

```
; {CR10X}
;
;TTS_3_0.csi
;March 9, 2000

; ----- 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

; 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.

; Instruction parameters are set for the following types of equipment:
;     Data loggers: CR510, CR500, CR10, or CR10X
;     Turbidity probe: D & A Instruments, OBS-3, 2.5V clamp
;     Pressure transducer: Druck PDCR 1830 differential mV output
;     Pumping sampler: ISCO 3700
; The use of different devices may require modification to specific parameters.
; 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,
; and samples turbidity at 0.5 second intervals for 30 seconds for a
; total of 60 readings. 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.

; A document containing program notes and definitions can be found at
; http://www.rsl.psw.fs.fed.us/projects/water/tts\_definitions.pdf

; 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.

; 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 5 (rev_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.

; OVERFLOW: Turbidity sensor output exceeds the data logger's
; millivolt limit setting (mv_limit). In this mode, one (lim_skip)
; interval 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
; 2: Indicates falling threshold
; 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 above mV limit collected

; 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 to initialize variables
; -High: Warm start, continue with normal execution

```

```

; FLAG 5:
;   -Low: Water temperature sensor not connected
;   -High: Water temperature sensor (107) connected
; FLAG 6:
;   -Low: Air temperature sensor not connected
;   -High: Air temperature sensor (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
; 2: Determines the median turbidity value
; 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 T-probe output is above mV limit
; 79: Threshold sampling code

;***** MEMORY ADDRESSES *****
; DO NOT move, insert, or delete instructions between the following
; address locations or the program will not function correctly.
; These locations are hard-coded in the program (locations, not
; variable names, are referenced). Always add new variables,
; via the Input Location Editor, following the highest address listed
; below. If a variable, from an address NOT listed below, is no
; longer needed delete the variable name but leave the location blank.
; Addr
;     1 - 61
;     63 - 64
;     66 - 67
;     71 - 74
;    130 - 131
;    140 - 185

;***** SENSOR/DEVICE CONNECTIONS *****
; Required sensors and connections for TTS:
; CR510, CR500, or CR10 data loggers
;     Druck PDCR 1830, mV output, Pressure Transducer
;     E1
;     L1 H1
; OBS-3 Turbidity Probe
;     E3 (connection not available)
;     SE3
;     C1 --> relay
; ISCO 3700 Flow connection
;     E2 --> FET transistor --> relay

```

```

; CR10X
; Druck PDCR 1830, mV output, Pressure Transducer
;   E1
;   L1 H1
; OBS-3 Turbidity Probe
;   E3 (reserved, but no connection)
;   SE3
;   C1 --> SW 12V
; ISCO 3700 Flow connection
;   C2 --> relay

; Optional sensors and connecitons
; Rain Gauge
;   P1
; CR10X only
;   Water temperature probe, Campbell Scientific 107
;   E2
;   SE4
; Air temperature probe, Campbell Scientific 107
;   E2
;   SE5

;***** DATA LOGGERS *****
; CR510, CR500, and CR10 (requires 2 relays, one controlled
; with FET transistor)
; CR10X (requires 1 relay)

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

; Table 1, Execution interval (600 or 900 seconds)
;           L62, Set 2: to 10 or 15, to match the execution
;           interval above
; Table 2, Sub 3, L62, Storage Area, array id (1 - 511)
;           Sub 7, L148, Minimum stage
;           L149, T-Probe multiplier
;           L150, T-Probe offset
;           L151, Pressure transducer multiplier
;           L152, Pressure transducer offset
;           L154, Flag, water temperature (CR10X only)
;           L155, Flag, air temperature (CR10X only)
;           L156, Flag, rain gage
;           L157, Flag, turbidity resolution
;           L158, isco_ex, excitation port for isco
;           L159, qcalc, flag to activate discharge calculation
;           L160, q_mult, multiplier for discharge equation
;           L161, q_exp, exponent for discharge equation
;           L174, Number of non-zero rising thresholds, set
;           parameter 2
;           L175 Number of non-zero falling thresholds, set
;           parameter 2
;           L185, Start threshold bulk load
;           L189, End threshold bulk load

=====

```

```

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

;Read battery voltage
1: Batt Voltage (P10)
1: 124      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: 108       Z Loc [ dump_cnt  ]

;Reset bottle counter to 0
8: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 112        Z Loc [ bot_cnt  ]

;Reset next ISCO counter to 1
9: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 111        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)

```

```

;-----
;Set sample code = 0, for no thresholds sampled
12: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 102        Z Loc [ smp_code   ]

;Set bottle = 0
13: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 86         Z Loc [ bottle     ]

;Initialize total stage = 0
14: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 79         Z Loc [ tot_stg   ]

;Initialize turbidity loop counter (k = 0)
15: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 65         Z Loc [ k         ]

;Initialize stage loop counter (n = 0)
16: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 126        Z Loc [ n         ]

;Increment interval counter each time the table is executed
17: Z=Z+1 (P32)
1: 95         Z Loc [ interval   ]

;Set Port 1 high to turn on T-probe
18: Do (P86)
1: 41         Set Port 1 High

;Reset timer
19: Timer (P26)
1: 0          Reset Timer

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

;Store timer value in location prob_time
21: Timer (P26)
1: 109        Loc [ prob_time  ]

;If [prob_time>=2.5], then exit loop
22: If (X<=>F) (P89)

```

```

1: 109      X Loc [ prob_time ]
2: 3          >=
3: 2.5      F
4: 31       Exit Loop if True

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

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

;Begin loop to read 60 turbidity values (mV)
24: Beginning of Loop (P87)
1: 0000      Delay
2: 60       Loop Count

;Reads turbidity probe (mV)
25: 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: 61         Loc [ raw_mv      ] ;     mV (raw value)
8: 1          Mult
9: 0.0        Offset

;-----

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

;-----

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

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

;Set Port 1 low to turn off T-probe
28: Do (P86)
1: 51         Set Port 1 Low

;-----

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

;-----

;Convert millivolts to FTU's, med_turb = med_mv * turb_mult
30: Z=X*Y (P36)
1: 76         X Loc [ med_mv      ]
2: 114        Y Loc [ turb_mult  ]

```

```

3: 77          Z Loc [ med_turb ] ;    FTUs

;Add offset to turbidity values, med_turb = med_turb + turb_off
31: Z=X+Y (P33)
1: 77          X Loc [ med_turb ]
2: 115         Y Loc [ turb_off ]
3: 77          Z Loc [ med_turb ]

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

;Round turbidity (by adding 0.5)
33: Z=X+F (P34)
1: 77          X Loc [ med_turb ]
2: .5           F
3: 77          Z Loc [ med_turb ]

;Truncate to integer
34: Z=INT(X) (P45)
1: 77          X Loc [ med_turb ]
2: 77          Z Loc [ med_turb ]

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

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

;Increment counter
37: Z=Z+1 (P32)
1: 126         Z Loc [ n       ]

;Reads pressure transducer
38: 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: 125          Loc [ raw_stg ]
7: 1.0          Mult
8: 0.0          Offset

;Adds up total stage each time the loop is executed
39: Z=X+Y (P33)
1: 125         X Loc [ raw_stg ]
2: 79          Y Loc [ tot_stg ]
3: 79          Z Loc [ tot_stg ]

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

```

```

;-----
;Compute average stage
41: Z=X/Y (P38)
1: 79      X Loc [ tot_stg    ]
2: 126      Y Loc [ n        ]
3: 80      Z Loc [ avg_stg   ]

;Convert millivolts to feet of stage, stage = stage * stg_mult
42: Z=X*Y (P36)
1: 80      X Loc [ avg_stg   ]
2: 116      Y Loc [ stg_mult  ]
3: 78      Z Loc [ stage     ]

;Add offset to stage, stage = stage + stg_off
43: Z=X+Y (P33)
1: 78      X Loc [ stage     ]
2: 117      Y Loc [ stg_off   ]
3: 78      Z Loc [ stage     ]

;-----
;*** Non-TTS Sensors ***
;If [qcalc = 1], then compute discharge
44: If (X<=>F) (P89)
1: 193      X Loc [ qcalc     ]
2: 1       =
3: 1       F
4: 30      Then Do

;Raise average stage to a power
45: Z=X^Y (P47)
1: 78      X Loc [ stage     ]
2: 128      Y Loc [ q_exp     ]
3: 129      Z Loc [ q_prod    ]

;Calc discharge
46: Z=X*Y (P36)
1: 127      X Loc [ q_mult    ]
2: 129      Y Loc [ q_prod    ]
3: 139      Z Loc [ disch     ]

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

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

;Read water temperature
49: Temp (107) (P11)
1: 1      Reps
2: 4      SE Channel
3: 2      Excite all reps w/E2

