

E-blocks general teachers' notes

EB772

V1.0

© Copyright Matrix Multimedia Limited 2006

Contents

| | | |
|------|--|----|
| 1. | Introduction | 2 |
| 1.1. | Resources | 2 |
| | E-blocks | 2 |
| | Actuators panel | 2 |
| | PICmicro buggy | 2 |
| | Flowcode | 2 |
| | C for PICmicro microcontrollers/Assembly for PICmicro MCUs | 2 |
| | ELSAM | 2 |
| 1.2. | Document aims | 2 |
| 2. | Getting started | 3 |
| 2.1. | Installing Flowcode 2.1 | 3 |
| 2.2. | Updating PPP and USB Multiprogrammer drivers | 4 |
| 2.3. | Installing C for PICmicro microcontrollers V3 | 4 |
| 2.4. | Installing Assembly for PICmicro microcontrollers V3 | 5 |
| 3. | Understanding all of the equipment | 6 |
| 3.1. | Where are the manuals? | 6 |
| | Flowcode | 6 |
| | E-blocks and hardware | 6 |
| | C for PICmicro microcontrollers | 6 |
| 4. | Knowing what to do if you get stuck | 6 |
| 5. | Using E-blocks equipment for teaching and learning | 8 |
| 5.1. | Our goal | 8 |
| 5.2. | Flowcode before C and Assembly | 8 |
| 5.3. | A learning framework | 8 |
| 5.4. | Crystals, PICmicros and settings | 9 |
| 6. | Worksheets | 10 |

1. Introduction

Thank you for purchasing some Matrix Multimedia products. This brief introduction will give you guidance on how to use the pack with your students. This document is available in Word form on request.

1.1. Resources

To make the most of the tutorials here you will need the resources listed below.

E-blocks

A set of hardware resources that are used for PICmicro programming and development work. This could be made up from the E-blocks range of hardware or could be based on a Version 3 PICmicro microcontroller development board.

Actuators panel

A servo, stepper, and DC motor with feedback which is used to help to understand motor control.

PICmicro buggy

A small robot buggy which is used to understand how robots are controlled.

Flowcode

A CD ROM containing software for programming PICmicros in flow charts, as well as a course in Flowchart programming.

C for PICmicro microcontrollers/Assembly for PICmicro MCUs

A CD ROM containing a complete C compiler and IDE for programming PICmicros in C as well as a course in C programming. The tutorials here can be carried out with Flowcode, C or assembly code.

ELSAM

This is a small CD ROM containing programming utilities and hardware datasheets. ELSAM is shipped with all Matrix Multimedia 'upstream' boards: programming boards and some application boards. The contents are currently also free on www.matrixmultimedia.com.

1.2. Document aims

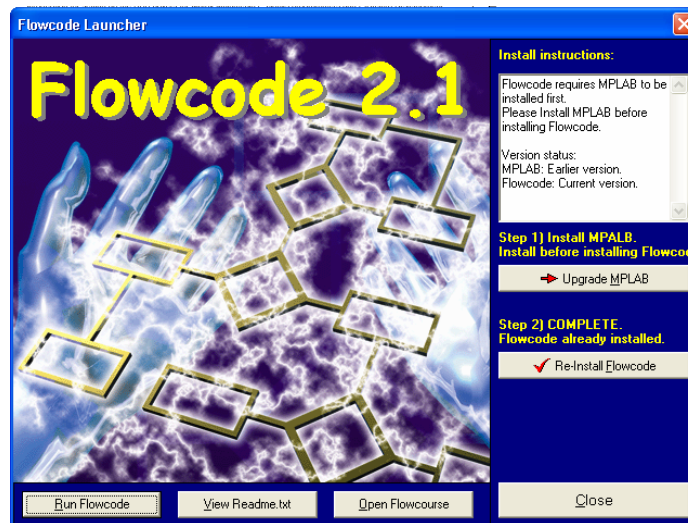
In this document we want to address 4 issues:

- How to get started
 - Installing Flowcode
 - Updating PPP and USB Multiprogrammer drivers
 - Installing C for PICmicros or ASM for PICmicros
- Understanding all of the equipment
 - Where are the manuals?
 - What is in my pack?
 - Using the equipment for teaching and learning
- Knowing what to do when you get stuck
- How do you use all this equipment in the classroom?

2. Getting started

2.1. Installing Flowcode 2.1

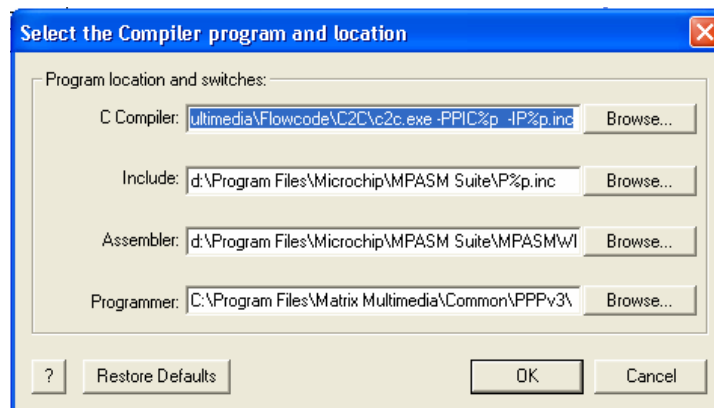
We suggest that you get started by installing Flowcode and programming a PICmicro for the first time. When you insert Flowcode into the CD ROM drive you will get a screen like this:



You should follow Step 1 – install / update MPLAB - and then step 2 – install/update Flowcode. MPLAB is a software utility supplied by Microchip – the makers of the PICmicro devices – which assembles Flowcode programs into a format the chips can understand.

Note: If you have already installed an earlier version of MPLAB for some other program you may wish to update your installation to the version. The version currently supplied with Flowcode 2.1 is V7.01, but check the CD for details. You can also check the Microchip website <http://www.microchip.com> for the latest version of MPLAB.

