LPC-H2294 Board

Introduction

The **LPC2294** are based on a 16/32 bit ARM7TDMI-S™ CPU with real-time emulation and embedded trace support, together with 128/256 kilobytes (kB) of embedded high speed flash memory. A 128-bit wide memory interface and a unique accelerator architecture enable 32-bit code execution at maximum clock rate. For critical code size applications, the alternative 16-bit Thumb Mode reduces code by more than 30% with minimal performance penalty.
With their 144 pin package, low power consumption, various 32-bit timers, 8-channel 10-bit ADC, PWM channels and up to 9 external interrupt pins these microcontrollers are particularly suitable for industrial control, medical systems, access control and point-of-sale. Number of available GPIOs ranges from 76 (with external memory) through 112 pins (single-chip). With a wide range of serial communications interfaces, they are also very well suited for communication gateways, protocol converters and embedded soft modems as well as many other general-purpose applications.

The **LPC-H2294** Development board is designed to evaluate LPC2124 processor. It has the following features:

- 1MB (256Kx32bit) 12 ns 71V416 SRAM
- 1MB (512Kx16bit) 55ns MX26LV800BTC FLASH
- standard JTAG connector with ARM 2x10 pin layout for programming/debugging with ARM-JTAG
- USB to RS232 convertor, board can take power only from USB port
- two on board voltage regulators 1.8V and 3.3V with up to 800mA current
- optional single power supply: +5VDC required if not connected to USB port
- power supply and status LED
- power supply filtering capacitor
- Slide switch for ICSP-RUN mode
- Slide switch for Boot mode from external internal flash
- RESET circuit with external control of Philips ISP utility via USB-RS232 virtual port
- RESET button
- DBG jumper for JTAG enable
- BSL jumper for bootloader enable
- JRST jumper for enable/disable external RESET control by RS232
- removable crystal on socket (default 14.7456 Mhz crystal installed)
- extension headers for all uC ports with IDC34 connectors (easy obtainable as used for HDD and Floppy on all PCs)
- dimensions: 86x55 mm (3.4x2.2")
- PCB: FR-4, 1.5 mm (0,062"), green soldermask, white silkscreen component print

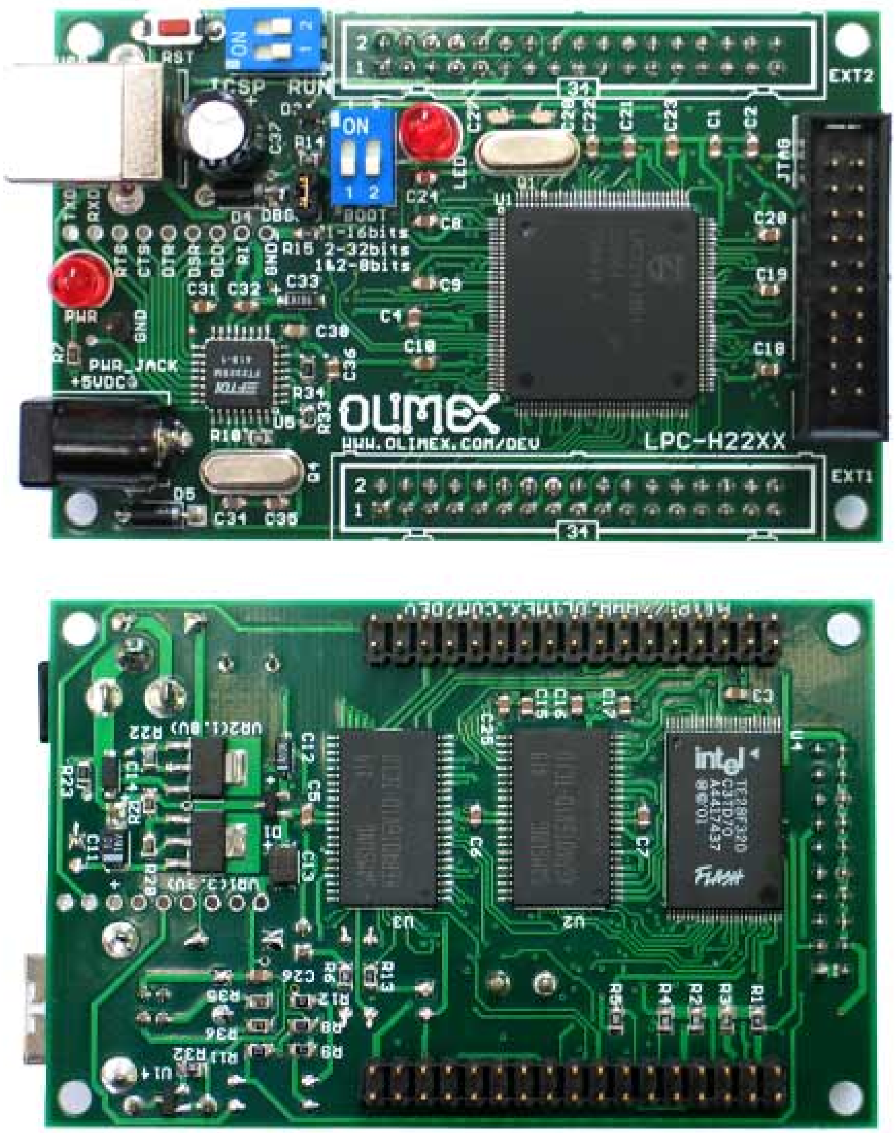
The purpose of this guide is to describe LPC-H2214 Development board.