```

```

4: 123      Loc [ wtemp      ]
5: 1.0      Mult
6: 0.0      Offset

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

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

;Read air temperature
52: Temp (107) (P11)
1: 1      Reps
2: 5      SE Channel
3: 2      Excite all reps w/E2
4: 192     Loc [ atemp      ]
5: 1.0      Mult
6: 0.0      Offset

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

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

;Read rain gage pulses
55: Pulse (P3)
1: 1      Reps
2: 1      Pulse Channel 1
3: 2      Switch Closure, All Counts
4: 122     Loc [ rain      ]
5: 0.01    Mult
6: 0.0      Offset

;Add rainfall from wakeup to daily total
56: Z=X+Y (P33)
1: 122     X Loc [ rain      ]
2: 186     Y Loc [ daily     ]
3: 186     Z Loc [ daily     ]

;Add rainfall from wakeup to season total
57: Z=X+Y (P33)
1: 122     X Loc [ rain      ]
2: 187     Y Loc [ season    ]
3: 187     Z Loc [ season    ]

;If midnight
58: 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

```

```

59: Z=F (P30)
1: 0.0      F
2: 00       Exponent of 10
3: 186      Z Loc [ daily      ]

;End if midnight
60: End (P95)

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

;-----
;If time is at the begining of a interval, then do
62: If time is (P92)
1: 0      Minutes (Seconds --) into a
2: 10     Interval (same units as above)
3: 30     Then Do

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

;Set sample code = 2 for DI sample taken
64: Z=F (P30)
1: 2      F
2: 00     Exponent of 10
3: 102    Z Loc [ smp_code   ]

;Call subroutine 6 to collect ISCO sample
65: Do (P86)
1: 6      Call Subroutine 6

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

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

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

;Set sample code = 3 for aux sample taken
69: Z=F (P30)
1: 3      F
2: 00     Exponent of 10
3: 102    Z Loc [ smp_code   ]

;Call subroutine 6 to collect ISCO sample
70: Do (P86)

```

```

1: 6           Call Subroutine 6

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

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

;-----
;If [interval < 3], then do
73: If (X<=>F) (P89)
1: 95           X Loc [ interval   ]
2: 4             <
3: 3             F
4: 30            Then Do

;-----
;If [interval = 1], then it is the first interval
74: If (X<=>F) (P89)
1: 95           X Loc [ interval   ]
2: 1             =
3: 1             F
4: 30            Then Do

;-----
;If [stage < minimum stage], then base flow
75: If (X<=>Y) (P88)
1: 78           X Loc [ stage      ]
2: 4             <
3: 92           Y Loc [ min_stg   ]
4: 30            Then Do

;Set threshold code to 0 for stage < min_stg (base flow)
76: Z=F (P30)
1: 0             F
2: 00            Exponent of 10
3: 101           Z Loc [ thr_code  ]

;Call subroutine 3 to write data to datalogger memory
77: Do (P86)
1: 3           Call Subroutine 3

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

;-----
;Set threshold code = 3 for unknown threshold condition
79: Z=F (P30)
1: 3             F
2: 00            Exponent of 10
3: 101           Z Loc [ thr_code  ]

```

```

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

;-----
;If [interval <> 1], then it is the second interval
81: If (X<=>F) (P89)
1: 95      X Loc [ interval   ]
2: 2      <>
3: 1      F
4: 30     Then Do

;-----
;If [thr_code = 0], then first reading was in base flow
82: If (X<=>F) (P89)
1: 101     X Loc [ thr_code   ]
2: 1      =
3: 0      F
4: 30     Then Do

;If [stage >= min_stg], then it is out of base flow
83: If (X<=>Y) (P88)
1: 78      X Loc [ stage      ]
2: 3      >=
3: 92      Y Loc [ min_stg   ]
4: 30     Then Do

;Call subroutine 4 for 1st interval above minimum stage
84: Do (P86)
1: 4      Call Subroutine 4

;End of case, [if stage >= min_stg]
85: End (P95)

;End of case, [if thr_code = 0]
86: End (P95)

;-----
;If [thr_code = 3], then first interval turbidity condition unknown
87: If (X<=>F) (P89)
1: 101     X Loc [ thr_code   ]
2: 1      =
3: 3      F
4: 30     Then Do

;Find the difference between old_turb and med_turb
88: Z=X-Y (P35)
1: 91      X Loc [ old_turb   ]
2: 77      Y Loc [ med_turb   ]
3: 113     Z Loc [ diff       ]

;-----
;If [diff < 0], then threshold is rising
89: If (X<=>F) (P89)

```

```

1: 113      X Loc [ diff      ]
2: 4          <
3: 0          F
4: 30         Then Do

;Set thr_code = 1 for rising threshold
90: Z=F (P30)
1: 1          F
2: 00        Exponent of 10
3: 101       Z Loc [ thr_code  ]

;Set tmax = med_turb
91: Z=X (P31)
1: 77        X Loc [ med_turb  ]
2: 82        Z Loc [ tmax      ]

;End of case, [If diff < 0]
92: End (P95)

;-----
;Else if [diff >= 0], then threshold is falling
93: If (X<=>F) (P89)
1: 113       X Loc [ diff      ]
2: 3          >=
3: 0          F
4: 30         Then Do

;Set thr_code = 2 for falling threshold
94: Z=F (P30)
1: 2          F
2: 00        Exponent of 10
3: 101       Z Loc [ thr_code  ]

;Set tmin = med_turb
95: Z=X (P31)
1: 77        X Loc [ med_turb  ]
2: 81        Z Loc [ tmin      ]

;End of case, [If diff >= 0]
96: End (P95)

;-----
;Call subroutine 5 to find threshold value
97: Do (P86)
1: 5          Call Subroutine 5

;End of case, [If thr_code = 3]
98: End (P95)

;-----
;End of case, [If interval <> 1]
99: End (P95)

;-----

```

```

;Call subroutine 3 to write data to datalogger memory
100: Do (P86)
  1: 3      Call Subroutine 3

;End of case, [If interval < 3]
101: End (P95)

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

;If [stage < minimum stage], then do
102: If (X<=>Y) (P88)
  1: 78      X Loc [ stage      ]
  2: 4        =
  3: 92      Y Loc [ min_stg   ]
  4: 30      Then Do

;Set thr_code = 0 for base flow conditions
103: Z=F (P30)
  1: 0        F
  2: 00       Exponent of 10
  3: 101     Z Loc [ thr_code  ]

;Call subroutine 3 to write data to datalogger memory
104: Do (P86)
  1: 3      Call Subroutine 3

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

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


;If [thr_code = 0], then previous interval was in base flow
106: If (X<=>F) (P89)
  1: 101     X Loc [ thr_code  ]
  2: 1        =
  3: 0        F
  4: 30      Then Do

;If [stage >= min_stg], then coming out of base flow
107: If (X<=>Y) (P88)
  1: 78      X Loc [ stage      ]
  2: 3        >=
  3: 92      Y Loc [ min_stg   ]
  4: 30      Then Do

;Call subroutine 4 for the 1st interval above minimum stage
108: Do (P86)
  1: 4      Call Subroutine 4

;End of case, [If stage >= min_stg]
109: End (P95)

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

```

```

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

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

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


;If [median turbidity >= threshold], then do
112: If (X<=>Y) (P88)
1: 77      X Loc [ med_turb   ]
2: 3          >=
3: 85      Y Loc [ thr       ]
4: 30         Then Do

;Increment threshold counter
113: Z=Z+1 (P32)
1: 84      Z Loc [ thr_count ]

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


;If [thr_count >= stay], number of intervals turbidity is above
;the threshold value
114: If (X<=>Y) (P88)
1: 84      X Loc [ thr_count ]
2: 3          >=
3: 96      Y Loc [ stay      ]
4: 30         Then Do

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

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

;-----


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

;-----


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

;If [med_mv >= mv_limit], then DO
119: If (X<=>Y) (P88)
1: 76      X Loc [ med_mv     ]

```

```

2: 3           >=
3: 190         Y Loc [ mv_limit   ]
4: 30          Then Do

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

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

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

;-----
;If [med_turb >= tmax], then do
123: If (X<=>Y) (P88)
1: 77         X Loc [ med_turb   ]
2: 3           >=
3: 82         Y Loc [ tmax       ]
4: 30          Then Do

;Set tmax = med_turb
124: Z=X (P31)
1: 77         X Loc [ med_turb   ]
2: 82         Z Loc [ tmax       ]

;Set reversal counter = 0
125: Z=F (P30)
1: 0           F
2: 00          Exponent of 10
3: 83          Z Loc [ rev_count  ]

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

;-----
;If [med_turb < old_turb], then do
127: If (X<=>Y) (P88)
1: 77         X Loc [ med_turb   ]
2: 4           <
3: 91         Y Loc [ old_turb   ]
4: 30          Then Do

;If [smp_code <> 5], then do
128: If (X<=>F) (P89)
1: 102        X Loc [ smp_code   ]
2: 2           <>
3: 5           F
4: 30          Then Do

;Increment reversal counter
129: Z=Z+1 (P32)
1: 83         Z Loc [ rev_count  ]

