CRASH DIVE!

by

Brian Moriarty

24K Cassette 32K Disk

You're on maintenance duty aboard the USS **Sea Moss**, *patrolling the icy North Atlantic waters with an arsenal of twenty nuclear missiles.*

The **Sea Moss** *is no ordinary sub. She's the first to carry the Navy's new experimental sonar-jammer that can make her "invisible" to even the most sophisticated enemy sensors. The 50-kiloton cruisers in her missile bay are the pride of the Pentagon: fast, silent, incredibly accurate.*

The enemy would love to get their hands on the **Sea Moss** *and her secrets. It's not likely to happen, though. The only way they could possibly breach the hull would be from the inside — and your fellow crewmembers have been carefully handpicked for their unswerving patriotism and utter lack of imagination. No "moles" in this bunch of sailors. No, sir!*

The intercom in the equipment bay clicks to life. "I've got a bad line in the forward escape tube," a voice from the command deck crackles. "Wanna come up here and take a look at it?" You grab a screwdriver, scoot up a ladder and slam the hatch of the escape tube behind you.

It's all over in a few seconds. The General Quarters klaxxon blares to life. You hear the shrieks and choked coughing of friends as they stumble through the passages outside, and a single hoarse shout: "Gas!" Some poor sucker pounds weakly on the escape hatch. Then the alarm cuts off as suddenly as it began. Everything is silent as death. Frozen with fear, you sit trembling in the airtight escape tube, knowing that now it's just you and the **Sea Moss** *against whoever shut off the alarm.*

The game.

Crash Dive! is a machine-language text adventure that pits you in a race against time. As the sole survivor of a terrible act of naval sabotage, you must find a way to keep your ship out of the hands of The Enemy. No sacrifice is too great to achieve this important goal. The question is, how do you get rid of a giant nuclear submarine and everything in it?

As the start of the game, the *Sea Moss* is assumed to be cruising along the surface of the ocean. Your mission is as follows:

1. Find a way to survive in the submarine's poisoned atmosphere.
2. Get the sub under water, so that enemy ships will not be able to reach it easily. You have a limited number of moves after the game begins to accomplish this, or the Enemy will capture the sub and kill you on the spot!
3. Find a way to completely destroy the *Sea Moss*.

Some of these goals will be relatively easy to accomplish. Others will require careful thought and a little bit of resourcefulness. Don't forget that there may be somebody left alive on the *Sea Moss* besides yourself — and that somebody might not be very friendly!

We'll discuss the details of playing **Crash Dive!** in a moment. First, let's take a look at the program itself, and how to get it up and running on your computer.

Typing it in.

Listing 1 is an Atari BASIC program that will create an auto-booting version of **Crash Dive!** on disk or cassette. The DATA statements are listed in hexadecimal (base 16) in order to make the program as small as possible. It makes typing a little more difficult, but it's a necessary evil.

Listing 1 will not fit in a 16K Atari system. You'll need at least 24K of memory if you're using cassette, or 32K if you're using disk. However, the machine-language file created by **Listing 1** does fit in 16K. If you only have 16K in your computer, ask a friend with a larger system to help you type in and RUN the BASIC listing. After the boot tape or disk is made, you'll be able to enjoy the game on your 16K system.

Listing 2 is the assembly-language source code for **Crash Dive!**, created with the MAC/65 Macro Assembler. You do *not* have to type **Listing 2** into your computer to play the game (thank goodness!). It's provided for those readers interested in learning how the program works.

Follow the instructions below to make either a cassette or disk version of **Crash Dive!**

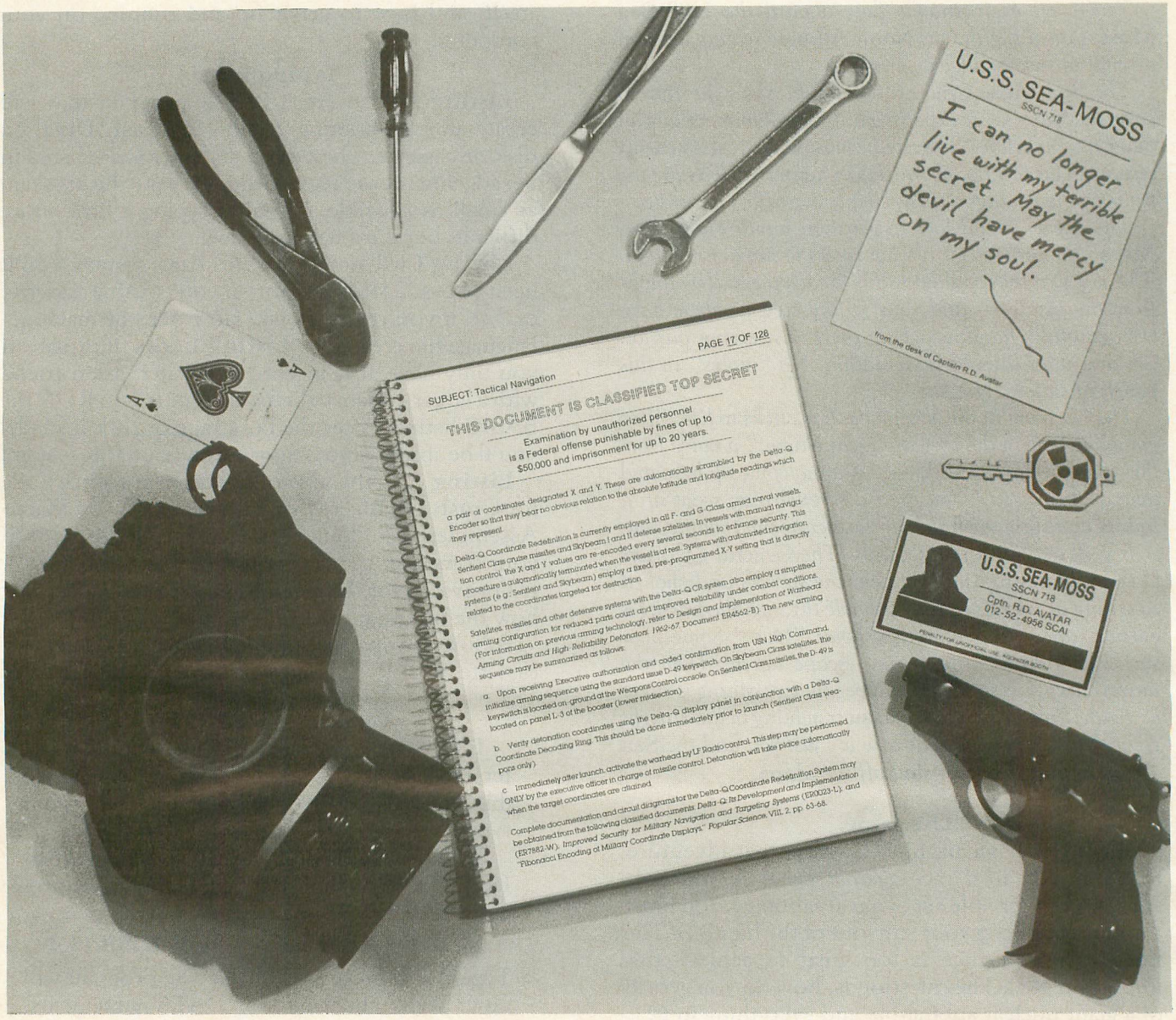
Cassette instructions.

1. Carefully type **Listing 1** into your computer (remember, you need at least 24K to do this). Use **C:CHECK** (page 30) to verify your typing.
2. When **C:CHECK** says the program is perfect, type RUN and press RETURN. The program will prompt you with:

MAKE CASSETTE (0) OR DISK (1)?

Type 0 and press RETURN. The program will now begin checking the DATA statements, printing the line number of each as it goes. It will alert you if it finds any problems. Fix any incorrect lines and re-RUN the program as necessary until all typos are eliminated.

3. When all DATA lines are correct, the computer will "beep" twice and prompt you to READY CASSETTE AND PRESS RETURN. Insert a blank cassette into your recorder, press the PLAY and RECORD buttons simultaneously and hit RETURN. The message WRITING FILE will appear, and the program will create a machine-language boot-tape version of **Crash Dive!**, printing the line number of each DATA statement as it goes. When the READY prompt reappears, the game is recorded and ready to play. CSAVE the BASIC program on a separate tape before continuing.
4. To play **Crash Dive!**, rewind the boot tape created by the BASIC program to the beginning. Turn your computer OFF and remove all cartridges. Press the PLAY button on your recorder



and turn ON your computer while holding down the START key. If you have a 600XL or 800XL computer, you must hold down the START and the OPTION keys together when you turn on the power. The computer will "beep" once. Hit the RETURN key and **Crash Dive!** will load and run automatically.

Disk instructions.

1. Type **Listing 1** into your computer and use **D:CHECK2** (see page 30) to verify your typing.

2. When **D:CHECK** says the BASIC code is perfect, type RUN and press RETURN. The program will ask:

MAKE CASSETTE (0) OR DISK (1)?

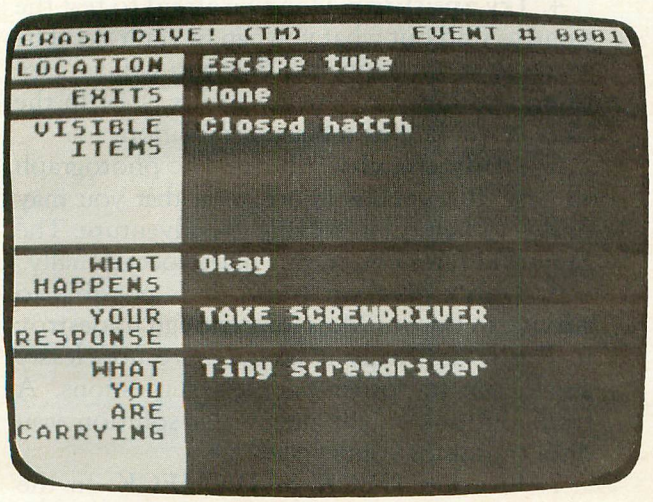
Type 1 and press RETURN. The program will begin checking the DATA statements, printing the line number of each statement as it proceeds. The program will alert you if it finds any problems. Fix incorrect lines and re-RUN the program as necessary until all typos are eliminated.

3. When all DATA lines are correct, the program will prompt you to INSERT DISK WITH DOS, PRESS RETURN. Put a disk containing Atari DOS 2.0S into drive #1 and press RETURN. The message WRITING FILE will appear and the program will create a binary AUTORUN.SYS file on the disk, displaying the line number of each DATA statement as it goes. When the READY prompt reappears, **Crash Dive!** is ready to play. Be sure the BASIC program is SAVEd out to a disk before continuing.

4. To play the game, insert the disk containing the AUTORUN.SYS file into drive #1. Turn your computer OFF, remove all cartridges and turn the computer back ON. **Crash Dive!** will load and run automatically.

Assuming everything went okay, you should now be looking at the **Crash Dive!** title screen, which includes the following prompt:

Press START to play new game
Press OPTION to restore old game



Crash Dive!

You haven't played the game before, so press the START key. Your TV screen should now look like the screen shown above. Note that the screen is divided into seven distinct sections or *windows*. From top to bottom, they are:

Event Window. The EVENT # counter in the top right corner keeps track of how many "events" have transpired since the start of the game. In general, each movement or other action you take during the course of the game counts as one event.

Location Window. This window contains a brief description of your current location.

Exit Window. The Exit Window tells you which directions you can go from your current location. Six vectors of movement are allowed: N (North), S (South), E (East), W (West), U (Up) and D (Down). "North" is towards the front of the submarine, "South" is aft, and so forth. If you can't move from a given location for some reason, the Exit Window will read "None."

Visible Items Window. This window displays a list of all the things you can see at your current location. Up to six items may be present in a location at any one time. Unoccupied locations will contain "Nothing."

What Happens Window. The What Happens Window reports on the results of your actions and lets you know if anything interesting is happening on board the *Sea Moss*. Keep an eye on this window — it may offer valuable information you will need to complete the adventure.

Response Window. This 2-line window is the communications link between you and the game. The commands and sentences you type into the Response Window tell the computer how you want to proceed. Each new line scrolls up into the top half of the window after you hit RETURN, so that you can see what you just typed. A blinking green underline keeps track of your position.

Inventory Window. Look here for a list of all the items you are carrying. Up to six items may be held at one time. If you're empty-handed, the window will say "Nothing."

Talking to Crash Dive!

Like most text adventures, **Crash Dive!** understands two-word sentences in the form:

VERB NOUN

The single space character between the verb and the noun is required. Don't worry about capitals vs. lower-case, numbers or funny characters — **Crash Dive!** has a "smart" keyboard handler that will snarl at you if you try entering anything illegal.

The best way to learn how to talk to **Crash Dive!** is to play with it. Let's use the opening screen as an example. The Location Window says you're in the escape tube, with no obvious exits and nothing in your inventory. You can "see" a closed hatch and a tiny screwdriver. Type the sentence TAKE SCREWDRIVER and you'll see the screwdriver vanish from the Visible Items list and reappear in your inventory. Simple, right?

You can interact with objects on the screen much like you can in real life. Type EXAMINE SCREWDRIVER and the What Happens Window will tell you that it "Seems ordinary." Now try EXAMINE HATCH and learn something interesting about the escape hatch. If you try to TAKE HATCH, you'll find out what happens when you attempt something impossible. DROP SCREWDRIVER will put the tiny screwdriver back in the Visible Items list.

You may be tempted to type OPEN HATCH, but if you read my little introductory tale carefully you'll know better than to try it. Think about your situation for a while and you'll discover a way to explore the rest of the *Sea Moss* without suffocating!

Commands.

Crash Dive! also understands a limited number of single-character commands. These are used to control your movement around the sub, and to perform special "internal" game functions. The following commands are recognized by **Crash Dive!**:

Movement Commands

N - North S - South E - East
W - West U - Up D - Down

Internal Commands

X - Mark Game Position
Q - Quit/Restart Game
A - Again (Repeat Last Sentence)

The movement commands are easy to use. Just consult the Exit Window to see which vectors are available, and type the initial of the direction you want to

go. The program will scold you if you type an illegal direction.

Saving your game.

The "X" (Mark Game Position) command is used when you want to save the current status of your game. Type X/RETURN and you'll see the following prompt:

SAVE GAME TO DISK OR CASSETTE?

If you're using a disk drive, insert a disk containing Atari DOS 2.0S into drive #1 and press the "D" key. Your game will be saved out in a few seconds and you'll return to the main screen.

If you're using cassette, insert a blank tape into your recorder and press the "C" key. The computer will "beep" twice. Press the PLAY and RECORD keys on the recorder simultaneously and hit RETURN. The game will be saved and you'll return to the main screen.

Starting over.

The "Q" (Quit/Restart) command is used when you want to restart the game from the beginning, or restore a game you have previously saved to disk or tape. Type Q/RETURN and you'll see the familiar **Crash Dive!** title screen. Press the START key if you want to start over from scratch. Press OPTION and you'll be asked:

RESTORE FROM DISK OR CASSETTE?

If your game was saved on disk, insert the game disk into drive #1 and press the "D" key. Your game will automatically resume at exactly the point where you left it.

If your game was saved on tape, cue the tape to the beginning of the saved game and press the "C" key. The computer will "beep" once. Press the PLAY key on the recorder and hit RETURN. Your game will resume at the point where you left it.

The A command.

The third and last command recognized by **Crash Dive!** is "A," which means Again. This command re-executes the last sentence you typed as if you had typed it in again yourself. The A command only repeats your last sentence (verb/noun); it will *not* repeat single-character commands.

Hints for successful play.

1. **Draw a map.** You'll have a hard time remembering the layout of the *Sea Moss* unless you draw a map. There are no mazes in this adventure, but a map will help you recall where interesting items are located and how the various rooms are connected.

2. **Examine everything.** Objects may have important features that will not be evident unless you examine them closely. Most of the items you discover in the game are essential to your success (though I may have left a couple of red herrings lying around. . .).

