG is altered
and the X-REG is cleared.  The sum mantissa is truncated to 24 bits.

Caution: Overflow may result if the sum is less that -2^128 or greater than
+2^128-1.  If so, a jump to location $3F5 is executed leaving 0 in X1, and
twice the proper sum in the mantissa M1.  The sign bit is left in the
carry, 0 for positive, 1 for negative.

        _____    __________
       |     |  |          |
FP1:   |  0  |  | X.YYY... |
       |_____|  |__________|

         X1        M1

(For carry=0, true sum=+X.YYY x 2^128)

Example: Prior to calling FADD, FP1 contains +12 and FP2 contains -5.

        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $83 |  | $60 |  |  0  |  |  0  |   (+12)
       |_____|  |_____|  |_____|  |_____|

         X1        M1

        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP2:   | $82 |  | $B0 |  |  0  |  |  0  |   (-5)
       |_____|  |_____|  |_____|  |_____|

         X2        M2

After calling FADD, FP1 contains +7 (FP2 contains +12).

        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1    | $82 |  | $70 |  |  0  |  |  0  |   (+7)
       |_____|  |_____|  |_____|  |_____|

         X1        M1
```

FSUB subroutine (address $F468)

Purpose: To subtract two floating point numbers.

Entry: The minuend is in FP1 and the subtrahend is in FP2. Both should be normalized to retain maximum precision prior to calling FSUB.

Uses: FCOMPL, ALGNSWP, FADD, ADD, NORM, RTLOG.

Exit: The normalized difference is in FP1 with the mantissa truncated to 24 bits. FP2 holds either the minued or the negated subtrahend, whichever is of greater magnitude. E is altered but SIGN and SCR are not. the A-REG is altered and the X-REG is cleared. The Y-REG is not disturbed.

Cautions: An exit to location S3F5 is taken if the result is less than -2^128 or greater than +2^128-1. or if the subtrahend is -2^128.

Example: Prior to calling FSUB, FP1 contains +7 (minuend) and FP2 contalns -5 (subtrahend).

```
         _____    _____    _____    _____
        |     |  |     |  |     |  |     |
FP1:    | $82 |  | $70 |  |  0  |  |  0  |   (+12)
        |_____|  |_____|  |_____|  |_____|

          X1       M1

         _____    _____    _____    _____
        |     |  |     |  |     |  |     |
FP2:    | $82 |  | $B0 |  |  0  |  |  0  |   (- 5)
        |_____|  |_____|  |_____|  |_____|

          X2       M2
```

After calling FSUB, FP1 contains +12 and FP2 contains +7.

```
         _____    _____    _____    _____
        |     |  |     |  |     |  |     |
FP1:    | $83 |  | $60 |  |  0  |  |  0  |   (+12)
        |_____|  |_____|  |_____|  |_____|

          X1       M1
```

FMUL subroutine (address $F48C)

Purpose: To multiply floating point numbers.

Entry: The multiplicand and multiplier must reside in FP1 and FP2 respectively. Both should be normalized prior to calling FMUL to retain maximum precision.

Uses: MD1, MD2, RTLOG1, ADD, MDEND.

Exit: The signed normalized floating point product is left in FP1. M1 is truncated to contain the 24 most significant mantissa bits (including sign). The absolute value of the multiplier mantissa (M2) is left in FP2. E, SIGN, and SCR are altered. The A- and X-REGs are altered and the Y-REG contains $FF upon exit.

Cautions: An exit to location $3F5 is taken if the product is less than -2^128 or greater than +2^128-1.

Notes: FMUL will run faster if the absolute value of the multiplier mantissa contains fewer '1's than the absolute value of the multiplicand mantissa.

Example: Prior to calling FMUL, FP1 contains +12 and FP2 contains -5.

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $83 |  | $60 |  |  0  |  |  0  |   (+12)
       |_____|  |_____|  |_____|  |_____|

         X1       M1

        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP2:   | $82 |  | $B0 |  |  0  |  |  0  |   (- 5)
       |_____|  |_____|  |_____|  |_____|

         X2       M2
```

After calling FMUL, FP1 contains -60 and FP2 contains +5.

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $85 |  | $88 |  |  0  |  |  0  |   (-60)
       |_____|  |_____|  |_____|  |_____|

         X1       M1

        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP2:   | $82 |  | $50 |  |  0  |  |  0  |   (+ 5)
       |_____|  |_____|  |_____|  |_____|

         X2       M2
```

FDIV subroutine (addr $F4B2)

Purpose: To perform division of floating point numbers.

Entry: The normalized dividend is in FP2 and the normalized divisor is in FP1.

Exit: The signed normalized floating point quotient is left in FP1. The mantissa (M1) is truncated to 24 bits. The 3-bit M1 extension (E) contains the absolute value of the divisor mantissa. MD2, SIGN, and SCR are altered. The A- and X-REGs are altered and the Y-REG is cleared.

Uses: MD1, MD2, MDEND.

Cautions: An exit to location $3F5 is taken if the quotient is less than -2^128 or greater than +2^128-1

Notes: MD2 contains the remainder mantissa (equivalent to the MOD function). The remainder exponent is the same as the quotient exponent, or 1 less if the dividend mantissa magnitude is less than the divisor mantissa magnitude.

Example: Prior to calling FDIV, FP1 contains -60 (dividend), and FP2 contains +12 (divisor).

```
           _____    _____    _____    _____
          |     |  |     |  |     |  |     |
    FP1:  | $85 |  | $80 |  |  0  |  |  0  |   (-60)
          |_____|  |_____|  |_____|  |_____|

            X1       M1

           _____    _____    _____    _____
          |     |  |     |  |     |  |     |
    FP2   | $83 |  | $60 |  |  0  |  |  0  |   (+12)
          |_____|  |_____|  |_____|  |_____|

            X1       M1
```

After calling FMUL, FP1 contains -5 and M2 contains 0.

```
           _____    _____    _____    _____
          |     |  |     |  |     |  |     |
    FP1:  | $82 |  | $B0 |  |  0  |  |  0  |   (-5)
          |_____|  |_____|  |_____|  |_____|

            X1       M1
```

FLOAT Subroutine (address $F451)

Purpose: To convert integers to floating point representation.

Entry: A signed (two's complement) 2-byte integer is stored in M1 (high-order byte) and M1+1 (low-order byte). M1+2 must be cleared by user prior to entry.

Uses: NORM1.

Exit: The normalized floating point equivalent is left in FP1. E, FP2, SIGN, and SCR are not disturbed. The A-REG contains a copy of the high-order mantissa byte upon exit but the X- and Y-REGs are not disturbed. The carry is cleared.

Notes: To float a 1-byte integer, place it in M1+1 and clear M1 as well as

M1+2 prior to calling FLOAT.

FLOAT takes approximately 3 msec. lonqer to convert zero to floating point form than other arguments. The user may check for zero prior to calling FLOAT and increase throughput.

```
           *
           *  LOW-ORDER INT. BYTE IN A-REG
           * HIGH-ORDER BYTE IN Y-REG
           *
85 FA      XFLOAT  STA  M1+1
84 F9              STY  M1    INIT MANT1
A0 00              LDY  #$0
84 FB              STY  M1+2
05 D9              ORA  M1    CHK BOTH
                              BYTES FOR
D0 03              BNE  TOFLOAT  ZERO
85 F8              STA  X1    IF SO CLR X1
60                 RTS        AND RETURN
4C 51 F4  TOFLOAT  JMP  FLOAT ELSE FLOAT
                              INTEGER
```

Example: Float +274 ($0112 hex)

```
             CALLING SEQUENCE

A0 01              LDY  #$01  HIGH-ORDER
                              INTEGER BYTE
A9 12              LDA  #$12  LOW-ORDER
                              INTEGER BYTE
84 F9              STY M1
85 FA              STA M1+1
A9 00              LDA #$00
85 F8              STA M1+2
20 51 F4           JSR FLOAT
```

Upon returning from FLOAT, FP1 contains the floating point representation of +274.

```
      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
FP1  | $88 |  | $44 |  | $80 |  |  0  |   (+274)
     |_____|  |_____|  |_____|  |_____|

       X1       M1
```

FIX subroutine (address $F640)

Purpose: To extract the integer portion of a floating point number with truncation (ENTIER function).

Entry: A floating point value is in FP1. It need not be normalized.

Uses: RTAR.

Exit: The two-byte signed two's complement representation of the integer portion is left in M1 (high-order byte) and M1+1 (low-order byte). The floating point values +24.63 and -61.2 are converted to the integers +24 and -61 respectively. FP1 and E are altered but FP2, E, SIGN, and SCR are not. The A- and X-REGs are altered but the Y-REG is not.

Example: The floating point value +274 is in FP1 prior to calling FIX.

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $88 |  | $44 |  | $80 |  |  0  |   (+274)
       |_____|  |_____|  |_____|  |_____|

         X1       M1
```

After calling FIX, M1 (high-order byte) and M1+1 (low-order byte) contain the integer representation of +274 ($0112).

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $8E |  | $01 |  | $12 |  |  0  |
       |_____|  |_____|  |_____|  |_____|

         X1       M1
```

Note: FP1 contains an unnormalized representation of +274 upon exit.

NORM Subroutine (address $F463)

Purpose: To normalize the value in FP1, thus insuring maximum precision.

Entry: A normalized or unnormalized value is in FP1.

Exit: The value in FP1 is normalized. A zero mantissa will exit with X1=0 (2 exponent). If the exponent on exit is -128 (X1=0) then the mantissa (M1) is not necessarily normalized (with the two high-order mantissa bits unequal). E, FP2, SIGN, AND SCR are not distubed. The A-REG is disturbed but the X- and Y-REGs are not. The carry is set.

Example: FP1 contains +12 in unnormalized form (as .0011 x 2).

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $86 |  | $0C |  |  0  |  |  0  |   (+12)
       |_____|  |_____|  |_____|  |_____|

         x1       M1
```

Upon exit from NORM, FP1 contains +12 in normalized form (as 1.1 x 2).

```
        _____    _____    _____    _____
       |     |  |     |  |     |  |     |
FP1:   | $83 |  | $60 |  |  0  |  |  0  |   (+12)
```

```
     |_____|  |_____|  |_____|  |_____|

       X1       M1
```

NORM1 subroutine (address $F455)

Purpose: To normalize a floating point value in FP1 when it is known the exponent is not -128 (X1=0) upon entry.

Entry: An unnormalized number is in FP1. The exponent byte should not be 0 for normal use.

Exit: The normalized value is in FP1. E, FP2, SIGN, and SCR are not not disturbed. The A-REG is altered but the X- and Y-REGs are not.

ADD Subroutine (address $F425)

Purpose: To add the two mantissas (M1 and M2) as 3-byte integers.

Entry: Two mantissas are in M1 (through M1+2) and M2 (through M2+2). They should be aligned, that is with identical exponents, for use in the FADD and FSUB subroutines.

Exit: the 24-bit integer sum is in M1 (high-order byte in M1, low-order byte in M1+2). FP2, X1, E, SIGN and SCR are not disturbed. The A-REG contains the high-order byte of the sum, the X-REG contains $FF and the Y-REG is not altered. The carry is the '25th' sum bit.

Example: FP1 contains +5 and FP2 contains +7 prior to calling ADD.

```
      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
FP1: | $82 |  | $50 |  |  0  |  |  0  |   (+5)
     |_____|  |_____|  |_____|  |_____|

       X1       M1

      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
FP2: | $82 |  | $70 |  |  0  |  |  0  |   (+7)
     |_____|  |_____|  |_____|  |_____|
```

Upon exit, M1 contains the overflow value for +12. Note that the sign bit is incorrect. This is taken care of with a call to the right shift routine.

```
      _____    _____    _____    _____
     |     |  |     |  |     |  |     |
FP:  | $82 |  | $C0 |  |  0  |  |  0  |   (+12)
     |_____|  |_____|  |_____|  |_____|
```

ABSWAP Subroutine (address $F437)

Purpose: To take the absolute value of FP1 and then swap FP1 with FP2. Note that two sequential calls to ABSWAP will take the absolute values of both FP1 and FP2 in preparation for a multiply or divide.

Entry: FP1 and FP2 contain floating point values.

Exit: The absolute value of the original FP1 contents are in FP2 and the original FP2 contents are in FP1. The least significant bit of SIGN is complemented if a negation takes place (if the original FP1 contents are negative) by means of an increment. SCR and E are used. The A-REG contains a copy of X2, the X-REG is cleared, and the Y-REG is not altered.

RTAR Subroutine (address $F47D)

Purpose: To shift M1 right one bit position while incrementing X1 to compensate for scale. This is roughly the opposite of the NORM subroutine.

Entry: A normalized or unnormalized floating point value is in FP1.

Exit: The 6-byte field MANT1 and E is shifted right one bit arithmetically and X1 is incremented by 1 to retain proper scale. The sign bit of MANT1 (MSB of M1) is unchanged. FP2, SIGN, and SCR are not disturbed. The A-REG contains the least significant byte of E (E+2), the X-REG is cleared, and the Y-REG is not disturbed.

Caution: If X1 increments of 0 (overflow) then an exit to location $3F5 is taken, the A-REG contains the high-order MANT1 byte, M1 and X1 is cleared. FP2, SIGN, SCR, and the X- and Y-REGs are not disturbed.

Uses: RTLOG

Example: Prior to calling RTAR, FP1 contains the normalized value -7.

```
       _____   _____   _____   _____
      |     | |     | |     | |     |
FP1   | $83 | | $A0 | |  0  | |  0  |   (-7)
      |_____| |_____| |_____| |_____|

        X1      M1
```

After calling RTAR, FP1 contains the unnormalized value -7 (note that precision is lost off the low-order end of M1).

```
       _____   _____   _____   _____
      |     | |     | |     | |     |
FP1   | $84 | | $D0 | |  0  | |  0  |   (-7)
      |_____| |_____| |_____| |_____|

        X1      M1
```

Note: M1 sign bit is unchanged.

RTLOG subroutine (address $F480)

Purpose: To shift the 6-byte field MANT1 and E one bit to the right (toward the least significant bit). The 6502 carry bit is shifted into the high-order M1 bit. This is useful in correcting binary sum overflows.

Entry: A normalized or unnormalized floating point value is in FP1. The carry must be cleared or set by the user since it is shifted Into the sign bit of M1.

Exit: Same as RTAR except that the sign of M1 is not preserved (it is set to the value of the carry bit on entry)

Caution: Same as RTAR.

Example: Prior to calling RTLOG, FP1 contains the normalized value -12 and the carry is clear.

```
         _____    _____    _____    _____
        |     |  |     |  |     |  |     |
FP1:    | $83 |  | $A0 |  |  0  |  |  0  |   (-12)
        |_____|  |_____|  |_____|  |_____|

          X1       M1
```

After calling RTLOG, M1 is shifted one bit to the right and the sign bit is clear. X1 is incremented by 1.

```
         _____    _____    _____    _____
        |     |  |     |  |     |  |     |
FP1:    | $84 |  | $50 |  |  0  |  |  0  |   (+20)
        |_____|  |_____|  |_____|  |_____|

          X1       M1
```

Note: The bit shifted off the end of MANT1 is rotated into the high-order bit of the 3-byte extension E. The 3-byte E field is also shifted one bit to the right.

RTLOG1 subroutine (address $F484)

Purpose: To shift MANT1 and E right one bit without adjusting X1. This is used by the multiply loop. The carry is shifted into the sign bit of MANT1.

Entry: M1 and E contain a 6-byte unsigned field. E is the 3-byte low-order extension of MANT1.

Exit: Same as RTLOG except that X1 is not altered and an overflow exit cannot occur.

MD2 subroutine (address $F4E2)

Purpose: To clear the 3-byte MANT1 field for FMUL and FDIV, check for inital result exponent overflow (and underflow), and initialize the X-REG to $17 for loop counting.

Entry: the X-REG is cleared by the user since it is placed in the 3 bytes of MANT1. The A-REG contains the result of an exponent addition (FMUL) or subtraction (FDIV). The carry and sign status bits should be set according to this addition or subtraction for overflow and underflow determination.

Exit: The 3 bytes of M1 are cleared (or all set to the contents of the X-REG on Entry) and the Y-REG is loaded with $17. The sign bit of the A-REG is complemented and a copy of the A-REG is stored in X1. FP2, SIGN, SCR, and the X-REG are not disturbed.

Uses: NORM.

Caution: Exponent overflow results in an exit to location $3F5. Exponent underflow results in an early return from the calling subroutine (FDIV or FMUL) with a floating point zero in FP1. Because MD2 pops a return address off the stack, it may only be called by another subroutine.

```
+-------------------------------------------------------------------------
|  TOPIC -- Apple II -- DDJ Floating point article
+-------------------------------------------------------------------------
```

Dr. Dobb's Journal, August 1976, pages 17-19.

Floating Point Routines for the 6502

by Roy Rankin, Department of Mechanical Engineering,
Stanford University, Stanford, CA 94305
(415) 497-1822

and

Steve Wozniak, Apple Computer Company
770 Welch Road, Suite 154
Palo Alto, CA 94304
(415) 326-4248

Editor's Note: Although these routines are for the 6502, it would appear that one could generate equivalent routines for most of the "traditional" microprocessors, relatively easily, by following the flow of the algorithms given in the excellent comments included in the program listing. This is particularly true of the transcendental functions, which were directly modeled after well-known and proven algorithms, and for which, the comments are relatively machine independent.

These floating point routines allow 6502 users to perform most of the more popular and desired floating point and transcendental functions, namely:

Natural Log - LOG
Common Log - LOG10
Exponential - EXP
Floating Add - FADD
Floating Subtract - FSUB
Floating Multiply - FMUL
Floating Divide - FDIV
Convert Floating to Fixed - FIX
Convert Fixed to Floating - FLOAT

They presume a four-byte floating point operand consisting of a one-byte exponent ranging from -128 to +127 and a 24-bit two's complement mantissa between 1.0 and 2.0.

The floating point routines were done by Steve Wozniak, one of the principals in Apple Computer Company. The transcendental functions were patterned after those offered by Hewlett-Packard for their HP2100 minicomputer (with some modifications), and were done by Roy Rankin, a Ph.D. student at Stanford University.

There are three error traps; two for overflow, and one for prohibited logarithm argument. ERROR (1D06) is the error exit used in the event of a non-positive log argument. OVFLW (1E3B) is the error exit for overflow occuring during calculation of e to some power. OVFL (1FE4) is the error exit for overflow in all of the floating point routines. There is no trap for underflow; in such cases, the result is set to 0.0.

All routines are called and exited in a uniform manner: The arguments(s) are placed in the specified floating point storage locations (for specifics, see the documentation preceeding each routine in the listing), then a JSR is used to enter the desired routine. Upon normal completion, the called routine is exited via a subroutine return instruction (RTS).

Note: The preceeding documentation was written by the Editor, based on phone conversations with Roy and studying the listing. There is a high probability that it is correct. However, since it was not written nor reviewed by the authors of these routines, the preceeding documentation may contain errors in concept or in detail.

-- JCW, Jr.

```
    In the Exponent:
    00 Represents -128
          ...
    7F Represents -1
    80 Represents 0
    81 Represents +1
          ...
    FF Represents +127


  Exponent    Two's Complement Mantissa
  SEEEEEEE  SM.MMMMMM  MMMMMMMM  MMMMMMMM
     n         n+1       n+2       n+3
```

```
               *           JULY 5, 1976
               *     BASIC FLOATING POINT ROUTINES
               *       FOR 6502 MICROPROCESSOR
               *       BY R. RANKIN AND S. WOZNIAK
               *
               *     CONSISTING OF:
               *        NATURAL LOG
               *        COMMON LOG
               *        EXPONENTIAL (E**X)
               *        FLOAT      FIX
               *        FADD       FSUB
               *        FMUL       FDIV
               *
               *
               *     FLOATING POINT REPRESENTATION (4-BYTES)
               *                     EXPONENT BYTE 1
               *                     MANTISSA BYTES 2-4
               *
               *     MANTISSA:    TWO'S COMPLIMENT REPRESENTATION WITH SIGN IN
               *       MSB OF HIGH-ORDER BYTE.  MANTISSA IS NORMALIZED WITH AN
               *       ASSUMED DECIMAL POINT BETWEEN BITS 5 AND 6 OF THE HIGH-ORDER
               *       BYTE.  THUS THE MANTISSA IS IN THE RANGE 1. TO 2. EXCEPT
```

```
                 *        WHEN THE NUMBER IS LESS THAN 2**(-128).
                 *
                 *      EXPONENT:    THE EXPONENT REPRESENTS POWERS OF TWO.  THE
                 *        REPRESENTATION IS 2'S COMPLIMENT EXCEPT THAT THE SIGN
                 *        BIT (BIT 7) IS COMPLIMENTED.  THIS ALLOWS DIRECT COMPARISON
                 *        OF EXPONENTS FOR SIZE SINCE THEY ARE STORED IN INCREASING
                 *        NUMERICAL SEQUENCE RANGING FROM $00 (-128) TO $FF (+127)
                 *        ($ MEANS NUMBER IS HEXADECIMAL).
                 *
                 *      REPRESENTATION OF DECIMAL NUMBERS:    THE PRESENT FLOATING
                 *        POINT REPRESENTATION ALLOWS DECIMAL NUMBERS IN THE APPROXIMATE
                 *        RANGE OF 10**(-38) THROUGH 10**(38) WITH 6 TO 7 SIGNIFICANT
                 *        DIGITS.
                 *
                 *
0003                       ORG 3        SET BASE PAGE ADRESSES
0003  EA         SIGN      NOP
0004  EA         X2        NOP          EXPONENT 2
0005  00 00 00   M2        BSS 3        MANTISSA 2
0008  EA         X1        NOP          EXPONENT 1
0009  00 00 00   M1        BSS 3        MANTISSA 1
000C             E         BSS 4        SCRATCH
0010             Z         BSS 4
0014             T         BSS 4
0018             SEXP      BSS 4
001C  00         INT       BSS 1
                 *
1D00                       ORG $1D00    STARTING LOCATION FOR LOG
                 *
                 *
                 *      NATURAL LOG OF MANT/EXP1 WITH RESULT IN MANT/EXP1
                 *
1D00  A5 09      LOG       LDA M1
1D02  F0 02                BEQ ERROR
1D04  10 01                BPL CONT     IF ARG>0 OK
1D06  00         ERROR     BRK          ERROR ARG<=0
                 *
1D07  20 1C 1F   CONT      JSR SWAP     MOVE ARG TO EXP/MANT2
1D0A  A5 04                LDA X2       HOLD EXPONENT
1D0C  A0 80                LDY =$80
1D0E  84 04                STY X2       SET EXPONENT 2 TO 0 ($80)
1D10  49 80                EOR =$80     COMPLIMENT SIGN BIT OF ORIGINAL EXPONENT
1D12  85 0A                STA M1+1     SET EXPONENT INTO MANTISSA 1 FOR FLOAT
1D14  A9 00                LDA =0
1D16  85 09                STA M1       CLEAR MSB OF MANTISSA 1
1D18  20 2C 1F             JSR FLOAT    CONVERT TO FLOATING POINT
1D1B  A2 03                LDX =3       4 BYTE TRANSFERS
1D1D  B5 04      SEXP1     LDA X2,X
1D1F  95 10                STA Z,X      COPY MANTISSA TO Z
1D21  B5 08                LDA X1,X
1D23  95 18                STA SEXP,X   SAVE EXPONENT IN SEXP
1D25  BD D1 1D             LDA R22,X    LOAD EXP/MANT1 WITH SQRT(2)
1D28  95 08                STA X1,X
1D2A  CA                   DEX
1D2B  10 F0                BPL SEXP1
1D2D  20 4A 1F             JSR FSUB     Z-SQRT(2)
1D30  A2 03                LDX =3       4 BYTE TRANSFER
1D32  B5 08      SAVET     LDA X1,X     SAVE EXP/MANT1 AS T
1D34  95 14                STA T,X
1D36  B5 10                LDA Z,X      LOAD EXP/MANT1 WITH Z
1D38  95 08                STA X1,X
1D3A  BD D1 1D             LDA R22,X    LOAD EXP/MANT2 WITH SQRT(2)
```

```
1D3D  95 04            STA X2,X
1D3F  CA               DEX
1D40  10 F0            BPL SAVET
1D42  20 50 1F         JSR FADD    Z+SQRT(2)
1D45  A2 03            LDX =3      4 BYTE TRANSFER
1D47  B5 14     TM2    LDA T,X
1D49  95 04            STA X2,X    LOAD T INTO EXP/MANT2
1D4B  CA               DEX
1D4C  10 F9            BPL TM2
1D4E  20 9D 1F         JSR FDIV    T=(Z-SQRT(2))/(Z+SQRT(2))
1D51  A2 03            LDX =3      4 BYTE TRANSFER
1D53  B5 08     MIT    LDA X1,X
1D55  95 14            STA T,X     COPY EXP/MANT1 TO T AND
1D57  95 04            STA X2,X    LOAD EXP/MANT2 WITH T
1D59  CA               DEX
1D5A  10 F7            BPL MIT
1D5C  20 77 1F         JSR FMUL    T*T
1D5F  20 1C 1F         JSR SWAP    MOVE T*T TO EXP/MANT2
1D62  A2 03            LDX =3      4 BYTE TRANSFER
1D64  BD E1 1D  MIC    LDA C,X
1D67  95 08            STA X1,X    LOAD EXP/MANT1 WITH C
1D69  CA               DEX
1D6A  10 F8            BPL MIC
1D6C  20 4A 1F         JSR FSUB    T*T-C
1D6F  A2 03            LDX =3      4 BYTE TRANSFER
1D71  BD DD 1D  M2MB   LDA MB,X
1D74  95 04            STA X2,X    LOAD EXP/MANT2 WITH MB
1D76  CA               DEX
1D77  10 F8            BPL M2MB
1D79  20 9D 1F         JSR FDIV    MB/(T*T-C)
1D7C  A2 03            LDX =3
1D7E  BD D9 1D  M2A1   LDA A1,X
1D81  95 04            STA X2,X    LOAD EXP/MANT2 WITH A1
1D83  CA               DEX
1D84  10 F8            BPL M2A1
1D86  20 50 1F         JSR FADD    MB/(T*T-C)+A1
1D89  A2 03            LDX =3      4 BYTE TRANSFER
1D8B  B5 14     M2T    LDA T,X
1D8D  95 04            STA X2,X    LOAD EXP/MANT2 WITH T
1D8F  CA               DEX
1D90  10 F9            BPL M2T
1D92  20 77 1F         JSR FMUL    (MB/(T*T-C)+A1)*T
1D95  A2 03            LDX =3      4 BYTE TRANSFER
1D97  BD E5 1D  M2MHL  LDA MHLF,X
1D9A  95 04            STA X2,X    LOAD EXP/MANT2 WITH MHLF (.5)
1D9C  CA               DEX
1D9D  10 F8            BPL M2MHL
1D9F  20 50 1F         JSR FADD    +.5
1DA2  A2 03            LDX =3      4 BYTE TRANSFER
1DA4  B5 18     LDEXP  LDA SEXP,X
1DA6  95 04            STA X2,X    LOAD EXP/MANT2 WITH ORIGINAL EXPONENT
1DA8  CA               DEX
1DA9  10 F9            BPL LDEXP
1DAB  20 50 1F         JSR FADD    +EXPN
1DAE  A2 03            LDX =3      4 BYTE TRANSFER
1DB0  BD D5 1D  MLE2   LDA LE2,X
1DB3  95 04            STA X2,X    LOAD EXP/MANT2 WITH LN(2)
1DB5  CA               DEX
1DB6  10 F8            BPL MLE2
1DB8  20 77 1F         JSR FMUL    *LN(2)
1DBB  60               RTS         RETURN RESULT IN MANT/EXP1
                *
```

```
                  *     COMMON LOG OF MANT/EXP1 RESULT IN MANT/EXP1
                  *
1DBC  20 00 1D  LOG10  JSR LOG      COMPUTE NATURAL LOG
1DBF  A2 03            LDX =3
1DC1  BD CD 1D  L10    LDA LN10,X
1DC4  95 04            STA X2,X     LOAD EXP/MANT2 WITH 1/LN(10)
1DC6  CA               DEX
1DC7  10 F8            BPL L10
1DC9  20 77 1F         JSR FMUL     LOG10(X)=LN(X)/LN(10)
1DCC  60               RTS
                  *
1DCD  7E 6F     LN10   DCM 0.4342945
      2D ED
1DD1  80 5A     R22    DCM 1.4142136   SQRT(2)
      02 7A
1DD5  7F 58     LE2    DCM 0.69314718  LOG BASE E OF 2
      B9 0C
1DD9  80 52     A1     DCM 1.2920074
      80 40
1DDD  81 AB     MB     DCM -2.6398577
      86 49
1DE1  80 6A     C      DCM 1.6567626
      08 66
1DE5  7F 40     MHLF   DCM 0.5
      00 00
                  *
1E00                   ORG $1E00    STARTING LOCATION FOR EXP
                  *
                  *     EXP OF MANT/EXP1 RESULT IN MANT/EXP1
                  *
1E00  A2 03     EXP    LDX =3       4 BYTE TRANSFER
1E02  BD D8 1E         LDA L2E,X
1E05  95 04            STA X2,X     LOAD EXP/MANT2 WITH LOG BASE 2 OF E
1E07  CA               DEX
1E08  10 F8            BPL EXP+2
1E0A  20 77 1F         JSR FMUL     LOG2(3)*X
1E0D  A2 03            LDX =3       4 BYTE TRANSFER
1E0F  B5 08     FSA    LDA X1,X
1E11  95 10            STA Z,X      STORE EXP/MANT1 IN Z
1E13  CA               DEX
1E14  10 F9            BPL FSA      SAVE Z=LN(2)*X
1E16  20 E8 1F         JSR FIX      CONVERT CONTENTS OF EXP/MANT1 TO AN INTEGER
1E19  A5 0A            LDA M1+1
1E1B  85 1C            STA INT      SAVE RESULT AS INT
1E1D  38               SEC          SET CARRY FOR SUBTRACTION
1E1E  E9 7C            SBC =124     INT-124
1E20  A5 09            LDA M1
1E22  E9 00            SBC =0
1E24  10 15            BPL OVFLW    OVERFLOW INT>=124
1E26  18               CLC          CLEAR CARRY FOR ADD
1E27  A5 0A            LDA M1+1
1E29  69 78            ADC =120     ADD 120 TO INT
1E2B  A5 09            LDA M1
1E2D  69 00            ADC =0
1E2F  10 0B            BPL CONTIN   IF RESULT POSITIVE CONTINUE
1E31  A9 00            LDA =0       INT<-120 SET RESULT TO ZERO AND RETURN
1E33  A2 03            LDX =3       4 BYTE MOVE
1E35  95 08     ZERO   STA X1,X     SET EXP/MANT1 TO ZERO
1E37  CA               DEX
1E38  10 FB            BPL ZERO
1E3A  60               RTS          RETURN
                  *
```

```
1E3B  00        OVFLW  BRK         OVERFLOW
                *
1E3C  20 2C 1F  CONTIN JSR FLOAT   FLOAT INT
1E3F  A2 03            LDX =3
1E41  B5 10     ENTD   LDA Z,X
1E43  95 04            STA X2,X    LOAD EXP/MANT2 WITH Z
1E45  CA               DEX
1E46  10 F9            BPL ENTD
1E48  20 4A 1F         JSR FSUB    Z*Z-FLOAT(INT)
1E4B  A2 03            LDX =3      4 BYTE MOVE
1E4D  B5 08     ZSAV   LDA X1,X
1E4F  95 10            STA Z,X     SAVE EXP/MANT1 IN Z
1E51  95 04            STA X2,X    COPY EXP/MANT1 TO EXP/MANT2
1E53  CA               DEX
1E54  10 F7            BPL ZSAV
1E56  20 77 1F         JSR FMUL    Z*Z
1E59  A2 03            LDX =3      4 BYTE MOVE
1E5B  BD DC 1E  LA2    LDA A2,X
1E5E  95 04            STA X2,X    LOAD EXP/MANT2 WITH A2
1E60  B5 08            LDA X1,X
1E62  95 18            STA SEXP,X  SAVE EXP/MANT1 AS SEXP
1E64  CA               DEX
1E65  10 F4            BPL LA2
1E67  20 50 1F         JSR FADD    Z*Z+A2
1E6A  A2 03            LDX =3      4 BYTE MOVE
1E6C  BD E0 1E  LB2    LDA B2,X
1E6F  95 04            STA X2,X    LOAD EXP/MANT2 WITH B2
1E71  CA               DEX
1E72  10 F8            BPL LB2
1E74  20 9D 1F         JSR FDIV    T=B/(Z*Z+A2)
1E77  A2 03            LDX =3      4 BYTE MOVE
1E79  B5 08     DLOAD  LDA X1,X
1E7B  95 14            STA T,X     SAVE EXP/MANT1 AS T
1E7D  BD E4 1E         LDA C2,X
1E80  95 08            STA X1,X    LOAD EXP/MANT1 WITH C2
1E82  B5 18            LDA SEXP,X
1E84  95 04            STA X2,X    LOAD EXP/MANT2 WITH SEXP
1E86  CA               DEX
1E87  10 F0            BPL DLOAD
1E89  20 77 1F         JSR FMUL    Z*Z*C2
1E8C  20 1C 1F         JSR SWAP    MOVE EXP/MANT1 TO EXP/MANT2
1E8F  A2 03            LDX =3      4 BYTE TRANSFER
1E91  B5 14     LTMP   LDA T,X
1E93  95 08            STA X1,X    LOAD EXP/MANT1 WITH T
1E95  CA               DEX
1E96  10 F9            BPL LTMP
1E98  20 4A 1F         JSR FSUB    C2*Z*Z-B2/(Z*Z+A2)
1E9B  A2 03            LDX =3      4 BYTE TRANSFER
1E9D  BD E8 1E  LDD    LDA D,X
1EA0  95 04            STA X2,X    LOAD EXP/MANT2 WITH D
1EA2  CA               DEX
1EA3  10 F8            BPL LDD
1EA5  20 50 1F         JSR FADD    D+C2*Z*Z-B2/(Z*Z+A2)
1EA8  20 1C 1F         JSR SWAP    MOVE EXP/MANT1 TO EXP/MANT2
1EAB  A2 03            LDX =3      4 BYTE TRANSFER
1EAD  B5 10     LFA    LDA Z,X
1EAF  95 08            STA X1,X    LOAD EXP/MANT1 WITH Z
1EB1  CA               DEX
1EB2  10 F9            BPL LFA
1EB4  20 4A 1F         JSR FSUB    -Z+D+C2*Z*Z-B2/(Z*Z+A2)
1EB7  A2 03            LDX =3      4 BYTE TRANSFER
1EB9  B5 10     LF3    LDA Z,X
```

```
1EBB  95 04            STA X2,X    LOAD EXP/MANT2 WITH Z
1EBD  CA               DEX
1EBE  10 F9            BPL LF3
1EC0  20 9D 1F         JSR FDIV    Z/(**** )
1EC3  A2 03            LDX =3      4 BYTE TRANSFER
1EC5  BD E5 1D  LD12   LDA MHLF,X
1EC8  95 04            STA X2,X    LOAD EXP/MANT2 WITH .5
1ECA  CA               DEX
1ECB  10 F8            BPL LD12
1ECD  20 50 1F         JSR FADD    +Z/(***)+.5
1ED0  38               SEC         ADD INT TO EXPONENT WITH CARRY SET
1ED1  A5 1C            LDA INT     TO MULTIPLY BY
1ED3  65 08            ADC X1      2**(INT+1)
1ED5  85 08            STA X1      RETURN RESULT TO EXPONENT
1ED7  60               RTS         RETURN ANS=(.5+Z/(-Z+D+C2*Z*Z-B2/(Z*Z+A2))*2**(INT+1)
1ED8  80 5C     L2E    DCM 1.4426950409   LOG BASE 2 OF E
      55 1E
1EDC  86 57     A2     DCM 87.417497202
      6A E1
1EE0  89 4D     B2     DCM 617.9722695
      3F 1D
1EE4  7B 46     C2     DCM .03465735903
      FA 70
1EE8  83 4F     D      DCM 9.9545957821
      A3 03
                *
                *
                *     BASIC FLOATING POINT ROUTINES
                *
1F00                   ORG $1F00   START OF BASIC FLOATING POINT ROUTINES
1F00  18        ADD    CLC         CLEAR CARRY
1F01  A2 02            LDX =$02    INDEX FOR 3-BYTE ADD
1F03  B5 09     ADD1   LDA M1,X
1F05  75 05            ADC M2,X    ADD A BYTE OF MANT2 TO MANT1
1F07  95 09            STA M1,X
1F09  CA               DEX         ADVANCE INDEX TO NEXT MORE SIGNIF.BYTE
1F0A  10 F7            BPL ADD1    LOOP UNTIL DONE.
1F0C  60               RTS         RETURN
1F0D  06 03     MD1    ASL SIGN    CLEAR LSB OF SIGN
1F0F  20 12 1F         JSR ABSWAP  ABS VAL OF MANT1, THEN SWAP MANT2
1F12  24 09     ABSWAP BIT M1      MANT1 NEG?
1F14  10 05            BPL ABSWP1  NO,SWAP WITH MANT2 AND RETURN
1F16  20 8F 1F         JSR FCOMPL  YES, COMPLIMENT IT.
1F19  E6 03            INC SIGN    INCR SIGN, COMPLEMENTING LSB
1F1B  38        ABSWP1 SEC         SET CARRY FOR RETURN TO MUL/DIV
                *
                *     SWAP EXP/MANT1 WITH EXP/MANT2
                *
1F1C  A2 04     SWAP   LDX =$04    INDEX FOR 4-BYTE SWAP.
1F1E  94 0B     SWAP1  STY E-1,X
1F20  B5 07            LDA X1-1,X  SWAP A BYTE OF EXP/MANT1 WITH
1F22  B4 03            LDY X2-1,X  EXP/MANT2 AND LEAVEA COPY OF
1F24  94 07            STY X1-1,X  MANT1 IN E(3BYTES). E+3 USED.
1F26  95 03            STA X2-1,X
1F28  CA               DEX         ADVANCE INDEX TO NEXT BYTE
1F29  D0 F3            BNE SWAP1   LOOP UNTIL DONE.
1F2B  60               RTS
                *
                *
                *
                *     CONVERT 16 BIT INTEGER IN M1(HIGH) AND M1+1(LOW) TO F.P.
                *     RESULT IN EXP/MANT1.  EXP/MANT2 UNEFFECTED
```

```
                *
                *
1F2C  A9 8E     FLOAT  LDA =$8E
1F2E  85 08            STA X1      SET EXPN TO 14 DEC
1F30  A9 00            LDA =0      CLEAR LOW ORDER BYTE
1F32  85 0B            STA M1+2
1F34  F0 08            BEQ NORM    NORMALIZE RESULT
1F36  C6 08     NORM1  DEC X1      DECREMENT EXP1
1F38  06 0B            ASL M1+2
1F3A  26 0A            ROL M1+1    SHIFT MANT1 (3 BYTES) LEFT
1F3C  26 09            ROL M1
1F3E  A5 09     NORM   LDA M1      HIGH ORDER MANT1 BYTE
1F40  0A               ASL         UPPER TWO BITS UNEQUAL?
1F41  45 09            EOR M1
1F43  30 04            BMI RTS1    YES,RETURN WITH MANT1 NORMALIZED
1F45  A5 08            LDA X1      EXP1 ZERO?
1F47  D0 ED            BNE NORM1   NO, CONTINUE NORMALIZING
1F49  60        RTS1   RTS         RETURN
                *
                *
                *     EXP/MANT2-EXP/MANT1 RESULT IN EXP/MANT1
                *
1F4A  20 8F 1F  FSUB   JSR FCOMPL  CMPL MANT1 CLEARS CARRY UNLESS ZERO
1F4D  20 5D 1F  SWPALG JSR ALGNSW  RIGHT SHIFT MANT1 OR SWAP WITH MANT2 ON CARRY
                *
                *     ADD EXP/MANT1 AND EXP/MANT2 RESULT IN EXP/MANT1
                *
1F50  A5 04     FADD   LDA X2
1F52  C5 08            CMP X1      COMPARE EXP1 WITH EXP2
1F54  D0 F7            BNE SWPALG  IF UNEQUAL, SWAP ADDENDS OR ALIGN MANTISSAS
1F56  20 00 1F         JSR ADD     ADD ALIGNED MANTISSAS
1F59  50 E3     ADDEND BVC NORM    NO OVERFLOW, NORMALIZE RESULTS
1F5B  70 05            BVS RTLOG   OV: SHIFT MANT1 RIGHT. NOTE CARRY IS CORRECT SIGN
1F5D  90 BD     ALGNSW BCC SWAP    SWAP IF CARRY CLEAR, ELSE SHIFT RIGHT ARITH.
1F5F  A5 09     RTAR   LDA M1      SIGN OF MANT1 INTO CARRY FOR
1F61  0A               ASL         RIGHT ARITH SHIFT
1F62  E6 08     RTLOG  INC X1      INCR EXP1 TO COMPENSATE FOR RT SHIFT
1F64  F0 7E            BEQ OVFL    EXP1 OUT OF RANGE.
1F66  A2 FA     RTLOG1 LDX =$FA    INDEX FOR 6 BYTE RIGHT SHIFT
1F68  A9 80     ROR1   LDA =$80
1F6A  B0 01            BCS ROR2
1F6C  0A               ASL
1F6D  56 0F     ROR2   LSR E+3,X   SIMULATE ROR E+3,X
1F6F  15 0F            ORA E+3,X
1F71  95 0F            STA E+3,X
1F73  E8               INX         NEXT BYTE OF SHIFT
1F74  D0 F2            BNE ROR1    LOOP UNTIL DONE
1F76  60               RTS         RETURN
                *
                *
                *     EXP/MANT1 X EXP/MANT2 RESULT IN EXP/MANT1
                *
1F77  20 0D 1F  FMUL   JSR MD1     ABS. VAL OF MANT1, MANT2
1F7A  65 08            ADC X1      ADD EXP1 TO EXP2 FOR PRODUCT EXPONENT
1F7C  20 CD 1F         JSR MD2     CHECK PRODUCT EXP AND PREPARE FOR MUL
1F7F  18               CLC         CLEAR CARRY
1F80  20 66 1F  MUL1   JSR RTLOG1  MANT1 AND E RIGHT.(PRODUCT AND MPLIER)
1F83  90 03            BCC MUL2    IF CARRY CLEAR, SKIP PARTIAL PRODUCT
1F85  20 00 1F         JSR ADD     ADD MULTIPLICAN TO PRODUCT
1F88  88        MUL2   DEY         NEXT MUL ITERATION
1F89  10 F5            BPL MUL1    LOOP UNTIL DONE
1F8B  46 03     MDEND  LSR SIGN    TEST SIGN (EVEN/ODD)
```

```
1F8D  90 AF     NORMX  BCC NORM    IF EXEN, NORMALIZE PRODUCT, ELSE COMPLEMENT
1F8F  38        FCOMPL SEC         SET CARRY FOR SUBTRACT
1F90  A2 03            LDX =$03    INDEX FOR 3 BYTE SUBTRACTION
1F92  A9 00     COMPL1 LDA =$00    CLEAR A
1F94  F5 08            SBC X1,X    SUBTRACT BYTE OF EXP1
1F96  95 08            STA X1,X    RESTORE IT
1F98  CA               DEX         NEXT MORE SIGNIFICANT BYTE
1F99  D0 F7            BNE COMPL1  LOOP UNTIL DONE
1F9B  F0 BC            BEQ ADDEND  NORMALIZE (OR SHIFT RIGHT IF OVERFLOW)
                *
                *
                *     EXP/MANT2 / EXP/MANT1 RESULT IN EXP/MANT1
                *
1F9D  20 0D 1F  FDIV   JSR MD1     TAKE ABS VAL OF MANT1, MANT2
1FA0  E5 08            SBC X1      SUBTRACT EXP1 FROM EXP2
1FA2  20 CD 1F         JSR MD2     SAVE AS QUOTIENT EXP
1FA5  38        DIV1   SEC         SET CARRY FOR SUBTRACT
1FA6  A2 02            LDX =$02    INDEX FOR 3-BYTE INSTRUCTION
1FA8  B5 05     DIV2   LDA M2,X
1FAA  F5 0C            SBC E,X     SUBTRACT A BYTE OF E FROM MANT2
1FAC  48               PHA         SAVE ON STACK
1FAD  CA               DEX         NEXT MORE SIGNIF BYTE
1FAE  10 F8            BPL DIV2    LOOP UNTIL DONE
1FB0  A2 FD            LDX =$FD    INDEX FOR 3-BYTE CONDITIONAL MOVE
1FB2  68        DIV3   PLA         PULL A BYTE OF DIFFERENCE OFF STACK
1FB3  90 02            BCC DIV4    IF MANT2<E THEN DONT RESTORE MANT2
1FB5  95 08            STA M2+3,X
1FB7  E8        DIV4   INX         NEXT LESS SIGNIF BYTE
1FB8  D0 F8            BNE DIV3    LOOP UNTIL DONE
1FBA  26 0B            ROL M1+2
1FBC  26 0A            ROL M1+1    ROLL QUOTIENT LEFT, CARRY INTO LSB
1FBE  26 09            ROL M1
1FC0  06 07            ASL M2+2
1FC2  26 06            ROL M2+1    SHIFT DIVIDEND LEFT
1FC4  26 05            ROL M2
1FC6  B0 1C            BCS OVFL    OVERFLOW IS DUE TO UNNORMALIZED DIVISOR
1FC8  88               DEY         NEXT DIVIDE ITERATION
1FC9  D0 DA            BNE DIV1    LOOP UNTIL DONE 23 ITERATIONS
1FCB  F0 BE            BEQ MDEND   NORMALIZE QUOTIENT AND CORRECT SIGN
1FCD  86 0B     MD2    STX M1+2
1FCF  86 0A            STX M1+1    CLR MANT1 (3 BYTES) FOR MUL/DIV
1FD1  86 09            STX M1
1FD3  B0 0D            BCS OVCHK   IF EXP CALC SET CARRY, CHECK FOR OVFL
1FD5  30 04            BMI MD3     IF NEG NO UNDERFLOW
1FD7  68               PLA         POP ONE
1FD8  68               PLA         RETURN LEVEL
1FD9  90 B2            BCC NORMX   CLEAR X1 AND RETURN
1FDB  49 80     MD3    EOR =$80    COMPLIMENT SIGN BIT OF EXP
1FDD  85 08            STA X1      STORE IT
1FDF  A0 17            LDY =$17    COUNT FOR 24 MUL OR 23 DIV ITERATIONS
1FE1  60               RTS         RETURN
1FE2  10 F7     OVCHK  BPL MD3     IF POS EXP THEN NO OVERFLOW
1FE4  00        OVFL   BRK
                *
                *
                *     CONVERT EXP/MANT1 TO INTEGER IN M1 (HIGH) AND M1+1(LOW)
                *      EXP/MANT2 UNEFFECTED
                *
1FE5  20 5F 1F         JSR RTAR    SHIFT MANT1 RT AND INCREMENT EXPNT
1FE8  A5 08     FIX    LDA X1      CHECK EXPONENT
1FEA  C9 8E            CMP =$8E    IS EXPONENT 14?
1FEC  D0 F7            BNE FIX-3   NO, SHIFT
```

```
1FEE  60        RTRN   RTS        RETURN
                       END
```

****

Dr. Dobb's Journal, November/December 1976, page 57.

ERRATA FOR RANKIN'S 6502
FLOATING POINT ROUTINES

Sept. 22, 1976

Dear Jim,

Subsequent to the publication of "Floating Point Routines for the 6502" (Vol.1, No.7) an error which I made in the LOG routine came to light which causes improper results if the argument is less than 1. The following changes will correct the error.

```
1.  After:            CONT JSR SWAP (1D07)
    Add:    A2 00          LDX =0    LOAD X FOR HIGH BYTE OF EXPONENT

2.  After:                 STA M1+1 (1D12)
    Delete:                LDA =0
                           STA M1
    Add:    10 01          BPL *+3   IS EXPONENT NEGATIVE
            CA             DEX       YES, SET X TO $FF
            86 09          STX M1    SET UPPER BYTE OF EXPONENT
```

3. Changes 1 and 2 shift the code by 3 bytes so add 3 to the addresses of the constants LN10 through MHLF whenever they are referenced. For example the address of LN10 changes from 1DCD to 1DD0. Note also that the entry point for LOG10 becomes 1DBF. The routines stays within the page and hence the following routines (EXP etc.) are not affected.

Yours truly,

Roy Rankin
Dep. of Mech. Eng.
Stanford University

```
+-----------------------------------------------------------------------------
|  TOPIC -- Apple II -- IA Floating point article
+-----------------------------------------------------------------------------
```

Interface Age, November 1976, pages 103-111.

Floating Point Routines for the 6502*

by Roy Rankin
Department of Mechanical Engineering, Stanford University

and Steve Wozniak
Apple Computer Company

*First appeared in Dr. DOBB's Journal of Computer Calisthenics & Orthodontia, Box 310, Menlo Park, CA 94025

The following floating point routines represent a joint effort between Steve Wozniak who wrote the basic floating point routines of FADD, FSUB, FMUL, FDIV and their support routines and myself, Roy Rankin, who added FIX, FLOAT, LOG, LOG10, and EXP. The basic floating point routines are failry Machine dependent, but the transcendental programs should be very easy to transport from one machine to another. The routines consist of the following math functions

```
     * LOG          Natural log
     * LOG10        Base 10 log
     * EXP          Exponential
     * FADD         Floating add
     * FSUB         Floating subtraction
     * FMUL         Floating multiplication
     * FDIV         Floating division
     * FIX          Convert floating to fixed
     * FLOAT        Convert fixed to floating
```

Two additional routines exchange the contents of exp/mant1 with exp/mant2 and compliments exp/mant1. These routines are

```
SWAP          Exchange the contents of exp/mant 1 with
              exp/mant 2
FCOMPL        Compliment exp/mant 1
```

Floating point numbers are represented by 4 bytes as shown in the following

```
 +- SIGN BIT     +- SIGN BIT
 |  0 = +        |  0 = +
 |  1 = -        |  1 = -
 v               v
|S|             |S| +- PRESUMED DECIMAL POINT
|B|             |B| v
|_|_ _ _ _ _ _ _|_|_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
```

```
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|7 6 5 4 3 2 1 0|7 6.5 4 3 2 1 0|7 6 5 4 3 2 1 0|7 6 5 4 3 2 1 0|
|               |               |               |               |
|    BYTE N     |   BYTE N+1    |  BYTE N+2     |   BYTE N+3    |
|               |               |               |               |
|               | MOST SIG BYTE |               | LEAST SIG BYTE|
|               | MANTISSA      |               | MANTISSA      |
|               |               |               |               |
|<- EXPONENT  ->|<---          THREE BYTE MANTISSA          --->|
|                        (TWOS COMPLEMENT REPRESENTATION)       |
|<----             FOUR-BYTE FLOATING POINT OPERAND        ---->|
```

The exponent byte is a binary scaling factor for the Mantissa. The exponent is a standard two's complement representation except that the sign bit is complemented and runs from +128 to +127. For example:

```
     $00 is -128
     $01 is -127
         *
         *
     $7F is -1
     $80 is 0
     $81 is -1
         *
         *
     $FF is 127
```

The mantissa is standard two's complement representation with the sign bit in the most significant bit of the high order byte. The assumed decimal point is between bits 6 and 7 of the most significant byte. Thus the normalized mantissa ranges in absolute value from 1 to 2. Except when the exponent has a value of +128 the mantissa is normalized to retain maximum precision. The mantissa is normalized if the upper two bits of the high-order mantissa byte are unequal. Thus a normalized mantissa is of the following form:

```
     01.xxxxxx  positive mantissa (high byte)
     10.xxxxxx  negative mantissa (high byte)
        Assumed binary point
```

Some sample floating point numbers in hex

```
     83 50 00 00     10.
     80 40 00 00     1.
     7C 66 66 66     .1
     00 00 00 00     0.
     FC 99 99 9A     -.1
     7F 80 00 00     -1.
     83 B0 00 00     -10.
```

The routines are all entered using a JSR instruction. Base page locations $004-$007 are referred to as exp/mant2 while $0008-000b are referred to as exp/

mant1 and act as floating point registers. On entry to the subroutines these registers contain the numbers to be operated upon and contain the result on return, The function of the registers is given before each entry point in the source listing. There are three error traps which will cause a software interrupts. ERROT (1D06) is encountered if the argument in the log routine is less than or equal to zero. OVFLW (1E3B) will be executed if the argument of EXP is too large. Overflow detected by the basic floating point routines will cause OVFL (1FE4) to be executed. The routines do not give underflow errors, but set the number to zero if underflow occurs.

Readers of Dr. Dobbs's journal should note that when these routines were published in that journal the math function LOG contained an error which prevented the correct result from being given if the argument was less than 1. This error has been correted in the present listing and marked with "MOD 9/76."

```
                  *          SEPTEMBER 11, 1976
                  *     BASIC FLOATING POINT ROUTINES
                  *       FOR 6502 MICROPROCESSOR
                  *       BY R. RANKIN AND S. WOZNIAK
                  *
                  *     CONSISTING OF:
                  *        NATURAL LOG
                  *        COMMON LOG
                  *        EXPONENTIAL (E**X)
                  *        FLOAT      FIX
                  *        FADD       FSUB
                  *        FMUL       FDIV
                  *
                  *
                  *     FLOATING POINT REPRESENTATION (4-BYTES)
                  *                   EXPONENT BYTE 1
                  *                   MANTISSA BYTES 2-4
                  *
                  *     MANTISSA:    TWO'S COMPLIMENT REPRESENTATION WITH SIGN IN
                  *       MSB OF HIGH-ORDER BYTE.  MANTISSA IS NORMALIZED WITH AN
                  *       ASSUMED DECIMAL POINT BETWEEN BITS 5 AND 6 OF THE HIGH-ORDER
                  *       BYTE.  THUS THE MANTISSA IS IN THE RANGE 1. TO 2. EXCEPT
                  *       WHEN THE NUMBER IS LESS THAN 2**(-128).
                  *
                  *     EXPONENT:    THE EXPONENT REPRESENTS POWERS OF TWO.  THE
                  *       REPRESENTATION IS 2'S COMPLIMENT EXCEPT THAT THE SIGN
                  *       BIT (BIT 7) IS COMPLIMENTED.  THIS ALLOWS DIRECT COMPARISON
                  *       OF EXPONENTS FOR SIZE SINCE THEY ARE STORED IN INCREASING
                  *       NUMERICAL SEQUENCE RANGING FROM $00 (-128) TO $FF (+127)
                  *       ($ MEANS NUMBER IS HEXADECIMAL).
                  *
                  *     REPRESENTATION OF DECIMAL NUMBERS:    THE PRESENT FLOATING
                  *       POINT REPRESENTATION ALLOWS DECIMAL NUMBERS IN THE
APPROXIMATE
                  *       RANGE OF 10**(-38) THROUGH 10**(38) WITH 6 TO 7 SIGNIFICANT
                  *       DIGITS.
                  *
                  *
     0003                ORG 3        SET BASE PAGE ADRESSES
```

```
0003  EA          SIGN   NOP
0004  EA          X2     NOP          EXPONENT 2
0005  00 00 00    M2     BSS 3        MANTISSA 2
0008  EA          X1     NOP          EXPONENT 1
0009  00 00 00    M1     BSS 3        MANTISSA 1
000C              E      BSS 4        SCRATCH
0010              Z      BSS 4
0014              T      BSS 4
0018              SEXP   BSS 4
001C  00          INT    BSS 1
                  *
1D00                     ORG $1D00    STARTING LOCATION FOR LOG
                  *
                  *     NATURAL LOG OF MANT/EXP1 WITH RESULT IN MANT/EXP1
                  *
1D00  A5 09       LOG    LDA M1
1D02  F0 02              BEQ ERROR
1D04  10 01              BPL CONT     IF ARG>0 OK
1D06  00          ERROR  BRK          ERROR ARG<=0
                  *
1D07  20 1C 1F    CONT   JSR SWAP     MOVE ARG TO EXP/MANT2
1D0A  A2 00              LDX =0       MOD 9/76: LOAD X FOR LATER
1D0C  A5 04              LDA X2       HOLD EXPONENT
1D0E  A0 80              LDY =$80
1D10  84 04              STY X2       SET EXPONENT 2 TO 0 ($80)
1D12  49 80              EOR =$80     COMPLIMENT SIGN BIT OF ORIGINAL EXPONENT
1D14  85 0A              STA M1+1     SET EXPONENT INTO MANTISSA 1 FOR FLOAT
1D16  10 01              BPL *+3      MOD 9/76: IS EXPONENT ZERO?
1D18  CA                 DEX          MOD 9/76: YES SET X TO $FF
1D19  86 09              STX M1       MOD 9/76: SET UPPER BYTE OF EXPONENT
1D1B  20 2C 1F           JSR FLOAT    CONVERT TO FLOATING POINT
1D1E  A2 03              LDX =3       4 BYTE TRANSFERS
1D20  B5 04       SEXP1  LDA X2,X
1D22  95 10              STA Z,X      COPY MANTISSA TO Z
1D24  B5 08              LDA X1,X
1D26  95 18              STA SEXP,X   SAVE EXPONENT IN SEXP
1D28  BD D4 1D           LDA R22,X    LOAD EXP/MANT1 WITH SQRT(2)
1D2B  95 08              STA X1,X
1D2D  CA                 DEX
1D2E  10 F0              BPL SEXP1
1D30  20 4A 1F           JSR FSUB     Z-SQRT(2)
1D33  A2 03              LDX =3       4 BYTE TRANSFER
1D35  B5 08       SAVET  LDA X1,X     SAVE EXP/MANT1 AS T
1D37  95 14              STA T,X
1D39  B5 10              LDA Z,X      LOAD EXP/MANT1 WITH Z
1D3B  95 08              STA X1,X
1D3D  BD D4 1D           LDA R22,X    LOAD EXP/MANT2 WITH SQRT(2)
1D40  95 04              STA X2,X
1D42  CA                 DEX
1D43  10 F0              BPL SAVET
1D45  20 50 1F           JSR FADD     Z+SQRT(2)
1D48  A2 03              LDX =3       4 BYTE TRANSFER
1D4A  B5 14       TM2    LDA T,X
1D4C  95 04              STA X2,X     LOAD T INTO EXP/MANT2
1D4E  CA                 DEX
1D4F  10 F9              BPL TM2
1D51  20 9D 1F           JSR FDIV     T=(Z-SQRT(2))/(Z+SQRT(2))
1D54  A2 03              LDX =3       4 BYTE TRANSFER
1D56  B5 08       MIT    LDA X1,X
1D58  95 14              STA T,X      COPY EXP/MANT1 TO T AND
1D5A  95 04              STA X2,X     LOAD EXP/MANT2 WITH T
1D5C  CA                 DEX
```

```
1D5D  10 F7              BPL MIT
1D5F  20 77 1F           JSR FMUL     T*T
1D62  20 1C 1F           JSR SWAP     MOVE T*T TO EXP/MANT2
1D65  A2 03              LDX =3       4 BYTE TRANSFER
1D67  BD E4 1D   MIC     LDA C,X
1D6A  95 08              STA X1,X     LOAD EXP/MANT1 WITH C
1D6C  CA                 DEX
1D6D  10 F8              BPL MIC
1D6F  20 4A 1F           JSR FSUB     T*T-C
1D72  A2 03              LDX =3       4 BYTE TRANSFER
1D74  BD E0 1D   M2MB    LDA MB,X
1D77  95 04              STA X2,X     LOAD EXP/MANT2 WITH MB
1D79  CA                 DEX
1D7A  10 F8              BPL M2MB
1D7C  20 9D 1F           JSR FDIV     MB/(T*T-C)
1D7F  A2 03              LDX =3
1D81  BD DC 1D   M2A1    LDA A1,X
1D84  95 04              STA X2,X     LOAD EXP/MANT2 WITH A1
1D86  CA                 DEX
1D87  10 F8              BPL M2A1
1D89  20 50 1F           JSR FADD     MB/(T*T-C)+A1
1D8C  A2 03              LDX =3       4 BYTE TRANSFER
1D8E  B5 14      M2T     LDA T,X
1D90  95 04              STA X2,X     LOAD EXP/MANT2 WITH T
1D92  CA                 DEX
1D93  10 F9              BPL M2T
1D95  20 77 1F           JSR FMUL     (MB/(T*T-C)+A1)*T
1D98  A2 03              LDX =3       4 BYTE TRANSFER
1D9A  BD E8 1D   M2MHL   LDA MHLF,X
1D9D  95 04              STA X2,X     LOAD EXP/MANT2 WITH MHLF (.5)
1D9F  CA                 DEX
1DA0  10 F8              BPL M2MHL
1DA2  20 50 1F           JSR FADD     +.5
1DA5  A2 03              LDX =3       4 BYTE TRANSFER
1DA7  B5 18      LDEXP   LDA SEXP,X
1DA9  95 04              STA X2,X     LOAD EXP/MANT2 WITH ORIGINAL EXPONENT
1DAB  CA                 DEX
1DAC  10 F9              BPL LDEXP
1DAE  20 50 1F           JSR FADD     +EXPN
1DB1  A2 03              LDX =3       4 BYTE TRANSFER
1DB3  BD D8 1D   MLE2    LDA LE2,X
1DB6  95 04              STA X2,X     LOAD EXP/MANT2 WITH LN(2)
1DB8  CA                 DEX
1DB9  10 F8              BPL MLE2
1DBB  20 77 1F           JSR FMUL     *LN(2)
1DBE  60                 RTS          RETURN RESULT IN MANT/EXP1
                 *
                 *     COMMON LOG OF MANT/EXP1 RESULT IN MANT/EXP1
                 *
1DBF  20 00 1D   LOG10   JSR LOG      COMPUTE NATURAL LOG
1DC2  A2 03              LDX =3
1DC4  BD D0 1D   L10     LDA LN10,X
1DC7  95 04              STA X2,X     LOAD EXP/MANT2 WITH 1/LN(10)
1DC9  CA                 DEX
1DCA  10 F8              BPL L10
1DCC  20 77 1F           JSR FMUL     LOG10(X)=LN(X)/LN(10)
1DCF  60                 RTS
                 *
1DD0  7E 6F      LN10    DCM  0.4342945
      2D ED
1DD4  80 5A      R22     DCM  1.4142136   SQRT(2)
      82 7A
```

```
1DD8  7F 58      LE2    DCM  0.69314718  LOG BASE E OF 2
      B9 0C
1DDC  80 52      A1     DCM  1.2920074
      B0 40
1DE0  81 AB      MB     DCM  -2.6398577
      86 49
1DE4  80 6A      C      DCM  1.6567626
      08 66
1DE8  7F 40      MHLF   DCM  0.5
      00 00
                 *
1E00                    ORG $1E00    STARTING LOCATION FOR EXP
                 *
                 *     EXP OF MANT/EXP1 RESULT IN MANT/EXP1
                 *
1E00  A2 03      EXP    LDX =3       4 BYTE TRANSFER
1E02  BD D8 1E          LDA L2E,X
1E05  95 04             STA X2,X     LOAD EXP/MANT2 WITH LOG BASE 2 OF E
1E07  CA                DEX
1E08  10 F8             BPL EXP+2
1E0A  20 77 1F          JSR FMUL     LOG2(3)*X
1E0D  A2 03             LDX =3       4 BYTE TRANSFER
1E0F  B5 08      FSA    LDA X1,X
1E11  95 10             STA Z,X      STORE EXP/MANT1 IN Z
1E13  CA                DEX
1E14  10 F9             BPL FSA      SAVE Z=LN(2)*X
1E16  20 E8 1F          JSR FIX      CONVERT CONTENTS OF EXP/MANT1 TO AN INTEGER
1E19  A5 0A             LDA M1+1
1E1B  85 1C             STA INT      SAVE RESULT AS INT
1E1D  38                SEC          SET CARRY FOR SUBTRACTION
1E1E  E9 7C             SBC =124     INT-124
1E20  A5 09             LDA M1
1E22  E9 00             SBC =0
1E24  10 15             BPL OVFLW    OVERFLOW INT>=124
1E26  18                CLC          CLEAR CARRY FOR ADD
1E27  A5 0A             LDA M1+1
1E29  69 78             ADC =120     ADD 120 TO INT
1E2B  A5 09             LDA M1
1E2D  69 00             ADC =0
1E2F  10 0B             BPL CONTIN   IF RESULT POSITIVE CONTINUE
1E31  A9 00             LDA =0       INT<-120 SET RESULT TO ZERO AND RETURN
1E33  A2 03             LDX =3       4 BYTE MOVE
1E35  95 08      ZERO   STA X1,X     SET EXP/MANT1 TO ZERO
1E37  CA                DEX
1E38  10 FB             BPL ZERO
1E3A  60                RTS          RETURN
                 *
1E3B  00         OVFLW  BRK          OVERFLOW
                 *
1E3C  20 2C 1F   CONTIN JSR FLOAT    FLOAT INT
1E3F  A2 03             LDX =3
1E41  B5 10      ENTD   LDA Z,X
1E43  95 04             STA X2,X     LOAD EXP/MANT2 WITH Z
1E45  CA                DEX
1E46  10 F9             BPL ENTD
1E48  20 4A 1F          JSR FSUB     Z*Z-FLOAT(INT)
1E4B  A2 03             LDX =3       4 BYTE MOVE
1E4D  B5 08      ZSAV   LDA X1,X
1E4F  95 10             STA Z,X      SAVE EXP/MANT1 IN Z
1E51  95 04             STA X2,X     COPY EXP/MANT1 TO EXP/MANT2
1E53  CA                DEX
1E54  10 F7             BPL ZSAV
```

```
1E56  20 77 1F          JSR FMUL      Z*Z
1E59  A2 03             LDX =3        4 BYTE MOVE
1E5B  BD DC 1E   LA2    LDA A2,X
1E5E  95 04             STA X2,X      LOAD EXP/MANT2 WITH A2
1E60  B5 08             LDA X1,X
1E62  95 18             STA SEXP,X    SAVE EXP/MANT1 AS SEXP
1E64  CA                DEX
1E65  10 F4             BPL LA2
1E67  20 50 1F          JSR FADD      Z*Z+A2
1E6A  A2 03             LDX =3        4 BYTE MOVE
1E6C  BD E0 1E   LB2    LDA B2,X
1E6F  95 04             STA X2,X      LOAD EXP/MANT2 WITH B2
1E71  CA                DEX
1E72  10 F8             BPL LB2
1E74  20 9D 1F          JSR FDIV      T=B/(Z*Z+A2)
1E77  A2 03             LDX =3        4 BYTE MOVE
1E79  B5 08      DLOAD  LDA X1,X
1E7B  95 14             STA T,X       SAVE EXP/MANT1 AS T
1E7D  BD E4 1E          LDA C2,X
1E80  95 08             STA X1,X      LOAD EXP/MANT1 WITH C2
1E82  B5 18             LDA SEXP,X
1E84  95 04             STA X2,X      LOAD EXP/MANT2 WITH SEXP
1E86  CA                DEX
1E87  10 F0             BPL DLOAD
1E89  20 77 1F          JSR FMUL      Z*Z*C2
1E8C  20 1C 1F          JSR SWAP      MOVE EXP/MANT1 TO EXP/MANT2
1E8F  A2 03             LDX =3        4 BYTE TRANSFER
1E91  B5 14      LTMP   LDA T,X
1E93  95 08             STA X1,X      LOAD EXP/MANT1 WITH T
1E95  CA                DEX
1E96  10 F9             BPL LTMP
1E98  20 4A 1F          JSR FSUB      C2*Z*Z-B2/(Z*Z+A2)
1E9B  A2 03             LDX =3        4 BYTE TRANSFER
1E9D  BD E8 1E   LDD    LDA D,X
1EA0  95 04             STA X2,X      LOAD EXP/MANT2 WITH D
1EA2  CA                DEX
1EA3  10 F8             BPL LDD
1EA5  20 50 1F          JSR FADD      D+C2*Z*Z-B2/(Z*Z+A2)
1EA8  20 1C 1F          JSR SWAP      MOVE EXP/MANT1 TO EXP/MANT2
1EAB  A2 03             LDX =3        4 BYTE TRANSFER
1EAD  B5 10      LFA    LDA Z,X
1EAF  95 08             STA X1,X      LOAD EXP/MANT1 WITH Z
1EB1  CA                DEX
1EB2  10 F9             BPL LFA
1EB4  20 4A 1F          JSR FSUB      -Z+D+C2*Z*Z-B2/(Z*Z+A2)
1EB7  A2 03             LDX =3        4 BYTE TRANSFER
1EB9  B5 10      LF3    LDA Z,X
1EBB  95 04             STA X2,X      LOAD EXP/MANT2 WITH Z
1EBD  CA                DEX
1EBE  10 F9             BPL LF3
1EC0  20 9D 1F          JSR FDIV      Z/(**** )
1EC3  A2 03             LDX =3        4 BYTE TRANSFER
1EC5  BD E8 1D   LD12   LDA MHLF,X
1EC8  95 04             STA X2,X      LOAD EXP/MANT2 WITH .5
1ECA  CA                DEX
1ECB  10 F8             BPL LD12
1ECD  20 50 1F          JSR FADD      +Z/(***)+.5
1ED0  38                SEC           ADD INT TO EXPONENT WITH CARRY SET
1ED1  A5 1C             LDA INT       TO MULTIPLY BY
1ED3  65 08             ADC X1        2**(INT+1)
1ED5  85 08             STA X1        RETURN RESULT TO EXPONENT
```

```
     1ED7  60                 RTS          RETURN ANS=(.5+Z/(-Z+D+C2*Z*Z-
B2/(Z*Z+A2))*2**(INT+1)
     1ED8  80 5C       L2E    DCM  1.4426950409   LOG BASE 2 OF E
           55 1E
     1EDC  86 57       A2     DCM  87.417497202
           6A E1
     1EE0  89 4D       B2     DCM  617.9722695
           3F 1D
     1EE4  7B 46       C2     DCM  .03465735903
           4A 70
     1EE8  83 4F       D      DCM  9.9545957821
           A3 03
                       *
                       *
                       *      BASIC FLOATING POINT ROUTINES
                       *
     1F00                     ORG $1F00    START OF BASIC FLOATING POINT ROUTINES
     1F00  18          ADD    CLC          CLEAR CARRY
     1F01  A2 02              LDX =$02     INDEX FOR 3-BYTE ADD
     1F03  B5 09       ADD1   LDA M1,X
     1F05  75 05              ADC M2,X     ADD A BYTE OF MANT2 TO MANT1
     1F07  95 09              STA M1,X
     1F09  CA                 DEX          ADVANCE INDEX TO NEXT MORE SIGNIF.BYTE
     1F0A  10 F7              BPL ADD1     LOOP UNTIL DONE.
     1F0C  60                 RTS          RETURN
     1F0D  06 03       MD1    ASL SIGN     CLEAR LSB OF SIGN
     1F0F  20 12 1F           JSR ABSWAP   ABS VAL OF MANT1, THEN SWAP MANT2
     1F12  24 09       ABSWAP BIT M1       MANT1 NEG?
     1F14  10 05              BPL ABSWP1   NO,SWAP WITH MANT2 AND RETURN
     1F16  20 8F 1F           JSR FCOMPL   YES, COMPLIMENT IT.
     1F19  E6 03              INC SIGN     INCR SIGN, COMPLEMENTING LSB
     1F1B  38          ABSWP1 SEC          SET CARRY FOR RETURN TO MUL/DIV
                       *
                       *      SWAP EXP/MANT1 WITH EXP/MANT2
                       *
     1F1C  A2 04       SWAP   LDX =$04     INDEX FOR 4-BYTE SWAP.
     1F1E  94 0B       SWAP1  STY E-1,X
     1F20  B5 07              LDA X1-1,X   SWAP A BYTE OF EXP/MANT1 WITH
     1F22  B4 03              LDY X2-1,X   EXP/MANT2 AND LEAVEA COPY OF
     1F24  94 07              STY X1-1,X   MANT1 IN E(3BYTES). E+3 USED.
     1F26  95 03              STA X2-1,X
     1F28  CA                 DEX          ADVANCE INDEX TO NEXT BYTE
     1F29  D0 F3              BNE SWAP1    LOOP UNTIL DONE.
     1F2B  60                 RTS
                       *
                       *
                       *
                       *      CONVERT 16 BIT INTEGER IN M1(HIGH) AND M1+1(LOW) TO F.P.
                       *      RESULT IN EXP/MANT1.  EXP/MANT2 UNEFFECTED
                       *
                       *
     1F2C  A9 8E       FLOAT  LDA =$8E
     1F2E  85 08              STA X1       SET EXPN TO 14 DEC
     1F30  A9 00              LDA =0       CLEAR LOW ORDER BYTE
     1F32  85 0B              STA M1+2
     1F34  F0 08              BEQ NORM     NORMALIZE RESULT
     1F36  C6 08       NORM1  DEC X1       DECREMENT EXP1
     1F38  06 0B              ASL M1+2
     1F3A  26 0A              ROL M1+1     SHIFT MANT1 (3 BYTES) LEFT
     1F3C  26 09              ROL M1
     1F3E  A5 09       NORM   LDA M1       HIGH ORDER MANT1 BYTE
     1F40  0A                 ASL          UPPER TWO BITS UNEQUAL?
```

```
1F41  45 09              EOR M1
1F43  30 04              BMI RTS1     YES,RETURN WITH MANT1 NORMALIZED
1F45  A5 08              LDA X1       EXP1 ZERO?
1F47  D0 ED              BNE NORM1    NO, CONTINUE NORMALIZING
1F49  60          RTS1   RTS          RETURN
                  *
                  *
                  *     EXP/MANT2-EXP/MANT1 RESULT IN EXP/MANT1
                  *
1F4A  20 8F 1F    FSUB   JSR FCOMPL   CMPL MANT1 CLEARS CARRY UNLESS ZERO
1F4D  20 5D 1F    SWPALG JSR ALGNSW   RIGHT SHIFT MANT1 OR SWAP WITH MANT2 ON CARRY
                  *
                  *     ADD EXP/MANT1 AND EXP/MANT2 RESULT IN EXP/MANT1
                  *
1F50  A5 04       FADD   LDA X2
1F52  C5 08              CMP X1       COMPARE EXP1 WITH EXP2
1F54  D0 F7              BNE SWPALG   IF UNEQUAL, SWAP ADDENDS OR ALIGN MANTISSAS
1F56  20 00 1F           JSR ADD      ADD ALIGNED MANTISSAS
1F59  50 E3       ADDEND BVC NORM     NO OVERFLOW, NORMALIZE RESULTS
1F5B  70 05              BVS RTLOG    OV: SHIFT MANT1 RIGHT. NOTE CARRY IS CORRECT
SIGN
1F5D  90 BD       ALGNSW BCC SWAP     SWAP IF CARRY CLEAR, ELSE SHIFT RIGHT ARITH.
1F5F  A5 09       RTAR   LDA M1       SIGN OF MANT1 INTO CARRY FOR
1F61  0A                 ASL          RIGHT ARITH SHIFT
1F62  E6 08       RTLOG  INC X1       INCR EXP1 TO COMPENSATE FOR RT SHIFT
1F64  F0 7E              BEQ OVFL     EXP1 OUT OF RANGE.
1F66  A2 FA       RTLOG1 LDX =$FA     INDEX FOR 6 BYTE RIGHT SHIFT
1F68  A9 80       ROR1   LDA =$80
1F6A  B0 01              BCS ROR2
1F6C  0A                 ASL
1F6D  56 0F       ROR2   LSR E+3,X    SIMULATE ROR E+3,X
1F6F  15 0F              ORA E+3,X
1F71  95 0F              STA E+3,X
1F73  E8                 INX          NEXT BYTE OF SHIFT
1F74  D0 F2              BNE ROR1     LOOP UNTIL DONE
1F76  60                 RTS          RETURN
                  *
                  *
                  *     EXP/MANT1 X EXP/MANT2 RESULT IN EXP/MANT1
                  *
1F77  20 0D 1F    FMUL   JSR MD1      ABS. VAL OF MANT1, MANT2
1F7A  65 08              ADC X1       ADD EXP1 TO EXP2 FOR PRODUCT EXPONENT
1F7C  20 CD 1F           JSR MD2      CHECK PRODUCT EXP AND PREPARE FOR MUL
1F7F  18                 CLC          CLEAR CARRY
1F80  20 66 1F    MUL1   JSR RTLOG1   MANT1 AND E RIGHT.(PRODUCT AND MPLIER)
1F83  90 03              BCC MUL2     IF CARRY CLEAR, SKIP PARTIAL PRODUCT
1F85  20 00 1F           JSR ADD      ADD MULTIPLICAN TO PRODUCT
1F88  88          MUL2   DEY          NEXT MUL ITERATION
1F89  10 F5              BPL MUL1     LOOP UNTIL DONE
1F8B  46 03       MDEND  LSR SIGN     TEST SIGN (EVEN/ODD)
1F8D  90 AF       NORMX  BCC NORM     IF EXEN, NORMALIZE PRODUCT, ELSE COMPLEMENT
1F8F  38          FCOMPL SEC          SET CARRY FOR SUBTRACT
1F90  A2 03              LDX =$03     INDEX FOR 3 BYTE SUBTRACTION
1F92  A9 00       COMPL1 LDA =$00     CLEAR A
1F94  F5 08              SBC X1,X     SUBTRACT BYTE OF EXP1
1F96  95 08              STA X1,X     RESTORE IT
1F98  CA                 DEX          NEXT MORE SIGNIFICANT BYTE
1F99  D0 F7              BNE COMPL1   LOOP UNTIL DONE
1F9B  F0 BC              BEQ ADDEND   NORMALIZE (OR SHIFT RIGHT IF OVERFLOW)
                  *
                  *
                  *     EXP/MANT2 / EXP/MANT1 RESULT IN EXP/MANT1
```

```
                        *
 1F9D  20 0D 1F   FDIV   JSR MD1      TAKE ABS VAL OF MANT1, MANT2
 1FA0  E5 08             SBC X1       SUBTRACT EXP1 FROM EXP2
 1FA2  20 CD 1F          JSR MD2      SAVE AS QUOTIENT EXP
 1FA5  38         DIV1   SEC          SET CARRY FOR SUBTRACT
 1FA6  A2 02             LDX =$02     INDEX FOR 3-BYTE INSTRUCTION
 1FA8  B5 05      DIV2   LDA M2,X
 1FAA  F5 0C             SBC E,X      SUBTRACT A BYTE OF E FROM MANT2
 1FAC  48                PHA          SAVE ON STACK
 1FAD  CA                DEX          NEXT MORE SIGNIF BYTE
 1FAE  10 F8             BPL DIV2     LOOP UNTIL DONE
 1FB0  A2 FD             LDX =$FD     INDEX FOR 3-BYTE CONDITIONAL MOVE
 1FB2  68         DIV3   PLA          PULL A BYTE OF DIFFERENCE OFF STACK
 1FB3  90 02             BCC DIV4     IF MANT2<E THEN DONT RESTORE MANT2
 1FB5  95 08             STA M2+3,X
 1FB7  E8         DIV4   INX          NEXT LESS SIGNIF BYTE
 1FB8  D0 F8             BNE DIV3     LOOP UNTIL DONE
 1FBA  26 0B             ROL M1+2
 1FBC  26 0A             ROL M1+1     ROLL QUOTIENT LEFT, CARRY INTO LSB
 1FBE  26 09             ROL M1
 1FC0  06 07             ASL M2+2
 1FC2  26 06             ROL M2+1     SHIFT DIVIDEND LEFT
 1FC4  26 05             ROL M2
 1FC6  B0 1C             BCS OVFL     OVERFLOW IS DUE TO UNNORMALIZED DIVISOR
 1FC8  88                DEY          NEXT DIVIDE ITERATION
 1FC9  D0 DA             BNE DIV1     LOOP UNTIL DONE 23 ITERATIONS
 1FCB  F0 BE             BEQ MDEND    NORMALIZE QUOTIENT AND CORRECT SIGN
 1FCD  86 0B      MD2    STX M1+2
 1FCF  86 0A             STX M1+1     CLR MANT1 (3 BYTES) FOR MUL/DIV
 1FD1  86 09             STX M1
 1FD3  B0 0D             BCS OVCHK    IF EXP CALC SET CARRY, CHECK FOR OVFL
 1FD5  30 04             BMI MD3      IF NEG NO UNDERFLOW
 1FD7  68                PLA          POP ONE
 1FD8  68                PLA          RETURN LEVEL
 1FD9  90 B2             BCC NORMX    CLEAR X1 AND RETURN
 1FDB  49 80      MD3    EOR =$80     COMPLIMENT SIGN BIT OF EXP
 1FDD  85 08             STA X1       STORE IT
 1FDF  A0 17             LDY =$17     COUNT FOR 24 MUL OR 23 DIV ITERATIONS
 1FE1  60                RTS          RETURN
 1FE2  10 F7      OVCHK  BPL MD3      IF POS EXP THEN NO OVERFLOW
 1FE4  00         OVFL   BRK
                  *
                  *
                  *     CONVERT EXP/MANT1 TO INTEGER IN M1 (HIGH) AND M1+1(LOW)
                  *      EXP/MANT2 UNEFFECTED
                  *
 1FE5  20 5F 1F          JSR RTAR     SHIFT MANT1 RT AND INCREMENT EXPNT
 1FE8  A5 08      FIX    LDA X1       CHECK EXPONENT
 1FEA  C9 8E             CMP =$8E     IS EXPONENT 14?
 1FEC  D0 F7             BNE FIX-3    NO, SHIFT
 1FEE  60         RTRN   RTS          RETURN
                         END
```

