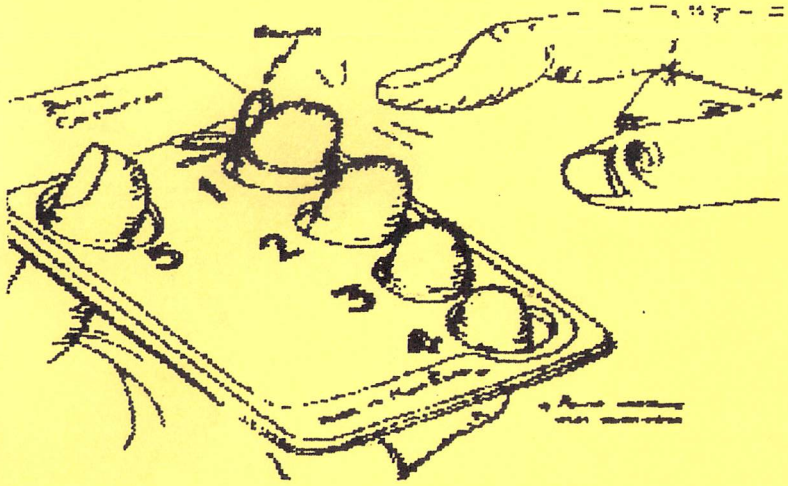
The Amazing Adventures of CAPTAIN COMAL™

BOOK 9: UTILITIES SET 2



Featuring COMAL QUICK and Printer Utilities
Compiled by Jesse Knight

ISBN 0-928411-07-9

COMAL UTILITY DISK #2 INSTRUCTIONS

This is the set of instructions for some of the programs on the second UTILITY disk. The **front** side of the disk contains a special loader that loads COMAL much quicker than normal, along with some general purpose utility programs. The **back** side of the disk contains an assortment of programs written for various printers.

The easiest method to load COMAL from this disk is to use the BASIC command:

```
LOAD"BOOT*",8,1
```

Another way is to use:

```
LOAD"BOOT*",8
RUN
```

Either one will start the loading process. COMAL will automatically be fast loaded by the special fast loader if a 1541, Indus, MSD (use drive 0 on dual), or 1571 drive is being used. If none of those drives are being used or if a Buscard II interface is used with device 8 selected as an IEEE device, COMAL will be loaded normally, which means slowly. With the fast loader, COMAL and the HI program are loaded in under 30 seconds. That's much better than the 2 minutes it used to take.

While this special fast loader will load the COMAL language itself faster than normal, it does not speed up the loading of programs running under COMAL 0.14.

You can create your own work disks with the fast loading COMAL on them. To do this, the files:

```
BOOT
LOAD1
LOAD2
0.14
-ERROR-MESSAGES-
HI
```

must be copied to your work disk. These files must be the first files on the disk and they must be in the same order.

The HI program may be changed to suit your own needs.

The programmers who wrote the programs on this disk deserve credit for their work. If they hadn't written the programs I could not have put this disk together. They deserve credit for not only writing useful programs but for providing good instructions with them. This book was to provide instructions for the programs on the disk. I found that most of the programs didn't require any more instructions than they already contained. That caused me to change my plans on what to write in this book. I have provided the program and author's name, along with a general description of the program. In some cases there was additional information I could provide about a program from my own experience. I hope you will find it useful.

Jesse Knight
August 1985

PROGRAM: 1541'Alignment
AUTHOR: Craig Van DeGrift
FILES: 1541'Align'1
1541'Align'2

These three programs provide detailed instructions for aligning the 1541. They are for technically oriented people. All the instructions needed for using the programs are contained in help menus. I would suggest you read all the help menus before using the programs. It wouldn't hurt if the CTRL-P text dump program was used to print some of the more important screens for reference.

Usually alignment problems are indicated by read errors when trying to read programs or information from a disk. This doesn't automatically mean you have an alignment problem. It could be a bad disk, or a 'copy protected' disk. If the drive has problems reading disks that are known to be good it should be fixed **immediately.** Continued use, especially writing to good disks, can result in data loss. I know because it has happened to me more than once. I finally learned my lesson and I hope you will take my warning. I promise not to lose any sleep if you don't.