3. **Save your game frequently.** Use the X command to save your current status after important discoveries and breakthroughs, and before trying anything that might be dangerous. Otherwise you may find yourself starting all over again in the escape tube.

4. **Try anything.** Don't be afraid to test the game to find out what you can or can't do. The worst that can happen is that you will be captured and killed by enemy agents, shot in the back or cooked by a blast of radiation.

5. **Study the clue photo.** The photograph on page 46 contains information that you may find very helpful in solving the adventure. The game will refer you to this photo occasionally.

6. **Don't give up hope.** It *is* possible to survive in the *Sea Moss* long enough to destroy it. Really and truly it is! If you're hopelessly stuck, ask for other people's suggestions. A fresh outlook might uncover a solution you didn't think of yourself.

7. **Use C:CHECK or D:CHECK** on the program before you try to use it. It only takes one byte in the wrong place to make **Crash Dive!** totally unplayable.

8. **Don't call ANALOG.** We are absolutely not giving out adventure hints over the telephone! If you're really stuck, send me a self-addressed, stamped envelope at the following address:

CRASH DIVE CLUES
c/o Brian Moriarty
ANALOG Computing Magazine
P.O. Box 23
Worcester, Mass. 01603

BASIC Listing.

```
10 REM *** CRASH DIVE ***
20 TRAP 20:? "MAKE CASSETTE (0), OR DI
SK (1)";:INPUT DSK:IF DSK>1 THEN 20
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,128:PUT #1,31:CLOSE #1:EN
D
```

```
160 FOR X=1 TO 25:PUT #1,0:NEXT X:CLOS
E #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,128:P
UT #1,31:PUT #1,190:PUT #1,58:GOTO 210
200 ? "READY CASSETTE AND PRESS RETURN
";:OPEN #1,8,128,"C:":RESTORE 230:FOR
X=1 TO 40:READ N:PUT #1,N:NEXT X
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
230 DATA 0,55,88,31,127,31,169,0,141,4
7,2,169,60,141,2,211,169,0,141,231,2,1
33,14,169,56,141,232,2
240 DATA 133,15,169,128,133,10,169,31,
133,11,24,96
1000 DATA A2008E4402E886092065E44CB81F
7070707042403C90100290100290100202020202
0202901002029010020280B0,119
1010 DATA 1002020202020270418E1FA2FF9A
20B22AA20CA00620772AA906A035205D2AA219
A00720772AA917A035205D2A,527
1020 DATA A208A00920772AA922A035205D2A
A205A01020772AA93BA035205D2AA203A01220
772AA95AA035205D2AA9068D,277
1030 DATA 2C3C8D2D3CA9228D2F02207C2AAD
1FD0C907F0F9AE1FD0E007D0F9C906F00AC903
F00C20802A4C142020E92F4C,59
1040 DATA 9B20A205A01420772AA97DA03520
5D2A207C2A20252BC944F00DC943F00F20382E
20802A4C472020422E4C6420,682
1050 DATA 20492EA9039D4203A9049D4A03A9
009D4B032056E430DAA210A9C09D4403A93A9D
4503A9489D4803A9019D4903,753
1060 DATA A9079D42032056E430BA20382E20
B22AA98E8D3002A91F8D3102A99CA035205D2A
A90C8553A9C1A035205D2AA9,28
1070 DATA D7A035205D2A20152B20152B2015
2B20152B201D2BA903A036205D2A201D2BA924
A036205D2A20152B20152BA9,715
1080 DATA 278553A9C48DC002A9308D01D0A9
CA8D02D0A9408D03D0A2FF8E0ED08E0FD08E10
D0E88E04D0E88E6F02A9038D,442
1090 DATA 89D08D0AD08D0BD0A9F08581A91E
8582A90D8552A085A22CA906205CE4A9318D00
02A92A8D0102A9C08D0ED4AD,551
1100 DATA C23A4CA823A2FF9A20BC2CEEC03A
D003EEC13AADC23AD01BA92020D82DF01420B2
2AA208A00320772AA928A03A,724
1110 DATA 205D2A4C192EADC03AC920D01EAD
C13AD019ADC73AD01420B22AA207A00320772A
A96DA039205D2A4C192EADCB,209
1120 DATA 3AF00ACED23AD005A9008DCB3AAD
CC3AF020A91C20D82DF019ADCB3AD01420B22A
A208A00320772AA911A03920,112
1130 DATA 5D2A4C192EADD33A101420B22AA2
09A00320772AA900A03A205D2A4C192EADC23A
C912D00DA90420EB2DD006CE,122
1140 DATA D33A4C0022A9018DD33AADCE3AD0
35AD0AD229F88DC33AAD0AD229F88DC43AADC9
3AF020ADC73A18D869088DC7,759
1150 DATA 3A10148DCE3A20112DA955A03820
5D2AA917A03A205D2AA9228D2F0220E92B2011
2DA684E001D027AD0006A208,480
1160 DATA DDD62EF010CA10F820092DA94DA0
36205D2A4CD02EBD53238591BD5C2385926C91
00A920858A858BA201BD0006,261
1170 DATA C920F012E8E48490F420092DA959
A036205D2A4CD02E868DE002900 2A202BD0006
9589CA10F8A200868E868CA0,417
1180 DATA 00B98900DDDF2ED008E8C8C00390
F2B00DE68EA68CE8E8E8E05A90E1B0BEA58EC9
1CD00AA92DA037205D2A4CD0,704
1190 DATA 2EA68DE8A000BD0006998900E8C8
C00390F4A200868F868CA000B98900DD6E2FD0
08E8C8C00390F2B018E68FA6,490
1200 DATA 8CE8E8E8E07E90E120092DA962A0
36205D2A4CD02EA58E85A90AAABD362F8591E8
BD362F8592A58F85AAC927F0,358
1210 DATA 0DC928F013AABDE23B85906C9100
A941A037205D2A4CD02EA956A037205D2A4CD0
2E8F8F8F8F8F8F65FE712323,354
1220 DATA 2323232323232400C20752AA98F
A036205D2AA222A00C20772A8680207C2A2025
2BC959F008A20C209E2A4CD0,17
1230 DATA 2E4CB81FB593100AA97DA036205D
2A4CD02ED8859F207C24A59F8DC23A20DC23A0
05B1A0999300B1A299990088,0
1240 DATA 10F3A201209E2AA00120752AAEC2
3ABD3E32BC5432205D2A205F2D202A2D20972D
4CC92D0A85900A1865908590,906
1250 DATA 1869DA85A0A93A690085A118A590
695E85A2A93B690085A360207C2420B22AA205
A00B20772AA966A03A205D2A,355
1260 DATA A9228D2F02207C2A20252BC944F0
0DC943F00F20382E20802A4C1A2420422E4C37
2420492EA9039D4203A9089D,771
1270 DATA 4A03A9009D4B032056E430DAA210
A9C09D4403A93A9D4503A9489D4803A9019D49
03A90B9D42032056E430BA20,981
1280 DATA 382E4C9B20A5A9858EA5AA858F4C
1923ADC23A20DC23A005B9930091A0B9990091
A28810F360A58FC916B034C,138
1290 DATA 572EA59020D82DD0034C8D2EA590
20EB2DF0034C842EA590C914D00AA900A03820
5D2A4CCA2EC92AD027ADC23A,4
1300 DATA D022A91B20D82DF00AA941A03A20
5D2A4CCA2EA90720EB2DD00AA985A03A205D2A
4CCA2E20D62DF0034C722EA6,581
1310 DATA A5A4A4B59999D43AA9FF9599205F
2D20972D4CC92D20D82DF0034C962EADC23AC9
11F01820E92DF0034C7B2EA6,958
1320 DATA A5A4A4B9D43A9599A9FF99D43AD0
D0207925D0E3A4A4A6A5B9D43A9D5E3BA9FF99
D43AA988A039205D2AA590C9,566
1330 DATA 2AF009205F2D20972D4CCA2EADCD
3AD0F2A926A2058DCD3A9DE23BA26C9D5E3B20
7925A6A5A9159D5E3BD0D7A2,873
1340 DATA 6DBD5E3B3007E8E07190F68A6086
A5A90060C916B0034C572EC91CF00EC920F00A
C92AF0034C692E4C92244C08,768
1350 DATA 2520EB2DF00AA59020D82DF0034C
842EA590C909D026ADD13AF0034CB12E20E92D
F0034C7B2EA993A037205D2A,709
1360 DATA A9188DD13AA6A59599205F2D4CCA
2EC90FD0152044DAADC73A85D420FB2DA9DBA0
38205D2A4CCA2EC911D04BAD,74
1370 DATA C23AC90BF008ADC33AAEC43AD006
ADC53AAEC63A85A786A8A9D3A038205D2AA211
A00920772A2044DAA5A785D4,613
1380 DATA 20FB2DA9D7A038205D2AA211A00A
20772A2044DAA5A885D420FB2DD051C920D026
ADD03AF0034CB12E20E92DF0,873
1390 DATA 034C7B2EA993A037205D2AA9218D
D03AA6A59599205F2D4CCA2EC915D017ADCF3A
D009A9C8A038205D2AD015A9,815
1400 DATA BAA038205D2AD00C0AAABD9F26E8
BC9F26205D2A4CCA2EA8373F398437C937DA37
84373F39DA37183884378437,371
1410 DATA 843784378437DA378437B6378437
84375B3800388437843 77D387D388437953884
37843784377D387D38843784,576
1420 DATA 3784378437B63784378437953 8DA
37843774380A209DD0527F006CA10F84C572E4C
A8250224080F111718 1A1E1F,303
1430 DATA 20D82DD0034CA82EA59020EB2DF0
034C842EA590C90AD02AA203ADC83AF012A900
8DC83AA9029DE23B8599205F,433
1440 DATA 2D4CCA2EAD5F3BC907F0034CBA2E
8EC83AA924D0E4C90BD021AEC93AF00EA9008D
C93AA9F3A038205D2AD00BE8,884
1450 DATA 8EC93AA902A039205D2A4CCA2EC9
0CD00DADC53AD81869088DC53A4C0F26C90DD0
0DADC63A38D8E9088DC63A4C,754
1460 DATA 0F26C90ED03BADC53ACDC33AD033
ADC63ACDC43AD02BADCA3AF02620B22AA90E8D
C6028DC802A9008DC502A20C,86
1470 DATA A00B20772AA955A03A205D2AA922
8D2F024CD4274CBA2EC90690034C572E20EB2D
F0034C842EA590D014A9228D,370
1480 DATA CC3A8599A9048598205F2D202A2D
4CC92DC901D009A93FA039205D2AD031C903D0
09A9C9A037205D2AD024C905,904
1490 DATA D009A9C7A036205D2AD017C922F0
0CC923F008C925F004C927D00AA953A039205D
2A4CCA2E4C572EA91520D82D,664
1500 DATA F0034CC32EA59020EB2DF00AA590
20D82DF0034C842EADCF3AD00AA9C8A038205D
2A4CCA2EA955A038205D2AA9,732
1510 DATA 008DCF3AA590C901D01EA99FA039
205D2AA9238599A2019DE23BE89DE23BA90385
96205F2D202A2D4CCA2E20D8,686
```

```
1520 DATA 2DF0034C962EA590C91AD00AA9AF
A039205D2A4CCA2EC918F0034C692EADC23AC9
13D0F6A9278599A2069DE23B,950
1530 DATA A9158594205F2D202A2D4CC92D20
EB2DF0034C842EA58FC91890034CA82EA590C9
14D0034C9224C903F0034C57,422
1540 DATA 2EA91D20D82DD00FA9258599A204
9DE23BA9118594D0C0A91620D82DD00AA9C4A0
39205D2A4CCA2E4CC32EA58F,549
1550 DATA C926F0034C692EADCB3AD00BA909
8DCB3A8DD23A4CC92D4C8D2EA92120D82DF003
4CC32EA59020EB2DF0034C84,612
1560 DATA 2EA590C901D00AA9DBA039205D2A
4CCA2EC906F0034C572EA9288599A2079DE23B
8DCA3A4CD52820EB2DF0034C,609
1570 DATA 842EA590C907F0034C572EA91920
D82DD02FADC83AF01420B22AA209A00320772A
A9E9A039205D2A4C192EA929,111
1580 DATA 859AA2089DE23BA9028D883BA203
9DE23B4CD5284CC32E20D82DF0034C962EA590
C91EF0034C572EA4A4A9FF99,475
1590 DATA D43AA941A038205D2AA91420EB2D
D00FA6A5A92A9599A2169DE23BE89DE23B205F
2D20972D4CCA2E20D82DD003,458
1600 DATA 4CA82EA91E20D82DF0034CC32EA5
9020EB2DF0BA4C842EC91CF007C920F0034C57
2E4C9224488A489848A685BD,514
1610 DATA 4F2ABC562A8D0AD48D18D08C0DD0
E68568A868AA68407060706070006000000000
00F000A2008D44038C4503A9,744
1620 DATA 098D4203A97F8D48038E49034C56
E4A20D86558454604A919D002A9648D00D2A9AA
8D01D2A9008514A514C905D0,425
1630 DATA FAA2008E01D2CA8EFC0260BDC32B
8586BDD62B8587A018A9009186B810FB60A940
8D0ED4856AA200A90C8D4203,815
1640 DATA 2056E4A2008E4B03A9038D4203A9
F48D4403A9348D4503A90C8D4A032056E4A900
8D00D48D2F028D0DD08D0ED0,166
1650 DATA 8D0FD08D10D0A2089DC002CA10FA
A90E8DC502A9748DC402A2018EF002E88652A9
7085108D0ED260A9EDA03520,568
1660 DATA 5D2A60A9F8A0352052D2A60ADFC02
C9FFF0F9A8A2FF8EFC0229C0F00620802A4C25
2B98A20DDD712BF0F2CA10F8,534
1670 DATA B97F2BC920F010C99BF00CC97EF0
08C96190DCD838E920A07F8483A4838C1FD0A2
08CA10FDC68310F2601C2C27,419
1680 DATA 3C36370F2002222606070E6C6A3B
8A8B6B2B2A6F8070759B692D3D7680638C8D62
787A34803361B3532312C20,818
1690 DATA 2E6E806D2F81728065797F747771
398030377E383C3E6668648082677361204000
6075759DC5ED153D658DB5DD,540
1700 DATA 052D557DA5CDF51D3C3C3C3C3C3D
3D3D3D3D3E3E3E3E3E3E3FD8A218A9209D
0006CA10FAA00C20752AA2FF,621
1710 DATA 86808EFC02E8868420252BC920F0
08C97EF004C99BD0062080204CF42B20FB2C20
56E4E68420252BC99BF032C9,70
1720 DATA 7ED010C68430E320FB2C2056E4A5
84F0BBD0E520FB2C2056E4E684A584C91890D7
20802A20252BC99BF006C97E,18
1730 DATA F0D4D0F020FB2C86802056E4A018
18B92D3E99053E2A2A2A2A2903AAB9053E291F
1DBF2B990006A900992D3E88,266
1740 DATA 10DF60A9008585A580F029D8A555
0A0A1869308D00D0AD2B02F00BA9F08D5B2A85
81A93C8582A581C682D008A0,526
1750 DATA 1E848249F085818D5B2A4C5FE4A9
B085AB85AC85ADA99B85AFADC03A85D4ADC13A
85D520AAD920E6D8D8A0FFC8,224
1760 DATA B1F310FBA203B1F3098095ABCA88
10F6A221A00020772AA9ABA000205D2A60A20B
8E4203A2008E48038E490360,941
1770 DATA A945A0362052D2A60A209209E2AE8
209E2AA0094C752AA20D209E2AE8E01390F860
A202209E2AA000A20086A6B5,496
1780 DATA 93300ABD592D999D3CE6A6C8C8E8
E00690EDA5A6D00CA00220752AA90CA039205D
2A602E33253735 24A203209E,231
1790 DATA 2AE8E00990F8A00320752AA20086
A6868CB599300CE6A6AABD9E34BCC934205D2A
A68CE8E00690E7A5A6D007A9,45
1800 DATA 25A037205D2A60201F2DA00D2075
2AA20086A6868CBDD43A300CE6A6AABD9E34BC
C934205D2AA68CE8E00690E6,2
1810 DATA A5A6D007A925A037205D2A602011
2DA9A4A036205D2A4CCA2EA9FFA205DDD43AF0
05CA10F88A6086A4A90060A9,123
1820 DATA FFA205D599F005CA10F98A6086A5
A9006020AAD920E6D8D8A0FFC8B1F310FB297F
91F3C8A99B91F3A5F3A4F44C,196
1830 DATA 5D2AA20AA00520772AA92AA03920
5D2AA218A00720772AA960A039205D2A4CE81F
A210A90C9D42034C56E4A9FA,384
1840 DATA A0344C4D2EA9F7A034A2109D4403
989D450360A96BA036205D2AD06AA9C7A03620
5D2AD061A902A037205D2AD0,92
1850 DATA 58A9D9A036205D2AD055A9EDA036
205D2AD04CA9BCA036205D2AD043A9A9A03620
5D2AD03AA913A037205D2AD0,87
1860 DATA 31A96BA037205D2AD022A978A037
205D2AD019A984A037205D2AD010A9E3A03820
5D2AD007A9A4A038205D2A20,589
1870 DATA 7C2A4C47212080 2A4C3A224E5345
57554451584154414B474554505 54C44524F52
454D4C4F4F45584153454152,910
1880 DATA 4541505553505 2454F5045434C4F
5553454B494C53484F464952425245534D4149
4E53554E53484F4C554E4C43,348
1890 DATA 5554504F554C554247524557454 1
474F2092249224C32E08258C25A825A825A825
F5260F270F27DA27A82E692E,68
1900 DATA C32E45284528A82EA82EA228DE28
292945297F29C929072A072A232A484154444F
4F4C4F43534341475241 5452,464
1910 DATA 4141495253574943414253494743
4150475245524544474F4C53494C5748494741
555045525343 4F4449534455,818
1920 DATA 43534C4F424F4C554E4950495353
43524E4F5449442043555443415257524 54D41
534741534B4E495348414D41,182
1930 DATA 4E5355494B4559425245425554 49
4E56A213A9009DC03ACA10FAA200A9FF9D5E3B
9DDA3AE8E08490F5A2059DD4,490
1940 DATA 3A95999593CA10F6A2018ECF3A8E
C23AA9208DC53AA9A88DC63AA225BD48309DE2
3BCA10F7A227BD6E30BC9630,254
1950 DATA 99DA3ACA10F4A21FBDBE30BCDE30
995E3BCA10F46000010102030405060708090A
0B0C0D0E0F10101112131414,334
1960 DATA 151617181911A1B1C1C1D1E1F2021
02000404020601 0F0604090705060906 0A0813
09150F0D0C0F0C13100E040F,1
1970 DATA 0E130F1412091313 0B010C0D1418
191C1D2024252627 2D323638393B3F47494A51
565A5B5C5D5E63666E727475,366
1980 DATA 767B7E8214070016080917011902
0A0F0B10111F110C0D1B1C031E1A1D12040513
0E2006000106070C1213181E,480
1990 DATA 2A2B3031363C3D4243444484E5455
5A60666C727374787E536F6E617220737068 65
72659B4573636170652074 75,551
2000 DATA 62659B4163636573732074756E6E
656C9B4361707461696E27732071756172746 5
72739B466F72776172642070,149
2010 DATA 617373616765 9B526164696F2072
6F6F6D9B4C6F6E6720636F727269646F729B53
6F6E6172207374617469 6F6E,831
2020 DATA 9B42616C6C61737420636F6E7472
6F6C9B436F6D6D616E642073746174696F6E9B
4E617669676174696F6E2063,480
2030 DATA 656E7465729B557070657220 6D69
7373696C65206261799B546F727065646F2072
6F6F6D9B576561706F6E7320,124
2040 DATA 6C6F636B65729B53686F77657220
7374616C6C739B437265772773207175617274
6572739B47616C6C65799B56,873
2050 DATA 656E74696C6174696F6E20647563
749B46616E20726F6F6D9B4D697373696C6520
636F6E74726F6C9B45717569,566
2060 DATA 706D656E74206261799B4C6F7765
72206D697373696C65206261799BFE0B172538
4853616F7F8FA1B3C0CFDDED,642
2070 DATA F4050E1E2C3031313131313131313
3131313131313131313232323236C6F736564
206861746368 9B4C6F636B65,863
2080 DATA 6420646F6F729B426C616E6B2073
63616E6E65729B436C6F736564206772617465
9B547261697 46F7220776974,440
2090 DATA 6820706973746F6C9B436C6F7365
64206169726C6F636B9B4C6F636B6564206172
6D696E672073776974636 89B,37
```

```
2100 DATA 506F7765722063616626C659B5369
676E9B44656164206361707461696E9B477265
656E20627574746F6E9B5265,632
2110 DATA 6420627574746F6E9B476F6C6420
627574746F6E9B53696C766572206275747 46F
6E9B57686974652062757474,308
2120 DATA 6F6E9B44657074682067617567 65
9B5065726973636F70659B4469676974616C20
646973706C61799B44756374,33
2130 DATA 20646F776E20746F2066616E2072
6F6F6D9B536C6F7420696E206169726C6F636B
9B426F6C7465642D646F776E,417
2140 DATA 20736F6E617220756E69749B5069
73746F6C9B54696E792073637265776472697 6
65729B537569636964652 06E,80
2150 DATA 6F74659B536563757269747920 49
449B4361626C652063757474657273 9B436172
649B5772656E63689B476173,769
2160 DATA 206D61736B9B44756C6C206B6E69
66659B5368616D706F6F9B5461637469637320
6D616E75616C9B5261646961,400
2170 DATA 74696F6E2073756974 9B4B65799B
4F70656E2068617463689B4F70656E20646F6F
729B4163746976652073636 1,15
2180 DATA 6E6E65729B4F70656E2067726174
659B4465616420747261697 46F729B4F70656E
206169726C6F636B9B416374,661
2190 DATA 697661746564207377697463689B
536576657265642063616 26C659B526164696F
6163746976652073 6F6E6172,261
2200 DATA 20756E69749B6A7783919EB2C1D6
E2E7F4010C1826333F49596F7F969DAEBBC7D5
DAE1EAF5FD0C1B1F2A34434E,994
2210 DATA 5B6879873232323232323232323232
3233333333333333333333333333333333333333
3333333434343434343434343434,534
2220 DATA 3434453A9B433A9B44313A47414D
452E4441549B4272769616E204D6F7269617274
7927739B435241534820444 9,305
2230 DATA 56459B2843293139383420414E41
4C4F4720436F6D707574696E679B5072657373
20A0D3D4C1D2D4A020746F20,786
2240 DATA 706C6179206E65772067616D659B
5072657373 20A0CFD0D4C9CFCEA020746F2072
6573746F7265206F6C642067,850
2250 DATA 616D659B526573746F7265206672
6F6D20C469736B206F7220C361737365747465
3F9BA0C3D2C1D3C8A0C4C9D6,478
2260 DATA C5A1A0A8D4CDA9A0A0A0A0A0A0C5
D6C5CED4A0A3A0A0A0A0A0A09BA0CCCFC3C1D4
C9CFCEA020A0A0A0A0C5D8C9,405
2270 DATA D4D3A09BA0A0D6C9D3C9C2CCC5A0
20A0A0A0A0C9D4C5CDD3A09BA0A0A0A0A0A0A0
A0A0A09BA0A0A0A0A0D7C8C1,221
2280 DATA D4A09BA0A0C8C1D0D0C5CED3A020
A0A0A0A0A0D9CFD5D2A020A0D2C5D3D0CFCED3
C5A09BA0A0A0A0A0A0D9CFD5,241
2290 DATA A020A0A0A0A0A0A0C1D2C5A020A0
C3C1D2D2D9C9CEC7A09B53796E7461783A9B42
616420636F6D6D616E649B42,402
2300 DATA 6164207665726 29B426164206E6F
756E9B5468617427732069 6D706F737369626C
659B43616E277420676F2074,759
2310 DATA 6861742077617 99B547970652059
20746F207175697420676 16D653A9B4F6B6179
9B416C726561647920686F6C,156
2320 DATA 64696E672069749B49736E277420
6865726 59B43616E277420646F207468617420
7965749B596F757220617 26D,487
2330 DATA 7320617265206 6756C6C219B4E6F
7420656E6F75676820726F6F6D206865726 59B
4265206D6F726520737065 63,761
2340 DATA 6966696 39B596F7520646F6E2774
2068617665206974 9B4E6F7468696E679B5479
7065204E2053204520572055,874
2350 DATA 206F722044 9B52656665722 0746F
2069742062792063 6F6C6F729B547970652049
20666F7220696E76656E746F,64
2360 DATA 72799B446F65736E27742068656C
709B5768792062 6F746865723F9B5365656D73
206F7264696E6172799B596F,700
2370 DATA 7520666F756E6420736F6D657468
696E67219B4974277320616972746967687 49B
456E656D7920617070726F61,158
2380 DATA 6368696E67219B536372657765 64
20696E20706C6163659B4C6F6F6B732064616E
6765726F75739B4E65656473,705
2390 DATA 206865792074 6F20616374697661
74659B426F6C747320617265207469674 20
2620727573747 99B44414E47,919
2400 DATA 45523A2052616469617469 6F6E20
7A6F6E65219B427261696E7320626C6F776E20
6F75749B5368616D706F6F20,222
2410 DATA 616C6C207573656420757 09B4241
4E47219B41636365707473207365637572697 4
792049420636172649B4974,470
2420 DATA 20676C6F77739B5365652070686F
746F20696E20414E414C4F472023313 89B4163
65206F662053706164657321,307
2430 DATA 9B4561736965722073616 9642074
68616E20646F6E659B4F6E6C7920312062756C
6C65749B4E6F2062756C6C65,667
2440 DATA 74739B58203D9B59203D9B464154
484F4D539B4E6F7468696E672068617070656E
739B537562206C6576656C73,57
2450 DATA 206F66669B5375622064697665 73
9B4E6F6E659B4120636C6F7564206F6620706F
69736F6E6F7573206761739B,556
2460 DATA 6B696C6C7320796F7520696E7374
616E746C79219B4C6F636B2069732076657279
207365637572659B416C7265,51
2470 DATA 616479206F70656E9B796F752061
726520444541449B456E656D79206361707475
726573207468652073756220,300
2480 DATA 616E649B49742066616C6C732064
6F776E2074686520706970659B4C6F636B2064
657374726F796564219B5472,745
2490 DATA 792065786 16D696E696E67207468
696E67739B536372657764726976657227732 0
746F6F2074696E799B4B6579,326
2500 DATA 20776F6E2774206669749B41206A
6F6C74206F662068696768207 66F6C74616765
9B54726169746F722073686F,638
2510 DATA 6F747320796F7520616E649B5375
622068697473206 26F74746F6D219B4120626C
617374206F662072616496F,941
2520 DATA 6163746976697479 9B426F6C7473
20776F6E2774206C657420796F759B436F6E67
726174756C6174696F6E7321,539
2530 DATA 9B536176520676 16D6520746F20
C469736B206F7220C361737365747465 3F9B43
6F6E6E65637465642074 6F20,18
2540 DATA 6361626C659B0000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000,676
```