OBJECT CODE DUMP

```
1D00  A5 09 F0 02 10 01 00 20 1C 1F A2 00 A5 04 A0 80
1D10  84 04 49 80 85 0A 10 01 CA 86 09 20 2C 1F A2 03
1D20  B5 04 95 10 B5 08 95 18 BD D4 1D 95 08 CA 10 F0
1D30  20 4A 1F A2 03 B5 08 95 14 B5 10 95 08 BD D4 1D
1D40  95 04 CA 10 F0 20 50 1F A2 03 B5 14 95 04 CA 10
1D50  F9 20 9D 1F A2 03 B5 08 95 14 95 04 CA 10 F7 20
```

```
1D60  77 1F 20 1C 1F A2 03 BD E4 1D 95 08 CA 10 F8 20
1D70  4A 1F A2 03 BD E0 1D 95 04 CA 10 F8 20 9D 1F A2
1D80  03 BD DC 1D 95 04 CA 10 F8 20 50 1F A2 03 B5 14
1D90  95 04 CA 10 F9 20 77 1F A2 03 BD E8 1D 95 04 CA
1DA0  10 F8 20 50 1F A2 03 B5 18 95 04 CA 10 F9 20 50
1DB0  1F A2 03 BD D8 1D 95 04 CA 10 F8 20 77 1F 60 20
1DC0  00 1D A2 03 BD D0 1D 95 04 CA 10 F8 20 77 1F 60
1DD0  73 6F 2D ED 80 5A 82 7A 7F 58 B9 0C 80 52 B0 40
1DE0  81 AB 86 49 80 6A 08 66 7F 40 00 00

1E00  A2 03 BD D8 1E 95 04 CA 10 F8 20 77 1F A2 03 B5
1E10  08 95 10 CA 10 F9 20 E8 1F A5 0A 85 1C 38 E9 7C
1E20  A5 09 E9 00 10 15 18 A5 0A 69 78 A5 09 69 00 10
1E30  0B A9 00 A2 03 95 08 CA 10 FB 60 00 20 2C 1F A2
1E40  03 B5 10 95 04 CA 10 F9 20 4A 1F A2 03 B5 08 95
1E50  10 95 04 CA 10 F7 20 77 1F A2 03 BD DC 1E 95 04
1E60  B5 08 95 18 CA 10 F4 20 50 1F A2 03 BD E0 1E 95
1E70  04 CA 10 F8 20 9D 1F A2 03 B5 08 95 14 BD E4 1E
1E80  95 08 B5 18 95 04 CA 10 F0 20 77 1F 20 1C 1F A2
1E90  03 B5 14 95 08 CA 10 F9 20 4A 1F A2 03 BD E8 1E
1EA0  95 04 CA 10 F8 20 50 1F 20 1C 1F A2 03 B5 10 95
1EB0  08 CA 10 F9 20 4A 1F A2 03 B5 10 95 04 CA 10 F9
1EC0  20 9D 1F A2 03 BD E8 1D 95 04 CA 10 F8 20 50 1F
1ED0  38 A5 1C 65 08 85 08 60 80 5C 55 1E 86 57 6A E1
1EE0  89 4D 3F 1D 7B 46 FA 70 83 4F A3 03

1F00  18 A2 02 B5 09 75 05 95 09 CA 10 F7 60 06 03 20
1F10  12 1F 24 09 10 05 20 8F 1F E6 03 38 A2 04 94 0B
1F20  B5 07 B4 03 94 07 95 03 CA D0 F3 60 A9 8E 85 08
1F30  A9 00 85 0B F0 08 C6 08 06 0B 26 0A 26 09 A5 09
1F40  0A 45 09 30 04 A5 08 D0 ED 60 20 8F 1F 20 5D 1F
1F50  A5 04 C5 08 D0 F7 20 00 1F 50 E3 70 05 90 BD A5
1F60  09 0A E6 08 F0 7E A2 FA A9 80 B0 01 0A 56 0F 15
1F70  0F 95 0F E8 D0 F2 60 20 0D 1F 65 08 20 CD 1F 18
1F80  20 66 1F 90 03 20 00 1F 88 10 F5 46 03 90 AF 38
1F90  A2 03 A9 00 F5 08 95 08 CA D0 F7 F0 BC 20 0D 1F
1FA0  E5 08 20 CD 1F 38 A2 02 B5 05 F5 0C 48 CA 10 F8
1FB0  A2 FD 68 90 02 95 08 E8 D0 F8 26 0B 26 0A 26 09
1FC0  06 07 26 06 26 05 B0 1C 88 D0 DA F0 BE 86 0B 86
1FD0  0A 86 09 B0 0D 30 04 68 68 90 B2 49 80 85 08 A0
1FE0  17 60 10 F7 00 20 5F 1F A5 08 C9 8E D0 F7 60
```

```
+------------------------------------------------------------------------------
|  TOPIC -- SYM Computer -- SYM Monitor listing
+------------------------------------------------------------------------------

SYM-1 SUPERMON AND AUDIO CASSETTE INTERFACE SOURCES
COMBINED AND CONVERTED TO TELEMARK ASSEMBLER (TASM) V3.1

0000             ;
0000             ;*****
0000             ;***** COPYRIGHT 1979 SYNERTEK SYSTEMS CORPORATION
0000             ;***** VERSION 2  4/13/79  "SY1.1"
A600                    *=$A600         ;SYS RAM (ECHOED AT TOP OF MEM)
A600             SCPBUF .BLOCK $20      ;SCOPE BUFFER LAST 32 CHARS
A620             RAM    =*              ;DEFAULT BLK FILLS STARTING HERE
A620             JTABLE .BLOCK $10      ; 8JUMPS - ABS ADDR, LO HI ORDER
A630             TAPDEL .BLOCK 1        ;KH TAPE DELAY
A631             KMBDRY .BLOCK 1        ;KIM TAPE READ BOUNDARY
A632             HSBDRY .BLOCK 1        ;HS TAPE READ BOUNDARY
A633             SCR3   .BLOCK 1        ;RAM SCRATCH LOCS 3-F
A634             SCR4   .BLOCK 1
A635             TAPET1 .BLOCK 1        ;HS TAPE 1/2 BIT TIME
A636             SCR6   .BLOCK 1
A637             SCR7   .BLOCK 1
A638             SCR8   .BLOCK 1
A639             SCR9   .BLOCK 1
A63A             SCRA   .BLOCK 1
A63B             SCRB   .BLOCK 1
A63C             TAPET2 .BLOCK 1        ;HS TAPE 1/2 BIT TIME
A63D             SCRD   .BLOCK 1
A63E             RC     =SCRD
A63E             SCRE   .BLOCK 1
A63F             SCRF   .BLOCK 1
A640             DISBUF .BLOCK 5        ;DISPLAY BUFFER
A645             RDIG   .BLOCK 1        ;RIGHT MOST DIGIT OF DISPLAY
A646                    .BLOCK 3        ;NOT USED
A649             PARNR  .BLOCK 1        ;NUMBER OF PARMS RECEIVED
A64A             ;
A64A             ; 3 16 BIT PARMS, LO HI ORDER
A64A             ; PASSED TO EXECUTE BLOCKS
A64A             ;
A64A             P3L    .BLOCK 1
A64B             P3H    .BLOCK 1
A64C             P2L    .BLOCK 1
A64D             P2H    .BLOCK 1
A64E             P1L    .BLOCK 1
A64F             P1H    .BLOCK 1
A650             PADBIT .BLOCK 1        ;PAD BITS FOR CARRIAGE RETURN
A651             SDBYT  .BLOCK 1        ;SPEED BYTE FOR TERMINAL I/O
A652             ERCNT  .BLOCK 1        ; ERROR COUNT  (MAX $FF)
A653             ; BIT 7 = ECHO /NO ECHO, BIT 6 = CTL O TOGGLE SW
A653             TECHO  .BLOCK 1        ;TERMINAL ECHO LAG
A654             ; BIT7 =CRT IN, 6 =TTY IN, 5 = TTY OUT, 4 = CRT OUT
A654             TOUTFL .BLOCK 1        ;OUTPUT FLAGS
A655             KSHFL  .BLOCK 1        ;KEYBOARD SHIFT FLAG
A656             TV     .BLOCK 1        ;TRACE VELOCITY (0=SINGLE STEP)
A657             LSTCOM .BLOCK 1        ;STORE LAST MONITOR COMMAND
A658             MAXRC  .BLOCK 1        ;MAXIMUM REC LENGTH FOR MEM DUMP
A659             ;
A659             ; USER REG'S FOLLOW
A659             ;
```

```
A659              PCLR   .BLOCK 1        ;PROG CTR
A65A              PCHR   .BLOCK 1
A65B              SR     .BLOCK 1        ;STACK
A65C              FR     .BLOCK 1        ;FLAGS
A65D              AR     .BLOCK 1        ;AREG
A65E              XR     .BLOCK 1        ;XREG
A65F              YR     .BLOCK 1        ;YREG
A660              ;
A660              ; I/O VECTORS FOLLOW
A660              ;
A660              INVEC  .BLOCK 3        ;IN CHAR
A663              OUTVEC .BLOCK 3        ;OUT CHAR
A666              INSVEC .BLOCK 3        ;IN STATUS
A669              URSVEC .BLOCK 3        ;UNRECOGNIZED SYNTAX VECTOR
A66C              URCVEC .BLOCK 3        ;UNRECOGNIZED CMD/ERROR VECTOR
A66F              SCNVEC .BLOCK 3        ;SCAN ON-BOARD DISPLAY
A672              ;
A672              ; TRACE, INTERRUPT VECTORS
A672              ;
A672              EXEVEC .BLOCK 2        ; EXEC CMD ALTERNATE INVEC
A674              TRCVEC .BLOCK 2        ;TRACE
A676              UBRKVC .BLOCK 2        ;USER BRK AFTER MONITOR
A678              UBRKV  =UBRKVC
A678              UIRQVC .BLOCK 2        ;USER NON-BRK IRQ AFTER MONITOR
A67A              UIRQV  =UIRQVC
A67A              NMIVEC .BLOCK 2        ;NMI
A67C              RSTVEC .BLOCK 2        ;RESET
A67E              IRQVEC .BLOCK 2        ;IRQ
A680              ;
A680              ;
A680              ;I/O REG DEFINITIONS
A680              PADA   =$A400          ;KEYBOARD/DISPLAY
A680              PBDA   =$A402          ;SERIAL I/O
A680              OR3A   =$AC01          ;WP, DBON, DBOFF
A680              DDR3A  =OR3A+2         ;DATA DIRECTION FOR SAME
A680              OR1B   =$A000
A680              DDR1B  =$A002
A680              PCR1   =$A00C          ; POR/TAPE REMOTE
A680              ;
A680              ; MONITOR MAINLINE
A680              ;
8000                     *=$8000
8000 4C 7C 8B     MONITR JMP MONENT      ;INIT S, CLD, GET ACCESS
8003 20 FF 80     WARM   JSR GETCOM      ;GET COMMAND + PARMS (0-3)
8006 20 4A 81            JSR DISPAT      ;DISPATCH CMD,PARMS TO EXEC BLKS
8009 20 71 81            JSR ERMSG       ;DISP ER MSG IF CARRY SET
800C 4C 03 80            JMP WARM        ;AND CONTINUE
800F              ;
800F              ; TRACE AND INTERRUPT ROUTINES
800F              ;
800F 08           IRQBRK PHP             ;IRQ OR BRK ?
8010 48                  PHA
8011 8A                  TXA
8012 48                  PHA
8013 BA                  TSX
8014 BD 04 01            LDA $0104,X     ;PICK UP FLAGS
8017 29 10               AND #$10
8019 F0 07               BEQ DETIRQ
801B 68                  PLA             ;BRK
801C AA                  TAX
801D 68                  PLA
801E 28                  PLP
```

```
801F 6C F6 FF              JMP ($FFF6)
8022 68           DETIRQ PLA                ;IRQ (NON BRK)
8023 AA                  TAX
8024 68                  PLA
8025 28                  PLP
8026 6C F8 FF            JMP ($FFF8)
8029 20 86 8B     SVIRQ  JSR ACCESS         ;SAVE REGS AND DISPLAY CODE
802C 38                  SEC
802D 20 64 80            JSR SAVINT
8030 A9 31               LDA #'1'
8032 4C 53 80            JMP IDISP
8035 08           USRENT PHP                ;USER ENTRY
8036 20 86 8B            JSR ACCESS
8039 38                  SEC
803A 20 64 80            JSR SAVINT
803D EE 59 A6            INC PCLR
8040 D0 03               BNE *+5
8042 EE 5A A6            INC PCHR
8045 A9 33               LDA #'3'
8047 4C 53 80            JMP IDISP
804A 20 86 8B     SVBRK  JSR ACCESS
804D 18                  CLC
804E 20 64 80            JSR SAVINT
8051 A9 30               LDA #'0'
8053              ; INTRPT CODES  0 = BRK
8053              ;               1 = IRQ
8053              ;               2 = NMI
8053              ;               3 = USER ENTRY
8053 48           IDISP  PHA                ;OUT PC, INTRPT CODE (FROM A)
8054 20 D3 80            JSR DBOFF          ;STOP NMI'S
8057 20 4D 83            JSR CRLF
805A 20 37 83            JSR OPCCOM
805D 68                  PLA
805E 20 47 8A            JSR OUTCHR
8061 4C 03 80            JMP WARM
8064 8D 5D A6     SAVINT STA AR             ;SAVE USER REGS AFTER INTRPT
8067 8E 5E A6            STX XR
806A 8C 5F A6            STY YR
806D BA                  TSX
806E D8                  CLD
806F BD 04 01            LDA $104,X
8072 69 FF               ADC #$FF
8074 8D 59 A6            STA PCLR
8077 BD 05 01            LDA $105,X
807A 69 FF               ADC #$FF
807C 8D 5A A6            STA PCHR
807F BD 03 01            LDA $103,X
8082 8D 5C A6            STA FR
8085 BD 02 01            LDA $102,X
8088 9D 05 01            STA $105,X
808B BD 01 01            LDA $101,X
808E 9D 04 01            STA $104,X
8091 E8                  INX
8092 E8                  INX
8093 E8                  INX
8094 9A                  TXS
8095 E8                  INX
8096 E8                  INX
8097 8E 5B A6            STX SR
809A 60                  RTS
809B 20 86 8B     SVNMI  JSR ACCESS         ;TRACE IF TV NE 0
809E 38                  SEC
```

```
809F 20 64 80           JSR SAVINT
80A2 20 D3 80           JSR DBOFF       ;STOP NMI'S
80A5 AD 56 A6           LDA TV
80A8 D0 05              BNE TVNZ
80AA A9 32              LDA #'2'
80AC 4C 53 80           JMP IDISP
80AF 20 37 83    TVNZ   JSR OPCCOM      ;TRACE WITH DELAY
80B2 AD 5D A6           LDA AR
80B5 20 4A 83           JSR OBCRLF      ;DISPLAY ACC
80B8 20 5A 83           JSR DELAY
80BB 90 10              BCC TRACON      ;STOP IF KEY ENTERED
80BD 4C 03 80           JMP WARM
80C0 20 86 8B    TRCOFF JSR ACCESS      ;DISABLE NMIS
80C3 38                 SEC
80C4 20 64 80           JSR SAVINT
80C7 20 D3 80           JSR DBOFF
80CA 6C 74 A6           JMP (TRCVEC)    ;AND GO TO SPECIAL TRACE
80CD 20 E4 80    TRACON JSR DBON        ;ENABLE NMI'S
80D0 4C FD 83           JMP GO1ENT+3    ;AND RESUME (NO WRITE PROT)
80D3 AD 01 AC    DBOFF  LDA OR3A        ;PULSE DEBUG OFF
80D6 29 DF              AND #$DF
80D8 09 10              ORA #$10
80DA 8D 01 AC           STA OR3A
80DD AD 03 AC           LDA DDR3A
80E0 09 30              ORA #$30
80E2 D0 0F              BNE DBNEW-3     ;RELEASE FLIP FLOP SO KEY WORKS
80E4 AD 01 AC    DBON   LDA OR3A        ;PULSE DEBUG ON
80E7 29 EF              AND #$EF
80E9 09 20              ORA #$20
80EB 8D 01 AC           STA OR3A
80EE AD 03 AC           LDA DDR3A
80F1 09 30              ORA #$30
80F3 8D 03 AC           STA DDR3A
80F6 AD 03 AC    DBNEW  LDA DDR3A       ;RELEASE FLIP FLOP
80F9 29 CF              AND #$CF
80FB 8D 03 AC           STA DDR3A
80FE 60                 RTS
80FF             ;
80FF             ; GETCOM - GET COMMAND AND 0-3 PARMS
80FF             ;
80FF 20 4D 83    GETCOM JSR CRLF
8102 A9 2E              LDA #'.'        ;PROMPT
8104 20 47 8A           JSR OUTCHR
8107 20 1B 8A    GETC1  JSR INCHR
810A F0 F3              BEQ GETCOM      ;CARRIAGE RETURN?
810C C9 7F              CMP #$7F        ;DELETE?
810E F0 F7              BEQ GETC1
8110 C9 00              CMP #0          ;NULL?
8112 F0 F3              BEQ GETC1
8114             ; L,S,U NEED TO BE HASHED 2 BYTES TO ONE
8114 C9 53              CMP #'S'
8116 F0 1B              BEQ HASHUS
8118 C9 55              CMP #'U'
811A F0 17              BEQ HASHUS
811C C9 4C              CMP #'L'
811E F0 0F              BEQ HASHL
8120 8D 57 A6    STOCOM STA LSTCOM
8123 20 42 83           JSR SPACE
8126 20 08 82           JSR PSHOVE      ;ZERO PARMS
8129 20 08 82           JSR PSHOVE
812C 4C 20 82           JMP PARM        ;AND GO GET PARMS
812F A9 01       HASHL  LDA #$01        ;HASH LOAD CMDS TO ONE BYTE
```

```
8131 10 02            BPL HASHUS+2
8133 0A        HASHUS ASL A           ;HASH 'USER' CMDS TO ONE BYTE A
8134 0A               ASL A           ;U0 = $14 THRU U17 =$1B
8135 8D 57 A6         STA LSTCOM
8138 20 1B 8A         JSR INCHR       ;GET SECOND
813B F0 C2            BEQ GETCOM
813D 18               CLC
813E 6D 57 A6         ADC LSTCOM
8141 29 0F            AND #$0F
8143 09 10            ORA #$10
8145 10 D9            BPL STOCOM
8147 FF FF FF         .DB $FF,$FF,$FF ;NOT USED
814A           ;
814A           ;DISPATCH TO EXEC BLK 0PARM, 1PARM, 2PARM, OR 3PARM
814A           ;
814A C9 0D     DISPAT CMP #$0D        ;C/R IF OK ELSE URSVEC
814C D0 20            BNE HIPN
814E AD 57 A6         LDA LSTCOM
8151 AE 49 A6         LDX PARNR
8154 D0 03            BNE M12
8156 4C 95 83         JMP BZPARM      ;0 PARM BLOCK
8159 E0 01     M12    CPX #$01
815B D0 03            BNE M13
815D 4C DA 84         JMP B1PARM      ;1 PARM BLOCK
8160 E0 02     M13    CPX #$02
8162 D0 03            BNE M14
8164 4C 19 86         JMP B2PARM      ;2 PARM BLOCK
8167 E0 03     M14    CPX #$03
8169 D0 03            BNE HIPN
816B 4C 14 87         JMP B3PARM      ;3 PARM BLOCK
816E 6C 6A A6  HIPN   JMP (URSVEC+1)  ;ELSE UNREC SYNTAX VECTOR
8171           ;
8171           ; ERMSG - PRINT ACC IN HEX IF CARRY SET
8171           ;
8171 90 44     ERMSG  BCC M15
8173 48               PHA
8174 20 4D 83         JSR CRLF
8177 A9 45            LDA #'E'
8179 20 47 8A         JSR OUTCHR
817C A9 52            LDA #'R'
817E 20 47 8A         JSR OUTCHR
8181 20 42 83         JSR SPACE
8184 68               PLA
8185 4C FA 82         JMP OUTBYT
8188           ;
8188           ; SAVER - SAVE ALL REG'S + FLAGS ON STACK
8188           ; RETURN WITH F,A,X,Y UNCHANGED
8188           ; STACK HAS         FLAGS,A,X,Y, PUSHED
8188 08        SAVER  PHP
8189 48               PHA
818A 48               PHA
818B 48               PHA
818C 08               PHP
818D 48               PHA
818E 8A               TXA
818F 48               PHA
8190 BA               TSX
8191 BD 09 01         LDA $0109,X
8194 9D 05 01         STA $0105,X
8197 BD 07 01         LDA $0107,X
819A 9D 09 01         STA $0109,X
819D BD 01 01         LDA $0101,X
```

```
81A0 9D 07 01              STA $0107,X
81A3 BD 08 01              LDA $0108,X
81A6 9D 04 01              STA $0104,X
81A9 BD 06 01              LDA $0106,X
81AC 9D 08 01              STA $0108,X
81AF 98                    TYA
81B0 9D 06 01              STA $0106,X
81B3 68                    PLA
81B4 AA                    TAX
81B5 68                    PLA
81B6 28                    PLP
81B7 60             M15    RTS
81B8                ; RESTORE EXCEPT A,F
81B8 08             RESXAF PHP
81B9 BA                    TSX
81BA 9D 04 01              STA $0104,X
81BD 28                    PLP
81BE                ; RESTORE EXCEPT F
81BE 08             RESXF  PHP
81BF 68                    PLA
81C0 BA                    TSX
81C1 9D 04 01              STA $0104,X
81C4                ; RESTORE ALL 100%
81C4 68             RESALL PLA
81C5 A8                    TAY
81C6 68                    PLA
81C7 AA                    TAX
81C8 68                    PLA
81C9 28                    PLP
81CA 60                    RTS
81CB                ;
81CB                ; MONITOR UTILITIES
81CB                ;
81CB C9 20          ADVCK  CMP #$20        ;SPACE?
81CD F0 02                 BEQ M1
81CF C9 3E                 CMP #'>'        ;FWD ARROW?
81D1 38             M1     SEC
81D2 60                    RTS
81D3 20 FA 82       OBCMIN JSR OUTBYT      ;OUT BYTE, OUT COMMA, IN BYTE
81D6 20 3A 83       COMINB JSR COMMA       ;OUT COMMA, IN BYTE
81D9 20 1B 8A       INBYTE JSR INCHR
81DC 20 75 82              JSR ASCNIB
81DF B0 14                 BCS OUT4
81E1 0A                    ASL A
81E2 0A                    ASL A
81E3 0A                    ASL A
81E4 0A                    ASL A
81E5 8D 33 A6              STA SCR3
81E8 20 1B 8A              JSR INCHR
81EB 20 75 82              JSR ASCNIB
81EE B0 11                 BCS OUT2
81F0 0D 33 A6              ORA SCR3
81F3 18             GOOD   CLC
81F4 60                    RTS
81F5 C9 3A          OUT4   CMP #':'        ;COLON ?
81F7 D0 05                 BNE OUT1
81F9 20 1B 8A              JSR INCHR
81FC D0 F5                 BNE GOOD        ;CARRIAGE RETURN?
81FE B8             OUT1   CLV
81FF 50 03                 BVC CRCHK
8201 2C 04 82       OUT2   BIT CRCHK
8204 C9 0D          CRCHK  CMP #$0D        ;CHECK FOR C/R
```

```
8206 38                     SEC
8207 60                     RTS
8208 A2 10        PSHOVE LDX #$10       ;PUSH PARMS DOWN
820A 0E 4A A6     PRM10  ASL P3L
820D 2E 4B A6            ROL P3H
8210 2E 4C A6            ROL P2L
8213 2E 4D A6            ROL P2H
8216 2E 4E A6            ROL P1L
8219 2E 4F A6            ROL P1H
821C CA                  DEX
821D D0 EB               BNE PRM10
821F 60                  RTS
8220 20 88 81     PARM   JSR SAVER      ;GET PARMS - RETURN ON C/R OR ERR
8223 A9 00               LDA #0
8225 8D 49 A6            STA PARNR
8228 8D 33 A6            STA SCR3
822B 20 08 82     PM1    JSR PSHOVE
822E 20 1B 8A     PARFIL JSR INCHR
8231 C9 2C               CMP #','       ;VALID DELIMETERS - ,
8233 F0 04               BEQ M21
8235 C9 2D               CMP #'-'
8237 D0 11               BNE M22
8239 A2 FF        M21    LDX #$FF
823B 8E 33 A6            STX SCR3
823E EE 49 A6            INC PARNR
8241 AE 49 A6            LDX PARNR
8244 E0 03               CPX #$03
8246 D0 E3               BNE PM1
8248 F0 1D               BEQ M24
824A 20 75 82     M22    JSR ASCNIB
824D B0 18               BCS M24
824F A2 04               LDX #4
8251 0E 4A A6     M23    ASL P3L
8254 2E 4B A6            ROL P3H
8257 CA                  DEX
8258 D0 F7               BNE M23
825A 0D 4A A6            ORA P3L
825D 8D 4A A6            STA P3L
8260 A9 FF               LDA #$FF
8262 8D 33 A6            STA SCR3
8265 D0 C7               BNE PARFIL
8267 2C 33 A6     M24    BIT SCR3
826A F0 03               BEQ M25
826C EE 49 A6            INC PARNR
826F C9 0D        M25    CMP #$0D
8271 18                  CLC
8272 4C B8 81            JMP RESXAF
8275 C9 0D        ASCNIB CMP #$0D       ;C/R?
8277 F0 19               BEQ M29
8279 C9 30               CMP #'0'
827B 90 0C               BCC M26
827D C9 47               CMP #'G'
827F B0 08               BCS M26
8281 C9 41               CMP #'A'
8283 B0 08               BCS M27
8285 C9 3A               CMP #':'
8287 90 06               BCC M28
8289 C9 30        M26    CMP #'0'
828B 38                  SEC            ;CARRY SET - NON HEX
828C 60                  RTS
828D E9 37        M27    SBC #$37
828F 29 0F        M28    AND #$0F
```

```
8291 18                  CLC
8292 60           M29    RTS
8293 EE 4A A6     INCP3  INC P3L          ;INCREMENT P3 (16 BITS)
8296 D0 03               BNE *+5
8298 EE 4B A6            INC P3H
829B 60                  RTS
829C AE 4D A6     P2SCR  LDX P2H          ;MOVE P2 TO FE,FF
829F 86 FF               STX $FF
82A1 AE 4C A6            LDX P2L
82A4 86 FE               STX $FE
82A6 60                  RTS
82A7 AE 4B A6     P3SCR  LDX P3H          ;MOVE P3 TO FE,FF
82AA 86 FF               STX $FF
82AC AE 4A A6            LDX P3L
82AF 86 FE               STX $FE
82B1 60                  RTS
82B2 E6 FE        INCCMP INC $FE          ;INCREM FE,FF, COMPARE TO P3
82B4 D0 14               BNE COMPAR
82B6 E6 FF               INC $FF
82B8 D0 10        WRAP   BNE COMPAR       ;TEST TO WRAP AROUND
82BA 2C BD 82            BIT EXWRAP
82BD 60           EXWRAP RTS
82BE A5 FE        DECCMP LDA $FE          ;DECREM FE,FF AND COMPARE TO P3
82C0 D0 06               BNE M32
82C2 A5 FF               LDA $FF
82C4 F0 F2               BEQ WRAP
82C6 C6 FF               DEC $FF
82C8 C6 FE        M32    DEC $FE
82CA 20 88 81     COMPAR JSR SAVER        ;COMPARE FE,FF TO P3
82CD A5 FF               LDA $FF
82CF CD 4B A6            CMP P3H
82D2 D0 05               BNE EXITCP
82D4 A5 FE               LDA $FE
82D6 CD 4A A6            CMP P3L
82D9 B8           EXITCP CLV
82DA 4C BE 81            JMP RESXF
82DD 08           CHKSAD PHP              ;16 BIT CKSUM IN SCR6,7
82DE 48                  PHA
82DF 18                  CLC
82E0 6D 36 A6            ADC SCR6
82E3 8D 36 A6            STA SCR6
82E6 90 03               BCC M33
82E8 EE 37 A6            INC SCR7
82EB 68           M33    PLA
82EC 28                  PLP
82ED 60                  RTS
82EE AD 59 A6     OUTPC  LDA PCLR         ;OUTPUT PC
82F1 AE 5A A6            LDX PCHR
82F4 48           OUTXAH PHA
82F5 8A                  TXA
82F6 20 FA 82            JSR OUTBYT
82F9 68                  PLA
82FA 48           OUTBYT PHA              ;OUTPUT 2 HEX DIGS FROM A
82FB 48                  PHA
82FC 4A                  LSR A
82FD 4A                  LSR A
82FE 4A                  LSR A
82FF 4A                  LSR A
8300 20 44 8A            JSR NBASOC
8303 68                  PLA
8304 20 44 8A            JSR NBASOC
8307 68                  PLA
```

```
8308 60                RTS
8309 29 0F      NIBASC AND #$0F        ;NIBBLE IN A TO ASCII IN A
830B C9 0A             CMP #$0A        ;LINE FEED
830D B0 04             BCS NIBALF
830F 69 30             ADC #$30
8311 90 02             BCC EXITNB
8313 69 36      NIBALF ADC #$36
8315 60         EXITNB RTS
8316 20 4D 83   CRLFSZ JSR CRLF        ;PRINT CRLF, FF, FE
8319 A6 FF             LDX $FF
831B A5 FE             LDA $FE
831D 4C F4 82          JMP OUTXAH
8320 A9 3F      OUTQM  LDA #'?'
8322 4C 47 8A          JMP OUTCHR
8325 20 3A 83   OCMCK  JSR COMMA       ;OUT COMMA, CKSUM LO
8328 AD 36 A6          LDA SCR6
832B 4C FA 82          JMP OUTBYT
832E A9 00      ZERCK  LDA #0          ;INIT CHECKSUM
8330 8D 36 A6          STA SCR6
8333 8D 37 A6          STA SCR7
8336 60                RTS
8337 20 EE 82   OPCCOM JSR OUTPC       ;PC OUT, COMMA OUT
833A 48         COMMA  PHA             ;COMMA OUT
833B A9 2C             LDA #','
833D D0 06             BNE SPCP3
833F 20 42 83   SPC2   JSR SPACE       ;2 SPACES OUT
8342 48         SPACE  PHA             ;1 SPACE OUT
8343 A9 20             LDA #$20        ;SPACE
8345 20 47 8A   SPCP3  JSR OUTCHR
8348 68                PLA
8349 60                RTS
834A 20 FA 82   OBCRLF JSR OUTBYT      ;BYTE OUT, CRLF OUT
834D 48         CRLF   PHA
834E A9 0D             LDA #$0D
8350 20 47 8A          JSR OUTCHR
8353 A9 0A             LDA #$0A        ;LINE FEED
8355 20 47 8A          JSR OUTCHR
8358 68                PLA
8359 60                RTS
835A AE 56 A6   DELAY  LDX TV          ;DELAY DEPENDS ON TV
835D 20 88 81   DL1    JSR SAVER
8360 A9 FF             LDA #$FF
8362 8D 39 A6          STA SCR9
8365 8D 38 A6          STA SCR8
8368 0E 38 A6   DLY1   ASL SCR8        ;(SCR9,8)=FFFF-2**X
836B 2E 39 A6          ROL SCR9
836E CA                DEX
836F D0 F7             BNE DLY1
8371 20 03 89   DLY2   JSR IJSCNV      ;SCAN DISPLAY
8374 20 86 83          JSR INSTAT      ;SEE IF KEY DOWN
8377 B0 0A             BCS DLY0
8379 EE 38 A6          INC SCR8        ;SCAN 2**X+1 TIMES
837C D0 03             BNE *+5
837E EE 39 A6          INC SCR9
8381 D0 EE             BNE DLY2
8383 4C BE 81   DLY0   JMP RESXF
8386            ; INSTAT - SEE IF KEY DOWN, RESULT IN CARRY
8386            ; KEYSTAT, TSTAT RETURN IMMEDIATELY W/STATUS
8386            ; INSTAT WAITS FOR RELEASE
8386 20 92 83   INSTAT JSR INJISV
8389 90 06             BCC INST2
838B 20 92 83   INST1  JSR INJISV
```

```
8 38E B0 FB             BCS INST1
8390 38                 SEC
8391 60          INST2  RTS
8392 6C 67 A6    INJISV JMP (INSVEC+1)
8395             ;
8395             ;
8395             ; *** EXECUTE BLOCKS BEGIN HERE
8395             ;
8395             BZPARM =*
8395             ; ZERO PARM COMMANDS
8395             ;
8395 C9 52       REGZ   CMP #'R'         ;DISP REGISTERS
8397 D0 5A              BNE GOZ          ;PC,S,F,A,X,Y
8399 20 4D 83    RGBACK JSR CRLF
839C A9 50              LDA #'P'
839E 20 47 8A           JSR OUTCHR
83A1 20 42 83           JSR SPACE
83A4 20 EE 82           JSR OUTPC
83A7 20 D6 81           JSR COMINB
83AA B0 13              BCS NH3
83AC 8D 34 A6           STA SCR4
83AF 20 D9 81           JSR INBYTE
83B2 B0 0B              BCS NH3
83B4 8D 59 A6           STA PCLR
83B7 AD 34 A6           LDA SCR4
83BA 8D 5A A6           STA PCHR
83BD 90 09              BCC M34
83BF D0 02       NH3    BNE NOTCR
83C1 18          EXITRG CLC
83C2 60          EXRGP1 RTS
83C3 20 CB 81    NOTCR  JSR ADVCK
83C6 D0 FA              BNE EXRGP1
83C8 A0 00       M34    LDY #0
83CA C8          M35    INY
83CB C0 06              CPY #6
83CD F0 CA              BEQ RGBACK
83CF 20 4D 83           JSR CRLF
83D2 B9 99 8F           LDA RGNAM-1,Y   ;GET REG NAME
83D5             ; OUTPUT 3 SPACES TO LINE UP DISPLAY
83D5 20 47 8A           JSR OUTCHR
83D8 20 42 83           JSR SPACE
83DB 20 3F 83           JSR SPC2
83DE B9 5A A6           LDA PCHR,Y
83E1 20 D3 81           JSR OBCMIN
83E4 B0 05              BCS M36
83E6 99 5A A6           STA PCHR,Y
83E9 90 DF              BCC M35
83EB F0 D4       M36    BEQ EXITRG
83ED 20 CB 81           JSR ADVCK
83F0 F0 D8              BEQ M35
83F2 60                 RTS
83F3 C9 47       GOZ    CMP #'G'
83F5 D0 20              BNE LPZB
83F7 20 4D 83           JSR CRLF
83FA 20 9C 8B    GO1ENT JSR NACCES       ;WRITE PROT MONITOR RAM
83FD AE 5B A6           LDX SR           ;RESTORE REGS
8400 9A                 TXS
8401 AD 5A A6           LDA PCHR
8404 48                 PHA
8405 AD 59 A6           LDA PCLR
8408 48          NR10   PHA
8409 AD 5C A6           LDA FR
```

```
840C 48                   PHA
840D AC 5F A6             LDY YR
8410 AE 5E A6             LDX XR
8413 AD 5D A6             LDA AR
8416 40                   RTI
8417 C9 11       LPZB     CMP #$11          ;LOAD PAPER TAPE
8419 F0 03                BEQ *+5
841B 4C A7 84             JMP DEPZ
841E 20 88 81             JSR SAVER
8421 20 4D 83             JSR CRLF
8424 A9 00                LDA #0
8426 8D 52 A6             STA ERCNT
8429 20 2E 83    LPZ      JSR ZERCK
842C 20 1B 8A    LP1      JSR INCHR
842F C9 3B                CMP #$3B          ;SEMI COLON
8431 D0 F9                BNE LP1
8433 20 A1 84             JSR LDBYTE
8436 B0 56                BCS TAPERR
8438 D0 09                BNE NUREC
843A AD 52 A6             LDA ERCNT         ;ERRORS ?
843D F0 01                BEQ *+3
843F 38                   SEC
8440 4C B8 81             JMP RESXAF
8443 8D 3D A6    NUREC    STA SCRD
8446 20 A1 84             JSR LDBYTE
8449 B0 43                BCS TAPERR
844B 85 FF                STA $FF
844D 20 A1 84             JSR LDBYTE
8450 B0 D7                BCS LPZ
8452 85 FE                STA $FE
8454 20 A1 84    MORED    JSR LDBYTE
8457 B0 35                BCS TAPERR
8459 A0 00                LDY #0
845B 91 FE                STA ($FE),Y
845D D1 FE                CMP ($FE),Y
845F F0 0C                BEQ LPGD
8461 AD 52 A6             LDA ERCNT
8464 29 0F                AND #$0F
8466 C9 0F                CMP #$0F
8468 F0 03                BEQ *+5
846A EE 52 A6             INC ERCNT
846D 20 B2 82    LPGD     JSR INCCMP
8470 CE 3D A6             DEC SCRD
8473 D0 DF                BNE MORED
8475 20 D9 81             JSR INBYTE
8478 B0 14                BCS TAPERR
847A CD 37 A6             CMP SCR7
847D D0 0C                BNE BADDY
847F 20 D9 81             JSR INBYTE
8482 B0 0A                BCS TAPERR
8484 CD 36 A6             CMP SCR6
8487 F0 A0                BEQ LPZ
8489 D0 03                BNE TAPERR        ;(ALWAYS)
848B 20 D9 81    BADDY    JSR INBYTE
848E AD 52 A6    TAPERR   LDA ERCNT
8491 29 F0                AND #$F0
8493 C9 F0                CMP #$F0
8495 F0 92                BEQ LPZ
8497 AD 52 A6             LDA ERCNT
849A 69 10                ADC #$10
849C 8D 52 A6             STA ERCNT
849F D0 88                BNE LPZ
```

```
84A1 20 D9 81    LDBYTE JSR INBYTE
84A4 4C DD 82           JMP CHKSAD
84A7 C9 44       DEPZ   CMP #'D'        ;DEPOSIT, 0 PARM - USE (OLD)
84A9 D0 03              BNE MEMZ
84AB 4C E1 84           JMP NEWLN
84AE C9 4D       MEMZ   CMP #'M'        ;MEM, 0 PARM - USE (OLD)
84B0 D0 03              BNE VERZ
84B2 4C 17 85           JMP NEWLOC
84B5 C9 56       VERZ   CMP #'V'        ;VERIFY, 0 PARM - USE (OLD)
84B7 D0 0D              BNE L1ZB        ; ... DO 8 BYTES (LIKE VER 1 PARM)
84B9 A5 FE              LDA $FE
84BB 8D 4A A6           STA P3L
84BE A5 FF              LDA $FF
84C0 8D 4B A6           STA P3H
84C3 4C 9A 85           JMP VER1+4
84C6 C9 12       L1ZB   CMP #$12        ;LOAD KIM, ZERO PARM
84C8 D0 05              BNE L2ZB
84CA A0 00              LDY #0          ;MODE = KIM
84CC 4C 78 8C    L1J    JMP LENTRY      ;GO TO CASSETTE ROUTINE
84CF C9 13       L2ZB   CMP #$13        ;LOAD HS, ZERO PARM
84D1 D0 04              BNE EZPARM
84D3 A0 80              LDY #$80        ;MODE - HS
84D5 D0 F5              BNE L1J         ;(ALWAYS)
84D7 6C 6D A6    EZPARM JMP (URCVEC+1)  ;ELSE UNREC COMMAND
84DA             B1PARM =*
84DA             ;
84DA             ; 1 PARAMETER COMMAND EXEC BLOCKS
84DA             ;
84DA C9 44       DEP1   CMP #'D'        ;DEPOSIT, 1 PARM
84DC D0 32              BNE MEM1
84DE 20 A7 82           JSR P3SCR
84E1 20 16 83    NEWLN  JSR CRLFSZ
84E4 A0 00              LDY #0
84E6 A2 08              LDX #8
84E8 20 42 83    DEPBYT JSR SPACE
84EB 20 D9 81           JSR INBYTE
84EE B0 11              BCS NH41
84F0 91 FE              STA ($FE),Y
84F2 D1 FE              CMP ($FE),Y     ;VERIFY
84F4 F0 03              BEQ DEPN
84F6 20 20 83           JSR OUTQM       ;TYPE "?" IF NG
84F9 20 B2 82    DEPN   JSR INCCMP
84FC CA                 DEX
84FD D0 E9              BNE DEPBYT
84FF F0 E0              BEQ NEWLN
8501 F0 0B       NH41   BEQ DEPEC
8503 C9 20              CMP #$20        ;SPACE = FWD
8505 D0 4C              BNE DEPES
8507 70 F0              BVS DEPN
8509 20 42 83           JSR SPACE
850C 10 EB              BPL DEPN
850E 18          DEPEC  CLC
850F 60                 RTS
8510 C9 4D       MEM1   CMP #'M'        ;MEMORY, 1 PARM
8512 D0 65              BNE GO1
8514 20 A7 82           JSR P3SCR
8517 20 16 83    NEWLOC JSR CRLFSZ
851A 20 3A 83           JSR COMMA
851D A0 00              LDY #0
851F B1 FE              LDA ($FE),Y
8521 20 D3 81           JSR OBCMIN
8524 B0 11              BCS NH42
```

```
8526 A0 00            LDY #$00
8528 91 FE            STA ($FE),Y
852A D1 FE            CMP ($FE),Y     ;VERIFY MEM
852C F0 03            BEQ NXTLOC
852E 20 20 83         JSR OUTQM       ;TYPE ? AND CONTINUE
8531 20 B2 82  NXTLOC JSR INCCMP
8534 18               CLC
8535 90 E0            BCC NEWLOC
8537 F0 3E     NH42   BEQ EXITM1
8539 50 04            BVC *+6
853B C9 3C            CMP #'<'
853D F0 D8            BEQ NEWLOC
853F C9 20            CMP #$20        ;SPACE ?
8541 F0 EE            BEQ NXTLOC
8543 C9 3E            CMP #'>'
8545 F0 EA            BEQ NXTLOC
8547 C9 2B            CMP #'+'
8549 F0 10            BEQ LOCP8
854B C9 3C            CMP #'<'
854D F0 06            BEQ PRVLOC
854F C9 2D            CMP #'-'
8551 F0 16            BEQ LOCM8
8553 38        DEPES  SEC
8554 60               RTS
8555 20 BE 82  PRVLOC JSR DECCMP      ;BACK ONE BYT
8558 18               CLC
8559 90 BC            BCC NEWLOC
855B A5 FE     LOCP8  LDA $FE         ;GO FWD 8 BYTES
855D 18               CLC
855E 69 08            ADC #$08
8560 85 FE            STA $FE
8562 90 02            BCC M42
8564 E6 FF            INC $FF
8566 18        M42    CLC
8567 90 AE            BCC NEWLOC
8569 A5 FE     LOCM8  LDA $FE         ;GO BACKWD 8 BYTES
856B 38               SEC
856C E9 08            SBC #$08
856E 85 FE            STA $FE
8570 B0 02            BCS M43
8572 C6 FF            DEC $FF
8574 18        M43    CLC
8575 90 A0            BCC NEWLOC
8577 18        EXITM1 CLC
8578 60               RTS
8579 C9 47     GO1    CMP #'G'        ;GO, 1 PARM (RTRN ADDR ON STK)
857B D0 19            BNE VER1        ; ... PARM IS ADDR TO GO TO
857D 20 4D 83         JSR CRLF
8580 20 9C 8B         JSR NACCES      ;WRITE PROT MONITR RAM
8583 A2 FF            LDX #$FF        ;PUSH RETURN ADDR
8585 9A               TXS
8586 A9 7F            LDA #$7F
8588 48               PHA
8589 A9 FF            LDA #$FF
858B 48               PHA
858C AD 4B A6         LDA P3H
858F 48               PHA
8590 AD 4A A6         LDA P3L
8593 4C 08 84         JMP NR10
8596 C9 56     VER1   CMP #'V'        ;VERIFY, 1 PARM (8 BYTES, CKSUM)
8598 D0 1A            BNE JUMP1
859A AD 4A A6         LDA P3L
```

```
859D 8D 4C A6          STA P2L
85A0 18                CLC
85A1 69 07             ADC #$07
85A3 8D 4A A6          STA P3L
85A6 AD 4B A6          LDA P3H
85A9 8D 4D A6          STA P2H
85AC 69 00             ADC #0
85AE 8D 4B A6          STA P3H
85B1 4C 40 86          JMP VER2+4
85B4 C9 4A      JUMP1  CMP #'J'        ;JUMP (JUMP TABLE IN SYS RAM)
85B6 D0 1F             BNE L11B
85B8 AD 4A A6          LDA P3L
85BB C9 08             CMP #8          ;0-7 ONLY VALID
85BD B0 26             BCS JUM2
85BF 20 9C 8B          JSR NACCES      ;WRITE PROT SYS RAM
85C2 0A                ASL A
85C3 A8                TAY
85C4 A2 FF             LDX #$FF        ;INIT STK PTR
85C6 9A                TXS
85C7 A9 7F             LDA #$7F        ;PUSH COLD RETURN
85C9 48                PHA
85CA A9 FF             LDA #$FF
85CC 48                PHA
85CD B9 21 A6          LDA JTABLE+1,Y  ;GET ADDR FROM TABLE
85D0 48                PHA             ;PUSH ON STACK
85D1 B9 20 A6          LDA JTABLE,Y
85D4 4C 08 84          JMP NR10        ;LOAD UP USER REG'S AND RTI
85D7 C9 12      L11B   CMP #$12        ;LOAD KIM FMT, 1 PARM
85D9 D0 14             BNE L21B
85DB A0 00             LDY #0          ;MODE = KIM
85DD AD 4A A6   L11C   LDA P3L
85E0 C9 FF             CMP #$FF        ;ID MUST NOT BE FF
85E2 D0 02             BNE *+4
85E4 38                SEC
85E5 60         JUM2   RTS
85E6 20 08 82          JSR PSHOVE      ;FIX PARM POSITION
85E9 20 08 82   L11D   JSR PSHOVE
85EC 4C 78 8C          JMP LENTRY
85EF C9 13      L21B   CMP #$13        ;LOAD TAPE, HS FMT, 1 PARM
85F1 D0 04             BNE WPR1B
85F3 A0 80             LDY #$80        ;MODE = HS
85F5 D0 E6             BNE L11C
85F7 C9 57      WPR1B  CMP #'W'        ;WRITE PROT USER RAM
85F9 D0 1B             BNE E1PARM
85FB AD 4A A6          LDA P3L         ; FIRST DIG IS 1K ABOVE 0,
85FE 29 11             AND #$11        ; SECOND IS 2K ABOVE 0
8600 C9 08             CMP #8          ; THIRD IS 3K ABOVE 0.
8602 2A                ROL A
8603 4E 4B A6          LSR P3H
8606 2A                ROL A
8607 0A                ASL A
8608 29 0F             AND #$0F
860A 49 0F             EOR #$0F        ;0 IS PROTECT
860C 8D 01 AC          STA OR3A
860F A9 0F             LDA #$0F
8611 8D 03 AC          STA DDR3A
8614 18                CLC
8615 60                RTS
8616 4C 27 88   E1PARM JMP CALC3
8619            B2PARM =*
8619            ;
8619            ; 2 PARAMETER EXEC BLOCKS
```

```
8619                ;
8619 C9 10          STD2   CMP #$10        ;STORE DOUBLE BYTE
861B D0 12                 BNE MEM2
861D 20 A7 82              JSR P3SCR
8620 AD 4D A6              LDA P2H
8623 A0 01                 LDY #1
8625 91 FE                 STA ($FE),Y
8627 88                    DEY
8628 AD 4C A6              LDA P2L
862B 91 FE                 STA ($FE),Y
862D 18                    CLC
862E 60                    RTS
862F C9 4D          MEM2   CMP #'M'        ;CONTINUE MEM SEARCH W/OLD PTR
8631 D0 09                 BNE VER2
8633 AD 4C A6              LDA P2L
8636 8D 4E A6              STA P1L
8639 4C 08 88              JMP MEM3C
863C C9 56          VER2   CMP #'V'        ;VERIFY MEM W/CHKSUMS , 2 PARM
863E D0 48                 BNE L12B
8640 20 9C 82              JSR P2SCR
8643 20 2E 83              JSR ZERCK
8646 20 16 83       VADDR  JSR CRLFSZ
8649 A2 08                 LDX #8
864B 20 42 83       V2     JSR SPACE
864E A0 00                 LDY #0
8650 B1 FE                 LDA ($FE),Y
8652 20 DD 82              JSR CHKSAD
8655 20 FA 82              JSR OUTBYT
8658 20 B2 82              JSR INCCMP
865B 70 11                 BVS V1
865D F0 02                 BEQ *+4
865F B0 0D                 BCS V1
8661 CA                    DEX
8662 D0 E7                 BNE V2
8664 20 25 83              JSR OCMCK
8667 20 86 83              JSR INSTAT
866A 90 DA                 BCC VADDR
866C 18                    CLC
866D 60                    RTS
866E 20 BE 82       V1     JSR DECCMP
8671 E0 08                 CPX #8
8673 F0 03                 BEQ *+5
8675 E8                    INX
8676 10 F6                 BPL V1
8678 20 25 83              JSR OCMCK
867B 20 4D 83              JSR CRLF
867E 20 42 83              JSR SPACE
8681 AE 37 A6              LDX SCR7
8684 20 F4 82              JSR OUTXAH
8687 60                    RTS
8688 C9 12          L12B   CMP #$12        ;LOAD KIM FMT TAPE, 2 PARMS
868A D0 0C                 BNE SP2B
868C AD 4C A6              LDA P2L
868F C9 FF                 CMP #$FF        ;ID MUST BE FF
8691 D0 F4                 BNE L12B-1      ;ERR
8693 A0 00                 LDY #0          ;MODE = HS
8695 4C E9 85              JMP L11D
8698 C9 1C          SP2B   CMP #$1C        ;SAVE PAPER TAPE, 2 PARMS
869A D0 75                 BNE E2PARM
869C 18                    CLC
869D 20 88 81              JSR SAVER
86A0 20 9C 82              JSR P2SCR
```

```
86A3 20 FA 86    SP2C   JSR DIFFZ
86A6 B0 03              BCS SP2D
86A8 4C C4 81    SPEXIT JMP RESALL
86AB 20 4D 83    SP2D   JSR CRLF
86AE CD 58 A6           CMP MAXRC
86B1 90 05              BCC SP2E
86B3 AD 58 A6           LDA MAXRC
86B6 B0 02              BCS SP2F
86B8 69 01       SP2E   ADC #1
86BA 8D 3D A6    SP2F   STA RC
86BD A9 3B              LDA #$3B        ;SEMI COLON
86BF 20 47 8A           JSR OUTCHR
86C2 AD 3D A6           LDA RC
86C5 20 F4 86           JSR SVBYTE
86C8 A5 FF              LDA $FF
86CA 20 F4 86           JSR SVBYTE
86CD A5 FE              LDA $FE
86CF 20 F4 86           JSR SVBYTE
86D2 A0 00       MORED2 LDY #$00
86D4 B1 FE              LDA ($FE),Y
86D6 20 F4 86           JSR SVBYTE
86D9 20 86 83           JSR INSTAT      ;STOP IF KEY DEPRESSED
86DC B0 CA              BCS SPEXIT
86DE 20 B2 82           JSR INCCMP
86E1 70 C5              BVS SPEXIT
86E3 CE 3D A6           DEC RC
86E6 D0 EA              BNE MORED2
86E8 AE 37 A6           LDX SCR7
86EB AD 36 A6           LDA SCR6
86EE 20 F4 82           JSR OUTXAH
86F1 18                 CLC
86F2 90 AF              BCC SP2C
86F4 20 DD 82    SVBYTE JSR CHKSAD
86F7 4C FA 82           JMP OUTBYT
86FA 20 2E 83    DIFFZ  JSR ZERCK
86FD AD 4A A6    DIFFL  LDA P3L
8700 38                 SEC
8701 E5 FE              SBC $FE
8703 48                 PHA
8704 AD 4B A6           LDA P3H
8707 E5 FF              SBC $FF
8709 F0 04              BEQ DIFF1
870B 68                 PLA
870C A9 FF              LDA #$FF
870E 60                 RTS
870F 68          DIFF1  PLA
8710 60                 RTS
8711 4C 27 88    E2PARM JMP CALC3       ;MAY BE CALC OR EXEC
8714             B3PARM =*
8714             ;
8714             ; 3 PARAMETER COMMAND EXECUTE BLOCKS
8714             ;
8714 C9 46       FILL3  CMP #'F'        ;FILL MEM
8716 D0 21              BNE BLK3
8718 20 9C 82           JSR P2SCR
871B A9 00              LDA #0
871D 8D 52 A6           STA ERCNT       ;ZERO ERROR COUNT
8720 AD 4E A6           LDA P1L
8723 A0 00       F1     LDY #0
8725 91 FE              STA ($FE),Y
8727 D1 FE              CMP ($FE),Y     ;VERIFY
8729 F0 03              BEQ F3
```

```
872B 20 C1 87          JSR BRTT       ;INC ERCNT (UP TO FF)
872E 20 B2 82    F3    JSR INCCMP
8731 70 7C             BVS B1
8733 F0 EE             BEQ F1
8735 90 EC             BCC F1
8737 B0 76       F2    BCS B1         ;(ALWAYS)
8739 C9 42       BLK3  CMP #'B'       ;BLOCK MOVE (OVERLAP OKAY)
873B F0 03             BEQ *+5
873D 4C CD 87          JMP S13B
8740 A9 00             LDA #0
8742 8D 52 A6          STA ERCNT
8745 20 9C 82          JSR P2SCR
8748 AD 4E A6          LDA P1L
874B 85 FC             STA $FC
874D AD 4F A6          LDA P1H
8750 85 FD             STA $FD
8752 C5 FF             CMP $FF        ;WHICH DIRECTION TO MOVE?
8754 D0 06             BNE *+8
8756 A5 FC             LDA $FC
8758 C5 FE             CMP $FE
875A F0 53             BEQ B1         ;16 BITS EQUAL THEN FINISHED
875C B0 14             BCS B2         ;MOVE DEC'NG
875E 20 B7 87    BLP   JSR BMOVE      ;MOVE INC'NG
8761 E6 FC             INC $FC
8763 D0 02             BNE *+4
8765 E6 FD             INC $FD
8767 20 B2 82          JSR INCCMP
876A 70 43             BVS B1
876C F0 F0             BEQ BLP
876E 90 EE             BCC BLP
8770 B0 3D             BCS B1
8772 A5 FC       B2    LDA $FC        ;CALC VALS FOR MOVE DEC'NG
8774 18                CLC
8775 6D 4A A6          ADC P3L
8778 85 FC             STA $FC
877A A5 FD             LDA $FD
877C 6D 4B A6          ADC P3H
877F 85 FD             STA $FD
8781 38                SEC
8782 A5 FC             LDA $FC
8784 E5 FE             SBC $FE
8786 85 FC             STA $FC
8788 A5 FD             LDA $FD
878A E5 FF             SBC $FF
878C 85 FD             STA $FD
878E 20 A7 82          JSR P3SCR
8791 AD 4C A6          LDA P2L
8794 8D 4A A6          STA P3L
8797 AD 4D A6          LDA P2H
879A 8D 4B A6          STA P3H
879D 20 B7 87    BLP1  JSR BMOVE      ;MOVE DEC'NG
87A0 A5 FC             LDA $FC
87A2 D0 02             BNE *+4
87A4 C6 FD             DEC $FD
87A6 C6 FC             DEC $FC
87A8 20 BE 82          JSR DECCMP
87AB 70 02             BVS B1
87AD B0 EE             BCS BLP1
87AF AD 52 A6    B1    LDA ERCNT      ;FINISHED, TEST ERCNT
87B2 38                SEC
87B3 D0 01             BNE *+3
87B5 18                CLC
```

```
87B6 60                RTS
87B7 A0 00     BMOVE   LDY #0           ;MOVE 1 BYT + VER
87B9 B1 FE             LDA ($FE),Y
87BB 91 FC             STA ($FC),Y
87BD D1 FC             CMP ($FC),Y
87BF F0 0B             BEQ BRT
87C1 AC 52 A6  BRTT    LDY ERCNT        ;INC ERCNT, DONT PASS FF
87C4 C0 FF             CPY #$FF
87C6 F0 04             BEQ *+6
87C8 C8                INY
87C9 8C 52 A6          STY ERCNT
87CC 60        BRT     RTS
87CD C9 1D     S13B    CMP #$1D         ;SAVE KIM FMT TAPE, 3 PARMS
87CF D0 15             BNE S23B
87D1 A0 00             LDY #$0          ;MODE = KIM
87D3 AD 4E A6  S13C    LDA P1L
87D6 D0 02             BNE *+4          ;ID MUST NOT = 0
87D8 38                SEC
87D9 60                RTS
87DA C9 FF             CMP #$FF         ;ID MUST NOT = FF
87DC D0 02             BNE *+4
87DE 38        S1NG    SEC
87DF 60                RTS
87E0 20 93 82          JSR INCP3        ;USE END ADDR + 1
87E3 4C 87 8E          JMP SENTRY
87E6 C9 1E     S23B    CMP #$1E         ;SAVE HS FMT TAPE, 3 PARMS
87E8 D0 04             BNE L23P
87EA A0 80             LDY #$80         ;MODE = HS
87EC D0 E5             BNE S13C         ;(ALWAYS)
87EE C9 13     L23P    CMP #$13         ;LOAD HS, 3 PARMS
87F0 D0 0F             BNE MEM3
87F2 AD 4E A6          LDA P1L
87F5 C9 FF             CMP #$FF         ;ID MUST BE FF
87F7 D0 E5             BNE S1NG         ;ERROR RETURN
87F9 20 93 82          JSR INCP3        ;USE END ADDR + 1
87FC A0 80             LDY #$80         ;MODE = HS
87FE 4C 78 8C          JMP LENTRY
8801 C9 4D     MEM3    CMP #'M'         ;MEM 3 SEARCH - BYTE
8803 D0 22             BNE CALC3
8805 20 9C 82          JSR P2SCR
8808 AD 4E A6  MEM3C   LDA P1L
880B A0 00             LDY #0
880D D1 FE             CMP ($FE),Y
880F F0 0B             BEQ MEM3E        ;FOUND SEARCH BYTE?
8811 20 B2 82  MEM3D   JSR INCCMP       ;NO, INC BUFFER ADDR
8814 70 04             BVS MEM3EX
8816 F0 F0             BEQ MEM3C
8818 90 EE             BCC MEM3C
881A 18        MEM3EX CLC
881B 60                RTS              ;SEARCHED TO BOUND
881C 20 17 85  MEM3E   JSR NEWLOC       ;FOUND SEARCH BYTE
881F 90 05             BCC MEM3F
8821 C9 47             CMP #'G'         ;ENTERED G?
8823 F0 EC             BEQ MEM3D
8825 38                SEC
8826 60        MEM3F   RTS
8827 C9 43     CALC3   CMP #'C'         ;CALCULATE, 1, 2 OR 3 PARMS
8829 D0 26             BNE EXE3         ;RESULT = P1+P2+P3
882B 20 4D 83  C1      JSR CRLF
882E 20 42 83          JSR SPACE
8831 18                CLC
8832 AD 4E A6          LDA P1L
```

```
8835 6D 4C A6          ADC P2L
8838 A8                TAY
8839 AD 4F A6          LDA P1H
883C 6D 4D A6          ADC P2H
883F AA                TAX
8840 38                SEC
8841 98                TYA
8842 ED 4A A6          SBC P3L
8845 A8                TAY
8846 8A                TXA
8847 ED 4B A6          SBC P3H
884A AA                TAX
884B 98                TYA
884C 20 F4 82          JSR OUTXAH
884F 18                CLC
8850 60                RTS
8851 C9 45      EXE3   CMP #'E'        ;EXECUTE FROM RAM, 1-3 PARMS
8853 D0 57             BNE E3PARM
8855            ; SEE IF VECTOR ALREADY MOVED
8855 AD 62 A6          LDA INVEC+2     ;INVEC MOVED TO SCRA, SCRB
8858            ; HI BYTE OF EXEVEC MUST BE DIFFERENT FROM INVEC
8858 CD 73 A6          CMP EXEVEC+1    ;$FA, $FB USED AS RAM PTR
885B F0 15             BEQ PTRIN
885D 8D 3B A6          STA SCRA+1      ;SAVE INVEC IN SCRA,B
8860 AD 61 A6          LDA INVEC+1
8863 8D 3A A6          STA SCRA
8866 AD 72 A6          LDA EXEVEC      ;PUT ADDR OF RIN IN INVEC
8869 8D 61 A6          STA INVEC+1
886C AD 73 A6          LDA EXEVEC+1
886F 8D 62 A6          STA INVEC+2
8872 AD 4B A6   PTRIN  LDA P3H         ;INIT RAM PTR IN $FA, $FB
8875 85 FB             STA $FB
8877 AD 4A A6          LDA P3L
887A 85 FA             STA $FA
887C 18                CLC
887D 60                RTS
887E 20 88 81   RIN    JSR SAVER       ;GET INPUT FROM RAM
8881 A0 00             LDY #$0         ;RAM PTR IN $FA, $FB
8883 B1 FA             LDA ($FA),Y
8885 F0 12             BEQ RESTIV      ;IF 00 BYTE, RESTORE INVEC
8887 E6 FA             INC $FA
8889 D0 02             BNE *+4
888B E6 FB             INC $FB
888D 2C 53 A6          BIT TECHO       ;ECHO CHARS IN ?
8890 10 03             BPL *+5
8892 20 47 8A          JSR OUTCHR
8895 18                CLC
8896 4C B8 81          JMP RESXAF
8899 AD 3A A6   RESTIV LDA SCRA        ;RESTORE INVEC
889C 8D 61 A6          STA INVEC+1
889F AD 3B A6          LDA SCRA+1
88A2 8D 62 A6          STA INVEC+2
88A5 18                CLC
88A6 20 1B 8A          JSR INCHR
88A9 4C B8 81          JMP RESXAF
88AC 6C 6D A6   E3PARM JMP (URCVEC+1)  ;... ELSE UNREC CMD
88AF            ; ***
88AF            ; *** HEX KEYBOARD I/O
88AF            ; ***
88AF 20 88 81   GETKEY JSR SAVER       ;FIND KEY
88B2 20 CF 88          JSR GK
88B5 C9 FE             CMP #$FE
```

```
88B7 D0 13              BNE EXITGK
88B9 20 CF 88           JSR GK
88BC 8A                 TXA
88BD 0A                 ASL A
88BE 0A                 ASL A
88BF 0A                 ASL A
88C0 0A                 ASL A
88C1 8D 3E A6           STA SCRE
88C4 20 CF 88           JSR GK
88C7 8A                 TXA
88C8 18                 CLC
88C9 6D 3E A6           ADC SCRE
88CC 4C B8 81    EXITGK JMP RESXAF
88CF A9 00       GK     LDA #0
88D1 8D 55 A6           STA KSHFL
88D4 20 03 89    GK1    JSR IJSCNV      ;SCAN KB
88D7 F0 FB              BEQ GK1
88D9 20 2C 89           JSR LRNKEY      ;WHAT KEY IS IT?
88DC F0 F6              BEQ GK1
88DE 48                 PHA
88DF 8A                 TXA
88E0 48                 PHA
88E1 20 72 89           JSR BEEP
88E4 20 23 89    GK2    JSR KEYQ
88E7 D0 FB              BNE GK2         ;Z=1 IF KEY DOWN
88E9 20 9B 89           JSR NOBEEP      ;DELAY (DEBOUNCE) W/O BEEP
88EC 20 23 89           JSR KEYQ
88EF D0 F3              BNE GK2
88F1 68                 PLA
88F2 AA                 TAX
88F3 68                 PLA
88F4 C9 FF              CMP #$FF        ;IF SHIFT, SET FLAG + GET NEXT KEY
88F6 D0 07              BNE EXITG
88F8 A9 19              LDA #$19
88FA 8D 55 A6           STA KSHFL
88FD D0 D5              BNE GK1
88FF 60          EXITG  RTS
8900 20 C1 89    HDOUT  JSR OUTDSP      ;CHAR OUT, SCAN KB
8903 6C 70 A6    IJSCNV JMP (SCNVEC+1)
8906 A9 09       SCAND  LDA #$9         ;SCAN DISPLAY FROM DISBUF
8908 20 A5 89           JSR CONFIG
890B A2 05              LDX #5
890D A0 00       SC1    LDY #0
890F BD 40 A6           LDA DISBUF,X
8912 8C 00 A4           STY PADA
8915 8E 02 A4           STX PBDA
8918 8D 00 A4           STA PADA
891B A0 10              LDY #$10
891D 88          SC2    DEY
891E D0 FD              BNE SC2
8920 CA                 DEX
8921 10 EA              BPL SC1
8923 20 A3 89    KEYQ   JSR KSCONF      ; KEY DOWN ? (YES THEN Z=1)
8926 AD 00 A4    H8926  LDA PADA
8929 49 7F              EOR #$7F
892B 60                 RTS
892C 29 3F       LRNKEY AND #$3F        ;DETERMINE WHAT KEY IS DOWN
892E 8D 3F A6           STA SCRF
8931 A9 05              LDA #$05
8933 20 A5 89           JSR CONFIG
8936 AD 02 A4           LDA PBDA
8939 29 07              AND #$07
```

```
893B 49 07              EOR #$07
893D D0 05              BNE LK1
893F 2C 00 A4           BIT PADA
8942 30 1A              BMI NOKEY
8944 C9 04       LK1    CMP #$04
8946 90 02              BCC LK2
8948 A9 03              LDA #$03
894A 0A          LK2    ASL A
894B 0A                 ASL A
894C 0A                 ASL A
894D 0A                 ASL A
894E 0A                 ASL A
894F 0A                 ASL A
8950 18                 CLC
8951 6D 3F A6           ADC SCRF
8954 A2 19              LDX #$19
8956 DD D6 8B    LK3    CMP SYM,X
8959 F0 05              BEQ FOUND
895B CA                 DEX
895C 10 F8              BPL LK3
895E E8          NOKEY  INX
895F 60                 RTS
8960 8A          FOUND  TXA
8961 18                 CLC
8962 6D 55 A6           ADC KSHFL
8965 AA                 TAX
8966 BD EF 8B           LDA ASCII,X
8969 60                 RTS
896A 20 23 89    KYSTAT JSR KEYQ         ;KEY DOWN? RETURN IN CARRY
896D 18                 CLC
896E F0 01              BEQ *+3
8970 38                 SEC
8971 60                 RTS
8972 20 88 81    BEEP   JSR SAVER        ;DELAY (BOUNCE) W/BEEP
8975 A9 0D       BEEPP3 LDA #$0D
8977 20 A5 89    BEEPP5 JSR CONFIG
897A A2 70              LDX #$70         ;DURATION CONSTANT
897C A9 08       BE1    LDA #8
897E 8D 02 A4           STA PBDA
8981 20 95 89           JSR BE2
8984 A9 06              LDA #6
8986 8D 02 A4           STA PBDA
8989 20 95 89           JSR BE2
898C CA                 DEX
898D D0 ED              BNE BE1
898F 20 A3 89           JSR KSCONF
8992 4C C4 81           JMP RESALL
8995 A0 1A       BE2    LDY #$1A
8997 88          BE3    DEY
8998 D0 FD              BNE BE3
899A 60                 RTS
899B 20 88 81    NOBEEP JSR SAVER        ;DELAY W/O BEEP
899E A9 01              LDA #$01
89A0 4C 77 89           JMP BEEPP5       ;(BNE BEEPP5, $FF)
89A3 A9 01       KSCONF LDA #$1          ;CONFIGURE FOR KEYBOARD
89A5 20 88 81    CONFIG JSR SAVER        ;CONFIGURE I/O FROM TABLE VAL
89A8 A0 01              LDY #$01
89AA AA                 TAX
89AB BD C8 8B    CON1   LDA VALSP2,X
89AE 99 02 A4           STA PBDA,Y
89B1 BD C6 8B           LDA VALS,X
89B4 99 00 A4           STA PADA,Y
```

```
89B7 CA               DEX
89B8 88               DEY
89B9 10 F0            BPL CON1
89BB 4C C4 81         JMP RESALL
89BE 20 AF 88  HKEY   JSR GETKEY      ;GET KEY FROM KB AND ECHO ON KB
89C1 20 88 81  OUTDSP JSR SAVER       ;DISPLAY OUT
89C4 29 7F            AND #$7F
89C6 C9 07            CMP #$07        ;BELL?
89C8 D0 03            BNE NBELL
89CA 4C 75 89         JMP BEEPP3      ;YES - BEEP
89CD 20 06 8A  NBELL  JSR TEXT        ;PUSH INTO SCOPE BUFFER
89D0 C9 2C            CMP #$2C        ;COMMA?
89D2 D0 0A            BNE OUD1
89D4 AD 45 A6         LDA RDIG
89D7 09 80            ORA #$80        ;TURN ON DECIMAL PT
89D9 8D 45 A6         STA RDIG
89DC D0 25            BNE EXITOD
89DE A2 3A     OUD1   LDX #$3A
89E0 DD EE 8B  OUD2   CMP ASCIM1,X
89E3 F0 05            BEQ GETSGS
89E5 CA               DEX
89E6 D0 F8            BNE OUD2
89E8 F0 19            BEQ EXITOD
89EA BD 28 8C  GETSGS LDA SEGSM1,X    ;GET CORR SEG CODE FROM TABLE
89ED C9 F0            CMP #$F0
89EF F0 12            BEQ EXITOD
89F1 A2 00            LDX #0
89F3 48               PHA
89F4 BD 41 A6  OUD3   LDA DISBUF+1,X  ;SHOVE DOWN DISPLAY BUFFER
89F7 9D 40 A6         STA DISBUF,X
89FA E8               INX
89FB E0 05            CPX #5
89FD D0 F5            BNE OUD3
89FF 68               PLA
8A00 8D 45 A6         STA RDIG
8A03 4C C4 81  EXITOD JMP RESALL
8A06 48        TEXT   PHA             ;UPDATE SCOPE BUFFER
8A07 8A               TXA             ;SAVE X
8A08 48               PHA
8A09 A2 1E            LDX #$1E        ;PUSH DOWN 32 CHARS
8A0B BD 00 A6  TXTMOV LDA SCPBUF,X
8A0E 9D 01 A6         STA SCPBUF+1,X
8A11 CA               DEX
8A12 10 F7            BPL TXTMOV
8A14 68               PLA             ;RESTORE X
8A15 AA               TAX
8A16 68               PLA             ;RESTORE CHR
8A17 8D 00 A6         STA SCPBUF      ;STORE CHR IN EMPTY SLOT
8A1A 60               RTS
8A1B           ;
8A1B           ;***
8A1B           ;*** TERMINAL I/O
8A1B           ;***
8A1B 20 88 81  INCHR  JSR SAVER       ;INPUT CHAR
8A1E 20 41 8A         JSR INJINV
8A21 29 7F            AND #$7F        ;DROP PARITY
8A23 C9 61            CMP #$61        ;ALPHA?
8A25 90 06            BCC INRT1
8A27 C9 7B            CMP #$7B
8A29 B0 02            BCS INRT1
8A2B 29 DF            AND #$DF        ;CVRT TO UPPER CASE
8A2D C9 0F     INRT1  CMP #$0F        ;CTL O ?
```

```
8A2F D0 0B              BNE INRT2
8A31 AD 53 A6           LDA TECHO
8A34 49 40              EOR #$40        ;TOGGLE CTL O BIT
8A36 8D 53 A6           STA TECHO
8A39 18                 CLC
8A3A 90 E2              BCC INCHR+3     ;GET GET ANOTHER CHAR
8A3C C9 0D      INRT2   CMP #$0D        ;CARRIAGE RETURN?
8A3E 4C B8 81           JMP RESXAF
8A41 6C 61 A6   INJINV  JMP (INVEC+1)
8A44 20 09 83   NBASOC  JSR NIBASC      ;NIBBLE TO ASCII, OUTCHR
8A47 20 88 81   OUTCHR  JSR SAVER
8A4A 2C 53 A6           BIT TECHO       ;LOOK AT CTRL O FLAG
8A4D 70 03              BVS *+5
8A4F 20 55 8A           JSR INJOUV
8A52 4C C4 81           JMP RESALL
8A55 6C 64 A6   INJOUV  JMP (OUTVEC+1)
8A58 20 88 81   INTCHR  JSR SAVER       ;IN TERMINAL CHAR
8A5B A9 00              LDA #0
8A5D 85 F9              STA $F9
8A5F AD 02 A4   LOOK    LDA PBDA        ;FIND LEADING EDGE
8A62 2D 54 A6           AND TOUTFL
8A65 38                 SEC
8A66 E9 40              SBC #$40
8A68 90 F5              BCC LOOK
8A6A 20 E9 8A   TIN     JSR DLYH        ;TERMINAL BIT
8A6D AD 02 A4           LDA PBDA
8A70 2D 54 A6           AND TOUTFL
8A73 38                 SEC
8A74 E9 40              SBC #$40        ;OR BITS 7,7 (TTY,CRT)
8A76 2C 53 A6           BIT TECHO       ;ECHO BIT?
8A79 10 06              BPL DMY1
8A7B 20 D4 8A           JSR OUT
8A7E 4C 87 8A           JMP SAVE
8A81 A0 07      DMY1    LDY #7
8A83 88         TLP1    DEY
8A84 D0 FD              BNE TLP1
8A86 EA                 NOP
8A87 66 F9      SAVE    ROR $F9
8A89 20 E9 8A           JSR DLYH
8A8C 48                 PHA             ;TIMING
8A8D B5 00              LDA 0,X
8A8F 68                 PLA
8A90 90 D8              BCC TIN
8A92 20 E9 8A           JSR DLYH
8A95 18                 CLC
8A96 20 D4 8A           JSR OUT
8A99 A5 F9              LDA $F9
8A9B 49 FF              EOR #$FF
8A9D 4C B8 81           JMP RESXAF
8AA0 85 F9      TOUT    STA $F9         ;TERMINAL CHR OUT
8AA2 20 88 81           JSR SAVER
8AA5 20 E9 8A           JSR DLYH        ;DELAY 1/2 BIT TIME
8AA8 A9 30              LDA #$30        ;SET FOR OUTPUT
8AAA 8D 03 A4           STA PBDA+1      ;DATA DIRECTION
8AAD A5 F9              LDA $F9         ;RECOVER CHR DATA
8AAF A2 0B              LDX #$0B        ;START BIT,8DATA, 3STOPS
8AB1 49 FF              EOR #$FF        ;INVERT DATA
8AB3 38                 SEC             ;START BIT
8AB4 20 D4 8A   OUTC    JSR OUT         ;OUTPUT BIT FROM CARRY
8AB7 20 E6 8A           JSR DLYF        ;WAIT FULL BIT TIME
8ABA A0 06              LDY #$06
8ABC 88         PHAKE   DEY
```

```
8ABD D0 FD              BNE PHAKE
8ABF EA                 NOP
8AC0 4A                 LSR A
8AC1 CA                 DEX
8AC2 D0 F0              BNE OUTC
8AC4 A5 F9              LDA $F9
8AC6 C9 0D              CMP #$0D        ;CARRIAGE RETURN?
8AC8 F0 04              BEQ GOPAD       ;YES-PAD IT
8ACA C9 0A              CMP #$0A        ;PAD LINE FEED TOO
8ACC D0 03              BNE LEAVE
8ACE 20 32 8B    GOPAD  JSR PAD
8AD1 4C C4 81    LEAVE  JMP RESALL
8AD4 48          OUT    PHA             ;TERMINAL BIT OUT
8AD5 AD 02 A4           LDA PBDA
8AD8 29 0F              AND #$0F
8ADA 90 02              BCC OUTONE
8ADC 09 30              ORA #$30
8ADE 2D 54 A6    OUTONE AND TOUTFL      ;MASK OUTPUT
8AE1 8D 02 A4           STA PBDA
8AE4 68                 PLA
8AE5 60                 RTS
8AE6             ;
8AE6 20 E9 8A    DLYF   JSR DLYH        ;DELAY FULL
8AE9 08          DLYH   PHP             ;DELAY HALF
8AEA 48                 PHA
8AEB 8A                 TXA
8AEC 48                 PHA
8AED 98                 TYA
8AEE AE 51 A6           LDX SDBYT
8AF1 A0 03       DLYX   LDY #3
8AF3 88          DLYY   DEY
8AF4 D0 FD              BNE DLYY
8AF6 CA                 DEX
8AF7 D0 F8              BNE DLYX
8AF9 A8                 TAY
8AFA 68                 PLA
8AFB AA                 TAX
8AFC 68                 PLA
8AFD 28                 PLP
8AFE 60                 RTS
8AFF A9 00       BAUD   LDA #0          ;DETERMINE BAUD RATE ON PB7
8B01 A8                 TAY
8B02 AD 02 A4    SEEK   LDA PBDA
8B05 0A                 ASL A
8B06 B0 FA              BCS SEEK
8B08 20 27 8B    CLEAR  JSR INK
8B0B 90 FB              BCC CLEAR
8B0D 20 27 8B    SET    JSR INK
8B10 B0 FB              BCS SET
8B12 8C 51 A6           STY SDBYT
8B15 BD 63 8C    DEAF   LDA DECPTS,X
8B18 CD 51 A6           CMP SDBYT
8B1B B0 07              BCS AGAIN
8B1D BD 69 8C           LDA STDVAL,X    ;LOAD CLOSEST STD VALUE
8B20 8D 51 A6           STA SDBYT
8B23 60                 RTS
8B24 E8          AGAIN  INX
8B25 10 EE              BPL DEAF
8B27 C8          INK    INY
8B28 A2 1C              LDX #$1C
8B2A CA          INK1   DEX
8B2B D0 FD              BNE INK1
```

```
8B2D AD 02 A4          LDA PBDA
8B30 0A                ASL A
8B31 60                RTS
8B32 AE 50 A6   PAD    LDX PADBIT       ;PAD CARRIAGE RETURN OR LF
8B35 20 E6 8A   PAD1   JSR DLYF         ;WITH EXTRA STOP BITS
8B38 CA                DEX
8B39 D0 FA             BNE PAD1
8B3B 60                RTS
8B3C 20 A3 89   TSTAT  JSR KSCONF       ;SEE IF BREAK KEY DOWN
8B3F AD 02 A4          LDA PBDA
8B42 2D 54 A6          AND TOUTFL
8B45 38                SEC
8B46 E9 40             SBC #$40
8B48 60                RTS
8B49 FF                .DB $FF          ;NOT USED
8B4A            ; ***
8B4A            ; *** RESET - TURN OFF POR, INIT SYS RAM, ENTER MONITOR
8B4A            ; ***
8B4A            ;
8B4A A2 FF      RESET  LDX #$FF
8B4C 9A                TXS              ;INIT STACK PTR
8B4D A9 CC             LDA #$CC
8B4F 8D 0C A0          STA PCR1         ;DISABLE POR, TAPE OFF
8B52 A9 04             LDA #4
8B54 48                PHA
8B55 28                PLP              ;INIT F, DISABLE IRQ DURING DFTXFR
8B56 20 86 8B          JSR ACCESS       ;UN WRITE PROT SYS RAM
8B59 A2 5F      DFTXFR LDX #$5F         ;INIT SYS RAM (EXCPT SCPBUF)
8B5B BD A0 8F          LDA DFTBLK,X
8B5E 9D 20 A6          STA RAM,X
8B61 CA                DEX
8B62 10 F7             BPL DFTXFR+2
8B64 A9 07      NEWDEV LDA #7           ;CHANGE DEVC/BAUD RATE
8B66 20 47 8A          JSR OUTCHR       ;BEEP
8B69 20 A3 89   SWITCH JSR KSCONF       ;KEYBOARD OR TERMINAL?
8B6C 20 26 89   SWLP   JSR KEYQ+3
8B6F D0 0B             BNE MONENT
8B71 2C 02 A4          BIT PBDA
8B74 10 F6             BPL SWLP
8B76 20 B7 8B          JSR VECSW        ;SWITCH VECTORS
8B79 20 FF 8A          JSR BAUD
8B7C A2 FF      MONENT LDX #$FF         ;MONITOR ENTRY
8B7E 9A                TXS
8B7F D8                CLD
8B80 20 86 8B          JSR ACCESS       ;UNWRITE PROT MONITOR RAM
8B83 4C 03 80          JMP WARM
8B86 20 88 81   ACCESS JSR SAVER        ;UN WRITE PROT SYS RAM
8B89 AD 01 AC          LDA OR3A
8B8C 09 01             ORA #1
8B8E 8D 01 AC   ACC1   STA OR3A
8B91 AD 03 AC          LDA DDR3A
8B94 09 01             ORA #1
8B96 8D 03 AC          STA DDR3A
8B99 4C C4 81          JMP RESALL
8B9C 20 88 81   NACCES JSR SAVER        ;WRITE PROT SYS RAM
8B9F AD 01 AC          LDA OR3A
8BA2 29 FE             AND #$FE
8BA4 18                CLC
8BA5 90 E7             BCC ACC1
8BA7 20 86 8B   TTY    JSR ACCESS       ;UN WRITE PROT RAM
8BAA A9 D5             LDA #$D5         ;110 BAUD
8BAC 8D 51 A6          STA SDBYT
```

```
8BAF AD 54 A6           LDA TOUTFL
8BB2 09 40              ORA #$40
8BB4 8D 54 A6           STA TOUTFL
8BB7 20 86 8B    VECSW  JSR ACCESS      ;UN WRITE PROT RAM
8BBA A2 08              LDX #$8
8BBC BD 6F 8C    SWLP2  LDA TRMTBL,X
8BBF 9D 60 A6           STA INVEC,X
8BC2 CA                 DEX
8BC3 10 F7              BPL SWLP2
8BC5 60                 RTS
8BC6             ;
8BC6             ;***
8BC6             ;*** TABLES (I/O CONFIGURATIONS, KEY CODES, ASCII CODES)
8BC6             ;***
8BC6 00 80 08 37 VALS   .DB $00,$80,$08,$37  ;KB SENSE, A=1
8BCA 00 7F 00 30        .DB $00,$7F,$00,$30  ;KB LRN, A=5
8BCE 00 FF 00 3F        .DB $00,$FF,$00,$3F  ;SCAN DSP, A=9
8BD2 00 00 07 3F        .DB $00,$00,$07,$3F  ;BEEP, A=D
8BD6             VALSP2 =VALS+2
8BD6             SYM    =*              ;KEY CODES RETURNED BY LRNKEY
8BD6             TABLE  =*
8BD6 01                 .DB $01         ;0/U0
8BD7 41                 .DB $41         ;1/U1
8BD8 81                 .DB $81         ;2/U2
8BD9 C1                 .DB $C1         ;3/U3
8BDA 02                 .DB $02         ;4/U4
8BDB 42                 .DB $42         ;5/U5
8BDC 82                 .DB $82         ;6/U6
8BDD C2                 .DB $C2         ;7/U7
8BDE 04                 .DB $04         ;8/JMP
8BDF 44                 .DB $44         ;9/VER
8BE0 84                 .DB $84         ;A/ASCII
8BE1 C4                 .DB $C4         ;B/BLK MOV
8BE2 08                 .DB $08         ;C/CALC
8BE3 48                 .DB $48         ;D/DEP
8BE4 88                 .DB $88         ;E/EXEC
8BE5 C8                 .DB $C8         ;F/FILL
8BE6 10                 .DB $10         ;CR/SD
8BE7 50                 .DB $50         ;-/+
8BE8 90                 .DB $90         ;>/<
8BE9 D0                 .DB $D0         ;SHIFT
8BEA 20                 .DB $20         ;GO/LP
8BEB 60                 .DB $60         ;REG/SP
8BEC A0                 .DB $A0         ;MEM/WP
8BED 00                 .DB $00         ;L2/L1
8BEE 40                 .DB $40         ;S2/S1
8BEF             ASCIM1 =*-1
8BEF             ASCII  =*              ;ASCII CODES AND HASH CODES
8BEF 30                 .DB $30         ;ZERO
8BF0 31                 .DB $31         ;ONE
8BF1 32                 .DB $32         ;TWO
8BF2 33                 .DB $33         ;THREE
8BF3 34                 .DB $34         ;FOUR
8BF4 35                 .DB $35         ;FIVE
8BF5 36                 .DB $36         ;SIX
8BF6 37                 .DB $37         ;SEVEN
8BF7 38                 .DB $38         ;EIGHT
8BF8 39                 .DB $39         ;NINE
8BF9 41                 .DB $41         ;A
8BFA 42                 .DB $42         ;B
8BFB 43                 .DB $43         ;C
8BFC 44                 .DB $44         ;D
```

```
8BFD 45             .DB $45         ;E
8BFE 46             .DB $46         ;F
8BFF 0D             .DB $0D         ;CR
8C00 2D             .DB $2D         ;DASH
8C01 3E             .DB $3E         ;>
8C02 FF             .DB $FF         ;SHIFT
8C03 47             .DB $47         ;G
8C04 52             .DB $52         ;R
8C05 4D             .DB $4D         ;M
8C06 13             .DB $13         ;L2
8C07 1E             .DB $1E         ;S2
8C08        ; KB UPPER CASE
8C08 14             .DB $14         ;U0
8C09 15             .DB $15         ;U1
8C0A 16             .DB $16         ;U2
8C0B 17             .DB $17         ;U3
8C0C 18             .DB $18         ;U4
8C0D 19             .DB $19         ;U5
8C0E 1A             .DB $1A         ;U6
8C0F 1B             .DB $1B         ;U7
8C10 4A             .DB $4A         ;J
8C11 56             .DB $56         ;V
8C12 FE             .DB $FE         ;ASCII
8C13 42             .DB $42         ;B
8C14 43             .DB $43         ;C
8C15 44             .DB $44         ;D
8C16 45             .DB $45         ;E
8C17 46             .DB $46         ;F
8C18 10             .DB $10         ;SD
8C19 2B             .DB $2B         ;+
8C1A 3C             .DB $3C         ;<
8C1B 00             .DB $00         ;SHIFT
8C1C 11             .DB $11         ;LP
8C1D 1C             .DB $1C         ;SP
8C1E 57             .DB $57         ;W
8C1F 12             .DB $12         ;L1
8C20 1D             .DB $1D         ;S1
8C21 2E             .DB $2E         ;.
8C22 20             .DB $20         ;BLANK
8C23 3F             .DB $3F         ;?
8C24 50             .DB $50         ;P
8C25 07             .DB $07         ;BELL
8C26 53             .DB $53         ;S
8C27 58             .DB $58         ;X
8C28 59             .DB $59         ;Y
8C29        ; SEGMENT CODES FOR ON-BOARD DISPLAY
8C29        SEGSM1 =*-1
8C29 3F             .DB $3F         ;ZERO
8C2A 06             .DB $06         ;ONE
8C2B 5B             .DB $5B         ;TWO
8C2C 4F             .DB $4F         ;THREE
8C2D 66             .DB $66         ;FOUR
8C2E 6D             .DB $6D         ;FIVE
8C2F 7D             .DB $7D         ;SIX
8C30 07             .DB $07         ;SEVEN
8C31 7F             .DB $7F         ;EIGHT
8C32 67             .DB $67         ;NINE
8C33 77             .DB $77         ;A
8C34 7C             .DB $7C         ;B
8C35 39             .DB $39         ;C
8C36 5E             .DB $5E         ;D
8C37 79             .DB $79         ;E
```

```
8C38 71                .DB $71          ;F
8C39 F0                .DB $F0          ;CR
8C3A 40                .DB $40          ;DASH
8C3B 70                .DB $70          ;>
8C3C 00                .DB $00          ;SHIFT
8C3D 6F                .DB $6F          ;G
8C3E 50                .DB $50          ;R
8C3F 54                .DB $54          ;M
8C40 38                .DB $38          ;L2
8C41 6D                .DB $6D          ;S2
8C42 01                .DB $01          ;U0
8C43 08                .DB $08          ;U1
8C44 09                .DB $09          ;U2
8C45 30                .DB $30          ;U3
8C46 36                .DB $36          ;U4
8C47 5C                .DB $5C          ;U5
8C48 63                .DB $63          ;U6
8C49 03                .DB $03          ;U7
8C4A 1E                .DB $1E          ;J
8C4B 72                .DB $72          ;V
8C4C 77                .DB $77          ;A
8C4D 7C                .DB $7C          ;B
8C4E 39                .DB $39          ;C
8C4F 5E                .DB $5E          ;D
8C50 79                .DB $79          ;E
8C51 71                .DB $71          ;F
8C52 6D                .DB $6D          ;SD
8C53 76                .DB $76          ;+
8C54 46                .DB $46          ;<
8C55 00                .DB $00          ;SHIFT
8C56 38                .DB $38          ;LP
8C57 6D                .DB $6D          ;SP
8C58 1C                .DB $1C          ;W
8C59 38                .DB $38          ;L1
8C5A 6D                .DB $6D          ;S1
8C5B 80                .DB $80          ;.
8C5C 00                .DB $00          ;SPACE
8C5D 53                .DB $53          ;?
8C5E 73                .DB $73          ;P
8C5F 49                .DB $49          ;BELL
8C60 6D                .DB $6D          ;S
8C61 64                .DB $64          ;X
8C62 6E                .DB $6E          ;Y
8C63 973D1F100800DECPTS .DB $97,$3D,$1F,$10,$08,$00  ; TO DETERMINE BAUD RATE
8C69                   .MSFIRST
8C69 D54C24100601STDVAL .DW $D54C,$2410,$0601  ;STD VALS FOR BAUD RATES
8C6F                   .LSFIRST
8C6F            ; 110,300,600,1200,2400,4800 BAUD
8C6F 4C 58 8A   TRMTBL JMP INTCHR
8C72 4C A0 8A          JMP TOUT
8C75 4C 3C 8B          JMP TSTAT
8C78            ;
8C78
8C78            ;****** VERSION 2 4/13/79  "SY1.1"
8C78            ;****** COPYRIGHT 1978 SYNERTEK SYSTEMS CORPORATION
8C78            ;******
8C78            BDRY   =$F8             ;0/1 BDRY FOR READ TIMING
8C78            OLD    =$F9             ;HOLD PREV INPUT LEVEL IN GETTR
8C78            CHAR   =$FC             ;CHAR ASSY AND DISASSY
8C78            MODE   =$FD             ;BIT7=1 IS HS, 0 IS KIM
8C78                                    ;... BIT6=1 - IGNORE DATA
8C78            BUFADL =$FE             ;RUNNING BUFFER ADR
```

```
8C78                BUFADH =$FF
8C78                ;TAPDEL =$A630         ;HI SPEED TAPE DELAY
8C78                ;KMBDRY =$A631         ;KIM READ BDRY
8C78                ;HSBDRY =$A632         ;HS READ BDRY
8C78                ;TAPET1 =$A635         ;HS FIRST 1/2 BIT
8C78                ;TAPET2 =$A63C         ;HS SECOND 1/2 BIT
8C78                ;SCR6   =$A636         ;SCR6
8C78                ;SCR7   =$8637         ;SCR7
8C78                ;SCR8   =$A638         ;SCR8
8C78                ;SCR9   =$A639         ;SCR9
8C78
A64A                       *=$A64A
A64A                EAL    .BLOCK 1        ;P3L - END ADDR +1 (LO)
A64B                EAH    .BLOCK 1        ;P3H -  (HI)
A64C                SAL    .BLOCK 1        ;P2L - START ADDR  (LO)
A64D                SAH    .BLOCK 1        ;P2H -  (HI)
A64E                ID     .BLOCK 1        ;P1L -  ID
A64F
A64F                EOT    = $04
A64F                SYN    = $16
A64F                TPBIT  =%1000          ;BIT 3 IS ENABLE/DISABLE TO DECODER
A64F                FRAME  =$FF            ;ERROR MSG # FOR FRAME ERROR
A64F                CHECK  =$CC            ;ERROR # FOR CHECKSUM ERROR
A64F                LSTCHR =$2F            ;LAST CHAR NOT '/'
A64F                NONHEX =$FF            ;NON HEX CHAR IN KIM REC
A64F
A64F                ;ACCESS =$8BB6         ;UNRITE PROTECT SYSTEM RAM
A64F                ;P2SCR  =$829C         ;MOVE P2 TO $FF,$FE IN PAGE ZERO
A64F                ;ZERCK  =$832E         ;MOVE ZERO TO CHECK SUM
A64F                ;CONFIG =$89A5         ;CONFIGURE I/O
A64F
A64F                ; I/O - TAPE ON/OFF IS CB2 ON VIA 1 (A000)
A64F                ;       TAPE IN IS PB6 ON VIA 1 (A000)
A64F                ;       TAPE OUT IS CODE 7 TO DISPLAY DECODER, THRU 6532,
A64F                ;             PB0-PB3 (A400)
A64F
A64F                VIAACR =$A00B
A64F                VIAPCR =$A00C          ;CONTROL CB2 TAPE ON/OFF, POR
A64F                TPOUT  =$A402
A64F                TAPOUT =TPOUT
A64F                DDROUT =$A403
A64F                TAPIN  =$A000
A64F                DDRIN  =$A002
A64F                TIMER  =$A406          ;6532 TIMER READ
A64F                TIM8   =$A415          ;6532 TIMER SET (8US)
A64F                DDRDIG =$A401
A64F                DIG    =$A400
A64F
A64F                ; LOADT ENTER W/ID IN PARM 2, MODE IN [Y]
A64F
8C78                       *=$8C78
8C78 20 A9 8D       LOADT  JSR START       ;INITIALIZE
8C7B 20 52 8D       LOADT2 JSR SYNC        ;GET IN SYNC
8C7E 20 E1 8D       LOADT4 JSR RDCHTX
8C81 C9 2A                 CMP #'*'        ;START OF DATA?
8C83 F0 06                 BEQ LOAD11
8C85 C9 16                 CMP #SYN        ;NO - SYN?
8C87 D0 F2                 BNE LOADT2      ;IF NOT, RESTART SYNC SEARCH
8C89 F0 F3                 BEQ LOADT4      ;IF YES, KEEP LOOKING FOR *
8C8B
8C8B 06 FD          LOAD11 ASL MODE        ;GET MODE IN A, CLEAR BIT6
8C8D 6A                    ROR A
```

```
8C8E 85 FD                 STA MODE
8C90 20 26 8E              JSR RDBYTX      ;READ ID BYTE ON TAPE
8C93 8D 00 A4              STA DIG         ;DISPLAY ON LED (NOT DECODED)
8C96 CD 4E A6              CMP ID          ;COMPARE WITH REQUESTED ID
8C99 F0 29                 BEQ LOADT5      ;LOAD IF EQUAL
8C9B AD 4E A6              LDA ID          ;COMPARE WITH 0
8C9E C9 00                 CMP #0
8CA0 F0 22                 BEQ LOADT5      ;IF 0, LOAD ANYWAY
8CA2 C9 FF                 CMP #$FF        ;COMPARE WITH FF
8CA4 F0 07                 BEQ LOADT6      ;IF FF, USE REQUEST SA TO LOAD
8CA6
8CA6 24 FD                 BIT MODE        ;UNWANTED RECORD, KIM OR HS?
8CA8 30 16                 BMI HWRONG
8CAA 4C 7B 8C              JMP LOADT2      ;IF KIM, RESTART SEARCH
8CAD
8CAD                ; SA (&EA IF USED) COME FROM REQUEST. DISCARD TAPE VALUES
8CAD                ;   (BUFAD ALREADY SET TO SA BY 'START')
8CAD                ;
8CAD 20 74 8E       LOADT6 JSR RDCHK       ;GET SAL FROM TAPE
8CB0 20 74 8E              JSR RDCHK       ;GET SAH FROM TAPE
8CB3 24 FD                 BIT MODE        ;HS OR KIM?
8CB5 10 52                 BPL LOADT7      ;IF KIM, START READING DATA
8CB7 20 74 8E              JSR RDCHK       ;HS, GET EAH, EAL FROM TAPE
8CBA 20 74 8E              JSR RDCHK       ; ... BUT IGNORE
8CBD 4C DE 8C              JMP LT7H        ;START READING HS DATA
8CC0
8CC0                ; SA ( & EA IF USED) COME FROM TAPE. SA REPLACES BUFAD
8CC0
8CC0 A9 C0          HWRONG LDA #$C0        ;READ THRU TO GE TO NEXT REC
8CC2 85 FD                 STA MODE        ;BUT DON'T CHECK CKSUM, NO FRAME ERR
8CC4
8CC4 20 74 8E       LOADT5 JSR RDCHK       ;GET SAL FROM TAPE
8CC7 85 FE                 STA BUFADL      ;PUT IN BUF START L
8CC9 20 74 8E              JSR RDCHK       ;SAME FOR SAH
8CCC 85 FF                 STA BUFADH
8CCE                ;(SAL - H STILL HAVE REQUEST VALUE)
8CCE 24 FD                 BIT MODE        ;HS OR KIM?
8CD0 10 37                 BPL LOADT7      ;IF KIM, START READING RECORD
8CD2 20 74 8E              JSR RDCHK       ;HS. GET & SAVE EAL,EAH
8CD5 8D 4A A6              STA EAL
8CD8 20 74 8E              JSR RDCHK
8CDB 8D 4B A6              STA EAH
8CDE
8CDE                ; READ HS DATA
8CDE
8CDE 20 E5 8D       LT7H   JSR RDBYTH      ;GET NEXT BYTE
8CE1 A6 FE                 LDX BUFADL      ;CHECK FOR END OF DATA + 1
8CE3 EC 4A A6              CPX EAL
8CE6 D0 07                 BNE LT7HA
8CE8 A6 FF                 LDX BUFADH
8CEA EC 4B A6              CPX EAH
8CED F0 14                 BEQ LT7HB
8CEF 20 77 8E       LT7HA  JSR CHKT        ;NOT END, UPDATE CHECKSUM
8CF2 24 FD                 BIT MODE        ;WRONG RECORD?
8CF4 70 04                 BVS LT7HC       ;IF SO, DONT STORE BYTE
8CF6 A0 00                 LDY #0          ;STORE BYTE
8CF8 91 FE                 STA (BUFADL),Y
8CFA E6 FE          LT7HC  INC BUFADL      ;BUMP BUFFER ADDR
8CFC D0 E0                 BNE LT7H
8CFE E6 FF                 INC BUFADH      ;CARRY
8D00 4C DE 8C              JMP LT7H
8D03
```

```
8D03 C9 2F      LT7HB  CMP #'/'       ;EA, MUST BE "/"
8D05 D0 29             BNE LCERR      ;LAST CHAR NOT '/'
8D07 F0 15             BEQ LT8A       ;(ALWAYS)
8D09
8D09            ; READ KIM DATA
8D09
8D09 20 2A 8E   LOADT7 JSR RDBYT
8D0C B0 26             BCS LDT7A      ;NONHEX OR LAST CHAR
8D0E 20 77 8E          JSR CHKT       ;UPDATE CHECKSUM (PACKED BYTE)
8D11 A0 00             LDY #0         ;STORE BYTE
8D13 91 FE             STA (BUFADL),Y
8D15 E6 FE             INC BUFADL     ;BUMP BUFFER ADR
8D17 D0 F0             BNE LOADT7     ;CARRY?
8D19 E6 FF             INC BUFADH
8D1B 4C 09 8D          JMP LOADT7
8D1E
8D1E            ; TEST CHECKSUM & FINISH
8D1E
8D1E            LOADT8 =*
8D1E 20 26 8E   LT8A   JSR RDBYTX     ;CHECK SUM
8D21 CD 36 A6          CMP SCR6
8D24 D0 16             BNE CKERR
8D26 20 26 8E          JSR RDBYTX
8D29 CD 37 A6          CMP SCR7
8D2C D0 0E             BNE CKERR      ;CHECK SUM ERROR
8D2E F0 11             BEQ OKEXIT     ;(ALWAYS)
8D30
8D30 A9 2F      LCERR  LDA #LSTCHR    ;LAST CHAR IS NOT '/'
8D32 D0 0A             BNE NGEXIT     ;(ALWAYS)
8D34
8D34 C9 2F      LDT7A  CMP #'/'       ;LAST OR NONHEX?
8D36 F0 E6             BEQ LOADT8     ;LAST
8D38            FRERR                 ;FRAMING ERROR
8D38 A9 FF      NHERR  LDA #NONHEX    ;KIM ONLY, NON HEX CHAR READ
8D3A D0 02             BNE NGEXIT     ;(ALWAYS)
8D3C
8D3C A9 CC      CKERR  LDA #CHECK     ;CHECKSUM ERROR
8D3E
8D3E 38         NGEXIT SEC            ;ERROR INDICATOR TO MONITOR IS CARRY
8D3F B0 01             BCS EXIT       ;(ALWAYS)
8D41
8D41 18         OKEXIT CLC            ;NO ERROR
8D42
8D42 24 FD      EXIT   BIT MODE
8D44 50 08             BVC EX10       ;READING WRONG REC?
8D46 A0 80             LDY #$80
8D48 4C 78 8C          JMP LOADT      ;RESTART SEARCH
8D4B
8D4B 68         USRREQ PLA            ;USER REQUESTS EXIT
8D4C 68                PLA
8D4D 38                SEC
8D4E A2 CC      EX10   LDX #$CC
8D50 D0 69             BNE STCC       ;STOP TAPE, RETURN
8D52 AD 02 A0   SYNC   LDA DDRIN      ;CHANGE DATA DIRECTION
8D55 29 BF             AND #$BF
8D57 8D 02 A0          STA DDRIN
8D5A A9 00             LDA #0
8D5C 8D 0B A0          STA VIAACR
8D5F AD 31 A6          LDA KMBDRY     ;SET UP BOUNDARY
8D62 24 FD             BIT MODE
8D64 10 03             BPL SY100
8D66 AD 32 A6          LDA HSBDRY
```

```
8D69 85 F8      SY100  STA BDRY
8D6B A9 6D             LDA #$6D
8D6D 8D 00 A4          STA DIG          ;INDICATE NO SYNC ON LEDS
8D70 A5 FD             LDA MODE         ;TURN ON OUT OF SYNC MODE
8D72 09 40             ORA #$40         ;BIT6
8D74 85 FD             STA MODE
8D76 A9 7F      SYNC5  LDA #$7F         ;TEST FOR CR DOWN ON HKB
8D78 8D 01 A4          STA DDRDIG
8D7B 2C 00 A4          BIT DIG
8D7E 10 CB             BPL USRREQ       ;CR KEY DOWN - EXIT (ERRORS)
8D80 20 9F 8D          JSR SYNBIT
8D83 66 FC             ROR CHAR
8D85 A5 FC             LDA CHAR
8D87 C9 16             CMP #SYN
8D89 D0 EB             BNE SYNC5
8D8B A2 0A      SYNC10 LDX #10          ;NOW MAKE SURE CAN GET 10 SYNS
8D8D 20 E1 8D          JSR RDCHTX
8D90 C9 16             CMP #SYN
8D92 D0 E2             BNE SYNC5
8D94 CA                DEX
8D95 D0 F6             BNE SYNC10+2
8D97 8E 00 A4          STX DIG          ;TURN OFF DISPLAY
8D9A CA                DEX              ;X=$FF
8D9B 8E 01 A4          STX DDRDIG
8D9E 60                RTS
8D9F            ;SYNBIT - GET BIT IN SYN SEARCH. IF HS, ENTER WITH
8D9F            ;  TIMER STARTED BY PREV BIT, BIT RETURNED IN CARRY.
8D9F
8D9F 24 FD      SYNBIT BIT MODE         ;KIM OR HS?
8DA1 10 69             BPL RDBITK       ;KIM
8DA3 20 CA 8D          JSR GETTR        ;HS
8DA6 B0 22             BCS GETTR        ;IF SHORT, GET NEXT TRANS
8DA8 60                RTS              ;BIT IS ZERO
8DA9
8DA9 84 FD      START  STY MODE         ;MODE PARM PASSED IN [Y]
8DAB 20 86 8B          JSR ACCESS       ;FIX BASIC WARM START BUG
8DAE A9 09             LDA #9
8DB0 20 A5 89          JSR CONFIG       ;PARTIAL I/O CONFIGURATION
8DB3 20 2E 83          JSR ZERCK        ;ZERO THE CHECK SUM
8DB6 20 9C 82          JSR P2SCR        ;MOVE SA TO FE,FF IN PAGE ZERO
8DB9 A2 EC             LDX #$EC
8DBB 8E 0C A0   STCC   STX VIAPCR       ;TAPE ON
8DBE 60                RTS
8DBF
8DBF            ; GETTR - GET TRANSITION TIME FROM 6532 CLOCK
8DBF            ; DESTROYS A,Y
8DBF
8DBF A9 00      KGETTR LDA #0           ;KIM GETTR - GET FULL CYCLE
8DC1 85 F9             STA OLD          ;FORCE GETTR POLARITY
8DC3 AD 00 A0   KG100  LDA TAPIN        ;WAIT TIL INPUT LO
8DC6 29 40             AND #$40
8DC8 D0 F9             BNE KG100
8DCA
8DCA A0 FF      GETTR  LDY #$FF
8DCC AD 00 A0   NOTR   LDA TAPIN
8DCF 29 40             AND #$40
8DD1 C5 F9             CMP OLD
8DD3 F0 F7             BEQ NOTR         ;NO CHANGE
8DD5 85 F9             STA OLD
8DD7 AD 06 A4          LDA TIMER
8DDA 8C 15 A4          STY TIM8         ;RESTART CLOCK
8DDD 18                CLC
```

```
8DDE 65 F8              ADC BDRY
8DE0 60                 RTS
8DE1
8DE1 24 FD       RDCHTX BIT MODE        ;READ HS OR KIM CHARACTER
8DE3 10 7A              BPL RDCHT       ;KIM
8DE5
8DE5             ; RDBYTH - READ HS BYTE
8DE5             ; Y DESTROYED, BYTE RETURNED IN CHAR AND A
8DE5             ; TIME FROM ONE CALL TO NEXT MUST BE LESS THAN
8DE5             ;    START BIT TIME (TIMER STILL RUNNING)
8DE5
8DE5 8E 38 A6    RDBYTH STX SCR8        ;SAVE X
8DE8 A2 08              LDX #8
8DEA 20 CA 8D           JSR GETTR       ;GET START BIT TIME
8DED B0 14              BCS RDBH90      ;IF NOT 0, FRAMING ERR
8DEF 20 CA 8D    RDBH10 JSR GETTR       ;GET BIT IN CARRY
8DF2 90 04              BCC RDASSY
8DF4 20 CA 8D           JSR GETTR       ;BIT IS ONE, WAIT HALF CYC
8DF7 38                 SEC             ;MAKE SURE "1"
8DF8 66 FC       RDASSY ROR CHAR
8DFA CA                 DEX
8DFB D0 F2              BNE RDBH10
8DFD A5 FC              LDA CHAR        ;GET IN ACC
8DFF AE 38 A6    H8DFF  LDX SCR8        ;RESTORE X
8E02 60                 RTS
8E03 24 FD       RDBH90 BIT MODE        ;NO ERR IF NOT IN SYNC
8E05 70 F8              BVS RDBH90-4    ;OR READING WRONG REC
8E07 68                 PLA             ;FIX STACK
8E08 68                 PLA
8E09 4C 38 8D           JMP FRERR
8E0C
8E0C             ; RDBITK - READ KIM BIT - X,Y,A DESTROYED, BIT RETURNED IN C
8E0C
8E0C 20 BF 8D    RDBITK JSR KGETTR      ;WAIT FOR LF
8E0F B0 FB              BCS RDBITK
8E11 20 BF 8D           JSR KGETTR      ;GET SECOND
8E14 B0 F6              BCS RDBITK
8E16 A2 00              LDX #0
8E18 E8          RDB100 INX             ;COUNT LF FULL CYCLES
8E19 20 BF 8D           JSR KGETTR
8E1C 90 FA              BCC RDB100
8E1E 20 BF 8D           JSR KGETTR      ;GET SECOND
8E21 90 F5              BCC RDB100
8E23 E0 08              CPX #$08        ;GET BIT TO CARRY
8E25 60                 RTS
8E26
8E26 24 FD       RDBYTX BIT MODE        ;READ HS OR KIM BYTE
8E28 30 BB              BMI RDBYTH      ;HS
8E2A
8E2A 20 5F 8E    RDBYT  JSR RDCHT       ;READ KIM BYTE INTO CHAR AND A
8E2D C9 2F              CMP #'/'        ;READ ONE CHAR IF LAST
8E2F F0 2C              BEQ PACKT3      ;SET CARRY AND RETURN
8E31 20 3C 8E           JSR PACKT
8E34 B0 26              BCS RDRTN       ;NON HEX CHAR?
8E36 AA                 TAX             ;SAVE MSD
8E37 20 5F 8E           JSR RDCHT
8E3A 86 FC              STX CHAR        ;MOVE MSD TO CHAR
8E3C             ; AND FALL INTO PACKT AGAIN
8E3C
8E3C             ;PACKT - ASCII HEX TO 4 BITS
8E3C             ;INPUT IN A, OUTPUT IN CHAR AND A, CARRY SET = NON HEX
8E3C
```

```
8E3C C9 30       PACKT  CMP #$30        ;LT "0"?
8E3E 90 1D              BCC PACKT3
8E40 C9 47              CMP #$47        ;GT "F" ?
8E42 B0 19              BCS PACKT3
8E44 C9 40              CMP #$40        ;A-F?
8E46 F0 15              BEQ PACKT3      ;40 NOT VALID
8E48 90 03              BCC PACKT1
8E4A 18                 CLC
8E4B 69 09              ADC #9
8E4D 2A          PACKT1 ROL A           ;GET LSD INTO LEFT NIBBLE
8E4E 2A                 ROL A
8E4F 2A                 ROL A
8E50 2A                 ROL A
8E51 A0 04              LDY #4
8E53 2A          RACKT2 ROL A           ;ROTATE 1 BIT AT A TIME INTO CHAR
8E54 26 FC              ROL CHAR
8E56 88                 DEY
8E57 D0 FA              BNE RACKT2
8E59 A5 FC              LDA CHAR        ;GET INTO ACCUM ALSO
8E5B 18                 CLC             ;OK
8E5C 60          RDRTN  RTS
8E5D 38          PACKT3 SEC             ;NOT HEX
8E5E 60                 RTS
8E5F
8E5F             ; RDCHT - READ KIM CHAR
8E5F             ; PRESERVES X, RETURNS CHAR IN CHAR (W/PARITY)
8E5F             ; AND A (W/O PARITY)
8E5F
8E5F 8A          RDCHT  TXA             ;SAVE X
8E60 48                 PHA
8E61 A9 FF              LDA #$FF        ;USE A TO COUNT BITS (BY SHIFTING)
8E63 48          KBITS  PHA             ;SAVE COUNTER
8E64 20 0C 8E           JSR RDBITK
8E67 66 FC              ROR CHAR
8E69 68                 PLA
8E6A 0A                 ASL A
8E6B D0 F6              BNE KBITS       ;DO 8 BITS
8E6D 68                 PLA             ;RESTORE X
8E6E AA                 TAX
8E6F A5 FC              LDA CHAR
8E71 2A                 ROL A
8E72 4A                 LSR A           ;DROP PARITY
8E73 60                 RTS
8E74
8E74             ; RDCHK - READ ONE BYT, INCLUDE IN CKSUM
8E74
8E74 20 26 8E    RDCHK  JSR RDBYTX      ;FALL INTO CHKT
8E77
8E77             ; CHKT - UPDATE CHECK SUM FROM BYTE IN A
8E77             ; DESTROYS Y
8E77
8E77 A8          CHKT   TAY             ;SAVE ACCUM
8E78 18                 CLC
8E79 6D 36 A6           ADC SCR6
8E7C 8D 36 A6           STA SCR6
8E7F 90 03              BCC CHKT10
8E81 EE 37 A6           INC SCR7        ;BUMP HI BYTE
8E84 98          CHKT10 TYA             ;RESTORE A
8E85 60                 RTS
8E86
8E86 FF                 .DB $FF         ;NOT USED
8E87                    *=$8E87         ;KEEP OLD ENTRY POINT
```

```
8E87 20 A9 8D    DUMPT  JSR START       ;INIT VIA & CKSUM, SA TO BUFAD & START
8E8A A9 07              LDA #7          ;CODE FOR TAPE OUT
8E8C 8D 02 A4           STA TAPOUT      ;BIT 3 USED FOR HI/LO
8E8F A2 01              LDX #1          ;KIM DELAY CONSTANT (OUTER)
8E91 A4 FD              LDY MODE        ;128 KIM, 0 HS
8E93 10 03              BPL DUMPT1      ;KIM - DO 128 SYNS
8E95 AE 30 A6           LDX TAPDEL      ;HS INITIAL DELAY (OUTER)
8E98 8A          DUMPT1 TXA
8E99 48                 PHA
8E9A A9 16       DMPT1A LDA #SYN
8E9C 20 0A 8F           JSR OUTCTX
8E9F 88                 DEY
8EA0 D0 F8              BNE DMPT1A      ;INNER LOOP (HS OR KIM)
8EA2 68                 PLA
8EA3 AA                 TAX
8EA4 CA                 DEX
8EA5 D0 F1              BNE DUMPT1
8EA7 A9 2A              LDA #'*'        ;WRITE START
8EA9 20 0A 8F           JSR OUTCTX
8EAC
8EAC AD 4E A6           LDA ID          ;WRITE ID
8EAF 20 3F 8F           JSR OUTBTX
8EB2
8EB2 AD 4C A6           LDA SAL         ;WRITE SA
8EB5 20 3C 8F           JSR OUTBCX
8EB8 AD 4D A6           LDA SAH
8EBB 20 3C 8F           JSR OUTBCX
8EBE
8EBE             ;
8EBE 24 FD              BIT MODE        ;KIM OR HS
8EC0 10 0C              BPL DUMPT2
8EC2
8EC2 AD 4A A6           LDA EAL         ;HS, WRITE EA
8EC5 20 3C 8F           JSR OUTBCX
8EC8 AD 4B A6           LDA EAH
8ECB 20 3C 8F           JSR OUTBCX
8ECE
8ECE A5 FE       DUMPT2 LDA BUFADL      ;CHECK FOR LAST BYTE
8ED0 CD 4A A6           CMP EAL
8ED3 D0 25              BNE DUMPT4
8ED5 A5 FF              LDA BUFADH
8ED7 CD 4B A6           CMP EAH
8EDA D0 1E              BNE DUMPT4
8EDC
8EDC A9 2F              LDA #'/'        ;LAST, WRITE "/"
8EDE 20 0A 8F           JSR OUTCTX
8EE1 AD 36 A6           LDA SCR6        ;WRITE CHECK SUM
8EE4 20 3F 8F           JSR OUTBTX
8EE7 AD 37 A6           LDA SCR7
8EEA 20 3F 8F           JSR OUTBTX
8EED
8EED A9 04              LDA #EOT        ;WRITE TWO EOT'S
8EEF 20 3F 8F           JSR OUTBTX
8EF2 A9 04              LDA #EOT
8EF4 20 3F 8F           JSR OUTBTX
8EF7
8EF7             DT3E   =*              ;(SET "OK" MARK)
8EF7 4C 41 8D           JMP OKEXIT
8EFA
8EFA A0 00       DUMPT4 LDY #0          ;GET BYTE
8EFC B1 FE              LDA (BUFADL),Y
8EFE 20 3C 8F           JSR OUTBCX      ;WRITE IT W/CHK SUM
```

```
8F01 E6 FE              INC BUFADL      ;BUMP BUFFER ADDR
8F03 D0 C9              BNE DUMPT2
8F05 E6 FF              INC BUFADH      ;CARRY
8F07 4C CE 8E           JMP DUMPT2
8F0A 24 FD       OUTCTX BIT MODE        ;HS OR KIM?
8F0C 10 48              BPL OUTCHT      ;KIM
8F0E
8F0E             ;  OUTBTH - NO CLOCK
8F0E             ; A,X DESTROYED
8F0E             ; MUST RESIDE ON ONE PAGE - TIMING CRITICAL
8F0E A2 09       OUTBTH LDX #9          ;8 BITS + START BIT
8F10 8C 39 A6           STY SCR9
8F13 85 FC              STA CHAR
8F15 AD 02 A4           LDA TAPOUT      ;GET PREV LEVEL
8F18 46 FC       GETBIT LSR CHAR
8F1A 49 08              EOR #TPBIT
8F1C 8D 02 A4           STA TAPOUT      ;INVERT LEVEL
8F1F             ; *** HERE STARTS FIRST HALF CYCLE
8F1F AC 35 A6           LDY TAPET1
8F22 88          A416   DEY             ;TIME FOR THIS LOOP IS 5Y-1
8F23 D0 FD              BNE A416
8F25 90 12              BCC NOFLIP      ;NOFLIP IF BIT ZERO
8F27 49 08              EOR #TPBIT      ;BIT IS ONE - INVERT OUTPUT
8F29 8D 02 A4           STA TAPOUT
8F2C             ; *** END OF FIRST HALF CYCLE
8F2C AC 3C A6    B416   LDY TAPET2
8F2F 88          B416B  DEY             ;LENGTH OF LOOP IS 5Y-1
8F30 D0 FD              BNE B416B
8F32 CA                 DEX
8F33 D0 E3              BNE GETBIT      ;GET NEXT BIT (LAST IS 0 START BIT)
8F35 AC 39 A6           LDY SCR9        ; (BY 9 BIT LSR)
8F38 60                 RTS
8F39 EA          NOFLIP NOP             ;TIMING
8F3A 90 F0              BCC B416        ;(ALWAYS)
8F3C             ;
8F3C 20 77 8E    OUTBCX JSR CHKT        ;WRITE HS OR KIM BYTE & CKSUM
8F3F 24 FD       OUTBTX BIT MODE        ;WRITE HS OR KIM BYTE
8F41 30 CB              BMI OUTBTH      ;HS
8F43
8F43             ;OUTBTC - OUTPUT ONE KIM BYTE
8F43
8F43             OUTBTC =*
8F43 A8          OUTBT  TAY             ;SAVE DATA BYTE
8F44 4A                 LSR A
8F45 4A                 LSR A
8F46 4A                 LSR A
8F47 4A                 LSR A
8F48 20 4B 8F           JSR HEXOUT      ;MORE SIG DIGIT
8F4B             ; FALL INTO HEXOUT
8F4B
8F4B 29 0F       HEXOUT AND #$0F        ;CVT LSD OF [A] TO ASCII, OUTPUT
8F4D C9 0A              CMP #$0A
8F4F 18                 CLC
8F50 30 02              BMI HEX1
8F52 69 07              ADC #$07
8F54 69 30       HEX1   ADC #$30
8F56
8F56             ; OUTCHT - OUTPUT ASCII CHAR (KIM)
8F56             ; CLOCK NOT USED
8F56             ; X,Y PRESERVED
8F56             ; MUST RESIDE ON ONE PAGE - TIMING CRITICAL
8F56
```

```
8F56 8E 38 A6    OUTCHT STX SCR8        ;PRESERVE X
8F59 8C 39 A6           STY SCR9        ;DITTO Y
8F5C 85 FC              STA CHAR
8F5E A9 FF              LDA #$FF        ;USE FF W/SHIFTS TO COUNT BITS
8F60 48          KIMBIT PHA             ;SAVE BIT CTR
8F61 AD 02 A4           LDA TPOUT       ;GET CURRENT OUTPUT LEVEL
8F64 46 FC              LSR CHAR        ;GET DATA BIT IN CARRY
8F66 A2 12              LDX #18         ;ASSUME 'ONE'
8F68 B0 02              BCS HF
8F6A A2 24              LDX #36         ;BIT IS ZERO
8F6C A0 19       HF     LDY #25
8F6E 49 08              EOR #TPBIT      ;INVERT OUTPUT
8F70 8D 02 A4           STA TPOUT
8F73 88          HFP1   DEY             ;PAUSE FOR 138 USEC
8F74 D0 FD              BNE HFP1
8F76 CA                 DEX             ;COUNT HALF CYCS OF HF
8F77 D0 F3              BNE HF
8F79 A2 18              LDX #24         ;ASSUME BIT IS ONE
8F7B B0 02              BCS LF20
8F7D A2 0C              LDX #12         ;BIT IS ZERO
8F7F A0 27       LF20   LDY #39
8F81 49 08              EOR #TPBIT      ;INVERT OUTPUT
8F83 8D 02 A4           STA TPOUT
8F86 88          LFP1   DEY             ;PAUSE FOR 208 USEC
8F87 D0 FD              BNE LFP1
8F89 CA                 DEX             ;COUNT HALF CYCS
8F8A D0 F3              BNE LF20
8F8C 68                 PLA             ;RESTORE BIT CTR
8F8D 0A                 ASL A           ;DECREMENT IT
8F8E D0 D0              BNE KIMBIT      ;FF SHIFTED 8X = 0
8F90 AE 38 A6           LDX SCR8
8F93 AC 39 A6           LDY SCR9
8F96 98                 TYA             ;RESTORE DATA BYTE
8F97 60                 RTS
8F98
8F98 FF FF              .DB $FF,$FF     ;NOT USED
8F9A
8F9A             ; REGISTER NAME PATCH
8F9A                    *=$8F9A
8F9A 53                 .DB "S"
8F9B 46                 .DB "F"
8F9C 41                 .DB "A"
8F9D 58                 .DB 'X'
8F9E 59                 .DB "Y"
8F9F 01                 .DB $01
8FA0             ;
8FA0             ;
8FA0             ;***
8FA0             ;*** DEFAULT TABLE
8FA0             ;***
8FA0                    *=$8FA0
8FA0             DFTBLK =*
8FA0 00 C0              .DW $C000       ;BASIC  *** JUMP TABLE
8FA2 A7 8B              .DW TTY
8FA4 64 8B              .DW NEWDEV
8FA6 00 00              .DW $0000       ;PAGE ZERO
8FA8 00 02              .DW $0200
8FAA 00 03              .DW $0300
8FAC 00 C8              .DW $C800
8FAE 00 D0              .DW $D000
8FB0 04                 .DB $04         ;TAPE DELAY (9.0 SEC)
8FB1 2C                 .DB $2C         ;KIM TAPE BOUNDARY
```

```
8FB2 46                     .DB $46          ;HS TAPE BOUNDARY
8FB3 00 00                  .DB $00,$00      ;SCR3,SCR4
8FB5 33                     .DB $33          ;HS TAPE FIRST 1/2 BIT
8FB6 00 00                  .DB $00,$00      ;SCR6,SCR7
8FB8 00 00 00 00            .DB $00,$00,$00,$00 ;SCR8-SCRB
8FBC 5A                     .DB $5A          ;HS TAPE SECOND 1/2 BIT
8FBD 00 00 00               .DB $00,$00,$00 ;SCRD-SCRF
8FC0 00006D6E8606           .DB $00,$00,$6D,$6E,$86,$06 ;DISP BUFFER (SY1.1)
8FC6 00 00 00               .DB $00,$00,$00 ;NOT USED
8FC9 00                     .DB $00          ;PARNR
8FCA 000000000000           .DW $0000,$0000,$0000 ;PARMS
8FD0 01                     .DB $01          ;PADBIT
8FD1 4C                     .DB $4C          ;SDBYT
8FD2 00                     .DB $00          ;ERCNT
8FD3 80                     .DB $80          ;TECHO
8FD4 B0                     .DB $B0          ;TOUTFL
8FD5 00                     .DB $00          ;KSHFL
8FD6 00                     .DB $00          ;TV
8FD7 00                     .DB $00          ;LSTCOM
8FD8 10                     .DB $10          ;MAXRC
8FD9 4A 8B                  .DW RESET        ;USER REG'S
8FDB FF                     .DB $FF          ;STACK
8FDC 00                     .DB $00          ;FLAGS
8FDD 00                     .DB $00          ;A
8FDE 00                     .DB $00          ;X
8FDF 00                     .DB $00          ;Y
8FE0                 ;VECTORS
8FE0 4C BE 89               JMP HKEY         ;INVEC
8FE3 4C 00 89               JMP HDOUT        ;OUTVEC
8FE6 4C 6A 89               JMP KYSTAT       ;INSVEC
8FE9 4C D1 81               JMP M1           ;UNRECOGNIZED SYNTAX (ERROR)
8FEC 4C D1 81               JMP M1           ;UNRECOGNIZED COMMAND (ERROR)
8FEF 4C 06 89               JMP SCAND        ;SCNVEC
8FF2 7E 88                  .DW RIN          ;IN PTR FOR EXEC FROM RAM
8FF4 C0 80                  .DW TRCOFF       ;USER TRACE VECTOR
8FF6 4A 80                  .DW SVBRK        ;BRK
8FF8 29 80                  .DW SVIRQ        ;USER IRQ
8FFA 9B 80                  .DW SVNMI        ;NMI
8FFC 4A 8B                  .DW RESET        ;RESET
8FFE 0F 80                  .DW IRQBRK       ;IRQ
9000
9000                 LENTRY =$8C78
9000                 SENTRY =$8C78+$20F
9000                 RGNAM  =$8F9A           ;REGISTER NAME PATCH
9000
9000                        .END

tasm: Number of errors = 0
```

```
+---------------------------------------------------------------------------
|  TOPIC -- AIM Computer -- AIM Monitor listing
+---------------------------------------------------------------------------

0000           ;TELEMARK CROSS ASSEMBLER (TASM) http://www.halcyon.com/squakvly/
0000
0000           ;****************************************************
0000           ;****************************************************
0000           ;**                                                **
0000           ;**                              PL-PA00-JOO1A     **
0000           ;**                                                **
0000           ;**     ROCKWELL R6500 MICROCOMPUTER SYSTEM        **
0000           ;**                                                **
0000           ;**               AIM 65 MONITOR                   **
0000           ;**                                                **
0000           ;**              PROGRAM LISTING                   **
0000           ;**                                                **
0000           ;**   REVISION A                  AUG 22, 1978     **
0000           ;**                                                **
0000           ;****************************************************
0000           ;****************************************************
0000
0000           ;ROCKWELL INTERNATIONAL
0000           ;MICROELECTRONIC DEVICES
0000           ;3310 MIRALOMA AVENUE
0000           ;P. O. BOX 3669
0000           ;ANAHEIM CA U.S.A. 92803
0000
0000           ;      **************************************
0000           ;      *  USER 6522 ADDRESSES (A000-A00F)   *
0000           ;      **************************************
A000                  *=$A000
A000           UDRB   .BLOCK 1        ;DATA REG B
A001           UDRAH  .BLOCK 1        ;DATA REG A
A002           UDDRB  .BLOCK 1        ;DATA DIR REG B
A003           UDDRA  .BLOCK 1        ;DATA DIR REG A
A004           UT1L   .BLOCK 1        ;TIMER 1 COUNTER LOW
A005           UT1CH  .BLOCK 1        ;TIMER 1 COUNTER HIGH
A006           UT1LL  .BLOCK 1        ;TIMER 1 LATCH LOW
A007           UT1LH  .BLOCK 1        ;TIMER 1 LATCH HIGH
A008           UT2L   .BLOCK 1        ;TIMER 2 LATCH & COUNTER LOW
A009           UT2H   .BLOCK 1        ;TIMER 2 COUNTER HIGH
A00A           USR    .BLOCK 1        ;SHIFT REGISTER
A00B           UACR   .BLOCK 1        ;AUX CONTROL REGISTER
A00C           UPCR   .BLOCK 1        ;PERIPHERAL CONTROL REGISTER
A00D           UIFR   .BLOCK 1        ;INTERRUPT FLAG REGISTER
A00E           UIER   .BLOCK 1        ;INTERRUPT ENABLE REGISTER
A00F           UDRA   .BLOCK 1        ;DATA REGISTER A
A010
A010           ASSEM  =$D000          ;ASSEMBLER ENTRY
A010           BASIEN =$B000          ;BASIC ENTRY (COLD)
A010           BASIRE =$B003          ;BASIC ENTRY (WARM)
A010
A010           ;       MONITOR RAM
A010           ;TEXT EDITOR EQUATES (PAG 0)
A010           ;OVERLAPS TABUF2+50 (TAPE OUTPUT BUFFER $AD-$FF)
00DF                  *=$00DF
00DF           NOWLN  .BLOCK 2        ;CURRENT LINE
00E1           BOTLN  .BLOCK 2        ;LAST ACTIVE , SO FAR
00E3           TEXT   .BLOCK 2        ;LIMITS OF BUFFER (START)
```

```
00E5            END    .BLOCK 2        ;LIMITS OF BUFFER (END)
00E7            SAVE   .BLOCK 2        ;USED BY REPLACE
00E9            OLDLEN .BLOCK 1        ;ORIG LENGTH
00EA            LENGTH .BLOCK 1        ;NEW LENGTH
00EB            STRING .BLOCK 20       ;FIND STRING
00FF
0100                   *=$0100
0100            ;BREAKPOINTS AND USER I/O HANDLERS
0100            BKS    .BLOCK 8        ;BRK LOCATIONS
0108            UIN    .BLOCK 2        ;USER INPUT HANDLER (VECTOR)
010A            UOUT   .BLOCK 2        ;USER OUTPUT HANDLER (VECTOR)
010C
010C            ;UNUSED KEYS TO GO TO USER ROUTINE
010C            KEYF1  .BLOCK 3        ;USER PUTS A JMP INSTRUCTION TO...
010F            KEYF2  .BLOCK 3        ;GO TO HIS ROUTINE ON EITHER KEY..
0112            KEYF3  .BLOCK 3        ;ENTRY
0115
0115            ;EQUATES FOR DISASSEMBLER (PAG 1)
0116                   *=$0116         ;SAME AS TAPE BUFFER I/O (TABUFF)
0116            FORMA  .BLOCK 1
0117            LMNEM  .BLOCK 1
0118            RMNEM  .BLOCK 14
0126
0126            ;EQUATES FOR MNEMONIC ENTRY
0126            MOVAD  .BLOCK 8
012E            TYPE   .BLOCK 2
0130            TMASK1 =MOVAD
0130            TMASK2 =MOVAD+1
0130            CH     .BLOCK 3
0133            ADFLD  .BLOCK 20
0147            HISTM  =$A42E          ;SHARE WITH NAME & HIST
0147            BYTESM =HISTM+1
0147            TEMPX  =HISTM+3
0147            TEMPA  =HISTM+5
0147            OPCODE =HISTM+6
0147            CODFLG =HISTM+9
0147
0147            ;      **********************************
0147            ;      *  6532 ADDRESSES (A400-A7FF)    *
0147            ;      **********************************
A400                   *=$A400
A400            MONRAM *=*
A400            ;JUMP VECTORS
A400            IRQV4  .BLOCK 2        ;IRQ AFTER MONITOR (NO BRK)
A402            NMIV2  .BLOCK 2        ;NMI
A404            IRQV2  .BLOCK 2        ;IRQ
A406
A406            ;I/O DEVICES
A406            DILINK .BLOCK 2        ;DISPL LINKAGE (TO ECHO TO DISP)
A408            TSPEED .BLOCK 1        ;TAPE SPEED (C7,5B,5A)
A409            GAP    .BLOCK 1        ;TIMING GAP BETWEEN BLOCKS
A40A            ;END OF USER ALTERABLE LOCATIONS
A40A            NPUL   .BLOCK 1        ;# OF HALF PULSES...
A40B            TIMG   .BLOCK 3        ;FOR TAPE
A40E            REGF   .BLOCK 1        ;REGS FLG FOR SINGLE STEP MODE
A40F            DISFLG .BLOCK 1        ;DISASSEM FLG FOR SINGLE STEP MODE
A410            BKFLG  .BLOCK 1        ;ENABLE OR DIS BREAKPOINTS
A411            PRIFLG .BLOCK 1        ;ENABLE OR DIS PRINTER
A412            INFLG  .BLOCK 1        ;INPUT DEVICE
A413            OUTFLG .BLOCK 1        ;OUTPUT DEVICE
A414            HISTP  .BLOCK 1        ;HISTORY PTR (SINGLE STEP) (Y)
A415            CURPO2 .BLOCK 1        ;DISPLAY POINTER
```

```
A416            CURPOS .BLOCK 1        ;PRINTER POINTER
A417            CNTH30 .BLOCK 1        ;BAUD RATE &...
A418            CNTL30 .BLOCK 1        ;DELAY FOR TTY
A419            COUNT  .BLOCK 1        ;# OF LINES (0-99)
A41A            S1     .BLOCK 2        ;START ADDRESS
A41C            ADDR   .BLOCK 2        ;END ADDRESS
A41E            CKSUM  .BLOCK 2        ;CHECKSUM
A420            S2     =BKS+6          ;VERTICAL COUNT (ONLY ON DUMP)
A420
A420            ;MONITOR REGISTERS
A420            SAVPS  .BLOCK 1        ;STATUS
A421            SAVA   .BLOCK 1        ;ACCUM
A422            SAVX   .BLOCK 1        ;X REG
A423            SAVY   .BLOCK 1        ;Y REG
A424            SAVS   .BLOCK 1        ;STACK POINTER
A425            SAVPC  .BLOCK 2        ;PROGR COUNTER
A427
A427            ;WORK AREAS FOR PAGE ZERO SIMULATION
A427            ;SIMULATE LDA (NNNN),Y ,WHERE NNNN IS ABSOLUTE
A427            STIY   .BLOCK 3        ;STA NM,Y
A42A            CPIY   .BLOCK 3        ;CMP NM,Y   OR  LDA NM,Y
A42D                   .BLOCK 1        ;RTS
A42E            LDIY   =CPIY           ;LDA NM,Y
A42E
A42E            ;VARIABLES FOR TAPE
A42E            NAME   .BLOCK 6        ;FILE NAME
A434            TAPIN  .BLOCK 1        ;IN  FLG (TAPE 1 OR 2)
A435            TAPOUT .BLOCK 1        ;OUT FLG (TAPE 1 OR 2)
A436            TAPTR  .BLOCK 1        ;TAPE BUFF POINTER
A437            TAPTR2 .BLOCK 1        ;TAPE OUTPUT BUFF PTR
A438            HIST   =NAME           ;FOUR LAST ADDR + NEXT (SINGL STEP)`
A438            BLK    =$0115          ;BLOCK COUNT
A438            TABUFF =$0116          ;TAPE BUFFER (I/O)
A438            BLKO   =$0168          ;OUTPUT BLOCK COUNT
A438            TABUF2 =$00AD          ;OUTPUT BUFF WHEN ASSEMB (PAG0)
A438            DIBUFF .BLOCK 40       ;DISPLAY BUFFER
A460
A460            ;VARIABLES USED IN PRINTING
A460            IBUFM  .BLOCK 20       ;PRINTER BUFFER
A474            IDIR   .BLOCK 1        ;DIRECTION == 0=>+ , FF=>-
A475            ICOL   .BLOCK 1        ;COLUMN  LEFTMOST=0,RIGHTMOST=4
A476            IOFFST .BLOCK 1        ;OFFSET  0=LEFT DGT,1=RIGHT DGT
A477            IDOT   .BLOCK 1        ;# OF LAST DOT ENCOUNTERED
A478            IOUTL  .BLOCK 1        ;LOWER 8 OUTPUTS(8 COLS ON RIGHT)
A479            IOUTU  .BLOCK 1        ;UPPER 2 DIGITS
A47A            IBITL  .BLOCK 1        ;1 BIT MSK FOR CURRENT OUTPUT
A47B            IBITU  .BLOCK 1
A47C            IMASK  .BLOCK 1        ;MSK FOR CURRENT ROW
A47D            JUMP   .BLOCK 2        ;INDIR & ADDR OF TABL FOR CURR ROW
A47F
A47F            ;VARIABLES FOR KEYBOARD
A47F            ROLLFL .BLOCK 1        ;SAVE LAST STROBE FOR ROLLOVER
A480            KMASK  =CPIY           ;TO MASK OFF CTRL OR SHIFT
A480            STBKEY =CPIY+1         ;STROBE KEY (1-8 COLUMNS)
A480
A480            ;                I/O ASSIGNMENT
A480                   *=$A480
A480            DRA2   .BLOCK 1        ;DATA REG A
A481            DDRA2  .BLOCK 1        ;DATA DIR REG A
A482            DRB2   .BLOCK 1        ;DATA REG B
A483            DDRB2  .BLOCK 1        ;DATA DIR REG B
A484
```

```
A484          ;  WRITE EDGE DETECT CONTROL (NOT USED BECAUSE KB)
A484                 *=$A484
A484          DNPA7  .BLOCK 1        ;DISABLE PA7 INT ,NEG EDGE DET
A485          DPPA7  .BLOCK 1        ;DIS PA7 INT ,POS EDGE DETE
A486          ENPA7  .BLOCK 1        ;ENA PA7 INT ,NEG EDG DET
A487          EPPA7  .BLOCK 1        ;ENA PA7 INT ,POS EDG DET
A488
A488          ;          READ AND CLEAR INTERRUPT
A485                 *=$A485
A485          RINT   .BLOCK 1        ;BIT 7=TIMER FLG , BIT 6=PA7 FLG
A486
A486          ;             TIMER INTERRUPT
A494                 *=$A494
A494          ;WRITE COUNT TO INTERVAL TIMER
A494          ;INTERRUPT DISABLE FOR THESE ADDRS
A494          DIV1   .BLOCK 1        ;DIV BY 1 (DISABLE);ADD 8 TO ENA
A495          DIV8   .BLOCK 1        ;DIV BY 8 (DIS) ; ADD 8 TO ENA
A496          DIV64  .BLOCK 1        ;DIV BY 64 (DIS) ; ADD 8 TO ENA
A497          DI1024 .BLOCK 1        ;DIV BY 1024 (DIS) ; ADD 8 TO ENA
A498
A498          ;     *********************************************
A498          ;     *    6522 ADDRESSES (MONIT) (A800-ABFF)     *
A498          ;     *********************************************
A800                 *=$A800
A800          DRB    .BLOCK 1        ;DATA REG B
A801          DRAH   .BLOCK 1        ;DATA REG A
A802          DDRB   .BLOCK 1        ;DATA DIR REG B
A803          DDRA   .BLOCK 1        ;DATA DIR REG A
A804          T1L    .BLOCK 1        ;TIMER 1 COUNTER LOW
A805          T1CH   .BLOCK 1        ;TIMER 1 COUNTER HIGH
A806          T1LL   .BLOCK 1        ;TIMER 1 LATCH LOW
A807          T1LH   .BLOCK 1        ;TIMER 1 LATCH HIGH
A808          T2L    .BLOCK 1        ;TIMER 2 LATCH & COUNTER LOW
A809          T2H    .BLOCK 1        ;TIMER 2 COUNTER HIGH
A80A          SR     .BLOCK 1        ;SHIFT REGISTER
A80B          ACR    .BLOCK 1        ;AUX CONTROL REGISTER
A80C          PCR    .BLOCK 1        ;PERIPHERAL CONTROL REGISTER
A80D          IFR    .BLOCK 1        ;INTERRUPT FLAG REGISTER
A80E          IER    .BLOCK 1        ;INTERRUPT ENABLE REGISTER
A80F          DRA    .BLOCK 1        ;DATA REGISTER A
A810
A810          ;DEFINE I/O CONTROL FOR PCR (CA1,CA2,CB1,CB2)
A810          DATIN  =$0E            ;DATA IN CA2=1
A810          DATOUT =$0C            ;DATA OUT CA2=0
A810          PRST   =$00            ;PRINT START (CB1) ,NEG DETEC
A810          SP12   =$01            ;STROBE P1,P2  (CA1) ,POS DETEC
A810          MON    =$C0            ;MOTOR ON (CB2=0)
A810          MOFF   =$E0
A810          ;MSKS TO OBTAIN EACH INTERRUPT
A810          MPRST  =$10            ;INT FLG FOR CB1
A810          MSP12  =$02            ;INT FLG FOR CA1
A810          MT2    =$20            ;INT FLG FOR T2
A810
A810          ;DEFINE I/O CONTROL FOR ACR  (TIMERS,SR)
A810          PRTIME =1700           ; PRINTING TIME =1.7M MSEC
A810          DEBTIM =5000           ; DEBOUNCE TIME (5 MSEC)
A810          T2I    =$00            ;T2 AS ONE SHOT (PRI,KB,TTY,TAPE)
A810          T1I    =$00            ;T1 AS ONE SHOT,PB7 DIS (TAPES)
A810          T1FR   =$C0            ;T1 IN FREE RUNNING (TAPE)
A810
A810          ;         ******************************
A810          ;         *    DISPLAY   (AC00-AFFF)   *
```

```
A810            ;         ******************************
A810            ; REGISTERS FOR DISPLAY (6520)
AC00                   *=$AC00
AC00            RA     .BLOCK 1        ;REGISTER A
AC01            CRA    .BLOCK 1        ;CONTROL REG A
AC02            RB     .BLOCK 1        ;REG B
AC03            CRB    .BLOCK 1        ;CONTROL REG B
AC04
AC04            ;CHR 00-03 ENA BY $AC04-AC07
AC04            ;CHR 04-07 ENA BY $AC08-AC0B
AC04            ;CHR 08-11 ENA BY $AC10-AC13
AC04            ;CHR 12-15 ENA BY $AC20-AC23
AC04            ;CHR 16-19 ENA BY $AC40-AC43
AC04
AC04            NULLC  =$FF
AC04            CR     =$0D
AC04            LF     =$0A
AC04            ESCAPE =$1B
AC04            RUB    =$08
AC04            EQS    =$BD
AC04            ;.FILE A1
AC04
AC04            ; E=ENTER EDITOR
AC04            ; T=RE-ENTER EDITOR TO RE-EDIT SOURCE
AC04            ; R=SHOW REGISTERS
AC04            ; M=DISPLAY MEMORY
AC04            ;  =SHOW NEXT 4 ADDRESSES
AC04            ; G=GO AT CURRENT P.C. (COUNT)
AC04            ; /=ALTER CURRENT MEMORY
AC04            ; L=LOAD OBJECT
AC04            ; D=DUMP OBJECT
AC04            ; N=ASSEMBLE
AC04            ; *=ALTER P.C.
AC04            ; A=ALTER ACCUMULATOR
AC04            ; X=ALTER X REGISTER
AC04            ; Y=ALTER Y REGISTER
AC04            ; P=ALTER PROCESSOR STATUS
AC04            ; S=ALTER STACK POINTER
AC04            ; B=SET BREAK ADDR
AC04            ; ?=SHOW BREAK ADDRESSES
AC04            ; #=CLEAR BREAK ADDRESSES
AC04            ; H=SHOW TRACE HISTORY STACK
AC04            ; V=TOGGLE REGISTER PRINT WITH DIS.
AC04            ; Z=TOGGLE DISASSEMBLER TRACE
AC04            ; \=TURN ON/OFF PRINTER
AC04            ;  =ADV PAPER
AC04            ; I=MNEMONIC ENTRY
AC04            ; K=DISASSEMBLE MEMORY
AC04            ; 1=TOGGLE TAPE 1 CONTRL (ON OR OFF)
AC04            ; 2=TOGGLE TAPE 2 CONTRL
AC04            ; 3=VERIFY CKSUM FOR TAPES
AC04            ; 4=ENABLE BREAKS
AC04            ; 5=BASIC ENTRY (COLD)
AC04            ; 6=BASIC REENTRY (WARM)
AC04
AC04            ;FOLLOWING KEYS ARE UNUSED BUT 'HOOKS'
AC04            ;ARE PROVIDED IN LOCATIONS 010C-0114
AC04            ;
AC04            ; KEYF1,KEYF2,KEYF3
AC04
E000                   *=$E000
E000            ;ALL MSGS HAVE MSB=1 OF LAST CHAR TO END IT
```

```
E000 46524F4DBD  M1     .DB "FROM",EQS
E005 54 4F BD    M3     .DB "TO",EQS
E008 202A2A2A2A20M4     .DB " **** PS AA XX YY S",$D3
E00E 5053204141205858205959205 3D3
E01C 4D4F5245BF  M5     .DB "MORE",$BF
E021 4F 4E A0    M6     .DB "ON",$A0     ;"ON "
E024 4F 46 C6    M7     .DB "OF",$C6     ;"OFF"
E027 42 52 CB    M8     .DB "BR",$CB     ;"BRK"
E02A 49 4E BD    M9     .DB "IN",EQS
E02D 4F 55 54 BD M10    .DB "OUT",EQS
E031 204D454D2046M11    .DB " MEM FAIL",$A0
E037 41494CA0
E03B 205052494E54M12    .DB " PRINTER DOW",$CE
E041 455220444F57CE
E048 2053524348  TMSG0  .DB " SRCH"
E04D 20 46 BD    TMSG1  .DB " F",EQS
E050 54 BD       TMSG2  .DB "T",EQS
E052 A0 C5 D2 D2 TMSG3  .DB $A0,$C5,$D2,$D2  ;PRINT " ERROR" ,MSB=1
E056 CFD2A0A0A0A0       .DB $CF,$D2,$A0,$A0,$A0,$A0,$A0,$A0,";"
E05C A0A03B
E05F 41 BD       TMSG5  .DB "A",EQS
E061 424C4B3DA0  TMSG6  .DB "BLK=",$A0
E066 A0CCCFC1C43BTMSG7  .DB $A0,$CC,$CF,$C1,$C4,";"
E06C 454449544FD2EMSG1  .DB "EDITO",$D2 ;EDITOR MESSAGES
E072 45 4E C4    EMSG2  .DB "EN",$C4
E075
E075                    ;VECTORS COME HERE FIRST AFTER JUMP THRU FFFA-FFFF
E075 6C 02 A4    NMIV1  JMP (NMIV2)     ;NMIV2 IS A VECTOR TO NMIV3
E078 6C 04 A4    IRQV1  JMP (IRQV2)     ;IRQV2 IS A VECTOR TO IRQV3
E07B
E07B                    ;SINGLE STEP ENTRY POINT (NMI)
E07B 8D 21 A4    NMIV3  STA SAVA        ;SAVE ACCUM
E07E 68                 PLA
E07F 8D 20 A4           STA SAVPS       ;SAVE PROCESSOR STATUS
E082 D8                 CLD
E083 8E 22 A4           STX SAVX        ;SAVE X
E086 8C 23 A4           STY SAVY
E089 68                 PLA
E08A 8D 25 A4           STA SAVPC       ;PROGRAM COUNTER
E08D 68                 PLA
E08E 8D 26 A4           STA SAVPC+1
E091 BA                 TSX             ;GET STACK PTR & SAVE IT
E092 8E 24 A4           STX SAVS
E095                    ;TRACE THE ADDRESS
E095 AC 14 A4           LDY HISTP       ;GET POINTER TO HISTORY STACK
E098 AD 26 A4           LDA SAVPC+1     ;SAVE HALT ADDR IN HISTORY STACK
E09B 99 2E A4           STA HIST,Y
E09E AD 25 A4           LDA SAVPC
E0A1 99 2F A4           STA HIST+1,Y
E0A4 20 88 E6           JSR NHIS        ;UPDATE POINTER
E0A7 AD 10 A4           LDA BKFLG       ;SOFT BREAKS ON?
E0AA F0 08              BEQ NMI5        ;NO ,DONT CHCK BRKPOINT LIST
E0AC 20 6B E7           JSR CKB         ;CHECK BREAKPOINT LIST
E0AF 90 03              BCC NMI5        ;DID NOT HIT BREAKPOINT
E0B1 4C 7F E1    NMI4   JMP IRQ2        ;HIT A BREAK-TRAP TO MONITOR
E0B4 20 90 E7    NMI5   JSR DONE        ;COUNT =0 ?
E0B7 F0 F8              BEQ NMI4        ;YES,TRAP TO MONITOR
E0B9 20 07 E9           JSR RCHEK       ;CHK IF HE WANTS TO INTERR
E0BC 4C 6D E2           JMP GOBK        ;NOT DONE-RESUME EXECUTION
E0BF
E0BF                    ;POWER UP AND RESET ENTRY POINT (RST TRANSFERS HERE)
E0BF D8          RSET   CLD             ;CLEAR DEC MODE
```

```
E0C0 78                    SEI              ;DISABLE INTERRUPT
E0C1 A2 FF                 LDX #$FF         ;INIT STACK PTR
E0C3 9A                    TXS
E0C4 8E 24 A4              STX SAVS         ;ALSO INIT SAVED STACK PTR
E0C7               ;INITIALIZE 6522
E0C7 A2 0E                 LDX #14
E0C9 BD 43 E7      RS1     LDA INTAB1,X     ;PB1-PB0,PA7-PA0 FOR PRNTR
E0CC 9D 00 A8              STA DRB,X        ;PB2=TTO,PB6=TTI
E0CF CA                    DEX              ;PB4-PB5=TAPE CONTROL,PB7=DATA
E0D0 10 F7                 BPL RS1          ;PB3 =SWITCH KB/TTY
E0D2               ;INITIALIZE 6532
E0D2 A2 03                 LDX #3           ;PORTS USED FOR KB
E0D4 BD 52 E7      RS2     LDA INTAB2,X     ;PA0-PA7 AS OUTPUT
E0D7 9D 80 A4              STA DRA2,X       ;PB0-PB7 AS INPUT
E0DA CA                    DEX
E0DB 10 F7                 BPL RS2
E0DD               ;INITIALIZE MONITOR RAM (6532)
E0DD AD 56 E7              LDA INTAB3       ;CHECK IF NMIV2 HAS BEEN CHANGED
E0E0 CD 02 A4              CMP NMIV2        ;IF IT HAS THEN ASSUME A COLD
E0E3 D0 0C                 BNE RS3A         ;START AND INITIALIZE EVERYTHING
E0E5 AD 57 E7              LDA INTAB3+1
E0E8 CD 03 A4              CMP NMIV2+1
E0EB D0 04                 BNE RS3A
E0ED A2 10                 LDX #16          ;THEY ARE EQUAL ,IT'S A WARM RESET
E0EF D0 02                 BNE RS3
E0F1 A2 00         RS3A    LDX #0           ;INIT EVERYTHING (POWER UP)
E0F3 BD 56 E7      RS3     LDA INTAB3,X
E0F6 9D 02 A4              STA NMIV2,X
E0F9 E8                    INX
E0FA E0 15                 CPX #21
E0FC 90 F5                 BCC RS3
E0FE               ;INITIALIZE DISPLAY (6520)
E0FE A9 00                 LDA #0           ;SET CONTR REG FOR DATA DIR REG
E100 A2 01                 LDX #1
E102 20 13 E1              JSR SETREG
E105 A9 FF                 LDA #$FF         ;SET DATA DIR REG FOR OUTPUT
E107 CA                    DEX
E108 20 13 E1              JSR SETREG
E10B A9 04                 LDA #$04         ;SET CONTR REG FOR PORTS
E10D E8                    INX
E10E 20 13 E1              JSR SETREG
E111 D0 07                 BNE RS3B
E113 9D 00 AC      SETREG  STA RA,X
E116 9D 02 AC              STA RB,X
E119 60                    RTS
E11A 58            RS3B    CLI              ;CLEAR INTERRUPT
E11B
E11B               ;KB/TTY SWITCH TEST AND BIT RATE MEASUREMENT
E11B A9 08                 LDA #$08         ;PB3=SWITCH KB/TTY
E11D 2C 00 A8      RS4     BIT DRB          ;A^M ,PB6-> V (OVERFLOW FLG)
E120 D0 22                 BNE RS7          ;BRANCH ON KB
E122 70 F9                 BVS RS4          ;START BIT=PB6=0?
E124 A9 FF                 LDA #$FF         ;YES ,INITIALIZE TIMER T2
E126 8D 09 A8              STA T2H
E129 2C 00 A8      RS5     BIT DRB          ;END OF START BIT ?
E12C 50 FB                 BVC RS5          ;NO ,WAIT UNTIL PB6 BACK TO 1
E12E AD 09 A8              LDA T2H          ;STORE TIMING
E131 49 FF                 EOR #$FF         ;COMPLEMENT
E133 8D 17 A4              STA CNTH30
E136 AD 08 A8              LDA T2L
E139 49 FF                 EOR #$FF
E13B 20 7C FE              JSR PATCH1       ;ADJUST IT
```

```
E13E 20 13 EA    RS6    JSR CRLOW        ;CLEAR DISPLAY
E141 4C 72 FF           JMP PAT21
E144 A2 13       RS7    LDX #19          ;CLEAR HARDWARE CURSORS
E146 8A          RS8    TXA
E147 48                 PHA
E148 A9 00              LDA #0
E14A 20 7B EF           JSR OUTDD1
E14D 68                 PLA
E14E AA                 TAX
E14F CA                 DEX
E150 10 F4              BPL RS8
E152 30 EA              BMI RS6
E154
E154             ;BRK INSTR (00) OR IRQ ENTRY POINT
E154 8D 21 A4    IRQV3  STA SAVA
E157 68                 PLA
E158 48                 PHA              ;GET STATUS
E159 29 10              AND #$10         ;SEE IF 'BRK' , ISOLATE B FLG
E15B D0 06              BNE IRQ1         ;TRAP WAS CAUSED BY "BRK" INSTRUC
E15D AD 21 A4           LDA SAVA         ;TRAP CAUSED BY IRQ SO TRANSFER
E160 6C 00 A4           JMP (MONRAM)     ;CONTROL TO USER THRU VECTOR
E163             ;IS 'BRK' INSTR ,SHOW PC & DATA
E163             ;PC IS OFF BY ONE , SO ADJUST IT
E163 68          IRQ1   PLA
E164 8D 20 A4           STA SAVPS        ;SAVE PROCESSOR STATUS
E167 8E 22 A4           STX SAVX
E16A 8C 23 A4           STY SAVY
E16D D8                 CLD
E16E 68                 PLA              ;PROGR CNTR
E16F 38                 SEC              ;SUBTRACT ONE FROM RETURN ADDR
E170 E9 01              SBC #1
E172 8D 25 A4           STA SAVPC
E175 68                 PLA
E176 E9 00              SBC #0
E178 8D 26 A4           STA SAVPC+1
E17B BA                 TSX              ;GET STACK PTR & SAVE IT
E17C 8E 24 A4           STX SAVS
E17F             ;SHOW PC AND DATA
E17F 20 61 F4    IRQ2   JSR REGQ         ;SHOW NEXT INSTRUCTION & CONTINUE
E182
E182             ;THIS ROUTINE WILL GET A CHR WITH "( )" FROM
E182             ;KB/TTY & THEN WILL GO TO THE RESPECTIVE COMMAND
E182 4C 59 FF    START  JMP PAT19        ;CLEAR DEC MODE & <CR>
E185 A9 BC       STA1   LDA #'<'+$80     ;"<" CHR WITH MSB=1 FOR DISP
E187 20 7A E9           JSR OUTPUT
E18A 20 96 FE           JSR RED1         ;GET CHR & ECHO FROM KB/TTY
E18D 48                 PHA
E18E A9 3E              LDA #'>'
E190 20 7A E9           JSR OUTPUT
E193 68                 PLA              ;SCAN LIST OF CMDS FOR ENTERED CHR
E194 A2 20              LDX #MCNT        ;COUNT OF COMMANDS
E196 DD C4 E1    MCM2   CMP COMB,X       ;CHECK NEXT COMMAND IN LIST
E199 F0 11              BEQ MCM3         ;MATCH , SO PROCESS THIS COMMAND
E19B CA                 DEX
E19C 10 F8              BPL MCM2
E19E             ;IS BAD COMMAND
E19E 20 D4 E7           JSR QM
E1A1 D8          COMIN  CLD
E1A2 20 FE E8           JSR LL
E1A5 AE 24 A4           LDX SAVS
E1A8 9A                 TXS
E1A9 4C 82 E1           JMP START
```

```
E1AC              ;HAVE VALID COMMAND
E1AC 8A           MCM3   TXA             ;CONVERT TO WORD (MULT BY 2)
E1AD 0A                  ASL A           ;2 BYTES (ADDR)
E1AE AA                  TAX
E1AF BD E5 E1            LDA MONCOM,X    ;GET ADDRESS OF COMMAND PROCESSOR
E1B2 8D 7D A4            STA JUMP
E1B5 BD E6 E1            LDA MONCOM+1,X
E1B8 8D 7E A4            STA JUMP+1
E1BB 20 C1 E1            JSR JMPR        ;CMD PROCESSORS CAN EXIT WITH 'RTS'
E1BE 4C 82 E1            JMP START
E1C1 6C 7D A4     JMPR   JMP (JUMP)      ;GO TO COMMAND
E1C4
E1C4              ;VALID COMMANDS
E1C4              MCNT   =32             ;COUNT
E1C4 4554524D472FCOMB    .DB "ETRMG/LDN*AXYPS "
E1CA 4C444E2A415859505320
E1D4 423F2348565A        .DB "B?#HVZIK123456[]",$5E
E1DA 494B3132333435365B5D5E
E1E5
E1E5 39F6CFF627E2MONCOM .DW EDIT,REENTR,REG,MEM,GO
E1EB 48E261E2
E1EF A0E2E6E23BE4        .DW CHNGG,LOAD,DUMP,ASSEM,CGPC,CGA
E1F5 00D0D4E5EEE5
E1FB F2E5F6E5EAE5        .DW CGX,CGY,CGPS,CGS,NXT5,BRKA
E201 FAE50DE61BE6
E207 4DE6FEE665E6        .DW SHOW,CLRBK,SHIS,REGT,TRACE
E20D D9E6DDE6
E211 9EFB0AE7BDE6        .DW MNEENT,KDISA,TOGTA1,TOGTA2,VECKSM
E217 CBE694E6
E21B E5E600B003B0        .DW BRKK,BASIEN,BASIRE
E221              ;USER DEFINED FUNCTIONS
E221 0C010F011201        .DW KEYF1,KEYF2,KEYF3
E227
E227              ;***** R COMMAND-DISPLAY REGISTERS *****
E227 20 13 EA     REG    JSR CRLOW       ;CLEAR DISP IF KB
E22A A0 08               LDY #M4-M1      ;MESSAG & <CR>
E22C 20 AF E7            JSR KEP
E22F 20 24 EA            JSR CRCK
E232 20 3E E8     REG1   JSR BLANK
E235 A0 09               LDY #SAVPC-ADDR ;OUTPUT PGR CNTR (SAVEPC+1,SAVEPC)
E237 20 DD E2            JSR WRITAD
E23A A9 20               LDA #SAVPS      ;NOW THE OTHER 5 REGS
E23C 8D 1C A4            STA ADDR
E23F A9 A4               LDA #SAVPS/256
E241 8D 1D A4            STA ADDR+1
E244 A2 05               LDX #5          ;COUNT
E246 D0 07               BNE MEM1        ;SHARE CODE
E248
E248              ;***** M COMMAND-DISPLAY MEMORY *****
E248 20 AE EA     MEM    JSR ADDIN       ;GET START ADDDRESS IN ADDR
E24B B0 13               BCS MEM3
E24D A2 04        MEIN   LDX #4
E24F A0 00        MEM1   LDY #0
E251 20 3E E8     MEM2   JSR BLANK
E254 A9 1C               LDA #ADDR
E256 20 58 EB            JSR LDAY        ;LOAD CONTENTS OF CURR LOCATION
E259 20 46 EA            JSR NUMA        ;AND DISPLAY IT AS 2 HEX DIGITS
E25C C8                  INY
E25D CA                  DEX             ;DECR COUNTER
E25E D0 F1               BNE MEM2
E260 60           MEM3   RTS             ;GET NEXT COMMAND
E261
```

```
E261               ;***** G COMMAND-RESTART PROCESSOR *****
E261 20 37 E8    GO     JSR PSL1        ;"/"
E264 20 85 E7           JSR GCNT        ;GET COUNT
E267 20 F0 E9           JSR CRLF
E26A 4C 86 E2           JMP GOBK1       ;RESUME EXECUTION
E26D AD 0E A4    GOBK   LDA REGF        ;DISPLAY REGISTERS ?
E270 F0 06              BEQ GOBK0       ;NO,BRANCH
E272 20 32 E2           JSR REG1        ;SHOW THE SIX REG
E275 20 24 EA           JSR CRCK        ;<CR>
E278 20 07 E9    GOBK0  JSR RCHEK       ;SEE IF HE WANTS TO INTERRUPT
E27B AD 0F A4           LDA DISFLG      ;DISASSEMBLE CURRENT INSTR ?
E27E F0 06              BEQ GOBK1       ;NO,BRANCH
E280 20 6C F4           JSR DISASM      ;DISASM THIS INSTRUCTION
E283 20 13 EA           JSR CRLOW
E286 AE 24 A4    GOBK1  LDX SAVS        ;RESTORE SAVED REGS FOR RTI
E289 9A                 TXS
E28A AC 23 A4           LDY SAVY
E28D AE 22 A4           LDX SAVX
E290 AD 26 A4           LDA SAVPC+1
E293 48                 PHA             ;PUT PC ON STACK
E294 AD 25 A4           LDA SAVPC
E297 48                 PHA
E298 AD 20 A4           LDA SAVPS       ;STATUS ALSO
E29B 48                 PHA
E29C AD 21 A4           LDA SAVA
E29F 40                 RTI             ;AND AWAY WE GO...
E2A0
E2A0               ;***** / COMMAND-ALTER MEMORY *****
E2A0 20 3E E8    CHNGG  JSR BLANK
E2A3 20 DB E2           JSR WRITAZ      ;WRITE ADDR
E2A6 20 3E E8    CHNG1  JSR BLANK
E2A9 20 5D EA           JSR RD2         ;GET VALUE
E2AC 90 0A              BCC CH2         ;ISN'T SKIP OR DONE
E2AE C9 20              CMP #' '
E2B0 D0 13              BNE CH3         ;NOT BLANK SO MUST BE DONE
E2B2               ;SKIP THIS LOCATION
E2B2 20 3E E8           JSR BLANK
E2B5 4C C0 E2           JMP CH4
E2B8               ;IS ALTER
E2B8 20 78 EB    CH2    JSR SADDR       ;STORE ENTERED VALUE INTO MEMORY
E2BB F0 03              BEQ CH4         ;NO ERROR IN STORE
E2BD 4C 33 EB           JMP MEMERR      ;MEMORY WRITE ERROR
E2C0 C8          CH4    INY
E2C1 C0 04              CPY #4
E2C3 D0 E1              BNE CHNG1       ;GO AGAIN
E2C5               ;HAVE DONE LINE OR HAVE <CR>
E2C5 20 CD E2    CH3    JSR NXTADD      ;UPDATE THE ADDRESS
E2C8 A9 0D              LDA #CR         ;CLEAR DISPL
E2CA 4C E9 FE           JMP PATC10      ;ONLY ONE <CR> & BACK TO MONITOR
E2CD
E2CD 98          NXTADD TYA             ;ADD Y TO ADDR+1,ADDR
E2CE 18                 CLC
E2CF 6D 1C A4           ADC ADDR
E2D2 8D 1C A4           STA ADDR
E2D5 90 03              BCC NXTA1
E2D7 EE 1D A4           INC ADDR+1
E2DA 60          NXTA1  RTS
E2DB
E2DB               ;WRITE CURRENT VALUE OF ADDR
E2DB               ;PART OF / & SPACE COMM
E2DB A0 00       WRITAZ LDY #0
E2DD B9 1D A4    WRITAD LDA ADDR+1,Y
```

```
E2E0 BE 1C A4            LDX ADDR,Y
E2E3 4C 42 EA            JMP WRAX
E2E6
E2E6              ;***** L COMMAND-GENERAL LOAD *****
E2E6              ;LOAD OBJECT FROM TTY,USER,TYPE OR TAPE IN KIM-1 FORMAT
E2E6 20 48 E8     LOAD   JSR WHEREI      ;WHERE INPUT
E2E9              ;GET ";"  , # OF BYTES AND SA
E2E9 20 93 E9     LOAD1  JSR INALL       ;GET FIRST CHAR
E2EC C9 3B               CMP #SEMICOLON  ;LOOK FOR BEGINNING
E2EE D0 F9               BNE LOAD1       ;IGNORE ALL CHARS BEFORE ";"
E2F0 20 4D EB            JSR CLRCK       ;CLEAR CHECHSUM
E2F3 20 4B E5            JSR CHEKAR      ;READ RECORD LENGTH
E2F6 AA                  TAX             ;SAVE IN X THE # BYTES
E2F7 20 4B E5            JSR CHEKAR      ;READ UPPER HALF OF ADDRESS
E2FA 8D 1D A4            STA ADDR+1
E2FD 20 4B E5            JSR CHEKAR      ;READ LOWER HALF OF ADDRESS
E300 8D 1C A4            STA ADDR
E303 8A                  TXA
E304 F0 1B               BEQ LOAD4       ;LAST RECORD (RECORD LENGTH=0)
E306              ;GET DATA
E306 20 FD E3     LOAD2  JSR RBYTE       ;READ NEXT BYTE OF DATA
E309 20 13 E4            JSR STBYTE      ;STORE AT LOC (ADDR+1,ADDR)
E30C CA                  DEX             ;DECR RECORD LENGTH
E30D D0 F7               BNE LOAD2
E30F              ;COMPARE CKSUM
E30F 20 FD E3            JSR RBYTE       ;READ UPPER HALF OF CHCKSUM
E312 CD 1F A4            CMP CKSUM+1     ;COMPARE TO COMPUTED VALUE
E315 D0 6E               BNE CKERR       ;CKSUM ERROR
E317 20 FD E3            JSR RBYTE       ;READ LOWER HALF OF CHECKSUM
E31A CD 1E A4            CMP CKSUM
E31D D0 66               BNE CKERR
E31F F0 C8               BEQ LOAD1       ;UNTIL LAST RECORD
E321 A2 05        LOAD4  LDX #5          ;READ 4 MORE ZEROS
E323 20 FD E3     LOAD5  JSR RBYTE
E326 CA                  DEX
E327 D0 FA               BNE LOAD5
E329 20 93 E9            JSR INALL       ;READ LAST <CR>
E32C 4C 20 E5            JMP DU13        ;SET DEFAULT DEV & GO BACK
E32F
E32F              ;LOAD ROUTINE FROM TAPE BY BLOCKS
E32F              ;CHECK FOR RIGHT FILE & LOAD FIRST BLOCK
E32F A9 00        LOADTA LDA #$00        ;CLEAR BLOCK COUNT
E331 8D 15 01            STA BLK
E334 20 53 ED            JSR TIBY1       ;LOAD BUFFER WITH A BLOCK
E337 CA                  DEX             ;SET X=0
E338 8E 15 A4            STX CURPO2      ;CLEAR DISPLAY PTR
E33B BD 16 01            LDA TABUFF,X    ;BLK COUNT SHOULD BE ZERO
E33E D0 EF               BNE LOADTA      ;NO, READ ANOTHER BLOCK
E340 E8                  INX
E341              ;AFTER FIRST BLOCK OUTPUT FILE NAME
E341 EE 11 A4            INC PRIFLG      ;SO DO NOT GO TO PRINT.
E344 A0 48               LDY #TMSG0-M1   ;PRINT "F="
E346 20 AF E7            JSR KEP
E349 BD 16 01     LOAD1A LDA TABUFF,X    ;OUTPUT FILE NAME
E34C 20 7A E9            JSR OUTPUT      ;ONLY TO DISPLAY
E34F E8                  INX
E350 E0 06               CPX #6
E352 D0 F5               BNE LOAD1A
E354 20 3E E8            JSR BLANK
E357 A0 61               LDY #TMSG6-M1   ;PRINT "BLK=  "
E359 20 AF E7            JSR KEP
E35C CE 11 A4            DEC PRIFLG      ;RESTORE PRINTR FLG
```

```
E35F 20 BD ED              JSR ADDBK1      ;JUST OUTPUT BLK CNT
E362 A2 01                 LDX #1          ;RESTORE X
E364                ;CHECK IF FILE IS CORRECT
E364 BD 16 01       LOADT2 LDA TABUFF,X    ;NOW CHCK FILE NAME
E367 DD 2D A4              CMP NAME-1,X
E36A D0 C3                 BNE LOADTA      ;IF NO FILENAME GET
E36C E8                    INX             ;ANOTHER BLOCK
E36D E0 06                 CPX #6          ;FILENAME=5 CHRS
E36F D0 F3                 BNE LOADT2
E371 8E 36 A4              STX TAPTR       ;SAVE TAPE BUFF PTR
E374 EE 11 A4              INC PRIFLG      ;OUTPUT MSG ONLY TO DISPLAY
E377 A9 00                 LDA #0          ;CLEAR DISPLAY POINTER
E379 8D 15 A4              STA CURPO2
E37C A0 66                 LDY #TMSG7-M1   ;PRINT "LOAD " WITHOUT CLR DISPL
E37E 20 96 E3              JSR CKER1
E381 CE 11 A4              DEC PRIFLG
E384 60                    RTS
E385
E385                ;LINE CKSUM ERROR
E385 20 8E E3       CKERR  JSR CKER0       ;SUBR SO MNEM ENTRY CAN USE IT
E388 20 DB E2              JSR WRITAZ      ;WRITE ADDR
E38B 4C A1 E1              JMP COMIN
E38E 20 FE E8       CKER0  JSR LL          ;SET DEFAULT DEVICES
E391 20 24 EA              JSR CRCK        ;<CR>
E394 A0 52          CKER00 LDY #TMSG3-M1   ;PRINT "ERROR"
E396 B9 00 E0       CKER1  LDA M1,Y        ;DONT CLR DISPLAY TO THE RIGHT
E399 C9 3B                 CMP #SEMICOLON
E39B F0 06                 BEQ CKER2
E39D 20 7A E9              JSR OUTPUT      ;ONLY TO TERMINAL
E3A0 C8                    INY
E3A1 D0 F3                 BNE CKER1
E3A3 60             CKER2  RTS
E3A4
E3A4                ;LOAD ROUTINE FROM TAPE WITH KIM-1 FORMAT
E3A4 20 4D EB       LOADKI JSR CLRCK       ;CLEAR CKSUM
E3A7 20 EA ED       LOADK1 JSR TAISET      ;SET TAPE FOR INPUT
E3AA 20 29 EE       LOADK2 JSR GETTAP      ;READ CHARACTER FROM TAPE
E3AD C9 2A                 CMP #'*'        ;BEGINNING OF FILE?
E3AF F0 06                 BEQ LOADK3      ;YES,BRNCH
E3B1 C9 16                 CMP #$16        ;IF NOT *  SHOULD BE SYN
E3B3 D0 F2                 BNE LOADK1
E3B5 F0 F3                 BEQ LOADK2
E3B7 20 FD E3       LOADK3 JSR RBYTE       ;READ ID FROM TAPE
E3BA 8D 21 A4              STA SAVA        ;SAVE ID
E3BD                ;NOW GET ADDR TO DISPLAY
E3BD                ;& COMPARE ID AFTERWARDS
E3BD 20 4B E5              JSR CHEKAR      ;GET START ADDR LOW
E3C0 8D 1C A4              STA ADDR
E3C3 20 4B E5              JSR CHEKAR      ;GET START ADDR HIGH
E3C6 8D 1D A4              STA ADDR+1
E3C9 20 25 E4              JSR GETID       ;ID FROM HIM
E3CC CD 21 A4              CMP SAVA        ;DO IDS MATCH?
E3CF D0 D3                 BNE LOADKI      ;NO ,GET ANOTHER FILE
E3D1 A2 02          LOADK5 LDX #$02        ;GET 2 CHARS
E3D3 20 29 EE       LOADK6 JSR GETTAP      ;1 CHAR FROM TAPE
E3D6 C9 2F                 CMP #'/'        ;LAST CHAR ?
E3D8 F0 0E                 BEQ LOADK7      ;YES,BRNCH
E3DA 20 84 EA              JSR PACK        ;CONVERT TO HEX
E3DD B0 A6                 BCS CKERR       ;NOT HEX CHAR SO ERROR
E3DF CA                    DEX
E3E0 D0 F1                 BNE LOADK6
E3E2 20 13 E4              JSR STBYTE      ;STORE & CHCK MEM FAIL
```

```
E3E5 4C D1 E3          JMP LOADK5      ;NEXT
E3E8 20 FD E3   LOADK7 JSR RBYTE       ;END OF DATA CMP CKSUM
E3EB CD 1E A4          CMP CKSUM       ;LOW
E3EE D0 95             BNE CKERR
E3F0 20 FD E3          JSR RBYTE
E3F3 CD 1F A4          CMP CKSUM+1     ;HIGH
E3F6 D0 8D             BNE CKERR
E3F8 68                PLA             ;CORRECT RTN INSTEAD OF WHEREI
E3F9 68                PLA
E3FA 4C 20 E5          JMP DU13        ;TELL HIM & GO BACK TO COMMAN
E3FD
E3FD            ;GET 2 ASCII CHRS INTO 1 BYTE
E3FD            ;FOR TAPE (T) GET ONLY ONE HEX CHR
E3FD AD 12 A4   RBYTE  LDA INFLG       ;INPUT DEVICE
E400 C9 54             CMP #'T'
E402 D0 03             BNE RBYT1
E404 4C 93 E9          JMP INALL       ;ONLY ONE BYTE FOR T (INPUT DEV)
E407 20 93 E9   RBYT1  JSR INALL
E40A 20 84 EA          JSR PACK
E40D 20 93 E9          JSR INALL
E410 4C 84 EA          JMP PACK
E413
E413            ;STORE AND CHECK MEMORY FAIL
E413 20 4E E5   STBYTE JSR CHEKA       ;ADD TO CKSUM
E416 A0 00             LDY #0
E418 20 78 EB          JSR SADDR       ;STORE AND CHCK
E41B F0 03             BEQ *+5
E41D 4C 33 EB          JMP MEMERR      ;MEMORY WRITE ERROR
E420 A0 01             LDY #1          ;INC ADDR+1,ADDR BY 1
E422 4C CD E2          JMP NXTADD
E425
E425            ;GET ID FROM LAST 2 CHR OF FILENAM
E425 A2 04      GETID  LDX #4          ;SEE WHAT HE GAVE US
E427 BD 2E A4   GID1   LDA NAME,X      ;GET LAST 2 CHARS
E42A CA                DEX
E42B C9 20             CMP #' '        ;<SPACE> ?
E42D F0 F8             BEQ GID1
E42F BD 2E A4          LDA NAME,X      ;CONVERT TO BINARY
E432 20 84 EA          JSR PACK
E435 BD 2F A4          LDA NAME+1,X
E438 4C 84 EA          JMP PACK        ;ID IS IN STIY
E43B
E43B            ;***** D COMMAND-GENERAL DUMP *****
E43B            ;TO TTY,PRINTR,USER,X ,TAPE,TAKIM-1
E43B AD 10 A4   DUMP   LDA BKFLG       ;SAVE IT TO USE IT
E43E 48                PHA
E43F A9 00             LDA #00
E441 8D 10 A4          STA BKFLG
E444 20 24 EA   DU1    JSR CRCK        ;<CR>
E447 20 A3 E7   DU0    JSR FROM        ;GET START ADDR
E44A B0 FB             BCS DU0         ;IN CASE OF ERROR DO IT AGAIN
E44C 20 3E E8          JSR BLANK
E44F 20 10 F9          JSR ADDRS1      ;TRANSFER ADDR TO S1
E452 20 A7 E7   DU1B   JSR TO          ;GET END ADDR
E455 B0 FB             BCS DU1B
E457 20 13 EA          JSR CRLOW
E45A AD 10 A4          LDA BKFLG       ;EXECUTE WHEREO ONLY ONCE
E45D D0 0E             BNE DU1A
E45F 20 71 E8          JSR WHEREO      ;WHICH DEV (OUTFLG)
E462 A9 00             LDA #0
E464 8D 06 01          STA S2          ;CLEAR RECORD COUNT
E467 8D 07 01          STA S2+1
```

```
E46A EE 10 A4           INC BKFLG       ;SET FLG
E46D              ;CHCK OUTPUT DEV
E46D AD 13 A4     DU1A  LDA OUTFLG
E470 C9 4B              CMP #'K'        ;TAPE FOR KIM?
E472 D0 04              BNE *+6
E474 68                 PLA             ;PULL FLG
E475 4C 87 E5           JMP DUMPKI      ;YES, GO OUTPUT WHOLE FILE
E478 A0 01              LDY #1          ;OUTPUT ONE MORE BYTE
E47A 20 CD E2           JSR NXTADD
E47D 20 F0 E9     DU2   JSR CRLF
E480 20 07 E9           JSR RCHEK       ;SEE IF HE WANTS TO INTERRUPT
E483              ;CALCULATE # OF BYTES YET TO BE DUMPED
E483 20 4D EB           JSR CLRCK       ;CLEAR CKSUM
E486 AD 1C A4           LDA ADDR        ;END ADDRESS-CURRENT ADDRESS
E489 38                 SEC
E48A ED 1A A4           SBC S1
E48D 48                 PHA             ;# OF BYTES LOW
E48E AD 1D A4           LDA ADDR+1
E491 ED 1B A4           SBC S1+1
E494 D0 09              BNE DU6         ;# OF BYTES HIGH
E496              ;SEE IF 24 OR MORE BYTES TO GO
E496 68                 PLA             ;# BYTES HIGH WAS ZERO
E497 F0 42              BEQ DU10        ;ARE DONE
E499 C9 18              CMP #24         ;# BYTES > 24 ?
E49B 90 05              BCC DU8         ;NO ,ONLY OUTPUT REMAINING BYTES
E49D B0 01              BCS DU7         ;YES ,24 BYTES IN NEXT RECORD
E49F 68           DU6   PLA
E4A0 A9 18        DU7   LDA #24
E4A2              ;OUTPUT ";" ,# OF BYTES AND SA
E4A2 48           DU8   PHA
E4A3 20 BA E9           JSR SEMI        ;SEMICOLON
E4A6 68                 PLA
E4A7 8D 19 A4           STA COUNT       ;SAVE # OF BYTES
E4AA 20 38 E5           JSR OUTCK       ;OUTPUT # OF BYTES
E4AD AD 1B A4           LDA S1+1        ;OUTPUT ADDRESS
E4B0 20 38 E5           JSR OUTCK
E4B3 AD 1A A4           LDA S1
E4B6 20 38 E5           JSR OUTCK
E4B9              ;OUTPUT DATA
E4B9 20 31 E5     DU9   JSR OUTCKS      ;GET CHAR SPEC BY S1 (NO PAG 0)
E4BC A9 00              LDA #0          ;CLEAR DISP PTR
E4BE 8D 15 A4           STA CURPO2
E4C1 20 5D E5           JSR ADDS1       ;INCR S1+1,S1
E4C4 CE 19 A4           DEC COUNT       ;DECREMENT BYTE COUNT
E4C7 D0 F0              BNE DU9         ;NOT DONE WITH THIS RECORD
E4C9              ;OUTPUT CKSUM
E4C9 AD 1F A4           LDA CKSUM+1
E4CC 20 3B E5           JSR OUTCK1      ;WITHOUT CHEKA
E4CF AD 1E A4           LDA CKSUM
E4D2 20 3B E5           JSR OUTCK1
E4D5 20 66 E5           JSR INCS2       ;INC VERTICAL COUNT
E4D8 4C 7D E4           JMP DU2         ;NEXT RECORD
E4DB              ;ALL DONE
E4DB A0 1C        DU10  LDY #M5-M1      ;PRINT "MORE ?#
E4DD 20 70 E9           JSR KEPR        ;OUTPUT MSG AND GET AN ANSWER
E4E0 C9 59              CMP #'Y'
E4E2 D0 03              BNE *+5
E4E4 4C 44 E4           JMP DU1         ;DUMP MORE DATA
E4E7 68                 PLA             ;RESTORE FLG
E4E8 8D 10 A4           STA BKFLG
E4EB              ;OUTPUT LAST RECORD
E4EB 20 66 E5           JSR INCS2
```

```
E4EE 20 BA E9          JSR SEMI       ;OUTPUT ';'
E4F1 A2 02             LDX #2
E4F3 A9 00             LDA #0         ;OUTPUT # OF BYTES (0-LAST RECORD)
E4F5 20 3B E5          JSR OUTCK1
E4F8 AD 07 01   DU10A  LDA S2+1       ;OUTPUT RECORD COUNT
E4FB 20 3B E5          JSR OUTCK1     ;CHECKCUM IS THE SAME
E4FE AD 06 01          LDA S2
E501 20 3B E5          JSR OUTCK1
E504 CA                DEX
E505 D0 F1             BNE DU10A
E507 20 F0 E9          JSR CRLF
E50A            ;CLOSE TAPE BLOCK IF ACTIVE
E50A AD 13 A4   DU11   LDA OUTFLG
E50D C9 54             CMP #'T'
E50F D0 0F             BNE DU13       ;NO ,BRANCH
E511 AD 37 A4   DU12   LDA TAPTR2     ;TAP OUTPUT BUFF PTR
E514 C9 01             CMP #1         ;BECAUSE FIRST ONE IS BLK CNT
E516 F0 08             BEQ DU13       ;NO DATA TO WRITE
E518 A9 00             LDA #0         ;FILL REST BUFF ZEROS
E51A 20 8B F1          JSR TOBYTE     ;OUTPUT TO BUFF
E51D 4C 11 E5          JMP DU12       ;FINISH THIS BLOCK
E520 20 13 EA   DU13   JSR CRLOW
E523 18                CLC            ;ENABLE INTERR
E524 A9 00             LDA #T1I       ;T1 FROM FREE RUNNING TO 1 SHOT
E526 8D 0B A8          STA ACR
E529 A9 34      DU14   LDA #$34       ;SET BOTH TAPES ON
E52B 8D 00 A8          STA DRB
E52E 4C FE E8          JMP LL
E531
E531            ;GET CHAR SPECIFIED BY START ADDR (S1)
E531 A9 1A      OUTCKS LDA #S1
E533 A0 00             LDY #0
E535 20 58 EB          JSR LDAY
E538
E538            ;ADD TO CHECKSUM AND PRINT
E538 20 4E E5   OUTCK  JSR CHEKA      ;CHCKSUM
E53B 48         OUTCK1 PHA
E53C AD 13 A4          LDA OUTFLG     ;IF TAPE DO NOT CNVRT
E53F C9 54             CMP #'T'       ;TO TWO ASCII CHRS
E541 D0 04             BNE OUTCK2
E543 68                PLA
E544 4C 8B F1          JMP TOBYTE     ;OUTPUT TO TAP BUFF
E547 68         OUTCK2 PLA
E548 4C 46 EA          JMP NUMA       ;TWO ASCII REPRE
E54B
E54B 20 FD E3   CHEKAR JSR RBYTE      ;TWO ASCII CHR---> 1 BYTE
E54E 48         CHEKA  PHA            ;ADD TO CHECKSUM
E54F 18                CLC
E550 6D 1E A4          ADC CKSUM
E553 8D 1E A4          STA CKSUM
E556 90 03             BCC *+5
E558 EE 1F A4          INC CKSUM+1
E55B 68                PLA
E55C 60                RTS
E55D
E55D            ;ADD ONE TO START ADDR (S1)
E55D EE 1A A4   ADDS1  INC S1
E560 D0 03             BNE ADD1
E562 EE 1B A4          INC S1+1
E565 60         ADD1   RTS
E566
E566 EE 06 01   INCS2  INC S2         ;INCR VERTICAL COUNT
```

```
E569 D0 03             BNE *+5
E56B EE 07 01          INC S2+1
E56E 60                RTS
E56F
E56F            ;OPEN A FILE FOR OUTPUT TO TAPE BY BLOCKS
E56F            ;OUTPUT FILENAME GIVEN BY JSR WHEREO TO TAPE BUFF
E56F A2 00      DUMPTA LDX #0           ;INITIALIZE TAPTR
E571 8A                TXA              ;TO OUTPUT
E572 8E 68 01          STX BLKO         ;BLOCK COUNTER
E575 8E 37 A4          STX TAPTR2       ;TAP OUTPUT BUFF PTR
E578 20 8B F1          JSR TOBYTE       ;TWO START OF FILE CHRS
E57B BD 2E A4   DUMPT1 LDA NAME,X       ;OUTPUT FILENAME
E57E 20 8B F1          JSR TOBYTE
E581 E8                INX
E582 E0 05             CPX #5
E584 D0 F5             BNE DUMPT1       ;5 FILENAME CHRS ?
E586 60                RTS
E587
E587            ;DUMP ROUTINE TO TAPE WITH KIM-1 FORMAT
E587 20 1D F2   DUMPKI JSR TAOSET       ;SET TAPE FOR OUTPUT
E58A A9 2A             LDA #'*'         ;TO EITHER 1 OR 2
E58C 20 4A F2          JSR OUTTAP       ;DIRECTLY TO TAPE
E58F            ;ID FROM LAST 2 CHRS OF FILENAME
E58F 20 25 E4          JSR GETID
E592 20 3B E5          JSR OUTCK1
E595 20 4D EB          JSR CLRCK
E598            ;STARTING ADDR
E598 AD 1A A4          LDA S1
E59B 20 38 E5          JSR OUTCK        ;WITH CHCKSUM
E59E AD 1B A4          LDA S1+1
E5A1 20 38 E5          JSR OUTCK
E5A4            ;OUTPUT DATA
E5A4 20 31 E5   DUK2   JSR OUTCKS       ;OUTPUT CHR SPECIFIED BY S1+1,S1
E5A7 20 5D E5          JSR ADDS1        ;INCREM S1+1,S1
E5AA AD 1A A4          LDA S1           ;CHCK FOR LAST BYTE
E5AD CD 1C A4          CMP ADDR         ;LSB OF END ADDR
E5B0 AD 1B A4          LDA S1+1
E5B3 ED 1D A4          SBC ADDR+1
E5B6 90 EC             BCC DUK2         ;NEXT CHR
E5B8            ;NOW SEND END CHR "/"
E5B8 A9 2F             LDA #'/'
E5BA 20 4A F2          JSR OUTTAP       ;DIRECTLY TO TAPE
E5BD            ;CHECKSUM
E5BD AD 1E A4          LDA CKSUM
E5C0 20 46 EA          JSR NUMA         ;ASCII REPRES
E5C3 AD 1F A4          LDA CKSUM+1
E5C6 20 46 EA          JSR NUMA
E5C9            ;TWO EOT CHRS
E5C9 A9 04             LDA #$04
E5CB 20 4A F2          JSR OUTTAP
E5CE 20 4A F2          JSR OUTTAP
E5D1            ;TURN TAPES ON
E5D1 4C 20 E5          JMP DU13
E5D4
E5D4            ;***** * COMMAND-ALTER PROGRAM COUNTER *****
E5D4 20 AE EA   CGPC   JSR ADDIN        ;ADDR <=ADDRESS ENTERED FROM KB
E5D7 20 DD E5   CGPC0  JSR CGPC1        ;TRANSFER ADDR TO SAVPC
E5DA 4C 13 EA          JMP CRLOW
E5DD AD 1D A4   CGPC1  LDA ADDR+1       ;THIS WAY MNEMONICS CAN USE IT
E5E0 8D 26 A4          STA SAVPC+1
E5E3 AD 1C A4          LDA ADDR
E5E6 8D 25 A4          STA SAVPC
```

```
E5E9 60                    RTS
E5EA
E5EA              ;***** P COMMAND-ALTER PROCESSOR STATUS *****
E5EA A2 00        CGPS     LDX #0
E5EC F0 0E                 BEQ CGALL
E5EE
E5EE              ;***** A COMMAND-ALTER ACCUMULATOR *****
E5EE A2 01        CGA      LDX #1
E5F0 D0 0A                 BNE CGALL
E5F2
E5F2              ;***** X COMMAND-ALTER X REGISTER *****
E5F2 A2 02        CGX      LDX #2
E5F4 D0 06                 BNE CGALL
E5F6
E5F6              ;***** Y COMMAND-ALTER Y REGISTER *****
E5F6 A2 03        CGY      LDX #3
E5F8 D0 02                 BNE CGALL
E5FA
E5FA              ;***** S COMMAND-ALTER STACK POINTER *****
E5FA A2 04        CGS      LDX #4
E5FC 20 D8 E7     CGALL    JSR EQUAL        ;PRINT PROMPT
E5FF 20 5D EA              JSR RD2          ;GET VALUE FROM KEYBOARD
E602 B0 04                 BCS GOERR
E604 9D 20 A4              STA SAVPS,X
E607 60                    RTS
E608 20 D4 E7     GOERR    JSR QM
E60B D0 EF                 BNE CGALL
E60D
E60D              ;***** <SPACE> COMMAND-SHOW NEXT 5 MEMORY LOC *****
E60D 20 3E E8     NXT5     JSR BLANK
E610 A0 04                 LDY #4           ;UPDATE ADDR FROM
E612 20 CD E2              JSR NXTADD       ;<M>=XXXX
E615 20 DB E2              JSR WRITAZ       ;OUTPUT ADDRESS
E618 4C 4D E2              JMP MEIN         ;DISPLAY CONTENTS OF NEXT 4 LOCS
E61B
E61B              ;***** B COMMAND-SET BREAKPOINT ADDR *****
E61B A0 27        BRKA     LDY #M8-M1       ;PRINT "BRK"
E61D 20 AF E7              JSR KEP
E620 20 37 E8     BRK1     JSR PSL1         ;PRINT "/"
E623 20 73 E9              JSR REDOUT       ;GET BREAK NUMBER
E626 38                    SEC
E627 E9 30                 SBC #'0'         ;0 THRU 3
E629 30 04                 BMI BKERR        ;CHARACTER < '0' -ILLEGAL
E62B C9 04                 CMP #4           ;FOUR BRK POINTS
E62D 30 05                 BMI BKOK         ;0 < CHARACTER < 4 -OK
E62F 20 D4 E7     BKERR    JSR QM           ;ERROR
E632 D0 EC                 BNE BRK1         ;ALLOW REENTRY OF BREAK NUMBER
E634 0A           BKOK     ASL A            ;*2 TO FORM WORD OFFSET
E635 48                    PHA              ;SAVE IT
E636 20 AE EA              JSR ADDIN        ;GET ADDRESS FOR BREAKPOINT
E639 68                    PLA
E63A B0 10                 BCS BKO2         ;BAD ADDRESS ENTERED
E63C 20 3D FF              JSR PATC18       ;<CR> & CLR BUFFERS
E63F AA                    TAX              ;# OF BRK
E640 AD 1C A4              LDA ADDR         ;STORE ENTERED ADDR IN BRKPT LIST
E643 9D 00 01              STA BKS,X
E646 AD 1D A4              LDA ADDR+1
E649 9D 01 01              STA BKS+1,X
E64C 60           BKO2     RTS              ;ALL DONE
E64D
E64D              ;***** ? COMMAND-SHOW CURRENT BREAKPOINTS *****
E64D A0 00        SHOW     LDY #0
```

```
E64F 20 13 EA           JSR CRLOW
E652 20 3E E8    SH1    JSR BLANK
E655 BE 00 01           LDX BKS,Y       ;ADDRESS OF NEXT BREAKPOINT
E658 B9 01 01           LDA BKS+1,Y
E65B 20 42 EA           JSR WRAX        ;SHOW BREAKPOINT ADDRESS
E65E C8                 INY
E65F C8                 INY
E660 C0 08              CPY #8
E662 D0 EE              BNE SH1
E664 60                 RTS
E665
E665             ;***** H COMMAND-SHOW TRACE STACK HISTORY *****
E665             ;LAST FIVE INSTR ADDRS
E665 A2 05       SHIS   LDX #5          ;NUMBER OF ENTRIES
E667 8E 29 A4           STX STIY+2
E66A AC 14 A4    SH11   LDY HISTP       ;POINTER TO LATEST ENTRY
E66D 20 13 EA           JSR CRLOW
E670 20 3E E8           JSR BLANK
E673 B9 2E A4           LDA HIST,Y      ;OUTPUT ADDRESS OF ENTRY
E676 20 46 EA           JSR NUMA
E679 B9 2F A4           LDA HIST+1,Y
E67C 20 46 EA           JSR NUMA
E67F 20 88 E6           JSR NHIS        ;UPDATE POINTER
E682 CE 29 A4           DEC STIY+2
E685 D0 E3              BNE SH11
E687 60                 RTS
E688
E688             ;UPDATE HISTORY POINTER (PART OF H)
E688 C8          NHIS   INY
E689 C8                 INY
E68A C0 0A              CPY #10
E68C D0 02              BNE NH1
E68E A0 00              LDY #0          ;WRAPAROUND AT 10
E690 8C 14 A4    NH1    STY HISTP
E693 60                 RTS
E694
E694             ;***** 3 COMMAND-VERIFY TAPES *****
E694             ;VERIFY CKSUM OF BLOCKS
E694 20 48 E8    VECKSM JSR WHEREI      ;GET THE FILE
E697 20 93 E9           JSR INALL       ;CHCK OBJ OR SOURCE
E69A C9 0D              CMP #CR         ;FIRST CHR IS <CR> IF OBJ
E69C D0 0E              BNE VECK2       ;ASSUME SOURCE CODE
E69E 20 93 E9    VECK1  JSR INALL       ;OBJECT FILE
E6A1 C9 3B              CMP #SEMICOLON
E6A3 D0 F9              BNE VECK1       ;IGNORE ALL CHARS BEFORE ';'
E6A5 20 93 E9           JSR INALL
E6A8 4C 60 FF           JMP PAT20
E6AB EA                 NOP
E6AC 20 93 E9    VECK2  JSR INALL       ;IT IS TEXT
E6AF C9 0D              CMP #CR
E6B1 D0 F9              BNE VECK2
E6B3 20 93 E9           JSR INALL       ;NEED TO <CR> TO FINISH
E6B6 C9 0D              CMP #CR
E6B8 D0 F2              BNE VECK2
E6BA 4C 20 E5           JMP DU13        ;CLOSE FILE, IT IS OKAY
E6BD
E6BD             ;***** 1 COMMAND-TOGGLE TAPE 1 CONTROL *****
E6BD AD 00 A8    TOGTA1 LDA DRB
E6C0 49 10              EOR #$10        ;INVERT PB4
E6C2 8D 00 A8           STA DRB
E6C5 29 10              AND #$10
E6C7 F0 28              BEQ BRK3        ;IF 0 TAPE CNTRL IS ON
```

```
E6C9 D0 2F              BNE BRK4        ;IF $10 TAPE CNTRL IS OFF
E6CB
E6CB             ;***** 2 COMMAND-TOGGLE TAPE 2 CONTROL *****
E6CB AD 00 A8    TOGTA2 LDA DRB
E6CE 49 20              EOR #$20        ;INVERT PB5
E6D0 8D 00 A8           STA DRB
E6D3 29 20              AND #$20
E6D5 F0 1A              BEQ BRK3
E6D7 D0 21              BNE BRK4
E6D9
E6D9             ;***** V COMMAND-TOGGLE REGISTER DISP FLG *****
E6D9             ;DISPLAY REGIST BEFORE EXEC
E6D9 A2 0E       REGT   LDX #REGF
E6DB D0 0A              BNE TOGL
E6DD
E6DD             ;****** Z COMMAND-TOGGLE DIS TRACE FLG *****
E6DD             ;DISPL NEXT INSTR BEFORE EXEC
E6DD A2 0F       TRACE  LDX #DISFLG
E6DF D0 06              BNE TOGL
E6E1
E6E1             ;***** \ COMMAND-TOGGLE PRINTER FLAG *****
E6E1 A2 11       PRITR  LDX #PRIFLG
E6E3 D0 02              BNE TOGL
E6E5
E6E5             ;***** 4 COMMAND-TOGGLE SOFT BRK ENABL FLG *****
E6E5 A2 10       BRKK   LDX #BKFLG
E6E7
E6E7 BD 00 A4    TOGL   LDA MONRAM,X    ;LOAD FLAG
E6EA F0 0A              BEQ TOGL1       ;FLAG IS OFF ,SO TURN ON
E6EC A9 00              LDA #0          ;FLAG IS ON ,SO TURN OFF
E6EE 9D 00 A4           STA MONRAM,X
E6F1 A0 24       BRK3   LDY #M7-M1      ;PRINT "OFF"
E6F3 4C AF E7    BRK2   JMP KEP
E6F6 38          TOGL1  SEC             ;TURN FLAG ON BY SETTING NON-ZERO
E6F7 7E 00 A4           ROR MONRAM,X    ;FLAG IS ON MSB
E6FA A0 21       BRK4   LDY #M6-M1      ;PRINT "ON"
E6FC D0 F5              BNE BRK2
E6FE
E6FE             ;***** # COMMAND-CLEAR ALL BREAKS *****
E6FE A9 00       CLRBK  LDA #0          ;STORE ZEROS INTO BRKPT LIST
E700 A2 07              LDX #7
E702 9D 00 01    RS20   STA BKS,X
E705 CA                 DEX
E706 10 FA              BPL RS20
E708 30 E7              BMI BRK3        ;PRINT "OFF"
E70A
E70A             ;***** K COMMAND-DISASSEMBLE MEMORY *****
E70A A9 2A       KDISA  LDA #'*'        ;GET START ADDRESS
E70C 20 7A E9           JSR OUTPUT
E70F 20 AE EA           JSR ADDIN
E712 B0 F6              BCS KDISA       ;IF ERROR DO IT AGAIN
E714 20 D7 E5           JSR CGPC0       ;GET IT INTO PROG CNTR
E717 20 37 E8           JSR PSL1        ;PRINT "/"
E71A 20 85 E7           JSR GCNT        ;GET COUNT
E71D 20 24 EA           JSR CRCK
E720 4C 2B E7           JMP JD2
E723 20 07 E9    JD1    JSR RCHEK       ;SEE IF HE WANTS TO INTERRUPT
E726 20 90 E7           JSR DONE
E729 F0 17              BEQ JD4
E72B 20 6C F4    JD2    JSR DISASM      ;GO TO DISASSEMBLER
E72E AD 25 A4           LDA SAVPC       ;POINT TO NEXT INSTRUC LOCAT
E731 38                 SEC             ;ONE MORE TO PROG CNTR
```

```
E732 65 EA              ADC LENGTH
E734 8D 25 A4           STA SAVPC
E737 90 03              BCC JD3
E739 EE 26 A4           INC SAVPC+1
E73C 20 24 EA    JD3    JSR CRCK        ;<CR>
E73F 4C 23 E7           JMP JD1
E742 60          JD4    RTS
E743
E743                    ;INITIALIZATION TABLE FOR 6522
E743 340037FF25FFINTAB1 .DB $34,$00,$37,$FF,$25,$FF,$25,$FF
E749 25FF
E74B FF FF 00 00        .DB $FF,$FF,$00,T1I+T2I
E74F E1 FF 7F           .DB MOFF+PRST+SP12,$FF,$7F
E752                    ;INITIALIZATION TABLE FOR 6532
E752 FF FF 00 00 INTAB2 .DB $FF,$FF,$00,$00
E756                    ;INITIALIZATION TABLE FOR MONITOR RAM
E756 7BE054E105EFINTAB3 .DW NMIV3,IRQV3,OUTDIS
E75C C70802CA0380       .DB $C7,$08,$02,$CA,$03,$80,$00,$00
E762 0000
E764 00800D0D0000       .DB $00,$80,$0D,$0D,$00,$00,$00
E76A 00
E76B                    ;SEE IF WE HIT A SOFT BREAKPOINT (PART OF NMV3)
E76B A2 07       CKB    LDX #7          ;COMPARE BRKPT LIST TO TRAP ADDR
E76D BD 00 01    CKB2   LDA BKS,X       ;GET ADDRESS OF NEXT BREAKPOINT
E770 CA                 DEX
E771 CD 26 A4           CMP SAVPC+1     ;COMPARE TO SAVED PROGRAM COUNTER
E774 D0 0A              BNE CKB1
E776 BD 00 01           LDA BKS,X
E779 CD 25 A4           CMP SAVPC
E77C D0 02              BNE CKB1        ;NO MATCH SO TRY NEXT BREAKPOINT
E77E 38                 SEC             ;MATCH-SET MATCH FLAG
E77F 60                 RTS
E780 CA          CKB1   DEX
E781 10 EA              BPL CKB2        ;MORE TO GO
E783 18                 CLC             ;NO MATCH -RESET MATCH FLAG
E784 60                 RTS
E785
E785                    ;GET # OF LINES COUNT FOR GO-COMMAND,LIST-COMM
E785 20 5D EA    GCNT   JSR RD2
E788 90 02              BCC GCN1
E78A 49 0C              EOR #$0C        ;<SPACE>---> $2C ,<CR>---> $01
E78C 8D 19 A4    GCN1   STA COUNT
E78F 60                 RTS
E790
E790                    ;CHECK IF COUNT HAS REACHED ZERO
E790                    ;COUNT=$2C MEANS FOREVER
E790 AD 19 A4    DONE   LDA COUNT       ;IF COUNT=0 WE ARE DONE
E793 C9 2C              CMP #$2C        ;THIS MEANS FOR EVER
E795 F0 09              BEQ DON1        ;SET ACC DIFF FROM ZERO
E797 F8                 SED             ;DECREMENT COUNT IN DECIMAL
E798 38                 SEC
E799 E9 01              SBC #1
E79B D8                 CLD
E79C 8D 19 A4           STA COUNT
E79F 60                 RTS
E7A0 A9 2C       DON1   LDA #$2C
E7A2 60                 RTS
E7A3
E7A3 A0 00       FROM   LDY #0          ;PRINT "FR="
E7A5 F0 02              BEQ TO1
E7A7
E7A7 A0 05       TO     LDY #M3-M1      ;PRINT "TO="
```

```
E7A9 20 AF E7    TO1    JSR KEP
E7AC 4C B1 EA           JMP ADDNE       ;GET ADDRESS
E7AF
E7AF                    ;PRINT MSG POINTED TO BY Y REG
E7AF B9 00 E0    KEP    LDA M1,Y
E7B2 48                 PHA
E7B3 29 7F              AND #$7F        ;STRIP OFF MSB
E7B5 20 7A E9           JSR OUTPUT
E7B8 C8                 INY
E7B9 68                 PLA
E7BA 10 F3              BPL KEP         ;MSB =1 ?
E7BC 60                 RTS
E7BD
E7BD                    ;PRINT "*" ,BUT NOT TO TAPE RECORDER, NOR LOADING....
E7BD                    ;PAPER TAPE OR TO DISPLAY
E7BD AD 12 A4    PROMPT LDA INFLG       ;WHICH DEV (FOR EDITOR)
E7C0 C9 54              CMP #'T'        ;NO PROMPT IF "T" OR "L"
E7C2 4C EF FE           JMP PATC11
E7C5 20 42 E8    PROMP1 JSR TTYTST      ;PROMPT ONLY TO TTY
E7C8 D0 05              BNE PR2         ;BRANCH ON KB
E7CA A9 2A              LDA #'*'
E7CC 4C 7A E9    PR1    JMP OUTPUT      ;ONLY TO TERMIN
E7CF A9 0D       PR2    LDA #CR         ;CLR DISP
E7D1 4C 05 EF           JMP OUTDIS
E7D4
E7D4 A9 3F       QM     LDA #'?'        ;PRINT "?"
E7D6 D0 F4              BNE PR1
E7D8
E7D8 A9 3D       EQUAL  LDA #'='        ;PRINT "="
E7DA D0 F0              BNE PR1
E7DC
E7DC                    ;ON DELETE KEY OUTPUT SLASH IF TTY & ....
E7DC                    ;BACK UP CURSOR IF KB (MAY NEED SCROLLING)
E7DC 20 42 E8    PSLS   JSR TTYTST      ;TTY OR KB ?
E7DF F0 56              BEQ PSL1        ;BRANCH ON TTY
E7E1 20 9E EB           JSR PHXY        ;SAVE X,Y
E7E4 CE 15 A4           DEC CURPO2      ;DECR DISP PNTR
E7E7 AE 15 A4           LDX CURPO2
E7EA E0 14              CPX #20         ;IF MORE THAN 20 JUST SCROLL THEM
E7EC B0 0D              BCS PSL0
E7EE A9 20              LDA #' '        ;< 20 ,SO CLR CUR
E7F0 20 02 EF           JSR OUTDP1
E7F3 CE 15 A4           DEC CURPO2
E7F6 4C 02 E8           JMP PSL00
E7F9 EA                 NOP
E7FA EA                 NOP
E7FB 20 F8 FE    PSL0   JSR PATC12      ;CLR PRIFLG
E7FE CA                 DEX             ;ONE CHR LESS
E7FF 20 2F EF           JSR OUTD2A      ;SCROLL THEM
E802 AD 15 A4    PSL00  LDA CURPO2      ;DISBUF---> PRIBUFF
E805 C9 15              CMP #21
E807 90 13              BCC PSL0B
E809 C9 29              CMP #41
E80B 90 07              BCC PSL0A
E80D A0 28              LDY #40         ;CHR 40-59
E80F E9 28              SBC #40
E811 4C 1E E8           JMP PSL0C
E814 A0 14       PSL0A  LDY #20         ;CHR 20-39
E816 38                 SEC
E817 E9 14              SBC #20
E819 4C 1E E8           JMP PSL0C
E81C A0 00       PSL0B  LDY #0          ;CHR 00-19
```

```
E81E 8D 16 A4    PSL0C  STA CURPOS
E821 A2 00              LDX #0
E823 B9 38 A4    PSL0D  LDA DIBUFF,Y    ;TRANSFER THEM
E826 9D 60 A4           STA IBUFM,X
E829 E8                 INX
E82A C8                 INY
E82B EC 16 A4           CPX CURPOS      ;PRI PNTR
E82E 90 F3              BCC PSL0D
E830 20 38 F0           JSR OUTPR       ;CLR PRI BUFF TO THE RIGHT
E833 20 AC EB           JSR PLXY        ;RESTORE X,Y
E836 60                 RTS
E837 A9 2F       PSL1   LDA #'/'        ;PRINT "/"
E839 D0 91              BNE PR1
E83B
E83B 20 3E E8    BLANK2 JSR BLANK       ;TWO SPACES
E83E A9 20       BLANK  LDA #' '
E840 D0 8A              BNE PR1
E842
E842             ;CHECK TTY/KBD SWITCH (Z=1 FOR TTY)
E842 A9 08       TTYTST LDA #$08        ;CHECK IF TTY OR KB
E844 2C 00 A8           BIT DRB         ;TTY OR KB SWICTH =PB3
E847 60                 RTS
E848
E848             ;WHERE IS INPUT COMING FROM?
E848             ;SET UP FOR INPUT ACTIVE DEVICE
E848 A0 2A       WHEREI LDY #M9-M1      ;PRINT "IN"
E84A 20 70 E9           JSR KEPR        ;OUTPUT MSG AND INPUT CHR
E84D 8D 12 A4           STA INFLG
E850 C9 54              CMP #'T'
E852 D0 08              BNE WHE1
E854 A2 00              LDX #0          ;FOR INPUT FILE FLG
E856 20 A2 E8           JSR FNAM        ;OPEN FILE FOR TAPE (1 OR 2)
E859 4C 2F E3           JMP LOADTA      ;GET FILE
E85C C9 4B       WHE1   CMP #'K'        ;TAPE WITH KIM FORMAT
E85E D0 08              BNE WHE2
E860 A2 00              LDX #0          ;FOR INPUT FILE FLG
E862 20 A2 E8           JSR FNAM        ;OPEN FILE FOR TAP (1 OR 2)
E865 4C A4 E3           JMP LOADKI      ;THE WHOLE FILE
E868 C9 55       WHE2   CMP #'U'        ;USER RTN?
E86A D0 04              BNE WHE3
E86C 18                 CLC             ;SET FLG FOR INITIALIZATION
E86D 6C 08 01           JMP (UIN)       ;USER INPUT SETUP
E870 60          WHE3   RTS
E871
E871             ;WHERE IS OUTPUT GOING TO?
E871             ;SET UP FOR OUTPUT ACTIVE DEVICE
E871 A0 2D       WHEREO LDY #M10-M1     ;PRINT "OUT"
E873 20 70 E9           JSR KEPR        ;OUTPUT MSG & INPUT CHR
E876 8D 13 A4           STA OUTFLG      ;DEVICE FLG
E879             ;TAPES
E879 C9 54              CMP #'T'
E87B D0 08              BNE WHRO1
E87D A2 01              LDX #1          ;FOR OUTPUT FILE FLG
E87F 20 A2 E8           JSR FNAM        ;FILENAME & TAPE (1 OR 2)
E882 4C 6F E5           JMP DUMPTA      ;INITIALIZE FILE
E885 C9 4B       WHRO1  CMP #'K'        ;TAPE WITH KIM FORMAT
E887 D0 05              BNE WHRO2
E889 A2 01              LDX #1          ;FOR OUTPUT FILE FLG
E88B 4C A2 E8           JMP FNAM
E88E             ;PRINTER
E88E C9 50       WHRO2  CMP #'P'        ;PRINTER?
E890 D0 05              BNE WHRO3
```

```
E892 A9 0D              LDA #CR         ;OUTPUT LAST LINE IF ON
E894 4C 00 F0           JMP OUTPRI      ;& CLEAR PRINTER PTR
E897             ;USER SET UP
E897 C9 55       WHRO3  CMP #'U'        ;USR RTN?
E899 D0 04              BNE WHRO4
E89B 18                 CLC             ;CLR FLG FOR INITIALIZATION
E89C 6C 0A 01           JMP (UOUT)      ;USER OUTPUT SETUP
E89F             ;ANY OTHER
E89F 4C 13 EA    WHRO4  JMP CRLOW
E8A2
E8A2             ;GET FILE NAME & TAPE UNIT
E8A2 20 9E EB    FNAM   JSR PHXY        ;SAVE IN/OUT FLG (X)
E8A5 20 CF E8           JSR NAMO        ;GET NAME
E8A8 A0 50       WHICHT LDY #TMSG2-M1   ;PRINT "T="
E8AA 20 70 E9           JSR KEPR        ;OUTPUT MSG & INPUT CHR
E8AD C9 0D              CMP #CR
E8AF D0 02              BNE TAP1
E8B1 A9 31              LDA #'1'        ;<CR> ==> TAPE 1
E8B3 38          TAP1   SEC
E8B4 E9 31              SBC #'1'        ;SUBTRACT 31
E8B6 30 04              BMI TAP2        ;ONLY 1,2 OK
E8B8 C9 02              CMP #2
E8BA 30 06              BMI TAP3        ;OK
E8BC 20 D4 E7    TAP2   JSR QM          ;ERROR
E8BF 4C A8 E8           JMP WHICHT
E8C2 20 AC EB    TAP3   JSR PLXY        ;IN/OUT FLG
E8C5 9D 34 A4           STA TAPIN,X     ;IF X=0 --> TAPIN (TAPE 1 OR 2)
E8C8 20 83 FE           JSR CUREAD      ;GET ANYTHING
E8CB 20 24 EA           JSR CRCK        ;<CR>
E8CE 60                 RTS             ;IF X=1 --> TAPOUT (TAPE 1 OR 2)
E8CF
E8CF             ;GET FILE NAME
E8CF A0 4D       NAMO   LDY #TMSG1-M1   ;PRINT "F="
E8D1 20 AF E7           JSR KEP         ;NO CRLF
E8D4 A0 00              LDY #0
E8D6 20 5F E9    NAMO1  JSR RDRUP       ;GET CHAR
E8D9 C9 0D              CMP #CR         ;DONE?
E8DB F0 0C              BEQ NAMO2
E8DD C9 20              CMP #' '
E8DF F0 08              BEQ NAMO2
E8E1 99 2E A4           STA NAME,Y      ;STORE
E8E4 C8                 INY
E8E5 C0 05              CPY #5
E8E7 D0 ED              BNE NAMO1
E8E9             ;BLANK REST OF NAME
E8E9 A9 20       NAMO2  LDA #' '
E8EB C0 05       NAMO3  CPY #5
E8ED F0 06              BEQ NAMO4
E8EF 99 2E A4           STA NAME,Y
E8F2 C8                 INY
E8F3 D0 F6              BNE NAMO3
E8F5 4C 3E E8    NAMO4  JMP BLANK
E8F8
E8F8             ;SET INPUT FROM TERMINAL (KB OR TTY)
E8F8 A9 0D       INLOW  LDA #CR
E8FA 8D 12 A4           STA INFLG
E8FD 60                 RTS
E8FE
E8FE             ;SET I/O  TO TERMINAL (KB & D/P ,OR TTY)
E8FE 20 F8 E8    LL     JSR INLOW
E901
E901             ;SET OUTPUT TO TERMINAL (D/P OR TTY)
```

```
E901 A9 0D       OUTLOW LDA #CR
E903 8D 13 A4           STA OUTFLG
E906 60          OUTL1  RTS
E907
E907                    ;ON <ESCAPE> STOPS EXECUTION & BACK TO MONITOR
E907                    ;ON <SPACE> STOPS EXECUTION & CONTINUE ON ANY OTHER KEY
E907 20 42 E8    RCHEK  JSR TTYTST       ;TTY OR KB ?
E90A F0 1A              BEQ RCHTTY
E90C 20 EF EC           JSR ROONEK       ;CLR MSK & GET A KEY
E90F 88                 DEY
E910 30 13              BMI RCH3         ;RTN ON NO KEY
E912 A2 00              LDX #0
E914 20 82 EC           JSR GETK2        ;GET THE KEY
E917 C9 1B              CMP #ESCAPE
E919 F0 3B              BEQ REA1         ;TO COMMAN & SET I/O TO TERMINAL
E91B C9 20              CMP #' '         ;WAIT KEY
E91D D0 06              BNE RCH3         ;RTN, IGNORE OTHER KEYS
E91F 20 EF EC    RCH2   JSR ROONEK       ;WAIT TILL HE RELEASE IT &
E922 88                 DEY              ;QUIT WAITING ON NEXT KEY
E923 30 FA              BMI RCH2
E925 60          RCH3   RTS
E926 70 13       RCHTTY BVS RCHT1        ;TTI=PB6 ---> V (OVERFL FLG)
E928 2C 00 A8    RCHT2  BIT DRB          ;WAIT TILL HE RELEASE IT
E92B 50 FB              BVC RCHT2
E92D 20 0F EC           JSR DELAY
E930 20 DB EB           JSR GETTTY       ;GET A CHAR
E933 C9 1B              CMP #ESCAPE
E935 F0 1F              BEQ REA1         ;TO COMMAN
E937 C9 20              CMP #' '
E939 D0 ED              BNE RCHT2
E93B 60          RCHT1  RTS              ;QUIT WAITING ON ANY KEY
E93C
E93C                    ;READ ONE CHAR FROM KB/TTY & PRESERVE X,Y
E93C 20 9E EB    READ   JSR PHXY         ;PUSH X & Y
E93F 20 42 E8           JSR TTYTST       ;TTY OR KB ?
E942 D0 06              BNE READ1
E944 20 DB EB           JSR GETTTY
E947 4C 4D E9           JMP READ2
E94A 20 40 EC    READ1  JSR GETKEY
E94D 20 AC EB    READ2  JSR PLXY         ;PULL X & Y
E950 29 7F              AND #$7F         ;STRIP PARITY
E952 C9 1B              CMP #ESCAPE
E954 D0 E5              BNE RCHT1        ;RTN
E956 20 3D FF    REA1   JSR PATC18       ;<CR> & CLR BUFFERS
E959 4C A1 E1           JMP COMIN        ;BOTH I/O TO TERMINAL
E95C
E95C                    ;READ WITH RUBOUT OR DELETE POSSIBLE
E95C 20 DC E7    RB2    JSR PSLS         ;SLASH OR BACK SPACE
E95F 20 83 FE    RDRUP  JSR CUREAD
E962 C9 08              CMP #RUB         ;RUBOUT
E964 F0 04              BEQ RDR1
E966 C9 7F              CMP #$7F         ;ALSO DELETE
E968 D0 0C              BNE RED2         ;ECHO IF NOT <CR>
E96A                    ;RUBOUT TO DELETE CHAR
E96A 88          RDR1   DEY
E96B 10 EF              BPL RB2
E96D C8                 INY
E96E F0 EF              BEQ RDRUP
E970
E970                    ;OUTPUT MESSAGE THEN INPUT CHR
E970 20 AF E7    KEPR   JSR KEP
E973
```

```
E973                ;READ AND ECHO A CHAR FROM KB OR TTY
E973 20 83 FE    REDOUT JSR CUREAD
E976 C9 0D       RED2   CMP #CR
E978 F0 C1              BEQ RCHT1       ;DO NOT ECHO <CR>
E97A
E97A                ;OUTPUTS A CHAR TO EITHER TTY OR D/P
E97A 48          OUTPUT PHA             ;SAVE IT
E97B AD 11 A4    OUT1   LDA PRIFLG      ;IF LSB=1 OUTPUT ONLY TO DISP
E97E 29 01              AND #$01
E980 F0 04              BEQ OUT1A
E982 68                 PLA
E983 4C 02 EF           JMP OUTDP1      ;ONLY TO DISPL
E986 20 42 E8    OUT1A  JSR TTYTST      ;TTY OR KB ?
E989 D0 04              BNE OUT2
E98B 68                 PLA
E98C 4C A8 EE           JMP OUTTTY      ;TO TTY
E98F 68          OUT2   PLA
E990 4C FC EE           JMP OUTDP       ;TO DISP & PRINTR
E993
E993                ;GET A CHR FROM CURRENT INPUT DEVICE (SET ON INFLG)
E993 AD 12 A4    INALL  LDA INFLG
E996 C9 54              CMP #'T'
E998 D0 03              BNE *+5
E99A 4C 3B ED           JMP TIBYTE      ;CHAR FROM BUFFER
E99D C9 4B              CMP #'K'        ;WITH KIM FORMAT
E99F D0 03              BNE *+5
E9A1 4C 29 EE           JMP GETTAP      ;DIRECTLY FROM TAPE
E9A4 C9 4D              CMP #'M'        ;MEMORY FOR ASM?
E9A6 D0 03              BNE *+5
E9A8 4C D0 FA           JMP MREAD
E9AB C9 55              CMP #'U'        ;USER ROUTINE?
E9AD D0 04              BNE *+6
E9AF 38                 SEC             ;SET FLG FOR NORMAL INPUT
E9B0 6C 08 01           JMP (UIN)
E9B3 C9 4C              CMP #'L'        ;TO LOAD PPR TAPE
E9B5 D0 A8              BNE RDRUP
E9B7 4C DB EB           JMP GETTTY      ; FROM TTY
E9BA
E9BA                ;.FILE A2
E9BA A9 3B       SEMI   LDA #SEMICOLON  ;OUTPUT A ";"
E9BC                ;WRITE A CHR TO OUTPUT DEVICE (SET ON OUTFLG)
E9BC 48          OUTALL PHA
E9BD AD 13 A4           LDA OUTFLG
E9C0                ;TAPE BY BLOCKS
E9C0 C9 54              CMP #'T'        ;TAPES ?
E9C2 D0 04              BNE OUTA1
E9C4 68                 PLA
E9C5 4C 8B F1           JMP TOBYTE      ;OUTPUT ONE CHAR TO TAPE BUFFER
E9C8                ;TAPE KIM FORMAT
E9C8 C9 4B       OUTA1  CMP #'K'        ;KIM-1 ?
E9CA D0 04              BNE OUTA2
E9CC 68                 PLA
E9CD 4C 4A F2           JMP OUTTAP
E9D0                ;PRINTER
E9D0 C9 50       OUTA2  CMP #'P'        ;PRINTER ?
E9D2 D0 0E              BNE OUTA3
E9D4 38                 SEC             ;TURN PRINTER ON
E9D5 6E 11 A4           ROR PRIFLG
E9D8 68                 PLA
E9D9 08                 PHP
E9DA 20 00 F0           JSR OUTPRI
E9DD 28                 PLP
```

```
E9DE 2E 11 A4           ROL PRIFLG      ;RESTORE FLG
E9E1 60                 RTS
E9E2             ;USER DEFINED
E9E2 C9 55       OUTA3  CMP #'U'        ;USER ROUTINE?
E9E4 D0 04              BNE OUTA4
E9E6 38                 SEC             ;SET FLG FOR NORMAL OUTPUT
E9E7 6C 0A 01           JMP (UOUT)      ;YES
E9EA             ;NOWHERE OR TO TTY ,D/P
E9EA C9 58       OUTA4  CMP #'X'        ;EAT IT?
E9EC D0 8D              BNE OUT1        ;OUTPUT TO TTY OR D/P
E9EE 68                 PLA
E9EF 60                 RTS
E9F0
E9F0             ;THIS ROUTINE OUTPUTS A CRLF TO ANY OUTPUT DEV
E9F0             ;LF AND NULL IS SENT ONLY TO TTY
E9F0 A9 0D       CRLF   LDA #CR
E9F2 20 BC E9           JSR OUTALL
E9F5 20 42 E8           JSR TTYTST      ;TTY OR KB ?
E9F8 D0 29              BNE CR2J
E9FA AD 13 A4           LDA OUTFLG      ;LF ONLY TO TTY
E9FD C9 54              CMP #'T'
E9FF F0 22              BEQ CR2J
EA01 C9 4B              CMP #'K'
EA03 F0 1E              BEQ CR2J
EA05 C9 50              CMP #'P'
EA07 F0 1A              BEQ CR2J
EA09 A9 0A              LDA #LF
EA0B 20 BC E9           JSR OUTALL
EA0E A9 FF              LDA #NULLC
EA10 4C BC E9           JMP OUTALL
EA13
EA13             ;CRLF TO TERMINAL (TTY OR D/P) ONLY
EA13 48          CRLOW  PHA             ;SAVE A
EA14 AD 13 A4           LDA OUTFLG
EA17 48                 PHA
EA18 20 01 E9           JSR OUTLOW
EA1B 20 F0 E9           JSR CRLF
EA1E 68                 PLA
EA1F 8D 13 A4           STA OUTFLG
EA22 68                 PLA
EA23 60          CR2J   RTS
EA24
EA24             ;OUTPUT <CR> TO TTY IF SWITCH ON TTY & INFLG NOT L
EA24             ;DONT CLR DISPLAY BUT CLEARS PNTRS FOR NEXT LINE
EA24             ;IF PRNTR HAS PRINTED ON 21RST CHR DONT OUTPUT <CR>
EA24 AD 12 A4    CRCK   LDA INFLG       ;NO <CR> IF "L"
EA27 C9 4C              CMP #'L'
EA29 D0 01              BNE CRCK1
EA2B 60                 RTS
EA2C 20 42 E8    CRCK1  JSR TTYTST      ;CHECK IF TTY OR KB
EA2F F0 E2              BEQ CRLOW       ;BRNCH IF TTY
EA31             ;IF PRINTR PTR=0 ,DO NOT CLR PRI
EA31 AD 16 A4           LDA CURPOS
EA34 F0 05              BEQ CRCK2       ;IF PTR=0 ,NO <CR>
EA36 A9 0D              LDA #CR
EA38 20 00 F0           JSR OUTPRI
EA3B A9 8D       CRCK2  LDA #CR+$80     ;<CR> ONLY FOR TV
EA3D 4C 02 EF           JMP OUTDP1
EA40 EA                 NOP
EA41 EA                 NOP
EA42
EA42             ;WRITE A THEN X IN ASCII TO THE OUTPUT DEV
```

```
EA42 20 46 EA    WRAX   JSR NUMA
EA45 8A                 TXA
EA46
EA46                    ;PRINT ONE BYTE=TWO ASCII CHARS TO OUTPUT DEVICE
EA46 48          NUMA   PHA
EA47 4A                 LSR A
EA48 4A                 LSR A
EA49 4A                 LSR A
EA4A 4A                 LSR A
EA4B 20 51 EA           JSR NOUT
EA4E 68                 PLA
EA4F 29 0F              AND #$F
EA51 18          NOUT   CLC
EA52 69 30              ADC #'0'
EA54 C9 3A              CMP #'9'+1
EA56 90 02              BCC LT10
EA58 69 06              ADC #6          ;CARRY IS SET
EA5A 4C BC E9    LT10   JMP OUTALL
EA5D
EA5D                    ;READ TWO CHR & PACK THEM INTO ONE BYTE
EA5D                    ;PART OF ALTER MEMORY , / COMM
EA5D 20 73 E9    RD2    JSR REDOUT
EA60 C9 0D              CMP #CR         ;<CR>?
EA62 F0 17              BEQ RSPAC
EA64 C9 20              CMP #' '        ;FOR MEMORY ALTER
EA66 F0 13              BEQ RSPAC
EA68 C9 2E              CMP #'.'        ;TREAT "." AS <SPACE>
EA6A D0 04              BNE RD1
EA6C A9 20              LDA #' '
EA6E D0 0B              BNE RSPAC
EA70 20 84 EA    RD1    JSR PACK
EA73 B0 06              BCS RSPAC
EA75 20 73 E9           JSR REDOUT
EA78 4C 84 EA           JMP PACK
EA7B                    ;WAS SPACE OR <CR>
EA7B 38          RSPAC  SEC
EA7C 60                 RTS
EA7D
EA7D                    ;CONVERT ACC IN ASCII TO ACC IN HEX (4 MSB=0)
EA7D 48          HEX    PHA             ;SAVE A
EA7E A9 00              LDA #0          ;CLEAR STIY IF HEX
EA80 8D 29 A4           STA STIY+2      ;BECAUSE ONLY ONCE
EA83 68                 PLA
EA84                    ;PACK TWO ASCII INTO ONE HEX (CALL SUBR TWO TIMES)
EA84                    ;RESULT IS GIVEN ON ACC WITH FIRST CHR INTO 4 MSB
EA84 C9 30       PACK   CMP #'0'        ;< 30 ?
EA86 90 F3              BCC RSPAC
EA88 C9 47              CMP #'F'+1      ; > 47 ?
EA8A B0 EF              BCS RSPAC
EA8C C9 3A              CMP #'9'+1      ; < $10
EA8E 90 06              BCC PAK1
EA90 C9 40              CMP #'A'-1      ; > $10 ?
EA92 90 E7              BCC RSPAC
EA94 69 08              ADC #8          ;ADD 9 IF LETTER (C IS SET)
EA96 2A          PAK1   ROL A           ;SHIFT A 4 TIMES
EA97 2A                 ROL A
EA98 2A                 ROL A
EA99 2A                 ROL A
EA9A 8E 2D A4           STX CPIY+3      ;SAVE X
EA9D A2 04              LDX #4
EA9F 2A          PAK2   ROL A           ;TRANSFER A TO STIY
EAA0 2E 29 A4           ROL STIY+2      ; THRU CARRY
```

```
EAA3 CA                 DEX
EAA4 D0 F9              BNE PAK2
EAA6 AE 2D A4           LDX CPIY+3      ;REST X
EAA9 AD 29 A4           LDA STIY+2
EAAC 18                 CLC
EAAD 60                 RTS
EAAE
EAAE            ;GET FOUR BYTE ADDR ,TAKE LAST FOUR CHR TO...
EAAE            ;CALCULATE ADDR .ALLOW DELETE ALSO
EAAE 20 D8 E7   ADDIN   JSR EQUAL
EAB1 AD 15 A4   ADDNE   LDA CURPO2      ;SAVE POSITION
EAB4 48                 PHA
EAB5 A0 00              LDY #0
EAB7 20 5F E9   ADDN1   JSR RDRUP
EABA C9 0D              CMP #CR
EABC F0 09              BEQ ADDN2
EABE C9 20              CMP #' '
EAC0 F0 05              BEQ ADDN2
EAC2 C8                 INY
EAC3 C0 0B              CPY #11         ;ALLOW 10
EAC5 90 F0              BCC ADDN1
EAC7 68         ADDN2   PLA
EAC8 8D 2D A4           STA CPIY+3      ;SAVE
EACB C0 00              CPY #0          ;IF FIRST CHR PUT DEFAULT VALUES
EACD D0 0D              BNE ADDN3
EACF A9 02              LDA #$02
EAD1 8D 1D A4           STA ADDR+1      ;DEFAULT OF 0200
EAD4 8D 1E A4           STA CKSUM       ;DEFAULT
EAD7 8C 1C A4           STY ADDR
EADA 18                 CLC
EADB 60                 RTS
EADC A2 00      ADDN3   LDX #0
EADE 88                 DEY             ;Y-4
EADF 88                 DEY
EAE0 88                 DEY
EAE1 88                 DEY
EAE2 10 13              BPL ADDN5       ;BRANCH IF > 4 CHR
EAE4 98                 TYA
EAE5 49 FF              EOR #$FF
EAE7 A8                 TAY             ;# OF LEADING 0
EAE8 A9 30      ADDN4   LDA #$30
EAEA 9D 1C A4           STA ADDR,X
EAED E8                 INX
EAEE 88                 DEY
EAEF 10 F7              BPL ADDN4
EAF1 AC 2D A4           LDY CPIY+3      ;NOW THE CHR
EAF4 4C FD EA           JMP ADDN6
EAF7 98         ADDN5   TYA             ;PUT CHR
EAF8 18                 CLC
EAF9 6D 2D A4           ADC CPIY+3
EAFC A8                 TAY
EAFD B9 38 A4   ADDN6   LDA DIBUFF,Y    ;FROM DISP BUFF
EB00 9D 1C A4           STA ADDR,X
EB03 C8                 INY
EB04 E8                 INX
EB05 E0 04              CPX #4
EB07 D0 F4              BNE ADDN6
EB09 A2 01              LDX #1
EB0B A0 00              LDY #0          ;CNVRT CHR TO HEX
EB0D B9 1C A4   ADDN7   LDA ADDR,Y
EB10 20 7D EA           JSR HEX
EB13 B0 16              BCS ADDN8
```

```
EB15 C8                    INY
EB16 B9 1C A4              LDA ADDR,Y
EB19 C8                    INY
EB1A 20 84 EA              JSR PACK        ;PACK TWO CHRS INTO 1 BYTE
EB1D B0 0C                 BCS ADDN8       ;BRCNH IF ERROR
EB1F 9D 1C A4              STA ADDR,X
EB22 CA                    DEX
EB23 10 E8                 BPL ADDN7
EB25 E8                    INX             ;X=0
EB26 8E 1E A4              STX CKSUM       ;TO INDICATE WE GOT AN ADDR
EB29 18                    CLC             ;NO INVALID CHARS
EB2A 60                    RTS
EB2B 20 94 E3       ADDN8  JSR CKER00      ;OUTPUT ERROR MSG
EB2E 20 24 EA              JSR CRCK        ;<CR>
EB31 38                    SEC             ;SET CARRY FOR INVALID CHR
EB32 60                    RTS
EB33
EB33                ;MEMORY FAIL TO WRITE MSG & SPECIFIC ADDRESS
EB33 20 24 EA       MEMERR JSR CRCK
EB36 20 CD E2              JSR NXTADD      ;ADD Y TO ADDR+1,ADDR
EB39 A0 31                 LDY #M11-M1     ;PRINT "MEM FAIL"
EB3B 20 AF E7              JSR KEP         ;FAIL MSG
EB3E 20 DB E2              JSR WRITAZ      ;PRINT ADDR+1 , ADDR
EB41 4C A1 E1              JMP COMIN
EB44
EB44                ;CLEAR DISPLAY & PRINTER POINTERS
EB44 A9 00          CLR    LDA #0
EB46 8D 15 A4              STA CURPO2      ;DISP PNTR
EB49 8D 16 A4              STA CURPOS      ;PRINTR PNTR
EB4C 60                    RTS
EB4D
EB4D                ;CLEAR CKSUM
EB4D A9 00          CLRCK  LDA #0
EB4F 8D 1F A4              STA CKSUM+1
EB52 8D 1E A4              STA CKSUM
EB55 60                    RTS
EB56
EB56                ;CODE FOR PAGE ZERO SIMULATION
EB56                ;SUBR LDAY-SIMULATES LDA (N),Y INSTR WITHOUT PAG 0
EB56                ;BY PUTTING INDIR ADDR INTO RAM & THEN EXEC LDA NM,Y
EB56 A9 25          PCLLD  LDA #SAVPC      ;FOR DISASSEMBLER
EB58 8C 2D A4       LDAY   STY CPIY+3      ;SAVE Y
EB5B A8                    TAY
EB5C B9 00 A4              LDA MONRAM,Y    ;MONRAM=MONITOR RAM
EB5F 8D 2B A4              STA LDIY+1
EB62 B9 01 A4              LDA MONRAM+1,Y
EB65 8D 2C A4              STA LDIY+2
EB68 AC 2D A4              LDY CPIY+3      ;REST Y
EB6B A9 B9                 LDA #$B9        ;INST FOR LDA NM,Y
EB6D 8D 2A A4              STA LDIY
EB70 A9 60                 LDA #$60        ;RTS
EB72 8D 2D A4              STA LDIY+3
EB75 4C 2A A4              JMP LDIY        ;START EXECUTING LDA (),Y
EB78
EB78                ;SUBR STORE AT ADDR & CMP WITHOUT PAG 0
EB78                ;REPLACES STA (ADDR),Y  &  CMP (ADDR),Y
EB78                ;LOOK THAT ADDR & ADDR+1 ARE NOT ON PAG 0
EB78 48             SADDR  PHA
EB79 AD 1C A4              LDA ADDR
EB7C 8D 28 A4              STA STIY+1
EB7F 8D 2B A4              STA CPIY+1
EB82 AD 1D A4              LDA ADDR+1
```

```
EB85 8D 29 A4           STA STIY+2
EB88 8D 2C A4           STA CPIY+2
EB8B A9 99              LDA #$99        ;STA INSTR
EB8D 8D 27 A4           STA STIY
EB90 A9 D9              LDA #$D9        ;CMP INSTR
EB92 8D 2A A4           STA CPIY
EB95 A9 60              LDA #$60        ;RTS
EB97 8D 2D A4           STA LDIY+3
EB9A 68                 PLA
EB9B 4C 27 A4           JMP STIY        ;START EXECUTING STA (),Y
EB9E
EB9E             ;PUSH X & Y WITHOUT CHANGING THE REGS
EB9E 8D 2D A4    PHXY   STA CPIY+3      ;SAVE ACC
EBA1 98                 TYA
EBA2 48                 PHA             ;PUSH Y
EBA3 8A                 TXA
EBA4 48                 PHA             ;PUSH X
EBA5 20 BA EB           JSR SWSTAK      ;SWAP X , Y WITH RTRN ADDR FROM S`
EBA8 AD 2D A4           LDA CPIY+3
EBAB 60                 RTS
EBAC
EBAC             ;PULL X & Y WITHOUT CHANGING ACC
EBAC             ;IT HAS TO BE CALLED BY JSR & NOT BY JMP INSTR
EBAC             ;SINCE IT SWAPS THE STACK
EBAC 8D 2D A4    PLXY   STA CPIY+3
EBAF 20 BA EB           JSR SWSTAK      ;SWAP X , Y WITH RTRN ADDR FROM`
EBB2 68                 PLA
EBB3 AA                 TAX             ;PULL X
EBB4 68                 PLA
EBB5 A8                 TAY             ;PULL Y
EBB6 AD 2D A4           LDA CPIY+3
EBB9 60                 RTS
EBBA
EBBA             ;SWAP STACK
EBBA BA          SWSTAK TSX
EBBB A9 02              LDA #2
EBBD 48          SWST1  PHA
EBBE BD 06 01           LDA $0106,X     ;GET PCH OR PCL
EBC1 BC 04 01           LDY $0104,X     ;GET Y OR X REGS
EBC4 9D 04 01           STA $0104,X
EBC7 98                 TYA
EBC8 9D 06 01           STA $0106,X
EBCB CA                 DEX
EBCC 68                 PLA
EBCD 38                 SEC
EBCE E9 01              SBC #1
EBD0 D0 EB              BNE SWST1
EBD2 BD 08 01           LDA $0108,X     ;RESTORE Y & X FROM STACK
EBD5 A8                 TAY
EBD6 BD 07 01           LDA $0107,X
EBD9 AA                 TAX
EBDA 60                 RTS
EBDB
EBDB             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EBDB             ;GET A CHAR FROM TTY SUBR INTO ACC ,SAVES X
EBDB 8A          GETTTY TXA             ;SAVE X
EBDC 48                 PHA
EBDD A2 07              LDX #$07        ;SET UP FOR 8 BIT CNT
EBDF 8E 2A A4           STX CPIY        ;CLR MSB
EBE2 2C 00 A8    GET1   BIT DRB         ;A^M ,  PB6->V
EBE5 70 FB              BVS GET1        ;WAIT FOR START BIT
EBE7 20 0F EC           JSR DELAY       ;DELAY 1 BIT
```

```
EBEA 20 23 EC          JSR DEHALF      ;DELAY 1/2 BIT TIME
EBED AD 00 A8   GET3   LDA DRB         ;GET 8 BITS
EBF0 29 40             AND #$40        ;MASK OFF OTHER BITS,ONLY PB6
EBF2 4E 2A A4          LSR CPIY        ;SHIFT RIGHT CHARACTER
EBF5 0D 2A A4          ORA CPIY
EBF8 8D 2A A4          STA CPIY
EBFB 20 0F EC          JSR DELAY       ;DELAY 1 BIT TIME
EBFE CA                DEX
EBFF D0 EC             BNE GET3        ;GET NEXT BIT
EC01 20 0F EC          JSR DELAY       ;DO NOT CARE FOR PARITY BIT
EC04 20 23 EC          JSR DEHALF      ;UNTIL WE GET BACK TO ONE AGAIN
EC07 68                PLA             ;RESTORE X
EC08 AA                TAX
EC09 AD 2A A4          LDA CPIY
EC0C 29 7F             AND #$7F        ;CLEAR PARITY BIT
EC0E 60                RTS
EC0F
EC0F            ;DELAY 1 BIT TIME AS GIVEN BY BAUD RATE
EC0F AD 18 A4   DELAY  LDA CNTL30      ;START TIMER T2
EC12 8D 08 A8          STA T2L
EC15 AD 17 A4          LDA CNTH30
EC18 8D 09 A8   DE1    STA T2H
EC1B AD 0D A8   DE2    LDA IFR         ;GET INT FLG FOR T2
EC1E 29 20             AND #MT2
EC20 F0 F9             BEQ DE2         ;TIME OUT ?
EC22 60                RTS
EC23
EC23            ;DELAY HALF BIT TIME
EC23            ;TOTAL TIME DIVIDED BY 2
EC23 AD 17 A4   DEHALF LDA CNTH30
EC26 4A                LSR A           ;LSB TO CARRY
EC27 AD 18 A4          LDA CNTL30
EC2A 6A                ROR A           ;SHIFT WITH CARRY
EC2B 8D 08 A8          STA T2L
EC2E AD 17 A4          LDA CNTH30
EC31 4A                LSR A
EC32 8D 09 A8          STA T2H
EC35 4C 1B EC          JMP DE2
EC38
EC38            ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EC38 A9 00      GETKD0 LDA #0
EC3A 8D 77 A4          STA IDOT        ;GO ANOTHER 90 DOTS
EC3D 20 50 F0          JSR IPO0        ;OUTPUT 90 DOTS TO PRI (ZEROS)
EC40
EC40            ;GET A CHAR FROM KB SUBROUTINE
EC40            ;FROM KB Y=ROW ,STBKEY=COLUMNS (STROBE)
EC40            ;X=CTRL OR SHIFT ,OTHERWISE X=0
EC40 20 EF EC   GETKEY JSR ROONEK      ;WAIT IF LAST KEY STILL DOWN
EC43 20 2A ED   GETKY  JSR DEBKEY      ;DEBOUNCE KEY (5 MSEC)
EC46            ;CTRL OR SHIFT ?
EC46 A9 8F             LDA #$8F        ;CHCK CLMN 5,6,7
EC48 8D 80 A4          STA DRA2
EC4B AD 82 A4          LDA DRB2        ;CHCK ROW 1
EC4E 4A                LSR A
EC4F B0 20             BCS GETK1       ;IF=1 ,NO CTRL OR SHIFT
EC51 A2 03             LDX #3          ;CLMN 5,6,7 (CNTRL,SHIFTL,SHIFTR)
EC53 A9 7F             LDA #$7F        ;CTRL OR SHIFT ,SO WHICH ONE?
EC55 38         GETK0  SEC
EC56 6A                ROR A
EC57 48                PHA
EC58 20 0B ED          JSR ONEK2       ;LETS GET CTRL OR SHIFT INTO X
EC5B AD 82 A4          LDA DRB2
```

```
EC5E 4A                    LSR A           ;ONLY ROW 1
EC5F 90 06                 BCC GETK00      ;GOT YOU
EC61 68                    PLA
EC62 CA                    DEX
EC63 D0 F0                 BNE GETK0
EC65 F0 DC                 BEQ GETKY       ;THERE IS A MISTAKE CHECK AGAIN
EC67 68             GETK00 PLA             ;NOW GET STBKEY INTO X
EC68 AD 2B A4              LDA STBKEY      ;CLMN INTO X
EC6B 49 FF                 EOR #$FF        ;COMPLEMENT BECAUSE STRBS ARE 0
EC6D AA                    TAX             ;CTRL OR SHIFT TO X
EC6E EE 2A A4              INC KMASK       ;SET MSK=$01
EC71                ;NOW GET ANY KEY
EC71 20 05 ED       GETK1  JSR ONEKEY      ;GET A KEY
EC74 88                    DEY             ;CHK THE ROW (1-8)
EC75 D0 09                 BNE GETK1B      ;CHK IF CTRL OR SHIFT
EC77 AD 2B A4              LDA STBKEY      ;WERE ENTERED AT THE LAST MOMENT
EC7A C9 F7                 CMP #$F7        ;IF CLMN 5,6,7,8 TO IT AGAIN
EC7C B0 04                 BCS GETK2
EC7E 90 C3                 BCC GETKY       ;SEND IT TO GET CTRL OR SHIFT
EC80 30 C1          GETK1B BMI GETKY       ;NO KEY ,CLEAR MSK
EC82                ;WE HAVE A KEY ,DECODE IT
EC82 20 2C ED       GETK2  JSR DEBK1       ;DEBOUNCE KEY (5 MSEC)
EC85 98                    TYA             ;MULT BY 8
EC86 0A                    ASL A
EC87 0A                    ASL A
EC88 0A                    ASL A
EC89 A8                    TAY             ;NOW Y HAS ROW ADDR FROM ROW 1
EC8A AD 2B A4              LDA STBKEY      ;ADD COLUMN TO Y
EC8D 4A             GETK3  LSR A
EC8E 90 03                 BCC GETK4
EC90 C8                    INY
EC91 D0 FA                 BNE GETK3
EC93 B9 21 F4       GETK4  LDA ROW1,Y      ;GET THE CHR
EC96 48                    PHA
EC97 8A                    TXA             ;SEE IF CTRL OR SHIFT WAS USED
EC98 F0 24                 BEQ GETK7       ;BRCH IF NO CTRL OR SHIFT
EC9A 29 10                 AND #$10        ;CTRL ?
EC9C F0 06                 BEQ GETK5       ;NO ,GO GETKS
EC9E 68                    PLA
EC9F 29 3F                 AND #$3F        ;MSK OFF 2 MSB FOR CONTROL
ECA1 4C BF EC              JMP GETK8       ;EXIT
ECA4 68             GETK5  PLA
ECA5 48                    PHA             ;SAVE IT
ECA6 29 40                 AND #$40        ;IF ALPHA CHARS DO NOT SHIFT
ECA8 D0 14                 BNE GETK7
ECAA 68                    PLA
ECAB 48                    PHA
ECAC 29 0F                 AND #$0F        ;ONLY LSB
ECAE F0 0E                 BEQ GETK7       ;DO NOT INTERCHANGE <SPACE> OR 0
ECB0 C9 0C                 CMP #$0C        ;ACC>=$0C ?
ECB2 B0 05                 BCS GETK6       ;YES ACC>=$0C
ECB4 68                    PLA             ;NO, ACC<$0C
ECB5 29 EF                 AND #$EF        ;STRIP OFF BIT 4
ECB7 D0 06                 BNE GETK8       ;EXIT
ECB9 68             GETK6  PLA             ;ACC>=$0C
ECBA 09 10                 ORA #$10        ;BIT 4= 1
ECBC D0 01                 BNE GETK8       ;EXIT
ECBE 68             GETK7  PLA
ECBF                ;CHECK FOR "ADV PAP","PRI LINE", OR "TOGL PRIFLG"
ECBF                ;IN THIS WAY WE DONT HAVE TO CHCK FOR THIS COMM
ECBF C9 60          GETK8  CMP #$60        ;ADV PAPER COMM
ECC1 D0 06                 BNE GETK11
```

```
ECC3 E0 00              CPX #0          ;IF SHIFT IS NOT ADV PAPER
ECC5 F0 25              BEQ GETK10      ;NO SHIFT ,SO ADVPAPER
ECC7 29 4F              AND #$4F        ;CONVRT TO "@"
ECC9 C9 1C       GETK11 CMP #$1C        ;SEE IF TOGGL PRIFLG (CONTRL PRI)
ECCB D0 14              BNE GETK13
ECCD 20 E1 E6           JSR PRITR       ;GO TOGGLE FLG
ECD0 A0 01              LDY #1          ;GET THE PTRS BACK 3 SPACES
ECD2 B9 15 A4    GETK12 LDA CURPO2,Y
ECD5 38                 SEC
ECD6 E9 03              SBC #3          ;BECAUSE "ON ,OFF" MSGS
ECD8 99 15 A4           STA CURPO2,Y
ECDB 88                 DEY
ECDC 10 F4              BPL GETK12
ECDE 4C 40 EC           JMP GETKEY
ECE1 C9 5C       GETK13 CMP #BACKSLASH  ;PRINT LINE COMMAND
ECE3 D0 06              BNE GETK14
ECE5 20 4A F0           JSR IPS0        ;PRINT WHATEVER IS IN BUFFER
ECE8 4C 40 EC           JMP GETKEY
ECEB 60          GETK14 RTS
ECEC 4C 38 EC    GETK10 JMP GETKD0
ECEF
ECEF             ;WAIT IF LAST KEY STILL DOWN  (ROLLOVER)
ECEF AD 82 A4    ROONEK LDA DRB2        ;SEE IF KEY STILL DOWN
ECF2 C9 FF              CMP #$FF
ECF4 F0 0A              BEQ ROO1        ;NO KEY AT ALL, CLR ROLLFL
ECF6 0D 7F A4           ORA ROLLFL      ;ACCEPT ONLY LAST KEY
ECF9 49 FF              EOR #$FF        ;STRBS ARE ZEROS TO INVER
ECFB D0 F2              BNE ROONEK
ECFD 20 2A ED           JSR DEBKEY      ;CLR KMASK & DEBOUNCE RELEASE
ED00 A9 00       ROO1   LDA #0          ;CLR KMASK
ED02 8D 2A A4           STA KMASK
ED05             ;GO THRU KB ONCE AND RTN ,IF ANY
ED05             ;KEY Y=ROW (1-8) & STBKEY=CLMN
ED05             ;IF NO KEY Y=0 ,STBKEY=$FF
ED05 A9 7F       ONEKEY LDA #$7F        ;FIRST STROBE TO MSB
ED07 D0 02              BNE ONEK2       ;START AT ONEK2
ED09 38          ONEK1  SEC             ;ONLY ONE PULSE (ZERO)
ED0A 6A                 ROR A           ;SHIFT TO RIGHT
ED0B 8D 80 A4    ONEK2  STA DRA2        ;OUTPUT CLMN STROBE
ED0E 8D 2B A4           STA STBKEY      ;SAVE IT
ED11 A0 08              LDY #8          ;CHECK 8 ROWS
ED13 AD 82 A4           LDA DRB2        ;ANY KEY ?
ED16 0D 2A A4           ORA KMASK       ;DISABLE ROW 1 IF CTRL OR SHIFT
ED19 8D 7F A4           STA ROLLFL      ;SAVE WHICH KEY IT WAS
ED1C 0A          ONEK3  ASL A
ED1D 90 0A              BCC ONEK4       ;JUMP IF KEY (ZERO)
ED1F 88                 DEY
ED20 D0 FA              BNE ONEK3
ED22 AD 2B A4           LDA STBKEY
ED25 C9 FF              CMP #$FF        ;LAST CLMN ?
ED27 D0 E0              BNE ONEK1       ;NO ,DO NEXT CLMN
ED29 60          ONEK4  RTS
ED2A
ED2A A2 00       DEBKEY LDX #0          ;CLEAR CNTRL OR SHIFT
ED2C A9 00       DEBK1  LDA #0          ;CLR KMASK
ED2E 8D 2A A4           STA KMASK
ED31 A9 88              LDA #DEBTIM     ;DEBOUNCE TIME FOR KEYBOARD
ED33 8D 08 A8           STA T2L
ED36 A9 13              LDA #DEBTIM/256
ED38 4C 18 EC           JMP DE1         ;WAIT FOR 5 MSEC
ED3B
ED3B             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
ED3B                ;GET A CHAR FROM TAPE SUBROUTINE
ED3B                ;A BUFFER IS USED TO GET BLOCKS OF DATA
ED3B                ;FROM TAPE ,EXCEPT WHEN FORMAT EQUAL TO
ED3B                ;KIM-1 (THE WHOLE FILE IS LOADED AT ONE TIME)
ED3B 20 9E EB       TIBYTE JSR PHXY        ;PUSH X
ED3E AE 36 A4              LDX TAPTR       ;POINTER FOR BUFFER
ED41 E0 50                 CPX #80         ;IS BUFFER EMPTY ?
ED43 D0 03                 BNE TIB1
ED45 20 53 ED              JSR TIBY1       ;LOAD ANOTHER BLOCK
ED48 BD 16 01       TIB1   LDA TABUFF,X
ED4B E8                    INX
ED4C 8E 36 A4              STX TAPTR
ED4F 20 AC EB              JSR PLXY        ;PULL X
ED52 60                    RTS
ED53                ;LOAD A BLOCK FROM TAPE INTO BUFFER
ED53 20 EA ED       TIBY1  JSR TAISET      ;SET TAPE FOR INPUT
ED56 20 29 EE       TIBY3  JSR GETTAP      ;GET A CHAR FROM TAPE
ED59 C9 23                 CMP #'#'        ;CHECK FIRST CHR FOR
ED5B F0 06                 BEQ TIBY4       ;START OF BLOCK
ED5D C9 16                 CMP #$16        ;IF NOT # SHOULD BE SYN
ED5F D0 F2                 BNE TIBY1
ED61 F0 F3                 BEQ TIBY3
ED63 A2 00          TIBY4  LDX #0
ED65 20 29 EE       TIBY5  JSR GETTAP      ;NOW LOAD INTO BUFFER
ED68 9D 16 01              STA TABUFF,X
ED6B E8                    INX
ED6C E0 52                 CPX #82
ED6E D0 F5                 BNE TIBY5
ED70 AD 00 A8              LDA DRB
ED73 29 CF                 AND #$CF
ED75 8D 00 A8              STA DRB         ;TURN OFF TAPES
ED78 58                    CLI             ;ENABL INTERR
ED79 20 BD ED              JSR ADDBK1      ;DISPLAY BLK COUNT
ED7C A2 00                 LDX #0          ;TO CLEAR PTR IN TIBYTE
ED7E AD 15 01              LDA BLK         ;CHECK THE BLOCK COUNT
ED81 F0 05                 BEQ TIBY5A      ;IF FIRST BLK ,DO NOT CMP
ED83 DD 16 01              CMP TABUFF,X
ED86 D0 28                 BNE TIBY7       ;BRANCH IF WE MISSED ONE BLOCK
ED88 E8             TIBY5A INX
ED89 8E 36 A4              STX TAPTR
ED8C EE 15 01              INC BLK         ;INCR BLK CONT
ED8F AD 67 01              LDA TABUFF+81   ;STORE THIS BLK CKSUM
ED92 48                    PHA
ED93 AD 66 01              LDA TABUFF+80
ED96 48                    PHA
ED97 CE 12 A4              DEC INFLG       ;SET INFLG DIFF FROM OUTFLG
ED9A 20 E7 F1              JSR BKCKSM      ;COMPUT BLK CKSUM FOR THIS BLK
ED9D 68                    PLA
ED9E CD 66 01              CMP TABUFF+80   ;DO THEY AGREE ?
EDA1 D0 0C                 BNE TIBY6
EDA3 68                    PLA
EDA4 CD 67 01              CMP TABUFF+81
EDA7 D0 07                 BNE TIBY7
EDA9 EE 12 A4              INC INFLG       ;RESTORE INPUT DEVICE
EDAC A2 01                 LDX #1          ;TO GET FIRST CHR IN TIBYTE
EDAE 60                    RTS
EDAF 68             TIBY6  PLA             ;RESTORE STACK PTR
EDB0 68             TIBY7  PLA
EDB1 68                    PLA
EDB2 68                    PLA
EDB3 68                    PLA
EDB4 20 8E E3              JSR CKER0
```

```
EDB7 4C A1 E1          JMP COMIN
EDBA
EDBA            ;ADD 1 TO BLK COUNT AND OUTPUT IT
EDBA EE 15 01   ADDBLK INC BLK          ;INCR BLK CNT
EDBD EE 11 A4   ADDBK1 INC PRIFLG       ;SO DONT OUTPUT TO PRINTR
EDC0 A9 12             LDA #18          ;ONLY OUTPUT IN THIS POSITION
EDC2 8D 15 A4          STA CURPO2
EDC5 AD 4A A4          LDA DIBUFF+18    ;SAVE DISBUF (FOR EDIT)
EDC8 48                PHA
EDC9 AD 4B A4          LDA DIBUFF+19
EDCC 48                PHA
EDCD AE 13 A4          LDX OUTFLG       ;SAVE OUTFLG
EDD0 A9 0D             LDA #CR
EDD2 8D 13 A4          STA OUTFLG       ;TO OUTPUT TO TERMINAL
EDD5 AD 16 01          LDA BLK+1        ;BLK CNT COMING FROM TAPE
EDD8 20 46 EA          JSR NUMA         ;OUTPUT IN ASCII
EDDB 8E 13 A4          STX OUTFLG       ;RESTORE OUTFLG
EDDE 68                PLA
EDDF 8D 4B A4          STA DIBUFF+19
EDE2 68                PLA
EDE3 8D 4A A4          STA DIBUFF+18
EDE6 CE 11 A4          DEC PRIFLG       ;RESTORE PRI FLG
EDE9 60                RTS
EDEA
EDEA            ;SET TAPE (1 OR 2) FOR INPUT
EDEA A9 37      TAISET LDA #$37         ;SET PB7 FOR INPUT
EDEC 8D 02 A8          STA DDRB
EDEF AD 34 A4          LDA TAPIN        ;INPUT FLG (TAP 1=2 OR TAP 2=1)
EDF2 20 1C EE          JSR TIOSET       ;RESET PB4 OR PB5
EDF5 A9 EE             LDA #MOFF+DATIN ;SET CA2=1 (DATA IN)
EDF7 8D 0C A8          STA PCR
EDFA A9 FF             LDA #$FF         ;PREPARE T2
EDFC 8D 08 A8          STA T2L          ;LACTH
EDFF            ;CHCK BIT BY BIT UNTIL $16
EDFF 20 3B EE   SYNC   JSR RDBIT        ;GET A BIT IN MSB
EE02 4E 2A A4          LSR CPIY         ;MAKE ROOM FOR BIT
EE05 0D 2A A4          ORA CPIY         ;PUT BIT INTO MSB
EE08 8D 2A A4          STA CPIY
EE0B C9 16             CMP #$16         ;SYN CHAR ?
EE0D D0 F0             BNE SYNC
EE0F A2 05             LDX #$05         ;TEST FOR 5 SYN CHARS
EE11 20 29 EE   SYNC1  JSR GETTAP
EE14 C9 16             CMP #$16
EE16 D0 E7             BNE SYNC         ;IF NOT 2 CHAR RE-SYNC
EE18 CA                DEX
EE19 D0 F6             BNE SYNC1
EE1B 60                RTS
EE1C
EE1C            ;SET PB4 OR PB5 OFF
EE1C            ;USED BY IN/OUT SET UPS
EE1C D0 04      TIOSET BNE TIOS1        ;BRCH IF TAP1
EE1E A9 14             LDA #$14         ;SET TAP 2 OFF (PB5=0)
EE20 D0 02             BNE TIOS2
EE22 A9 24      TIOS1  LDA #$24         ;SET TAP 1 OFF (PB4=0)
EE24 8D 00 A8   TIOS2  STA DRB
EE27 78                SEI              ;DISABLE INTERR WHILE TAP
EE28 60                RTS
EE29
EE29            ;GET 1 CHAR FROM TAPE AND RETURN
EE29            ;WITH CHR IN ACC, USE CPIY TO ASM CHR ,USES Y
EE29 A0 08      GETTAP LDY #$08         ;READ 8 BITS
EE2B 20 3B EE   GETA1  JSR RDBIT        ;GET NEXT DATA BIT
```

```
EE2E 4E 2A A4          LSR CPIY         ;MAKE ROOM FOR MSB
EE31 0D 2A A4          ORA CPIY         ;OR IN SIGN BIT
EE34 8D 2A A4          STA CPIY         ;REPLACE CHAR
EE37 88                DEY
EE38 D0 F1             BNE GETA1
EE3A 60                RTS
EE3B            ;GET ONE BIT FROM TAPE AND
EE3B            ;RETURN IT IN SIGN OF A (MSB)
EE3B AD 08 A4   RDBIT  LDA TSPEED       ;ARE WE IN C7 OR 5B,5A FREQUENC`
EE3E 30 27             BMI RDBIT4       ;JUMP TO C7 FREQ FORMAT
EE40 20 75 EE          JSR CKFREQ       ;START BIT IN HIGH FREQ
EE43 20 75 EE   RDBIT1 JSR CKFREQ       ;HIGH TO LOW FREQ TRANS
EE46 B0 FB             BCS RDBIT1
EE48 AD 96 A4          LDA DIV64        ;GET HIGH FREQ TIMING
EE4B 48                PHA
EE4C A9 FF             LDA #$FF         ;SET UP TIMER
EE4E 8D 96 A4          STA DIV64
EE51 20 75 EE   RDBIT2 JSR CKFREQ       ;LOW TO HIGH FREQ TRANS
EE54 90 FB             BCC RDBIT2       ;WAIT TILL FREQ IS HIGH
EE56 68                PLA
EE57 38                SEC
EE58 ED 96 A4          SBC DIV64        ;(256-T1) - (256-T2) =T2-T1
EE5B 48                PHA              ;LOW FREQ TIME-HIGH FREQ TIME
EE5C A9 FF             LDA #$FF
EE5E 8D 96 A4          STA DIV64        ;SET UP TIMER
EE61 68                PLA
EE62 49 FF             EOR #$FF
EE64 29 80             AND #$80
EE66 60                RTS
EE67            ;EACH BIT STARTS WITH HALF PULSE OF 2400 & THEN
EE67            ;3 HALF PULSES OF 1200 HZ FOR 0 ,3 PUSLES OF 2400 FOR 1
EE67            ;THE READING IS MADE ON THE FOURTH 1/2 PULSE ,WHERE
EE67            ;THE SIGNAL HAS STABILIZED
EE67 20 75 EE   RDBIT4 JSR CKFREQ       ;SEE WHICH FREQ
EE6A 90 FB             BCC RDBIT4
EE6C 20 75 EE          JSR CKFREQ
EE6F 20 75 EE          JSR CKFREQ
EE72 4C B5 FF          JMP PATC24       ;NOW READ THE BIT
EE75
EE75 2C 00 A8   CKFREQ BIT DRB          ;ARE WE HIGH OR LOW ?
EE78 30 27             BMI CKF4
EE7A 2C 00 A8   CKF1   BIT DRB          ;WAIT TILL HIGH
EE7D 10 FB             BPL CKF1
EE7F 65 00             ADC $00          ;EQUALIZER
EE81 AD 09 A8   CKF2   LDA T2H          ;SAVE CNTR
EE84 48                PHA
EE85 AD 08 A8          LDA T2L
EE88 48                PHA
EE89 A9 FF             LDA #$FF
EE8B 8D 09 A8          STA T2H          ;START CNTR
EE8E AD 08 A4          LDA TSPEED
EE91 30 06             BMI CKF3         ;SUPER SPEED ?
EE93 68                PLA
EE94 CD 08 A4          CMP TSPEED       ;HIGH OR LOW FREC
EE97 68                PLA              ;C=1 IF HIGH ,C=0 IF LOW
EE98 60                RTS
EE99 68         CKF3   PLA
EE9A CD 08 A4          CMP TSPEED       ;CENTER FREQ
EE9D 68         CKF3A  PLA
EE9E E9 FE             SBC #$FE
EEA0 60                RTS
EEA1 2C 00 A8   CKF4   BIT DRB          ;WAIT TILL LOW
```

```
EEA4 30 FB              BMI CKF4
EEA6 10 D9              BPL CKF2        ;GO GET TIMING
EEA8
EEA8             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EEA8             ;OUTPUT ACC TO TTY SUBROUTINE
EEA8             ;X,Y ARE PRESERVED
EEA8 48          OUTTTY PHA             ;SAVE A
EEA9 20 9E EB           JSR PHXY        ;PUSH X
EEAC 8D 27 A4           STA STIY        ;PUT CHAR HERE
EEAF 20 0F EC           JSR DELAY       ;STOP BIT FROM LAST CHAR
EEB2 AD 00 A8           LDA DRB
EEB5 29 FB              AND #$FB        ;START BIT PB2=0
EEB7 8D 00 A8           STA DRB         ;TTO=PB2
EEBA 8D 28 A4           STA STIY+1      ;SAVE THIS PATTERN
EEBD 20 0F EC           JSR DELAY
EEC0 A2 08              LDX #$08        ;8 BITS
EEC2 2E 27 A4           ROL STIY        ;GET FIRST LSB INTO BIT 2
EEC5 2E 27 A4           ROL STIY
EEC8 2E 27 A4           ROL STIY
EECB 6E 27 A4    OUTT1  ROR STIY
EECE AD 27 A4           LDA STIY
EED1 29 04              AND #$04        ;GET ONLY BIT 2 FOR PB2
EED3 0D 28 A4           ORA STIY+1      ;PUT BIT INTO PATTERN
EED6 8D 00 A8           STA DRB         ;NOW TO TTY
EED9 08                 PHP             ;PRESERVE CARRY FOR ROTATE
EEDA 20 0F EC           JSR DELAY
EEDD 28                 PLP
EEDE CA                 DEX
EEDF D0 EA              BNE OUTT1
EEE1 A9 04              LDA #$04        ;STOP BIT
EEE3 0D 28 A4           ORA STIY+1
EEE6 8D 00 A8           STA DRB
EEE9 20 0F EC           JSR DELAY       ;STOP BIT
EEEC 20 AC EB           JSR PLXY        ;PULL X
EEEF 68                 PLA
EEF0 C9 0A              CMP #LF
EEF2 F0 07              BEQ OUTT2
EEF4 C9 FF              CMP #NULLC
EEF6 F0 03              BEQ OUTT2
EEF8 4C 05 EF           JMP OUTDIS      ;USE THAT BUFF
EEFB 60          OUTT2  RTS
EEFC
EEFC             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EEFC             ;OUTPUT A CHR TO D/P SUBR (SINGLE ENTRY FOR BOTH SUBR)
EEFC             ;IF CHAR=<CR> CLEAR DISPLAY & PRINTER
EEFC 20 00 F0    OUTDP  JSR OUTPRI      ;FIRST TO PRI THEN TO DISP
EEFF EA                 NOP
EF00 EA                 NOP
EF01 EA                 NOP
EF02 6C 06 A4    OUTDP1 JMP (DILINK)    ;HERE HE COULD ECHO SOMEWHERE ELSE`
EF05
EF05             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
EF05             ;OUTPUT ACC TO DISPLAY SUBROUTINE
EF05             ;IF SIGN BIT (MSB)=1 DISP DO NOT CLR TO THE RIGHT
EF05 48          OUTDIS PHA             ;SAVE A
EF06 20 9E EB           JSR PHXY        ;PUSH X
EF09 C9 0D              CMP #CR         ;<CR> ?
EF0B D0 07              BNE OUTD1
EF0D A2 00              LDX #0          ;YES
EF0F 8E 15 A4           STX CURPO2      ;CLEAR DISP POINTER
EF12 F0 42              BEQ OUTD5       ;GO CLEAR DISP
EF14 4C 9C FE    OUTD1  JMP PATCH4
```

```
EF17 E0 3C       OUTD1A CPX #60         ;LAST CHAR FOR DISP?
EF19 90 05              BCC OUTD2
EF1B 20 AC EB           JSR PLXY        ;GO BACK
EF1E 68                 PLA             ;DO NOT STORE
EF1F 60                 RTS
EF20 9D 38 A4    OUTD2  STA DIBUFF,X    ;PUT CHAR IN BUFF
EF23 EE 15 A4           INC CURPO2      ;INC POINTER
EF26 E0 14              CPX #20         ;DISPLAY FULL?
EF28 90 1E              BCC OUTD4
EF2A 20 2F EF           JSR OUTD2A      ;THIS WAY SCROLL IS A SUBR
EF2D 30 47              BMI OUTD7       ;EXIT DISP
EF2F             ;YES, SCROLL CHARS TO THE LEFT
EF2F 8A          OUTD2A TXA             ;X---> Y
EF30 A8                 TAY
EF31 A2 13              LDX #19         ;ADDR FOR DISP DO NOT
EF33 8E 27 A4    OUTD3  STX STIY        ;DECREM IN BINARY
EF36 B9 38 A4           LDA DIBUFF,Y    ;FROM BUFFER TO DISP
EF39 09 80              ORA #$80        ;NO CURSOR
EF3B 20 7B EF           JSR OUTDD1      ;CONVERT X INTO REAL ADDR
EF3E 88                 DEY
EF3F CE 27 A4           DEC STIY
EF42 AE 27 A4           LDX STIY
EF45 10 EC              BPL OUTD3       ;AGAIN UNTIL WHOLE DISP
EF47 60                 RTS
EF48 48          OUTD4  PHA
EF49 09 80              ORA #$80        ;NO CURSOR
EF4B 20 7B EF           JSR OUTDD1      ;X=<$19 ,CONVRT TO REAL ADDR
EF4E 68                 PLA
EF4F 29 80              AND #$80        ;IF MSB=0 CLEAR REST OF DISPLAY
EF51 D0 23              BNE OUTD7
EF53 AE 15 A4           LDX CURPO2
EF56             ;CLEAR DISP TO THE RIGHT
EF56 E0 14       OUTD5  CPX #20
EF58 B0 1C              BCS OUTD7
EF5A 8E 27 A4           STX STIY
EF5D A9 A0              LDA #' '+$80    ;<SPACE>
EF5F 20 7B EF           JSR OUTDD1      ;CONVRT TO REAL ADDR
EF62 EE 27 A4           INC STIY
EF65 AE 27 A4           LDX STIY
EF68 D0 EC              BNE OUTD5       ;GO NEXT
EF6A 4C 76 EF           JMP OUTD7
EF6D EA                 NOP
EF6E EA                 NOP
EF6F EA                 NOP
EF70 EA                 NOP
EF71 EA                 NOP
EF72 EA                 NOP
EF73 EA                 NOP
EF74 EA                 NOP
EF75 EA                 NOP
EF76 20 AC EB    OUTD7  JSR PLXY        ;REST ,SO PRINTR INDEPEN
EF79 68                 PLA
EF7A 60                 RTS
EF7B
EF7B             ;CONVERT X INTO REAL ADDR FOR DISPLAY
EF7B             ;AND OUTPUT IT  PB=DATA ; PA=W,CE ,A0 A1 (6520)
EF7B 48          OUTDD1 PHA             ;SAVE DATA
EF7C 8A                 TXA
EF7D 48                 PHA             ;SAVE X
EF7E 4A                 LSR A           ;DIVIDE X BY 4
EF7F 4A                 LSR A           ;TO GET CHIP SELECT
EF80 AA                 TAX             ;BACK TO X
```

```
EF81 A9 04              LDA #4          ;FIRST CHIP SELECT
EF83 E0 00              CPX #0          ;FIRST CHIP ?
EF85 F0 04              BEQ OUTDD3
EF87 0A          OUTDD2 ASL A
EF88 CA                 DEX
EF89 D0 FC              BNE OUTDD2      ;BACK TILL RIGH CS
EF8B 8D 28 A4    OUTDD3 STA STIY+1      ;SAVE CS TEMPORARILY
EF8E 68                 PLA             ;GET X AGAIN FOR CHAR
EF8F 29 03              AND #$03        ;IN THAT CHIP
EF91 0D 28 A4           ORA STIY+1      ;OR IN CS AND CHAR
EF94             ;STORE ADDR AND DATA INTO DISPL
EF94 49 FF              EOR #$FF        ;W=1 , CE=0 & A1,A0
EF96 8D 00 AC           STA RA
EF99 AA                 TAX             ;SAVE A IN X
EF9A 68                 PLA             ;GET DATA
EF9B 48                 PHA
EF9C 8D 02 AC           STA RB
EF9F 8A                 TXA
EFA0 49 80              EOR #$80        ;SET W=0
EFA2 8D 00 AC           STA RA
EFA5 EA                 NOP
EFA6 09 7C              ORA #$7C        ;SET CE=1
EFA8 8D 00 AC           STA RA
EFAB A9 FF              LDA #$FF        ;SET W=1
EFAD 8D 00 AC           STA RA
EFB0 68                 PLA             ;RETURN DATA
EFB1 60                 RTS
EFB2
EFF9                    *=$EFF9
EFF9 EA                 .DB $EA
F000                    *=$F000
F000             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
F000             ;OUTPUT ACC TO PRINTER SUBROUTINE
F000             ;PRINTS ON 21RST CHAR OR WHEN <CR>
F000             ;IT WILL PUT IT ON BUBFFER BUT WONT PRINT IF
F000             ;PRIFLG=0
F000 48          OUTPRI PHA             ;SAVE CHR TO BE OUTPUT
F001 20 9E EB           JSR PHXY        ;SAVE X
F004 C9 0D              CMP #CR         ;SEE IF CR
F006 F0 07              BEQ OUT01       ;YES SO PRINT THE BUFF
F008 AE 16 A4           LDX CURPOS      ;PTR TO NEXT POS IN BUFF
F00B E0 14              CPX #20         ;SEE IF BUFF FULL
F00D D0 16              BNE OUT04       ;NOT FULL SO RETURN
F00F             ;<CR> SO FILL REST OF BUFFER WITH BLANKS
F00F 48          OUT01  PHA
F010 A9 00              LDA #0          ;CURPOS = 0
F012 AE 16 A4           LDX CURPOS      ;SEE IF ANYTHING IN BUFFER
F015 8D 16 A4           STA CURPOS
F018 20 38 F0           JSR OUTPR       ;CLEAR PRIBUF TO THE RIGHT
F01B             ;BUFFER FILLED SO PRINT IT
F01B 20 45 F0           JSR IPST        ;START THE PRINT
F01E A2 00              LDX #0          ;STORE CHR IN BUFF (FIRST LOC)
F020 68                 PLA             ;GET IT
F021 C9 0D              CMP #CR         ;DONT STORE IF <CR>
F023 F0 0E              BEQ OUT05
F025 9D 60 A4    OUT04  STA IBUFM,X     ;STORE CHR IN BUFF
F028 EE 16 A4           INC CURPOS      ;INCR BUFF PNTR
F02B E8                 INX
F02C 29 80              AND #$80
F02E D0 03              BNE OUT05       ;DONT CLR IF MSB=1
F030 20 38 F0           JSR OUTPR       ;CLEAR PRIBUFF TO THE RIGHT
F033 20 AC EB    OUT05  JSR PLXY        ;RESTORE REGS
```

```
F036 68                    PLA
F037 60                    RTS
F038 A9 20        OUTPR  LDA #' '        ;FILL REST OF BUFF WITH BLANKS
F03A E0 14        OUTPR1 CPX #20         ;SEE IF END OF BUFF
F03C F0 06               BEQ OUTPR2
F03E 9D 60 A4            STA IBUFM,X     ;NO SO STORE BLANK
F041 E8                  INX             ;INCR BUFF PNTR
F042 10 F6               BPL OUTPR1
F044 60           OUTPR2 RTS
F045
F045              ;SUB TO OUTPUT BUFFER, 70 DOTS (10 DOTS AT
F045              ;A TIME BY 7 ROWS) FOR EACH LINE OF PRINTING
F045 2C 11 A4     IPST   BIT PRIFLG      ;PRINT FLG ON ?
F048 10 2E               BPL IPO4
F04A 20 CB F0     IPS0   JSR PINT        ;INITIALIZE VALUES
F04D 20 E3 F0            JSR IPSU        ;SET UP FIRS OUTPUT PATTERN
F050 A9 C1        IPO0   LDA #PRST+SP12+MON ;TURN MOTOR ON
F052 8D 0C A8            STA PCR
F055 20 A0 FF            JSR PAT23       ;TIME OUT ?
F058 D0 0C               BNE IPO2        ;NO, START SIGNAL RECEIVED
F05A 20 A0 FF            JSR PAT23       ;YES, TRY AGAIN
F05D D0 07               BNE IPO2        ;OK
F05F 4C 79 F0            JMP PRIERR      ;TWO TIME OUTS - ERROR
F062 EA                  NOP
F063 EA                  NOP
F064 EA                  NOP
F065 EA                  NOP
F066 20 87 F0     IPO2   JSR PRNDOT      ;STRB P1=1 PRINT DOTS (1.7MSEC)
F069 20 87 F0            JSR PRNDOT      ;STRB P2=1 PRINT DOTS (1.7MSEC)
F06C              ;CHECK FOR 90, WHEN 70 PRNDOT WILL OUTPUT ZEROS
F06C AD 77 A4            LDA IDOT
F06F C9 5A               CMP #90
F071 90 F3               BCC IPO2        ;L.T. 90 THEN GO STROB P1
F073 A9 E1        IPO3   LDA #PRST+SP12+MOFF ;TURN MOTOR OFF
F075 8D 0C A8            STA PCR
F078 60           IPO4   RTS
F079
F079 20 44 EB     PRIERR JSR CLR         ;CLEAR PRI PNTR
F07C 20 B1 FE            JSR PATCH5      ;TURN PRI OFF
F07F A0 3B               LDY #M12-M1
F081 20 AF E7            JSR KEP
F084 4C A1 E1            JMP COMIN       ;BACK WHERE SUBR WAS CALLED
F087
F087              ;SUBR TO INCR DOT COUNTER,WHEN
F087              ;NEG TRANS OUTPUT CHR FOR 1.7 MSEC
F087              ;CLEAR & SET UP NEXT PATTERN
F087 A9 00        PRNDOT LDA #0          ;CLR INTERRPTS
F089 8D 01 A8            STA DRAH
F08C AD 0D A8     PRDOT0 LDA IFR
F08F 29 02               AND #MSP12      ;ANY STROBES ?
F091 F0 F9               BEQ PRDOT0
F093 AD 0C A8            LDA PCR
F096 49 01               EOR #$01
F098 8D 0C A8            STA PCR
F09B EE 77 A4            INC IDOT
F09E AD 79 A4            LDA IOUTU       ;2 LEFT ELEM
F0A1 0D 00 A8            ORA DRB         ;DO NOT TURN TTY OUTPUT OFF
F0A4 8D 00 A8            STA DRB
F0A7 AD 78 A4            LDA IOUTL       ;7 RIGHT ELEM, CLR CA1 INTER FLG
F0AA 8D 01 A8            STA DRAH
F0AD A9 A4               LDA #PRTIME
F0AF 8D 08 A8            STA T2L
```

```
F0B2 A9 06               LDA #PRTIME/256 ;START T2 FOR 1.7 MSEC
F0B4 8D 09 A8            STA T2H
F0B7 20 E3 F0            JSR IPSU        ;SET NEXT PATTERN WHILE WAITING
F0BA 20 1B EC            JSR DE2         ;WAIT TILL TIME OUT
F0BD A9 00               LDA #0          ;THERMAL ELEM OFF
F0BF 8D 01 A8            STA DRAH
F0C2 AD 00 A8            LDA DRB         ;BUT DONT CHANGE TAPE CONTROLS
F0C5 29 FC               AND #$FC
F0C7 8D 00 A8            STA DRB
F0CA 60                  RTS
F0CB
F0CB              ; SUBROUTINE PINT -- INIT VARS FOR PRINTER
F0CB A9 FF        PINT   LDA #$FF
F0CD 8D 74 A4            STA IDIR        ;DIRECTION <= -
F0D0 A9 05               LDA #5
F0D2 8D 75 A4            STA ICOL        ;COLUMN <= LEFTMOST +1
F0D5 A9 01               LDA #1
F0D7 8D 76 A4            STA IOFFST      ;OFFSET <= LEFT CHARACTER
F0DA 8D 7C A4            STA IMASK
F0DD A9 00               LDA #0
F0DF 8D 77 A4            STA IDOT        ;DOT COUNTER <= 0
F0E2 60                  RTS
F0E3
F0E3              ;THE VARIABLES FOR THE PRINTER ARE AS FOLLOWS:
F0E3              ;
F0E3              ;IDIR   DIRECT HEAD IS CURRENTLY MOVING (0=+, $FF=-)
F0E3              ;ICOL   CLMN TO BE PRNTED NEXT (LEFTMOST=0,RIGHTMOST=4)
F0E3              ;IOFFST OFFSET N PRINT BUFF (0=LEFT CHR, 1=RIGHT CHR)
F0E3              ;IDOT   COUNT OF NUMBER OF DOTS PRINTED THUS FAR
F0E3              ;IOUTL  SOLENOID PATTERN (8 CHRS ON RIGHT)
F0E3              ;IOUTU  SOLENOID PATTERN (2 CHRS ON LEFT)
F0E3              ;IBITL  1 BIT MSK USED IN SETTING NEXT SOLENOID VALUE
F0E3              ;IBITU  UPPER PART OF MASK
F0E3              ;IBUFM  START OF PRINT BUFFER (LEFTMOST CHR FIRST)
F0E3              ;IMASK  MASK FOR CURRENT ROW BEING PRINTED
F0E3              ;JUMP   ADDRESS OF TABLE FOR CURRENT COLUMN
F0E3              ;
F0E3              ;   THE DOT PATTERNS FOR THE CHRS ARE STORED SO THAT...
F0E3              ;EACH BYTE CONTAINS THE DOTS FOR ONE COLUMN OF ONE...
F0E3              ;CHR. SINCE EACH COLUMN CONTAINS SEVEN DOTS ,
F0E3              ;THIS MEANS THAT ONE BIT PER BYTE IS UNUSED.
F0E3              ;    THE PATTERNS ARE ORGANIZED INTO 5 TABLES OF 64...
F0E3              ;BYTES WHERE EACH TABLE CONTAINS ALL THE DOT...
F0E3              ;PATTERNS FOR A PARTICULAR COLUMN. THE BYTES IN EACH...
F0E3              ;TABLE ARE ORDERED ACCORDING TO THE CHR CODE OF...
F0E3              ;THE CHR BEING REFERENCED. THE CHR CODE CAN...
F0E3              ;THUS BE USED TO DIRECTLY INDEX INTO THE TABLE.
F0E3
F0E3              ;SUBROUTINE IPSU -- SET UP OUTPUT PATTERN FOR PRINTER
F0E3              ;   THIS ROUTINE IS CALLED IN ORDER TO
F0E3              ;SET UP THE NEXT GROUP OF SOLENOIDS TO
F0E3              ;BE OUTPUT TO THE PRINTER.
F0E3              ;   ON ENTRY THE CONTENTS OF ALL REGISTERS
F0E3              ;ARE ARBITRARY
F0E3              ;   ON EXIT THE CONTENTS OF A,X,Y ARE UNDEFINED
F0E3 A2 00        IPSU   LDX #0          ;X POINTS TO VAR BLOCK FOR PRNTR
F0E5 20 21 F1            JSR INCP        ;ADVANCE PTRS TO NXT DOT POSITION
F0E8              ;X NOW CONTAINS INDEX INTO PRINT BUFFER
F0E8 BD 60 A4     IPS1   LDA IBUFM,X     ;LOAD NEXT CHAR FROM BUFFER
F0EB 29 3F               AND #$3F
F0ED A8                  TAY
F0EE A9 7D               LDA #JUMP       ;A<= DOT PATTERN FOR CHAR & COL
```

```
F0F0 20 58 EB            JSR LDAY
F0F3 2C 7C A4            BIT IMASK       ;SEE IF DOT IS SET
F0F6 F0 16               BEQ IPS2        ;NO SO GO ON TO NEXT CHAR
F0F8 AD 7A A4            LDA IBITL       ;DOT ON SO SET THE CURR SOLENOID
F0FB F0 08               BEQ IPS3        ;LSB OF SOL MASK IS 0 , DO MSB
F0FD 0D 78 A4            ORA IOUTL       ;SET THE SOLENOID IN THE PATTERN
F100 8D 78 A4            STA IOUTL
F103 D0 09               BNE IPS2        ;BRANCH ALWAYS
F105 AD 7B A4     IPS3   LDA IBITU       ;SOLENOID IS ONE OF THE 2 MSD
F108 0D 79 A4            ORA IOUTU       ;SET THE BIT IN THE PATTERN
F10B 8D 79 A4            STA IOUTU
F10E 0E 7A A4     IPS2   ASL IBITL       ;SHIFT MSK TO NXT CHR POSITION
F111 2E 7B A4            ROL IBITU
F114 CA                  DEX             ;DECR PTR INTO BUFFER
F115 CA                  DEX
F116 10 D0               BPL IPS1        ;NOT END YET
F118              ;SOLENOID PATTERN IS SET UP IN IOUTU,IOUTL
F118 AD 79 A4            LDA IOUTU       ;LEFTMOST 2
F11B 29 03               AND #$03        ;DISABLE FOR SEGMENTS
F11D 8D 79 A4            STA IOUTU
F120 60                  RTS
F121
F121              ; SUBROUTINE INCP
F121              ;THIS SUBROUTINE IS USED TO UPDATE THE PRINTER VARIABLES
F121              ;TO POINT TO THE NEXT DOT POSITION TO BE PRINTED
F121              ;X REG IS USED TO POINT TO THE VARIABLE BLOCK OF
F121              ;BEING UPDATED
F121              ;ON EXIT X CONTAINS THE POINTER TO THE LAST CHARACTER IN
F121              ;THE PRINT BUFFER
F121              ;CONTENTS OF A,Y ON EXIT ARE ARBITRARY
F121 BD 74 A4     INCP   LDA IDIR,X      ;EXAMINE DIRECTION(+ OR -)
F124 10 1E               BPL OP03        ;DIRECTION = +
F126              ;*DIRECTION = -
F126 BD 75 A4            LDA ICOL,X      ;SEE WHAT THE COLUMN IS
F129 F0 05               BEQ OP04        ;COLUMN = 0 SO END OF DIGIT
F12B              ;**COLUMN # 0 SO JUST DECREMENT COLUMN
F12B DE 75 A4            DEC ICOL,X
F12E 10 33               BPL NEWCOL      ;BRANCH ALWAYS
F130              ;**COLUMN = 0 SO SEE IF EVEN OR ODD DIGIT
F130 BD 76 A4     OP04   LDA IOFFST,X
F133 F0 0A               BEQ OP07        ;OFFSET = 0 SO DIRECTION CHANGE
F135              ;***OFFSET = 1 SO MOVE TO RIGHT DIGIT
F135 DE 76 A4            DEC IOFFST,X    ;OFFSET <= 0 (LEFT CHARACTER)
F138 A9 04               LDA #4          ;COLUMN <= 4
F13A 9D 75 A4            STA ICOL,X
F13D 10 24               BPL NEWCOL      ;BRANCH ALWAYS
F13F              ;***OFFSET = 0 SO CHANGE DIRECTION TO +
F13F FE 74 A4     OP07   INC IDIR,X      ;DIRECTION <= $00 (+)
F142 10 1C               BPL NEWROW      ;BRANCH ALWAYS
F144              ;*DIRECTION = +
F144 BD 75 A4     OP03   LDA ICOL,X      ;SEE IF LAST COLUMN IN DIGIT
F147 C9 04               CMP #4
F149 F0 05               BEQ OP05        ;COLUMN = 4 SO GO TO NEXT DIGIT
F14B FE 75 A4            INC ICOL,X      ;JUST INCR COLUMN-NOT END OF DIGIT
F14E 10 13               BPL NEWCOL      ;BRANCH ALWAYS
F150              ;**AT COLUMN 4 -- SEE IF LEFT OR RIGHT DIGIT
F150 BD 76 A4     OP05   LDA IOFFST,X
F153 D0 08               BNE OP06        ;OFFSET # 0 SO RIGHT DIGIT
F155 9D 75 A4            STA ICOL,X      ;COLUMN <= 0
F158 FE 76 A4            INC IOFFST,X    ;OFFSET <= 1 (RIGHT CHARACTER)
F15B 10 06               BPL NEWCOL      ;BRANCH ALWAYS
F15D              ;***OFFSET = 1 SO DIRECTION CHANGE
```

```
F15D DE 74 A4    OP06   DEC IDIR,X      ;DIRECTION <= $FF (-)
F160
F160             ;START OF NEW PRINT ROW
F160 1E 7C A4    NEWROW ASL IMASK,X     ;UPDATE ROW MASK FOR DOT PATTERNS
F163             ;START OF NEW PRINT COLUMN
F163 A9 00       NEWCOL LDA #0          ;CLEAR OUTPUT PATTERN
F165 9D 78 A4           STA IOUTL,X     ;PATTERN FOR 8 RIGHT CHRS
F168 9D 79 A4           STA IOUTU,X     ;PATTERN FOR 2 LEFT SOLEN
F16B 9D 7B A4           STA IBITU,X     ;OUTPUT MSK FOR LEFTMOST SOLEN
F16E A9 01              LDA #1
F170 9D 7A A4           STA IBITL,X     ;OUTPUT MSK FOR RIGHTMOST SOLEN
F173             ;GET ADDRESS OF DOT PATTERN TABLE FOR NEXT COLUMN
F173 BD 75 A4           LDA ICOL,X      ;GET COLUMN NUMBER (0-4)
F176 0A                 ASL A           ;*2 ,INDEX INTO TBL OF TBL ADDRS
F177 A8                 TAY
F178 B9 D7 F2           LDA MTBL,Y      ;LSB OF ADDR OF TABLE
F17B 9D 7D A4           STA JUMP,X      ;PTR TO TBL WITH DOT PATTERNS
F17E B9 D8 F2           LDA MTBL+1,Y    ;MSB OF TABLE ADDRESS
F181 9D 7E A4           STA JUMP+1,X
F184 A9 12              LDA #18         ;COMPUTE INDEX INTO PRNTR BUFFER
F186 1D 76 A4           ORA IOFFST,X    ;+1 IF RIGHT CHR
F189 AA                 TAX
F18A 60                 RTS
F18B
F18B             ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
F18B             ;OUTPUT ACC TO TAPE BUFFER SUBROUTINE
F18B             ; & WHEN FULL OUTPUT BUFF TO TAPE.
F18B             ; IF INFLG=OUTFLG= T USE TWO BUFFERS
F18B             ;OTHERWISE USE SAME BUFFER FOR INPUT
F18B             ;AND OUTPUT (MONIT BUFFER)
F18B 20 9E EB    TOBYTE JSR PHXY        ;SAVE X
F18E AE 37 A4           LDX TAPTR2      ;TAPE BUFFER POINTER FOR OUTPUT
F191 20 0F F2           JSR BKCK2       ;STORE IN BUFFER
F194 E8                 INX
F195 8E 37 A4           STX TAPTR2      ;FOR NEXT
F198 E0 50              CPX #80         ;BUFFER FULL?
F19A D0 32              BNE TABY3       ;NO , GO BACK
F19C             ;OUTPUT A BLOCK FROM BUFFER TO TAPE
F19C 20 E7 F1           JSR BKCKSM      ;COMPUT BLOCK CHECKSUM
F19F 20 1D F2           JSR TAOSET      ;SET TAPE FOR OUTPUT
F1A2 A9 23              LDA #'#'        ;CHAR FOR BEGINNING
F1A4 20 4A F2           JSR OUTTAP      ;OF BLOCK
F1A7             ;OUTPUT CHRS FROM ACTIVE BUFFER
F1A7 20 D2 F1    TABY2  JSR CKBUFF      ;LOAD CHR FROM ACTIVE BUFFER
F1AA 20 4A F2           JSR OUTTAP      ; FROM BUFFER
F1AD E8                 INX
F1AE E0 53              CPX #83         ;2 BLOCK CKSUM CHR + 1 EXTRA CHR..
F1B0 D0 F5              BNE TABY2       ;OTHERWISE ERROR
F1B2 AD 00 A8           LDA DRB
F1B5 29 CF              AND #$CF        ;TURN TAPES OFF PB5,PB4
F1B7 8D 00 A8           STA DRB
F1BA 58                 CLI             ;ENABLE INTERRUPT
F1BB A9 00              LDA #0
F1BD 8D 37 A4           STA TAPTR2      ;CLR TAPE BUFF PTR
F1C0 A9 00              LDA #T1I        ;RESET FREE RUNNING TO 1 SHOT
F1C2 8D 0B A8           STA ACR
F1C5 20 9A FF           JSR PAT22       ;ADD 1 TO BLK COUNT & OUTPUT
F1C8 AD 68 01           LDA BLKO        ;PUT BLK CNT IN FIRST LOC (TABUFF)
F1CB 20 8B F1           JSR TOBYTE
F1CE 20 AC EB    TABY3  JSR PLXY
F1D1 60                 RTS
F1D2
```

```
F1D2                ;CHCK ACTIVE BUFFER AND LOAD A CHR
F1D2                ;CARRY=0 IF ONLY 1 BUFFER ,C=1 IF 2 BUFFERS
F1D2 AD 12 A4       CKBUFF LDA INFLG
F1D5 CD 13 A4              CMP OUTFLG
F1D8 D0 08                 BNE CBUFF1
F1DA C9 54                 CMP #'T'        ;SEE IF INFLG=OUTFLG = T
F1DC D0 04                 BNE CBUFF1
F1DE 38                    SEC             ;USE PAGE 1 FOR OUTPUT BUFFER
F1DF B5 AD                 LDA TABUF2,X
F1E1 60                    RTS
F1E2 18             CBUFF1 CLC             ;USE SAME BUFFER FOR I/O
F1E3 BD 16 01              LDA TABUFF,X
F1E6 60                    RTS
F1E7
F1E7                ;COMPUTE BLOCK CHECKSUM & PUT IT
F1E7                ;AT THE END OF ACTIVE BUFFER
F1E7 A9 00          BKCKSM LDA #0          ;CLEAR BLK CKSUM LOCAT
F1E9 8D 66 01              STA TABUFF+80
F1EC 8D 67 01              STA TABUFF+81
F1EF A2 4F                 LDX #79
F1F1 20 D2 F1       BKCK1  JSR CKBUFF      ;GET CHR FROM EITHER BUFFER
F1F4 18                    CLC
F1F5 6D 66 01              ADC TABUFF+80   ;ADD TO CKSUM
F1F8 8D 66 01              STA TABUFF+80
F1FB 90 03                 BCC *+5
F1FD EE 67 01              INC TABUFF+81
F200 CA                    DEX
F201 10 EE                 BPL BKCK1       ;DO THE WHOLE BUFFER
F203 A2 50                 LDX #80
F205 AD 66 01              LDA TABUFF+80   ;PUT CKSUM INTO RIGHT BUFFER
F208 20 0F F2              JSR BKCK2
F20B E8                    INX
F20C AD 67 01              LDA TABUFF+81
F20F 48             BKCK2  PHA             ;OUTPUT A CHAR TO RIGHT BUFFER
F210 20 D2 F1              JSR CKBUFF      ;GET WHICH BUFFER
F213 68                    PLA
F214 B0 04                 BCS BKCK3       ;BRNCH TO SECOND BUFFER
F216 9D 16 01              STA TABUFF,X
F219 60                    RTS
F21A 95 AD          BKCK3  STA TABUF2,X    ;TO PAG 1
F21C 60                    RTS
F21D
F21D                ;SET TAPE (1 OR 2) FOR OUTPUT
F21D 20 C0 F2       TAOSET JSR SETSPD      ;SET UP SPEED (# OF HALF PULSES)
F220 AD 35 A4              LDA TAPOUT      ;OUTPUT FLG (TAPE 1 OR 2)
F223 20 1C EE              JSR TIOSET      ;SET PB4 OR PB5 TO ZERO
F226 A9 EC                 LDA #DATOUT+MOFF ;SET CA2=0 (DATA OUT)
F228 8D 0C A8              STA PCR
F22B A9 C0                 LDA #T1FR       ;SET TIMER IN FREE RUNNING
F22D 8D 0B A8              STA ACR
F230 A9 00                 LDA #00
F232 8D 05 A8              STA T1CH        ;START TIMER T1
F235 AE 09 A4              LDX GAP         ;OUTPUT 4*GAP SYN BYTES
F238 A9 16          TAOS1  LDA #$16        ;SYN CHAR
F23A 20 4A F2              JSR OUTTAP      ;TO TAPE
F23D 20 4A F2              JSR OUTTAP
F240 20 4A F2              JSR OUTTAP
F243 20 4A F2              JSR OUTTAP
F246 CA                    DEX
F247 D0 EF                 BNE TAOS1
F249 60                    RTS
F24A
```

```
F24A                 ;OUTPUT ACC TO TAPE
F24A 8E 2D A4    OUTTAP STX CPIY+3      ;SAVE X
F24D A0 07              LDY #$07        ;FOR THE 8 BITS
F24F 8C 27 A4           STY STIY
F252 AE 08 A4           LDX TSPEED
F255 30 39              BMI OUTTA1      ;IF ONE IS SUPER HIPER
F257 48                 PHA
F258 A0 02       TRY    LDY #2          ;SEND 3 UNITS
F25A 8C 28 A4           STY STIY+1      ;STARTING AT 3700 HZ
F25D BE 0A A4    ZON    LDX NPUL,Y      ;#OF HALF CYCLES
F260 48                 PHA
F261 B9 0B A4    ZON1   LDA TIMG,Y      ;SET UP LACTH FOR NEXT
F264 8D 06 A8           STA T1LL        ;PULSE (80 OR CA) (FREC)
F267 A9 00              LDA #0
F269 8D 07 A8           STA T1LH
F26C 2C 0D A8    ZON2   BIT IFR         ;WAIT FOR PREVIOUS
F26F 50 FB              BVC ZON2        ;CYCLE (T1 INT FLG)
F271 AD 04 A8           LDA T1L         ;CLR INTERR FLG
F274 CA                 DEX
F275 D0 EA              BNE ZON1        ;SEND ALL CYCLES
F277 68                 PLA
F278 CE 28 A4           DEC STIY+1
F27B F0 05              BEQ SETZ        ;BRCH IF LAST ONE
F27D 30 07              BMI ROUT        ;BRCH IF NO MORE
F27F 4A                 LSR A           ;TAKE NEXT BIT
F280 90 DB              BCC ZON         ;...IF IT'S A ONE...
F282 A0 00       SETZ   LDY #0          ;SWITCH TO 2400 HZ
F284 F0 D7              BEQ ZON         ;UNCONDITIONAL BRCH
F286 CE 27 A4    ROUT   DEC STIY        ;ONE LESS BIT
F289 10 CD              BPL TRY         ;ANY MORE? GO BACK
F28B 68          ROUT1  PLA             ;RECOVER CHR
F28C AE 2D A4           LDX CPIY+3      ;RESTORE X
F28F 60                 RTS
F290
F290             ;OUTPUT HALF PULSE FOR 0 (1200 HZ) &
F290             ;TWO HALF PULSES FOR 1 (2400 HZ)  (00 TSPEED)
F290 48          OUTTA1 PHA
F291 8D 28 A4           STA STIY+1      ;STORE ACC
F294 A2 02       OUTTA2 LDX #2          ;# OF HALF PULSES
F296 A9 D0              LDA #$D0        ;1/2 PULSE OF 2400
F298 8D 06 A8           STA T1LL
F29B A9 00              LDA #00
F29D 8D 07 A8           STA T1LH
F2A0 20 BC FF           JSR PATC25      ;WAIT TILL COMPLETED
F2A3 4E 28 A4           LSR STIY+1      ;GET BITS FROM CHR
F2A6 B0 0A              BCS OUTTA3
F2A8 A9 A0              LDA #$A0        ;BIT=0 ,OUTPUT 1200 HZ
F2AA 8D 06 A8           STA T1LL
F2AD A9 01              LDA #$01
F2AF 8D 07 A8           STA T1LH
F2B2 20 BC FF    OUTTA3 JSR PATC25
F2B5 CA                 DEX
F2B6 10 FA              BPL OUTTA3      ;OUTPUT 3 HALF PULSES
F2B8 88                 DEY
F2B9 10 D9              BPL OUTTA2      ;ALL BITS ?
F2BB 4C 8B F2           JMP ROUT1       ;RESTORE REGS
F2BE EA                 NOP
F2BF EA                 NOP
F2C0
F2C0             ;SET SPEED FROM NORMAL TO 3 TIMES NORMAL
F2C0 AD 08 A4    SETSPD LDA TSPEED      ;SPEED FLG
F2C3 6A                 ROR A           ;NORMAL OR 3* NORM
```

```
F2C4 A9 0C              LDA #12
F2C6 90 02              BCC SETSP1
F2C8 A9 04              LDA #4
F2CA 8D 0A A4    SETSP1 STA NPUL
F2CD A9 12              LDA #18
F2CF 90 02              BCC SETSP2
F2D1 A9 06              LDA #6
F2D3 8D 0C A4    SETSP2 STA TIMG+1
F2D6 60                 RTS
F2D7             ;.FILE A3/2
F2D7
F2D7             ; ADDRESS TABLE FOR EACH PRINT COLUMN
F2D7             ; EACH TBL CONTAINS DOT PATTERNS FOR 1 OF THE 5 COLUMNS.
F2D7             ;   DATA ARE STORED WITH EACH BYTE DEFINING ONE COLUMN...
F2D7             ; OF A CHARACTER, WITH THE TOP DOT CORRESPONDING TO THE..
F2D7             ; LSB IN THE BYTE
F2D7 E1F221F361F3MTBL   .DW COL0,COL1,COL2,COL3,COL4
F2DD A1F3E1F3
F2E1
F2E1             ;DOT PATTERNS FOR COLUMN ZERO (LEFTMOST COLUMN)
F2E1 3E7E7F3E7F7FCOL0   .DB $3E,$7E,$7F,$3E,$7F,$7F,$7F,$3E  ;@ -- G
F2E7 7F3E
F2E9 7F00207F7F7F       .DB $7F,$00,$20,$7F,$7F,$7F,$7F,$3E  ;H -- O
F2EF 7F3E
F2F1 7F3E7F46013F       .DB $7F,$3E,$7F,$46,$01,$3F,$07,$7F  ;P -- W
F2F7 077F
F2F9 6307617F0300       .DB $63,$07,$61,$7F,$03,$00,$02,$40  ;X -- (
F2FF 0240
F301 000000142463       .DB $00,$00,$00,$14,$24,$63,$60,$00  ;  -- '
F307 6000
F309 000014084008       .DB $00,$00,$14,$08,$40,$08,$40,$60  ;( -- /
F30F 4060
F311 3E4462411827       .DB $3E,$44,$62,$41,$18,$27,$3C,$01  ;0 -- 7
F317 3C01
F319 364600400814       .DB $36,$46,$00,$40,$08,$14,$41,$02  ;8 -- ?
F31F 4102
F321
F321             ;DOT PATTERNS FOR COLUMN 1
F321 410949414149COL1   .DB $41,$09,$49,$41,$41,$49,$09,$41  ;@ -- G
F327 0941
F329 084140084002       .DB $08,$41,$40,$08,$40,$02,$06,$41  ;H -- O
F32F 0641
F331 094109490140       .DB $09,$41,$09,$49,$01,$40,$18,$20  ;P -- W
F337 1820
F339 140851410400       .DB $14,$08,$51,$41,$04,$00,$01,$40  ;X -- (
F33F 0140
F341 0000077F2A13       .DB $00,$00,$07,$7F,$2A,$13,$4E,$04  ;  -- '
F347 4E04
F349 1C4108083008       .DB $1C,$41,$08,$08,$30,$08,$00,$10  ;( -- /
F34F 0010
F351 514251411445       .DB $51,$42,$51,$41,$14,$45,$4A,$71  ;0 -- 7
F357 4A71
F359 494900341414       .DB $49,$49,$00,$34,$14,$14,$41,$01  ;8 -- ?
F35F 4101
F361
F361             ;DOT PATTERNS FOR COLUMN 2
F361 5D0949414149COL2   .DB $5D,$09,$49,$41,$41,$49,$09,$41  ;@ -- G
F367 0941
F369 087F4114400C       .DB $08,$7F,$41,$14,$40,$0C,$08,$41  ;H -- O
F36F 0841
F371 095119497F40       .DB $09,$51,$19,$49,$7F,$40,$60,$18  ;P -- W
F377 6018
```

```
F379 087849410841       .DB $08,$78,$49,$41,$08,$41,$01,$40  ;X -- (
F37F 0140
F381 004F00147F08       .DB $00,$4F,$00,$14,$7F,$08,$59,$02  ;  -- '
F387 5902
F389 22223E3E0008       .DB $22,$22,$3E,$3E,$00,$08,$00,$08  ;( -- /
F38F 0008
F391 497F51491245       .DB $49,$7F,$51,$49,$12,$45,$49,$09  ;0 -- 7
F397 4909
F399 494944002214       .DB $49,$49,$44,$00,$22,$14,$22,$51  ;8 -- ?
F39F 2251
F3A1
F3A1                ;DOT PATTERNS FOR COLUMN 3
F3A1 550949412249COL3   .DB $55,$09,$49,$41,$22,$49,$09,$49  ;@ -- G
F3A7 0949
F3A9 08413F224002       .DB $08,$41,$3F,$22,$40,$02,$30,$41  ;H -- O
F3AF 3041
F3B1 092129490140       .DB $09,$21,$29,$49,$01,$40,$18,$20  ;P -- W
F3B7 1820
F3B9 140845001041       .DB $14,$08,$45,$00,$10,$41,$01,$40  ;X -- (
F3BF 0140
F3C1 0000077F2A64       .DB $00,$00,$07,$7F,$2A,$64,$26,$01  ;  -- '
F3C7 2601
F3C9 411C08080008       .DB $41,$1C,$08,$08,$00,$08,$00,$04  ;( -- /
F3CF 0004
F3D1 454049557F45       .DB $45,$40,$49,$55,$7F,$45,$49,$05  ;0 -- 7
F3D7 4905
F3D9 492900004114       .DB $49,$29,$00,$00,$41,$14,$14,$09  ;8 -- ?
F3DF 1409
F3E1                ;DOT PATTERNS FOR COLUMN 4
F3E1 1E7E36221C41COL4   .DB $1E,$7E,$36,$22,$1C,$41,$01,$7A  ;@ -- G
F3E7 017A
F3E9 7F000141407F       .DB $7F,$00,$01,$41,$40,$7F,$7F,$3E  ;H -- O
F3EF 7F3E
F3F1 065E4631013F       .DB $06,$5E,$46,$31,$01,$3F,$07,$7F  ;P -- W
F3F7 077F
F3F9 63074300607F       .DB $63,$07,$43,$00,$60,$7F,$02,$40  ;X -- (
F3FF 0240
F401 000000141263       .DB $00,$00,$00,$14,$12,$63,$50,$00  ;  -- '
F407 5000
F409 000014080008       .DB $00,$00,$14,$08,$00,$08,$00,$03  ;( -- /
F40F 0003
F411 3E4046221039       .DB $3E,$40,$46,$22,$10,$39,$31,$03  ;0 -- 7
F417 3103
F419 361E00004114       .DB $36,$1E,$00,$00,$41,$14,$08,$06  ;8 -- ?
F41F 0806
F421
F421                ;ASCII CHARACTERS FOR KB
F421 2008000D0000ROW1   .DB $20,$08,$00,$0D,$00,$00,$00,$00
F427 0000
F429 00605C000000ROW2   .DB $00,$60,'\',$00,$00,$00,$7F,$00
F42F 7F00
F431 2E4C502D3A30ROW3   .DB ".LP-:0;/"
F437 3B2F
F439 4D4A494F3938ROW4   .DB "MJIO98K,"
F43F 4B2C
F441 424759553736ROW5   .DB "BGYU76HN"
F447 484E
F449 434452543534ROW6   .DB "CDRT54FV"
F44F 4656
F451 5A4157453332ROW7   .DB "ZAWE32SX"
F457 5358
F459 00001B51315EROW8   .DB $00,$00,$1B,"Q1",$5E,"]["
```

```
F45F 5D5B
F461
F461               ;DISASSEMBLE INSTRUCTIONS AND SHOW REGS IS REGF SET
F461 AD 0E A4      REGQ   LDA REGF         ;GET FLAG
F464 F0 06                BEQ DISASM
F466 20 32 E2             JSR REG1         ;SHOW THE SIX REGS
F469 20 24 EA             JSR CRCK         ;<CR>
F46C
F46C 20 45 F5      DISASM JSR PRBL2
F46F 20 3C F5             JSR PRPC         ;OUTPUT PROG COUNTR
F472 A0 00                LDY #0
F474 20 56 EB             JSR PCLLD
F477 A8                   TAY
F478 4A                   LSR A
F479 90 0B                BCC IEVEN
F47B 4A                   LSR A
F47C B0 17                BCS ERR
F47E C9 22                CMP #$22
F480 F0 13                BEQ ERR
F482 29 07                AND #7
F484 09 80                ORA #$80
F486 4A            IEVEN  LSR A
F487 AA                   TAX
F488 BD 5B F5             LDA MODE,X
F48B B0 04                BCS RTMODE
F48D 4A                   LSR A
F48E 4A                   LSR A
F48F 4A                   LSR A
F490 4A                   LSR A
F491 29 0F         RTMODE AND #$F
F493 D0 04                BNE GETFMT
F495 A0 80         ERR    LDY #$80
F497 A9 00                LDA #0
F499 AA            GETFMT TAX
F49A BD 9F F5             LDA MODE2,X
F49D 8D 16 01             STA FORMA
F4A0 29 03                AND #3
F4A2 85 EA                STA LENGTH
F4A4 98                   TYA              ;OPCODE
F4A5 29 8F                AND #$8F
F4A7 AA                   TAX
F4A8 98                   TYA              ;OPCODE IN A AGAIN
F4A9 A0 03                LDY #3
F4AB E0 8A                CPX #$8A
F4AD F0 0B                BEQ MNNDX3
F4AF 4A            MNNDX1 LSR A
F4B0 90 08                BCC MNNDX3
F4B2 4A                   LSR A
F4B3 4A            MNNDX2 LSR A
F4B4 09 20                ORA #$20
F4B6 88                   DEY
F4B7 D0 FA                BNE MNNDX2
F4B9 C8                   INY
F4BA 88            MNNDX3 DEY
F4BB D0 F2                BNE MNNDX1
F4BD 48                   PHA              ;SAVE MNEMONIC TABLE INDEX
F4BE 20 56 EB             JSR PCLLD
F4C1 20 46 EA             JSR NUMA
F4C4 20 45 F5             JSR PRBL2        ;PRINT LAST BLANK
F4C7 68                   PLA
F4C8 A8                   TAY
F4C9 B9 B9 F5             LDA MNEML,Y
```

```
F4CC 8D 17 01           STA LMNEM
F4CF B9 F9 F5           LDA MNEMR,Y
F4D2 8D 18 01           STA RMNEM
F4D5 A2 03              LDX #3          ;MUST BE
F4D7 A9 00       PRMN1  LDA #0
F4D9 A0 05              LDY #5
F4DB 0E 18 01    PRMN2  ASL RMNEM
F4DE 2E 17 01           ROL LMNEM
F4E1 2A                 ROL A
F4E2 88                 DEY
F4E3 D0 F6              BNE PRMN2
F4E5 69 BF              ADC #'?'+$80    ;ADD "?" OFFSET
F4E7 20 BC E9           JSR OUTALL
F4EA CA                 DEX
F4EB D0 EA              BNE PRMN1
F4ED 20 45 F5           JSR PRBL2
F4F0 A2 06              LDX #6
F4F2 A9 00              LDA #0
F4F4 8D 29 A4           STA STIY+2      ;FLAG
F4F7 E0 03       PRADR1 CPX #3
F4F9 D0 1E              BNE PRADR3      ;IF X=3 PRINT ADDR VALUE
F4FB A4 EA              LDY LENGTH
F4FD F0 1A              BEQ PRADR3      ;1 BYTE INSTR
F4FF AD 16 01    PRADR2 LDA FORMA
F502 C9 E8              CMP #$E8        ;RELATIVE ADDRESSING
F504 20 56 EB           JSR PCLLD
F507 B0 27              BCS RELADR
F509             ;SE IF SYMBOL
F509 48                 PHA
F50A AD 29 A4           LDA STIY+2
F50D D0 03              BNE MR11A
F50F EE 29 A4           INC STIY+2      ;SHOW WE WERE HERE
F512
F512 68          MR11A  PLA
F513 20 46 EA           JSR NUMA
F516 88                 DEY
F517 D0 E6              BNE PRADR2
F519 0E 16 01    PRADR3 ASL FORMA
F51C 90 0E              BCC PRADR4
F51E BD AC F5           LDA CHAR1-1,X
F521 20 BC E9           JSR OUTALL
F524 BD B2 F5           LDA CHAR2-1,X
F527 F0 03              BEQ PRADR4
F529 20 BC E9           JSR OUTALL
F52C CA          PRADR4 DEX
F52D D0 C8              BNE PRADR1
F52F 60                 RTS
F530 20 4D F5    RELADR JSR PCADJ3
F533 AA                 TAX
F534 E8                 INX
F535 D0 01              BNE PRNTXY
F537 C8                 INY
F538 98          PRNTXY TYA
F539 4C 42 EA           JMP WRAX        ;PRINT A &X
F53C AD 26 A4    PRPC   LDA SAVPC+1     ;PRINT PC
F53F AE 25 A4           LDX SAVPC
F542 20 42 EA           JSR WRAX
F545 A9 20       PRBL2  LDA #' '
F547 4C BC E9           JMP OUTALL
F54A A5 EA              LDA LENGTH
F54C 38                 SEC
F54D AC 26 A4    PCADJ3 LDY SAVPC+1     ;PRG CNTR HIGH
```

```
F550 AA                 TAX
F551 10 01              BPL PCADJ4
F553 88                 DEY
F554 6D 25 A4    PCADJ4 ADC SAVPC       ;PROG CNTR LOW
F557 90 01              BCC RTS1
F559 C8                 INY
F55A 60          RTS1   RTS
F55B
F55B 40024503D008MODE   .DB $40,2,$45,3,$D0,8,$40,9
F561 4009
F563 30224533D008       .DB $30,$22,$45,$33,$D0,8,$40,9
F569 4009
F56B 40024533D008       .DB $40,2,$45,$33,$D0,8,$40,9
F571 4009
F573 400245B3D008       .DB $40,2,$45,$B3,$D0,8,$40,9
F579 4009
F57B 00224433D08C       .DB 0,$22,$44,$33,$D0,$8C,$44,0
F581 4400
F583 11224433D08C       .DB $11,$22,$44,$33,$D0,$8C,$44,$9A
F589 449A
F58B 10 22 44 33        .DB $10,$22,$44,$33
F58F D0 08 40 09        .DB $D0,8,$40,9
F593 10224433D008       .DB $10,$22,$44,$33,$D0,8,$40,9
F599 4009
F59B 62 13 78 A9        .DB $62,$13,$78,$A9
F59F
F59F 002101020080MODE2  .DB 0,$21,1,2,0,$80,$59,$4D
F5A5 594D
F5A7 1112064A051D       .DB $11,$12,6,$4A,5,$1D
F5AD
F5AD 2C292C23282ECHAR1  .DB ",",$29,",#(",","."
F5B3 590058000041CHAR2  .DB "Y",0,"X",0,0,"A"
F5B9
F5B9 1C8A1C235D8BMNEML  .DB $1C,$8A,$1C,$23,$5D,$8B,$1B
F5BF 1B
F5C0 A1                 .DB $A1
F5C1 9D8A1D239D8B       .DB $9D,$8A,$1D,$23,$9D,$8B,$1D,$A1
F5C7 1DA1
F5C9 002919AE69A8       .DB 0,$29,$19,$AE,$69,$A8,$19,$23
F5CF 1923
F5D1 24531B232453       .DB $24,$53,$1B,$23,$24,$53,$19,$A1
F5D7 19A1
F5D9 001A5B5BA569       .DB 0,$1A,$5B,$5B,$A5,$69,$24,$24
F5DF 2424
F5E1 AEAEA8AD2900       .DB $AE,$AE,$A8,$AD,$29,0,$7C,0
F5E7 7C00
F5E9 159C6D9CA569       .DB $15,$9C,$6D,$9C,$A5,$69,$29,$53
F5EF 2953
F5F1 84133411A569       .DB $84,$13,$34,$11,$A5,$69,$23,$A0
F5F7 23A0
F5F9
F5F9 D8625A482662MNEMR  .DB $D8,$62,$5A,$48,$26,$62,$94
F5FF 94
F600 88                 .DB $88
F601 5444C8546844       .DB $54,$44,$C8,$54,$68,$44,$E8,$94
F607 E894
F609 00B4088474B4       .DB 0,$B4,8,$84,$74,$B4,$28,$6E
F60F 286E
F611 74F4CC4A72F2       .DB $74,$F4,$CC,$4A,$72,$F2,$A4,$8A
F617 A48A
F619 00AAA2A27474       .DB 0,$AA,$A2,$A2,$74,$74,$74,$72
F61F 7472
```

```
F621 4468B232B200       .DB $44,$68,$B2,$32,$B2,0,$22,0
F627 2200
F629 1A1A26267272       .DB $1A,$1A,$26,$26,$72,$72,$88,$C8
F62F 88C8
F631 C4CA26484444       .DB $C4,$CA,$26,$48,$44,$44,$A2,$C8
F637 A2C8
F639
F639              ;*******************************
F639              ;***    AIM TEXT EDITOR      ***
F639              ;***       05/01/78          ***
F639              ;*******************************
F639
F639              ; R=READ FROM ANY INPUT DEVICE
F639              ; I=INSERT A LINE FROM INPUT DEV
F639              ; K=DELETE A LINE
F639              ; U-GO UP ONE LINE
F639              ; D=GO DOWN ONE LINE
F639              ; L=LIST LINES TO OUTPUT DEV
F639              ; T=GO TO TOP OF TEXT
F639              ; B=GO TO BOTTOM OF TEXT
F639              ; F=FIND STRING
F639              ; C=CHANGE STRING TO NEW STRING
F639              ; Q=QUIT EDITOR
F639              ; <SPACE>=DISPLAY CURRENT LINE
F639
F639              ;***** E COMMAND-EDITOR ENTRY (FROM MONITOR) *****
F639 20 13 EA     EDIT   JSR CRLOW
F63C A0 6C               LDY #EMSG1-M1
F63E 20 AF E7            JSR KEP          ;START UP MSG
F641 20 13 EA            JSR CRLOW
F644 20 A3 E7     EDI0   JSR FROM
F647 B0 FB               BCS EDI0
F649 AD 1E A4            LDA CKSUM        ;IS CLR IF ADDR WAS INPUTTED
F64C F0 03               BEQ *+5
F64E 20 DB E2            JSR WRITAZ       ;OUTPUT DEFAULT ADDR (0200)
F651 A2 01               LDX #1
F653 BD 1C A4     EDI1   LDA ADDR,X
F656 95 E3               STA TEXT,X
F658 95 E1               STA BOTLN,X
F65A 9D 1A A4            STA S1,X         ;FOR MEMORY TEST
F65D CA                  DEX
F65E 10 F3               BPL EDI1
F660 20 3B E8            JSR BLANK2
F663 20 A7 E7     EDI2   JSR TO           ;END
F666 B0 FB               BCS EDI2
F668 20 BC F8            JSR TOPNO        ;TRANSF TEXT TO ADDR FOR RAM CHECK
F66B AD 1E A4            LDA CKSUM        ;IS CLR IF ADDR WAS INPUTTED
F66E F0 10               BEQ EDI4         ;BRNCH IF NOT DEFAULT VALUE
F670 20 34 F9            JSR SAVNOW
F673 20 B6 F6     EDI3   JSR EDI          ;CARRY IS SET IF NO RAM THERE
F676 90 FB               BCC EDI3
F678 A9 00               LDA #0           ;SET UPPER LIMIT TO BEGINNING...
F67A 8D 1C A4            STA ADDR         ;OF PAGE
F67D 20 DB E2            JSR WRITAZ       ;OUTPUT DEFAULT VALUE ,UPPER LIMIT
F680 AD 1C A4     EDI4   LDA ADDR
F683 85 E5               STA END
F685 AD 1D A4            LDA ADDR+1
F688 85 E6               STA END+1
F68A 20 34 F9            JSR SAVNOW
F68D              ;NOW SEE IF MEMORY IS THERE
F68D 20 B6 F6     EDI5   JSR EDI
F690 90 FB               BCC EDI5
```

```
F692 A5 E6               LDA END+1        ;CMP WITH END
F694 CD 1D A4            CMP ADDR+1
F697 F0 11               BEQ EDI7
F699 B0 13               BCS EDI8
F69B 20 BC F8    EDI6    JSR TOPNO        ;RESTORE NOWLN
F69E A9 00               LDA #0
F6A0 91 DF               STA (NOWLN),Y    ;END OF TEXT MARKER
F6A2 20 13 EA            JSR CRLOW
F6A5 A9 52               LDA #'R'         ;FORCE READ COMMAND
F6A7 4C 8D FA            JMP ENTRY
F6AA A5 E5       EDI7    LDA END          ;IF ZERO MEM IS OKAY
F6AC F0 ED               BEQ EDI6
F6AE A9 00       EDI8    LDA #0
F6B0 8D 1C A4            STA ADDR
F6B3 4C 33 EB            JMP MEMERR       ;NO MEMORY FOR THOSE LIMITS
F6B6
F6B6 A0 00       EDI     LDY #0           ;CHCK IF MEMORY WRITES
F6B8 20 B7 FE            JSR PATCH6       ;GET BYTE ADDR BY ADDR,ADDR+1
F6BB 48                  PHA              ;SAVE IT
F6BC A9 AA               LDA #$AA         ;SET THIS PATTERN
F6BE 20 78 EB            JSR SADDR        ;CHCK IT
F6C1 D0 09               BNE EDI2B
F6C3 68                  PLA
F6C4 20 78 EB            JSR SADDR        ;RESTORE CHR
F6C7 EE 1D A4            INC ADDR+1       ;NEXT PAG
F6CA 18                  CLC              ;IT WROTE
F6CB 60                  RTS
F6CC 38          EDI2B   SEC              ;DIDNT WRITE
F6CD 68                  PLA
F6CE 60                  RTS
F6CF
F6CF             ;***** T COMMAND-REENTRY EDITOR *****
F6CF             ;RE-ENTRY POINT,TEXT ALREADY THERE
F6CF 20 24 EA    REENTR JSR CRCK          ;<CR> IF PRI ON
F6D2 20 BC F8    TP      JSR TOPNO        ;GO TO TOP
F6D5 4C B9 F7            JMP IN03A        ;DISPLAY LINE
F6D8
F6D8             ;***** U COMMAND-UP LINE *****
F6D8             ;GO UP ONE LINE BUT...
F6D8             ;DOWN IN ADDRESSING MEMORY
F6D8 20 DB F8    DNNO    JSR ATTOP        ;THIS RTN DOESNT PRINT
F6DB 90 06               BCC DOW1         ;NOT TOP
F6DD 20 27 F7            JSR PLNE         ;ARE AT TOP
F6E0 4C 78 FA            JMP ERR0
F6E3 A0 00       DOW1    LDY #0
F6E5 20 1D F9            JSR SUB          ;DECREMENT NOWLN PAST <CR>
F6E8 20 1D F9    DOW2    JSR SUB
F6EB 20 DB F8            JSR ATTOP
F6EE B0 30               BCS UP4
F6F0 B1 DF               LDA (NOWLN),Y
F6F2 C9 0D               CMP #CR
F6F4 D0 F2               BNE DOW2
F6F6 4C 28 F9            JMP AD1
F6F9
F6F9             ;***** D COMMAND-DOWN LINE *****
F6F9             ;GO DOWN ONE LINE BUT...
F6F9             ;UP IN ADDRESSING MEMORY
F6F9 20 09 F7    UP      JSR UPNO
F6FC 20 27 F7            JSR PLNE         ;DISPLAY LINE & CHCK BOTTOM
F6FF 20 E9 F8            JSR ATBOT
F702 90 1C               BCC UP4
F704 A0 72               LDY #EMSG2-M1    ;PRINT "END"
```

```
F706 4C AF E7           JMP KEP
F709 A0 00      UPNO    LDY #0
F70B 20 E9 F8           JSR ATBOT
F70E 90 03              BCC UP1
F710 4C 5C FA           JMP ENDERR
F713 B1 DF      UP1     LDA (NOWLN),Y
F715 F0 09              BEQ UP4
F717 C8                 INY
F718 C9 0D              CMP #CR
F71A D0 F7              BNE UP1
F71C 98                 TYA
F71D 20 2A F9           JSR ADDA        ;ADD LENGTH TO CURRENT LINE
F720 60         UP4     RTS
F721
F721            ;***** B COMMAND-GO TO BOTTOM *****
F721 20 C5 F8   BT      JSR SETBOT
F724            ;START U-COMMAND HERE
F724 20 D8 F6   DOWN    JSR DNNO        ;U COMMAND
F727
F727            ;***** <SPACE> COMMAND-DISPLAY CURRENT LINE *****
F727 A0 00      PLNE    LDY #0          ;PRINT CURRENT LINE
F729 B1 DF      P02     LDA (NOWLN),Y
F72B F0 0E              BEQ P01         ;PAST END ?
F72D C9 0D              CMP #CR         ;DONE?
F72F F0 0A              BEQ P01
F731 20 BC E9           JSR OUTALL      ;PUT IT SOMEWHERE
F734 99 38 A4           STA DIBUFF,Y
F737 C8                 INY
F738 4C 29 F7           JMP P02
F73B 84 EA      P01     STY LENGTH
F73D 84 E9              STY OLDLEN
F73F AC 13 A4   P03     LDY OUTFLG      ;ONE MORE <CR> FOR TAPE
F742 C0 0D              CPY #CR
F744 F0 03              BEQ P00
F746 4C F0 E9           JMP CRLF        ;TO OUTPUT DEV
F749 4C 24 EA   P00     JMP CRCK        ;<CR>, & DONT CLR DISPL
F74C
F74C            ;***** K COMMAND-KILL LINE *****
F74C            ;DELETE CURRENT LINE
F74C 20 B6 F8   DLNE    JSR KIFLG       ;CLR K OR I COMM FLG
F74F EA                 NOP
F750 EA                 NOP
F751 EA                 NOP
F752 20 27 F7           JSR PLNE
F755 20 E9 F8           JSR ATBOT
F758 B0 CD              BCS PLNE        ;AT END OF TEXT
F75A A0 00              LDY #0
F75C 84 EA              STY LENGTH
F75E 20 3F F9           JSR REPLAC      ;KILL LINE
F761 4C 27 F7           JMP PLNE
F764
F764            ;***** I COMMAND-INSERT LINE *****
F764 20 6D F7   IN      JSR INL
F767 20 F9 F6           JSR UP          ;DISPLAY NEXT LINE DOWN
F76A 4C 78 FA           JMP ERR0        ;IF AT BOTTOM PRINT "END"
F76D 20 B6 F8   INL     JSR KIFLG       ;CLR K OR I COMM FLG
F770 A0 00              LDY #0          ;GET LINE INTO DIBUFF
F772 84 E9              STY OLDLEN
F774 20 BD E7           JSR PROMPT
F777 20 44 EB           JSR CLR
F77A 20 93 E9   IN02    JSR INALL
F77D 20 F8 FE           JSR PATC12      ;CLR, SO WE CAN OUTPUT TO PRI
```

```
F780 C9 7F               CMP #$7F        ;RUB
F782 4C 2A FF            JMP PATC17      ;NO ZEROS IN CASE OF PAPER TAPE
F785 C9 0A      IN02A    CMP #LF
F787 F0 F1               BEQ IN02
F789 C9 0D               CMP #CR
F78B F0 1B               BEQ IN03
F78D C0 3C               CPY #60         ;DO NOT INCR Y IF 60
F78F B0 08               BCS IN03B
F791 99 38 A4            STA DIBUFF,Y
F794 C8                  INY
F795 C0 3C               CPY #60
F797 D0 E1               BNE IN02        ;CONTIN , DISP WONT ALLOW > 60 CHR`
F799 A0 3C      IN03B    LDY #60         ;SET Y TO MAX OF 60
F79B A9 01               LDA #$01
F79D 0D 11 A4            ORA PRIFLG      ;DO NOT OUTPUT TO PRI ANY MORE
F7A0 8D 11 A4            STA PRIFLG      ;OTHERWISE CLOBBERS THE BUFFER
F7A3 8C 15 A4            STY CURPO2
F7A6 D0 D2               BNE IN02        ;GO BACK
F7A8 84 EA      IN03     STY LENGTH
F7AA C0 00               CPY #0          ;FIRST CHAR?
F7AC D0 17               BNE IN05
F7AE AD 19 A4            LDA COUNT       ;K OR I COMM FLG ?
F7B1 D0 12               BNE IN05        ;BRANCH IF C COMMAND
F7B3 20 24 EA            JSR CRCK        ;<CR> IF PRI PNTR DIFF FROM 0
F7B6 20 03 FF            JSR PATC13      ;TURN ON TAPES & SET DEFAULT DEV
F7B9 20 27 F7   IN03A    JSR PLNE        ;DISPLAY NEXT LINE DOWN
F7BC 20 09 F7            JSR UPNO        ;PRINT "END" IF BOTTOM
F7BF 20 D8 F6            JSR DNNO
F7C2 4C 78 FA            JMP ERR0
F7C5 20 3F F9   IN05     JSR REPLAC      ;INSERT THE LINE
F7C8 4C 24 EA            JMP CRCK        ;<CR> IF PRI PTR NOT 0
F7CB
F7CB            ;***** R COMMAND-READ LINE *****
F7CB            ;READ TEXT FROM ANY INPUT DEVICE UNTIL
F7CB            ;TWO CONSECUTIVE <CR> ARE ENCOUNTER.
F7CB 20 48 E8   INPU     JSR WHEREI
F7CE AC 12 A4            LDY INFLG       ;IF TAPE DO NOT ERRASE BUFFER
F7D1 C0 54               CPY #'T'
F7D3 F0 03               BEQ INPU1
F7D5 20 13 EA            JSR CRLOW
F7D8 20 6D F7   INPU1    JSR INL
F7DB 20 09 F7            JSR UPNO        ;NEXT LINE
F7DE 4C D8 F7            JMP INPU1
F7E1
F7E1            ;***** L COMMAND-LIST LINES *****
F7E1            ;PRINT FROM HERE N LINES TO ACTIVE OUTPUT DEV
F7E1 20 37 E8   LST      JSR PSL1        ;PRINT "/"
F7E4 20 85 E7            JSR GCNT        ;GET LINES COUNT
F7E7 20 13 EA            JSR CRLOW
F7EA 20 71 E8            JSR WHEREO      ;WHERE TO
F7ED 4C F8 F7            JMP LST02       ;ONE MORE LINE
F7F0 20 07 E9   LST01    JSR RCHEK
F7F3 20 90 E7            JSR DONE
F7F6 F0 0B               BEQ LST3
F7F8 20 27 F7   LST02    JSR PLNE
F7FB 20 09 F7            JSR UPNO        ;NEXT LINE
F7FE 20 E9 F8            JSR ATBOT
F801 90 ED               BCC LST01       ;NO
F803 20 3F F7   LST3     JSR P03         ;ONE MORE CRLF FOR TAPE
F806 20 0D FF            JSR PATC14      ;CLOSE TAPE IF NEEDED
F809 4C 5C FA            JMP ENDERR
F80C
```

```
F80C               ;***** F COMMAND-FIND STRING *****
F80C               ;FIND STRING AND PRINT LINE TO TERMINAL
F80C 20 1E F8      FCHAR  JSR FCH
F80F AD 15 A4      FCHA1  LDA CURPO2      ;SAVE BUFFER PNTR
F812 48                   PHA
F813 20 44 EB             JSR CLR         ;CLEAR DISP PNTR
F816 20 27 F7             JSR PLNE
F819 68                   PLA
F81A 8D 15 A4             STA CURPO2
F81D 60                   RTS
F81E               ;FIND A CHARACTER STRING
F81E A0 00         FCH    LDY #0
F820 20 BD E7             JSR PROMPT
F823 20 5F E9      FC1    JSR RDRUP       ;GET THE CHARACTER
F826 C9 0D                CMP #CR         ;REUSE OLD ARGUMENT??
F828 D0 0A                BNE FC3
F82A C0 00                CPY #0          ;FIRST CHAR?
F82C D0 06                BNE FC3
F82E 20 09 F7      FC2    JSR UPNO        ;NEXT LINE DOWN
F831 4C 49 F8             JMP FC5
F834 C9 0D         FC3    CMP #CR         ;DONE
F836 F0 0B                BEQ FC4
F838 99 EB 00             STA STRING,Y
F83B C8                   INY
F83C C0 14                CPY #20         ;MAX LENGTH
F83E D0 E3                BNE FC1
F840 4C 72 FA             JMP ERROR
F843 20 24 EA      FC4    JSR CRCK        ;CLEAR DISPLAY
F846 8C 29 A4             STY STIY+2      ;COUNT OF CHARACTERS
F849 A0 00         FC5    LDY #0
F84B 8C 15 A4             STY CURPO2      ;START AT BEGINNING OF LINENTR IS
F84E AC 15 A4      FC6    LDY CURPO2      ;CLOBBER
F851 A2 00                LDX #0
F853 B1 DF         FC7    LDA (NOWLN),Y   ;GET THE CHARACTER
F855 D0 03                BNE FC8         ;NOT AT END
F857 4C 5C FA             JMP ENDERR
F85A C9 0D         FC8    CMP #CR         ;END OF LINE
F85C F0 D0                BEQ FC2
F85E D5 EB                CMP STRING,X
F860 F0 06                BEQ FC9
F862 EE 15 A4             INC CURPO2
F865 4C 4E F8             JMP FC6
F868 C8            FC9    INY
F869 E8                   INX
F86A EC 29 A4             CPX STIY+2      ;DONE?
F86D D0 E4                BNE FC7
F86F 60                   RTS
F870
F870               ;***** Q COMMAND-EXIT EDITOR *****
F870               ; EXIT THE TEXT EDITOR NEATLY
F870 20 13 EA      STOP   JSR CRLOW
F873 4C A1 E1             JMP COMIN
F876
F876               ;***** C COMMAND-CHANGE STRING *****
F876               ;CHANGE STRING TO ANOTHER STRING IN A LINE
F876 20 B2 F8      CHNG   JSR CFLG        ;SET C COMMAND FLG
F879 20 0C F8             JSR FCHAR       ;FIND CORRECT LINE
F87C 20 3C E9      CHN1   JSR READ        ;IS <CR> IF OK
F87F C9 0D                CMP #CR
F881 F0 09                BEQ CHN2
F883 20 2E F8             JSR FC2         ;TRY NEXT ONE
F886 20 0F F8             JSR FCHA1       ; SHOW LINE
```

```
F889 4C 7C F8          JMP CHN1
F88C AD 29 A4   CHN2   LDA STIY+2      ;GET CHAR COUNT
F88F 85 E9             STA OLDLEN      ;GET READY FOR REPLAC
F891 AD 15 A4          LDA CURPO2      ;PNTR TO BEGINNING OF STRING
F894 48                PHA             ;SAVE IT
F895 20 2A F9          JSR ADDA        ;ADD TO NOWLN (LINE PNTR)
F898 20 44 EB          JSR CLR         ;CLEAR DISP
F89B A0 05             LDY #M3-M1      ;PRINT "TO"
F89D 20 AF E7          JSR KEP
F8A0 A0 00             LDY #0
F8A2 20 7A F7          JSR IN02        ;GET NEW STRING & REPLAC
F8A5 68                PLA
F8A6 AA                TAX
F8A7 F0 06             BEQ CHN4
F8A9 20 1D F9   CHN3   JSR SUB         ;RESTORE NOWLN WHERE IT WAS
F8AC CA                DEX
F8AD D0 FA             BNE CHN3
F8AF 4C 27 F7   CHN4   JMP PLNE        ;DISPLAY THE CHANGED LINE
F8B2
F8B2            ;THE FOLLOWING ARE SUBROUTINES USED BY COMMANDS
F8B2 A9 01      CFLG   LDA #1          ;SET FLG FOR C COMMAND
F8B4 D0 02             BNE KI2
F8B6 A9 00      KIFLG  LDA #0          ;CLR K OR I COMMAND FLG
F8B8 8D 19 A4   KI2    STA COUNT
F8BB 60                RTS
F8BC
F8BC A5 E3      TOPNO  LDA TEXT        ;SET CURRENT LINE TO TOP
F8BE A6 E4             LDX TEXT+1
F8C0 85 DF      TPO1   STA NOWLN
F8C2 86 E0             STX NOWLN+1
F8C4 60                RTS
F8C5
F8C5 A5 E1      SETBOT LDA BOTLN       ;SET CURRENT LINE TO BOTTOM
F8C7 A6 E2             LDX BOTLN+1
F8C9 85 E7             STA SAVE
F8CB 86 E8             STX SAVE+1
F8CD 4C C0 F8          JMP TPO1
F8D0
F8D0 AD 1C A4   RESNOW LDA ADDR        ;RESTORE CURRENT LINE ADDRESS
F8D3 85 DF             STA NOWLN
F8D5 AD 1D A4          LDA ADDR+1
F8D8 85 E0             STA NOWLN+1
F8DA 60                RTS
F8DB
F8DB            ; SEE IF CURRENT LINE AT TOP (C SET IF SO)
F8DB A5 DF      ATTOP  LDA NOWLN
F8DD C5 E3             CMP TEXT
F8DF D0 16             BNE AT01
F8E1 A5 E0             LDA NOWLN+1
F8E3 C5 E4             CMP TEXT+1
F8E5 D0 10             BNE AT01
F8E7 38                SEC
F8E8 60                RTS
F8E9
F8E9            ; SEE IF CURRENT LINE AT BOTTOM (C SET IF SO)
F8E9 A5 DF      ATBOT  LDA NOWLN
F8EB A6 E0             LDX NOWLN+1
F8ED C5 E1             CMP BOTLN
F8EF D0 06             BNE AT01
F8F1 E4 E2             CPX BOTLN+1
F8F3 D0 02             BNE AT01
F8F5 38         AT02   SEC
```

```
F8F6 60                   RTS
F8F7 18            AT01   CLC
F8F8 60                   RTS
F8F9
F8F9               ;SEE IF WE RAN PAST END OF BUFFER LIMIT
F8F9 A5 E1         ATEND  LDA BOTLN
F8FB A6 E2                LDX BOTLN+1
F8FD E4 E6                CPX END+1       ;HIGH BYTE > OR = ?
F8FF 90 F6                BCC AT01
F901 D0 F2                BNE AT02
F903 C5 E5                CMP END         ;LOW BYTE > OR = ?
F905 90 F0                BCC AT01
F907 B0 EC                BCS AT02
F909
F909               ; SAVE CURRENT LINE (NEWLN) IN S1
F909 A5 DF         NOWS1  LDA NOWLN
F90B A6 E0                LDX NOWLN+1
F90D 4C 16 F9             JMP ADDS1A
F910
F910               ; MOVE ADDR INTO S1
F910 AD 1C A4      ADDRS1 LDA ADDR
F913 AE 1D A4             LDX ADDR+1
F916 8D 1A A4      ADDS1A STA S1
F919 8E 1B A4             STX S1+1
F91C 60                   RTS
F91D
F91D               ; SUBTRACT ONE FROM CURRENT LINE (NOWLN)
F91D C6 DF         SUB    DEC NOWLN
F91F A5 DF                LDA NOWLN
F921 C9 FF                CMP #$FF
F923 D0 02                BNE SUB1
F925 C6 E0                DEC NOWLN+1
F927 60            SUB1   RTS
F928
F928               ; ADD ACC TO CURRENT LINE (NOWLN)
F928 A9 01         AD1    LDA #1
F92A 18            ADDA   CLC
F92B 65 DF                ADC NOWLN
F92D 85 DF                STA NOWLN
F92F 90 02                BCC ADDA1
F931 E6 E0                INC NOWLN+1
F933 60            ADDA1  RTS
F934
F934 A5 DF         SAVNOW LDA NOWLN       ;SAVE CURRENT LINE INTO ADDR
F936 8D 1C A4             STA ADDR
F939 A5 E0                LDA NOWLN+1
F93B 8D 1D A4             STA ADDR+1
F93E 60            REP2   RTS
F93F
F93F               ;MOVE CURRENT TEXT AROUND TO HAVE
F93F               ;SPACE TO PUT IN THE NEW BUFFER
F93F A4 EA         REPLAC LDY LENGTH
F941 C4 E9                CPY OLDLEN      ;COMPARE OLD AND NEW LENGTHS
F943 D0 1A                BNE R2W         ;BRANCH IF DIFF
F945 F0 07                BEQ R87         ;LENGTHS ARE EQUAL. JUST REPLACE
F947 A9 0D         R8     LDA #CR
F949 91 DF                STA (NOWLN),Y
F94B 20 4A FA             JSR GOGO
F94E
F94E               ;LENGTH = OLDLEN
F94E 88            R87    DEY
F94F C0 FF                CPY #$FF
```

```
F951 F0 EB              BEQ REP2
F953 B9 38 A4    R88    LDA DIBUFF,Y
F956 91 DF              STA (NOWLN),Y
F958 20 4A FA           JSR GOGO
F95B 88                 DEY
F95C 10 F5              BPL R88
F95E 60                 RTS
F95F B0 6E       R2W    BCS R100        ;LENGTH > OLDLEN
F961
F961             ;LENGTH < OLDLEN
F961 20 34 F9           JSR SAVNOW      ;PUT NOWLN INTO ADDR
F964 20 10 F9           JSR ADDRS1      ;PUT IT IN S1 ALSO
F967 A5 E9              LDA OLDLEN
F969 38                 SEC
F96A E5 EA              SBC LENGTH      ;GET DIFFERENCE IN LENGTHS
F96C A4 EA              LDY LENGTH
F96E D0 07              BNE RQP
F970 AE 19 A4           LDX COUNT       ;C-COMM ?
F973 D0 02              BNE RQP         ;YES, JUMP
F975 69 00              ADC #0          ;INCLUDE <CR>
F977 48          RQP    PHA
F978 18                 CLC
F979 6D 1A A4           ADC S1
F97C 8D 1A A4           STA S1
F97F 90 03              BCC R6
F981 EE 1B A4           INC S1+1
F984 A9 1A       R6     LDA #S1
F986 20 58 EB           JSR LDAY
F989 91 DF              STA (NOWLN),Y   ;...AND NOVE IT UP (DOWN IN ADDR)
F98B 20 4A FA           JSR GOGO
F98E AA                 TAX
F98F AD 1A A4           LDA S1
F992 C5 E1              CMP BOTLN       ;DONE ??
F994 D0 07              BNE R5
F996 AD 1B A4           LDA S1+1
F999 C5 E2              CMP BOTLN+1
F99B F0 0E              BEQ R7
F99D 20 28 F9    R5     JSR AD1
F9A0 EE 1A A4           INC S1
F9A3 D0 03              BNE R55
F9A5 EE 1B A4           INC S1+1
F9A8 4C 84 F9    R55    JMP R6
F9AB 20 D0 F8    R7     JSR RESNOW      ;RESTORE NOWLN
F9AE 68                 PLA             ;RESTORE DIFFERENCE
F9AF 8D 2A A4           STA CPIY        ;SAVE IT
F9B2 A5 E1              LDA BOTLN
F9B4 38                 SEC
F9B5 ED 2A A4           SBC CPIY        ;AND SUBTRACT IT FROM BOTTOM
F9B8 85 E1              STA BOTLN
F9BA B0 02              BCS R9
F9BC C6 E2              DEC BOTLN+1
F9BE AD 19 A4    R9     LDA COUNT       ;C COMM OR K ,I COMM ?
F9C1 D0 04              BNE R10
F9C3 A4 EA              LDY LENGTH
F9C5 D0 05              BNE R11
F9C7 A4 EA       R10    LDY LENGTH
F9C9 D0 83              BNE R87
F9CB 60                 RTS
F9CC 4C 47 F9    R11    JMP R8
F9CF
F9CF             ;LENGTH > OLDLEN
F9CF A5 EA       R100   LDA LENGTH      ;NEW LINE IS LONGER
```

```
F9D1 38                 SEC
F9D2 E5 E9              SBC OLDLEN
F9D4 A4 E9              LDY OLDLEN
F9D6 D0 02              BNE R101        ;ALREADY HAVE ROOM FOR CR
F9D8 69 00              ADC #0          ;ADD ONE TO DIFFERENCE
F9DA 48        R101     PHA
F9DB 20 34 F9           JSR SAVNOW      ;NOWLN INTO S1
F9DE 20 C5 F8           JSR SETBOT
F9E1 A0 00              LDY #0
F9E3 B1 DF     R102     LDA (NOWLN),Y
F9E5 C9 00              CMP #0
F9E7 F0 06              BEQ R108
F9E9 20 28 F9           JSR AD1
F9EC 4C E3 F9           JMP R102
F9EF 68        R108     PLA
F9F0 48                 PHA
F9F1 18                 CLC
F9F2 65 E1              ADC BOTLN       ;ADD DIFFERENCE TO END
F9F4 85 E1              STA BOTLN       ;STORE NEW END
F9F6 90 02              BCC R103
F9F8 E6 E2              INC BOTLN+1
F9FA 20 F9 F8  R103     JSR ATEND
F9FD 90 0B              BCC R107
F9FF A5 E7              LDA SAVE        ;RESTORE OLD BOTTOM
FA01 85 E1              STA BOTLN
FA03 A5 E8              LDA SAVE+1
FA05 85 E2              STA BOTLN+1
FA07 4C 5C FA           JMP ENDERR      ;RAN PAST BUFFER END
FA0A 20 09 F9  R107     JSR NOWS1       ;SAVE CURRENT END
FA0D 68                 PLA
FA0E 18                 CLC
FA0F 65 DF              ADC NOWLN
FA11 85 DF              STA NOWLN
FA13 90 02              BCC R104
FA15 E6 E0              INC NOWLN+1
FA17 A9 1A     R104     LDA #S1
FA19 20 58 EB           JSR LDAY
FA1C 91 DF              STA (NOWLN),Y
FA1E 20 4A FA           JSR GOGO
FA21 AD 1A A4           LDA S1
FA24 CD 1C A4           CMP ADDR
FA27 D0 08              BNE R105
FA29 AD 1B A4           LDA S1+1
FA2C CD 1D A4           CMP ADDR+1      ;BACK WHERE WE STARTED ??
FA2F F0 13              BEQ R106        ;BRANCH IF DONE
FA31 20 1D F9  R105     JSR SUB
FA34 CE 1A A4           DEC S1
FA37 AD 1A A4           LDA S1
FA3A C9 FF              CMP #$FF
FA3C D0 03              BNE R1051
FA3E CE 1B A4           DEC S1+1
FA41 4C 17 FA  R1051    JMP R104
FA44 20 D0 F8  R106     JSR RESNOW
FA47 4C BE F9           JMP R9
FA4A
FA4A           ;SEE IF IT WROTE INTO MEMORY
FA4A D1 DF     GOGO     CMP (NOWLN),Y
FA4C F0 0D              BEQ GOGO1
FA4E           ;MOVE ADDRESS
FA4E A5 DF              LDA NOWLN
FA50 8D 1C A4           STA ADDR
FA53 A5 E0              LDA NOWLN+1
```

```
FA55 8D 1D A4          STA ADDR+1
FA58 4C 33 EB          JMP MEMERR
FA5B 60         GOGO1  RTS             ;OK
FA5C
FA5C 20 44 EB   ENDERR JSR CLR         ;CLEAR PNTR
FA5F A0 72             LDY #EMSG2-M1   ;PRINT "END"
FA61 20 AF E7          JSR KEP
FA64 20 D8 F6          JSR DNNO        ;BACK UP TO LAST LINE
FA67 20 42 E8          JSR TTYTST      ;IF TTY <CR>
FA6A D0 03             BNE ENDE2
FA6C 20 13 EA          JSR CRLOW
FA6F 4C 78 FA   ENDE2  JMP ERR0
FA72 20 FE E8   ERROR  JSR LL
FA75 20 D4 E7          JSR QM
FA78 20 44 EB   ERR0   JSR CLR
FA7B A2 FF             LDX #$FF
FA7D            COM    =ERR0
FA7D 9A                TXS
FA7E 20 FE E8          JSR LL          ;I/O TO TERMINAL (KB,D/P OR TTY)
FA81 D8                CLD
FA82 20 88 FA          JSR COMM
FA85 4C 78 FA          JMP ERR0
FA88
FA88            ;GET EDITOR COMMANDS & DECODE
FA88 A2 00      COMM   LDX #0
FA8A 20 BC FE          JSR PATCH8      ;READ A CHAR WITH "=< >"
FA8D A2 0B      ENTRY  LDX #COMCN1
FA8F DD AC FA   CD02   CMP COMTBL,X    ;COMPARE WITH ALLOWABLE COMMANDS
FA92 F0 0C             BEQ CFND1       ;MATCH ,SO PROCESS COMMAND
FA94 CA                DEX
FA95 10 F8             BPL CD02
FA97 20 D4 E7          JSR QM          ;NOT IN LIST ,SO NOT LEGAL COMMAND
FA9A 20 24 EA          JSR CRCK
FA9D 4C 78 FA          JMP ERR0
FAA0 20 17 FF   CFND1  JSR PATC15      ;<CR> & START DECODING COMMAND
FAA3 BD B9 FA          LDA JTBL+1,X
FAA6 8D 1B A4          STA S1+1
FAA9 6C 1A A4          JMP (S1)
FAAC
FAAC            COMCN1 =11
FAAC            ;COMMAND TABLE
FAAC 4B2052495544COMTBL .DB "K RIUDLTBFQC"
FAB2 4C5442465143
FAB8 4CF727F7CBF7JTBL   .DW DLNE,PLNE,INPU,IN,DOWN,UP
FABE 64F724F7F9F6
FAC4 E1F7D2F621F7       .DW LST,TP,BT,FCHAR,STOP,CHNG
FACA 0CF870F876F8
FAD0
FAD0            ;READ FROM MEMORY FOR ASSEMBLER
FAD0 98         MREAD  TYA
FAD1 48                PHA
FAD2 A0 00             LDY #0
FAD4 B1 DF             LDA (NOWLN),Y
FAD6 8D 2A A4          STA CPIY
FAD9 20 28 F9          JSR AD1
FADC 68                PLA
FADD A8                TAY
FADE AD 2A A4          LDA CPIY
FAE1 60                RTS
FAE2
FAE2            ;THIS PROGRAM CONVERS MNEMONIC INSTRUCTIONS INTO MACHINE
FAE2            ;CODE AND STORES IT IN THE DESIGNATED MEMORY AREA
```

```
FAE2
FAE2                ;ROM TABLE LOCATIONS:
FAE2 00020008F2FFTYPTR1 .DB 00,02,00,08,$F2,$FF,$80,01
FAE8 8001
FAEA C0E2C0C0FF00       .DB $C0,$E2,$C0,$C0,$FF,00,00
FAF0 00
FAF1 0800108040C0TYPTR2 .DB 08,00,$10,$80,$40,$C0,00,$C0
FAF7 00C0
FAF9 00400000E420       .DB $00,$40,00,00,$E4,$20,$80
FAFF 80
FB00 00FC000808F8CORR   .DB 00,$FC,00,08,08,$F8,$FC,$F4
FB06 FCF4
FB08 0C1004F40020       .DB $0C,$10,04,$F4,00,$20,$10
FB0E 10
FB0F 00000F010101SIZEM  .DB 00,00,$0F,01,01,01,$11,$11
FB15 1111
FB17 020211110212       .DB 02,02,$11,$11,02,$12,00
FB1D 00
FB1E
FB1E 000810182028STCODE .DB $00,$08,$10,$18,$20,$28,$30,$38
FB24 3038
FB26 404850586068       .DB $40,$48,$50,$58,$60,$68,$70,$78
FB2C 7078
FB2E 80889098ACA8       .DB $80,$88,$90,$98,$AC,$A8,$B0,$B8
FB34 B0B8
FB36 CCC8D0D8ECE8       .DB $CC,$C8,$D0,$D8,$EC,$E8,$F0,$F8
FB3C F0F8
FB3E 0C2C4C4C8CAC       .DB $0C,$2C,$4C,$4C,$8C,$AC,$CC,$EC
FB44 CCEC
FB46 8A9AAABACADA       .DB $8A,$9A,$AA,$BA,$CA,$DA,$EA,$FA
FB4C EAFA
FB4E 0E2E4E6E8EAE       .DB $0E,$2E,$4E,$6E,$8E,$AE,$CE,$EE
FB54 CEEE
FB56 0D2D4D6D8DAD       .DB $0D,$2D,$4D,$6D,$8D,$AD,$CD,$ED
FB5C CDED
FB5E 0D0D0C0D0E0DTYPTB  .DB 13,13,12,13,14,13,12,13
FB64 0C0D
FB66 0D0D0C0D0D0D       .DB 13,13,12,13,13,13,12,13
FB6C 0C0D
FB6E 0F0D0C0D090D       .DB 15,13,12,13,9,13,12,13
FB74 0C0D
FB76 080D0C0D080D       .DB 8,13,12,13,8,13,12,13
FB7C 0C0D
FB7E 0F060B0B040A       .DB 15,6,11,11,4,10,8,8
FB84 0808
FB86 0D0D0D0D0D0F       .DB 13,13,13,13,13,15,13,15
FB8C 0D0F
FB8E 070707070509       .DB 7,7,7,7,5,9,3,3
FB94 0303
FB96 010101010201       .DB 1,1,1,1,2,1,1,1
FB9C 0101
FB9E
FB9E                ;PROGRAM STARTS HERE
FB9E AD 25 A4    MNEENT LDA SAVPC       ;TRANSF PC TO ADDR
FBA1 8D 1C A4           STA ADDR
FBA4 AD 26 A4           LDA SAVPC+1
FBA7 8D 1D A4           STA ADDR+1
FBAA 20 24 EA    STARTM JSR CRCK        ;<CR> IF PRI PTR DIFF FROM 0
FBAD A9 00              LDA #0
FBAF 8D 37 A4           STA CODFLG
FBB2 20 3E E8           JSR BLANK
FBB5 20 DB E2           JSR WRITAZ      ;WRITE ADDRESS
```

```
FBB8 20 3B E8             JSR BLANK2
FBBB 20 3B E8             JSR BLANK2
FBBE 4C 06 FE             JMP MNEM         ;JUMP TO INPUT MNEMONIC OPCODE
FBC1 A9 00         MODEM  LDA #00          ;SET UP TO FORM MODE MATCH
FBC3 8D 26 01             STA TMASK1
FBC6 8D 27 01             STA TMASK2
FBC9 20 3E E8             JSR BLANK
FBCC AC 2E 01             LDY TYPE
FBCF 38                   SEC
FBD0 6E 26 01      PNTLUP ROR TMASK1       ;SHIFT POINTER TO INSTRUCTION TYPE
FBD3 6E 27 01             ROR TMASK2
FBD6 88                   DEY
FBD7 D0 F7                BNE PNTLUP
FBD9
FBD9               ;TEST FOR ONE BYTE INSTRUCTION
FBD9 AC 2E 01             LDY TYPE
FBDC C0 0D                CPY #$0D
FBDE D0 05                BNE RDADDR
FBE0 A2 00                LDX #00
FBE2
FBE2               ;INPUT ADRESS FIELD
FBE2 4C CB FC             JMP OPCOMP
FBE5 A0 06         RDADDR LDY #06          ;CLEAR ADDRESS FIELD (NON HEX)
FBE7 A9 51                LDA #'Q'
FBE9 99 32 01      CLRLUP STA ADFLD-1,Y
FBEC 88                   DEY
FBED D0 FA                BNE CLRLUP       ;(LEAVES Y = 0 FOR NEXT PHASE)
FBEF 20 5F E9             JSR RDRUP        ;WITH RUBOUT
FBF2 C9 20                CMP #' '         ;IGNORE SPACE CHARACTERS
FBF4 F0 EF                BEQ RDADDR
FBF6 99 33 01      STORCH STA ADFLD,Y      ;STORE ADDRESS CHARACTER
FBF9 C8                   INY
FBFA C0 07                CPY #07
FBFC B0 5C                BCS TRY56
FBFE 20 5F E9             JSR RDRUP        ;READ REMAINDER OF ADDRESS CHARS
FC01 C9 20                CMP #' '         ;THRU WHEN <SPACE> OR <CR>
FC03 D0 05                BNE STOR1
FC05 EE 37 A4             INC CODFLG       ;SET CODE FLG
FC08 D0 04                BNE EVAL
FC0A C9 0D         STOR1  CMP #CR          ;CHECK FOR <CR>
FC0C D0 E8                BNE STORCH
FC0E
FC0E               ;SEPARATE ADDRESS MODE FROM ADDRESS FIELD
FC0E 8C 31 A4      EVAL   STY TEMPX        ;TEMPX NOW HAS NUMBER OF CHAR
FC11 AD 33 01             LDA ADFLD        ;CHECK FIRST CHAR FOR # OR (
FC14 C9 23                CMP #'#'
FC16 F0 25                BEQ HATCJ
FC18 C9 28                CMP #'('
FC1A F0 5A                BEQ PAREN
FC1C AD 31 A4             LDA TEMPX        ;CHECK FOR ACCUMULATOR MODE
FC1F C9 01                CMP #01
FC21 D0 05                BNE TRYZP
FC23 A2 01         ACCUM  LDX #01
FC25 4C CB FC             JMP OPCOMP
FC28 C9 02         TRYZP  CMP #02          ;CHECK FOR ZERO PAGE MODE
FC2A D0 14                BNE TRY34
FC2C AD 2E 01             LDA TYPE         ;CHCK FOR BRNCH WITH RELATIVE ADDR`
FC2F C9 0C                CMP #$0C
FC31 D0 05                BNE ZPAGE
FC33 A2 02                LDX #02
FC35 4C CB FC             JMP OPCOMP
FC38 A2 05         ZPAGE  LDX #05
```

```
FC3A 4C CB FC              JMP OPCOMP
FC3D 4C B6 FC     HATCJ    JMP HATCH
FC40 A9 04        TRY34    LDA #04          ;CHECK FOR ABSOLUTE OR ZP,X ORZP,`
FC42 CD 31 A4              CMP TEMPX
FC45 90 15                 BCC ABSIND
FC47 A2 02                 LDX #02
FC49 20 F1 FD              JSR XORYZ        ;CC = X, CS = Y, NE = ABSOLUTE
FC4C D0 58                 BNE ABSOL
FC4E 90 05                 BCC ZPX
FC50 A2 03        ZPY      LDX #03          ;CARRY SET SO ZP,Y MODE
FC52 4C CB FC              JMP OPCOMP
FC55 A2 04        ZPX      LDX #04          ;CARRY CLEAR SO ZP,X MODE
FC57 4C CB FC              JMP OPCOMP
FC5A B0 69        TRY56    BCS ERRORM
FC5C 20 EF FD     ABSIND   JSR XORY         ;CC=ABS,X   CS=ABS,Y  NE=ERROR
FC5F D0 64                 BNE ERRORM
FC61 90 0F                 BCC ABSX
FC63 A9 09        ABSY     LDA #09
FC65 CD 2E 01              CMP TYPE
FC68 D0 04                 BNE ABSY1
FC6A A2 0E                 LDX #$0E
FC6C D0 5D                 BNE OPCOMP
FC6E A2 08        ABSY1    LDX #$08
FC70 D0 59                 BNE OPCOMP
FC72 A2 09        ABSX     LDX #09          ;CARRY CLEAR SO ABS,X MODE
FC74 D0 55                 BNE OPCOMP
FC76 AD 36 01     PAREN    LDA ADFLD+3      ;SEE IF (HH,X),(HH)Y OR (HHHH)
FC79 C9 2C                 CMP #','         ;(HHX) (HH),Y  ARE OK TOO
FC7B F0 04                 BEQ INDX         ;COMMA IN 4TH POSITION = (HH,X)
FC7D C9 58                 CMP #'X'         ;X IN 4TH POSITION = (HHX)
FC7F D0 04                 BNE TRYINY
FC81 A2 0B        INDX     LDX #$0B
FC83 D0 46                 BNE OPCOMP
FC85 C9 29        TRYINY   CMP #')'         ;")" IN 4TH POS = (HH)Y OR (HH),Y
FC87 D0 0B                 BNE TRYJMP
FC89 20 EF FD              JSR XORY         ;CHCK TO SEE IF Y INDEX REG DESIRE
FC8C D0 37                 BNE ERRORM
FC8E 90 35                 BCC ERRORM
FC90 A2 0A                 LDX #$0A
FC92 D0 37                 BNE OPCOMP
FC94 AD 38 01     TRYJMP   LDA ADFLD+5      ;CHECK FOR FINAL PAREN
FC97 C9 29                 CMP #')'
FC99 D0 2A                 BNE ERRORM
FC9B AD 2E 01              LDA TYPE         ;CONFIRM CORRECT ADDRESS TYPE
FC9E C9 0B                 CMP #$0B
FCA0 D0 23                 BNE ERRORM
FCA2 A2 0D                 LDX #$0D         ;OK, FORM IS JMP (HHHH)
FCA4 D0 25                 BNE OPCOMP
FCA6 AD 2E 01     ABSOL    LDA TYPE         ;CHECK FOR BRANCH TO ABSOLUTE LOC
FCA9 C9 0C                 CMP #$0C
FCAB D0 05                 BNE ABSOL1
FCAD A2 02                 LDX #02
FCAF 4C CB FC              JMP OPCOMP
FCB2 A2 0C        ABSOL1   LDX #$0C
FCB4 D0 15                 BNE OPCOMP
FCB6              ;SELECT IMMEDIATE ADDRESSING TYPE
FCB6 AD 2E 01     HATCH    LDA TYPE
FCB9 C9 01                 CMP #01
FCBB F0 04                 BEQ IMMED1
FCBD A2 07                 LDX #07
FCBF D0 0A                 BNE OPCOMP
FCC1 A2 06        IMMED1   LDX #06
```

```
FCC3 D0 06              BNE OPCOMP
FCC5 20 94 E3    ERRORM JSR CKER00      ;OUTPUT ERROR MESSAGE
FCC8 4C AA FB           JMP STARTM
FCCB
FCCB             ;COMPUTE FINAL OP CODE FOR DEFINED ADDRESING MODE
FCCB BD E2 FA    OPCOMP LDA TYPTR1,X    ;MATCH TYPE MASK WITH VALID MODE
FCCE F0 05              BEQ OPCMP1      ;PATTERNS & SKIP 1ST WORD TEST IF
FCD0 2D 26 01           AND TMASK1      ;ALREADY ZERO
FCD3 D0 08              BNE VALID
FCD5 BD F1 FA    OPCMP1 LDA TYPTR2,X    ;TEST 2ND PART
FCD8 2D 27 01           AND TMASK2      ;INST DOES NOT HAVE SPECIFIED MODE
FCDB F0 E8              BEQ ERRORM
FCDD 18          VALID  CLC             ;FORM FINAL OP CODE
FCDE BD 00 FB           LDA CORR,X
FCE1 6D 34 A4           ADC OPCODE
FCE4 8D 34 A4           STA OPCODE
FCE7
FCE7             ;PROCESS ADRESSES TO FINAL FORMAT
FCE7 BD 0F FB           LDA SIZEM,X     ;OBTAIN ADDRESS FORMAT FROM TABLE
FCEA C9 00              CMP #00
FCEC F0 50              BEQ ONEBYT
FCEE C9 0F              CMP #$0F        ;NEED BRANCH COMPUTATION?
FCF0 F0 1D              BEQ BRNCHC
FCF2 8D 33 A4           STA TEMPA       ;SAVE START POINT & CHAR COUNT
FCF5 29 0F              AND #$0F        ;SEPARATE CHARACTER COUNT
FCF7 A8                 TAY             ;LOAD ADDR BYTES INTO Y (0,1,OR 2)
FCF8 8D 2F A4           STA BYTESM      ;SAVE IN BYTES
FCFB EE 2F A4           INC BYTESM      ;TO INSTR LENGTH (1,2,OR 3 BYTES)
FCFE AD 33 A4           LDA TEMPA       ;SEPARATE STARTING POINT
FD01 29 F0              AND #$F0
FD03 4A                 LSR A
FD04 4A                 LSR A
FD05 4A                 LSR A
FD06 4A                 LSR A
FD07 AA                 TAX             ;AND PUT IT IN X
FD08 20 12 FD           JSR CONVRT      ;CONVERT ASCII ADDRESS TO HEX
FD0B B0 B8              BCS ERRORM      ;SKIP OUT IF ERROR IN INPUT
FD0D 90 1D              BCC STASH
FD0F 4C 86 FD    BRNCHC JMP BRCOMP
FD12
FD12             ;############ SUBROUTINE ###############
FD12             ;CONVERT FORMATTED ADDRESS INTO PROPER HEX ADDRESS
FD12 BD 33 01    CONVRT LDA ADFLD,X     ;PICK UP 1ST ADDRES CHARACTER
FD15 20 7D EA           JSR HEX         ;CONVERT TO MOST SIG HEX
FD18 B0 11              BCS ERRFLG
FD1A E8                 INX             ;GET NEXT ASCII CHARACTER
FD1B BD 33 01           LDA ADFLD,X
FD1E E8                 INX             ;POINT TO NEXT CHARACTER, IF ANY
FD1F 20 84 EA           JSR PACK
FD22 B0 07              BCS ERRFLG
FD24 99 34 A4           STA OPCODE,Y    ;SAVE IN MOST SIG. BYTE LOCATION
FD27 88                 DEY             ;SET UP FOR NEXT ADDR BYTE, IF ANY
FD28 D0 E8              BNE CONVRT      ;IF NECESSARY, FORM NEXT ADDR BYTE
FD2A 18                 CLC
FD2B 60          ERRFLG RTS             ;NON HEX CLEARED CARRY
FD2C             ;#############
FD2C
FD2C AC 2F A4    STASH  LDY BYTESM      ;SET UP TO STORE COMMAND
FD2F 88                 DEY
FD30 B9 34 A4    STSHLP LDA OPCODE,Y
FD33 20 78 EB           JSR SADDR       ;STORE ONE BYTE OF COMMAND
FD36 C0 00              CPY #00
```

```
FD38 F0 0B              BEQ FORMDS
FD3A 88                 DEY
FD3B B8                 CLV
FD3C 50 F2              BVC STSHLP      ;REPEAT TILL THRU
FD3E
FD3E A9 01      ONEBYT LDA #01          ;SET BYTES = 1
FD40 8D 2F A4           STA BYTESM
FD43 D0 E7              BNE STASH
FD45
FD45            ;FORMAT FOR SYSTEM 65 DISPLAY (REFORMAT FOR AIM)
FD45 20 44 EB   FORMDS JSR CLR
FD48 20 DD E5           JSR CGPC1       ;ADDR TO SAVPC FOR  DISASSEMBLY
FD4B 20 42 E8           JSR TTYTST      ;IF TTY DO NOT GO TO DISASS
FD4E D0 08              BNE FORMD1
FD50 20 3B E8           JSR BLANK2      ;IT IS TTY
FD53 20 3B E8           JSR BLANK2
FD56 D0 11              BNE FORMD2      ;OUTPUT OPCODE
FD58 20 6C F4   FORMD1 JSR DISASM
FD5B 20 24 EA           JSR CRCK        ;<CR> IF PRI PTR DIFF FROM 0
FD5E AD 37 A4           LDA CODFLG      ;SEE IF HE WANTS CODE ALSO
FD61 F0 1A              BEQ FORM1
FD63 20 3E E8           JSR BLANK
FD66 20 3C F5           JSR PRPC        ;PROG CNTR
FD69            ;OUTPUT OPCODE
FD69 AE 2F A4   FORMD2 LDX BYTESM
FD6C A0 00              LDY #00
FD6E A9 1C      DISPLY LDA #ADDR        ;DO LDA (ADDR),Y ,WHITOUT PAG 0
FD70 20 58 EB           JSR LDAY
FD73 20 46 EA           JSR NUMA
FD76 20 3E E8           JSR BLANK
FD79 C8                 INY
FD7A CA                 DEX
FD7B D0 F1              BNE DISPLY
FD7D
FD7D            ;POINT TO NEXT INSTRUCTION LOCATION
FD7D AC 2F A4   FORM1  LDY BYTESM       ;ADD BYTESM TO ADDR
FD80 20 CD E2           JSR NXTADD
FD83 4C 24 FF           JMP PATC16      ;UPDATE PC
FD86
FD86            ;RELATIVE BRANCH ADDRESS COMPUTATION
FD86 AD 31 A4   BRCOMP LDA TEMPX
FD89 C9 02              CMP #02         ;IF REL BRANCH INPUT, USE IT
FD8B D0 11              BNE COMPBR
FD8D A2 00              LDX #00
FD8F A0 01              LDY #01
FD91 20 12 FD           JSR CONVRT
FD94 B0 40              BCS ERRJMP
FD96 A9 02              LDA #02
FD98 8D 2F A4           STA BYTESM      ;SET PROPER BYTES
FD9B 4C 2C FD           JMP STASH
FD9E A2 00      COMPBR LDX #00
FDA0 A0 02              LDY #02
FDA2 20 12 FD           JSR CONVRT
FDA5 B0 2F              BCS ERRJMP
FDA7 AD 1D A4           LDA ADDR+1      ;ADD BRANCH OFFSET
FDAA 8D 27 01           STA MOVAD+1
FDAD AD 1C A4           LDA ADDR
FDB0 18                 CLC
FDB1 69 02              ADC #02
FDB3 8D 26 01           STA MOVAD
FDB6 90 03              BCC CMPBR1
FDB8 EE 27 01           INC MOVAD+1
```

```
FDBB 38          CMPBR1 SEC              ;COMPUTE BRANCH RELATIVE ADDRESS
FDBC AD 35 A4           LDA OPCODE+1
FDBF ED 26 01           SBC MOVAD
FDC2 8D 35 A4           STA OPCODE+1
FDC5 AD 36 A4           LDA OPCODE+2
FDC8 ED 27 01           SBC MOVAD+1
FDCB 8D 36 A4           STA OPCODE+2
FDCE C9 00              CMP #00
FDD0 F0 0E              BEQ FORWRD
FDD2 C9 FF              CMP #$FF
FDD4 F0 03              BEQ BACKWD
FDD6 4C C5 FC    ERRJMP JMP ERRORM
FDD9 AD 35 A4    BACKWD LDA OPCODE+1     ;CHECK IN RANGE
FDDC 30 09              BMI OK
FDDE 10 F6              BPL ERRJMP
FDE0 AD 35 A4    FORWRD LDA OPCODE+1
FDE3 10 02              BPL OK
FDE5 30 EF              BMI ERRJMP
FDE7 A9 02       OK     LDA #02          ;SET UP FOR STASH
FDE9 8D 2F A4           STA BYTESM
FDEC 4C 2C FD           JMP STASH
FDEF
FDEF             ;###### SUBROUTINE ########
FDEF             ;SUBROUTINE FOR DETERMINING X OR Y OR NEITHER
FDEF A2 04       XORY   LDX #04
FDF1 BD 33 01    XORYZ  LDA ADFLD,X
FDF4 C9 2C              CMP #','
FDF6 D0 04              BNE XORY1
FDF8 E8                 INX
FDF9 BD 33 01           LDA ADFLD,X
FDFC C9 58       XORY1  CMP #'X'
FDFE F0 03              BEQ ISX
FE00 C9 59              CMP #'Y'
FE02             XORYRT
FE02 60                 RTS              ;NOT ZERO IS NOT X OR NOT Y
FE03 18          ISX    CLC              ;CARRY SET IS Y
FE04 90 FC              BCC XORYRT       ; CARRY CLEAR IS X
FE06             ;####### END OF SUB ########
FE06
FE06             ; INPUT FOR MNEMONIC CODE
FE06 A0 00       MNEM   LDY #00
FE08 8C 34 A4           STY OPCODE
FE0B 8C 35 A4           STY OPCODE+1
FE0E 8C 36 A4           STY OPCODE+2     ;CLEARS OPCODE FOR NEW INPUT
FE11 8C 26 01           STY MOVAD        ;CLEARS UNUSED BIT IN FINAL FORMAT
FE14 20 5F E9    RDLUP  JSR RDRUP
FE17 C9 2A              CMP #'*'         ;COMMAND TO LOAD POINTER
FE19 F0 58              BEQ STLOAD       ;GO TO SET CURRENT ADDRESS POINTER
FE1B C9 20              CMP #' '         ;IGNORE SPACE BAR INPUT
FE1D F0 F5              BEQ RDLUP
FE1F 29 1F              AND #$1F         ;MASK OFF UPPER 3 BITS
FE21 99 30 01           STA CH,Y
FE24 98                 TYA
FE25 AA                 TAX              ;Y----> X
FE26 FE 30 01           INC CH,X         ;FORMAT TO MATCH DISASSEMBLER TBL
FE29 C8                 INY
FE2A C0 03              CPY #03          ;REPEAT FOR EACH OF 3 CHARACTERS
FE2C D0 E6              BNE RDLUP
FE2E
FE2E             ;COMPRESS 3 FORMATED CHARACTERS TO MOVAD & MOVAD+1
FE2E A0 03              LDY #03          ;SET UP OUTER LOOP
FE30 B9 2F 01    OUTLUP LDA CH-1,Y       ;COMPRESS 3 CHARACTERS
```

```
FE33 A2 05              LDX #05         ;SET UP INNER LOOP
FE35 4A          INLUP  LSR A           ;SHIFT 5 BITS ACC TO MOVAD,MOVAD+1
FE36 6E 26 01           ROR MOVAD
FE39 6E 27 01           ROR MOVAD+1
FE3C CA                 DEX
FE3D D0 F6              BNE INLUP
FE3F 88                 DEY
FE40 D0 EE              BNE OUTLUP
FE42
FE42             ;SEARCH FOR MATCHING COMPRESSED CODE
FE42 A2 40              LDX #$40
FE44 AD 26 01    SRCHLP LDA MOVAD
FE47 DD B8 F5    SRCHM  CMP MNEML-1,X   ;MATCH LEFT HALF
FE4A F0 05              BEQ MATCH
FE4C CA                 DEX
FE4D D0 F8              BNE SRCHM       ;IF NO - TRY AGAIN
FE4F F0 0B              BEQ MATCH1
FE51 AD 27 01    MATCH  LDA MOVAD+1     ;ALSO MATCH RIGHT HALF
FE54 DD F8 F5           CMP MNEMR-1,X
FE57 F0 06              BEQ GOTIT
FE59 CA                 DEX
FE5A D0 E8              BNE SRCHLP
FE5C 4C C5 FC    MATCH1 JMP ERRORM
FE5F
FE5F             ;GET INSTRUCTION TYPE FROM TYPE TABLE
FE5F BD 5D FB    GOTIT  LDA TYPTB-1,X
FE62 8D 2E 01           STA TYPE
FE65
FE65             ;GET OPCODE FROM OP CODE UE
FE65 BD 1D FB           LDA STCODE-1,X
FE68 8D 34 A4           STA OPCODE
FE6B 4C C1 FB           JMP MODEM
FE6E
FE6E             ;THIS SECTION SETS THE CURRENT ADDRESS POINTER
FE6E A9 2A       STLO   LDA #'*'
FE70 20 7A E9           JSR OUTPUT
FE73 20 AE EA    STLOAD JSR ADDIN       ;GET ADDR
FE76 B0 F6              BCS STLO        ;IN CASE OF ERROR
FE78 4C 24 FF           JMP PATC16      ;ADDR TO PC THEN TO STARTM
FE7B
FE7B             ;PATCHES TO CORRECT PROBLEMS WITHOUT
FE7B             ;CHANGING ENTRY POINTS TO THE ROUTINES
FE7B 41                 .DB "A"
FE7C 38          PATCH1 SEC             ;ADJUST BAUD
FE7D E9 2C              SBC #44
FE7F 8D 18 A4           STA CNTL30
FE82 60                 RTS
FE83
FE83 8A          CUREAD TXA             ;SAVE X  , OUTPUT CUR
FE84 48                 PHA
FE85 AE 15 A4           LDX CURPO2
FE88 E0 14              CPX #20         ;ONLY IF < 20
FE8A B0 05              BCS PAT2A
FE8C A9 DE              LDA #$DE
FE8E 20 7B EF           JSR OUTDD1
FE91 68          PAT2A  PLA
FE92 AA                 TAX
FE93 4C 3C E9           JMP READ        ;CONTINUE
FE96
FE96 20 3C E9    RED1   JSR READ        ;READ & ECHO WITHOUT CURSOR
FE99 4C 76 E9           JMP RED2
FE9C
```

```
FE9C AE 15 A4    PATCH4 LDX CURPO2       ;DONT DO ANYTHING IF "8D"
FE9F C9 8D              CMP #CR+$80      ;SO <CR> FOR TV & NOT FOR DISP
FEA1 D0 0B              BNE PAT4A
FEA3 A9 A0              LDA #' '+$80     ;CLR CURSOR
FEA5 20 7B EF           JSR OUTDD1
FEA8 20 44 EB           JSR CLR          ;CLR PNTRS
FEAB 4C 76 EF           JMP OUTD7        ;EXIT
FEAE 4C 17 EF    PAT4A  JMP OUTD1A       ;CONTINUE
FEB1
FEB1 8D 11 A4    PATCH5 STA PRIFLG       ;TURN PRI OFF
FEB4 4C 73 F0           JMP IPO3
FEB7
FEB7 A9 1C       PATCH6 LDA #ADDR        ;SIMULATE LDA (ADDR),Y
FEB9 4C 58 EB           JMP LDAY
FEBC
FEBC 20 3C E9    PATCH8 JSR READ         ;READ & ECHO WITH CARROTS
FEBF 48                 PHA
FEC0 20 D8 E7           JSR EQUAL
FEC3 A9 3C              LDA #'<'
FEC5 20 7A E9           JSR OUTPUT
FEC8 68                 PLA
FEC9 48                 PHA
FECA C9 0D              CMP #CR
FECC F0 03              BEQ PATC8C
FECE 20 7A E9           JSR OUTPUT
FED1 A9 3E       PATC8C LDA #'>'
FED3 20 7A E9           JSR OUTPUT
FED6 68                 PLA
FED7 60                 RTS
FED8
FED8 C9 F7       PATCH9 CMP #$F7         ;CHCK LOWER TRANSITION OF TIMER
FEDA B0 06              BCS PAT9A
FEDC CD 08 A4           CMP TSPEED
FEDF 4C 9D EE           JMP CKF3A
FEE2 CD 08 A4    PAT9A  CMP TSPEED
FEE5 68                 PLA
FEE6 C9 FF              CMP #$FF
FEE8 60          PAT9B  RTS
FEE9
FEE9 20 F0 E9    PATC10 JSR CRLF         ;CLR DISP (ONLY 1 <CR>)
FEEC 4C 85 E1           JMP STA1
FEEF
FEEF F0 F7       PATC11 BEQ PAT9B        ;GO OUTPUT PROMPT
FEF1 C9 4C              CMP #'L'         ;NO PROMPT FOR "T" OR "L"
FEF3 F0 F3              BEQ PAT9B
FEF5 4C C5 E7           JMP PROMP1
FEF8
FEF8 48          PATC12 PHA              ;CLEAR PRIFLG SO WE CAN OUTPUT
FEF9 AD 11 A4           LDA PRIFLG       ;TO PRINTER IF FLG WAS ON (MSB)
FEFC 29 F0              AND #$F0
FEFE 8D 11 A4           STA PRIFLG
FF01 68                 PLA
FF02 60                 RTS
FF03
FF03 AD 12 A4    PATC13 LDA INFLG        ;TURN TAPES ON ONLY IF TAPES
FF06 C9 54              CMP #'T'
FF08 D0 DE              BNE PAT9B
FF0A 4C 29 E5           JMP DU14         ;TURN ON TAPES & SET DEF DEV
FF0D
FF0D AD 13 A4    PATC14 LDA OUTFLG       ;TURN ON TAPES ONLY IF TAPES
FF10 C9 54              CMP #'T'
FF12 D0 D4              BNE PAT9B
```

```
FF14 4C 0A E5           JMP DU11
FF17
FF17 20 F0 E9    PATC15 JSR CRLF        ;DECODE COMMAND
FF1A 8A                 TXA             ;SAVE INDEX
FF1B 0A                 ASL A
FF1C AA                 TAX
FF1D BD B8 FA           LDA JTBL,X      ;PART OF ENTRY
FF20 8D 1A A4           STA S1
FF23 60                 RTS
FF24
FF24 20 DD E5    PATC16 JSR CGPC1       ;ADDR TO PC
FF27 4C AA FB           JMP STARTM      ;BACK TO MNEMONIC START
FF2A
FF2A F0 0E       PATC17 BEQ PAT17B      ;RUB, SO READ ANOTHER
FF2C C9 00              CMP #0
FF2E F0 03              BEQ PAT17A
FF30 4C 85 F7           JMP IN02A       ;NEITHER ,CONTINUE
FF33 20 93 E9    PAT17A JSR INALL       ;SKIP ON ZEROS
FF36 C9 7F              CMP #$7F        ;UNTILL RUB
FF38 D0 F9              BNE PAT17A
FF3A 4C 7A F7    PAT17B JMP IN02        ;GO BACK
FF3D
FF3D 20 F8 FE    PATC18 JSR PATC12      ;RESET PRIFLG
FF40 48                 PHA
FF41 20 42 E8           JSR TTYTST      ;IF TTY JUST RTN
FF44 D0 02              BNE PAT18A
FF46 68                 PLA
FF47 60                 RTS
FF48 20 FE E8    PAT18A JSR LL          ;SET TO LOW SPEED
FF4B 20 45 F0           JSR IPST        ;PRINT WHAT IS IN BUFFER
FF4E 20 44 EB           JSR CLR         ;CLR PRINTER BUFFER BY OUTPUTTING
FF51 20 3E E8           JSR BLANK       ;AN SPACE
FF54 20 44 EB           JSR CLR
FF57 68                 PLA             ;RTN ACC
FF58 60                 RTS
FF59
FF59 D8          PAT19  CLD
FF5A 20 24 EA           JSR CRCK
FF5D 4C 85 E1           JMP STA1
FF60
FF60 F0 0D       PAT20  BEQ VECK4       ;END (DATA BYTES=0)
FF62 18                 CLC
FF63 69 04              ADC #4
FF65 AA                 TAX
FF66 20 93 E9    VECK5  JSR INALL       ;SKIP OVER DATA
FF69 CA                 DEX
FF6A D0 FA              BNE VECK5
FF6C 4C 9E E6           JMP VECK1       ;PROCESS NEXT RCD
FF6F 4C 20 E5    VECK4  JMP DU13
FF72
FF72 A0 00       PAT21  LDY #0
FF74 B9 88 FF    PAT21A LDA POMSG,Y     ;RESET MSG
FF77 F0 06              BEQ PAT21B
FF79 20 7A E9           JSR OUTPUT
FF7C C8                 INY
FF7D D0 F5              BNE PAT21A
FF7F 20 F0 E9    PAT21B JSR CRLF
FF82 20 F0 E9           JSR CRLF
FF85 4C 82 E1           JMP START
FF88
FF88 2020524F434BPOMSG  .DB "  ROCKWELL AIM 65"
FF8E 57454C4C2041494D203635
```

```
FF99 00                 .DB 0
FF9A
FF9A EE 68 01    PAT22  INC BLKO
FF9D 4C BD ED           JMP ADDBK1
FFA0
FFA0 A9 FF       PAT23  LDA #$FF        ;START TIMER
FFA2 8D 97 A4           STA DI1024
FFA5 AD 85 A4    PAT23A LDA RINT        ;TIME OUT?
FFA8 30 08              BMI PAT23B      ;YES
FFAA AD 0D A8           LDA IFR         ;START SIGNAL?
FFAD 29 10              AND #MPRST
FFAF F0 F4              BEQ PAT23A      ;NO
FFB1 60                 RTS             ;YES
FFB2 A9 00       PAT23B LDA #0          ;TIME OUT RETURN
FFB4 60                 RTS
FFB5
FFB5 20 75 EE    PATC24 JSR CKFREQ      ;READ BIT FROM FOURTH HALF PULSE
FFB8 6A                 ROR A
FFB9 29 80              AND #$80
FFBB 60                 RTS
FFBC
FFBC 2C 0D A8    PATC25 BIT IFR         ;WAIT TILL TIMES OUT
FFBF 50 FB              BVC PATC25
FFC1 AD 04 A8           LDA T1L         ;CLR INTERRUPT FLG
FFC4 60                 RTS
FFC5
FFF9                    *=$FFF9
FFF9             ;INTERRUPT VECTORS
FFF9 FA                 .DB $FA
FFFA 75E0BFE078E0       .DW NMIV1,RSET,IRQV1    ;SET UP VECTORS
10000             ;.END A0/1
10000             SEMICOLON =$3B
10000             BACKSLASH =$5C
10000                    .END M1
```

