❖ *Programming Protocol*

The program for the CR3000 datalogger will provide all instructions for automation of the chamber system. The program controlling the autochamber system needs to: 1) collect data from the temperature sensors and gas analyzers, and 2) automate the control of the chambers' sample air and piston air, by way of controlling solenoids. The programming involved with these tasks will be based around If... Then... instructions within a 30 minute schedule. It is important to recognize that the architecture of this program is designed for a particular data output format, whereas the data table allocates columns to each of the consecutive 20 CO_2 and O_2 measurements. This format allows for convenient placement of the flux calculation column at the end of the datatable. If it is decided that this data table format should be different, the majority of this program will be obsolete.

```
'CR3000 Series Datalogger

'Declare Public Variables

Public PTemp_C
Public relayset(16)
Public CO2(20)
Public O2(20)
Dim j
Public AirTemp
Public SoilTemp
Public ChamberNumber As Long
Public S_Scans

'Create boolean variables for do/don't do logic
Public SampleNow As Boolean
```

Begin building your program by creating intuitive aliases for the variables for which you will be programming instructions, in the Declare Public Variables section. Some of the variables will be created as data output destinations for particular sample measurements, while others will be used as variables within the program's logic.

Variable	Description
`Public PTemp_C`	Variable for recording datalogger's panel temperature. This variable is used as a reference in the thermocouple instructions found later in the program
`Public relayset(16)`	Variable array representing the 16 separate channels on the SDM controller. To avoid redundant programming, numbers in paranthesis following a variable name constitute a "variable array." All piston and air sample solenoids are controlled by the separate channels on the SDM. This variable will be used to schedule the timing of those solenoids.

Variable	Description
`Public CO2(20)`	Variable array for recording the 20 consecutive CO2 measurements during each chamber's 4-min cycle.
`Public O2(20)`	Variable array for recording the 20 consecutive O2 measurements during each chamber's 4-min cycle.
`Dim j`	The Dim statement is used to declare variables and variable arrays, and allocate storage space for these variables. Here the Dim statement will provide a variable array for the CO2 and O2 sampling instructions and storage.
`Public AirTemp`	Variable for recording air temperature for each chamber's sampling cycle. To be placed last in the data output table.
`Public SoilTemp`	Variable for recording soil temperature for each chamber's sampling cycle. To be placed last in the data output table.
`Public ChamberNumber As Long`	Variable for recording the chamber's number (1-6) along with it's recorded data. Allows for quick reference of which chamber each data record was taken from.
`Public S_Scans`	Records the datalogger's total "skipped scans" while running the program. Quick reference of possible programming compiling errors placed in the datatable.
`Public SampleNow as Boolean`	Variable to represent the subroutine within the program which will instruct all air sample measurements. Boolean variables allow for Do/Don't logic within the program.

```
'Declare Units
Units PTemp_C=Deg C
Units CO2=Volts
Units O2=Volts
Units AirTemp=Deg C
Units SoilTemp=Deg C
```

For all variables created as data output destinations in the datatable, appropriate units must be chosen to accompany their data.

```
'Define Data Tables
DataTable (MasterTB,1,-1)
  DataInterval (0,5,min,10)
  Sample (1,ChamberNumber,Long)
  Average (1,PTemp_C,FP2,False)
  Sample (20,CO2(),FP2)
  Sample (20,O2(),IEEE4)
  Sample (1,AirTemp,FP2)
  Sample (1,SoilTemp,FP2)
EndTable
```

The Define Data Tables section of the program is where you instruct the datalogger on how, when, and where to output data. At a glance, this program outputs data to a single table, called MasterTB, every 5 minutes; the table will record 20 CO2 and O2 measurements, 1 AirTemp and SoilTemp record and so forth. Instructions are discussed specifically in the chart below.

