

CC1110/CC2430/CC2510

IAR IDE User Manual Rev. 1.2







Table of contents

1	INTRODUCTION	3
2	DEFINITIONS	3
3	INSTALLATION	.4
3.1 3.1.1 3.1.2 3.2 3.2.1 3.2.2	CHIPCON CC1110/ CC2510 EXTENSIONS	4 4 5 5 5
4	CC1110/ CC2510 LINKER FILE	6
4.1 4.2 <i>4.2.1</i>	XDATA RAM CODE Create HEX-file	6 7 7
5	CC2430 LINKER FILE	.8
5.1 5.2 5.2.1 5.2.2 5.2.3	XDATA RAM CODE Near code model Banked code model Create HEX-file	8 9 10 11 12
-	P P P V G G P P	1.4
6	DEBUGGER	14
6 7	ABSOLUTE PLACEMENT OF CODE AND VARIABLES	14 15
6 7 7.1	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE	14 15 15
6 7 7.1 7.2	DEBUGGER	14 15 15 15
6 7 7.1 7.2 8	DEBUGGER	 14 15 15 15 16
6 7 7.1 7.2 8 8.1	DEBUGGER	 14 15 15 15 16
6 7 7.1 7.2 8 8.1 8.2	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE	14 15 15 15 16 16
6 7 7.1 7.2 8 8.1 8.2 9	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES	 14 15 15 16 16 16 18
6 7 7.1 7.2 8 8.1 8.2 9 9.1	DEBUGGER	14 15 15 16 16 16 18 18
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE VARIABLES CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS Comment Ontions Contained Ontions Comment Ontions	14 15 15 16 16 16 18 18 19 20
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2 9.2 9.2	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES	14 15 15 16 16 16 16 18 18 19 <i>20</i> <i>23</i>
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2 9.2 9.2 9.2 9.2 9.2 3	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE. VARIABLES. CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS General Options, Code Model Near (Applies also to CCxx10) General Options, Code Model Banked (Does not apply to CCxx10) Linker	14 15 15 15 15 16 16 18 19 20 23 24
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE VARIABLES CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS General Options, Code Model Near (Applies also to CCxx10) General Options, Code Model Banked (Does not apply to CCxx10) Linker Debugger Word Decement	14 15 15 15 15 16 16 18 19 20 23 24 25
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2.1 9.2.2 9.2.3 9.2.4 9.3 9.4	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE VARIABLES CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS General Options, Code Model Near (Applies also to CCxx10) General Options, Code Model Banked (Does not apply to CCxx10) Linker Debugger INCLUDE SOURCE FILES COMPUTE AND LINK	14 15 15 15 15 16 16 16 18 19 20 23 24 25 26 27
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2.1 9.2.2 9.2.3 9.2.4 9.3 9.4 9.5	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE VARIABLES. CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS General Options, Code Model Near (Applies also to CCxx10) General Options, Code Model Banked (Does not apply to CCxx10) Linker Debugger. INCLUDE SOURCE FILES COMPILE AND LINK DEBUG	14 15 15 15 16 16 18 20 23 24 25 26 27 28
6 7 7.1 7.2 8 8.1 8.2 9 9.1 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2 9.2	DEBUGGER ABSOLUTE PLACEMENT OF CODE AND VARIABLES CODE VARIABLES CC2430 IEEE ADDRESS WRITE AN ADDRESS READ AN ADDRESS TUTORIAL CREATE A NEW PROJECT. OPTIONS. General Options, Code Model Near (Applies also to CCxx10). General Options, Code Model Banked (Does not apply to CCxx10). Linker Debugger. INCLUDE SOURCE FILES COMPILE AND LINK DEBUG DOCLIMENT HISTORY	14 15 15 15 16 16 18 19 20 23 24 25 26 27 28







1 Introduction

This manual is written to clarify some aspects that have to be taken into consideration when working with CC2430, CC2510 or CC1110 together with the IAR Workbench.

The CC2431 is in most of the aspects described in this manual equal to CC2430, and will not be mentioned explicit. CC2431 is only delivered with a flash size of 128 kB. Every configuration file available for CC2430 is also available for CC2431.

2 Definitions

CC2430-F32	CC2430 with 32KB FLASH
CC2430-F64	CC2430 with 64KB FLASH
CC2430-F128	CC2430 with 128KB FLASH
CCxx10	Refers to either CC2510 or the CC1110.







