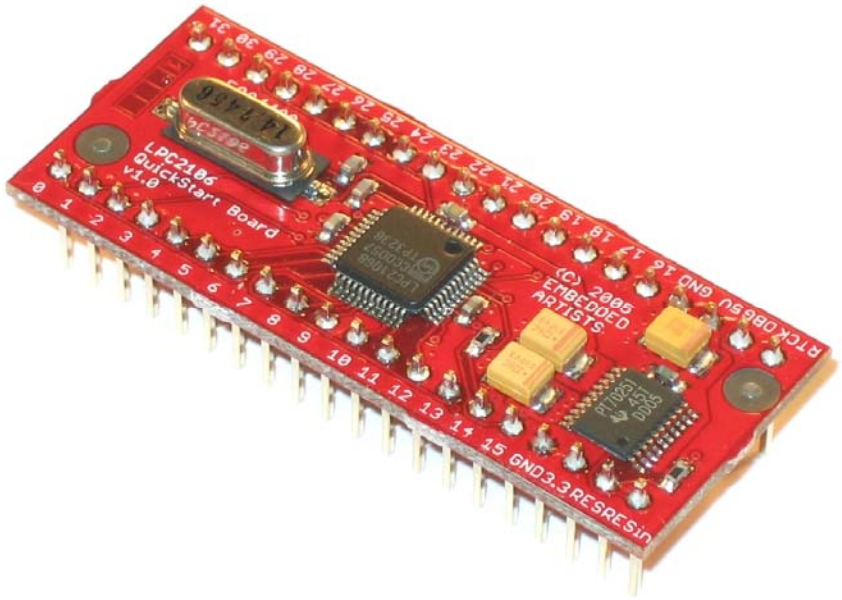
LPC2106 QuickStart Board User's Guide



Get Up-and-Running Quickly and Start Developing on Day 1…

Embedded Artists

Embedded Artists AB
Friisgatan 33
SE-214 21 Malmö
Sweden

info@EmbeddedArtists.com
http://www.EmbeddedArtists.com

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Please send your comments to support@EmbeddedArtists.com.

Trademarks

InfraBed and ESIC are trademarks of Embedded Artists AB. All other brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1 Introduction

Thank you for buying Embedded Artists' *LPC2106 QuickStart Board* based on Philips ARM7TDMI LPC2106 microcontroller.

This document is a User's Guide that describes the *LPC2106 QuickStart Board* design along with the accompanying software and program development tools. The document contains information on how to use and integrate the board in your own designs, including electrical and mechanical information.

1.1 Contents

The box received when ordering the *LPC2106 QuickStart Board* contains the following:

- The *LPC2106 QuickStart Board*.
- CD-ROM which includes additional material and programs, including complete and evaluation versions of different development environments. Observe that bulk orders (10 or 100 boards) only include one CD-ROM.

In addition, the following is needed in order to start developing applications with the *LPC2106 QuickStart Board*:

- A DC power supply, 4-6 volt, capable of providing at least 150 mA (more if external circuits need power from the 3.3 volt supply). Observe that the *LPC2106 QuickStart Board* does not contain any reverse polarity protection. If voltage is applied with wrong polarity, the board will likely be damaged. Also observe that 6.0 volt is the absolute maximum voltage that can be applied without damaging the on-board voltage regulator (TPS70251). Consult the TPS70251 datasheet for exact details.
- A serial ISP interface for downloading application programs (via a PC).
- An optional JTAG interface, for program development debugging.

1.2 Features

Embedded Artists' *LPC2106 QuickStart Board* lets you get up-and-running quickly with Philips ARM7TDMI LPC2106 microcontroller. The small form factor board offers many unique features that ease your development.

- Philips ARM7TDMI LPC2106 microcontroller with 128 Kbyte program Flash and 64 Kbyte SRAM
- All LPC2106 I/O pins are available on connectors
- 14.7456 MHz crystal for maximum execution speed and standard serial bit rates
  - Phase-locked loop (PLL) multiplies frequency with four; 4 x 14.7456 MHz = 58.9825 MHz
- Onboard low-dropout voltage and reset generation.
  - Generates +3.3V and +1.8V from a single +5V supply
  - +3.3V available for external circuits, up to 300 mA
  - Power supply: 4-6 VDC, at least 150 mA
- Simple and automatic program download (ISP) via serial channel
- Easy to connect to JTAG signals
- Dimensions: 21.5 x 53.25 mm

- DIL form factor with 2x20 pins (15.24 mm, 0.60" between pin rows)
- Four layer PCB (FR-4 material) for best noise immunity

1.3 Low Cost

The *LPC2106 QuickStart Board* is very low cost and can be used for prototyping / development as well as for OEM production. Modifications for OEM production can easily be done for volumes > 1k. Contact Embedded Artists for further information about design and production services.

Bulk orders (> 10 boards) can be delivered without pin rows soldered for easy integration with original equipment.

1.3.1 Design and Production Services

Embedded Artists provide design services for custom designs, either completely new or modification to existing boards. Specific peripherals and/or I/O can easily be added to the different designs, for example communication interfaces, specific analogue or digital I/O, and power supplies. Embedded Artists has a broad, and long, experience in designing industrial electronics in general, and specifically with Philips LPC2xxx microcontroller family.

- Prototype and low-volume production takes place in Sweden for best flexibility and short lead times.
- High-volume production takes place in China for lowest possible cost.

1.4 Other QuickStart Boards and Kits

Visit Embedded Artists' home page, www.EmbeddedArtists.com, for information about other *QuickStart* boards / kits or contact your local distributor.

2 Board Design

This chapter contains detailed information about the electrical and mechanical design of the *LPC2106 QuickStart Board*. A number of example circuits are also presented that will lower the threshold of start developing with the board.

2.1 Board Schematics

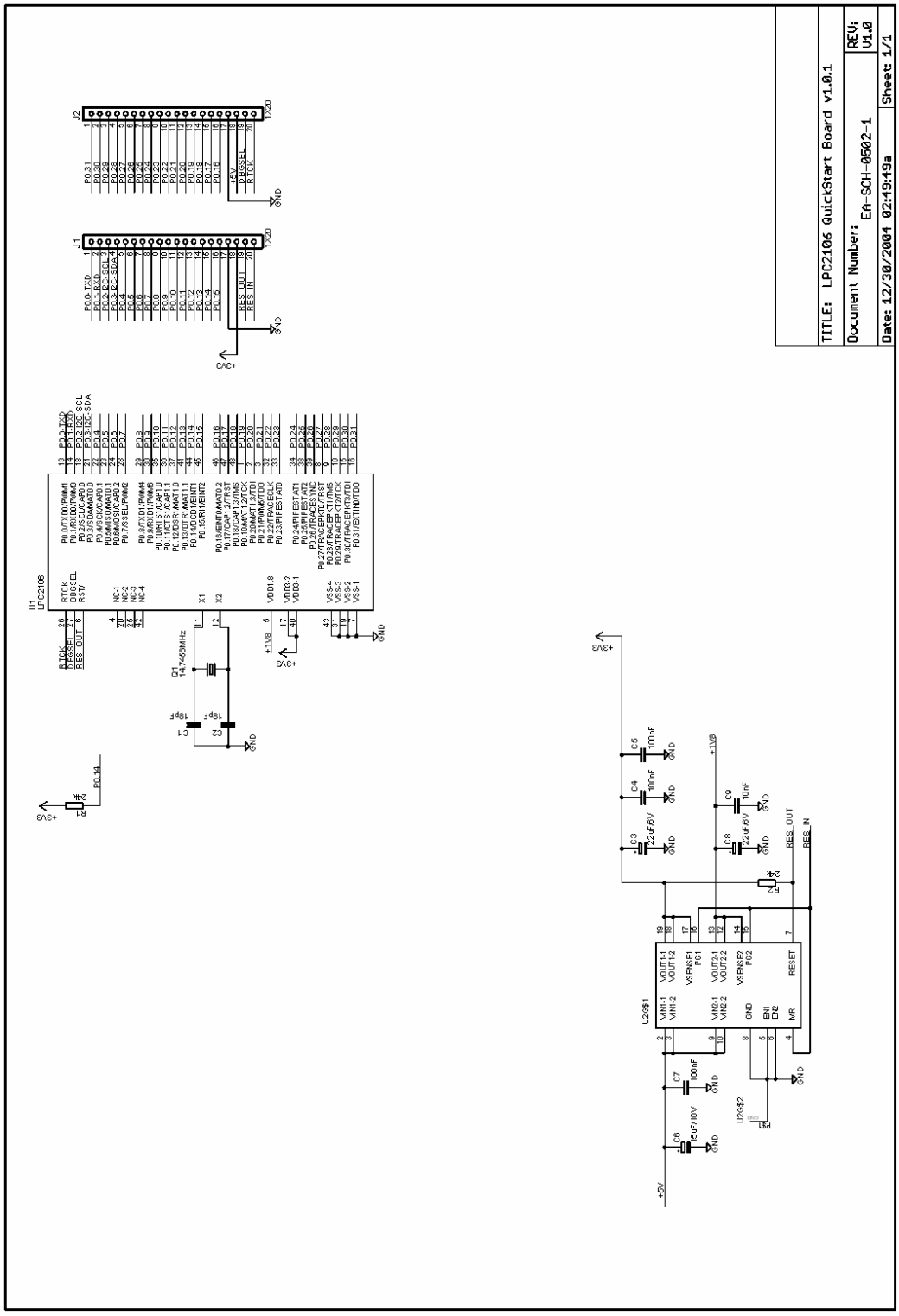


Figure 1 - LPC2106 QuickStart Board Schematic

Besides the LPC2106 microcontroller from Philips, the board contains a dual voltage regulator with an internal reset generator.

The microcontroller crystal frequency is 14.7456 MHz. This frequency has been selected in order to allow close to maximum execution speed (4 x 14.7456 MHz = 58.9824 MHz, which is very close to the maximum frequency, 60 MHz) as well as to provide standard serial communication bit rates. The crystal frequency can be changed to any desired value for OEM orders, provided that the conditions in the LPC2106 datasheet are met. Current requirements are (but consult the most current datasheet for latest details):

- 1-30 MHz if the on-chip phase-locked loop (PLL) is not used, or
- 10-25 MHz if the PLL is to be used.

The board interface connectors are placed in two 20 pin rows along the board edges. They are 600 mil (0.60 inch, 15.24 mm) apart and can be viewed as a standard DIL-40 package. *Figure 2* below illustrates how the two row connectors can be viewed as a DIL package.