If you have MPLAB version 7.01 or later installed you can skip the Install MPLAB option. Flowcode will check for MPLAB when it is installed. However if there are problems you can check the MPLAB links in Flowcode To check the links open Flowcode and select COMPILER OPTIONS from the PIC menu:

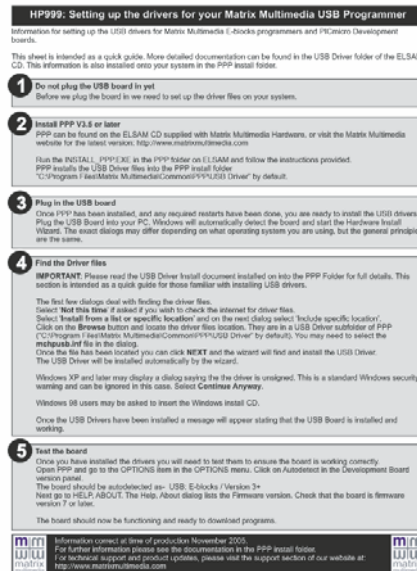


Check that the file paths are correct, and that the command line parameters match those detailed in the Flowcode help file (Press the ? button to access the help file).

2.2. Updating PPP and USB Multiprogrammer drivers

Flowcode, 'Assembler for PICmicro microcontrollers' and 'C for PICmicro microcontrollers' rely on a software utility called 'PPP' to send your programs into the PICmicro Multiprogrammer hardware. Because the range of PICmicro devices is always being updated the PPP software also changes from time to time. You will find a fairly up-to-date version of PPP on the ELSAM CD ROM supplied in your kit. The latest version can always be found on the Matrix Multimedia web site – <http://www.matrixmultimedia.com> if you wish to download it.

You should follow the instructions in Document HP999 which will allow you to set up PPP and the USB drivers for your Multiprogrammer. Document HP999 is supplied in the papers you received with your kit. HP999 looks something like this:



2.3. Installing C for PICmicro microcontrollers V3

Follow the instructions for installing C for PICmicro Microcontrollers. You will need to enter a licence key which is on the inside of the jewel case. Microchips MPLAB and the SourceBoost C Code compiler are installed as part of C for PICmicro. SourceBoost can be launched from C for PICmicro, or independently from the SourceBoost program group.

Note: MPLAB V7.01 or later is required. You may need to check the C compiler settings for MPLAB in SourceBoost by selecting SETTINGS...OPTIONS...TOOLS and checking that the program paths are correct, and that the command parameters match those in the C for PICmicros *Compiling and running C programs* chapter.

Note that the course in C for PICmicro microcontrollers uses a Java menu which your browser security system may flag up as being a security risk. You can prevent this from reoccurring by altering the security settings in your system. Alternatively click on the message and 'Allow blocked content'

2.4. Installing Assembly for PICmicro microcontrollers V3

Follow the instructions for installing Assembly for PICmicro Microcontrollers. You will need to enter a licence key which is on the inside of the jewel case. Microchip's MPLAB is installed for you.

Note that the course in Assembly for PICmicro microcontrollers uses a Java menu which your browser security system may flag up as being a security risk. You can prevent this from reoccurring by altering the security settings in your system. Alternatively click on the message and 'Allow blocked content'

3. Understanding all of the equipment

3.1. Where are the manuals?

There is a large amount of equipment in you pack. The information on all this equipment is supplied electronically in a modular format – mostly on CD ROM. Because items are continually being updated then you can find the latest datasheets on our web site. If you don't find something you want then please email us and we will do our best to provide it for you.

Flowcode

- The marketing datasheet is currently at www.matrixmultimedia.com/datasheets/TEFLC-60-2.pdf
- You will also find free software updates and component plug-ins like CAN bus, SPI, EEPROM Web server and more at: www.matrixmultimedia.com/flowcode.php
- There is no manual for Flowcode – all the relevant information is included in the Help file that is shipped with Flowcode
- There are 30 tutorial files shipped with Flowcode which demonstrate how the software is used. If you choose the default installation options then these will be installed in C:\PROGRAM FILES\MATRIX MULTIMEDIA\FLOWCODE\TUTORIALS
- There is a complete on-screen course in Flowcode programming on the Flowcode CD ROM. This is not installed with Flowcode but can be found in the FLOWCOURSE directory on the CD ROM and copied to your hard drive if necessary.
- Lastly there are two online videos explaining the basic operation of Flowcode which can be seen at www.matrixmultimedia.com.

E-blocks and hardware

The technical datasheets for E-blocks are on your ELSAM CD ROM and up-to-date versions are in the E-blocks members area at <http://www.matrixmultimedia.com/eblocks/index.php> username is 'eblocks', password is 'halifax'. Note lower case letters. Technical datasheets include circuit diagrams. Datasheets on the Actuators panel and the buggy are on the ELSAM CD ROM.

A user guide to the E-blocks system is available on your ELSAM CD ROM and in the E-blocks members' area at www.matrixmultimedia.com.

C for PICmicro microcontrollers

C for PICmicro microcontrollers contains a full C compiler called 'SourceBoost'. This has its own help file that is accessed from within SourceBoost itself.

4. Knowing what to do if you get stuck

We have a set of user forums on our web site at <http://www.matrixmultimedia.com> that contain answers to many common problems. You can post technical support question there, and also general questions about PICmicro programming, Flowcode and C for PICmicros etc. You can also post project and code related questions to see if anyone can help you with your difficulties.

Should you require assistance with technical issues you can contact our support team at support@matrixmultimedia.co.uk

Note that technical support is only available 9-5 Monday to Friday GMT (excluding holidays). Whilst we do endeavour to sort out problems as fast as possible please note that it may take some time to respond or look into issues.

