NOAA Technical Memorandum NMFS

This TM series is used for documentation and timely communication of preliminary results, interim reports, or special purpose information; and have not received complete formal review, editorial control, or detailed editing.

NOVEMBER 1990

A PERSONAL COMPUTER BASED SYSTEM FOR ANALOG-TO-DIGITAL AND SERIAL COMMUNICATION DATA ACQUISITION

Robert C. Holland

National Oceanic and Atmospheric Administration
National Marine Fisheries Service
Southwest Fisheries Science Center
P.O. Box 271
La Jolla, California 92038

NOAA-TM-NMFS-SWFSC-152

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
National Oceanic and Atmospheric Administration
John A. Knauss, Under Secretary for Oceans and Atmosphere
National Marine Fisheries Service
William W. Fox, Jr., Assistant Administrator for Fisheries

NOAA Technical Memorandum NMFS

The National Oceanic and Atmospheric Administration (NOAA), organized in 1970, has evolved into an agency which establishes national policies and manages and conserves our oceanic, coastal, and atmospheric resources. An organizational element within NOAA, the Office of Fisheries is responsible for fisheries policy and the direction of the National Marine Fisheries Service (NMFS).

In addition to its formal publications, the NMFS uses the NOAA Technical Memorandum series to issue informal scientific and technical publications when complete formal review and editorial processing are not appropriate or feasible. Documents within this series, however, reflect sound professional work and may be referenced in the formal scientific and technical literature.

NOAA Technical Memorandum NMFS

NOVEMBER 1990

A PERSONAL COMPUTER BASED SYSTEM FOR ANALOG-TO-DIGITAL AND SERIAL COMMUNICATION DATA ACQUISITION

Robert C. Holland

NOAA-TM-NMFS-SWFSC-152

U.S. DEPARTMENT OF COMMERCE
National Oceanic and Atmospheric Administration
National Marine Fisheries Service
Southwest Fisheries Science Center

A PERSONAL COMPUTER BASED SYSTEM FOR ANALOG-TO-DIGITAL AND SERIAL COMMUNICATION DATA ACQUISITION

by

Robert C. Holland
Southwest Fisheries Science Center
National Marine Fisheries Service
P.O. Box 271
La Jolla, CA. 92038

TABLE OF CONTENTS

INTRODUCTION 1

PCPLUSS 2

OPERATION 2
- Standard Mode 2
- Waiting Mode 3
- Blank Mode 3

PROGRAM DEFINITIONS 3
- Configuration 3
  - BOD1 3
  - BOD2 3
  - COMM 4
  - FORM 4
  - DIRC 4
  - GAIN 4
  - ADDR 4
  - TICK 4
- PCPLUSS Files 4
  - CONFIGUR.EXE 4
  - PCPLUSS.BAS 4
  - PCPLUSS.MAP 5
  - PCPLUSS.EXE 5
  - PCPLUSS.INF 5
  - PCPLUSSM.WP5 5
- AIO8 5

EXP-16 6
- ODEC Thermosalinograph 6
- Output 7

ACKNOWLEDGEMENTS 8

REFERENCES 8

APPENDIX 1 - Specifications 9
- Fluorometer 9
- EXP-16 Multiplexer 9
- AIO8-B Analog-Digital Board 10

APPENDIX 2 - Program PCPLUSS.BAS 11

APPENDIX 3 - Program INSTALL.PAS 20

APPENDIX 4 - Program CONFIGUR.PAS 25

INTRODUCTION

PCPLUSS and the companion programs were developed to sample continuous and discrete data for a variety of environmental sensors during transits of a research vessel; it was specifically designed for the MOPS (Monitoring Of Porpoise Stocks) program to utilize an inexpensive personal computer (PC), but it also has potential use in any R/V oceanographic program due to its inherent expandibility.

PCPLUSS is a compiled BASIC[1] program designed to sample, average and store multiple streams of data from a variety of sources. It is presently configured to directly access an **AIO8**[2] analog-to-digital (A/D) board and convert analog data from a **Turner Designs**[3] Fluorometer (flow-thru model, series 10) to a 12-bit digital form, collect temperature and salinity data from an ODEC Thermosalinograph through the communications port #1 (COM1:), and to collect satellite navigation (**SATNAV**) latitude and longitude locations through the communications port #2 (COM2:). The input data are averaged and stored as: date, time, latitude, longitude, temperature, salinity, fluorescence voltage, fluorometer scale, mode of operation, and speed indicator. The program can be modified to handle other analog data inputs as needed.

Data are collected at 1 second intervals and are averaged over 2 minutes (default). The data file is opened only when data are to be written, thereby safeguarding data integrity from power loss or equipment malfunction. The **AIO8** is an 8-bit, XT style board that uses an expansion slot in a PC (Personal Computer). The **AIO8** can be attached to an **EXP-16**[2] multiplexer board through an external cable to allow multiple analog inputs.

The **EXP-16** can be cascaded with 7 other multiplexer boards to provided a maximum of 128 current or voltage analog inputs. Turbo BASIC can operate 2 communication ports (COM1 and COM2). Thus, **PCPLUSS** can control 130 data inputs, although the 1 second interval may not be possible on a computer with an effective clock rate of less than 8 MHz. (megahertz) due to the constraints of data input/output and array management.

PCPLUSS has been executed successfully with 5 inputs (3 analog, 2 serial) sampled at 1 second intervals on a 4.77 MHz. COMPAQ portable. A math coprocessor (Intel 8087/80287) must be used since high-speed math calculations are required in **PCPLUSS**. An AT class computer (80286/80386 microprocessor) with an effective clock rate of 12 MHz. (or better) should be sufficient for any array of

---

[1]Turbo BASIC 1.1. Copyright Borland International, Inc.

[2]Industrial Computer Source, San Diego, CA. 92123.

[3]Turner Designs, Mountain View, Ca. 94043.

data inputs.

PCPLUSS should be installed on a hard disk with an average seek time of 65 ms. (milliseconds) or better. **PCPLUSS** can be installed on a bootable floppy diskette, but floppy drives are extremely slow and are often less reliable than a hard disk (or hard card).

PCPLUSS

The following files are necessary for the operation of **PCPLUSS**:

CONFIGUR.EXE	- Reconfigures the **PCPLUSS.INF** file.
PCPLUSS.BAS	- The source program.
PCPLUSS.MAP	- The graphics screen cruise track map.
PCPLUSS.INF	- The configuration file.
PCPLUSS.EXE	- The executable program.
PCPLUSSM.WP5	- This document in **WordPerfect 5.1** format.

