Matrix Multimedia PICmicro microcontroller development board
Information datasheet: Using external sensors and actuators

Contents of this document

Contents of this disk (part TEFWS)

SensAct.doc	This document
ActuatorsTest1.FCF	Actuators test program for Flowcode
Actuators Test1.ASM.	Actuators test program for Assembly
sensor_library.asm	Macros for completing worksheets
LibraryTest.fcf	An example flowchart using the sensor_library.asm code.

1 Introduction

The information in this document is designed to act as a framework to allow students to carry out certain exercises using a core range of sensors and actuators which attach to the Matrix Multimedia version 2 PICmicro® development board.

The worksheets in this document allow students to carry out a number of exercises using sensors and actuators. The worksheets can be used as a framework of assessment in many electronics, control, and robotics courses. The worksheets will require that student's reference the information in this document and in the PICmicro development board datasheet, which is included on the floppy disk/CD, supplied with the board.

We do not supply completed code, or flowcharts, for the sensors although code has been written to verify that these exercises are practical. However, a code library, containing a number of useful functions, has been supplied on the floppy disk to aid completion of the exercises.

Most of the worksheet exercises involving sensors do not require the use of actuators and vice-versa.

We hope that you will find the exercises involving the DC motor and feedback loop particularly effective in teaching the use of real feedback systems.

Please do contact us with feedback and comments.

John Dobson john@matrixmultimedia.co.uk
Ian Hill ian@matrixmultimedia.co.uk
Matrix Multimedia Ltd.

Sensors Overview

4 sensors have been chosen which require students to gather real world data using different programming strategies:

Sensor	**Output/action**	**Coding strategy**
Temperature probe	Simple potential divider	A/D conversion, calibration, value look up, display
Motion detector	Gives out a digital pulse correlating to distance	Pulse time measurement, conversion display
Heart rate monitor	Gives out an analogue voltage pulse	Data slicing, timing,
Photogate and pulley wheel	When IR light path is interrupted, digital output changes	Various

This document does not list code which carries out these Coding strategies: that is for the student to produce.

These sensors can be used with any Matrix PICmicro CD ROM: Flowcode for PICmicros, C for PICmicros, or Assembly for PICmicros.

These sensors are for educational use only.

Actuators overview

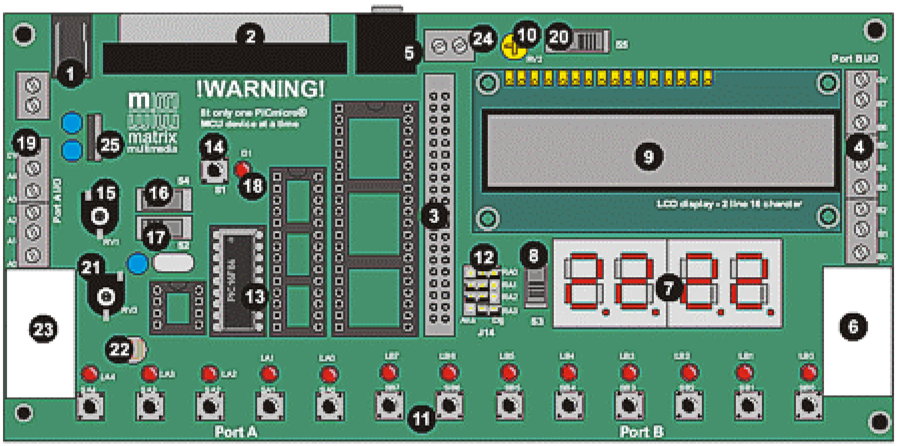
Matrix supplies two robTT5 1 Tf▯ Tc 0 Tw 10146.▯0.02 0 0 105 frsd .gei4 .00 0 10.02 150.5684 EMC▯P ▯▯/MCI 4 .00 0

2 Sensor information

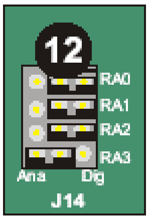
Using external analogue sensors

Analogue sensors require a PICmicro chip with analogue capabilities, such as the PIC16F874 and PIC16F877.

Sensors need to be plugged in to the analogue sensor port (white connector on the left hand side of the PICmicro development board – no 23) or the digital sensor port (white connector on the right hand side).



To set the PICmicro development board to the external analogue sensor mode set jumper 3 on the Port A mode selector J14.

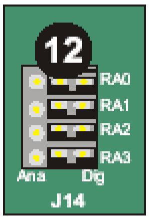


Using external Digital sensors

Digital sensors can be used with all PICmicro chips, such as the PIC16F84 and PIC16F627.

Sensors need to be plugged in to the digital sensor port (white connector on the right hand side of the PICmicro development board – no 6).

To set the PICmicro development board to the external digital sensor mode reset all 4 jumpers to the right on the Port A mode selector J14.



Heart rate monitor

The heart rate monitor is an analogue sensor and requires a PICmicro device with A/D inputs

Description

The Exercise heart rate monitor is ideal for determining the heart rate of actively moving individuals. With this sensor, a person's heart rate can be monitored during, as well as after exercise. The Exercise heart rate monitor consists of a wireless transmitter belt and a receiver module. The transmitter belt senses the electrical signals generated by the heart much like an ECG. For each heartbeat detected, a signal is transmitted to the receiver module, and a heart rate is determined.

