MicroLab

500-series

Programming Manual

Programming in MicroLab

This manual is designed to introduce you to the functions and operations you can employ with MicroLab programming. This brief introduction will show you how to automate tasks in MicroLab that you *could* do manually, but is easier and more reliable to perform these tasks by the use of programming steps.

Introduction

MicroLab interfaces operate by following a series of steps used to control the way the interface runs. Let's take a look at an example of having a temperature probe stop taking temperature at 30 degrees C. Before starting any programming steps, be sure that all the sensors you will need for your experiments are connected and any timers you need have been selected.

Ordinarily you would need to start your experiment and monitor it until it reached the desired temperature. This is not a difficult process but it does prevent you from carrying out any other tasks required in the lab.

What if you were performing a whole series of these temperature runs and you also needed to track time accurately? Imagine watching the temperature and the time while trying to get the next run ready!

Rather than get involved like that, set up a series of program steps that tells the program when to stop.

Look at the **Experiment Steps** window in MicroLab now. Notice it has a program already in it. This simple program is the default set of steps telling MicroLab when to collect data. Currently, the program tells MicroLab to collect data by reading the sensors every 0.5 seconds until the **Stop** button is pressed.

We are going to modify the **Until** condition to allow the program to stop taking data
Double click the **Until** statement in the program. You will get a dialog box with all the various options associated with the **Until** statement (Figure 1). In our scenario – we want the program to stop when a condition is met, so select **Condition True**. Now we need to tell it what condition to make true. Select **Edit Condition** and the Condition dialog will appear (Figure 2). Here we want our temperature sensor to

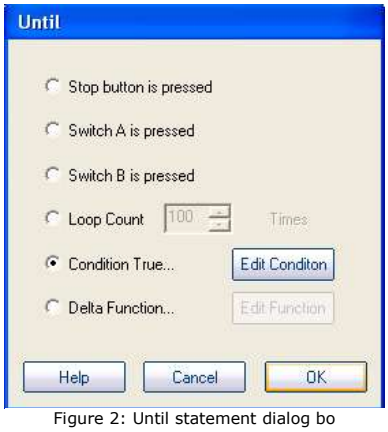


Figure 2: Until statement dialog bo

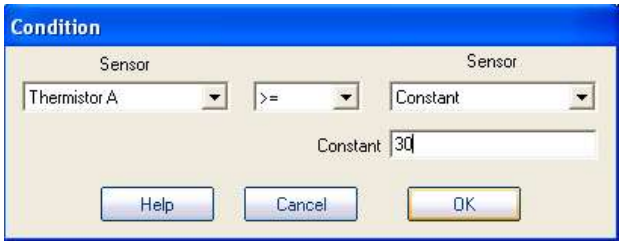


Figure 1: Edit Condition dialog box

stop when it reaches 30°C so we want the **Until** statement to continue until the value of the sensor is greater than or equal to 30°C.

In the first sensor selection box, select your temperature sensor. In the second selection box choose **>=**. In the third selection box, select **Constant**, and enter the value of 30 in the constant box. Select **OK**

Figure 3: New experiment algorithm

Congratulations! You've written your first MicroLab program! Take a look at your Experiment Steps box. It should resemble Figure 3. To try this algorithm out, press **Start** and hold the temperature probe in your hand – it should stop when the temperature hits 30 degrees.

Designing a program

This section is designed to give you an introduction in program design. The steps you put into an algorithm will determine how the program runs and how your data is taken. Ultimately, the program design affects your results.

The first step in designing a program is to ask yourself "*What am I trying to measure?*" and "*How do I want to measure it?*" You also need to account for any conditions that affect how you take a measurement. Next, make a list of the steps you would take during your experiment if there were no programming to consider. Many of the steps you would normally do, you can have the program take care of, freeing up your time up for other things you may need to do.

MicroLab has several available programming steps available that are accessed through a set of buttons. You can show the buttons by right-clicking in the Programming Steps area, and selecting Show Programming Buttons. You will see the programming buttons appear on the right of the Programming Steps area as shown below (Figure 4).

Figure 4: Programming Buttons

Program Boundaries

All programs need a way to start and stop. Starting a program is easy – press the **Start** button. There are other ways to start the taking of data within an algorithm and we will discuss these in just a bit. When to stop a program from running is one of the most important things to decide when designing a program. At what point do you want the program to stop taking data?

