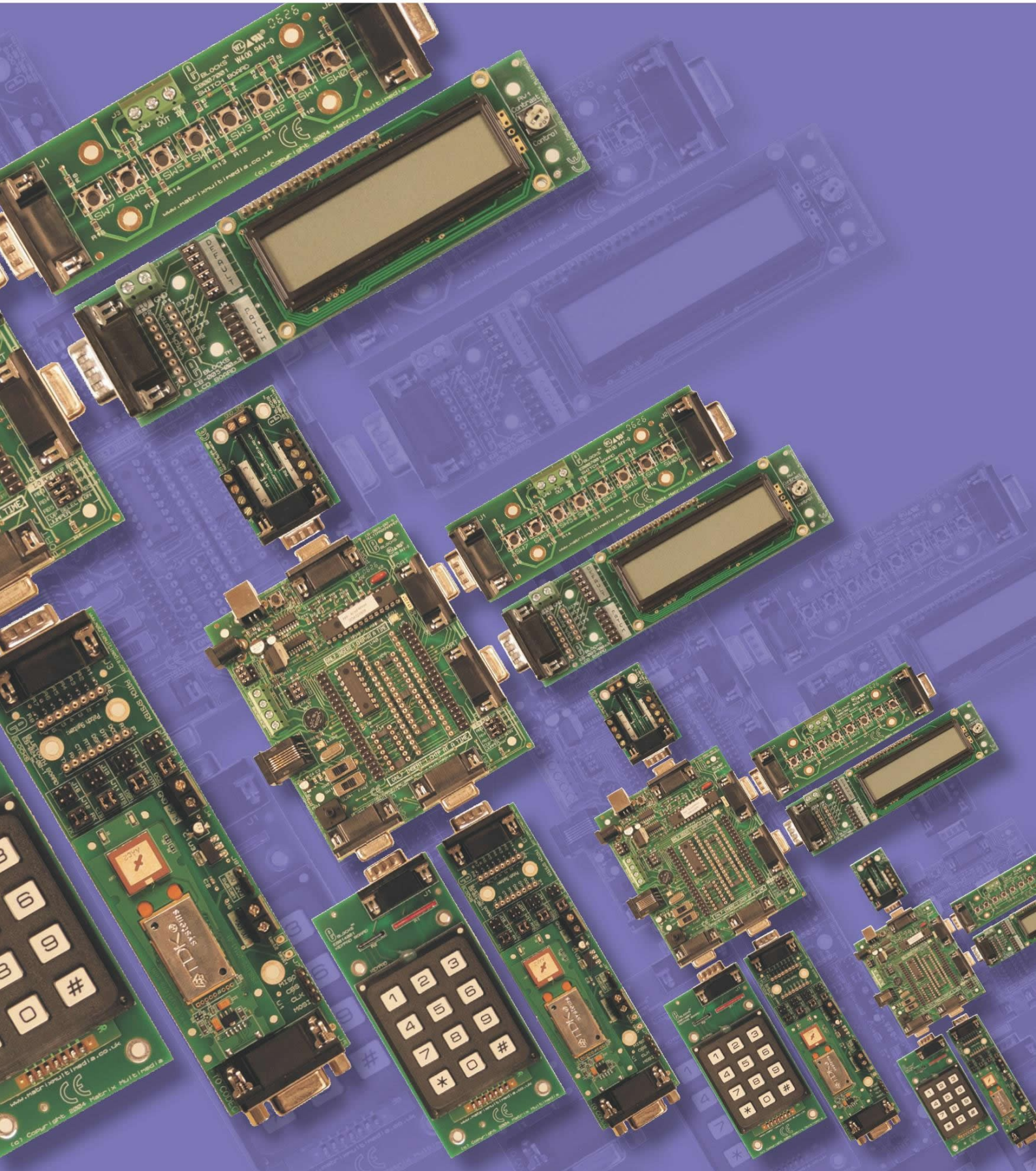
MATRIX

getting started guide

# E-Blocks Datalogger Bundle



# Flowcode

EB833-80-1
**E-Blocks Data-logger Bundle**

### Installing Flowcode

Instruction for installing Flowcode can be found inside the installation booklet located inside the Flowcode DVD case.

Before starting with the course it is recommended to update your version of Flowcode to the latest released version. This allows for the latest bug fixes and components to run on your machine. The latest version of Flowcode can be found by visiting the Matrix TSL website and clicking on the Flowcode page.

http://www.matrixtsl.com/Flowcode3a-X.php

### Getting Started with Flowcode

There is a free online course available for helping with getting started with learning Flowcode. This course covers basic principals through to designing your own programs and programming the devices. It is recommended that you take time to go through this course before proceeding with the bundle exercises to give you a better grasp of what the Flowcode program is doing.

The online course is available from the learning centre on our website or by visiting the following address:

http://www.matrixtsl.com/lc_microcontroller.php

### Flowcode Examples

A number of pre-made example files are available for download from the main Flowcode page on the Matrix TSL website. These files are also located on the Flowcode CD. Before the example files can be used you must first copy them into a folder on your hard drive.

The example programs referenced in this bundle can be found at the following web address:
http://www.matrixtsl.com/lc_bundle_manuals.php

Or by clicking the bundle manuals link from the Learning Centre area of our website.

### Flowcode Help

There is a help file available that covers all the main features of Flowcode. This help file can be accessed by clicking the question mark icon in the main Flowcode toolbar or alternatively clicking the help menu and selecting contents.

There are also help files available for each and every component in Flowcode which explain the functionality of the component and the component macros. The component help files can be found by selecting the component on the panel and the clicking the Help button in the properties toolbar.

### General Support

Support for frequently encountered problems can be found online on our FAQ's site.
Our online forums can also be used as a general discussion area or for help or advice.

