

# CanSat Software Development

In this section, you will learn how to develop software in the BASIC language. You will learn how to use the development software tools. You will learn the basics of writing software.

When you complete this section you will have covered:

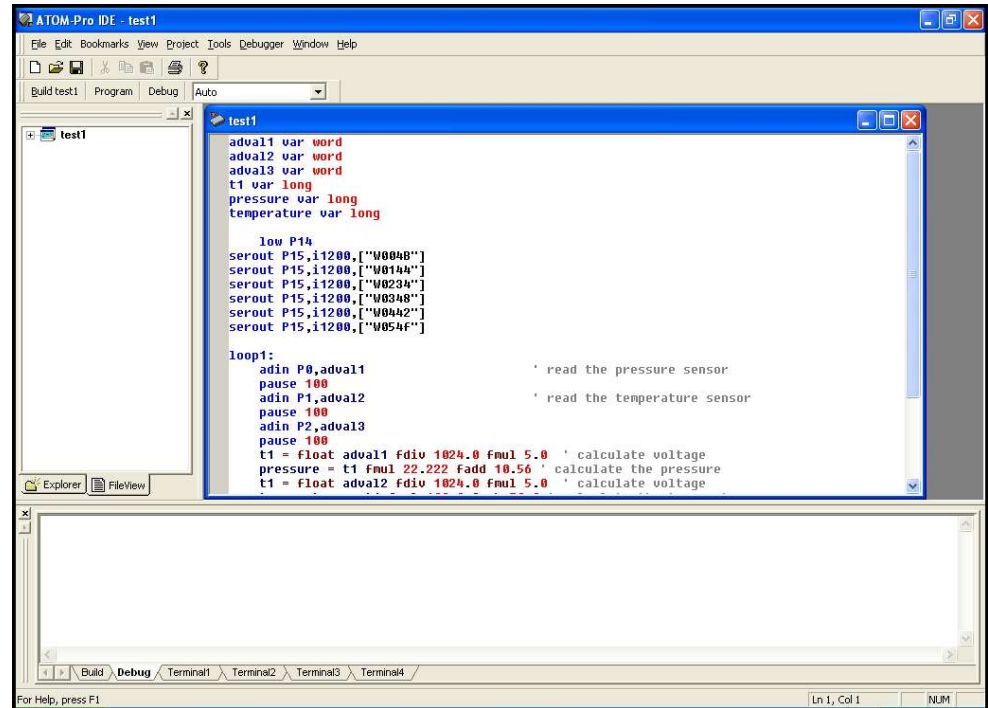
- Using the development software tools
- Program structure
- Printing to the display
- Using variables
- Loops
- Programming math equations

Requirements:

- Know how to use a windows based computer.
- Start programs.
- Use a file dialog window.
- Know where the 9 pin serial connector port is on the computer.

# Software Development Tools

- The BASICATOM PRO 24-M processor uses a BASIC interpreter for a programming language. This requires a PC with a serial port. The included development software uploads the programs into the processor using the serial cable. The serial cable connects the PC to the CanSat processor.
- Insert the CanSat CD.
- Double click on the System icon to access the CDROM. Double click on the CDROM icon.
- Double click on the the icon named BASICATOMPro. Follow the instructions to install the program.