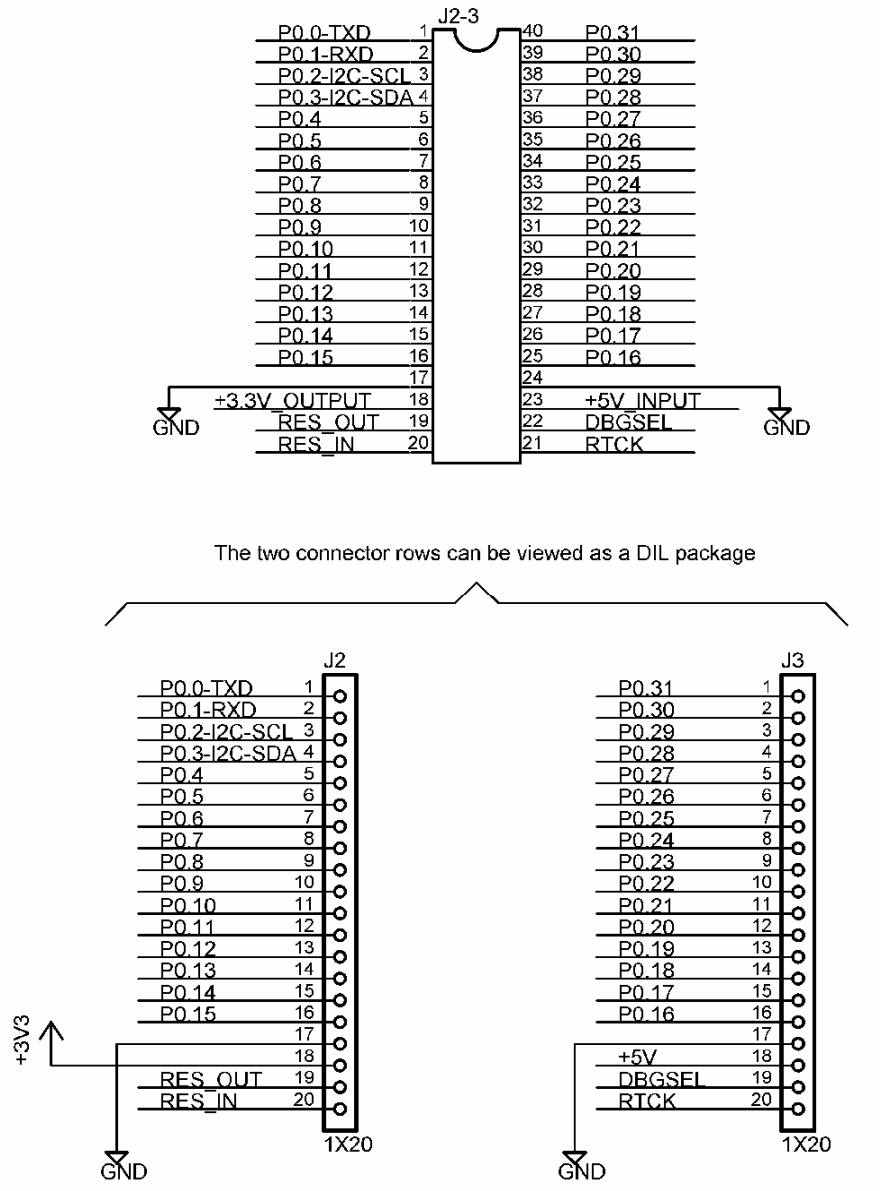


Figure 2 - LPC2106 QuickStart Board Interface Connectors Viewed as a DIL Package

2.2 Mechanical Dimensions

Figure 3 below contains a drawing of the board that includes mechanical measures.

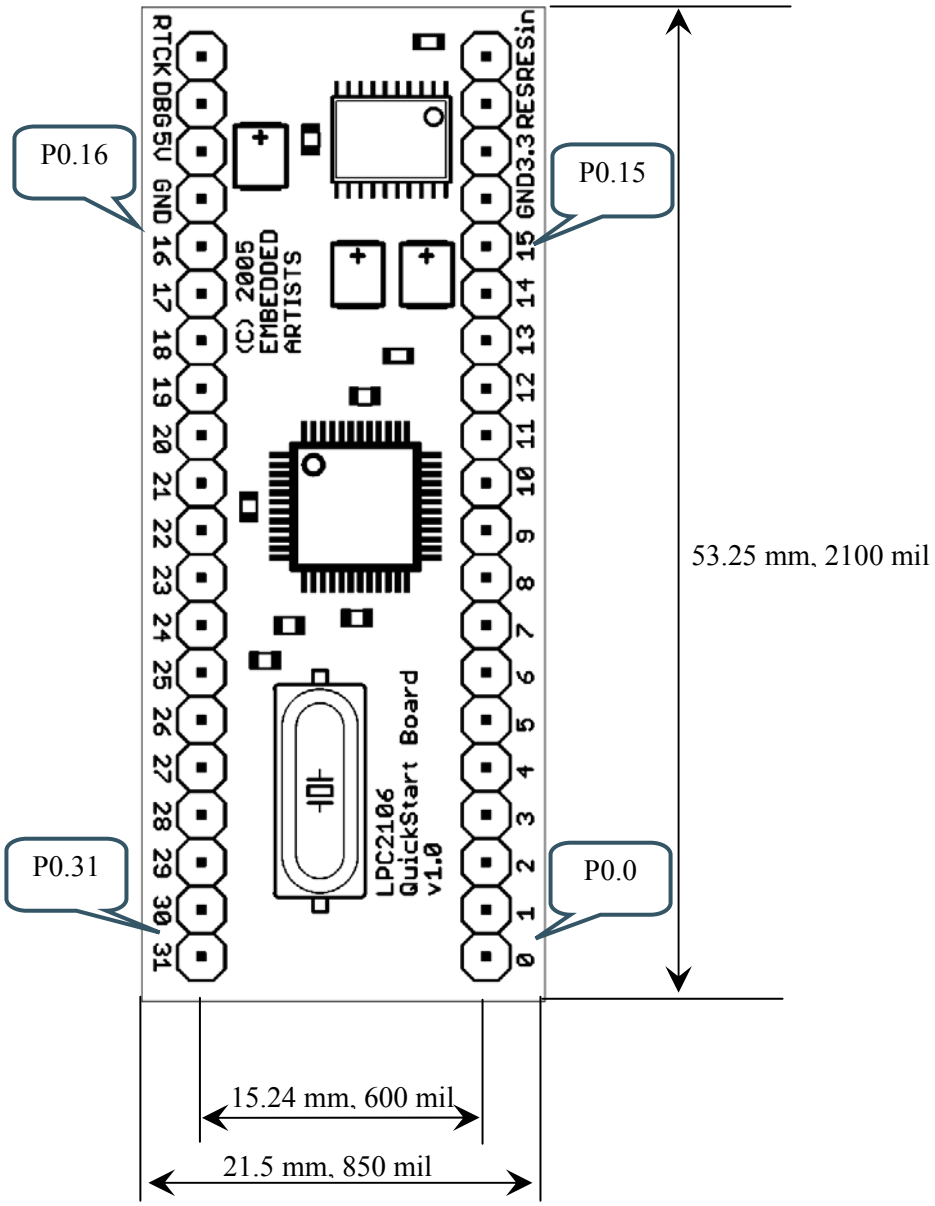


Figure 3 - LPC2106 QuickStart Board Mechanical Dimensions

2.3 Examples

This section contains a few sample / illustrative circuit examples that will help you to quickly get up-and-running with the board interface design. Detailed information about the on-chip peripheral units can be found in the LPC2106 User's Manual.

2.3.1 ISP

The LPC2106 microcontroller contains ISP functionality that allows you to easily download a program via the serial channel (UART #0, pin P0.0 and P0.1). The circuit in *Figure 4* below illustrates a simple RS232 interface. It also contains automatic bootloader activation functionality, by controlling the state of pin P0.14 and the reset signal. Pin P0.14 is sampled by the microcontroller after reset and determines if the internal bootloader program shall be started, or not. A low signal after reset enters the bootloader mode. The RS232 signal RTS is connected to pin P0.14 and the RS232 signal DTR controls the reset signal to the LPC2106. Philips FLASH Utility program (for ISP) supports automatic bootloader activation.

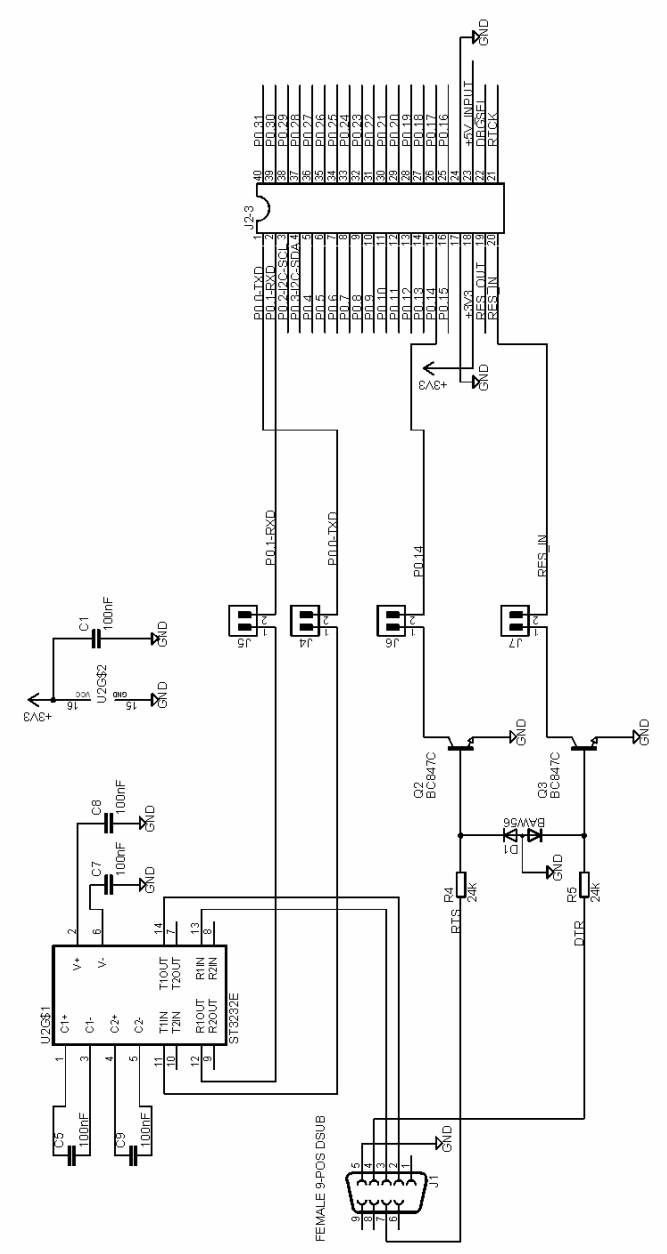


Figure 4 – Example ISP (serial, RS232) Interface

2.3.2 JTAG

The LPC2106 microcontroller contains a JTAG interface that can be used for debug purposes during program development. The circuit in *Figure 5* below works for many JTAG interfaces on the market, including J-link from Segger, Ulink from Keil, and Wiggler from MacRaigor.

The signal DBGSEL on the LPC2106 microcontroller is sampled during reset. Jumper J10 drives the signal high. If the signal is found high, the JTAG interface is enabled. Pin P0.17-P0.21 then changes from being general I/O pins to dedicated JTAG pins.

Some JTAG interfaces require the signal RTCK to be held low (jumper J11) and some that the JTAG signals TRST and RST are connected (optional 0 ohm resistor). Consult your JTAG manual for specific information.

Note that many Wiggler JTAG interfaces do not work with a processor crystal frequency above about 10 MHz. If this is the case, the crystal frequency can be changed by desoldering the 14.7456 MHz crystal and replace it with another suitable one.