```

```

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

;Set rev_mult = tmax * revpct1
131: Z=X*Y (P36)
1: 82      X Loc [ tmax      ]
2: 88      Y Loc [ revpct1  ]
3: 100     Z Loc [ rev_mult  ]

;Find the difference between rev_val and rev_mult
132: Z=X-Y (P35)
1: 110     X Loc [ rev_val   ]
2: 100     Y Loc [ rev_mult   ]
3: 113     Z Loc [ diff       ]

;-----
;If [diff < 0], then rev_mult > rev_val
133: If (X<=>F) (P89)
1: 113     X Loc [ diff      ]
2: 4       <
3: 0       F
4: 30      Then Do

;Set rev_thr = tmax - rev_mult
134: Z=X-Y (P35)
1: 82      X Loc [ tmax      ]
2: 100     Y Loc [ rev_mult  ]
3: 90      Z Loc [ rev_thr   ]

;End of case, [If diff < 0]
135: End (P95)

;-----
;Else if [diff >= 0], then rev_mult =< rev_val
136: If (X<=>F) (P89)
1: 113     X Loc [ diff      ]
2: 3       >=
3: 0       F
4: 30      Then Do

;Set rev_thr = tmax - rev_val
137: Z=X-Y (P35)
1: 82      X Loc [ tmax      ]
2: 110     Y Loc [ rev_val   ]
3: 90      Z Loc [ rev_thr   ]

;End of case, [If diff >= 0]
138: End (P95)

;-----
;Find the difference between rev_thr and med_turb
139: Z=X-Y (P35)
1: 90      X Loc [ rev_thr   ]
2: 77      Y Loc [ med_turb   ]

```

```

3: 113      Z Loc [ diff      ]
;If [diff >= 0], then rev_thr >= med_turb]
140:  If (X<=>F) (P89)
1: 113      X Loc [ diff      ]
2: 3        >=
3: 0        F
4: 30       Then Do

;-----
;If [rev_count >= stay], then do
141:  If (X<=>Y) (P88)
1: 83      X Loc [ rev_count ]
2: 3        >=
3: 96      Y Loc [ stay      ]
4: 30       Then Do

;Set thr_code = 2 for falling threshold
142:  Z=F (P30)
1: 2        F
2: 00      Exponent of 10
3: 101     Z Loc [ thr_code  ]

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

;Set tmin = med_turb
144:  Z=X (P31)
1: 77      X Loc [ med_turb  ]
2: 81      Z Loc [ tmin      ]

;Set scratch1 = nextindex - 1
145:  Z=X+F (P34)
1: 137     X Loc [ nextindex ]
2: -1      F
3: 130     Z Loc [ SCRATCH1  ]

;Set scratch2 = 106, this is the location of fthr
146:  Z=F (P30)
1: 106     F
2: 00      Exponent of 10
3: 131     Z Loc [ SCRATCH2  ]

;Store old threshold in fthr
147:  Indirect Move (P61)
1: 130     Source Loc [ SCRATCH1  ]
2: 131     Destination Loc [ SCRATCH2  ]

;Find the difference between fthr and tmax
148:  Z=X-Y (P35)
1: 106     X Loc [ fthr      ]
2: 82      Y Loc [ tmax      ]
3: 113     Z Loc [ diff      ]

;-----

```

```

;If [diff < 0], then fthr < tmax
149: If (X<=>F) (P89)
1: 113      X Loc [ diff      ]
2: 4          <
3: 0          F
4: 30         Then Do

;-----
;If [fthr <> last_fthr], then do
150: If (X<=>Y) (P88)
1: 106      X Loc [ fthr      ]
2: 2          <>
3: 94      Y Loc [ last_fthr ]
4: 30         Then Do

;Call subroutine 79 for reversal threshold sample
151: Do (P86)
1: 79         Call Subroutine 79

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

;-----
;Set rev_inter = rev_wait + last_fsam
153: Z=X+Y (P33)
1: 98      X Loc [ last_fsam ]
2: 104      Y Loc [ rev_wait  ]
3: 105      Z Loc [ rev_inter ]

;Find the difference between rev_inter and interval
154: Z=X-Y (P35)
1: 105      X Loc [ rev_inter ]
2: 95      Y Loc [ interval   ]
3: 113      Z Loc [ diff      ]

;-----
;If [diff < 0], then rev_inter < interval
155: If (X<=>F) (P89)
1: 113      X Loc [ diff      ]
2: 4          <
3: 0          F
4: 30         Then Do

;Call subroutine 79 for reversal threshold sample
156: Do (P86)
1: 79         Call Subroutine 79

;End of case, [If rev_inter < interval]
157: End (P95)

```

```

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

;-----
;Set reversal counter = 0
159: Z=F (P30)
1: 0      F
2: 00     Exponent of 10
3: 83     Z Loc [ rev_count ]

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

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

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

;Call subroutine 3 to write data to datalogger memory
163: Do (P86)
1: 3      Call Subroutine 3

;-----
;End of case, [if thr_code = 1], thresholds rising
164: End (P95)

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

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

;=====
;Find the difference between thr and med_turb
166: Z=X-Y (P35)
1: 85      X Loc [ thr      ]
2: 77      Y Loc [ med_turb ]
3: 113    Z Loc [ diff      ]

;If [diff >= 0], then thr >= med_turb
167: If (X<=>F) (P89)
1: 113    X Loc [ diff      ]
2: 3      >=

```

```

3: 0          F
4: 30        Then Do

;Increment threshold counter
168: Z=Z+1 (P32)
1: 84        Z Loc [ thr_count ]

;-----
;If [thr_count = stay], then do
169: If (X<=>Y) (P88)
1: 84        X Loc [ thr_count ]
2: 1         =
3: 96        Y Loc [ stay      ]
4: 30        Then Do

;-----
;If [inbounds = 1], then still inside threshold array
170: If (X<=>F) (P89)
1: 103       X Loc [ inbounds  ]
2: 1         =
3: 1         F
4: 30        Then Do

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

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

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

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

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

;-----
;If [med_mv >= mv_limit], then DO
176: If (X<=>Y) (P88)
1: 76        X Loc [ med_mv     ]
2: 3         >=
3: 190       Y Loc [ mv_limit   ]
4: 30        Then Do

```

```

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

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

;-----
;Find the difference between tmin and med_turb
179: Z=X-Y (P35)
1: 81          X Loc [ tmin      ]
2: 77          Y Loc [ med_turb  ]
3: 113         Z Loc [ diff      ]

;-----
;If [diff >= 0], then tmin >= med_turb
180: If (X<=>F) (P89)
1: 113         X Loc [ diff      ]
2: 3           >=
3: 0           F
4: 30          Then Do

;Set tmin = med_turb
181: Z=X (P31)
1: 77          X Loc [ med_turb  ]
2: 81          Z Loc [ tmin      ]

;Set reversal counter = 0
182: Z=F (P30)
1: 0           F
2: 00          Exponent of 10
3: 83          Z Loc [ rev_count ]

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

;-----
;Find the difference between old_turb and med_turb
184: Z=X-Y (P35)
1: 91          X Loc [ old_turb  ]
2: 77          Y Loc [ med_turb  ]
3: 113         Z Loc [ diff      ]

;-----
;If [diff < 0], then old_turb < med_turb
185: If (X<=>F) (P89)
1: 113         X Loc [ diff      ]
2: 4           <
3: 0           F
4: 30          Then Do

;Increment reversal counter

```

```

186: Z=Z+1 (P32)
1: 83      Z Loc [ rev_count ]

;Set rev_mult = tmin * revpct2
187: Z=X*Y (P36)
1: 81      X Loc [ tmin      ]
2: 89      Y Loc [ revpct2   ]
3: 100     Z Loc [ rev_mult  ]

;Find difference between rev_val and rev_mult
188: Z=X-Y (P35)
1: 110     X Loc [ rev_val   ]
2: 100     Y Loc [ rev_mult   ]
3: 113     Z Loc [ diff      ]

;-----
;If [diff < 0 ], then rev_val < rev_mult
189: If (X<=>F) (P89)
1: 113     X Loc [ diff      ]
2: 4       <
3: 0       F
4: 30     Then Do

;Set rev_thr = tmin + rev_mult
190: Z=X+Y (P33)
1: 81      X Loc [ tmin      ]
2: 100     Y Loc [ rev_mult   ]
3: 90      Z Loc [ rev_thr   ]

;End of case, [If rev_val < rev_mult]
191: End (P95)

;-----
;If [diff >= 0] then rev_val >= rev_mult
192: If (X<=>F) (P89)
1: 113     X Loc [ diff      ]
2: 3       >=
3: 0       F
4: 30     Then Do

;Set rev_thr = tmin + rev_val
193: Z=X+Y (P33)
1: 81      X Loc [ tmin      ]
2: 110     Y Loc [ rev_val   ]
3: 90      Z Loc [ rev_thr   ]

;End of case, [if rev_val >= rev_mult]
194: End (P95)

;-----
;If [med_turb >= rev_thr], then do
195: If (X<=>Y) (P88)
1: 77      X Loc [ med_turb  ]
2: 3       >=