3 Installation

Install the IAR Embedded Workbench as described in the IAR User Manual.

3.1 Chipcon CC1110/ CC2510 Extensions

3.1.1 Configuration Files

C2510/CC1110 has 4 configuration files associated. For CC2510 these files are installed with IAR Workbench, for CC1110 they can be downloaded from our web site. The configuration files should be located according to Table 2.

File name	Location	Description
CCxx10.i51	C:\Program Files\IAR Systems\Embedded	Setup some parts of an
	Workbench 4.05\8051\config\derivatives\chipcon	IAR Project.
CCxx10.ddf	C:\Program Files\IAR Systems\Embedded	Describe each register
	Workbench 4.05\8051\config\derivatives\chipcon	used in the debugger.
Ink51ew_ccxx10.x	C:\Program Files\IAR Systems\Embedded	Set up the linker to reflect
cl	Workbench 4.05\8051\config	CCxx10.
ioCCxx10.h	C:\Program Files\IAR Systems\Embedded	Header file for CCxx10.
	Workbench 4.05\8051\inc	

Table 1: Configuration files

3.1.2 Chipcon Library Files

The files which are described here are part of the Chipcon library and can be downloaded from www.chipcon.com.

In the IAR standard library, cexit will by default occupy one of the four hardware breakpoints. To free this breakpoint include the file below in the project.

• Chipcon_cexit.s51

Note:

Add \$TOOLKIT_DIR\$\SRC\LIB\ to Options -> Assembler -> Additional include directives.







3.2 Chipcon CC2430 Extensions

3.2.1 Configuration Files

CC2430 has five configuration files associated. These files are installed with IAR Workbench. The configuration files are located according to Table 2.

File name	Location	Description
CC2430.i51	C:\Program Files\IAR Systems\Embedded	Setup some parts of an
	Workbench	IAR Project.
	4.05\8051\config\derivatives\chipcon	
CC2430.ddf	C:\Program Files\IAR Systems\Embedded	Describe each register
	Workbench	used in the debugger.
	4.05\8051\config\derivatives\chipcon	
Ink51ew_cc2430.xcl	C:\Program Files\IAR Systems\Embedded	Set up the linker to reflect
	Workbench 4.05\8051\config	CC2430.
Ink51ew_cc2430b.xcl	C:\Program Files\IAR Systems\Embedded	Set up the linker to reflect
	Workbench 4.05\8051\config	CC2430 with banked code
		model.
ioCC2430.h	C:\Program Files\IAR Systems\Embedded	Header file for CC2430.
	Workbench 4.05\8051\inc	

Table 2: Configuration files

3.2.2 Chipcon Library Files

Some of the functions in the library included with IAR Workbench cannot be used with CC2430.

The files which are described here are part of the Chipcon library and can be downloaded from www.chipcon.com.

In the IAR standard library, cexit will by default occupy one of the four hardware breakpoints. To free this breakpoint include the file below in the project.

• Chipcon_cexit.s51

To use banked code model the following files have to be included in the project, they will overwrite the files that are included by default in the IAR library.

- Chipcon_cstartup.s51
- Chipcon_banked_code_support.s51

Note:

Add \$TOOLKIT_DIR\$\SRC\LIB\ to Options -> Assembler -> Additional include directives.







4 CC1110/ CC2510 Linker File

There is one linker file for the CC2510 and one for CC1110.

4.1 XData RAM

The linker files set up 4k of XData RAM. The range from 0xFDAA – 0xFEFF is used to store register values during power mode (PM) 2 and 3. Hence, any data stored here before entering PM 2/3 will be lost. If PM 2/3 is not used, the entire range from 0xF000 – 0xFEFF can be used safely.

The XData RAM memory map in Figure 3 is reflected in the linker file.

0xFFFF	IData
0xFF00	IDala
0xFEFF	
	Lose data in PM2/3
0xFDAA	
0xFDA9	
	Retain data in all PM
0xF000	

Figure 1: XData RAM Memory

Figure 3 shows how XData is organized. To place variables in the memory space which is not retained will have to be explicitly located in that memory segment (see chapter 7.1).

	- 4	
NI	ATA'	
	ULC.	