Note on use in institutions. Some institutions, particularly those with mixed sex classes, have reported that students have shown some embarrassment/reluctance in using the heart rate monitor, as it needs to be placed under articles of clothing. Please consider this issue when using the heart rate monitor.

Instructions for use

Place the chest band/transmitter on the person being tested.
For maximum efficiency wet the two electrodes with a saline solution (e.g. a 5% salt solution).
Plug the receiver into the PICmicro board analogue input. The receiver has a range of 80-100cm.

The input is received as an analogue signal on Pin A4. We suggest that before attempting any coding you connect an oscilloscope to RA4 and view the signal the heart rate monitor produces

Temperature probe

The temperature probe is an analogue sensor and requires a PICmicro device with A/D inputs

Description

This rugged and durable temperature probe has a sealed stainless steel shaft and tip that can be used in organic liquids, salt solutions, acids, and bases.

Instructions for use

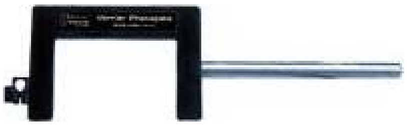
Simply place the probe in the liquid, or against a solid and wait for a short while.
The probe will heat up or cool down until it matches the temperature or the item being measured.
The input is received as an analogue signal on Pin A4.
The reading from the probe is a raw data value not directly related to any of the temperature scales.
However by plotting value against known temperature a simple temperature scale can be constructed.

The value is the resistance value of a Thermistor and can be converted to °C via the following formula:
T = [K0 + K1(ln 1000R) + K2(ln 1000R)3]-1 – 273.15
Where T is temperature (°C), R is the measured resistance in kΩ, K0 = 1.02119 X 10-3,
K1 = 2.22468 X 10-4, and K2 = 1.33342 X 10-7.
Or you can simply use the raw Thermistor value.

The probe is designed to measure from –25 to 125°C.
Maximum recommended temperature is 150°C.

Note that at the time of going to press (July 2002) version 2.0 development boards will require slight modification to enable this sensor to operate. See our web site for details.

Photogate and smart pulley



Description
Photogates can be used to study free fall, rolling objects, air track collisions, pendulums, etc. These inexpensive, ready-to-use photogates are similar to the traditional PASCO photogate, but have no stand. They can be easily mounted on a ring stand. The photogate sensor consists of the sensor unit and a rod, which allows the photogate to be clamped into position. The smart pulley attaches to the photogate. The pulley has a slotted wheel, which passes through the photogate's beam.

Instructions for use
The photogate sends an infrared beam between the two arms.
The status of the beam is sent to Pin A4 as an input.
When the beam is being received the sensor output is high (Pin A4 = 1).
When the beam is blocked the output is low (Pin A4 = 0)

Motion detector

Description

The Motion detector functions like the automatic range finder on a Polaroid camera. This sonar device emits ultrasonic pulses and waits for an echo. The time it takes for the reflected pulses to return is used to calculate distance, velocity, and acceleration. The range is 0.4 to 6 meters. Our Motion detector has a pivoting head, rubber feet, and a clamp for mounting.