Board	LPC-H2294
Hardware details	Describes the hardware peripherials in detail
Programming	describes how to write programs for the H2294 Board.

Revision 01.01.2005 Creating

Picture

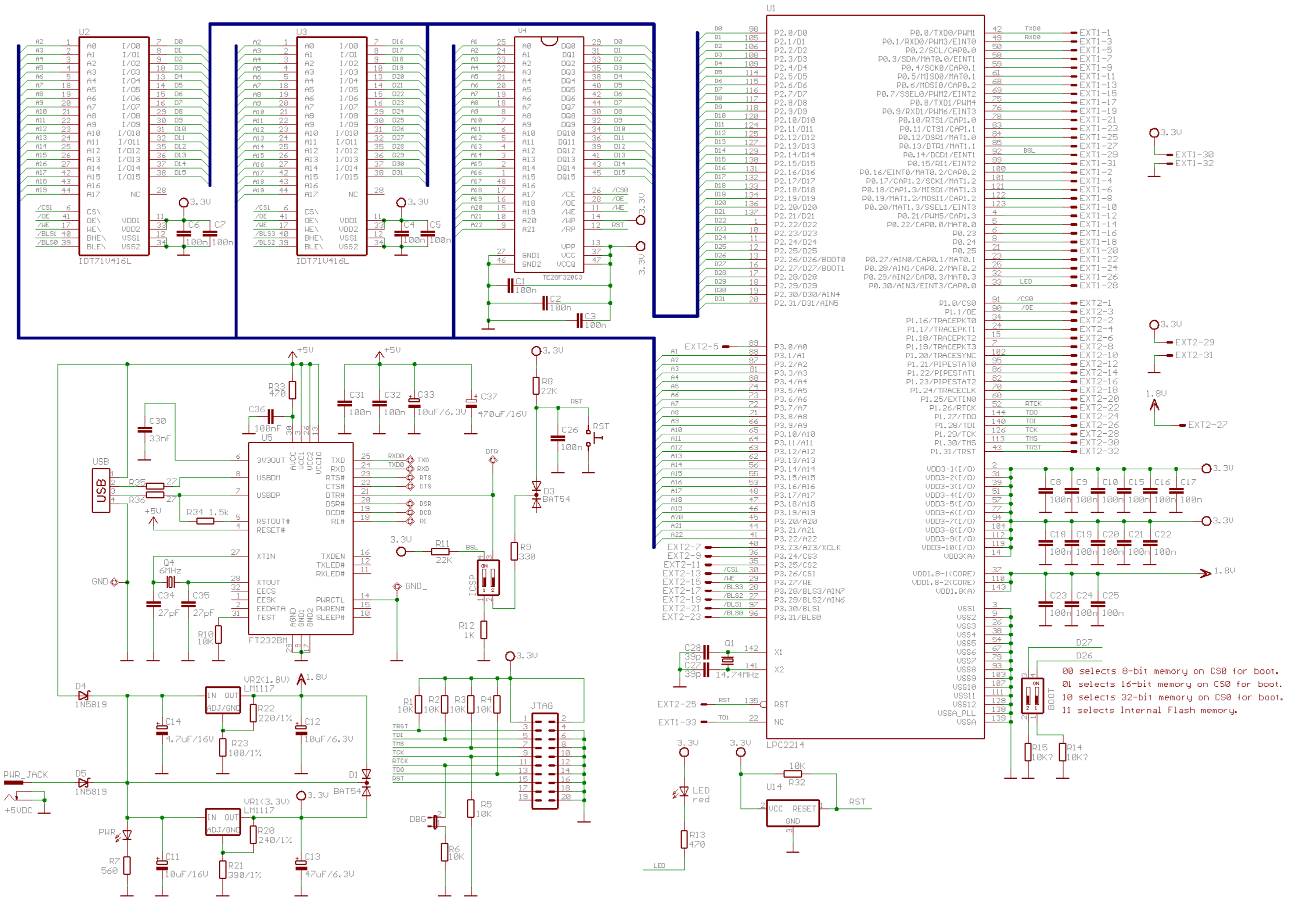
This is picture of LPC-H2294 Development board.