The program will locate the necessary information and the data files will be written to the pre-configured directory.

OPERATION

When the program is running, it is in the **Standard Mode** (record type = 1) of data acquisition. The date, time, temperature, salinity, fluorometer voltage, fluorometer scale, and record type indicator are constantly shown and updated. Also, the output data file name is shown.

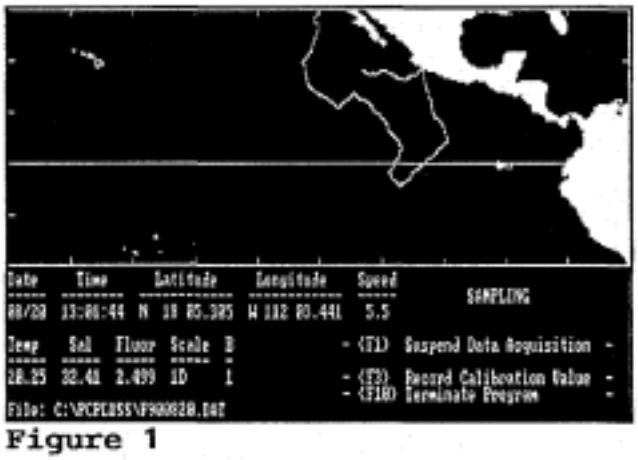


Figure 1

Figure 1 shows the normal operation of the **PCPLUSS** graphics screen with the cruise track map shown on the top half of the screen and the collected data shown on the bottom half of the screen. The line bisecting the map is the equator. The current map dimensions are:

Latitude: 20 South to 30 North
Longitude: 170 West to 70 West

In the **Standard Mode**, there are three function keys available:

F1 - Suspend Data Acquisition

F3 - Record Calibration Value
F10 - Terminate Program

Pressing **F1** places you in the **Waiting Mode**, where only two function keys are now available:

F1 - Restart Data Acquisition
F2 - Record Blank Value

Pressing **F1** places you back in the **Standard Mode**, while pressing **F2** places you in the **Blank Mode**, which will run data acquisition for 30 seconds (record type = 2) and place you back in the **Waiting Mode**. This will continue until you press **F1** to go back to the **Standard Mode**. The **Blank Mode** is used to blank the fluorometer at one or more scale settings (Smith et al., 1981).

In the **Standard Mode**, pressing **F3** will clear the function keys and mark the current data record as a calibration record (record type = 3). **F3** is pressed when a water sample is collected from the fluorometer for discrete chlorophyll analysis and used to later calibrate the data collected by **PCPLUSS** (Smith et al., 1981). At the start of the next data acquisition interval, **PCPLUSS** will return to the **Standard Mode**.

In the **Standard Mode**, pressing **F10** will immediately terminate the program and place you back at the **DOS** (Disk Operating System) prompt.

PROGRAM DEFINITIONS

The default **PCPLUSS.INF** configuration settings are:

```
BOD1:COM1:300,E,7,2
BOD2:COM2:110,N,8,2
COMM:014,000,001,005,007,005
FORM:\   \ \      \ ###.### ###.### ##.## ##.## #.### \  \ #\\
DIRC:C:\PCPLUSS
GAIN:2
ADDR:768
TICK:120
```

Configuration

BOD1 specifies the communications port #1 (**COM1**) used by the ODEC Thermosalinograph, the baud rate (**300**), parity (**Even**), data bits (**7**), and the number of stop bits (**2**).

BOD2 specifies the communications port #2 (**COM2**) used by the GPS SATNAV, the baud rate (**110**), parity (**None**), data bits (**8**), and the number of stop bits (**2**).

COMM specifies the number of characters for 1 serial input (014), the ASCII start character (000), the position of the beginning character of temperature after the start character (001), the number of characters representing temperature (005), the position of the beginning character of salinity after the start character (007), and the number of characters representing salinity (005). Each part (6 parts) MUST be present and each part MUST be 3 characters long, each separated by a comma. The current **BAUD** and **COMM** strings represent the output of the ODEC thermosalinograph (see **ODEC Thermosalinograph** for more information on the ODEC output). These strings must be changed for other forms of serial data input.

FORM specifies the TIME and DATE fields (\ \ \ \), the latitude and longitude (###.### ###.###), the temperature and salinity fields (##.## ##.##), the fluorometer voltage (#.###), the fluorometer scale (\ \), the blanking indicator (#), and the ship speed indicator (\\) (see **Output** for more information).

DIRC specifies the current directory where the data will be placed when collected (C:\PCPLUSS). This is determined during installation.

GAIN specifies the gain setting on the **EXP-16** multiplexer board. This setting is used on all input signals. The **GAIN** value in **PCPLUSS.INF** MUST be the same on the **EXP-16** board to insure correct input values. This dip switch setting on the **EXP-16** board should only be changed by a knowledgeable technician with the proper **EXP-16** manual.

ADDR specifies the **AIO8** dip switch setting for the Input/Output port (768). The **ADDR** value in **PCPLUSS.INF** MUST be the same on the **AIO8** board to insure correct input values. This dip switch setting on the **AIO8** board should only be changed by a knowledgeable technician with the proper **AIO8** manual.

TICK specifies the number of seconds to average the input data (120). This does not change the blanking period (30 seconds).

PCPLUSS Files

CONFIGUR.EXE allows you to change any of the above settings without changing the program itself. To run the program, just type: **CONFIGUR**↵ (you do not have to be in the **PCPLUSS** directory to run **CONFIGUR**). Then enter the number corresponding to the line you wish to edit. When finished, just press <**ENTER**>. **CONFIGUR.EXE** will locate and change the **PCPLUSS.INF** file in the **PCPLUSS** directory.

PCPLUSS.BAS is the source program in Turbo BASIC. If a major problem occurs where the program needs to be changed, consult a programmer before any changes are made, and the executable program is changed.

PCPLUSS.MAP is the cruise track map file that is updated every time new latitude and longitude data are received. The map is saved each time the program is terminated normally. If the computer is turned off while **PCPLUSS** is running, all visual cruise track map data is lost since the last time **PCPLUSS** was terminated normally.