instruction	Description
`DataTable (MasterTB,1,-1)`	**DataTable** (Name, TrigVar, Size) Used to define the name, trigger condition, and size of an output table. The EndTable statement designates the end of the output table.
`DataInterval (0,5,min,10)`	**Datainterval** (TintoInt, Interval, Units, Lapses) Sets the time interval for data storage on the designated data table. Time into Interval allows for an offset of time to the specified interval (not used here). Our program needed data to be output with the end of each chamber's sampling cycle, which is 5 minutes.
`Sample (1,ChamberNumber,Long)`	**Sample** (Reps, Source, DataType) Used to record the current value of the source variable when output to the DataTable is instructed to occur. The ChamberNumber will be recorded first in the table, next to the TimeStamp.
`Average (1,PTemp_C,FP2,False)`	**Average** (Reps, Source, DataType, DisableVar) Records the average value of the source variable over the output tim interval to the DataTable.
`Sample (20,CO2(),FP2)`	**Sample** (Reps, Source, DataType) Here the Sample instruction is used, with the repetitions parameter set at 20 and the source as a variable array of 20. This will create an array of columns in the datatable to be populated by each consecutive CO2 measurement in each chamber's 4-min sample cycle.
`Sample (20,O2(),IEEE4)`	Sample instruction for O2 as described above. This Sample instruction uses a IEEE4 (four-byte floating point) data type, instead of the standard FP2 data type, because the O2 data requires 5 decimal places provided by the IEEE4.
`Sample (1,AirTemp,FP2)`	Sample instruction for a single air temperature measurement, to be placed at the end of the table.
`Sample (1,SoilTemp,FP2)`	Sample instruction for a single soil temperature measurement, to be placed at the end of the table.

```
Sub SampleNow
  For j = 1 To 20
    VoltDiff (O2(j),1,mV5000,1,True ,0,_60Hz,0.005,0)
    VoltDiff (CO2(j),1,mV5000,2,True,0,_60Hz,.4,0)
    Delay (0,12,sec)
```

```
  Next j
EndSub
```

A subroutine is a discrete procedure within the main program, able to take arguments and perform a series of instructions. Here a subroutine is used to consolidate the large package of instructions required for sampling O2 and CO2 120 times every 30 minutes. At a glance, this subroutine instructs the program to record a measurement from the CO2 and O2 gas analyzers every 12 seconds for 4 minutes. The scheduling of when this subroutine begins or is "Called" is written later in the program.

Instruction	Description

`Sub SampleNow`
`...`
`EndSub`
	Sub (Subroutine Name)
	EndSub
	Declaration identifies the beginning of the Subroutine.

`For j = 1 To 20`
`...`
`Next j`
	Here the *j* variable represents the 20 air sample measurements to populate the CO2() and O2() destinations in the DataTable. A 12-sec delay is placed between each measurement, with the subroutine reaching the EndSub instruction after the 20th *j.*

`Delay` `(0,12,sec)`	**Delay** (Option, Delay, Units) A 12-second delay is used between each *j*, or air sample measurement.

VoltDiff (*O2(j)*,1,mV5000,1,True,0,_60Hz,0.005,0)
VoltDiff(Dest, Reps, Range, DiffChan, RevDiff, SettlingTime, Integ, Mult, Offset)

The VoltDiff instruction is used to make a differential voltage measurement on one or more of the datalogger's analog channels. Here the VoltDiff instruction takes a measurement from the O2 analyzer connected to DiffChan 1 on the logger, outputting the data to the O2(j) columns on the DataTable. Notice that there is only one repetition in the instruction, as the 20 required sampling repetitions are accounted for by the Subroutine that the VoltDiff instruction is called within.

The voltage Range parameter is the expected voltage range of the input from the sensor, here an O2 gas analyzer. mV5000 is the standard voltage range, however the instrument's manual should describe its output voltage range to be accounted for in the program. Once the range is determined, the instruction's Multiplier and Offset parameters should be calculated and adjusted in order to match the datalogger's data output to that of the instrument (also found in the instrument's manual). Here a 0.005 multiplier was required to match the data outputs. Note that matching the instrument and logger are not necessarily needed during data collection but will eventually be needed later, during data analysis. These small adjustments save time with data processing. Consider the following equation when creating the Offset and Multiplier:

$$y = x(m) + q$$

Where y = logger data output; x = instrument data output; m = VoltDiff multiplier; q = VoltDiff offset

Lastly, the Integration for all instructions should be set at _60Hz. This setting will best reduce noise from the electronics on the datalogger's panel.