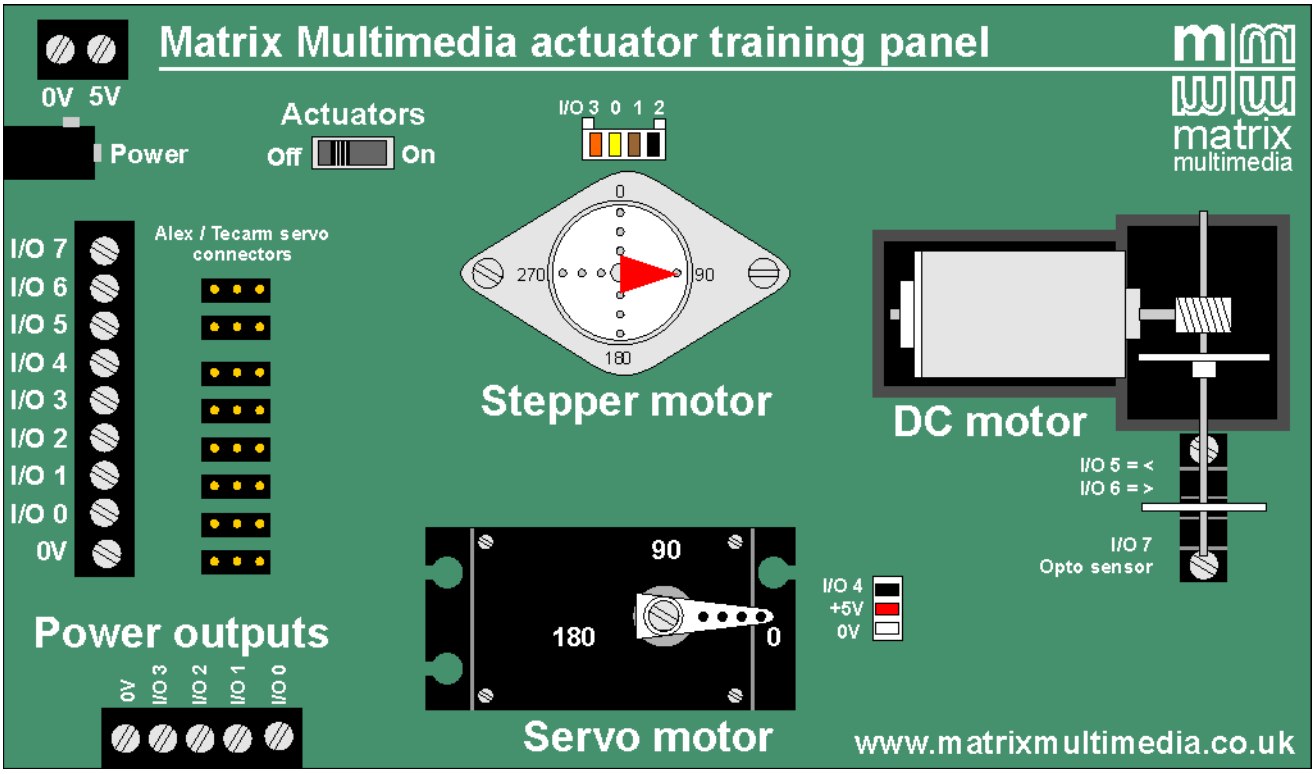
Instructions for use

Ultrasound signal speed is 343m/s in air
The detection cone is 15 to 20°.
If you having trouble getting a return value check that there is no object in the detection cone that may be interfering with the experiment.

The motion detector requires a trigger to emit a bust of ultrasound. The outbound pulse can be triggered by an output on Pin A2. You should be able to hear a small click for each outbound pulse.
When the returning echo is picked up by the sensor a signal is sent to pin A4.
By calculating the time taken for the signal to return a distance value can be determined.
By comparing incoming signals with a stored base signal, motion in the sensor area can be detected.

If you are using the motion detector to detect movement in a preset area you may need to wait for the initial reading to steady before it starts to search for movement.

3 Actuators panel information



WARNING – use only 6V DC supply maximum.

Purpose

The actuators panel has three purposes:

Firstly it contains 3 different types of motor that allow students to understand how different motor systems work.

Secondly it provides an interface between the Alex animated head and the TecArm robot systems and the PICmicro development board.

Thirdly it provides power outputs for general purpose work such as powering external motors and relays.

How to use the on/off switch

Setting the switch to the right ON position turns power onto the actuators.

Setting the switch to the left turns the actuators off and allows the actuator board to be used with either the external power outputs or the Alex animated head /TecArm connection. Note that the power outputs and the servo connections for Alex / TecArm are always on.

How to connect the board

There are 8 Input/Output connections on the left hand side of the actuators board, numbered 0-7, and a 0V connection. A matching set of input/outputs for Port B of the PICmicro can be found on the right hand side of the PICmicro development board (no – 4 on the diagram). If you wish to use the display on the development board then you can use a 28 or 40 pin device and use port C or D (from the expansion bus on the development board) as the driving port.

You can connect some or all of the connections depending on what elements of the actuator board you wish to use.

The following table lists the actuator board connections:

Actuator I/O connection no.	Description	Recommended PICmicro connection
0V	0V connection	0V
I/O 0	Stepper motor output (Yellow)	Pin B0
I/O 1	Stepper motor output (Brown)	Pin B1
I/O 2	Stepper motor output (Black)	Pin B2
I/O 3	Stepper motor output (orange)	Pin B3
I/O 4	Servo motor output	Pin B4
I/O 5	DC motor output	Pin B5
I/O 6	DC motor output	Pin B6
I/O 7	Opto sensor input	Pin B7

Using with TecArm/Alex
There is a set of connections for the Alex animated head and TecArm, just to the right of the I/O connectors. Connect the servo leads from Alex animated head or TecArm and you can drive the servos from the PICmicro development board.

Power outputs
A set of 4 extra output connections, and a 0V connection, can be found on the bottom edge of the actuators panel.
These outputs are from I/O connections 0-3 buffered by an L293D driver chip.
The power outputs cannot be used as inputs.
Note that these are wired in parallel with the stepper motor so if you are using the power outputs for an application and you have the switch in the 'ON' position you can expect the stepper motor to judder about a bit.

Power supply
The panel takes nominal 5V. The power connector is positive outer – please make sure you have it the right way round before powering it up. The actuators board cannot be powered from the PICmicro board unless you fit a heatsink to the 7805 regulator on the board (there should be room to do this). With a heatsink you may be able to use the on-board actuators without too much difficulty using the 5V power output from the development board. You need to use some judgement here as heatsinks vary in performance. If your regulator starts to shut down then you will need to use an external power supply.

When powering Alex or the TecArm you will definitely need to use an external power supply as these systems can easily consume more than 500mA.

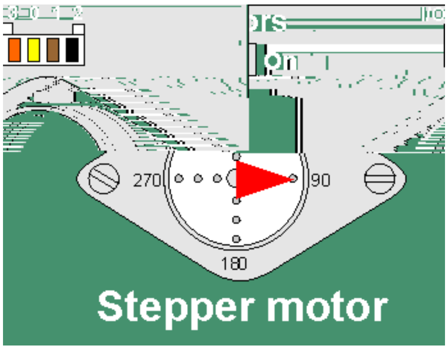
Actuator board test routine
The test routine ActuatorsTest1.fcf has been developed inside Flowcode. This is also supplied as Actuators Test1.ASM. This program tests the following items on the actuators board:

- Stepper motor clockwise
- Stepper motor counter clockwise
- DC motor clockwise
- DC motor counter clockwise

- Opto sensor (4 rotations of the DC motor)
- Stepper motor 0°
- Stepper motor 90°
- Stepper motor 180°

When you get your actuators panel please check that it is fully operational using this test. It is designed for a PIC16F84 and you should be able to use the PPP (PICmicro Parallel Programmer) software supplied with the development board to send this to the PIC16F84.

