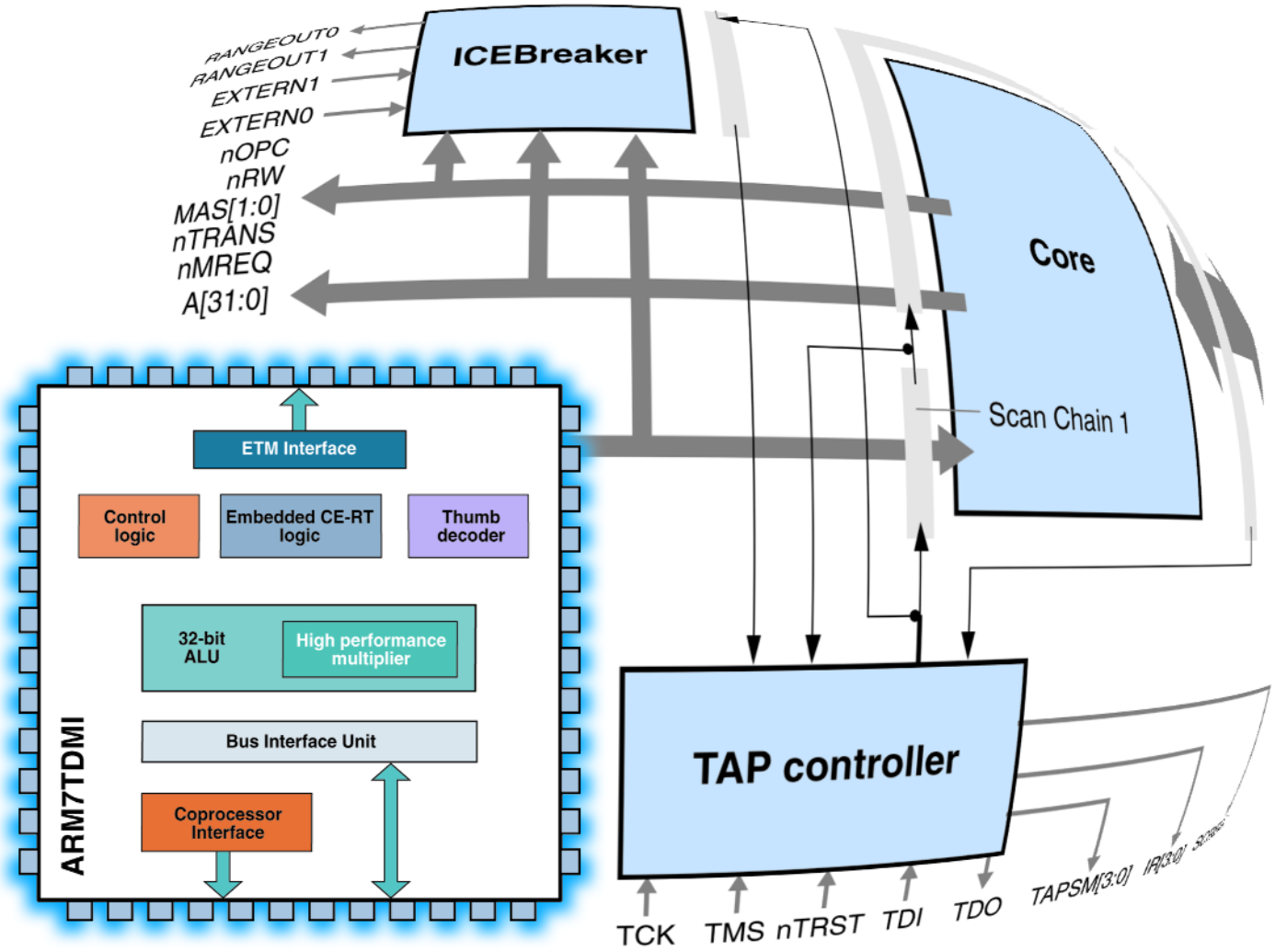
LPC210x 'ARMee'

Part 1: an ARM processor survey

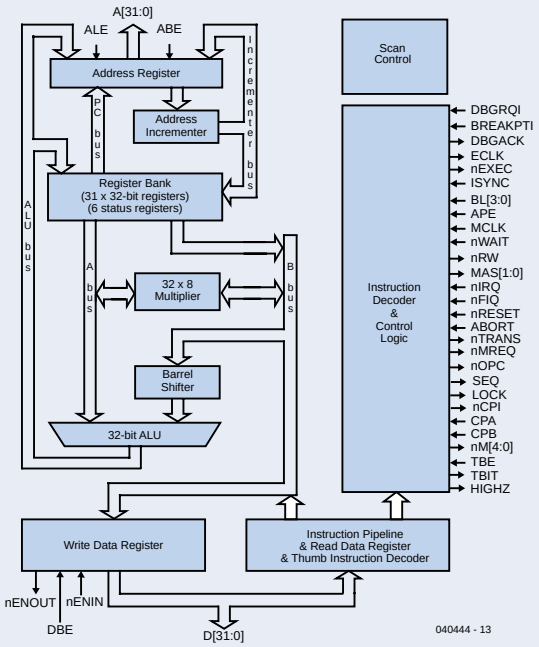
Tony Dixon



In the first instalment of a three-part article we will look at recent developments in 32-bit ARM based microcontrollers, concentrating on those devices that are available, do not cost an arm and a leg or require a major investment in surface mount soldering equipment!

DEVELOPMENT BOARD (1)

Overview of the ARM7TDMI core



The ARM7 core is a 32-bit Reduced Instruction Set Computer (RISC). It uses a single 32-bit bus for instruction and data. The length of the data can be 8, 16 or 32 bits and the length of the instruction word is 32 bits.

What Does TDMI™ mean?

The ARM7TDMI is an ARM7 core with 4 additional features identified with letter codes, as follows

T: support for the Thumb (16 bit) instruction set.
D: support for debug
M: support for long multiplies
I: include the EmbeddedICE module to support embedded system debugging.

Thumb mode (T)

An ARM instruction is 32-bits long. The ARM7TDMI processor supports a second instruction set that has been compressed into 16-bits, the Thumb instruction set. Faster execution from 16-bit memory and greater code density can usually be achieved by using the Thumb instruction set instead of the ARM instruction set, which makes the ARM7TDMI core particularly suitable for embedded applications.
However the Thumb mode has two limitations and these are: Thumb code usually uses more instructions for the same job, so ARM code is usually best for maximising the performance of the time-critical code.
The Thumb instruction set does not include some instructions that are needed for exception handling, so the ARM code needs to be used for exception handling. See ARM7TDMI User Guide for details on the core architecture, the programming model and both the ARM and ARM Thumb instruction sets.

Long multiple (M)

A 32-bit Multiplier function qualifies the core for complex arithmetic tasks usually performed by a DSP. The ARM7TDMI instruction set includes four extra instructions that perform 32-bit by 32-bit multiplication with 64-bit result and 32-bit by 32-bit multiplication-accumulation (MAC) with 64-bit result.

Debugging (D)

A special hardware extension allows for Debugging within an application. This is made possible by means of a boundary-scan cell array around the core driven by a JTAG port and a TAP controller.

EmbeddedICE (I)

The EmbeddedICE extends the debugging functions and this module contains the breakpoint and watch point registers that allow the code to be halted for debugging purposes. These registers are controlled through the JTAG test port with the aid of software debugging tools running on a computer. When a breakpoint or watch point is encountered the processor halts and enters debug state. Once in a debug state, the processor registers may be inspected as well as the Flash/EE, SRAM and the Memory Mapped Registers

ARM stands for *Advanced RISC Machine*, where RISC means *Reduced Instruction Set Computer*. The ARM 32-bit architecture has been around for a number of years and has been used in products where low power consumption is essential such as for mobile phones and PDA's.
Its 32-bit core is available in several guises including the ARM7, ARM9, ARM10 and the recently announced ARM11, each of which offers enhanced levels of computing power.

The ARM was usually only available as a microprocessor device, where it required external program and data memory to complete a system. However several companies are now offering a 32-bit ARM based microcontroller with sufficient memory options for them to be considered worthy microcontroller alternatives. **Table 1** shows a selection of ARM based microcontrollers from three companies, Analog Devices, OKI Semiconductors and Philips Semiconductors. Other companies such as Atmel, NetSilicon, Samsung and TI offer similar devices

Table 1. ARM processor comparison matrix.

Device	Package	RAM	Flash	Clock	I/Os	UARTs	SPI
AduC7020	40-pin LFCSP	8 kB	62 kB	0.3-45 MHz	14	1	1
AduC7021	40-pin LFCSP	8 kB	62 kB	0.3-45 MHz	13	1	1
AduC7022	40-pin LFCSP	8 kB	62 kB	0.3-45 MHz	13	1	1
AduC7024	64-pin LQFP	8 kB	62 kB	0.3-45 MHz	30	1	1
AduC7025	64-pin LQFP	8 kB	62 kB	0.3-45 MHz	30	1	1
AduC7026	80-pin LQFP	8 kB	62 kB	0.3-45 MHz	40	1	1
AduC7027	80-pin LQFP	8 kB	62 kB	0.3-45 MHz	40	1	1
LPC2104	48-pin TQFP	16 kB	128 kB	0-60 MHz	32	2	1
LPC2105	48-pin TQFP	32 kB	128 kB	0-60 MHz	32	2	1
LPC2106	48-pin TQFP	64 kB	128 kB	0-60 MHz	32	2	1
LPC2114	64-pin LQFP	16 kB	128 kB	0-60 MHz	46	2	2
LPC2124	64-pin LQFP	16 kB	256 kB	0-60 MHz	46	2	2
LPC2212	144-pin LQFP	16 kB	256 kB	0-60 MHz	112	2	2
LPC2214	144-pin LQFP	16 kB	256 kB	0-60 MHz	112	2	2
LPC2119	64-pin LQFP	16 kB	128 kB	0-60 MHz	46	2	2
LPC2129	64-pin LQFP	16 kB	256 kB	0-60 MHz	46	2	2
LPC2194	64-pin LQFP	16 kB	256 kB	0-60 MHz	46	2	2
LPC2292	144-pin LQFP	16 kB	256 kB	0-60 MHz	112	2	2
LPC2294	144-pin LQFP	16 kB	256 kB	0-60 MHz	112	2	2
ML674001	144-pin LQFP	32 kB	256 kB	1-60 MHz	42	2	1
ML674001	144-pin LQFP	32 kB	512 kB	1-60 MHz	42	2	1
ML675001A	144-pin LQFP	32 kB	256 kB	1-60 MHz	42	2	1
ML675001A	144-pin LQFP	32 kB	512 kB	1-60 MHz	42	2	1

but these are usually available in larger BGA (ball grid array) packaging only, making them less suitable for hand prototyping. So what's available on the ARM market we can actually obtain and handle? Let's have a look what a number of major manufacturers in the ARM arena have on offer.