```

```

3: 90          Y Loc [ rev_thr    ]
4: 30          Then Do

;-----

;If [rev_count >= stay], then do
196: If (X<=>Y) (P88)
1: 83          X Loc [ rev_count  ]
2: 3           >=
3: 96          Y Loc [ stay       ]
4: 30          Then Do

;Set thr_code = 1 for rising threshold
197: Z=F (P30)
1: 1           F
2: 00          Exponent of 10
3: 101         Z Loc [ thr_code  ]

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

;Set tmax = med_turb
199: Z=X (P31)
1: 77          X Loc [ med_turb   ]
2: 82          Z Loc [ tmax      ]

;Set scratch1 = nextindex - 1
200: Z=X+F (P34)
1: 137         X Loc [ nextindex  ]
2: -1           F
3: 130         Z Loc [ SCRATCH1  ]

;Set scratch2 = 107, this is the location of rthr
201: Z=F (P30)
1: 107         F
2: 00          Exponent of 10
3: 131         Z Loc [ SCRATCH2  ]

;Move the last threshold in rthr
202: Indirect Move (P61)
1: 130         Source Loc [ SCRATCH1  ]
2: 131         Destination Loc [ SCRATCH2  ]

;-----

;If [tmin < rthr], then do
203: If (X<=>Y) (P88)
1: 81          X Loc [ tmin      ]
2: 4           <
3: 107         Y Loc [ rthr      ]
4: 30          Then Do

;-----

;If [rthr <> last_rthr], then do
204: If (X<=>Y) (P88)

```

```

1: 107      X Loc [ rthr      ]
2: 2          <>
3: 93      Y Loc [ last_rthr ]
4: 30      Then Do

;Call subroutine 79 for reversal threshold sample
205: Do (P86)
1: 79      Call Subroutine 79

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

;-----
;Set rev_inter = last_rsam + rev_wait
207: Z=X+Y (P33)
1: 97      X Loc [ last_rsam ]
2: 104      Y Loc [ rev_wait ]
3: 105      Z Loc [ rev_inter ]

;Find the difference between rev_inter and interval
208: Z=X-Y (P35)
1: 105      X Loc [ rev_inter ]
2: 95      Y Loc [ interval ]
3: 113      Z Loc [ diff      ]

;-----
;If [diff < 0], then rev_inter < interval
209: If (X<=>F) (P89)
1: 113      X Loc [ diff      ]
2: 4          <
3: 0          F
4: 30      Then Do

;Call subroutine 79 for reversal threshold sample
210: Do (P86)
1: 79      Call Subroutine 79

;End of case, [If rev_inter < interval]
211: End (P95)

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

;-----
;Set reversal counter = 0
213: Z=F (P30)
1: 0          F
2: 00      Exponent of 10
3: 83      Z Loc [ rev_count ]

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

```

```

;-----
;End of case, [If med_turb >= rev_thr
215: End (P95)

;-----

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

;Call subroutine 3 to write data to datalogger memory
217: Do (P86)
1: 3          Call Subroutine 3

;-----
;End of case, [If thr_code = 2], thresholds falling
218: End (P95)

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

;End of case, [If at the begining of an interval]
219: End (P95)

;=====
;=====
;=====
```

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

; Not Used

\*Table 3 Subroutines

```

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

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

;Increment counter k
2: Z=Z+1 (P32)
1: 65          Z Loc [ k           ]

;Set counter j = k
3: Z=X (P31)
1: 65          X Loc [ k           ]
2: 66          Z Loc [ j           ]
```

```

;Set insert flag = 0
4: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 68         Z Loc [insert    ]

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

;Set continue flag = 0
6: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 69         Z Loc [ continue  ]

;-----
;If [ j >= 2 ], then do
7: If (X<=>F) (P89)
1: 66         X Loc [ j          ]
2: 3          >=
3: 2          F
4: 30         Then Do

;-----
;If [insert flag = 0], then do
8: If (X<=>F) (P89)
1: 68         X Loc [ insert    ]
2: 1          =
3: 0          F
4: 30         Then Do

;Set jminus1 = j - 1
9: Z=X+F (P34)
1: 66         X Loc [ j          ]
2: -1         F
3: 67         Z Loc [ jminus1   ]

;Set LAST_LOC = 62, this is the location for last_mv
10: Z=F (P30)
1: 62         F
2: 00         Exponent of 10
3: 64         Z Loc [ LAST_LOC  ]

;Set last_mv = raw_mv(j-1)
11: Indirect Move (P61)
1: 67         Source Loc [ jminus1   ]
2: 64         Destination Loc [ LAST_LOC  ]

;-----

```

```

;If [raw_mv >= last_mv], then do
12: If (X<=>Y) (P88)
1: 61      X Loc [ raw_mv      ]
2: 3       >=
3: 62      Y Loc [ last_mv      ]
4: 30      Then Do

;Set insert flag = 1
13: Z=F (P30)
1: 1      F
2: 00     Exponent of 10
3: 68     Z Loc [ insert      ]

;End of case, [If raw_mv >= last_mv]
14: End (P95)

;-----
;If [raw_mv < last_mv], then do
15: If (X<=>Y) (P88)
1: 61      X Loc [ raw_mv      ]
2: 4       <
3: 62      Y Loc [ last_mv      ]
4: 30      Then Do

;Set raw_mv(j) = raw_mv(j-1)
16: Indirect Move (P61)
1: 67      Source Loc [ jminus1      ]
2: 66      Destination Loc [ j      ]

;Set j = j - 1
17: Z=X+F (P34)
1: 66      X Loc [ j      ]
2: -1      F
3: 66      Z Loc [ j      ]

;End of case, [If raw_mv < last_mv]
18: End (P95)

;-----
;Set continue flag = 1
19: Z=F (P30)
1: 1      F
2: 00     Exponent of 10
3: 69     Z Loc [ continue      ]

;End of case, [If insert flag = 0]
20: End (P95)

;-----
;End of case, [If j >= 2]
21: End (P95)

;-----

```

```

;If [continue flag = 0], then do
22: If (X<=>F) (P89)
1: 69      X Loc [ continue   ]
2: 1      =
3: 0      F
4: 31      Exit Loop if True

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

;-----
;Set TURB_LOC = 61
24: Z=F (P30)
1: 61      F
2: 00      Exponent of 10
3: 63      Z Loc [ TURB_LOC   ]

;Insert raw_mv(j) = raw_mv
25: Indirect Move (P61)
1: 63      Source Loc [ TURB_LOC   ]
2: 66      Destination Loc [ j           ]

;End of subroutine 1
26: End (P95)

;-----
;-----
;SUBROUTINE 2 find median turbidity (millivolts)
27: 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
28: Z=X MOD F (P46)
1: 65      X Loc [ k           ]
2: 2      F
3: 70      Z Loc [ rem         ]

;-----
;If [remainder <> 0], then k is odd
29: If (X<=>F) (P89)
1: 70      X Loc [ rem         ]
2: 2      <>
3: 0      F
4: 30      Then Do

;Set k = k + 1
30: Z=Z+1 (P32)
1: 65      Z Loc [ k           ]

;Divide k in half to find the location which holds the median value
31: Z=X*F (P37)
1: 65      X Loc [ k           ]

```

```

2: .5          F
3: 71          Z Loc [ source1    ]

;Add .5 to source1 to round correctly (this takes care of possible
;floating point error)
32: Z=X+F (P34)
1: 71          X Loc [ source1    ]
2: .5          F
3: 71          Z Loc [ source1    ]

;Take integer of source1
33: Z=INT(X) (P45)
1: 71          X Loc [ source1    ]
2: 71          Z Loc [ source1    ]

;Set scratch location for MED1
34: Z=F (P30)
1: 73          F
2: 00          Exponent of 10
3: 73          Z Loc [ MED1      ]

;Move median to temporary location addressed by MED1
35: Indirect Move (P61)
1: 71          Source Loc [ source1    ]
2: 73          Destination Loc [ MED1      ]

;Set med_mv = MED1
36: Z=X (P31)
1: 73          X Loc [ MED1      ]
2: 76          Z Loc [ med_mv     ]

;End of case, [If remainder <> 0]
37: End (P95)

;-----
;If [remainder = 0], then k is even
38: If (X<=>F) (P89)
1: 70          X Loc [ rem      ]
2: 1          =
3: 0          F
4: 30          Then Do

;Divide k by 2 to find the middle value
39: Z=X*F (P37)
1: 65          X Loc [ k      ]
2: .5          F
3: 71          Z Loc [ source1    ]

;Add .5 to source1 to round correctly (this takes care of possible
;floating point error)
40: Z=X+F (P34)
1: 71          X Loc [ source1    ]
2: .5          F
3: 71          Z Loc [ source1    ]

;Take the integer of source1