Label	Value	Label	Value	Label	Value
ASSEM	D000	ADFLD	0133	ADDR	A41C
ACR	A80B	ADDS1	E55D	ADD1	E565
ADDIN	EAAE	ADDNE	EAB1	ADDN1	EAB7
ADDN2	EAC7	ADDN3	EADC	ADDN4	EAE8
ADDN5	EAF7	ADDN6	EAFD	ADDN7	EB0D
ADDN8	EB2B	ADDBLK	EDBA	ADDBK1	EDBD
ATTOP	F8DB	ATBOT	F8E9	AT02	F8F5
AT01	F8F7	ATEND	F8F9	ADDRS1	F910
ADDS1A	F916	AD1	F928	ADDA	F92A
ADDA1	F933	ACCUM	FC23	ABSIND	FC5C
ABSY	FC63	ABSY1	FC6E	ABSX	FC72
ABSOL	FCA6	ABSOL1	FCB2	BASIEN	B000
BASIRE	B003	BOTLN	00E1	BKS	0100
BYTESM	A42F	BKFLG	A410	BLK	0115
BLKO	0168	BRKA	E61B	BRK1	E620
BKERR	E62F	BKOK	E634	BKO2	E64C
BRKK	E6E5	BRK3	E6F1	BRK2	E6F3
BRK4	E6FA	BLANK2	E83B	BLANK	E83E
BKCKSM	F1E7	BKCK1	F1F1	BKCK2	F20F
BKCK3	F21A	BT	F721	BRNCHC	FD0F
BRCOMP	FD86	BACKWD	FDD9	BACKSLASH	005C
CH	0130	CODFLG	A437	CURPO2	A415
CURPOS	A416	CNTH30	A417	CNTL30	A418

