```
;{CR10X}
;TTS_4_0.csi
;Revised: 10/23/2002
;
; ---------- TURBIDITY THRESHOLD SAMPLING PROGRAM (TTS) ----------
;              Jack Lewis, Rand Eads, and Noah Campbell-Lund
;                    Redwood Sciences Laboratory
;                        USDA Forest Service
;                        1700 Bayview Drive
;                         Arcata, CA  95521


;=========================>>> NOTICE <<<============================

; U.S. Government employees produced this software on government time
; and, thus, no copyright can be claimed and this program is available
; for use by the public.

; This software is provided "as is" without warranty of any kind.
; The authors and the Redwood Sciences Laboratory do not warrant,
; guarantee or make any representations regarding the use or
; correctness, reliability, or otherwise. The user of this program
; assumes the entire risk as to the results and performance of the
; software. In no case will any party involved with creation or use
; of the software be liable for any damage that may result.

; The use of trade names is not an endorsement by the USDA Forest Service.

;************** CAMPBELL DATA LOGGER PROGRAM DISCRIPTION ************

; The program wakes up every 10 or 15 minutes, depending on the site.
; The OBS-3 sensor samples turbidity at 0.5-second intervals for 30
; seconds for a total of 60 readings.  Or, the DTS-12 collects 100
; samples in about 7 seconds. The median turbidity is saved.
; The median turbidity and average stage values are used to
; trigger a pumping sampler if the sampling criteria are satisfied.


; The conditions which govern sampling are as follows:

;   BASE FLOW:  This condition occurs when the stage is less than the
;   specified minimum stage (min_stg).  Minimum stage is the lowest
;   stage where both the turbidity probe and pumping sampler intake
;   are adequately submerged and functional.  No sampling takes place
;   in this mode.

;   RISING:  A start-up sample is collected at the first interval
;   above minimum stage, if the turbidity is above the second rising
;   threshold (the first threshold is always zero), and no rising
;   thresholds have been sampled within the past 3 hours (now hard-
;   coded, but could be a parameter).  The condition becomes rising
;   at the first interval above base flow.  For sampling to occur in
;   a rising turbidity condition, the turbidity must be equal to, or
;   greater than, the next rising threshold for 2 (stay) intervals.
;   In addition, a threshold will not be resampled if it has already
;   been sampled in the past 8 (rep_wait) intervals.
```

```
;   REVERSALS:  A turbidity reversal occurs when the turbidity has
;   changed direction for 2 (stay) intervals, and has dropped 10%
;   (revpct1) from the prior peak or risen 20% (revpct2) from the
;   prior trough, and changed least 5 NTUs (rev_val) in both cases.
;   A sample is collected if a threshold has been crossed since the
;   previous peak or trough, unless that threshold has already been
;   sampled in the past 8 (rep_wait) intervals.

;   FALLING:  Sampling occurs in a falling turbidity condition when
;   the turbidity is less than, or equal to, the next falling
;   threshold for 2 (stay) intervals.  And a threshold will not be
;   resampled within 8 (rep_wait) intervals.

;   OVERFLOW:  Turbidity sensor output exceeds the data logger's
;   millivolt limit setting (mv_limit-->OBS-3) or (ntu_limit-->DTS-12).
;   In this mode, two (lim_skip) intervals will be skipped between
;   each sampled interval.

; Recorded information is written to the datalogger memory for all the
; above conditions.

; Between each execution interval the datalogger goes into a
; quiescent state.

;******************** PROGRAM PARAMETERS **********************

; THRESHOLD CODES:
; 0:  Indicates when the average stage < the minimum stage
; 1:  Indicates rising threshold (rising turbidity)
; 2:  Indicates falling threshold (falling turbidity)
; 3:  Indicates the first interval into the program and the status
;     of whether the thresholds are rising or falling has not yet been
;     established.

; SAMPLING CODES:
; 0:  No sample collected
; 1:  Threshold sample collected
; 2:  Depth Integrated sample collected
; 3:  Auxillary sample collected (set to 1 if also a threshold sample)
; 4:  Start-up sample collected
; 5:  Fixed-time sample collected when turbidity probe is above mV limit

; FLAG USAGE:
; FLAG 1:
;   -Low:   No action, normal program execution
;   -High:  Increments dump counter and resets bottle counter to 0
; FLAG 2:
;   -Low:   No action, normal program execution
;   -High:  For a Depth-Integrated sample to be taken
; FLAG 3:
;   -Low:   No action, normal program execution
;   -High:  For an Auxillary sample to be taken
; FLAG 4:
;   -Low:   Cold start, subroutine initializes variables
;   -High:  Warm start, continue with normal execution
; FLAG 5:
;   -Low:   Water temperature sensor not connected
```

```
;   -High:  Water temperature sensor (Campbell 107) connected
; FLAG 6:
;   -Low:   Air temperature sensor not connected
;   -High:  Air temperature sensor (Campbell 107) connected
; FLAG 7:
;   -Low:   Tipping bucket rain gage not connected
;   -High:  Tipping bucket rain gage is connected
; FLAG 8:
;   -Low:   Turbidity saved as an integer value
;   -High:  Turbidity saved as a floating point value

;********************** SUBROUTINES *******************************

; 1:  Sorts turbidity values into ascending order (OBS-3 only)
; 2:  Determines the median turbidity value (OBS-3 only)
; 3:  Writes data to datalogger final storage
; 4:  First interval average stage is above minimum stage
; 5:  Determines rising/falling thresholds
; 6:  Triggers pumped sample and increments bottle counter
; 7:  Initializes variables
; 8:  Determines next threshold location
; 9:  Fixed time sample when turbidity sensor output is above maximum
; 79: Threshold sampling code
; 80: Inputs current turbidity values
; 81: Inputs current stage
; 82: Measures water/air temperature
; 83: Calculate discharge

;********************** MEMORY ADDRESSES ****************************

; Memory addresses are utilized as follows
;      1 -  60  array for sorting turbidity values
;     62 -  63  hard-coded source and destination for indirect moves (P61)
;     65 - 104  turbidity thresholds
;    106 - 166  alphabetized list of miscellaneous variables
;    168 - 176  DTS arrays, must be contiguous locations
;    178 - 200  variables of user interest grouped to
;               facilitate pasting into PC208W numeric window
; IMPORTANT: DO NOT move, insert, or delete variables in addresses
; 1 to 104 or the program will not function correctly without
; further modification.  These locations are hard-coded into the
; program (locations, not variable names, are referenced).  Variables
; whose locations are hard-coded are named in upper case as a reminder.
; If they are moved, program instructions in subroutine 7 must be
; manually modified. Always add new variables, or delete variables no
; longer needed, via the Input Location Editor above memory location 105.
; See SITE-SPECIFIC INSTRUCTIONS below for modification of thresholds.

;****************** SENSOR/DEVICE CONNECTIONS **********************

; See the Redwood Sciences Lab's web site for wiring connections at:
;   http://www.rsl.psw.fs.fed.us/projects/water/tts_webpage
; under "Implementation"

;************************ EQUIPMENT *************************

; Instruction parameters are set for the following types of equipment:
```

```
;     Data loggers:  Campbell CR510 and CR10X
;     Turbidity probe:  D & A Instruments, OBS-3, with 2.5V clamp installed
;          or the FTS DTS-12 digital probe with wiper
;     Pressure transducer:  Druck PDCR 1830 differential mV output
;     Pumping samplers:  ISCO 2100, 2700, 3700, and Sigma model 900
; The use of different devices may require modification to specific parameters.

; CR10X requires 1 relay (accepts either OBS-3 or DTS-12)
; CR510 requires 1 relay (use only with DTS-12)
; CR21X, CR10, and CR500 will not operate with this code (contact RSL
;   for additional information)

;****************** SITE-SPECIFIC INSTRUCTIONS ********************

; Check the program locations below and change the default values to
;   meet the specific site and equipment requirements

; Table 1, Execution interval seconds (600 or 900 recommended)
;
; Table 3, Sub 3, L45  Storage Area, use as site id (1 - 511),
;                      contact RSL be assigned a station number
;          Sub 7, L151 to L166  Site & equipment variables that can
;                      be modified by the user are denoted
;                      with "<==========" for easy identification
;                 L167 to L174  Logic parameters
;                 L191 and L192 set locations of threshold arrays
;                 L193 and L194 Set number of rising/falling thresholds
;                 L195 to L199 Enter rising/falling thresholds
;
; Note: if adding other sensors write the extra values at the end of the
;   TTS data records (Sub 3) to permit proper functioning of plotting &
;   analysis software

;=====================================================================


*Table 1 Program
  01: 600       Execution Interval (seconds)

;-----------------------==========-----------------------

;Read battery voltage
1:  Batt Voltage (P10)
 1: 200      Loc [ bat_volt  ]

;---------------------------------------------------------------------

;If [Flag 4 is low], then cold start
2:  If Flag/Port (P91)
 1: 24       Do if Flag 4 is Low
 2: 30       Then Do

;Call subroutine 7 to initialize variables
3:  Do (P86)
 1: 7        Call Subroutine 7

;Set Flag 4 high to continue normal program execution
```

```
4:  Do (P86)
 1: 14       Set Flag 4 High

;End of case, [If Flag 4 is low]
5:  End (P95)

;---------------------------------------------------------------------

;If [Flag 1 is high], then it is a new dump
6:  If Flag/Port (P91)
 1: 11       Do if Flag 1 is High
 2: 30       Then Do

;Increment dump counter
7:  Z=Z+1 (P32)
 1: 192      Z Loc [ dump_cnt  ]

;Reset bottle counter to 0
8:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 107      Z Loc [ bot_cnt   ]

;Reset next pump sample counter to 1
9:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 193      Z Loc [ nxt_bot   ]

;Set flag 1 low for warm start
10:  Do (P86)
 1: 21       Set Flag 1 Low

;End of case, [If Flag 1 is high]
11:  End (P95)

;---------------------------------------------------------------------

;Initialize variables at start of each wake-up
;Set sample code = 0, for no thresholds sampled
12:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Set bottle = 0
13:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 108      Z Loc [ bottle    ]

;Increment interval counter each time the table is executed
14:  Z=Z+1 (P32)
 1: 117      Z Loc [ interval  ]

;---------------------------------------------------------------------
```

```
;Call subroutine for temperature
15:  Do (P86)
 1: 82       Call Subroutine 82

;---------------------------------------------------------------------

;Call subroutine to get stage
16:  Do (P86)
 1: 81       Call Subroutine 81

;---------------------------------------------------------------------

;Call subroutine to get turbidity
17:  Do (P86)
 1: 80       Call Subroutine 80

;---------------------------------------------------------------------

;Call subroutine to calculate water discharge
18:  Do (P86)
 1: 83       Call Subroutine 83

;---------------------------------------------------------------------

;If [Flag 7 is high], then measure rainfall
19:  If Flag/Port (P91)
 1: 17       Do if Flag 7 is High
 2: 30       Then Do

;Read rain gage pulses (instruction must be in main prgm!)
