This is the published version:


Available from Deakin Research Online:

http://hdl.handle.net/10536/DRO/DU:30005135

Reproduced with the kind permission of the copyright owner.

Flowchart and Kernel Software for Modular Robot

Matthew. A. Joordens
School of Engineering and Technology
Deakin University
Waurn Ponds, Vic, 3216, Australia
majoor@deakin.edu.au

Abstract

Flowcharting is a common method of setting out the requirements for a piece of code. It is simple with few rules to follow. Rarely however, is it used as the code itself. This paper describes the outline of a software package that uses the flowchart as the code for a small, autonomous, modular robot, designed for use in High Schools. It also describes the code used by the robot to complement the flowchart software creating a system that can be used by students and their teachers to design, build and program a robot without previous programming experience.

Keywords

Robotics, Education, Modular, Flowchart, Programming

1 INTRODUCTION

During a project the author had at a high school that involved enabling a small group of students to build a robot that could move over a piece of paper, drawing as it when, a discussion developed between the author and the Teachers. The point of the discussion was that the Teachers were looking for a robot system that could take the students beyond the Lego “Mindstorms”® robot. The result was a small, autonomous, modular robot that the students could plug together in any configuration.

One hurdle in this robot system was the programming of the robot. The teachers did not want a programming language that would take long for the teachers to learn and so it was decided to use flowcharting, which the teachers had experience with, as the language.

2 Background

The robot needed a kernel program. This kernel had two functions; to download the students programs and to run the students programs. The first function to tackle was the download code.

2.1 Kernel Download Code

The microcontroller chosen for was the 68HC08. The version that was used had 1K of RAM, 512Bytes of EEPROM and 32K of Flash.[1] This version was selected for the large amount of Flash on board, allowing for more program space than any student is likely to need. The student’s program was to reside at the beginning of the Flash memory.

Now in order for code to write data to Flash, that code must not be run from Flash. For this reason the download code was placed in the EEPROM. It was designed to accept S19 data from the micro controller’s serial port and to program it to the beginning of the Flash memory. This code was squeezed into the EEPROM with 10 bytes to spare!

The decision to accept S19 data, even though it was harder to fit into the limited EEPROM was so the download code and the robot could be tested with the S19 data generated by most compilers for the 68HC08. Without this, the author would have had to find some other way to generate the data in an acceptable format.
2.2 Kernel Runtime Code

Once the student’s program has been downloaded, it needs to be run. In order to simplify the student’s program, it was decided to limit the program to reading from and writing to memory location. The runtime kernel would take care of the hardware.

The first consideration was the motors. The robot could use either DC or servomotors and those motors were to be controlled with Pulse Width Modulation (PWM). A servomotor likes its PWM signal to have a period of 20ms, so an interrupt that would occur every 20ms was set up.

The first function of this interrupt was to service the motors. It would adjust the width of the signal based on memory locations set by the student’s program.

Its next task was to increment the time memory locations so that the student’s program may keep track of the time or create timed delays.

The third task was to update all the outputs based on memory locations set by the student’s program and take an image of all the inputs. All inputs and outputs are on a pseudo bus controlled by lines on three of the micro controllers I/O bits.

Its last task was to send a few bytes of relevant data back through the serial port so that the student’s program may diagnose any problems in the program.

Thus the interrupt routine handles all the hardware every 20ms while the student’s program runs in the foreground manipulating the memory.

3 The Modular Robot

The robot itself comprises of a system of modules, all based on single circuit boards. The first module is the micro controller board. This module contains the microcontroller, a serial port and several connections for the motor bus and the I/O bus.

Figure 3 The modules of the robot.

The next module is the power module. It uses 4 AA batteries or a 6V external supply to power the robot. This module also has several motor and I/O connectors. The micro controller and the power modules form the nucleus of the robot. All other modules are built up around these two modules.

The basic robot comes with one motor module. This module contains two for the 4 possible motors that the robot can use. This module drives two independent wheels to give the robot mobility.

Figure 2 Exploded view of the Modular Robot

501
There are several modules that contain the different sensors such as infrared, Light dependent resistors and micro switches. Each module has one connector for the I/O bus and one rotary micro switch to allow the sensor module to be set to one of the possible 16 addresses. There are several generic I/O modules, either digital or analogue, to allow students to create there own sensors.

The final modules are extension modules. These modules contain many I/O bus connectors that allow the sensors to be connected to the micro controller module, via the extension modules, in many different orientations. All I/O connectors are built with redundancy so that the sensor modules may be inserted in different orientations and they will still work.

4 Flowchart Software

With the robot programming complete, the interface software was required. It needed to be both simple and effective for use by high school students.

In discussion with high school teachers it transpired that most of the students did not have any programming experience by they did have experience with Lego's 'Mindstorm'® iconic system. This was a flowchart like system with different icons for specialised tasks.

As the Flowcharts were to be a stage that followed the Lego Mindstorms icon based system, the author looked in this area and found the "Chart Programming Language".[2] This is a flowchart based system. Unfortunately on a more detailed examination is was found that the system allowed the student to write a flowchart then to write the code from the flowchart. Scratch the first option.

Dr. Tia Watts from the Sonoma State University is working on a flowchart editor.[3] This is just a GUI flowchart editor, but could be turned into a compiler with a little more work. It is very comprehensive and may be a little to detail for the job at hand.