•

CHECKSUM DATA

(See p. 30)

```
10 DATA 205,351,496,811,423,729,200,60
3,555,573,694,613,29,205,197,6684
160 DATA 749,198,962,93,491,30,155,941
,287,88,522,600,216,516,706,6554
1060 DATA 900,534,157,861,15,983,935,8
81,220,633,824,685,180,903,116,8827
1210 DATA 944,700,783,728,657,542,707,
886,830,993,213,263,281,961,898,10386
1360 DATA 420,104,44,85,96,544,679,976
,333,222,350,953,865,898,28,6597
1510 DATA 911,24,703,994,156,61,875,18
2,961,52,490,763,912,733,907,8724
1660 DATA 951,36,677,622,150,85,45,928
,938,343,792,708,780,219,815,8089
1810 DATA 905,315,845,974,48,770,505,4
86,805,645,202,458,956,165,618,8697
1960 DATA 134,577,302,470,410,679,737,
552,510,779,876,285,582,825,549,8267
2110 DATA 598,676,767,688,464,638,859,
679,393,228,920,433,555,654,755,9307
2260 DATA 384,174,307,915,532,571,503,
510,508,671,692,543,607,411,645,7973
2410 DATA 278,309,691,821,709,697,376,
584,566,567,528,718,673,490,8007
```