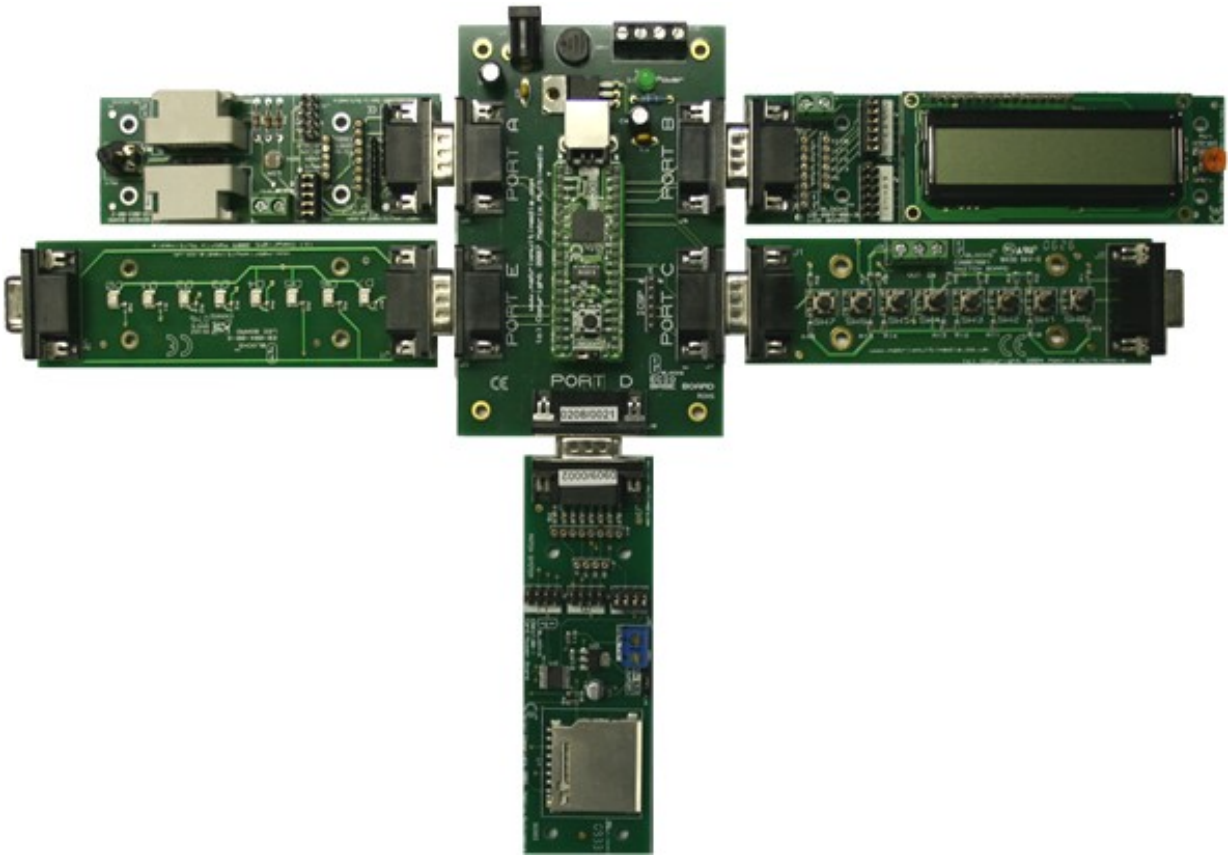
# Wiring & Testing

EB833-80-1

## E-Blocks Data-logger Bundle

To setup your E-Blocks for use with the example programs you must perform the following actions:

- Insert the ECIO40P into the EB061;
- Connect up the E-Blocks as shown below.
- The Card Reader, LCD, Sensor and Switch E-Blocks all need to be connected to the VDD on the EB061 via single core wire.




Before you can begin you must install the driver for the ECIO using the ELSAM CD or by visiting the Matrix TSL website: http://www.matrixtsl.com

The system can be tested by compiling and sending one of the example programs to the hardware. This is done by opening one of the example files in Flowcode and then clicking the compile to chip button.

The example Flowcode files contain a correct configuration so you will not have to modify the configuration to allow them to run on the hardware.

Any program you create from scratch will have to be configured to use the ECIO40 rather then using the standard PICmicro device.

# FAT Component

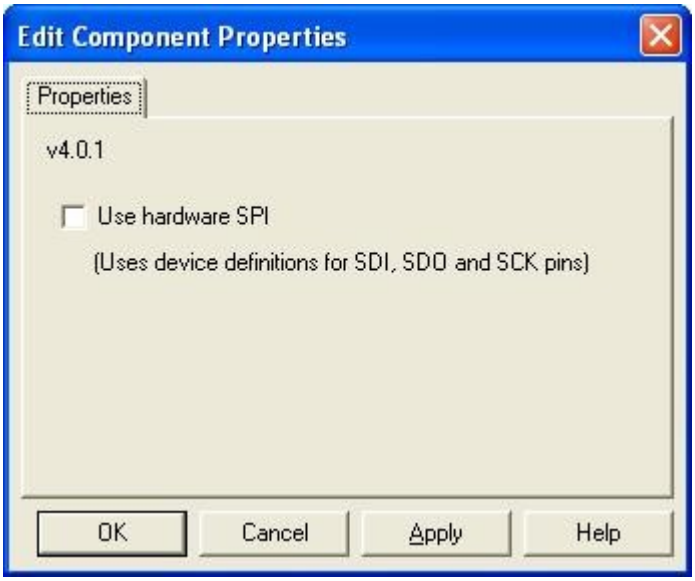
EB833-80-1
**E-Blocks Data-logger Bundle**

The Flowcode FAT16 component can be added to your program by finding the component in the Peripheral section of the Flowcode component toolbar.

