

Standard Operating Procedure (SOP) 6: Datalogger Programming, Version 2.01

Weather and Climate Monitoring Protocol

Arctic (ARCN), Central Alaska (CAKN), and Southeast Alaska (SEAN) Inventory and Monitoring Networks

Change History

Prev. Version #	Revision Date	Revised By	Changes	Justification	New Version #
1.00	12/19/2006	P. Sousanes	Added reference to testing prior to field deployment	Incorporated peer reviewed comments	1.01
1.01	March 2017	P. Sousanes K. Hill, M. Bower, B. Johnson	Updated instructions for programming in CRBasic. Minor changes in program from original.	Datalogger and software updates. Data management deliverable schedule enhanced.	2.0
2.0	February 2024	P. Sousanes	Added reference to new CR1000X datalogger	New datalogger model available	2.01

Please cite this Standard Operating Procedure as:

Sousanes, P., K. Hill, M. Bower, W. Johnson, and S. Miller. 2024. Standard Operating Procedure (SOP) 6: Datalogger Programming, Version 2.01. Weather and Climate Monitoring Protocol, Arctic (ARCN), Central Alaska (CAKN), and Southeast Alaska (SEAN) Inventory and Monitoring Networks. National Park Service, Fairbanks, Alaska. Available online at <https://irma.nps.gov/DataStore/Reference/Profile/2303371>.

Overview

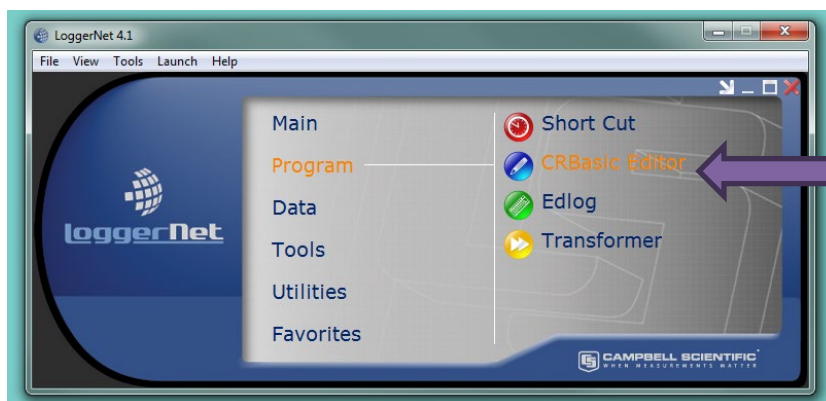
This protocol describes the process for programming the weather station datalogger. The datalogger scans a variety of different measurements, performs programmed calculations, converts data, transmits data packages, and stores data in memory. This SOP describes the programming for a CR1000 and CR1000X datalogger using CRBasic, the program language included in the LoggerNet software package from Campbell Scientific, Inc. Additionally, a wiring diagram consistent with the current measurements and programming may be generated using this software. It is recommended

that staff working on the weather stations periodically attend LoggerNet and CRBasic training offered at Campbell Scientific in Logan, Utah.

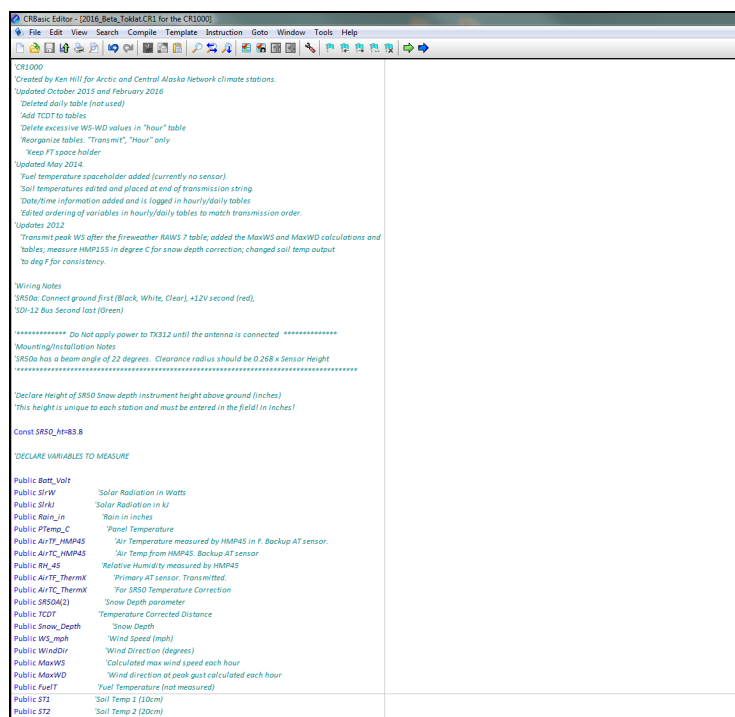
The NPS weather station dataloggers are programmed to measure most sensors every 60 seconds, store the values, and compute the hourly summary statistic (mean, total, minimum, etc.) Snow depth and relative humidity are measured and stored once each hour, on the hour. Summary statistics or sampled values are output to the satellite transmitter at 5 minutes past every hour. A standard order of measurements are output via satellite for processing at the National Interagency Fire Center in Boise. The data stream order is 1) cumulative precipitation, 2) mean wind speed, 3) mean wind direction, 4) mean air temperature, 5) relative humidity, 6) mean battery voltage, 7) peak wind direction, 8) peak wind speed, 9) mean solar radiation, 10) sample snow depth, 11) minimum air temperature, 12) maximum air temperature, 13) mean soil temperature 1 (10 cm depth), 14) mean soil temperature 2 (20 cm depth), and 15) mean soil temperature 3 (50 cm depth). All measurements are processed and stored on an hourly basis in a table that is downloaded during each annual maintenance visit.

Procedures

To start programming the CR1000 or CR1000X datalogger open LoggerNet and click the CRBasic button.



The software will open the last program you were working with.



To create a new program click on File|New. To open a program click on File|Open and navigate to the correct file. The current programs are located on network drives and are also stored on USB sticks that are used in the field. The nomenclature for naming program versions should include location and year (e.g. Asik_2015.CR1).