PROGRAM: COMAL'keypad0.14
AUTHOR: James Borden

This program can be a real time saver for typing numeric data. By using the interrupts it makes the m, j, k, l, u, i, and o keys produce the digits 0 through 6, respectively. The result is as good as having a numeric keypad attached. The keyboard can be toggled between normal and keypad mode by pressing the pound symbol located next to the CLR/HOME key. The toggle key can be changed by using POKE 52051,ORD("X"), where X is the new key to use.

PROGRAM: dir'manipulator
AUTHOR: David Stidolph

This program can be a real life-saver. As the name implies, it allows a disk directory to be manipulated. Only one directory block can be worked with at one time. That happens to be eight directory entries. The directory entry to work on is selected by using cursor up/down. A 'menu' of functions appears at the bottom of the display. A function

is selected by using cursor left/right followed by return. The functions are:

change file name and type
scratch protect a file
unscratch protect a file
restore a scratched file
next directory block
previous directory block
quit

In addition, the F1 and F3 keys are used to save and cancel changes, respectively.

The F1 key should be used to save any changes before going to another block or stopping the program (if that's what you want). It takes a couple of seconds for the changes to be saved. Don't get impatient and press it more than once because it will save the changes once for each time you press F1.

I think some elaboration on how to recover a scratched file would be helpful. If a file gets scratched when it shouldn't have, it is important that the proper steps are followed to recover it. Above all else, DO NOT write anything else to that disk before trying to recover the lost file. When the file is scratched, all the DOS does is change one byte of the directory entry and free the sectors used by the file. All the data is still on the disk and can be recovered, unless it has been overwritten. That's why it's important not to write anything else to the disk.

When dir'manipulator tries to restore a file, it follows the files sector chain to verify that all the blocks are free. If it finds a block in the file that has been allocated it indicates the file can't be recovered. There is a slim chance that the block may actually be free. Try using PASS "v0" to validate the disk, then try to recover the file again using dir'manipulator.

When dir'manipulator restores the file it leaves the file type as DEL. The **change name/type** option can then be used to correct the file type. If you are not sure what the file type was originally, try SEQ and see if the file contains readable data. If it doesn't it was probably a PRG file.

After all the above has been done, F1 should be used to save the changes. Use PASS "v0" to validate the disk.

PROGRAM: directory'editor
AUTHOR: Robert Ross

Warning: This program works only with the 1541 disk drive, and only as unit 8. It will not work with the SX-64 or other drives.

This program provides you with many new easy to use commands allowing you to modify and change the disk directory. These commands include the following:

r read directory of disk
L list a range of entries
l list up to current entry
+ add 1 to current entry
- subtract 1 from current entry
C set new current entry
c copy one entry to another
i make space to insert entries
d delete entries (old entries saved at top)
W write entries, 1 to high
e edit an entry
s swap 2 entries
S sort entries*
p protect range of entries
u unprotect range of entries
z zero a range of entries
h set high entry
H help screen
RUN/STOP usually stops the program

* sort:
24 data bytes are stored for each entry; sorting may be done on from 1 to all 24 bytes of the entries. Sort direction (><) may be set for each byte: > means ascending order ("a" will preceed "b" after the sort). **width** (1-24) is the # of characters used for comparing entries. **1st** and **last** give the sort range. **Give width and range for each sort. specify** allows the order and direction of sorting to be set; the letters a to x stand for the 24 bytes:

a=file type
b,c=track,sector of first block
d-s=file name
t,u=track,sector for 1st side sector of a relative file
v=relative file record length
w,x=low byte,high byte of file length

Example to sort by decreasing file groups:

width=1
order="a..."
direction="<..."

names sets a default order with d-s at the start and all directions=">"

PROGRAM: disk'editor
AUTHOR: Ian MacPhedran

This is an easy to use, menu oriented program. As would be expected, disk'editor allows you to read, write, display, and edit sectors from a disk. The data can be displayed in either hex or decimal. It also has a feature to allow a sector chain to be followed.

The first thing to do is select the option to read a block from disk. Until this is done, don't try the write, display/edit, or read next block options.

If you tried using dir'manipulator to restore a scratched file and it said the file was not recoverable, you may have more luck with disk'editor. The most likely cause for dir'manipulator to indicate a file is unrecoverable is that part of it has been overwritten, but may still be recoverable. To find out how much of the file is still there, use disk'editor to find its starting track/sector from the directory entry, then follow the sector chain. You should have some idea of what the file contained. When you find the last 'good' sector of the file, terminate the sector chain there, and try using dir'manipulator to recover it again.

