AND9371/D

AxCode::Blocks Quick Guide

ON Semiconductor®

www.onsemi.com

APPLICATION NOTE

Introduction

Figure 1 shows a diagram of the ON Semiconductor AX8052 Development System Architecture.

Radio Link parameters are set using the AX−RadioLAB GUI. AX−RadioLAB produces source code, compiles it and downloads it into the target board.

AxCode::Blocks is the graphical Integrated Development Environment (IDE) for AX8052 projects. It is a customized version of the popular Code::Blocks IDE. It can be used to further customize the AX−RadioLAB generated code, or it can be used to create new projects (such as those that do not involve a radio link).

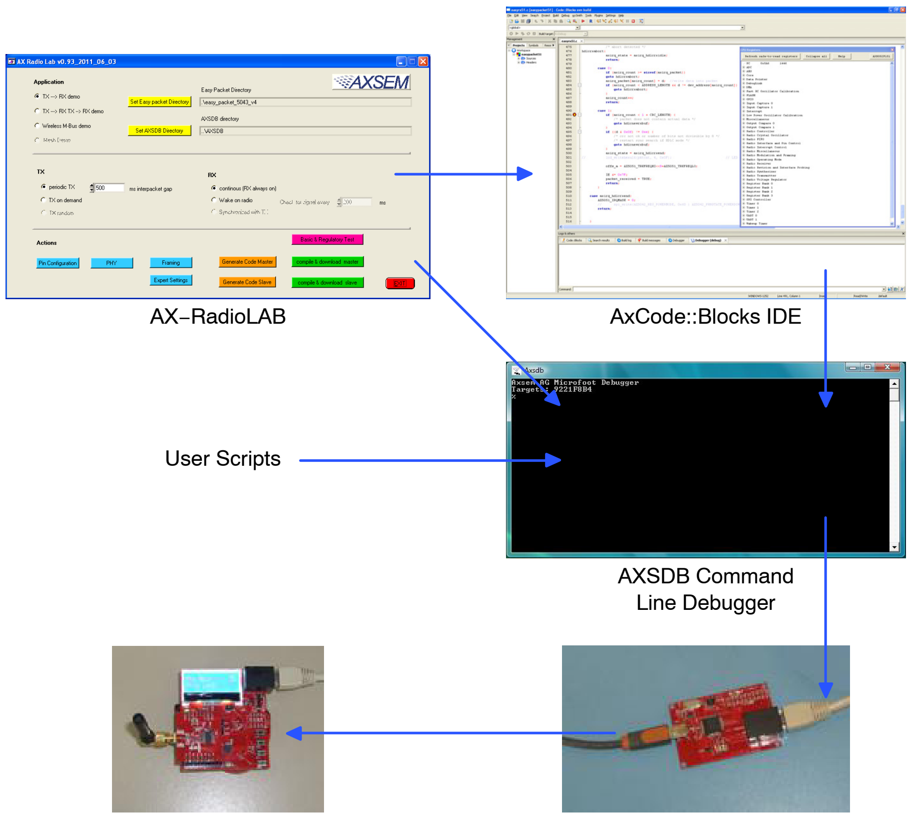
Both AX−RadioLAB and AxCode::Blocks talk to the ON Semiconductor Symbolic (command line) Debugger (AXSDB). Normally, Users need not directly interact with AXSDB. AXSDB can however be useful for automated or scripted tasks, thanks to its command line and TCL scripting features.

The Debug Adapter provides the link between the developers workstation and the target board.

This document should guide the reader through the installation of AxCode::Blocks and its use to create, compile and debug a little project.

AX−RadioLAB is documented in a separate document.

For general issues regarding Code::Blocks, please refer to its manual:
http://www.codeblocks.org/docs/manual_en.pdf.



```
Axsem AG Microfoot Debugger
Target: 9221F8B4
```

Figure 1. On Semiconductor AX8052 Development System Architecture

Installing AxCode::Blocks

The installer contains everything you need: the SDCC compiler, the AXSDB debugger, example files and libraries and, of course, AxCode::Blocks.

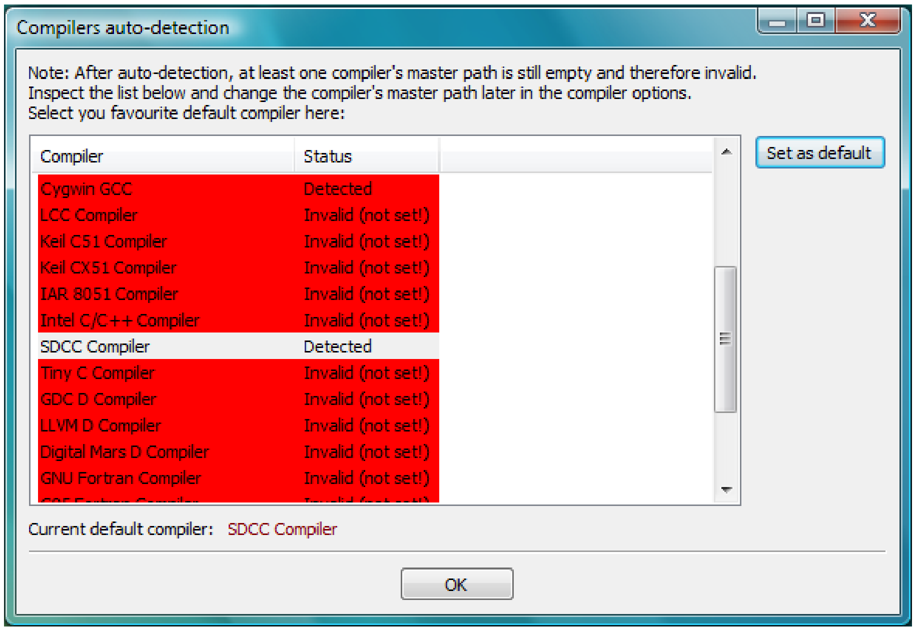
- Launch the installer.
- After accepting the terms of agreement you are asked to select the components to be installed. We strongly suggest to install all components. Failure to do so could result in missing links in your toolchain.
- Choose where to install AXSDB (it is recommended to keep the default settings). Hit *Install*.
- When asked if you want to install AxCode::Blocks too, click *Yes*: the corresponding setup wizard is started. This one is quite similar to the previous one: go through it.
- Next, you want to install the SDCC compiler. Again, the installation is pretty intuitive. Be sure to tick the option for adding the SDCC directory to the system path.
- Wait until the process finishes and you are done!

Connecting the Hardware

Please refer to the application note *AX−RF−DVK2 Quick Start Instructions* available from the ON Semiconductor website: http://www.onsemi.com.

Creating a New Project

Start AxCode::Blocks. The first time AxCode::Block starts, it scans for installed compilers and presents a list of the compilers found. Select SDCC as default.



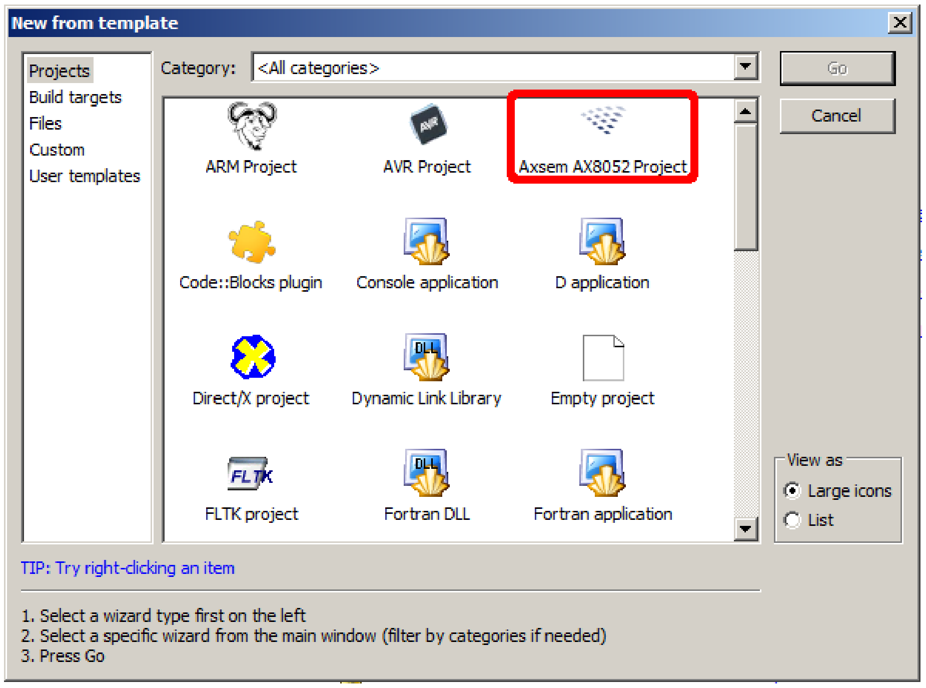
Note: After auto-detection, at least one compiler's master path is still empty and therefore invalid.
Inspect the list below and change the compiler's master path later in the compiler options.
Select you favourite default compiler here:

Compiler	Status
Cygwin GCC	Detected
LCC Compiler	Invalid (not set!)
Keil C51 Compiler	Invalid (not set!)
Keil CX51 Compiler	Invalid (not set!)
IAR 8051 Compiler	Invalid (not set!)
Intel C/C++ Compiler	Invalid (not set!)
SDCC Compiler	Detected
Tiny C Compiler	Invalid (not set!)
GDC D Compiler	Invalid (not set!)
LLVM D Compiler	Invalid (not set!)
Digital Mars D Compiler	Invalid (not set!)
GNU Fortran Compiler	Invalid (not set!)

Current default compiler: SDCC Compiler

Figure 2. Creating a New Project − Compilers Auto-Detection

- Click on *File →New →Project.*
- Choose *Axsem AX8052 Project:*

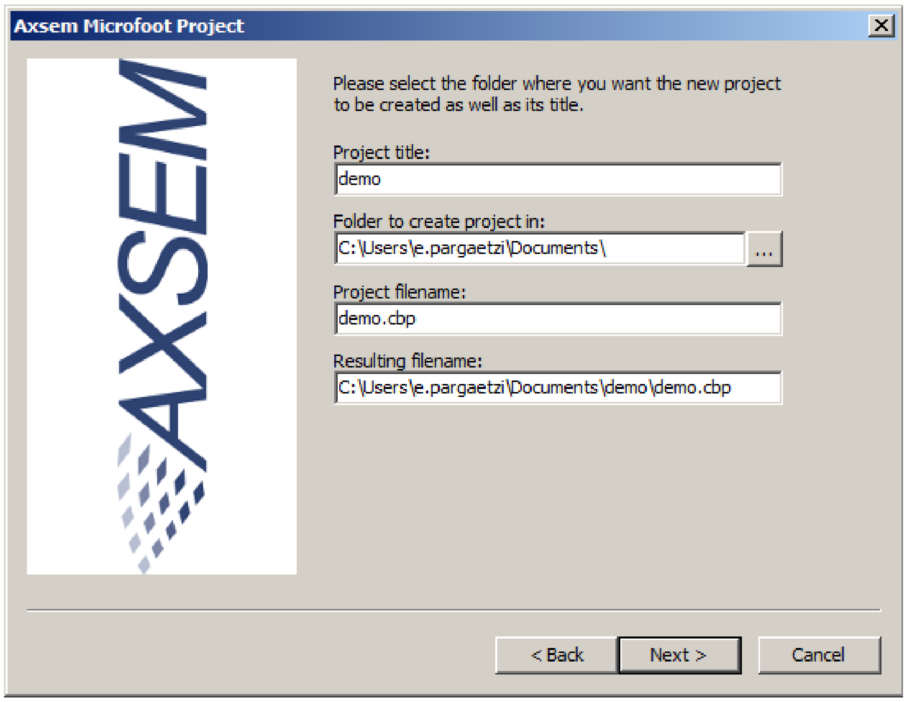


TIP: Try right-clicking an item

1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Figure 3. Creating a New Project – New from Template

A dialog will pop-up. Go through it and change the settings if needed. The following screenshots are intended as examples:



Please select the folder where you want the new project to be created as well as its title.