•

(Assembly language listing starts next page.)

```
; ------------------------
; CRASH DIVE! (TM)
; by Brian Moriarty
; ANALOG Computing #18
; (C)1984 ANALOG Computing
; ------------------------
;
; MACRO DEFINITIONS
; -----------------
;
;         POSITION MACRO
;         --------------
;
; SYNTAX:
; POSITION xpos,ypos
;
        .MACRO POSITION
        .IF %0<>2 .OR %1>39 .OR %2>23
          .ERROR "POS parameters"
          .ELSE
          .IF %1=13
            LDY #%2
            JSR POSIT13
            .ELSE
            LDX #%1
            LDY #%2
            JSR POSIT
            .ENDIF
          .ENDIF
        .ENDM
;
;         PRINT MACRO
;         -----------
;
; SYNTAX:
; PRINT <addr of EOL-terminated string>
;
        .MACRO PRINT
        .IF %0<>1
          .ERROR "PRINTE parameters"
          .ELSE
          LDA # <%1
          LDY # >%1
          JSR EPRINT
          .ENDIF
        .ENDM
;
;         TEXT MACRO
;         ----------
;
; SYNTAX:
; TEXT <"string">
;
        .MACRO TEXT
        .IF %0<>1 .OR %1>127
          .ERROR "TEXT parameters"
          .ELSE
          .BYTE %*1,EOL
          .ENDIF
        .ENDM
;
; SYSTEM  EQUATES
; ---------------
;
;         ZERO-PAGE
;         ---------
;
BOOT?   =   $09     ; OS boot flag
POKMSK  =   $10     ; interrupt mask
RTCLOK  =   $14     ; system clock
LMARGN  =   $52     ; left margin
RMARGN  =   $53     ; right margin
ROWCRS  =   $54     ; cursor row
COLCRS  =   $55     ; cursor column
RAMTOP  =   $6A     ; # pages of RAM
FR0     =   $D4     ; floating point register
CIX     =   $F2     ; FP index register
INBUFF  =   $F3     ; FP pointer
;
;         PAGES 2-3
;         ---------
;
VDSLST  =   $0200   ; DLI vector
SRTIMR  =   $022B   ; key repeat timer
SDMCTL  =   $022F   ; DMA control
SDLSTL  =   $0230   ; D-list addr
COLDST  =   $0244   ; coldstart flag
GPRIOR  =   $026F   ; PM0 priority
PCOLR0  =   $02C0   ; player 0 color
PCOLR1  =   $02C1   ; player 1 color
PCOLR2  =   $02C2   ; player 2 color
PCOLR3  =   $02C3   ; player 3 color
COLOR0  =   $02C4   ; playfield 0 color
COLOR1  =   $02C5   ; playfield 1 color
COLOR2  =   $02C6   ; playfield 2 color
COLOR4  =   $02C8   ; background color
CRSINH  =   $02F0   ; cursor inhibit
CH      =   $02FC   ; keypress register
ICCOM   =   $0342   ; CIO command
ICBADR  =   $0344   ; CIO addr
ICBLEN  =   $0348   ; CIO length
ICAUX1  =   $034A   ; AUX byte 1
ICAUX2  =   $034B   ; AUX byte 2
;
;         CTIA/GTIA
;         ---------
;
HPOSP0  =   $D000   ; h-pos player 0
HPOSP1  =   $D001   ; "          " 1
HPOSP2  =   $D002   ; "          " 2
HPOSP3  =   $D003   ; "          " 3
SIZEP0  =   $D004   ; width player 0
SIZEP1  =   $D009   ; "           " 1
SIZEP2  =   $D00A   ; "           " 2
SIZEP3  =   $D00B   ; "           " 3
GRAFP0  =   $D00D   ; graphics player 0
GRAFP1  =   $D00E   ; "              " 1
GRAFP2  =   $D00F   ; "              " 2
GRAFP3  =   $D010   ; "              " 3
COLPF2  =   $D018   ; color register 2
CONSOL  =   $D01F   ; console keys
;
;         POKEY
;         -----
;
AUDF1   =   $D200   ; frequency channel 1
AUDC1   =   $D201   ; vol/dist channel 1
AUDCTL  =   $D208   ; audio control
RANDOM  =   $D20A   ; random # generator
IRQEN   =   $D20E   ; interrupt enable
;
;         ANTIC
;         -----
;
DMACTL  =   $D400   ; DMA control
WSYNC   =   $D40A   ; wait for horz sync
NMIEN   =   $D40E   ; NMI enable
;
;         FLOATING POINT
;         --------------
;
FASC    =   $D8E6   ; FP-to-ATASCII
IFP     =   $D9AA   ; integer-to-FP
ZFR0    =   $DA44   ; clear FR0
;
;         OS ROUTINES
;         -----------
;
CIOV    =   $E456   ; CIO entry
SETVBV  =   $E45C   ; set v-blank vector
SYSVBV  =   $E45F   ; OS VBI entry
SIOINV  =   $E463   ; SIO init
;
; INTERNAL EQUATES
; ----------------
;
;         MEMORY ALLOCATION
;         -----------------
;
INLINE  =   $0600   ; text input buffer
GAMEDATA =  $3AC0   ; working area
EDLIST  =   $3C20   ; start of E: display list
SCREEN  =   $3C40   ; start of screen RAM
;
; Game play database
;
        *=  GAMEDATA
;
EVENT   *=  *+2     ; event #
PLACE   *=  *+1     ; location
SLAT    *=  *+1     ; ship's latitude
SLON    *=  *+1     ; ship's longitude
MLAT    *=  *+1     ; missile latitude
MLON    *=  *+1     ; missile longitude
DEPTH   *=  *+1     ; sub's depth
GREEN   *=  *+1     ; green button on/off
RED     *=  *+1     ; red button on/off
SWITCH  *=  *+1     ; arming switch on/off
BREATH  *=  *+1     ; holding breath?
HATCH   *=  *+1     ; hatch opened?
UNIT    *=  *+1     ; unit dropped in rm 17?
CRASHED *=  *+1     ; sub crashed?
BULLET  *=  *+1     ; bullet used?
SUIT    *=  *+1     ; suit searched?
CAPTAIN *=  *+1     ; captain searched?
BHOLD   *=  *+1     ; breath holding counter
TRAITOR *=  *+1     ; traitor wait counter
HOLDING8S *= *+6    ; current inventory
VECTORS *=  *+132   ; current vector table
OBJECTS *=  *+132   ; current object table
NTRANS  *=  *+38    ; current translation matrix
;
;         MISCELLANEOUS
;         -------------
;
EOL     =   $9B
SPACE   =   $20
;
;         ZERO-PAGE
;         ---------
;
        *=  $80
;
; Cursor control
;
CURSEN  *=  *+1     ; cursor on/off flag
CSHAPE  *=  *+1     ; current cursor shape
BLINK   *=  *+1     ; cursor blink timer
;
; Keyboard handler
;
CLICK   *=  *+1     ; key click counter
LENGTH  *=  *+1     ; line length register
;
; DLI control
;
DLICOL  *=  *+1     ; DLI color index
;
; Screen clearing
;
CLPOINT *=  *+2     ; screen clear pointer
CINDEX  *=  *+1     ; window clear index
;
; Parser
;
PBUFF   *=  *+3     ; parsing buffer
PDEX    *=  *+1     ; scanning index
LBREAK  *=  *+1     ; pos. of space char
VCODE   *=  *+1     ; verb code #
UCODE   *=  *+1     ; untranslated noun code #
NCODE   *=  *+1     ; translated noun code #
DOVECT  *=  *+2     ; verb execution vector
CURVECT *=  *+6     ; current room vectors
CUROBJS *=  *+6     ; current room objects
NEWPLACE *= *+1     ; next room code #
VPNT    *=  *+2     ; addr of current v-buffer
OPNT    *=  *+2     ; addr of current o-buffer
INVPOS  *=  *+1     ; position in inventory buffer
ROOMPOS *=  *+1     ; position in room buffer
ANY?    *=  *+1     ; flag for empty room/inv
LATSHOW *=  *+1     ; latitude to display
LONSHOW *=  *+1     ; longitude to display
VLAST   *=  *+1     ; last verb
ULAST   *=  *+1     ; last noun
;
; Event counter
;
EBUFF   *=  *+5     ; printing buffer
;
;         INITIALIZATION
;         --------------
;
        *=  $1F80
;
ENTRY
        LDX #0
        STX COLDST
        INX         ; = 1
        STX BOOT?
        JSR SIOINV  ; for sound init
        JMP TITLE
;
;         CUSTOM DISPLAY LIST
;         -------------------
;
DLIST
        .BYTE $70,$70,$70,$70 ; blank 32 lines
        .BYTE $42, <SCREEN, >SCREEN ; text w/LMS
        .BYTE $90,$10 ; blank 4 w/DLI
        .BYTE $02     ; text (location)
        .BYTE $90,$10 ; blank 4 w/DLI
        .BYTE $02     ; text (exits)
        .BYTE $90,$10 ; blank 4 w/DLI
        .BYTE $02,$02,$02,$02,$02,$02 ; text (v-items)
        .BYTE $90,$10 ; blank 4 w/DLI
        .BYTE $02,$02 ; text (events)
        .BYTE $90,$10 ; blank 4 w/DLI
        .BYTE $02,$02 ; text (response)
        .BYTE $80,$80,$10 ; blank 4 w/DLIs
        .BYTE $02,$02,$02,$02,$02,$02 ; text (i-items)
        .BYTE $70     ; blank 8
        .BYTE $41, <DLIST, >DLIST ; JVB
;
;         TITLE SCREEN
;         ------------
;
TITLE
        LDX #$FF
        TXS
        JSR NEWSCREEN ; reset E:
          POSITION  12,6
          PRINT   T0  ; "Brian Moriarty's"
          POSITION  25,7
          PRINT   T1  ; "CRASH DIVE"
          POSITION  8,9
          PRINT   T2  ; "(C)1984 ANALOG Computing"
;
STARTOP
          POSITION  5,16
          PRINT   T3  ; "Press  START  to play new game"
          POSITION  3,18
          PRINT   T4  ; "Press  OPTION  to restore old game"
;
        LDA #6
        STA EDLIST+12 ; modify
        STA EDLIST+13 ; display list
;
        LDA #$22
        STA SDMCTL  ; restore screen
        JSR BEEP
;
; Wait for selection
;
POLL
        LDA CONSOL
        CMP #7      ; scan until a
        BEQ POLL    ; key is pressed
LET80
        LDX CONSOL
        CPX #7
        BNE LET80
        CMP #6      ; START pressed?
        BEQ NEWGAME ; yes, do a new game
        CMP #3      ; OPTION pressed?
        BEQ RESTORE ; yes, restore old game
BADPOLL
        JSR BOOP    ; else
        JMP POLL    ; resume scan
;
; Start new game
;
```

```
NEWGAME
        JSR INITDATA
        JMP PLAYSCREEN
;
; Restore a previous game
;
RESTORE
         POSITION  5,20
         PRINT   T5  ; "Restore from Disk or Tape?"
        JSR BEEP
DORT
        JSR GETKEY
        CMP #'D     ; disk?
        BEQ GETDISK
        CMP #'C     ; tape?
        BEQ GETAPE
DORTERR
        JSR CLOSE1
        JSR BOOP
        JMP DORT
;
; Get game from disk
;
GETDISK
        JSR DPOINT
        JMP READOPEN
;
; Get game from tape
;
GETAPE
        JSR TPOINT
;
READOPEN
        LDA #3      ; OPEN command
        STA ICCOM,X
        LDA #4      ; READ
        STA ICAUX1,X
        LDA #0
        STA ICAUX2,X
        JSR CIOV
        BMI DORTERR
;
; Get game data thru IOCB #1
;
GETDATA
        LDX #$10
        LDA # <GAMEDATA
        STA ICBADR,X
        LDA # >GAMEDATA
        STA ICBADR+1,X
        LDA #$48
        STA ICBLEN,X
        LDA #$01
        STA ICBLEN+1,X
        LDA #7      ; GET RECORD
        STA ICCOM,X
        JSR CIOV
        BMI DORTERR
        JSR CLOSE1
;
;         PLAYSCREEN INIT
;         ---------------
PLAYSCREEN
        JSR NEWSCREEN ; reset E:
;
; Enable custom display list
;
        LDA # <DLIST
        STA SDLSTL
        LDA # >DLIST
        STA SDLSTL+1
;
; Print title & labels
;
         PRINT  T6  ; "CRASH DIVE! (TM) EVENT #"
        LDA #12     ; set right margin
        STA RMARGN  ; for wraparound
         PRINT  T7  ; "LOCATION/EXITS"
         PRINT  T8  ; "VISIBLE ITEMS"
        JSR BAR
        JSR BAR
        JSR BAR
        JSR BAR     ; 4 blank bars
        JSR SAYWHAT ; "WHAT"
         PRINT  T11 ; "HAPPENS/YOUR RESPONSE"
        JSR SAYWHAT ; "WHAT"
         PRINT  T12 ; "YOU ARE CARRYING"
        JSR BAR
        JSR BAR     ; more blank bars
        LDA #39
        STA RMARGN  ; reset
;
        LDA #$C4    ; green
        STA PCOLR0  ; cursor
;
; Setup P/M borders & mask
;
        LDA #48
        STA HPOSP1  ; position left
        LDA #202    ; and
        STA HPOSP2  ; right borders
        LDA #64     ; and
        STA HPOSP3  ; title cover
        LDX #255
        STX GRAFP1  ; set up
        STX GRAFP2  ; side borders and
        STX GRAFP3  ; title mask
        INX         ; = 0
        STX SIZEP0  ; set cursor width
        INX         ; = 1
        STX GPRIOR  ; set player priority
        LDA #3
        STA SIZEP1  ; set border
        STA SIZEP2  ; and mask
        STA SIZEP3  ; widths
;
        LDA #$F0    ; init
        STA CSHAPE  ; cursor shape
        LDA #30     ; and
        STA BLINK   ; blink timer
;
        LDA #13
        STA LMARGN  ; init left margin
;
        LDY # <IMMVBI ; set VBI vector,
        LDX # >IMMVBI ; specifying
        LDA #6      ; immediate-mode
        JSR SETVBV
;
        LDA # <DLI
        STA VDSLST  ; tell OS where
        LDA # >DLI  ; we put our
        STA VDSLST+1 ; DLI service routine
        LDA #$C0    ; set bits 6 & 7 of NMIEN
        STA NMIEN   ; to enable DLIs and VBIs
;
        LDA PLACE
        JMP REENTRY
;
;         EVENT GENERATOR
;         ---------------
;
NEXTEVENT
        LDX #$FF    ; empty stack
        TXS         ; just in case!
        JSR SHOWEV  ; show event #
        INC EVENT   ; update
        BNE RADIO   ; event
        INC EVENT+1 ; counter
;
; Handle room 0
;
RADIO
        LDA PLACE
        BNE ENEMY
        LDA #32     ; got suit?
        JSR OWNIT?
        BEQ ENEMY
        JSR NEWSCREEN
         POSITION  8,3
         PRINT  T79 ; "A blast of radioactivity"
        JMP KILLS
;
; Check for enemy takeover
;
ENEMY
        LDA EVENT
        CMP #32
        BNE LUNGS
        LDA EVENT+1
        BNE LUNGS
        LDA DEPTH
        BNE LUNGS
        JSR NEWSCREEN
         POSITION  7,3
         PRINT  T67 ; "Enemy captures the sub and"
        JMP KILLS
;
; Limit breath-holding
;
LUNGS
        LDA BREATH  ; holding it?
        BEQ POISON  ; no - skip this
        DEC BHOLD   ; else decrement count
        BNE POISON  ; continue if NZ
        LDA #0      ; else
        STA BREATH  ; release hold
;
; Handle poison atmosphere
;
POISON
        LDA HATCH   ; hatch opened?
        BEQ DOTRAIT ; not yet
        LDA #28     ; got the
        JSR OWNIT?  ; mask?
        BEQ DOTRAIT ; yup - you're safe
        LDA BREATH  ; holding breath?
        BNE DOTRAIT ; good thing, otherwise ...
        JSR NEWSCREEN
         POSITION  8,3
         PRINT  T62 ; "A cloud of poisonous gas"
        JMP KILLS
;
; Handle traitor
;
DOTRAIT
        LDA TRAITOR
        BPL NTRAIT
        JSR NEWSCREEN
         POSITION  9,3
         PRINT  T77 ; "Traitor shoots you and"
        JMP KILLS
NTRAIT
        LDA PLACE   ; is this
        CMP #18     ; room 18?
        BNE TRESET  ; we're safe
        LDA #4      ; is the traitor
        JSR INROOM? ; lurking?
        BNE TRESET  ; thankfully not
        DEC TRAITOR ; else reduce wait time
        JMP DOCRASH
TRESET
        LDA #1
        STA TRAITOR
;