I suggest that anyone who is interested in what goes on inside their 1541 purchase a copy of 'Inside Commodore DOS'. At over 500 pages, it goes into detail about the inner workings of the drive, more than can be covered here.

PROGRAM: disk'edit/protect
AUTHOR: Glen Colbert and Dave Powell

This program allows you to protect your disk so that it cannot be written on by the disk drive. Plus it allows you various other options including:

A - Protect the disk
B - Unprotect the disk
c - Protect one file
D - Display DOS RAM
M - Display C64 Memory
S - Show sector
V - View BAM
Z - Allocate sector
SHIFT @ - Check all sectors on disk

PROGRAM: find
AUTHOR: Doug Weick

This program gives 0.14 users something almost as good as the **FIND** command in 2.0 COMAL. It doesn't work the same way and it has a few problems, but it is a whole lot better then nothing.

When the program is first run, it sets up some machine code and ties it into the interrupts. The ML does all the work. After it is in place, other programs can be loaded and the find function used with them.

To use the find function you first have to set the search string. This is done by hitting **CLR/HOME** and typing the string at the top of the screen. End the string with the **back arrow** key at the top left of the keyboard. The command **EDIT** will appear on the bottom line of the screen. Press return and the program will be listed, in edit mode. The lines the search string appears in will be flagged by a white back arrow just after the line number.

Before another search string can be specified, the current one must be cleared by pressing **CONTROL** and **C**. This should always be done when the cursor is on a blank screen line.

More detailed information is provided in the program in the comments beginning at line 1290.

PROGRAM: find'string/fast
AUTHOR: Jesse Knight

PROGRAM: find'string/full
AUTHOR: Ray Carter

Both of the programs do the same thing: locate a string variable in memory. The fast version is quicker but more limited than the full version. Both are demo programs for a single procedure. As they stand now they aren't extremely useful. However they could be used to improve other utility or applicatition programs. Here's what I'm talking about:

COMAL is good at string handling. A few simple commands allow you to dice, slice, chop, and even puree strings. In version 0.14, getting a single byte from a disk drive requires machine code. The procedures already written for this work very slowly, especially if a large number of bytes are to be transfered. Now what if someone (don't look at me) were to write the ML for getting a number of bytes into a string? It seems to me you would have the best of both worlds. The ML quickly transfers the data directly into a string. COMAL can then work wonders on the string. On that happy note, I'll stop.

PROGRAM: ml'setup
AUTHOR: David Stidolph
FILE: ml'procs

As the name **ml'setup** implies, it sets up some machine language code. The machine code performs assorted functions and can be accessed by POKE and SYS. Specific information is given when the program is run.

If you dread the POKE and SYS commands, ml'procs will make things easier for you. Ml'procs contains procedures to access all the functions of the machine code. The procedures and their functions are:

quote'mode(true/false)
true - enable quote mode
false - disable quote mode

clock'display(true/false)
true - display clock
false - don't display clock

sprite'flip'left'right(block)
flips defined sprite in block
left and right (mirror image)

sprite'flip'up'down(block)
flips defined sprite in block
up and down (inverts)

setup'irq // no parameters
ties code for quote mode and clock into interrupts

restore'irq // no parameters
removes code for quote mode and clock from interrupts

reverse'screen // no parameters
reverses the graphics screen
(bit map pixels are inverted)

save'screen(filename$)
saves hires screen (bitmap only) to filename$.
can be loaded back with obj'load

read'sprite'h(block)
reads hires sprite from data statements
and stores it in given block

read'sprite'm(block)
reads multicolor sprite from data statements
and stores it in given block

The procedures read'sprite'h and read'sprite'm use a similar data format. Instead of numbers, a character string is used for each pixel row. For hires sprites, a '.' is used for a transparent pixel, '#' is used for a colored pixel. For

multicolor sprites, each of the possible colors, transparent, multicolor register #1, sprite color register, and multicolor register #2, are indicated by the digits 0, 1, 2, and 3, respectively. If fewer than the maximum number of characters (24 for hires, 12 for multicolor) are used, the rest will default to transparent. If fewer than the maximum number of lines are used (21 for both), the rest will default to transparent. Regardless of the number of lines used, the last line for hires sprites should be "h" and the last line for multicolor sprites should be "m". Below are two examples.

```
  Multicolor:                    Hi-Res:

data "333333111111"          data "..########"
data "333333222222"          data "..##....##"
data "333333111111"          data "..########"
data "222222222222"          data "..##....##"
data "111111111111"          data "..##....##"
data "222222222222"          data "h"
data "111111111111"
data "m"
```