The common boundary commands are :

Repeat – Until : This is a useful set of program boundaries when the pair act as the first and last step of the algorithm, or even as a pair of steps in a branched algorithm. This pair is a default set of commands added every time you open a new MicroLab experiment. The **Repeat** portion tells your interface when to perform a set of steps (Figure 5). The options are to repeat the set of steps based on a time interval, when an attached counter device changes its count, or when the program receives keyboard input. The default is set to repeat an algorithm every 0.5 seconds. You can change your **Repeat** settings by double-clicking on the **Repeat** command in the **Experiment Steps** section. *Note: For the counter option to be active, a counter-type sensor must have already been added.*

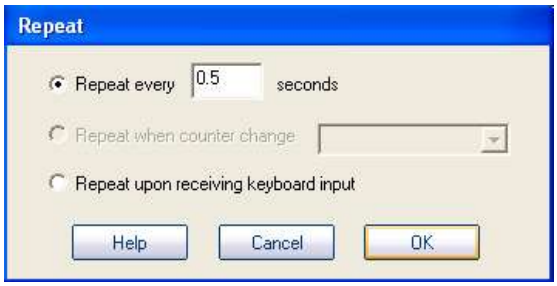


Figure 5: Repeat dialog box

After the repeat statement, a list of steps follows. When the algorithm reaches the **Until** statement it checks the condition you have set for stopping the **Repeat** command. If the condition is met, the algorithm will end, if not, the algorithm will go to the **Repeat** statement and continue running until the **Until** condition is met. There are several options for setting up the **Until** portion of the loop (Figure 6). You can access these by double-clicking on the **Until** step in the algorithm.

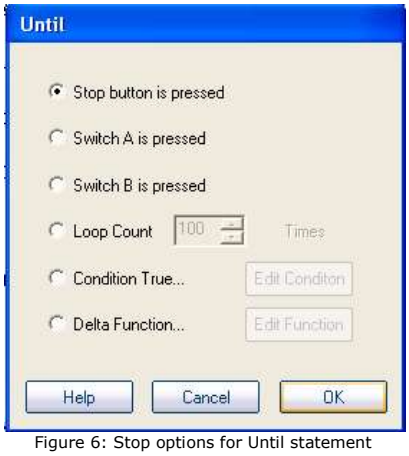


Figure 6: Stop options for Until statement

You can set up several **Repeat – Until** steps as nested loops inside an experiment's algorithm to suit the data acquisition requirements to an experiment's conditions.

Wait – Until : This is primarily a **Start** condition where you tell the algorithm to wait until a certain condition is met, then continue. Perhaps you want to take pressure readings at a certain temperature after heating a gas sample - this is the command step you would use to start taking pressure readings after the desired temperature is reached.

Figure 7: Wait Until dialog box

Decision Structures

Many programs use decision structures which give the program a direction to take dependent upon conditions. Generally, additional program steps are included inside the decision structure.

If – Then: This is a common decision structure also known as If – Else or If – Then – Else. The main purpose of this command is to give the program the opportunity to branch based on a measured value, such as temperature or time, or a manual command, such as activating buttons A or B in the software.

Here are the different options when the **If** statement is double-clicked in the programming steps (Figure 8).

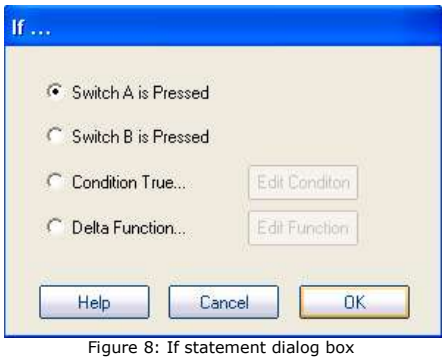


Figure 8: If statement dialog box

Switch A or Switch B is pressed (Figure 9): These are manual branching options that can be activated within the MicroLab software. The A & B soft switches are located just below the **Start** button.

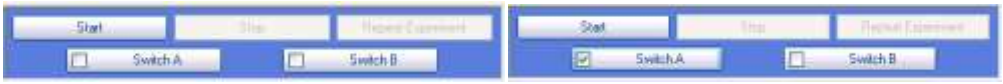


Figure 9: Manual soft switches A & B. The left figure shows both switches in an 'Off' state while the right figure shows Switch A in an 'On' state

Add a set of steps you wish the program to follow when the appropriate button is activated.

