

Compiler for ARM, ARM Programming Tutorials, ARM Guide, ARM Projects, ARM Led Blinking Program, ARM Debugger, How to use Jlink for ARM, IAR Workbench Guide

IAR workbench is very nice development tools for ARM. You can download their kickstart version from their website. Kickstart version is free to download and it comes with restriction of 32KB code size limit. Once you download then you can easily install it. This article describe project creating and debugging.



Before going through in details of working of IAR workbench, let me explain basics of ARM application project. ARM application can be of two types:

1.ROM Run Application 2.RAM Run Application

ROM Run Application

Rom run application will be downloaded to flash memory and it will start executing from there only. The executable file in this case will be pure binary file i.e. this will not contain debugging information. Extension of this file can be either bin or hex. (User can configure project options to generate any type of file). In IAR workbench there is predefined setting of project options for generating output file suitable for ROM run application, name of this configuration is Release configuration.

In Release configuration code is optimized in order to generate smaller and fast executable output file. And Rom version of linker script file will be used. In this file text section of code is directed to load into ROM memory.

RAM Run Application

Ram run application will be downloaded to ram memory and it will start executing from there only. The executable file in this case will contain debugging information. So that debugging will be easy. Purpose of type of application is only debugging and this will be not be finally released file. These files are of ELF format. (Executable and Linkable Format)

In general these files do not have any extension. In IAR workbench there is predefined setting of project options for generating elf output file le for RAM run application, name of this configuration is Debug configuration. And Ram version of linker script file will be used while building the project. In this file text section of code is directed to load into RAM memory. Downloading of such a file need JTAG debugger or bootloader program.

We have created a sample project for blinking of LED. This application is written for Olimex LPC2148 Board. In this board two LEDs are connected at P0.10 and P0.11. You can easily modify code according to your LPC2148 board. This code should work for any LPC2xxx processor.

You can download from following location Click Here todo give the location of code.

Creating Project

After installing IAR workbench open IAR workbench. For that goto Start > All Programs > IAR Systems > IAR Embedded Workbench for ARM 5.30 Kickstart > IAR Embedded Workbench

🔏 IAR Embedded Work	whench IDE		
<u> Eile E</u> dit <u>V</u> iew Project]	<u>Tools W</u> indow <u>H</u> elp		
🗅 🚅 🖬 🕼 🐰	8 略 @ の cv		🚳 🚳 🛤 🕅 🥭 ⊿
Workspace	x		
Files			
Ready		12	

Go to Project > Create New Project

Create New Pr	oject		X
<u>T</u> ool chain:	ARM	•	·
Project templates	:		
Empty project ⊕ asm	ət		
← C++ ← C ← main ← Externally by	iilt executable		
Description:			
C project using o	efault tool settings incl	uding an empty main.	c file.
		ОК	Cancel

Here we will choose C main project and click Ok. After that you will see option to save project file. My preferred location is D:\lpc2148\ledblink and project name I have given ledblink. IAR project file extension name is ewp. so my project file is ledblink.ewp.

🔏 IAR Embedded Workbench IDE		
Eile Edit View Project Simulator Tools	<u>W</u> indow <u>H</u> elp	
0 🗳 🖬 🕼 🥌 👗 🖻 🛍 🗠	C*	💽 🗸 🏷 🍡 🖾 🚳 🐡 🦛 🏤 🔛 🛤 😕 🕗
Workspace	main.c	▼ ×
Debug		
Files 8	int main()	-
🗆 🗊 ledblink - Debug 🗸 🗸	return 0:	
Hanc	}	
ledblink	fo 📢	
Ready		Ln 6, Col 1

After saving project let us save workspace. workspace is a group of projects. For example, in case of lpc2148 board I can create more then one projects, so my all these project should goes under lpc2148 workspace. So its up to user how he manage project files.

I have selected workspace name as Ipc2148work and its location will be in d:\lpc2148. For saving project workspace goto File > Save Workspace

and give lpc2148work name to workspace file.