Before creating or modifying a program, learn the fundamentals of CRBasic. There are manuals and video tutorials available on [LoggerNet](https://www.campbellsci.com/loggernet) on the CSI website (<https://www.campbellsci.com/loggernet>). The Short Cut button from the LoggerNet tool bar will generate a simple program based on sensors you pick from a list. Using Short Cut is the best starting point to learn CRBasic programming.

The current datalogger program being used has four main steps:

1. Declare variables to measure,
2. Define data tables to be transmitted and downloaded,
3. Instructing sensors to take measurements (the main program),
4. Transmitting data to the GOES satellite.

CRBasic uses colors to differentiate types of instructions. **Operators** are displayed in blue text by default and include the declaration of public variables, measurement instructions, etc. **Normal text** is displayed in black. **Comments** are shown in green and can be designated anywhere in the program as long as the apostrophe notation (') precedes the comment. The current programs contain

comments at each step for explanation. Instructions are added from the frame on the right-hand side of the screen. Instruction information is entered within the main CRBasic text or by right-clicking on instructions within the program. Help is available by right-clicking instructions and pressing **Help** or by highlighting text on the right frame and clicking **Help**. The help files are extremely useful when creating or modifying a program.

Declare Variables to Measure

The snow depth measurement is calculated based on the known distance to the ground from the bottom of the sensor. This variable is unique to each site and must be entered into the program in the field. The height is measured in inches and entered as a constant near the beginning of the program as the **SR50_ht** variable (example: `Const SR50_ht=112`). If a new program is being sent to the datalogger measure the distance, change the number in the program and then send it to the datalogger (SOP 9).

In CRBasic create a list of public variables to be measured, stored, transmitted, and/or made available to view when connected to the datalogger. These include battery voltage, solar radiation, rain, temperature, relative humidity, snow depth, wind speed, wind direction, soil temperature, etc. Parameters to identify the status, location, and time of the transmitter are also declared at this stage. See the comments in the sample program for details.

Define Data tables to be Transmitted and Downloaded

There are two tables: transmit and hourly. The transmit table includes the cumulative rain total, hourly average wind speed, hourly average wind direction, hourly average air temperature, relative humidity sampled on the hour, hourly average battery voltage, hourly peak wind speed, hourly peak wind direction, hourly average solar radiation, snow depth sampled on the hour, hourly average minimum air temperature, hourly average maximum air temperature, and hourly average soil temperatures at 10 cm, 20 cm, and 50 cm (and 100 cm at some sites).

In addition to all transmitted data, the hourly table stores a time stamp, as well as temperature and relative humidity values from the air temperature / relative humidity sensors for redundancy and the hourly rainfall totals (as opposed to the cumulative total that is transmitted). The hourly tables are downloaded from the datalogger to the laptop during each annual maintenance visit (SOP 9). These are the basis for the WC_F raw data product defined in SOP 10.

Main Program and Scanning Interval

Begin the main program and define a scan interval of 60 seconds. All sensors take a measurement every minute except the relative humidity sensor and the snow depth sensor. Snow depth is only measured once each hour to avoid erroneous data caused by outliers. Additionally, measuring the snow depth only once each hour reduces the power draw of the station. Relative humidity is sampled once per hour as recommended by the sensor manual due to the lag in response to temperature changes.

Use CRBasic's instrument-specific instructions to make sensor measurements. Use the sensor manuals to determine which instructions are appropriate. Short Cut can also be used to create a

preliminary program as it contains lists of common sensors which can be added to a program. Instructions often involve some preliminary processing and calculations. For example, it is common to eliminate negative solar radiation values and relative humidity values above 100%. This can be accomplished by using processing instructions. Data variables are also maintained in the main program to make available transmitter properties for potential troubleshooting (GOESStatus, GOESGPS).

Transmitting Data to the GOES Satellite

Call the **transmit** table and send hourly values to the satellite transmitter that is connected to the datalogger. The transmitter is programmed to send the data package to the GOES satellite during a unique transmission window assigned through the National Environmental Satellite, Data, and Information Service (NESDIS). See SOP 7 for details on the satellite transmitter processes.

Save and Compile

After the programming is complete, the file is saved and compiled. Errors are displayed and can be remedied. The final file is saved as a .CR1 or .CR1X file which can be sent to the datalogger (see SOP 9). Before the program is used at a remote station it should be tested. Having a test station located near the office is useful for testing programs, transmitters, sensors, and any station design modifications.

The datalogger programs typically vary by site, depending on which sensor models are in use and what parameters are measured.

The datalogger program must be archived with the annual data download for each station each year as a discreet deliverable WC_D (see SOP 10). The program source code defines to future users exactly which parameters were collected during its execution and how the measurements were calculated. It is important to keep data and program together since programs may be altered occasionally and program content is required in order for researchers to correctly understand the exact nature of the data collected by it.

An example program is included in Appendix A.

Create a Wiring Diagram

The program determines the wiring scheme for the datalogger. An example wiring diagram is included as Appendix B.