Analog Devices

Analog Devices (www.analog.com) is not a name normally associated with microcontrollers, however they are about to change this view by creating a microcontroller product with a range of precision analogue interfaces. *Elektor Electronics* (leading the way!) have already published a series of articles on Analog Devices AduC812 8052 based controllers (Ref. 1). Analog Devices have released an updated range using an ARM7TDMI as the computing engine.
The ADuC702x family of devices from Analog Devices integrates a 32-bit ARM7TDMI core with a 12-bit data converter that can have up to 16 channels supporting one million samples/sec. The ADuC702X devices also feature up to four 12-bit DACs with a precision bandgap reference sensitive to 10 ppm/°C. Other peripherals include a comparator, a small programmable-logic array (PLA) for glue logic, an on-chip temperature sensor (±3°C) and a three-phase 16-bit PWM generator. Of these peripherals the programmable-logic array is the most interesting to find on a microcontroller.
A JTAG interface is provided for debugging the chip, while the UART can be used to program the Flash memory *in-situ*.

An on-chip oscillator will drive the ADuC702x at speeds of up to 35 MHz and is 2% accurate. An external clock is required to run at speeds up to the 45- MHz limit. Memory options include both a 32-kByte Flash memory for the ADuC7024 and a 62-kByte Flash memory for the ADuC7026 and all include 8 kBytes of RAM. Packaging options range from a 6×6-mm, 40-lead CSP, a 64-pin LQFP and an 80-pin LQFP. These 3-V devices can operate within a temperature range of –40 to +85, or at extended temperatures up to +105, or +125°C.
Analog Devices offer a low-cost quick start development system called QuickStart, which includes a power supply, cables, evaluation board, JTAG emulator and software-development tools from Keil Software and IAR Systems. The QuickStart Development System sells for $249 and is available directly from Analog Devices.
Of the devices offered, the 64-pin and 80-pin devices

I2C	CAN	Timers	PWM	ADC	DAC	Notes
2	-	2	-	5 x 12-bit	4 x 12-bit	PLA, Temp Sensor
2	-	2	-	8 x 12-bit	2 x 12-bit	PLA, Temp Sensor
2	-	2	-	10 x 12-bit	-	PLA, Temp Sensor
2	-	2	3	10 x 12-bit	2 x 12-bit	PLA, Temp Sensor, 3-Phase
1	-	2	3	12 x 12-bit	-	PLA, Temp Sensor, 3-Phase
1	-	2	3	12 x 12-bit	4 x 12-bit	PLA, Temp Sensor, 3-Phase
1	-	2	3	16 x 12-bit	-	PLA, Temp Sensor, 3-Phase
1	-	4 x 16-bit	6-ch	-	-	
1	-	4 x 16-bit	6-ch	-	-	
1	-	4 x 16-bit	6-ch	-	-	
1	-	4 x 16-bit	6-ch	4 x 10-bit	-	
1	-	4 x 16-bit	6-ch	4 x 10-bit	-	
1	-	4 x 16-bit	6-ch	8 x 10-bit	-	with external memory interface
1	-	4 x 16-bit	6-ch	8 x 10-bit	-	with external memory interface
1	2	4 x 16-bit	6-ch	4 x 10-bit	-	
1	2	4 x 16-bit	6-ch	4 x 10-bit	-	
1	4	4 x 16-bit	6-ch	4 x 10-bit	-	
1	2	4 x 16-bit	6-ch	8 x 10-bit	-	with external memory interface
1	4	4 x 16-bit	6-ch	8 x 10-bit	-	with external memory interface
1	-	7 x 16-bit	2-ch	4 x 10-bit	-	with external memory interface
1	-	7 x 16-bit	2-ch	4 x 10-bit	-	with external memory interface
1	-	7 x 16-bit	2-ch	4 x 10-bit	-	external memory i/f, 8K cache
1	-	7 x 16-bit	2-ch	4 x 10-bit	-	external memory i/f, 8K cache

are probably the most usable by our readership for their prototyping ease.

Philips

Of all the companies offering ARM microcontrollers Philips (www.semiconductors.philips.com) seem to be the company pushing the ARM microcontroller the most and have already released an extensive range of microcontrollers based on a 32-bit ARM7TDMI-S core (see **inset**). Philips initially offered the LPC210x which featured 16 to 64 kB of RAM dependant on the device, together with 128 kB of Flash memory and all operating at 60 MHz. Other peripherals include two UARTs, SPI and I²C interfaces, 6-channel PWM and 32-bit digital I/O port. All of which are fitted in a small 48-pin LQFP package!
All three controller chips are based on a common system architecture approach which offers the same memory map, vectored interrupt controller and similar peripheral complements. Also common to them are the same Flash programming and updating mechanism, JTAG debugging and emulation facilities.
These devices operate from 1.8 V for the core CPU functions and 3.3 V for the I/O and peripherals, with the general I/O being 5 V tolerant.

Philips has extended the LPC21xx family to include new devices packaged either in a 64-pin or a 144-pin LQFP. These new family members offer larger Flash memory options, an additional SPI interface and additional digital I/O lines. They also included either a 4 or 8 channel ADC with 10-bit resolution, 2- or 4-channel CAN bus interface and the option of an external memory interface on the larger 144-pin devices.
The LPC210x devices have a number of development and evaluation boards from companies such as Hitek, Keil, IAR and Nohau.
According to press releases from Philips we can expect future members of the LPC21xx and LPC22xx family to include Ethernet, USB, and 802.11 capabilities. Something to look forward to!

OKI Semiconductors

OKI Semiconductors (www.oki.com) are a Japanese company who offer a broad range of ICs and have been providing 32-bit ARM-based solutions for a number of years. Oki have extended its microcontroller portfolio by introducing a new series of general-purpose 32-bit microcontrollers based on an ARM7TDMI core.
These new two lines consist of the ML674001 and the

ML675001 series. The ML674001 series comprises of three products: the ML674001, the ML67Q4002 and the ML67Q4003. While the ML675001 series consisting of the ML675001, the ML67Q5002 and the ML67Q5003. The ML674001 and ML675001 are ROMless parts.

The ML67Q4002/3 and ML67Q5002/3 microcontrollers offer large Flash memory options up to 512 kB and 32 kB of RAM. Other peripherals include 1 × system timer, 6 × general purpose timers, 2 × PWM, watch dog timer, general purpose I/O ports, ADC converters and 2 × DMA channels. Communications are provided 2 × UARTs; one UART is an industry standard 16550A and has 16 bytes FIFO for both send and receive, with the other having no FIFO; an I^2C and SPI interface. The chips also include an external memory interface that features a SDRAM controller allowing for ROMs (including Flash memories), SRAMs, DRAMs, or I/O devices can be directly connected to the on-board SDRAM controller. A standard JTAG interface is provided for debugging and device programming. These chips can also be programmed by using a special Boot mode program built into the device. In boot mode, the on-chip boot ROM downloads a Flash writing application into the internal RAM area of the MCU. This application then handles the serial transfer and writing of internal Flash through the UART interface of the MCU.

The chips require 2.5 V for the core CPU functions and 3.3 V for the I/O and peripherals. The series operate in a wide temperature range of –40°C to +85°C. The ML674001 series can operate at a maximum frequency of 33 MHz, while the ML675001 series operates at a maximum frequency of 60 MHz. The ML675001 series has an 8-kB unified cache memory allowing the chip to operate at the higher clock speed.

The ML67Q4002/3 and ML67Q5002/3 are packaged in a 144-pin LQFP and all the microcontrollers are of a pin-compatible design, allowing for easier upgrade from the ML674001 series to the ML675001 series with a minimum of program and board layout change.

Next month's issue

has a heavy focus on microcontrollers and it is no coincidence that you will be able to read about an extremely powerful ARM microcontroller development system you can build at home, in class or in the lab. As far as we know, this is a first in electronics magazine publishing but then again who else but Elektor?

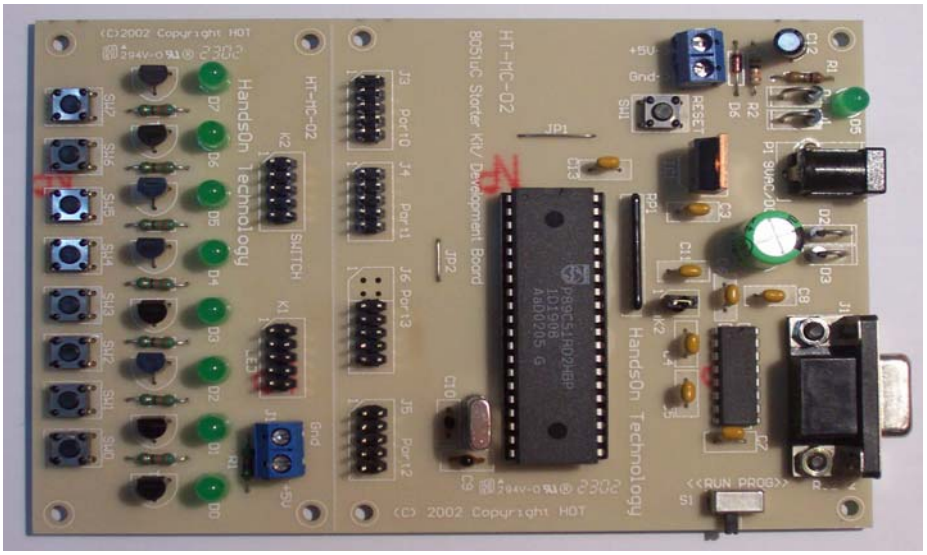
(040444-1)

Reference:

1. Intelligent Sensor/Actuator Controller (ISAC), parts 1-4, Elektor Electronics October – December 2001 and January 2002.