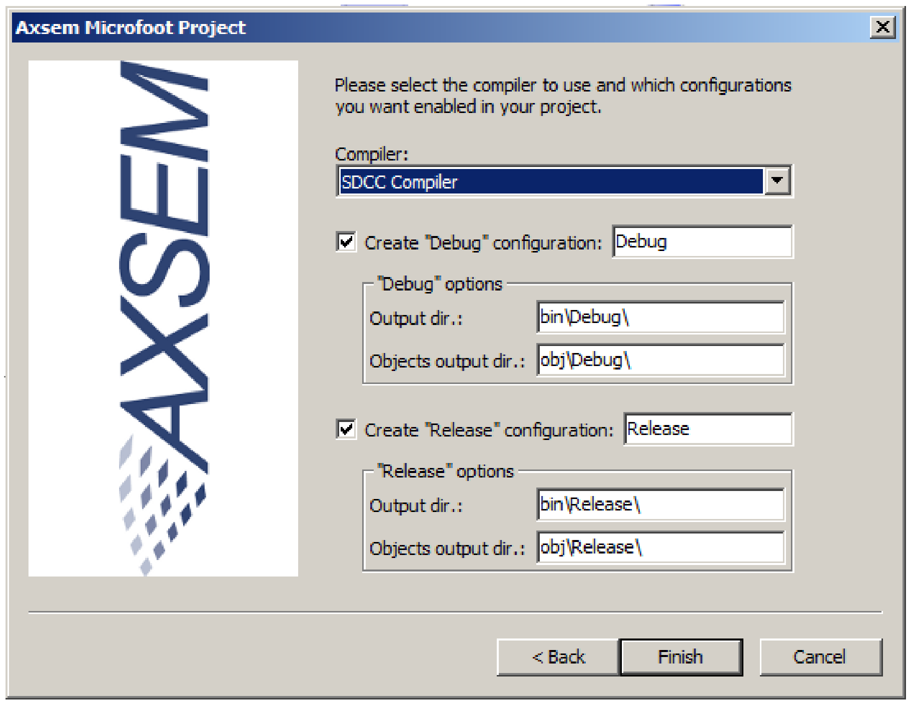
Project title: demo

Folder to create project in: C:\Users\e.pargaetzi\Documents\

Project filename: demo.cbp

Resulting filename: C:\Users\e.pargaetzi\Documents\demo\demo.cbp

Figure 4. Screenshot No. 1 – Axem Microfoot Project



Please select the compiler to use and which configurations you want enabled in your project.

Compiler: SDCC Compiler

[✓] Create "Debug" configuration: Debug

"Debug" options
Output dir.: bin\Debug\
Objects output dir.: obj\Debug\

[✓] Create "Release" configuration: Release

"Release" options
Output dir.: bin\Release\
Objects output dir.: obj\Release\

Figure 5. Screenshot No. 2 – Axem Microfoot Project

Clicking on *Finish* will create the new project. For your convenience the most important build options and compiler preferences are set automatically.

ON Semiconductor also distributes example projects. Check the ON Semiconductor homepage for the available examples.

Adding and Editing Files

An example source file has been included. You can open it by double-clicking its name on the project tree:

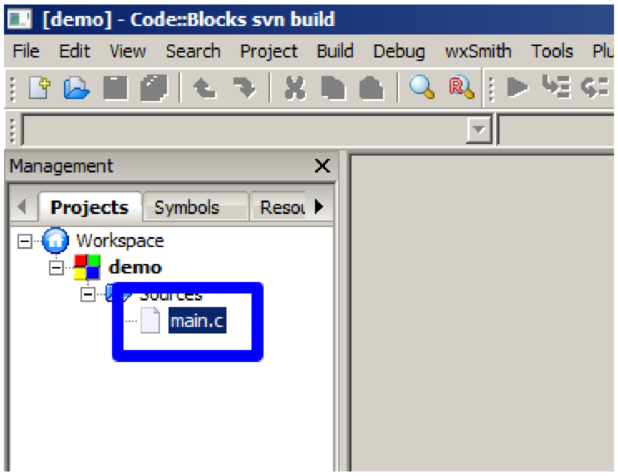


Figure 6. Code::Blocks SVN Build

You can add existing files to the project using the menu entry *Project →Add Files.*

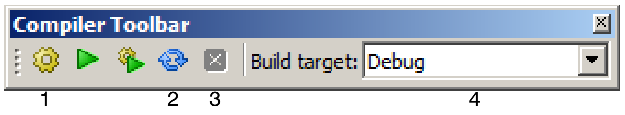
You can add new files to the project using the menu entry *File →New →File.*

Open files are shown on the right pane and can be edited.

WARNING: SDCC seems to have a bug. Do not use hyphens in your filenames. Underscores are fine.

Compiling the Project

The most important function can be accessed through the compiler toolbar:

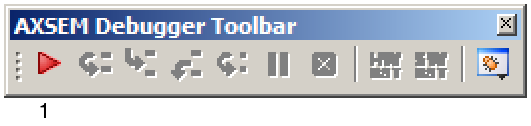


1. Build – Compile and link the project
2. Rebuild – Delete existing files and build
3. Abort – Stop the building process
4. Build target – The Debug target generates automatically debug informations

Debugging the Project

Select the AXSEM Debugger under *Debug → Active debuggers* and make sure the corresponding toolbar is visible (*View →Toolbars →AXSEM Debugger*).

Before the debugging process is started, the toolbar looks like this:

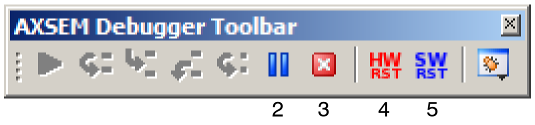


By hitting 1, the debugger is started. If no devices are found, an error message is issued[1]. If exactly one device is found, the device is automatically connected. If more than one device is found, the user is prompted to select one.

Furthermore, the user can decide to load the firmware to the device or to use the firmware already stored in it.

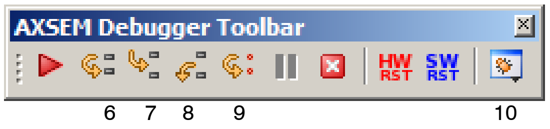
If changes are made to the project since the last build, the project is automatically compiled.

When running, the toolbar changes it’s appearance to:



The pause button 2 stops the execution of the program to examine its state. The cursor inside the editor is moved to the line corresponding to the current instruction. Button 3 does not stop the execution, but disconnects the device and exits the debugger. 4 and 5 are used to reset the microcontroller.

After hitting the pause button 2, other functions become enabled:



6. Next line – Execute the next line in the sourcecode
7. Step into – Execute the next line, if it’s a function step into it
8. Step out – Continue execution until the end of the current frame
9. Next instruction – Execute the next assembly instruction

Due to limitations in the debug information of 8052 compilers, the stepping commands 6, 7 and 8 require the microcontroller to be single-stepped. It can therefore take a long time until these commands terminate. It is always possible to stop one of these commands prematurely by hitting the pause button 2. Usually, it is preferable to set breakpoints or use the Run-To-Cursor feature over the stepping commands.

The drop-down menu 10 can be used to open the debugging windows described in the next chapter.

1 If you get an error message although a device is connected, you likely are missing the drivers. Open the *Control Panel* and navigate to the *Device Manager*. Find the unrecognized devices (look for the exclamation marks) named “Microfoot Debug Interface”, right-click them and choose to install or update the driver. Do not search online for the driver. As search directory give the directory where you installed AXSDB followed by “\ftdi”, e.g. “C:\Program Files\AXSEM\AXSDB\ftdi”. Eventually, you need to disconnect the device and restart AxCode::Blocks.

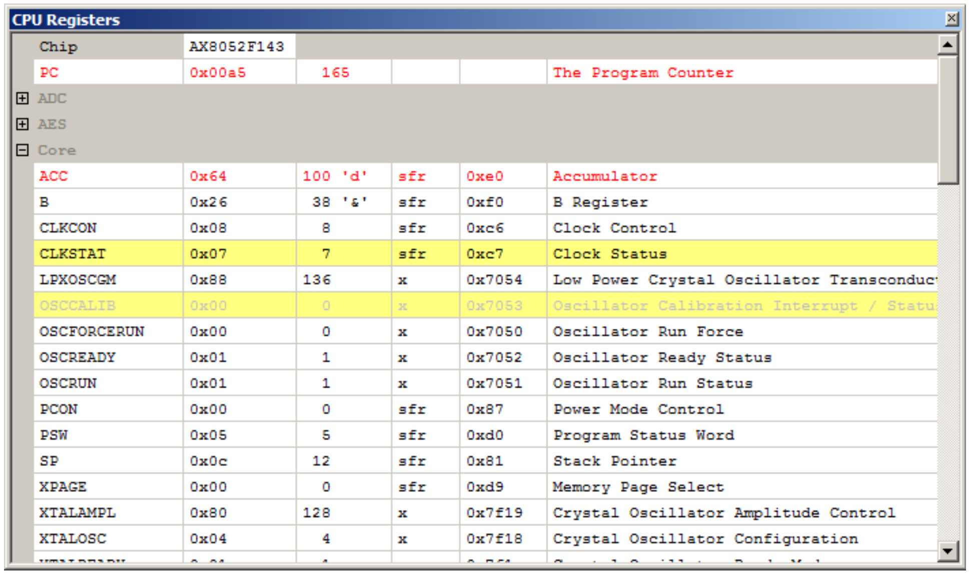
Debugging Windows

Breakpoints

This window simply displays a list of the breakpoints set. In order to set a breakpoint, click on the left of the corresponding line in the editor or hit *F5*. The same procedure removes an already set bookmark.

Registers

This window shows Microcontroller register contents while the Microcontroller is stopped. Registers of ON Semiconductor Radio Chips or SoC functions are also displayed.



