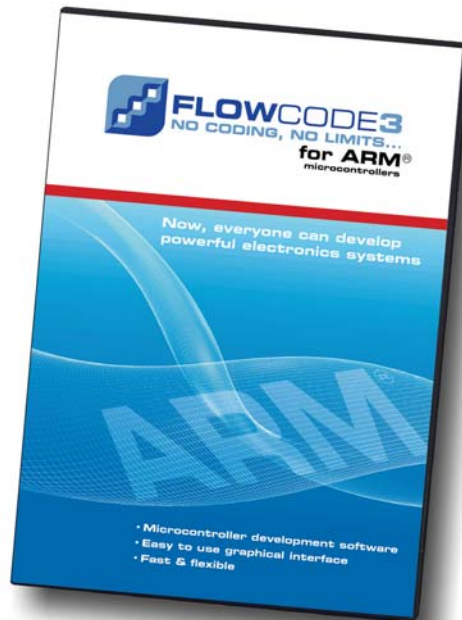


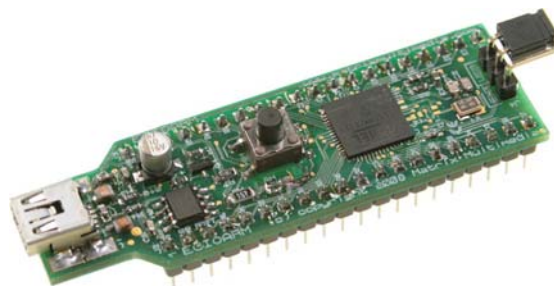
Moving up to 32 bit

How Flowcode and ECIO for ARM can help you move into
32 bit microcontroller programming

Introducing Flowcode for ARM....



...and the new ARM ECIO:



MATRIX

Introduction

Single chip microcontrollers have been around as electronic components for around 30 years now. Maybe a bit longer. In this time there have been several step changes in the technology that have had significant impact on the way engineers develop products. There is a long list of innovations here but notable ones in my mind are: the move from masked products (long lead times very expensive) to electrically programmable microcontrollers, the advent of parts that had A/D converters embedded in them, the development of low cost design tools that allowed even hobbyists to develop microcontroller circuits, and so on. What you may not realise is that we are entering a phase of one of the most significant changes in product development in the last 20 years – the advent of affordable 32 bit microcontroller technology. In a previous article we touched on the power of 32 bit ARM processors programmed with C – nasty. Here we look at how you can harness the power of 32 bit with an affordable and easy to use combination of ECIO ARM and Flowcode for ARM and we show you a range of new features that you can use to take advantage of these innovations.

The advantages of 32 bit

In moving to 32 bit architecture processors the advantages over 8 bit is not readily apparent – whilst I/O pin count for 32 bit families is larger, there are packages with similar I/O count to 8 bit microcontroller families. Whilst A/D speed for 32 bit processors is faster, it is not dramatically so. Whilst 32 bit cores tend to have more than one USART, this is not a ‘must-have’ feature either. Whilst 32 bit microcontroller families do offer memories of 128K ROM and 32 RAM, and more, this is not unheard of in 8 bit family ranges. But the general trend here is that 32 bit microcontroller families do generally offer ‘more’, of just about everything should you want it. This is the key point: 32 bit processors offer more I/O, more RAM/ROM, more internal features, and more speed. Not all these points will be of interest to all engineers because they are operating effectively with current 8 bit families, but let’s have a look at a few applications that may give you an idea as to why engineers who are pushing the barriers are starting to look at moving to 32 bit.

Sensors made easy

Most analogue sensors can be characterised by a mathematical formula. Take a temperature probe for example: a typical formula that converts a temperature sensor resistance to a temperature value would be given by the following equation:

$$T = [K_0 + K_1(\ln 1000R) + K_2(\ln 1000R)^3]^{-1} - 273.15$$

Where T is the temperature, R is the resistance in Kohms, K_0 is 1.02119×10^{-3} , K_1 is 2.22468×10^{-4} , and K_2 is 1.33342×10^{-7} .

If you were incorporating temperature data into a control program, then the normal way you would deal with this in code is to use Excel to generate a series of values for each of, say, 256 separate A/D readings on your microcontroller into the corresponding temperature value. You would then use this data, or a sub-set of it, as a basis for making decisions in your program.

If you had a program that needed to display the data in ‘human’ form you would either use further look up tables to allow you to display the decimal temperature reading, or you would implement floating point and a maths library in your 8 bit micro which would have quite a significant impact in terms of code size and speed.

In either case an ARM with floating point and maths functions built-in would allow you to simply enter the mathematical formula into a single line of code. For many applications this could mean the death of the look up table? Way hay! Thank goodness!!