20:  Pulse (P3)
 1: 1        Reps
 2: 1        Pulse Channel 1 ;           <-- P1
 3: 2        Switch Closure, All Counts
 4: 183      Loc [ rain      ]
 5: 0.01     Mult
 6: 0.0      Offset

;Add rainfall from wakeup to daily total
21:  Z=X+Y (P33)
 1: 183      X Loc [ rain      ]
 2: 184      Y Loc [ daily     ]
 3: 184      Z Loc [ daily     ]

;Add rainfall from wakeup to season total
22:  Z=X+Y (P33)
 1: 183      X Loc [ rain      ]
 2: 185      Y Loc [ season    ]
 3: 185      Z Loc [ season    ]

;If midnight
23:  If time is (P92)
 1: 0000     Minutes (Seconds --) into a
 2: 1440     Interval (same units as above)
 3: 30       Then Do

;Set daily rain total to zero when midnight
```

```
24:  Z=F (P30)
 1: 0.0      F
 2: 00       Exponent of 10
 3: 184      Z Loc [ daily     ]

;End if midnight
25:  End (P95)

;End of case [If Flag 7 is high]
26:  End (P95)

;If [Flag 2 is high], then take a depth integrated sample
27:  If Flag/Port (P91)
 1: 12       Do if Flag 2 is High
 2: 30       Then Do

;Set sample code = 2 for DI sample taken
28:  Z=F (P30)
 1: 2        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to collect pumped sample
29:  Do (P86)
 1: 6        Call Subroutine 6

;Set Flag 2 low for normal program execution
30:  Do (P86)
 1: 22       Set Flag 2 Low

;End of case, [If Flag 2 is high]
31:  End (P95)

;-----------------------==========----------------------------

;If [Flag 3 is high], then take an auxillary sample
32:  If Flag/Port (P91)
 1: 13       Do if Flag 3 is High
 2: 30       Then Do

;Set sample code = 3 for aux sample taken
33:  Z=F (P30)
 1: 3        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to collect pumped sample
34:  Do (P86)
 1: 6        Call Subroutine 6

;Set Flag 3 low for normal program execution
35:  Do (P86)
 1: 23       Set Flag 3 Low

;End of case, [If Flag 3 is high]
36:  End (P95)
```

```
;-----------------------================================----------------------
;-----------------------================================----------------------

;If [stage < minimum stage], then do
37:  If (X<=>Y) (P88)
 1: 179      X Loc [ stage     ]
 2: 4        <
 3: 189      Y Loc [ min_stg   ]
 4: 30       Then Do

;Set thr_code = 0 for baseflow condition
38:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Write a record to data logger memory and jump to end of program
39:  Do (P86)
 1: 3        Call Subroutine 3

;End of case, [If stage < min_stg]
40:  End (P95)

;-----------------------================================----------------------
;-----------------------================================----------------------

;If [interval = 1] write a record and jump to end of program
41:  If (X<=>F) (P89)
 1: 117      X Loc [ interval  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Set threshold code to 3 (initial turbidity condition unknown)
42:  Z=F (P30)
 1: 3        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Write a record to data logger memory and jump to end of program
43:  Do (P86)
 1: 3        Call Subroutine 3

;End of case, [If interval = 1]
44:  End (P95)

;-----------------------================================----------------------
;-----------------------================================----------------------

;If [thr_code = 0], then previous interval was in base flow
45:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 0        F
 4: 30       Then Do

;Call subroutine 4 for the 1st interval above minimum stage
```

```
46:  Do (P86)
 1: 4        Call Subroutine 4

;End of case, [If thr_code = 0]
47:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------

;If [interval = 2], then determine turbidity condition
48:  If (X<=>F) (P89)
 1: 117      X Loc [ interval  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;-----------------------------------------------------------------

;If [old_turb < med_turb], then turbidity is rising
49:  If (X<=>Y) (P88)
 1: 138      X Loc [ old_turb  ]
 2: 4        <
 3: 178      Y Loc [ med_turb  ]
 4: 30       Then Do

;Set thr_code = 1 for rising turbidity
50:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Set tmax = med_turb
51:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 163      Z Loc [ tmax      ]

;Else [old_turb >= med_turb], turbidity is falling or constant
52:  Else (P94)

;Set thr_code = 2 for falling turbidity
53:  Z=F (P30)
 1: 2        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Set tmin = med_turb
54:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 162      Z Loc [ tmin      ]

;End of case, [Else old_turb >= med_turb]
55:  End (P95)

;-----------------------------------------------------------------

;Call subroutine 5 to find threshold value
56:  Do (P86)
```

```
 1: 5        Call Subroutine 5

;Write a record to data logger memory and jump to end of program
57:  Do (P86)
 1: 3        Call Subroutine 3

;End of case, [If interval = 2]
58:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------

;If [thr_code = 1], then turbidity is rising
59:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;-----------------------------------------------------------------

;If [median turbidity >= threshold], then do
60:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 188      Y Loc [ thr       ]
 4: 30       Then Do

;Increment threshold counter
61:  Z=Z+1 (P32)
 1: 187      Z Loc [ thr_count ]

;-----------------------------------------------------------------
;-----------------------------------------------------------------

;If [thr_count >= stay], number of intervals turbidity is above
;the threshold value
62:  If (X<=>Y) (P88)
 1: 187      X Loc [ thr_count ]
 2: 3        >=
 3: 159      Y Loc [ stay      ]
 4: 30       Then Do

;Set rthr = thr for call to pass to subroutine 79
63:  Z=X (P31)
 1: 188      X Loc [ thr       ]
 2: 152      Z Loc [ rthr      ]

;Call subroutine 79 for possible threshold sample
64:  Do (P86)
 1: 79       Call Subroutine 79


;Call subroutine 5 to determine next threshold
65:  Do (P86)
 1: 5        Call Subroutine 5
```

```
;-------------------------------------------------------------------

;End of case, [If thr_count >= stay]
66:  End (P95)

;-------------------------------------------------------------------

;Else [med_turb < thr]
67:  Else (P94)

;-------------------------------------------------------------------

;If[turb_dev = 1], then DO (OBS-3)
68:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do
;-------------------------------------------------------------------

;If [med_mv >= mv_limit], then DO
69:  If (X<=>Y) (P88)
 1: 132      X Loc [ med_mv    ]
 2: 3        >=
 3: 133      Y Loc [ mv_limit  ]
 4: 30       Then Do

;Call turbidity overflow subroutine 9
70:  Do (P86)
 1: 9        Call Subroutine 9

;End of case [if med_mv >= mv_limit]
71:  End (P95)

;-------------------------------------------------------------------

;End of case [if turb_dev = 1]
72:  End (P95)

;-------------------------------------------------------------------

;If[turb_dev = 2], then DO (DTS-12)
73:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;-------------------------------------------------------------------
;If [med_turb >= ntu_limit], then DO
74:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 137      Y Loc [ ntu_limit ]
 4: 30       Then Do

;Call turbidity overflow subroutine 9
```

```
75:  Do (P86)
 1: 9        Call Subroutine 9

;End of case [if med_turb >= ntu_limit]
76:  End (P95)

;-----------------------------------------------------------------

;End of case [if turb_dev = 2]
77:  End (P95)

;-----------------------------------------------------------------

;End of case, [else med_turb < thr]
78:  End (P95)

;-------------------------------------------------------------------

;If [med_turb >= tmax], then do
79:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 163      Y Loc [ tmax      ]
 4: 30       Then Do

;Set tmax = med_turb
80:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 163      Z Loc [ tmax      ]

;Set reversal counter = 0
81:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;End of case, [If med_turb >= tmax]
82:  End (P95)

;----------------------=====================-----------------------

;If [med_turb < old_turb], then do
83:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 4        <
 3: 138      Y Loc [ old_turb  ]
 4: 30       Then Do

;If [smp_code <> 5], then do
84:  If (X<=>F) (P89)
 1: 191      X Loc [ smp_code  ]
 2: 2        <>
 3: 5        F
 4: 30       Then Do

;Increment reversal counter
85:  Z=Z+1 (P32)
```

```
 1: 186      Z Loc [ rev_count ]

;End of case [smp_code <> 5]
86:  End (P95)

;Set rev_mult = tmax * revpct1
87:  Z=X*Y (P36)
 1: 163      X Loc [ tmax      ]
 2: 148      Y Loc [ revpct1   ]
 3: 147      Z Loc [ rev_mult  ]

;If [rev_val < rev_mult]
88:  If (X<=>Y) (P88)
```

```
;Call subroutine 5 to determine next threshold
96:  Do (P86)
 1: 5        Call Subroutine 5

;Set tmin = med_turb
97:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 162      Z Loc [ tmin      ]

;Set scratch = nextindex - 1
98:  Z=X+F (P34)
 1: 135      X Loc [ nextindex ]
 2: -1       F
 3: 153      Z Loc [ scratch   ]

;Store old threshold at nextindex - 1 in DEST
99:  Indirect Move (P61)
 1: 153      Source Loc [ scratch   ]
 2: 110      Destination Loc [ dest_loc  ]

;Assign DEST to fthr
100:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 114      Z Loc [ fthr      ]

;If [fthr < tmax]
101:  If (X<=>Y) (P88)
 1: 114      X Loc [ fthr      ]
 2: 4        <
 3: 163      Y Loc [ tmax      ]
 4: 30       Then Do

;Call subroutine 79 for possible threshold sample
102:  Do (P86)
 1: 79       Call Subroutine 79

;End of case, [If fthr < tmax]
103:  End (P95)

;---------------------------------------------------------------------

;Set reversal counter = 0
104:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;End of case, [If rev_count >= stay]
105:  End (P95)

;---------------------------------------------------------------------