Condition True…(Figure 10): When this option is selected for your If statement, you must also set the condition for the branch. To do this, select Condition True… then click on Edit Condition.

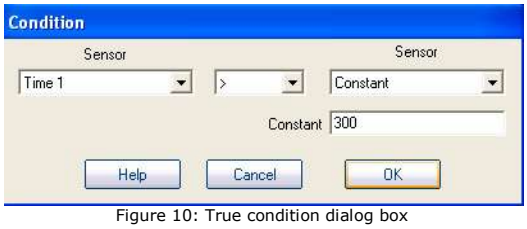


Figure 10: True condition dialog box

Here we have chosen to have something happen after a count of 300 on Timer 1. The constant will utilize the input value for whatever unit is chosen for that sensor. The timer was set up with seconds as its unit, so the If statement will branch when Timer 1 is greater than 300 seconds.

Delta Function (Figure 11): The Delta function is used when you want a branch to activate every time a sensor changes by a certain amount. If we want the algorithm to branch when the temperature changes by 2 degrees, the Delta function is set up like this:

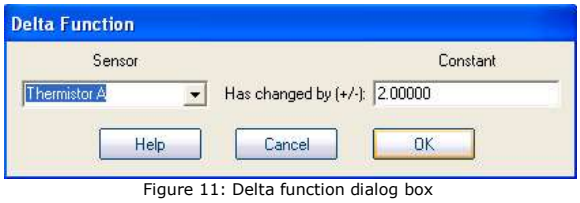


Figure 11: Delta function dialog box

Simple Steps

These are single step commands that do not branch that are placed into an experiment's data acquisition algorithm.

Message: (Figure 12) The message function allows a user to program a message to come up at a certain time during an algorithm. The message will show up in the message bar at the bottom of the MicroLab software window. Click and drag the step to where you want it in your program steps and double-click the message step to input your message to the user.

Figure 12: Message dialog box

Output: The output function allows a user to send a 5-volt (logic 1) signal out of a specified CAT5 port on the MicroLab. The most common use of this function is to serve as a logic on/off signal (Figure 13).

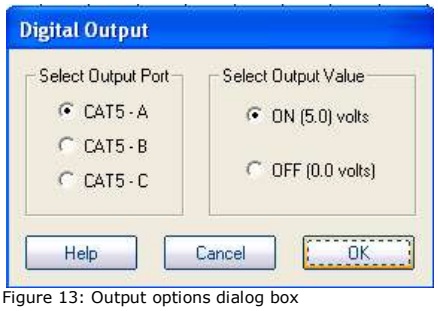


Figure 13: Output options dialog box

This function is of particular use with X10 AC controllers that can control the on/off state of any AC device such as heaters or lights.

Pause: (Figure 14) This is a useful instructional tool where an instructor can pause the data-taking algorithm and relay a message to a student to do something in the experiment. The step will show up in the algorithm as 'Pause with prompt "" ' and the user can set the displayed message by double-clicking the step. The message for the pause will show up in a dialog box in the center of the screen. Clicking on **OK** will resume the experiment's algorithm.

Figure 14: Pause dialog box

Read: This program step is already included in the base algorithm and shows up as **Read Sensors**. You may need to add this step into a loop that is outside the main algorithm where you still want to be reading data.

Timer (Figure 15): The Timer function is very useful if you want to track the time for a certain event, reset the time when a certain condition is met, or to wait until a condition is met to start the experiment (Figure 16). In order to use the Timer function, a Timer sensor must be used in the experiment, and it must also be set up for program control rather than automatic. If more than one timer is in use, this function allows you to control how each one tracks time individually.