Stepper Motor



Description

Stepper motors use a set of internal electro-magnets to attract the rotor. By activating magnets next to the one that the rotor is currently at, the rotor will move or step to the new position. By repeating this process around a set of magnets the rotor will appear to move round in a circle.

Stepper motors are designed to step a set number of degrees in a single step.
The stepper motor on the actuators board moves in 15° steps, i.e. every step is 15°.
So a set of 4 steps would move the rotor 60°.

This motor does not support half steps.

Instructions for use

The stepper motor moves in 15° steps.
To move the stepper motor clockwise use the following sequence.

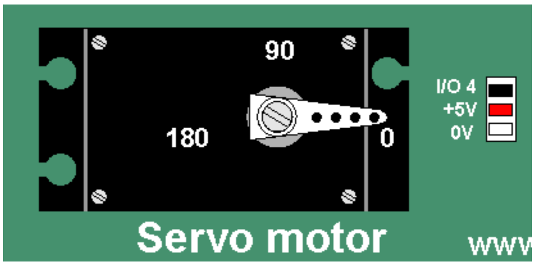
	Orange (I/O 3)	Yellow (I/O 0)	Brown (I/O 1)	Black (I/O 2)	Input on pins 0-3 (bin, hex, dec)
Step 1	1	0	1	0	1010, 0AH, 10
Step 2	0	1	1	0	0011,03H, 3
Step 3	0	1	0	1	0101, 05H, 5
Step 4	1	0	0	1	1100, 0CH, 12

Note: The inputs are listed in the same order as the wires in the connection socket.
The input number assumes that the control inputs are on output pins 0-3 of the control device.

To move counter clockwise simply reverse the sequence.

If you want to be able to move the stepper by a certain number of steps you may need to keep track of what the current step is in relation to the above chart. Otherwise you may find the rotor jumping around when it gets a next step that is not in the correct sequence.

Servo motor



Description
Servo motors consist of a motor and an arm that rotates around the motor.
Unlike DC and stepper motors the servo motor does not rotate freely but instead rotates a number of degrees from 0 according to the pulse signal sent to it.
Changing the pulse signal changes the amount of rotation from 0 degrees that the arm moves.
The arm can be directly responsible for rotational movement, or can use a mechanical linkage to change the rotational movement to a single plane of movement (e.g. horizontal).

Instructions for use
The servo motor is driven from I/O line 4.
The servo motor operates with a pulse of twenty milliseconds.
This 20ms pulse can be further divided into two sections, the high positioning section and the low section.
The high positioning section is a high signal pulse at the beginning of the 20ms time period determines how great an angle the servo moves through.
A signal length of 1ms moves the servo arm 0°
A signal length of 1.5ms moves the arm 90°
A signal length of 2ms moves the arm 180°

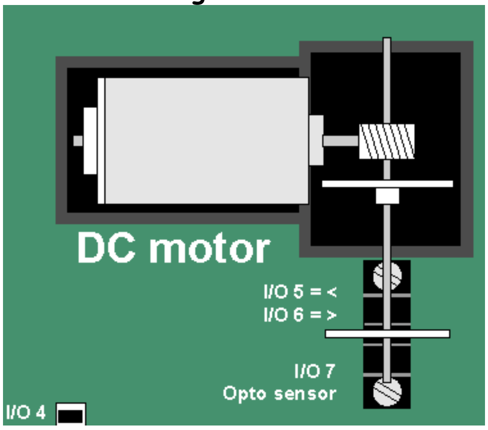
Servo motors normally have internal stops to prevent movement outside of the 180° arc.

The second section is a low signal that lasts from the end of the high section until the 20ms pulse duration is over (normally between 18 and 19 ms).

To maintain rotor position the whole pulse needs repeating every 20ms.

Note: Clock speed of the control device may affect positional accuracy of the servo arm, as the device may not be able to generate signal pulses with sufficient time accuracy for degree perfect positioning.

DC motor and gearbox with feedback



Description

DC motors use internal magnets and brushes to turn the rotor or shaft.
Rotation is either clockwise or counter clockwise.
The motor turns continuously when a voltage is present. To control the speed of the movement the motor can be turned on and off in bursts. The longer the motor is off compared to the time it is on, the slower the motor rotates. This is called Pulse Width Modulation (PWM).
Unlike stepper motors and servo motors accurate positioning is very difficult to achieve via the dc motor alone. DC motors are normally used in conjunction with a sensor that is triggered by movement caused by the motor. Examples are car park barriers when the raised arm trips a sensor that tells the controller that the barrier has been raised high enough.
Another example, found on the actuators board is an opto-electronic sensor that is triggered by the rotation of the motor. In this case a disc with holes at known intervals has been attached to a shaft driven by the motor. By counting how many of the holes have gone past it is possible to calculate how far the shaft has rotated. PWM is useful here as if the motor is too fast it may have moved past the point that you want before the stop signal arrives. One useful trick is to slow the motor down as it nears the stop point, gaining greater accuracy for stopping. (This is the basis for fuzzy logic.)