PCPLUSS.EXE is the executable program that collects the data. To run, just type: **PCPLUSS**↵ (you do not have to be in the **PCPLUSS** directory to run **PCPLUSS**).

PCPLUSS.INF is the configuration file that contains the changeable input/output and timing parameters of **PCPLUSS**.

PCPLUSSM.WP5 is the document you are reading now. It was written using WordPerfect 5.1.

AIO8

There are 3 channels currently being accessed by the **AIO8** on the **EXP-16**: 0) Fluorometer Scale, 1) Fluorescence Voltage, and 2) Voltage of x1/x100 knob. All channels have inputs ranging from 0 to 1 volt. Thus the calculations in the program for the on-screen and in-file data are calculated as:

The input from the **AIO8** board is in digital form representing a 10 volt range (= 12 bits). This number is then multiplied by 5 (volts) and divided by its digital equivalent 2048 (to put the equation into a volt unit). This value is then subtracted by 5 (to put the whole equation into a -5 to 5 volt range and then the whole equation is divided by the gain setting on the **EXP-16** board to represent the actual input voltage on the **EXP-16** (see equation (1) and table (1))).

$$INPUT[volts] = \frac{INPUT[digital] * \frac{5[volts]}{2048[digital]} - 5}{GAIN} \qquad (1)$$

Table (1):

Binary	Hex	Analog Input Voltage
0000 0000 0000	000	-5.0000 v (-Full scale)
0000 0000 0001	001	-4.9976 v
. . .	.	.
0100 0000 0000	400	-2.5000 v (-½ scale)
. . .	.	.

Binary	Hex	Analog Input Voltage
1000 0000 0000	800	±0 v (zero)
1000 0000 0001	801	+0.0024 v
. . .	.	.
1100 0000 0000	C00	+2.5000 v (+½ scale)
. . .	.	.
1111 1111 1111	FFF	+4.9976 v (+Full scale)

EXP-16

The **EXP-16** has 8 channels (0 - 7) which can be multiplexed through the **AIO8** board. Each channel has "signal hi" and "signal lo" inputs for devices with grounded signals. Also, there is an **EXP-16** ground associated with each channel for floating signals (all fluorometer inputs have grounded (return) signals). There may need to be a resistor (load) across each "hi" and "lo" channel to scale each input signal to a proper voltage range.

Each channel is then multiplexed through gain circuitry for input signal amplification. The **EXP-16** has 8 gain settings: 0.5, 1, 2, 10, 50, 100, 200, 1000. Since the fluorometer inputs are in the range 0 - 1 volt, a gain setting of 2 can be used for each signal (since the AIO8 can only digitize -5 to 5 volts, a gain setting of 10 can not be possible).

ODEC Thermosalinograph

The temperature and salinity are accessed from the **ODEC** thermosalinograph device through the communications port at baud 300, 7 data bits, even parity, and two stop bits. The form of the input is as follows:

<null>TT.TT<sp>SS.SS<cr><lf> ◄= 014 characters

where **<null>** is the null character **000** (start character), **TT.TT** is the input temperature which begins **001** character after the start character and is **005** characters long, **<sp>** is a space, **SS.SS** is the input salinity which begins **007** characters after the start character and is **005** characters long, **<cr>** is a carriage return, and **<lf>** is a line feed. The **ODEC** sends the data constantly, but the program will only access the data once a second (and empty the communications buffer for new data). With the above information, the line in the **PCPLUSS.INF** file is: **COMM:014,000,001,005,007,005.**

Output

The averaged data are stored in a data file in a format specified by **FORM**:

MMDD HH:MM:SS LLL.LLL lll.lll TT.TT SS.SS F.FFF SSSS B S

where **MMDD** is the month and day of the current file, **HH:MM:SS** is the hours, minutes and seconds, **LLL.LLL** is the latitude, **lll.lll** is the longitude, **TT.TT** is temperature, **SS.SS** is salinity, **F.FFF** is fluorescence voltage, **SSSS** is the scale set on the fluorometer with the following possible scale values:

1A	= x1	x1
1B	= x1	x3.16
1C	= x1	x10
1D	= x1	x31.6
100A	= x100	x1
100B	= x100	x3.16
100C	= x100	x10
100D	= x100	x31.6

and **B** is the record type where:

1 = Normal mode
2 = Blank (blanking mode)
3 = Calibration

and **S** is the speed indicator where:

U = Ship speed ≥ 1 knot.
D = Ship speed < 1 knot.
= Did not receive SATNAV information

The output file name is in the form:

FYYMMDD.DAT

where **F** stands for File, **YY** is the year, **MM** is the month, and **DD** is the day (the extension **.DAT** refers to it as a data file).

In a normal day's operation, a file will be about 41Kb in size. Therefore, when backing up the data, about 8 days of data can fit onto one 360Kb-5¼ inch diskette.

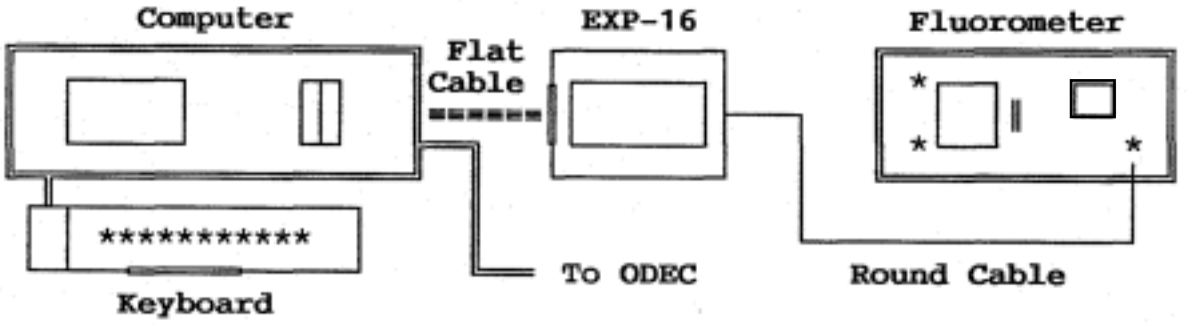
ACKNOWLEDGEMENTS

I would like to thank Dr. Stephen Reilly and Dr. Paul Fiedler for their guidance and insight into both the hardware and software aspects of this project. Their continued support engendered this project to be both viable and successful.