If your application is not using PM2/ PM3 you can change the linker file to use the full XData memory. Change the line below in Ink51ew_cc1110.xcl or Ink51ew_cc2510.xcl: From: -D_IXDATA_END=FDA9 To: -D_IXDATA_END=FEFF







4.2 Code

The CCxx10 has 32 kB of code memory. This is reflected in the linker file the following way:

-D_CODE_START=0x0000	
-D_CODE_END=0x7FFF	// 32KB code

4.2.1 Create HEX-file

To generate a hex-file that can be downloaded to the CCxx10 by the Chipcon Flash Programmer, setup the Project Options as shown in Figure 7.

Options for node "appE Category: General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	×_cc2430" Output Extra Output #define Diagnostics List Config Proce Output Extra Output #define Diagnostics List Config Proce Output file Secondary output file: [ilename.hex (None for the selected format Eormat © Debug information for C-SPY ✓ With runtime control modules [With I/O emulation modules ✓ With functime control modules [Buffered terminal output [Allow C-SPY-specific extra output file [Other Output format: intel-extended [[[[Module-local symbols: Include all [[[[[
	Module-local symbols: Include all	<u>a</u>
	OK Can	

Figure 2: HEX-file generation







5 CC2430 Linker File

CC2430 is delivered in three different versions CC2430-F32, CC2430-F64 and CC2430-F128. There are two linker files for CC2430, lnk51ew_cc2430.xcl and lnk51ew_cc2430b.xcl. They both reflect the physical aspects of CC2430. Lnk51ew_cc2430b.xcl is for CC2430-F128 and banked code model, lnk51ew_cc2430.xcl is for use with non-banked code, and can be edited to reflect any chip model.

5.1 XData RAM

The linker file is by default set up to use 4k of XData RAM.

The XData RAM memory map in Figure 3 is reflected in the linker file.

0xFFFF IData
0xFEFF Lose data in PM2/ PM3 0xFD58
0xFD57
Retain data in all PM
0xF000
^{0xEFFF} Lose data in PM2/ PM3
0xE000

Figure 3: XData RAM Memory

CC2430 has 8k XData RAM memory, but only 4k is preserved during Power Mode 2 and Power Mode 3. Figure 3 shows how XData is organized. To place variables in the memory space which is not retained will have to be explicitly located in that memory segment (see chapter 7.1).

Note:

If your application is not using PM2/ PM3 you can change the linker file to use the full XData memory. Change the line below in Ink51ew_cc2430[b].xcl: From: -D_IXDATA_START=F000 // The internal xdata is 4k. To: -D_IXDATA_START=E000 // using low power RAM as normal RAM







5.2 Code

When using Near Code Model, a maximum of 64k (bank 0 and 1, Figure 4) of code memory is available (16 bits is used for address). When using Banked Code Model for CC2430-F128, 128k is available.

Lnk51ew_cc2430.xcl is set up to reflect CC2430-F128 and CC2430-F64 (both has 64 k flash for non-banked code). To reflect CC2430-F32, edit the file as described below as well as in the linker file. With use of this linker file, maximum code size is 64K.

Ink51ew_cc2430.xcl contains this fragment:

-D_CODE_END=0xFFFF //-D_CODE_END=0x7FFF	// Last address for code, CC2430-F64 and CC2430-F128 // Last address for code, CC2430-F32
-D_NEAR_CODE_END=0xFFFF	// Last address for near code, CC2430-F64 // and CC2430-F128
//-D_NEAR_CODE_END=0x7FFF	// Last address for near code, CC2430-F32

For use of CC2430-F32, interchange the lines that are commented out.



Figure 4: Code banking







5.2.1 Near code model

For unbanked code, the address space is continuous and straight forward to use, see Figure 5. The "Physical View" is showing the address space as it is observed physically in the CC2430-F64 and CC2430-F128. The "Linker View" is showing how the linker map addresses to physical address, and "Debugger view" is showing where in the "Disassembly Window" the code can be found.



Figure 5: Near Code Memory







5.2.2 Banked code model

In Banked Code Model things get a bit trickier. Figure 6 shows the address space.

Note:		

In "Linker View" the address space is not continuous.

To explicitly locate code in each bank, see chapter 7.1



Figure 6: Banked Code Memory

"Debugger View" is upside-down in the IAR Disassembly Window in relation to Figure 6. "Bank 0*" in "Debugger View" is a copy of "Bank 0" and should not be taken into account.