Save Workspac	e As					? 🔀
Save jn:	Dc2148		•	(- E		
My Recent Documents	iedblink 🔁					
Desktop						
My Documents						
My Computer						
S		Internet internet				
My Network Places	File <u>n</u> ame:	Ipc2148work			<u> </u>	Save
	Save as type:	Workspace Files (*.eww)			<u> </u>	Cancel

Extension to workspace file is eww. now click on save button and save it. Now let us add source code in main file. We are writing code for Led blinking.

#include "iolpc2148.h" #define LED0 0x0400 #define LED1 0x0800 #include "main.h"

int main()
{
 /**** Initialization ****/
 PLLInit();
 SCS =0x3;
 /* make led line as output*/
 FIOODIR |= LED0 | LED1;
 FIOOSET = LED0 | LED1;
 while(1)
 {
 /* turn off LED*/
 FIOOSET = LED0 | LED1 ;
 delay();
 /* turn on LED*/

FIO0CLR = LED0 LED1 ; delay();
}
keep main.h and iolpc2148.h files in your project directory. There are two configuration for executable file, ram run or ram run as discussed in first paragraph of this article. Let us discuss building of RAM Run application first. For RAM run application we have to add LPC2148_RAM.icf linker script file to the project.
Click here for Part2 of the article.
Part 1 Part 2 Part 3
Receive Regular Updates Want to receive regular updated of Tutorials published on this site, Register for free subscription of cost Infoletter Click Here for registration.

 Best viewed in 1024x768 pixels
 [Home] [Query] [FAQ] [Aboutus] [Contact us] [Sitemap] [Privacy Policy] [Advertise]
 RSS FEED

 Logos and brand names used in this site are belonging to their respected owners. We have used them here only for the purpose of information. Enable Active X control from internet options of internet explorer to view all element of this site.

You 📷 🚹 🔄 🔜 GoogleGroups InfoLetter

Files

Embedded SITE SEARCH mCard EMBEDDED SYSTEM, EMBEDDED LINUX, 8051, ARM PROCESSOR, ECLIPSE IDE, RTOS, STEPPER MOTOR, muc MICROPROCESSOR PRESENTATIONS EMBEDDED LINUX NEWS TUTORIALS/ARTICLES CODE BANK QUERY ABOUT US FAQ INFOLETTER mCARD because information matters Tutorials | Infoletters | News | Search Part 1 | Part 2 | Part 3 Embedded ARM Development by IAR workbench (2) Step by step Last Edited: APR-2010 Compiler for ARM, ARM Programming Tutorials, ARM Guide, ARM Projects, ARM Led Blinking Program, ARM Debugger, How to use Jlink for ARM, IAR Workbench Guide For adding linker script file right click on project options and choose options as shown in 🔏 IAR Embedded Workt above picture. or choose Project > options or use (ALT + F7). File Edit View Project T D 🛩 🖬 🖪 🖨 /ork Debug

🔏 IAR Embedded Workbench IDE	
File Edit View Project Simulator Tools W	indow Help
D 📽 🖬 🕼 🕼 🖁 🛍 🗠 🗠	- 📃 🗸 🦙 🐂 🔄 📼 🐡 🍓 🌬 🛤 👷 🏓 🥭
Workspace	x main.c io_macros.h
Debug Files	<pre> /************************************</pre>
ledblink	ce Code Control >
× Hassages	roperties
Set a	s Active
4 file(s) deleted. Updating build tree main.c Linking Build Debug Log	×
Edit options for the selected item	Errors 0, Warnings 0

After that options for node "ledblink" window will open. Here goto Linker > Config > Linker configuration file override default and give path of LPC2148_RAM.icf file. For specifying path you can use IAR global variable \$PROJ_DIR\$ to specify path of project directory. So according to that complete path will be following