; Change sub coords, handle crash
;
DOCRASH
        LDA CRASHED
        BNE PARSER
        LDA RANDOM
        AND #$F8
        STA SLAT
        LDA RANDOM
        AND #$F8
        STA SLON
        LDA RED
        BEQ PARSER
        LDA DEPTH
        CLC
        CLD
        ADC #8
        STA DEPTH
        BPL PARSER
        STA CRASHED
        JSR CLWH
         PRINT  T45 ; "BANG!"
         PRINT  T78 ; "Sub hits bottom!"
;
;         INPUT PARSER
;         ------------
;
PARSER
        LDA #$22
        STA SDMCTL
        JSR GETLINE ; put line into INLINE
        JSR CLWH
        LDX LENGTH
        CPX #1      ; if length is 1
        BNE DOCLAUSE ; check for legality
;
; Check for a legal single-char command
;
        LDA INLINE  ; get the character
        LDX #8      ; init search index
LEGSING
        CMP SCOMS,X
        BEQ EXSING  ; matched! go do it
        DEX         ; otherwise
        BPL LEGSING ; keep searching
        JSR SYNERR  ; error, so print
         PRINT  T14 ; "Invalid command"
        JMP BADPARSE ; and try again
;
; Execute a single-char command
;
EXSING
        LDA SVECTL,X ; fetch the lsb
        STA DOVECT  ; and
        LDA SVECTH,X ; msb of the
        STA DOVECT+1 ; execution addr
        JMP (DOVECT) ; and do it!
;
; Find the 1st space character
; in the user's response
;
DOCLAUSE
        LDA #SPACE
        STA PBUFF+1
        STA PBUFF+2
        LDX #1
FIND1
        LDA INLINE,X ; length is in X
        CMP #SPACE  ; is it a space?
        BEQ ENDV    ; yes!
        INX         ; else keep scanning
        CPX LENGTH
        BCC FIND1
BADVERB
        JSR SYNERR  ; verb is no good, so print
         PRINT  T15 ; "Verb not recognized"
        JMP BADPARSE ; and try again
;
; Space char found, so record its
; position and move the first half
; of the clause into the parsing buffer
;
ENDV
        STX LBREAK
        CPX #2
        BCC TOVB
        LDX #2
TOVB
```

```
        LDA INLINE,X
        STA PBUFF,X
        DEX
        BPL TOVB
;
; Check for a legal verb
;
        LDX #0      ; init verb index
        STX VCODE
VNEXT
        STX PDEX
        LDY #0      ; init buffer char index
VSCAN
        LDA PBUFF,Y ; get a char from buffer
        CMP VERBS,X ; match?
        BNE NEXTRY  ; nope - try another verb
        INX
        INY
        CPY #3
        BCC VSCAN   ; if all 3 chars match
        BCS LEGALV  ; the verb is legal
NEXTRY
        INC VCODE
        LDX PDEX
        INX
        INX
        INX
        CPX #NV*3+3 ; out of verbs?
        BCC VNEXT   ; nope - keep scanning
        BCS BADVERB ; else verb is worthless
;
; Verb is legal, so fetch its execution
; vector
;
LEGALV
        LDA VCODE   ; fetch verb #
        CMP #28     ; if it's 80,
        BNE MOVE2   ; warn user:
         PRINT  T29 ; "Type N S E W U or D"
        JMP BADPARSE
;
; Move the second half of the
; clause into the parsing buffer
;
MOVE2
        LDX LBREAK  ; fetch pos. of space char
        INX         ; plus 1
        LDY #0      ; init buffer char index
MOVEN
        LDA INLINE,X ; fetch character
        STA PBUFF,Y ; stuff into buffer
        INX
        INY
        CPY #3      ; until 3 characters
        BCC MOVEN   ; have been moved
;
; Check for a legal noun
;
        LDX #0      ; init noun index
        STX UCODE
NNEXT
        STX PDEX
        LDY #0      ; init buffer char index
NSCAN
        LDA PBUFF,Y ; get a char from buffer
        CMP NOUNS,X ; match?
        BNE NEXTRY2 ; nope - try another noun
        INX
        INY
        CPY #3
        BCC NSCAN   ; if all 3 chars match
        BCS LEGALN  ; the noun is legal
NEXTRY2
        INC UCODE
        LDX PDEX
        INX
        INX
        INX
        CPX #NN*3+3 ; out of nouns?
        BCC NNEXT   ; nope - keep scanning
        JSR SYNERR  ; else noun is garbage
         PRINT  T16 ; "Noun not in vocabulary"
        JMP BADPARSE
;
; Noun's code # is in UCODE;
; verb's code # is in VCODE;
; verb execution addr is in DOVECT
;
LEGALN
        LDA VCODE   ; fetch
        STA VLAST
        ASL A       ; execution addr
        TAX         ; and
        LDA VVECTS,X ; save it
        STA DOVECT  ; in DOVECT
        INX
        LDA VVECTS,X
        STA DOVECT+1
        LDA UCODE
        STA ULAST
        CMP #39     ; was it BUTTON?
        BEQ DOBUTT
        CMP #40     ; was it INVENTORY?
        BEQ DOINVE
        TAX
        LDA NTRANS,X ; translate noun
        STA NCODE
        JMP (DOVECT) ; execute verb
;
; Handle BUTTON
;
DOBUTT
         PRINT  T30 ; "Refer to it by color"
        JMP BADPARSE
;
; Handle INVENTORY
;
DOINVE
         PRINT  T31 ; "Type I for inventory"
        JMP BADPARSE
;
;        EXECUTE SINGLE-CHAR COMMANDS
;        ----------------------------
;
;        COMMAND VECTOR TABLES
;        ---------------------
;
SVECTL
        .BYTE  <DOM, <DOM, <DOM, <DOM, <DOM
        .BYTE  <DOM, <DOQ, <DOX, <DOA
SVECTH
        .BYTE  >DOM, >DOM, >DOM, >DOM, >DOM
        .BYTE  >DOM, >DOQ, >DOX, >DOA
;
;        HANDLE "Q" (QUIT)
;        -----------------
;
DOQ
         POSITION  13,12
         PRINT  T19 ; "Type Y to quit game:"
         POSITION  34,12
        STX CURSEN  ; enable cursor
        JSR BEEP
        JSR GETKEY
        CMP #'Y
        BEQ DOQUIT
        LDX #12
        JSR ERASE
        JMP BADPARSE
DOQUIT
        JMP TITLE
;
;        HANDLE MOVEMENT
;        ---------------
;
; ENTRY: Vector (0-5) in X
;
DOM
        LDA CURVECT,X
        BPL EXMOVE
CANTGO
         PRINT  T18 ; "You can't go that way."
        JMP BADPARSE
EXMOVE
        CLD         ; for safety
        STA NEWPLACE ; save destination
        JSR SAVELOC ; save status
        LDA NEWPLACE ; get destination,
        STA PLACE   ; make it current, and
REENTRY
        JSR BPOINT  ; point to the new buffers
;
; Get new buffer data
;
        LDY #5
RLOOP
        LDA (VPNT),Y
        STA CURVECT,Y
        LDA (OPNT),Y
        STA CUROBJS,Y
        DEY
        BPL RLOOP
;
; Refresh screen
;
SHOWPLACE
        LDX #1
        JSR ERASE   ; clear location window
         POSITION  13,1
        LDX PLACE   ; get loc #
        LDA RDLS,X  ; fetch lsb and
        LDY RDHS,X  ; msb of text addr and
        JSR EPRINT  ; print it
;
        JSR SHOWVIS ; display visible items
        JSR SHOWVECTS ; display new vectors
        JSR SHOWINV ; show inventory
        JMP POKAY   ; congratulations!
;
;        POINT TO NEW BUFFERS
;        --------------------
;
; ENTRY: Buffer # (0-23) in A
;
BPOINT
        ASL A       ; * 2
        STA NCODE   ; save it
        ASL A       ; * 4
        CLC
        ADC NCODE   ; *2 + *4 = *6
        STA NCODE   ; save it
        CLC
        ADC # <VECTORS
        STA VPNT
        LDA # >VECTORS
        ADC #0
        STA VPNT+1
        CLC
        LDA NCODE
        ADC # <OBJECTS
        STA OPNT
        LDA # >OBJECTS
        ADC #0
        STA OPNT+1
        RTS
;
;        HANDLE "X" (SAVE GAME)
;        ----------------------
;
DOX
        JSR SAVELOC ; save current status
        JSR NEWSCREEN
         POSITION  5,11
         PRINT  T82 ; "Save game to Disk or Cassette?"
        LDA #$22
        STA SDMCTL
        JSR BEEP
SAVEPOLL
        JSR GETKEY
        CMP #'D
        BEQ DSAVE
        CMP #'C
        BEQ CSAVE
BADWRITE
        JSR CLOSE1
        JSR BOOP
        JMP SAVEPOLL
;
; Save to disk
;
DSAVE
        JSR DPOINT
        JMP GSAVE
;
; Save to cassette
;
CSAVE
        JSR TPOINT
;
GSAVE
        LDA #3
        STA ICCOM,X
        LDA #8
        STA ICAUX1,X
        LDA #0
        STA ICAUX2,X
        JSR CIOV
        BMI BADWRITE
;
; Write out game data
;
WRITE
        LDX #$10
        LDA # <GAMEDATA
        STA ICBADR,X
        LDA # >GAMEDATA
        STA ICBADR+1,X
        LDA #$48
        STA ICBLEN,X
        LDA #$01
        STA ICBLEN+1,X
        LDA #11
        STA ICCOM,X
        JSR CIOV
        BMI BADWRITE
        JSR CLOSE1
        JMP PLAYSCREEN
;
;        HANDLE "A" (AGAIN)
;        ------------------
;
DOA
        LDA VLAST   ; restore old verb
        STA VCODE
        LDA ULAST   ; and noun
        STA UCODE
        JMP LEGALN  ; and do it again!
;
;        SAVE LOC STATUS
;        ---------------
;
SAVELOC
        LDA PLACE
        JSR BPOINT
        LDY #5
SLOOP
        LDA CURVECT,Y
        STA (VPNT),Y
        LDA CUROBJS,Y
        STA (OPNT),Y
        DEY
        BPL SLOOP
        RTS
;
;        VERB EXECUTORS
;        --------------
;
; ENTRY: Translated noun code in A
;        and in NCODE;
;        untranslated code in UCODE
```

```
;
;         TAKE
;         ----
DOTAKE
        LDA UCODE
        CMP #22       ; is it moveable?
        BCS DT0       ; yes
        JMP IMPOSS
DT0
        LDA NCODE
        JSR OWNIT?    ; already have it?
        BNE DT1
        JMP ALREADY
DT1
        LDA NCODE
        JSR INROOM?   ; is it here?
        BEQ DT2
        JMP NOTHERE
DT2
        LDA NCODE
        CMP #20       ; bolted unit?
        BNE DT3
         PRINT  T41   ; "Bolts are tight & rusty"
        JMP GOODPARSE
DT3
        CMP #42       ; free unit?
        BNE DT3B
        LDA PLACE     ; room 0?
        BNE DT3B
        LDA #27       ; got wrench?
        JSR OWNIT?
        BEQ DT3A
         PRINT  T80   ; "Bolts won't let you"
        JMP GOODPARSE
DT3A
        LDA #7        ; power cable
        JSR INROOM?
        BNE DT3B
         PRINT  T83   ; "Connected to cable"
        JMP GOODPARSE
DT3B
        JSR INVSPACE? ; arms full?
        BEQ DT4
        JMP ARMSFULL
DT4
        LDX ROOMPOS   ; get object position
        LDY INVPOS    ; and inv position
        LDA CUROBJS,X ; pick it up
        STA HOLDINGS,Y ; add to inventory
        LDA #$FF
        STA CUROBJS,X ; leave a blank slot
SHOWALL
        JSR SHOWVIS   ; show room
        JSR SHOWINV   ; and inventory
        JMP POKAY     ; done!
;
;         DROP
;         ----
DODROP
        JSR OWNIT?    ; do you have it?
        BEQ DD0
        JMP DONTHAVE
DD0
        LDA PLACE
        CMP #17       ; is this room 17?
        BEQ DROP17    ; special handling
DD1
        JSR ROOMSPACE? ; enough room for it?
        BEQ DROPIT
        JMP ROOMFULL
DROPIT
        LDX ROOMPOS
        LDY INVPOS
        LDA HOLDINGS,Y
        STA CUROBJS,X
        LDA #$FF
        STA HOLDINGS,Y
        BNE SHOWALL
;
; Handle room 17
;
DROP17
        JSR ROOMIN18? ; space in room 18?
        BNE DD1       ; no - drop it in 17
        LDY INVPOS    ; get inv position
        LDX ROOMPOS   ; and pos in room 18
        LDA HOLDINGS,Y ; pick up item
        STA OBJECTS,X ; and put in 18
        LDA #$FF      ; clear inventory
        STA HOLDINGS,Y ; slot
         PRINT  T68   ; "It falls down the pipe"
        LDA NCODE
        CMP #42       ; dropped the unit?
        BEQ DROPUNIT  ; special handling
D17B
        JSR SHOWVIS
        JSR SHOWINV
        JMP GOODPARSE
;
; Handle UNIT in 17
;
DROPUNIT
        LDA UNIT      ; init = 0
        BNE D17B
        LDA #38       ; update traitor
        LDX #5        ; status in
        STA UNIT      ; unit flag,
        STA NTRANS,X  ; translation matrix
        LDX #108      ; and object
        STA OBJECTS,X ; matrix
        JSR ROOMIN18? ; find room
        LDX ROOMPOS   ; for pistol
        LDA #21       ; and
        STA OBJECTS,X ; drop it in 18
        BNE D17B
;
; Find empty slot in room 18
;
ROOMIN18?
        LDX #109      ; skip 1st object
RN18
        LDA OBJECTS,X
        BMI RN18A     ; found a blank!
        INX
        CPX #113      ; scan to end
        BCC RN18      ; of room
        TXA           ; return NZ status
        RTS
RN18A
        STX ROOMPOS   ; save room pos
        LDA #0        ; set zero status
        RTS
;
;         REMOVE
;         ------
DOREMOVE
        CMP #22       ; moveable?
        BCS DRM
        JMP IMPOSS
DRM
        CMP #28       ; mask?
        BEQ GODROP
        CMP #32       ; suit?
        BEQ GODROP
        CMP #42       ; unit?
        BEQ GOGRAB
        JMP BESPEC
GOGRAB
        JMP DOTAKE
GODROP
        JMP DODROP
;
;         LOOK/EXAM
;         ---------
DOLOOK
        JSR INROOM?   ; is it in room?
        BEQ LOOKOK    ; if not,
        LDA NCODE
        JSR OWNIT?    ; do you have it?
        BEQ LOOKOK
        JMP NOTHERE   ; guess not
LOOKOK
        LDA NCODE
        CMP #9        ; captain?
        BNE LK0
;
; Search captain
;
EXAMCAP
        LDA CAPTAIN
        BEQ EC0
        JMP SEEMSORD
EC0
        JSR ROOMSPACE?
        BEQ EC1
        JMP ROOMFULL
EC1
         PRINT  T35   ; "Found something!"
        LDA #24
        STA CAPTAIN
        LDX ROOMPOS
        STA CUROBJS,X
        JSR SHOWVIS
        JMP GOODPARSE
;
LK0
        CMP #15       ; gauge?
        BNE LK1
;
; Read depth gauge
;
READGAUGE
        JSR ZFR0
        LDA DEPTH
        STA FR0
        JSR VPRINT
         PRINT  T55   ; "Fathoms"
        JMP GOODPARSE
;
LK1
        CMP #17       ; display?
        BNE LK2
;
; Read navigation displays
;
READISP
        LDA PLACE
        CMP #11       ; missile room?
        BEQ SHOWMD
        LDA SLAT
        LDX SLON
        BNE DISHOW
SHOWMD
        LDA MLAT
        LDX MLON
DISHOW
        STA LATSHOW
        STX LONSHOW
         PRINT  T53   ; "X ="
         POSITION  17,9
        JSR ZFR0
        LDA LATSHOW
        STA FR0
        JSR VPRINT
         PRINT  T54   ; "Y ="
         POSITION  17,10
        JSR ZFR0
