Flowcode

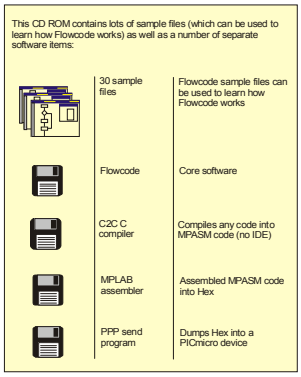
Allows those with no programming skills to design complex electronic systems

Flowcode is a very high level language programming system for PICmicro® microcontrollers based on flowcharts. Flowcode allows students to design and simulate complex robotics and control systems in a matter of minutes.

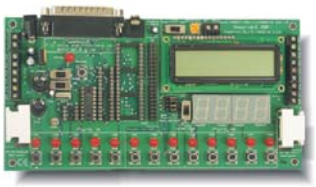
Flowcode is a powerful language that uses macros to facilitate the control of complex devices like 7-sement displays, motor controllers, and LCD displays. The use of macros allows students to control highly complex electronic devices without getting bogged down in understanding the programming involved.

A complementary development board allows Flowcode to program virtually any PICmicro microcontroller and easily build complex projects.

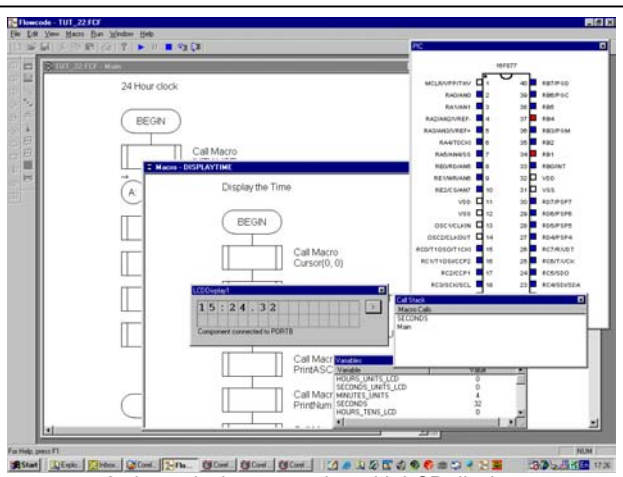
A demonstration version is available from the downloads section of our web site.



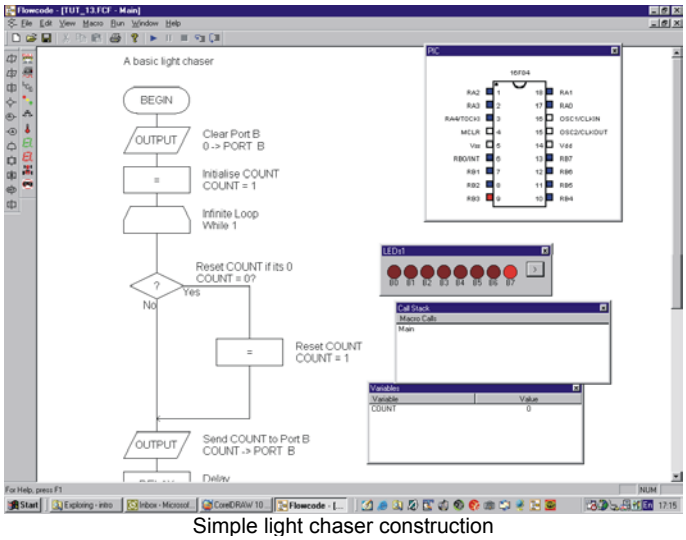
The software on the CD



When used with the development board this CD ROM provides a simple working environment for learning and programming PICmicro microcontrollers



24 hour clock construction with LCD display



Simple light chaser construction

- Requires no programming experience
- Allows complex PICmicro microcontrollers to be designed quickly
- Uses international standard flow chart symbols (ISO5807)
- Full on-screen simulation allows debugging and speeds up the development process
- Facilitates learning via full suite of demonstration tutorials and virtual systems (burglar alarms etc.)
- Produces ASM code for a range of 18, 28 and 40 pin devices
- Works with most programmers
- Can be used to teach PLC programming skills
- Allows C code or assembly code to be embedded as a macro
- Supports interrupts and A/D converters