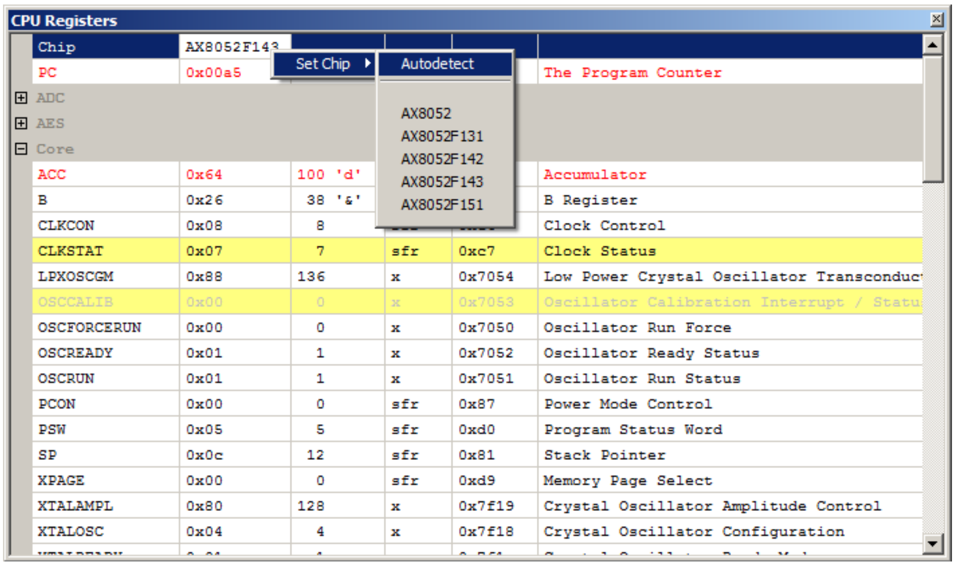
CPU Registers

Chip	AX8052F143				
PC	0x00a5	165			The Program Counter
⊞ ADC					
⊞ AES					
⊟ Core					
ACC	0x64	100 'd'	sfr	0xe0	Accumulator
B	0x26	38 '&'	sfr	0xf0	B Register
CLKCON	0x08	8	sfr	0xc6	Clock Control
CLKSTAT	0x07	7	sfr	0xc7	Clock Status
LPXOSCGM	0x88	136	x	0x7054	Low Power Crystal Oscillator Transconduc
OSCCALIB	0x00	0	x	0x7053	Oscillator Calibration Interrupt / Statu
OSCFORCERUN	0x00	0	x	0x7050	Oscillator Run Force
OSCREADY	0x01	1	x	0x7052	Oscillator Ready Status
OSCRUN	0x01	1	x	0x7051	Oscillator Run Status
PCON	0x00	0	sfr	0x87	Power Mode Control
PSW	0x05	5	sfr	0xd0	Program Status Word
SP	0x0c	12	sfr	0x81	Stack Pointer
XPAGE	0x00	0	sfr	0xd9	Memory Page Select
XTALAMPL	0x80	128	x	0x7f19	Crystal Oscillator Amplitude Control
XTALOSC	0x04	4	x	0x7f18	Crystal Oscillator Configuration

Figure 7. CUP Registers – Microcontroller Register Contents

Registers are grouped into sections. Light gray bars show the section name, the + or – sign to the left of the section name allows to show or hide the section.

The first line of the dialog shows the name of the chip. The chip is normally auto-detected. Should auto-detection fail, the chip type can be manually set by right-clicking on the chip name.



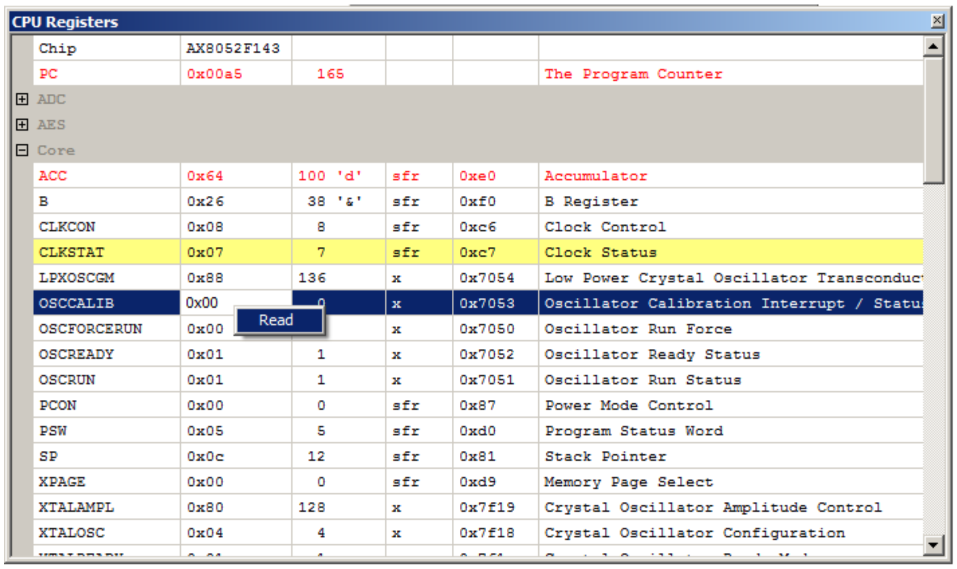
CPU Registers

Chip	AX8052F143				
PC	0x00a5				The Program Counter
⊞ ADC					
⊞ AES					
⊟ Core					
ACC	0x64	100 'd'			Accumulator
B	0x26	38 '&'			B Register
CLKCON	0x08	8			Clock Control
CLKSTAT	0x07	7	sfr	0xc7	Clock Status
LPXOSCGM	0x88	136	x	0x7054	Low Power Crystal Oscillator Transconduc
OSCCALIB	0x00	0	x	0x7053	Oscillator Calibration Interrupt / Statu
OSCFORCERUN	0x00	0	x	0x7050	Oscillator Run Force
OSCREADY	0x01	1	x	0x7052	Oscillator Ready Status
OSCRUN	0x01	1	x	0x7051	Oscillator Run Status
PCON	0x00	0	sfr	0x87	Power Mode Control
PSW	0x05	5	sfr	0xd0	Program Status Word
SP	0x0c	12	sfr	0x81	Stack Pointer
XPAGE	0x00	0	sfr	0xd9	Memory Page Select
XTALAMPL	0x80	128	x	0x7f19	Crystal Oscillator Amplitude Control
XTALOSC	0x04	4	x	0x7f18	Crystal Oscillator Configuration

Figure 8. CUP Registers – Example of Registers Sections

Registers and their contents are shown in the remaining rows. The first column shows the register name. The second column shows the current register contents as a hexadecimal number. The third column displays the register contents as a decimal number; if this number is greater or equal 32, it is also displayed as ASCII character. The fourth and fifth columns display the address space and the address of the register, while the sixth column displays a short description of the register.

Registers that have changed since the last processor break are displayed in red. Registers that can cause side effects when read are not automatically read. They are displayed with a light yellow background. Their values are also shown in light-gray, if they have not yet been read.

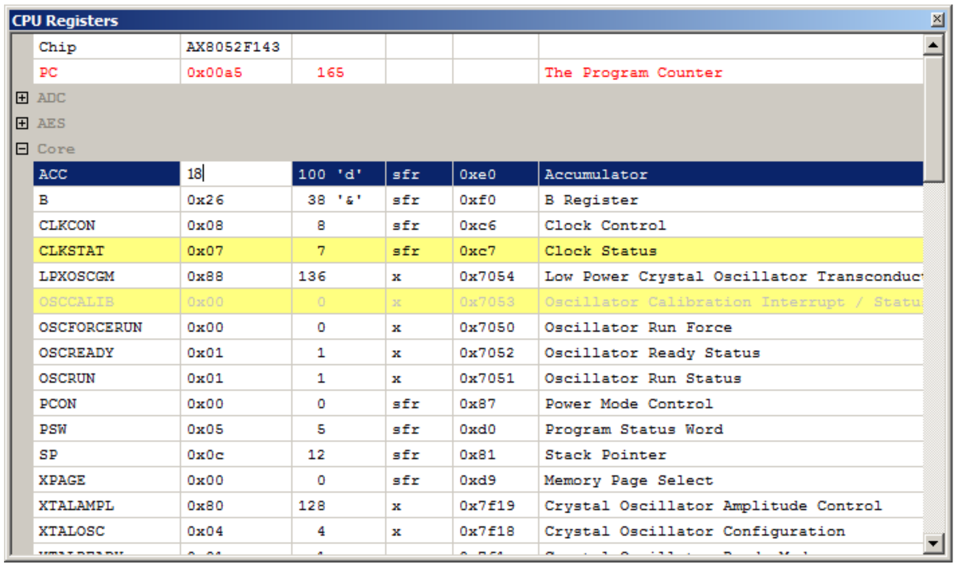


Chip	AX8052F143				
PC	0x00a5	165			The Program Counter
⊞ ADC					
⊞ AES					
⊟ Core					
ACC	0x64	100 'd'	sfr	0xe0	Accumulator
B	0x26	38 '&'	sfr	0xf0	B Register
CLKCON	0x08	8	sfr	0xc6	Clock Control
CLKSTAT	0x07	7	sfr	0xc7	Clock Status
LPXOSCGM	0x88	136	x	0x7054	Low Power Crystal Oscillator Transconduc
OSCCALIB	0x00	0	x	0x7053	Oscillator Calibration Interrupt / Statu
OSCFORCERUN	0x00		x	0x7050	Oscillator Run Force
OSCREADY	0x01	1	x	0x7052	Oscillator Ready Status
OSCRUN	0x01	1	x	0x7051	Oscillator Run Status
PCON	0x00	0	sfr	0x87	Power Mode Control
PSW	0x05	5	sfr	0xd0	Program Status Word
SP	0x0c	12	sfr	0x81	Stack Pointer
XPAGE	0x00	0	sfr	0xd9	Memory Page Select
XTALAMPL	0x80	128	x	0x7f19	Crystal Oscillator Amplitude Control
XTALOSC	0x04	4	x	0x7f18	Crystal Oscillator Configuration

Figure 9. CUP Registers – Example of Registers Sections

Registers can be (re-)read by either right-clicking into the row and selecting Read, or by selecting the row and pressing the space-bar.

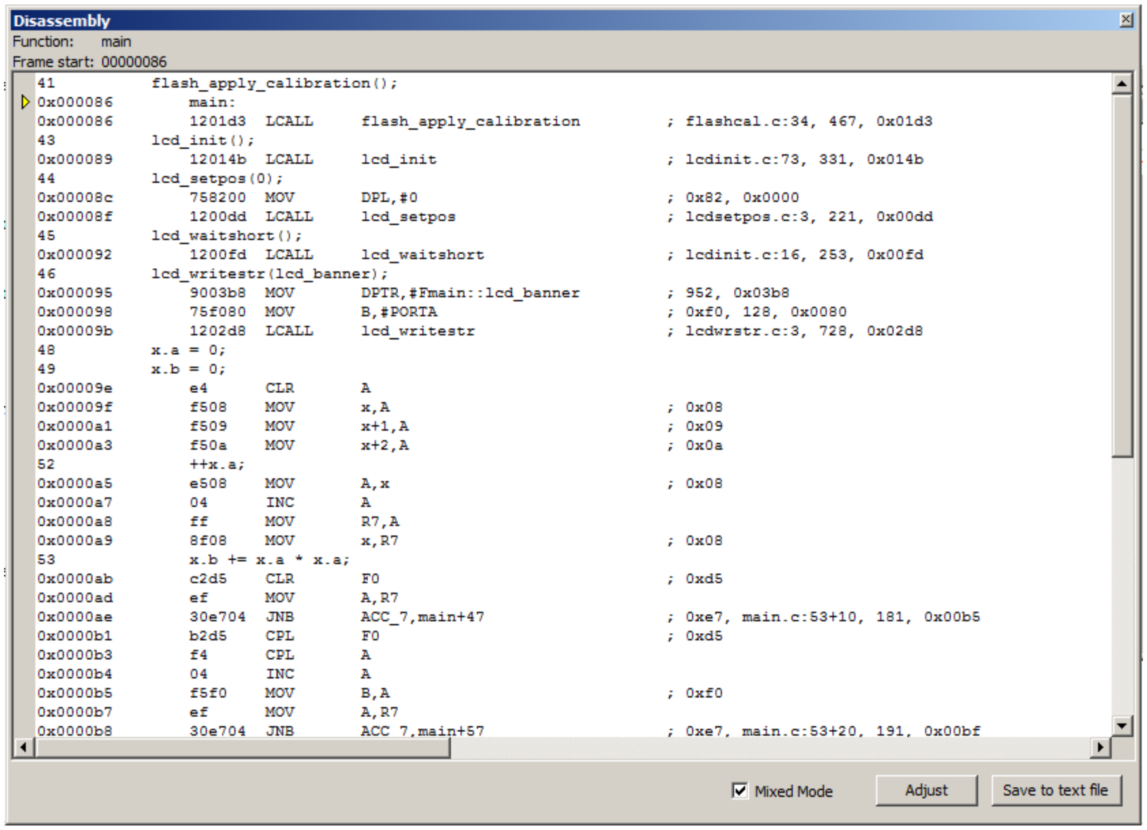
Register values can be changed by clicking into the second column and entering a number. Numbers can be entered both in decimal format (eg. 18), as well as hexadecimal format (eg. 0 x 12).