PROGRAM: names'printout
AUTHOR: David Stidolph

This program shows the names defined in a program and their type (procedure, function, real, integer, etc.). The program works on programs SAVEd to disk. To view the name table of a program, first run the program, then save it. Next load and run **names'printout**, giving it the file name of your program you previously saved.

PROGRAM: print'2'col'dir
AUTHOR: Ray Carter

This program prints a disk directory in two coloumns on the printer. It helps save paper and space for directory listings.

PROGRAM: remove comments
AUTHOR: UniCOMAL Aps?

This program will remove all comments from a COMAL program that has been listed to disk. When the program asks for the input file, respond with the program file name to remove the comments from. When the program asks for the output file, respond with the file to write the new program to.

PROGRAM: sd2'copier
AUTHOR: Colin Thompson

This program was written for the MSD dual drive. It will work with other dual drives such as the 4040. It is most useful when you have a large number of disks to copy. Three options are offered:

copy a file
duplicate a disk
view directory

They are selected by C, D, and the British pound symbol, respectively. Drive 0 is used as the source and drive 1 is used as the destination drive.

Because of limitations of COMAL 0.14, if the directory listing is slowed by using the control key, the program will stop. You may issue the command CON to continue the program if this happens.

PROGRAM: sd2'copy&label
AUTHOR: Kevin Quiggle

This program was written for the MSD dual drive but will work on the 4040 as well. For each source disk used (drive 0 for source), the disk name and ID is read. You are then asked how many copies you wish to make. The program gives a prompt for inserting each destination disk (in drive 1). The disk is duplicated and the disk name and ID are printed on the labels in the printer. The labels used should be one-across mailing labels.

PROGRAMS: speed'to'seq and seq'tospeed
AUTHOR: Phyrne Bacon

This set of programs allows you to convert a SPEEDSCRIPT text file into a normal SEQ type text file and back again. SEQ type text files are loadable by **EasyScript, PaperClip,** and other wordprocessors.

PROGRAM: sprite'converter
AUTHOR: David Stidolph

This program will convert sprite image files created by the Access Software program **Sprite Master** into normal COMAL sprite shape files.

PROGRAM: sprite'editor
AUTHOR: UniCOMAL & Len Lindsay

With this program you can create sprite images for use in COMAL programs. Some instructions are given at the start of the program. I want to mention a few things not covered there. The SAVE option saves the sprite data in a form that can be read from other programs by using READ FILE to read a 64 byte string. For an example look at the program code starting at line 1060. The append option can be used to store several sprites in a single file. Each time the append option is used, the data for the current sprite is written to the end of the file given.

PROGRAM: text'dump'ctl-p
AUTHOR: Doug Bruhn & Jesse Knight

There is just one input to answer before the text dump can be activated, that is to tell it which page (0-15) to use in the $C000 block of RAM. To make a long story short, I think that the safest place would be page 9 ($c900). That is just after where the error messages are placed (if you're using RAM error message option). I have noticed most COMAL 0.14 programs using ML place it either in the first few pages at $c000, or the last few, so page 9 should be out of the way of most of it.

Once the machine code is in place, pressing the control key and the P key causes the text screen to be printed on the printer.

SOME USEFUL PROCEDURES

PROCEDURE: buffer.proc
AUTHOR: David Stidolph
PROC BUFFER(A$)

The contents of A$ are placed in the keyboard buffer. The number of characters in the buffer is set to the length of A$. A$ should not contain more than 10 characters.

PROCEDURE: cat'.proc
AUTHOR: Mike Miller
PROC CAT'

This procedure allows the disk directory to be shown from inside a running program. There are no parameters. **Warning:** if the control key is pressed while the procedure is executing, the program will not continue after showing the disk directory.