Matrix Multimedia Limited
Tel 0870 700 1831 Fax 0870 700 1832
sales@matrixmultimedia.co.uk w ww.matrixmultimedia.co.uk

Specifications

General Functions

Menu link	Menu items
FILE	NEW, OPEN, CLOSE, SAVE, SAVE AS, PRINT, PRINT PREVIEW, PRINT SETUP,TARGET PIC
EDIT	CUT, COPY, PASTE, DELETE, VARIABLES, KEY MAPPING, PROPERTIES
VIEW	COMMAND TOOLBOX, COMPONENTS TOOLBOX, PIC, VARIABLES, CALL STACK, ATTACHED COMPONENTS, ANALOGUE INPUTS, TOOL BAR, STATUS BAR
MACRO	NEW, EDIT, DELETE, EXPORT, IMPORT
RUN	GO/CONTINUE, STEP INTO, STEP OVER, PAUSE, STOP, CLOCK SPEED, COMPILE TO PIC, COMPILER OPTIONS
WINDOW	CASCADE, TILE, ARRANGE ICONS
HELP	HELP TOPICS, ABOUT Flowcode

Flow chart icons

Icon	Specification
Input	Input on any PICmicro pin of logic level or variable on any port on the device. Optional masking
Output	Output on any PICmicro PIN of logic level or variable
Delay	Absolute value in milliseconds/seconds or delay from a variable
Decision	Decision making branch based on a calculation in which variables can be used along with the following operators: (,) - Parentheses. = , <> -Equal to, Not equal to. +, -, *, /, ~, % - Addition, Subtraction, Multiplication, Division, Inversion & Modulus. <, <=, >, >= - Less than, Less than or equal to, Greater than, Greater than or equal to. >>, < < - Shift right, Shift left. Also Boolean operators AND, OR, XOR, NOT
Connection points	Connection points facilitate a 'GOTO' connection in flow charts.
Calculation	Calculation in which variables can be used with the following operators: (,) - Parentheses. = , <> -Equal to, Not equal to. +, -, *, /, ~, % - Addition, Subtraction, Multiplication, Division, Inversion & Modulus. <, <=, >, >= - Less than, Less than or equal to, Greater than, Greater than or equal to. >>, < - Shift right, Shift left. Also Boolean operators AND, OR, XOR, NOT
Interrupts	Three types of interrupt are supported: RB0/INT - external interrupt on RB0. RB0 interrupt Macro is called.

	Port B bits 4-8 – interrupts when there is a change in the state of bits 4 to 8. Port B interrupt is called. Timer - interrupt via internal timer with prescaler control. Timer interrupt macro is called.
Code embed	Allows any C code or assembly code to be embedded to have access to more complex functions.Variables can be passed to both C and ASM
Loop	Easily facilitates icons to be embedded in a loop for simple counters. Decision making is based on a calculation in which variables can be used along with the following operators: (,) - Parentheses. = , <> -Equal to, Not equal to. +, -, *, /, ~, % - Addition, Subtraction, Multiplication, Division, Inversion & Modulus. <, <=, >, >= - Less than, Less than or equal to, Greater than, Greater than or equal to. >>, < - Shift right, Shift left. Also Boolean operators AND, OR, EOR, NOT
Breakpoint	Breakpoints can be set to limit code execution.
Macro	See below.

Macro functions

The macro function allows a sheet of flowchart programming to be called from another flowchart as a single icon. This facility is designed to allow hierarchy in the flow chart systems which makes using flowcharts easier. This scaleability also allows complex designs to be greatly simplified.

The macro function also has several special invocations:

RB0 Interrupt	Interrupts from RB0 call the RB0 macro flowchart sheet
Timer interrupt	Interrupts from the timer call the RB0 macro flowchart sheet
I/O device macro	Allows calls to be made to on-screen components which are developed in Visual Basic. Several components are supplied with the basic version of Flowcode (switches, 7-segment displays etc.) and others can be designed using Visual Basic.

All variables are available in macros.

On-screen components

The following are supplies as standard - others may be available from www.matrixmultimedia.co.uk at a later date.

Component	Description

	declare them as outputs.
Quad 7-segment display	Quad 7-segment display in common anode configuration (each segment will be activated by a low voltage on the PICmicro). 8 pins are required for each segment and 4 further pins are used to 'strobe' which 7-segment display is used. Connecting the single 7-segment display to pins on the PICmicro will declare them as outputs.