Chip	AX8052F143				
PC	0x00a5	165			The Program Counter
⊞ ADC					
⊞ AES					
⊟ Core					
ACC	18	100 'd'	sfr	0xe0	Accumulator
B	0x26	38 '&'	sfr	0xf0	B Register
CLKCON	0x08	8	sfr	0xc6	Clock Control
CLKSTAT	0x07	7	sfr	0xc7	Clock Status
LPXOSCGM	0x88	136	x	0x7054	Low Power Crystal Oscillator Transconduc
OSCCALIB	0x00	0	x	0x7053	Oscillator Calibration Interrupt / Statu
OSCFORCERUN	0x00	0	x	0x7050	Oscillator Run Force
OSCREADY	0x01	1	x	0x7052	Oscillator Ready Status
OSCRUN	0x01	1	x	0x7051	Oscillator Run Status
PCON	0x00	0	sfr	0x87	Power Mode Control
PSW	0x05	5	sfr	0xd0	Program Status Word
SP	0x0c	12	sfr	0x81	Stack Pointer
XPAGE	0x00	0	sfr	0xd9	Memory Page Select
XTALAMPL	0x80	128	x	0x7f19	Crystal Oscillator Amplitude Control
XTALOSC	0x04	4	x	0x7f18	Crystal Oscillator Configuration

Figure 10. CUP Registers – Example of Registers Values

Disassembly



```
Function:    main
Frame start: 00000086
41          flash_apply_calibration();
0x000086        main:
0x000086        1201d3  LCALL     flash_apply_calibration        ; flashcal.c:34, 467, 0x01d3
43          lcd_init();
0x000089        12014b  LCALL     lcd_init                       ; lcdinit.c:73, 331, 0x014b
44          lcd_setpos(0);
0x00008c        758200  MOV       DPL,#0                         ; 0x82, 0x0000
0x00008f        1200dd  LCALL     lcd_setpos                     ; lcdsetpos.c:3, 221, 0x00dd
45          lcd_waitshort();
0x000092        1200fd  LCALL     lcd_waitshort                  ; lcdinit.c:16, 253, 0x00fd
46          lcd_writestr(lcd_banner);
0x000095        9003b8  MOV       DPTR,#Fmain::lcd_banner        ; 952, 0x03b8
0x000098        75f080  MOV       B,#PORTA                       ; 0xf0, 128, 0x0080
0x00009b        1202d8  LCALL     lcd_writestr                   ; lcdwrstr.c:3, 728, 0x02d8
48          x.a = 0;
49          x.b = 0;
0x00009e        e4      CLR       A
0x00009f        f508    MOV       x,A                            ; 0x08
0x0000a1        f509    MOV       x+1,A                          ; 0x09
0x0000a3        f50a    MOV       x+2,A                          ; 0x0a
52              ++x.a;
0x0000a5        e508    MOV       A,x                            ; 0x08
0x0000a7        04      INC       A
0x0000a8        ff      MOV       R7,A
0x0000a9        8f08    MOV       x,R7                           ; 0x08
53              x.b += x.a * x.a;
0x0000ab        c2d5    CLR       F0                             ; 0xd5
0x0000ad        ef      MOV       A,R7
0x0000ae        30e704  JNB       ACC_7,main+47                  ; 0xe7, main.c:53+10, 181, 0x00b5
0x0000b1        b2d5    CPL       F0                             ; 0xd5
0x0000b3        f4      CPL       A
0x0000b4        04      INC       A
0x0000b5        f5f0    MOV       B,A                            ; 0xf0
0x0000b7        ef      MOV       A,R7
0x0000b8        30e704  JNB       ACC_7,main+57                  ; 0xe7, main.c:53+20, 191, 0x00bf
```

Figure 11. Disassembly Dialog Box

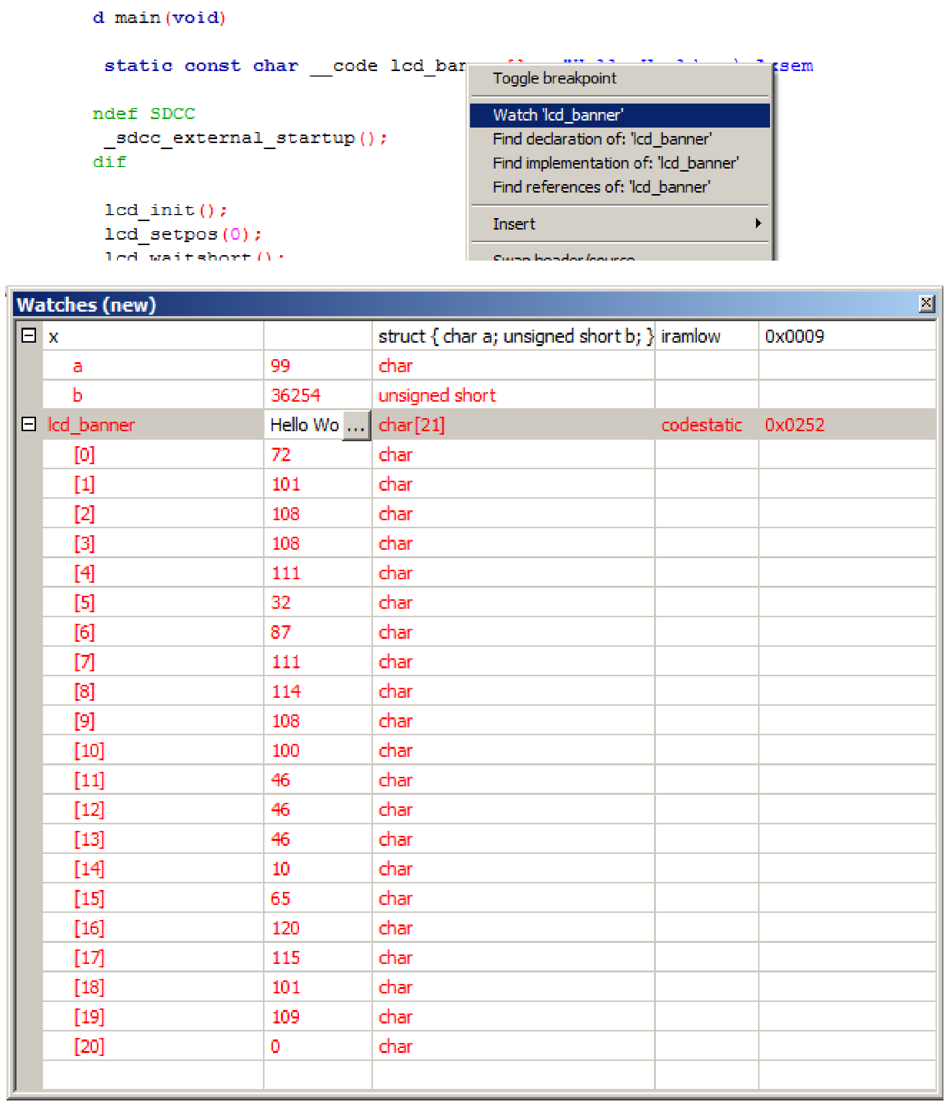
The disassembly window shows the disassembly of the current function. A yellow triangle indicates the next instruction that is executed when the microcontroller is

Memory Dump

This dialog allows to read a range in the memory of the microcontroller unit and displays it.

Watches

Watches allow the user to monitor the content of a variable which may be defined only in the high-level programming language. The easiest way to add a watch is to right click on the name of the variable and to choose the corresponding menu entry:



```
d main(void)

 static const char __code lcd_bar                        sem

ndef SDCC
 _sdcc_external_startup();
dif

 lcd_init();
 lcd_setpos(0);
```

Toggle breakpoint
Watch 'lcd_banner'
Find declaration of: 'lcd_banner'
Find implementation of: 'lcd_banner'
Find references of: 'lcd_banner'
Insert
Watches (new)

⊟ x		struct { char a; unsigned short b; }	iramlow	0x0009
a	99	char		
b	36254	unsigned short		
⊟ lcd_banner	Hello Wo …	char[21]	codestatic	0x0252
[0]	72	char		
[1]	101	char		
[2]	108	char		
[3]	108	char		
[4]	111	char		
[5]	32	char		
[6]	87	char		
[7]	111	char		
[8]	114	char		
[9]	108	char		
[10]	100	char		
[11]	46	char		
[12]	46	char		
[13]	46	char		
[14]	10	char		
[15]	65	char		
[16]	120	char		
[17]	115	char		
[18]	101	char		
[19]	109	char		
[20]	0	char		

Figure 12. Example of Adding a Watch

The watch window is able to display complex variable types. The last line is empty; its purpose is to allow adding watches simply by typing a name into the first column of the last empty row. Watch variables can be deleted by right-clicking into the name field of the watch to be deleted. A context menu then allows to rename or delete the watch.