Low Cost 8051μC Starter Kit/ Development Board *HT-MC-02*

HT-MC-02 *is an ideal platform for small to medium scale embedded systems development and quick 8051 embedded design prototyping.* ***HT-MC-02*** *can be used as stand-alone 8051μC Flash programmer or as a development, prototyping and educational platform*



Main Features:

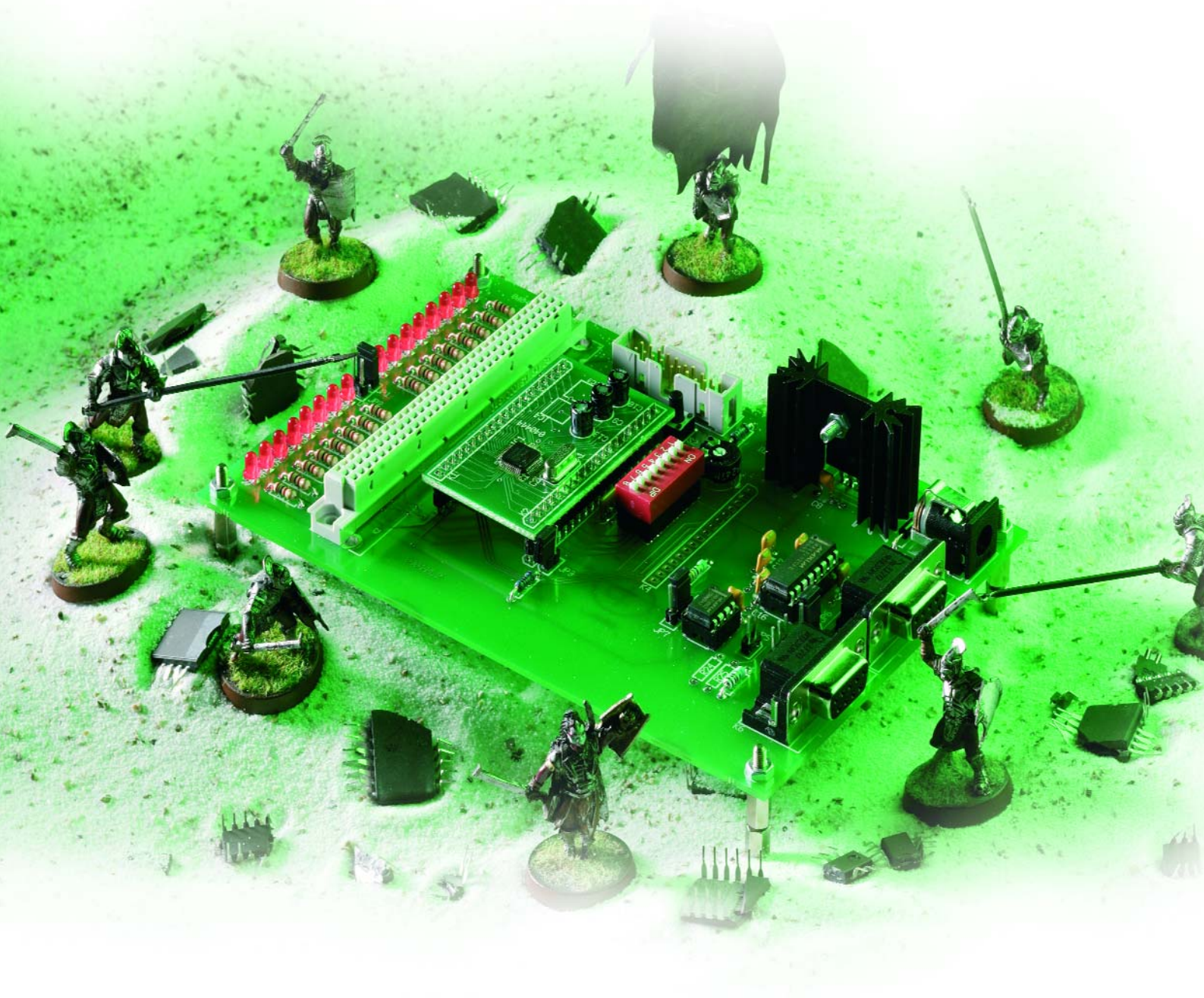
- 8051 Central Processing Unit.
- On-chip Flash Program Memory with In-System Programming (ISP) and In Application Programming (IAP) capability.
- Boot ROM contains low level Flash programming routines for downloading code via the RS232.
- Flash memory reliably stores program code even after 10,000 erase and program cycles.
- 10-year minimum data retention.
- Programmable security for the code in the Flash. The security feature protects against software piracy and prevents the contents of the Flash from being read.
- 4 level priority interrupt & 7 interrupt sources.
- 32 general purpose I/O pins connected to 10pins header connectors for easy I/O pins access.
- Full-duplex enhanced UART **–** Framing error detection Automatic address recognition.
- Programmable Counter Array (PCA) & Pulse Width Modulation (PWM).
- Three 16-bits timer/event counters.
- AC/DC (9~12V) power supply – easily available from wall socket power adapter.
- On board stabilized +5Vdc for other external interface circuit power supply.
- Included 8x LEDs and pushbuttons test board (free with ***HT-MC-02*** while stock last) for fast simple code testing.
- Industrial popular window ***Keil*** C compiler and assembler included (Eval. version).
- Free ***Flash Magic*** Windows software for easy program code down loading.

PLEASE READ ***HT-MC-02 GETTING STARTED MANUAL*** BEFORE OPERATE THIS BOARD
INSTALL ACROBAT READER (AcrobatReader705 Application) TO OPEN AND PRINT ALL DOCUMENTS

LPC210x 'ARMee'

Part 2: build and program the ARMee board

Tony Dixon



Development System

Bored with PICs, AVRs and 8051s everyone else is doing? Last month we covered 32-bit ARM microcontrollers and the undeniable star from all the devices discussed in that article was the Philips LPC210x. This month the real thing is upon us: enter **ARMee**, an incredibly powerful ARM development board you can build and program yourself.

Over the years *Elektor Electronics* have published a number of microcontroller development boards and this article is the latest in a long line of such articles. What's different about this article and this design is that instead of being based on an 8-bit device such as a PIC [1], AVR [2] or 8051 [3] controller, this design will use a **32-bit ARM microcontroller**: the LPC210x from Philips Semiconductors. This device has just about everything you could want from a microcontroller, see the '**ARMee & LPC210x Main Features**' inset. However, we should also mention that it lacks any ADC interface and an external bus interface. But then the LPC210x is a true microcontroller and has no external bus interface to extend the memory capabilities using external Flash or SRAM chips, or add new peripherals.

Hardware

The circuit diagram of the development system is shown in **Figure 1**. The motherboard may be powered from a standard DC mains adapter capable of supplying between 9 V and 15 V DC. An adapter capable of supplying around 500 mA will be more than sufficient for powering the development system board and any other reasonable amount of hardware connected to the development board.
On the processor daughterboard, voltage regulators provide the various voltages required by the LPC210x: 1.8 V for its CPU core and 3.3 V for its peripherals and I/O. The LPC210x micro has 5-V tolerant I/O and is capable of directly driving 5-V TTL logic.
Like many modern processors the LPC210x provides a reset circuit internally, so an external pull-up resistor is all that is required to finish the design. Pushbutton S2 is available to allow for manual resets to occur.

RS232 serial communication is provided at connector K5 and the interface is built around IC1, a MAX3232 RS232 Transceiver chip providing two driver and two receiver circuits. The MAX3232 is powered from the 3.3-V supply and an on-board charge pump provides the +12 V and –12 V needed by the RS232 interface. The board has the option of a second RS232 interface at connector K6 provided by the second RS232 driver and receiver circuits of IC1. Jumpers JP6 and JP7 should be fitted to the correct position to enable the second RS232 port.

If two RS232 interfaces are not required then connector K6 can provide a Modbus pin-compatible RS485 interface. The RS485 interface is driven by IC2, a MAX3082 RS485 Transceiver chip from Maxim, which is also powered from the 3.3-V supply. Jumpers JP6 and JP7 should be fitted to the correct position to enable the RS485 port instead of the second RS232 port.
As already stated, the RS485 interface is Modbus pin-compatible. However, Modbus uses pin 5 of the 9-way sub-D connector as a signal line while for RS232 pin 5 is connected to signal ground. To overcome this problem, jumper JP4 also needs to be fitted correctly to select either pin 5 as an RS485 signal or as RS232 signal ground.
Connector K4 provides the option of linking a standard alphanumeric LCD to the system board. The LCD interface shares some the same LPC210x I/O pins with the 8-way DIL switch. To use the LCD, every switch position of the DIL switch must be set to Off. A preset P1 is used to adjust the LCD contrast. The LCD R/$\overline{W}$ signal pin 5 of K4 is tied to 0 V thus preventing a program to read back any data from the LCD. Therefore a program should use a simple delay to allow the LCD sufficient time to process its last command. Digital I/O lines P0.22 and P0.23 control the LCD RS and E signals respectively. If the LCD module has an integrated backlight it may be powered through current limiting resistor R22.