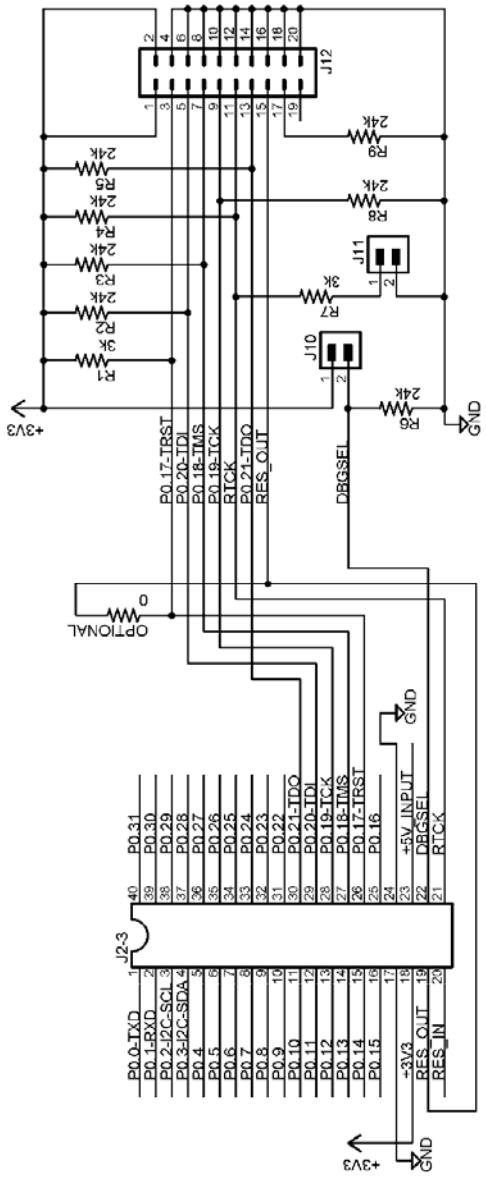


Figure 5 – Example JTAG Interface

2.3.3 Reset

The on-board voltage regulator also contains a reset generator. The reset signal will be held active (i.e., low) until both internal voltages (i.e., +3.3V and +1.8V) are within margins. The reset duration is about 120 mS (consult the TPS70251 datasheet for exact details). The output reset signal is an open-collector / open-drain output. An external reset source can also control the reset generator. *Figure 6* below illustrate how an external push-button can generate a reset. Observe that an external driver must be an open-collector / open-drain driver.

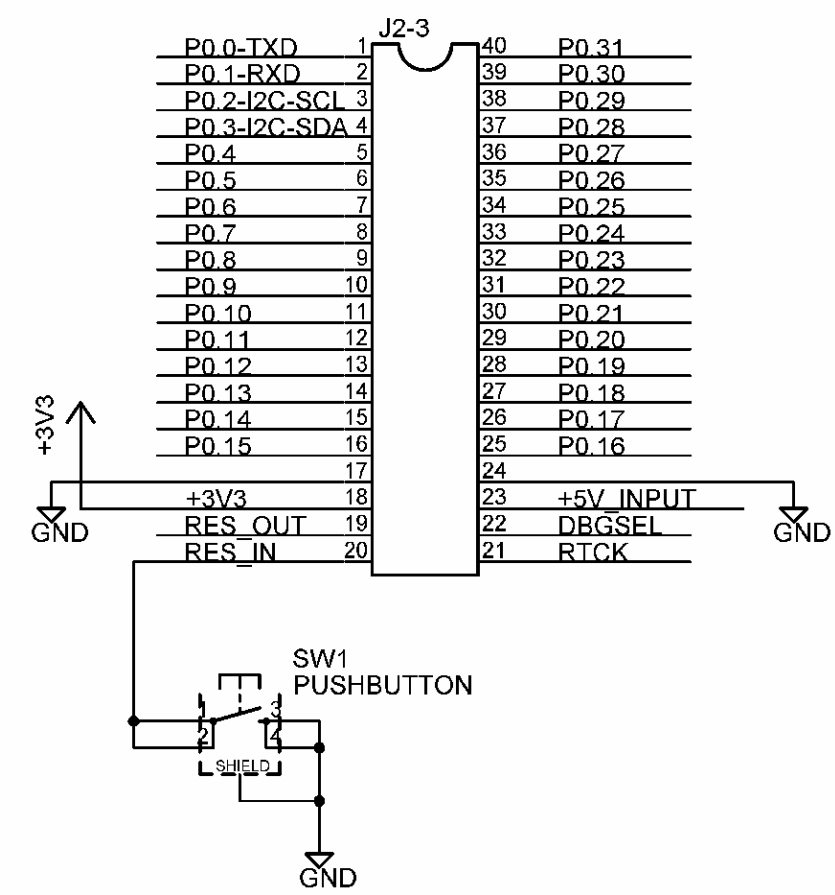


Figure 6 – Example External Reset Push-button

2.3.4 I²C

The LPC2106 microcontroller has an on-chip I²C communication channel. I²C peripheral units are easily connected to the two-wire bus. *Figure 7* below illustrates how an E²PROM can be connected to the I²C bus.

Observe that the pull-up resistors (which are always needed on I²C busses) are not included on the *LPC2106 QuickStart Board*., and are hence needed on the external circuit. Typical values for these pull-up resistors are 3000 ohm each, but consult the datasheet of your specific circuit for exact details.

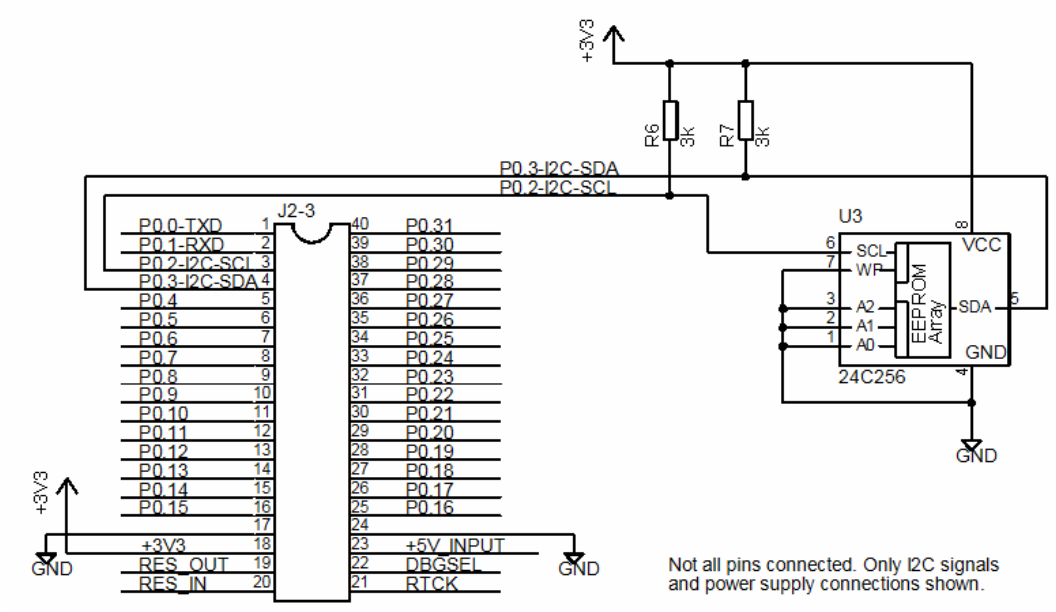


Figure 7 – Example I2C Interface

2.3.5 SPI

The LPC2106 microcontroller has an on-chip SPI serial communication channel. *Figure 8* below illustrates how serial E^2PROM chip is connected to the *LPC2106 QuickStart Board*. Observe that signal SSEL (i.e., P0.7) must be high when the SPI controller (in LPC2106) operates as a master, and master operation is the normal operating mode. In *Figure 8* below, signal P0.25 is used as an example to control the chip select to the serial E^2PROM chip. Any other pin can be used to control chip select signals. Observe that one chip select signal is requires for each external chip that is connected to the SPI bus.

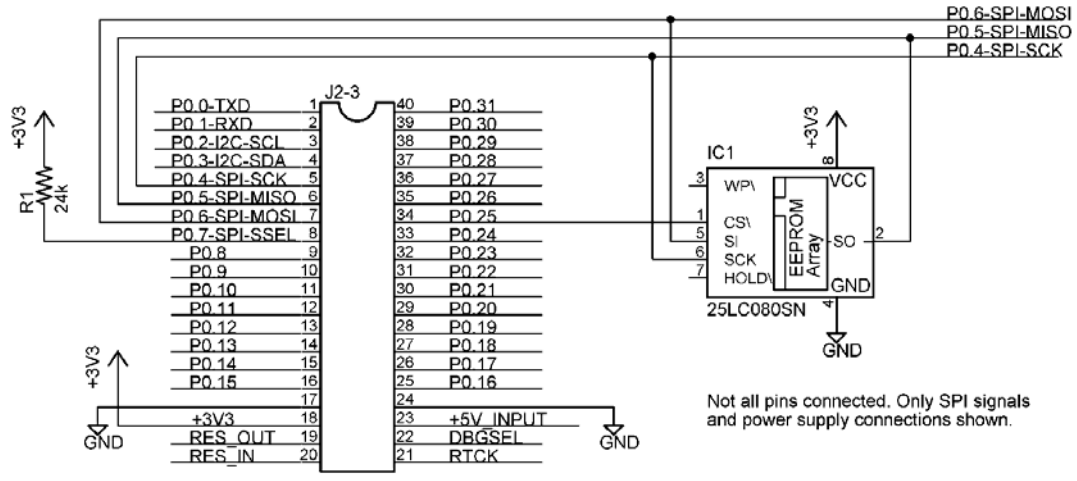


Figure 8 – Example SPI Interface

2.3.6 LEDs

The port pins of the LPC2106 microcontrollers have a 4 mA driving capacity, just enough to directly drive LEDs. *Figure 9* below illustrates how current is sunk into the microcontroller to drive the LEDs. Current sourcing is possible just as well. The resistors limit the current to about 4 mA. The preloaded test program (described in *Section 3.1*) outputs a running-zero on the port pins (P0.2 – P0.31), in order to create a running-one pattern on the LEDs. A circuit like the one in *Figure 9* below can be used to verify correct operation.