NOTE: Technical support via email is for technical and installation related issues only. For Code and project questions please visit our forums.

5. Using E-blocks equipment for teaching and learning

5.1. Our goal

With E-blocks, the Version 3 development board, Flowcode, and our C and Assembly CD ROMs, our goal is to provide you with a flexible set of resources that can be used for learning about microcontrollers, at high or low level, and for project work. We do not provide a rigorous framework for learning that students must follow sequentially. The learning resources provided are open architecture and will need an additional framework (probably just worksheets) to provide students with a goal and learning objectives.

The bulk of our learning materials are supplied on CD ROM, or online, that students can access on a computer screen. Note that networkable versions of all software are available if you need to allow more than one student to access the learning materials. The courses on the C for PICmicro microcontrollers CD ROM, Assembly for PICmicro microcontrollers CD ROM and on Flowcode include exercises that are designed to test students' understanding of programming as they work through the material on each course. The contents list of the tutorials on the CD ROMs effectively dictates the 'curriculum' that the students undertake, but teachers will need to review this material to decide how much of it their students should undertake, and may want to provide additional exercises for assessment.

The time taken for students to go through these courses is dictated by their academic ability, and their flair for programming. The latter is the most important factor here: some students pick up programming very quickly, and other struggle, with age not being an effective arbiter of programming ability. However as a point of reference you can assume that Flowcode will take a student around 20 hours of learning, C or Assembly for PICmicro microcontrollers will take around 30 hours. C and/or Assembly for PICmicro microcontrollers are used by university undergraduates as the backbone of a 60 hour module, with a 30 hour project making up the shortfall.

5.2. Flowcode before C and Assembly

It is our intention that Flowcode and Flowcourse are used as a precursor to learning to program microcontrollers in C or assembly. Flowcode provides a highly effective method of introducing students to the concepts of programming and the concepts of developing circuits and systems that are based on microcontroller technology. Flowcode allows students to learn about microcontrollers and programming techniques without getting bogged down in the details of syntax. Introducing students to the concepts of microcontrollers and microcontroller programming before introducing them to the complexities of, say, C programming has a definite advantage: the techniques of programming are largely independent of the programming language used., and if we can teach about the concepts of programming independently of the syntax and complexities of C programming then students will find the learning process easier.

5.3. A learning framework

The flexibility of the learning resources provided has the disadvantage that there potentially no overall structure to a course. In order to rectify this we have included 14 task based worksheets with this pack that are as follows:

1. Simple input and output sequences
2. Loops and counters
3. Making decisions
4. Using subroutines and macros
5. A/D conversion
6. Building a PICmicro circuit
7. Using interrupts
8. A burglar alarm
9. DC motor control

10. Stepper motor control
11. Servo motor control
12. Simple buggy control
13. Obstacle avoidance
14. Solving a maze

Most of these worksheets can be completed using Flowcode, Assembly or C. The maze and burglar alarm are only available in Flowcode and are a good test of a student's overall understanding of programming strategy.

There are no answers to these worksheets: with programming there are many ways of completing a task and most programs are unique. The journey here is more important than the destination!

5.4. Crystals, PICmicros and settings

There are many types of PICmicro: an 'F88 device is shipped on the EB006 Multiprogrammer board, and on the HP488 Version 3 development board. The buggy is shipped with a 'F627. You will need to be aware of the different settings within Flowcode, Assembly or C for PICmicros here: if your settings are not correct for the appropriate chip then your program may not run.

The current versions of Assembly for PICmicro microcontrollers and C for PICmicro microcontrollers are written around a 16F84 and a 3.2768MHz crystal. Current hardware is shipped with a 'F88 device and a 19MHz crystal. If you are using Assembly for PICmicro microcontrollers and C for PICmicro microcontrollers then you should make sure you have the correct crystal and PIC16F84 device.

If you find that your students are having trouble in getting their program to run, the most frequent reason for this is that they have not understood the PICmicro clock settings. Or that they have chosen Crystal operation in the set up software, but have the hardware set for RC oscillation, or vice-versa.

6. Worksheets

The following 14 Worksheets give the student a phased introduction to both Flowcode programming and Robotics, culminating in a set of buggy related exercises. These can all be undertaken with C or Assembly code programming as well.

The context of all of these worksheets is robotics. Students are able to build their experience and then use this on a small buggy that we make available.

Worksheets 1-8 cover basics such as input/output, control and decisions. The group of worksheets finishes with an example of a complete control system – the Burglar Alarm.

Worksheets 9-14 cover the various motor types used in robotics. The final four worksheets examine the control principles and decision making elements of a Buggy, leading up to the student's final task of solving a maze.