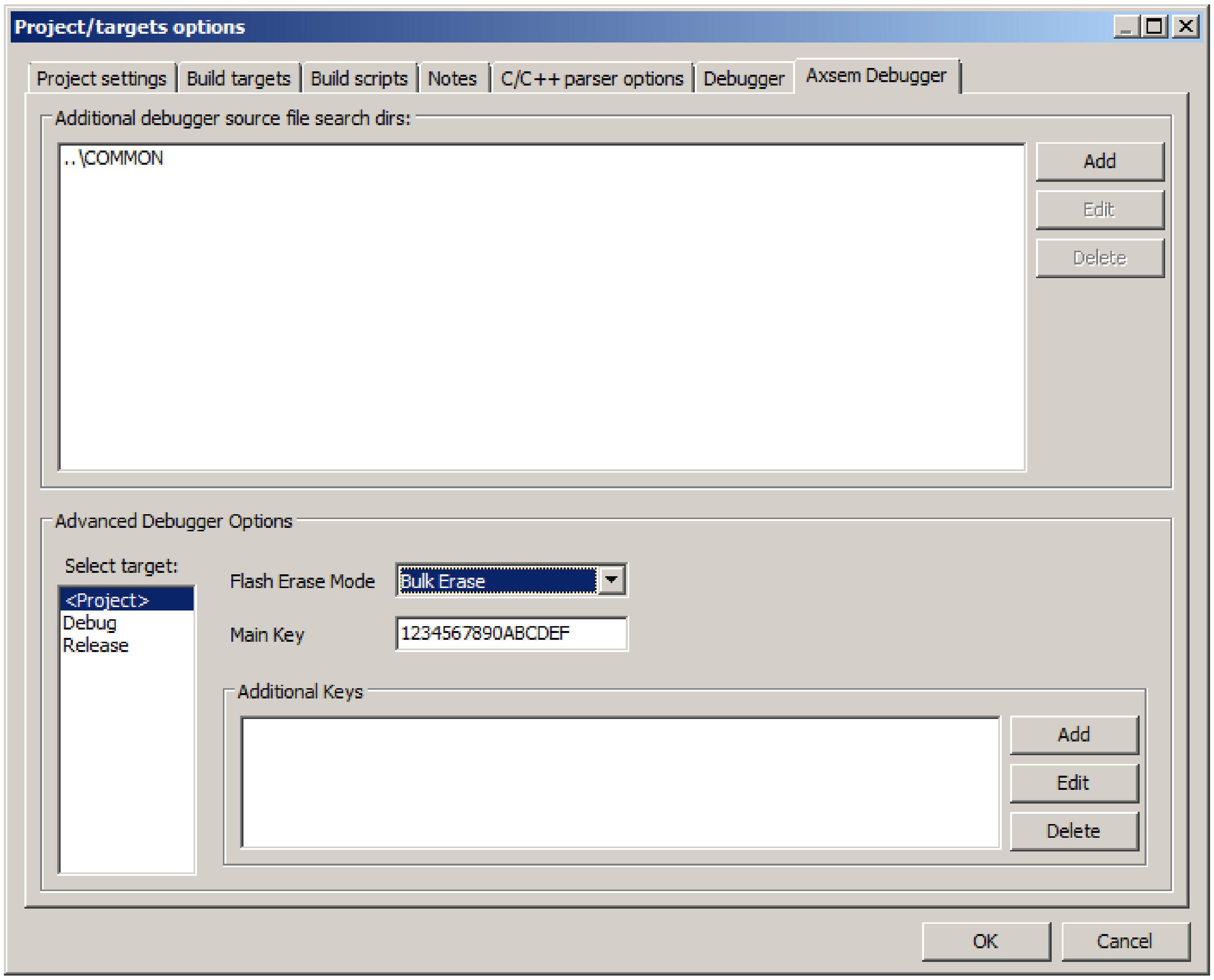
Pin Emulation

This function emulates the two pins of the microcontroller occupied by the debugger. The direction (input or output) is automatically detected. For both directions, the logic state of the pin is shown. Additionally, the state of inputs can be toggled.

Debuglink

This function emulates the two pins of the microcontroller occupied by the debugger. The direction (input or output) is automatically detected. For both directions, the logic state of the pin is shown. Additionally, the state of inputs can be toggled.

Advanced Debugger Configuration



Additional debugger source file search dirs: ..\COMMON

Flash Erase Mode: Bulk Erase

Main Key: 1234567890ABCDEF

Figure 13. Advanced Debugger Configuration – Project / Target Options

The advanced Debugger configuration window can be reached by selecting *Project→Properties…* from the menu bar, and then clicking on the “Axsem Debugger” tab.

The advanced Debugger configuration window can be reached by selecting Project→Properties… from the menu bar, and then clicking on the *“Axsem Debugger”* tab.

Whenever the CPU stops, for example because it hits a breakpoint or the user hits the pause button, the debugger reads the current PC and looks up the file name and line number corresponding to that PC in the debug information. If this file is found neither in the open editor tabs, nor in the project directories, the debugger searches the directories listed under *“Additional debugger source file search dirs”*. It is recommended to add the source and include directories of all used libraries (such as libmf) to the directory list to enable source level debugging even in library code.

Flash Erase Mode specifies the strategy the debugger uses to erase the flash memory before reprogramming. Bulk Erase, the default, sends a bulk erase command to the microcontroller, thus erasing the complete memory. If the debugger knows the device key, it saves and restores the calibration data in the last flash sector. Bulk erase may be issued even without knowing the device key; the calibration data will be lost however. The debugger warns you if it does not know the correct key and asks you to confirm to continue. After a Bulk Erase, the device key is set to the Main Key configured below.

All Sectors Erase instructs the debugger to issue individual page erase commands to all flash sectors that need to be reprogrammed. Flash sectors not needed by the program to be loaded are erased if they contain data. The device key cannot be changed, unless the device was using the default key. This option may be faster than Bulk Erase if only little changes between download cycles.

Needed Sectors Erase instructs the debugger to issue individual page erase commands to all flash sectors that need to be reprogrammed. Flash sectors not needed by the program will be left as-is. The device key cannot be changed, unless the device was using the default key. This option may be the fastest if only little changes between download cycles, however left-over contents are not necessarily cleared.

In order to protect the intellectual property of the customer while still allowing full debugging capability, use of the debug interface is protected by a 64 bit key. The debug interface can only be used if the device key is known. The default key (as shipped by ON Semiconductor, and after a bulk erase) is FFFFFFFFFFFFFFFF. *Main Key* specifies the key the debugger should set to protect the debug interface.

Additional Keys lists keys that are tried as well when connecting the device when the main key does not unlock the device. This is useful if multiple projects use different keys, and devices from other projects should be used. If their keys are listed under Additional Keys, the debugger will be able to retain the calibration data and reprogram the key to the Main Key, when the Flash Erase Mode is set to Bulk Erase.

On Semiconductor Project Wizard

The new project wizard supports the creation of a skeleton project template for ON Semiconductor microcontroller projects.

Both SDCC and IAR compilers are fully supported, with selectable code models.

The code structure of the example project's main.c looks like this:

```
AX5051_commsleepexit();
#if defined(__ICC8051__)
#define coldstart 1
#define warmstart 0
//
// If the code model is banked, low_level_init must be declared
// __near_func elsa a ?BRET is performed
//
#if (__CODE_MODEL__ == 2)
__near_func __root char
#else
__root char
#endif
__low_level_init(void) @ "CSTART"
#else
#define coldstart 0
#define warmstart 1
uint8_t _sdcc_external_startup(void)
#endif
{
   DPS = 0;
   ...
   GPIOENABLE = 1;
   if (PCON & 0x40)
      return warmstart;
   return coldstart;
}
#undef coldstart
#undef warmstart
#if defined(SDCC)
extern uint8_t _start__stack[];
#endif
void main(void)
{
#if !defined(SDCC) && !defined(__ICC8051__)
    _sdcc_external_startup();
#endif
#if defined(SDCC)
    __asm
    G$_start__stack$0$0 = __start__stack
    .globl G$_start__stack$0$0
    __endasm;
#endif
    ...
}
```

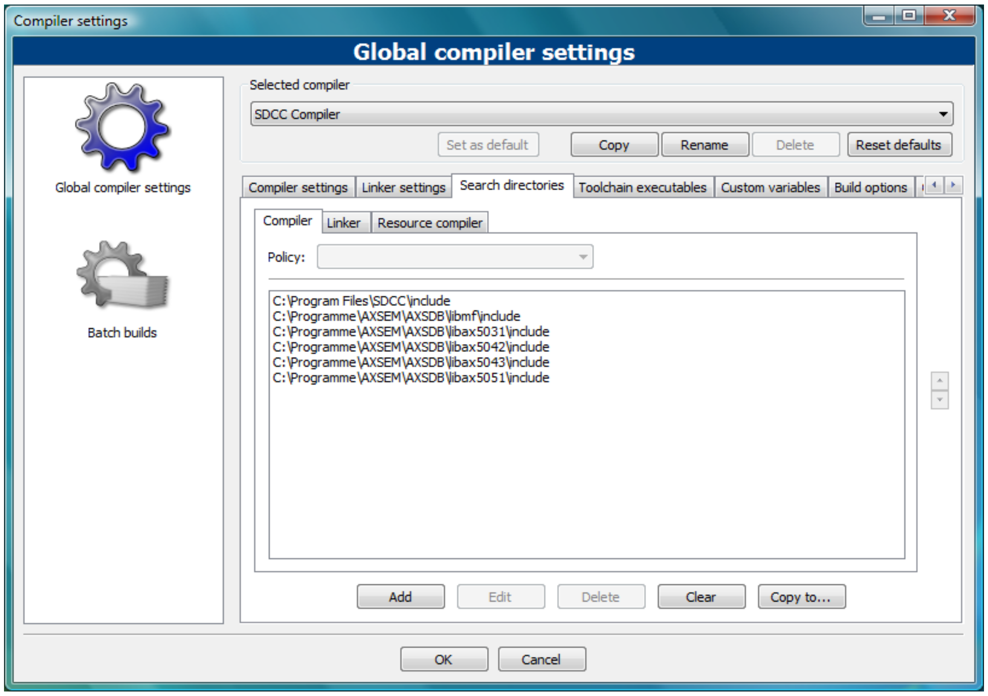
We need to distinguish the different compilers.

- SDCC:
  If available, SDCC arranges for a function *named _sdcc_external_startup* to be called before main.
  The return value of this function determines whether static variables should be initialized. The example code uses this to avoid overwriting static variables on a wake-up from sleep.
  Stack: The two ifdefs *enclosing _start__stack* define a global symbol so that the debugger knows where the stack starts, and can find the call stack.
- IAR:
  If available, SDCC arranges for a function *named _low_level_init* to be called before main.
  The return value of this function determines whether static variables should be initialized. The function needs to reside in segment CSTART and its calling convention is determined by the code model selected.
  The example code uses this to avoid overwriting static variables on a wake-up from sleep.
- Keil:
  Keil does not call any startup routine, its runtime library does not support bypassing static variable initialization. Therefore, the example code calls it explicitly if the compiler is Keil.

Troubleshooting Guide

Compiler Auto-Detection Fails on First Start

When AxCode::Blocks is run for the first time, it tries to auto-detect the installers. If the auto-detection fails, configure the compiler (SDCC) manually as shown in the screen shots below. The path *C:\Program Files* must be replaced by the actual installation path if a non-default installation has been selected.

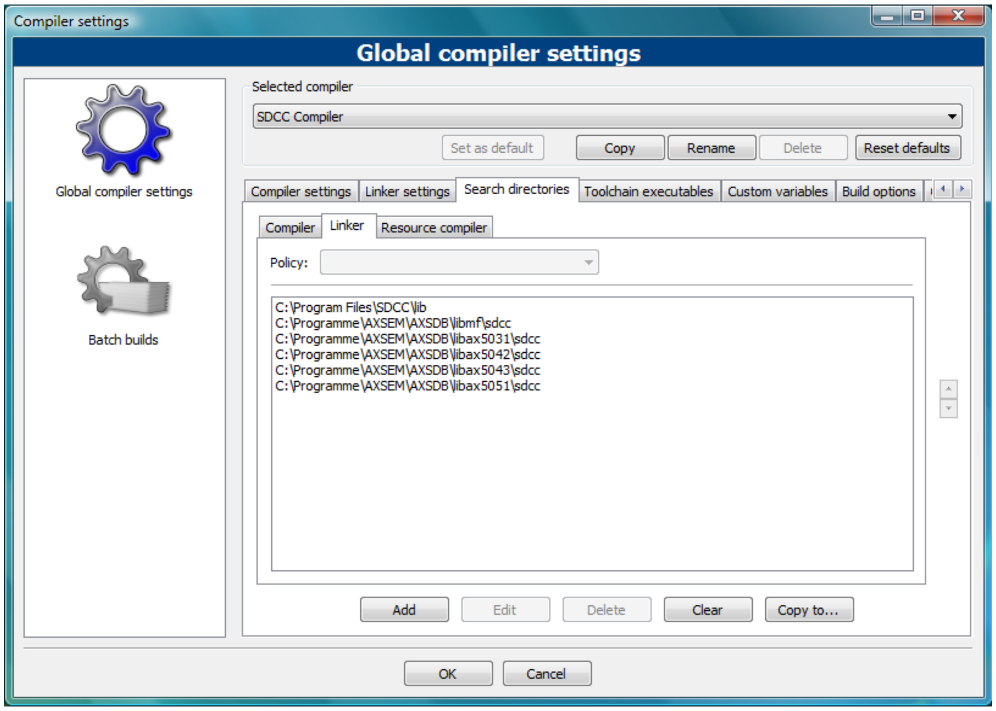


Selected compiler: SDCC Compiler

```
C:\Program Files\SDCC\include
C:\Programme\AXSEM\AXSDB\libmf\include
C:\Programme\AXSEM\AXSDB\libax5031\include
C:\Programme\AXSEM\AXSDB\libax5042\include
C:\Programme\AXSEM\AXSDB\libax5043\include
C:\Programme\AXSEM\AXSDB\libax5051\include
```

Figure 14. Global Compiler Settings – Search Directories

Open the compiler settings dialog by clicking on *Settings →Compiler*. Select SDCC (under selected compiler). Click on “Set as default”. Click on “Search Directories” and “Compiler”, and enter the directories as in the picture above.