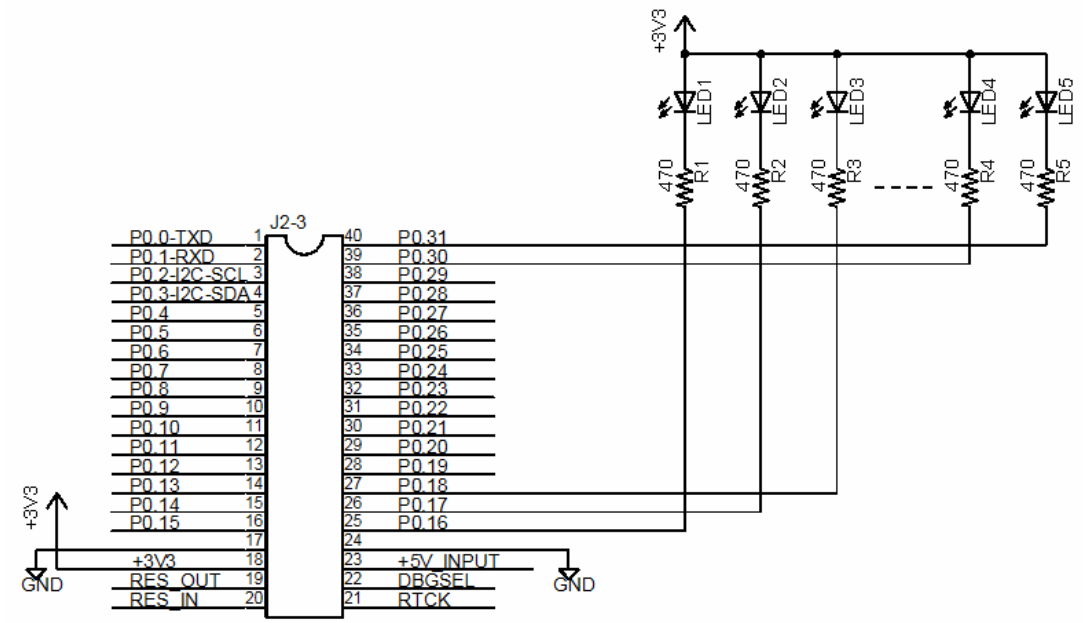


Figure 9 – Example LED Driving

3 Getting Started

3.1 Test program

The *LPC2106 QuickStart Board* comes preloaded with a test program. This program can be used to verify that the board operates correctly. A circuit, like the one found in *Figure 9*, can be used to attach LEDs to port pins P0.2 – P0.31. Pins P0.0 – P0.1 are tested via the serial channel.

The test program outputs a running-zero to port pins P0.2 – P0.31, meaning that one LED at a time will light (in a running-one pattern). Also, a terminal program can be attached to the serial channel (UART #0), assuming a circuit like the one on *Figure 4*. The UART/RS232 channel can be tested by typing characters in the terminal program.

The settings for the terminal program are: 115.2 kbps, 8 data bits, no parity bits, and one stop bit (i.e., 8N1).

The output from the test program will look something like in *Figure 10* below.

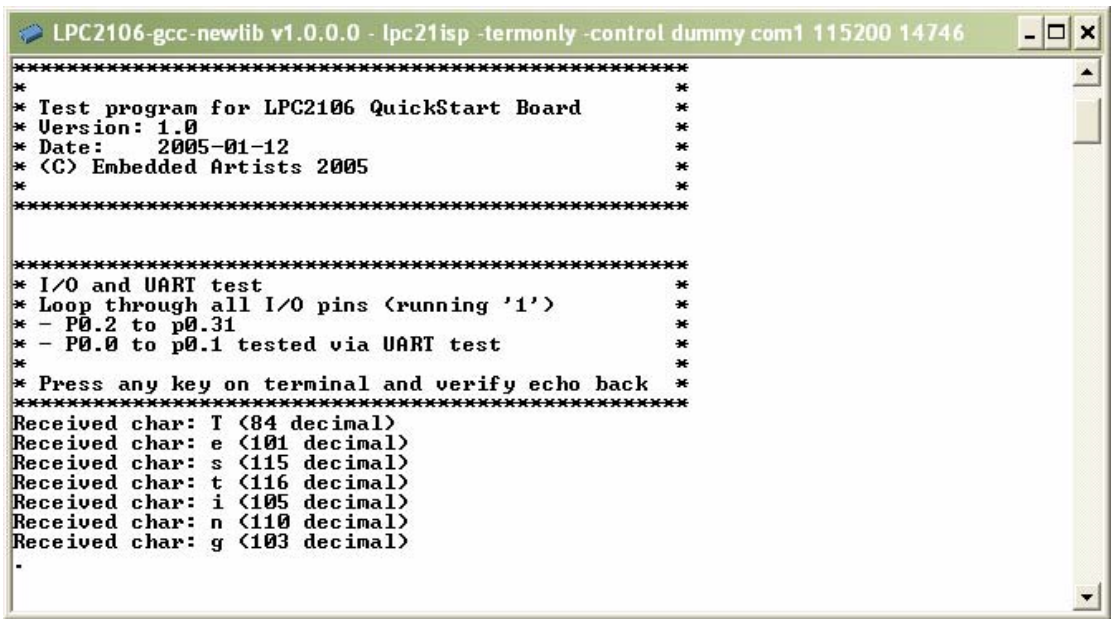


Figure 10 – Example Test Program Output

3.2 Program Download

For now, it is assumed that the program to be downloaded is already developed and there exist a HEX-file to be downloaded. This HEX-file represents the binary image of the application program.

There are basically two ways of downloading a program into the LPC2106 microcontroller:

- ISP – In-System Programming
  The LPC2106 microcontroller provides on-chip bootloader software that allows programming of the internal flash memory over the serial channel. The bootloader is activated by pulling port pin P0.14 low during reset of the microcontroller. The circuit in *Figure 4* can be used to implement automatic control of the bootloader.
    - Philips provides a utility program for In-System Flash (ISP) programming called *LPC2000 Flash Utility*.
    - Alternatively, there is a program called LPC21ISP that can be used. Source code is available. This program also provides a terminal functionality, which

can be very helpful when developing your application program. The same serial channel that is used to download the program is typically also used for printing out information from the running program. The program immediately switch to terminal mode after program download and will hence not miss any characters sent on the serial channel directly after program start.

The installation files for both programs can be found on the accompanying CD-ROM.

- JTAG
  For specific information about program download (i.e., Flash programming) with a JTAG interface, consult the manual for the specific JTAG interface that is used (e.g., J-link from Segger, Ulink from Keil, or Wiggler from MacRaigor).

3.2.1 Philips LPC2000 Flash Utility

Philips LPC2000 Flash Utility program looks like *Figure 11* below.

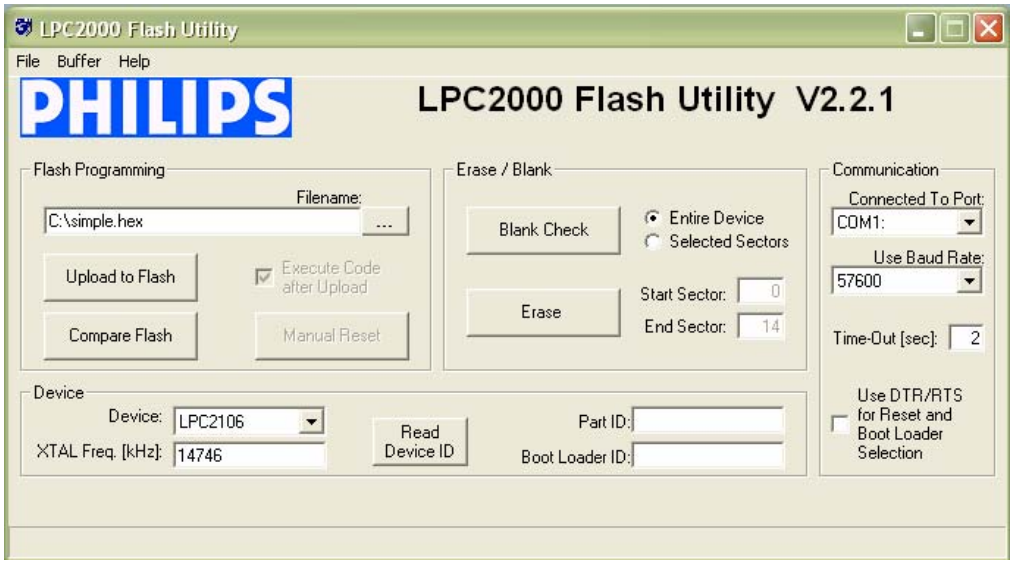


Figure 11 – Philips LPC2000 Flash Utility Screenshot

Configure the dialog as shown above. If automatic control of the bootloader is implemented in hardware the program will control the RS232 signals DTR and RTS if the appropriate checkbox is checked (checkbox on bottom right), and hence provide fully automated program download.

Test connection with the *LPC2106 QuickStart Board* by pressing the *Read Device ID* button. The text fields for *Part ID* and *Boot Loader ID* will then contain uploaded information from the microcontroller. Observe that the XTAL Freq. must be set to appropriate value. The default mounted crystal frequency on the *LPC2106 QuickStart Board* is 14.7456 MHz. In this case the value 14746 shall be written in the text box. If the crystal frequency has been changed, make sure the appropriate value is set. If no connection can be established test with a low *Baud Rate*, for example 1200 bps. Also verify that the correct COM-port has been selected (under *Connected to Port*).

Select the HEX file to be downloaded and then press the *Upload to Flash* button.

The downloaded program will immediately start after the download (i.e. the *Upload to Flash* operation is ready) is the option *Execute Code after Upload* is checked.

3.2.2 LPC21ISP

The LPC21ISP program is made publicly available by Martin Maurer. Source code is also available at: http://engelschall.com/~martin/lpc21xx/isp/index.html. *Figure 12* below shows the command syntax for the program.

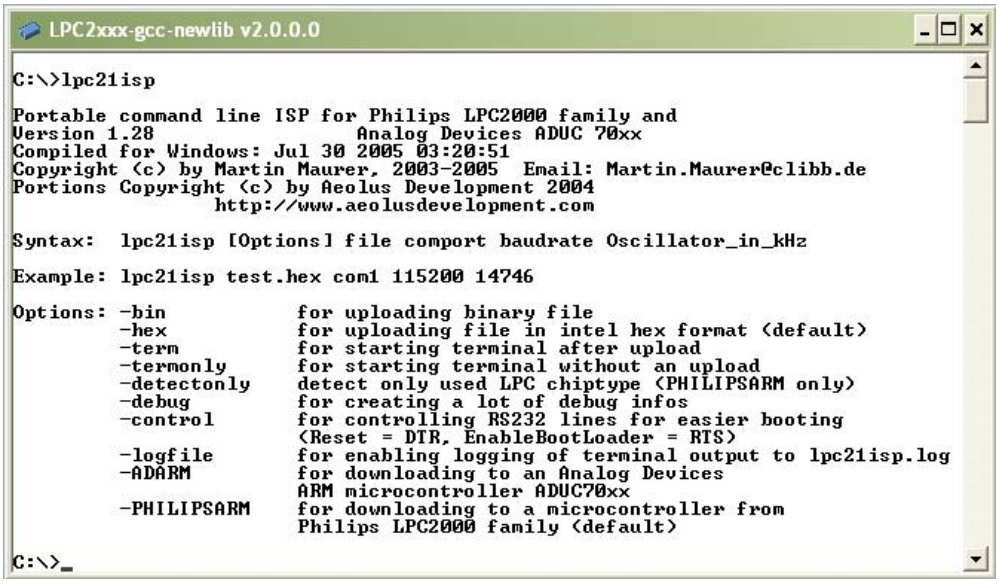


Figure 12 – LPC21ISP Portable Command Line ISP Screenshot

A typical program download sequence may look like in *Figure 13* below. Here, the test program is downloaded. As seen, the first part is the actual program download phase. Then this is done, the program switches to being a terminal (the second part) and the messages from the test program is displayed. It also sends anything typed on the keyboard back to the *LPC2106 QuickStart Board*. As seen the program ends when ESC is pressed.

This sequence illustrates the benefits from using the program as a terminal directly after program download. No characters are missed after program start.