PROJ_DIR\$\LPC2148_RAM.icf

Category:	Factory Setting:
Seneral Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Driver	Config Library Input Output List #define Diagnostics Ch. Linker configuration file Coverride default STODLKIT_DIR\$\CONFIG\generic.icf Edit Configuration file symbol definitions: (one per line)

after setting of linker script file this window should be like following:

ategory:		Factory Settings
eneral Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Driver	Config Library Input Output List #define Linker configuration file Image: Configuration file Image: Configuration file Image: Configuration file SPR0J_DIR\$\LPC2148_RAM.icf Image: Configuration file symbol definitions: (one per line) Image: Configuration file symbol definitions: (one per line)	Diagnostics Ch.
		1231

After that we have to choose project options which are following:

Processor options Project > options > General options > Target > Processor variant > Device click on button next to Device text box

seneral Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Driver	Target Output Library Processor variant C Cgre ARM7T C Device None Endian mode C Little Big C BE32 C BE8	Configuration Library DMI EPUNone	Options MISRA-C:200	
---	--	-------------------------------------	-----------------------	--

and choose NXP > NXP LPC2148.

Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace RDI ST-Link C Lore IARM/1DMFS Device NXP LPC2148 Endian mode Endian mode Endian Mode Endian State Big ST-Link C Big ST-Link C BE32	
---	--

Now choose j-Link as a debugger. For that go to Project > options > Debugger > Setup > Driver select J-Link/J-Trace from drop down menu. Left all options as it is. and click on OK.

eneral Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator	Setup Download Extra Options Plugins Driver Run to	Factory Settings
Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Driver	Use macro file(s) Devige description file Qverride default (\$TOOLKIT_DIR\$\CONFIG\debugger\NXP\iolpc2148.ddf	

now project is ready to build.

To build project pres F7.

and Start debugger. For that go to Project > Download and Debug or press (Ctrl + D). Code will start downloading on target. Once downloading is completed then you can see debugger view of IAR workbench. Now you can see that code is stopped at main.

🔏 IAR Embedded Workbench II	DE		
<u>File Edit View Project Debug</u> D	isassembly <u>J</u> -Link <u>T</u> ools <u>W</u> indow <u>H</u> elp		
0 🛩 🖬 🕼 👹 🖓 🐚 😭	Ina - √ * *	🔄 🕟	🐢 🍓 🌬 🛤 🐏 🖬 🗶 🟚 g
5 - 326332	* ×		
Workspace 🗙	main.c in macros.b Imain.b	• x	Disassembly
Debug	finclude "iolnc2148 h"		Go to 🗾 🗸 M
P1- 2* D3	#define LEDO 0x0400	-	Next label is a Thumb I
	#define LED1 0x0800		main
🛛 🗖 ledblink 🗸 👘	#include "main.h"		main.
H ⊢⊞ ∰ main.c			40000134 B580 F
Gutput	shint main()		PLLInit();
			40000136 F7FF ;
	/**** Tritialization ****/		40000138 FF85 F
	PLLInit():		4000013A 480D I
	SCS =0x3.		4000013C 2103 M
	/f make led line as output f/		4000013E 6001 9
	FIGDDIR I= LEDO LED1.		40000140 480C T
	FIGSET = LEDG LED1;		40000142 6800 I
	riddsei - hebo hebi,		40000144 21C0 N
	willie (1)		40000146 0109 I
	No. 1 Contraction		40000148 4301 C
	/* turn off LED*/		4000014C 6001 S
	FIOUSET = LEDU LED1 ;		FIO0SET = LED0 LEDJ
	delay();	-1	4000014E 480A I
ledblink	f0 4 /* turn on T.FD*/		
×			
T + 1010 4017 0010			
Tue Apr 13 12:48:17 2010:	JLINK command: ProjectHile = D:\/jpc2148\/ledblink\setting	gs∖ledb	link_Debug.jlink, return = 0
Tue Apr 13 12:48:17 2010:	JLINK command: device = LPC2148, return = 0		
Debug Log Build			
Ready		1	Ln 6, Col 1

Click here for Part3 of the article.

Part 1 | Part 2 | Part 3

Receive Regular Updates

Want to receive regular updated of Tutorials published on this site, Register for free subscription of cost Infoletter Click Here for registration.

www.embeddedcraft.org/iar_arm1.ht...