Instructions for use

An output on I/O 5 turns the motor clockwise.
An output on I/O 6 turns the motor counter clockwise.
An output on both I/O 5 and I/O 6 is ignored and there is no movement.
If the output is continuous then the motor will run at full speed.
If the output is Pulse Width Modulated then the speed of the motor will decrease in the same ratio of that of the 'on' time compared to the 'off' time.
For example if there an 'on' pulse of 1ms followed by an 'off' pulse of 9ms then the motor is only on for one tenth of the time so it is running at 1/10th of full speed.
At large ratios the motor may appear to pulse or operate in bursts.

The opto-sensor is connected to I/O 7.
The sensor is permanently on, and will register high with an uninterrupted beam.
If the beam is blocked than the signal becomes low.
The sensor is associated with a shaft-mounted disc with holes at 90-degree intervals.
When the hole passes the sensor the signal changes from low to high.

4 Using the worksheets

These worksheets are fully editable – please feel free to modify them for your own uses.

We have split the worksheets into different levels of accomplishment. The amount of work that your students will get through will depend on their competence with programming, and how long you want them to spend on the task. In practice we have found that there are usually one or two bright students who finish tasks quicker than one would like so we have tried to include some quite tricky problems to keep these students busy.

These worksheets are designed to be used with Assembly, C or Flowcode. For the sensors worksheets Flowcode users will not usually get past level 1. In order to allow Flowcode users to work with the sensors we have had to provide some pre-written Assembly code which can be embedded into Flowcode in the form of Macros. Details are given below. Flowcode students may need help in understanding how to embed this assembly code.

How you want to use these worksheets will depend on the course you are teaching and the language that your students are working in. The mix of problems we have set should be usable over a wide range of courses from Schools Design and Technology (using Flowcode) through to University microprocessor courses using Assembly.

The worksheets are fairly open ended – the tasks set allow students to use their imagination in developing solutions rather than following a particular coding strategy.

The worksheets are in no particular order: please read them to see which your students should start with.

To use these worksheets with your students you will need to give them the information on the sensors and actuators (see previous sections) as well as the worksheet itself. You may also need to give them pre-written routines or macros that they can use to complete the worksheets. You can be selective about how much information you provide for them.

Please give us your feedback on these worksheets.

5 Using Macros

In addition to the worksheet supervisors should also give a copy of this section to their students.

Additional code, written in Assembly language, has been provided to make it simpler/faster to complete the worksheet exercises.

The code contains functions for retrieving analogue sample data, or motion detector distance measurement, and outputting this data to an LCD display. Details are also given on how to set up a computed-GOTO jump table (advanced users only).

The code can be added into a "C" icon in Flowcode – instructions are given below.

This code can also be embedded as Assembler in C. The code can be used in its native form if you are using the Assembly for PICmicros CD ROM. Students using C or Assembly may need to carry out further investigation here.

The Assembly code for the macros can be found in the file: sensor_library.asm, which will be on the disk provided (you may need to see your supervisor for this)

Instructions on how to include and use this code within your own programs can be found below. Please refer to the appropriate section.

FlowCode users:

The library functions consist of two parts:

a) The library Assembly code itself
b) A set of variables required for the code to function

Adding the library code

You should add a "C" icon at the end of the "`Main`" flowchart, which will contain the additional code. The code should be placed within an "`asm{...}`" construct as follows:

```
asm
{

<Place the entire code from sensor_library.asm here>

}
```

Alternatively copy the "C" icon with the code from an already existing flowchart (e.g. LibraryTest.fcf).

Adding the variables

The following variables need to be added to the "`Main`" :

```
   MV_STORE1            ; Temporary store
   MV_STORE2            ; Temporary store
   MV_OFFSET            ; Offset for tables
   MV_DECVAL            ; Decimal value to display
```

You can add these variables by including a calculation icon, double-clicking on the icon to edit its properties, selecting BROWSE, ADD NEW VARIABLE.

Calling functions
Functions within this Assembly code can be called from any point within your flowchart (and from within any macro) by adding a "C" icon with the following syntax:

```
asm
{
      CALL  <fn_name>
}
```

(where `<fn_name>` is the name of the appropriate function as detailed below).

'C' users:

The following global variables need to be defined at the beginning of your C program:

```
FCV_MV_STORE1  ; Temporary store
FCV_MV_STORE2  ; Temporary store
FCV_MV_OFFSET  ; Offset for tables
FCV_MV_DECVAL  ; Decimal value to display
```

The code itself should be inserted into your C file. We recommend that you place it at the end of your "`main()`" function.

Functions within this code can be called from any point within your program by using the following syntax:

```
asm
{
      CALL  <fn_name>
}
```

(where `<fn_name>` is the name of the appropriate function as detailed below).

To see a pre-defined example of how this is done refer to the file LibraryTest.fcf which is shipped with this document.

Assembly code users:

The following registers need to be defined at the beginning of your program:

```
_FCV_MV_STORE1 ; Temporary store
_FCV_MV_STORE2 ; Temporary store
_FCV_MV_OFFSET ; Offset for tables
_FCV_MV_DECVAL ; Decimal value to display
```

The code itself should be inserted into your ASM file. We recommend that you place it just before the final "`END`" statement in the ASM file.

Functions within this code can be called from any point within your program by using the following syntax:

```
CALL <fn_name>
```

(where `<fn_name>` is the name of the appropriate function as detailed below).

The Functions

These are available to Flowcode, C and Assembly users.

(1) **LCD Display**

To utilise the LCD display, you must make use of 2 function calls. You need to call "`MX_SETUPLCD`" at the beginning of your program to set up the LCD. When you want to display a number, you need to populate the "`MV_DECVAL`" register with the value you want to display and then call the "`MX_LCDBYTE`" subroutine.