Board LPC-H2294

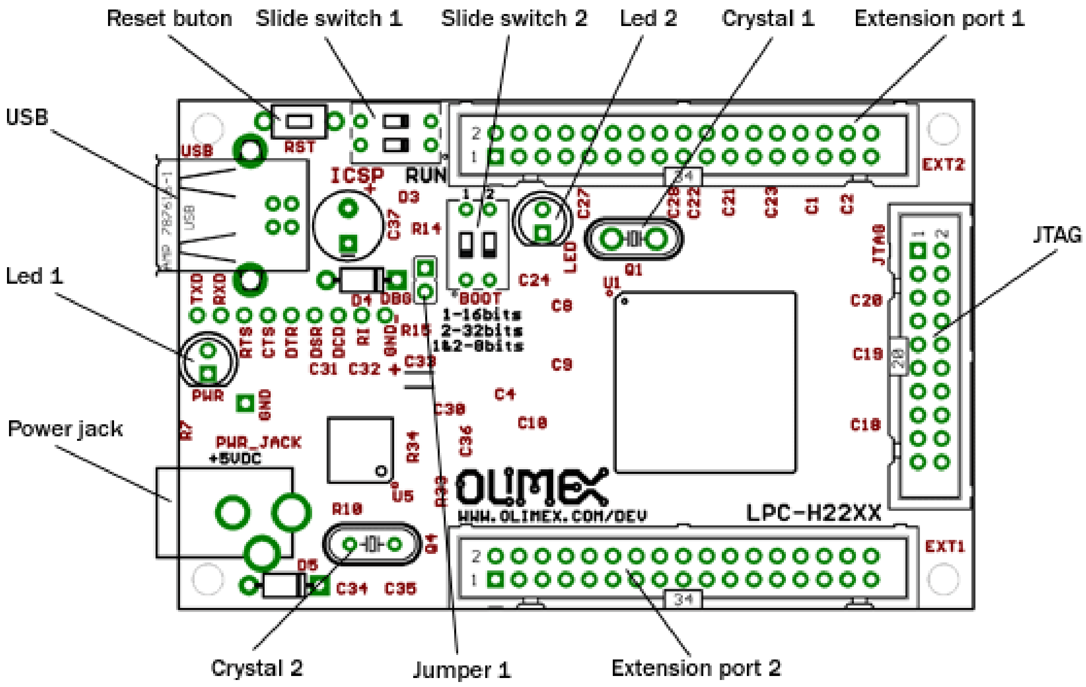
LPC-H2294 Board

H2294



00 selects 8-bit memory on CS0 for boot.
01 selects 16-bit memory on CS0 for boot.
10 selects 32-bit memory on CS0 for boot.
11 selects Internal Flash memory.

H2294 Board


Reset buton
Slide switch 1
Slide switch 2
Led 2
Crystal 1
Extension port 1
USB
JTAG
Led 1
Power jack
Crystal 2
Jumper 1
Extension port 2
RST
ICSP
RUN
BOOT
1-16bits
2-32bits
1&2-8bits
PWR_JACK
+5VDC
EXT1
EXT2
JTAG
WWW.OLIMEX.COM/DEV
LPC-H22XX