```

```

41: Z=INT(X) (P45)
1: 71      X Loc [ source1    ]
2: 71      Z Loc [ source1    ]

;Set source2 = source1 + 1
42: Z=X+F (P34)
1: 71      X Loc [ source1    ]
2: 1      F
3: 72      Z Loc [ source2    ]

;Set scratch location for MED1
43: Z=F (P30)
1: 73      F
2: 00      Exponent of 10
3: 73      Z Loc [ MED1      ]

;Set scratch location for MED2
44: Z=F (P30)
1: 74      F
2: 00      Exponent of 10
3: 74      Z Loc [ MED2      ]

;Move first turbidity value into temporary storage location 1
45: Indirect Move (P61)
1: 71      Source Loc [ source1    ]
2: 73      Destination Loc [ MED1      ]

;Move second turbidity value into temporary storage location 2
46: Indirect Move (P61)
1: 72      Source Loc [ source2    ]
2: 74      Destination Loc [ MED2      ]

;Add the two turbidity values together
47: Z=X+Y (P33)
1: 73      X Loc [ MED1      ]
2: 74      Y Loc [ MED2      ]
3: 75      Z Loc [ sum      ]

;Divide this sum by 2 to get the median value
48: Z=X*F (P37)
1: 75      X Loc [ sum      ]
2: .5      F
3: 76      Z Loc [ med_mv      ]

;End of case, [If remainder = 0]
49: End (P95)

;End of subroutine 2
50: End (P95)

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

;SUBROUTINE 3 write data to memory  
51: Beginning of Subroutine (P85)  
1: 3 Subroutine 3

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

;Site specific identifier (1 - 511)
53: Set Active Storage Area (P80)
 1: 01      Final Storage Area 1
 2: 0001    Array ID

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

;Writes out dump number
55: Sample (P70)
 1: 1       Reps
 2: 108    Loc [ dump_cnt   ]

;Writes out the number of bottles collected
56: Sample (P70)
 1: 1       Reps
 2: 86     Loc [ bottle     ]

;Writes out the threshold code
57: Sample (P70)
 1: 1       Reps
 2: 101    Loc [ thr_code   ]

;Writes out the sample code
58: Sample (P70)
 1: 1       Reps
 2: 102    Loc [ smp_code   ]

;Writes out the average stage
59: Sample (P70)
 1: 1       Reps ;
 2: 78     Loc [ stage     ]

;Writes out the median turbidity
60: Sample (P70)
 1: 1       Reps
 2: 77     Loc [ med_turb   ]

;If [Flag 5 is high], then write water temp
61: If Flag/Port (P91)
 1: 15      Do if Flag 5 is High
 2: 30      Then Do

;Writes out water temperature
62: Sample (P70)
 1: 1       Reps
 2: 123    Loc [ wtemp     ]

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

```

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

;Writes out air temperature
65: Sample (P70)
 1: 1      Reps
 2: 192     Loc [ atemp      ]

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

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

;Totals up the rainfall (inches)
68: Totalize (P72)
 1: 1      Reps
 2: 122     Loc [ rain      ]

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

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

;Set old_turb = med_turb for next wakeup
71: Z=X (P31)
 1: 77      X Loc [ med_turb  ]
 2: 91      Z Loc [ old_turb  ]

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

;End of subroutine 3
73: End (P95)

-----
-----
-----
```

**SUBROUTINE 4 first interval above minimum stage**

```

74: Beginning of Subroutine (P85)
 1: 4      Subroutine 4

;Set timer = interval - last_rsam
75: Z=X-Y (P35)
 1: 95      X Loc [ interval  ]
 2: 97      Y Loc [ last_rsam  ]
 3: 99      Z Loc [ timer      ]
```

```

;Set scratch1 = start1 + 1, location of the first rising threshold
76: Z=X+F (P34)
 1: 132      X Loc [ start1      ]
 2: 1         F
 3: 130      Z Loc [ SCRATCH1    ]

;Set scratch2 = 134, this is the location of frst_rthr
77: Z=F (P30)
 1: 134      F
 2: 00       Exponent of 10
 3: 131      Z Loc [ SCRATCH2    ]

;Place the first rising threshold into frst_rthr
78: Indirect Move (P61)
 1: 130      Source Loc [ SCRATCH1    ]
 2: 131      Destination Loc [ SCRATCH2    ]

;-----
;If [timer >= 18], then 3 hours have passed
79: If (X<=>F) (P89)
 1: 99      X Loc [ timer      ]
 2: 3       >=
 3: 18      F
 4: 30      Then Do

;-----
;If [med_turb >= frst_rthr], the turbidity is above the first rising
;threshold
80: If (X<=>Y) (P88)
 1: 77      X Loc [ med_turb    ]
 2: 3       >=
 3: 134     Y Loc [ frst_rthr  ]
 4: 30      Then Do

;-----
;If [smp_code = 0], then no threshold samples have been taken
81: If (X<=>F) (P89)
 1: 102     X Loc [ smp_code   ]
 2: 1       =
 3: 0       F
 4: 30      Then Do

;Set smp_code for threshold sampled
82: Z=F (P30)
 1: 4         F
 2: 00       Exponent of 10
 3: 102     Z Loc [ smp_code   ]

;Call subroutine 6 to collect ISCO sample
83: Do (P86)
 1: 6         Call Subroutine 6

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

```

```

;-----
;Set last_rthr = 0
85: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 93         Z Loc [ last_rthr ]

;Set last_rsam = interval
86: Z=X (P31)
1: 95         X Loc [ interval ]
2: 97         Z Loc [ last_rsam ]

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

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

;-----
;If [last_rsam = 0], then do
89: If (X<=>F) (P89)
1: 97         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
90: If (X<=>Y) (P88)
1: 77         X Loc [ med_turb ]
2: 3          =
3: 134        Y Loc [ frst_rthr ]
4: 30         Then Do

;-----
;If [smp_code = 0], then no threshold samples have been taken
91: If (X<=>F) (P89)
1: 102        X Loc [ smp_code ]
2: 1          =
3: 0          F
4: 30         Then Do

;Set smp_code for threshold sampled
92: Z=F (P30)
1: 4          F
2: 00         Exponent of 10
3: 102        Z Loc [ smp_code ]

;Call subroutine 6 to collect ISCO sample

```

```

93: Do (P86)
1: 6      Call Subroutine 6

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

;-----

;Set last_rthr = 0
95: Z=F (P30)
1: 0      F
2: 00     Exponent of 10
3: 93     Z Loc [ last_rthr ]

;Set last_rsam = interval
96: Z=X (P31)
1: 95     X Loc [ interval ]
2: 97     Z Loc [ last_rsam ]

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

;-----

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

;=====

;Set tmax = med_turb
99: Z=X (P31)
1: 77     X Loc [ med_turb ]
2: 82     Z Loc [ tmax ]

;Set thr_code = 1 for rising threshold
100: Z=F (P30)
1: 1      F
2: 00     Exponent of 10
3: 101    Z Loc [ thr_code ]

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

;Call subroutine 3 to write data to datalogger memory
102: Do (P86)
1: 3      Call Subroutine 3

;End of subroutine 4
103: End (P95)

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

;SUBROUTINE 5 determine rising/falling thresholds

```

```

104: Beginning of Subroutine (P85)
1: 5          Subroutine 5

;Set inbounds = 1
105: Z=F (P30)
1: 1          F
2: 00          Exponent of 10
3: 103         Z Loc [ inbounds  ]

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

;Set nextindex = start1, start of rising threshold array
107: Z=X (P31)
1: 132         X Loc [ start1    ]
2: 137         Z Loc [ nextindex  ]

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

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

;If [nextindex >= maxrindex], exit loop
110: If (X<=>Y) (P88)
1: 137         X Loc [ nextindex  ]
2: 3          >=
3: 135         Y Loc [ maxrindex  ]
4: 31         Exit Loop if True

;If [med_turb < next_thr], exit loop
111: If (X<=>Y) (P88)
1: 77          X Loc [ med_turb   ]
2: 4          =
3: 138          Y Loc [ next_thr   ]
4: 31          Exit Loop if True

;Increment nextindex
112: Z=Z+1 (P32)
1: 137         Z Loc [ nextindex  ]

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

;End of loop
114: End (P95)

```

```

;-----
;Set thr = next_thr
115: Z=X (P31)
1: 138      X Loc [ next_thr   ]
2: 85       Z Loc [ thr       ]