PROCEDURE: joystick.proc
AUTHOR: Captain COMAL
PROC JOYSTICK (PORT,DIRECTION,FIRE)

The joystick (in PORT 1 or 2) is read and DIRECTION is set to the value for the current direction. Values for the directions are the same as used by the JOYSTICK package in the COMAL 2.0 Cartridge. FIRE is set to 1 if the fire button is pressed, 0 if not pressed.

The values and corresponding direction are:

0 no direction selected
1 up
2 up and right
3 right
4 right and down
5 down
6 down and left
7 left
8 left and up

PROCEDURE: load'obj.proc
AUTHOR: Phyrne Bacon
PROC LOAD'OBJ(NAME$)

The machine code or other data in the file NAME$ is loaded into memory to its original position.

PROCEDURE: paddle.proc
AUTHOR: Captain COMAL
PROC PADDLE(PORT,VAL1,VAL2,FIRE1,FIRE2)

The paddle pair (in PORT 1 or 2) is read. The values for the paddles are returned in VAL1 and VAL2. FIRE1 and FIRE2 will return 1 if the corresponding paddle button was pushed, or 0 if not.

PROCEDURE: plot'char.proc
AUTHOR: David Stidolph
FILES: convert
convert'setup
PROC PLOT'CHAR(ROW,COL,C$)

The text in C$ is plotted on the graphics screen at ROW,COL. ROW and COL correspond to the row and column on the text screen. They are automatically converted to graphic coordinates by the procedure. Unlike the PLOTTEXT command, PLOT'CHAR allows lower case charaters to be plotted on the graphics screen. The procedures CONVERT and CONVERT'SETUP are used by PLOT'CHAR to perform character conversions.

PROCEDURE: repeat'key.proc
AUTHOR: David Stidolph
PROC REPEAT'KEY(STATE)

If STATE is TRUE the repeat key feature is activated. If it is FALSE, the feature is turned off.

PROCEDURE: restore'lbl.proc
AUTHOR: John Eldredge
PROC RESTORE'LABEL(NAME$)

A RESTORE operation is performed to the line with the label in NAME$. An error is generated if NAME$ is not found or is not immediately followed by a DATA statement.

PROGRAM: tod.proc
AUTHOR: David Stidolph
PROC SET'TOD(HOURS,MINUTES,SECONDS,AM'PM)

The Time Of Day (TOD) clock is set to the time in HOURS, MINUTES, and SECONDS. If AM'PM is 0 the time is AM. If it is 1, then the time is PM. The tenths of a second register is set to 0.

PROC READ'TOD(HRS,MINS,SECS,TENTHS,AM'PM)

The TOD clock is read and the time is returned in HRS, MINS, SECS, and TENTHS. AM'PM will be 0 to indicate AM or 1 to indicate PM.

PROCEDURE: wait'n'go.proc
AUTHOR: Captain COMAL
PROC KEY'WAIT'N'GO(X,RESP$)

The procedure waits for a key to be pressed, returning it in RESP$. If a key is not pressed within X seconds, the procedure returns.

PRINTER UTILITIES

PROGRAM: dump'1525
AUTHOR: Eric Giguere

This program will dump a hires picture to a Commodore 1525 printer. After creating the picture, LOAD and RUN this program.

PROGRAM: dumpscreen'1525
AUTHOR: David Stidolph

This program produces a true 8 bit dump of a graphic screen on the 1525. It does not use machine code, so it's slow. LOAD and RUN this program after creating the picture.

PROGRAM: pretty'printer
AUTHOR: UniCOMAL

This program makes 'pretty' listings of COMAL programs listed to disk. The number of lines per page, margins, line length, and indentation are user selectable. Line numbers and a heading at the top of each page are optional.

PROGRAM: dual'epson'dump
AUTHOR: Jesse Knight

This program will produce either a single or double size hardcopy of a graphic screen. LOAD and RUN the program after the picture has been created.

PROGRAM: fx-80'cmds.proc
AUTHOR: Mark Finley

The procedures in this file provide an easy means to control the functions of the FX-80 printer. None of the procedures require any parameters. The procedures are:

BEEP
BACKSPACE
ENLARGE
UNENLARGE

CONDENSE
UNCONDENSE
UNDERLINE
NONUNDERLINE
EMPHASIZE
UNEMPHASIZE
BOLD
UNBOLD
ELITE
PICA
SUPERSCRIPT
SUBSCRIPT
ENDSCRIPT
PROPORTIONAL
UNPROPORTIONAL

PROGRAM: dump'nec8023a
AUTHOR: Tom Kuiper & Harry Seidman

This program dumps the current picture in memory to a NEC 8023 printer. It is written mostly in COMAL, although some Machine Language subroutines are used.