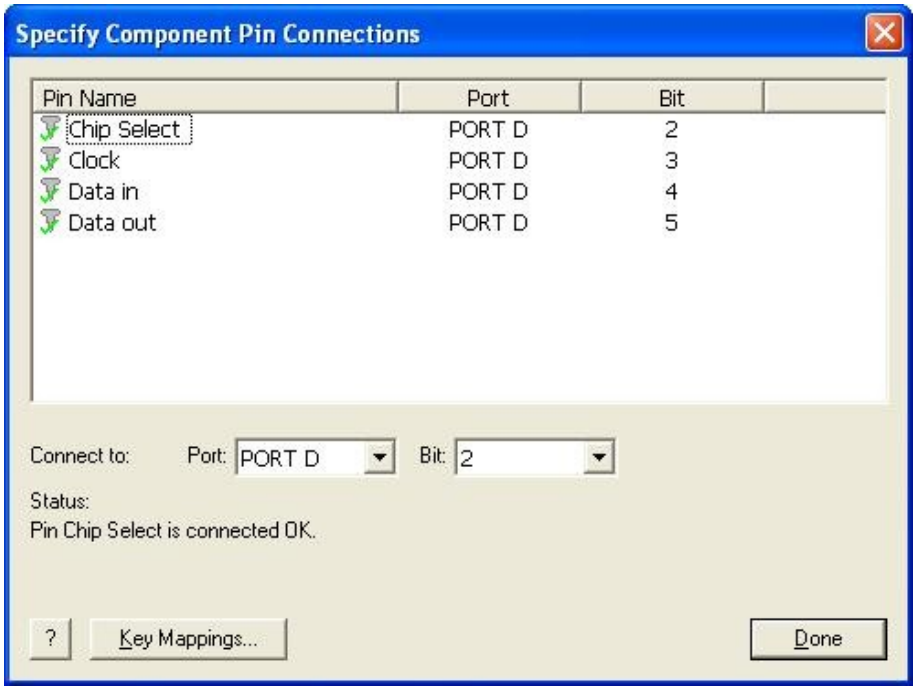

**FAT16Component Icon**

Once the component has been added to your program you will find that the FAT16 component icon has been added to your panel.

The properties for the FAT component simply allow you to turn on or off the hardware SPI. If the hardware SPI is turned off then you can use any four I/O pins to communicate with the memory card. The ECIO has its SPI bus split across two ports so it is not practical to use the hardware SPI on an ECIO device. Instead we leave the tick box un-ticked and specify that the connections are all on Port D.




An extensive help file for the component can be found by selecting the component on the Panel and clicking the help button on the properties sidebar. Note that to view help files in Vista or Windows 7 you will need to download the Microsoft help runtime.



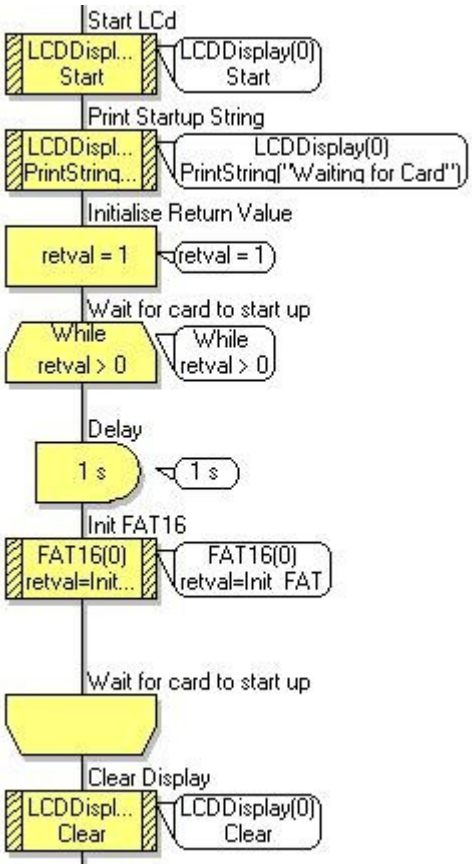

The pin numbers shown here are correct for jumper position A on the Card reader E-Block.

# Example 1

EB833-80-1

## E-Blocks Data-logger Bundle

Exercise 1 shows you how to perform basic write functionality on the memory card by sampling an analogue channel and recording the sampled data onto the card in text form. This example will also demonstrate how to create and open a file and how to check for errors along the way.

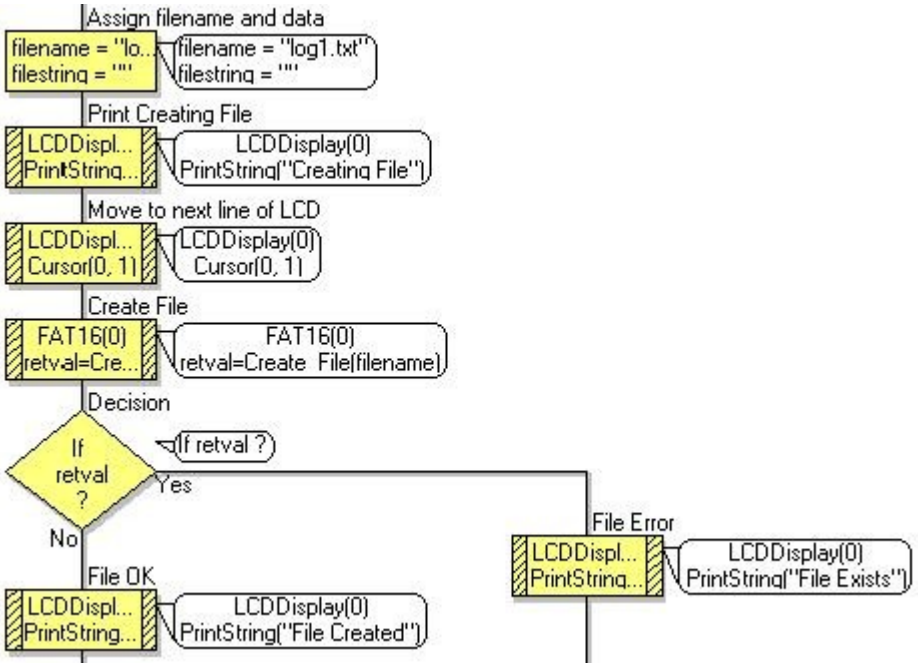



At the start of the program we initialise the LCD and print a message so we know the system is running. We then enter a loop to try and initialise the memory card so we can talk to it. If the card is disconnected or not compatible then we will get a value greater then 0 returned from the hardware macro.