Variables

All variables are 8 bit positive.
10 bit analogue inputs are read as high byte, low byte.

Compiler, assembler and send packages

Flowcode is built on a C compiler - C2C. This is a general purpose 8/16 bit compiler designed specifically for PICmicro devices.

Flowcode assembles a C code file from the flow chart. This is automatically compiled and assembled using Arizona Microchip's MPASM assembler into an ASM file.

Any third party PIC programmer can then be used to send the resulting file into the target PICmicro.

If Flowcode is used with Matrix Multimedia's development board then this whole operation of compiling assembling and sending is carried out with one button providing a totally seamless PICmicro development tool.

Versions and licences

Virtual systems are not available with student/home versions.

Licence	Virtual systems	Support
student/home	no	e-mail only
single user	yes	e-mail/phone
site	yes	e-mail/phone

Target devices

PIC16C62, PIC16C620, PIC16C621, PIC16C622, PIC16C63, PIC16C64, PIC16C65, PIC16C65A, PIC16C66, PIC16C67, PIC16C71, PIC16C710, PIC16C712, PIC16C716, PIC16C72, PIC16C73, PIC16C74, PIC16C76, PIC16C77, PIC16CE623, PIC16CE624, PIC16CE625, PIC16C627, PIC16C628, PIC16F84, PIc16F870, PIC16F871, PIC16F872, PIC16F873, PIC16F874, PIC16F876, PIC16F877

Technical specification

Flowcode operates on all Windows™ 98, 2000, NT, ME, XP. Network versions are available.

Tutorials

There is no formal course with Flowcode. Instead we have provided a comprehensive help file and a suite of 28 tutorials that novices can work through to gain an understanding of how Flowcode works. The complete list of tutorials is:

TUT_01.FCF
Tutorial 1: Lighting an LED
Lighting up an LED (A0). Demonstrates how to send output to a port.
Clears PORT A by sending 0 to the port, which turns off all the LEDs.
Then sends 1 to PORT A, which lights LED A0

TUT_02.FCF
Tutorial 2: Outputting a value to a port.
Sends 5 to PORT A, which lights up two LEDs.
The two LEDs correspond to the binary pattern for 5.

Try this:
Change the value in the OUTPUT icon and see what affect this has on the LEDs

TUT_03.FCF
Tutorial 3: Single bits and ports.
Sends 1 to each individual bit of PORT A, then clears the port. Finally it lights up all of PORT A.
Sending 1 to a specific bit will light that bit. Sending a value will light the corresponding pattern of bits.
Try this:
Change the order of how the LEDs light up.
Change which LEDs light up.

TUT_04.FCF
Tutorial 4: Using variables.

Sends MY_OUTPUT to PORT A.
When output to a port variables work the same as values.
Try this:
Change the value that is assigned to MY_OUTPUT in the Calculation icon.
Add a new variable OUTPUT_A and change the calculation to use this variable instead.

TUT_05.FCF
Tutorial 5: Basic calculations.
Performs basic calculations and sends the result to PORT B.
Try this:
Change the calculation, or add new ones.
Add a second variable and use this to try calculations with more than one variable.

TUT_06.FCF
Tutorial 6: Input from switches.
Input is taken from switches on PORT A and displayed on PORT B.
Note that this tutorial also uses connection points to run continuously.
Try this:
Try inputting from a single bit rather than the whole port.

TUT_07.FCF
Tutorial 7: Boolean logic calculations.
Calculates RESULT based on a variable, input from PORT A and the Boolean logic AND function.
Try this:
Change the value of BOOLVAR.

Try other Boolean functions such as OR and XOR.

TUT_08.FCF
Tutorial 8: Using masking.
Passes the input from PORT A to PORT B, but uses a mask to select only certain bits from PORT A.
Try this:
Change what is being masked.
See what happens when you mask the output.

TUT_09.FCF
Tutorial 9: A basic counter.
A basic counter on PORT B.
Note that once the counter reaches 255 (all LEDs on) it overflows and starts again at 0.

TUT_10.FCF

Tutorial 10: A timed counter.
As Tutorial 9, but with a 1 second delay to aid timing
Try this:
Change the delay to speed up or slow down the count.

TUT_11.FCF
Tutorial 11: Using loops.