COUNT	A419	CKSUM	A41E	CPIY	A42A
CRA	AC01	CRB	AC03	CR	000D
COMIN	E1A1	COMB	E1C4	CHNGG	E2A0
CHNG1	E2A6	CH2	E2B8	CH4	E2C0
CH3	E2C5	CKERR	E385	CKER0	E38E
CKER00	E394	CKER1	E396	CKER2	E3A3
CHEKAR	E54B	CHEKA	E54E	CGPC	E5D4
CGPC0	E5D7	CGPC1	E5DD	CGPS	E5EA
CGA	E5EE	CGX	E5F2	CGY	E5F6
CGS	E5FA	CGALL	E5FC	CLRBK	E6FE
CKB	E76B	CKB2	E76D	CKB1	E780
CRLF	E9F0	CRLOW	EA13	CR2J	EA23
CRCK	EA24	CRCK1	EA2C	CRCK2	EA3B
CLR	EB44	CLRCK	EB4D	CKFREQ	EE75
CKF1	EE7A	CKF2	EE81	CKF3	EE99
CKF3A	EE9D	CKF4	EEA1	CKBUFF	F1D2
CBUFF1	F1E2	COL0	F2E1	COL1	F321
COL2	F361	COL3	F3A1	COL4	F3E1
CHAR1	F5AD	CHAR2	F5B3	CHNG	F876
CHN1	F87C	CHN2	F88C	CHN3	F8A9
CHN4	F8AF	CFLG	F8B2	COM	FA78
COMM	FA88	CD02	FA8F	CFND1	FAA0
COMCN1	000B	COMTBL	FAAC	CORR	FB00
CLRLUP	FBE9	CONVRT	FD12	COMPBR	FD9E
CMPBR1	FDBB	CUREAD	FE83	DILINK	A406
DISFLG	A40F	DIBUFF	A438	DRA2	A480
DDRA2	A481	DRB2	A482	DDRB2	A483
DNPA7	A484	DPPA7	A485	DIV1	A494
DIV8	A495	DIV64	A496	DI1024	A497
DRB	A800	DRAH	A801	DDRB	A802
DDRA	A803	DRA	A80F	DATIN	000E
DATOUT	000C	DEBTIM	1388	DUMP	E43B
DU1	E444	DU0	E447	DU1B	E452
DU1A	E46D	DU2	E47D	DU6	E49F
DU7	E4A0	DU8	E4A2	DU9	E4B9
DU10	E4DB	DU10A	E4F8	DU11	E50A
DU12	E511	DU13	E520	DU14	E529
DUMPTA	E56F	DUMPT1	E57B	DUMPKI	E587
DUK2	E5A4	DONE	E790	DON1	E7A0
DELAY	EC0F	DE1	EC18	DE2	EC1B
DEHALF	EC23	DEBKEY	ED2A	DEBK1	ED2C
DISASM	F46C	DNNO	F6D8	DOW1	F6E3
DOW2	F6E8	DOWN	F724	DLNE	F74C
DISPLY	FD6E	END	00E5	ENPA7	A486
EPPA7	A487	ESCAPE	001B	EQS	00BD
EMSG1	E06C	EMSG2	E072	EQUAL	E7D8
ERR	F495	EDIT	F639	EDI0	F644
EDI1	F653	EDI2	F663	EDI3	F673
EDI4	F680	EDI5	F68D	EDI6	F69B
EDI7	F6AA	EDI8	F6AE	EDI	F6B6
EDI2B	F6CC	ENDERR	FA5C	ENDE2	FA6F
ERROR	FA72	ERR0	FA78	ENTRY	FA8D
EVAL	FC0E	ERRORM	FCC5	ERRFLG	FD2B
ERRJMP	FDD6	FORMA	0116	FROM	E7A3
FNAM	E8A2	FCHAR	F80C	FCHA1	F80F
FCH	F81E	FC1	F823	FC2	F82E
FC3	F834	FC4	F843	FC5	F849
FC6	F84E	FC7	F853	FC8	F85A
FC9	F868	FORMDS	FD45	FORMD1	FD58
FORMD2	FD69	FORM1	FD7D	FORWRD	FDE0
GAP	A409	GO	E261	GOBK	E26D
GOBK0	E278	GOBK1	E286	GETID	E425