An array of 16 LEDs (D1-D16) is provided on the board. The LEDs are connected to the first 16 digital input / output (I/O) lines of the LPC210x and can be used to indicate status or monitor peripheral activity such as activity on the UARTs. LED D1 is connected to

ARMee & LPC210x Main Features

LPC210x (2104/2105/2106):

- 16/32-bit ARM7TDMI-S processor core
- 128 kB program Flash memory
- 16/32/64 kB SRAM data memory
- ISP and IAP
- digital I/O; 2 UARTs
- SPI and I²C interfaces
- timer capture and PWM outputs
- JTAG debugging interface

ARMee Development System:

- direct lineage with earlier *Elektor Electronics* development systems
- detachable processor module available ready-stuffed with LPC2106
- LEDs for viewing port status
- 8-way DIL switch for switch inputs
- connector for linking to a standard alphanumeric LCD module
- real-world connectivity via two 9-way sub-D connectors: 2 x RS232 or 1 x RS232 and 1 x Modbus pin-compatible RS485
- DIN41612 I/O expansion connector on motherboard
- 160×100 mm Eurocard
- Free, multi-platform GNU GCC 'C' compilers
- Wide selection of free and commercial ARM development software

P0.00, LED D2 to P.01 and so on. It should be noted that the LEDs also share the LPC210x I/O pins with other functions such as the UART0 and UART1.
The LEDs are divided into two groups of 8. The first group, D1-D8 are enabled by jumper JP1 connecting them to 0 V and the second group D8-D16 are enabled by jumper JP2 again connecting them to 0 V.
An 8-way DIL switch, S1, shares the same LPC210x I/O pins as the LCD interface connector and should only be used if no LCD module is fitted to the system. You may use this switch for debugging purposes or use it as a configuration switch.

On the motherboard a DIN41612 A+C connector, K2, is available for the connection of external circuitry to the development system. This connector carries the voltage supply rails +5 V and +3.3 V, clock and reset signals and all 32 I/O signals from the LPC210x. The connector also carries signals for separate I²C and SPI interfaces. It should be noted that on the LPC210x an I/O pin may provide more than one function, in this case the I/O pin may be shared with the UART, I²C, SPI or other signals. It should also be noted that the on board LEDs, 8-way DIL switch and LCD connection also share I/O signals on the DIN41612 connector.

Printed circuit boards

The PCB artwork for the motherboard and detachable processor board is shown in **Figure 2**.
The processor daughterboard is available ready-built as item **040444-91** (with LPC2106 fitted), or bare as item **040444-1**; see Readers Services pages or go to our Online Shop.

The LPC210x is a 48-pin surface mount device (SMD) and will need to be soldered directly to the PCB. The bare PCB **040444-1** comes pre-tinned. Position IC1 carefully onto the SMD pads, then using a fine tipped soldering iron, solder each corner of the IC. If you are happy with the IC's position then continue and carefully solder the rest of the IC pins. Solder carefully, but don't worry if excess solder connects the

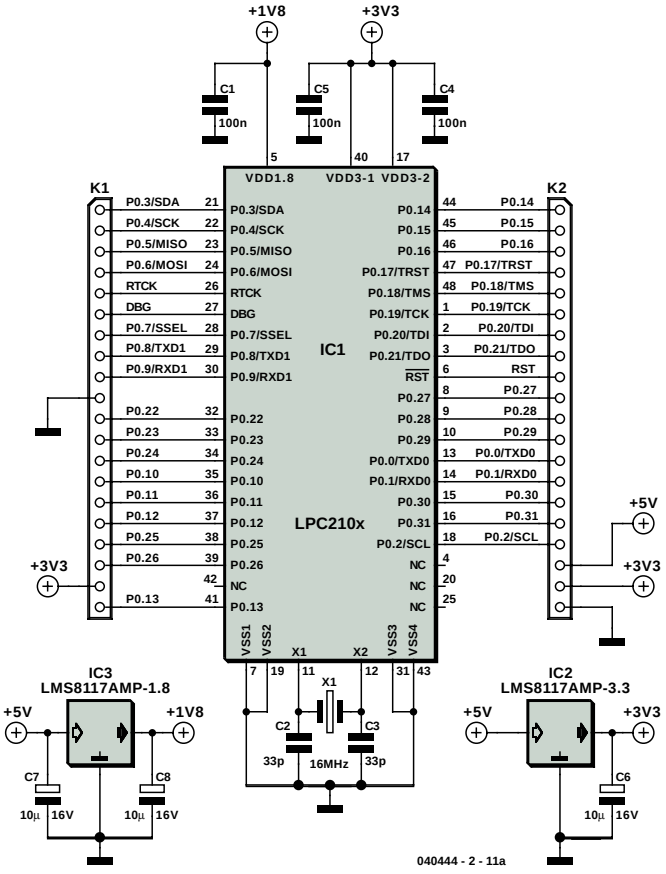


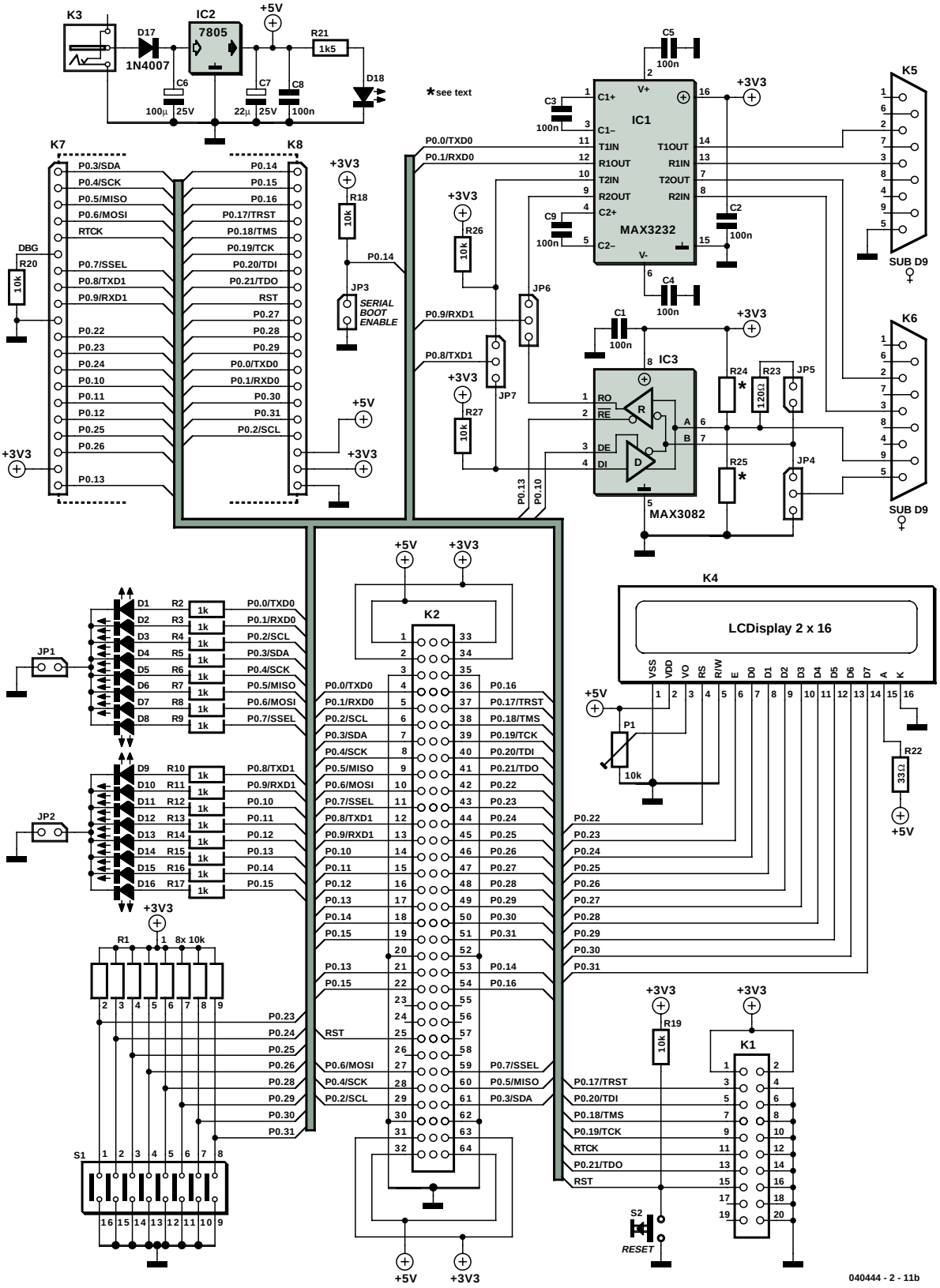
Figure 1.Circuit diagram of the ARMee Development System. Note that the processor module is detachable!

pins. Using some fresh desolder braid apply to each solder bridge and heat with a soldering iron for a moment, allowing the braid to absorb the solder from the bridge. Once the bridge is removed carefully resolder with the fine tipped soldering iron.

The bare motherboard PCB is available ready-made from our Readers Services and online Shop as item no. **040444-2**. When fitting components to the PCB, do observe the polarities of the diodes, the electrolytic capacitors and the LEDs. IC1 and IC3 on the motherboard can be fitted into sockets. With all components are fitted do one last visual inspection and compare the fruit of your efforts with our working prototype shown in **Figure 3** before applying power to the board.

Programming Tools

The LPC210x is easily programmed using the C programming language. A number of commercial C-Compilers are available for the ARM architecture (see



web links). Those users who do not want to purchase a commercial C-compiler can use the GNU C-Compiler. This is an open-source C-compiler that is considered by many as good as most of the commercial offerings. The downside of using the GNU compiler is that you don't have the polished installation process of a commercial compiler.
The LPC210x IC itself can be programmed using the on-board JTAG or the internal Serial Boot Loader.

Go JTAG!