Speed speed speed

The ARM chip operates off an external supply of 3.3V and an internal supply of 1.5V. This reduces the power consumption of the chip and also means that each gate in the device can be made smaller – which in turn increases the speed of operation: our ARM devices are clocked at 18MHz, but have an internal phase locked loop that boosts the clock frequency up to a whopping 47MHz. (Actually the ARM can go faster but we keep the speed at 47MHz because this is the frequency at which the USB connection works.) So how much faster, in practical terms, is it? Well funnily enough that

Part	ROM	RAM	I/O pins	Package	Speed	A/D	Price (US\$)
AT91SAM7XC128-AU	128K x 8	32K x 8	62	100-LQFP	55MHz	8x10bit	11.93
AT91SAM7X128-AU	128K x 8	32K x 8	62	100-LQFP	55MHz	8x10bit	10.97
AT91SAM7A3-AU	256K x 8	32K x 8	62	100-LQFP	60MHz	8x10bit	14.84
AT91SAM7SE256-AU	256K x 8	32K x 8	88	128-LQFP	48MHz	8x10bit	14.19
AT91SAM7SE256-CU	256K x 8	32K x 8	88	144-LFBGA	48MHz	8x10bit	15.7
AT91SAM7SE512-AU	512K x 8	32K x 8	88	128-LQFP	48MHz	8x10bit	16.77

Table I - One off ARM costs from Digikey, May 2008

is not a straight forward question: the answer is that it depends on just what the microcontroller is doing. For simple floating point mathematical operations you can expect the ARM to be between 5 to 10 times faster than 8 bit microcontrollers clocked at the same frequency. For more advanced mathematical operations – like Taylor’s series approximations – the ARM can be more than 100 times quicker. Our experiments here surprised us – we thought it would be universally much quicker for all operations – especially floating point. However we are relying on a compiler which may not be terribly well optimised for 32 bit operation. Still the ARM is a multiple of at least a few times quicker than any 8 bit microcontroller we have tried.

More memory

There is a mindset for engineers working with 8 bit processors: keep code size small, keep code efficient, minimise the software development overhead. All this means that developers have to minimise the available features in products. However a 32 bit fast core with almost-but-not-quite unlimited amounts of RAM and ROM, and more I/O should allow us to add radical features to a microcontroller and hence to our products. For example: let’s make our single chip microcontroller talk. There are a couple of routes open to designers who want talking chips: either digitise speech using a computer, or generate speech inside the microcontroller itself. Let’s look at each in turn:

Option 1: if you are digitising speech then for audio you should be able to get away with a bit rate of 8,000 samples per second at 8 bits. 100Kbytes of ROM can therefore store around 12.5 seconds of audio. That’s not so useful, but with 512K ROM micros readily available it is realistic.

Option 2: a better alternative might be to split up human speech into ‘phonemes’ which consist of all the basic sounds used in human speech, to store these in the ROM and then to selectively stream sequential phonemes to make as many different words as you like. The 65 phonemes that make up human speech take up around 65K of ROM space – not a lot for a 32 bit micro, but not practical in many 8 bit micros. The audio produced is definitely computer generated, but is also quite acceptable for use in electronic equipment.

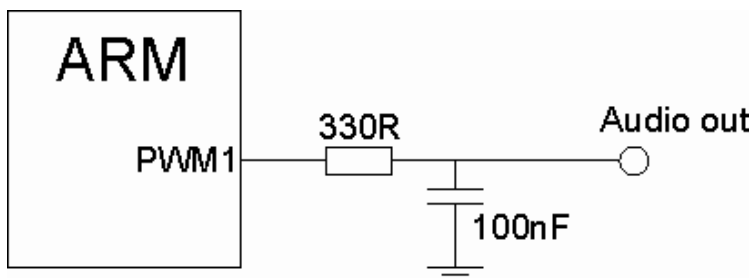


Figure 1 – single transistor D/A converter

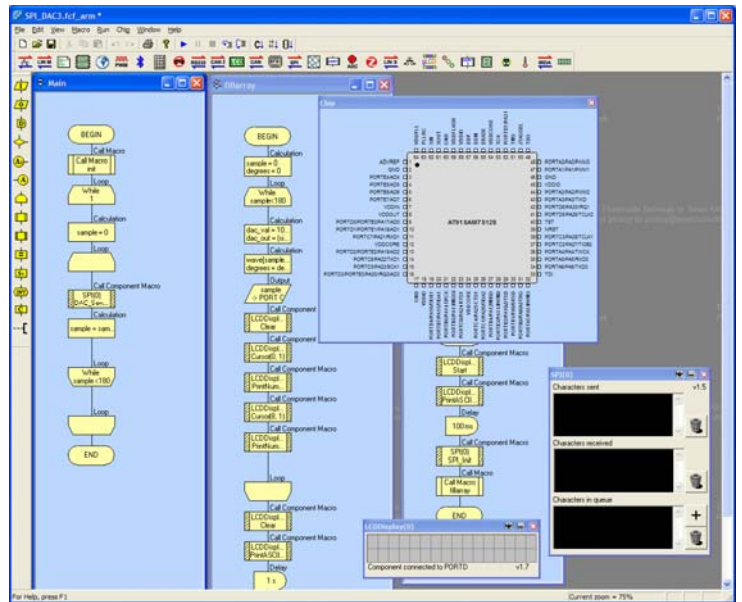


Figure 2 – Flowcode for ARM

In either case the speed of the 32 bit devices allows us to use a single transistor and a capacitor to make a D/A converter by modulating the internal PWM circuit of the ARM. You can see this in figure 1. This is the technique used for making audio on many mobile phones. So with 65k of internal memory you can now add sound to your microcontroller projects for just the cost of a few of passive components.