GID1	E427	GOERR	E608	GCNT	E785
GCN1	E78C	GETTTY	EBDB	GET1	EBE2
GET3	EBED	GETKD0	EC38	GETKEY	EC40
GETKY	EC43	GETK0	EC55	GETK00	EC67
GETK1	EC71	GETK1B	EC80	GETK2	EC82
GETK3	EC8D	GETK4	EC93	GETK5	ECA4
GETK6	ECB9	GETK7	ECBE	GETK8	ECBF
GETK11	ECC9	GETK12	ECD2	GETK13	ECE1
GETK14	ECEB	GETK10	ECEC	GETTAP	EE29
GETA1	EE2B	GETFMT	F499	GOGO	FA4A
GOGO1	FA5B	GOTIT	FE5F	HISTM	A42E
HISTP	A414	HIST	A42E	HEX	EA7D
HATCJ	FC3D	HATCH	FCB6	IRQV4	A400
IRQV2	A404	INFLG	A412	IBUFM	A460
IDIR	A474	ICOL	A475	IOFFST	A476
IDOT	A477	IOUTL	A478	IOUTU	A479
IBITL	A47A	IBITU	A47B	IMASK	A47C
IFR	A80D	IER	A80E	IRQV1	E078
IRQV3	E154	IRQ1	E163	IRQ2	E17F
INCS2	E566	INTAB1	E743	INTAB2	E752
INTAB3	E756	INLOW	E8F8	INALL	E993
IPST	F045	IPS0	F04A	IPO0	F050
IPO2	F066	IPO3	F073	IPO4	F078
IPSU	F0E3	IPS1	F0E8	IPS3	F105
IPS2	F10E	INCP	F121	IEVEN	F486
IN	F764	INL	F76D	IN02	F77A
IN02A	F785	IN03B	F799	IN03	F7A8
IN03A	F7B9	IN05	F7C5	INPU	F7CB
INPU1	F7D8	INDX	FC81	IMMED1	FCC1
ISX	FE03	INLUP	FE35	JUMP	A47D
JMPR	E1C1	JD1	E723	JD2	E72B
JD3	E73C	JD4	E742	JTBL	FAB8
KEYF1	010C	KEYF2	010F	KEYF3	0112
KMASK	A42A	KDISA	E70A	KEP	E7AF
KEPR	E970	KIFLG	F8B6	KI2	F8B8
LENGTH	00EA	LMNEM	0117	LDIY	A42A
LF	000A	LOAD	E2E6	LOAD1	E2E9
LOAD2	E306	LOAD4	E321	LOAD5	E323
LOADTA	E32F	LOAD1A	E349	LOADT2	E364
LOADKI	E3A4	LOADK1	E3A7	LOADK2	E3AA
LOADK3	E3B7	LOADK5	E3D1	LOADK6	E3D3
LOADK7	E3E8	LL	E8FE	LT10	EA5A
LDAY	EB58	LST	F7E1	LST01	F7F0
LST02	F7F8	LST3	F803	MOVAD	0126
MONRAM	A400	MON	00C0	MOFF	00E0
MPRST	0010	MSP12	0002	MT2	0020
M1	E000	M3	E005	M4	E008
M5	E01C	M6	E021	M7	E024
M8	E027	M9	E02A	M10	E02D
M11	E031	M12	E03B	MCM2	E196
MCM3	E1AC	MCNT	0020	MONCOM	E1E5
MEM	E248	MEIN	E24D	MEM1	E24F
MEM2	E251	MEM3	E260	MEMERR	EB33
MTBL	F2D7	MNNDX1	F4AF	MNNDX2	F4B3
MNNDX3	F4BA	MR11A	F512	MODE	F55B
MODE2	F59F	MNEML	F5B9	MNEMR	F5F9
MREAD	FAD0	MNEENT	FB9E	MODEM	FBC1
MNEM	FE06	MATCH	FE51	MATCH1	FE5C
NOWLN	00DF	NMIV2	A402	NPUL	A40A
NAME	A42E	NULLC	00FF	NMIV1	E075
NMIV3	E07B	NMI4	E0B1	NMI5	E0B4
NXTADD	E2CD	NXTA1	E2DA	NXT5	E60D

