Nohau by ICE Technology

Evaluation Board

For NXP - Philips LPC2106

Contents

About This Guide

EMUL–ARM is a PC-based hardware debugger for the ARM™ Core (currently ARM7 and ARM9 cores). Seehau is the name of the user interface of EMUL-ARM – Seehau and EMUL-ARM is often used interchangeably.

This User's Guide helps you to understand how to use the Philips LPC210X series in general and the Nohau LPC2106 TESTER board with:

- EMUL-ARM
- HI-TECH ARM-C and HI-TIDE
- IAR EWARM C/C++

Note that the LPC210X series does not have an external data and address bus – which means that most of the information in this document is relevant for any LPC210X based target.

Note that there might exist two versions of this document (and other BSP/Target Board Related documents):

- If it is located in the "SeehauARM\Documents" install directory then it relates to currently installed Seehau.
- If it is located in the "C:\Nohau\BSP_ARM\Doc" directory then it relates to the currently installed BSP.

Please note that it is required to install BSPs to "C:\Nohau\BSP_ARM" because compiler project files are often dependent on file location.

1 INTRODUCTION

Important Notes

The target board comes with a manual – please read it. This document is intended to complement the board documentation – not replace it.

Following web site has additional information about the Philips LPC2100 series. It offers a preliminary user's manual for download.

- http://www.lpc2100.com

It is assumed and required that the BSP is installed to c:\Nohau\BSP_ARM.

Memory Configuration

The LPC2106 have internal memory only:

SRAM:	64 KB.	Address 4000_0000 to 4000_7FFF.
Flash:	512 KB.	Address 0 to 1DFFF

The LCP2105 has 32 KB of SRAM, and the LPC2104 has 16 KB of SRAM.

Remap the internal RAM

The exception vectors – addresses 0-0x3F can be remapped, which allows:

- Makes it easy for software to modify exception vectors. I.e. it is not required to link the addresses of the handlers into the application.
- Debug in RAM. It is also possible to debug in a mix of RAM and Flash, for instance interrupt vectors in flash and the rest in RAM.

When the exception vectors are remapped, there is RAM at addresses 0-0x3F. The RAM is mirrored from address 4000_0000, which means that writing to address zero gives the same effect as writing to address 4000_0000.

Seehau supports this remapping, and it is enabled by default. Change by menu:

- Config | Emulator – then use the Misc Tab.

Flash Programming with EMUL-ARM

Seehau supports flash programming directly. However it is disabled by default. When flash programming is enabled, normal “load” causes flash to erased and written as needed. To enable, use menu:

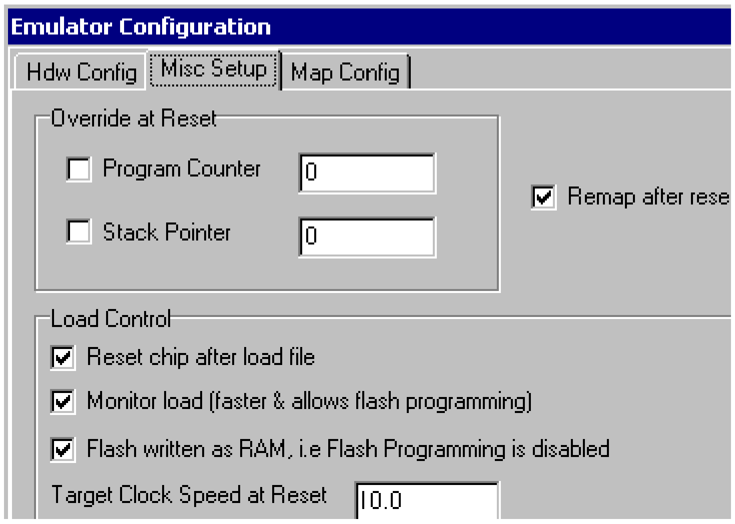
- Config | Emulator – then use the Misc Tab, and uncheck “Flash Written as RAM”.

Remapping cannot be active when flash programming is active. Currently, it is required manually configure for hardware breakpoints when debugging in Flash by menu:

- Run | Force Hardware Step.

For flash programming to work correctly, the driver needs to know the target board crystal frequency. It is available on the emulator configuration, the Misc Tab as show below. Enter value in MHz in the “Target Clock Speed at Reset” – it is possible to use decimal points such as “3.2”.

For the Nohau LPC2106 TESTER (10 MHz), use “10.0”.



- [ ] Program Counter: 0
- [ ] Stack Pointer: 0
- [✓] Remap after rese
- [✓] Reset chip after load file
- [✓] Monitor load (faster & allows flash programming)
- [✓] Flash written as RAM, i.e Flash Programming is disabled

Target Clock Speed at Reset: 10.0

Board Configuration

For the Nohau LPC2106 TESTER board, there is very little board configuration to do for basic operation. Only one jumper is of importance – JP1, which is required to allow debugging.