REFERENCES

Smith, R.C., K.S. Baker, and P. Dustan. 1981. Fluorometric techniques for the measurement of oceanic chlorophyll in the support of remote sensing. SIO Ref. 81-17, 14 pp.

APPENDIX 1 - Specifications



Fluorometer

Description	Pin	Voltage Range
Analog Output Signal	Pin M (w/ 1kΩ resistor)	0 - 1 Volt
Analog Range Output	Pin N (w/ 1kΩ resistor)	0.0v = x0 0.4v = x3.16 0.7v = x10 1.0v = x31.6
Analog Precision Ground	Pin L	
TTL x1/x100 knob	Pin S (w/ 1.1kΩ resistor)	0.0v = x100 0.5v = x1
TTL Return Ground	Pin D	

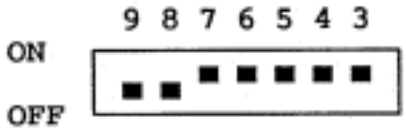
EXP-16 Multiplexer

Input (fluorometer)	Channel (EXP-16 Board)
Pin N	Channel 0 (hi)
Pin M	Channel 1 (hi)
Pin S	Channel 2 (hi)
Pin L	Channel 0 (lo) Channel 1 (lo)
Pin D	Channel 2 (lo)

1) 1k resistor across channel 0 (hi) and channel 0 (lo)
2) 1k resistor across channel 1 (hi) and channel 1 (lo)
3) 1.1k resistor across channel 2 (hi) and channel 2 (lo)
4) Toggle switch on gain control set to gain of 2

AIO8-B Analog-Digital Board

AIO8 board set on base address of 768 (300 Hexadecimal).



APPENDIX 2 - Program PCPLUSS.BAS

Program used to collect input data from sensors
(written in Turbo BASIC 1.1 (Borland Intl.))

```
'*************************************************************************
'*                                                                       *
'*    PCPLUSS.BAS (V. 3G) - DATA LOGGING PROGRAM FOR AIO8 BOARD          *
'*                                                                       *
'*                                                / Version 1  06-06-87  *
'*    Row-Bare Software, 1987.                   /  Version 2  06-24-88  *
'*                                               \  Version 3  03-14-89  *
'*                                                \ Version 3G 02-23-90  *
'*************************************************************************
     (Row-Bare Software is a trademark of Robert C. Holland)

'---------- Description of Program ----------------------------------
     This program sets up the AIO8 to log data from the first 3 analog
channels (no digital inputs): 0) Fluorometer scale, 1) Fluorescence,
2) Voltage of x1 & X100 knob. The data is transferred to a random
access data file on disk at 2 minute intervals.

DIM DI%(3)                              'Dimension channel data array

'                    Set up EVENT traps

ON KEY(1) GOSUB Waiting                 'Function key 1 waits for blanking
ON KEY(2) GOSUB BlankOn                 'Function key 2 turns blanking ON
ON KEY(3) GOSUB Calibrate               'Function key 3 sets calibration ON
ON KEY(10) GOSUB GoodNight              'Ends program with Function key 10
CLS

'              Get Program Information from information file

BOD1$ = "COM1:300,E,7,2"                'COM1 Port, Baud, Parity, Data, Stop
BOD2$ = "COM2:110,N,8,2"                'COM2 Port, Baud, Parity, Data, Stop
COMM$ = "014,000,001,005,007,005"       'Initial temp/sal input form
FORM$ = "\    \ \       \ ###.### ###.### ##.## ##.## #.### \   \ #\\"
FORM$ =    "\    \ \       \ ###.### ###.### ##.## ##.##\\"
DIRC$ = ""                              'Initial output file directory string
ADDR% = 768                             'Initial Base address for AIO8 board
GAIN% = 2                               'Initial EXP-16 gain setting
TICK% = 120                             'Initial seconds count between writes

PCPLUSS$ = ENVIRON$("PCPLUSS") + "\PCPLUSS.INF"
OPEN PCPLUSS$ FOR APPEND AS #1
```

```
CLOSE #1
OPEN PCPLUSS$ FOR INPUT AS #1           'Open PCPLUSS information file
WHILE NOT EOF(1)
  LINE INPUT #1,INSTRING$               'Get input string from input file
  T$ = MID$(INSTRING$,6,LEN(INSTRING$) - 5)
  SELECT CASE LEFT$(INSTRING$,5)
    CASE "BOD1:"                        'If COM1 input, set BOD1 string
      BOD1$ = T$
    CASE "BOD2:"                        'If COM2 input, set BOD2 string
      BOD2$ = T$
    CASE "COMM:"                        'If COM1 input form, set COMM string
      COMM$ = T$
    CASE "FORM:"                        'If form input, set form string
      FORM$ = T$
    CASE "DIRC:"                        'If directory input, set DIRC string
      DIRC$ = T$ + "\"
    CASE "ADDR:"                        'If PCPLUSS address, set address port
      ADDR% = VAL(T$)
    CASE "GAIN:"                        'If EXP-16 gain, then set gain value
      GAIN% = VAL(T$)
    CASE "TICK:"                        'If seconds tick, set TICK variable
      TICK% = VAL(T$)
  END SELECT
WEND
CLOSE #1

NUMCHARS% = VAL(MID$(COMM$,1,3))        'Number of characters from TempSal
STARTCHAR% = VAL(MID$(COMM$,5,3))       'Character that signifies start
TEMPCHAR% = VAL(MID$(COMM$,9,3))        'Position of temp after STARTCHAR%
TEMPLENGTH% = VAL(MID$(COMM$,13,3))     'Number of characters for Temperature
SALCHAR% = VAL(MID$(COMM$,17,3))        'Position of sal after STARTCHAR%
SALLENGTH% = VAL(MID$(COMM$,21,3))      'Number of characters for Salinity
NUMCHARS% = NUMCHARS%*2                 'Double NUMCHARS% so to include 1 line
SATNAV$ = ""
LAT$ = "#  ## ##.###"                   'Latitude indicator
LON$ = "# ### ##.###"                   'Longitude indicator
SPEED$ = "##.#"                         'Ship speed indicator
SPD$ = " #"                             'Output speed indicator
OPEN BOD1$ + ",DS,RS,CS,CD" FOR INPUT AS #3   'Open COM1 port for TS input
OPEN BOD2$ + ",DS,RS,CS,CD" FOR INPUT AS #4   'Open COM2 port for SATNAV
$COM1 2048                              'Set up 2k buffer for COM1 port
$COM2 2048                              'Set up 2k buffer for COM2 port

'
'   Turn keys ON to allow traps to happen
'

KEY(1) ON                               'Turn Waiting key F1 ON
KEY(3) ON                               'Turn Calibrate key F3 ON
KEY(10) ON                              'Turn Exit key F10 ON
```