Appendix SOP 6 - A. Examples of CR1000 Datalogger Program for SEAN and CAKN

SEAN Program Example:

```
' ===== IDENTIFICATION =====
'Covering this instrumentation:
' CR1000 controller
' TE525 precip gauge
' HMP60 temp/RH sensor
' RM Young 5103 wind
' T107 soil temp sensors (3)
' LI200X solar radiation sensor
' Judd snow depth sensor
' TX320 transmitter
'*****
' Purpose: Manage CSI weather station using CR1000 controller
' Filename: WC_D_2016_DBWA2.crl
' Based on file: WC_D_2016 Nunatak_Upper.crl
' Activated on station: 20160502 (burn-in period; not yet deployed)
' Updated by: Bill Johnson ' Change Log:
'   2016-05-16 Make JuddHeight in a public variable instead of constant
'   2016_05_02 Adapted for Dry Bay by copying Nunatak_Upper program

'===== DECLARATIONS =====
Public RainCumulative_in      'Total rain in inches - until reset on October 1 or at maintenance
Public Rain_in                'Rain in inches
Public WindSpeed_mph          'Wind Speed (mph)
Public WindDirection_degrees  'Wind Direction (degrees)
Public AirTemp_F              'Air Temperature measured by HMP60 in F
Public FuelT_F                'Fuel temp in F - not measured but used to maintain RAWS data order
Public RelHumidity_pct        'Relative Humidity measured by HMP60 as %
Public Battery_V              'Battery voltage
Public SolarRadiation_W       'Solar Radiation in Watts/m**2
Public SnowDepth_in           'Distance to snow measured by Judd snow depth sensor
Public SoilTemp1_F            'Soil Temp at 10cm
Public SoilTemp2_F            'Soil Temp at 20cm
Public SoilTemp3_F            'Soil Temp at 50cm
Public Max_WindSpeed_mph      'Hourly maximum wind
Public Max_WindDirection      'Direction of hourly max wind
Public Pressure_inHg          'Barometer in inches mercury
Public JuddHeight_in = 120.0   'Starting point for height adjustment - must be corrected in the
field
Public JuddAirTemp_C          'AirTemp measured by Judd sensor to correct depth calculation
'Declare Windsonic variables (if used)
'Public windsonic(4) As String
'Public in_bytes_str As String * 21
'Public nmbr_bytes_rtrnd

'Declare GOES variables used for diagnostic purposes
Public DataRC                'Public variable holds the success/failure of transmission to the GOES
satellite.
                                'A result code other than 0 indicates an error

Public status(13)            'generated by the GOESStatus instruction
Public time(7)               'generated by the GOESGPS instruction
Public location(6)           'generated by the GOESGPS instruction

'Declare variables used to create well-formatted GOES message
Public obs(30)               ' extracted HourlyObservation table data to be reformatted
Public goesMsg As String * 128 'GOES message that is sent to transmitter hourly

Public CR1000Time(9) 'CR1000 time (used to reset RainCumulative_in on October 1)
'*****
'*****
'CR1000Time: Controller time variables
Alias CR1000Time(1)= CR1000Year
Alias CR1000Time(2)= CR1000Month
Alias CR1000Time(3)= CR1000Day
```

```

Alias CR1000Time(4)= CR1000Hour
Alias CR1000Time(5)= CR1000Minute
'Remaining aliases are for diagnostic information available in the Public table
'Location: station location
Alias location(1)=GPSdataRC
Alias location(2)=GPSSeconds
Alias location(3)=Latitude
Alias location(4)=Longitude
Alias location(5)=Elevation
Alias location(6)=MagDec
'Time: transmitted time
Alias time(1)=Year
Alias time(2)=Month
Alias time(3)=Days
Alias time(4)=Hours
Alias time(5)=Minutes
Alias time(6)=Seconds
Alias time(7)=Microseconds
'Status: transmitter status
Alias status(1)=RC_Status
Alias status(2)=ST_Bytes
Alias status(3)=ST_Day
Alias status(4)=ST_Hour
Alias status(5)=ST_Minute Alias status(6)=ST_Second
Alias status(7)=RDM_Bytes
Alias status(8)=RDM_Hour
Alias status(9)=RDM_Minute
Alias status(10)=RDM_Second
Alias status(11)=RDM_FailSafe
Alias status(12)=ST_BattVolt
Alias status(13)=Ave_GPS
'*****
'Define the units of the parameter variables.
Units RainCumulative_in=in
Units Rain_in=in
Units WindSpeed_mph=miles/hour
Units WindDirection_degrees=degrees
Units AirTemp_F=Deg F
Units FuelT_F=Deg F
Units RelHumidity_pct=%
Units Battery_V=Volts
Units Max_WindDirection=degrees
Units Max_WindSpeed_mph=miles/hour
Units SolarRadiation_W=W/m2
Units SnowDepth_in = in
Units SoilTemp1_F=Deg F
Units SoilTemp2_F=Deg F
Units SoilTemp3_F=Deg F
Units Pressure_inHg=in Hg
'===== TABLES =====
'Table with hourly averages of all parameters - stored for annual download.
' Every hour GOES transmits the contiguous block of 16 params beginning at RainCululative_in.
' All parameters (except the last) must be present, even if there's no sensor attached - no
deleting lines.
' This makes the column order of all our sites consistent, which prevents future confusion and
errors.
' GOES parameter #16 must be either soil temperature 3 or barometric pressure - one must be
specified, but not both.
' Parameters for which there are no sensors present are given the flag value -99, but are still
present in the table.
DataTable(HourlyObservation, True, -1)
DataInterval(0, 60, Min, 10)
'Explicitly timestamp the row
Sample (1, CR1000Year, FP2)
Sample (1, CR1000Month, FP2)
Sample (1, CR1000Day, FP2)
Sample (1, CR1000Hour, FP2)
Sample (1, CR1000Minute, FP2)

'GOES transmitted values are in table elements 6 through 21
Sample(1, RainCumulative_in, FP2)