(2) **Read External Sensor Value**

Calling the function "`MX_READ_A3`" will read the value of the external analogue sensor and place the most significant 8-bits into the "`MV_DECVAL`" variable.

(3) **Read the Distance Sensor**

Calling the function "`MX_READ_DIST`" will read the value of the distance sensor and place this reading into the "`MV_DECVAL`" variable. Note that it is best to call this function no more than 100 times a second (because the clicking of the distance sensor can get annoying!). If the distance sensor is not connected, this function will run forever and your program will appear to be stuck.

Implementing a computed-GOTO jump table

Computed-GOTO jump tables are an advanced topic and have been provided primarily for 'C' and ASM users.

'C' users please note: the following code is in ASM and will need to be placed in an `asm {...}` construct.

A computed-GOTO jump table is a way of implementing an array in ASM. An offset is passed to the function and the corresponding data value in the table is returned.
In the following example the "offset" is passed in the `MV_OFFSET` variable, and the result is placed into the `MV_DECVAL` variable.
Further examples can be found towards the end of the sensors library code in the "`MESSAG`" and "`ASCII`" subroutines.

```
<FunctionName>
      MOVF _FCV_MV_OFFSET, W
      ANDLW H'0F'             ; AND to get nibble
      MOVWF _FCV_MV_OFFSET

      MOVLW LOW <TableName    ; get low 8 bits of address
      ADDWF _FCV_MV_OFFSET, F ; do an 8-bit add operation
      MOVLW HIGH <TableName   ; get high 5 bits of address
      BTFSC STATUS, C               ; page crossed?
      ADDLW D'01'             ; yes then increment high address
      MOVWF PCLATH            ; load high address in latch
      MOVF  _FCV_MV_OFFSET, W ; load computed offset in w reg
      MOVWF PCL               ; load computed offset in PCL
            MOVWF _FCV_MV_DECVAL
<TableName>:
      RETLW 'A'                     ; 1st value offset = 0
      RETLW 'B'                     ; 2nd value offset = 1
      RETLW 'C'                     ; 3rd value offset = 2 etc…
```

For 'C' and Flowcode the function will need wrapping in an `asm{...}` construct.
Flowcode users will need to add the code to their program in a 'C' icon.

The function can be called the same as any of the other library functions:

```
      asm
      {
            CALL  < FunctionName >
      }
```

(where `< FunctionName >` is the name of the your function).

6 Heart rate monitor worksheet

Equipment required
Development board, Heart rate sensor, PICmicro device with A/D converters on board – e.g. PIC16F874 and PIC16F877. This can be carried out in C, Assembly or Flowcode.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of how to convert an analogue quantity into a digital one. An understanding of PICmicro architecture and the architecture of the development board. Binary and hexadecimal.

Objectives
To understand the process of converting a time-varying analogue quantity into a digital one
To carry out simple signal processing to turn a varying analogue quantity into a digital one
To manipulate this data to provide meaningful readings for human beings
To develop this work into a meaningful system such as a game or a medical instrument

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should e recorded in you workbook.

Exercises
Level 1

1. Configure the jumper setting on J14 for analogue work. Make sure you have an appropriate PICmicro device in the development board. Use the crystal clock on the development board.
2. Make sure you are able to convert an analogue signal into a digital value by using the on-board potentiometer RV3. Write a simple program that displays the value on the LCD display.
3. Read the information supplied with the heart rate sensor
4. Make sure you understand how the heart rate sensor is connected to the development board – you will need to refer to the development board datasheet for this.
5. With the heart rate sensor round your chest and the receiver plugged in to the development board attach an oscilloscope to A4. Make sure you can see the signal on the oscilloscope. Refer to the sensor data sheet if you get problems. Look at the voltage levels and decide on a suitable threshold for detecting the signal.
6. Devise a simple program that reads the analogue value on A3 and flashes an LED on port B every time a beat signal is detected. To do this continually read the value from the heart rate monitor and light the LED only if the value is above a certain threshold value (try 100 first). If the LED flashing seems too erratic or if it is not flashing at all, you should try a different threshold value.

Level 2

a) Using interrupts or another timing mechanism, create a program to calculate the time between pulses. Display this value on the LCD display.

b) Manipulate this value of time between pulses to provide a reading for the number of beats per minute.

Level 3

a) By averaging the beats per minute for 5 or 10 beats, a more constant and accurate value of pulse rate can be calculated.
b) Once a valid display of beats per minute can be achieved, create a program that monitors heart rate and lights LEDs if it is too fast or too slow.

Macros required (see Using Macros page)

a) Display a decimal value onto LCD display.
b) Read an analogue value from sensor on A3 (as an 8-bit number)

7 Temperature probe worksheet

Equipment required
Development board, temperature probe, PICmicro device with A/D converters on board – e.g. PIC16F874 and PIC16F877. This can be carried out in C, Assembly or Flowcode. Freezer spray would be useful.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of how to convert an analogue quantity into a digital one. An understanding of PICmicro architecture and the architecture of the development board. Binary and hexadecimal.

Objectives
To understand the process of converting a simple analogue quantity into a digital one
To develop a 'control system' based on this data
To use look up tables to manipulate this data to provide meaningful readings for human beings

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should e recorded in you workbook.