Determine Screen characteristics (Monochrome or Color)

```
CLS
MAP%=0
REG 1,&H0F00                              'Set register AH with 0F
CALL INTERRUPT &H10                       'Call BIOS interrupt for screen ID
IF (REG(1) AND &H00FF) <> 7 THEN          'If AL=7 then Monochrome, else color
  SCREEN 2                                'Choose 640x200 resolution
  DEF SEG = &HB800                        'Set segment pointer to screen memory
  BLOAD DIRC$ + "PCPLUSS.MAP",0           'Load ETP map
  WINDOW (-170,30) - (-70,-50)            'Define screen boundaries
  FOR II%=17 TO 25
    LOCATE II%,1
    PRINT SPACE$(80);                     'Clear old graphics below map
  NEXT II%
  MAP%=1                                  'Enable cruise track to be plotted
END IF                                    '  if in color mode (AL <> 7)
```

Get name of data file

```
ODAT$ = DATE$                             ' if file already exists
FILE$ = "F" + MID$(ODAT$,9,2) + LEFT$(ODAT$,2) + MID$(ODAT$,4,2) + ".DAT"
OPEN DIRC$ + FILE$ FOR APPEND AS #1       'Create the data file then close it
CLOSE #1
ON ERROR GOTO Errorcom                    ' to occur with a communication error
```

Initialize Screen with colors and text

```
LOCATE 17,1
PRINT "Date      Time       Latitude      Longitude     Speed"
PRINT "-----  --------  ------------  ------------  -----"
LOCATE 21,1
PRINT "Temp     Sal    Fluor  Scale  B"
PRINT "-----  -----  -----  -----  -"
PRINT "Temp     Sal"
PRINT "-----  -----"
LOCATE 25,1
PRINT "File: ";DIRC$ + FILE$;

Start:
 LOCATE 21,44
 PRINT "- <F1>  Suspend Data Acquisition  -";
 LOCATE 22,44
 PRINT "                                  ";
 LOCATE 23,44
 PRINT "- <F3>  Record Calibration Value  -";
```

```
  LOCATE 24,44
  PRINT "- <F10> Terminate Program          -";

'
'                    Initialize Working Variables
'

  COL% = 0                                  'Variable for count of samples taken
  TSCOL% = 0
  BLANK% = 1                                'Blanking occurs when BLANK% = 0
  FLUOR = 0                                 'Set fluor, temp, and sal to 0 for
  TEMP = 0                                  ' sampling
  SAL = 0
  T = 0                                     'Set temp and sal values to 0
  S = 0
  SECONDS% = TICK%                          'Number of seconds to sample data
  BLANK$ = " SAMPLING "                     'Set flashing indicator to "SAMPLING"
  BLKPRESSED% = 1                           'Indicator for blanking key pressed
  OLDTIME$ = TIME$                          'Save current time for checking later

'
'          Fetch A/D data from channels 0 thru 2 and return in DI%(*)
'

MainLoop:
 $EVENT OFF                                 'Temporarily turn off all event traps
  LOCATE 18,60
  PRINT BLANK$;                             'Print "SAMPLING"

  FOR I% = 0 TO 2
    ADDRESS% = I%*16                        'Set multiplexer channel to 0 and
    OUT (ADDR% + 2),ADDRESS%                ' use analog channels 0 - 2
    OUT (ADDR% + 1),0                       'Start 12-bit conversion
    XL% = INP(ADDR%)                        'Get low byte of data
    XH% = INP(ADDR% + 1)                    'Get high byte of data
    DI%(I%) = XH%*16 + XL%\16               'Combine high and low bytes
  NEXT I%
  INCR COL%                                 'Increment collector for sampling #

'
'                        Get time and date
'

  TI$ = TIME$
  DAT$ = DATE$
  DAT$ = MID$(DAT$,1,2) + "/" + MID$(DAT$,4,2)  'Set in MM/DD

'
'                      Display current data
'

  C0 = (DI%(0)*5!/2048! - 5)/GAIN%          'Convert digital data to voltage
```

```
C1 = (DI%(1)*5!/2048! - 5)/GAIN%          '  representation of input values
C2 = (DI%(2)*5!/2048! - 5)/GAIN%
IF C1 < 0 OR C1 > 11 THEN C1 = 0          'In case fluorometer is not working
FLUOR = FLUOR + C1                        'Add latest value of fluorescence and
C1 = FLUOR/COL%                           '  average over the number collected
```

Determine x1/x100 knob setting and fluorometer scale

```
IF C2 > 0.25 THEN SCALE$ = "1" ELSE SCALE$ = "100"
SELECT CASE C0
  CASE > 0.85
    SCALE$ = SCALE$ + "D"
  CASE > 0.55
    SCALE$ = SCALE$ + "C"
  CASE > 0.2
    SCALE$ = SCALE$ + "B"
  CASE ELSE
    SCALE$ = SCALE$ + "A"
END SELECT
IF COL% = 1 THEN OLDSCALE$ = SCALE$
```

Get temperature, salinity, and SATNAV information

```
IF LOC(3) > NUMCHARS% THEN                'If NUMCHARS% are present then
  TEMPSAL$ = INPUT$(LOC(3),#3)            '  get those characters
  POSITION% = INSTR(1,TEMPSAL$,CHR$(STARTCHAR%))      'Find the start
  TEMP = TEMP + VAL(MID$(TEMPSAL$,POSITION% + TEMPCHAR%,TEMPLENGTH%))
  SAL = SAL + VAL(MID$(TEMPSAL$,POSITION% + SALCHAR%,SALLENGTH%))
  INCR TSCOL%                             'Increment the temp sal counter
  T = TEMP/TSCOL%                         'Get the on-going temperature and
  S = SAL/TSCOL%                          '  salinity for the screen
END IF
GOSUB Satnav
```

Output the data to screen

```
LOCATE 19,1
PRINT USING "\    \  \       \";DAT$;TI$;
LOCATE 23,1
PRINT USING "##.##  ##.##  #.###  \  \   #";T;S;C1;SCALE$;BLANK%;
PRINT USING "##.##  ##.##";T;S;
```