The used command is:

```
lpc21isp -term -control testprogram_lpc2106_qsb.hex com1 115200 14746
```

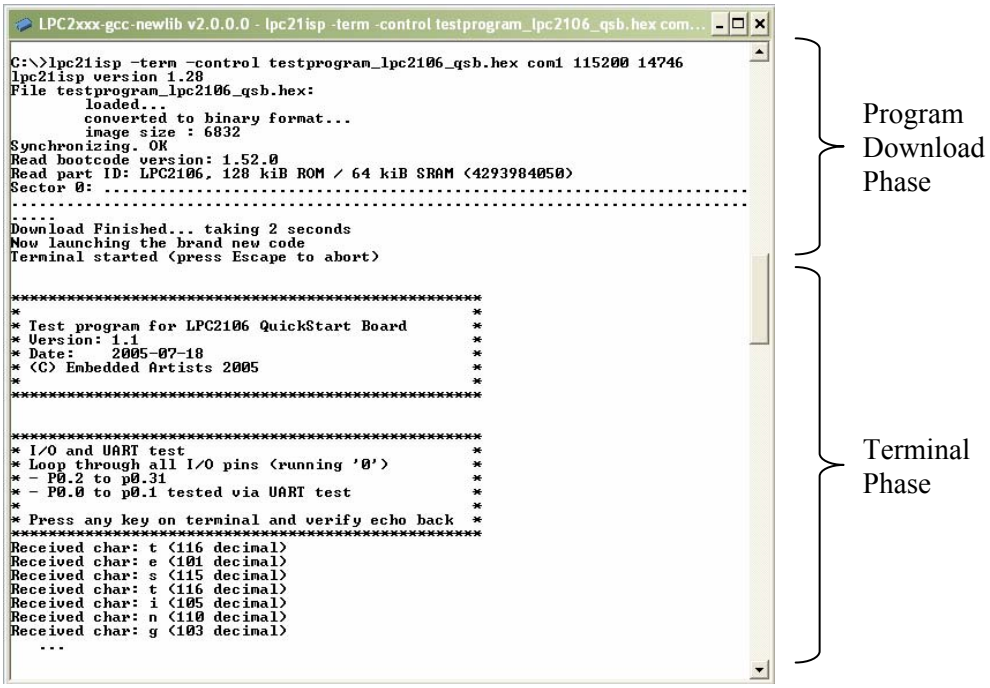


Figure 13 – LPC21ISP Portable Command Line ISP Download Screenshot

Another benefit with this program is that it runs under Linux.

Use version 1.28, or later, of LPC21ISP.EXE since older versions must be recompiled with increased reset timeout (when the program tries to synchronize to the *LPC213x QuickStart Board*). The timeout should be increased to at least 350 ms.

3.3 Program Development

There are many options when it comes to the actual application program development. First of all, you must select a development environment, i.e., an editor (preferably with project management capabilities), a compiler package (compiler plus linker), and a debugger. Fortunately, there are many different choices for ARM program development, each with its pros and cons. The list below is far from complete but gives a general overview. The accompanying CD-ROM (see *Section 4.1* for more details) contains many of these programs / environments.

- ***QuickStart Build Environment from Embedded Artists***
  Embedded Artists has created a complete GCC build environment for all QuickStart boards. This will ease program development for novel users. By installing the *QuickStart Build Environment* you will automatically get a complete setup of the build environment.
- ***Rowley Associates CrossWorks for ARM***
  A complete development environment from Rowley Associates, including an editor, project manager, a complete compiler build environment, and a debugger. The version included on the CD is a 30-day fully functional evaluation version.
- ***IAR Embedded Workbench***
  A complete development environment from IAR Systems, including an editor, project manager, a complete compiler build environment, and a debugger. The version shipped with the *LPC2106 QuickStart Board* has a 32 Kbyte program size limit, but is fully functional in all other aspects.
- ***Keil uVision***
  This is another complete development environment, but from Keil. It includes an

editor, project manager, a complete compiler build environment, and a debugger. An evaluation version can be downloaded from Keils homepage. One version of the development environment is based on the GCC compiler (currently version 3.3.1 of GCC).

- ***Programmers notepad***
  This is a very good editor and project manager that is increasing in popularity. The program can easily be integrated with the GCC compiler.
- ***Eclipse + CDT***
  This is a very good development environment (editor and project manager) with specific support for C/C++ code development. It does not contain a compiler but can easily be connected to one, for example GCC.
- ***GCC distribution GNUARM***
  A complete distribution of GCC, specifically for ARM processors. Current version of GCC is 3.4.3 this it is constantly updated.
- ***WinARM***
  This is another distribution that not only contains GCC but also Programmers Notepad, LPC21ISP, a terminal program, and JTAG drivers.

3.3.1 QuickStart Build Environment

The *QuickStart Build Environment* is a complete build environment for GCC including program downloading via ISP. The build environment is built around a bash script. This script sets up all necessary paths. When installing the *QuickStart Build Environment* you will automatically get shortcuts to this bash script. A practical feature is that there can be different scripts for different hardware platforms, for controlling different hardware specific details of the platforms. There can also be many different compilers (including different versions of the same compiler) without conflicting with each other.

The use of the bash script is optional but is recommended for non-experienced users.

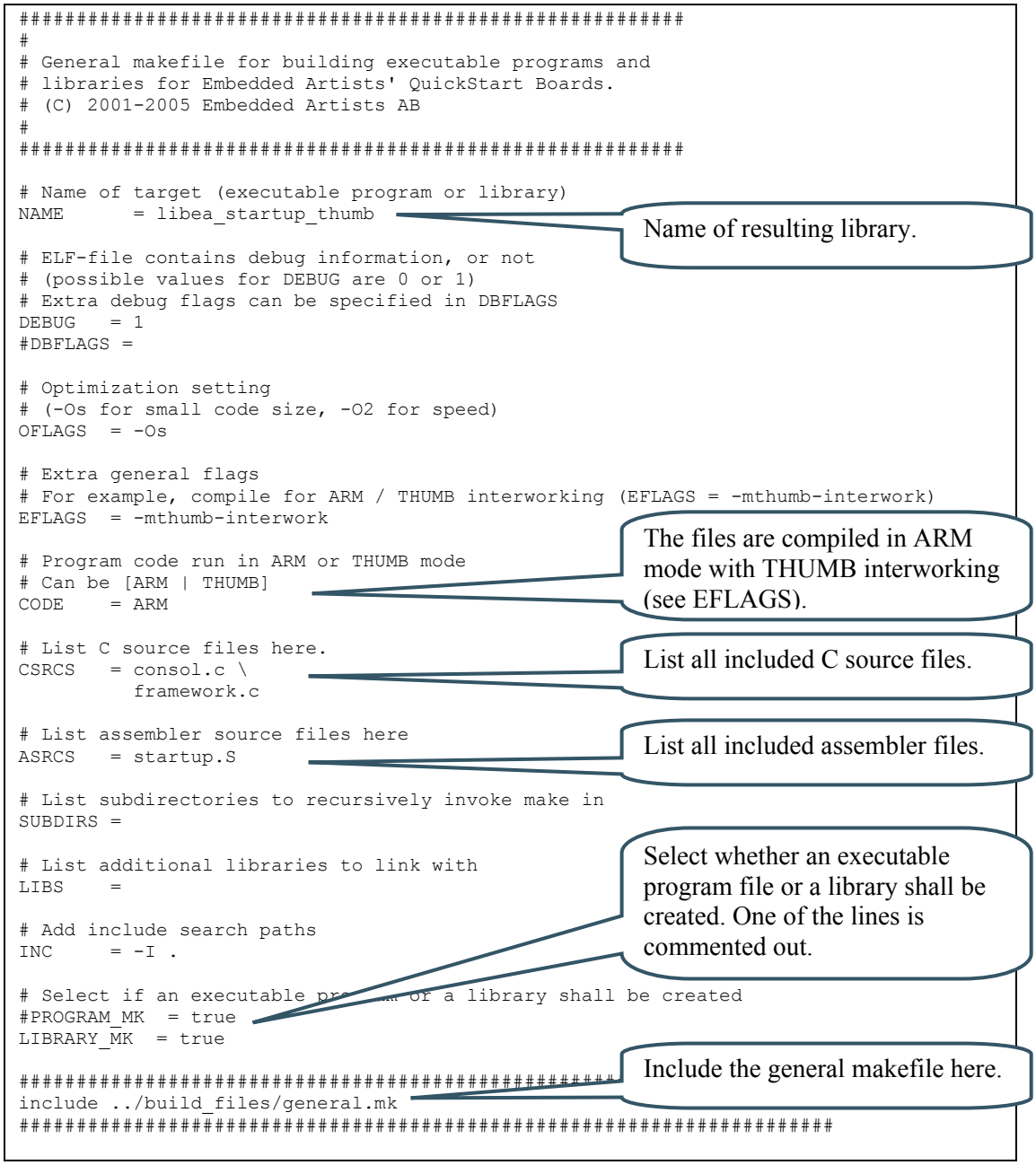
A typical project has two subdirectories; **`build_files`** and **`startup`**. *Figure 14* below illustrate the general structure.

Figure 14 – Typical Project Directory Structure

The subdirectory **`build_files`** contains a general makefile and linker script files. The subdirectory startup contains a configurable startup framework for QuickStart Board projects. The startup files form a library that is linked to the main application.

The makefiles have a hierarchical structure. Each project, either an executable program file or a library, has a simple **`makefile`** that just describe the specifics of the project. This simple **`makefile`** includes the general **`makefile`** that is placed in the **`build_files`** subdirectory.

Figure 15 below illustrates the simple **`makefile`**. The example comes from the startup library, found under the **`startup`** subdirectory. The name of the resulting library is **`libea_startup_thumb.a`**. Two C-source code files are listed: **`consol.c`** and **`framework.c`**. An assembler file called **`startup.S`** is also included in the library.




```
############################################################
#
# General makefile for building executable programs and
# libraries for Embedded Artists' QuickStart Boards.
# (C) 2001-2005 Embedded Artists AB
#
############################################################

# Name of target (executable program or library)
NAME      = libea_startup_thumb

# ELF-file contains debug information, or not
# (possible values for DEBUG are 0 or 1)
# Extra debug flags can be specified in DBFLAGS
DEBUG   = 1
#DBFLAGS =

# Optimization setting
# (-Os for small code size, -O2 for speed)
OFLAGS  = -Os

# Extra general flags
# For example, compile for ARM / THUMB interworking (EFLAGS = -mthumb-interwork)
EFLAGS  = -mthumb-interwork

# Program code run in ARM or THUMB mode
# Can be [ARM | THUMB]
CODE    = ARM

# List C source files here.
CSRCS   = consol.c \
          framework.c

# List assembler source files here
ASRCS   = startup.S

# List subdirectories to recursively invoke make in
SUBDIRS =

# List additional libraries to link with
LIBS    =

# Add include search paths
INC     = -I .

# Select if an executable program or a library shall be created
#PROGRAM_MK  = true
LIBRARY_MK  = true

#############################################################
include ../build_files/general.mk
#############################################################
```

Figure 15 – Example QuickStart Build Environment Makefile from Startup Library

As seen in *Figure 15* above the makefile ends with the command: **`include build_files/general.mk`**. This is a general make file that is part of the complete build environment. This part contains all specific details of compiler and linker invocation.