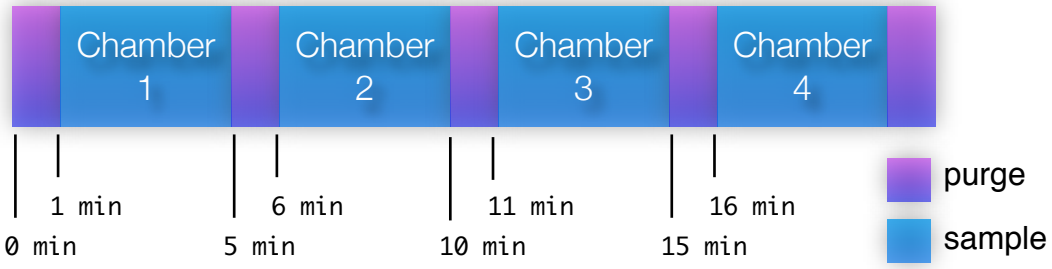
```
'Main Program
BeginProg
  Scan (1,min,0,0)
    PanelTemp (PTemp_C,_60Hz)
```

Having created all variables and designated their respective data output destinations in a data table, the main program can now be written, starting with the BeginProg instruction. Note that the program controlling the autochamber system needs to: 1) collect data from the sensors and analyzers, and 2) automate the control of the chambers' sample air and piston air, by way of controlling solenoids. Task 1 has mostly been completed by defining the Datatable and SampleNow subroutine, leaving task 2 for the main program. With all sample air and piston solenoids connected to the SDM, the main program has been written to provide a scheduling protocol for the 12 populated channels on the SDM device. That said, this section of the program is entirely comprised of timing instructions using If... Then... statements.

It is important to understanding the relayset(16) variable array with regard to the SDM wiring. Recall that the 6 chamber piston solenoids occupy channels 1-6 of the SDM, while each pair of in/out air sample solenoids populate channels 7-12. (Channels 13-16 of the SDM are not used in this configuration)

SDM configuration

CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
Air 6 In/Out	Air 5 In/Out	Air 4 In/Out	Air 3 In/Out	Air 2 In/Out	Air 1 In/Out	Piston 6	Piston 5	Piston 4	Piston 3	Piston 2	Piston 1



Above is a figure showing the scheduling protocol written in the main program. Observe that each chamber closes for a 4 minute sampling period, with a 1-min purge between–this totals a 30-min cycle for the whole system.

TCDiff (AirTemp,1,mV20c,13,TypeT,PTemp_C,True,0,_60Hz,1,0)
TCDiff (Dest, Reps, Range, DiffChan, TCType, TRef, RevDiff, SettlingTime, Integ, Mult, Offset)

The TCDiff instruction is used to make a thermocouple measurement on a differential channel and convert the measurement to degrees Celsius. Several of the instruction's parameters are described above under the VoltDiff instruction (e.g. Dest, Reps, Range etc.). For the TCDiff instruction, the TCType parameter identifies the type of thermocouple being measured (here Type T). The instruction's TRef paramater is the name of the variable which holds the result of the reference temperature, here set as the logger's internal panel temperature.

```
 'Relay sequencing section
  If  TimeIntoInterval (0,30,min)  Then
```

```
 relayset(6) = false.....chamber 6 opens
 relayset(12) = false.....sample intake 6 closes
 relayset(7) = true.....sample intake 1 opens
 TCDiff (AirTemp,1,mV20c,13,TypeT,PTemp_C,True,0,_60Hz,1,0).....chamber 6 air temp
 TCDiff (SoilTemp,1,mV20c,14,TypeT,PTemp_C,True,0,_60Hz,1,0).....chamber 6 soil temp
EndIf

If TimeIntoInterval (1,30,min)  Then
    relayset(1) = true.....chamber 1 closes
    ChamberNumber = 1.....chamber number 1 recorded
 EndIf

If TimeIntoInterval (5,30,min)  Then
    relayset(1) = false.....chamber 1 opens
    relayset(7) = false.....sample intake 1 closes
    relayset(8) = true.....sample intake 2 opens
    TCDiff (AirTemp,1,mV20c,3,TypeT,PTemp_C,True,0,_60Hz,1,0).....chamber 1 air temp
    TCDiff (SoilTemp,1,mV20c,4,TypeT,PTemp_C,True,0,_60Hz,1,0).....chamber 1 soil temp
EndIf
```