```

```

Average(1, WindSpeed_mph, FP2, False)
WindVector (1, WindSpeed_mph, WindDirection_degrees, FP2, False, 0, 0, 3)
FieldNames("WindDirection_deg")
Average(1, AirTemp_F, FP2, False)
Average(1, FuelT_F, FP2, False)
Sample(1, RelHumidity_pct, FP2)
Average(1, Battery_V, FP2, False)
Sample(1, Max_WindDirection, FP2)
Sample(1, Max_WindSpeed_mph, FP2)
Average(1, SolarRadiation_W, FP2, False)
Average(1, SnowDepth_in, FP2, False)
Maximum(1, AirTemp_F, FP2, False, False)
Minimum(1, AirTemp_F, FP2, False, False)
Average(1, SoilTemp1_F, FP2, False)
Average(1, SoilTemp2_F, FP2, False)
Average(1, SoilTemp3_F, FP2, False)      ' transmitted param #16 is barometer if present, else it's
Soil temp #3
'Sample(1, Pressure_inHg, FP2)          ' barometer reading

'additional values not supported over GOES transmission
Totalize(1, Rain_in, FP2, False)        'rain over past hour
Minimum(1, Battery_V, FP2, False, False) 'minimum hourly battery voltage
Maximum(1, Battery_V, FP2, False, False) 'maximum hourly battery voltage
'Average(1, SoilTemp3_F, FP2, False)     ' if barometer is in GOES param #16 above, then collect
soil temperature #3 here
EndTable

' Table for holding HourlyObservation parameters reformatted as a string to send to GOES transmitter
DataTable(FWReformatted, True, 24)
  DataInterval(0, 60, Min, 10)
  Sample(1, goesMsg, String)
EndTable

'==== "PROGRAM" =====
' DEFINE THE SCAN INTERVAL, RECORD PARAMETER VALUES IN TABLES
' Starting at 5 minutes into each hour, send tables to TX312/320 buffer for transmission
' Transmission time is dependent on unique transmission window of NESDIS ID
BeginProg
'  SerialOpen (Com3,9600,3,0,108)      ' for Windsonic instrument otherwise comment out
Scan(60,Sec,1,0)
' Datalogger Battery Voltage measurement Battery_V
Battery(Battery_V)

' TE525 Rain Gauge measurement RainCumulative_in
PulseCount(Rain_in,1,2,2,0,0.01,0)
RainCumulative_in = RainCumulative_in + Rain_in

RealTime(CR1000Time)  'Reset accumulated rainfall on October 1 each year
If CR1000Month = 10 AND CR1000Day = 1 AND CR1000Hour = 0 AND CR1000Minute = 0 Then
RainCumulative_in = 0

' T107 Soil Temperature Probe measurement SoilTemp_F
Therm107(SoilTemp1_F, 1, 5, 1, 0, _60Hz, 1.8, 32)
Therm107(SoilTemp2_F, 1, 6, 1, 0, _60Hz, 1.8, 32)
Therm107(SoilTemp3_F, 1, 8, 1, 0, _60Hz, 1.8, 32)

' T109 Soil Temperature Probe measurement SoilTemp_F
' Therm109(SoilTemp1_F, 1, 6, 2, 0, _60Hz, 1.8, 32)
' Therm109(SoilTemp2_F, 1, 5, 1, 0, _60Hz, 1.8, 32)
' Therm109(SoilTemp3_F, 1, 8, 1, 0, _60Hz, 1.8, 32)

' HMP60 Temperature & Relative Humidity Sensor measurements AirTemp_F and RelHumidity_pct:
VoltSe(AirTemp_F, 1, mV2500, 1, 0, 0, _60Hz, 0.18, -40.0)
VoltSe(RelHumidity_pct, 1, mV2500, 7, 0, 0, _60Hz, 0.1, 0)
If (RelHumidity_pct > 100) AND (RelHumidity_pct < 108) Then RelHumidity_pct=100

' LI200X Pyranometer measurements SolarRadiation_W
VoltDiff(SolarRadiation_W, 1, mV7_5, 2, True, 0, _60Hz, 1, 0)
If SolarRadiation_W < 0 Then SolarRadiation_W = 0
SolarRadiation_W = SolarRadiation_W * 200

```

```

'Judd Depth sensor measurements for SnowDepth_in
PortSet(1, 1)      'Turn ON the snow depth sensor

Delay(0,800,MSEC)   'wait 0.8 seconds for the sensor to output the Air Temp.
'Air Temp is measured first for sonar correction, scaled to Kelvin and then converted to
Celsius.
VoltDiff(JuddAirTemp_C, 1, mV2500, 5, True, 0, 250, 0.2, -273 )
'The snow depth sensor needs another 1.8 seconds to make up to 10 measurements
'    and perform error checking and temperature compensation.
Delay(0,1800,MSEC)
'Distance is output in millivolts, scaled to inches, and then converted to depth by
subtracting the distance measured from the reference distance.
'Use a multiplier of -.19685 for output in inches and -.5 for output in centimeters.
'The following instruction is configured to output the measured distance in inches.
'Offset value is distance from ground to sensor in inches.
VoltDiff(SnowDepth_in, 1, mV2500, 5, True, 0, 250, -.19685, JuddHeight_in)
If SnowDepth_in < 0 Then SnowDepth_in = 0
PortSet(1, 0)      'Turn OFF depth sensor.

'Reset Max_WindSpeed_mph and Max_WindDirection each hour
If TimeIntoInterval (1,60,Min) Then
Max_WindSpeed_mph=0
Max_WindDirection=0
EndIf

'RM Young 5103 Wind Speed & Dir. Sensor for WindSpeed_mph and WindDir
'comment out if using WINDSONIC
PulseCount(WindSpeed_mph, 1, 1, 1, 1, 0.2192, 0)
BrHalf(WindDirection_degrees, 1, mV2500, 2, 1, 1, 2500, True, 0 , _60Hz, 355, 0)
If WindDirection_degrees>=360 Then WindDirection_degrees=0

'WindSonic for WindSpeed_mph and WindDirection_degrees
'comment out if using RM Young
SerialInRecord (Com3,in_bytes_str,&h02,0,&h0D0A,nmbr_bytes_rtr nd,01)
SplitStr (windsonic(),in_bytes_str,"",4,4) 'Split the string and convert to floats.
WindDirection_degrees = windsonic(1)
WindSpeed_mph = windsonic(2)      ' must do calculation in two statements, otherwise type
mismatch
WindSpeed_mph = WindSpeed_mph * 2.23694

If WindSpeed_mph > Max_WindSpeed_mph Then
Max_WindSpeed_mph = WindSpeed_mph
Max_WindDirection = WindDirection_degrees
EndIf

'Set values to -99 for unmeasured parameters (WFMI should skip any of these)
FuelT_F = -99
SoilTemp2_F = -99
SoilTemp3_F = -99
Pressure_inHg = -99

'Refresh authoritative table that gets downloaded at annual maintenance
CallTable(HourlyObservation)
'Build properly formatted RAWs message directly from the hourly table
GetRecord(obs, HourlyObservation, 1)
goesMsg = CHR(13) & CHR(10) & Trim(FormatFloat(obs(6), "%6.2f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(7), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(8), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(9), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(10), "%4.0f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(11), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(12), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(13), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(14), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(15), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(16), "%6.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(17), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(18), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(19), "%5.1f"))
goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(20), "%5.1f"))