Worksheet 1: Simple input/output sequences

Worksheet 2: Loops and counters

Worksheet 3: Making decisions

Worksheet 4: Using subroutines and macros

Worksheet 5: A/D conversion

Worksheet 6: Using interrupts

Worksheet 7: Building a PICmicro circuit

Worksheet 8: A burglar alarm

Worksheet 9: DC motor control

Worksheet 10: Stepper motor control

Worksheet 11: Servo motor control

Worksheet 12: Simple buggy control

Worksheet 13: Obstacle avoidance

Worksheet 14: Solving a maze

| Robotics Worksheet 1 | | | |
|---|-------------------------------|--|-----------|
| Topic | Simple Input/Output sequences | Time: | 1-2 Hours |
| Subject | Flowcode fundamentals | | |
| Objectives: Light LEDs both singularly and in patterns. Receive input via switches, and react to these inputs. Understanding the basics of using outputs to communicate. | | | |
| Learning objectives: Basic Flowcode programming Input and Output principles Sequences and patterns | | | |
| Pre-requisites: None | | | |
| Resources: Software: Flowcourse - Flowcode, Outputting Data Flowcourse - Flowcode, Inputting Data Hardware: PICmicro/E-blocks board | | | |
| | | Programmer board/PPP settings | |
| | | Setting | |
| | | PICmicro device | |
| | | 16F88 | |
| | | Power supply | |
| | | 13.5V Positive outer + Batteries for the Buggy | |
| | | SW1 (Fast/Slow) | |
| | | N/A | |
| | | SW2 (RC/XT) | |
| | | XT | |
| | | Clock frequency | |
| | | 19.6608MHz | |
| | | Port A | |
| | | EB007 Switches board | |
| | | Port B | |
| | | EB004 LED board | |
| | | Watchdog timer | |
| | | OFF | |
| | | LVP | |
| | | Disabled | |
| | | Oscillator setting | |
| | | HS | |
| Instructions: The basis of robotics is communication. The robot must be able to receive input from its environment, and to be able to communicate with that environment where necessary. The most basic forms of communication are signal lights and input switches. | | | |
| Tasks: Task 1: Light a single LED (B0). Expand the program to light a sequence of LEDs (On-Off-On-Off-On-Off-On-Off) Task 2: Set an LED (B0) to flash at approximately 1 second on/one second off. Task 3: Set up an LED (B0) to light when switch A0 is pressed. Task 4: Set up the following input/output systems: B0 – Power on LED. A0 – Left hand bumper input. B1 – Left turn indicator – flashes 1Hz when Left hand bumper activated. A1 – Right hand bumper input. B2 – Right turn indicator – flashes 1Hz when Right hand bumper activated. | | | |
| Results: Demonstration of the programs running as detailed in the task instructions. | | | |

| Robotics Worksheet 2 | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|-------|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|----------------------|--------|-----------------|----------------|-----|-----|----------|--------------------|----|
| Topic | Loops and Counters | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Flowcode fundamentals | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Understanding the basics of using Loops and counters to control repeated commands. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Loops as control structures Counters and exit conditions | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: Worksheet 1: Simple Input/Output sequences | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: <ul style="list-style-type: none"> Flowcourse - Flowcode, Outputting Data Flowcourse - Flowcode, Outputting Data Flowcourse - Flowcode, Inputting Data Flowcourse - Flowcode, Doing Calculations Hardware: PICmicro/E-blocks board <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Programmer board/PPP settings</th> <th>Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>EB007 Switches board</td> </tr> <tr> <td>Port B</td> <td>EB004 LED board</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | EB007 Switches board | Port B | EB004 LED board | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | EB007 Switches board | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | EB004 LED board | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instructions: Many sequences of commands need to be performed repeatedly, often a certain number of times, or until a certain condition is met. Putting commands into a loop allows them to be repeated. However, you need to decide how and at what point the program exits the loop, and you need to implement any counters or variables required to handle exit condition. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Create a loop that lasts forever. Add in a sequence of commands that flashes port B at 1Hz. Task 2: Modify the program from Task 1 to loop 5 times. Task 3: Set up a Loop flashing Port B at 1Hz. Continue the loop until switch A0 is pressed. Task 4: Expand the program from Task 3 to exit the loop once switch A0 has been pressed 5 times. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Results: Demonstration of the programs running as detailed in the task instructions. | | | | | | | | | | | | | | | | | | | | | | | | | |

| Robotics Worksheet 3 | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|---|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|----------------------|--------|-----------------|----------------|-----|-----|----------|--------------------|----|
| Topic | Making decisions | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Flowcode fundamentals | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Using Decisions to control program flow. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Basic Flowcode programming Output principles Input principle Sequences and patterns | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: Worksheets 1-2 | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: Flowcourse - Flowcode, Decisions decisions! Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Programmer board/PPP settings</th> <th>Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>EB007 Switches board</td> </tr> <tr> <td>Port B</td> <td>EB004 LED board</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | EB007 Switches board | Port B | EB004 LED board | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | EB007 Switches board | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | EB004 LED board | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: The most important part of intelligence is the ability to make decisions. Deciding what to do requires us to examine the conditions we meet and reacting to them appropriately. Even deciding not to do anything is a decision in its own right. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Create a program that lights B0 if switch A0 is pressed. Task 2: Modify the program from Task 1 so that it turns on and stays on if A0 is pressed, and turns off if switch A1 is pressed. Task 3: Modify the program from Task 1 so that B0 only lights if A0 has been pressed 5 times. Task 4: Modify the program from Task 3 so that variable updated when A0 is pressed will only update if switch A1 is also pressed. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Results: Demonstration of the programs running as detailed in the task instructions. | | | | | | | | | | | | | | | | | | | | | | | | | |

| Robotics Worksheet 4 | | | |
|---|-------------------------------|--|-----------|
| Topic | Using Sub-routines and Macros | Time: | 1-2 Hours |
| Subject | Flowcode fundamentals | | |
| Objectives: Using and reusing code. Macros and parameters Importing and exporting macros. | | | |
| Learning objectives: Principles of code reuse Using macros Import and exporting macros Using LCD Displays | | | |
| Pre-requisites: Worksheets 1-2 | | | |
| Resources: Software: Flowcourse – Flowcode, Macros Hardware: PICmicro/E-blocks board | | | |
| Programmer board/PPP settings | | Setting | |
| PICmicro device | | 16F88 | |
| Power supply | | 13.5V Positive outer + Batteries for the Buggy | |
| SW1 (Fast/Slow) | | N/A | |
| SW2 (RC/XT) | | XT | |
| Clock frequency | | 19.6608MHz | |
| Port A | | EB007 Switches board | |
| Port B | | EB004 LED board | |
| Watchdog timer | | OFF | |
| LVP | | Disabled | |
| Oscillator setting | | HS | |
| Instruction: Code reuse is a key programming technique. Why reinvent the wheel? Or do the same thing 20 different times in the same program? Using macros you can perform tasks multiple times using just one set of code. | | | |
| Tasks: Task 1: Initialize the LCD display and display “Hello World”. Task 2: Create a basic macro that increments a variable and then displays it on the LCD. Use a loop and a delay to slow down the display so that the message is readable. Use Clear to erase the old value. Task 3: Modify the program from Task 2 to use the Cursor command to overwrite the old value. Consider how to resolve issue arising from trying the old value. Task 4: Create a basic macro that displays a variable on the LCD display (e.g. “MyVar = 0”). Create a second macro that increments the variable by 1 if switch A0 is pressed. Create a simple program loop in the main flowchart that calls the two macros once per second. Task 5: Export the macros from Task 4 to FCM files. Import the macros into a new program. Complete the program so that it resets the variable when it reaches 16. | | | |