This counter uses a loop to count up to 16 on PORT A.
Try this:
Change the LEDs to PORT B and Increase the counter.

TUT_12.FCF
Tutorial 12: A basic light chaser.
A light moves across the row of LEDs
By multiplying by 2 the next binary number is activated and the light appears to move.

Try this:
Can you make it work in reverse?

TUT_13.FCF
Tutorial 13: Improved light chaser.
In the previous tutorial the light appeared to fall off the end of the LEDs. So we have added a simply decision box to see if its fallen off, and if so re-initialise the count.
Try this:
How would you make this work in reverse?

TUT_14.FCF
Tutorial 14: Bitwise shifting.
Another light chaser, but this one uses bitshifting.
Bitshifting actually moves the bits along to the next one.

Try this:
Change the value of count and see what happens.
Change the direction of the bitshift.

TUT_15.FCF
Tutorial 15: Moving LED display.
Creates a moving LED display using loops.
Try this:
Compare this tutorial and the next one to see different ways of solving the same problems.

TUT_16.FCF
Tutorial 16: Moving LED display.
Creates a moving LED display using decisions and connections.
Try this:
Compare this tutorial and the previous one to see different ways of solving the same problems.

TUT_17.FCF
Tutorial 17: Using a seven segment display.
Displays a number on a seven segment display.
Rather than use complex code to pass the specific inputs that the seven segment display needs to display the number we are using a macro that handles this for us.
Try this:
Look at the macro properties to see how it works.
Change what digit is displayed.
Display the decimal point.

TUT_18.FCF
Tutorial 18: Counting on a seven segment display.

Uses a simple loop to turn the seven segment display into a counter.

TUT_19.FCF
Tutorial 19: Counting on a seven segment display using timers.
In this tutorial we use a timer interrupt to make the seven segment display update every second.
We have set the clock speed to 3276800Hz, and the prescaler value to 1:128
This now gives us a timer interrupt frequency of 25Hz (25 times a second)
We can now edit the TMR0 interrupt to update the display variable every second.

Try this:
Change the clock speed and prescaler values to see how they affect the timer interrupt frequency.
Change the code to work with the new values.

TUT_20.FCF
Tutorial 20: Counting on a quad seven segment display using multiplexing.

The PICmicro can only illuminate one of the four displays at a time. However the PICmicro can light up the four displays one after the other so fast that to the human eye they appear to be continuously lit. This is called multiplexing.

Note that this tutorial uses a macro UPDATE_VALUES.
Macros can be used to neaten up the flowchart by removing large or complex blocks of code, or to separate a piece of code that may be useful elsewhere as well.
Try this:
Change the code so that it no longer multiplexes - what is the display like now?
Create two new macros - one to initialise the display, and one to output to the display.

TUT_21.FCF
Tutorial 21: Using the LCD display.
Uses the LCD display to display a message.

Try this:
Change the message (see the LCD display help page for more details).

TUT_22.FCF
Tutorial 22: A 24 Hour clock.
Displays a 24 hour clock on the LCD display.
Note the use of macros to improve the layout of the code.
Try this:
Change the code to make a 12 hour AM/PM clock.

TUT_23.FCF
Tutorial 23: PORT B0 interrupt.
Uses an interrupt on PORT B0 to increment a counter.
Try this:
Change the code so that the counter updates continually, and is reset by the interrupt.

TUT_24.FCF
Tutorial 24: Generating sound.
Note: Sound will not simulate in Flowcode.
Turns the PORT B switches into piano keys.
Sound is generated by waggling Pin A0.
A headphone or speaker needs to be connected to the audio output for the sound to audible.
Try this:
Alter the TONE values to see how they affect the sound.

TUT_25.FCF
Tutorial 25: Using embedded C and Assembly code.
This tutorial demonstrates the use of embedded C and Assembly code.

Note: C and ASSEMBLY code will not simulate in Flowcode.

TUT_26.FCF
Tutorial 26: Using Analogue inputs.
Note: This tutorial requires a PICmicro with an analogue input, such as a PIC16F877.
Reads the analogue level of the input and outputs this to a LCD display.

TUT_27.FCF
Tutorial 27: Advanced calculations.
This tutorial demonstrates a number of complex calculations.

TUT_28.FCF
Tutorial 28: Using macros.
This tutorial demonstrates the use of macros.