PROGRAM: nec'ml'dump
AUTHOR: Stephen Anson-Cartwright & Peter Foiles

This program dumps the current picture in memory to a NEC 8023 printer. It is written in machine code for speed. The file ML'DUMP.OBJ is loaded by this program.

PROGRAM: nec'comal'dump
AUTHOR: Captain COMAL and friends

This program dumps the current picture in memory to a NEC 8023 printer. It written entirely in COMAL. It takes about 17 minutes to print the picture.

PROGRAM: bigdump'nec.src

This is the source code for the NEC'ML'DUMP.

PROGRAM: dir'print'nec
AUTHOR: David Skinner

This program prints disk directories on the NEC or Prowriter printers. Besides the file names, types, and lengths, the starting track and sector is also given (although these are meaningless for COMAL program files). Multiple copies of the same directory can be printed without having to read it again.

PROGRAM: 8023p'options
AUTHOR: David Maven

This program provides an easy method of setting the options available on the Commodore 8023 printer.

PROGRAM: oki92'hi
AUTHOR: Craig Van Degrift

This program can be used to replace the normal HI program. It allows diferent print styles and quality to be selected for the Okidata 92 printer using either the Cardco or Turboprint interfaces.

PROGRAM: oki92'screen'io
AUTHOR: Craig Van Degrift
FILE: OKI92.DUMP.OBJ

This is a machine language screen dump for the Okidata 92 printer. As the program is now, the two files 1ST 80 KANJI.HRG and 2ND 80 KANJI.HRG are loaded and printed. The file OKI92.DUMP.OBJ contains the Machine Language used by this program.

PROGRAM: gem10x'lister
AUTHOR: T. S. Creasy

This program will list either saved or listed COMAL 0.14 programs to the Gemini 10x printer. The printing is done using elite characters and the skip over perforation in effect. It may be helpful to save this program on the same disk as any files to be listed since the file to be listed is either ENTERed or LOADed, overwriting the program in

memory. After the listing is finished, pressing return will chain the lister program again. More instructions are in the program.

PROGRAM: print'calendar
AUTHOR: Kevin Quiggle

This program will print a calendar for any year using a Gemini 10x printer and Card?/A interface. The calendar can be printed starting and ending with any month. Three print quality options are offered. The higher the quality, the slower the printing.

PROGRAM: bit'map'print.l
AUTHOR: Tom Kiuper

This is a procdure for printing the graphic screen to a Gemini printer with a Cardco interface. A feature is included to allow labels to be printed for the x and y axes on graphs. To use the procedure, merge this file with your program and add the following statements to your program:

```
DIM XLABEL$ of 40, YLABEL$ OF 40
READ'HIDDEN'RAM'SETUP
*** your code to draw the picture ***
XLABEL$="** the label you want **"
YLABEL$="** the label you want **"
BIT'MAP'PRINT(XLABEL$,YLABEL$) // print the picture
```

PROGRAM: imp'dump
AUTHOR: Captain COMAL and friends

This program prints the Hi-Res screen to the Impact Matrix Printer (IMP) made by Fidelity Electronics. The resolution is only 320 by 144 so part of the picture is lost.

PROGRAM: dump'prowriter
AUTHOR: Tom Kiuper and Harry Seidman

This program prints the Hi-Res screen to a Prowriter printer. An option is given for the screen to be reversed.

PROGRAM: 1520/0.14demo
AUTHOR: Kevin Quiggle

These programs produce a variety of images on the plotter. If you own a 1520 plotter, run these programs and see some nice results.

There are 14 demo programs. Each has a number from 1 to 14 added at the end of its name (1520/0.14demo1 on through 1520/0.14demo14).

PROGRAM: 1520'driver.proc
AUTHOR: Kevin Quiggle

This file contains procedures for driving the 1520 plotter. These procedures allow you to control various aspects of the Commodore 1520 plotter.

TABLE OF CONTENTS

SOME USEFUL PROCEDURES

PRINTER UTILITIES

ISBN 0-928411-07-9