Exercises
Level 1

1. Configure the jumper setting on J14 for analogue work. Make sure you have an appropriate PICmicro device in the development board. Use the crystal clock on the development board.
2. Make sure you are able to convert an analogue signal into a digital value by using the on-board potentiometer RV3. Write a simple program that displays the value on the LCD display.
3. Read the information supplied with the temperature sensor
4. Make sure you understand how the temperature sensor is connected to the development board – you will need to refer to the development board datasheet for this.
5. With the temperature sensor plugged in to the development board attach an oscilloscope to A4. Make sure you can see the DC signal vary as you warm up, or cool down, the sensor.
6. Using the code/macros provided, display the value from the temperature probe onto the LCD display. Update this value regularly (e.g. every second). Note that this is an uncalibrated value and does not represent the temperature in °C or °F.
7. Create a program that monitors the temperature of a bath and lets you know if it is too cold or too hot. Firstly, choose an optimum temperature for your bath (say 50°C) and heat a beaker of water to this temperature (use a thermometer to measure the temperature). With your program running, note down the value of the temperature probe when it is placed in this beaker of hot water (note that you will need to wait a few seconds for the value to settle). You then need to choose temperatures where the water is too hot and too cold (selecting 55°C and 45°C respectively should work OK) and similarly note down the value of the temperature probe when placed in water of these temperatures. Once you have calculated these threshold values, you can create a program that lights an LED (e.g. B7) when the

temperature is too cold, one when it is just right (e.g. B6) and another when it is too hot (e.g. B5). Using a cup of boiling water and one of cold water, try to create a "bath" around 50°C in another container using your program to see if the temperature is too hot or too cold. When you think you have created an optimum bath, check its temperature with a thermometer.

Level 2

8. Construct a temperature chart for 0°C to 100°C using the data on the temperature probe or by calibrating the probe with a known thermometer.
9. Place the probe in a cup of boiling water and remove the heat from the beaker so it begins to cool. Note down the reading of the temperature probe at regular intervals (e.g. every xxxx seconds for about xxxx minutes). Convert each of these values to °C using your chart and plot the results on graph paper. Repeat this experiment to see if you can speed up the cooling process by blowing on the cooling water or by placing next to an open window.

Level 3

10. There are lots of applications for monitoring temperature once you have created a calibration chart.
11. Developing the temperature display program to display actual temperature in °C is a valid extension, but this can be tricky due to the complexity of PICmicro programming. The best way would be to form a look-up table (see Using Macros page).

Macros required (see Using Macros page)

a) Display a decimal value onto LCD display.
b) Read an analogue value from sensor on A3 (as an 8-bit number)
c) Example code for creating a computed-GOTO jump table.

8 Photogate sensor worksheet

Equipment required
Development board, Photogate sensor and smart pulley, standard PICmicro device– e.g. PIC16F84. This can be carried out in C, Assembly or Flowcode. Spring, string and weights.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of PICmicro architecture and the architecture of the development board.

Objectives
To understand the process of detecting a digital signal
To monitor and record time-varying digital signals
To convert time-varying digital signals into meaningful data for human beings
To carry our more complex processing on irregular digital signals

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should e recorded in you workbook.

Exercises
Level 1
1. Configure the jumper setting on J14 for digital work. Make sure you have an appropriate PICmicro device in the development board. Use the crystal clock on the development board.
2. With the photogate plugged in to the development board attach an oscilloscope to A4. Make sure you can see the DC signal vary as you block the infrared beam of the photogate.
3. The photogate acts like a simple switch attached to pin A4 of the PICmicro. When the beam is broken, the value of A4 is low (zero). When it is not broken, A4 is high. Write a small program that mirrors the status of this sensor input onto B4.
4. Extend this program to create a simple burglar alarm. When the signal is broken, light all of the LED's on Port B and keep them on.
5. We will also need to turn the alarm off. Do this by resetting the state of the alarm (i.e. turn the Port B LED's off) when A0 is pressed.

Level 2
6. Attach the smart pulley to the photogate. Write a program to count the number of pulses detected on A4 and display this value on the LCD display (see the Using Macros sections for some code to help you do this). Make this reset to zero each time a switch on the development board is pressed.
7. Alter you program to actually count the number of revolutions the pulley wheel makes.
8. Calculate the circumference of the pulley wheel. Develop a program that measures the approximate distance travelled when the smart pulley and photogate are run along the desk. Think of applications for this technology (albeit with more rugged components) you have developed.

Level 3

9. The burglar alarm concept can be enhanced to provide things such as flashing lights and an audio output, a countdown before the alarm is activated and even some kind of keyed sequence to disable the alarm.
10. The pulley wheel concept can be extended to measure rotational speed. To do this you will need to calculate the time interval between the pulses on A4 using PICmicro interrupts.

Macros required (see Using Macros page)
Display a value as a decimal on the LCD display.

9 Motion detector worksheet

Equipment required
Development board, Motion detector, standard PICmicro device– e.g. PIC16F84. This can be carried out in C, Assembly or Flowcode. A football and a tape measure.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of PICmicro architecture and the architecture of the development board.

Objectives
To understand and use an active sensor in simple systems
To develop systems based on an active sensor that have real practical value

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should e recorded in you workbook.