Also at the end, the target must be decided; either an executable program or a library. Either **`PROGRAM_MK`** or **`LIBRARY_MK`** must be set to **`true`**.

The example makefile above is quite simple to its structure. It is possible to create more complex project structures that contain many subprojects. A typical example is to have an application project in a root folder. Under this root folder a number of subdirectories exist containing different blocks of functionality. For example, this can be a Real-Time Operating System and a TCP/IP stack. This calls for a recursive **`makefile`** structure.

The **makefile** in the root filer will create an executable program. It also includes the **makefile** in each of the subdirectories. The **makefiles** that exist in subdirectories will create libraries. An example of a root make file is presented in *Figure 16* below.

```
############################################################
#
# General makefile for building executable programs and
# libraries for Embedded Artists' QuickStart Boards.
# (C) 2001-2005 Embedded Artists AB
#
############################################################

# Name of target (executable program or library)
NAME      = testprogram

# Path and name of linker script file
# Only needed for executable program files
LD_SCRIPT = build_files/link_rom.ld

# ELF-file contains debug information, or not
# (possible values for DEBUG are 0 or 1)
# Extra debug flags can be specified in DBFLAGS
DEBUG   = 1
#DBFLAGS =

# Optimization setting
# (-Os for small code size, -O2 for speed)
OFLAGS  = -Os

# Extra general flags
# For example, compile for ARM / THUMB interworking (EFLAGS = -mthumb-interwork)
EFLAGS  =

# Program code run in ARM or THUMB mode
# Can be [ARM | THUMB]
CODE    = THUMB

# List C source files here.
CSRCS   = main.c

# List assembler source files here
ASRCS   =

# List subdirectories to recursively invoke make in
SUBDIRS = startup \
          tcpip \
          pre_emptive_os

# List additional libraries to link with
LIBS    = startup/libea_startup_thumb.a \
          tcpip/tcpip.a \
          pre_emptive_os/pre_emptive_os.a

# Add include search path for startup files, and othe
INC     = -I./startup

# Select if an executable program or a library shall be created
PROGRAM_MK  = true
#LIBRARY_MK  = true

# Output format on hex file (if making a program); can be [srec | ihex]
HEX_FORMAT  = ihex

# Program to download executable program file into microcontroller's FLASH
DOWNLOAD    = lpc21isp.exe

# Configurations for download program
DL_COMPORT  = com1
DL_BAUDRATE = 115200
DL_CRYSTAL  = 14746

######################################################################
include build_files/general.mk
```

Name of resulting program file.

Define linker script.

The files are compiled in THUMB mode.

The root folder only contains one file, the main-file.

Three different subdirectories that contains different blocks of functions in the final application.

The three libraries that are created in the recursive invocation of make are included in the final application.
Note the startup library.

```
##############################################################################
```

Figure 16 – Example Root Makefile and Recursive Invocation

To build the application program, start a command prompt (the bash script), change directory to the project root, and type: **`make`**. Depending on the make file content, either an executable program or a library will be created. To also download the executable program, type: **`make deploy`** instead of just **`make`**.

A final note about the make file; **`make clean`** will erase all object files and **`make depend`** will recreate dependency files (this is also always done when typing just **`make`**). Finally, **`make terminal`** will just start the terminal function in the download program (lpc21isp). The specific settings for using the ISP download program can be set with the **`DL_XXX`** variables (as seen at the end of *Figure 16* above).

As already mentioned, the startup files form a configurable startup framework. This is often called a Board Support package or BSP for short. It contains the very basic startup and initialization code as well as a console with printf()- and scanf()-like functionality. The BSP is very configurable and can be changed according to your specific needs. Each project can have its specific settings. The configuration file is listed in *Figure 17* below, and can be found in file **`config.h`** in the **`startup`** subdirectory.

```
/*****************************************************************************
 *
 * Copyright:
 *    (C) 2000 - 2005 Embedded Artists AB
 *
 * Description:
 *    Framework for ARM7 processor
 *
 *****************************************************************************/
#ifndef _config_h_
#define _config_h_

/*****************************************************************************
 * Defines, macros, and typedefs
 *****************************************************************************/

#define FOSC 14745600                /* External clock input frequency
                                        (must be between 10 MHz and 25 MHz) */

#define USE_PLL 1                    /* 0 = do not use on-chip PLL,
                                        1 = use on-chip PLL) */
#define PLL_MUL 4                    /* PLL multiplication factor (1 to 32) */
#define PLL_DIV 2                    /* PLL division factor (1, 2, 4, or 8) */
#define PBSD    4                    /* Peripheral bus speed divider (1, 2, or 4) */

/* initialize the MAM (Memory Accelerator Module) */
#if (FOSC * PLL_MUL) < 20000000
#define MAM_TIMING   1               /* number of CCLK to read from the FLASH */
#elif (FOSC * PLL_MUL) < 40000000
#define MAM_TIMING   2               /* number of CCLK to read from the FLASH */
#else
#define MAM_TIMING   3               /* number of CCLK to read from the FLASH */
#endif
#define MAM_SETTING  2               /* 0=disabled,
                                        1=partly enabled (enabled for code prefetch,
                                          but not for data),
                                        2=fully enabled */

#define IRQ_HANDLER  1               /* 0 = Jump to common IRQ handler
                                        1 = Load vector directly from VIC, i.e.,
                                            LDR PC,[PC,#-0xFF0] */

/*initialize the exception vector mapping */
#define MAM_MAP      1                /* 1 = exception vectors are in FLASH
                                            at 0x0000 0000,
                                         2 = exception vectors are in SRAM
```

```
                                            at 0x4000 0000   */

/*
 * CHIP         SRAM SIZE     SRAM START ADDRESS
 * LPC2104       16 * 1024     0x40000000
 * LPC2105       32 * 1024     0x40000000
 * LPC2106       64 * 1024     0x40000000
 * LPC2114       16 * 1024     0x40000000
 * LPC2119       16 * 1024     0x40000000
 * LPC2124       16 * 1024     0x40000000
 * LPC2129       16 * 1024     0x40000000
 * LPC2194       16 * 1024     0x40000000
 * LPC2131        8 * 1024     0x40000000
 * LPC2132       16 * 1024     0x40000000
 * LPC2134       16 * 1024     0x40000000
 * LPC2136       32 * 1024     0x40000000
 * LPC2138       32 * 1024     0x40000000
 * LPC2210       16 * 1024     0x40000000
 * LPC2214       16 * 1024     0x40000000
 * LPC2220       64 * 1024     0x40000000
 * LPC2290       16 * 1024     0x40000000
 * LPC2292       16 * 1024     0x40000000
 * LPC2294       16 * 1024     0x40000000
 */
#define SRAM_SADDR   0x40000000              /* SRAM starting address */
#define SRAM_SIZE    (64 * 1024)             /* LPC2106 */
#define SRAM_TOP     (SRAM_SADDR+SRAM_SIZE)  /* SRAM end address + 1 */
#define SRAM_EADDR   (SRAM_SADDR+SRAM_SIZE-1) /* SRAM end address */

#define stackSize_SYS    600
#define stackSize_SVC     64
#define stackSize_UND     64
#define stackSize_ABT     64
#define stackSize_IRQ    600
#define stackSize_FIQ     64

#define STK_SIZE      (stackSize_SYS+stackSize_SVC+stackSize_UND+stackSize_ABT+
                       stackSize_IRQ+stackSize_FIQ)
#define STK_SADDR     (SRAM_EADDR+1-STK_SIZE)  /* Stack start address */



#define CONSOL_UART             0
#define CONSOL_BITRATE      115200

#define USE_NEWLIB           0   /* 0 = do not use newlib (= save about 22k FLASH),
                                    1 = use newlib = full implementation of printf(),
                                        scanf(), and malloc() */
#define CONSOLE_API_PRINTF   1   /* 0 = printf() = sendString,
                                    1 = simple, own implementation of printf() */
#define CONSOLE_API_SCANF    0   /* 0 = none,
                                    1 = simple, own implementation of scanf() */

#endif  /* _config_h_ */
```

Figure 17 – Board Support Package (BSP) Configuration File

There are three versions of the consol in order to best fit different situations:

- A very simple version that basically only supports printing strings (without any formatting parts) and printing numbers (decimal or hexadecimal).
- A simple printf() implementation that supports the simplest formatting tags. The implementation has been designed for lest possible stack usage (about 40 bytes).
- A full ANSI printf() implementation from newlib (part of the compiler environment that comes with GNUARM). This routine requires about 600 bytes of stack space and should normally not be used in resource constraint systems.

The code size for the first two alternatives is minimal (about 2k in program size for the entire framework). When using printf() from newlib, the code size is about 30 k for the entire framework (including a large part of the newlib library).

Just edit the configuration file above and recompile your project. The recursive nature of the makefiles will make sure that the startup library is recompiled and linked with the final executable program.

You can find an example project under the QuickStart Build Environment installation. See Figure 18 below for the path. It is typically:
`c:/program/InfraBed/evboards/LPC2xxx-gcc-newlib-vX_X_X_X`.
The beginning of the path can be specific for your installation and the ending of the path is specific for the version of the build environment. The figure below illustrates version 2_0_0_0.

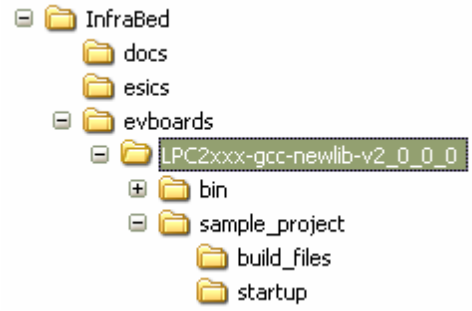


Figure 18 – Sample Project Files under QuickStart Build Environment Installation

The startup framework (BSP) is very simple and can best be understood by studying the source code files. If using the console functionality (printf()- and scanf()-like functions) observe that the function eaInit() must be called before printf() and the console can be used. The following code segment illustrates this.

```
#include <ea_init.h>
...
...
int main(void)
{
  eaInit();  //Now, the console/printf can be used
  ...
}
```

Also observe that whenever the BSP printf() should be used, the following include file must be included into the source code file.

```
#include <printf_P.h>
```

As a summary; Embedded Artists' *QuickStart Build Environment* is comprised of:

- A make build environment, controlled by bash script. A program or library build is started via the command: **`make`**.
- A program download feature, by using the LPC21ISP program. A program build and download is started via the command: **`make deploy`**.
- A Board Support Package (BSP) with startup code and console functions (i.e., printf() and scanf()-like functionality).

3.3.2 GCC

This will be very similar to the *QuickStart Build Environment* example, except that you will have to set up all paths manually and create your own startup files. The **`make`** files will also be a bit more complex. An example **`makefile`** is presented in *Figure 19* below. More complex examples than the **`makefile`** below also exist.