5.2.3 Create HEX-file

Because of the non-continuous address space when using banked code model, the HEX file generated will not be usable with the Chipcon Flash Programmer. To map this address space into one continuous address range, include the line below into the xcl-file or insert it in the IAR IDE as shown in Figure 8. If this line is included the C-Spy debugger cannot be used.

-M(CODE)[(_CODEBANK_START+_FIRST_BANK_ADDR)-(_CODEBANK_END+_FIRST_BANK_ADDR)]*_NR_OF_BANKS+10000=0x8000

To generate the file, setup the Project Options as shown in Figure 7.

Options for node "appE Category:	x_cc2430"
General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Output Extra Output #define Diagnostics List Config Proce Image: Config Output file Image: Config Image: Config Proce Image: Config Image: Config <t< td=""></t<>
	OK Cancel

Figure 7: HEX-file generation







Options for node "appE	к_сс2430"
Category: General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Factory Settings #define Diagnostics List Config Processing Extra Options Define Options Command line options: (one per line) M(CODE)[[_CODEBANK_START+_FIRST_BANK_ADDR]-[_CODEI]
	OK Cancel

Figure 8: Address translation for Banked Code







6 Debugger

The debugger is configured to match the specific chip with use of a definition file. For CC2430 the configuration file is named CC2430.ddf. For the CCxx10 the file is named CCxx10.ddf (xx refers to either 11 or 25). To setup your project correctly, see chapter 9.

Some of the internal registers will disturb the state of the chip when they are read. These registers will consequently not be shown with correct values in the register view window, see Figure 9. They will always be shown with value 0x00. (e.g. RFD, ADCH, ADCL, ENDO)

Register		×
1/0		•
P 0	-	0xFF
- P0_0	=	1
- P0_1	=	1
- P0_2	=	1
- P0_3	=	1
- P0_4	=	1
- P0_5	=	1
- P0_6	=	1
- P0_7	=	1
⊞ P1	=	0xFF
⊞ P2	=	0x19
HPODIR	=	0x00
HP1DIR	=	0x00
HP2DIR	=	0x00
HP1IEN	=	0x00
DERCFG	=	0x00
HPICTL	=	0x00
POIFG	=	0x20
P1IFG	=	0x00
P2IFG	=	0x1E
DOSEL	=	0x00
 P1SEL	=	0x00
HP2SEL	=	0x00
H POINP	=	0x00
\pm P1INP	=	0x00
HP2INP	=	0x00
ADCCFG	=	0x00









7 Absolute Placement of Code and Variables

This chapter shows how to place code and variables at absolute addresses, and the chapter is mainly intended for the CC2430. For an extended description please refer to the IAR user manual.

7.1 Code

The linker file for CC2430-F128 has defined one code space for each bank. It is not required to use these definitions for banked code model. The linker will distribute the code automatically.

```
int f() @ "BANK1"{
    return 1;
}
#pragma location="BANK1"
int g(){
    return 2;
}
____near_func int main( void )
{
    int a = g();
    int b = f();
    return 0;
}
```

Both f() and g() will be placed in code segment named BANK1. The main routine will be placed in BANK 1 by default. To place main (or some other routines) in bank 0 use the keyword near func, as indicated in the listing above.

The linker file for CC2430-F128 defines "BANK1", "BANK2" and "BANK3".

Note:

There is no banking for the CCxx10, as they only have 32 kB code memory size.

7.2 Variables

Absolute placement of variables is described below.

__no_init int myArray[128] @ "PM0_XDATA"; __no_init int myInteger @ 0xE000;

"PM0_XDATA" is defined in the linker file. This is the address space from 0xE000 to 0xEFFF. It is necessary to explicitly define variables to be located in this area; if variables are "placed by default" they will be located in address range 0xF000 to 0xFD57.







8 CC2430 IEEE address

When delivered in a kit each CC2430 has an IEEE address stored in code memory. The segment is defined to be the last eight byte on the last flash page. Table 3 gives the address.

Chip	Code model	Physical Address	IAR Linker Address
CC2430-F32	Near	0x07FF8	0x07FF8
CC2430-F64	Near	0x0FFF8	0x0FFF8
CC2430-F128 ¹	Near	0x0FFF8	0x0FFF8
CC2430-F128	Banked	0x1FFF8	0x3FFF8

Table 3: IEEE address space

The linker file defines a segment for the IEEE address, named "IEEE_ADDRESS_SPACE".