If the initialise function returns a 0 then we know that a card is connected and that communications have been established. The card should now be ready to be manipulated.

You will always start in the root directory after starting up the card.

Creating a new file is fairly simple. Basically you would navigate the card using the Open_Folder and Scan_Current_Folder component macros and then call the Create_File component macro. If the file already exists then the macro will return number that is not 0. If the file was created successfully then the macro will return a 0.

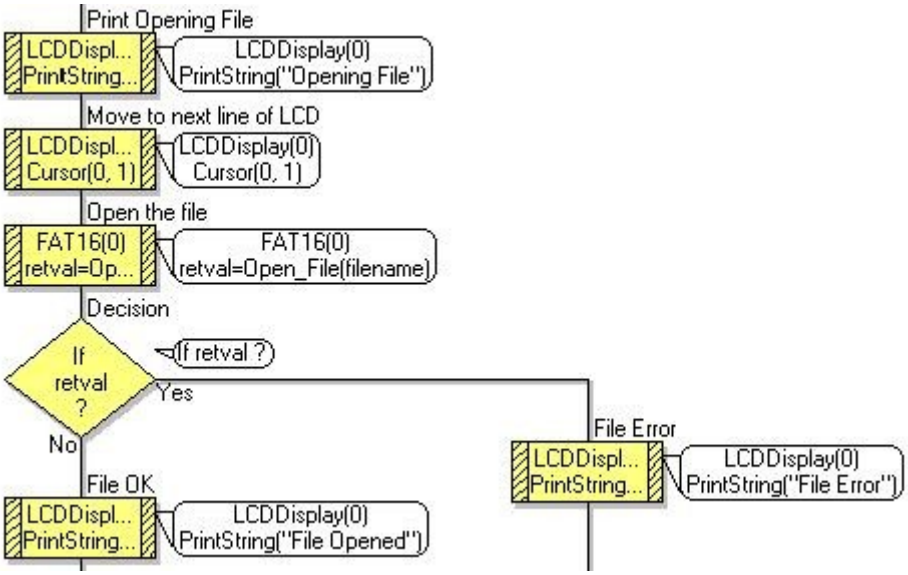


# Example 1

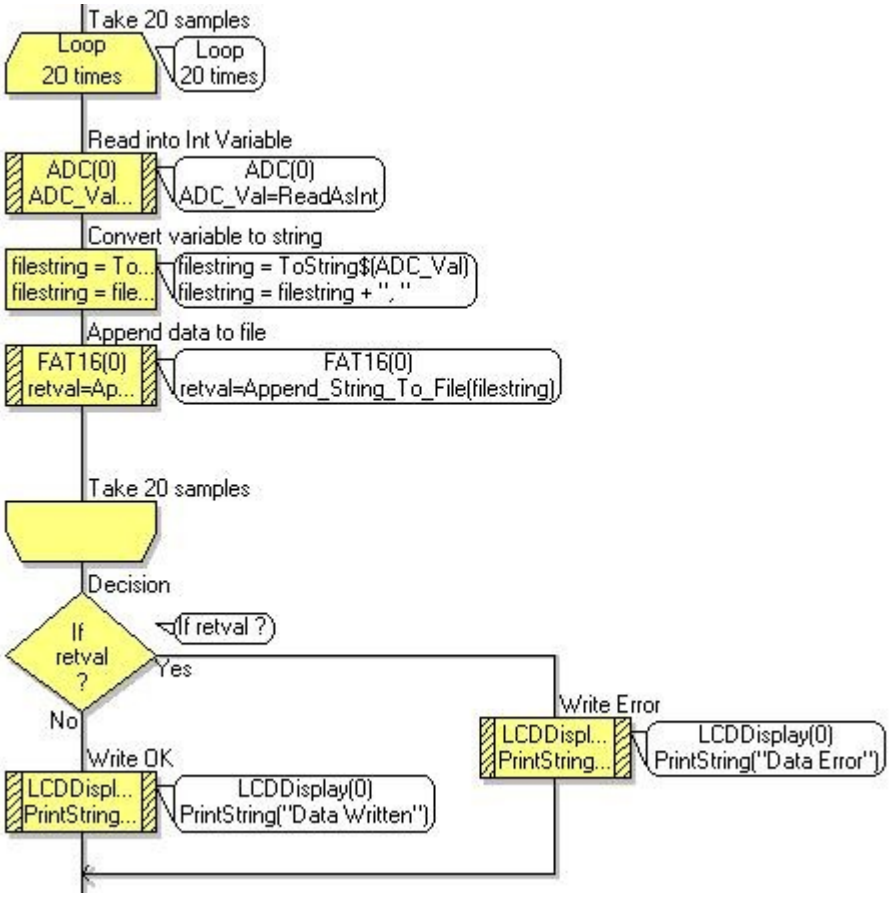
EB833-80-1

## E-Blocks Data-logger Bundle

When working with files you always have to open the file before you can start reading or writing data. Even if you have just created a new file then you must open the file before you can do any data manipulations. Opening a file is as simple as passing the file-name string parameter to the Open_File component macro. Like the other functions this macro also returns a value of 0 if the file was found and opened correctly. Otherwise the macro will return a non 0 value.



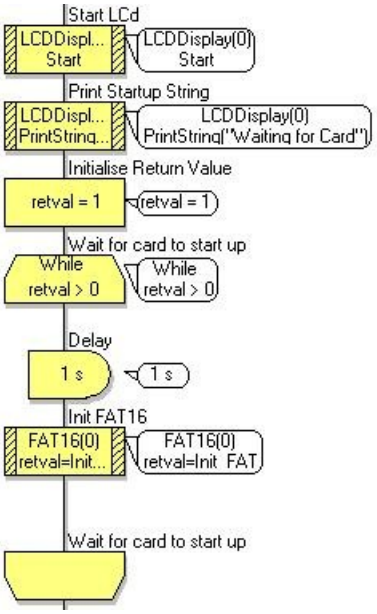

The Append_String_To_File component macro automatically moves to the end of an open file and then appends the string parameter to the end of the file. The file is also updated in size and if need be the file system is manipulated to make room for the additional portions of the file. This function is fairly slow and inefficient as it requires a lot of processing to work out where the end of the file is and then to append the data. This is made worse if the data happens to cross a sector boundary on the disk. The main benefits of this function is that all of the complexity of dealing with FAT16 file systems is removed from the user.