;End of case, [If rev_thr >= med_turb]
106:  End (P95)

;---------------------------------------------------------------------
```

```
;End of case, [If med_turb < old_turb]
107:  End (P95)

;Write a record to data logger memory and jump to end of program
108:  Do (P86)
 1: 3        Call Subroutine 3

;-----------------------------------------------------------------

;End of case, [if thr_code = 1], turbidity rising
109:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------

;If [thr_code = 2], then turbidity is falling
110:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;-----------------------------------------------------------------

;If [thr >= med_turb]
111:  If (X<=>Y) (P88)
 1: 188      X Loc [ thr       ]
 2: 3        >=
 3: 178      Y Loc [ med_turb  ]
 4: 30       Then Do

;Increment threshold counter
112:  Z=Z+1 (P32)
 1: 187      Z Loc [ thr_count ]

;-----------------------------------------------------------------

;If [thr_count = stay], then do
113:  If (X<=>Y) (P88)
 1: 187      X Loc [ thr_count ]
 2: 3        >=
 3: 159      Y Loc [ stay      ]
 4: 30       Then Do

;-----------------------------------------------------------------

;If [inbounds = 1], then still inside threshold array
114:  If (X<=>F) (P89)
 1: 115      X Loc [ inbounds  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Set fthr = thr to pass to subroutine 79
115:  Z=X (P31)
 1: 188      X Loc [ thr       ]
```

```
 2: 114      Z Loc [ fthr      ]

;Call subroutine 79 for possible threshold sample
116:  Do (P86)
 1: 79       Call Subroutine 79

;Call subroutine 5 to determine next threshold
117:  Do (P86)
 1: 5        Call Subroutine 5


;--------------------------------------------------------------------

;End of case, [If inbounds = 1]
118:  End (P95)

;--------------------------------------------------------------------

;End of case, [If thr_count = stay]
119:  End (P95)

;--------------------------------------------------------------------

;End of case, [If thr >= med_turb]
120:  End (P95)

;----------------------=====================-----------------------
;If[turb_dev = 1], then DO (OBS-3)
121:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do
;--------------------------------------------------------------------

;If [med_mv >= mv_limit], then DO
122:  If (X<=>Y) (P88)
 1: 132      X Loc [ med_mv    ]
 2: 3        >=
 3: 133      Y Loc [ mv_limit  ]
 4: 30       Then Do

;Call turbidity overflow subroutine 9
123:  Do (P86)
 1: 9        Call Subroutine 9

;End of case [if med_mv >= mv_limit]
124:  End (P95)

;--------------------------------------------------------------------

;End of case [if turb_dev = 1]
125:  End (P95)

;--------------------------------------------------------------------

;If[turb_dev = 2], then DO (DTS-12)
```

```
126:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;--------------------------------------------------------------------
;If [med_turb >= ntu_limit], then DO
127:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 137      Y Loc [ ntu_limit ]
 4: 30       Then Do

;Call turbidity overflow subroutine 9
128:  Do (P86)
 1: 9        Call Subroutine 9

;End of case [if med_turb >= ntu_limit]
129:  End (P95)

;--------------------------------------------------------------

;End of case [if turb_dev = 2]
130:  End (P95)

;--------------------------------------------------------------------

;If [tmin >= med_turb]
131:  If (X<=>Y) (P88)
 1: 162      X Loc [ tmin      ]
 2: 3        >=
 3: 178      Y Loc [ med_turb  ]
 4: 30       Then Do

;Set tmin = med_turb
132:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 162      Z Loc [ tmin      ]

;Set reversal counter = 0
133:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;End of case, [If tmin >= med_turb]
134:  End (P95)

;--------------------------------------------------------------------

;If [old_turb < med_turb]
135:  If (X<=>Y) (P88)
 1: 138      X Loc [ old_turb  ]
 2: 4        <
 3: 178      Y Loc [ med_turb  ]
 4: 30       Then Do
```

```
;Increment reversal counter
136:  Z=Z+1 (P32)
 1: 186      Z Loc [ rev_count ]

;Set rev_mult = tmin * revpct2
137:  Z=X*Y (P36)
 1: 162      X Loc [ tmin      ]
 2: 149      Y Loc [ revpct2   ]
 3: 147      Z Loc [ rev_mult  ]

;---------------------------------------------------------------------

;If [rev_val < mult]
138:  If (X<=>Y) (P88)
 1: 151      X Loc [ rev_val   ]
 2: 4        <
 3: 147      Y Loc [ rev_mult  ]
 4: 30       Then Do

;Set rev_thr = tmin + rev_mult
139:  Z=X+Y (P33)
 1: 162      X Loc [ tmin      ]
 2: 147      Y Loc [ rev_mult  ]
 3: 150      Z Loc [ rev_thr   ]

;Else [rev_val >= rev_mult]
140:  Else (P94)

;Set rev_thr = tmin + rev_val
141:  Z=X+Y (P33)
 1: 162      X Loc [ tmin      ]
 2: 151      Y Loc [ rev_val   ]
 3: 150      Z Loc [ rev_thr   ]

;End of case, [else rev_val >= rev_mult]
142:  End (P95)

;---------------------------------------------------------------------

;If [med_turb >= rev_thr], then do
143:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 150      Y Loc [ rev_thr   ]
 4: 30       Then Do

;---------------------------------------------------------------------

;If [rev_count >= stay], then do
144:  If (X<=>Y) (P88)
 1: 186      X Loc [ rev_count ]
 2: 3        >=
 3: 159      Y Loc [ stay      ]
 4: 30       Then Do

;Set thr_code = 1 for rising turbidity
```

```
145:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Call subroutine 5 to determine next threshold
146:  Do (P86)
 1: 5        Call Subroutine 5

;Set tmax = med_turb
147:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 163      Z Loc [ tmax      ]

;Set scratch = nextindex - 1
148:  Z=X+F (P34)
 1: 135      X Loc [ nextindex ]
 2: -1       F
 3: 153      Z Loc [ scratch   ]

;Move the threshold at nextindex - 1 into DEST
149:  Indirect Move (P61)
 1: 153      Source Loc [ scratch   ]
 2: 110      Destination Loc [ dest_loc  ]

;Copy DEST (last threshold) to rthr
150:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 152      Z Loc [ rthr      ]

;---------------------------------------------------------------------

;If [tmin < rthr], then do
151:  If (X<=>Y) (P88)
 1: 162      X Loc [ tmin      ]
 2: 4        <
 3: 152      Y Loc [ rthr      ]
 4: 30       Then Do

;Call subroutine 79 for possible threshold sample
152:  Do (P86)
 1: 79       Call Subroutine 79

;End of case, [If tmin < rthr]
153:  End (P95)

;---------------------------------------------------------------------

;Set reversal counter = 0
154:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;End of case, [If rev_count >= stay]
155:  End (P95)
```

```
;---------------------------------------------------------------------

;End of case, [If med_turb >= rev_thr]
156:  End (P95)

;---------------------------------------------------------------------

;End of case, [If old_turb < med_turb]
157:  End (P95)

;Write a record to data logger memory and jump to end of program
158:  Do (P86)
 1: 3        Call Subroutine 3

;---------------------------------------------------------------------

;End of case, [If thr_code = 2], turbidity falling
159:  End (P95)

;-----------------------=====================-----------------------
;-----------------------=====================-----------------------
;-----------------------=====================-----------------------



*Table 2 Program
  02: 0         Execution Interval (seconds)

; Not Used



*Table 3 Subroutines

;-----------------------=====================-----------------------
;-----------------------=====================-----------------------
;-----------------------=====================-----------------------


;SUBROUTINE 1 sorts turbidity mV values into ascending order (OBS-3)
1:  Beginning of Subroutine (P85)
 1: 1        Subroutine 1

;Increment counter k
2:  Z=Z+1 (P32)
 1: 120      Z Loc [ k         ]

;Set counter j = k
3:  Z=X (P31)
 1: 120      X Loc [ k         ]
 2: 118      Z Loc [ j         ]

;Set insert flag = 0
4:  Z=F (P30)
 1: 0.0      F
 2: 00       Exponent of 10
 3: 116      Z Loc [ insert    ]
```

```
;-------------------------================================-----------------------

;Loop for inserting new values
5:  Beginning of Loop (P87)
 1: 0        Delay
 2: 0        Loop Count

;If [j = 1], then do
6:  If (X<=>F) (P89)
 1: 118      X Loc [ j         ]
 2: 1        =
 3: 1        F
 4: 31       Exit Loop if True

;If [insert flag = 1], then do
7:  If (X<=>F) (P89)
 1: 116      X Loc [ insert    ]
 2: 1        =
 3: 1        F
 4: 31       Exit Loop if True

;-------------------------------------------------------------------------

;Set jminus1 = j - 1
8:  Z=X+F (P34)
 1: 118      X Loc [ j         ]
 2: -1       F
 3: 119      Z Loc [ jminus1   ]

;Assign mv(j-1) to DEST
9:  Indirect Move (P61)
 1: 119      Source Loc [ jminus1   ]
 2: 110      Destination Loc [ dest_loc  ]

;Assign DEST to last_mv
10:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 123      Z Loc [ last_mv   ]

;-------------------------------------------------------------------------

;If [raw_mv >= last_mv], then do
11:  If (X<=>Y) (P88)
 1: 142      X Loc [ raw_mv    ]
 2: 3        >=
 3: 123      Y Loc [ last_mv   ]
 4: 30       Then Do

;Set insert flag = 1
12:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 116      Z Loc [ insert    ]

;Else [raw_mv < last_mv]
13:  Else (P94)
```

```
;Set mv(j) = mv(j-1)
14:  Indirect Move (P61)
 1: 119      Source Loc [ jminus1   ]
 2: 118      Destination Loc [ j         ]

;Set j = j - 1
15:  Z=X+F (P34)
 1: 118      X Loc [ j         ]
 2: -1       F
 3: 118      Z Loc [ j         ]

;End of case, [else raw_mv < last_mv]
16:  End (P95)

;---------------------------------------------------------------------

;End of loop to insert turbidity values
17:  End (P95)

;----------------------=====================-----------------------------

;Copy turbidity (raw_mv) to temporary storage (SOURCE)
18:  Z=X (P31)