;End of case, [If thr_code = 1], thresholds rising
116: End (P95)
;-----

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

;Set nextindex = start2
118: Z=X (P31)
1: 133      X Loc [ start2    ]
2: 137      Z Loc [ nextindex  ]

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

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

;If [nextindex >= maxfindex], exit loop
121: If (X<=>Y) (P88)
1: 137      X Loc [ nextindex  ]
2: 3        >=
3: 136      Y Loc [ maxfindex ]
4: 31      Exit Loop if True

;Find difference between next_thr and med_turb
122: Z=X-Y (P35)
1: 138      X Loc [ next_thr   ]
2: 77       Y Loc [ med_turb   ]
3: 113      Z Loc [ diff       ]

;If [diff < 0], then med_turb > next_thr
123: If (X<=>F) (P89)
1: 113      X Loc [ diff       ]
2: 4        =
3: 0        F
4: 31      Exit Loop if True

```

```
;Increment nextindex  
124: Z=Z+1 (P32)  
1: 137      Z Loc [ nextindex ]
```

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

```
;End of loop
```

```
126: End (P95)
```

```
;Set thr = next_thr  
127: Z=X (P31)  
1: 138      X Loc [ next_thr ]  
2: 85       Z Loc [ thr ]
```

```
-----
```

```
;If [nextindex = maxfindex], then do  
128: If (X<=>Y) (P88)  
1: 137      X Loc [ nextindex ]  
2: 1         =  
3: 136      Y Loc [ maxfindex ]  
4: 30       Then Do
```

```
;Set inbounds = 0  
129: Z=F (P30)  
1: 0         F  
2: 00        Exponent of 10  
3: 103      Z Loc [ inbounds ]
```

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

```
-----
```

```
;End of case, [If thr_code = 2], thresholds falling  
131: End (P95)
```

```
-----
```

```
;Set threshold counter = 0  
132: Z=F (P30)  
1: 0.0      F  
2: 00       Exponent of 10  
3: 84       Z Loc [ thr_count ]
```

```
;End of subroutine 5  
133: End (P95)
```

```
-----
```

```
-----
```

```
-----
```

```
;SUBROUTINE 6 collect and ISCO sample  
134: Beginning of Subroutine (P85)
```

```

1: 6          Subroutine 6

;Increment bottle counter
135: Z=Z+1 (P32)
1: 112      Z Loc [ bot_cnt    ]

;Increment next ISCO counter
136: Z=Z+1 (P32)
1: 111      Z Loc [ nxt_bot    ]

;Set bottle = bot_cnt (bottle numbers are written out
;only when they are collected, otherwise they are recorded as 0)
137: Z=X (P31)
1: 112      X Loc [ bot_cnt    ]
2: 86       Z Loc [ bottle     ]

;If [bottle < 25], then do
138: If (X<=>F) (P89)
1: 86       X Loc [ bottle     ]
2: 4        =
3: 25       F
4: 30       Then Do

;If [isco_ex = 0], then excite E2 (CR510, CR500, CR10)
139: If (X<=>F) (P89)
1: 189      X Loc [ isco_ex    ]
2: 1        =
3: 0        F
4: 30       Then Do

;Pulse E2 to trigger ISCO
140: Excite-Delay (SE) (P4)
1: 1        Reps
2: 5        2500 mV Slow Range
3: 4        SE Channel ;           <-- no connection
4: 2        Excite all reps w/Exchan 2
5: 10       Delay (units 0.01 sec)
6: 2500     mV Excitation
7: 188      Loc [ dummy       ]
8: 1.0      Mult
9: 0.0      Offset

;End case [If isco_ex = 0]
141: End (P95)

;If [isco_ex = 1], then excite C2 (CR10X)
142: If (X<=>F) (P89)
1: 189      X Loc [ isco_ex    ]
2: 1        =
3: 1        F
4: 30       Then Do

;Pulse Port 2 to trigger ISCO
143: Pulse Port w/Duration (P21)
1: 2        Port
2: 87      Pulse Length Loc [ delay     ]

```



```

158: Z=F (P30) ; isco excite <===== Set to 0 for CR510,
1: 0 F ; CR500, and CR10
2: 00 Exponent of 10 ;
3: 189 Z Loc [ isco_ex ] ;

159: Z=F (P30) ; <===== Calculate discharge
1: 0 F ; 0 --> not active
2: 00 Exponent of 10 ;
3: 193 Z Loc [ qcalc ] ; 1 --> active
Set next 2 instruct.

160: Z=F (P30) ; <===== Enter multiplier for
1: 68.349 F ; discharge calc.
2: 00 Exponent of 10 ;
3: 127 Z Loc [ q_mult ] ; (F * stg**exp)

161: Z=F (P30) ; <===== Enter exponent for
1: 1.677 F ; discharge calc.
2: 00 Exponent of 10 ;
3: 128 Z Loc [ q_exp ] ; (raise stg to power
held in F)

162: Z=F (P30) <===== Enter value for the
1: 0.1 F ; reversal % from rising
2: 00 Exponent of 10 ;
3: 88 Z Loc [ revpctl ] ; to falling thresholds

163: Z=F (P30) <===== Enter value for the
1: 0.2 F ; reversal % from
2: 00 Exponent of 10 ;
3: 89 Z Loc [ revpct2 ] ; falling to rising

164: Z=F (P30) <===== Enter # of intervals
1: 2 F ; before a reversal
2: 00 Exponent of 10 ;
3: 96 Z Loc [ stay ] ; or new threshold

165: Z=F (P30) <===== Enter # of intervals
1: 5 F ; before isco is taken
2: 00 Exponent of 10 ;
3: 104 Z Loc [ rev_wait ] ; within 2 similar
; threshold values

166: Z=F (P30) <===== Enter delay (0.01s)
1: 8 F ; that port #2 is high
2: 00 Exponent of 10 ;
3: 87 Z Loc [ delay ] ; to trigger a pumped
; sample

167: Z=F (P30) ; <===== Enter the value to
1: 5 F ; collect a sample when
2: 00 Exponent of 10 ;
3: 110 Z Loc [ rev_val ] ; > (tmax or tmin)*
reversal percentage

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

```

```

3: 190      Z Loc [ mv_limit ] ;           at fixed time
;                                         intervals (lim_skip)

169: Z=F (P30) ;             ===== Number of intervals
1: 1          F ;           skipped (not sampled)
2: 00         Exponent of 10 ;
3: 191         Z Loc [ lim_skip ] ;       between samples, when
                                         mV limit exceeded

;Initialize the dump counter = 1
170: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 108        Z Loc [ dump_cnt ] 

;Set interval = 0
171: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 95          Z Loc [ interval ] 

;Set START1 = 140, this is the location where the rising threshold
;array begins
172: Z=F (P30)
1: 140        F
2: 00         Exponent of 10
3: 132        Z Loc [ start1 ] 

;Set START2 = 160, this is the location where the falling threshold
;array begins
173: Z=F (P30)
1: 160        F
2: 00         Exponent of 10
3: 133        Z Loc [ start2 ] 

;Set maxrindex = START1, plus the number of rising thresholds
174: Z=X+F (P34)
1: 132        X Loc [ start1 ] 
2: 11          F
3: 135        Z Loc [ maxrindex ] 

;Set maxfindex = START2, plus the number of falling thresholds
175: Z=X+F (P34)
1: 133        X Loc [ start2 ] 
2: 17          F
3: 136        Z Loc [ maxfindex ] 

;Set bottle counter to 0
176: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 112        Z Loc [ bot_cnt ] 

;Set next ISCO counter to 1
177: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 111        Z Loc [ nxt_bot ] 

```

```

;Set reversal counter = 0
178: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 83         Z Loc [ rev_count ]

;Set threshold counter to 0
179: Z=F (P30)
1: 0.0        F
2: 00         Exponent of 10
3: 84         Z Loc [ thr_count ]

;Set inbounds = 1
180: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 103        Z Loc [ inbounds ]

;Set last_rsam = 0
181: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 97         Z Loc [ last_rsam ]

;Set last_fsam = 0
182: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 98         Z Loc [ last_fsam ]

;Set last_rthr = 0
183: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 93         Z Loc [ last_rthr ]

;Set last_fthr = 0
184: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 94         Z Loc [ last_fthr ]

;-----
;Enter in the Rising threshold values, maximum of 16 values
185: 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: 140        Loc [ up_1      ]

```

```
186: 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: 148       Loc [ up_9      ]
```

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

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

```
187: 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: 160       Loc [ dn_1      ]
```

```
188: 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: 168       Loc [ dn_9      ]
```

```
189: Bulk Load (P65)
1: 30         F
2: 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: 176       Loc [ dn_17     ]
```

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

```
;End of subroutine 7
```

```
190: End (P95)
```

```
;=====
```

```
;=====
```

```
;=====
```

```

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

;Set scratch1 = nextindex, this is the threshold location
192: Z=X (P31)
1: 137      X Loc [ nextindex ]
2: 130      Z Loc [ SCRATCH1  ]