NHIS	E688	NH1	E690	NAMO	E8CF
NAMO1	E8D6	NAMO2	E8E9	NAMO3	E8EB
NAMO4	E8F5	NUMA	EA46	NOUT	EA51
NEWROW	F160	NEWCOL	F163	NOWS1	F909
OLDLEN	00E9	OPCODE	A434	OUTFLG	A413
OUTCKS	E531	OUTCK	E538	OUTCK1	E53B
OUTCK2	E547	OUTLOW	E901	OUTL1	E906
OUTPUT	E97A	OUT1	E97B	OUT1A	E986
OUT2	E98F	OUTALL	E9BC	OUTA1	E9C8
OUTA2	E9D0	OUTA3	E9E2	OUTA4	E9EA
ONEKEY	ED05	ONEK1	ED09	ONEK2	ED0B
ONEK3	ED1C	ONEK4	ED29	OUTTTY	EEA8
OUTT1	EECB	OUTT2	EEFB	OUTDP	EEFC
OUTDP1	EF02	OUTDIS	EF05	OUTD1	EF14
OUTD1A	EF17	OUTD2	EF20	OUTD2A	EF2F
OUTD3	EF33	OUTD4	EF48	OUTD5	EF56
OUTD7	EF76	OUTDD1	EF7B	OUTDD2	EF87
OUTDD3	EF8B	OUTPRI	F000	OUT01	F00F
OUT04	F025	OUT05	F033	OUTPR	F038
OUTPR1	F03A	OUTPR2	F044	OP04	F130
OP07	F13F	OP03	F144	OP05	F150
OP06	F15D	OUTTAP	F24A	OUTTA1	F290
OUTTA2	F294	OUTTA3	F2B2	OPCOMP	FCCB
OPCMP1	FCD5	ONEBYT	FD3E	OK	FDE7
OUTLUP	FE30	PRIFLG	A411	PCR	A80C
PRST	0000	PRTIME	06A4	PRITR	E6E1
PROMPT	E7BD	PROMP1	E7C5	PR1	E7CC
PR2	E7CF	PSLS	E7DC	PSL0	E7FB
PSL00	E802	PSL0A	E814	PSL0B	E81C
PSL0C	E81E	PSL0D	E823	PSL1	E837
PACK	EA84	PAK1	EA96	PAK2	EA9F
PCLLD	EB56	PHXY	EB9E	PLXY	EBAC
PRIERR	F079	PRNDOT	F087	PRDOT0	F08C
PINT	F0CB	PRMN1	F4D7	PRMN2	F4DB
PRADR1	F4F7	PRADR2	F4FF	PRADR3	F519
PRADR4	F52C	PRNTXY	F538	PRPC	F53C
PRBL2	F545	PCADJ3	F54D	PCADJ4	F554
PLNE	F727	P02	F729	P01	F73B
P03	F73F	P00	F749	PNTLUP	FBD0
PAREN	FC76	PATCH1	FE7C	PAT2A	FE91
PATCH4	FE9C	PAT4A	FEAE	PATCH5	FEB1
PATCH6	FEB7	PATCH8	FEBC	PATC8C	FED1
PATCH9	FED8	PAT9A	FEE2	PAT9B	FEE8
PATC10	FEE9	PATC11	FEEF	PATC12	FEF8
PATC13	FF03	PATC14	FF0D	PATC15	FF17
PATC16	FF24	PATC17	FF2A	PAT17A	FF33
PAT17B	FF3A	PATC18	FF3D	PAT18A	FF48
PAT19	FF59	PAT20	FF60	PAT21	FF72
PAT21A	FF74	PAT21B	FF7F	POMSG	FF88
PAT22	FF9A	PAT23	FFA0	PAT23A	FFA5
PAT23B	FFB2	PATC24	FFB5	PATC25	FFBC
QM	E7D4	RMNEM	0118	REGF	A40E
ROLLFL	A47F	RINT	A485	RA	AC00
RB	AC02	RUB	0008	RSET	E0BF
RS1	E0C9	RS2	E0D4	RS3A	E0F1
RS3	E0F3	RS3B	E11A	RS4	E11D
RS5	E129	RS6	E13E	RS7	E144
RS8	E146	REG	E227	REG1	E232
RBYTE	E3FD	RBYT1	E407	REGT	E6D9
RS20	E702	RCHEK	E907	RCH2	E91F
RCH3	E925	RCHTTY	E926	RCHT2	E928
RCHT1	E93B	READ	E93C	READ1	E94A