 1: 142      X Loc [ raw_mv    ]
 2: 62       Z Loc [ SOURCE    ]

;Assign SOURCE (raw_mv) to mv(j)
19:  Indirect Move (P61)
 1: 156      Source Loc [ sourc_loc ]
 2: 118      Destination Loc [ j         ]

;End of subroutine 1
20:  End (P95)

;----------------------=====================-----------------------------
;----------------------=====================-----------------------------
;----------------------=====================-----------------------------


;SUBROUTINE 2 find median turbidity (millivolts) -- OBS-3
21:  Beginning of Subroutine (P85)
 1: 2        Subroutine 2

;Take the mod 2 of k to see if there is an even or odd number of
;turbidity values that were read
22:  Z=X MOD F (P46)
 1: 120      X Loc [ k         ]
 2: 2        F
 3: 144      Z Loc [ rem       ]

;---------------------------------------------------------------------

;If [remainder <> 0], then k is odd
23:  If (X<=>F) (P89)
 1: 144      X Loc [ rem       ]
 2: 2        <>
```

```
 3: 0        F
 4: 30       Then Do

;Set k = k + 1
24:  Z=Z+1 (P32)
 1: 120      Z Loc [ k         ]

;Divide k in half to find the location which holds the median value
25:  Z=X*F (P37)
 1: 120      X Loc [ k         ]
 2: .5       F
 3: 154      Z Loc [ source1   ]

;Add .5 to source1 to round correctly (this takes care of possible
;floating point error)
26:  Z=X+F (P34)
 1: 154      X Loc [ source1   ]
 2: .5       F
 3: 154      Z Loc [ source1   ]

;Take integer of source1
27:  Z=INT(X) (P45)
 1: 154      X Loc [ source1   ]
 2: 154      Z Loc [ source1   ]

;Move median at source1 to temporary location (DEST)
28:  Indirect Move (P61)
 1: 154      Source Loc [ source1   ]
 2: 110      Destination Loc [ dest_loc  ]

;Set med_mv = DEST (median)
29:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 132      Z Loc [ med_mv    ]

;---------------------------------------------------------------------

;Else [remainder = 0]
30:  Else (P94)

;Divide k by 2 to find the middle value
31:  Z=X*F (P37)
 1: 120      X Loc [ k         ]
 2: .5       F
 3: 154      Z Loc [ source1   ]

;Add .5 to source1 to round correctly (this takes care of possible
;floating point error)
32:  Z=X+F (P34)
 1: 154      X Loc [ source1   ]
 2: .5       F
 3: 154      Z Loc [ source1   ]

;Take the integer of source1
33:  Z=INT(X) (P45)
 1: 154      X Loc [ source1   ]
 2: 154      Z Loc [ source1   ]
```

```
;Set source2 = source1 + 1
34:  Z=X+F (P34)
 1: 154      X Loc [ source1   ]
 2: 1        F
 3: 155      Z Loc [ source2   ]

;Move first turbidity value into temporary storage (DEST)
35:  Indirect Move (P61)
 1: 154      Source Loc [ source1   ]
 2: 110      Destination Loc [ dest_loc  ]

;Copy DEST (first value) to med1
36:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 130      Z Loc [ med1      ]

;Move second turbidity value into temporary storage (DEST)
37:  Indirect Move (P61)
 1: 155      Source Loc [ source2   ]
 2: 110      Destination Loc [ dest_loc  ]

;Copy DEST (second value) to med2
38:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 131      Z Loc [ med2      ]

;Add the two turbidity values together
39:  Z=X+Y (P33)
 1: 130      X Loc [ med1      ]
 2: 131      Y Loc [ med2      ]
 3: 160      Z Loc [ sum       ]

;Divide this sum by 2 to get the median value
40:  Z=X*F (P37)
 1: 160      X Loc [ sum       ]
 2: .5       F
 3: 132      Z Loc [ med_mv    ]

;End of case, [else remainder = 0]
41:  End (P95)

;End of subroutine 2
42:  End (P95)

;-----------------------==========================--------------------
;-----------------------==========================--------------------
;-----------------------==========================--------------------


;SUBROUTINE 3 write data to memory and jump to end of program
43:  Beginning of Subroutine (P85)
 1: 3        Subroutine 3

;Set output flag high to write data
44:  Do (P86)
 1: 10       Set Output Flag High (Flag 0)
```

```
;Site specific identifier (1 - 511)
45:  Set Active Storage Area (P80)
 1: 01       Final Storage Area 1
 2: 511      Array ID

;Writes out time from datalogger clock
46:  Real Time (P77)
 1: 1120     (Same as 1220) Y,D,Hr/Mn

;Writes out dump number
47:  Sample (P70)
 1: 1        Reps
 2: 192      Loc [ dump_cnt  ]

;Writes out the number of sample bottles collected
48:  Sample (P70)
 1: 1        Reps
 2: 108      Loc [ bottle    ]

;Writes out the threshold code
49:  Sample (P70)
 1: 1        Reps
 2: 190      Loc [ thr_code  ]

;Writes out the sample code
50:  Sample (P70)
 1: 1        Reps
 2: 191      Loc [ smp_code  ]

;Writes out the average stage
51:  Sample (P70)
 1: 1        Reps ;
 2: 179      Loc [ stage     ]

;Writes out the median turbidity
52:  Sample (P70)
 1: 1        Reps
 2: 178      Loc [ med_turb  ]

;If [turb_dev = 3], then write DTS-12 water temp
53:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 3        F
 4: 30       Then Do

;Writes out DTS-12 water temperature
54:  Sample (P70)
 1: 1        Reps
 2: 174      Loc [ dts_temp  ]

;End case If[turb_dev = 3]
55:  End (P95)

;If [Flag 5 is high], then write water temp
56:  If Flag/Port (P91)
```

```
 1: 15       Do if Flag 5 is High
 2: 30       Then Do

;Writes out water temperature (sensor T107)
57:  Sample (P70)
 1: 1        Reps
 2: 182      Loc [ wtemp     ]

;End of case [Flag 5 is high]
58:  End (P95)

;If [Flag 6 is high], then write air temp
59:  If Flag/Port (P91)
 1: 16       Do if Flag 6 is High
 2: 30       Then Do

;Writes out air temperature (sensor T107)
60:  Sample (P70)
 1: 1        Reps
 2: 181      Loc [ atemp     ]

;End of case [Flag 6 is high]
61:  End (P95)

;If [Flag 7 is high], then write rain
62:  If Flag/Port (P91)
 1: 17       Do if Flag 7 is High
 2: 30       Then Do

;Totals up the rainfall (inches)
63:  Totalize (P72)
 1: 1        Reps
 2: 183      Loc [ rain      ]

;End of case [Flag 7 is high]
64:  End (P95)

;Send data to Storage Module, if connected
65:  Serial Out (P96)
 1: 71       SM192/SM716/CSM1

;Set old_turb = med_turb for next wakeup
66:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 138      Z Loc [ old_turb  ]

;Goes to the end of program table 1
67:  Do (P86)
 1: 0        Go to end of Program Table

;End of subroutine 3
68:  End (P95)

;-----------------------==========-----------------------
;-----------------------==========-----------------------
;-------------------------------------------------------
```

```
;SUBROUTINE 4 first interval above minimum stage
69:  Beginning of Subroutine (P85)
 1: 4        Subroutine 4

;Set timer = interval - last_rsam
70:  Z=X-Y (P35)
 1: 117      X Loc [ interval  ]
 2: 124      Y Loc [ last_rsam ]
 3: 161      Z Loc [ timer     ]

;Set scratch = start1 + 1, location of first nonzero rising threshold
71:  Z=X+F (P34)
 1: 157      X Loc [ start1    ]
 2: 1        F
 3: 153      Z Loc [ scratch   ]

;Place the threshold at start1 + 1 into temporary storage (DEST)
72:  Indirect Move (P61)
 1: 153      Source Loc [ scratch   ]
 2: 110      Destination Loc [ dest_loc  ]

;Copy DEST (first rising threshold) to frst_rthr
73:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 113      Z Loc [ frst_rthr ]
```

```
 1: 4        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to collect pumped sample
78:  Do (P86)
 1: 6        Call Subroutine 6

;End of case, [If smp_code = 0]
79:  End (P95)

;---------------------------------------------------------------------

;Set last_rthr = 0
80:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 125      Z Loc [ last_rthr ]

;Set last_rsam = interval
81:  Z=X (P31)
 1: 117      X Loc [ interval  ]
 2: 124      Z Loc [ last_rsam ]

;End of case, [med_turb >= frst_rthr]
82:  End (P95)

;---------------------------------------------------------------------

;End of case, [timer >= 18]
83:  End (P95)

;------------------------=====================-----------------------

;If [last_rsam = 0], then do
84:  If (X<=>F) (P89)
 1: 124      X Loc [ last_rsam ]
 2: 1        =
 3: 0        F
 4: 30       Then Do

;---------------------------------------------------------------------

;If [med_turb >= frst_rthr], then turbidity is above the first rising
;threshold
85:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 3        >=
 3: 113      Y Loc [ frst_rthr ]
 4: 30       Then Do

;---------------------------------------------------------------------

;If [smp_code = 0], then no threshold samples have been taken
86:  If (X<=>F) (P89)
 1: 191      X Loc [ smp_code  ]
 2: 1        =
```

```
 3: 0        F
 4: 30       Then Do

;Set smp_code for threshold sample collected
87:  Z=F (P30)
 1: 4        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to collect pumped sample
88:  Do (P86)
 1: 6        Call Subroutine 6

;End of case, [If smp_code = 0]
89:  End (P95)

;---------------------------------------------------------------------

;Set last_rthr = 0
90:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 125      Z Loc [ last_rthr ]

;Set last_rsam = interval
91:  Z=X (P31)
 1: 117      X Loc [ interval  ]
 2: 124      Z Loc [ last_rsam ]

;End of case, [If med_turb >= frst_rthr]
92:  End (P95)

;---------------------------------------------------------------------

;End of case, [If last_rsam = 0]
93:  End (P95)
;---------------------------------------------------------------------

;Set tmax = med_turb
94:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 163      Z Loc [ tmax      ]

;Set thr_code = 1 for rising threshold
95:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Call subroutine 5 to determine next threshold
96:  Do (P86)
 1: 5        Call Subroutine 5

;Write a record to data logger memory and jump to end of program
97:  Do (P86)
 1: 3        Call Subroutine 3
```

```
;End of subroutine 4
98:  End (P95)

;-------------------------------------------------------------------
;-------------------------------------------------------------------
;-------------------------------------------------------------------