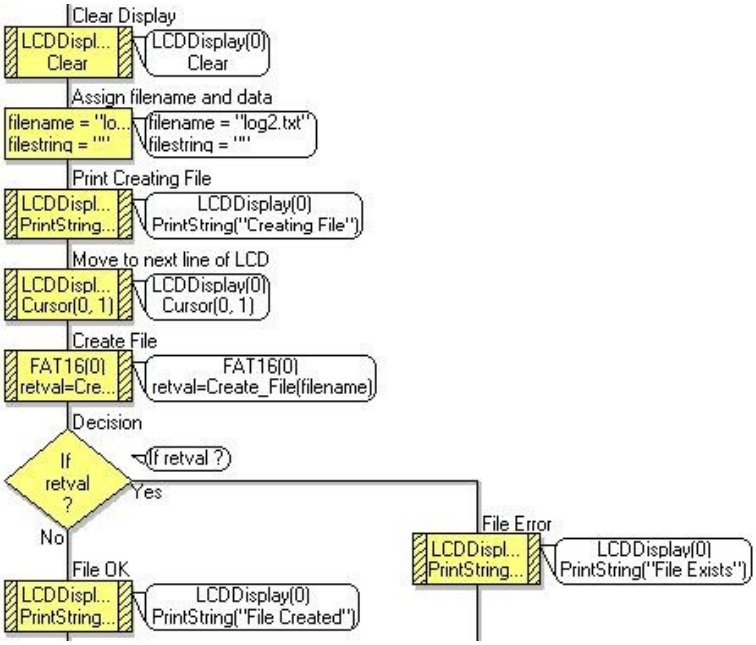
# Example 2

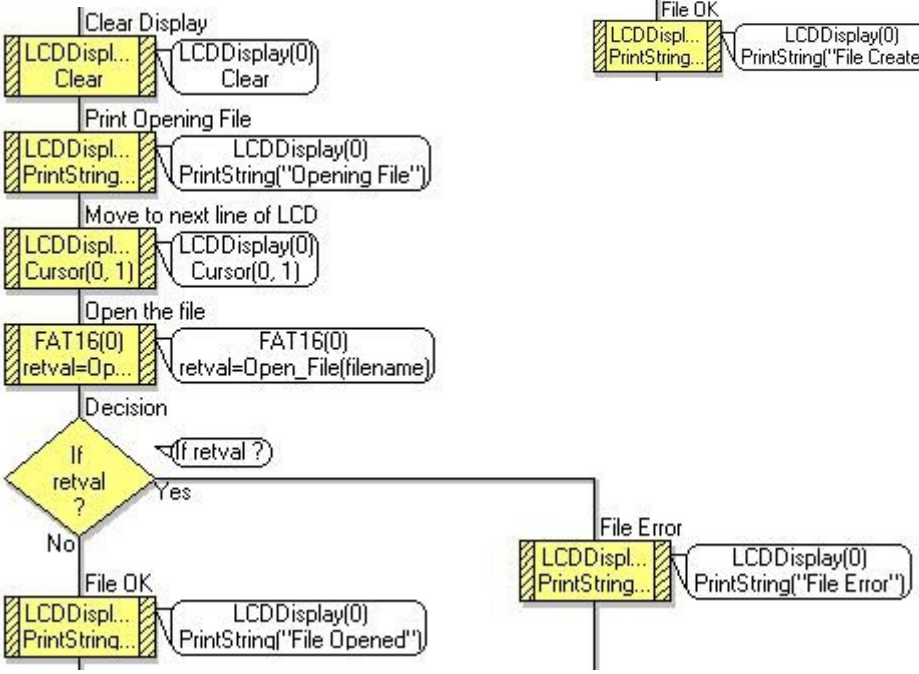
EB833-80-1
**E-Blocks Data-logger Bundle**

Exercise 2 performs the same task as exercise 1 but now instead of using the slow hand holding data append function we will be writing whole sectors to the card using a single transaction. This is much faster and allows for fairly high speed data logging.