READ2	E94D	REA1	E956	RB2	E95C
RDRUP	E95F	RDR1	E96A	REDOUT	E973
RED2	E976	RD2	EA5D	RD1	EA70
RSPAC	EA7B	ROONEK	ECEF	ROO1	ED00
RDBIT	EE3B	RDBIT1	EE43	RDBIT2	EE51
RDBIT4	EE67	ROUT	F286	ROUT1	F28B
ROW1	F421	ROW2	F429	ROW3	F431
ROW4	F439	ROW5	F441	ROW6	F449
ROW7	F451	ROW8	F459	REGQ	F461
RTMODE	F491	RELADR	F530	RTS1	F55A
REENTR	F6CF	RESNOW	F8D0	REP2	F93E
REPLAC	F93F	R8	F947	R87	F94E
R88	F953	R2W	F95F	RQP	F977
R6	F984	R5	F99D	R55	F9A8
R7	F9AB	R9	F9BE	R10	F9C7
R11	F9CC	R100	F9CF	R101	F9DA
R102	F9E3	R108	F9EF	R103	F9FA
R107	FA0A	R104	FA17	R105	FA31
R1051	FA41	R106	FA44	RDADDR	FBE5
RDLUP	FE14	RED1	FE96	SAVE	00E7
STRING	00EB	S1	A41A	S2	0106
SAVPS	A420	SAVA	A421	SAVX	A422
SAVY	A423	SAVS	A424	SAVPC	A425
STIY	A427	STBKEY	A42B	SR	A80A
SP12	0001	SETREG	E113	START	E182
STA1	E185	STBYTE	E413	SHOW	E64D
SH1	E652	SHIS	E665	SH11	E66A
SEMI	E9BA	SADDR	EB78	SWSTAK	EBBA
SWST1	EBBD	SYNC	EDFF	SYNC1	EE11
SETZ	F282	SETSPD	F2C0	SETSP1	F2CA
SETSP2	F2D3	STOP	F870	SETBOT	F8C5
SUB	F91D	SUB1	F927	SAVNOW	F934
SIZEM	FB0F	STCODE	FB1E	STARTM	FBAA
STORCH	FBF6	STOR1	FC0A	STASH	FD2C
STSHLP	FD30	SRCHLP	FE44	SRCHM	FE47
STLO	FE6E	STLOAD	FE73	SEMICOLON	003B
TEXT	00E3	TYPE	012E	TMASK1	0126
TMASK2	0127	TEMPX	A431	TEMPA	A433
TSPEED	A408	TIMG	A40B	TAPIN	A434
TAPOUT	A435	TAPTR	A436	TAPTR2	A437
TABUFF	0116	TABUF2	00AD	T1L	A804
T1CH	A805	T1LL	A806	T1LH	A807
T2L	A808	T2H	A809	T2I	0000
T1I	0000	T1FR	00C0	TMSG0	E048
TMSG1	E04D	TMSG2	E050	TMSG3	E052
TMSG5	E05F	TMSG6	E061	TMSG7	E066
TOGTA1	E6BD	TOGTA2	E6CB	TRACE	E6DD
TOGL	E6E7	TOGL1	E6F6	TO	E7A7
TO1	E7A9	TTYTST	E842	TAP1	E8B3
TAP2	E8BC	TAP3	E8C2	TIBYTE	ED3B
TIB1	ED48	TIBY1	ED53	TIBY3	ED56
TIBY4	ED63	TIBY5	ED65	TIBY5A	ED88
TIBY6	EDAF	TIBY7	EDB0	TAISET	EDEA
TIOSET	EE1C	TIOS1	EE22	TIOS2	EE24
TOBYTE	F18B	TABY2	F1A7	TABY3	F1CE
TAOSET	F21D	TAOS1	F238	TRY	F258
TP	F6D2	TOPNO	F8BC	TPO1	F8C0
TYPTR1	FAE2	TYPTR2	FAF1	TYPTB	FB5E
TRYZP	FC28	TRY34	FC40	TRY56	FC5A
TRYINY	FC85	TRYJMP	FC94	UDRB	A000
UDRAH	A001	UDDRB	A002	UDDRA	A003
UT1L	A004	UT1CH	A005	UT1LL	A006

```
UT1LH          A007      UT2L           A008      UT2H           A009
USR            A00A      UACR           A00B      UPCR           A00C
UIFR           A00D      UIER           A00E      UDRA           A00F
UIN            0108      UOUT           010A      UP             F6F9
UPNO           F709      UP1            F713      UP4            F720
VECKSM         E694      VECK1          E69E      VECK2          E6AC
VALID          FCDD      VECK5          FF66      VECK4          FF6F
WRITAZ         E2DB      WRITAD         E2DD      WHEREI         E848
WHE1           E85C      WHE2           E868      WHE3           E870
WHEREO         E871      WHRO1          E885      WHRO2          E88E
WHRO3          E897      WHRO4          E89F      WHICHT         E8A8
WRAX           EA42      XORY           FDEF      XORYZ          FDF1
XORY1          FDFC      XORYRT         FE02      ZON            F25D
ZON1           F261      ZON2           F26C      ZPAGE          FC38
ZPY            FC50      ZPX            FC55

tasm: Number of errors = 0
```

AIM 65 MICROCOMPUTER MONITOR PROGRAM LISTING
Rockwell International
Document No. 29650 N36L
Rev. 1, April 1979

I used the Telemark Cross Assembler v3.1 (TASM) to re-create the source code.
See http://www.halcyon.com/squakvly/

I tried to exactly duplicate the original source but some errors may exist. The exceptions are when the original had a hexadecimal constant instead of an ASCII constant or ASCII equate (especially CR) in some immediate mode instructions; I changed them to ASCII constants or an equate.

For example, line 468 in the printed listing is:

```
0468  E185  A9 BC       STA1   LDA #$BC        ;"<" CHR WITH MSB=1 FOR DISP
```

My version is:

```
0468   E185 A9 BC       STA1   LDA #'<'+$80    ;"<" CHR WITH MSB=1 FOR DISP
```

The TASM assembler is not the same one that Rockwell used to write the code, so some assembler directives and opcode formats are different. However, the ASM file uses the same line numbering as the printed listing. That is, line 1000 in the printed listing corresponds to line 1000 in the ASM file and line 1000 in the LST file.

I could not fully read eight lines in the program listing because I was looking at a scanned copy, not the original. The rightmost characters were lost in the binding. These are the lines:

```
0149  HIST   =NAME           ;FOUR LAST ADDR + NEXT (SINGL STEP)
1796         JSR SWSTAK      ;SWAP X , Y WITH RTRN ADDR FROM S
1804         JSR SWSTAK      ;SWAP X , Y WITH RTRN ADDR FROM
2159  RDBIT  LDA TSPEED      ;ARE WE IN C7 OR 5B,5A FREQUENC
2262  OUTDP1 JMP (DILINK)    ;HERE HE COULD ECHO SOMEWHERE ELSE
3205         BNE IN02        ;CONTIN , DISP WONT ALLOW > 60 CHR
3719         LDA TYPE        ;CHCK FOR BRNCH WITH RELATIVE ADDR
3727  TRY34  LDA #04         ;CHECK FOR ABSOLUTE OR ZP,X ORZP,
```

NOTE: I have since been told that the cut-off lines above exist in the original manual.

```
+--------------------------------------------------------------------------
|  TOPIC -- AIM Computer -- AIM BASIC Language Reference Manual
+--------------------------------------------------------------------------
```

AIM 65 MICROCOMPUTER BASIC LANGUAGE REFERENCE MANUAL

Rockwell International Corporation
Document No 29650 N49
March 1979

TABLE OF CONTENTS

INTRODUCTION

Before a computer can perform any useful function, it must be "told" what to do. Unfortunately, at this time, computers are not capable of understanding English or any other "human" language. This is primarily because our languages are rich with ambiguities and implied meanings. The computer must be told precise instructions and the exact sequence of operations to be performed in order so accomplish any specific task. Therefore, in order to facilitate human communication with a computer, programming languages have been developed.

Rockwell AIM 65 8K BASIC by Microsoft is a programming language both easily understood and simple to use. It serves as an excellent "tool" for applications in areas such as business, science, and education. After only a few hours of using BASIC, you will find that you can already write programs with an ease that few other computer languages can duplicate.

Originally developed at Dartmouth University, the BASIC language has found wide acceptance in the computer field. Although it is one of the simplest computer languages to use, it is very powerful. BASIC uses a small set of common English words as its "comnmands." Designed specifically as an "interactive" language, you can give a command such as "PRINT 2 + 2," and BASIC will immediately reply with "4." It is not necessary to submit a card deck wish your program on it and then wait hours for the results. Instead, the full power of the computer is "at your fingertips."

We hope that you enjoy BASIC, and are successful in using it to solve all of your programming problems.

100 INSTALLING BASIC IN THE AIM 65

ROM INSTALLATION PROCEDURE

Before handling the BASIC ROM circuits, be sure to observe the precautions outlined in Section 1.4 of the AIM 65 User's Guide.

To install the ROMs, turn off power to the AIM 65. Inspect the pins on the two BASIC ROMs to ensure that they are straight and free of foreign material. While supporting the AIM 65 Master Module beneath the ROM socket, insert ROM number R3225 into Socket Z25, being careful to observe the device orientation. Now insert ROM number R3226 into Socket Z26. Be certain that both ROM's are completely inserted into their sockets, then turn on power to the AIM 65.

ENTERING BASIC

To enter and initialize BASIC, type 5 after the monitor prompt is displayed. AIM 65 will respond with:

```
    <5>

    MEMORY SIZE? ^
```

Type the highest address in memory that is to be allocated to the BASIC program, in decimal. End the entry by typing RETURN. BASIC will allocate memory from 530 (212 in hex) through the entered address. If BASIC is to use all available memory, type RETURN without entering an address. The highest address is 1024 (400 hex) in the 1K RAM version of AIM 65, and 4096 (1000 hex) in the 4K RAM version.

BASIC will then ask:

```
    WIDTH? ^
```

Type in the output line width of the printer (or any other output device that is being used) and end the
input with RETURN.

The entered number may vary from 1 to 255, depending on the output device. If RETURN is typed without entering a number, the output line width is set to a default value of 20, which is the column width of the AIM 65 printer.

BASIC will respond with:

```
    XXXX BYTES FREE
```

where XXXX is the number of bytes available for BASIC program, variables, matrix storage, and string space. If all available memory was allocated, BASIC will reply with:

```
    494 BYTES FREE (for 1K RAM; i.e., 1024-530)
```

or

```
    3566 BYTES FREE (for 4K RAM; i.e., 4096-530)
```

BASIC will display:

```
    ^ AIM 65 BASIC Vn.n
```

where n.n is the version number.

BASIC is now in the command entry mode as indicated by the BASIC prompt (^) in the display column 1. Subject 201 gets you started into the BASIC commands.

Read the following paragraphs first, however, so understand how to exit and reenter the BASIC and how the BASIC cursor prompt operates.

CAUTION

Entering BASIC with the 5 key causes the allocated memory to be initialized with AA (hex) in all bytes, starting with address 532. This, of course, destroys any previous BASIC programs, data in the AIM 65 Editor Text Buffer, or machine level routines that may have been stored in this portion of memory. Be sure to save any desired data or programs that may exist in this area before entering BASIC with the 5 key.

Note that text in the Text Buffer or machine level routine may co-exist in memory with BASIC by locating such text or routines in upper memory and entering the highest BASIC address with a value lower than the starting address of such text or routines.

EXITING BASIC

To escape from BASIC and return to the AIM 65 Monitor, type ESC any time the BASIC command cursor is displayed. You can also escape BASIC while a program is running, by pressing the F1 key (see Subject 301).

Pressing RESET will also cause the AIM 65 Monitor to be entered as well as performing a hardware reset of AIM 65.

REENTERING BASIC

BASIC may be reentered by typing 6 whenever the AIM 65 Monitor prompt is displayed. In this case, however, any existing BASIC program is retained in memory. AIM 65 will respond to a Key 6 entry with:

```
<6>

^6>
```

BASIC CURSOR

The BASIC cursor (^), displayed in column 1 whenever BASIC is in the command entry mode, indicates that a BASIC command can be entered. The last displayed data resulting from the previous command is retained except for column 1 to provide information continuity with the previous command or displayed output data. This is especially helpful when the printer control is turned off to preserve printer paper.

When the first character of the next command is typed, the display will blank except for the newly typed character. The cursor then advances across the display in accordance with typed characters to indicate the character input position.

The displayed cursor does not appear on the printer output, thus any data printed in column 1 will be retained.

CAUTION

The minus sign associated with any negative values that are displayed starting in column 1 will be replaced with the cursor in the BASIC command entry mode. In the case of direct commands, the minus sign will only flash before the cursor is displayed if the printer control is on or may not appear at all if the printer control is off. In order to retain the minus sign, a leading blank should be displayed before the value is displayed (see Subject 204).

PRINTER CONTROL

While in the BASIC command entry mode, the printer may be turned on or off by typing PRINT while CNTL is pressed (CNTL PRINT). The on/off state of the printer is displayed after typing PRINT.

If the printer is turned off, statements in the BASIC command entry mode and data output from

PRINT commands will be directed to the display only. If the printer is turned on, all commands and data from PRINT commands will be directed to both the printer and display. With the printer off, data can still be directed to the printer by using the PRINT) command (see Subject 305).

Similarly, INPUT statements will output data to the printer in response to the printer control state. An INPUT! statement will output data to the printer even if the printer control is off (see Subject 305).

200 GETTING STARTED WITH BASIC

201 BASIC COMMAND SET

This section is not intended to be a detailed course in BASIC programming. It will, however, serve as an excellent introduction for those of you unfamiliar with the language.

We recommend that you try each example in this section as it is presented. This will enhance your "feel" for BASIC and how it is used. Table 201-1 lists all the AIM 65 BASIC commands.

NOTE

Any time the cursor (^) is displayed in column 1 a BASIC command may be typed in. End all commands to BASIC by typing RETURN. The RETURN tells BASIC that you have finished typing the command. If you make an error, type a DEL (RUBOUT on a TTY) to eliminate the last character. Repeated use of DEL will eliminate previous characters. An @ symbol will eliminate that entire line being typed.

Table 201.1. AIM 65 BASIC Commands

```
Commands                      Input/Output
--------                      ------------
CLEAR                         DATA
CONT                          GET
FRE                           INPUT
LIST                          POS
LOAD                          PRINT
NEW                           READ
PEEK                          SPC
POKE                          TAB
RUN
SAVE
                              String Functions
                              ----------------
Program Statements            ASC
------------------            CHR$
DEF FN                        LEFT$
DIM                           LEN
END                           MID$
FOR                           RIGHT$
GOSUB                         STR$
GOTO                          VAL
IF...GOTO
IF...THEN
LET                           Arithmetic Functions
NEXT                          --------------------
ON...GOSUB                    ABS
ON...GOTO                     ATN*
REM                           COS
RESTORE                       EXP
RETURN                        INT
STOP                          LOG
USR                           RND
WAIT                          SIN
                              SGN
                              SQR
                              TAN
```

* Although the ATN function is not included in AIM 65 BASIC,
  the ATN command is recognized (see Appendix H).

202 DIRECT AND INDIRECT COMMANDS

DIRECT COMMANDS

Try typing in the following:

```
    PRINT 10-4 (end with RETURN)
```

BASIC will immediately print:

```
    6
```

The print statement you typed in was executed as soon as you hit the RETURN key. This is called a direct command. BASIC evaluated the formula after the "PRINT" and then typed out its value, in this case "6".

Now try typing in this:

```
    PRINT 1/2,3*10          ("*" means multiply, "/" means divide)
```

BASIC will print:

```
    .5        30
```

As you can see, BASIC can do division and multiplication as well as subtraction. Note how a "," (comma) was used in the print command to print two values instead of just one. The command divides a line into 10-character-wide columns. The comma causes BASIC to skip to the next 10-column field on the terminal, where the value 30 is printed.

INDIRECT COMMANDS

There is another type of command called an Indirect Command. Every Indirect command begins with a Line Number. A Line Number is any integer from 0 to 63999.

Try typing in these lines:

```
    10 PRINT 2+3
    20 PRINT 2-3
```

A sequence of Indirect Commands is called a "Program." Instead of executing indirect statements immediately, BASIC saves Indirect Commands in memory. When you type in RUN, BASIC will execute the lowest numbered indirect statement that has been typed in first, then the next higher, etc., for as many as were typed in.

In the example above, we typed in line 10 first and line 20 second. However, it makes no difference in what order you type in indirect statements. BASIC always puts them into correct numerical order according to the Line Number.

Suppose we type in

RUN

BASIC will print:

```
    5
    -1
```

203 OPERATING ON PROGRAMS AND LINES

In Subject 202, we typed a two-line program into memory. Now let's see how BASIC can be used to operate on either or both lines.

LISTING A PROGRAM

If we want a listing of the complete program currently in memory, we type in

```
    LIST
```

BASIC will reply with:

```
10 PRINT 2+3
20 PRINT 2-3
```

DELETING A LINE

Sometimes it is desirable to delete a line of a program altogether. This is accomplished by typing the Line Number of the line so be deleted, followed by a carriage return.

Type in the following:

```
10
LIST
```

BASIC will reply with:

```
20 PRINT 2-3
```

We have now deleted line 10 from the program.

REPLACING A LINE

You can replace line 10, rather than just deleting it, by typing the new line 10 and hitting RETURN.

Type in the following:

```
10 PRINT 3-3
LIST
```

BASIC will reply with:

```
10 PRINT 3-3
20 PRINT 2-3
```

It is not recommended that lines be numbered consecutively. It may become necessary to insert a new line between two existing lines. An increment of 10 between line numbers is generally sufficient.

DELETING A PROGRAM

If you want to delete the complete program currently stored in memory, type in "NEW." If you are finished running one program and are about to read in a new one, be sure to type in "NEW" first.

Type in the following:

```
NEW
```

Now type in:

```
LIST
```

204 PRINTING DATA

If is often desirable to include explanatory text along with answers that are printed out.

Type in the following:

```
PRINT "ONE HALF EQUALS", 1/2
```

BASIC will reply with:

```
ONE THIRD EOUALS
.5
```

As explained in Subject 202, including a "," in a PRINT statement causes it to space over to the next 10-column field before the value following the "," is printed.

If we use a ";" instead of a comma, the next value will be printed immediately following the previous value.

NOTE

> Numbers are always printed with at least one trailing space. Any text to be printed must always be enclosed in double quotes.

Try the following examples:

```
1.  PRINT "ONE HALF EQUALS"; 1/2
    ONE HALF EQUALS .5

2.  PRINT 1,2,3
    1         2
    3
    ...

3.  PRINT 1;2;3
    1 2 3

4.  PRINT -1;2;-3
    -1 2 -3
```

205 NUMBER FORMAT

We will digress for a moment to explain the format of numbers in BASIC. Numbers are stored internally to over nine digits of accuracy. When a number is printed, only nine digits are shown. Every number may also have an exponent (a power of ten scaling factor).

The largest number that may be presented in AIM 65 BASIC is 1.70141183*10^38, while the smallest positive number is 2.93873588*10^-39.

When a number is printed, the following rules define the format:

1. If the number is negative, a minus sign (-) is printed. If the number is positive, a space is printed.

2. If the absolute value of the number is an integer in the range 0 to 999999999, it is printed as an integer.

3. If the absolute value of the number is greater than or equal to 0.01 and less than or equal to 999999999, it is printed in fixed point notation, with no exponent.

4. If the number does not fall under categories 2 or 3, scientific notation is used.

Scientific notation is formatted as follows: SX.XXXXXXXXESTT. (Each X is some integer, 0 to 9.)

The leading "S" is the sign of the number: a space for a positive number and a "-" for for a negative one. One non-zero digit is printed before the decimal point. This it followed by the decimal point and then the other eight digits of the mantissa. An "E" is then printed (for exponent), followed by the sign (S) of the exponent; then the two digits (TT) of the exponent itself. Leading zeroes are never printed; i.e., the digit before the decimal is never zero. Trailing zeroes are never printed. If there is only one digit to print after all trailing zeroes are suppressed, no decimal point is printed. The exponent sign will be "+" for positive and "-" for negative. Two digits of the exponent are always printed; that is, zeroes are not suppressed in the exponent field. The value of any number expressed thus is the number so the left of the "E" times 10 raised to the power of the number to the right of the "E".

Regardless of what format is used, a space is always printed following a number. BASIC checks to see if the entire number will fit on the current line. If it cannot, a carriage return/line feed is executed before printing the number.

Following are examples of various numbers and the output format in which BASIC will output them:

```
    NUMBER              OUTPUT FORMAT
    -------------       -------------
```

```
+1                  1
-1                 -1
 6523               6523
-23.460            -23.46
 1E20               1E+20
-12.3456E-7        -1.23456E-06
 1.234567E-10       1.23457E-10
 1000000000         1E+09
 999999999          999999999
.1                 .1
.01                .01
.000123             1.23 E-04
```

A number input from the keyboard or a numeric constant used in a BASIC program may have as many digits as desired, up to the maximum length of a line (72 characters) or maximum numeric value. However, only the first 10 digits are significant, and tenth digit is rounded up.

```
PRINT 1.23456789876543210
1.2345679
```

206 VARIABLES

ASSIGNING VARIABLES WITH AN INPUT STATEMENT

Following is an example of a program that reads a value from the keyboard and uses that value to calculate and print a result:

```
10 INPUT R
20 PRINT 3.14159*R*R
RUN
?10
314.159
```

Here's what's happening: When BASIC encounters the input statement, it outputs a question mark (?) on the display and then waits for you to type in a number. When you do (in the above example, 10 was typed), execution continues with the next statement in the program after the variable (R) has been set (in this case to 10). In the above example, line 20 would now be executed. When the formula after the PRINT statement is evaluated, the value 10 is substituted for the variable R each time R appears in the formula. Therefore, the formula becomes 3.14159*10*10, or 314.159.

If we wanted so calculate the area of various circles, we could rerun the program for each successive circle. But, there's an easier way to do it simply by adding another line to the program, as follows:

```
30 GOTO 10
RUN
?10
314.159
?3
28.27431
?4.7
69.3977231
?
```

By putting a "GOTO" statement on the end of our program, we have caused it to go back to line 10 after it prints each answer for the successive circles. This could have gone on indefinitely, but we decided to stop after calculating the area for three circles. This was accomplished by typing a carriage return to the input statement (thus a blank line).

VARIABLE NAMES

The letter "R" in the program above is a "variable." A variable name can be any alphabetic character and may be followed by any alphanumeric character (letters A to Z, numbers 0 to 9).

Any alphanumeric characters after the first two are ignored.

Here are some examples of legal and illegal variable names:

```
Legal     Illegal

A         % (first character must be alphabetic)
```

```
Z1        ZIABCD (variable name too long)

TP        TO (variable names cannot be reserved words)
PSTG$     RGOTO (variable names cannot contain reserved words)
COUNT
```

ASSIGNING VARIABLES WITH A LET OR ASSIGNMENT STATEMENT

Besides having values assigned to variables with an input statement, you can also set the value of a variable with a LET or assignment statement.

Try the following examples:

```
A=5

PRINT A, A*2
5         10

LET Z=7

PRINT Z, Z-A
7         2
```

As you will notice from the examples, the "LET" is optional in an assignment statement.

BASIC "remembers" the values that have been assigned to variables using this type of statement. This "remembering" process uses space in the memory to store the data.

The values of variables are discarded (and the space in memory used to store them is released) when one of four conditions occur:

* A new line is typed into the program or an old line is deleted
* A CLEAR command is typed in
* A RUN command is typed in
* NEW is typed in

Another important fact is that if a variable is encountered in a formula before it is assigned a value, it is automatically assigned the value zero. Zero is then substituted as the value of the variable in the particular formula. Try the example below:

```
PRINT Q;Q+2;Q*2
0 2 0
```

RESERVED WORDS

The words used as BASIC statements are "reserved" for this specific purpose. You cannot use these words as variable names or inside of any variable name. For instance, "FEND" would be illegal because "END" is a reserved word.

Table 206-1 is a list of the reserved words in BASIC.

Table 206-1. AIM 65 BASIC Reserved Words

ABS	FN	LIST	PRINT	SPC
AND	FOR	LOAD	POS	SQR
ASC	FRE	LOG	READ	STEP
ATN	GET	MID$	REM	STOP
CHR$	GOSUB	NEW	RESTORE	STR$
CLEAR	GOTO	NEXT	RETURN	TAB
CONT	IF	NOT	RIGHT$	TAN
COS	INPUT	NULL	RND	THEN
DATA	INT	ON	RUN	TO
DEF	LEFT$	OR	SAVE	USR
DIM	LEN	PEEK	SGN	VAL
END	LET	POKE	SIN	WAIT
EXP				

REMARKS

The REM (short for "remark") statement is used to insert comments or notes into a program. When BASIC encounters a REM statement, the rest of the line is ignored.

This serves mainly as an aid for the programmer and serves no useful function as far as the operation of the program in solving a particular problem.

207 RELATIONAL TESTS

Suppose we wanted to write a program to check whether a number is zero. With the statements we've gone over so far, this could not be done. What is needed is a statement which can be used to conditionally branch to another statement. The "IF-THEN" statement does just that.