The JTAG interface can be used for more than just programming the chip. It can be used to debug a program during execution on LPC210x. To use the JTAG interface, a JTAG debugging interface module is required to connect between the LPC210x development system and a host PC. There are several low cost commercial JTAG interface modules available, as well as several non-commercial interpretations. These can be found though a simple search on the web, or by checking out the web links provided at the end of this article.
If JTAG debugging is not required then

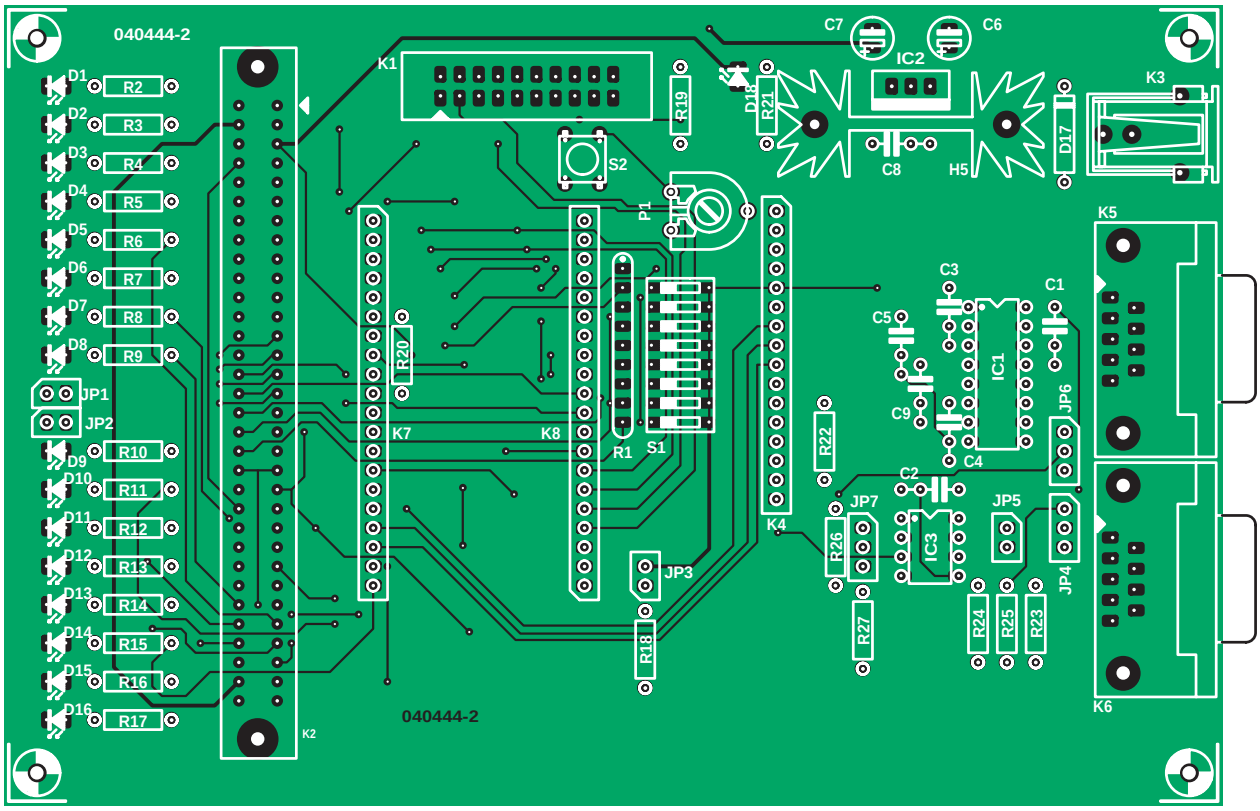


Figure 2a. PCB design for the Eurocard-size ARMee motherboard.

COMPONENTS LIST

ARMee motherboard

Resistors:
R1 = 8-way 10kΩ SIL array
R2-R17 = 1kΩ
R18,R19,R20,R26,R27 = 10kΩ
R21 = 1kΩ5
R22 = 33Ω
R23 = 120Ω
R24,R25 = see text
P1 = 10kΩ preset

Capacitors:
C1-C5,C8,C9 = 100nF
C6 = 100µF 25V, radial
C7 = 10µF 25V, radial

Semiconductors:
D1-D16,D18 = LED, low current, 3mm
D17 = 1N4007
IC1 = MAX3232CPE
IC2 = 7805
IC3 = MAX3082CP

Miscellaneous:
JP1,JP2,JP3,JP5 = 2-way pinheader with jumper
JP4,JP6,JP7 = 3-way pinheader with jumper
K1 = 20-way boxheader
K2 = 2x32-way DIN41612AC connector
K3 = mains adapter connector, PCB mount
K4 = general purpose LCD module, 2x16 characters
K5,K6 = 9-way sub-D socket (female), angled, PCB mount
K7,K8 = 20-way SIL socket
S1 = 8-way DIP switch
S2 = pushbutton, PCB mount, e.g., DTS6
Heatsink, Fischer SK104 25,4 STC
PCB, order code **040444-2**
Disk, project software (LEDTest), order code **040444-11** or Free Download

the LPC210x can be programmed using the built-in serial boot loader. This allows the 5 digital I/O lines used by the JTAG interface to be used for something else.

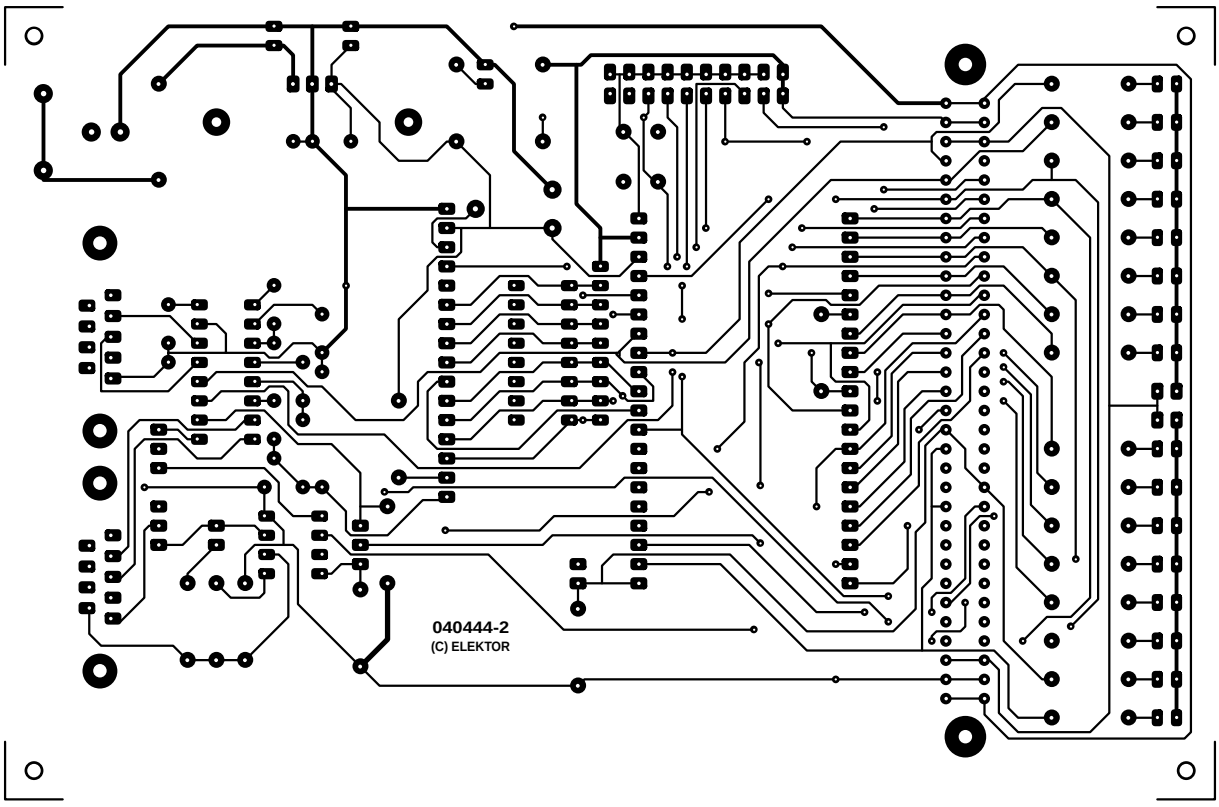
Serial Boot Loader

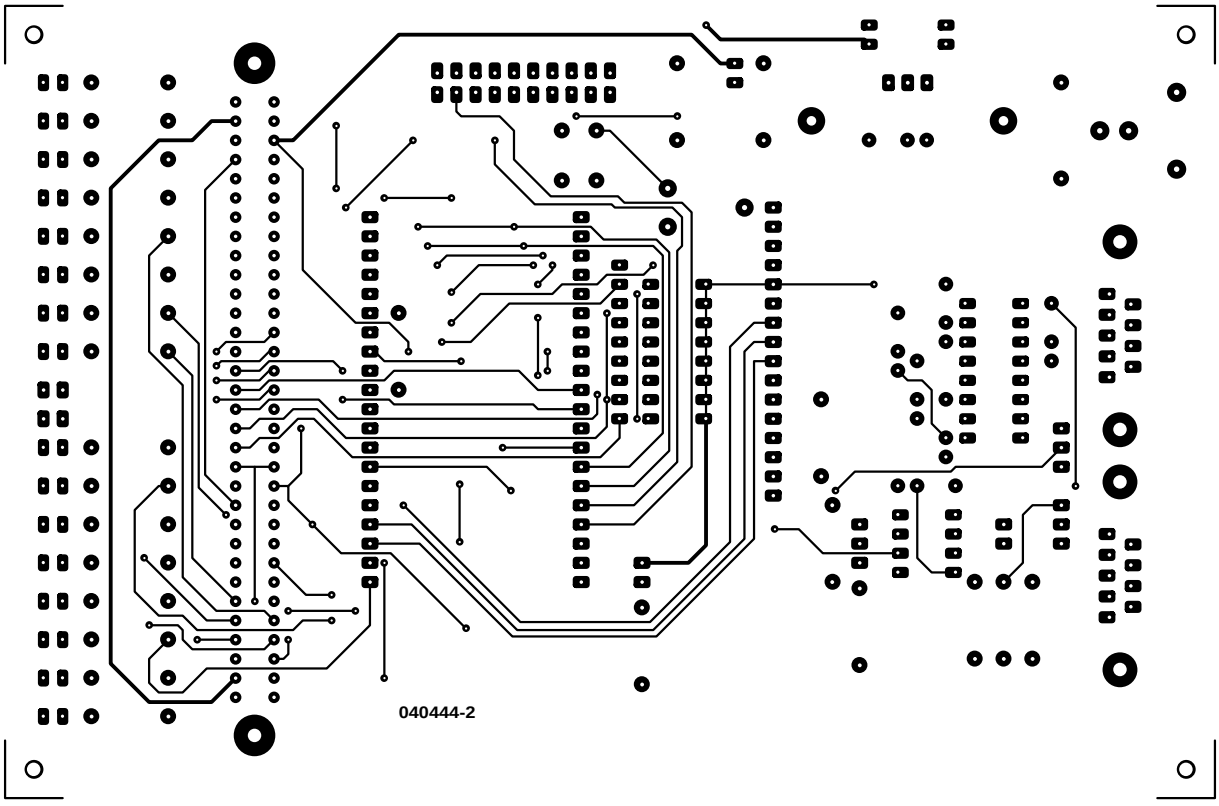
To use the serial boot loader jumper JP3 needs to be fitted. JP3 pulls P0.14 to ground and releasing the LPC210x from reset forces the LPC210x to execute an internal serial boot loader program from the top 8 k of its Flash memory.