| |
|--|
| <p>Consider if the macro needs to be changed, or can you do it without changing the macro. Discuss the consequences of changing the macro.</p> |
| <p>Results: Demonstration of the programs running as detailed in the task instructions.</p> |

| Robotics Worksheet 5 | | | |
|---|---------------------------------|--|-----------|
| Topic | Analogue to Digital conversions | Time: | 1-2 Hours |
| Subject | Flowcode fundamentals | | |
| Objectives: To be able to read analogue signal inputs and | | | |
| Learning objectives: Read in Analogue signals Work with High and Low bytes of the Analogue signal | | | |
| Pre-requisites: None | | | |
| Resources: Software: Flowcourse – Flowcode, Analogue inputs Flowcourse – Project 4: Temperature probe projects Flowcourse – Project 5: Light control projects Hardware: PICmicro/E-blocks board | | | |
| Programmer board/PPP settings | | Setting | |
| PICmicro device | | 16F88 | |
| Power supply | | 13.5V Positive outer + Batteries for the Buggy | |
| SW1 (Fast/Slow) | | N/A | |
| SW2 (RC/XT) | | XT | |
| Clock frequency | | 19.6608MHz | |
| Port A | | EB003 Sensors board | |
| Port B | | EB005 LCD Board | |
| Watchdog timer | | OFF | |
| LVP | | Disabled | |
| Oscillator setting | | HS | |
| Instruction: Many sensors use analogue signals rather than digital signals. ADC equipped Microcontrollers can take an analogue signal and convert it into a numeric value that can then be used programmatically. | | | |
| Tasks: Task 1: Read in the analogue value from ADC0 (the Light sensor). Display the HIGH byte value on the LCD Display. Task 2: Modify the program from Task 1 to display the LOW byte value. Compare the outputs from Task 1 and Task 2 to see how the 10 bit value is broken down into 8 bit values that the PICmicro can handle. Task 3: Modify the program from Task 1 to output the full analogue value (10 bit 0-1024 value). Task 4: Create a program that monitors the ADC0 channel and displays the HIGH byte value. Flash a “WARNING” message on the LCD if the value slips below 50. | | | |
| Results: Demonstration of the programs running as detailed in the task instructions. | | | |

| <h2 style="margin: 0;">Robotics Worksheet 6</h2> | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|-------|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|-----------|--------|-----------|----------------|-----|-----|----------|--------------------|----|
| Topic | Using Interrupts | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Flowcode fundamentals | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Use timer interrupts to create a clock display Use the B0 interrupt to implement an emergency button. Use the Port B interrupt to implement an indicator system. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Timer interrupts and pre-scalers B0 and Port B interrupts | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: None | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: Flowcourse – Flowcode, Interrupts Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Programmer board/PPP settings</th> <th style="text-align: left;">Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>See notes</td> </tr> <tr> <td>Port B</td> <td>See notes</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | See notes | Port B | See notes | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | See notes | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | See notes | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Task 1-3 - PICmicro/E-blocks board with Switches on Port B and LEDs on Port A. Task 4-5 - PICmicro/E-blocks board with Switches on Port A and an LCD on Port B. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: There are times you need to react to events immediately. The best way to do this is to stop what you are doing, jump to the code that needs to be run and then return to where you were. This is what interrupts do. Using them enables you to react as needed rather than having to constantly check to see if action is needed. In addition the Timer interrupt can be used to control time sensitive events, or to provide clock reference values. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Set up a B0 interrupt. Edit the Interrupt code to increment a variable every time the Switch B0 is pressed Display the variable on Port A Task 2: Set up a Port B interrupt. Light up LED A0 when a Port B interrupt is triggered. Note which Port B pins are part of the Port B interrupt. Task 3: Set up an program that uses Port B interrupt to light up indicators as follows: Pin B5 lights A0 | | | | | | | | | | | | | | | | | | | | | | | | | |

| |
|---|
| <p>Pin B6 lights A1 Pin B7 lights all available LEDs on Port A. Task 4: Set up the Timer macro to provide an accurate seconds timer. Use the Pre-scaler to provide a convenient timer overflow rate that can be used in conjunction with counters for an accurate update. Task 5: Modify the program from Task 4 to become a 24 hour clock.</p> |
| <p>Results: Demonstrate the interrupts in action via basic IO, or by messages/clock data on LCD displays.</p> |