Other jumper / options are there to allow prototyping. See LPC2106_TESTER.pdf for further description of those.

Other Documents

Please also see following documents:

- **ARM_BSP.pdf** – overview of general source code that is supplied with EMUL-ARM for Philips LPC210X – see ARM_BSP.pdf. This document discusses Timer, different Target Console applications and uC/OS-II.
- **ARM_Compilers.pdf** – how to set up different compilers to be used with EMUL-ARM.
- **Nohau_Monitor .pdf –** using the Nohau Monitor to get printf() from target application in Seehau etc.
- **LPC2106_TESTER.pdf** – documentation of the Nohau LPC2106 TESTER board.

Target Board CD

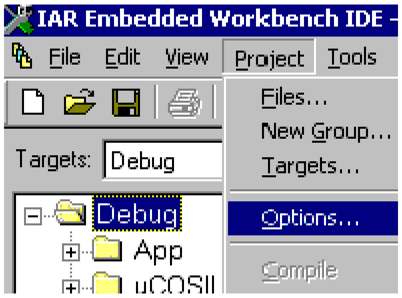
The Nohau LPC2106 TESTER board comes with a CD with documentation and schematics.

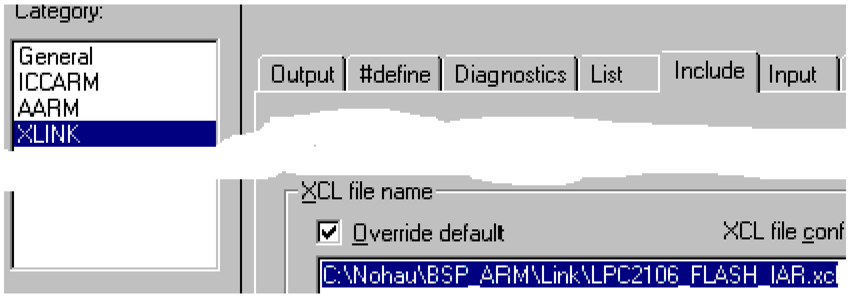
Supported Compilers

All examples can be compiled with HI-TECH ARM-C and IAR C/C++, unless otherwise explicitly stated. See description for each example. There are ready to use project files in each target application directory. Almost all examples are setup to be compiled and linked for the internal SRAM memory region. The sections below will help you to change the projects to use internal Flash instead.

IAR Compiler – Generate Code for Flash

With the IAR compiler, memory regions to be used is controlled in the linker control file – extension *.xcl. See additional information below. To change linkage, open the IAR project file (*.pew), select menu “Project|Options”, and in the popup, select XLINK in the left pane, and set the XCL file name as shown in the pictures:



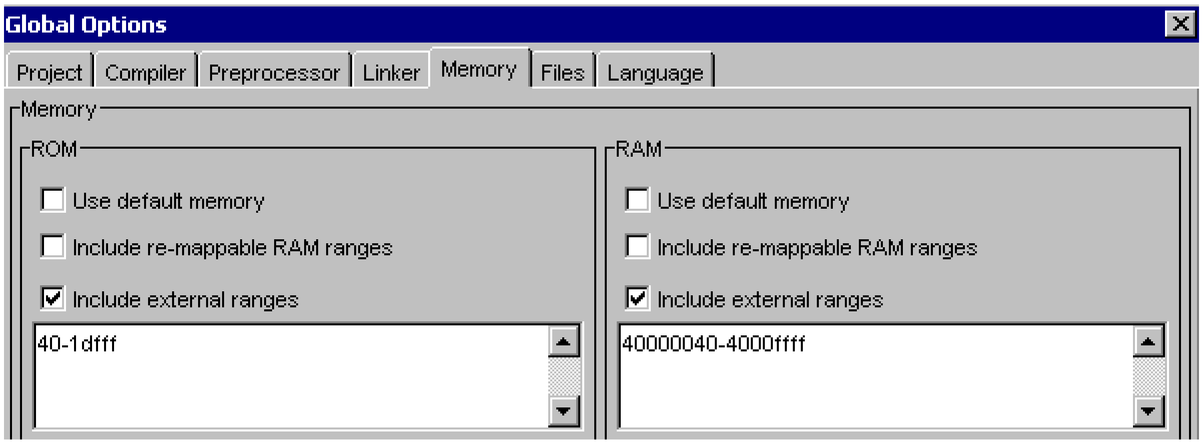


☑ Override default

C:\Nohau\BSP_ARM\Link\LPC2106_FLASH_IAR.xcl

HI-TECH Compiler – Generate Code for Flash

The HI-TECH ARM-C compiler does not use a link control file to define memory regions to be used. Instead, you have to modify the memory regions manually. In HI-TIDE, select menu “Project|Global Options”, then go to the Memory tab.