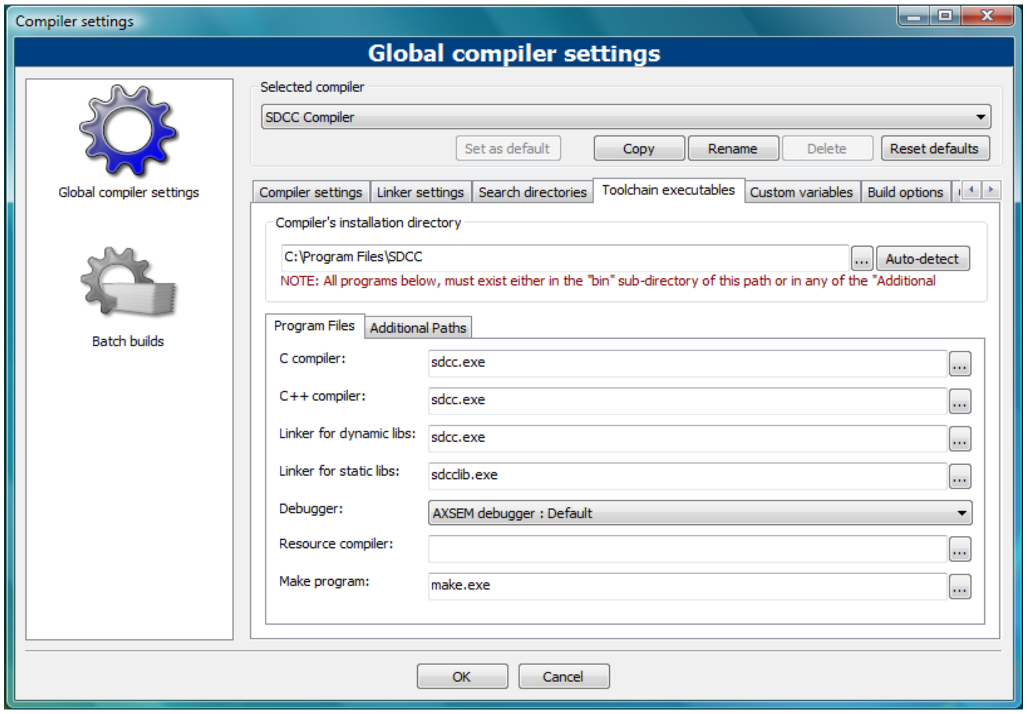
Selected compiler: SDCC Compiler

Policy:

```
C:\Program Files\SDCC\lib
C:\Programme\AXSEM\AXSDB\libmf\sdcc
C:\Programme\AXSEM\AXSDB\libax5031\sdcc
C:\Programme\AXSEM\AXSDB\libax5042\sdcc
C:\Programme\AXSEM\AXSDB\libax5043\sdcc
C:\Programme\AXSEM\AXSDB\libax5051\sdcc
```

Figure 15. Global Compiler Settings – Search Directories (Linker)

Click on “Linker”, and enter the directories as in the picture above.



Selected compiler: SDCC Compiler

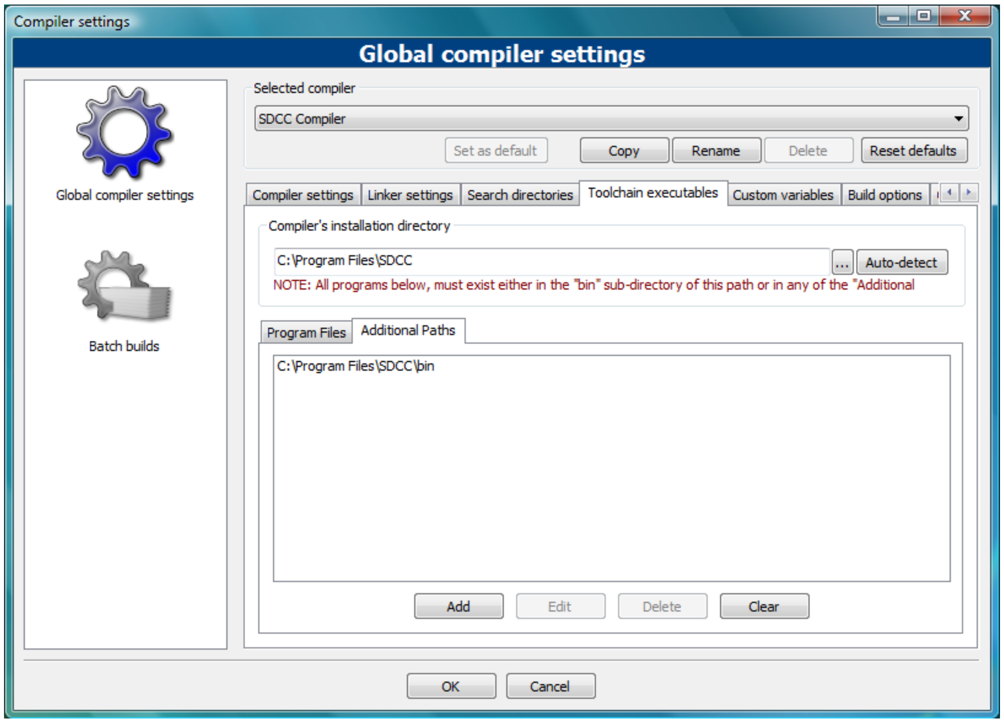
Compiler's installation directory: C:\Program Files\SDCC

NOTE: All programs below, must exist either in the "bin" sub-directory of this path or in any of the "Additional

C compiler:	sdcc.exe
C++ compiler:	sdcc.exe
Linker for dynamic libs:	sdcc.exe
Linker for static libs:	sdcclib.exe
Debugger:	AXSEM debugger : Default
Resource compiler:	
Make program:	make.exe

Figure 16. Global Compiler Settings – Toolchain Executables

Click on “Toolchain Executables”, and verify the “Compiler’s Installation directory”.



Selected compiler: SDCC Compiler

Compiler's installation directory: C:\Program Files\SDCC

NOTE: All programs below, must exist either in the "bin" sub-directory of this path or in any of the "Additional

C:\Program Files\SDCC\bin

Figure 17. Global Compiler Settings – Toolchain Executables (Additional Paths)

Click on “Additional Paths”, and enter the path as shown above.

Remove All Saved User Settings

Open an explorer. Type “%APPDATA%” (without double quotes) into the address bar. Type enter. The directory now displayed should contain a folder (subdirectory) named “axCodeBlocks”. This is where AxCode::Blocks stores per-user configuration settings. Delete that subdirectory.

SDCC Project Does Not Compile

First, make sure that AxCode::Blocks finds the compiler executable. Open the compiler settings window.

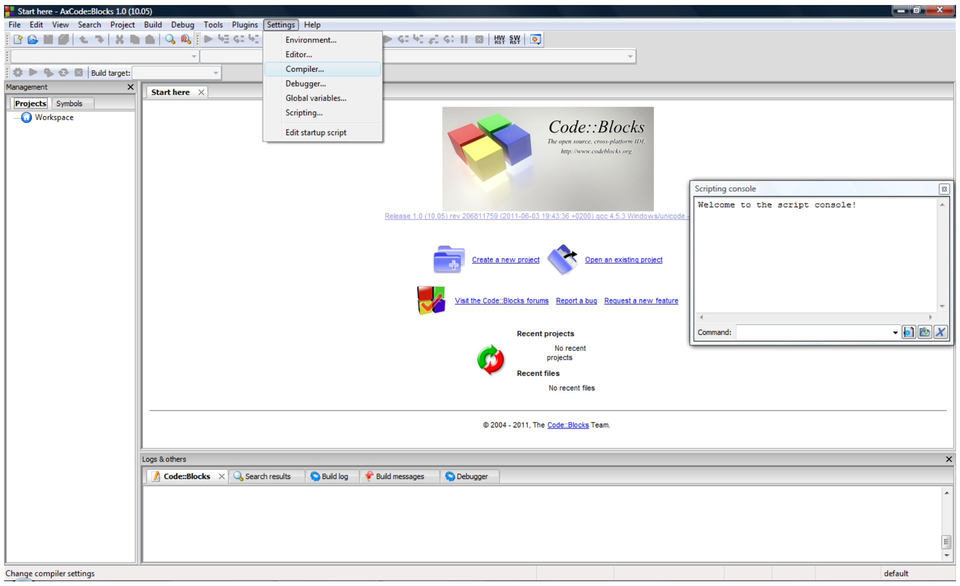
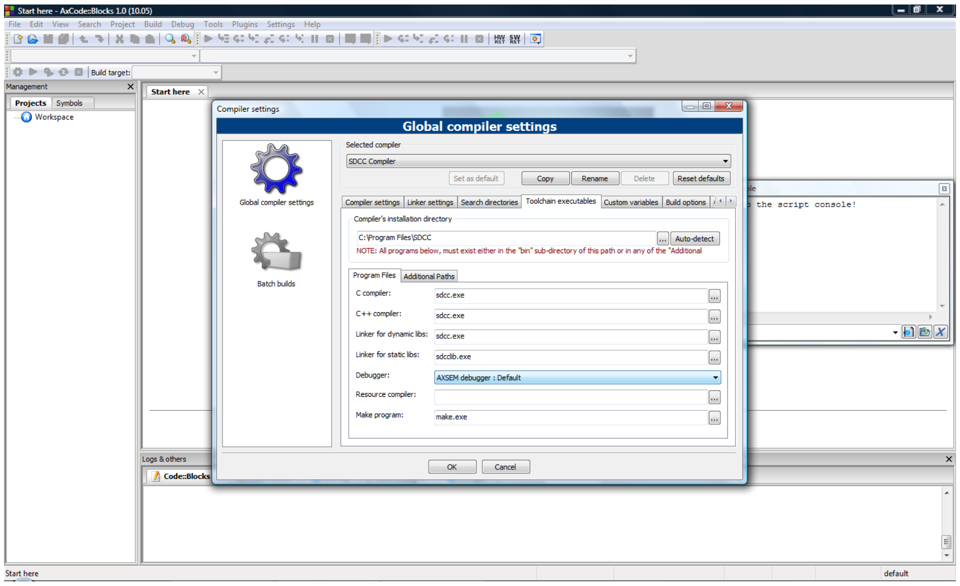


Figure 18. AxCode::Blocks 1.0 – Settings

Check that SDCC is the default compiler and that the Compiler’s Installation directory points to SDCC’s installation location. Normally, this is automatically set up correctly, but if you manually move the SDCC directory, or re-install it at another location, you must manually update the location.