The screenshot shows the ATOM-Pro IDE window titled 'test1'. The main editor displays a BASIC program with the following code:

```
adva11 var word
adva12 var word
adva13 var word
t1 var long
pressure var long
temperature var long

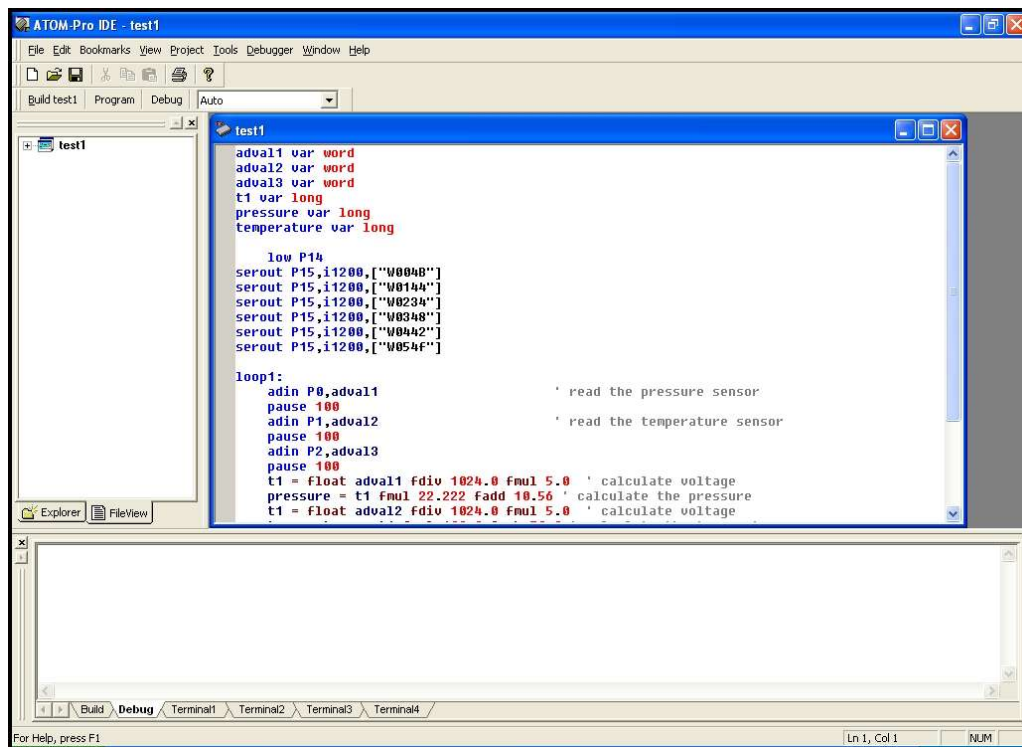
low P14
serout P15,11200,["M0048"]
serout P15,11200,["M0144"]
serout P15,11200,["M0234"]
serout P15,11200,["M0348"]
serout P15,11200,["M0442"]
serout P15,11200,["M054F"]

loop1:
  adin P0,adva11           ' read the pressure sensor
  pause 100
  adin P1,adva12           ' read the temperature sensor
  pause 100
  adin P2,adva13
  pause 100
  t1 = float adva11 fdiv 1024.0 fmul 5.0 ' calculate voltage
  pressure = t1 fmul 22.222 fadd 10.56 ' calculate the pressure
  t1 = float adva12 fdiv 1024.0 fmul 5.0 ' calculate voltage
```

The interface includes a menu bar (File, Edit, Bookmarks, View, Project, Tools, Debugger, Window, Help), a toolbar, and a status bar at the bottom with the text 'For Help, press F1' and 'Ln 1, Col 1'.

# Writing a Program

- The development software includes a program editor.
- Start the program.
- Select the menu 'Project' and select 'New'.
- A file browser will appear. Make sure it is in the C:\cansat directory.
- Select 'Basic Project'
- Specify the file name to be 'test1' and click on 'Ok'
- There will be three windows, a text editor window which is the upper right, the file list window on the left, and a status window on the bottom. The text editor window is where you will type in your program.