LPC H2294 Hardware description

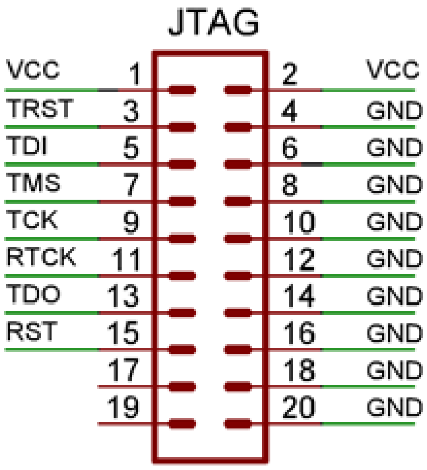
Peripherials

Unit	Description
USB Connector	USB Connector Type B
SRAM memory	1MB (256Kx32bit) 12 ns 71V416 SRAM
Flash memory	1MB (512Kx16bit) 55ns MX26LV800BTC FLASH
JTAG Connector	2x10 0,1" step connector for programming with ARM-JTAG.
Extension ports	headers for all uC ports with IDC34 connectors (easy obtainable as used for HDD and Floppy on all PCs)
Leds	Status led connected to P0.30 / AIN3 / EINT3 / CAP0.0 (PIN 33) and power led indicator.

Technical characteristics

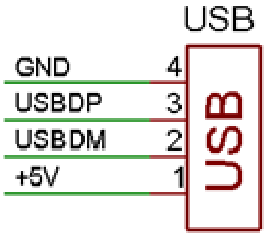
Parameter	Description
Voltage Supply	5.0V DC (power supply from USB) min 5.0V DC max 7.5V DC (power supply from Power jack)
CPU	LPC2214
Crystals	Crystal 1 - Q1 - 14,745 MHz crystal Crystal 2 - Q2 - 6 MHz crystal
Board dimensions	86x55 mm (3.4x2.2")
PCB	FR-4, 1.5 mm (0,062"), green soldermask, white silkscreen component print
Operating Temperature	form 0єC to 70єC

JTAG Connector



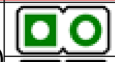
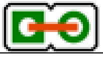
Pin / Name	Connected to:	Functionality
1 - VCC	VCC	-
2 - VCC	VCC	-
3 - TRST	PIN 43	P1.31 / TRST
4 - GND	GROUND	-
5 - TDI	PIN 140	P1.28 / TDI
6 - GND	GROUND	-
7 - TMS	PIN 113	P1.30 / TMS
8 - GND	GROUND	-
9 - TCK	PIN 126	P1.29 / TCK
10 - GND	GROUND	-
11 - RTCK	PIN 52	P1.26 / RTCK
12 - GND	GROUND	-
13 - TDO	PIN 144	P1.27 / TDO
14 - GND	GROUND	-
15 - RST	PIN 135	RST
16 - GND	GROUND	-
17 - -	no connected	-
18 - GND	GROUND	-
19 - -	no connected	-
20 - GND	GROUND	-

USB Connector

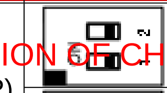
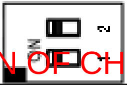
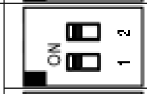
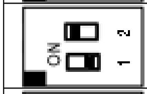
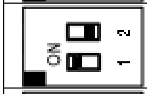
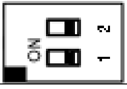


Pin / Name	Connected to:	Functionality
1 - +5V	+5V DC	-
2 - USBDM	FT232BM (PIN 8)	USBDM
3 - USBDP	FT232BM (PIN 7)	USBDP
4 - GND	GROUND	-

Jumpers

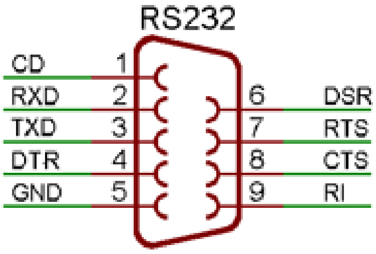
Jumpers	Position	Description
Jumper 1 (DBG)		Disable JTAG programming.
		Enable JTAG programming.

Slide switch

Slide switch	Position	Description
Slide switch 1 (ICSP)		Disable ICSP programming.
		Enable ICSP programming.
Slide switch 2 (BOOT)		Select 8-bit memory on CS0 for boot.
		Select 16-bit memory on CS0 for boot.
		Select 32-bit memory on CS0 for boot.
		Select internal Flash memory.