        LDA LONSHOW
        STA FR0
        JSR VPRINT
        BNE LKX
;
LK2
        CMP #32       ; suit?
        BNE LK3
;
; Examine suit
;
EXAMSUIT
        LDA SUIT
        BEQ ES0
        JMP SEEMSORD
ES0
        JSR ROOMSPACE?
        BEQ ES1
        JMP ROOMFULL  ; "Not enough room here."
ES1
         PRINT  T35   ; "Found something!"
        LDA #33       ; key
        STA SUIT
        LDX ROOMPOS
        STA CUROBJS,X
        JSR SHOWVIS
        JMP GOODPARSE
;
LK3
        CMP #21       ; pistol?
        BNE LK4
;
; Examine pistol
;
EXAMPIST
        LDA BULLET
        BNE PX0       ; init = 1
         PRINT  T52   ; "No bullets"
        BNE LKX
PX0
         PRINT  T51   ; "Only 1 bullet"
        BNE LKX
;
LK4
        ASL A         ; * 2
        TAX           ; use as an index
        LDA LKLK,X    ; fetch lsb
        INX           ; and
        LDY LKLK,X    ; msb of text addr
        JSR EPRINT    ; print text
LKX
        JMP GOODPARSE ; and exit
;
;         EXAM TEXT LOOKUP TABLE
;         ----------------------
LKLK
        .WORD T36,T64,T34,T38
        .WORD T39,T34,T64,T39
        .WORD T42,T34,T34,T34
        .WORD T34,T34,T39,T34
        .WORD T37,T34,T34,T46
        .WORD T41,T34,T34,T48
        .WORD T48,T34,T49,T34
        .WORD T34,T34,T48,T48
        .WORD T34,T34,T34,T34
        .WORD T37,T34,T34,T49
        .WORD T39,T34,T47
;
;         READ
;         ----
DOREAD
        LDX #9
RDLOOP
        CMP READS,X
        BEQ READOK
        DEX
        BPL RDLOOP
        JMP IMPOSS
READOK
        JMP DOLOOK
;
; READable nouns
;
READS
        .BYTE 2,36,8,15,17
        .BYTE 23,24,26,30,31
;
```

```
;          PUSH
;          ----
DOPUSH
          JSR OWNIT?
          BNE DPH
          JMP WHYBOTH
DPH
          LDA NCODE
          JSR INROOM?
          BEQ DPH0
          JMP NOTHERE
DPH0
          LDA NCODE
          CMP #10       ; green button?
          BNE DPH1
;
; Handle green button push
;
          LDX #3
          LDA GREEN
          BEQ SCANON
          LDA #0
          STA GREEN
          LDA #2        ; blank scanner
SCANNER
          STA NTRANS,X
          STA CUROBJS
          JSR SHOWVIS
          JMP GOODPARSE
SCANON
          LDA OBJECTS+1 ; check cable
          CMP #7
          BEQ ONGREEN
          JMP NOTHAP    ; "Nothing happens"
ONGREEN
          STX GREEN
          LDA #36       ; active scanner
          BNE SCANNER
;
DPH1
          CMP #11       ; red button?
          BNE DPH2
;
; Handle red button push
;
          LDX RED
          BEQ REDON
          LDA #0
          STA RED
           PRINT  T59 ; "Sub levels off"
          BNE REDX
REDON
          INX           ; = 1
          STX RED
           PRINT  T60 ; "Sub dives!"
REDX
          JMP GOODPARSE
;
DPH2
          CMP #12       ; gold button?
          BNE DPH3
;
; Handle gold button
;
          LDA MLAT
          CLD
          CLC
          ADC #8
          STA MLAT
          JMP SHOWMD
;
DPH3
          CMP #13       ; silver button?
          BNE DPH4
;
; Handle silver button
;
          LDA MLON
          SEC
          CLD
          SBC #8
          STA MLON
          JMP SHOWMD
;
DPH4
          CMP #14       ; white button?
          BNE SORRY
;
; Handle white button
;
PUSHWHITE
          LDA MLAT      ; missile =
          CMP SLAT      ; sub?
          BNE SORRY
          LDA MLON      ; missile =
          CMP SLON      ; sub?
          BNE SORRY
          LDA SWITCH    ; missile armed?
          BEQ SORRY
          JSR NEWSCREEN
          LDA #14
          STA COLOR2
          STA COLOR4
          LDA #0
          STA COLOR1
           POSITION  12,11
           PRINT   T81 ; "Congratulations!"
          LDA ##22
          STA SDMCTL
SOLVED
          JMP SOLVED
SORRY
          JMP NOTHAP    ; "Nothing happens."
;
;          OPEN
;          ----
DOOPEN
          CMP #6        ; can't be OPENed
          BCC OP0       ; if NCODE > 6
          JMP IMPOSS
OP0
          JSR INROOM?   ; is it here?
          BEQ OP1
          JMP NOTHERE   ; nope
OP1
          LDA NCODE     ; hatch?
          BNE OP2
HOPEN
          LDA #34
          STA HATCH     ; mark hatch as opened
          STA CUROBJS   ; change in current objects
          LDA #4        ; open path to
          STA CURVECT+5 ; room 4
          JSR SHOWVIS
          JSR SHOWVECTS
          JMP POKAY
OP2
          CMP #1        ; door?
          BNE OP3
           PRINT  T64 ; "Lock is very secure"
          BNE OPX
OP3
          CMP #3        ; grate?
          BNE OP4
           PRINT  T38 ; "Screwed in place"
          BNE OPX
OP4
          CMP #5        ; airlock?
          BNE OP5
           PRINT  T23 ; "Can't do that yet"
          BNE OPX
OP5
          CMP #34       ; open hatch?
          BEQ ALOPEN
          CMP #35       ; open door?
          BEQ ALOPEN
          CMP #37       ; open grate?
          BEQ ALOPEN
          CMP #39       ; open airlock?
          BNE OP6
ALOPEN
           PRINT  T65 ; "Already open!"
OPX
          JMP GOODPARSE
OP6
          JMP IMPOSS
;
;          SHOOT
;          -----
DOSHOOT
          LDA #21       ; do you have
          JSR OWNIT?    ; the pistol?
          BEQ SHT0
          JMP EASIER
SHT0
          LDA NCODE
          JSR INROOM?
          BEQ SHT1
          LDA NCODE
          JSR OWNIT?
          BEQ SHT1
          JMP NOTHERE
SHT1
          LDA BULLET
          BNE SHT3
           PRINT  T52 ; "No bullets"
          JMP GOODPARSE
SHT3
           PRINT  T45 ; "BANG!"
          LDA #0
          STA BULLET
          LDA NCODE
          CMP #1        ; Locked door?
          BNE SHOOTX
           PRINT  T71 ; "Lock destroyed!"
          LDA #35       ; change door status in
          STA CUROBJS   ; object matrix
          LDX #1        ; and
          STA NTRANS,X  ; in the
          INX           ; translation
          STA NTRANS,X  ; table
          LDA #3        ; open west wall
          STA CURVECT+3 ; to room #3
          JSR SHOWVIS   ; show door change
          JSR SHOWVECTS ; and new vector
SHOOTX
          JMP GOODPARSE
;
;          INSERT
;          ------
DOINSERT
          JSR OWNIT?
          BEQ INS0
          JMP DONTHAVE
INS0
          LDA NCODE
          CMP #26       ; card?
          BNE INS1
           PRINT  T72 ; "Try examining things"
          JMP GOODPARSE
INS1
          CMP #24       ; ID?
          BEQ INS2
INSX
          JMP BESPEC
INS2
          LDA PLACE
          CMP #19       ; room 19?
          BNE INSX
          LDA #39       ; update object
          STA CUROBJS   ; matrix
          LDX #6        ; and
          STA NTRANS,X  ; translator
          LDA #21       ; open south wall
          STA CURVECT+1 ; to room 21
INEXIT
          JSR SHOWVIS
          JSR SHOWVECTS
          JMP POKAY
;
;          UNSCREW
;          -------
DOUNSCREW
          JSR INROOM?
          BEQ UNS0
          JMP NOTHERE
UNS0
          LDA UCODE
          CMP #24       ; nothing you can carry
          BCC UNS1      ; is unscrewable
          JMP WHYBOTH
UNS1
          LDA NCODE
          CMP #20       ; bolted unit?
          BNE UNS2
          JMP DOTAKE
UNS2
          CMP #3        ; closed grate?
          BEQ UNS3
          JMP IMPOSS
UNS3
          LDA #29       ; do you have
          JSR OWNIT?    ; the knife?
          BNE UNS4      ; nope
          LDA #37       ; patch
          STA CUROBJS   ; object
          LDX #4        ; and
          STA NTRANS,X  ; translator tables
          LDA #17       ; open south wall to
          STA CURVECT+1 ; room 17
          BNE INEXIT
UNS4
          LDA #22       ; screwdriver?
          JSR OWNIT?
          BNE UNSX
           PRINT  T73 ; "Blade's too tiny"
          JMP GOODPARSE
UNSX
          JMP EASIER
;
;          HOLD
;          ----
DOHOLD
          LDA UCODE
          CMP #38       ; breath?
          BEQ DHLD0
          JMP BESPEC
DHLD0
          LDA BREATH    ; already
          BNE DHLD1     ; holding
          LDA #9        ; it?
          STA BREATH    ; if not, set timer
          STA BHOLD     ; to 8 events
          JMP POKAY
DHLD1
          JMP ALREADY
;
;          UNLOCK
;          ------
DOUNLOCK
          LDA #33       ; key?
          JSR OWNIT?
          BEQ UNL0
          JMP EASIER
UNL0
          LDA NCODE
          JSR INROOM?
          BEQ UNL1
          JMP NOTHERE
UNL1
          LDA NCODE
          CMP #1        ; locked door?
          BNE UNL2
           PRINT  T75 ; "Key doesn't fit"
          JMP GOODPARSE
```

```
UNL2
        CMP #6      ; locked switch?
        BEQ UNL3
        JMP IMPOSS
UNL3
        LDA #40     ; unlocked
        STA CUROBJS
        LDX #7
        STA NTRANS,X
        STA SWITCH
        JMP INEXIT
;
;        CUT
;        ---
DOCUT
        JSR INROOM?
        BEQ DCT0
        JMP NOTHERE
DCT0
        LDA NCODE
        CMP #7      ; hummer?
        BEQ DCT1
        JMP IMPOSS
DCT1
        LDA #25     ; cutters?
        JSR OWNIT?
        BNE DCT2
        LDA GREEN
        BEQ CUTIT
        JSR NEWSCREEN
         POSITION  9,3
         PRINT  T76 ; "A jolt of high voltage"
        JMP KILLS
CUTIT
        LDA #41     ; severed
        STA CUROBJS+1
        LDX #8
        STA NTRANS,X
        LDA #2      ; blank scanner
        STA OBJECTS+42
        LDX #3
        STA NTRANS,X
        JMP INEXIT
DCT2
        JMP EASIER
;
;        POUR
;        ----
DOPOUR
        JSR OWNIT?
        BEQ DPR
        JMP DONTHAVE
DPR
        LDA NCODE
        CMP #30     ; shampoo?
        BEQ DOSHAM
        JMP IMPOSS
DOSHAM
        LDY INVPOS  ; remove
        LDA #$FF    ; shampoo from
        STA HOLDINGS,Y ; inventory
         PRINT  T44 ; "Shampoo all used up"
        LDA #20     ; is bolted unit
        JSR INROOM? ; nearby?
        BNE POURX   ; nope
        LDX ROOMPOS ; else change to
        LDA #42     ; free unit
        STA CUROBJS,X
        LDX #22     ; patch
        STA NTRANS,X ; translation
        INX         ; table
        STA NTRANS,X
POURX
        JSR SHOWVIS
        JSR SHOWINV
        JMP GOODPARSE
;
;        LUBRICATE
;        ---------
DOLUBE
        JSR OWNIT?
        BNE DOL
        JMP WHYBOTH
DOL
        LDA #30     ; got the shampoo?
        JSR OWNIT?
        BEQ DOL1
DOL0
        JMP EASIER
DOL1
        LDA NCODE
        JSR INROOM?
        BEQ DOSHAM
        JMP NOTHERE
;
;        WEAR
;        ----
DOWEAR
        CMP #28     ; mask?
        BEQ WEAROK
        CMP #32     ; suit?
        BEQ WEAROK
        JMP IMPOSS
WEAROK
        JMP DOTAKE
;
IMPOSS
         PRINT  T17 ; "That's impossible."
        BNE GOODPARSE
;
NOTYET
         PRINT  T23 ; "You can't do that yet."
        BNE GOODPARSE
;
BESPEC
         PRINT  T26 ; "Be more specific."
        BNE GOODPARSE
;
ARMSFULL
         PRINT  T24 ; "You're carrying too much!"
        BNE BADPARSE
;
ROOMFULL
         PRINT  T25 ; "Not enough room here."
        BNE BADPARSE
;
NOTHERE
         PRINT  T22 ; "It isn't here."
        BNE BADPARSE
;
ALREADY
         PRINT  T21 ; "You already have it."
        BNE BADPARSE
;
DONTHAVE
         PRINT  T27 ; "You don't have it."
        BNE BADPARSE
;
NOHELP
         PRINT  T32 ; "Doesn't help."
        BNE GOODPARSE
;
WHYBOTH
         PRINT  T33 ; "Why bother?"
        BNE GOODPARSE
;
SEEMSORD
         PRINT  T34 ; "Seems ordinary."
        BNE GOODPARSE
;
NOTHAP
         PRINT  T56 ; "Nothing happens."
        BNE GOODPARSE
;
EASIER
         PRINT  T50 ; "Easier said than done"
;
GOODPARSE
        JSR BEEP
        JMP NEXTEVENT
;
BADPARSE
        JSR BOOP
        JMP PARSER
;
;        SUBROUTINES
;        -----------
;
;        DLI ROUTINE
;        -----------
DLI
        PHA         ; save A,
        TXA
        PHA         ; X
        TYA         ; and
        PHA         ; Y registers
        LDX DLICOL  ; fetch color index
        LDA DCOLS,X ; fetch new color
        LDY CURSES,X ; and cursor shape
        STA WSYNC   ; wait for scan
        STA COLPF2  ; change color
        STY GRAFP0  ; and player shape
        INC DLICOL  ; update index
        PLA
        TAY         ; restore Y,
        PLA         ; X
        TAX         ; and
        PLA         ; A registers
        RTI         ; back to mainline
;
DCOLS
        .BYTE $70,$60,$70,$60,$70,$00,$60
CURSES
        .BYTE $00,$00,$00,$00,$00
CURSOR
        .BYTE $00,$00
;
;        PRINT MACRO SUBROUTINE
;        ----------------------
; ENTRY: Addr of EOL-terminated string
;        in A/Y registers (LSB/MSB).
;
EPRINT
        LDX #0      ; IOCB #0 (E:)
        STA ICBADR  ; lsb of string addr
        STY ICBADR+1 ; msb of string addr
        LDA #$09    ; PUT LINE command
        STA ICCOM
        LDA #$7F
        STA ICBLEN  ; lsb max string length
        STX ICBLEN+1 ; msb (0)
        JMP CIOV
;
;        POSITION MACRO ROUTINE
;        ----------------------
; ENTRY: X-pos in X, Y-pos in Y.
;
POSIT13
        LDX #13
POSIT
        STX COLCRS
        STY ROWCRS
        RTS
;
;        AUDIO PROMPTS
;        -------------
BEEP
        LDA #25     ; high tone
        BNE BTONE
;
BOOP
        LDA #100    ; low tone
BTONE
        STA AUDF1
        LDA #$AA    ; dist/vol = 10
        STA AUDC1
        LDA #0
        STA RTCLOK
SOUND
        LDA RTCLOK
        CMP #5      ; 5 jiffies
        BNE SOUND
        LDX #0
        STX AUDC1   ; silence!
        DEX
        STX CH      ; clear last key
        RTS