Using a free PC program provided by Philips Semiconductor (**Figure 4**) and a serial cable connected between a host PC and UART0 of the LPC210x (RS232 Port 1 on the ARMee board) allows the programming of the LPC210x to take place.

Using GNU GCC with ARM Processors

"Why do we want to use the GNU GCC compiler?" you may ask? Well for a start the tools by virtue of their beginnings and the method of their commu-





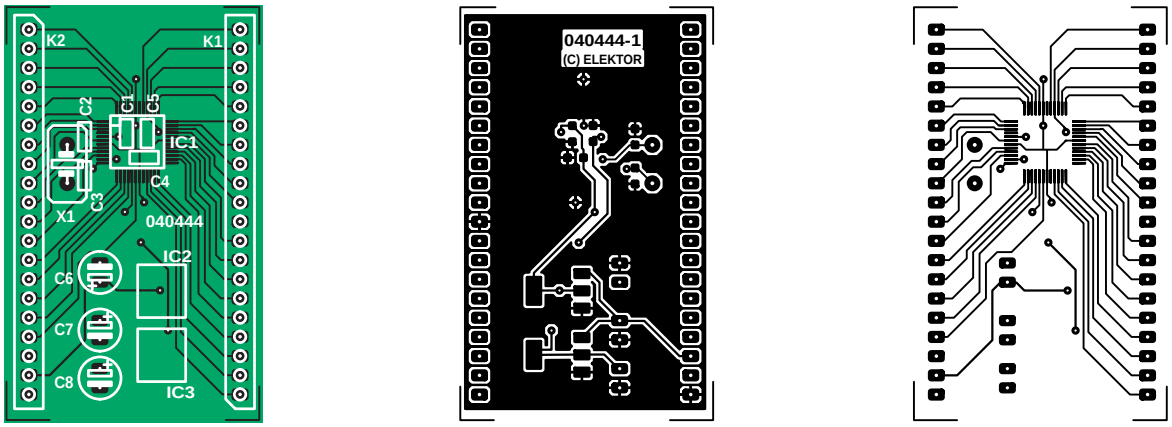


Figure 2b. PCB design for the detachable LPC210x processor board.

COMPONENTS LIST

LPC210x daughterboard

Capacitors:
C1,C4,C5 = 100nF, shape 0805
C2,C3 = 33pF, shape 0805
C6,C7,C8 = 10µF 16 V, radial

Semiconductors:
IC1 = LPC210x (2104/2105/2106)
IC2 = LMS8117AMP-3.3
IC3 = LMS8117AMP-1.8

Miscellaneous:
K1,K2 = 20-way SIL pinheader
X1 = 16MHz quartz crystal
PCB (bare), order code 040444-1
PCB (ready populated, with LPC2106), order code **040444-91**

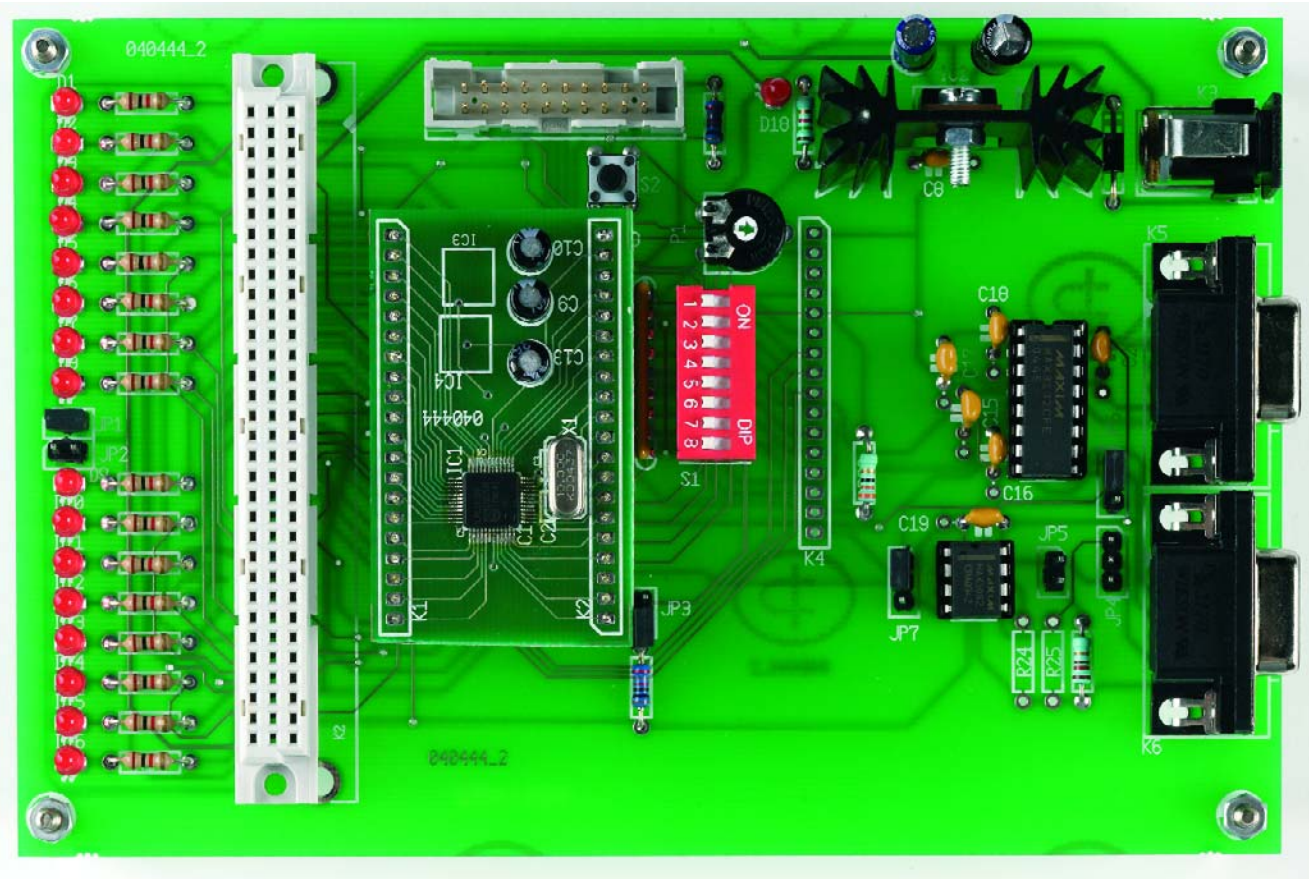


Figure 3. Our finished and tested prototype of the ARMee Development Board. The processor module can be migrated to the target application!

nal user support they are essentially provided for free at the Free Software Foundation (FSF) web site. These tools are made available through the GNU project under the GNU general public license. Historically, the software from GNU/FSF was aimed at the UNIX operating system and GNU is short for 'GNU's Not Unix', while GCC is short for 'GNU Compiler Collection'.

GNU GCC can be found of dozens of different host platforms from Linux, Microsoft Windows, UNIX and Solaris and many others. GNU GCC has been targeted to produce executables for many different microprocessor architectures ranging from Intel/AMD x86,

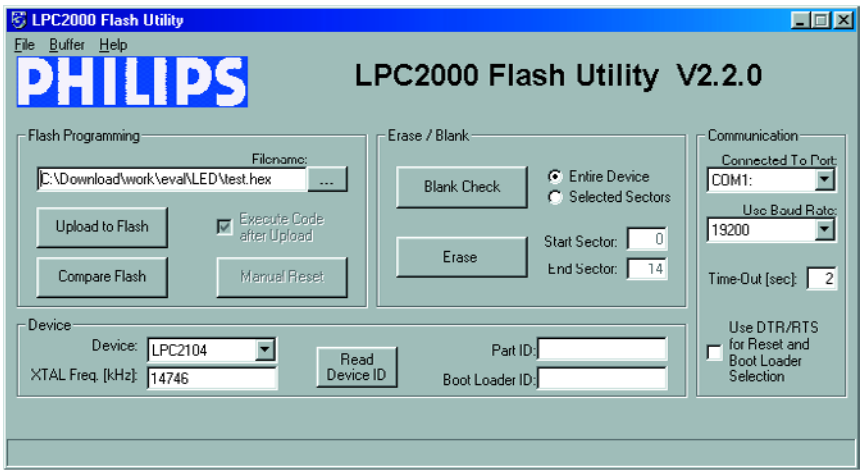


Figure 4. Flash Downloader from Philips.

Simple Input / Output Bus – SIOB

The Simple Input / Output Bus (SIOB) is about as simple as an expansion bus can be. 32 digital I/O signals are brought to a DIN41612 board connector and made available as an expansion port. The digital I/O lines may be either inputs or outputs dependant on the type of expansion interface required. The programmer of the development board is responsible for configuring the data direction register of the microcontroller according to their digital I/O requirement.