Write data to disk

```
$EVENT ON                                    'Turn on event trap for keyboard keys
 IF (COL% < SECONDS%) AND (BLKPRESSED%) AND (SCALE$ = OLDSCALE$) THEN EndScan
 IF COL% < SECONDS% THEN EndScan
 IF SCALE$ <> OLDSCALE$ THEN                 'Has the scale changed since last
   C1 = OLDC1                                '  sample? If so, then get the last
   SCALE$ = OLDSCALE$                        '  value of fluorescence
 END IF
 COL% = 0                                    'Reset the Fluoresence and tempsal
 TSCOL% = 0                                  '  counters to 0 for more data
 OPEN DIRC$ + FILE$ FOR APPEND AS #1         'Temporarily open file for output
 PRINT #1,USING FORM$;DAT$;TI$;LAT;LON;T;S;SPD$
 CLOSE #1                                    '  then close the file immediately
 IF BLANK% = 3 THEN                          'If in calibration mode, then reset
   BLANK% = 1                                '  back to sampling mode, and turn
   KEY(1) ON                                 '  trap keys back on
   KEY(3) ON
   KEY(10) ON
   GOTO Start
 END IF
 IF BLKPRESSED% = 0 THEN WaitBlank           'If the Blank key has been pressed or
 IF SECONDS% = 30 THEN WaitBlank             ' 30 seconds elapsed, go to WaitBlank

VarReset:
 COL% = 0                                    'Reset input variables and collection
 TSCOL% = 0                                  '  counters
 FLUOR = 0
 TEMP = 0
 SAL = 0
 LAT$ = "#   ## ##.###"
 LON$ = "# ### ##.###"
 SPEED$ = "##.#"
 SPD$ = " #"

EndScan:
 OLDSCALE$ = SCALE$                          'Make old scale equal new scale and
 OLDC1 = C1                                  ' old fluorescence to new fluorescence
 LOCATE 18,60
 PRINT SPACE$(20);                           'Remove sampling marker

'
'          Delay for sampling interval
'

 DO
 LOOP UNTIL OLDTIME$ <> TIME$

 OLDTIME$ = TIME$
 C$ = INKEY$                                 'Remove any unwanted keys pressed

'
'          Loop back for next data
'
```

```
GOTO MainLoop
```

Event Key Traps Modules Begin here (Trap routines are further below)

```
WaitBlank:
 BLKPRESSED% = 1                          'Reset blank pressed key to 1
 BLKON% = 0                               ' and wait until <F2> is pressed
 KEY(1) ON
 KEY(2) ON                                'Enable <F2> and disable <F3> and <F10>
 KEY(3) OFF
 KEY(10) OFF
 LOCATE 21,44
 PRINT "- <F1>  Restart Data Acquisition  -";
 LOCATE 22,44
 PRINT "- <F2>  Record Blank Value        -";
 LOCATE 23,44
 PRINT "                                   ";
 LOCATE 24,44
 PRINT "                                   ";

 LOCATE 18,60
 PRINT " WAITING  ";

 Waitloop:
   OLDTIME$ = TIME$
   DO
     INTTIME$ = TIME$
   LOOP UNTIL INTTIME$ <> OLDTIME$
   IF LOC(3) > 0 THEN TEMPSAL$ = INPUT$(LOC(3),#3)     'Clear buffer
   GOSUB Satnav                           'See if any SATNAV information
   IF BLKON% THEN Blanking                'If <F2> pressed then start Blanking
 IF BLKPRESSED% THEN Waitloop             'If <F1> wasn't pressed then wait
 BLKPRESSED% = 1                          'Reset <F1> pressed variable
 KEY(1) ON                                'Restore previous key configuations
 KEY(2) OFF
 KEY(3) ON
 KEY(10) ON
 GOTO Start

Blanking:
 KEY(1) OFF                               'Disable <F1> and <F2> while blanking
 KEY(2) OFF
 LOCATE 21,44
 PRINT "                                   ";
 LOCATE 22,44
 PRINT "                                   ";
 SECONDS% = 30                            'Blank for 30 seconds
 BLANK% = 2                               'Show that program is in blanking mode
 BLANK$ = "BLANKING "                     '   and flash a "BLANKING" message
 GOTO VarReset
```

```
'
'                    Satellite navigation input routine
'
Satnav:
  IF LOC(4) > 0 THEN SATNAV$ = SATNAV$ + INPUT$(LOC(4),#4)
  IF INSTR(1,SATNAV$,STRING$(5,10)) > 0 THEN SATNAV$ = ""
  IF INSTR(1,SATNAV$,"HDG") > 0 THEN
    POSITION% = INSTR(1,SATNAV$,"LAT")
    IF POSITION% > 0 THEN LAT$ = MID$(SATNAV$,POSITION% + 4,12)
    POSITION% = INSTR(1,SATNAV$,"LON")
    IF POSITION% > 0 THEN LON$ = MID$(SATNAV$,POSITION% + 4,12)
    POSITION% = INSTR(1,SATNAV$,"SPEED")
    IF POSITION% > 0 THEN SPEED$ = MID$(SATNAV$,POSITION% + 6,4)
    SATNAV$ = ""
    LAT = VAL(MID$(LAT$,4,2)) + (VAL(MID$(LAT$,7,6)))/60.0
    IF LEFT$(LAT$,1)="S" THEN LAT = -LAT
    LON = VAL(MID$(LON$,3,3)) + (VAL(MID$(LON$,7,6)))/60.0
    IF MAP%=1 THEN PSET (-LON,LAT),1
    IF VAL(SPEED$) < 1.0 THEN SPD$ = " D" ELSE SPD$ = " U"
  END IF
  LOCATE 19,18
  PRINT USING "\          \  \          \  \  \";LAT$;LON$;SPEED$;
  RETURN
'
'              The following are keyboard error traps.
'                      Waiting is for <F1> key trap
'                      BlankOn is for <F2> key trap
'                      Calibrate is for <F3> key trap
'                      ErrorCom is for communications error trap
'                      GoodNight is for <F10> key trap
'
 Waiting:
  BLKPRESSED% = 0                          'If <F1> pressed, set the flag to 0
  RETURN

 BlankOn:
  BLKON% = 1                               'If <F2> pressed, set the flag to 0
  RETURN

 Calibrate:
  BLANK% = 3                               'Set mode to calibration = 3
  KEY(1) OFF                               'Turn off all keyboard traps
  KEY(3) OFF
  KEY(10) OFF
  FOR II% = 21 TO 24
    LOCATE II%,44
    PRINT "                                ";
  NEXT II%
  RETURN
```

```
Errorcom:
 RESUME

GoodNight:
 LOCATE 18,60
 PRINT "TERMINATED ";
 DEF SEG = &HB800
 BSAVE DIRC$ + "PCPLUSS.MAP",0,16384
 CLOSE
 END
```

```
'Don't ask why an error occurred, just
' go back and try again

'With <F10> pressed, terminate the
'   program
```