ROM:
☐ Use default memory
☐ Include re-mappable RAM ranges
☑ Include external ranges
40-1dfff

RAM:
☐ Use default memory
☐ Include re-mappable RAM ranges
☑ Include external ranges
40000040-4000ffff

2 DEVELOPING SOFTWARE

Include Files

The compiler will generally have #include files for MCU registers. If yours doesn't have it, you can use following file (where the initial N is used to denote Nohau and avoiding name conflicts):

- c:\Nohau\BSP_ARM\Src\Inc\ **NLPC210X.h**

All examples in Nohau BSP ARM are based on a couple of .h files. These are:

- c:\Nohau\BSP_ARM\Src\Inc\ **NDefs.h**
- c:\Nohau\BSP_ARM\Src\Inc\ **NArm.h**
- **NConfig.h** (located in the application directory). "NDefs.h" includes this file.

The Nohau Monitor uses an additional .h file in the application directory:

- **NMon_Cfg.h** – Nohau Monitor.

Startup Code

HI-TECH ARM-C does require startup code or special vector definitions. However, interrupt driven examples from Nohau use the following files:

- C:\Nohau\BSP_ARM\Boards\Philips\LPC210X\Startup \ **vectors_HTC_LPC210x.as**
- C:\Nohau\BSP_ARM\Boards\Philips\LPC210X\Startup \ **stacks_HTC.s**

For IAR C/C++, an example c startup code is given in:

- C:\Nohau\BSP_ARM\Boards\Philips\LPC210X\Startup\ **cstartup_IAR_ LPC210x.s**

Memory Map / Linking

With the LPC2106, there are two main options – debug in Flash or RAM. When possible, Nohau recommend debugging in RAM as long as it is big enough because debugging in RAM allows unlimited amount of breakpoints.

With the HI-TECH compiler, you will have to configure memory map manually. For the LPC2106 we recommend as initial values using menu "Project|Global Options" – memory tab:

- RAM based debug: ROM= 40000040-40007FFF, RAM= 40008000- 4000FFFF
- Flash based debug: ROM=40-1DFFF, RAM= 40000040- 4000FFFF

With the IAR compiler, this is reflected in the link control maps as supplied in Nohau BSP ARM (there are also similar flash based files for the 2104 and 21056):

- C:\Nohau\BSP_ARM\Boards\Philips\LPC210X\Link\LPC2106_FLASH_IAR.xcl
- C:\Nohau\BSP_ARM\Boards\Philips\LPC210X\Link\LPC2106_RAM_IAR.xcl

Please note that you may have to modify the xcl files to change the size of the stack and heap.

3 SUPPLIED SOURCE CODE

General Examples

These are applications that are shared between most supported boards. See ARM_BSP.pdf for additional information.

The application categories are available:

- **TargetConsole** - Shows the capabilities of the EMUL-ARM debugger-target communication.
- **Interrupt** – Shows how to do basic interrupts with and without the debugger target-communication.
- **uCOSII** – Shows the uC/OS-II RTOS.

µC/OS-II

- No Thumb mode examples with the HI-TECH compiler (which does not yet support Thumb).
- The "Full" application is not yet tested.

MCU / Board specific examples

MCU/Board specific examples are located in the "LPC2106\Apps\Hw" directory. Following examples are currently available:

- **IAP** – demonstrates the IAP driver. Not yet available for the HI-TECH compiler (because it does not support Thumb mode).

Drivers

Drivers are located in the "LPC2106\Drv" directory. Following drivers are currently available:

- **IAP** – In Application flash programming.

4 MACROS

PLL_Set.bas

The Nohau LPC2106 TESTER board starts at 10MHz after reset. (The clock frequency must be in the range of 10 MHz to 25 MHz.)

The registers PLLCON and PLLCFG control the frequency.

The PLL_Set.bas macro sets PLLCON=3. With PLLCON=3 and 10MHz clock, following PLLCFG values gives frequencies:

PLLCFG = 1 → 20MHz
PLLCFG = 2 → 30 MHz
PLLCFG = 3 → 40 MHz
PLLCFG = 4 → 50 MHz
PLLCFG = 5 → 60 MHz

System Reset clears the effect of setting the PLL.

Running the macro copies a small program into address 0x40001000. When desired, set the PC to that address, then set PLLCFG and finally – GO and Stop – and the frequency is updated.

Note that whatever that was loaded at that address was overwritten.

5 OTHER TARGET BOARDS

This document relates to the Nohau LPC2106 Tester Board. But most of the information in this document is relevant for any LPC210X based implementation – especially since the LPC210X series does not have an external data and address bus.

The section(s) below are intended to give some pointers when using other boards.

Ashling EVB-A7

Board Configuration

To allow debug with EMUL-ARM:

- Remove JP12 (Enable On Board Debug).

IAR LPC2106

Default settings can be used.