;SUBROUTINE 5 determine rising/falling thresholds
99:  Beginning of Subroutine (P85)
 1: 5        Subroutine 5

;Set inbounds = 1
100:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 115      Z Loc [ inbounds  ]

;-------------------------------------------------------------------

;If [thr_code = 1], then use rising thresholds
101:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Set nextindex = start1, start of rising threshold array
102:  Z=X (P31)
 1: 157      X Loc [ start1    ]
 2: 135      Z Loc [ nextindex ]

;Call subroutine 8 to determine the next threshold location
103:  Do (P86)
 1: 8        Call Subroutine 8

;-------------------------------------------------------------------
;Loop through the threshold locations until conditions are met
104:  Beginning of Loop (P87)
 1: 0000     Delay
 2: 0000     Loop Count

;If [nextindex >= maxrindex], exit loop
105:  If (X<=>Y) (P88)
 1: 135      X Loc [ nextindex ]
 2: 3        >=
 3: 129      Y Loc [ maxrindex ]
 4: 31       Exit Loop if True

;If [med_turb < next_thr], exit loop
106:  If (X<=>Y) (P88)
 1: 178      X Loc [ med_turb  ]
 2: 4        <
 3: 136      Y Loc [ next_thr  ]
 4: 31       Exit Loop if True

;Increment nextindex
```

```
107:  Z=Z+1 (P32)
 1: 135      Z Loc [ nextindex ]

;Call subroutine 8 to determine the next threshold location
108:  Do (P86)
 1: 8        Call Subroutine 8

;End of loop
109:  End (P95)
;-----------------------------------------------------------------------

;Set thr = next_thr
110:  Z=X (P31)
 1: 136      X Loc [ next_thr  ]
 2: 188      Z Loc [ thr       ]

;End of case, [If thr_code = 1], turbidity rising
111:  End (P95)

;-----------------------------------------------------------------------

;If [thr_code = 2], then use falling thresholds
112:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;Set nextindex = start2
113:  Z=X (P31)
 1: 158      X Loc [ start2    ]
 2: 135      Z Loc [ nextindex ]

;Call subroutine 8 to determine the next threshold location
114:  Do (P86)
 1: 8        Call Subroutine 8

;-----------------------------------------------------------------------

;Loop through the threshold locations until conditions are met
115:  Beginning of Loop (P87)
 1: 0000     Delay
 2: 0000     Loop Count

;If [nextindex >= maxfindex], exit loop
116:  If (X<=>Y) (P88)
 1: 135      X Loc [ nextindex ]
 2: 3        >=
 3: 128      Y Loc [ maxfindex ]
 4: 31       Exit Loop if True

;If [med_turb > next_thr], exit loop
117:  If (X<=>Y) (P88)
 1: 136      X Loc [ next_thr  ]
 2: 4        <
 3: 178      Y Loc [ med_turb  ]
 4: 31       Exit Loop if True
```

```
;Increment nextindex
118:  Z=Z+1 (P32)
 1: 135      Z Loc [ nextindex ]

;Call subroutine 8 to determine the next threshold location
119:  Do (P86)
 1: 8        Call Subroutine 8

;End of loop
120:  End (P95)

;Set thr = next_thr
121:  Z=X (P31)
 1: 136      X Loc [ next_thr  ]
 2: 188      Z Loc [ thr       ]

;---------------------------------------------------------------------

;If [nextindex = maxfindex], then do
122:  If (X<=>Y) (P88)
 1: 135      X Loc [ nextindex ]
 2: 1        =
 3: 128      Y Loc [ maxfindex ]
 4: 30       Then Do

;Set inbounds = 0
123:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 115      Z Loc [ inbounds  ]

;End of case, [If nextindex = maxfindex]
124:  End (P95)

;---------------------------------------------------------------------

;End of case, [If thr_code = 2]
125:  End (P95)

;----------------------=====================----------------------

;Set threshold counter = 0
126:  Z=F (P30)
 1: 0.0      F
 2: 00       Exponent of 10
 3: 187      Z Loc [ thr_count ]

;End of subroutine 5
127:  End (P95)

;----------------------=====================----------------------
;----------------------=====================----------------------
;----------------------=====================----------------------


;SUBROUTINE 6 collect pumped sample
```

```
128:  Beginning of Subroutine (P85)
 1: 6        Subroutine 6

;If [dts_temp>= 0.5 deg. C] then sample (water above freezing)
;Default if no sensor is 999
129:  If (X<=>F) (P89)
 1: 174      X Loc [ dts_temp  ]
 2: 3        >=
 3: 0.5      F
 4: 30       Then Do

;If [wtemp>= 0.5 deg. C] then sample (water above freezing)
;Default if no sensor is 999
130:  If (X<=>F) (P89)
 1: 182      X Loc [ wtemp     ]
 2: 3        >=
 3: 0.5      F
 4: 30       Then Do

;If [atemp >= 0.5 deg. C] then sample (air above freezing)
;Default if no sensor is 999
131:  If (X<=>F) (P89)
 1: 181      X Loc [ atemp     ]
 2: 3        >=
 3: 0.5      F
 4: 30       Then Do

;Increment bottle counter
132:  Z=Z+1 (P32)
 1: 107      Z Loc [ bot_cnt   ]

;Increment next pumped sample counter
133:  Z=Z+1 (P32)
 1: 193      Z Loc [ nxt_bot   ]

;Set bottle = bot_cnt (bottle numbers are writen out
;only when they are collected, otherwise they are recorded as 0)
134:  Z=X (P31)
 1: 107      X Loc [ bot_cnt   ]
 2: 108      Z Loc [ bottle    ]

;If [bottle < 25], then do
;Empty sample bottles are available
135:  If (X<=>F) (P89)
 1: 108      X Loc [ bottle    ]
 2: 4        <
 3: 25       F
 4: 30       Then Do

;If [turb_dev <> 1] DTS-12 in use (CR510 or CR10X)
136:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 2        <>
 3: 1        F
 4: 30       Then Do

;Pulse Port 1 to trigger pumped sampler
```

```
137:  Pulse Port w/Duration (P21)
 1: 1        Port ;                         <-- using C1
 2: 109      Pulse Length Loc [ delay     ]

;Else [turb_dev = 1] OBS-3 in use (CR510 or CR10X)
138:  Else (P94)

;If [logger = 0], CR510 in use
139:  If (X<=>F) (P89)
 1: 127      X Loc [ logger    ]
 2: 1        =
 3: 0        F
 4: 30       Then Do

;Pulse E2 to trigger samper; control ports not used
140:  Excite-Delay (SE) (P4)
 1: 1        Reps
 2: 5        2500 mV Slow Range
 3: 5        SE Channel ;                  <-- no connection
 4: 2        Excite all reps w/Exchan 2 ; <-- using E2
 5: 500      Delay (units 0.01 sec)
 6: 2500     mV Excitation
 7: 112      Loc [ dummy     ]
 8: 1.0      Mult
 9: 0.0      Offset

;Else [logger = 1] CR10X in use
141:  Else (P94)

;Pulse Port 3 to trigger pumped sampler
142:  Pulse Port w/Duration (P21)
 1: 3        Port ;                         <-- using C3
 2: 109      Pulse Length Loc [ delay     ]

;End case [else logger = 1]
143:  End (P95)

;End case [else turb_dev = 1]
144:  End (P95)

;End of case, [If bottle < 25]
145:  End (P95)

;End case [if dts_temp >= 0.5]
146:  End (P95)

;End case [if wtemp >= 0.5]
147:  End (P95)

;End case [if atemp >= 0.5]
148:  End (P95)

;End of subroutine 6
149:  End (P95)

;---------------------------------------------------------------
;---------------------------------------------------------------
```

```
;-----------------------------------------------------------------


;SUBROUTINE 7 initialize variables
150:  Beginning of Subroutine (P85)
 1: 7        Subroutine 7


;---------------------------------------------------------------------
;Site and equipment variables - modify values indicated by <========

151:  Z=F (P30)
 1: 0.0      F ;                  <========      Enter minimum stage
 2: 00       Exponent of 10
 3: 189      Z Loc [ min_stg   ]

152:  Z=F (P30)
 1: 00.800   F ;                  <========      Enter multiplier for
 2: 00       Exponent of 10 ;                    the turbidity probe
 3: 194      Z Loc [ turb_mult ] ;               (OBS-3 only)

153:  Z=F (P30)
 1: 0        F ;                  <========      Enter offset for the
 2: 00       Exponent of 10 ;                    turbidity probe
 3: 195      Z Loc [ turb_off  ] ;

154:  Z=F (P30)
 1: 1        F ;                  <========      Enter multiplier for
 2: 00       Exponent of 10 ;                    the pressure trans.
 3: 196      Z Loc [ stg_mult  ]

155:  Z=F (P30)
 1: 0        F ;                  <========      Enter offset for the
 2: 00       Exponent of 10 ;                    pressure transducer
 3: 197      Z Loc [ stg_off   ]

156:  Do (P86) ;                                 Cold Start (low)
 1: 24       Set Flag 4 Low ;     <========      Warm Start (high)

157:  Do (P86) ;                                 Water temp (T107 sensor)
 1: 25       Set Flag 5 Low ;     <========      low --> not active

158:  Do (P86) ;                                 Air temp (T107 sensor)
 1: 26       Set Flag 6 Low ;     <========      low --> not active

159:  Do (P86) ;                                 Rain gage
 1: 27       Set Flag 7 Low ;     <========      low --> not active

160:  Do (P86) ;                                 Turbibity resolution
 1: 28       Set Flag 8 Low ;     <========      low --> integer

161:  Z=F (P30) ;                                Data logger type
 1: 1        F ;                  <========      Set to 0 for CR510
 2: 00       Exponent of 10 ;                    or set to 1 for CR10X
 3: 127      Z Loc [ logger    ] ;

162:  Z=F (P30) ;                                Calculate discharge
```

```
 1: 0        F ;                  <========      0 --> not active
 2: 00       Exponent of 10 ;                    1 --> active
 3: 140      Z Loc [ qcalc     ] ;               Set next 2 instructions

163:  Z=F (P30) ;                                Enter multiplier for
 1: 1        F ;                  <========      discharge calc.
 2: 00       Exponent of 10 ;                    (F * stg**exp)
 3: 198      Z Loc [ q_mult    ]

164:  Z=F (P30) ;                                Enter exponent for
 1: 1        F ;                  <========      discharge calc.
 2: 00       Exponent of 10 ;                    (raise stg to power
 3: 199      Z Loc [ q_exp     ] ;               held in F)

165:  Z=F (P30)
 1: 1        F ;                  <=========     Select turbidity sensor
 2: 00       Exponent of 10 ;                    1 = OBS-3
 3: 165      Z Loc [ turb_dev  ] ;               2 = DTS-12
;                                                3 = DTS-12, record water temp.