The screenshot shows the ATOM-Pro IDE interface. The main window displays a C program named 'test1'. The code includes variable declarations for 'adval1', 'adval2', 'adval3', 't1', 'pressure', and 'temperature'. It features a 'loop1' section with 'adin' statements for reading sensors, 'pause' statements, and calculations for 't1' and 'pressure' using 'float' and 'fdiv' functions. The program also includes 'serout' statements for serial output. The IDE interface includes a menu bar, a toolbar, a file explorer on the left, and a status bar at the bottom.

```
adval1 var word
adval2 var word
adval3 var word
t1 var long
pressure var long
temperature var long

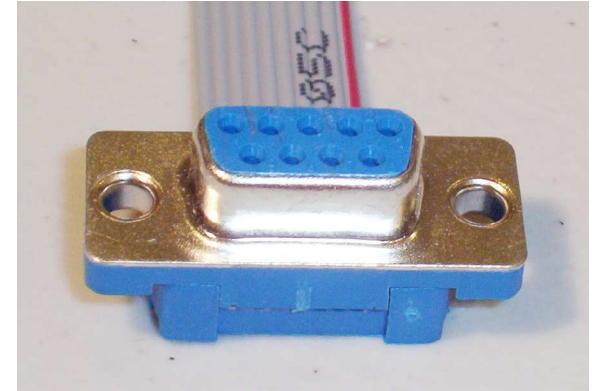
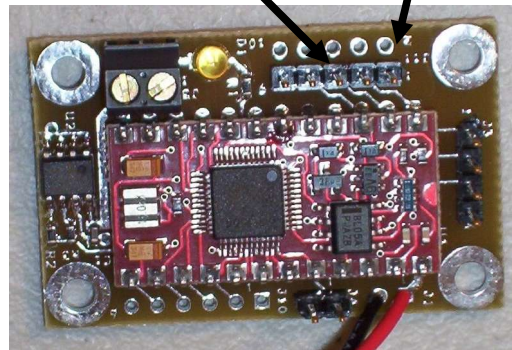
low P14
serout P15,11200,["W0040"]
serout P15,11200,["W0140"]
serout P15,11200,["W0234"]
serout P15,11200,["W0348"]
serout P15,11200,["W0442"]
serout P15,11200,["W054F"]

loop1:
  adin P0,adval1          ' read the pressure sensor
  pause 100
  adin P1,adval2          ' read the temperature sensor
  pause 100
  adin P2,adval3
  pause 100
  t1 = float adval1 fdiv 1024.0 fmul 5.0 ' calculate voltage
  pressure = t1 fmul 22.222 fadd 10.56 ' calculate the pressure
  t1 = float adval2 fdiv 1024.0 fmul 5.0 ' calculate voltage
```

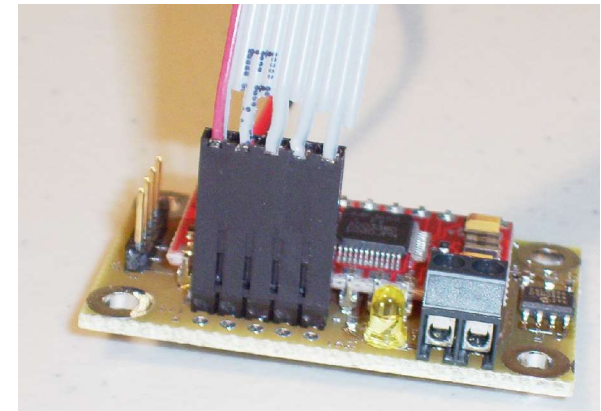
# Connecting the Electronics

- A nine pin serial cable is needed to connect the cansat electronics to the computer so your program can be uploaded. Plug the 9 pin D connector to the serial port on the computer.
- Connect the 5 pin connector to the cansat electronics board. The cable has a red line on one edge. This indicates pin 1. Make sure this is on the same side as the '1' on the circuit board.
- Plug in the 9 volt battery. Make sure you don't try connecting the battery in backwards.

Serial port      Pin 1



9-pin D connector that plugs in computer



5-pin connector that plugs in CanSat processor board

# Program Structure

The structure of the program is pretty straight forward. At the top of the program, all variables must be declared. After the variables are the instructions. Below is an example.

```
A var byte      ' this is where variables are declared  
b var byte
```

```
a = 3          ' this is where the instructions go  
b = 4  
debug [dec a," ", dec b]  
end
```

You will notice that two of the lines have an apostrophe in it with text afterwards. This is called a comment. Anything to the right of the apostrophe is not considered an instruction. This allows notes to be included in the software to help in programming.

# Writing the First Program

- The first program will print text to the computer.
- In the editor, enter the following:

**Debug [“Hello, this is my first program”]**

**End**

- Once entered, click on the Debug button above the editor window. This will compile the program and load it into the processor. The debugger program will start. This allows you to run your program and see the results.
- If an error is indicated in the bottom status window, look in the editor window for any misstyping.
- Open the menu “Debugger” and select “Run”. This will start the program. In the bottom window, you should see the output appear.