8.1 Write an address

Use Chipcon Flash Programmer, available from www.chipcon.com, to write an IEEE address to CC2430 or see the next section. Information about Chipcon Flash Programmer is available in Chipcon Flash Programmer User Manual.

It is possible to write the IEEE address as part of a C program. "IEEE_ADDRESS_SPACE" is defined in the linker file. To write an address to this segment, include the code below in the C program (change the address). The IEEE address which the chip was delivered with is written on the development card.

__root __code const unsigned char IEEE_ADDRESS[8] @ "IEEE_ADDRESS_SPACE" = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};

If the chip already contains an address, this address will be overwritten by inclusion of this example code.

8.2 Read an address

If the IEEE address is written to code as the example code in chapter 8.1 shows, it can easily be read with use of the same variable.

If the address is stored in the chip, it can be read with use of the example code shown below. If interrupt is enabled, remember to turn this off before calling the routine.

If the project is using code model near, set flash size to 64 for both CC2430-F64 and CC2430-F128.

¹ Not programmed when delivered







```
// #define CC2430_FLASH_SIZE 32
#define CC2430_FLASH_SIZE 64
// #define CC2430_FLASH_SIZE 128
#if (CC2430 FLASH SIZE == 32)
  #define IEEE ADDRESS ARRAY 0x7FF8
#elif (CC2430 FLASH SIZE == 64) || (CC2430 FLASH SIZE == 128)
  #define IEEE ADDRESS ARRAY 0xFFF8
#endif
  near func void getIEEEAddress(unsigned char *a){
#if ( CC2430_FLASH_SIZE == 128 )
  unsigned char bank;
  bank = MEMCTR;
 // switch to bank 3
  MEMCTR \mid = 0x30;
#endif
  a[0] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 0);
  a[1] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 1);
 a[2] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 2);
a[3] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 3);
  a[4] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 4);
  a[5] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 5);
 a[6] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 6);
a[7] = *(unsigned char __code *)(IEEE_ADDRESS_ARRAY + 7);
#if ( CC2430 FLASH SIZE == 128 )
 // restore old bank settings
  MEMCTR = bank:
#endif
}
unsigned char addr[8];
int main(void)
{
 getIEEEAddress(addr);
  for(;;);
}
```





CC1110/CC2430/CC2510

9 Tutorial

This chapter shows how to set up a new project for CC2430 in the IAR Workbench. A project for CCxx10 can be set up in a similar fashion by replacing the CC2430-specific files with the corresponding files for the chip in question.

The setup which is shown here use settings used for common projects. Both setup for project using near code model and banked code model is presented.

The tutorial shows only screen shots from settings which need to be changed.

9.1 Create a new project

To quickly create a CC2430 application Chipcon recommends that the user begins by creating a software project.

From the "Project" menu, choose "Create New Project...".

X 14	AR En	nbedde	ed Work	oench (IDE		
File	Edit	View	Project	Tools	Window	Help	
Work	space		Add F Add C Impor Edit C	iles iroup t File Li ionfigur	st ations		HTX
Fil	es		Remo	ve			
⊢			Creat	e New F	Project		
			Add E	xisting	Project		
⊢			Optio	ns		ALT+F7	
			Sourc	e Code	Control		•
			Make Comp Rebui Clean Batch	ile Id All build		F7 CTRL+F7 F8	7
			Stop	Build			
			Debu Make	J 8. Resta	art Debugg	CTRL+D jer	

Figure 10: Create New Project

Choose "Empty project" and press OK.







Create New Project	×
Tool chain: 8051	•
Project templates:	
Empty project asm C++ C CLIB DLIB DLIB	
Description:	
Creates an empty project.	
	OK Cancel

Figure 11: Choose Project Type

Save As					? ×
Save jn:	AR_Project		•	• 🗈 💣 🎟•	
History Desktop My Documents My Computer My Network P	File <u>n</u> ame: Save as <u>t</u> ype:	Test1 Project Files (*.ewp)		•	<u>S</u> ave Cancel

Figure 12: Save Project

Give the new project a name and press "Save".

9.2 Options

To set up the project for use with CC2430 choose "Options" from the "Project" menu.



Chipcon Products from Texas Instruments



Chapter 9.2.1 describes all settings that need to be changed for CC2430 for code model near. This also applies to CCxx10. Chapter 9.2.2 describes the settings that differ from code model near to code model banked. This does not apply to CCxx10.