```

```

goesMsg = goesMsg & CHR(13) & CHR(10) & Trim(FormatFloat(obs(21), "%5.1f"))
'Move the string to table
CallTable(FWReformatted)

'Send selected tables to GOES transmitter
'Data are sent to the buffer beginning at 5 minutes past the hour.
If TimeIntoInterval(5,60,Min) Then 'Make status, location, and time information of the
transmitter available in Public for diagnostic purposes.
    GOESStatus (status(),1)
    GOESGPS(location(), time())
    'Send the parameter string to GOES transmitter
    GOESData(DataRC, FWReformatted, 0, 0, 0)
EndIf
NextScan
EndProg

```

CAKN Program Example:

```
'CR1000
'Created by Ken Hill for Arctic and Central Alaska Network weather stations.
'Updated March 2016. Reordered variables. Renamed variables to match SOPs.
'Updated February 2016
'Deleted daily table (not used)
'Add TCDT to tables
'Delete excessive WS-WD values in "hour" table
'Reorganize tables. "Transmit", "Hour" only
'Updated May 2015
'Accumulated rainfall calculated and transmitted in RAWS7 (instead of rain_hour)
'Time array added for CR1000
'RNIN_Cumul reset on Oct. 1 each year.
'Hourly rain total sent at end of transmission window and recorded in hourly/daily tables.
'Backup Air Temperature measurement transmitted at end of transmission window.
'Updated May 2014.
'Fuel temperature placeholder added (currently no sensor).
'Soil temperatures edited and placed at end of transmission string.
'Date/time information added and is logged in hourly/daily tables
'Edited ordering of variables in hourly/daily tables to match transmission order.
'Updates 2012
'Transmit peak WS after the fireweather RAWS 7 table; added the WSPMPH and WDDP calculations
'tables; measure HMP155 in degree C for snow depth correction; changed soil temp output
'to deg F for consistency.
'Wiring Notes
'SR50a: Connect ground first (Black, White, Clear), +12V second (red),
'SDI-12 last (Green)
'***** Do Not apply power to TX312 until the antenna is connected *****
'Mounting/Installation Notes
'SR50a has a beam angle of 22 degrees. Clearance radius should be 0.268 x Sensor Height
'*****
'Declare Height of SR50 Snow depth instrument height above ground (inches)
'This height is unique to each station and must be entered in the field! In Inches!
Const SR50_ht=72
'DECLARE VARIABLES TO MEASURE
Public RNIN_Cumul 'Rainfall. Cumulative total since January 1 or last reset
Public WSM 'Wind Speed (mph)
Public WDD 'Wind Direction (degrees)
Public ATF_109 'Air Temperature in F from primary 109 sensor. Transmitted.
Public RHP_45 'Relative Humidity from AT-RH sensor
'Public RHP_155 'Relative Humidity from HMP155
Public FuelT 'Fuel Temperature (not measured)
Public BVV 'Battery Voltage
Public WSPMPH 'Peak Wind Speed; calculated each hour
Public WDDP 'Wind direction of peak gust; calculated each hour
Public SRW 'Solar Radiation in Watts per square meter
Public SlrkJ 'Solar Radiation in kJ
Public SDI 'Snow Depth
Public STF 'Soil Temp 1 (10cm)
Public STF2 'Soil Temp 2 (20cm)
Public STF3 'Soil Temp 3 (50cm)
'Public STF4 'Soil Temp 4 (75cm). Not at all sites.
Public RNIN_Tot 'Rain in inches. (hourly total)
Public TCDT 'Temperature Corrected Distance from SR50a sensor.
Public ATC_109 'Air temperature from 109 sensor in C. Primary sensor. For SR50a
Public ATF_HMP45 'Air Temperature from HMP45 in F. Backup sensor.
Public ATC_HMP45 'Air Temperature from HMP45 in C. Backup sensor.
Page 1 of 6
Program: 2016_Beta2.CR1
Public ATF_ThermX 'Air temperature from ThermX in F. Backup sensor.
Public ATC_ThermX 'Air temperature from ThermX in C. Backup sensor.
'Public ATF_HMP155 'Air temperature from HMP155 in F. Backup sensor.
'Public ATC_HMP155 'Air temperature from HMP155 in C. Backup sensor.
Public PTemp_C 'Panel Temperature in C
Public SR50A(2) 'Snow Depth parameter
'Declare GOES and TX312 Troubleshooting Variables
'A "DataRC" (result code) of 0 indicates a successful transmission
'A "DataRC" (result code) other than 0 indicates an error
'See the help section for the GOESData instruction for details
Public DataRC 'DataRC=0 indicates successful transmission
Public status(13) 'This is an array of values generated by the GOESStatus instruction
```

```

Public time(7) 'This is an array of values generated by the GOESGPS instruction
Public location(6) 'This is an array of values generated by the GOESGPS instruction
'See Alias variables below for details.
'See the GOESStatus and GOESGPS instruction help files for details.
'Also see the TX312 Transmitter manual for additional help.
Public CR1000Time(9) 'CR1000 time (sent to tables; used to reset RNIN_Cumul annually)
Public Elevation_ft
'Alias snow depth values
Alias SR50A(1)=DT 'Distance to Snow
Alias SR50A(2)=SR50_Quality 'Outputs quality of snow depth measurement (see manual)
'Alias Location Variables: Public values that indicate the location transmitter
Alias location(1)=GPSdataRC
Alias location(2)=GPSSecondsSince
Alias location(3)=Latitude
Alias location(4)=Longitude
Alias location(5)=Elevation
Alias location(6)=MagDec
'Alias Time Variables: Public variables the indicate time settings of the transmitter
Alias time(1)=GPSYear
Alias time(2)=GPSMonth
Alias time(3)=GPSDays
Alias time(4)=GPSHours
Alias time(5)=GPSMinutes
Alias time(6)=GPSSeconds
Alias time(7)=GPSMicroseconds
'Alias Status Variables: Public variables that indicate the status of the transmitter
Alias status(1)=RC_Status
Alias status(2)=ST_Bytes
Alias status(3)=ST_Day
Alias status(4)=ST_Hour
Alias status(5)=ST_Minute
Alias status(6)=ST_Second
Alias status(7)=RDM_Bytes
Alias status(8)=RDM_Hour
Alias status(9)=RDM_Minute
Alias status(10)=RDM_Second
Alias status(11)=RDM_FailSafe
Alias status(12)=ST_BattVolt
Alias status(13)=Ave_GPS
'Alias CR1000Time Variables: Public variables that indicate datalogger time. Used for resetting
Alias CR1000Time(1)= CR1000Year
Alias CR1000Time(2)= CR1000Month