;---------------------------------------------------------------------
;Logic parameters - modify values indicated by <========

166:  Z=F (P30)
 1: 100      F ;                  <========      Enter delay (0.01s)
 2: 00       Exponent of 10 ;                    that selected port is
 3: 109      Z Loc [ delay     ] ;               high to trigger a pumped
;                                                sample

167:  Z=F (P30) ;                                Enter the minimum
 1: 5        F ;                  <========      absolute change in NTU
 2: 00       Exponent of 10 ;                    for program to detect
 3: 151      Z Loc [ rev_val   ] ;               a turbidity reversal

168:  Z=F (P30)
 1: 0.1      F ;                  <========      Enter value for the
 2: 00       Exponent of 10 ;                    reversal % from rising
 3: 148      Z Loc [ revpct1   ] ;               to falling turbidity

169:  Z=F (P30)
 1: 0.2      F ;                  <========      Enter value for the
 2: 00       Exponent of 10 ;                    reversal % from
 3: 149      Z Loc [ revpct2   ] ;               falling to rising

170:  Z=F (P30)
 1: 2        F ;                  <========      Enter # of intervals
 2: 00       Exponent of 10 ;                    before a reversal
 3: 159      Z Loc [ stay      ] ;               or new threshold

171:  Z=F (P30)
 1: 8        F ;                  <========      Enter # of intervals
 2: 00       Exponent of 10 ;                    between two utilizations
 3: 146      Z Loc [ rep_wait  ] ;               of the same threshold

172:  Z=F (P30) ;                                mV tubidity limit
 1: 2500     F ;                  <========      above this value,
 2: 00       Exponent of 10 ;                    samples are collected
```

```
 3: 133      Z Loc [ mv_limit  ] ;              at fixed time
;                                              intervals (lim_skip),
;                                              OBS-3 only

173:  Z=F (P30)
 1: 1850     F ;                  <========     NTU turbidity limit
 2: 00       Exponent of 10 ;                   DTS-12 only (see
 3: 137      Z Loc [ ntu_limit ] ;              instuction above)


174:  Z=F (P30) ;                               Number of intervals
 1: 2        F ;                  <========     skipped (not sampled)
 2: 00       Exponent of 10 ;                   between samples, when
 3: 126      Z Loc [ lim_skip  ] ;              mV limit exceeded

;---------------------------------------------------------------------
;Other initializations

;Initialize the dump counter = 1
175:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 192      Z Loc [ dump_cnt  ]

;Initialize DTS-12 temperature (dts_temp)
176:  Z=F (P30)
 1: 999      F
 2: 00       Exponent of 10
 3: 174      Z Loc [ dts_temp  ]

;Initialize water temp (wtemp)
177:  Z=F (P30)
 1: 999      F
 2: 00       Exponent of 10
 3: 182      Z Loc [ wtemp     ]

;Initialize air temp (atemp)
178:  Z=F (P30)
 1: 999      F
 2: 00       Exponent of 10
 3: 181      Z Loc [ atemp     ]

;Set interval = 0
179:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 117      Z Loc [ interval  ]

;Set bottle counter to 0
180:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 107      Z Loc [ bot_cnt   ]

;Set next pump sampler counter to 1
181:  Z=F (P30)
 1: 1        F
```

```
 2: 00       Exponent of 10
 3: 193      Z Loc [ nxt_bot   ]

;Set reversal counter = 0
182:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;Set threshold counter to 0
183:  Z=F (P30)
 1: 0.0      F
 2: 00       Exponent of 10
 3: 187      Z Loc [ thr_count ]

;Set inbounds = 1
184:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 115      Z Loc [ inbounds  ]

;Set last_rsam = 0
185:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 124      Z Loc [ last_rsam ]

;Set last_fsam = 0
186:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 121      Z Loc [ last_fsam ]

;Set last_rthr = 0
187:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 125      Z Loc [ last_rthr ]

;Set last_fthr = 0
188:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 122      Z Loc [ last_fthr ]

;---------------------------------------------------------------------

;Set source location address for indirect moves
189:  Z=F (P30)
 1: 62       F
 2: 00       Exponent of 10
 3: 156      Z Loc [ sourc_loc ]

;Set destination location address for indirect moves
190:  Z=F (P30)
 1: 63       F
 2: 00       Exponent of 10
```

```
 3: 110      Z Loc [ dest_loc  ]

;---------------------------------------------------------------------
;Threshold locations - modify values indicated by <========

;Set the location where the rising threshold array begins
191:  Z=F (P30)
 1: 65       F
 2: 00       Exponent of 10
 3: 157      Z Loc [ start1    ]

;Set the location where the falling threshold array begins
192:  Z=F (P30)
 1: 81       F
 2: 00       Exponent of 10
 3: 158      Z Loc [ start2    ]

;Set maxrindex = start1, plus the number of rising thresholds - 1
193:  Z=X+F (P34)
 1: 157      X Loc [ start1    ]
 2: 11       F ;                 <========   Enter # of rising thresholds
 3: 129      Z Loc [ maxrindex ]

;Set maxfindex = start2, plus the number of falling thresholds - 1
194:  Z=X+F (P34)
 1: 158      X Loc [ start2    ]
 2: 17       F ;                 <========   Enter # of falling thresholds
 3: 128      Z Loc [ maxfindex ]

;---------------------------------------------------------------------

;Enter the Rising threshold values, maximum of 16 values
195:  Bulk Load (P65)
 1: 0        F
 2: 20       F
 3: 77       F
 4: 170      F
 5: 300      F
 6: 467      F
 7: 670      F
 8: 910      F
 9: 65       Loc [ UP_1      ]

196:  Bulk Load (P65)
 1: 1187     F
 2: 1500     F
 3: 1850     F
 4: 9999     F
 5: 0.0      F
 6: 0.0      F
 7: 0.0      F
 8: 0.0      F
 9: 73       Loc [ up_9      ]

;---------------------------------------------------------------------

;Enter the Falling threshold values, maximum of 24
```

```
197:  Bulk Load (P65)
 1: 1900     F
 2: 1698     F
 3: 1507     F
 4: 1328     F
 5: 1160     F
 6: 1004     F
 7: 858      F
 8: 724      F
 9: 81       Loc [ DN_1      ]

198:  Bulk Load (P65)
 1: 602      F
 2: 491      F
 3: 391      F
 4: 302      F
 5: 225      F
 6: 159      F
 7: 105      F
 8: 62       F
 9: 89       Loc [ dn_9      ]

199:  Bulk Load (P65)
 1: 30       F
 2: 0.0      F
 3: 0.0      F
 4: 0.0      F
 5: 0.0      F
 6: 0.0      F
 7: 0.0      F
 8: 0.0      F
 9: 97       Loc [ dn_17     ]

;---------------------------------------------------------------------

;End of subroutine 7
200:  End (P95)

;----------------------=====================----------------------
;----------------------=====================----------------------
;----------------------=====================----------------------


;SUBROUTINE 8 find the next threshold location
201:  Beginning of Subroutine (P85)
 1: 8        Subroutine 8

;Place threshold at nextindex into temporary storage (DEST)
202:  Indirect Move (P61)
 1: 135      Source Loc [ nextindex ]
 2: 110      Destination Loc [ dest_loc  ]

;Copy threshold from DEST to next_thr
203:  Z=X (P31)
 1: 63       X Loc [ DEST      ]
 2: 136      Z Loc [ next_thr  ]
```

```
;End of subroutine 8
204:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------
;-----------------------------------------------------------------

;SUBROUTINE 9 control sampling above senor's max turbidity

205:  Beginning of Subroutine (P85)
 1: 9        Subroutine 9

;If [thr_code = 2], then turbidity condition is falling
206:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 2        F
 4: 30       Then Do

;Set thr_code = 1 (rising)
207:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 190      Z Loc [ thr_code  ]

;Set tmax = med_turb
208:  Z=X (P31)
 1: 178      X Loc [ med_turb  ]
 2: 163      Z Loc [ tmax      ]

;Set rev_count = 0
209:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;Call subroutine 5 to determine max threshold
210:  Do (P86)
 1: 5        Call Subroutine 5

;End of case [if thr_code = 2]
211:  End (P95)

;Find the sum of last_rsam and lim_skip
212:  Z=X+Y (P33)
 1: 124      X Loc [ last_rsam ]
 2: 126      Y Loc [ lim_skip  ]
 3: 160      Z Loc [ sum       ]

;If [interval > last_rsam + lim_skip)
213:  If (X<=>Y) (P88)
 1: 160      X Loc [ sum       ]
 2: 4        <
 3: 117      Y Loc [ interval  ]
 4: 30       Then Do

;Find the sum of last_fsam and lim_skip
```

```
214:  Z=X+Y (P33)
 1: 121      X Loc [ last_fsam ]
 2: 126      Y Loc [ lim_skip  ]
 3: 160      Z Loc [ sum       ]

;If [interval > last_fsam + lim_skip]
215:  If (X<=>Y) (P88)
 1: 160      X Loc [ sum       ]
 2: 4        <
 3: 117      Y Loc [ interval  ]
 4: 30       Then Do

;Set smp_code = 5 (overflow sample)
216:  Z=F (P30)
 1: 5        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to collect pumped sample
217:  Do (P86)
 1: 6        Call Subroutine 6

;Set last_rsam = interval
218:  Z=X (P31)
 1: 117      X Loc [ interval  ]
 2: 124      Z Loc [ last_rsam ]

;Set rev_count = 0
219:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 186      Z Loc [ rev_count ]

;End of case [If interval > last_fsam + lim_skip]
220:  End (P95)

;End of case [If interval > last_rsam + lim_skip]
221:  End (P95)

;End of subroutine 9
222:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------
;-----------------------------------------------------------------

;SUBROUTINE 79 threshold samples.