Exercises
Level 1
1. Configure the jumper setting on J14 for digital work. Make sure you have an appropriate PICmicro device in the development board. Use the crystal clock on the development board.
2. With the motion sensor plugged in to the development board attach an oscilloscope to A4.
3. Write a small program that simply emits a pulse on A2 every second. Make sure you can hear the sensor click and that you can see the output on A2 and the input on A4.
4. Using the code/macros provided, display the distance value from the sensor onto the LCD display. Update this value regularly (e.g. every second).
5. Extend this code so that an alarm (e.g. pin A0) is set when something is close to the motion sensor. You will also need to use a switch (e.g. on pin A1) to reset the alarm.

Level 2
6. In exercise 4 the value of distance displayed on the LCD is not in any specific units - it would be more useful if the value displayed was in centimetres. Create a table of the displayed value and the actual distance in cm and see if you can work out a formula relating them (hint: it should be a simple linear relationship).
7. Using this formula, alter your program to display the distance in actual centimetres and check that the value displayed is correct. Due to the limitation of using 8-bit numbers (0 - 255), you may only be able to work in a range of distances. We suggest calculating a value where 1 represents 10cm and displaying an extra zero after the number (you should then be able to display values between 40cm and 2550cm).

Level 3
8. See "further work" in the Photogate Sensor worksheet for ideas on extending the burglar alarm concept.
9. By using 2 bytes to represent the distance, exercise 2(b) can be extended to display the full range of distances (up to 6m). Altering the code/macro provided could also allow you to have greater accuracy.

10. Now that distance can be measured, velocity can be measured by sampling the distance at faster regular intervals (e.g. 10ms).

Macros required (see Using Macros page)

a) Display a decimal value onto LCD display.
b) Read distance value from sensor (as an 8-bit number)

10 Stepper motor worksheet

Equipment required
Development board, standard PICmicro device– e.g. PIC16F84. This can be carried out in C, Assembly or Flowcode. Actuators panel.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of PICmicro architecture and the architecture of the development board. Binary.

Objectives
To teach students how to control a stepper motor and make it part of a system.

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high-level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should be recorded in you workbook.

Exercises
Attach the actuators panel to port B of the development board. Attach power to the actuators panel. Make sure the actuators switch is in the ON position.

Level 1

1. Design a simple program, which uses the LEDs on B0 to B3 as a simple light chaser with 4 states: 0001, 0010, 0100, 1000.
2. Alter the counting sequence to 1010, 0011, 0101, 1100 using a look up table or 'if' statements. You should now see the stepper motor continuously turning.

Level 2

3. Use a switch on Port A to step the motor turn step-by-step clockwise when the button is pressed.
4. Use another switch on Port A to step the motor turn step-by-step anti-clockwise when the button is pressed
5. Write a new program, which uses two switches to turn the motor forwards and backwards by one 15-degree step for each press of each switch.

11 Servo motor worksheet

Equipment required

Development board, standard PICmicro device– e.g. PIC16F84. This can be carried out in C, Assembly or Flowcode. Actuators panel.

Assumed knowledge

A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of PICmicro architecture and the architecture of the development board. Binary.

Objectives

To teach students how to control a servo motor and make it part of a system.

Presenting your work

In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high-level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should be recorded in you workbook.

Exercises

Attach the actuators panel to port B of the development board. Attach power to the actuators panel. Make sure the actuators switch is in the ON position.

Level 1

1. Design a simple program that pulses B4 for 1millisecond every 20milliseconds (i.e. 1millisecond on, 19milliseconds off). Attach an oscilloscope to B4 to check your results. This is the 0 degree point of the servo.
2. Modify the program so that B4 is at 5V for 1.5milliseconds and at 0V for 18.5milliseconds. Attach an oscilloscope to B4 to check your results. Your servo should now be positioned at 90 degrees.
3. Use switches on Port A to modify the pulse duration between 1 and 2 milliseconds within the 20millisecond period.

12 DC motor worksheet

Equipment required
Development board, standard PICmicro device– e.g. PIC16F84. This can be carried out in C, Assembly or Flowcode. Actuators panel.

Assumed knowledge
A reasonable level of competence in coding for a PICmicro microcontroller in the language of your choice – assembly, C, or Flowcode. An understanding of PICmicro architecture and the architecture of the development board. Binary.

Objectives
To teach students how to control a DC motor and make it part of a system.
To teach students how to control DC motor speed
To teach students how to use DC motor feedback for positioning and speed control

Presenting your work
In developing your work you should keep copies of all print outs of code or flowcharts. If you are using a low level language (Assembly or C) then you should produce high-level system diagrams showing the algorithms used. All code should be properly commented. For each of the tasks below keep a written record of problems encountered and their solutions. Make sure you keep copies of each program for each level. Be sure to make notes on the exercises as you work through them. Your programs, your notes and your comments should be recorded in you workbook.

Exercises
Attach the actuators panel to port B of the development board. Attach power to the actuators panel. Make sure the actuators switch is in the ON position.

Level 1

1. Design a simple program with two switches on port A that control the output levels of B5 and B6. Clicking on the switches should make the DC motor go either forward and backwards.
2. Add two further switches on Port A to make the motor go forwards and backwards at one quarter speed Using Pulse Width Modulation.

Level 2

3. Using the feedback from the opto-sensor on B7 make a program that turns the DC motor through two complete clockwise revolution when a switch on port A is pressed. (Note – you may require the disc to move slightly upon start up for accuracy). Monitor how fast your motor stops and how close you get to one complete revolution.
4. Using feedback from the opto-sensor on B7 make your motor slow down as it is about to reach its target. Monitor how fast your and how close you get to two complete revolutions and the difference in accuracy between the two programs.