Programming: RS232

RS232 Connector



Pin / Name	Description
1 - CD	**Carrier Detected.**
2 - RXD	**Received Data.**
3 - TXD	**Transmited Data.**
4 - DTR	**Data Terminal Ready.**
5 - GND	**Signal Ground.**
6 - DSR	**Data Set Ready.**
7 - RTS	**Request to Send.**
8 - CTS	**Clear to Send..**
9 - RI	**Ring Indicator.**

Register description

Register	Address	Function
U0RBR	0xE000C000 DLAB = 0	**Receiver Buffer Register.** Input data buffer.
U0THR	0xE000C000 DLAB = 0	**Transmit Holding Register.** Output data buffer.
U0DLL	0xE000C000 DLAB = 1	**Divisor Latch LSB.**
U0DLM	0xE000C000 DLAB = 1	**Divisor Latch MSB.**
U0IER	0xE000C004 DLAB = 0	**Interrupt Enable Register.**
U0IIR	0xE000C008	**Interrupt ID Register.**
U0FCR	0xE000C008	**FIFO Control Register.**
U0LCR	0xE000C00C	**Line Control Register.**
U0LSR	0xE000C014	**Line Status Register.**
U0SCR	0xE000C01C	**Scratch Pad Register.**
U0TER	0xE000C030	**Transmit Enable.**

1.Initialization

1.1. Set Line Control Register

U0LCR	Function	Description	Rese Valu
1:0	Word Length Select	00: 5 bit character length 01: 6 bit character length 10: 7 bit character length 11: 8 bit character length	0
2	Stop Bit Select	0: 1 stop bit 1: 2 stop bits (1.5 if U0LCR[1:0]=00)	0
3	Parity Enable	0: Disable parity generation and checking 1: Enable parity generation and checking	0
5:4	Parity Select	00: Odd parity 01: Even parity 10: Forced “1” stick parity 11: Forced “0” stick parity	0
6	Break Control	0: Disable break transmission 1: Enable break transmission. Output pin UART0 TxD is forced to logic 0 when U0LCR6 is active high.	0
7	Divisor Latch Access Bit	0: Disable access to Divisor Latches 1: Enable access to Divisor Latches	0

1.2. UART0 Baudrate Calculation

The U0DLL and U0DLM registers together form a 16 bit divisor where U0DLL contains the lower 8 bits of the divi and U0DLM contains the higher 8 bits of the divisor.

```
devisor = pclk / (16 * baud);
```

1.3. Set Functionality to pins

Set functionality to P0.0 -> TX0 and P0.1 -> RXD0

2. RS232 Communication

2.1. Write to RS232

Use follow algorithm to send data:
- fill U0THR register with data to write
- wait shift all data
- clear interrupt flag

2 .2. Read from RS232

Use follow algorithm to receive data:
- wait read all data
- clear interrupt flag
- get data from U0RBR

3. Example

Initialize:

```
 //set Line Control Register (8 bit, 1 stop bit, no parity, enable DLAB)
U0LCR_bit.WLS = 0x3;     //8 bit
U0LCR_bit.SBS = 0x0;     //1 stop bit
```

```
   U0LCR_bit.PE = 0x0;      //no parity
   U0LCR_bit.DLAB = 0x1;    //enable DLAB

   //devisor
   U0DLL = Pclk / (16 * baud);      //low bite
   U0DLM = Pclk / (16 * baud)>>8;   //high bite
   U0LCR &= ~0x80;

   //set functionality to pins: port0.0 -> TX0, port0.1 -> RXD0
   PINSEL0_bit.P0_0 = 0x1;
   PINSEL0_bit.P0_1 = 0x1;
```

Read Data:

```
   //when U0LSR_bit.DR is 1 - U0RBR contains valid data
   while (U0LSR_bit.DR == 0);
   return U0RBR;
```

Write Data:

```
   //when U0LSR_bit.THRE is 0 - U0THR contains valid data.
   while (U0LSR_bit.THRE == 0);
   U0THR = ch0;
```

Programming: Real Time Clock

Register description

Register	Address	Function
ILR	0xE0024000	**Interrupt Location.** Reading this location indicates the source of an interrupt. Writing clears the associated interrupt.
CTC	0xE0024004	**Clock Tick Counter.** Value from the clock divider.
CCR	0xE0024008	**Clock Control Register.** Controls the function of the clock divider.
CIIR	0xE002400C	**Counter Increment Interrupt.** Selects which counters will generate an interrupt when
AMR	0xE0024010	**Alarm Mask Register.** Controls which of the alarm registers are masked. RW
CTIME0	0xE0024014	**Consolidated Time Register 0**
CTIME1	0xE0024018	**Consolidated Time Register 1**
CTIME2	0xE002401C	**Consolidated Time Register 2**