Instruction	Description
`If... Then... EndIf`	Statements which allow instructions to be processed conditionally, based on the evaluation of an expression. Here conditional statements are designated by time intervals
`TimeIntoInterval`	**TimeIntoInterval**(TintoInt, Interval, Units) Instruction designates the interval of time in which to perform a set of instructions. Throughout the main program, the same 30 min interval is used while the TintoInt, or offset parameter is adjusted per chamber. (i.e. 1 minute into a 30 minute interval, Chamber 1 closes; 5 minutes into a 30 minute interval, Chamber 1 reopens.)

```
  If TimeIntoInterval (6,30,min)  Then
    relayset(2) = true  'Chamber 2 closes
    ChamberNumber = 2
  EndIf

  If  TimeIntoInterval (10,30,min)  Then
    relayset(2) = false  'Chamber 2 opens
    relayset(8) = false  'Intake 2 closes
    relayset(9) = true  'Intake 3 opens
    TCDiff (AirTemp,1,mV20c,5,TypeT,PTemp_C,True ,0,_60Hz,1,0)
    TCDiff (SoilTemp,1,mV20c,6,TypeT,PTemp_C,True ,0,_60Hz,1,0)
  EndIf

  If TimeIntoInterval (11,30,min)  Then
    relayset(3) = true  'Chamber 3 closes
    ChamberNumber = 3
```

```
    EndIf

    If  TimeIntoInterval (15,30,min)  Then
      relayset(3) = false  'Chamber 3 opens
      relayset(9) = false  'Intake 3 closes
      relayset(10) = true  'Intake 4 opens
      TCDiff (AirTemp,1,mV20c,7,TypeT,PTemp_C,True ,0,_60Hz,1,0)
      TCDiff (SoilTemp,1,mV20c,8,TypeT,PTemp_C,True ,0,_60Hz,1,0)
    EndIf

    If TimeIntoInterval (16,30,min)  Then
      relayset(4) = true  'Chamber 4 closes
      ChamberNumber = 4
    EndIf

    If  TimeIntoInterval (20,30,min)  Then
      relayset(4) = false  'Chamber 4 opens
      relayset(10) = false  'Intake 4 closes
      relayset(11) = true  'Intake 5 opens
      TCDiff (AirTemp,1,mV20c,9,TypeT,PTemp_C,True ,0,_60Hz,1,0)
      TCDiff (SoilTemp,1,mV20c,10,TypeT,PTemp_C,True ,0,_60Hz,1,0)
    EndIf

    If TimeIntoInterval (21,30,min)  Then
      relayset(5) = true  'Chamber 5 closes
      ChamberNumber = 5
    EndIf

    If  TimeIntoInterval (25,30,min)  Then
      relayset(5) = false  'Chamber 5 opens
      relayset(11) = false  'Intake 5 closes
      relayset(12) = true  'Intake 6 opens
      TCDiff (AirTemp,1,mV20c,11,TypeT,PTemp_C,True ,0,_60Hz,1,0)
      TCDiff (SoilTemp,1,mV20c,12,TypeT,PTemp_C,True ,0,_60Hz,1,0)
    EndIf

    If TimeIntoInterval (26,30,min)  Then
      relayset(6) = true  'Chamber  closes
      ChamberNumber = 6
    EndIf

    SDMCD16AC (relayset(),1,0)


'Sample Subroutine

  If  TimeIntoInterval (1,30,min)  Then Call SampleNow.....sample from Chamber 1
  If  TimeIntoInterval (6,30,min)  Then Call SampleNow..... sample from Chamber 2
  If  TimeIntoInterval (11,30,min)  Then Call SampleNow.....sample from Chamber 3
  If  TimeIntoInterval (16,30,min)  Then Call SampleNow.....sample from Chamber 4
  If  TimeIntoInterval (21,30,min)  Then Call SampleNow.....sample from Chamber 5
  If  TimeIntoInterval (26,30,min)  Then Call SampleNow.....sample from Chamber 6
```

Lastly, the SampleNow subroutine is called on schedule with all of the solenoids opening/closing in order .

```
  S_Scans = status.skippedscan(1,1)

    CallTable(MasterTB)

   SetStatus ("skippedscan",0)

  NextScan
EndProg
```

Instruction	Description
`SDMCD16AC (relayset(),1,0)`	**SDMCD16AC**(Source, Reps, SDMAddress) Instruction for directing the *relayset()* instructions to control the SDM relay control port device.
`CallTable(MasterTB)`	**CallTable**(Table Name) The CallTable instruction is used in the main program to call a Data Table. When the Data Table is called, it will check the output condition and process data as programmed.
`NextScan`	**Scan** **NextScan...** Redirects the datalogger back to the beginning of the program, or Scan instruction, creating a loop. Without this instruction, the program would reach the “EndProg” instruction after 30 minutes and not repeat itself.