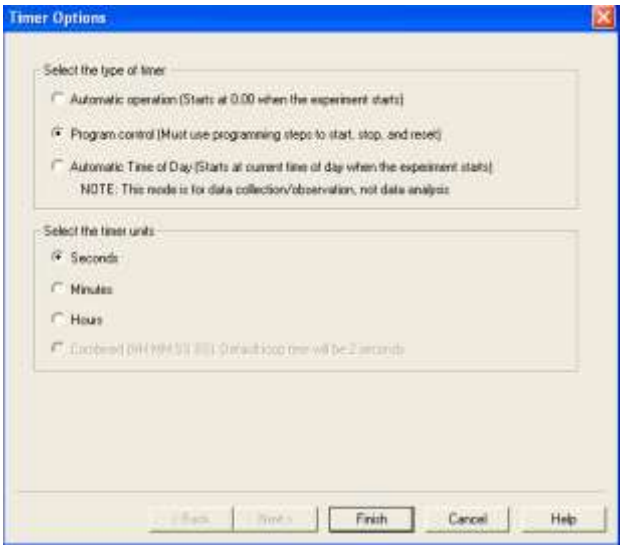


Figure 15: Timer options dialog

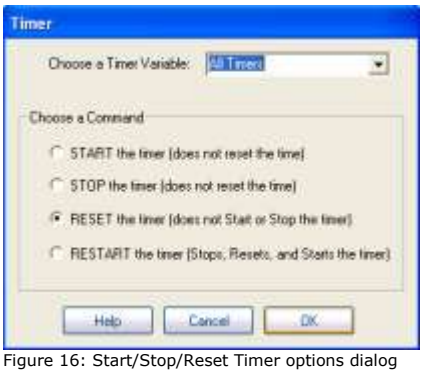


Figure 16: Start/Stop/Reset Timer options dialog

Comment: This is a non-functional step that allows the programmer to place a reminder or note to the operator about the program at a certain point.

Counter: The Counter function allows the programmer to manipulate the way that MicroLab reports values for a counter variable. The counter can be started, stopped, reset, or restarted (Figure 17)

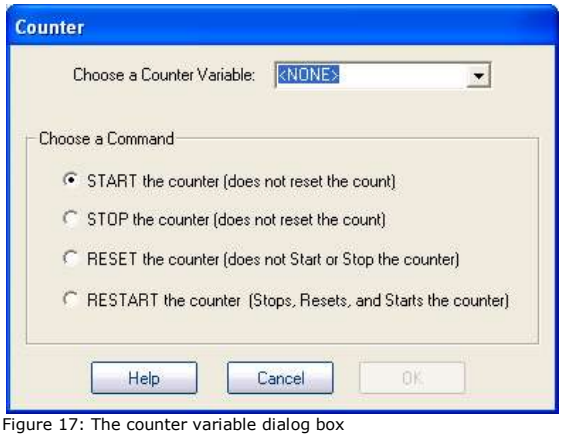


Figure 17: The counter variable dialog box

DAC: The DAC is a Digital-to-Analog Converter that the programmer can use to control a voltage output to any of the CAT5 jacks. The DAC has two modes: Bipolar mode with a -2.5V to +2.5V range (-2000mV - +2000mV); and Unipolar mode with a 0V to +5V range (0mV – 5000mV). Both modes can be controlled to the millivolt (0.001V). The DAC dialog allows the programmer to increment, decrement, or set a voltage output (Figure 18).

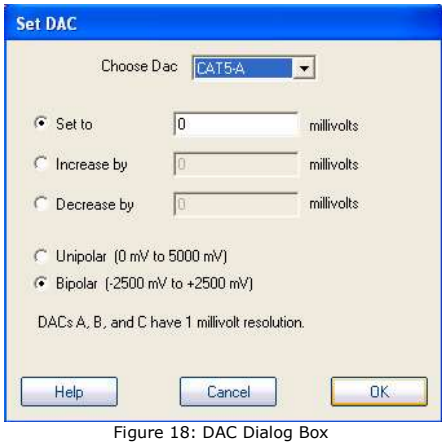


Figure 18: DAC Dialog Box

Delay: Allows the programmer to insert a delay at any point during the algorithm. The delay is set in seconds. (Figure 19)

Figure 19: Delay dialog box

Programming Examples

Constant Temperature Bath Control

A constant temperature bath is a key element in performing chemical kinetics experiments, particularly if an experiment is being carried out at different temperatures to find activation energy. A MicroLab unit and an optional X10 AC controller give an excellent way to perform such an experiment.

Hardware Set Up: A temperature sensor is plugged into CAT5-A and the transmitter portion of an AC controller pair is plugged into CAT5-B. The receiver portion of the AC controller is plugged into an AC outlet, and a heat source (typically a hot plate or cup heater) is plugged into the receiver.