;Set scratch2 = 138, this is the location of next_thr
193: Z=F (P30)
1: 138      F
2: 00       Exponent of 10
3: 131      Z Loc [ SCRATCH2  ]

;Place threshold into next_thr
194: Indirect Move (P61)
1: 130      Source Loc [ SCRATCH1  ]
2: 131      Destination Loc [ SCRATCH2  ]

;End of subroutine 8
195: End (P95)

```

```

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

;SUBROUTINE 9 control sampling above turbidity mv\_limit

```

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

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

```

```

;Set thr_code = 1 (rising)
198: Z=F (P30)
1: 1          F
2: 00        Exponent of 10
3: 101      Z Loc [ thr_code  ]

```

```

;Set tmax = med_turb
199: Z=X (P31)
1: 77      X Loc [ med_turb  ]
2: 82      Z Loc [ tmax      ]

```

```

;Set rev_count = 0
200: Z=F (P30)
1: 0          F
2: 00        Exponent of 10
3: 83      Z Loc [ rev_count ]

```

;Call subroutine 5 to determine max threshold

```

201: Do (P86)
1: 5          Call Subroutine 5

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

;Find difference between interval and last_rsam
203: Z=X-Y (P35)
1: 97          X Loc [ last_rsam ]
2: 95          Y Loc [ interval   ]
3: 113         Z Loc [ diff       ]

;Add lim_skip to diff
204: Z=X+Y (P33)
1: 113         X Loc [ diff       ]
2: 191         Y Loc [ lim_skip   ]
3: 113         Z Loc [ diff       ]

;If [diff < 0], then interval > last_rsam + lim_skip]
205: If (X<=>F) (P89)
1: 113         X Loc [ diff       ]
2: 4            <
3: 0            F
4: 30           Then Do

;Find difference between interval and last_fsam
206: Z=X-Y (P35)
1: 98          X Loc [ last_fsam ]
2: 95          Y Loc [ interval   ]
3: 113         Z Loc [ diff       ]

;Add lim_skip to diff
207: Z=X+Y (P33)
1: 113         X Loc [ diff       ]
2: 191         Y Loc [ lim_skip   ]
3: 113         Z Loc [ diff       ]

;If [diff < 0], then interval > last_fsam + lim_skip]
208: If (X<=>F) (P89)
1: 113         X Loc [ diff       ]
2: 4            <
3: 0            F
4: 30           Then Do

;Set smp_code = 5 (overflow sample)
209: Z=F (P30)
1: 5            F
2: 00           Exponent of 10
3: 102          Z Loc [ smp_code  ]

;Call subroutine 6 to collect ISCO sample
210: Do (P86)
1: 6          Call Subroutine 6

;Set last_rsam = interval
211: Z=X (P31)
1: 95          X Loc [ interval   ]

```

```

2: 97          Z Loc [ last_rsam ]

;Set rev_count = 0
212: Z=F (P30)
1: 0          F
2: 00         Exponent of 10
3: 83         Z Loc [ rev_count ]

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

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

;End of subroutine 9
215: End (P95)

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

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

;If [smp_code = 0], then no ISCO samples have been taken
217: If (X<=>F) (P89)
1: 102         X Loc [ smp_code ]
2: 1          =
3: 0          F
4: 30         Then Do

;Set smp_code = 1 for threshold sampled
218: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 102         Z Loc [ smp_code ]

;Call subroutine 6 to take ISCO sample
219: Do (P86)
1: 6          Call Subroutine 6

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

;If [smp_code = 3], then AUX was already selected
221: If (X<=>F) (P89)
1: 102         X Loc [ smp_code ]
2: 1          =
3: 3          F
4: 30         Then Do

;Set sample code = 1 (override sample code 3)
222: Z=F (P30)
1: 1          F
2: 00         Exponent of 10
3: 102         Z Loc [ smp_code ]