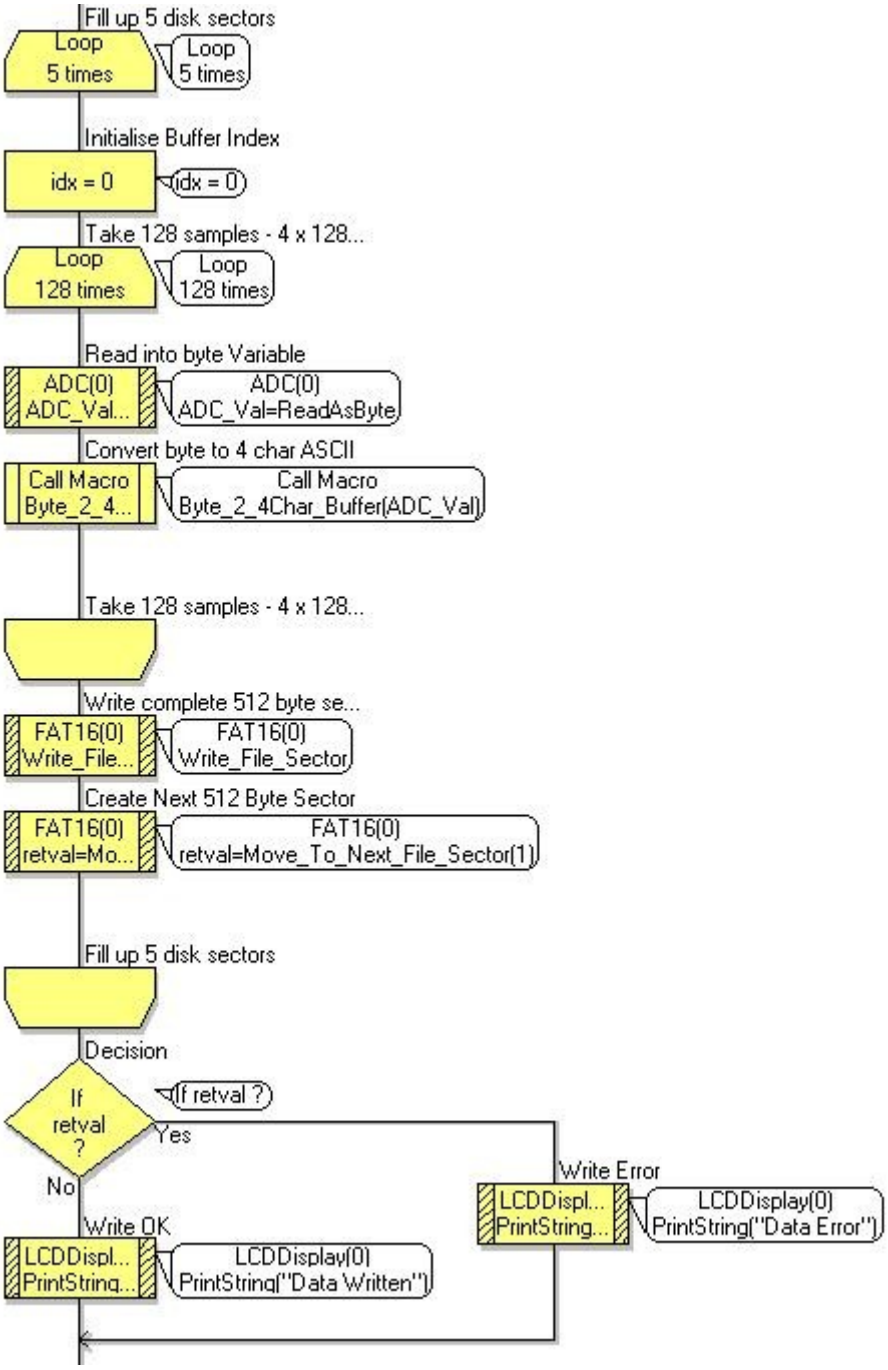

The start of the file is the same as example 1. We first wait for a valid card startup confirming that a card is present and working with the system. Once this is done we create a new file then attempt to open the file. If this all goes well then we move onto writing the data to the file. We will write to a newly created file because this way we are certain that the file contains no data and therefore will not cause a problem when we append a sector to it.



# Example 2

EB833-80-1
**E-Blocks Data-logger Bundle**




The program then attempts to write 5 sectors to the newly created file. Each sector on the disk is 512 bytes so 5 sectors would be the equivalent of writing 2560 bytes.

To ensure that the 512 bytes of the sector are filled exactly, we need to format the string version of the sample data to ensure that no matter what the sample is, there is always a fixed number of characters in the number. The Byte_To_4Char software macro is responsible for ensuring that each ADC sample has three ASCII bytes to represent 0—255 and a final comma to segment the sample data. We then perform 128 samples between each sector write. 128 x 4 = 512 bytes or 1 complete sector.

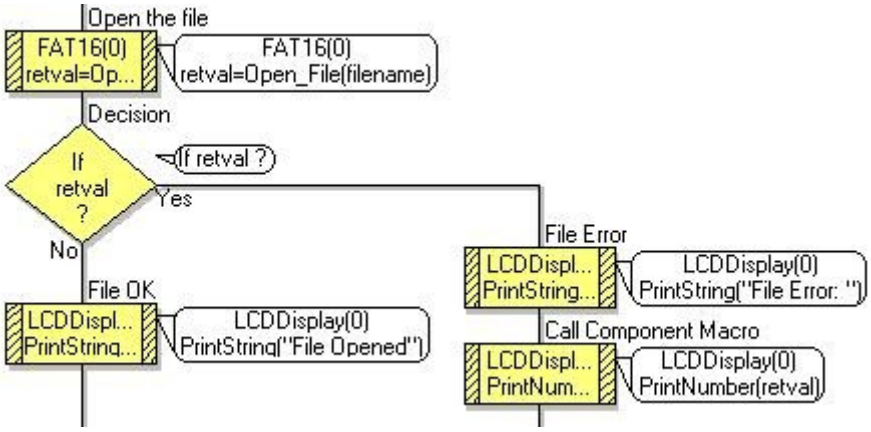
# Example 3

EB833-80-1

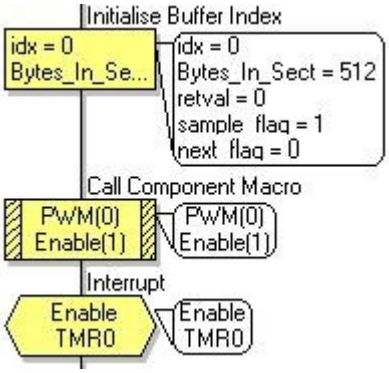
## E-Blocks Data-logger Bundle

Exercise 3 uses a combination of reading from the disk and the Flowcode PWM component to generate a basic audio player. This audio player can stream a file of 800hz to an audio system or headphones using the PWM output component. To allow the file to stream correctly we must set up a timer interrupt to run at exactly 800hz. This means that every time an interrupt occurs we must move to the next byte in the file to continue the audio playback. Because the FAT component is fairly large to allow the file table system to be handled correctly we also have to use a secondary buffer to allow the 800hz output streaming to continue while we perform the function to move to the next sector of the audio file on the card.