Selected compiler: SDCC Compiler

Compiler's installation directory: C:\Program Files\SDCC

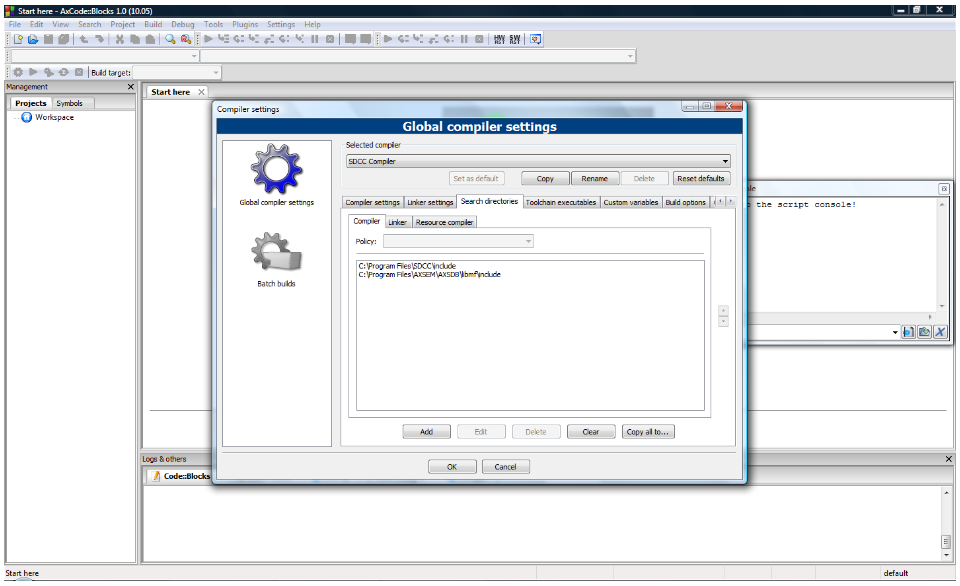
NOTE: All programs below, must exist either in the "bin" sub-directory of this path or in any of the "Additional

C compiler:	sdcc.exe
C++ compiler:	sdcc.exe
Linker for dynamic libs:	sdcc.exe
Linker for static libs:	sdcclib.exe
Debugger:	AXSEM debugger : Default
Resource compiler:	
Make program:	make.exe

Figure 19. AxCode::Blocks 1.0 – Global Compiler Settings

Make sure that SDCC finds all required header files. Check that the compiler’s default include paths contain SDCC’s own includes, as well as the AX8052 convenience library (such as libmf) includes.

Check that the linker’s default paths contain SDCC’s own libraries path, as well as the AX8052 convenience library (such as libmf) path.

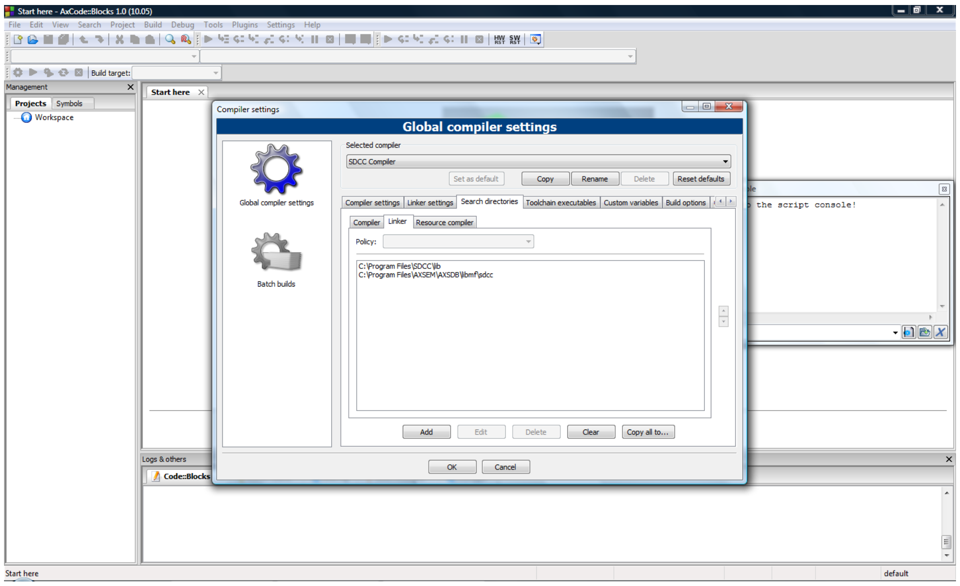


Selected compiler: SDCC Compiler

Policy:

```
C:\Program Files\SDCC\include
C:\Program Files\AXSEM\AXSDB\libmf\include
```

Figure 20. AxCode::Blocks 1.0 – Global Compiler Settings (Compiler)



Selected compiler: SDCC Compiler

Policy:

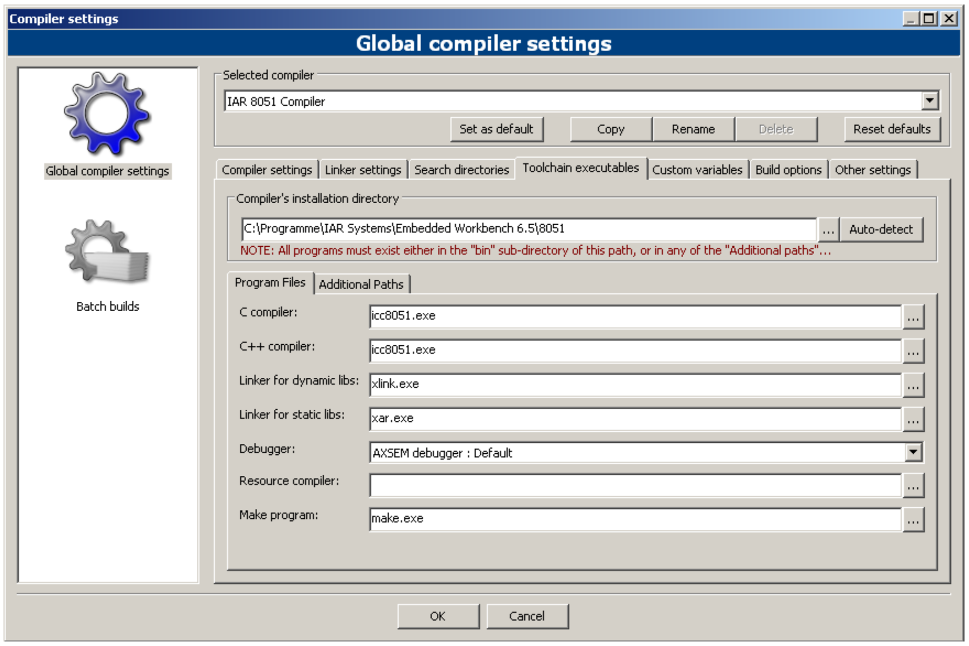
```
C:\Program Files\SDCC\lib
C:\Program Files\AXSEM\AXSDB\libmf\sdcc
```

Figure 21. AxCode::Blocks 1.0 – Global Compiler Settings (Linker)

IAR Project Does Not Compile

First, make sure that AxCode::Blocks finds the compiler executable. Open the compiler settings window. Then select “IAR 8051 Compiler” in the “Selected Compiler” combobox.

Check that the Compiler’s Installation directory points to IAR’s installation location. Normally, this is automatically set up correctly, but if you manually move the IAR directory, or re-install it at another location, or update it to a new version, you must manually update the location.



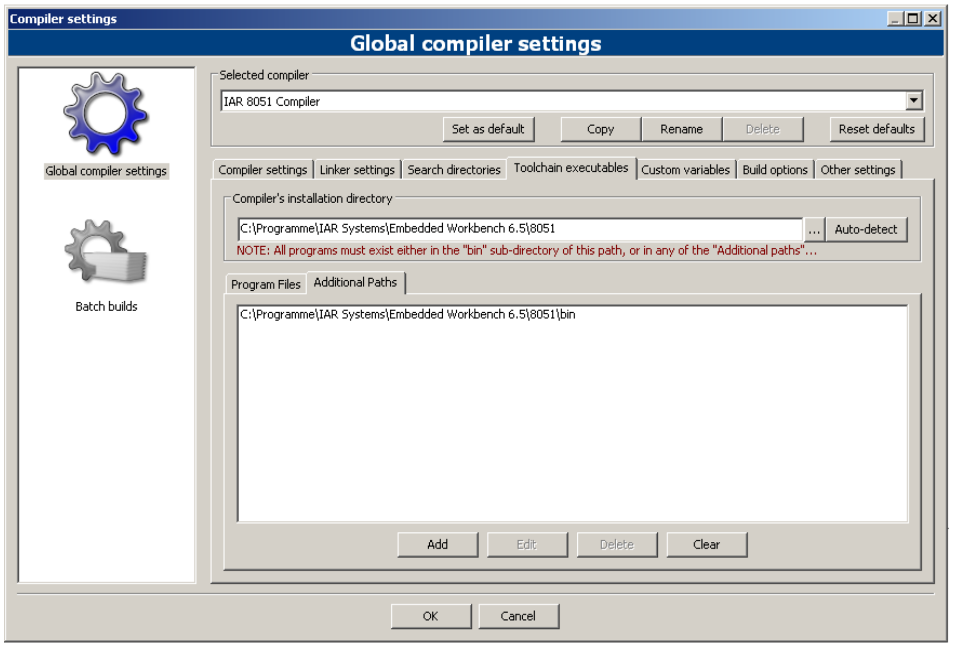
Selected compiler: IAR 8051 Compiler

Compiler's installation directory: C:\Programme\IAR Systems\Embedded Workbench 6.5\8051

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional paths"...