1.Initialization

1.1. Turn on the 32KHz external clock

CLKSRC (bit 4 from CCR Register)

0 - Disable 32kHz external clock
1 - Enable 32kHz external clock

1.2. Enable Interrupt

CIIR	Function	Description
0	IMSEC	When one, an increment of the Second value generates an interrupt.
1	IMMIN	When one, an increment of the Minute value generates an interrupt.
2	IMHOUR	When one, an increment of the Hour value generates an interrupt.
3	IMDOM	When one, an increment of the Day of Month value generates an interrupt.
4	IMDOW	When one, an increment of the Day of Week value generates an interrupt.
5	IMDOY	When one, an increment of the Day of Year value generates an interrupt.
6	IMMON	When one, an increment of the Month value generates an interrupt.
7	IMYEAR	When one, an increment of the Year value generates an interrupt.

1.3. Start the Real Time Clock

CLKEN (bit 0 from CCR Register) Enable/Disable Real Time Clock

0 - Disable Real Time Clock
1 - Enable Real Time Clock

2 . Example

Initialize:

```
   CCR_bit.CLKEN  = 0;     //rtc disable
   CCR_bit.CLKSRC = 1;     //set external 32kHz oscillator
   CCR_bit.CTCRST = 0;     //disable reset
   CCR_bit.CTTEST = 0;     //disable test
   AMR            = 0;     //initialize interrupt mask register of RTC
   CIIR_bit.IMSEC = 1;     //enable interupt every seconds
   ILR=0x3;                //clear all interrupt of RTC
   CCR_bit.CLKEN  = 1;     //rtc enable
```

[redacted]

Programming: Blinking LED

GPIO Register map

Generic name	Description
IOPIN	**GPIO Port Pin value register** . The current state of the GPIO configured port pins can always be read from this register, regardless of pin direction and mode. Activity on non-GPIO configured pi will not be reflected in this register.
IOSET	**GPIO Port Output set register.** This register controls the state of output pins in conjunction with the IOCLR register. Writing ones produces highs at the corresponding port pins. Writing zeroes has no effect.
IODIR	**GPIO Port Direction control register.** This register individually controls the direction of each po pin.
IOCLR	**GPIO Port Output clear register.** This register controls the state of output pins. Writing ones produces lows at the corresponding port pins and clears the corresponding bits in the IOSET register. Writing zeroes has no effect.

Pin Connect Block Register Map

Register name	Description
PINSEL0	PINSEL0 Pin function select register 0 (from P0.0 to P0.15)
PINSEL1	PINSEL0 Pin function select register 1 (from P0.16 to P0.31)
PINSEL2	PINSEL0 Pin function select register 2

1. Initialization

1.1. Set first functionality to port

```
PINSEL0 = 0x00;  //set first functionality to port (from P0.0 to P0.15)
```

1.2. Set port which is connected to LED as output

```
IO0PIN = 0xFF;    //set all port to output
```

2. Led blink

```
IO0CLR = 0xFF;    // set all port to low
IO0SET = 0xFF;    // set all port to high
```

3. Example - blink led, which is connected to P0.7

//Initialization

```
PINSEL0_bit.P0_7 = 0x00;  //set first functionality to port (from P0.0 to P0.15)
IO0DIR_bit.P0_7 = 0x01;    //set P0.7 port to output
IO0SET_bit.P0_7 = 0x01;   //set P0.7 port to high
```

```
//loop forever
while(1)
{
   Delay(1000);                //Simple delay
   IO0SET_bit.P0_7 = 0x01;   // set P0.7 port to high
   Delay(1000);                //Simple delay
   IO0CLR_bit.P0_7 = 0x01;   // set P0.7 port to high
}
```

Links

1. Philips web site

LPC2294 product datasheets, application notes, etc info:
http://www.semiconductors.philips.com/

2. LPC microcontrollers discussion forum

http://groups.yahoo.com/group/lpc2000/ - forum for discussions on LPC2000 ARM microcontrollers
http://groups.yahoo.com/group/arm-olimex/ - forum for discussions on Olimex ARM boards

3. IAR Systems EW-ARM C compiler and debugger

http://www.iar.com/Products/?name=EWARM

4. Rowley associates CrossWorks for ARM C compiler and debugger

http://www.rowley.co.uk