Matrix Multimedia has a very promising package called "Flowcode".[4] This package allows students to design complex electronic system straight from the Flowchart. Its output is used to program a PIC microprocessor. It is however, very detailed, found take a while to learn and a different processor had already been chosen for the modular robot.

4.1 Inhouse Software

In the end, it was decided that the author would write this software. This gave the author full control over the code and no licensing issues. As the author also had to write the code that resided in the robot, it was easy to ensure compatibility and to make the job easier. This software was written using Microsoft's Visual C++ v6.0® as a multi-document application. This software was broken into four sections; Coding, compiling, downloading and diagnostic.

4.2 Coding

The decision was made to use a simple flowchart, with just the basic flowchart blocks as the programming language. There would be three basic blocks used in the flowchart; a process block, a decision block and a run block.

4.2.1 Process Block

The process block is the workhorse. This is where the action takes place. Not that much action is needed. That's to the nature of the kernel code in the robot, all the students program needs to do is read and write to different memory locations and to manipulate the data in those memory locations.

The process block then is set up to allow the data in one memory location be operated on with data in another memory location or with immediate data and written to a memory location. The available operations are; equal, invert, add, subtract, multiply, divide, modulus, AND, OR and XOR. This can be seen in the following figure.

Figure 4 The Process Dialog box.

All memory locations, including the memory locations for the 16 I/O sensors, are given a name that can be change by the student to allow the flowchart to be easier to read.
4.2.2 The Decision Block

To make a decision, this block allows the data in one memory location to be compared in another memory location or immediate data. Thus two sensors can be compared against each other or a sensor can be compared with a threshold value. The comparisons available are: equals, not equal, greater than, greater than or equal, less than, and less than or equal as can be seen in the following figure.

![Figure 5 The Decision Dialog box](image)

To create a flowchart these two blocks, process and decision, are all that are needed as seen in the following example.

![Figure 6 The software with a flowchart example](image)

4.2.3 The Run Block

Since flowcharts based the on last two simple blocks can get very large and messy a third block, the run block is required. This block simply gives the location in the flowchart in which to run another flowchart. This means that flowcharts can be written as subroutines to be used in other flowcharts. Thus the students can create their own library of functions. As the next example illustrates.

![Figure 7 Example of flowchart calling another flowchart](image)

To show the flow of the chart, simple lines connecting the blocks are used. There are no flow arrows on the lines. Lines that enter the top of a block are in flowing and lines from the bottom or sides of a block are out flowing.

The data for all the different blocks are stored in a dynamic linked list.

4.3 Compiling

With the defined structure of the flowchart and the limited blocks, compiling is relatively easy. The flowchart is compiled in a two-pass process.

4.3.1 First Pass

Each block is compiled on its own. The machine code has already been figured out for each block and each operation within each block. Thus when compiling a block the correct machine code is placed in an array that represents the Flash RAM and the addresses of the memory locations specified in the block, or the immediate data, is inserted.

Each block ends with a jump to the next block. To facilitate this, each block is given a unique ID based on the time it was created. A link list of the addresses of each block is created during the first pass.
The run block is interpreted as a jump to the first block of the new flowchart. The file containing the flowchart is opened and the blocks within it are appended to the linked list of blocks currently being compiled.

4.3.2 Second Pass

After the first pass the software searches the compiled code for the jumps and, searching the block address linked list, inserts the correct addresses.

4.4 Downloading

Once the flowchart is compiled the software converts the machine code into the S19 record and then uses the ActiveX Microsoft Communication Control® to download the code, via a serial port, to the robot.

The serial port was chosen over USB as all high schools will have serial port on their PC’s while only the most up to date schools will have PCs with USB. In the worst case, a school with only USB can get USB to serial converters very cost effectively.

While serial communications is more complex under Windows 95 and later, it also has a greater capability.[5] Or does it?

The biggest problem the author had in writing this software was with the Serial port communications. The creators of the MSComm routines wrote these routines for Visual Basic and Microsoft converted them for Visual C++. This means that there is very little support and documentation for them and the author found that a lot of trial and error was required to get these routines to work correctly. For example, the comm. Port likes to shut itself down and so the software must regularly check if the port is closed and reopen it.

The MSComm control is an interrupt driven system set up to interrupt on every character received and when ready to transmit. The transmit interrupt was ignored as the MSComm Control has a function to transmit strings of characters. Thus each line of the S19 record was sent with this function.

4.5 Diagnostic

Once the program is downloaded and the robot is running, the software can report back on the status of all the memory locations that are available to the students program. The robot, over about a third of a second, sends out on the serial line, seven set characters followed by the data from the 76 available memory locations.

The software is interrupted on each character received. At this point the software stores each character in an array. When the software sees the seventh set character it resets the array index to 0 and increments the index on all other characters. This array is used to display all the data in a dialog box so that the student may use the data to diagnose any problems in the program.

5 CONCLUSION

The flowchart software together with the modular robot creates the basis of a system that allows high school students to design, build and program a robot. The tasks can be quite varied and the student does not need any programming experience. Just important, the teacher does not need programming knowledge either.

The author wishes to acknowledge the Department of Education and Training, Victoria, Australia, for without their support this project would not have been possible.

REFERENCES