223:  Beginning of Subroutine (P85)
 1: 79       Subroutine 79

;Set do_sample = 0
224:  Z=F (P30)
 1: 0.0      F
 2: 00       Exponent of 10
 3: 111      Z Loc [ do_sample ]

;If [thr_code = 1], turbidity is rising, then
```

```
225:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 01       =
 3: 1        F
 4: 30       Then Do

;If [rthr <> last_rthr], then do
226:  If (X<=>Y) (P88)
 1: 152      X Loc [ rthr      ]
 2: 2        <>
 3: 125      Y Loc [ last_rthr ]
 4: 30       Then Do

;Set do_sample = 1 for threshold sample
227:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 111      Z Loc [ do_sample ]

;End of case, [If rthr <> last_rthr]
228:  End (P95)

;Set rep_inter = rep_wait + last_rsam
229:  Z=X+Y (P33)
 1: 124      X Loc [ last_rsam ]
 2: 146      Y Loc [ rep_wait  ]
 3: 145      Z Loc [ rep_inter ]

;If [rep_inter < interval], then do
230:  If (X<=>Y) (P88)
 1: 145      X Loc [ rep_inter ]
 2: 4        <
 3: 117      Y Loc [ interval  ]
 4: 30       Then Do

;Set do_sample = 1 for reversal sample
231:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 111      Z Loc [ do_sample ]

;End of case, [If rep_inter < interval]
232:  End (P95)

;Else [thr_code = 2], turbidity is falling
233:  Else (P94)

;If [fthr <> last_fthr], then do
234:  If (X<=>Y) (P88)
 1: 114      X Loc [ fthr      ]
 2: 2        <>
 3: 122      Y Loc [ last_fthr ]
 4: 30       Then Do

;Set do_sample = 1 for threshold sample
235:  Z=F (P30)
 1: 1        F
```

```
 2: 00       Exponent of 10
 3: 111      Z Loc [ do_sample ]

;End of case, [If fthr <> last_fthr]
236:  End (P95)

;Set rep_inter = rep_wait + last_fsam
237:  Z=X+Y (P33)
 1: 121      X Loc [ last_fsam ]
 2: 146      Y Loc [ rep_wait  ]
 3: 145      Z Loc [ rep_inter ]

;If [rep_inter < interval], then do
238:  If (X<=>Y) (P88)
 1: 145      X Loc [ rep_inter ]
 2: 4        <
 3: 117      Y Loc [ interval  ]
 4: 30       Then Do

;Set do_sample = 1 for threshold sample
239:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 111      Z Loc [ do_sample ]

;End of case, [If rep_inter < interval]
240:  End (P95)

;End of case [else thr_code = 2]
241:  End (P95)

;If [do_sample = 1], then proceed
242:  If (X<=>F) (P89)
 1: 111      X Loc [ do_sample ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;If [smp_code = 0], then no pumped samples have been taken
243:  If (X<=>F) (P89)
 1: 191      X Loc [ smp_code  ]
 2: 1        =
 3: 0        F
 4: 30       Then Do

;Set smp_code = 1 for threshold sample collected
244:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;Call subroutine 6 to take pumped sample
245:  Do (P86)
 1: 6        Call Subroutine 6

;End of case, [If smp_code = 0]
246:  End (P95)
```

```
;If [smp_code = 3], then AUX was already selected
247:  If (X<=>F) (P89)
 1: 191      X Loc [ smp_code  ]
 2: 1        =
 3: 3        F
 4: 30       Then Do

;Set sample code = 1 (overide sample code 3)
248:  Z=F (P30)
 1: 1        F
 2: 00       Exponent of 10
 3: 191      Z Loc [ smp_code  ]

;End of case, [If sample code = 3]
249:  End (P95)

;-----------------------------------------------------------------------

;If [thr_code = 1], turbidity is rising
250:  If (X<=>F) (P89)
 1: 190      X Loc [ thr_code  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Set last_rsam = interval
251:  Z=X (P31)
 1: 117      X Loc [ interval  ]
 2: 124      Z Loc [ last_rsam ]

;Set last_rthr = rthr
252:  Z=X (P31)
 1: 152      X Loc [ rthr      ]
 2: 125      Z Loc [ last_rthr ]

;Else [thr_code = 2], turbidity is falling
253:  Else (P94)

;Set last_fsam = interval
254:  Z=X (P31)
 1: 117      X Loc [ interval  ]
 2: 121      Z Loc [ last_fsam ]

;Set last_fthr = fthr
255:  Z=X (P31)
 1: 114      X Loc [ fthr      ]
 2: 122      Z Loc [ last_fthr ]

;End of case [Else thr_code = 2]
256:  End (P95)

;End of case [If do_sample = 1]
257:  End (P95)

;End of subroutine 79
258:  End (P95)
```

```
;-----------------------==================-----------------------
;-----------------------==================-----------------------
;-----------------------==================-----------------------


259:  Beginning of Subroutine (P85)
 1: 80       Subroutine 80 ; measure turbidity

; If [turb_dev = 1], OBS-3 connected
260:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Initialize turbidity loop counter (k = 0)
;Assumes turbidity mv array starts at memory location 1
261:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 120      Z Loc [ k         ]

;Set Port 1 high 12V --> OBS-3
262:  Do (P86)
 1: 41       Set Port 1 High

;Reset timer
263:  Timer (P26)
 1: 0000     Reset Timer

;---------------------------------------------------------------------

;Begin loop to warm up T-probe
264:  Beginning of Loop (P87)
 1: 0000     Delay
 2: 9999     Loop Count

;Store timer value in location prob_time
265:  Timer (P26)
 1: 139      Loc [ prob_time ]

;If [prob_time>=2.5], then exit loop
266:  If (X<=>F) (P89)
 1: 139      X Loc [ prob_time ]
 2: 3        >=
 3: 2.5      F
 4: 31       Exit Loop if True

;End of loop to warm up T-probe
267:  End (P95)

;-----------------------==================-----------------------
;-----------------------==================-----------------------

;Begin loop to read 60 turbidity values (mV)
268:  Beginning of Loop (P87)
```

```
 1: 0000     Delay
 2: 60       Loop Count

;Reads turbidity probe (mV)
269:  Excite-Delay (SE) (P4)
 1: 1        Reps
 2: 5        2500 mV Slow Range
 3: 3        SE Channel
 4: 3        Excite all reps w/Exchan 3 ; not connected
 5: 30       Delay (units 0.01 sec)
 6: 0        mV Excitation
 7: 142      Loc [ raw_mv    ] ;    mV (raw value)
 8: 1        Mult
 9: 0.0      Offset

;---------------------------------------------------------------------

;Call subroutine 1 to sort mV values into ascending order
270:  Do (P86)
 1: 1        Call Subroutine 1

;---------------------------------------------------------------------

;End of loop to read turbidity
271:  End (P95)

;----------------------=====================----------------------------

;Set Port 1 low to turn off turbidity probe
272:  Do (P86)
 1: 51       Set Port 1 Low

;----------------------=====================----------------------------

;Call subroutine 2 to determine the median mV value
273:  Do (P86)
 1: 2        Call Subroutine 2

;----------------------=========================--------------------

;Convert millivolts to NTUs, med_turb = med_mv * turb_mult
274:  Z=X*Y (P36)
 1: 132      X Loc [ med_mv    ]
 2: 194      Y Loc [ turb_mult ]
 3: 178      Z Loc [ med_turb  ] ;   NTUs

;Add offset to turbidity values, med_turb = med_turb + turb_off
275:  Z=X+Y (P33)
 1: 178      X Loc [ med_turb  ]
 2: 195      Y Loc [ turb_off  ]
 3: 178      Z Loc [ med_turb  ]

;End of case [if turb_dev = 1] OBS-3
276:  End (P95)

;---------------------------------------------------------------------
```

```
;If [turb_dev <> 1] then DTS-12 connected (CR10X or CR510 OK)
277:  If (X<=>F) (P89)
 1: 165      X Loc [ turb_dev  ]
 2: 2        <>
 3: 1        F
 4: 30       Then Do

;Set wiper flag condition to false (no wipe)
278:  Z=F (P30)
 1: 0        F ;                        Set false condition
 2: 00       Exponent of 10
 3: 166      Z Loc [ wiper     ]

;Measure DTS-12 temperature
279:  SDI-12 Recorder (P105)
 1: 0        SDI-12 Address ;           DTS-12 device address 0
 2: 0        Start Measurement (aM0!) ; Measure only
 3: 2        Port ;                     <-- C2
 4: 174      Loc [ dts_temp  ] ;        Start address of 3 inputs
 5: 1.0      Mult ;                     Not used
 6: 0.0      Offset ;                   Not used

;If [dts_temp >= 0.5 deg C] water temperature is above freezing
280:  If (X<=>F) (P89)
 1: 174      X Loc [ dts_temp  ]
 2: 3        >=
 3: 0.5      F
 4: 30       Then Do

;If [stage >= min_stg] sensor is submerged
281:  If (X<=>Y) (P88)
 1: 179      X Loc [ stage     ]
 2: 3        >=
 3: 189      Y Loc [ min_stg   ]
 4: 30       Then Do

;Set wipe flag to true when sensor is above freezing AND submerged
282:  Z=F (P30)
 1: 1        F ;                        Set true condition
 2: 00       Exponent of 10
 3: 166      Z Loc [ wiper     ]

;End case [If stage >= min_stg]
283:  End (P95)

;End case [If dts_temp >= 0.5 deg C]
284:  End (P95)

;If[wipe = 1] then wipe optics
285:  If (X<=>F) (P89)
 1: 166      X Loc [ wiper     ]
 2: 1        = ;                         OK to activate wiper
 3: 1        F
 4: 30       Then Do

;Measure turbidity with DTS-12
286:  SDI-12 Recorder (P105)
```

```
 1: 0        SDI-12 Address ;            DTS-12 device address 0
 2: 2        Start Measurement (aM2!) ;  Activate wiper, then measure
 3: 2        Port ;                      <-- C2
 4: 168      Loc [ dts_mean  ] ;         Start address of 6 results
 5: 1.0      Mult ;                      Not used
 6: 0        Offset ;                    Not used

;Else ice may be forming or sensor is dry, don't activate wiper
287:  Else (P94)

;Measure turbidity with DTS-12
288:  SDI-12 Recorder (P105)
 1: 0        SDI-12 Address ;            DTS-12 device address 0
 2: 1        Start Measurement (aM1!) ;  Measure only
 3: 2        Port ;                      <-- C2
 4: 168      Loc [ dts_mean  ] ;         Start address of 6 results
 5: 1.0      Mult ;                      Not used
 6: 0.0      Offset ;                    Not used

;End case [if wipe = 1]
289:  End (P95)