```

Page 2 of 6

Program: 2016_Beta2.CR1

```

Alias CR1000Time(3)= CR1000DayOfMonth
Alias CR1000Time(4)= CR1000HourOfDay
Alias CR1000Time(5)= CR1000Minutes
Alias CR1000Time(6)= CR1000Seconds
Alias CR1000Time(7)= CR1000Microseconds
Alias CR1000Time(8)= CR1000DayOfWeek
Alias CR1000Time(9)= CR1000DayOfYear
'Define the units of the important public variables.
Units RNIN_Cumul=inches
Units WSM=miles per hour
Units WDD=degrees
Units ATF_109 = degrees F
Units RHP_45=%
'Units RHP_155=%
Units BVV=volts
Units WSPMPH=miles per hour
Units WDDP=degrees
Units SRW=W/m2
Units SlrkJ=kJ/m2
Units SDI = inches
Units STF = degrees F
Units STF2 = degrees F
Units STF3 = degrees F
'Units ST4 = degrees F
Units RNIN_Tot=inches
Units TCDT=inches
Units ATC_109 = degrees C

```

```

Units ATF_HMP45 = degrees F
Units ATC_HMP45=degrees C
Units ATF_ThermX=degrees F
Units ATC_ThermX=degrees C
'Units ATF_HMP155 = degrees F
'Units ATC_HMP155=degrees C
Units PTemp_C=degrees C
*****
'DEFINE DATA TABLES TO BE TRANSMITTED AND DOWNLOADED
'Write table for variables to be transmitted:
'precip, average wind speed, wind direction, air temperature, fuel temperature (skipped)
'relative humidity, BattV, WSPMPH, WDDP, Solar, SnowDepth, ATmin, ATmax, STF, STF2, STF3, ST4
DataTable(transmit,True,-1)
DataInterval(0,60,Min,10)
Sample(1,RNIN_Cumul,FP2)
Average(1,WSM,FP2,False)
WindVector (1,WSM,WDD,FP2,False,0,0,3)
FieldNames("WDD D1 WVT")
Average(1,ATF_109,FP2,False)
Average(1,FuelT,FP2,False)
Sample(1,RHP_45,FP2)
Average(1,BVV,FP2,False)
Sample(1,WSPMPH,FP2)
Sample (1,WDDP,FP2)
Average(1,SRW,FP2,False)
Sample(1,SDI,FP2)
Minimum(1,ATF_109,FP2,False, False)
Maximum(1,ATF_109,FP2,False, False)
Average(1,STF,FP2,False)
Page 3 of 6
Program: 2016_Beta2.CR1
Average(1,STF2,FP2,False)
Average(1,STF3,FP2,False)
'Average(1,ST4,FP2,False)
Totalize(1, RNIN_Tot, FP2,False) 'Not picked up by WFMI. Diagnostic purposes on
Average(1,ATF_HMP45,FP2,False) 'Not picked up by WFMI. Replicate AT measureme
Average(1,ATF_ThermX,FP2,False) 'Not picked up by WFMI. Replicate AT measureme
'Average(1,ATF_HMP155,FP2,False) 'Not picked up by WFMI. Replicate AT measureme
EndTable
'Write a table with hourly averages of all parameters
DataTable(hourly,True,-1)
DataInterval(0,60,Min,10)
Sample (1,GPSYear,FP2) 'Record year in table
Sample (1,GPSMonth,FP2) 'Record month in table
Sample (1,GPSDays,FP2) 'Record day of month in table
Sample (1,GPSHours,FP2) 'Record hour in table
Sample (1,CR1000Year,FP2) 'Record CR1000 Year
Sample (1,CR1000Month,FP2) 'Record CR1000 Month
Sample (1,CR1000DayOfMonth,FP2) 'Record CR1000 DayOfMonth
Sample (1,CR1000HourOfDay,FP2) 'Record CR1000 HourOfDay
Sample (1,CR1000DayOfYear,FP2) 'Record CR1000 DayOfYear
Totalize(1,RNIN_Tot,FP2,False) 'Total Rain over the hour
Sample(1,RNIN_Cumul,FP2) 'Accumulated Rain since Jan. 1
Average(1,WSM,FP2,False) 'Average Wind Speed
WindVector (1,WSM,WDD,FP2,False,0,0,3) 'Average Wind Direction
Average(1,ATF_109,FP2,False) 'Average Air Temperature from 109 (F)
Sample(1,RHP_45,FP2) 'Relative Humidity HMP45 (Sampled on the hour)
'Sample(1,RHP_155,FP2) 'Relative Humidity HMP155 (Sampled on the hour)
Average(1,BVV,FP2,False) 'Average Battery Voltage
Sample(1,WDDP,FP2) 'Direction of peak gust
Sample (1,WSPMPH,FP2) 'Max WS for the hour
Average(1,SRW,FP2,False) 'Average Solar Radiation for the hour
Average(1,SDI,FP2,False) 'Average snow depth (avg 1 value taken on hour
Minimum(1,ATF_109,FP2,False, False) 'Minimum Air Temperature from ThermX
Maximum(1,ATF_109,FP2,False, False) 'Maximum Air Temperature from ThermX
Average(1,STF,FP2,False) 'Average Soil Temperature (5cm)
Average(1,STF2,FP2,False) 'Average Soil Temperature 20cm
Average(1,STF3,FP2,False) 'Average Soil Temperature 50cm
'Average(1,ST4,FP2,False) 'Average Soil Temperature 75cm
Minimum(1,ATF_HMP45,FP2,False, False) 'Minimum Air Temperature from HMP45 Backup
Average(1,ATF_HMP45,FP2,False) 'Average Air Temperature from HMP45 Backup