Disadvantages of the ARM

Of course there is a down side – or several:

- ARM devices – to us - are more difficult to program: I/O lines, data direction registers and the guts of these devices are addressed using a system of pointer registers which is just plain awkward. (or maybe it is just different!).
- The main supply voltage is 3.3V rather than the traditional 5V. That’s not so much of a problem when starting a development from scratch but it can have issues when upgrading existing systems. (however the plus side here is that the power consumption is low). The packaging is all SMD which has implications for the design cycle and – for small companies – also production implications.

Easy adoption - hardware

If some of the benefits we have highlighted here are of interest then how do you get involved?

We can suggest two routes:

Flowcode takes much of the pain away of using the AT91 ARM family of microcontrollers – it has been developed to allow users to easily port designs from 8 bit to 32 bit with a system of ports (A to E) that mirrors those found on smaller micros. This includes full software for handling floating point numbers and a full mathematics library, and it is compatible with programs from Flowcode for AVR and Flowcode for

PIC so porting your programs to this new platform should be easy.

We have also produced a 40 pin 0.6" version of the AT91 which fits into a standard 40 pin header – the ECRM40. You can see this in Figure 2. To help you get started we are giving this away free (it is worth £25) with every Professional version of Flowcode for ARM.

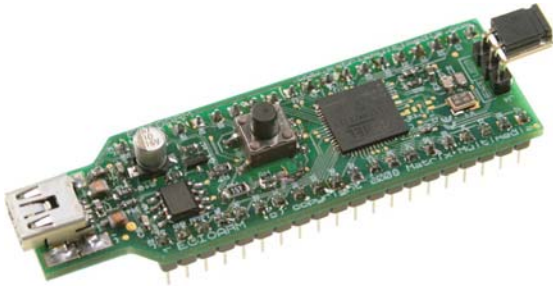


Figure 3 – ECRM40 ARM – free with Flowcode for ARM

If you would like a more robust development kit then there is also a suite of ARM based E-blocks that is compatible with Flowcode which is available at a reduction of 30%.

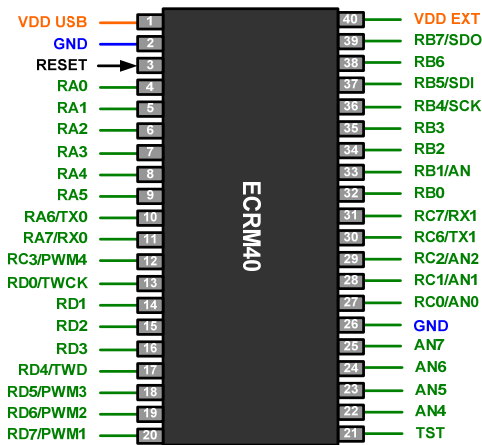


Figure 3b – ECRM40 pin out



Figure 4 – the ARM E-blocks bundle.

The following example files are available on request:


- Temperature-calculation.FCF_ARM
- Phoneme_based_speech.FCF_ARM
- WAV_file_speech.FCF_ARM


Introducing the Atmel SAM7 ARM



32 bit core AT91SAM7128 features

For our first 32 bit core we have used an Atmel AT91 ARM 7 device. We chose an ARM because we wanted a core that was supported by the GNU compiler and tool set, and because

the ARM is a British design. ( God save

our gracious queen, long live..... ). We also wanted one that was directly USB programmable and we decided to standardise on the Atmel AT91SAM7128 which has the following features:

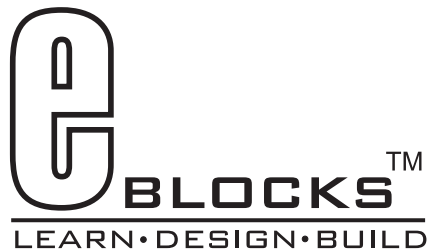
128K flash ROM
32K RAM
80MHz internal clock speed
2 USARTs
USB programming and communications interface
32 I/O lines
4 channel 16 bit PWM outputs
32 bit processor
8 x 300kHz 10 bit A/D converters



Compatible with.....



...and also...



Details correct at time of going to press. Matrix Multimedia reserves the right to change specification.

Matrix Multimedia Limited
The Factory
Emscote Street South
Halifax
HX1 3AN
England

+44 1422 252380

www.matrixmultimedia.com

Details correct :



ange specification.

TE-60-4