| Robotics Worksheet 7 | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|---|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|----------------------|--------|-----------------|----------------|-----|-----|----------|--------------------|----|
| Topic | Building a PICmicro circuit | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Circuitry | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Light LEDs both singularly and in patterns. Receive input via switches, and react to these inputs. Understanding the basics of using outputs to communicate. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Constructing a circuit based on a PICmicro device. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: None | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: Flowcourse – Building circuits with PICmicros EB006 Multiprogrammer datasheet PPP help file. Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Programmer board/PPP settings</th> <th>Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>EB007 Switches board</td> </tr> <tr> <td>Port B</td> <td>EB004 LED board</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | EB007 Switches board | Port B | EB004 LED board | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | EB007 Switches board | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | EB004 LED board | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: So far you have only used E-blocks for running your programs. In this section we want you to program a PICmicro device, remove it from the Multiprogrammer, and use the Prototype board to construct a fully working PICmicro system. For this exercise you will need to look at the datasheets on the ELSAM CD ROM to derive the correct circuit. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: using the Prototype board construct a circuit with two input switches on A0 and A1 and two LEDs, with resistors, on B0 and B1. Use a crystal at 19MHz. Develop a program that reflects the state of the input switches on A0, A1 to B0, B1. Task 2: PICmicro devices are programmed using B6, B7, and MCLR. Wire up these three lines directly from the Multiprogrammer to the PICmicro on the development board so that you can use the Multiprogrammer as a remote programming system for the PICmicro on the prototype board. Task 3: Remove the crystal. Using an 'f88 device experiment with the settings in the PPP software to get the PICmicro device to use its internal oscillator. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Results: Demonstration of the programs running as detailed in the task instructions. | | | | | | | | | | | | | | | | | | | | | | | | | |

| <h2 style="margin: 0;">Robotics Worksheet 8</h2> | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|-------|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|----------------------|--------|-----------------|----------------|-----|-----|----------|--------------------|----|
| Topic | A Burglar alarm system | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Systems design | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Control a system with inputs and outputs System planning and design | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Control a system with inputs and outputs System planning and design | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: None | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: | | | | | | | | | | | | | | | | | | | | | | | | | |
| Software: Flowcourse – Flowcode, Project 2: A Burglar alarm. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Programmer board/PPP settings</th> <th style="text-align: left;">Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>EB007 Switches board</td> </tr> <tr> <td>Port B</td> <td>EB004 LED board</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | EB007 Switches board | Port B | EB004 LED board | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | EB007 Switches board | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | EB004 LED board | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: The Burglar alarm may not be a robotics system, but it is a complete system. The Burglar alarm enables the user to consider strategy and design. When doing the tasks consider both the current task and how they fit into the final system. What inputs are available, how should they react. What overall system is required? What sub systems are needed? | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Create a simple burglar alarm system that sets off an alarm when any window or door is open Task 2: Modify the program from Task 1 so that it can be turned on and off. The alarm will not go off if the system is turned off within 10 seconds. Task 3: Modify the program from task 3 to require a Code to turn the system on or off. Task 4: Modify the program from task 3 so that specific areas such as the main room can be turned on or off separately. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Results: Demonstration of the burglar alarm system in operation. | | | | | | | | | | | | | | | | | | | | | | | | | |

| <h2>Robotics Worksheet 9</h2> | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|--|---|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|------|--------|-----------------------|----------------|-----|-----|----------|--------------------|----|
| Topic | DC Motor control | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Motors and movement | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Move the DC motor both forwards and backwards. Investigate controlling motor speed Investigate motor positioning | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Basic DC motor driving Directional control Speed control Positioning | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: None | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: Flowcourse – Flowcode, Outputting Data Robotics and Mechatronics – DC Motors Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Programmer board/PPP settings</th> <th style="text-align: left;">Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>None</td> </tr> <tr> <td>Port B</td> <td>HPACT Actuators panel</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | None | Port B | HPACT Actuators panel | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | None | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | HPACT Actuators panel | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: Motors provide rotational movement which can be used for movement of both vehicles and for systems such as cranes that use rotational forces to drive linear motion. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Create a program that runs a motor as follow: Forwards when switch A0 is pressed Backwards when switch A1 is pressed Task 2: Modify the program so that the motor is pulsed on/off to provide basic half speed control. Task 3: Using the program from Task 2 experiment with different speed control methods. Try using large pulses - on for the period/off for the period, and smaller pulses - on/off/on/off and on/on/off/on/on/off etc. to see how they affect motor movement Task 4: Use the sensor input to work out when the DC motor has revolved one single revolution. Extend the program to count revolutions. Task 5: Modify the program from Task 4 to move the motor backwards and forwards a set number of turns. This simulates a system such as a crane or a lift where the rotational forces are used to raise or | | | | | | | | | | | | | | | | | | | | | | | | | |

lower an object between set limits.
Start the count of turns at 0 and allow the count to go up to 10.
Use A0 for Forwards
Use A1 for backwards
Move the motor at a set speed.

Results:

Demonstrate the DC Motor moving as appropriate.

Robotics Worksheet 10

| | | | |
|---|---------------------|--|-----------|
| Topic | Stepper motors | Time: | 1-2 Hours |
| Subject | Motors and movement | | |
| Objectives: Move stepper motors forwards and backwards Large steps and small steps Counting revolutions Positioning | | | |
| Learning objectives: Stepper motors Directional control Stepper motor sequences and patterns | | | |
| Pre-requisites: Basic Flowcode I/O | | | |
| Resources: Software: Flowcourse – Flowcode, Outputting Data Robotics and Mechatronics – Stepper Motors Hardware: PICmicro/E-blocks board | | | |
| Programmer board/PPP settings | | Setting | |
| PICmicro device | | 16F88 | |
| Power supply | | 13.5V Positive outer + Batteries for the Buggy | |
| SW1 (Fast/Slow) | | N/A | |
| SW2 (RC/XT) | | XT | |
| Clock frequency | | 19.6608MHz | |
| Port A | | None | |
| Port B | | HPACT Actuators panel | |
| Watchdog timer | | OFF | |
| LVP | | Disabled | |
| Oscillator setting | | HS | |
| Instruction: Stepper motors move in rotational increments which allow a greater amount of control as to the amount of rotation. A greater level of positional accuracy is also possible. However, precise control is limited due to the set size nature of the step movements. | | | |
| Tasks: Task 1: Investigate moving the stepper motor in a single direction by switching on single magnets. Investigate what sequence of motors produce a forwards motion. Task 2: Investigate moving the stepper motor with half steps. Task 3: Modify the program from Task 2 to provide forwards and backwards motion. Task 4: Investigate positioning the stepper motor: A mark on the stepper disk is required as start and reference point Calculate the angle turned for each single step for both full and half steps. Given a desired angle of 73 degrees move the mark to as close as that position as possible. Task 5: Modify the program from Task 4 so that the stepper motor moves to the following positions: Switch A0 = 0 degrees from start | | | |

Switch A1 = 90 degrees from start
Switch A2 = 143 degrees from start
Switch A3 = 260 degrees from start
Note that it may not be possible to get to each precise position. If it is not then try to get as close as possible.