**Hello, this is my first program**

- Congratulations, you ran your first program. Select “Disconnect” in the Debugger menu. This exits the debugger and returns to the editor.

# Printing some more

- Add another line to the top of the program as shown below.

**Debug ["Hello, this is the first line."]**

**Debug ["This is the second line"]**

**End**

- Activate the debugger and run the program. You should see that both print statements print on the same line one after the other. To make them print on separate lines change the first statement to the following:

**Debug ["Hello, this is the first line.",13]**

**Debug ["This is the second line"]**

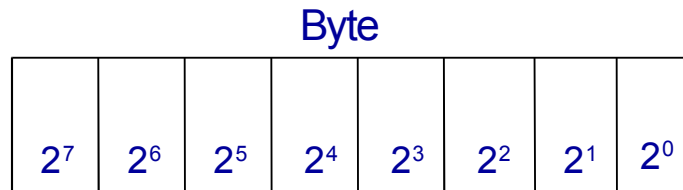
**End**

- Run the program and see what happens. The two statements should be on separate lines. The number 13 is an ascii value for carriage return. The display interprets the number as move the cursor down to the next line.

# Understanding Bits

To understand variables which will be discussed soon, bits need to be understood. Computers use bits in many ways. A bit is a single piece of information that can be at two states, a logic level 1 and a logic level 0. The state is called a binary state which means a bit can only be any of two states. Not much can be done with a single bit. To be useful, the computer can group a number of bits together to represent numbers or other information. There are standard groupings:

nibble is four bits together  
byte is eight bits together  
word is 16 bits together  
long is 32 bits together



There is an order to the bits and a weighting scheme so that numbers can be represented. Let's take a byte. It has eight bits. As shown, the least significant bit has a weight of zero. To calculate the value, calculate 2 to the power of the digit which is zero. The result is one. Take the next bit. It has a weight of 1.  $2^1$  is equal to 2. The next bit is  $2^2$  which is equal to 4.  $2^7$  equals 128. To figure out the value of the byte, add up all the weights for the bits that equal 1.

For example, the byte "10111001" equals 185.  $2^7 + 2^5 + 2^4 + 2^3 + 2^1 = 185$ .

# Using Variables

- Now, it is time to learn about variables. Variables are used to store temporary information in memory during program execution. There are five different types of variables. Each is used to contain a certain size value.
  - \_ Bit - defines a variable to be a single digital bit in size with a value of 1 or 0.
  - \_ Nib - defines a four bit variable that can have a range from 0 to 15.
  - \_ Byte - defines an eight bit variable that can have a range from 0 to 255.
  - \_ Word - defines a 16-bit variable that can have a range from 0 to 65535.
  - \_ Long - defines a 32-bit variable that can have a range from 0 to 4294967295.
- Variables are declared with the first character being a letter.

**dog var byte** ‘ this declares variable dog to be a byte.

**t34 var word** ‘ this declares variable t34 to be a word.

# How to Print Variables

- The Debug command is used to print information to the display. It is only used during testing of the software. When the CanSat is performing its mission, there is no need to print information.
- To print variables, use the following:

**a var byte**

**a = 78**

**Debug [ dec a] ‘ print variable a as a decimal number**

- The above prints the value of variable a as a decimal number. The modifier 'dec' specifies the value to be shown as a decimal integer number. Other modifiers include 'hex', 'bin', and 'real'.
  - Hex - print the variable in hexadecimal format.
  - Bin - print the variable in binary format.
  - Real - print the variable in floating point format.
- Write a new program that sets an integer variable to the value of 78. Print it in all the modifier formats.

# Printing Multiple Variables

- Printing multiple variables using one Debug command is simple. Each variable is separated by a comma. Try the following program:

**A var byte**

**B var byte**

**A = 5**

**B = 8**

**Debug [dec a, dec b]**

**End**

- Run the program and see what happens. The numbers should be next to each other. Now change the Debug command to the following:

**Debug [dec a," ",dec b]**

- The above command inserted a space in between the numbers. You can see the Debug command can print any variable and statements at one time.