APPENDIX 3 - Program INSTALL.PAS

```
program install;
uses crt, dos;

const NumFiles = 6;
      files  : array[1..NumFiles] of string[13] = (' PCPLUSS.EXE',' PCPLUSS.BAS',
                                                   ' PCPLUSS.WP5',' PCPLUSS.MAP',
                                                   ' CONFIGUR.EXE',' MAP.BIN');
var  dirinfo  : searchrec;
     s,s1     : string;
     auto     : array[1..100] of string[150];
     drv,boot : string[2];
     dir      : string[50];
     f        : text;
     ch       : char;
     a,b,c    : integer;
     sets     : boolean;
     valid    : boolean;

Procedure ReBoot;
begin
  inline(
  $B8/$40/
  $00/$8E/
  $D8/$B8/
  $34/$12/
  $A3/$72/
  $00/$EA/
  $5B/$E0/
  $00/$F0);
end;

procedure CopyFile(a:integer);
var FromF,ToF           : file;
    NumRead,NumWritten : word;
    buf                 : array[1..4096] of char;
    s                   : string[13];

begin
  gotoxy(12,a + 3);
  write('->');
  s := copy(files[a],2,length(files[a]) - 1);
  assign(FromF,s1 + '\' + s);
  reset(FromF,1);
  assign(ToF,drv + dir + '\' + s);
  rewrite(ToF,1);
  repeat
    blockread(FromF,buf,sizeof(buf),NumRead);
    blockwrite(ToF,buf,NumRead,NumWritten);
  until (NumRead = 0) or (NumWritten <> NumRead);
  close(FromF);
```

```
  close(ToF);
  for NumRead := 30 to 53 do
  begin
    gotoxy(NumRead - 1,a + 3);
    write('                ');
    gotoxy(NumRead,a + 3);
    write(files[a]);
    delay(25);
  end;
  gotoxy(12,a + 3);
  write('  ');
end;

begin
  clrscr;
  sets := false;
  findfirst('INSTALL.EXE',anyfile,dirinfo);
  if doserror = 0 then
  begin
    s1 := fexpand('INSTALL.EXE');
    delete(s1,pos('INSTALL.EXE',s1) - 1,12);
    repeat
      boot := 'C:';
      gotoxy(33,9);
      write('BOOT drive: ',boot);
      gotoxy(45,9);
      ch:=readkey;
      if upcase(ch) in ['C','D','E','F'] then
      begin
        write(upcase(ch));
        boot[1] := ch;
      end;
    until upcase(ch) in ['C','D','E','F',#13];
    for b := 1 to length(boot) do boot[b] := upcase(boot[b]);

    repeat
      drv := 'C:';
      gotoxy(30,10);
      write('PCPLUSS drive: ',drv);
      gotoxy(45,10);
      ch:=readkey;
      if upcase(ch) in ['C','D','E','F'] then
      begin
        write(upcase(ch));
        drv[1] := ch;
      end;
    until upcase(ch) in ['C','D','E','F',#13];
    for b := 1 to length(drv) do drv[b] := upcase(drv[b]);

    valid := false;
    repeat
      dir := '\PCPLUSS';
```

```
  gotoxy(26,11);
  write('PCPLUSS directory: ',dir,'                          ');
  gotoxy(45,11);
  ch := readkey;
  if ch <> #13 then
  begin
    write('                                    ');
    gotoxy(45,11);
    write(ch);
    readln(s);
    dir := ch + s;
    if dir[1] <> '\' then insert('\',dir,1);
    if dir[length(dir)] = '\' then delete(dir,length(dir),1);
    c := 0;
    for b := 1 to length(dir) do
    begin
      dir[b] := upcase(dir[b]);
      if dir[b] = '\' then inc(c);
    end;
    if c = 1 then valid := true;
  end else valid := true;
until valid;

findfirst(drv + dir,directory,dirinfo);
if doserror <> 0 then mkdir(drv + dir);

clrscr;
gotoxy(34,1);
write('Copying Files');
gotoxy(((20 - length(s1)) div 2) + 10,2);
write(s1);
gotoxy(40,2);
write('To');
gotoxy(((20 - length(drv+dir)) div 2) + 50,2);
write(drv+dir);
gotoxy(10,3);
write('----------------------');
gotoxy(50,3);
write('----------------------');
for a:=1 to NumFiles do
begin
  gotoxy(14,a + 3);
  writeln(files[a]);
end;
for a:=1 to NumFiles do CopyFile(a);

gotoxy(1,10);
writeln('Current PCPLUSS input file information:');
writeln;
writeln('   BOD1:COM1:300,E,7,2');
writeln('   BOD2:COM2:110,N,8,2');
writeln('   COMM:014,000,001,005,007,005');
```

```
writeln('   FORM:\   \ \      \ ###.### ###.### ##.## ##.## #.### \  \ #\\');
writeln('    DIRC:',drv + dir);
writeln('    GAIN:2');
writeln('    ADDR:768');
writeln('    TICK:120');
writeln;
writeln('Use CONFIGUR.EXE to change PCPLUSS information');
assign(f,drv + dir + '\PCPLUSS.INF');
rewrite(f);
writeln(f,'BOD1:COM1:300,E,7,2');
writeln(f,'BOD2:COM2:110,N,8,2');
writeln(f,'COMM:014,000,001,005,007,005');
writeln(f,'FORM:\   \ \      \ ###.### ###.### ##.## ##.## #.### \  \ #\\');
writeln(f,'FORM:\   \ \      \ ###.### ###.### ##.## ##.##\\');
writeln(f,'DIRC:',drv + dir);
writeln(f,'GAIN:2');
writeln(f,'ADDR:768');
writeln(f,'TICK:120');
close(f);

gotoxy(1,20);
writeln('Editing AUTOEXEC.BAT file');
findfirst(boot + '\AUTOEXEC.BAT',anyfile,dirinfo);
if doserror = 0 then
begin
  assign(f,boot + '\AUTOEXEC.BAT');
  reset(f);
  a := 0;
  while not eof(f) do
  begin
    inc(a);
    readln(f,auto[a]);
    if pos('SET PCPLUSS',auto[a]) > 0 then sets := true;
  end;
  close(f);
  if not sets then
  begin
    findfirst(boot + '\AUTOEXEC.BAK',anyfile,dirinfo);
    if doserror = 0 then
    begin
      assign(f,boot + '\AUTOEXEC.BAK');
      erase(f);
      assign(f,boot + '\AUTOEXEC.BAT');
    end;
    rename(f,boot + '\AUTOEXEC.BAK');
    assign(f,boot + '\AUTOEXEC.BAT');
    rewrite(f);
    for b := 1 to a do
    begin
      if (pos('PATH',auto[b]) > 0) or (pos('path',auto[b]) > 0) then
        if pos('PCPLUSS',auto[b]) = 0 then auto[b] := auto[b]+';'+drv +dir;
      if (not sets) and (b = 1) then writeln(f,'SET PCPLUSS=',drv + dir);
```

```
        if (auto[b] = 'nc') or (auto[b] = 'NC') or (auto[b] = 'ncsmall') or
           (auto[b] = 'NCSMALL') then writeln(f,'PCPLUSS');
        writeln(f,auto[b]);
      end;
      if (auto[a] <> 'nc') and (auto[a] <> 'NC') and (auto[a] <> 'ncsmall') and
         (auto[a] <> 'NCSMALL') then writeln(f,'PCPLUSS');
      close(f);
    end;
  end
  else
  begin
    assign(f,boot + '\AUTOEXEC.BAT');
    rewrite(f);
    writeln(f,'PATH ',drv + dir);
    writeln(f,'SET PCPLUSS=',drv + dir);
    writeln(f,'PCPLUSS');
    close(f);
  end;

  gotoxy(20,22);
  writeln('PCPLUSS Installed Successfully!');
  gotoxy(20,23);
  textcolor(white + blink);
  writeln('Remove Diskette from drive and press Return!');
  readln;
  reboot;
end else writeln('INSTALL must be executed from the diskette!');
end.
```