тор

S RSS FEED

[Home] [Query] [FAQ] [Aboutus] [Contact us] [Sitemap] [Privacy Policy] [Advertise] Best viewed in 1024x768 pixels Logos and brand names used in this site are belonging to their respected ow ners. We have used them here only for the purpose of information. Enable Active X control from internet options of internet explorer to view all element of this site.

Embedded ARM Programming | IAR W...

Yw 🌆 🚹 🔄 🔝 GoogleGroups InfoLetter

Embedded crafting of intellig	Craft gent systems								mCard	SITE	SEARCH
	EMBEDDED SYSTEM	, EMBEDDED LINUX, 805	1, ARM PROCESSOR,	, ECLIPSE IDE, F	RTOS, STEPPER	MOTOR, muc	h more W	ELCOME	To The World o	f Know le	edge
MICROPROCESSOR	PRESENTATION	S EMBEDDED LINUX	NEWS TUTORIALS	S/ARTICLES	CODE BANK	QUERY A	BOUT US	FAQ I	NFOLETTER	mCA	RD
because information	n matters						Tuto	orials	Infoletters	News	Search
	E	mbedded ARM De	velopment by I/	AR workbei	n ch (3)			P	Part 1 Pai	rt 2	Part 3
	Ste Las Coi Gu	p by step t Edited: APR-2010 npiler for ARM, ARM Prograr de	nming Tutorials, ARM Gu	uide, ARM Projects	s, ARM Led Blink	ing Program, Al	RM Debugger,	How to u	ise Jlink for AR	M, IAR W	/orkbench
	W	e can do single stepping	I. Some of the usefu	ul commands a	are following:					nbedde	ed Workt
	Ste	ep over F10							File Edit	View	Project T
	St	p into F11									
	St	p out Shift + F11							Workspace		1998 60
	Pr	ess F5 and you can see	LED must be blinki	ing on Board.					Debug Files		

ROM run program or Flash run program

For making program to execute from run IAR has Release configuration. For that choose Release from drop down button below the Workspace window.

🔏 IAR Embedded Workbench IDE	
Eile Edit View Project Simulator Tools Window Help	
	🗹 🖌 🌾 注 🖻 🗭 🦓 🛤 🛤 😣 🕭 .
Workspace × Release × Files \$	<pre>main.c io_macros.h main.h PLLInit(); SCS =0x3; /* make led line as output*/ FIOODIR = LED0 LED1; FIOOSET = LED0 LED1; while(1) (</pre>
ledblink	<pre>/* turn on LED*/ FIOOCLR = LED0 LED1 ; delay(); } fo] [4]</pre>
× Messages	
Building configuration: ledblink - Debug Updating build tree Configuration is up-to-date.	
v <	
Build Debug Log	
Ready	Errors 0, Warnings 0

For release configuration we have to make following changes

1.Optimization

choose none optimization. Project > options > C/C++ Compiler > Optimization > Level

2.Linker script file for flash Project > options > Linker > Config > Linker configuration file choose LPC2148_flash.icf file \$PROJ_DIR\$\LPC2148_flash.icf

3.Generate HEX file

choose none

Project > options > Output Converter > output check on Generate additional output choose intel extended

General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace LMI FTDI Macraigor RDI ST-Link Third-Party Driver	Output Image: Constraint of the second output Output format: Intel extended Qutput file Oyerride default Iedblink.hex
---	---

now build the project and you can see that hex file is generated in Release folder of project directory.

This file can be downloaded into flash memory by flash magic software.

Part 1 | Part 2 | Part 3

Receive Regular Updates Want to receive regular updated of Tutorials published on this site, Register for free subscription of cost Infoletter Click Here for registration.

Best viewed in 1024x768 pixels

[Home] [Query] [FAQ] [Aboutus] [Contact us] [Sitemap] [Privacy Policy] [Advertise]

🔝 RSS FEED

TOP

Logos and brand names used in this site are belonging to their respected ow ners. We have used them here only for the purpose of information. Enable Active X control from internet options of internet explorer to view all element of this site.