Options for node "ieee_	test"	×
Category: C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Target Data Pointer Arithmetic Unit Code Bank Output Library C Derivative information Derivative: CC2430	
	OK Cancel	

9.2.1 General Options, Code Model Near (Applies also to CCxx10)

Figure 13: Target, Code Model Near

In the "Derivative information" box select the CC2430.i51 file located in folder: Embedded Workbench 4.05 \8051\config\derivatives\Chipcon.

Use ito select the file.

Set up rest of the settings on the "Target" settings as Figure 13 shows.







Use one data pointer. It's important to set "DPTR select" to "Set using XOR/ AND" even if only one DPTR is in use.

Options for node "Test:	l" >
Category: C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Target Data Pointer Arithmetic Unit Code Bank Output Library C<
	OK Cancel









Change XDATA stack size to 0x1FF.

Options for node "Test: Category:	1"	×
General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Library Configuration Library options Stack/Heap MISRA C Stack sizes IDATA: 0x40 Heap sizes PDATA: 0x80 XDATA: 0xFFF XDATA: 0x1FF Huge: 0xFFF Extended: 0x3FF Image: Control of the second secon	
	OK Car	ncel

Figure 15: Stack/ Heap settings







9.2.2 General Options, Code Model Banked (Does not apply to CCxx10)

Note: Remember to include Chipcon_cstartup.s51 and Chipcon_banked_code_support.s51 (chapter 3.2.2 when using code model banked).	
Options for node "Test1" Category: C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator Category: Large Number of vitual registers: B CoDe memory Category:	
OK Cancel	

Figure 16: Target, Code Model Banked

The settings in Figure 17 are only available when Code Model Banked is active

Options for node "Test1					×
Category:					
General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices	Target Data Pointer Register address Bank start Bank end	Arithmetic Unit 0xC7 0x8000 0xFFFF	Code Bank	Output Lib	rary C_▲ ▶

Figure 17: Code Bank







9.2.3 Linker

Choose correct linker file, see chapter 4, as "Linker command file".

Code Model	File
Near	Ink51ew_cc2430.xcl
Banked	Ink51ew_cc2430b.xcl

Table 4: Linker files

Options for node "Test	L
Category: General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Factory Settings Output Extra Output #define Diagnostics List Config Proce Image: Transformed set and transformed se
	OK Cancel

Figure 18: Linker







9.2.4 Debugger

Use "Chipcon" as debugger, and check that the "Device Description file" is set to CC2430.ddf. The ddf file is located in: Embedded Workbench 4.05 \8051\config\derivatives\Chipcon.

Options for node "Test:		x
Options for node "Test: Category: General Options C/C++ Compiler Assembler Custom Build Build Actions Linker Debugger Third-Party Driver	Factory Settings Setup Extra Options Plugins Driver Chipcon	×
Debugger Third-Party Driver Chipcon ROM-Monitor Analog Devices Silabs Simulator	Setup macros	
	✓ Overide default \$TOOLKIT_DIR\$\config\derivatives\Chipcon\CC2430.ddf OK	

Figure 19: Debugger





CC1110/CC2430/CC2510

9.3 Include source files

To create a new source file choose File -> New -> File.

💥 IAR Embedded Workbench IDE							
File	Edit	View	Project	Simulator	Tools	Window	Help
N	ew			•	File	CTRI	L+N
0	pen			•	Worksp	bace	
C	lose			T	LESU		
						1	

Figure 20: Create new source file

Edit the file and save it into your project directory.

To add this, or another file to your project, choose Project -> Add files... Locate correct file and click "Open".

🔏 IAR Embedded Workbench IDE						
File Edit View	Project	Simulator	Tools	Window	Help	
🗅 🚅 🖬 🗐	Add Files					
Workspace	Add Group					
Debug	Edit Configurations					
Files	Remove					
	Create New Project					

Figure 21: Add files to project







9.4 Compile and Link

To compile and link you project choose "Make" from the "Project" menu, or press F7.



Figure 22: Compile and Link







9.5 Debug

Start the debugger with "Debug" from the "Project" menu, or press CTRL+D.



Figure 23: Start debugger

Use of the debugger is described in the IAR user manual.







10 Document history

Revision	Date	Description/Changes
1.2	2006-02-16	Added CC2510 and CC1110. Changed layout.
1.1	2006-01-18	Removed link to configuration files on web
1.0	2005-12-15	Initial release.