;
;        ERASE A STATUS LINE
;        -------------------
;
; ENTRY: Target line (0-18) in X.
;
ERASE
        LDA LADRSL,X ; lsb of line addr
        STA CLPOINT
        LDA LADRSH,X ; msb of addr
        STA CLPOINT+1
        LDY #24     ; clear 25 characters
        LDA #0
CLLOOP
        STA (CLPOINT),Y
        DEY
        BPL CLLOOP
        RTS
;
;        INIT SCREEN
;        -----------
NEWSCREEN
        LDA #$40
        STA NMIEN
        STA RAMTOP  ; set system to 16K
;
; Close IOCB #0 (E:)
;
        LDX #0      ; IOCB #0 (E:)
        LDA #$0C    ; CLOSE command
        STA ICCOM
        JSR CIOV    ; slam!
;
; Re-open E: with new RAM size
;
        LDX #0      ; E: again
        STX ICAUX2  ; zero this byte
        LDA #3      ; OPEN command
        STA ICCOM
        LDA # <EADR ; point to "E:"
        STA ICBADR
        LDA # >EADR
        STA ICBADR+1
        LDA #$0C    ; allow read/write
        STA ICAUX1
        JSR CIOV    ; do the OPEN
;
        LDA #0
        STA DMACTL
        STA SDMCTL  ; shut off ANTIC
        STA GRAFP0
        STA GRAFP1
        STA GRAFP2
        STA GRAFP3  ; blank out borders
;       LDX #8      ; all colors black
COLOFF
        STA PCOLR0,X
        DEX
        BPL COLOFF
        LDA #14
        STA COLOR1  ; except text
        LDA #$74
        STA COLOR0
;
        LDX #1
        STX CRSINH  ; disable system cursor
        INX         ; = 2
        STX LMARGN  ; fix margin
        LDA #$70
        STA POKMSK
        STA IRQEN   ; disable BREAK key
        RTS         ; and return
```

```
;
;         BLANK BAR
;         ---------
BAR
          PRINT  T9
          RTS
;
;         "WHAT" BAR
;         ----------
SAYWHAT
          PRINT  T10
          RTS
;
;         KEYBOARD INPUT HANDLER
;         ----------------------
; These routines are based in part on
; Steve Howard's "Alternative Keyboard Handler"
; (ANALOG Computing #15, pp. 96-103).
;
;         FETCH A KEYPRESS
;         ----------------
GETKEY
          LDA CH
          CMP #$FF      ; key pressed?
          BEQ GETKEY    ; not yet - keep scanning
;
; Analyze keycode
;
ANALYZE
          TAY           ; save key for later
          LDX #$FF
          STX CH        ; reset key
          AND #$C0      ; bit 6 or 7 set?
          BEQ LEGAL?    ; nope
;
; Handle a bad keypress
;
BADKEY
          JSR BOOP      ; razz user and
          JMP GETKEY    ; try again
;
; Look for illegal keys
;
LEGAL?
          TYA           ; restore keycode
          LDX #13
KLOOP
          CMP ILLEGAL,X
          BEQ BADKEY    ; razz if illegal key
          DEX
          BPL KLOOP
;
; Get ATASCII equivalent
;
          LDA ATASCI,Y
;
; Screen out numbers, pass EOL and BS
;
          CMP #SPACE    ; space bar?
          BEQ CLK1      ; that's okay
          CMP #EOL      ; RETURN?
          BEQ CLK1      ; fine by me
          CMP #$7E      ; backspace?
          BEQ CLK1      ; love 'em
          CMP #'a
           BCC BADKEY
          CLD
          SEC
          SBC #$20      ; convert to upper case
;
; Click the speaker
;
CLK1
          LDY #$7F
          STY CLICK
CLK2
          LDY CLICK
          STY CONSOL    ; tick!
          LDX #8        ; click freq
DELAY
          DEX
          BPL DELAY
          DEC CLICK
          BPL CLK2      ; 128 times
;
          RTS           ; ATASCII code in A
;
;         ILLEGAL KEYS
;         ------------
ILLEGAL
          .BYTE $1C     ; escape
          .BYTE $2C     ; tab
          .BYTE $27     ; atari
          .BYTE $3C     ; caps
          .BYTE $36     ; <
          .BYTE $37     ; >
          .BYTE $0F     ; =
          .BYTE $20     ; ;
          .BYTE $02     ; ;
          .BYTE $22     ; .
          .BYTE $26     ; /
          .BYTE $06     ; +
          .BYTE $07     ; *
          .BYTE $0E     ; -
;
;         ATASCII CONVERSION TABLE
;         ------------------------
; We use our own table because the
; location of the ROM-based table varies
; depending on which computer you have.
;
ATASCI
          .BYTE $6C,$6A,$3B,$8A,$8B,$6B,$2B,$2A
          .BYTE $6F,$80,$70,$75,$9B,$69,$2D,$3D
          .BYTE $76,$80,$63,$8C,$8D,$62,$78,$7A
          .BYTE $34,$80,$33,$36,$1B,$35,$32,$31
          .BYTE $2C,$20,$2E,$6E,$80,$6D,$2F,$81
          .BYTE $72,$80,$65,$79,$7F,$74,$77,$71
          .BYTE $39,$80,$30,$37,$7E,$38,$3C,$3E
          .BYTE $66,$68,$64,$80,$82,$67,$73,$61
;
;         INTERNAL CONVERSION TABLE
;         -------------------------
INTATA
          .BYTE $20,$40,$00,$60
;
;         Y-OFFSET TABLES
;         ---------------
; These two tables contain the
; starting address of each status line
; (absolute screen line address + 13).
; LADRSL holds the LSBs, LADRSH the MSBs.
;
LADRSL
          .BYTE  <SCREEN+53,  <SCREEN+53,  <SCREEN+93,  <SCREEN+133
          .BYTE  <SCREEN+173,  <SCREEN+213,  <SCREEN+253,  <SCREEN+293
          .BYTE  <SCREEN+333,  <SCREEN+373,  <SCREEN+413,  <SCREEN+453
          .BYTE  <SCREEN+493,  <SCREEN+533,  <SCREEN+573,  <SCREEN+613
          .BYTE  <SCREEN+653,  <SCREEN+693,  <SCREEN+733
LADRSH
          .BYTE  >SCREEN+53,  >SCREEN+53,  >SCREEN+93,  >SCREEN+133
          .BYTE  >SCREEN+173,  >SCREEN+213,  >SCREEN+253,  >SCREEN+293
          .BYTE  >SCREEN+333,  >SCREEN+373,  >SCREEN+413,  >SCREEN+453
          .BYTE  >SCREEN+493,  >SCREEN+533,  >SCREEN+573,  >SCREEN+613
          .BYTE  >SCREEN+653,  >SCREEN+693,  >SCREEN+733
;
;         FETCH INPUT LINE
;         ----------------
GETLINE
;
; Clear line input buffer
;
          CLD
          LDX #24
          LDA #SPACE
CLINL
          STA INLINE,X
          DEX
          BPL CLINL
;
; Get first character of line
;
GETONE
           POSITION  13,12
          LDX #$FF
          STX CURSEN    ; turn on PM8 cursor
          STX CH        ; clear key
          INX
          STX LENGTH    ; zero line length
          JSR GETKEY    ; fetch a keycode
          CMP #SPACE    ; first char not be
          BEQ BADONE    ; a space
          CMP #$7E      ; a backspace
          BEQ BADONE
          CMP #EOL      ; or an EOL
          BNE PUT1
;
; Handle bad first character
;
BADONE
          JSR BOOP      ; razz user and
          JMP GETONE    ; try again
;
; Print 1st char
;
PUT1
          JSR SETCIO    ; to E:
          JSR CIOV
          INC LENGTH
;
; Get rest of input line
;
REST
          JSR GETKEY    ; grab another keycode
          CMP #EOL      ; if it's an EOL,
          BEQ GOTEOL    ; line entry complete
          CMP #$7E      ; backspace?
          BNE PUTNEXT   ; no - send to screen
;
; Handle a backspace
;
BACKS
          DEC LENGTH    ; if 1st char of line
          BMI BADONE    ; signal error
          JSR SETCIO    ; E:
          JSR CIOV      ; let CIO do backspace
          LDA LENGTH    ; if length=0,
          BEQ GETONE    ; handle as 1st char
          BNE REST      ; else continue
;
; Print latest character
;
PUTNEXT
          JSR SETCIO
          JSR CIOV      ; print character
          INC LENGTH    ; next position
          LDA LENGTH
          CMP #24       ; end of line?
          BCC REST      ; not yet - get another key
;
; Too many chars in line!
;
GETLAST
          JSR BOOP      ; a warning razz
          JSR GETKEY
          CMP #EOL      ; must have an EOL
          BEQ GOTEOL
          CMP #$7E      ; or a backspace
          BEQ BACKS
          BNE GETLAST ; insist!
;
; EOL recieved
;
GOTEOL
          JSR SETCIO    ; specify E:
          STX CURSEN    ; shut off cursor
          JSR CIOV      ; and send EOL
;
; Convert screen bytes to ATASCII
; and move to INLINE
;
          LDY #24
TOBUFF
          CLC
          LDA SCREEN+493,Y ; grab screen byte
          STA SCREEN+453,Y ; move to upper line
          ROL A
          ROL A
          ROL A
          ROL A
          AND #3        ; transform byte, and
          TAX           ; use as an index
          LDA SCREEN+453,Y ; restore original value
          AND #$1F      ; clear bits 5-7
          ORA INTATA,X ; merge with code table
          STA INLINE,Y ; send to buffer
          LDA #0
          STA SCREEN+493,Y ; clear response line
          DEY
          BPL TOBUFF
          RTS
;
;         IMMEDIATE VBI ROUTINE
;         ---------------------
; Positions and blinks cursor,
; resets DLI color index
;
IMMVBI
;
; Reset DLI color index
;
          LDA #0
          STA DLICOL
;
; Okay to update cursor?
;
          LDA CURSEN    ; if enable flag = 0,
          BEQ VEXIT     ; don't redraw cursor
;
; Calculate cursor X-position:
; XNEW = ( XOLD * 4 ) + 48
;
          CLD
          LDA COLCRS
          ASL A
          ASL A         ; times 4
          CLC
          ADC #48       ; plus 48
          STA HPOSP0    ; use as h-pos
;
; Don't blink cursor if a key
; is being pressed.
;
          LDA SRTIMR    ; 0 = no press
          BEQ BLINK?
;
          LDA #$F0
          STA CURSOR
          STA CSHAPE    ; force cursor on
          LDA #60       ; for at least
          STA BLINK     ; 1 second
;
BLINK?
          LDA CSHAPE
          DEC BLINK     ; next jiffy
          BNE VEXIT     ; don't blink until 0
;
; Blink the cursor
;
          LDY #30
          STY BLINK     ; reset timer
          EOR #$F0      ; flip the cursor shape
          STA CSHAPE    ; and save it for later
VEXIT
          STA CURSOR    ; plot the cursor
          JMP SYSVBV    ; sayonara
;
;         DISPLAY EVENT COUNTER
;         ---------------------
```

```
SHOWEV
;
; Initialize EBUFF
;
        LDA #'Ø
        STA EBUFF
        STA EBUFF+1
        STA EBUFF+2
        LDA #EOL
        STA EBUFF+4
;
; Convert event # to ATASCII
;
ECON
        LDA EVENT
        STA FRØ
        LDA EVENT+1
        STA FRØ+1
        JSR IFP     ; convert to floating point
        JSR FASC    ; then to ATASCII
        CLD
;
; Determine length of number
;
        LDY #$FF    ; init loop index
FINDE
        INY
        LDA (INBUFF),Y ; check characters
        BPL FINDE
;
; Change # to inverse video and
; move to EBUFF
;
        LDX #3      ; move 3 chars maximum
TOEB
        LDA (INBUFF),Y
        ORA #$80    ; set msb
        STA EBUFF,X ; put in EBUFF
        DEX
        DEY
        BPL TOEB
;
; Display contents of EBUFF
;
         POSITION  33,Ø
         PRINT  EBUFF
        RTS
;
;         SET CIO TO PUT CHAR MODE
;         ------------------------
SETCIO
        LDX #$ØB
        STX ICCOM
        LDX #Ø
        STX ICBLEN
        STX ICBLEN+1
        RTS
;
;         SYNTAX ERROR
;         ------------
SYNERR
         PRINT  T13 ; "Syntax:"
        RTS
;
;         CLEAR WINDOWS
;         -------------
; What Happens window
;
CLWH
        LDX #9
        JSR ERASE
        INX
        JSR ERASE
        LDY #9
        JMP POSIT13
;
; Inventory window
;
CLINV
        LDX #13
CLINV1
        JSR ERASE
        INX
        CPX #19
        BCC CLINV1
        RTS
;
;         DISPLAY ROOM VECTORS
;         --------------------
SHOWVECTS
        LDX #2
        JSR ERASE
        LDY #Ø
        LDX #Ø
        STX ANY?
SVL
        LDA CURVECT,X
        BMI VSKIP
        LDA VNAMES,X
        STA SCREEN+93,Y
        INC ANY?
        INY
        INY
VSKIP
        INX
        CPX #6
        BCC SVL
        LDA ANY?
        BNE VECTEX
         POSITION  13,2
         PRINT  T61 ; "None"
VECTEX
        RTS
;
; Vector initials
;
VNAMES
        .SBYTE "NSEWUD"
;
;         DISPLAY ROOM OBJECTS
;         --------------------
;
SHOWVIS
        LDX #3
CLVIS
        JSR ERASE
        INX
        CPX #9
        BCC CLVIS
         POSITION  13,3
        LDX #Ø
        STX ANY?
SHV1
        STX PDEX
        LDA CUROBJS,X
        BMI SHV2
        INC ANY?
        TAX
        LDA OBDLS,X
        LDY OBDHS,X
        JSR EPRINT
SHV2
        LDX PDEX
        INX
        CPX #6
        BCC SHV1
        LDA ANY?
        BNE SHEXIT
         PRINT  T28 ; "Nothing"
SHEXIT
        RTS
;
;         DISPLAY INVENTORY
;         -----------------
SHOWINV
        JSR CLINV
         POSITION  13,13
        LDX #Ø
        STX ANY?
SHI1
        STX PDEX
        LDA HOLDINGS,X
        BMI SHI2
        INC ANY?
        TAX
        LDA OBDLS,X
        LDY OBDHS,X
        JSR EPRINT
SHI2
        LDX PDEX
        INX
        CPX #6
        BCC SHI1
        LDA ANY?
        BNE SIEXIT
         PRINT  T28 ; "Nothing"
SIEXIT
        RTS
;
;         "OKAY" PROMPT
;         -------------
POKAY
        JSR CLWH
         PRINT  T20 ; "Okay"
        JMP GOODPARSE
;
;         SEARCH INVENTORY
;         ----------------
INVSPACE?
        LDA #$FF
OWNIT?
        LDX #5
OLOOP
        CMP HOLDINGS,X
        BEQ FOUND
        DEX
        BPL OLOOP
        TXA
        RTS
FOUND
        STX INVPOS
        LDA #Ø
        RTS
;
;         SEARCH ROOM
;         -----------
ROOMSPACE?
        LDA #$FF
INROOM?
        LDX #5
IRLOOP
        CMP CUROBJS,X
        BEQ FOUND2
        DEX
        BPL IRLOOP
        TXA
        RTS
FOUND2
        STX ROOMPOS
        LDA #Ø
        RTS
;
;         PRINT INTEGER IN FRØ
;         --------------------
;
VPRINT
        JSR IFP
        JSR FASC
        CLD
        LDY #$FF
VLOOP
        INY
        LDA (INBUFF),Y
        BPL VLOOP
        AND #$7F
        STA (INBUFF),Y
        INY
        LDA #EOL
        STA (INBUFF),Y
        LDA INBUFF
        LDY INBUFF+1
        JMP EPRINT
;
;         DEATH
;         -----
;
KILLS
         POSITION  10,5