```

```

Maximum(1,ATF_HMP45,FP2,False, False) 'Maximum Air Temperature from HMP45 Backup
Minimum(1,ATF_ThermX,FP2,False, False) 'Minimum Air Temperature from ThermX Backup
Average(1,ATF_ThermX,FP2,False) 'Average Air Temperature from ThermX Backup
Maximum(1,ATF_ThermX,FP2,False, False) 'Maximum Air Temperature from ThermX Backup
'Minimum(1,ATF_155,FP2,False, False) 'Minimum Air Temperature from 109 Backup
'Average(1,ATF_155,FP2,False) 'Average Air Temperature from 109 Backup
'Maximum(1,ATF_155,FP2,False, False) 'Maximum Air Temperature from 109 Backup
Average(1,DT, FP2, False) 'Average non-temperature corrected distance in
Average(1, TCDT, FP2, False) 'Average distance to ground (avg 1 value taken
EndTable
'*****
'MAIN PROGRAM: DEFINE A SCAN INTERVAL and TAKE MEASUREMENTS
'Main Program: for each scan, take measurements, store info in tables.
'Starting at 5 minutes into each hour, send tables to TX312 buffer for transmission
'Transmission at ~40 minutes into the hour
'Transmission time is dependent on unique transmission window of NESDIS ID
Page 4 of 6
Program: 2016_Beta2.CR1
BeginProg
Scan(60,Sec,1,0)
'Rainfall
PulseCount(RNIN_Tot,1,2,2,0,0.01,0) 'TE525/TE525WS Rain Gauge measurement RNIN_Tot
RNIN_Cumul=RNIN_Cumul+RNIN_Tot 'Calculate accumulated rainfall
If CR1000Month=10AND CR1000DayOfMonth=1AND CR1000HourOfDay=0AND CR1000Minutes=0 Then
RNIN_Cumul=0 'Reset accumulated rainfall on Oct. 1 each year
EndIf
'Wind Speed/Wind Direction (05103 Sensor)
PulseCount(WSM,1,1,1,1,0.2192,0) 'Multiplier is 0.2192 for mph (from manual).
BrHalf(WDD,1,mV2500,3,2,1,2500,True,0,_60Hz,355,0)
If WDD>=360 Then WDD=0
'Track the max wind gust each hour and the direction of the gust
If WSM> WSPMPH Then
WSPMPH= WSM
WDDP = WDD
EndIf
'Reset WSPMPH and WDDP each hour
If TimeIntoInterval (1,60,Min) Then
WSPMPH=0
WDDP=0
EndIf
'Air Temperature T109 (primary sensor)
Therm109 (ATF_109,1,16,Vx1,0,_60Hz,1.8,32)
ATC_109 = (ATF_109-32)/1.8
'Relative Humidity and Air Temperature(HMP45 6-wire, HMP45 7-wire, or HMP155)
'HMP45 (6-wire continuous power; optional switch power)
'PortSet(9,1)
'Delay(0,150,mSec)
VoltSe(ATF_HMP45,1,mV2500,11,0,0,_60Hz,0.18,-40)
VoltSe (ATC_HMP45,1,mV2500,11,0,0,_60Hz,0.1,-40)
VoltSe(RHP_45,1,mV2500,12,0,0,_60Hz,0.1,0)
If RHP_45>100 AND RHP_45<108 Then RHP_45=100
'PortSet(9,0)
'HMP45 (7-wire switch power to control port #1)
'PortSet(1,1)
'Delay(0,150,mSec)
'VoltSe(ATF_HMP45,1,mV2500,11,0,0,_60Hz,0.18,-40)
'VoltSe (ATC_HMP45,1,mV2500,11,0,0,_60Hz,0.1,-40)
'VoltSe(RHP_45,1,mV2500,12,0,0,_60Hz,0.1,0)
'If RHP_45>100 AND RHP_45<108 Then RHP_45=100
'PortSet(1,0)
'HMP155 (6-wire continuous power; optional switch power)
'PortSet(9,1) 'Turn on SW 12 V power
'Delay(0,2,Sec) 'HMP155A takes 2 seconds to warm up
'VoltSe(ATF_HMP155,1,mV2500,11,0,0,_60Hz,0.252,-112)
'VoltSe (ATC_HMP155,1,mV2500,11,0,0,_60Hz,0.14,-80)
'VoltSe(RHP_155,1,mV2500,12,0,0,_60Hz,0.1,0)
'If RHP_155>100 AND RHP_155<108 Then RHP_155=100
'PortSet(9,0)
'Fuel Temperature (not measured, skipped at WFMI)
Therm107(FuelT,1,7,Vx3,0,_60Hz,1.8,32)
'Battery Voltage

```

Battery (BVV)