Type in the following program: (remember, type NEW first)

```
10 INPUT B
20 IF B=0 THEN 55
30 PRINT "NON-ZERO"
40 GOTO 10
50 PRINT "ZERO"
60 GOTO 10
```

When this program is typed and run, it will ask for a value for B. Type in any value you wish. The AIM 65 will then come to the "IF" statement. Between the "IF" and the "THEN" portion of the statement there are two expressions separated by a "relation."

A relation is one of the following six symbols:

RELATION	MEANING
=	EQUAL TO
>	GREATER THAN
<	LESS THAN
<>	NOT EQUAL TO
<= or =<	LESS THAN OR EQUAL TO
=> or >=	GREATER THAN OR EQUAL TO

The IF statement is either true or false, depending upon whether the two expressions satisfy the relation. For example, in the program we just did, if 0 was typed in for B the IF statement would be true because 0=0. In this case, since the number after the THEN is 50, execution of the program would continue at line 50. Therefore, "ZERO" would be printed and then the program would jump back to line 10 (because of the GOTO statement in line 60).

Suppose a 1 was typed in for B. Since 1=0 is false, the IF statement would be false and the program would continue execution with the next line. Therefore, "NON-ZERO" would be printed and the GOTO in line 40 would send the program back to line 10.

A PROGRAM USING RELATIONS

Now try the following program for comparing two numbers:

```
10 INPUT A,B
20 IF A<=B THEN 50
30 PRINT "A IS BIGGER"
40 GOTO 10
50 IF A<B THEN 80
60 PRINT "THEY ARE THE SAME"
70 GOTO 10
80 PRINT "B IS BIGGER"
90 GOTO 10
```

When this program is run, line 10 will input two numbers from the keyboard. At line 20, if A is greater than B, A<=B will be false. This will cause the next statement to be executed, printing "A IS BIGGER" and then line 40 sends the computer back to line 10 to begin again.

At line 20, if A has the same value as B, A<=B is true so we go to line 50. At line 50, since A has the same value as B, A<B is false; therefore, we go to the following statement and print "THEY ARE THE SAME." Then line 70 sends us back to the beginning again.

At line 20, if A is smaller than B, A<=B is true so we goto line 50. At line 50, A<B will be true so we then go to line 80. "B IS BIGGER" is then printed and again we go back to the beginning.

Try running the last two programs several times. It may be easier to understand if you try writing your own program at this time using the IF-THEN statement. Actually trying programs of your own is the quickest and easiest way to understand how BASIC works. Remember, to stop these programs just give a RETURN to the input statement.

208 LOOPING

One advantage of computers is their ability to perform repetitive tasks. Let's take a closer look and see how this works.

A SQUARE ROOT PROGRAM

Suppose we want a table of square roots from 1 to 9. The BASIC function for square root is "SQR"; the form being SORIX), X being the number whose square root is to be calculated. We could write the program as follows:

```
10 PRINT 1,SQR(1)
20 PRINT 2,SQR(2)
30 PRINT 3,SQR(3)
40 PRINT 4,SQR(4)
50 PRINT 5,SQR(5)
60 PRINT 6,SQR(6)
70 PRINT 7,SQR(7)
80 PRINT 8,SQR(8)
90 PRINT 9,SQR(9)
```

AN IMPROVED SQUARE ROOT PROGRAM

This program will do the job, but is terribly inefficient. We can improve the program considerably by using the IF statement just introduced as follows:

```
10 N=1
20 PRINT N;SQR(N)
3D N=N+1
40 IF N<=9 THEN 20
```

When this program is run, its output will look exactly like that of the 9 statement program above it. Let's look at how it works:

At line 10 we have a LET statement which sets the value of the variable N equal to 1. At line 20 we print N and the square root of N using its current value. It thus becomes 20 PRINT 1;SQR(1), and this calculation is printed out.

At line 30 we use what will appear at first to be a rather unusual LET statement. Mathematically, the statement N=N+1 is nonsense. However, the important thing to remember is that in a LET statement, the symbol "=" does not signify equality. In this case, "=" means "to be replaced with." All the statement does is to take the current value of N and add 1 to it. Thus, after the first time through line 30, N becomes 2.

At line 40, since N now equals 2, N<=9 is true so the THEN portion branches us back to line 20, with N now at a value of 2.

The overall result is that lines 20 through 40 are repeated, each time adding 1 to the value of N. When N finally equals 9 at line 20, the next line will increment it to 11. This results in a false statement at line 40, and since there are no further statements to the program it stops.

BASIC STATEMENTS FOR LOOPING

This technique is referred to as "looping" or "iteration." Since it is used quite extensively in programming, there are special BASIC statements for using it. We can show these with the following program:

```
10 FOR N=1 TO 9
20 PRINT N;SQR(N)
30 NEXT N
```

The output of the program listed above will be exactly the same as the previous two programs.

At line 10, N is set to equal 1. Line 20 causes the value of N and the square root of N so be printed. At line 30 we sees new type of statement. The "NEXT N" statement causes one to be added to N, and then if N<=9 we go back to the statement following the "FOR" statement. The overall operation then is the same as with the previous program.

Notice that the variable following the "FOR" is exactly the same as the variable after the "NEXT." There is nothing special about the N in this case. Any variable could be used, as long as it is the same in both the "FOR" and the "NEXT" statements. For instance, "Z1" could be substituted everywhere there is an "N" in the above program and it would function exactly the same.

ANOTHER SQUARE ROOT PROGRAM

Suppose we want to print a table of square roots of each even number from 10 to 20. The following program performs this task:

```
10 N=10
20 PRINT N;SQR(N)
30 N=N+2
40 IF N<=20 THEN 20
```

Note the similarity between this program and our "improved" square root program. This program can also be written using the "FOR" loop just introduced.

```
10 FOR N=10 TO 20 STEP 2
20 PRINT N;SQR(N)
30 NEXT N
```

Notice that the only major difference between this program and the previous one using "FOR" loops is the addition of the "STEP 2" clause.

This tells BASIC to add 2 to N each time, instead of 1 as in the previous program. If no "STEP" is given in a "FOR" statement, BASIC assumes that 1 is to be added each time. The "STEP" can be followed by any expression.

A COUNT-BACKWARD PROGRAM

Suppose we wanted to count backward from 10 to 1. A program for doing this would be as follows:

```
10 I=10
20 PRINT I
30 I=I-1
40 IF I>=1 THEN 20
```

Notice that we are now checking to see that I is greater than or equal to the final value. The reason is that we are now counting by a negative number. In the previous examples it was the opposite, so we were checking for a variable less than or equal to the final value.

SOME OTHER LOOPING OPERATIONS

The "STEP" statement previously shown can also be used with negative numbers to accomplish this same result. This can be done using the same format as in the other program:

```
10 FOR I=10 TO 1 STEP -1
20 PRINT I
30 NEXT I
```

"FOR" loops can also be "nested." For example:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT J
50 NEXT I
```

Notice that "NEXT J" precedes "NEXT I." This is because the J-Ioop is inside the I-loop. The following program is incorrect; run it and see what happens:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT I
50 NEXT J
```

It does not work because when the "NEXT I" is encountered, all knowledge of the J-loop is lost. This happens because the J-loop is "inside" the I-loop.

209 MATRIX OPERATIONS

It is often convenient to be able to select any element in a table of numbers. BASIC allows this to be done through the use of matrices.

A matrix is a table of numbers. The name of this table (the matrix name) is any legal variable name, "A" for example. The matrix name "A" is distinct and separate from the simple variable "A," and you could use both in the same program.

To select an element of the table, we subscript "A": that is, to select the I'th element, we enclose I in parentheses "(I)" and then follow "A" by this subscript. Therefore, "A(I)" is the I'th element in the matrix "A."

"A(1)" is only one element of matrix A, and BASIC must be told how much space so allocate for the entire matrix. This is done with a "DIM" statement, using the format "DIM A(15)." In this case, we have reserved space for the matrix index "I" to go from 0 to 15. Matrix subscripts always start as 0; therefore, in the above example, we have allowed for 16 numbers in matrix A.

If "A(1)" is used in a program before is has been dimensioned, BASIC reserves space for 11 elements (0 through 10).

A SORT PROGRAM

As an example of how matrices are used, try the following program so sort a list of 8 numbers, in which you pick the numbers to be sorted:

```
10 DIM A(8)
20 FOR I=1 TO 8
30 INPUT A(I)
50 NEXT I
70 F=0
80 FOR I=1 TO 7
90 IF A(I)<=A(I+1) THEN 140
100 T=A(I)
110 A(I)=A(I+1)
120 A(I+1)=T
130 F=1
140 NEXT I
150 IF F=1 THEN 70
160 FOR I=1 TO 8
170 PRINT A(I)
180 NEXT I
```

When line 10 is executed, BASIC sets aside space for 9 numeric values, A(0) through A(8). Lines 20 through 50 get the unsorted list from the user. The sorting itself is done by going through the list of numbers and switching any two that are not in order. "F" is used to indicate if any switches were made; if any were made, line 150 tells BASIC to go back and check some more.

If we did not switch any numbers, or after they are all in order, lines 160 through 180 will print out the sorted list. Note that a subscript can be any expression.

210 SUBROUTINES

If you have a program that performs the same action in several different places, you could duplicate the same statements for the action in each place within the program.

The "GOSUB" and "RETURN" statements can be used to avoid this duplication. When a "GOSUB" is encountered, BASIC branches to the line whose number follows the "GOSUB." However, BASIC remembers where it was in the program before it branches. When the "RETURN" statement is encountered, BASIC goes back to the first statement following the last "GOSUB" that was executed. Observe the following program:

```
10 PRINT "WHAT IS THE NUMBER";
30 GOSUB 100
40 T=N
50 PRINT "SECOND NUMBER";
70 GOSUB 100
80 PRINT "THE SUM IS"; T+N
90 STOP
```

```
100 INPUT N
110 IF N=INT(N) THEN 140
120 PRINT "MUST BE INTEGER."
130 GOTO 100
140 RETURN
```

This program asks for two numbers (which must be integers), and then prints their sum. The subroutine in this program is lines 100 to 140. The subroutine asks for a number, and if it is not an integer, asks for a new number. It will continue to ask until an integer value is typed in.

The main program prints "WHAT IS THE NUMBER," and then calls the subroutine so get the value of the number into N. When the subroutine returns (to line 40), the value input is saved in the variable T. This is done so that when the subroutine is called a second time, the value of the first number will not be lost.

"SECOND NUMBER" is then printed, and the second value is entered when the subroutine is again called.

When the subroutine returns the second time, "THE SUM IS" is printed, followed by the sum. T contains the value of the first number that was entered and N contains the value of the second number.

STOPPING A PROGRAM

The next statement in the program is a "STOP" statement. This causes the program to stop execution at line 90. If the "STOP" statement was excluded from the program, we would "fall into" the subroutine at line 100. This is undesirable because we would be asked to input another number. If we did, the subroutine would try to return; and since there was no "GOSUB" which called the subroutine, an RG error would occur. Each "GOSUB" executed in a program should have a matching "RETURN" executed later. The opposite also applies: a "RETURN" should be encountered only if it is part of a subroutine which has been called by a "GOSUB."

Either "STOP" or "END" can be used to separate a program from its subroutines. "STOP" will print a message saying at what line the "STOP" was encountered.

211 ENTERING DATA

Suppose you had to enter numbers to your program that did not change each time the program was run, but you would like it to be easy to change them if necessary. BASIC contains special statements, "READ" and "DATA," for this purpose.

Consider the following program:

```
 10 PRINT "GUESS A NUMBER";
 20 INPUT G
 30 READ D
 40 IF D = -999999 THEN 90
 50 IF D<>G THEN 30
 60 PRINT "YOU ARE CORRECT"
 70 END
 90 PRINT "BAD GUESS, TRY AGAIN."
 95 RESTORE
100 GOTO 10
110 DATA 1,393,-39,28,391,-8,0,3.14,90
120 DATA 89,5,10,15,-34,-999999
```

When the "READ" statement is encountered, the effect is the same as an INPUT statement. But, instead of getting a number from the keyboard, a number is read from the "DATA" statements.

The first time a number is needed for a READ, the first number in the first DATA statement is read. The second time one is needed, the second number in the first DATA statement is read. When the all numbers of the first DATA statement have been read in this manner, the second DATA statement will be used. DATA is always read sequentially in this manner, and there may be any number of DATA statements in your program.

The purpose of this program is to play a little game in which you try to guess one of the numbers contained in the DATA statements. For each guess that is typed in, we read through all of the numbers in the DATA statements until we find one that matches the guess.

If more values are read than there are numbers in the DATA statements, an out of data (OD) error occurs. That is why in line 40 we check to see if -999999 was read. This is not one of the numbers to be matched, but is used as a flag to indicate that all of the data (possible correct guesses) has been read. Therefore, if -999999 was read, we know that the guess was incorrect.

Before going back to line 10 for another guess, we need to make the READ's begin with the first piece of data again. This is the function of the "RESTORE." After the RESTORE is encountered, the next piece of data read will be the first number in the first DATA statement again.

DATA statements may be placed anywhere within the program. Only READ statements make use of the DATA statements in a program, and any other time they are encountered during program execution they will be ignored.

212 STRINGS

A list of characters is referred to as a "String." Rockwell, R6500, and THIS IS A TEST are all strings. Like numeric variables, string variables can be assigned specific values. String variables are
distinguished from numeric variables by a "$" after the variable name.

For example, try the following:

```
A$="ROCKWELL R6500"
PRINT A$
ROCKWELL R6500
```

In this example, we set the string variable A$ to the string value "ROCKWELL R6500." Note that we also enclosed the character string so be assigned to A$ in quotes.

LEN FUNCTION

Now that we have set A$ to a string value, we can find out what the length of this value is (the number of characters it contains). We do this as follows:

```
PRINT LEN(A$),LEN("MICROCOMPUTER")
14        13
```

The "LEN" function returns an integer equal to the number of characters in a string.

A string expression may contain from 0 to 255 characters. A string containing 0 characters is called the "null" string. Before a string variable is set to a value in the program, it is initialized to the null
string. Printing a null string on the terminal will cause no characters to be printed, and the printer or cursor will not be advanced to the next column. Try the following:

```
PRINT LEN(Q$);Q$;3
0 3
```

Another way to create the null string is: Q$=""

Setting a string variable to the null string can be used to free up the string space used by a non-null string variable.

LEFT$ FUNCTION

It is often desirable to access parts of a string and manipulate them. Now that we have set A$ to "ROCKWELL R6500," we might want to print out only the first eight characters of A$. We would do so like this:

```
PRINT LEFT$(A$,8)
ROCKWELL
```

"LEFT$" is a string function which returns a string composed of the leftmost N characters of its string argument. Here is another example:

```
FOR N=1 TO LEN(A$):PRINT LEFT$(A$,N):NEXT N
R
RO
ROC
ROCK
```

```
    ROCKW
    ROCKWE
    ROCKWEL
    ROCKWELL
    ROCKWELL R
    ROCKWELL R6
    ROCKWELL R65
    ROCKWELL R650
    ROCKWELL R6500
```

Since A$ has 14 characters this loop will be executed with N=1,2,3,...,13,14. The first time through only the first character will be printed, the second time the first two characters will be printed, etc.

RIGHT$ FUNCTION

Another string function, called "RIGHT$," returns the right N characters from a string expression. Try substituting "RIGHT$" for "LEFT$" in the previous example and see what happens.

MID$ FUNCTION

There is also a string function which allows us to take characters from the middle of a string. Try the following:

```
    FOR N=1 TO LEN(A$):PRINT MID$(A$,N):NEXT N
    ROCKWELL R6500
    OCKWELL R6500
    CKWELL R6500
    KWELL R6500
    WELL R6500
    ELL R6500
    LL R6500
    L R6500
     R6500
    R6500
    6500
    500
    00
    0
```

"MID$" returns a string starting at the Nth position of A$ so the end (last character) of A$. The first position of the string is position 1 and the last possible position of a string is position 255.

Very often it is desirable to extract only the Nth character from a string. This can be done by calling MID$ with three arguments. The third argument specifies the number of characters to return.

For example:

```
    FOR N=1 TO LEN(A$):PRINT MID$(A$,N,1),MID$(A$,N,2):NEXT N
    R         RO
    O         OC
    C         CK
    K         KW
    W         WE
    E         EL
    L         LL
    L         L
               R
    R         R6
    6         65
    5         50
    0         00
    0         0
```

CONCATENATION-JOINING STRINGS

Strings may also be concatenated (put or joined together) through the use of the "+" operator. Try the following:

```
B$="BASIC FOR"+" "+A$
PRINT B$
BASIC FOR ROCKWELL R6500
```

Concatenation is especially useful if you wish to take a string apart and then put it back together with slight modifications. For instance:

```
C$=LEFT$(B$,9)+"-"+MID$(B$,11,8)+"-"+RIGHT$(B$,5)
PRINT C$
BASIC FOR-ROCKWELL-R6500
```

VAL AND STRS FUNCTIONS

Sometimes it is desirable to convert a number to its string representation, and vice-versa. "VAL" and "STR$" perform these functions.

Try the following:

```
STRING$="567.8"
PRINT VAL(STRING$)
567.8
STRING$=STR$(3.1415)
PRINT STRINGS$,LEFT$(STRING$,5)
3.1415    3.14
```

"STR$" can be used to perform formatted I/O on numbers. You can convert a number to a string and then use LEFT$, RIGHT$, MID$ and concatenation to reformat the number as desired.

"STR$" can also be used to conveniently find out how many print columns a number will take. For example:

```
PRINT LEN(STR$(3.157))
6
```

If you have an application in which a user is typing in a question such as "WHAT IS THE VOLUME OF A CYLINDER OF RADIUS 5.36 FEET, OF HEIGHT 5.1 FEET?" you can use "VAL" to extract the numeric values 5.36 and 5.1 from the question.

CHR$ FUNCTION

CHR$ is a string function which returns a one character string which contains the alphanumeric equivalent of the argument, according so the conversion table in Appendix E. ASC takes the first character of a string and converts it to its ASCII decimal value.

One of the most common uses of CHR$ is to send a special character to a terminal.

```
100 DIM A$(15)
110 FOR I=1 TO 15
120 READ A$(I)
130 NEXT I
120 F=0:I=1
130 IF A$(I)<=A$(I+1) THEN 180
140 T$=A$(I+1)
150 A$(I+1)=A$(I)
160 A$(I)=T$
170 F=1
180 I=I+1
185 IF I<15 THEN 130
190 IF F THEN 120
200 FOR I=1 TO 15
202 PRINT A$(I)
204 NEXT I
220 DATA AIM 65,DOG
230 DATA CAT,R6500
240 DATA ROCKWELL,RANDOM
250 DATA SATURDAY,"***ANSWER***"
260 DATA MICRO,FOO
270 DATA COMPUTER,MED
280 DATA NEWPORT BE-ACH,DALLAS,ANAHEIM
```

ADDITIONAL STRING CONSIDERATIONS

1. A string may contain from 0 to 255 characters. All string variable names end in a dollar sign ($); for example, A$, B9$, K$, HELLO$.

2. String matrices may be dimensioned exactly like numeric matrices. For instance, DIM A$(10,10) creates a string matrix of 121 elements, eleven rows by elevon columns (rows 0 to 10 and columns 0 to 10). Each string matrix element is a complete string, which can be up to 255 characters in length.

NAME	EXAMPLE	PURPOSE/USE
DIM	25 DIM A$(10,10)	Allocates space for a pointer and length for each element of a string matrix. No string space is allocated.
LET	27 LET A$="FOO"+V$	Assigns the value of a string expression to a string variable. LET is optional.
= > < <= or =< >= or => <>		String comparison operators. Comparison is made on the basis of ASCII codes, a character at a time until a difference is found. If during the comparison of two strings, the end of one is reached, the shorter string is considered smaller. Note that "A " is greater than "A" since trailing spaces are significant.
+	30 LET Z$=R$+Q$	String concatenation. The resulting string must be less than 256 characters in length or an LS error will occur.
INPUT	40 INPUT X$	Reads a string from the keyboard. String does not have to be quoted; but if not, leading blanks will be ignored and the string will be terminated on a "," or ":" character.
READ	50 READ X$	Reads a string from DATA statements within the program. Strings do not have to be quoted; but if they are not, they are terminated on a "," or ":" character and leading spaces are ignored. See DATA for the format of string data.
PRINT	60 PRINT X$ 70 PRINT "FOO"+A$	Prints the string expression on the display/printer.

300 STATEMENT DEFINITIONS

301 SPECIAL CHARACTERS

CHARACTER	USE
@	Erases current line being typed, and types a carriage return/line feed.
DEL	Erases last character typed. If no more characters are left on the line, types a carriage return/line feed.
RETURN	A RETURN must end every line typed in. Returns cursor to the first position (leftmost) on line, and prints the line if the printer is on.
F1	Interrupts execution of a program or a list command. F1 has effect when a statement finishes execution, or in the case of interrupting a LIST command, when a complete line has finished printing. In both cases a return is made to BASIC's

```
                command level and OK is typed.

                Prints "BREAK IN LINE XXXX," where XXXX is the line
                number of the next statement to be executed.

                There is no F1 key on a TTY.  However, when TTY is being
                used, the AIM 65's F1 key is operational and can be used.

    : (colon)   A colon is used to separate statements on a line.  Colons may
                be used in direct and indirect statements.  The only limit on
                the number of statements per line is the line length.  It is not
                possible to GOTO or GOSUB to the middle of a line.

    ?           Question marks are equivalent to PRINT.  For instance, ? 2+2
                is equivalent to PRINT 2+2.  Question marks can also be used
                in indirect statements.  10 ? X, when listed, will be typed as
                10 PRINT X.

    $           A dollar sign ($) suffix on a variable name establishes the
                variable as a character string.

    %           A percent sign (%) suffix on a variable name establishes the
                variable as an integer

    !           An exclamation sign (!) suffix on an INPUT, PRINT, or ?
                command causes the input or output to be printed even
                though the printer is turned off.

    ESC         Returns control to the Monitor.

    CNTL PRINT  Turns the AIM 65 printer on if it is off, and off if it is on.

302  OPERATORS

    SYMBOL          SAMPLE STATEMENT        PURPOSE/USE
    ------          ----------------        -----------

    =               A=100                   Assigns a value to a variable

                    LET Z=2.5               The LET is optional

    -               B=-A                    Negation.  Note that 0-A is subtraction,
                                            while -A is negation.

    ^ (F3 key)      130 PRINT X^3           Exponentiation (equal to X*X*X in
                                            in the sample statement)

                                            0^0=1    0 to any other power = 0

                                            A^B, with A negative and B not an
                                            integer gives an FC error.

    *               140 X=R*(B*D)           Multiplication.

    /               150 PRINT X/1.3         Division.

    +               160 Z=R+T+Q             Addition

    -               170 J=100-I             Subtraction

RULES FOR EVALUATING EXPRESSIONS:

1)  Operations of higher precedence are performed before operations of lower precedence.
    This means the multiplication and divisions are performed before additions and subtractions.
    As an example, 2+10/5 equals 4, not 2.4.  When operations of equal precedence are found
    in a formula, the left hand one is executed first:  6-3+5=8, not -2.

2)  The order in which operations are performed can always be specified explicitly through the
    use of parentheses.  For instance, to add 5 to 3 and then divided that by 4, we would use
    (5+3)/4, which equals 2.  If instead we had used 5+3/4, we would get 5.75 as a result
```

(5 plus 3/4).

The precedence of operators used in evaluating expressions is as follows, in order beginning with the highest precedence :

NOTE

Operators listed on the same line have the same precedence.

1) Expressions in parentheses are always evaluated first

2) ^ (F3 KEY) — ExponentiatiOn

3) NEGATION — -X where X may be a formula

4) * and / — Multiplication and Division

5) + and - — Addition and Subtraction

6) RELATIONAL OPERATORS: (equal precedence for all six)

=	Equal
<>	Not Equal
<	Less Than
>	Greater Than
=< or <=	Less Than or Equal
=> or >=	Greater Than or Equal

(These three below are Logical Operators)

7) NOT — Logical and bitwise "NOT" like negation, not takes only the formula to its right as an argument

8) AND — Logical and bitwise "AND"

9) OR — Logical and bitwise "OR"

A relational expression can be used as part of any expression.

Relational Operator expressions will always have a value of True (-1) or a value of False (0). Therefore, (5=4)=0, (5=5)=-1, (4>5)=0, (4<5)=-1, etc.

The THEN clause of an IF statement is executed whenever the formula after the IF is not equal to 0. That is to say, IF X THEN ... is equivalent to IF X<>0 THEN ....

SYMBOL	SAMPLE STATEMENT	PURPOSE/USE
-	10 IF A=15 THEN 40	Expression Equals Expression
<>	70 IF A<>0 THEN 5	Expression Does Not Equal Expression
>	30 IF B>100 THEN 8	Expression Greater Than Expression
<	160 IF B<2 THEN 10	Expression Less Than Expression
<=,=<	180 IF 100<=B+C THEN 10	Expression Less Than or Equal To Expression
>=,=>	190 IF Q=>R THEN 50	Expression Greater Than Or Equal To Expression
AND	2 IF A<5 AND B<2 THEN 7	If expression 1 (A<5) AND expression 2 (B<2) are both true, then branch to line 7
OR	IF A<1 OR B<2 THEN 2	If either expression 1 (A<1) OR expression 2 (B<2) is true, then branch to line 2

NOT	IF NOT Q3 THEN 4	If expression "NOT Q3" is true (Because Q3 is false), then branch to line 4 Note: NOT -1=0 (NOT true=false)

AND, OR, and NOT can be used for bit manipulation, and for performing boolean operations.

These three operators convert their arguments to sixteen bit, signed two's-complement integers in the range -32768 to +32767. They then perform the specified logical operation on them and return a result within the same range. If the arguments are not in this range, an "FC" error results.

The operations are performed in bitwise fashion, this means that each bit of the result is obtained by examining the bit in the same position for each argument.

The following truth table shows the logical relationship between bits:

OPERATOR	ARGUMENT 1	ARGUMENT 2	RESULT
AND	1	1	1
	0	1	0
	1	0	0
	0	0	0
OR	1	1	1
	1	0	1
	0	1	1
	0	0	0
NOT	1	-	0
	0	-	1

EXAMPLES: (In all of the examples below, leading zeroes on binary numbers are not shown.)

63 AND 16=16	Since 63 equals binary 111111 and 16 equals binary 10000, the result of the AND is binary 10000 or 16.
15 AND 14=14	15 equals binary 1111 and 14 equals binary 1110, so 15 AND 14 equals binary 1110 or 14.
-1 AND 8=8	-1 equals binary 1111111111111111 and 8 equals binary 1000, so the result is binary 1000 or 8 decimal.
4 AND 2=0	4 equals binary 100 and 2 equals binary 10, so the result is binary 0 because nons of the bits in either argument match to give a 1 bit in the result.
4 OR 2=6	Binary 100 OR'd with binary 10 equals binary 110, or 6 decimal.
10 OR 10=10	Binary 1010 OR'd with binary 1010 equals binary 1010, or 10 decimal.
-1 OR -2=-1	Binary 1111111111111111 (-1) OR'd with binary 1111111111111110 (-2) equals binary 1111111111111111, or -1.
NOT 0=-1	The bit complement of binary 0 to 16 places is sixteen ones (1111111111111111) or -1. Also NOT -1=0.
NOT X	NOT X is equal to -(X+1). This is because to form the sixteen bit two's complement of the number, you take the bit (one's) complement and add one.
NOT 1=-2	The sixteen bit complement of 1 is 1111111111111110, which is equal to -(1+1) or -2.

A typical use of the bitwise operators is to test bits set in the computer's locations which reflect the
state of some external device.

Bit position 7 is the most significant bit of a byte, while position 0 is the least significant.

For instance, suppose bit 1 of location 40963 is 0 when the door to Room X is closed, and 1 if the door is open. The following program will print "Intruder Alert" if the door is opened:

10 IF NOT (PEEK(40963) AND 2) THEN 10	This line will execute over and over until bit 1 (masked or selected by the 2) becomes a 1. When that happens, we go to line 20.
20 PRINT "INTRUDER ALERT"	Line 20 will output "INTRUDER ALERT."

However, we can replace statement 10 with a "WAIT" statement, which has exactly the same effect.

10 WAIT 40963,2	This line delays the execution of the next statement in the program until bit 1 of location A003 becomes 1. The WAIT is much faster than the equivalent IF statement and also takes less bytes of program storage.

The following is another useful way of using relational operators:

125 A=-(B>C)*B-(B<=C)*C	This statement will set the variable A to MAX(B,C) = the larger of the two variables B and C.

303 COMMANDS

A BASIC command may be entered when the cursor is displayed. This is called the "Command Level." Commands may be used as program statements. Certain commands, such as LIST, NEW, and LOAD will terminate program execution when they finish. Each command may require one or more arguments in addition to the command statement, as defined in the syntax/function description. An argument without parenthesis is required to be entered without parenthesis. Arguments contained within parenthesis are required to be entered with the shown parenthesis. Arguments within brackets are optional. Optional arguments, if included, must be entered with or without accompanying parenthesis, however shown.

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
CLEAR	CLEAR Clears all program variables, resets "FOR" and "GOSUB" state, and restores data.	CLEAR

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
CONT	CONT Continues program execution after the F1 key or a STOP or INPUT statement terminates execution. You cannot continue after any error, after modifying your program, or before your program has been run. One of the main purposes of CONT is debugging. Suppose at some point after running your program, nothing is printed. This may be because your program is performing some time consuming calculation, but it may be because you have fallen into an "infinite loop." An infinite loop is a series of BASIC statements from which there is no excape. BASIC will keep executing the series of statements over and over; until you intervene or until power to the AIM 65 is turned off. If you suspect your program is in an infinite loop, press F1 until the BREAK message is displayed. The line number of the statement BASIC was executing will be displayed. After BASIC has displayed the cursor, you can use PRINT to type out some of the values of your variables. After examining these	CONT

values you may become satisfied that your program is functioning correctly. You should then type in CONT to Continue executing your program where it left off, or type a direct GOTO statement to resume execution of the program at a different line. You could also use assignment statements to set some of your variables to different values. Remember, if you interrupt a program with the F1 key and expect to continue it later, you must not get any errors or type in any new program lines. If you do, you won't be able to continue and will get a "CN" (continue not) error. It is impossible to continue a direct command. CONT always resumes execution at the next statement to be executed in your program when F1 was typed.

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
FRE	FRE (expression) Gives the number of memory bytes currently unused by BASIC. A dummy operand--0 or 1--must be used.	270 PRINT FRE(0)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LIST	LIST [[start line] [-[end line]]] Lists current program optionally starting at specified line. List can be interrupted with the F1 key. (BASIC will finish listing the current line.)	
	Lists entire program	LIST
	Lists just line 100.	LIST 100
	Lists lines 100 to 1000.	LIST 100-1000
	Lists from current line to line 1000.	LIST -1000
	Lists from line 100 to end of program.	LIST 100-

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LOAD	LOAD Loads a BASIC program from the cassette tape. When done, the LOAD will display the cursor. See Appendix G for more information.	LOAD

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
NEW	NEW Deletes current program and all variables.	NEW

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
PEEK	PEEK (address) The PEEK function returns the contents of memory address I in decimal. The value returned will be =>0 and <=255. If I is >65535 or <0, an FC error will occur. An attempt to read a non-existent memory address will return an unknown value.	356 PRINT PEEK(I)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
POKE	POKE location, byte The POKE statement stores the byte specified by its second argument (J) into the location given by its first argument (I). The byte to be stored must be =>0 and <=255, or an FC error will occur. The address (I) must be =>0 and <=65535, or an FC error result. Caution: Careless use of the POKE statement may cause your program, BASIC, or the Monitor functions to operate incorrectly, to hang up, and/or cause loss of your program. Note that Pages 0 and 1 in memory are reserved for use by BASIC and should not be used for user program variable storage. A POKE to a non-existent memory location is harmless. One of the main uses of POKE is to pass arguments to machine language subroutines. (See Appendix F.) You could also use PEEK and POKE to write a memory diagnostic or an assembler in BASIC.	357 POKE I,J

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
RUN	RUN line number Starts execution of the program currently in memory at the specified line number. RUN deletes all variables [does a CLEAR) and restores DATA. If you have stopped your program and wish to continue execution at some point in the program, use a direct GOTO statement to start execution of your program at the desired line, or CONT to continue after a break.	RUN 200
	Start program execution at the lowest numbered statement.	RUN

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
SAVE	SAVE Saves the current program in the AIM 65 memory on cassette tape. The program in memory is left unchanged. More than one program may be stored on cassette using this command. See Appendix G for more information.	SAVE

304 PROGRAM STATEMENTS

In the following description of statements, an argument of B, C, V or W denotes a numeric variable, X denotes a numeric expression, X$ denotes a string expression and an I or J denotes an expression that is truncated to an integer before the statement is executed. Truncation means that any fractional part of the number is lost, e.g., 3.9 becomes 3, 4.01 becomes 4.

An expression is a series of variables, operators, function calls and constants which after the operations and function calls are performed using the precedence rules, evaluates to a numeric or string value.

A constant is either a number (3.14) or a string literal ("FOO").

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
DEF	DEF FNx [(argument list)] = expression The user can define functions like the built-in functions (SQR, SGN, ABS, etc.) through the use of the DEF statement. The name of the function is "FN" followed by any legal variable name, for example: FNX,	100 DEF FNA(V)=V/B+C

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
	FNJ7, FNKO, FNR2. User defined functions are restricted to one line. A function may be defined to be any expression, but may only have one argument. In the example, B and C are variables that are used in the program. Executing the DEF statement defines the function. User defined functions can be redefined by executing another DEF statement for the same function. "V" is called the dummy variable.	
	Execution of this statement following the above would cause Z to be set to 3/B+C, but the value of V would be unchanged.	100 Z=FNA(3)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
DIM	DIM variable (size 1, [size 2...]) Allocates space for matrices. All matrix elements are set to zero by the DIM statement.	113 DIM A(3),B(10)
	Matrices can have from one to 255 dimensions.	114 DIM R3(5,5), D$(2,2,2)
	Matrices can be dimensioned dynamically during program execution. If a matrix is not explicitly dimensioned with a DIM statement, it is assumed to be a single dimensioned matrix of whose single subscript may range 0 to 10 (eleven elements).	115 DIM Q1(N),Z(2*I)
	If this statement was encountered before a DIM statement for A was found in the program, it would be as if a DIM A(10) had been executed previous to the execution of line 117. All subscripts start at zero (0), which means that DIM X(100) really allocates 101 matrix elements.	117 A(8)=4

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
END	END Terminates program execution without printing a BREAK message. (See STOP.) CONT after an END statement causes execution to resume at the statement after the END Statement. END can be used anywhere in the program, and is optional.	999 END

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
FOR	FOR variable = expression to expression [STEP expression] (See NEXT statement) V is set equal to the value of the expression following the equal sign, in this case 1. This value is called the initial value. Then the statements between FOR and NEXT are executed. The final value is the value of the expression following the TO. The step is the value of the expression following STEP. When the NEXT statement is encountered, the step is added to the variable.	300 FOR V=1 TO 9.3 STEP .6
	If no STEP was specified, it is assumed to be one. If the step is positive and the new value of the variable is <= the final value (9.3 in this example), or the step value is negative and the new value of the variable	310 FOR V=1 TO 9.3

is => the final value, then the first statement following the FOR statement is executed. Otherwise, the statement following the NEXT statement is executed. All FOR loops execute the statements between the FOR and the NEXT at least once, even in cases like FOR V=1 TO 0.

	Note that expressions (formulas) may be used for the initial, final and step values in a FOR loop. The values of the expressions are computed only once, before the body of the FOR...NEXT loop is executed.	315 FOR V=10*N TO 3.4/Q STEP SQR(R)
	When the statement after the NEXT is executed, the loop variable is never equal to the final value, but is equal to whatever value caused the FOR...NEXT loop to terminate. The statements between the FOR and its corresponding NEXT in both examples above (310 and 320) would be executed nine times.	320 FOR V=9 TO 1 STEP -1
	Error: do not use nested FOR...NEXT loops with the same index variable.	330 FOR W=1 TO 10: FOR W=1 TO 5:NEXT W:NEXT W
	FOR loop nesting is limited only by the available memory. (See Appendix C.)	
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
GOSUB	GOSUB line number Branches to the specified statement (910) until a RETURN is encountered; when a branch is then made to the statement after the GOSUB. GOSUB nesting is limited only by the available memory.	10 GOSUB 910
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
GOTO	GOTO line number Branches to the statement specified.	50 GOTO 100
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
IF...GOTO	IF expression GOTO line number ... Equivalent to IF...THEN, except that IF...GOTO must be followed by a line number, while IF...THEN can be followed by either a line number or another statement.	32 IF X<=Y+23.4 GOTO 92
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
IF...THEN	IF expression THEN line number ...	IF X<10 THEN 5
	Branches to specified statement if the relation is True.	
	Executes all of the statements on the remainder of the THEN if the relation is True.	20 IF X<0 THEN PRINT "X LESS THAN 0"
	WARNING: The "Z=A" will never be executed because if the relation is true, BASIC will branch to line 50. If the relation is false BASIC will proceed to to the line following line 25.	25 IF X=5 THEN 50:Z=A

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
	In this example, if X is less than 0, the PRINT statement will be executed and then the GOTO statement will branch to line 350. If the X was 0 or positive, BASIC will proceed to execute the lines after line 26.	26 IF X<0 THEN PRINT "ERROR, X NEGATIVE": GOTO 350

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LET	[LET] variable = expression Assigns a value to a variable,	300 LET W=X
	"LET" is optional.	310 V=5.1

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
NEXT	NEXT [variable] [,variable] ... Marks the end of a FOR loop.	340 NEXT V
	If no variable is given, matches the most recent FOR loop,	345 NEXT
	A single NEXT may be used to match multiple FOR statements. Equivalent to NEXT V:NEXT W.	350 NEXT V,W

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
ON...GOSUB	ON expression GOSUB line [,line] ... Identical to "ON...GOTO," except that a subroutine call (GOSUB) is executed instead of a GOTO. RETURN from the GOSUB branches to the statement after the ON...GOSUB.	110 ON I GOSUB 50,60

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
ON...GOTO	ON expression GOTO line [, line] ... Branches to the line indicated by the I'th number after the GOTO. That is: IF I=1, THEN GOTO LINE 10 IF I=2, THEN GOTO LINE 20 IF I=3, THEN GOTO LINE 30 IF I=4, THEN GOTO LINE 40. If I=0, or I attempts to select a nonexistent line (>=5 in this case), the statement after the ON statement is executed. However, if I is >255 or <0, an FC error message will result. As many line numbers as will fit on a line can follow an ON...GOTO.	100 ON I GOTO 10,20, 30,40
	This statement will branch to line 40 if the expression X is less than zero, to line 50 if it equals zero, and to line 60 if it is greater than zero.	105 ON SGN(X)+2 GOTO 40,50,60

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
REM	REM any text Allows the programmer to put comments in his program. REM statements are not executed, but can be branched to. A REM statement is terminated by end of line, but not by a ":".	500 REM NOW SET V=0
	In this case the V=0 will never be executed by BASIC.	505 REM SET V=0: V=0

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
	In this case V=0 will be executed,	505 V=0: REM SET V=0

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
RESTORE	RESTORE Allows the re-reading of DATA statements, After a RESTORE, the next piece of data read will be the first piece listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on as in a normal READ operation.	510 RESTORE

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
RETURN	RETURN Causes a subroutine to return to the statement after the most recently executed GOSUB.	50 RETURN

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
STOP	STOP Causes a program to stop execution and to enter command mode. Prints BREAK IN LINE 900. (As per this example.) CONT after a STOP branches to the statement following the STOP.	900 STOP

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
USR	USR (argument) Calls the user's machine language subroutine with the argument. See PEEK and POKE in Subject 303, and Appendix F.	200 V=USR(W)

SYMBOL	SYNTAX/FUNCTION	EXAMPLE
WAIT	WAIT (address, mask [, select]) This statement reads the contents of the addressed location, does an Exclusive-OR with the select value, and then ANDs the result with the mask. This sequence is repeated until a non-zero result is obtained, at which time execution continues at the statement that follows WAIT. If the WAIT statement has no select argument, the select value is assumed to be zero. If you are waiting for a bit to become zero, there should be a "one" in the corresponding bit position of the select value. The select value (K) and the mask value (J) can range from 0 to 255. The address (I) can range from 0 to 65535.	805 WAIT I,J,K 806 WAIT I,J

305 INPUT/OUTPUT STATEMENTS

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
DATA	DATA item [, item...] Specifies data, read from left to right. Information appears in data statements in the same order as it will be read in the program.	10 DATA 1,3,-1E3,.04
	Strings may be read from DATA statements. If you want the string to contain leading spaces (blanks), colons (:) or	20 DATA "FOO",Z1

```
                commas (,), you must enclose the string
                in double quotes.  It is illegal so have a
                double quote within string data or a
                string literal.  (""BASIC"" is illegal.)

STATEMENT       SYNTAX/FUNCTION                               EXAMPLE

INPUT           INPUT [!] ["prompt string literal";]          3 INPUT V,W,W2
                variable [, variable] ...
                Requests data from the keyboard (to be
                typed in).  Each value must be separated
                from the preceeding value by a comma (,).
                The last value typed should be followed by
                a carriage return.  A "?" is displayed as a
                prompt character.  Only constants may be
                typed in as a response to an INPUT state-
                ment, such as 4.5E-3 or "CAT."  If more
                data was requested in an INPUT statement
                than was typed in, a "??" is printed and
                the rest of the data should be typed in.  If
                more data was typed in than was requested,
                the warning "EXTRA IGNORED" will be
                displayed.  Strings must be input in the
                same format as they are specified in DATA
                statements.

                Optionally displays a prompt string           5 INPUT "VALUE";V
                ("VALUE") before requesting data from
                the keyboard.  If RETURN is typed to an
                input statement, BASIC returns to com-
                command mode.  Typing CONT after an
                INPUT command has been interrupted
                will cause execution to resume at the
                INPUT statement.

                If the optional character ! is included       15 INPUT! "VALUE";V
                following INPUT, then the prompts from
                the INPUT statement and the user's entries
                will be printed (even if the printer is
                turned off) and displayed.

STATEMENT       SYNTAX/FUNCTION                               EXAMPLE

POS             POS (expression)                              260 PRINT POS(I)
                Gives the current position of the cursor on
                the display.  The leftmost character position
                on the display is position zero.  A dummy
                operand--0 or 1--must be used.

STATEMENT       SYNTAX/FUNCTION                               EXAMPLE

PRINT           PRINT [!] expression [, expression]           360 PRINT X,Y;Z
                Prints the value of expressions on the        370 PRINT " "
                display/printer.  If the list of values to be 380 PRINT X,Y;
                printed out does not end with a comma         390 PRINT "VALUE IS";A
                (,) or a semicolon (;), then a carriage       400 PRINT A2,B,
                return/line feed is executed after all the
                values have been printed.  Strings enclosed
                in quotes (") may also be printed.  If a
                semicolon separates two expressions in
                the list, their values are printed next to
                each other.  If a comma appears after an
                expression in the list, and the print head
                is at print position 11 or more, then a
                carriage return/line feed is executed.  If
                the print head is before print position 11,
                then spaces are printed until the carriage
                is at the beginning of the next 10 column
                field.  If there is a blank string enclosed in
                quotes, as in line 370 of the examples,
```

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
	then a carriage return/line feed is executed.	
	"VALUE IS" will be displayed and printed.	410 PRINT ! "VALUE IS";A
	String expressions may be printed.	420 PRINT MID$(A$,2);

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
READ	READ variable [, variable] Read data into specified variables from a DATA statement. The first piece of data read will be the first piece of data listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on. When all of the data have been read from the first DATA statement, the next piece of data to be read will be the first piece listed in the second DATA statement of the program. Attempting to read more data than there is in all the DATA statements in a program will cause an OD (out of data) error.	490 READ V,W

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
SPC	SPC (expression) Prints I space [or blank) characters on the terminal. May be used only in a PRINT statement. I must be =>0 and <=255 or an FC error will result.	250 PRINT SPC(I)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
TAB	TAB (expression) Spaces to the specified print position (column) on the printer. May be used only in PRINT statements. Zero is the leftmost column on the termainl, 19 the rightmost. If the carriage is beyond pos position I, then no printing is done. I must be =>0 and <=255. If I is greater than 19, the printer will skip the required number of lines to arrive at the specified position.	240 PRINT TAB(I)

306 STRING FUNCTIONS

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
ASC	ASC (string expression) Returns the ASCII numeric value of the first character of the string expression X$. See Appendix E for an ASCII/number conversion table. An FC error will occur if X$ is the null string.	300 PRINT ASC(X$)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
CHR$	CHR$ (expression) Returns one character, the ASCII equivalent of the argument (I) which must be a number between 0 and 255. See Appendix E.	275 PRINT CHR$(I)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
GET	GET string variable	10 GET A$

Inputs a single character from the keyboard. If data is at the keyboard, it is put in the variable specified in the GET statement. If no data is available, the BASIC program will continue execution.

GET can only be used as an indirect command.

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LEFT$	LEFT$ (string expression, length) Gives the leftmost I characters of the string expression X$. If I<=0 or >255 an FC error occurs.	310 PRINT LEFT$(X$,I)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LEN	LEN (string expression) Gives the length of the string expression X$ in characters (bytes). Non-printing characters and blanks are counted as part of the length.	220 PRINT LEN(X$)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
MID$	MID$ [string expression, start [, length]) MID$ called with two arguments returns characters from the string expression X$ starting at character position I. If I>LEN(X$), then MID$ returns a null (zero length) string. If I<=0 or >255, an FC error occurs,	330 PRINT MID$(X$,I)
	MID$ called with three arguments returns a string expression composed of the characters of the string expression X$ starting at the Ith character for J characters. If I>LEN(X$), MID$ returns a null string. If I or J <=0 or >255, an FC error occurs. If J specifies more characters than are left in the string, all characters from the Ith on are returned.	340 PRINT MID$(X$, I,J)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
RIGHT$	RIGHT$ (string expression, length) Gives the rightmost I characters of the string expression X$. When I<=0 or >255 an FC error will occur. If I>=LEN(X$) then RIGHT$ returns all of X$.	320 PRINT RIGHT$ (X$,I)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
STR$	STR$ (expression) Gives a string which is the character representation of the numeric expression X. For instance, STR$(3.1)="3.1."	290 PRINT STR$(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
VAL	VAL (string expression) Returns the string expression X$ converted to a number. For instance. VAL("3.1")=3.1. If the first non-space character of the string is not a plus (+) or minus (-) sign; a digit or a decimal point (.) then zero will be returned.	280 PRINT VAL(X$)

307 ARITHMETIC FUNCTIONS

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
ABS	ABS (expression) Gives the absolute value of the expression X. ABS returns X if X>=0, -X otherwise.	120 PRINT ABS(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
ATN	ATN (expression) Gives the arcTangent of the expression X. The result is returned in radians and ranges from -PI/2 to PI/2 (PI/2=1.5708). If you want to use this function, you must provide the code in memory. See Appendix H for implementation details.	210 PRINT ATN(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
COS	COS (expression) Gives the cosine of the expression X. X is interpreted as being in radians.	200 PRINT COS(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
EXP	EXP (expression) Gives the constant "E" (2.71828) raised so the power X (E^X). The maximum argu- ment that can be passed to EXP without overflow occurring is 88.0296.	150 PRINT EXP(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
INT	INT (expression) Returns the largest integer less than or equal to its expression X. For example: INT(.23)=0, INT(7)=7, INT(-.1)=-1, INT(-2)=-2, INT(1.1)=1. The following would round X to 0 decimal places: INT(X*10^D+.5)/10^D	140 PRINT INT(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
LOG	LOG (expression) Gives the natural (Base E) logarithm of its expression X. To obtain the Base Y logarithm of X use the formula LOG(X)/ LOG(Y). Example: The base 10 (com- mon) log of 7 = LOG(7)/LOG(10).	160 PRINT LOG(X)

STATEMENT	SYNTAX/FUNCTION	EXAMPLE
RND	RND (parameter) Generates a random number between 0 and 1. The parameter X controls the generation of random numbers as follows: X<0 starts a new sequence of random numbers using X. Calling RND with the same X starts the same random number sequence. X=0 gives the last random number generated. Repeated calls to RND(0) will always return the same random number. X>0 generates a new random number between 0 and 1.	170 PRINT RND(X)

	Note that (B-A)*RND(1)+A will generate a random number between A and B.	
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
SGN	SGN (expression) Gives 1. If X>0, 0 if X=0, and -1 if X<0.	230 PRINT SGN(X)
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
SIN	SIN (expression) Gives the sine of the expression X. X is interpreted as being in radians. Note: COS(X) =SIN(X+3.14159/2) and that 1 Radian = 180/PI degrees = 57.2958 degrees; so that the sine of X degrees= SIN(X/57.2958).	190 PRINT SIN(X)
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
SQR	SQR (expression) Gives the square root of the expression X. An FC error will occur if X is less than zero,	180 PRINT SQR(X)
STATEMENT	SYNTAX/FUNCTION	EXAMPLE
TAN	TAN (expression) Gives the tangent of the expression X. X is interpreted as being in radians.	200 PRINT TAN(X)

DERIVED FUNCTIONS

The following functions, while not intrinsic to BASIC, can be calculated using the existing BASIC functions:

FUNCTION	FUNCTION EXPRESSED IN TERMS OF BASIC FUNCTIONS
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENT	COT(X) = 1/TAN(X)
INVERSE SINE*	ARCSIN(X) = ATN(X/SQR(-X*X+1))
INVERSE COSINE*	ARCCOS(X) = -ATN(X/SQR(-X*X+1))+1.5708
INVERSE SECANT*	ARCSEC(X) = ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708
INVERSE COSECANT*	ARCCSC(X) = ATN(1/SQR(X*X-1))+(SGN(X)-1)*1.5708
INVERSE COTANGENT*	ARCCOT(X) = -ATN(X)+1.5708
HYPERBOLIC SINE	SINH(X) = (EXP(X)-EXP(-X))/2
HYPERBOLIC COSINE	COSH(X) = (EXP(X)+EXP(-X))/2
HYPERBOLIC TANGENT	TANH(X) = -EXP(-X)/(EXP(X)+EXP(-X))*2+1
HYPERBOLIC SECANT	SECH(X) = 2/(EXP(X)+EXP(-X))
HYPERBOLIC COSECANT	CSCH(X) = 2/(EXP(X)-EXP(-X))
HYPERBOLIC COTANGENT	COTH(X) = EXP(-X)/(EXP(X)-EXP(-X))*2+1

*These functions require the user-defined ATN function. See Appendix H for details.

FUNCTION	FUNCTION EXPRESSED IN TERMS OF BASIC FUNCTIONS
INVERSE HYPERBOLIC SINE	ARGSINH(X) = LOG(X+SQR(X*X+1))
INVERSE HYPERBOLIC COSINE	ARGCOSH(X) = LOG(X+SQR(X*X-1))
INVERSE HYPERBOLIC TANGENT	ARGTANH(X) = LOG((1+X)/(1-X))/2
INVERSE HYPERBOLIC SECANT	ARGSECH(X) = LOG((XQR(-X*X+1)+1)/X
INVERSE HYPERBOLIC COSECANT	ARGCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
INVERSE HYPERBOLIC COTANGENT	ARGCOTH(X) = LOG((X+1)/(X-1))/2

A ERROR MESSAGES

If an error occurs, BASIC outputs an error message, returns to command level and displays the cursor. Variable values and the program text remain intact, but the program can not be continued and all GOSUB and FOR context is lost.

When an error occurs in a direct statement, no line number is printed.

Format of error messages:

Direct Statement	?XX ERROR
Indirect Statement	?XX ERROR IN YYYYY

In both of the above examples, "XX" will be the error code. The "YYYYY" will be the line number where the error occured for the indirect statement.

The following are the possible error codes and their meanings:

ERROR CODE	MEANING
BS	Bad Subscript. An attempt was made to reference a matrix element which is outside the dimensions of the matrix. This error can occur if the wrong number of dimensions are used in a matrix reference; for instance, LET A(1,1,1)=Z when A has been dimensioned DIM A(2,2).
CN	Continue error, Attempt to continue a program when none exists, an error occured, or after a new line was typed into the program.
DD	Double Dimension. After a matrix was dimensioned, another DIM statement for the same matrix was encountered. This error often occurs if a matrix has been given the default dimension 10 because a statement like A(I)=3 is encountered and then later in the program a DIM A(100) is found,
FC	Function Call error, The parameter passed to a math or string function was out of range. FC errors can occur due to:

1. A negative matrix subscript (LET A(-1)=0)
2. An unreasonably large matrix subscript (>32767)
3. LOG-negative or zero argument
4. SQR-negative argument
5. A^B with A negative and B not an integer
6. A call to USR before the address of the machine language

subroutine has been patched in

7. Calls to MID$, LEFT$, RIGHT$, WAIT, PEEK, POKE, TAB, SPC or ON...GOTO with an improper argument.

ID — Illegal Direct. You cannot use an INPUT, DEF or GET statement as a direct command.

LS — Long String. Attempt was made by use of the concantenation operator to create a string more than 255 characters long.

NF — NEXT without FOR. The variable in a NEXT statement corresponds to no previously executed FOR statement.

OD — Out of Data. A READ statement was executed but all of the DATA statements in the program have already been read. The program tried to read too much data or insufficient data was included in the program.

OM — Out of Memory. Program too large, too many variables, too many FOR loops, too many GOSUB's, too complicated an expression, or any combination of the above. (see Appendix B)

OV — Overflow. The result of a calculation was too large to be represented in BASIC's number format. If an underflow (too small result) occurs, zero is given as the result and execution continues without any error message being printed.

RG — RETURN without GOSUB. A RETURN statement was encountered without a previous GOSUB statement being executed,

SN — Syntax error. Missing parenthesis in an expression, illegal character in a line, incorrect punctuation, etc.

ST — String Temporaries. A string expression was too complex. Break it into two or more shorter expressions.

TM — Type Mismatch. The left side of an assignment statement was a numeric variable and the right side was a string, or vice versa; or, a function which expected a string argument was given a numeric one or vice versa.

UF — Undefined Function. Reference was made to a user function which has never been defined.

US — Undefined Statement. An attempt was made to GOTO, GOSUB or THEN to a statement which does not exist.

/0 — Division by Zero

B SPACE HINTS

In order to make your program smaller and save space, the following hints may be useful.

1. Use multiple statements per line. There is a five-byte of overhead associated with each line in the program. Two of these five bytes contain the line number of the line in binary. This means that no matter how many digits you have in your line number (minimum line number is 0, maximum is 63999), it takes the same number of bytes. Putting as many statements as possible on a line will cut down on the number of bytes used by your program.

2. Delete all unnecessary spaces from your program. For instance:

```
10 PRINT X, Y, Z
```

uses three more bytes than

```
10 PRINTX,Y,Z
```

Note: All spaces between the line number and the first non-blank character are ignored.

3. Delete all REM statements. Each REM statement uses at least one byte plus the number in the comment text. For instance, the statement 130 REM THIS IS A COMMENT uses 24 bytes of memory.

   In the statement 140 X=X+Y: REM UPDATE SUM, the REM uses 14 bytes of memory including the colon before the REM.

4. Use variables instead of constants. Suppose you use the constant 3.14159 ten times in your program. If you insert a statement

   ```
   10 P=3.1.4159
   ```

   in the program, and use P instead of 3.14159 each time it is needed, you will save 40 bytes. This will also result in a speed improvement.

5. A program need not end with an END, so an END statement at the end of a program may be deleted.

6. Reuse variables. If you have a variable T which is used so hold a temporary result in one part of the program and you need a temporary variable later in your program, use it again. Or, if you are asking the terminal user to give a YES or NO answer to two different questions at two different times during the execution of the program, use the same temporary variable A$ to store the reply.

7. Use GOSUB's to execute sections of program statements that perform identical actions.

8. Use the zero elements of matrices; for instance, A(0), B(0,X).

STORAGE ALLOCATION INFORMATION

Simple (non-matrix) numeric and strong variables like V use 7 bytes; 2 for the variable name, and 5 for the value. Simple non-matrix string variables also use 7 bytes; 2 for the variable name, 1 for the length, 2 for a pointer, and 2 are unused.

Matrix variables require 7 bytes to hold the header, plus additional bytes to hold each matrix element. Each element that is an integer variable requires 2 bytes. Elements that are string variables or floating point variables require 3 bytes or 5 bytes, respectively.

String variables also use one byte of string space for each character in the string. This is true whether the string variable is a simple string variable like A$, or an element of a string matrix such as Q1$(5,2).

When a new function is defined by a DEF statement, 7 bytes are used to store the definition.

Reserved words such as FOR, GOTO or NOT, and the names of the intrinsic functions such as COS, INT and STR$ take up only one byte of program storage. All other characters in programs use one byte of program storage each.

When a program is being executed, space is dynamically allocated on the stack as follows:

1. Each active FOR...NEXT loop uses 22 bytes.

2. Each active GOSUB (one that has not returned yet) uses 6 bytes.

3. Each parenthesis encountered in an expression uses 4 bytes and each temporary result calculated in an expression uses 12 bytes.

C SPEED HINTS

The hints below should improve the execution time of your BASIC program. Note that some of these hints are the same as those used to decrease the space used by your programs. This means that in many cases you can increase the efficiency of both the speed and size of your programs at the same time.

1. Delete all unnecessary spaces and REM's from the program. This may cause a small decrease in execution time because BASIC would otherwise have to ignore or skip over spaces and REM statements.

2. THIS IS PROBABLY THE MOST IMPORTANT SPEED HINT.

   Use variables instead of constants. It takes more time to convert a constant to its floating point representation than it does to fetch the value of a simple or matrix variable. This is especially important within FOR...NEXT loops or other code that is executed repeatedly.

3. Variables which are encountered first during the execution of a BASIC program are allocated at the start of the variable table. This means that a statement such as 5 A=0:B=A:C=A, will place A first, B second, and C third in the symbol table (assuming line 5 is the first statement executed in the program). Later in the program, when BASIC finds a reference to the variable A, it will search only one entry in the symbol table to find A, two entries to find B and three entries to find C, etc.

4. Use NEXT statements without the index variable. NEXT is somewhat faster than NEXT I because no check is made to see whether the variable specified in the NEXT is the same as the variable in the most recent FOR statement.

D CONVERTING BASIC PROGRAMS NOT WRITTEN FOR AIM 65 BASIC

Though implementations of BASIC on different computers are in many ways similar, there are some incompatibilities which you should watch for if you are planning to convert some BASIC programs that were not written in AIM 65 BASIC.

1. Matrix subscripts. Some BASICs use "[" and "]" to denote matrix subscripts. AIM 65 BASIC uses "(" and ")".

2. Strings. A number of BASICs force you to dimension (declare) the length of strings before you use them. You should remove all dimension statements of this type from the program. In some of these BASICs, a declaration of the form DIM A$(I,J) declares a string matrix of J elements each of which has a length I. Convert DIM statements of this type to equivalent ones in AIM 65 BASIC: DIM A$(J).

   AIM 65 BASIC uses "+" for string concatenation, not "," or "&".

   AIM 65 BASIC uses LEFT$, RIGHT$ and MID$ to take substrings of strings. Other BASICs uses A$(I) to access the Ith character of the string A$, and A$(I,J) to take a substring of A$ from character position I to character position J. Convert as follows:

OLD	AIM 65
A$(I)	MID$(A$,I,1)
A$(I,J)	MID$(A$,I,J-I+1)

   This assumes that the reference to a substring of A$ is in an expression or is on the right side of an assignment. If the reference to A$ is on the left hand side of an assignment, and X$ is the string expression used to replace characters in A$, convert as follows:

OLD	AIM 65
A$(I)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,J+1)

3. Multiple assignments. Some BASICs allow statements of the form: 500 LET B=C=0. This statement would set the variables B & C to zero.

   In AIM 65 BASIC this has an entirely different effect. All the "='s" to the right of the first one would be interpreted as logical comparison operators. This would set the variable B to -1 if C equaled 0. If C did not equal 0, B would be set to 0. The easiest way to convert statements like this one is to rewrite them as follows:

   ```
   500 C=0:B=C
   ```

4. Some BASICs use "/" instead of ":" to delimit multiple statements per line. Change all occurrences of "/" to ":" in the program.

5. Programs which use the MAT functions available in some BASICs will have to be re-written using FOR...NEXT loops to perform the appropriate operations.

6. A PRINT statement with no arguments will not cause a paper feed on the printer. To generate a paper feed (blank line), use PRINT "space"

E ASCII CHARACTER CODES

DECIMAL	CHAR.	DECIMAL	CHAR.	DECIMAL	CHAR.
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1.	092	/
007	BEL	050	2	093	]
008	BS	051	3	094	^
009	HT	052	4	095	_
010	LF	053	5	096	`
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

LF=Line Feed FF=Form Feed CR=Carriage Return DEL=Rubout on TTY

F ASSEMBLY LANGUAGE SUBROUTINES

AIM 65 BASIC allows a user to link to assembly language subroutines, via the USR(W) function. This function allows one parameter to be passed between BASIC and a subroutine.

The first step is to allocate sufficient memory for the subroutine. AIM 65 BASIC always uses all RAM memory locations, beginning at decimal location 530 (hex location 212), unless limited by the user. You can limit BASIC's memory useage by answering the prompt MEMORY SIZE? (see Subject 100) with some number less than 4096, assuming a 4K system. This will leave sufficient space for the subroutine as the top of RAM.

For example, if your response to MEMORY SIZE? is "2048", 1518 bytes at the top of RAM will be free for assembly language subroutines.

Parameter (W), passed to a subroutine by USR(W), will be converted to floating-point accumulator located at $A9. The floating-point accumulator has the following format:

ADDRESS	CONTENT
$A9	Exponent + $81 ($80 if mantissa = 00)
$AA-$AD	Mantissa, normalized so that Bit 7 of MSB is set. $AA is MSB, $AD is LSB.
$AE	Sign of mantissa

A parameter passed to an assembly language subroutine from BASIC can be truncated by the subroutine to a 2-byte integer and deposited in $AC (MSB) and $AD (LSB). If the parameter is greater than 32767 or less than -32768, an FC error will result. The address of the subroutine that converts a floating-point number to an integer is located in $B006, $B007.

A parameter passed to BASIC from an assembly language subroutine will be converted to floating-point. The address of the subroutine that performs this conversion is in $B008, $B009. The integer MSB ($AC) must be in the accumulator; the integer LSB ($AD) must be in the Y register.

Prior to executing USR, the starting address of the assembly language subroutine must be stored in locations $04 (LSB) and $05 (MSB). This is generally performed using the POKE command. Note that more than one assembly language subroutine may be called from a BASIC program, by changing the starting address in $04 and $05.

Figure F-1 is the listing for a BASIC program that calls an assembly language subroutine located at $A00. Here's what the BASIC program does:

* Line 10 - Stores the starting address of the assembly language subroutine ($A00) into locations $04 and $05, using POKE.
* Line 20 - Asks for a number "N".
* Line 30 - Calls the subroutine, with N as the parameter.
* Line 40 - Upon return from the subroutine, the BASIC program prints X, the parameter passed from the subroutine to the BASIC program.
* Line 50 - Loops back to get a new N

```
ROCKWELL AIM 65

<5>
MEMORY SIZE? 2048
WIDTH?
 1518 BYTES FREE
AIM 65 BASIC V1.1
OK
10 POKE 04,0: POKE 05
,10
20 INPUT"NUMBER";N
30 X=USR(N)
40 PRINTX
50 GOTO 20
```

Figure F-1. BASIC Program That Calls Assembly Language Subroutine

The assembly language subroutine (Figure F-2) performs these operations:

* Prints the floating-point accumulator ($A9-$AE), using Monitor subroutines NUMA ($EA46), BLANK ($E83E) and CRLF ($E9F0),
* Converts the floating-point accumulator to an integer, using the subroutine at $BF00. The address $BF00 was found in locations $B006, $B007. (Address $BF00 may vary with different versions of BASIC. Be sure to check locations $B006 and $B007 for the correct address.)
* After conversion, the program again prints the floating point accumulator,

* The program then swaps the bytes of the integer.

* Finally, the program converts the result to floating point and returns to BASIC (JMP C0D3). Address $C0D3 was found in locations $B008, $B009. (Address $C0D3 may vary with different versions of BASIC. Be sure to check locations $B008 and $B009 for the correct address.

```
<1>
 0A26    *=A00
 0A00 A0 LDY #00
 0A02 A2 LDX #00
 0A04 B5 LDA A9,X
 0A06 20 JSR EA46
 0A09 20 JSR E83E
 0A0C E8 INX
 0A0D E0 CPX #06
 0A0F D0 BNE 0A04
 0A11 20 JSR E9F0
 0A14 C0 CPY #00
 0A16 F0 BEQ 0A1F
 0A13 A5 LDA AD
 0A18 A4 LDY AC
 0A1C 4C JMP C0D3
 0A1F 20 JSR BF00
 0A22 C8 INY
 0A23 D0 BNE 0A02
 0A25 00 BRK
 0A26
```

Figure F-2 Assembly Language Subroutine

Figure F-3 shows the print-out for various values of "N".

```
<6>
OK
RUN
NUMBER? 128
88 80 00 00 00 00
88 00 00 00 80 00
-32768

NUMBER? 1
81 80 00 00 00 00
81 00 00 00 01 00
 256

NUMBER? 4097
8D 80 06 00 00 00
8D 00 00 10 01 00
 272

NUMBER? 256
89 80 00 00 00 00
89 00 00 01 00 00
 1
```

Figure F-3. Output for Example

G STORING AIM 65 BASIC PROGRAMS ON CASSETTE

AIM 65 BASIC Programs can be stored on cassette tape by using BASIC's SAVE and LOAD commands, or by using the AIM 65 Editor. Before employing either procedure be sure to carefully observe the recorder installation and operation procedures given in Section 9 of the AIM 65 User's Guide.

RECORDING ON CASSETTE USING THE BASIC SAVE COMMAND

The procedure to store a BASIC program is:

1. Install a cassette in the recorder, and manually position the tape to the program record

position. Be sure to initialise the counter at the start of the tape.

Note: Since remote control must be used to retrieve a BASIC program, observe the tape gap CAUTION in Section 9.1.5 (Step 1) of the AIM 65 User's Guide.

2. While in BASIC, type in SAVE. BASIC will respond with:

   OUT=

3. Enter a T (for "Tape"). BASIC will display:

   OUT=T F=

4. Enter the file name (up to five characters). If the file name is FNAME, BASIC will display:

   OUT=T F=FNAME T=

5. Put the recorder into Record mode.

6. Enter the recorder number (1 or 2) and type RETURN.

7. If remote control is being used, observe the procedures outlined in Section 9.1.5 of the AIM 65 User's Guide.

8. When recording has been completed, BASIC will display the cursor.

9. Switch the recorder out of record mode.

RETRIEVING A PROGRAM FROM CASSETTE USING THE BASIC LOAD COMMAND

The procedure to retrieve a BASIC program is:

1. Install the cassette in the recorder., and manually position the tape to about five counts before the beginning of the desired file.

   Note: Remote control must be used when retrieving a file via BASIC.

2. While in BASIC, type in LOAD. BASIC will respond with:

   IN=

3. Enter a T (for "Tape"). BASIC will display:

   IN=T F=

4. Enter the file name. If the file name is FNAME, BASIC will display:

   IN=T F=FNAME T=

5. Enter the recorder number (1 or 2) and type RETURN.

6. Put the recorder into play mode. Be sure to observe the procedures outlined in Section 9.1.6 of the AIM 65 User's Guide.

   While the file is being read, each line will be displayed (and printed, if the printer is on). If the printer is on, the tape gap ($A409) will probably have to be increased.

   The file being loaded will not overlay any BASIC statements already entered unless the statement numbers are the same.

7. When loading has been completed. BASIC will display the cursor.

8. Switch the recorder out of play mode.

CASSETTE OPERATIONS USING THE AIM 65 EDITOR

AIM 65 BASIC programs can also be stored and retrieved from cassette using the AIM 65 Editor. However, if the program is to be retrieved by BASIC at some future time, one rule must be observed:

When BASIC stores a program on cassette, it inserts a CTRL/Z after the last line. The AIM 65 Editor will strip off the CTRL/Z when it retrieves the program. Therefore, before storing a BASIC program from the Editor, the user must insert a CTRL/Z following the last line of the program.

H ATN IMPLEMENTATION

The ATN function (see Subject 307) can be programmed in RAM using the AIM 65 Mnemonic Entry (I) and Alter Memory Locations (/) commands, as shown below. The program is written for the AIM 65 with 4K bytes of RAM. The ATN function can be relocated elsewhere in memory by changing the starting addresses of the instructions and constants, the conditional branch addresses, the vector to the constants start address and the vector to the ATN function starT address.

ATN FUNCTION CONSTANTS ENTERED BY ALTER MEMORY <M>

```
<M> = 0F80  XX  XX  XX  XX    Constants Starting Address = 0F80
</> = 0F80  0B  76  83  83                                     8
</>   0F84  BD  D3  79  1E
</>   0F88  F4  A6  F5  7B
</>   0F8C  83  FC  B0  10
</>   0F90  7C  0C  1F  67
</>   0F94  CA  7C  DE  53
</>   0F98  CB  C1  7D  14
</>   OF9C  64  70  4C  7D
</>   0FA0  B7  EA  51. 7A
</>   0FA4  7D  63  30  88
</>   0FA8  7E  7E  92  44
</>   0FAC  99  3A  7E  4C
</>   0FR0  CC  91  C7  7F
</>   0FB4  AA  AA  AA  13
</>   0FR8  81  00  00  00
</>   0FBC  00
```

ATN FUNCTION INSTRUCTIONS STORED BY MNEMONIC ENTRY (I)

```
<I>
XXXX *=0FBD                   Instructions Starting Address = 0F8D
0FBD A5 LDA AE
0FBF 48 PHA
0FC0 10 BPL 0FC5
0FC2 20 JSR CCB8
0FC5 AS LDA A9
0FC7 48 PHA
0FC8 C9 CMP #81
0FCA 90 BCC 0FD3
0FCC A9 LDA #FB
0FCE A0 LDY #C6
0FD0 20 JSR C84E
0FD3 A9 LDA #80 \             Starting Address of Constants = 0F80
0FD5 A0 LDY #0F /
0FD7 20 JSR CD44
0FDA 68 PLA
0FDB C9 CMP #81
0FDD 90 BCC 0FE6
0FDF A9 LDA #4E
0FE1 A0 LDY #CE
0FE3 20 JSR C58F
0FE6 68 PLA
0FE7 10 BPL 0FEC
0FE9 4C JMP CCB8
0FEC 60 RTS
0FEC
```

BASIC INITIALIZATION FOR ATN FUNCTION

BASIC memory must be initialized below the memory allocated to the ATN function. The ATN vector in RAM must also be changed from the address of the FC error message to the starting address of the ATN function instructions. This can be done using BASIC initialization, as follows:

```
<5>
MEMORY SIZE? 3968              Limit BASIC to F80
WIDTH?                                            16
 3438 BYTES FREE
  AIM 65 BASIC V1.1
POKE 188,189                   Change ATN function vector low to $BD
POKE 189,15                    Change ATN function vector high to $0F
?ATN (TAN(.5))                 Test case to verify proper ATN function program
 .5                            Expected answer = .5
```

SAVING ATN OBJECT CODE ON CASSETTE

The object code for the ATN function can be saved on cassette by dumping addresses $00BB through $00BD (Jump instruction to ATN) and $0F80 through $0FEC (constants and instructions) after the function is initially loaded and verified.

The ATN function can then be loaded from cassette by executing the Monitor L command after BASIC has been initialized via the 5 command. After the ATN function has been loaded, reenter BASIC with the 6 command.

###