APPENDIX 4 - Program CONFIGUR.PAS

```
program configur;
uses crt,dos;
var dirinfo:searchrec;
    ps      :pathstr;
    io      :text;
    s       :array[1..20] of string[80];
    streng :string[80];
    a,n,c   :integer;

begin
  clrscr;
  ps := getenv('PCPLUSS');
  if ps <> '' then
  begin
    ps := ps + '\PCPLUSS.INF';
    findfirst(ps,anyfile,dirinfo);
    if doserror = 0 then
    begin
      assign(io,ps);
      reset(io);
      n := 0;
      while not eof(io) do
      begin
        inc(n);
        readln(io,s[n]);
      end;
      close(io);
      repeat
        clrscr;
        gotoxy(1,1);
        for a := 1 to n do writeln(a:2,': ',s[a]);
        repeat
          gotoxy(1,22);
          write('Edit #:                                                    ');
          gotoxy(9,22);
          a := 0;
          readln(streng);
          if streng <> '' then val(streng,a,c);
        until ((a in [1..20]) and (a <= n)) or (streng = '');
        if a in [1..20] then
        begin
          gotoxy(3,23);
          write(copy(s[a],1,5));
          readln(streng);
          for c := 1 to length(streng) do streng[c] := upcase(streng[c]);
          if streng = '' then streng := s[a] else s[a] := copy(s[a],1,5)+streng;
        end;
      until streng = '';
      rewrite(io);
      for a:=1 to n do writeln(io,s[a]);
```

```
      close(io);
    end else writeln('Could not find: ',ps);
  end else writeln('PCPLUSS has not been installed!');
end.
```

RECENT TECHNICAL MEMORANDUMS

Copies of this and other NOAA Technical Memorandums are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22167. Paper copies vary in price. Microfiche copies cost $4.50. Recent issues of NOAA Technical Memorandums from the NMFS Southwest Fisheries Science Center are listed below:

NOAA-TM-NMFS-SWFSC- 143 Report of a marine mammal survey of the eastern tropical Pacific aboard the research vessel *McArthur* July 29-December 7, 1989.
P.S. HILL, A. JACKSON and T. GERRODETTE
(June 1990)

144 Atlas of eastern tropical Pacific oceanographic variability and cetacean sightings, 1986-1989.
P.C. FIEDLER, L.J. LIERHEIMER, S.B. REILLY, S.N. SEXTON, R.S. HOLT and D.P. DEMASTER
(July 1990)

145 Trends in landings, species composition, length-frequency distributions, and sex ratios of 11 rockfish species (Genus *Sebastes*) from central and northern California ports (1978-88).
D.E. PEARSON and S. RALSTON
(July 1990)

146 Field manual for phocid necropsies (specifically *Monachus schauinslandi*).
J.M. WINCHELL
(July 1990)

147 Survey of the abundance and distribution of pelagic young-of-the-year rockfishes, *Sebastes*, off central California.
T.W. ECHEVERRIA, W.H. LENARZ and C.A. REILLY
(September 1990)

148 United states agency for international development and national marine fisheries service workshop on tropical fish stock assessment, 5-26 July 1989, Honolulu, Hawaii.
J.J. POLOVINA and R.S. SHOMURA
(September 1990)

149 Summary of the 1988 U.S. tuna/porpoise observer data.
A.R. JACKSON
(September 1990)

150 Population monitoring of the Hawaiian Monk Seal, *Monachus schauinslandi*, and captive maintenance project at Kure Atoll, 1988.
J.R. HENDERSON and M.R. FINNEGAN
(September 1990)

151 The Hawaiian Monk Seal on Laysan Island, 1988.
T.C. JOHANOS, B.L. BECKER, M.A. BROWN, B.K. CHOY, L.M. HURUKI, R.E. BRAINARD and R.L. WESTLAKE
(September 1990)

152 A personal computer based system for analog-to-digital and serial communication data acquisition.
R.C. HOLLAND
(November 1990)