Results:

Demonstration of the stepper motor moving as appropriate for the task.

Robotics Worksheet 11

| Topic | Servo motors | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
|---|--|-------|-----------|-------------------------------|---------|-----------------|-------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|------|--------|-----------------------|----------------|-----|-----|----------|--------------------|----|
| Subject | Motors and movement | | | | | | | | | | | | | | | | | | | | | | | | |
| Objectives: Angular positioning of Servo motors Signal pulse duration and timing | | | | | | | | | | | | | | | | | | | | | | | | | |
| Learning objectives: Servo motors Servo timing pulses Setting angles | | | | | | | | | | | | | | | | | | | | | | | | | |
| Pre-requisites: None | | | | | | | | | | | | | | | | | | | | | | | | | |
| Resources: Software: Flowcourse – Flowcode, Outputting Data Robotics and Mechatronics – Servo Motors Hardware: PICmicro/E-blocks board | | | | | | | | | | | | | | | | | | | | | | | | | |
| <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Programmer board/PPP settings</th> <th style="text-align: left;">Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F88</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>None</td> </tr> <tr> <td>Port B</td> <td>HPACT Actuators panel</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | | | Programmer board/PPP settings | Setting | PICmicro device | 16F88 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | None | Port B | HPACT Actuators panel | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F88 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | None | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | HPACT Actuators panel | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| Instruction: A Servo motor is a motor that can hold a precise angle. The angle is dependant on the duration of a pulse sent at a specific timing interval. Note: Due to the very accurate timing requirements, and the shortness of the timing pulses you may need to use C Code commands to help create the signal pulses. | | | | | | | | | | | | | | | | | | | | | | | | | |
| Tasks: Task 1: Create a program that sends a 2ms pulse every 20ms. Task 2: Modify the program from Task 1 to produce pulses as follows: Switch A0 = 1ms Switch A1 = 1.5 ms Switch A2 = 2ms Task 3: Modify the program from Task 1 to produce a pulse that increases from 1 to 2 ms. Task 3: Set up an LED (B0) to light when switch A0 is pressed. Task 4: Modify the program from Task 3 to move the Servo | | | | | | | | | | | | | | | | | | | | | | | | | |
| Results: Demonstrate the Servo motor in action. | | | | | | | | | | | | | | | | | | | | | | | | | |

Robotics Worksheet 12

| | | | |
|--|---------------|--|-----------|
| Topic | Simple buggy | Time: | 1-2 Hours |
| Subject | Buggy control | | |
| Objectives: Move the buggy Turn the buggy | | | |
| Learning objectives: Basic DC Motor control Turning with motors | | | |
| Pre-requisites: DC Motors | | | |
| Resources: Software: Flowcourse – Flowcode, Outputting Data Robotics and Mechatronics – Example systems Robotics and Mechatronics – Case Studies Hardware: PICmicro/E-blocks board and PIC Buggy | | | |
| | | Programmer board/PPP settings | |
| | | Setting | |
| | | PICmicro device | |
| | | 16F627 | |
| | | Power supply | |
| | | 13.5V Positive outer + Batteries for the Buggy | |
| | | SW1 (Fast/Slow) | |
| | | N/A | |
| | | SW2 (RC/XT) | |
| | | XT | |
| | | Clock frequency | |
| | | 19.6608MHz | |
| | | Port A | |
| | | None | |
| | | Port B | |
| | | None | |
| | | Watchdog timer | |
| | | OFF | |
| | | LVP | |
| | | Disabled | |
| | | Oscillator setting | |
| | | HS | |
| Instruction: The Buggy is a vehicle and like all vehicles needs to move. This is accomplished by controlling the two DC motors. Independent and simultaneous control of the motors allows the direction of movement to be controlled. | | | |
| Tasks: Task 1: Move the buggy forwards and backwards in 4 second bursts. Use speed control to make the Buggy go at a smooth and moderate speed. Task 2: Move the buggy forwards with both motors. After 4 seconds of forward motion turn the buggy to the left by stopping or slowing the right motor. Task 3: Modify the program from Task 2 so that the motors turn in opposite directions. What affect does this have on turning distance? | | | |
| Results: Demonstrate the Buggy in motion, and turning. | | | |

Robotics Worksheet 13

| | | | |
|---|--------------------|--|-----------|
| Topic | Obstacle avoidance | Time: | 1-2 Hours |
| Subject | Buggy control | | |
| Objectives: Detect and react to obstacle in the Buggy's path. | | | |
| Learning objectives: Detecting and reacting to obstacles Turning away from obstacles Moving set distances and rotating set turn angles | | | |
| Pre-requisites: Worksheet 9 - DC Motors Worksheets 11-13 – Buggy control | | | |
| Resources: | | | |
| Software: | | | |
| Flowcourse – Flowcode, Outputting Data | | | |
| Robotics and Mechatronics – Example systems | | | |
| Robotics and Mechatronics – Case Studies | | | |
| Hardware: | | | |
| PICmicro/E-blocks board and PIC Buggy | | | |
| Programmer board/PPP settings | | Setting | |
| PICmicro device | | 16F627 | |
| Power supply | | 13.5V Positive outer + Batteries for the Buggy | |
| SW1 (Fast/Slow) | | N/A | |
| SW2 (RC/XT) | | XT | |
| Clock frequency | | 19.6608MHz | |
| Port A | | None | |
| Port B | | None | |
| Watchdog timer | | OFF | |
| LVP | | Disabled | |
| Oscillator setting | | HS | |
| Instruction: We cannot go forwards forever; sooner or later we will bump into something. When we do we need to realize that we have bumped into something and react to the obstacle. At this stage we are more concern with racing rather than any kind of decision as to which way to go. The following tasks introduce some basic obstacle avoidance principles. | | | |
| Tasks: Task 1: Move the buggy forwards until it meets an obstacle with either bumper. Stop the vehicle once an obstacle has been hit. Task 2: Modify the program from Task 1 so that the Buggy reverses back for 2 seconds. Task 3: Modify the program from Task 2 so that the Buggy turns to the Left for 2 seconds once it has reversed. Task 4: Modify the program from Task 3 so that the Buggy reverses a set distance (15cm). Task 5: Modify the program from Task 4 so that the Buggy turns approximately 90 degrees to the left. | | | |
| Results: Demonstrate that the Buggy can successfully react to obstacles placed in its path. | | | |