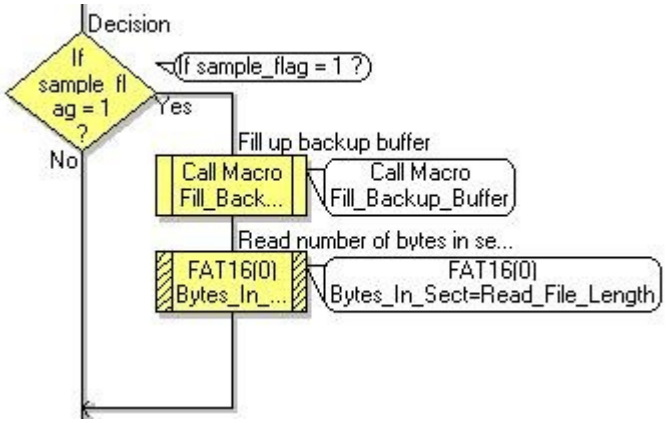
If you want to make your own WAV file recordings then you can simply use Winamp or another audio program to convert your audio file into a 800Hz MONO audio file. Then simply save the file to your SD / MMC card and specify the file name in the Flowcode program. The FAT component only supports DOS file names so you may want to keep the filename below 8 characters to avoid any problems.




The program starts by searching for a card and then tries to open the file specified in the filename variable from the root directory.




If the file is found and opened correctly then we initialise the variables used in the program and start the timer interrupt running before entering a while loop.

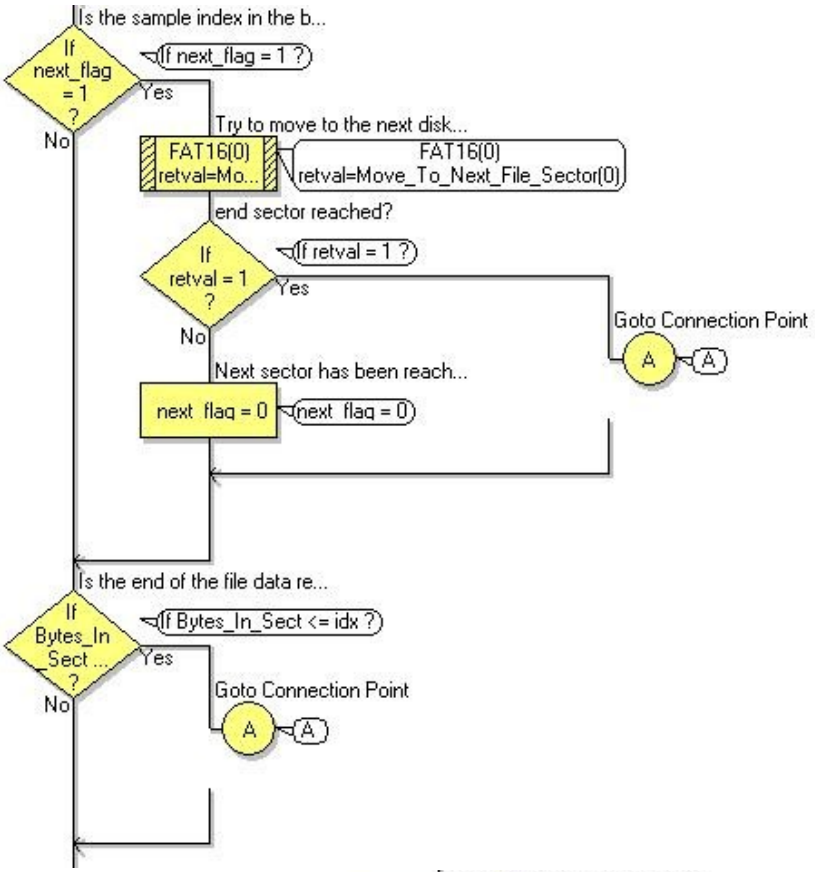



Inside the while loop we start by reading the end of the sector into our local buffer. We do this so that later when we are moving to the next sector of the file we can continue streaming the audio from the local buffer. We also read the number of bytes in the sector to ensure that the sector is not the last sector in the file.