```

```

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

;If [thr_code = 1], condition is rising
224: If (X<=>F) (P89)
1: 101      X Loc [ thr_code   ]
2: 1         =
3: 1         F
4: 30        Then Do

;Set last_rsam = interval
225: Z=X (P31)
1: 95      X Loc [ interval   ]
2: 97      Z Loc [ last_rsam ]

;Set last_rthr = thr
226: Z=X (P31)
1: 85      X Loc [ thr       ]
2: 93      Z Loc [ last_rthr ]

;End of case [If thr_code = 1]
227: End (P95)

;If [thr_code = 2], condition is falling
228: If (X<=>F) (P89)
1: 101      X Loc [ thr_code   ]
2: 1         =
3: 2         F
4: 30        Then Do

;Set last_fsam = interval
229: Z=X (P31)
1: 95      X Loc [ interval   ]
2: 98      Z Loc [ last_fsam ]

;Set last_fthr = thr
230: Z=X (P31)
1: 85      X Loc [ thr       ]
2: 94      Z Loc [ last_fthr ]

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

;End of subroutine 79
232: End (P95)

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

End Program

-Input Locations-

1	_____	1	0	0
2	_____	1	0	0
3	_____	1	0	0

4	_____	1	0	0
5	_____	1	0	0
6	_____	1	0	0
7	_____	1	0	0
8	_____	1	0	0
9	_____	1	0	0
10	_____	1	0	0
11	_____	1	0	0
12	_____	1	0	0
13	_____	1	0	0
14	_____	1	0	0
15	_____	1	0	0
16	_____	1	0	0
17	_____	1	0	0
18	_____	1	0	0
19	_____	1	0	0
20	_____	1	0	0
21	_____	1	0	0
22	_____	1	0	0
23	_____	1	0	0
24	_____	1	0	0
25	_____	1	0	0
26	_____	1	0	0
27	_____	1	0	0
28	_____	1	0	0
61	raw_mv	1	2	1
62	last_mv	1	2	0
63	TURB_LOC	1	1	1
64	LAST_LOC	1	0	2
65	k	1	4	3
66	j	1	3	4
67	jminus1	1	2	1
68	insert	1	1	2
69	continue	1	1	2
70	rem	1	2	1
71	source1	1	7	6
72	source2	1	1	1
73	MED1	1	2	4
74	MED2	1	1	2
75	sum	1	1	1
76	med_mv	1	3	2
77	med_turb	1	27	4
78	stage	1	8	3
79	tot_stg	1	2	2
80	avg_stg	1	3	1
81	tmin	1	5	3
82	tmax	1	5	5
83	rev_count	1	2	9
84	thr_count	1	2	4
85	thr	1	4	3
86	bottle	1	2	2
87	delay	1	1	1
88	revpct1	1	1	1
89	revpct2	1	1	1
90	rev_thr	1	2	4
91	old_turb	1	3	1
92	min_stg	1	6	1

93 last\_rthr 1 1 4  
94 last\_fthr 1 1 2  
95 interval 1 13 2  
96 stay 1 4 1  
97 last\_rsam 1 4 5  
98 last\_fsam 1 2 2  
99 timer 1 1 1  
100 rev\_mult 1 4 2  
101 thr\_code 1 12 10  
102 smp\_code 1 6 8  
103 inbounds 1 1 3  
104 rev\_wait 1 2 1  
105 rev\_inter 1 2 2  
106 fthr 1 2 1  
107 rthr 1 2 0  
108 dump\_cnt 1 1 2  
109 prob\_time 1 1 2  
110 rev\_val 1 4 1  
111 nxt\_bot 1 0 3  
112 bot\_cnt 1 1 3  
113 diff 1 18 15  
114 turb\_mult 1 1 1  
115 turb\_off 1 1 1  
116 stg\_mult 1 1 1  
117 stg\_off 1 1 1  
118 \_\_\_\_\_ 1 1 1  
119 \_\_\_\_\_ 1 1 1  
120 \_\_\_\_\_ 1 2 0  
121 temp\_off 1 2 0  
122 rain 1 5 3  
123 wtemp 1 3 3  
124 bat\_volt 1 0 1  
125 raw\_stg 1 1 1  
126 n 1 1 2  
127 q\_mult 1 1 1  
128 q\_exp 1 1 1  
129 q\_prod 1 1 1  
130 SCRATCH1 1 4 4  
131 SCRATCH2 1 0 8  
132 start1 1 3 1  
133 start2 1 2 1  
134 frst\_rthr 1 2 0  
135 maxrindex 1 1 1  
136 maxfindex 1 2 1  
137 nextindex 1 6 4  
138 next\_thr 1 4 0  
139 disch 1 0 1  
140 up\_1 5 0 1  
141 up\_2 9 0 1  
142 up\_3 9 0 1  
143 up\_4 9 0 1  
144 up\_5 9 0 1  
145 up\_6 9 0 1  
146 up\_7 9 0 1  
147 up\_8 17 0 1  
148 up\_9 5 0 1  
149 up\_10 9 0 1

150	up_11	9	0	1
151	up_12	9	0	1
152	up_13	9	0	1
153	up_14	9	0	1
154	up_15	9	0	1
155	up_16	17	0	1
156	_____	0	0	0
157	_____	0	0	0
158	_____	0	0	0
159	_____	0	0	0
160	dn_1	5	2	1
161	dn_2	9	0	1
162	dn_3	9	0	1
163	dn_4	9	0	1
164	dn_5	9	0	1
165	dn_6	9	0	1
166	dn_7	9	0	1
167	dn_8	17	0	1
168	dn_9	5	0	1
169	dn_10	9	0	1
170	dn_11	9	0	1
171	dn_12	9	0	1
172	dn_13	9	0	1
173	dn_14	9	0	1
174	dn_15	9	0	1
175	dn_16	17	0	1
176	dn_17	5	0	1
177	dn_18	9	0	1
178	dn_19	9	0	1
179	dn_20	9	0	1
180	dn_21	9	0	1
181	dn_22	9	0	1
182	dn_23	9	0	1
183	dn_24	17	0	1
184	_____	0	0	0
185	_____	0	0	0
186	daily	1	1	2
187	season	1	1	1
188	dummy	1	0	1
189	isco_ex	1	2	1
190	mv_limit	1	2	1
191	lim_skip	1	2	1
192	atemp	1	3	3
193	qcalc	1	1	1
194	_____	0	0	0
195	_____	0	0	0
196	_____	0	0	0
197	_____	0	0	0
198	_____	0	0	0
199	_____	0	0	0
200	_____	0	0	0
201	_____	0	0	0
202	_____	0	0	0
203	_____	0	0	0
204	_____	0	0	0
205	_____	0	0	0
206	_____	0	0	0

207	_____	0	0	0
208	_____	0	0	0
209	_____	0	0	0
210	_____	0	0	0
211	_____	0	0	0
212	_____	0	0	0
213	_____	0	0	0
214	_____	0	0	0
215	_____	0	0	0
216	_____	0	0	0
217	_____	0	0	0
218	_____	0	0	0
219	_____	0	0	0
220	_____	0	0	0
221	_____	0	0	0
222	_____	0	0	0
223	_____	0	0	0
224	_____	0	0	0
225	_____	0	0	0
226	_____	0	0	0
227	_____	0	0	0
228	_____	0	0	0
229	_____	0	0	0
230	_____	0	0	0
231	_____	0	0	0
232	_____	0	0	0
233	_____	0	0	0
234	_____	0	0	0
235	_____	0	0	0
236	_____	0	0	0
237	_____	0	0	0
238	_____	0	0	0
239	_____	0	0	0
240	_____	0	0	0
241	_____	0	0	0
242	_____	0	0	0
243	_____	0	0	0
244	_____	0	0	0
245	_____	0	0	0
246	_____	0	0	0
247	_____	0	0	0
248	_____	0	0	0
249	_____	0	0	0
250	_____	0	0	0
251	_____	0	0	0
252	_____	0	0	0
253	_____	0	0	0
254	_____	0	0	0
255	_____	0	0	0
256	_____	0	0	0
257	_____	0	0	0
258	_____	0	0	0
259	_____	0	0	0
260	_____	0	0	0
261	_____	0	0	0
262	_____	0	0	0
263	_____	0	0	0

264	_____	0	0	0
265	_____	0	0	0
266	_____	0	0	0
267	_____	0	0	0
268	_____	0	0	0
269	_____	0	0	0
270	_____	0	0	0
271	_____	0	0	0
272	_____	0	0	0
273	_____	0	0	0
274	_____	0	0	0
275	_____	0	0	0
276	_____	0	0	0
277	_____	0	0	0
278	_____	0	0	0
279	_____	0	0	0
280	_____	0	0	0
281	_____	0	0	0
282	_____	0	0	0
283	_____	0	0	0
284	_____	0	0	0
285	_____	0	0	0
286	_____	0	0	0
287	_____	0	0	0
288	_____	0	0	0
289	_____	0	0	0
290	_____	0	0	0
291	_____	0	0	0
292	_____	0	0	0
293	_____	0	0	0
294	_____	0	0	0
295	_____	0	0	0
296	_____	0	0	0
297	_____	0	0	0
298	_____	0	0	0
299	_____	0	0	0
300	_____	0	0	0
301	_____	0	0	0
302	_____	0	0	0
303	_____	0	0	0
304	_____	0	0	0
305	_____	0	0	0
306	_____	0	0	0
307	_____	0	0	0
308	_____	0	0	0
309	_____	1	0	0
310	_____	0	0	0
311	_____	0	0	0
312	_____	0	0	0
313	_____	0	0	0
314	_____	0	0	0
315	_____	0	0	0
316	_____	0	0	0
317	_____	0	0	0
318	_____	0	0	0
319	_____	0	0	0
320	_____	0	0	0

321	_____	0	0	0
322	_____	0	0	0
323	_____	0	0	0
324	_____	0	0	0
325	_____	0	0	0
326	_____	0	0	0
327	_____	0	0	0
328	_____	0	0	0
329	_____	0	0	0
330	_____	0	0	0
331	_____	0	0	0
332	_____	0	0	0
333	_____	0	0	0
334	_____	0	0	0
335	_____	0	0	0
336	_____	0	0	0
337	_____	0	0	0
338	_____	0	0	0
339	_____	0	0	0
340	_____	0	0	0
341	_____	0	0	0
342	_____	0	0	0
343	_____	0	0	0
344	_____	0	0	0
345	_____	0	0	0
346	_____	0	0	0
347	_____	0	0	0
348	_____	0	0	0
349	_____	0	0	0
350	_____	0	0	0
351	_____	0	0	0
352	_____	0	0	0
353	_____	0	0	0
354	_____	0	0	0
355	_____	0	0	0
356	_____	0	0	0
357	_____	0	0	0
358	_____	0	0	0
359	_____	0	0	0
360	_____	0	0	0
361	_____	0	0	0
362	_____	0	0	0
363	_____	0	0	0
364	_____	0	0	0
365	_____	0	0	0
366	_____	0	0	0
367	_____	0	0	0
368	_____	0	0	0
369	_____	0	0	0
370	_____	0	0	0
371	_____	0	0	0
372	_____	0	0	0
373	_____	0	0	0
374	_____	0	0	0
375	_____	0	0	0
376	_____	0	0	0
377	_____	0	0	0

378 \_\_\_\_\_ 0 0 0  
379 \_\_\_\_\_ 0 0 0  
380 \_\_\_\_\_ 0 0 0  
381 \_\_\_\_\_ 0 0 0  
382 \_\_\_\_\_ 0 0 0  
383 \_\_\_\_\_ 0 0 0  
384 \_\_\_\_\_ 0 0 0  
385 \_\_\_\_\_ 0 0 0  
386 \_\_\_\_\_ 0 0 0  
387 \_\_\_\_\_ 0 0 0  
388 \_\_\_\_\_ 0 0 0  
389 \_\_\_\_\_ 0 0 0  
390 \_\_\_\_\_ 0 0 0  
391 \_\_\_\_\_ 0 0 0  
392 \_\_\_\_\_ 0 0 0  
393 \_\_\_\_\_ 0 0 0  
394 \_\_\_\_\_ 0 0 0  
395 \_\_\_\_\_ 0 0 0  
396 \_\_\_\_\_ 0 0 0  
397 \_\_\_\_\_ 0 0 0  
398 \_\_\_\_\_ 0 0 0  
399 \_\_\_\_\_ 0 0 0  
400 \_\_\_\_\_ 0 0 0  
401 \_\_\_\_\_ 0 0 0  
402 \_\_\_\_\_ 0 0 0  
403 \_\_\_\_\_ 0 0 0  
404 \_\_\_\_\_ 0 0 0  
405 \_\_\_\_\_ 0 0 0  
406 \_\_\_\_\_ 0 0 0  
407 \_\_\_\_\_ 0 0 0  
408 \_\_\_\_\_ 0 0 0  
409 \_\_\_\_\_ 0 0 0  
410 \_\_\_\_\_ 0 0 0  
411 \_\_\_\_\_ 0 0 0  
412 \_\_\_\_\_ 0 0 0  
413 \_\_\_\_\_ 0 0 0  
414 \_\_\_\_\_ 0 0 0  
415 \_\_\_\_\_ 0 0 0  
416 CSI\_R 0 0 0  
417 OF 0 0 0  
418 END 0 0 0  
419 CSI\_1 0 0 0  
420 \_\_\_\_\_ 1 0 0  
421 \_\_\_\_\_ 0 0 0  
422 \_\_\_\_\_ 0 0 0

-Program Security-

0000  
0000  
0000

-Mode 4-

-Final Storage Area 2-

0

-CR10X ID-

0

-CR10X Power Up-

3