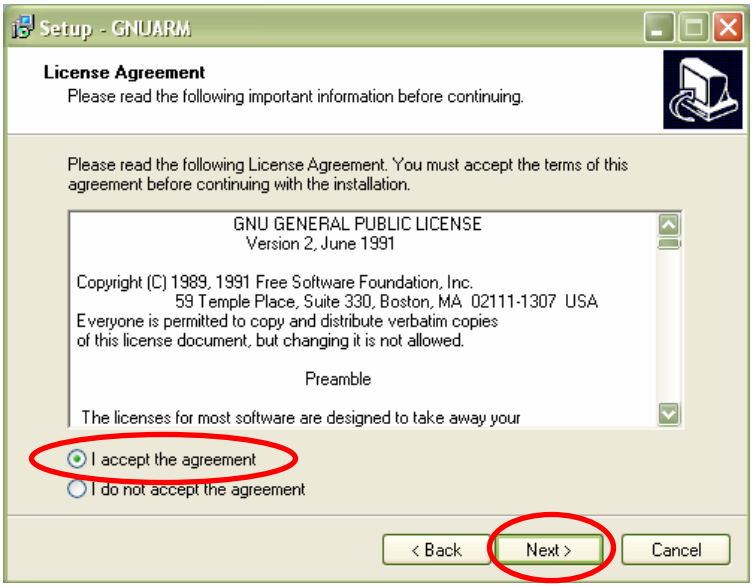
```
#
# Example makefile that creates a program called 'test', containing the
# C-source code files: main.c, eeprom.c, and i2c.c plus the assembler
# file startup.S
#

LIBS    =
DEBUG   = -g
CFLAGS  = -Wall -nostartfiles -mthumb-interwork -mthumb
INCLUDE = -Iinc/ -Iinc/specific/                #specify include paths here
ARMCC   = arm-elf-gcc
OBJS    = main.o eeprom.o i2c.o startup.o
LDFLAGS = -Wl,-Trom.ld                          #this file controls the linker

all: test.hex

test: $(OBJS)
        arm-elf-gcc $(CFLAGS) $(LDFLAGS) $(OBJS) $(LIBS) -o test.elf
%.o: %.c
        arm-elf-gcc -c $(INCLUDE) $(CFLAGS) $<
%.o: %.S
        arm-elf-gcc -c $(INCLUDE) $(CFLAGS) $<
%.o: %.c
        arm-elf-gcc -c $(INCLUDE) $(CFLAGS) $<
%.hex: %
        arm-elf-objcopy -O ihex $<.elf $@
clean:
        rm -f *.o test.elf test.hex
```

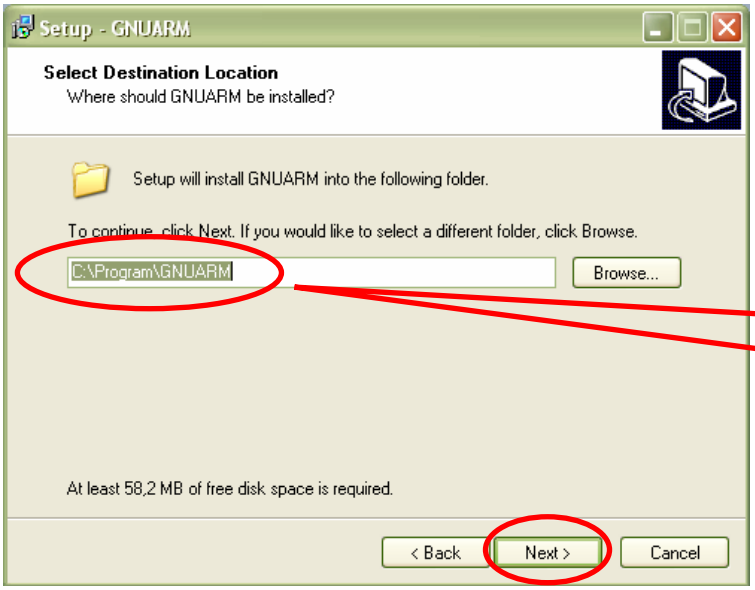
Figure 19 – Example GCC Makefile

3.4 Installing QuickStart Build Environment

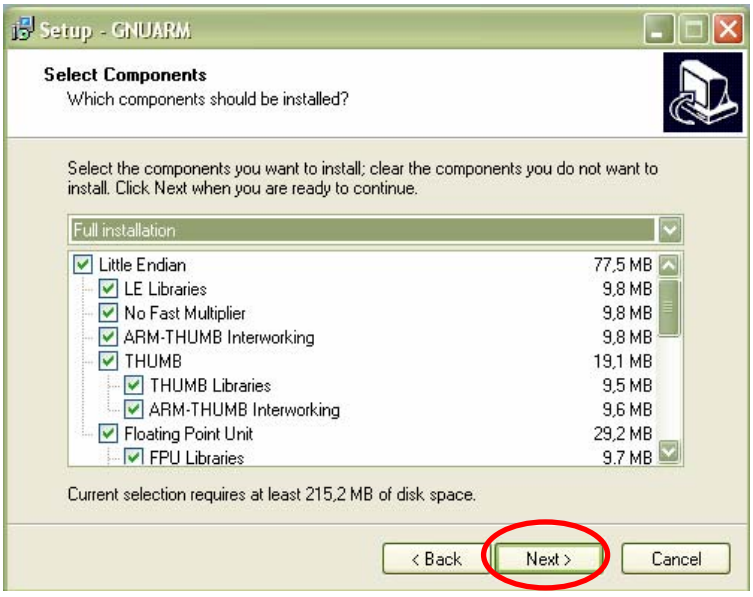
This section describes the necessary steps of program installation that is needed to get the QuickStart Build Environment ready for your use.

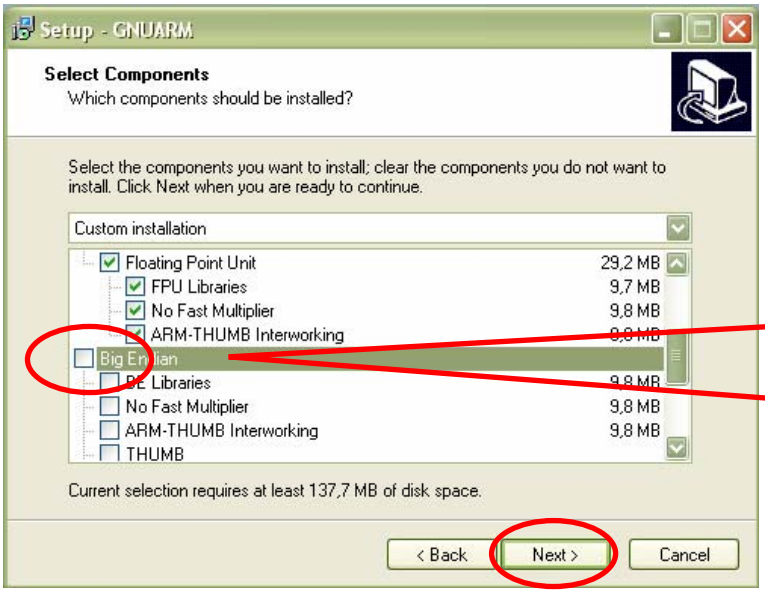
- Start with installing the GNUARM distribution that is included in the CD-ROM. The current version of the file is called: **`bu-2.15_gcc-3.4.3-c-c++-java_nl-1.12.0_gi-6.1.exe`**. There is also a newer, but less well tested, version (based on GCC v4.0.0). Only use this newer version if you are an experienced user.
  The installation is very simple and straightforward. It's just following the default installation steps as illustrated in the pictures below:



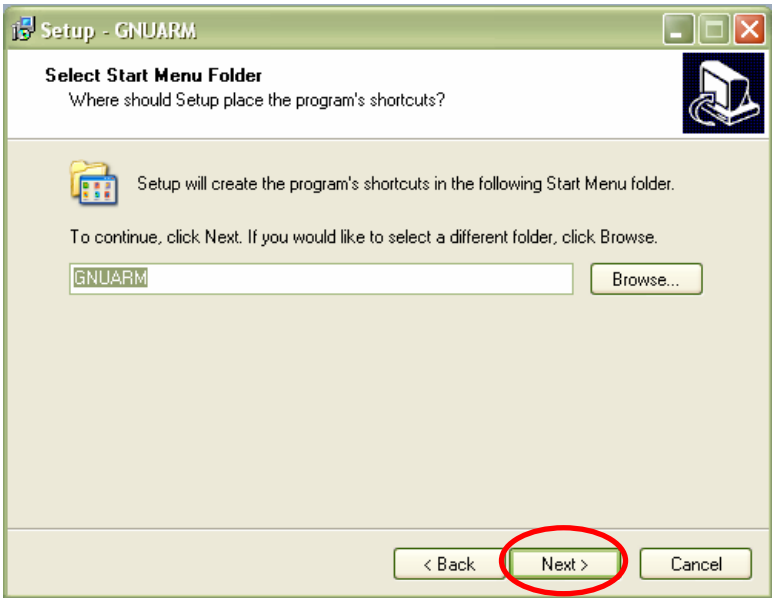


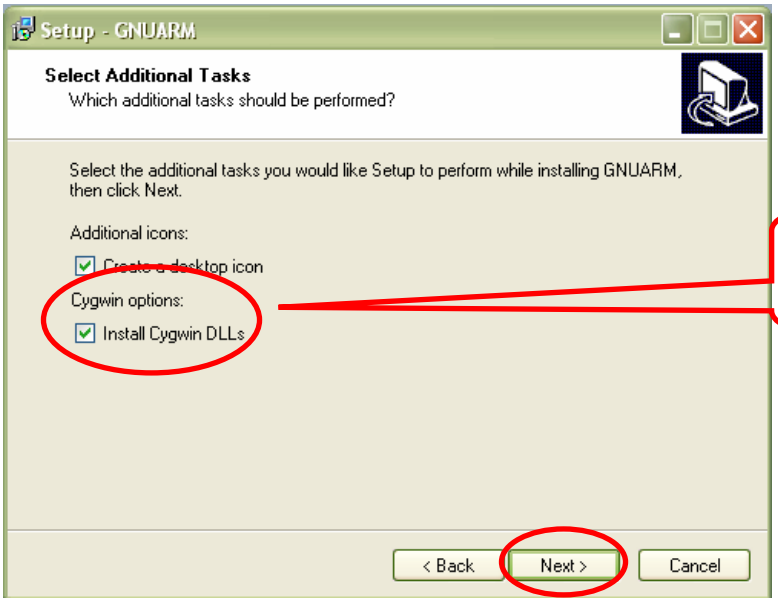
Use the default installation directory



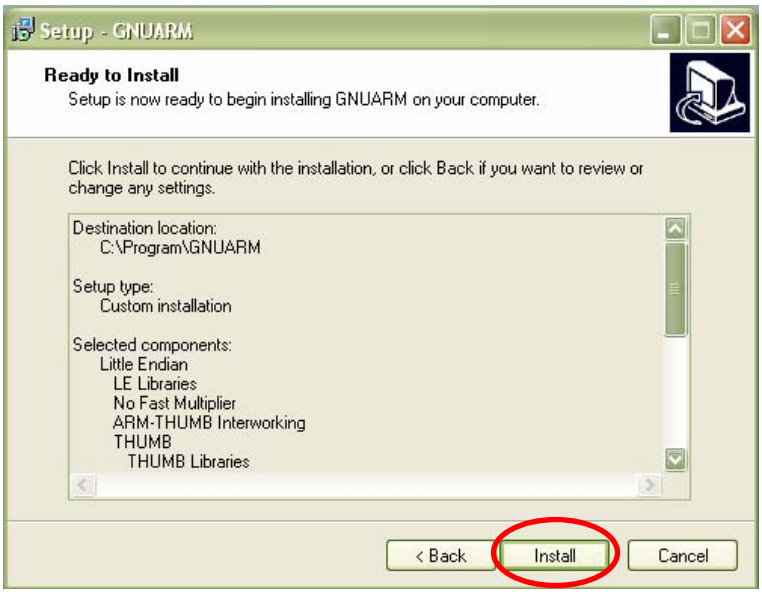


If you want to save space on your harddisk, you can deseclect the *Big Endian* component.