Page 5 of 6

Program: 2016_Beta2.CR1

```
'Solar Radiation (Li200x Sensor)
VoltDiff(SRW,1,mV7_5,3,True,0,_60Hz,1,0)
If SRW<0 Then SRW=0
SlrkJ=SRW*12 'Total flux density = kJ/m2 = mV * Execution Interval * 0.2 = 60*0.2 = 12 (
SRW=SRW*200 'Average Flux = Watts/m2 = mV * 200 = SRW * 200
'Snow Depth (SR50a SDI-12).
'SR50A Sonic Ranging Sensor (SDI-12 Output). Measurements include DT, Q, TCDT, and DBTC DT
If TimeIntoInterval (59,60,min) Then 'Sample once per hour at 59 minutes to avoid e
SDI12Recorder(SR50A(),3,"0","M6!",1,0)
TCDT=DT*SQR((ATC_ThermX+273.15)/273.15) 'Correct for temperature
SDI=SR50_ht-TCDT 'Calculate depth. SR50_ht is a constant entered at the beginning of t
If SDI<0 Then SDI=0
EndIf
'Soil Temperatures (10 cm, 20 cm, 50 cm, 75 cm)
Therm107(STF,1,8,Vx3,0,_60Hz,1.8,32) 'STF (10 cm)
Therm107(STF2,1,9,Vx3,0,_60Hz,1.8,32) 'STF2 (20 cm)
Therm107(STF3,1,10,Vx3,0,_60Hz,1.8,32) 'STF3 (50 cm)
Therm107(ST4,1,13,Vx3,0,_60Hz,1.8,32) 'ST 4 (75 cm)
'Air Temperature Backup ThermX Sensor
BrHalf (ATF_ThermX,1,mV2500,4,Vx1,1,2500,False,100,250,355,-89.1335) 'ThermX Air Tempera
ATC_ThermX = (ATF_ThermX-32)/1.8
'Panel Temperature
PanelTemp (PTemp_C,_60Hz) 'Wiring Panel Temperature measurement PTemp_C
'These instructions make status, location, and time information from the...
'transmitter and datalogger available as Public Variables
'See Instruction help files for details.
GOESStatus (status(),1)
GOESGPS(location(), time())
RealTime(CR1000Time)
Elevation_ft=Elevation*0.0328
'*****
'CALL DATA TABLES TO STORE and TRANSMIT DATA
CallTable(transmit)
CallTable(hourly)
'Send data tables to TX312 for transmission to GOES.
'Data is sent to the TX buffer beginning at 5 minutes past the hour.
'Fireweather RAWS variables (Rain, WS, WD, ATF_HMP45 (HMP45), FuelT, RH, BattV)
If TimeIntoInterval (5,60,min) Then
GOESData(DataRC,transmit,0,0,0) 'sends data to transmit to GOES self-timed buffer in
EndIf
NextScan
EndProg
```

Appendix SOP 6 - B. Example of Wiring Diagrams for SEAN and CAKN

CR1000 Wiring Schematic - CAKN

With HMP45 and ThermX AT Sensor

Date: May 2015

Program: Toklat_2015

[1] 1H	[2] 1L	G	[3] 2H	[4] 2L	G	[5] 3H	[6] 3L	G	[7] 4H	[8] 4L	G	EX1	G	P1	G	P2	G
			WIND Green	ThermX1 White	WIND White Shield	SOLAR Red	SOLAR Black	SOLAR White Shield		SOIL 10cm Red	SOIL 5cm Purple	ThermX1 Black	ThermX1 Red	WIND Red	WIND Black	PRECIP Black	PRECIP White, Shield
					 			 									 

[9] 5H	[10] 5L	G	[11] 6H	[12] 6L	G	[13] 7H	[14] 7L	G	[15] 8H	[16] 8L	G	EX2	G	EX3
SOIL 20cm Red	SOIL 50cm Red	SOIL All Depth Purple	HMP 45 Yellow	HMP 45 Blue	HMP 45 Black,White,Shield	SOIL 100cm Red						WIND Blue	SOIL All Depth Clear	SOIL All Depth Black
					  									

G	5V	G	SW 12V	G	12 V	12V	G	C1	C2	C3	C4	G	C5	C6	C7	C8	G
					HMP45 Red	SNOW Red	SNOW Black, White, Shield	HMP45 (*7wire Only)		SNOW Green							
							  										

*6-wire Switched Power = Jumper from C1 to SW 12 Volt

CR1000 Wiring Schematic - SEAN

Common wiring scheme accommodates all instruments in current use

TE525/TE525WS - Rain_in

White	$\frac{\perp}{\equiv}$ (Ground)
Clear	$\frac{\perp}{\equiv}$ (Ground)
Black	P2

HMP50/HMP60 - AirTF, RH

Black	1H
White	4H
Brown	12v\V
Blue	G
Clear	$\frac{\perp}{\equiv}$ (Ground)

JC Depth (analog) - DistT, Depth, AirTF_2

White	5H
Brown	5L
Red	12v
Green	C1
Black	G
Clear	$\frac{\perp}{\equiv}$ (Ground)

LI200X - SirW, SirMJ

Red	P1
Black	2L
Clear	$\frac{\perp}{\equiv}$ (Ground)
White	$\frac{\perp}{\equiv}$ (Ground)

CR1000

05103 - WS_mph, WindDir_2

Green	1L
White	$\frac{\perp}{\equiv}$ (Ground)
Clear	$\frac{\perp}{\equiv}$ (Ground)
Black	$\frac{\perp}{\equiv}$ (Ground)
Red	P1
Blue	VX1 or EX1

CR1000

107 (1) - T107_F

Red	3H
Clear	$\frac{\perp}{\equiv}$ (Ground)
Purple	$\frac{\perp}{\equiv}$ (Ground)
Black	VX1 or EX1

CR1000

107 (2) - T107_F

Red	3L
Clear	$\frac{\perp}{\equiv}$ (Ground)
Purple	$\frac{\perp}{\equiv}$ (Ground)
Black	VX1 or EX1

CR1000

107 (3) - T107_F

Red	3H
Clear	$\frac{\perp}{\equiv}$ (Ground)
Purple	$\frac{\perp}{\equiv}$ (Ground)
Black	VX1 or EX1

CR1000

WindSonic1 (9.6K) - WindDir, WS_mph_2**CR1000**

Red	12v
Green	C3
White	C4
Black	G
Clear	$\perp$ (Ground)

CS100 - BP_inHg**CR1000**

Blue	6H
Red	12V
Green	C2
Black	G
Clear	$\perp$ (Ground)
Yellow	$\perp$ (Ground)

HMP45C (6WPS) – AirTF_3, RH_2**CR1000**

Yellow	6L
Blue	7H
Clear	$\perp$ (Ground)
White	$\perp$ (Ground)
Black	$\perp$ (Ground)
Red	SW-12

Note: if T109 soil temp sensor is used in place of T107, wiring remains the same.