;-----------------------------------------------------------------

;Add [turb_off + dts_med]
290:  Z=X+Y (P33)
 1: 170      X Loc [ dts_med   ]
 2: 195      Y Loc [ turb_off  ]
 3: 170      Z Loc [ dts_med   ]

;Let [med_turb = dts_med]                Assign DTS-12 median to
291:  Z=X (P31) ;                        med_turb
 1: 170      X Loc [ dts_med   ] ;       dts_med is the 3rd value
 2: 178      Z Loc [ med_turb  ] ;       in the list of 6 inputs

;End case [if turb_dev <> 1]
292:  End (P95)

;If [Flag 8 is low], then convert turbidity to integer
293:  If Flag/Port (P91)
 1: 28       Do if Flag 8 is Low
 2: 30       Then Do

;Round turbidity (by adding 0.5)
294:  Z=X+F (P34)
 1: 178      X Loc [ med_turb  ]
 2: .5       F
 3: 178      Z Loc [ med_turb  ]

;Truncate to integer
295:  Z=INT(X) (P45)
 1: 178      X Loc [ med_turb  ]
 2: 178      Z Loc [ med_turb  ]

;End of case [Flag 8 is low]
296:  End (P95)
```

```
;-----------------------------------------------------------------

;End of subroutine 80
297:  End (P95)

;-----------------------------------------------------------------
;-----------------------------------------------------------------
;-----------------------------------------------------------------

298:  Beginning of Subroutine (P85)
 1: 81       Subroutine 81 ; get stage from Druck 1830 pressure transducer

;Initialize total stage = 0
299:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 164      Z Loc [ tot_stg   ]

;Initialize stage loop counter (n = 0)
300:  Z=F (P30)
 1: 0        F
 2: 00       Exponent of 10
 3: 134      Z Loc [ n         ]

;Begining of loop to read the pressure transducer
301:  Beginning of Loop (P87)
 1: 0000     Delay
 2: 150      Loop Count ;         <-- collect 150 measurements

;Increment counter
302:  Z=Z+1 (P32)
 1: 134      Z Loc [ n         ]

;Reads pressure transducer
303:  Full Bridge (P6)
 1: 1        Reps
 2: 3        25 mV Slow Range
 3: 1        DIFF Channel
 4: 1        Excite all reps w/Exchan 1
 5: 2500     mV Excitation
 6: 143      Loc [ raw_stg   ]
 7: 1.0      Mult
 8: 0.0      Offset

;Adds up total stage each time the loop is executed
304:  Z=X+Y (P33)
 1: 143      X Loc [ raw_stg   ]
 2: 164      Y Loc [ tot_stg   ]
 3: 164      Z Loc [ tot_stg   ]

;End of loop to read pressure transducer
305:  End (P95)

;-----------------------------------------------------------------

;Compute average stage
306:  Z=X/Y (P38)
```

```
 1: 164      X Loc [ tot_stg   ]
 2: 134      Y Loc [ n         ]
 3: 106      Z Loc [ avg_stg   ]

;Convert millivolts to feet of stage, stage = stage * stg_mult
307:  Z=X*Y (P36)
 1: 106      X Loc [ avg_stg   ]
 2: 196      Y Loc [ stg_mult  ]
 3: 179      Z Loc [ stage     ]

;Add offset to stage, stage = stage + stg_off
308:  Z=X+Y (P33)
 1: 179      X Loc [ stage     ]
 2: 197      Y Loc [ stg_off   ]
 3: 179      Z Loc [ stage     ]

;---------------------------------------------------------------------

;End of subroutine 81
309:  End (P95)

;----------------------=================-----------------------
;----------------------=================-----------------------
;----------------------=================-----------------------

310:  Beginning of Subroutine (P85)
 1: 82       Subroutine 82 ; Measure water/air temperature

;If [Flag 5 is high], then measure water temperaure T107
311:  If Flag/Port (P91)
 1: 15       Do if Flag 5 is High
 2: 30       Then Do

;If [logger = 1] then CR10X in use
312:  If (X<=>F) (P89)
 1: 127      X Loc [ logger    ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Read water temperature
313:  Temp (107) (P11)
 1: 1        Reps
 2: 5        SE Channel ;              <-- SE 5 CR10X only
 3: 2        Excite all reps w/E2
 4: 182      Loc [ wtemp     ]
 5: 1.0      Mult
 6: 0.0      Offset

;End case [If logger = 1]
314:  End (P95)

;End of case [If Flag 5 is high]
315:  End (P95)

;If [Flag 6 is high], then measure air temperature T107
316:  If Flag/Port (P91)
```

```
 1: 16       Do if Flag 6 is High
 2: 30       Then Do

;Read air temperature
317:  Temp (107) (P11)
 1: 1        Reps
 2: 4        SE Channel ;             <-- SE 4 CR10X or CR510
 3: 2        Excite all reps w/E2
 4: 181      Loc [ atemp     ]
 5: 1.0      Mult
 6: 0.0      Offset

;End of case [If Flag 6 is high]
318:  End (P95)

;End of subroutine 82
319:  End (P95)

;-----------------------================================-----------------------
;-----------------------================================-----------------------
;-----------------------================================-----------------------

320:  Beginning of Subroutine (P85)
 1: 83       Subroutine 83 ; Calculate water discharge
;If [qcalc = 1], then compute discharge

321:  If (X<=>F) (P89)
 1: 140      X Loc [ qcalc     ]
 2: 1        =
 3: 1        F
 4: 30       Then Do

;Raise average stage to a power
322:  Z=X^Y (P47)
 1: 179      X Loc [ stage     ]
 2: 199      Y Loc [ q_exp     ]
 3: 141      Z Loc [ q_pow     ]

;Calc discharge
323:  Z=X*Y (P36)
 1: 198      X Loc [ q_mult    ]
 2: 141      Y Loc [ q_pow     ]
 3: 180      Z Loc [ disch     ]

;End of case [qcalc = 1]
324:  End (P95)

;End of subroutine 83
325:  End (P95)

;---------------------------------------------------------------------------

End Program

-Input Locations-
1 mv_1      1 0 0
2 mv_2      1 0 0
```

```
3 mv_3      1 0 0
4 mv_4      1 0 0
5 mv_5      1 0 0
6 mv_6      1 0 0
7 mv_7      1 0 0
8 mv_8      1 0 0
9 mv_9      1 0 0
10 mv_10     1 0 0
11 mv_11     1 0 0
12 mv_12     1 0 0
13 mv_13     0 0 0
14 mv_14     0 0 0
15 mv_15     1 0 0
16 mv_16     1 0 0
17 mv_17     1 0 0
18 mv_18     1 0 0
19 mv_19     1 0 0
20 mv_20     1 0 0
21 mv_21     1 0 0
22 mv_22     1 0 0
23 mv_23     1 0 0
24 mv_24     1 0 0
25 mv_25     1 0 0
26 mv_26     1 0 0
27 mv_27     1 0 0
28 mv_28     1 0 0
29 mv_29     1 0 0
30 mv_30     1 0 0
31 mv_31     0 0 0
32 mv_32     0 0 0
33 mv_33     0 0 0
34 mv_34     0 0 0
35 mv_35     0 0 0
36 mv_36     0 0 0
37 mv_37     0 0 0
38 mv_38     0 0 0
39 mv_39     0 0 0
40 mv_40     0 0 0
41 mv_41     0 0 0
42 mv_42     0 0 0
43 mv_43     0 0 0
44 mv_44     0 0 0
45 mv_45     0 0 0
46 mv_46     0 0 0
47 mv_47     0 0 0
48 mv_48     0 0 0
49 mv_49     0 0 0
50 mv_50     0 0 0
51 mv_51     0 0 0
52 mv_52     0 0 0
53 mv_53     0 0 0
54 mv_54     0 0 0
55 mv_55     0 0 0
56 mv_56     0 0 0
57 mv_57     0 0 0
58 mv_58     0 0 0
59 mv_59     0 0 0
```

```
60 mv_60      0 0 0
61 _________ 0 0 0
62 SOURCE     1 0 1
63 DEST       1 8 0
64 _________ 0 0 0
65 UP_1       5 0 1
66 up_2       9 0 1
67 up_3       9 0 1
68 up_4       1 0 0
69 up_5       9 0 1
70 up_6       9 0 1
71 up_7       9 0 1
72 up_8       17 0 1
73 up_9       5 0 1
74 up_10      9 0 1
75 up_11      9 0 1
76 up_12      9 0 1
77 up_13      9 0 1
78 up_14      9 0 1
79 up_15      9 0 1
80 up_16      17 0 1
81 DN_1       5 2 1
82 dn_2       9 0 1
83 dn_3       9 0 1
84 dn_4       9 0 1
85 dn_5       9 0 1
86 dn_6       9 0 1
87 dn_7       9 0 1
88 dn_8       17 0 1
89 dn_9       5 0 1
90 dn_10      9 0 1
91 dn_11      9 0 1
92 dn_12      9 0 1
93 dn_13      9 0 1
94 dn_14      9 0 1
95 dn_15      9 0 1
96 dn_16      17 0 1
97 dn_17      5 0 1
98 dn_18      9 0 1
99 dn_19      9 0 1
100 dn_20      9 0 1
101 dn_21      9 0 1
102 dn_22      9 0 1
103 dn_23      9 0 1
104 dn_24      17 0 1
105 _________ 0 0 0
106 avg_stg   1 3 1
107 bot_cnt   1 1 3
108 bottle    1 2 2
109 delay     1 2 1
110 dest_loc  1 0 9
111 do_sample 1 1 4
112 dummy     1 0 1
113 frst_rthr 1 2 1
114 fthr      1 3 3
115 inbounds  1 1 3
116 insert    1 1 3
```

```
117 interval  1 12 2
118 j         1 3 4
119 jminus1   1 2 1
120 k         1 4 3
121 last_fsam 1 2 2
122 last_fthr 1 1 2
123 last_mv   1 1 1
124 last_rsam 1 4 5
125 last_rthr 1 1 4
126 lim_skip  1 2 1
127 logger    1 2 1
128 maxfindex 1 2 1
129 maxrindex 1 1 1
130 med1      1 1 1
131 med2      1 1 1
132 med_mv    1 3 2
133 mv_limit  1 2 1
134 n         1 1 2
135 nextindex 1 6 4
136 next_thr  1 4 1
137 ntu_limit 1 2 1
138 old_turb  1 3 1
139 prob_time 1 1 2
140 qcalc     1 1 1
141 q_pow     1 1 1
142 raw_mv    1 2 1
143 raw_stg   1 1 1
144 rem       1 1 1
145 rep_inter 1 2 2
146 rep_wait  1 2 1
147 rev_mult  1 4 2
148 revpct1   1 1 1
149 revpct2   1 1 1
150 rev_thr   1 2 4
151 rev_val   1 4 1
152 rthr      1 3 2
153 scratch   1 3 3
154 source1   1 7 6
155 source2   1 1 1
156 sourc_loc 1 1 1
157 start1    1 3 1
158 start2    1 2 1
159 stay      1 4 1
160 sum       1 3 3
161 timer     1 1 1
162 tmin      1 5 3
163 tmax      1 5 5
164 tot_stg   1 2 2
165 turb_dev  1 9 2
166 wiper     1 1 2
167 _________ 0 0 0
168 dts_mean  1 0 2
169 dts_var   0 0 0
170 dts_med   1 2 1
171 dts_bes   0 0 0
172 dts_min   0 0 0
173 dts_max   0 0 0
```

```
174 dts_temp  1 3 2
175 dts_mean1 0 0 0
176 dts_var1  0 0 0
177 _________ 0 0 0
178 med_turb  1 29 5
179 stage     1 6 3
180 disch     1 0 1
181 atemp     1 4 4
182 wtemp     1 4 4
183 rain      1 5 3
184 daily     1 1 2
185 season    1 1 1
186 rev_count 1 2 9
187 thr_count 1 2 4
188 thr       1 4 3
189 min_stg   1 4 1
190 thr_code  1 10 9
191 smp_code  1 6 8
192 dump_cnt  1 1 2
193 nxt_bot   1 0 3
194 turb_mult 1 1 1
195 turb_off  1 2 1
196 stg_mult  1 1 1
197 stg_off   1 1 1
198 q_mult    1 1 1
199 q_exp     1 1 1
200 bat_volt  1 0 1
-Program Security-
0000
0000
0000
-Mode 4-
-Final Storage Area 2-
0
-CR10X ID-
0
-CR10X Power Up-
3
```