In addition to the digital I/O signals the SIOB also offers serial expansion in the form of a SPI and I²C interface. A small number of SPI chip select signals are also present on the bus. The presence of the SPI and I²C interfaces will allow the development system to be expanded in many ways. For example a system could be expanded with I²C based Analogue to Digital (ADC) and Digital to Analogue (DAC) converters or SPI driven digital I/O.

The bus is completed by the +5 V and +3.3 V power signals, clock signal, $\overline{\text{RESET}}$ and RESET.

One final note on the SIOB, the signal pins of the LPC210x are not only used for digital I/O but have other functions such as UART, I²C, SPI, PWM, Timer Capture, JTAG debugging etc. Where a signal pin is used to say drive a UART interface it will no longer be available as a digital I/O signal and thus no longer be available for the SIOB. If a system requires the use of the UART, JTAG debugging and LCD interfaces then there will be very few digital I/O signals available for the SIOB. In such cases system expansion is best done through the I²C or SPI interfaces available on the SIOB.

Motorola 68x00, PowerPC, ARM7 and ARM9. GNU GCC also covers several microcontroller architectures, some of these being TI's MSP430, Atmel's AVR and Motorola's 68HC11.
Another good reason is bugs, software bugs. The GNU tool chain gets updated regularly, meaning any reported problems are fixed relatively fast.
Your next question might be "What's the downside of using GNU GCC?" Well, for a start it's the staggering range of hosts and targets available. You have to check if GCC is available for your host platform, next check if your target processor is available and finally decide if you want to build (i.e., compile) the GCC yourself or want a pre-built binary. However, for the most part you will find the most common target architectures available for both Win32 and Linux hosts as both source code and binaries. But one word of warning, don't expect to find a fancy

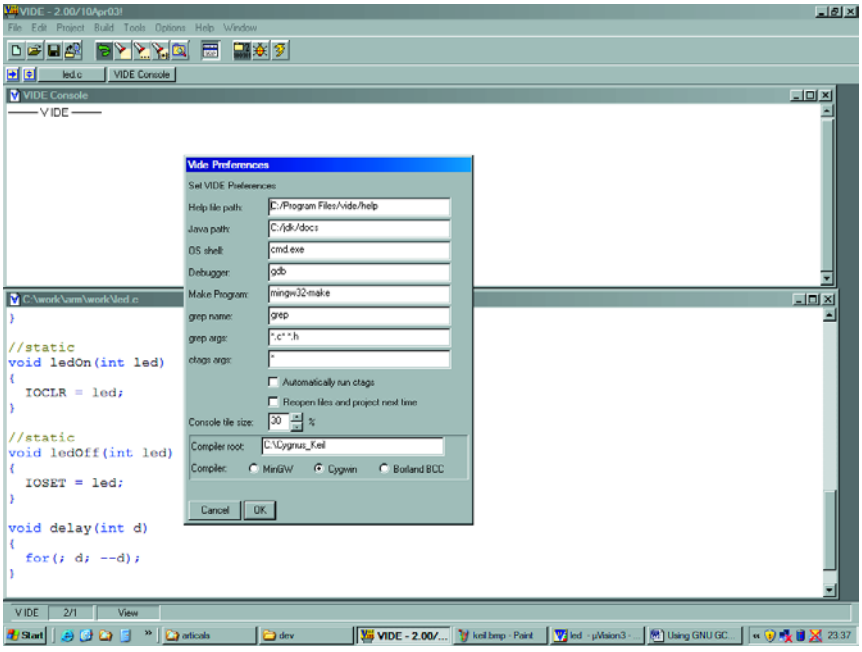


Figure 5. VIDE Programmer's Editor.

More ARM binutils.

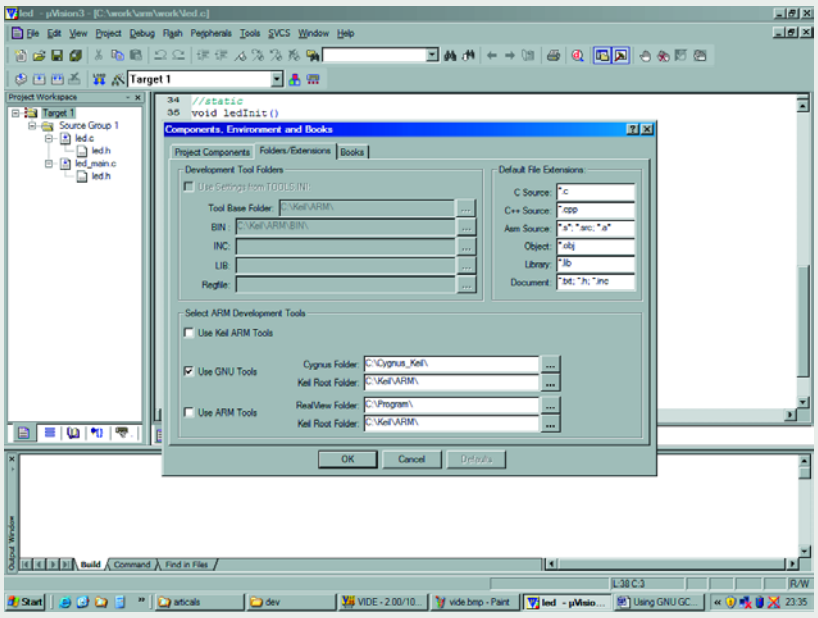
addr2line	Converts addresses into filenames and line numbers.
ar	Archiving utility (creating, modifying and extracting from archives)
gprof	Displays profiling information.
nlmconv	Converts object code into an NLM.
nm	Lists symbols from object files.
ranlib	Generates an index to the contents of an archive.
readelf	Displays information from any ELF format object file.
size	Lists the section sizes of an object or archive file.
strings	Lists printable strings from files.
strip	Discards symbols.

installation program or an Integrated Development Environment (IDE) — this is 'storm trooper' DIY development. If you want a fancy installer and IDE check out one of the companies that offer pre-configured, ready to go versions of the GNU tools. GNU-X tools from Microcross and GNUPro from Redhat are two examples of commercial GNU tools.
If you wanted to build your own binaries then download them from the FSF/GNU web site and follow the instructions provided on the site. However, we will take one of the many pre-built binaries and use it to compile our C code with.
For our example we will use the **GNU GCC version 3.4 Win32 pre-built binaries** from www.gnuarm.com. You can select the Linux host if you wish. Other pre-built binaries are available on the web, Google a search with [GNU], [ARM] and [binaries] and see how many hits you get. So let's download the binaries from GNUARM and install them on our PC.

Commercial offerings

If you are happy to provide a few personal details to Keil you can freely download their uVision IDE and a pre-built copy of GNU tools. These tools are unrestricted for generating code with the only limitation (at this time) is that the debugger is limited to 16 k program size. The Keil tools are based around their uVision 3 IDE, the **screendump** shows the IDE configured for GNU GCC.

Redhat and Microcross provide pre-built GNU tool chains for your chosen target and host platform. Rowley Associates also provide a commercial development environment based on the GNU tool chain but using their own libraries and IDE.



Configuring Keil uVision IDE for GNU GCC.

GNU C Compiler
Go to the GNUARM sub-directory you installed, look down the list of program files and you will see a program called *arm-elf-gcc.exe*. Similarly, if you were using a microprocessor based on the SH3 architecture you would expect to find *sh3-elfs-gcc.exe* in the sub-directory.
Now, *arm-elf-gcc* is a GNU GCC compiler targeted for the ARM architecture using the Extended Linker Format (ELF) object file format. If you were using the Common Object File Format (COFF) you would find *arm-coff-gcc.exe*. ELF and COFF are two different formats of file structure that the GCC can save the compiler object output to.
Our first task is to make sure the GCC compiler has been correctly installed. A quick test is to display the version message from the GCC compiler. Open a Command Window or MS-DOS window and type the command *arm-elf-gcc –v*. If you don't see the GCC version message, then you most likely need to add the GCC directory to our environment path.
To use the compiler, open a DOS window and type

```
arm-elf-gcc  -c ledtst.c
```

The actual program is found in the '**LEDTest – an example' inset**. An archive file containing the assembly code, linker and compiler result files is available as a **free download** with this article (**040444-11**).
The GCC compiler has a raft of options we can invoke, too many to list in this article! Some of the main command line options are:

–c	for compile only and do not link the object file;
-Wall	to enable every warning and error conditions the compiler will produce;
-o filename	to change the file name of the object file produced;
–S	to produce an assembly listing interspaced with the original C code;
–g	to include debugging information.

The command

```
arm-elf-gcc  -Wall ledtst.c –o ledtst
```

will faithfully compile *ledtst.c* with all compiler warnings enabled and pro-

duce an object file called *ledtst.o*.
When we invoke GCC compiler, it normally does the preprocessing, compilation, assembly and linking. However we can tell the compiler not to link any of the object files by using -c option. Where this is done we can use the GNU linker to combine our object files into a single output file.

GNU Linker
The object files produced during the compilation stage are by themselves incomplete. We still need to resolve internal variables and function references for the object files and this is where the linker comes in. The linker's job is to combine the object files and resolve any unresolved symbols and function references.
The command line GNU linker *ld* (or *arm-elf-ld*) takes the names of the object files to be linked together as arguments, as well as any options and outputs a single relocatable object file. If we had several files to compile and link we might type:

```
arm-elf-gcc  -c module1.c
arm-elf-gcc  -c module2.c
arm-elf-gcc  -c module3.c

arm-elf-ld module1.o module2.o module3.o  -o prog1
```

There are no memory addresses assigned to the code and data sections in the relocatable object file. To do this we need a locator program that will assign physical memory addresses to each code and data sections. The GNU linker has a built-in locator function that can locate the code and data sections of the program to specific areas of memory, which results in an absolutely located binary image.
We can now download this binary image to our development system.
The GNU linker includes a scripting language that can be used to control the linking process and the exact order of the code and data sections within the relocatable program. We can also assign memory address to the code and data sections. We call a script file by using the command line option –T followed by the name of the script file, for example:

```
arm-elf-ld -Tlpc2106-rom.ld
```

GNU Make
The make file is used to mechanise the compilation process. In essence, the make utility follows a user defined set of rules and these rules are processed in sequence. For instance, instead of typing the four command lines for the linker example shown previously we would type and save these commands into separate text file *makefile* or another suitable filename. To compile this example we would type:

```
make makefile
```

The Make utility will determine what source code files have changed since the last time is was run and will only compile those changed files. You may also include conditional compilation choices by making use of makes' preprocessor functions.