| <h2 style="margin: 0;">Robotics Worksheet 14</h2> | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|-------|-----------|-------------------------------|---------|-----------------|--------|--------------|--|-----------------|-----|-------------|----|-----------------|------------|--------|------|--------|------|----------------|-----|-----|----------|--------------------|----|
| Topic | Solving a Maze | Time: | 1-2 Hours | | | | | | | | | | | | | | | | | | | | | | |
| Subject | Buggy control | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Objectives: Decision making strategies and methods. Pre-set behaviours and random behaviours Co-ordination of controls for movement. Decision making skills to for navigating the maze</p> | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Learning objectives: System design Control and movement Decision making</p> | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Pre-requisites: Worksheet 9 - DC Motors Worksheets 11-13 – Buggy control</p> | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Resources:</p> <p>Software:</p> <ul style="list-style-type: none"> Flowcourse – Flowcode, Outputting Data Robotics and Mechatronics – Example systems Robotics and Mechatronics – Case Studies <p>Hardware:</p> <p>PICmicro/E-blocks board and PIC Buggy</p> <table border="1" style="width: 100%; border-collapse: collapse; margin-left: 20px;"> <thead> <tr> <th style="text-align: left;">Programmer board/PPP settings</th> <th style="text-align: left;">Setting</th> </tr> </thead> <tbody> <tr> <td>PICmicro device</td> <td>16F627</td> </tr> <tr> <td>Power supply</td> <td>13.5V Positive outer + Batteries for the Buggy</td> </tr> <tr> <td>SW1 (Fast/Slow)</td> <td>N/A</td> </tr> <tr> <td>SW2 (RC/XT)</td> <td>XT</td> </tr> <tr> <td>Clock frequency</td> <td>19.6608MHz</td> </tr> <tr> <td>Port A</td> <td>None</td> </tr> <tr> <td>Port B</td> <td>None</td> </tr> <tr> <td>Watchdog timer</td> <td>OFF</td> </tr> <tr> <td>LVP</td> <td>Disabled</td> </tr> <tr> <td>Oscillator setting</td> <td>HS</td> </tr> </tbody> </table> | | | | Programmer board/PPP settings | Setting | PICmicro device | 16F627 | Power supply | 13.5V Positive outer + Batteries for the Buggy | SW1 (Fast/Slow) | N/A | SW2 (RC/XT) | XT | Clock frequency | 19.6608MHz | Port A | None | Port B | None | Watchdog timer | OFF | LVP | Disabled | Oscillator setting | HS |
| Programmer board/PPP settings | Setting | | | | | | | | | | | | | | | | | | | | | | | | |
| PICmicro device | 16F627 | | | | | | | | | | | | | | | | | | | | | | | | |
| Power supply | 13.5V Positive outer + Batteries for the Buggy | | | | | | | | | | | | | | | | | | | | | | | | |
| SW1 (Fast/Slow) | N/A | | | | | | | | | | | | | | | | | | | | | | | | |
| SW2 (RC/XT) | XT | | | | | | | | | | | | | | | | | | | | | | | | |
| Clock frequency | 19.6608MHz | | | | | | | | | | | | | | | | | | | | | | | | |
| Port A | None | | | | | | | | | | | | | | | | | | | | | | | | |
| Port B | None | | | | | | | | | | | | | | | | | | | | | | | | |
| Watchdog timer | OFF | | | | | | | | | | | | | | | | | | | | | | | | |
| LVP | Disabled | | | | | | | | | | | | | | | | | | | | | | | | |
| Oscillator setting | HS | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Instruction: Once an obstacle is met the Buggy must decide which way to turn. We being intelligent can tell the Buggy which way to turn next, but that only works for a set maze. What the buggy really needs is a method to select which way it turns itself which is what we will be investigating here.</p> | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p>Tasks:</p> <p>Task 1: Set up a simple maze and examine what turns would be needed to complete the course. Program the buggy to turn as appropriate to finish the maze.</p> <p>Task 2: Set up the Buggy so that when it hits an obstacle it: Reverses for 4 seconds Turns to the Left for approx 90 degrees (modify turn timing as appropriate) Continues forwards. See how a single turn direction affects the Buggy's ability to solve a simple maze</p> <p>Task 3: Modify the program from Task 1 so that the turn direction alternates Left and Right.</p> | | | | | | | | | | | | | | | | | | | | | | | | | |

Note how the changes affect the Buggy's ability to solve a simple maze
Task 4: Create a program that turns the Buggy to a random direction each time.
(Note: You can use a timer interrupt to increment a variable and use bit 0 of that variable as the
Left or right direction.)
How does random turning affect the Buggy's ability to solve a simple maze

Results:

Demonstrate that the Buggy can react to obstacles in the appropriate manner.
Demonstrate that the Buggy turns randomly in Task 4.