For our example, we want to keep the water bath at 40°C. Before changing the experiment steps, be sure to have all the sensors involved in the experiment set up properly in the program. Now we can set up the programming steps.

The first thing to do is to right-click in the Experiment Steps area and select Show Programming Buttons. This changes the Experiment Steps area as shown in Figure 20. We see that we have the default data acquisition algorithm in place, but that will not suit our purposes here. We need to set up an If-Then statement to tell the interface what to do if the temperature is above or below our desired temperature.

Click and drag the If-Then button to the line just below the Read Sensors command in the steps. We need to edit the If statement since our branching decision will involve temperature. Double click the If statement to bring up its options. Select Condition True and click on Edit to set the condition. We want the program to branch if the temperature drops below 40°C. Change the Condition settings to look like the example in Figure 21.

Figure 20: Experiment view after adding If-Then statement

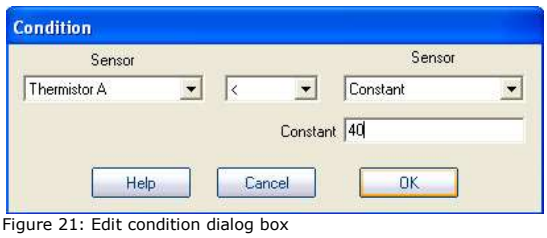
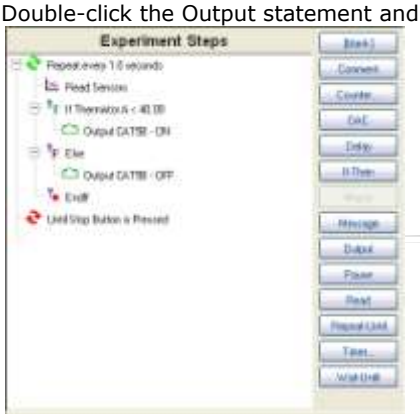


Figure 21: Edit condition dialog box

So what does the MicroLab interface need to do when the temperature drops below 40°C? It needs to turn on the heater. The AC controller pair is turned off and on by a 5-volt (logic 1) signal that the CAT5 port provides. When the program branches into the If statement we want the CAT5-B port to put out a 5V (logic 1) signal to turn the AC controller on, thus turning the heater on. We do this with an Output programming statement. Click and drag an Output statement to the line just below the If statement. Double-click the Output statement and change the output to CAT5-B and be sure the Output Value is set to On.



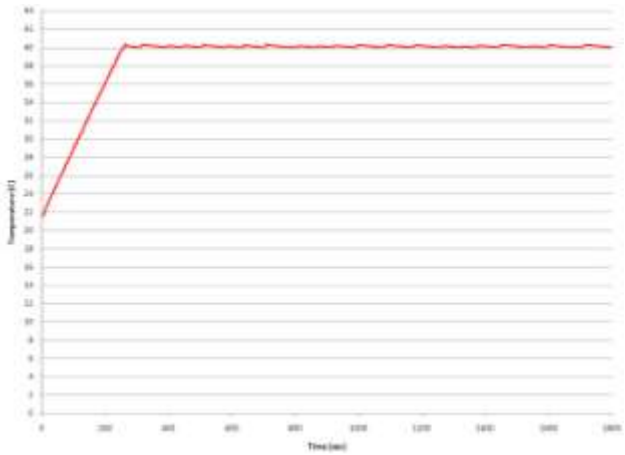
The only other possibility for the temperature is to be greater than or equal to 40°C. This is a situation where we want the heater to be off. This condition is covered by the Else statement. Insert another Output step just below the Else statement and set the CAT5-B output value to Off. This step will send a 0-volt signal

(logic 0) through the CAT5 port, which will deactivate the AC controller and the heater will not have power.

Once the water bath has reached its desired temperature, the experiment loop cycles the heater on and off to maintain that temperature. Here is a result of this algorithm with all its steps (Figure 22).

Figure 22: Experiment steps after adding On/Off conditions

The results show that the MicroLab can maintain the desired temperature within a few tenths of a degree. The parameters of the program can be adjusted to meet the specifics of your heating system.



The standard deviation for this example is 0.077 degrees over a period of 30 minutes with an average temperature of 40.11°C (Figure 23).

Figure 23 : Final temperature vs. time graph for constant temperature bath