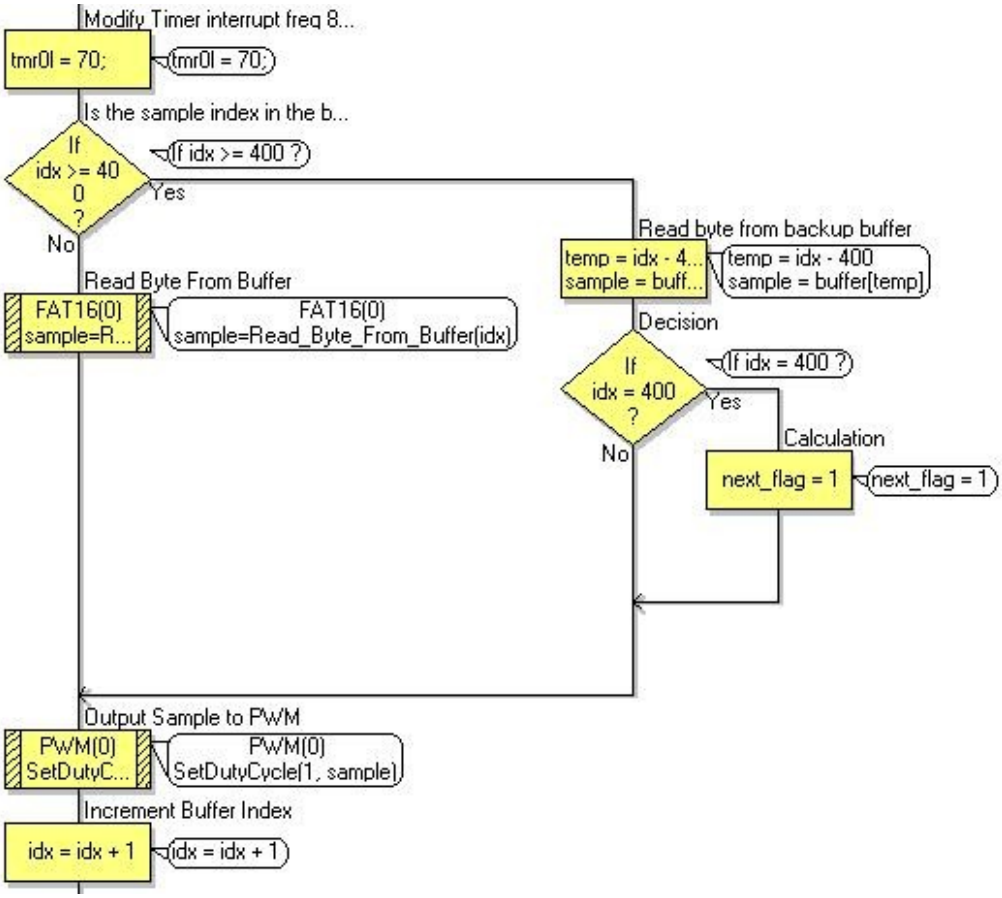
# Example 3

EB833-80-1

## E-Blocks Data-logger Bundle




If we are nearing the end of the sector and the end of the buffer has been stored then we call the function to fetch the next sector of the file. If no further sectors exist then we end the loop and disable the PWM output.




The timer interrupt starts by assigning a value to the timer count register. This allows the interrupt to run at exactly 800hz.

We then decide which buffer to read from based on the current position specified by the idx variable.

After reading the correct buffer position we simply forward the sample to the PWM.

# Troubleshooting

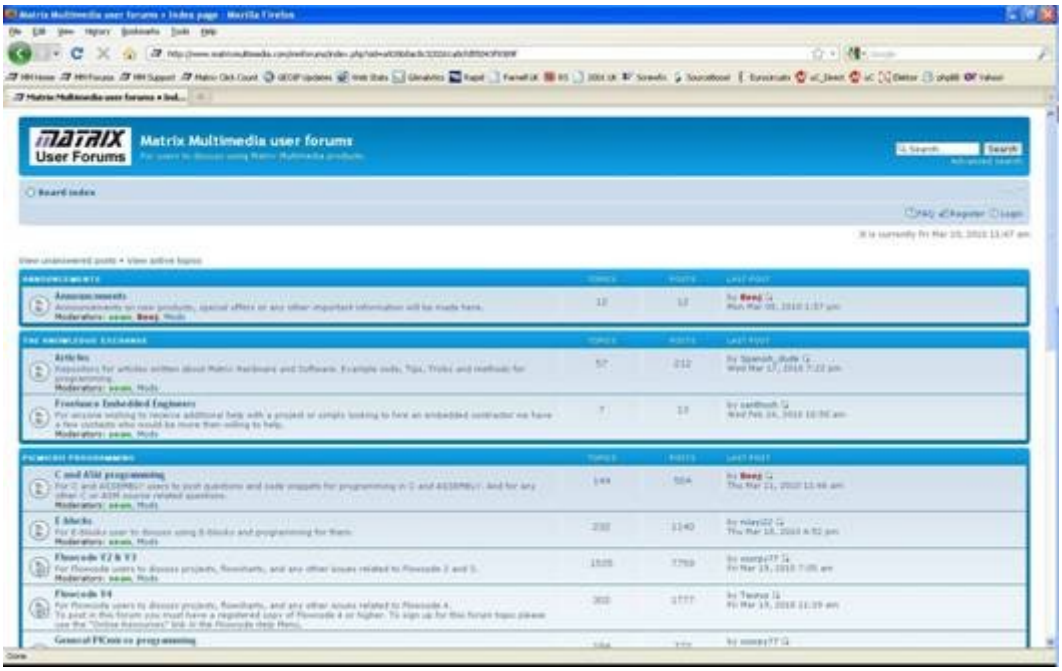
EB833-80-1
**E-Blocks Data-logger Bundle**

If you are having any problems getting up and running with any of the examples or any of the Flowcode components then the first port of call is to ensure you have your boards plugged together and wired correctly. As a rule of thumb any board with a screw terminal and a +V marking should be connected via a single core wire to the +V screw terminal on the corresponding Multiprogrammer.

If you do run into any problems then there is help and advice available from our online user forums located here:

http://www.matrixtsl.com/mmforums/

The Articles section contains quite a few examples, as well as hints and tips to aid in your applications




There is also an online video demonstrating Flowcode, available from the videos section of our website: http://www.matrixtsl.com/lc_videos.php

# Other Products

EB833-80-1

## E-Blocks Data-logger Bundle

Matrix TSL is a leading global technology company. Over the years we have developed a portfolio of award-winning products which have applications in Education, Industry and in the home.

Learning is at the heart of much of what our company does, and the philosophy of all Matrix learning products is based on 'learning by doing'. Each year Matrix spends around 25% of turnover on research and development to ensure that our learning and development resources are world class.

### MIAC PLC

MIAC (Matrix Industrial Automotive Controller) is an industrial grade control unit which can be used to control a wide range of different electronic systems including sensing, monitoring and automotive. It has a number of applications in industry and learning.

### Formula Flowcode

Formula Flowcode is a robot vehicle which is used to teach robotics, and to provide a platform for competing in robotics events.

### ECIO

ECIO devices are powerful USB programmable microcontrollers with either 28 or 40 pin standard DIL (0.6") footprints. They are perfect for student use at home, project work and building fully integrated embedded systems.

### FlowKit


The FlowKit allows for in circuit debugging directly from within Flowcode. This is the same ICD debugging feature that is included with our version 7 EB006 Multiprogrammer boards.

### Flowcode + E-Block Technology bundles

Matrix TSL technology bundles are based on a combination of two of our most popular products, E-Blocks and Flowcode.

FLOWCODE
NO CODING, NO LIMITS...

### Other bundles in the range

- Easy Mobile Communications Pack
- Easy Zigbee Pack
- Easy RFID
- Easy Internet
- Easy CAN Bus
- Easy GPS
- Easy USB
- Build your own PC Interface
- Build your own PLC
- Build your own Data-logger