# Jumping Around

- To jump to different parts of the program, there is the goto statement. To specify the location to jump to, a label needs to be inserted.

**Debug ["Print the next statement continuously",13]**

**Loop1: <<— This is the label**

**Debug ["Hello world",13]**

**Goto loop1**

**End**

- Remember entering the above program. The label is inserted where you want the program to jump to. This example will print the “Hello world” statement for ever.

# Control the Light

- Now it is time to learn how to use the digital control ports. A light emitting diode or LED is included on the CanSat electronics board. It is a semiconductor device that emits light when electricity flows through it.
- The LED is connected to pin fourteen or P14.
- To turn the LED off, use the command:

## **High P14**

- To turn the LED on, use the command:

## **Low P14**

- That is all there is to controlling the digital ports. When you set the port with the high command, you are setting the port signal to a high level which is a logic level 1 which is 5 volts.
- When you set the port low, you are setting the port signal to a low level which is a logic level 0 which is 0 volts.

# More Light Control

- Now, let's write a program that will continuously flash the LED on and off. To do this, the program has to run in a loop. Loops are easy to set up. First a label is used to indicate the start location of the loop. Then later in the program is the goto statement that specifies the label for the program to go to. Enter and run the following example:

**Debug ["flashing LED"]**

**Loop1:**

**High P14**

**Low P14**

**Goto loop1**

**End**

- The program will flash the LED as fast as it can execute the program. You probably cannot see the flashing. That can be fixed next. Exit the debugger and go on to the next page.

# Adding Delay

- There is a command called pause. The pause command tells the processor to stop operating for a period of time. The pause command will stop the processor in one millisecond increments. That is 1/1000<sup>th</sup> of a second. Insert the pause command in program as shown:

**Debug [“flashing LED”]**

**Loop1:**

**High P14**

**Pause 1000**

**Low P14**

**Pause 1000**

**Goto loop1**

**End**

# Delays

- The program should have turned the LED on for one second and turned it off for one second. Experiment and change the pause time to different values.

# Doing Math

- Performing mathematical functions is pretty simple. It is like writing equations. Try the following in a new program.

**A var byte**

**B var byte**

**C var byte**

**A = 5**

**B = 6**

**C = a + b**

**Debug ["The answer is: ",dec c]**

**end**

# Floating Point Math

- Up to now, any numbers and math were based on integer numbers which are whole numbers. Floating point numbers allows the processor to work with fractional numbers.
- The floating point format uses 32-bit variables. To use floating point, declare the variables as long. The following example shows how it works:

**Temp var long**

**Temp = 1.543**

**Debug [real temp]**

**End**

- Temp is declared to be along integer just so that there is enough bits to hold a floating point value. To set the variable to a floating point value, the number must have a decimal point such as 1.0, 2.435. The BASIC interpreter needs the decimal point to recognize the number being floating point. This is a requirement of the language.

# Floating Point Math

- Writing math operations is a little different than with integer math.
  - To add, use fadd
  - To subtract, use fsub
  - To multiply, use fmul
  - To divide, use fdiv
- Below are examples of how to use the math functions. They are similar to the earlier math except the operation is spelled out.

**Temp1 var long**

**Temp2 var long**

**Temp1 = 3.4156**

**Temp2 = temp1 fmul 2.345 ' multiply temp1 with 2.345**

**Temp2 = temp1 fadd 3.234 ' add temp1 and 3.234**

**Temp2 = temp1 fdiv 5.0 ' divide temp1 by 5.0**

**Temp2 = temp1 fsub 34.0 ' subtract 34.0 from temp1**

- This is a little strange but that is just how the BASIC processor works.

# Working with Floating Point

- **Write a program to add two floating point numbers together and store the result into a variable and print the result. Assign the two floating point numbers to variables. Remember that all variables must be declared before any instructions. Refer to the previous two pages to help in writing the program.**

# Summary

- **In this section, you should be able to write simple programs. You should have an understanding of:**
  - **How to declare and use variables**
  - **Write program loops**
  - **Do math operations on variables**
  - **Do floating point operations on variables**