Debugging

The GNU GCC tool chain also comes with a command line debugging tool called GDB.
GNU GDB is one of those tools whose operation and large number of options could fill an entire book, let alone a single article. We do not intend to use GDB with the ARMee Development Board, instead you will be using the Flash downloader provided Philips (**Figure 4**). Therefore, for this article we skim over most of GDB functionality but those readers wanting more information on GNU GDB are referred to the web where lurks Bill Gatliff's excellent introduction.
In essence GDB is a source debugger. It is connected to a target board through a serial connection where it communicates and interacts with a monitor type program on the target board. To launch GDB type:

```
arm-elf-gdb ledtst
```

Once GDB is running you will be working within the GDB console where the debugging commands will be typed. Next, you can load your program to the target board by using the command

```
(gdb) load
```

Once loaded, you can step through your source code, set break points, investigate what functions were called to get to the current source line or use one of the other debugging options GDB offers.

Other tools

The GNU tool chain also includes a number of other tools, the *binutils*, which we can us during our development. Two of the most useful tools are objcopy and objdump.
The objcopy (or *arm-elf-objcopy*) utility is used to translate object files from one format to another format. For example, we can translate a file from binary image into s-record format.

```
arm-elf-objcopy -O srec ledtst.o ledtst.s19
```

while the objdump (or *arm-elf-objdump*) utility can be used to disassembly the object files. With the disassembled file you can see where the linker has located the various text, data and bss sections of our object file.

```
arm-elf-objdump.  —disassemble ledtst.o
```

You may also add source code to the

Start-up file

To use our compiled C program we first need to initialise the processors stack pointers and memory settings. A separate object file called the start up file usually does this. Typically a start up file will be called startup.s, boot.s or crt0.s. The start up file is linked to the rest of the compiled object files during the program linking stage. Check the example files from the LPC2000 group on Yahoo for crt0.s or boot.s, which are typical examples of ARM start up files.

The start up code must provide the following:

1. Disable all interrupts.
2. Copy initialised variables from ROM to RAM.
3. Zero uninitialised data variables.
4. Allocate space for and initialise stack area.
5. Initialise the stack.
6. Create and initialise the heap area.
7. Enable interrupts.
8. Jump to main ().

disassembled output by adding the option *–S* to the command line. If you want to view the symbol table of our object file, simply include the *—syms* option to the command line.
Some of the GNU other *binutils* are listed in **Table 1**.

IDE

When installed, GNU doesn't provide us with any form of integrated development environment (IDE). To resolve this we could install VIDE (**Figure 5**), one of the more popular IDEs that can be used with the GNU tool chain.
VIDE provides us with a programmers text editor and a GUI for running the GNU tools. Other IDEs, both free and commercial, are available for us to use with our GNU tools. Do a search of the web to see what's out there, or use the Forum on our website to get in touch with fellow ARM twisters.

(040444-2)

References:
1. PICee Development System, Elektor Electronics February 2003.
2. AVRee Development System, Elektor Electronics March 2003.
3. 89S8252 Flash Microcontroller Board, Elektor Electronics December 2001.

For further reading
1. Programming Embedded Systems in C and C++, Michael Ball; O'Reilly (www.oreilly.com)
2. An Introduction to GCC, Brian J. Gough, foreword by Richard M. Stallman; pdf copy at http://www.network-theory.co.uk/gcc/intro/

Websites:
www.geocities.com/tonydixon2k1/index.html (ARMee support website)
www.gnu.org (Main GNU web site)
www.fsf.org (Free Software Foundation web site)
www.gnuarm.com (GNU Binaries for ARM processors)
http://groups.yahoo.com/group/lpc2000/ (Philips LPC2000 ARM Discussion Group)
http://groups.yahoo.com/group/gnuarm/ (GNUARM Discussion Group)
www.objectcentral.com/vide.htm (VIDE Windows IDE)
www.billgatliff.com (Excellent GNU tutorials by Bill Gatliff)
http://www.dreamislife.com/arm/ (LPC2106 tutorials by Senz)
www.keil.com (GNU GCC and Commercial C-Compilers)
www.rowley.co.uk (GNU GCC and Commercial C-Compilers)
www.redhat.co.uk (Commercial GNU GCC tool chains)
www.microcross.co.uk (Commercial GNU

LEDTest - an example using the ARMee board

In the embedded development community the first program you usually write with a new compiler or target platform is a LED-flashing program. This is as ubiquitous as the "hello world" program is for PC development and we will be no different. Our example program will consist of a C program that will flash one the LEDs on the ARMee development board. See Listing 1!
Include the Philips ARM LPC2100 C header file, ARM start up file (boot.o) and GCC linker script (lpc2106-rom.ld) in the same directory as our LED test program. From the command line prompt, type the following GCC commands:

arm-elf-gcc -c ledtst.c

arm-elf-ld -Tlpc2106-rom.ld -nostartfiles, -nostdlib -s -o led boot.o ledtst.o

arm-elf-objcopy —output-target ihex test test.hex

We should now have a compiled and linked hex file ready for sending to the ARMee Development board. To save retyping the GCC commands, we could type the above GCC commands into a text editor and save them as a batch file ledtst.bat or create a Makefile.

Finally, instead of using the GDB debugging tool included with GNU GCC we will op to use the simple Flash download provided by Philips Semiconductor, Figure 4. Select the filename of our LED hex file and upload to the target board.

Once uploaded, press reset on the ARMee board and LED D1 should start to flash.

Congratulations! You've have successfully compiled, linked and uploaded our test program using the GNU GCC compiler and tools.

```
#include "LPC2000.h"


void Delay (unsigned long a)
{
    while (—a!=0);
}


int main (void) {

  /* System Init */
  /* Init PLL */
  SCB_PLLCFG = 0x23;
  SCB_PLLFEED = 0xAA; SCB_PLLFEED = 0x55;

  SCB_PLLCON = 0x01;              /* Enable PLL */
  SCB_PLLFEED = 0xAA; SCB_PLLFEED = 0x55;

  while ( !(SCB_PLLSTAT & PLOCK) ); /* Wait for PLL to lock */

  SCB_PLLCON = 0x03;              /* Connect PLL as clock source */
  SCB_PLLFEED = 0xAA; SCB_PLLFEED = 0x55;

  /* Init MAM & Flash memory fetch */
  MAM_MAMCR=0x2;                   /* MAM = flash */
  MAM_MAMTIM=0x4;
  SCB_VPBDIV=0x1;                  /* PCLK = CCLK */

  /* Init GPIO */
  GPIO_IODIR = 0x000000001;      /* P0.0 as output */
  GPIO_IOSET |= 0x00000001;      /* LED off */


  /* main loop */
  while (1) {

    GPIO_IOSET |= 0x00000001;    /* LED off */
    Delay (1000000);

    GPIO_IOCLR |= 0x00000001;    /* LED on */
    Delay (2000000);

  }

} /* End of main () */
```

HandsOn Technology

http://www.handsontec.com

creativity for tomorrow's better living...

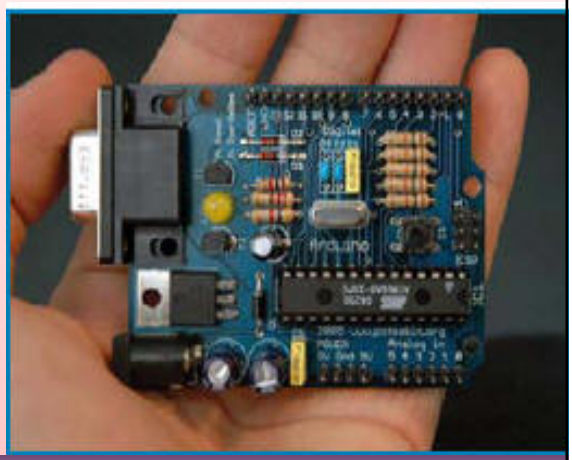
HandsOn Technology is a manufacturer of high quality educational and professional electronics kits and modules, uController development/evaluation boards. Inside you will find Electronic Kits and fully assembled and tested Modules for all skill levels. Please check back with us regularly as we will be adding many new kits and products to the site in the near future.

Do you want to stay up to date with electronics and computer technology? Always looking for useful hints, tips and interesting offers?

Inspiration and goals...

HandsOn Technology **provides a multimedia and interactive platform for everyone interested in electronics. From beginner to diehard, from student to lecturer... Information, education, inspiration and entertainment. Analog and digital; practical and theoretical; software and hardware...**

HandsOn Technology provides Designs, ideas and solutions for today's engineers and electronics hobbyists.



Creativity for tomorrow's better living...

HandsOn Technology believes everyone should have the tools, hardware, and resources to play with cool electronic gadgetry. HandsOn Technology's goal is to get our "hands On" current technology and information and pass it on to you! We set out to make finding the parts and information you need easier, more intuitive, and affordable so you can create your awesome projects. By getting technology in your hands, we think everyone is better off

We here at HandsOn like to think that we exist in the same group as our customers >> curious students, engineers, prototypers, and hobbyists who love to create and share. We are snowboarders and rock-climbers, painters and musicians, engineers and writers - but we all have one thing in common...we love electronics! We want to use electronics to make art projects, gadgets, and robots. We live, eat, and breathe this stuff!!

If you have more questions, go ahead and poke around the website, or send an email to sales@handsontec.com. And as always, feel free to let your geek shine - around here, we encourage it...