         PRINT  T63 ; "kills you instantly!"
DEATH
         POSITION  24,7
         PRINT  T66 ; "YOU ARE DEAD"
        JMP STARTOP
;
;         CLOSE IOCB #1
;         -------------
;
CLOSE1
        LDX #$1Ø
        LDA #12
        STA ICCOM,X
        JMP CIOV
;
;         POINT TO FILENAMES
;         ------------------
;
DPOINT
        LDA # <FILE
        LDY # >FILE
        JMP POINT
TPOINT
        LDA # <CADR
        LDY # >CADR
POINT
        LDX #$1Ø
        STA ICBADR,X
        TYA
        STA ICBADR+1,X
        RTS
;
;         INIT DATABASE
;         -------------
;
INITDATA
;
; Set status flags
;
        LDX #19
        LDA #Ø
IDØ
        STA EVENT,X
        DEX
        BPL IDØ
;
; Clear vector/object matrix
;
        LDX #Ø
        LDA #$FF
ID1
        STA OBJECTS,X
        STA VECTORS,X
        INX
        CPX #132
        BCC ID1
;
; Clear working arrays
;
        LDX #5
ID2
        STA HOLDINGS,X
        STA CUROBJS,X
        STA CURVECT,X
        DEX
        BPL ID2
        LDX #1
        STX BULLET  ; 1 bullet
        STX PLACE   ; start in escape tube
        LDA #32
        STA MLAT
        LDA #168
        STA MLON
;
; Init tranlation table
;
        LDX #37
ID3
```

```
        LDA TRANS,X
        STA NTRANS,X
        DEX
        BPL ID3
;
; Init vector matrix
;
        LDX #39
ID4
        LDA VDATA,X
        LDY VSETS,X
        STA VECTORS,Y
        DEX
        BPL ID4
;
; Init object matrix
;
        LDX #31
ID5
        LDA ODATA,X
        LDY OSETS,X
        STA OBJECTS,Y
        DEX
        BPL ID5
        RTS
;
; Translator data
;
TRANS
        .BYTE 0,1,1,2,3,4,5,6
        .BYTE 7,8,9,10,11,12,13,14
        .BYTE 15,16,16,17,18,19,20,20
        .BYTE 21,22,23,24,25,26,27,28
        .BYTE 28,29,30,31,32,33
;
; Vector data
;
VDATA
        .BYTE 2,0,4,4,2,6,1,15
        .BYTE 6,4,9,7,5,6,9,6
        .BYTE 10,8,19,9,21,15,13,12
        .BYTE 15,12,14,16,14,4,15,14
        .BYTE 19,15,20,18,9,19,19,11
VSETS
        .BYTE 1,12,13,20,24,25,28,29
        .BYTE 32,36,37,38,39,45,50,54
        .BYTE 56,57,59,63,71,73,74,81
        .BYTE 86,90,91,92,93,94,99,102
        .BYTE 110,114,116,117,118,123,126,130
;
; Object data
;
ODATA
        .BYTE 20,7,0,22,8,9,23,1
        .BYTE 25,2,10,15,11,16,17,31
        .BYTE 17,12,13,27,28,3,30,26
        .BYTE 29,18,4,5,19,14,32,6
OSETS
        .BYTE 0,1,6,7,12,18,19,24
        .BYTE 30,42,43,48,49,54,60,61
        .BYTE 66,67,68,72,78,84,85,90
        .BYTE 96,102,108,114,115,116,120,126
;
;          DICTIONARY
;          ----------
;
; SINGLE-CHAR COMMANDS
; --------------------
SCOMS
        .BYTE "N"    ; 0 - North
        .BYTE "S"    ; 1 - South
        .BYTE "E"    ; 2 - East
        .BYTE "W"    ; 3 - West
        .BYTE "U"    ; 4 - Up
        .BYTE "D"    ; 5 - Down
        .BYTE "Q"    ; 6 - Quit
        .BYTE "X"    ; 7 - Save
        .BYTE "A"    ; 8 - Again
;
; COMPOUND VERBS
; --------------
VERBS
        .BYTE "TAK" ; 0 - take
        .BYTE "GET" ; 1 - get
        .BYTE "PUL" ; 2 - pull
        .BYTE "DRO" ; 3 - drop
        .BYTE "REM" ; 4 - remove
        .BYTE "LOO" ; 5 - look
        .BYTE "EXA" ; 6 - examine
        .BYTE "SEA" ; 7 - search
        .BYTE "REA" ; 8 - read
        .BYTE "PUS" ; 9 - push
        .BYTE "PRE" ; 10 - press
        .BYTE "OPE" ; 11 - open
        .BYTE "CLO" ; 12 - close
        .BYTE "USE" ; 13 - use
        .BYTE "KIL" ; 14 - kill
        .BYTE "SHO" ; 15 - shoot
        .BYTE "FIR" ; 16 - fire
        .BYTE "BRE" ; 17 - break
        .BYTE "SMA" ; 18 - smash
        .BYTE "INS" ; 19 - insert
        .BYTE "UNS" ; 20 - unscrew
        .BYTE "HOL" ; 21 - hold
        .BYTE "UNL" ; 22 - unlock
        .BYTE "CUT" ; 23 - cut
        .BYTE "POU" ; 24 - pour
        .BYTE "LUB" ; 25 - lubricate
        .BYTE "GRE" ; 26 - grease
        .BYTE "WEA" ; 27 - wear
        .BYTE "GO " ; 28 - go
;
NV      =   29      ; total # of verbs
;
; VERB VECTOR TABLE
; -----------------
VVECTS
        .WORD DOTAKE ; 0
        .WORD DOTAKE ; 1
        .WORD EASIER ; 2
        .WORD DODROP ; 3
        .WORD DOREMOVE ; 4
        .WORD DOLOOK ; 5
        .WORD DOLOOK ; 6
        .WORD DOLOOK ; 7
        .WORD DOREAD ; 8
        .WORD DOPUSH ; 9
        .WORD DOPUSH ; 10
        .WORD DOOPEN ; 11
        .WORD WHYBOTH ; 12
        .WORD BESPEC ; 13
        .WORD EASIER ; 14
        .WORD DOSHOOT ; 15
        .WORD DOSHOOT ; 16
        .WORD WHYBOTH ; 17
        .WORD WHYBOTH ; 18
        .WORD DOINSERT ; 19
        .WORD DOUNSCREW ; 20
        .WORD DOHOLD ; 21
        .WORD DOUNLOCK ; 22
        .WORD DOCUT ; 23
        .WORD DOPOUR ; 24
        .WORD DOLUBE ; 25
        .WORD DOLUBE ; 26
        .WORD DOWEAR ; 27
;
; COMPOUND NOUNS
; --------------
NOUNS
        .BYTE "HAT" ; 0 - hatch
        .BYTE "DOO" ; 1 - door
        .BYTE "LOC" ; 2 - lock
        .BYTE "SCA" ; 3 - scanner
        .BYTE "GRA" ; 4 - grate
        .BYTE "TRA" ; 5 - traitor
        .BYTE "AIR" ; 6 - airlock
        .BYTE "SWI" ; 7 - switch
        .BYTE "CAB" ; 8 - cable
        .BYTE "SIG" ; 9 - sign
        .BYTE "CAP" ; 10 - captain
        .BYTE "GRE" ; 11 - green
        .BYTE "RED" ; 12 - red
        .BYTE "GOL" ; 13 - gold
        .BYTE "SIL" ; 14 - silver
        .BYTE "WHI" ; 15 - white
        .BYTE "GAU" ; 16 - gauge
        .BYTE "PER" ; 17 - periscope
        .BYTE "SCO" ; 18 - scope
        .BYTE "DIS" ; 19 - display
        .BYTE "DUC" ; 20 - duct
        .BYTE "SLO" ; 21 - slot
        .BYTE "BOL" ; 22 - bolt
        .BYTE "UNI" ; 23 - unit
        .BYTE "PIS" ; 24 - pistol
        .BYTE "SCR" ; 25 - screwdriver
        .BYTE "NOT" ; 26 - note
        .BYTE "ID " ; 27 - ID
        .BYTE "CUT" ; 28 - cutters
        .BYTE "CAR" ; 29 - card
        .BYTE "WRE" ; 30 - wrench
        .BYTE "MAS" ; 31 - mask
        .BYTE "GAS" ; 32 - gas
        .BYTE "KNI" ; 33 - knife
        .BYTE "SHA" ; 34 - shampoo
        .BYTE "MAN" ; 35 - manual
        .BYTE "SUI" ; 36 - suit
        .BYTE "KEY" ; 37 - key
        .BYTE "BRE" ; 38 - breath
        .BYTE "BUT" ; 39 - button
        .BYTE "INV" ; 40 - inventory
;
NNN     =   41      ; total # of nouns
;
;         ROOM DESCRIPTIONS
;         -----------------
;
RD0
        TEXT  "Sonar sphere"
RD1
        TEXT  "Escape tube"
RD2
        TEXT  "Access tunnel"
RD3
        TEXT  "Captain's quarters"
RD4
        TEXT  "Forward passage"
RD5
        TEXT  "Radio room"
RD6
        TEXT  "Long corridor"
RD7
        TEXT  "Sonar station"
RD8
        TEXT  "Ballast control"
RD9
        TEXT  "Command station"
RD10
        TEXT  "Navigation center"
RD11
        TEXT  "Upper missile bay"
RD12
        TEXT  "Torpedo room"
RD13
        TEXT  "Weapons locker"
RD14
        TEXT  "Shower stalls"
RD15
        TEXT  "Crew's quarters"
RD16
        TEXT  "Galley"
RD17
        TEXT  "Ventilation duct"
RD18
        TEXT  "Fan room"
RD19
        TEXT  "Missile control"
RD20
        TEXT  "Equipment bay"
RD21
        TEXT  "Lower missile bay"
;
;         ROOM DESC. ADDRESS TABLES
;         -------------------------
;
RDLS
        .BYTE  <RD0, <RD1, <RD2, <RD3, <RD4, <RD5
        .BYTE  <RD6, <RD7, <RD8, <RD9, <RD10, <RD11
        .BYTE  <RD12, <RD13, <RD14, <RD15, <RD16, <RD17
        .BYTE  <RD18, <RD19, <RD20, <RD21
;
RDHS
        .BYTE  >RD0, >RD1, >RD2, >RD3, >RD4, >RD5
        .BYTE  >RD6, >RD7, >RD8, >RD9, >RD10, >RD11
        .BYTE  >RD12, >RD13, >RD14, >RD15, >RD16, >RD17
        .BYTE  >RD18, >RD19, >RD20, >RD21
;
;         OBJECT DESCRIPTIONS
;         -------------------
;
OBD0
        TEXT  "Closed hatch"
OBD1
        TEXT  "Locked door"
OBD2
        TEXT  "Blank scanner"
OBD3
        TEXT  "Closed grate"
OBD4
        TEXT  "Traitor with pistol"
OBD5
        TEXT  "Closed airlock"
OBD6
        TEXT  "Locked arming switch"
OBD7
        TEXT  "Power cable"
OBD8
        TEXT  "Sign"
OBD9
        TEXT  "Dead captain"
OBD10
        TEXT  "Green button"
OBD11
        TEXT  "Red button"
OBD12
        TEXT  "Gold button"
OBD13
        TEXT  "Silver button"
OBD14
        TEXT  "White button"
OBD15
        TEXT  "Depth gauge"
OBD16
        TEXT  "Periscope"
OBD17
        TEXT  "Digital display"
OBD18
        TEXT  "Duct down to fan room"
OBD19
        TEXT  "Slot in airlock"
OBD20
        TEXT  "Bolted-down sonar unit"
OBD21
        TEXT  "Pistol"
OBD22
        TEXT  "Tiny screwdriver"
OBD23
        TEXT  "Suicide note"
OBD24
        TEXT  "Security ID"
OBD25
        TEXT  "Cable cutters"
OBD26
        TEXT  "Card"
OBD27
        TEXT  "Wrench"
OBD28
        TEXT  "Gas mask"
OBD29
        TEXT  "Dull knife"
OBD30
        TEXT  "Shampoo"
OBD31
        TEXT  "Tactics manual"
OBD32
```

```
OBD33   TEXT  "Radiation suit"
OBD34   TEXT  "Key"
OBD35   TEXT  "Open hatch"
OBD36   TEXT  "Open door"
OBD37   TEXT  "Active scanner"
OBD38   TEXT  "Open grate"
OBD39   TEXT  "Dead traitor"
OBD40   TEXT  "Open airlock"
OBD41   TEXT  "Activated switch"
OBD42   TEXT  "Severed cable"
        TEXT  "Radioactive sonar unit"
        ;
        ;  DESCRIPTION ADDR TABLES
        ;  ----------------------
OBDLS   .BYTE <OBD0, <OBD1, <OBD2, <OBD3
        .BYTE <OBD4, <OBD5, <OBD6, <OBD7
        .BYTE <OBD8, <OBD9, <OBD10, <OBD11
        .BYTE <OBD12, <OBD13, <OBD14, <OBD15
        .BYTE <OBD16, <OBD17, <OBD18, <OBD19
        .BYTE <OBD20, <OBD21, <OBD22, <OBD23
        .BYTE <OBD24, <OBD25, <OBD26, <OBD27
        .BYTE <OBD28, <OBD29, <OBD30, <OBD31
        .BYTE <OBD32, <OBD33, <OBD34, <OBD35
        .BYTE <OBD36, <OBD37, <OBD38, <OBD39
        .BYTE <OBD40, <OBD41, <OBD42, <OBD43
        ;
OBDHS   .BYTE >OBD0, >OBD1, >OBD2, >OBD3
        .BYTE >OBD4, >OBD5, >OBD6, >OBD7
        .BYTE >OBD8, >OBD9, >OBD10, >OBD11
        .BYTE >OBD12, >OBD13, >OBD14, >OBD15
        .BYTE >OBD16, >OBD17, >OBD18, >OBD19
        .BYTE >OBD20, >OBD21, >OBD22, >OBD23
        .BYTE >OBD24, >OBD25, >OBD26, >OBD27
        .BYTE >OBD28, >OBD29, >OBD30, >OBD31
        .BYTE >OBD32, >OBD33, >OBD34, >OBD35
        .BYTE >OBD36, >OBD37, >OBD38, >OBD39
        .BYTE >OBD40, >OBD41, >OBD42, >OBD43
        ;
        ;  TEXT
        ;  ----
EADR    TEXT  "E:"
CADR    TEXT  "C:"
FILE    TEXT  "D1:GAME.DAT"
T0      TEXT  "Brian Moriarty's"
T1      TEXT  "CRASH DIVE"
T2      TEXT  "(C)1984 ANALOG Computing"
T3      TEXT  "Press  START  to play new game"
T4      TEXT  "Press  OPTION  to restore old game"
T5      TEXT  "Restore from Disk or Cassette?"
T6      TEXT  "  CRASH DIVE! (TM)     EVENT #                "
T7      TEXT  "  LOCATION      EXITS "
T8      TEXT  "  VISIBLE       ITEMS "
T9      TEXT  "                "
T10     TEXT  "    WHAT "
T11     TEXT  "  HAPPENS       YOUR   RESPONSE "
T12     TEXT  "                YOU    ARE  CARRYING "
T13     TEXT  "Syntax:"
T14     TEXT  "Bad command"
T15     TEXT  "Bad verb"
T16     TEXT  "Bad noun"
T17     TEXT  "That's impossible"
T18     TEXT  "Can't go that way"
T19     TEXT  "Type Y to quit game:"
T20     TEXT  "Okay"
T21     TEXT  "Already holding it"
T22     TEXT  "Isn't here"
T23     TEXT  "Can't do that yet"
T24     TEXT  "Your arms are full!"
T25     TEXT  "Not enough room here"
T26     TEXT  "Be more specific"
T27     TEXT  "You don't have it"
T28     TEXT  "Nothing"
T29     TEXT  "Type N S E W U or D"
T30     TEXT  "Refer to it by color"
T31     TEXT  "Type I for inventory"
T32     TEXT  "Doesn't help"
T33     TEXT  "Why bother?"
T34     TEXT  "Seems ordinary"
T35     TEXT  "You found something!"
T36     TEXT  "It's airtight"
T37     TEXT  "Enemy approaching!"
T38     TEXT  "Screwed in place"
T39     TEXT  "Looks dangerous"
T40     TEXT  "Needs key to activate"
T41     TEXT  "Bolts are tight & rusty"
T42     TEXT  "DANGER: Radiation zone!"
T43     TEXT  "Brains blown out"
T44     TEXT  "Shampoo all used up"
T45     TEXT  "BANG!"
T46     TEXT  "Accepts security ID card"
T47     TEXT  "It glows"
T48     TEXT  "See photo in ANALOG #18"
T49     TEXT  "Ace of Spades!"
T50     TEXT  "Easier said than done"
T51     TEXT  "Only 1 bullet"
T52     TEXT  "No bullets"
T53     TEXT  "X ="
T54     TEXT  "Y ="
T55     TEXT  "FATHOMS"
T56     TEXT  "Nothing happens"
T59     TEXT  "Sub levels off"
T60     TEXT  "Sub dives"
T61     TEXT  "None"
T62     TEXT  "A cloud of poisonous gas"
T63     TEXT  "Kills you instantly!"
T64     TEXT  "Lock is very secure"
T665    TEXT  "Already open"
T66     TEXT  "you are DEAD"
T67     TEXT  "Enemy captures the sub and"
T68     TEXT  "It falls down the pipe"
T71     TEXT  "Lock destroyed!"
T72     TEXT  "Try examining things"
T73     TEXT  "Screwdriver's too tiny"
T75     TEXT  "Key won't fit"
T76     TEXT  "A jolt of high voltage"
T77     TEXT  "Traitor shoots you and"
T78     TEXT  "Sub hits bottom!"
T79     TEXT  "A blast of radioactivity"
T80     TEXT  "Bolts won't let you"
T81     TEXT  "Congratulations!"
T82     TEXT  "Save game to Disk or Cassette?"
T83     TEXT  "Connected to cable"
        ;
```

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of Atari computer owners. **ANALOG** pays between $30.00-$360.00 for all articles. All submissions for publication must be typed, upper and lower case with double spacing. Program listings should be provided in printed form, and on cassette or disk. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned. please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number. Send programs to: Editor, **ANALOG Computing**, P.O. Box 23, Worcester, MA 01603.