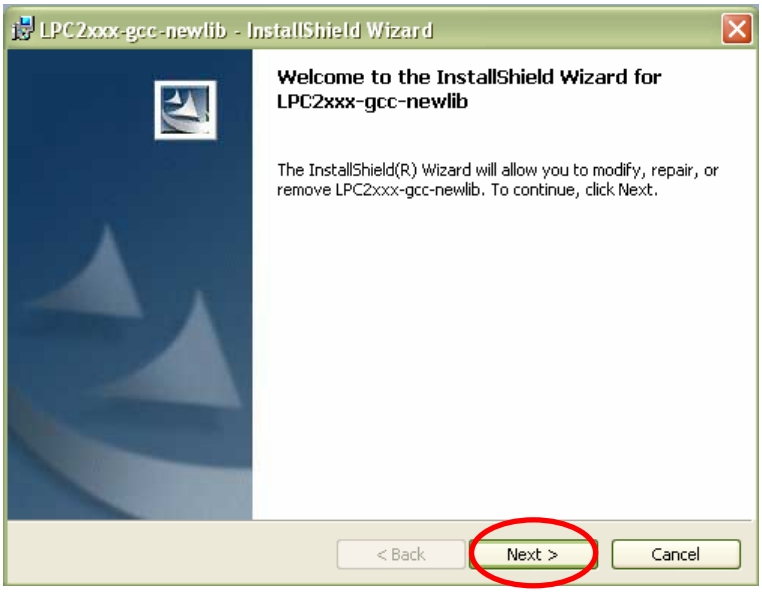


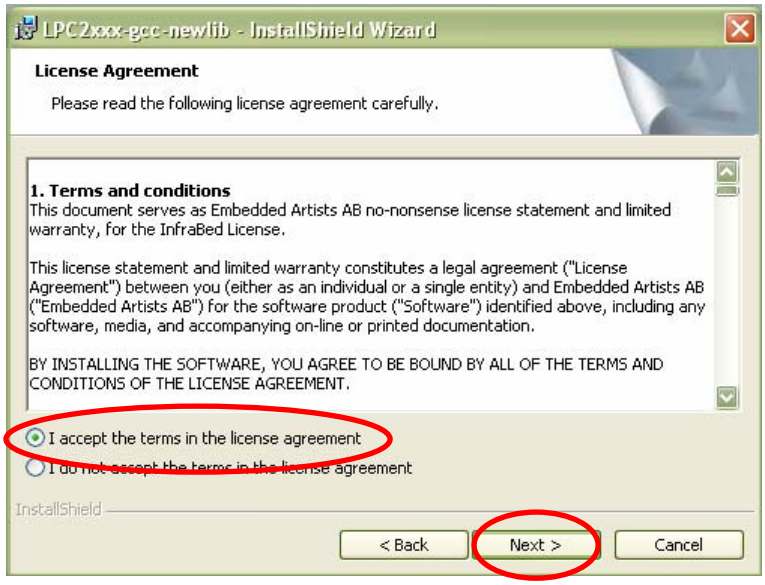


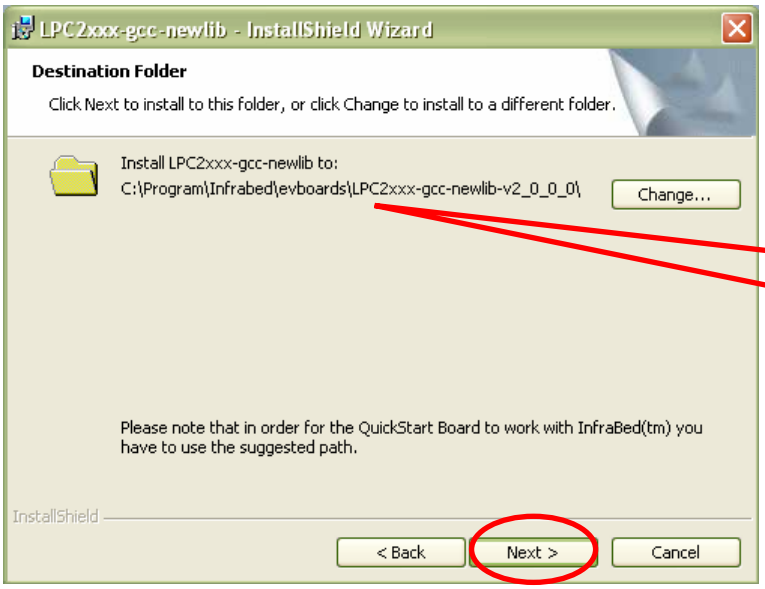
Install the Cygwin DLLs.



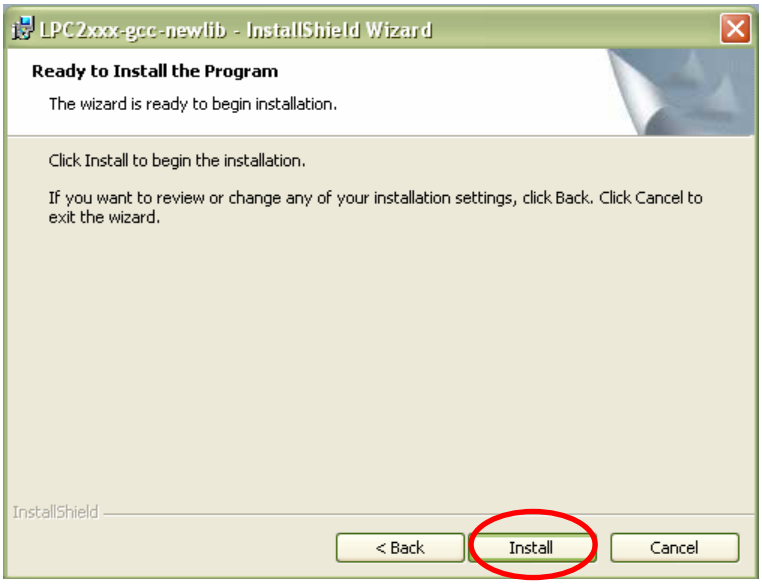
- Now install the **`LPC2xxx-gcc-newlib_vX_X_X_X`** *QuickStart Build Environment* (vX_X_X_X is the current version of the file). The installation is also in this case very simple and straightforward. Just follow the default installation steps.







Use the default installation directory



Observe that if the compiler is not installed on the default location (**c:/Program/GNUARM/**) the new path must be set in the files **build.sh** and **build_environment.sh**. Both files can be found in: **C:\Program\InfraBed\evboards\LPC2xxx-gcc-newlib-vX_X_X_X\bin**). It is the variable **COMPILERDIR2** that must be set (can be found on line 13 in both files). The compiler path must be to the **GNUARM/bin** directory.

Observe that the path above must contain the correct version number instead of …**vX_X_X_X\bin**. It may for example be: …**v2_1_0_0\bin**.

4 CD-ROM and Product Registration

The accompanying CD-ROM contains a lot of information and programs that will QuickStart your program development! Observe that there may be newer versions of different documents and programs available than the ones on the CD-ROM. See *Section 4.2* for information about the product registration process, which allows you to always have access to the latest versions.

4.1 CD-ROM

The following is included on the CD-ROM:

- The preloaded test program, in source code format and as a HEX-file.
- The two different ISP download programs.
- Datasheets of all circuits on the *LPC2106 QuickStart Board*.
- *QuickStart Build Environment* from Embedded Artists, which contains a complete setup of a build environment for GCC.
- A complete development environment: Rowley Associates CrossWorks for ARM, 30-day evaluation version.
- A complete development environment: IAR Embedded Workbench for ARM, Kickstart Edition with 32 Kbyte program size limit.
- Another complete development environment: GCC, GNUARM distribution, including compiler, linker, make, and debugger.
- The program Programmers Notepad, which is a very good program development editor and project manager.
- The Eclipse development environment including the CDT (C/C++ Development Tools) project.

4.2 Product Registration

By registering as a customer of Embedded Artists you will get access to more valuable material that will get you up-and-running instantly:

- Access to a Real-Time Operating System (RTOS), in the form of a library that can be used for non-commercial applications.
- Access to a number of sample applications that demonstrated different (peripheral) functions in the LPC2106 processor.
- Access to the latest versions of all information and programs on the CD-ROM.

Registering is easy and done quickly.

1) Go to http://www.EmbeddedArtists.com, select *Support* and then *Register*.
2) Type in the products serial number (can be found on the *LPC2106 QuickStart Board* or on the package carrying the board) along with your personal information.

5 Further Information

The LPC2106 microcontroller is a complex circuit and there exist a number of other documents with a lot more information. The following documents are recommended as a complement to this document.

[1] Philips LPC2106 Datasheet
http://www.semiconductors.philips.com/acrobat/datasheets/LPC2104_2105_2106-05.pdf

[2] Philips LPC2106 User's Manual
http://www.semiconductors.philips.com/acrobat/usermanuals/UM_LPC2106_2105_2104_1.pdf

[3] Philips LPC2106 Errata Sheet
http://www.semiconductors.philips.com/acrobat/erratasheets/2106.pdf

[4] ARM7TDMI Technical Reference Manual. Document identity: DDI0029G
http://www.arm.com/pdfs/DDI0029G_7TDMI_R3_trm.pdf

[5] ARM Architecture Reference Manual. Document identity: DDI0100E
Book, Second Edition, edited by David Seal, Addison-Wesley: ISBN 0-201-73719-1
Also available in PDF form on the ARM Technical Publications CD

[6] ARM System Developer's Guide – Designing and Optimizing System Software, by A.N. Sloss, D Symes, C. Wright. Elsevier: ISBN 1-55860-874-5

[7] Embedded System Design on a Shoestring, by Lewin Edwards.
Newnes: ISBN 0750676094.

[8] GNU Manuals
http://www.gnu.org/manual/

[9] GNU ARM tool chain for Cygwin
http://www.gnuarm.com

[10] An Introduction to the GNU Compiler and Linker, by Bill Gatliff
http://www.billgatliff.com

[11] LPC2000 Yahoo Group. A discussion forum dedicated entirely to the Philips LPC2xxx series of microcontrollers.
http://groups.yahoo.com/group/lpc2000/

[12] The Insider's Guide to the Philips ARM7-Based Microcontrollers, by Trevor Martin.
http://www.hitex.co.uk/arm/lpc2000book/index.html

Especially observe document [3]. There exist a number of bugs in the processor that is important to be aware of.

Observe that there can be newer versions of the documents than the ones linked to here. Always check for the latest information / version.

Datasheets for all circuits on the *LPC2106 QuickStart Board* are included on the accompanying CD-ROM.