C compiler:	icc8051.exe
C++ compiler:	icc8051.exe
Linker for dynamic libs:	xlink.exe
Linker for static libs:	xar.exe
Debugger:	AXSEM debugger : Default
Resource compiler:	
Make program:	make.exe

Figure 22. Global Compiler Settings – Toolchain Executables (Program Files)



Selected compiler: IAR 8051 Compiler

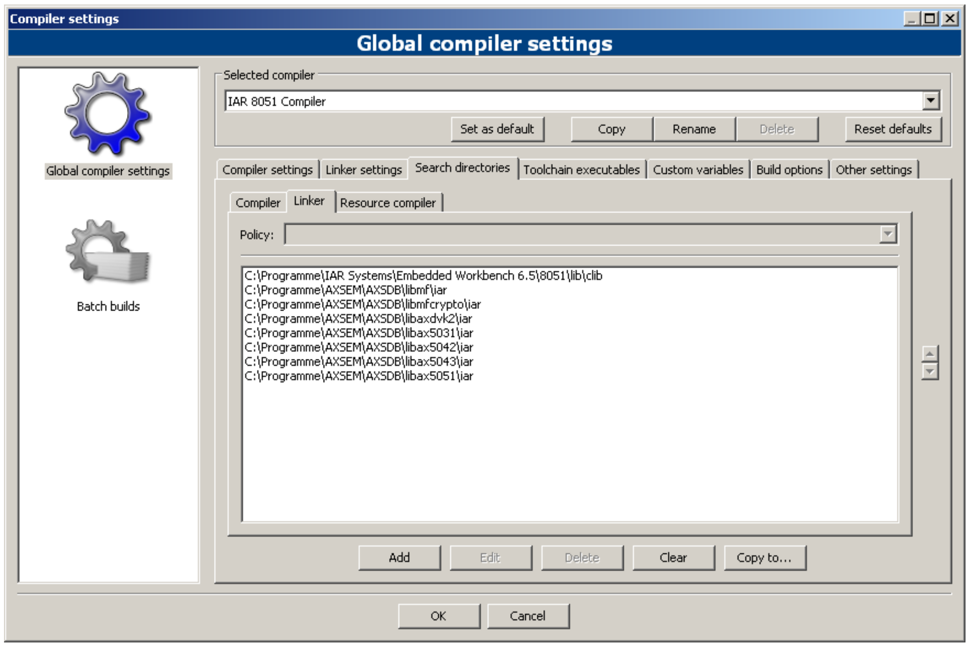
Compiler's installation directory: C:\Programme\IAR Systems\Embedded Workbench 6.5\8051

NOTE: All programs must exist either in the "bin" sub-directory of this path, or in any of the "Additional paths"...

Additional Paths: C:\Programme\IAR Systems\Embedded Workbench 6.5\8051\bin

Figure 23. Global Compiler Settings – Toolchain Executables (Additional Paths)

Make sure that IAR finds all required header files. Check that the compiler’s default include paths contain IAR’s own includes, as well as the AX8052 convenience library (such as libmf) includes.

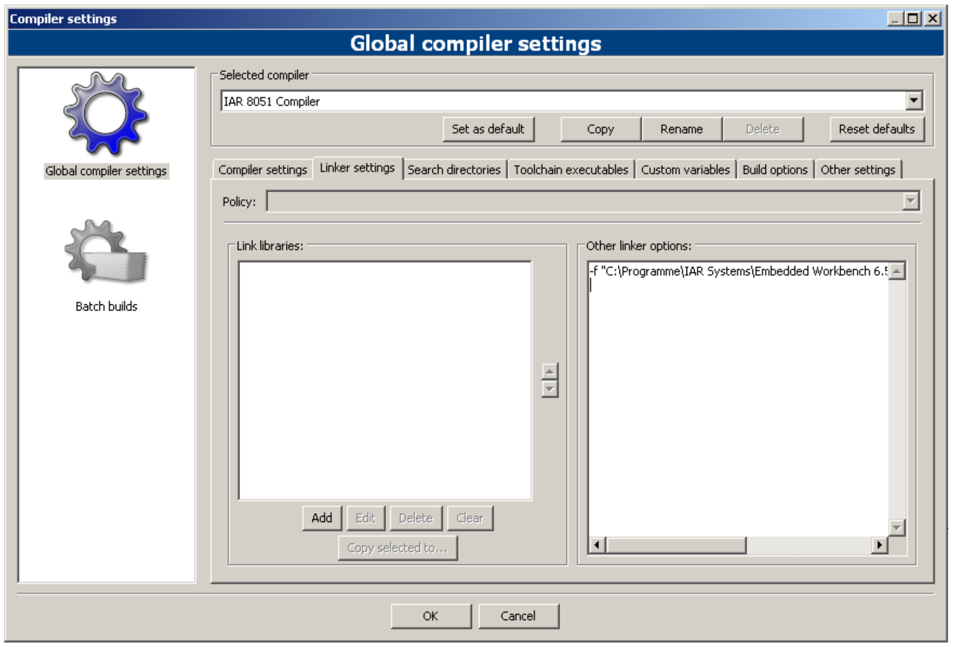


Selected compiler: IAR 8051 Compiler

```
C:\Programme\IAR Systems\Embedded Workbench 6.5\8051\lib\clib
C:\Programme\AXSEM\AXSDB\libmf\iar
C:\Programme\AXSEM\AXSDB\libmfcrypto\iar
C:\Programme\AXSEM\AXSDB\libaxdvk2\iar
C:\Programme\AXSEM\AXSDB\libax5031\iar
C:\Programme\AXSEM\AXSDB\libax5042\iar
C:\Programme\AXSEM\AXSDB\libax5043\iar
C:\Programme\AXSEM\AXSDB\libax5051\iar
```

Figure 24. Global Compiler Settings – Search Directories (Linker)

Check that the linker's default paths contain IAR's own libraries path, as well as the AX8052 convenience library (such as libmf) path.

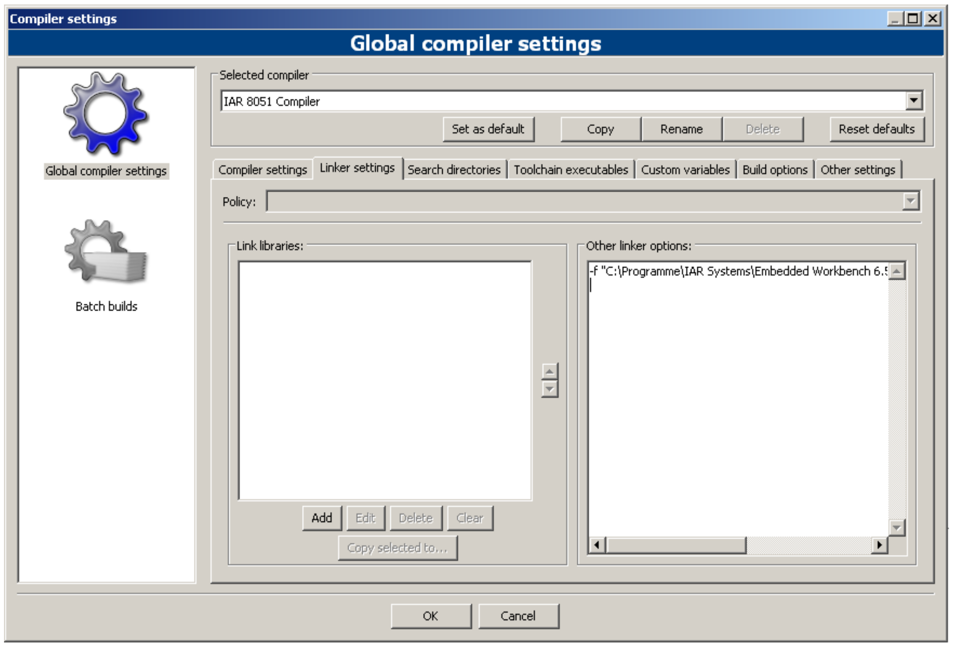


Selected compiler: IAR 8051 Compiler

Other linker options:

```
-f "C:\Programme\IAR Systems\Embedded Workbench 6.
```

Figure 25. Global Compiler Settings – Linker Settings



Selected compiler: IAR 8051 Compiler

Other linker options:

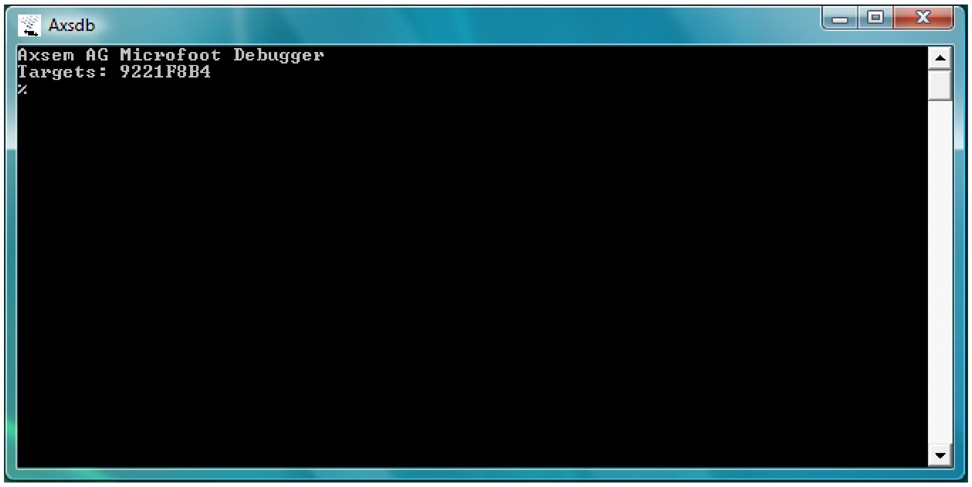
```
-f "C:\Programme\IAR Systems\Embedded Workbench 6.5
```

Figure 26. Global Compiler Settings – Linker Settings

Project Compiles, but Debugging Does Not Work

First check whether the Axsem Command Line Debugger, AXSDB, works. Start AXSDB (the installer places a link into the program section of the start menu).

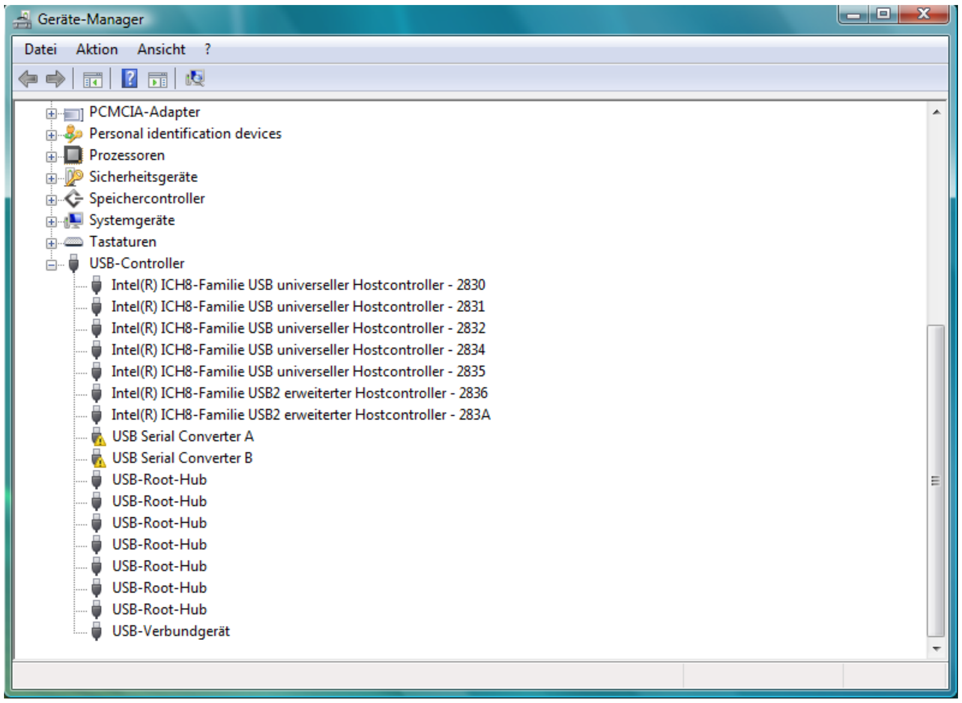
AXSDB should print the serial number of a target after the “Targets:” prompt. The top two reasons for an empty line after “Targets:” are that no debug adapter is connected to a working USB port, or that the USB drivers have not been correctly installed.



```
Axsem AG Microfoot Debugger
Targets: 9221F8B4
%
```

Figure 27. Axsdb

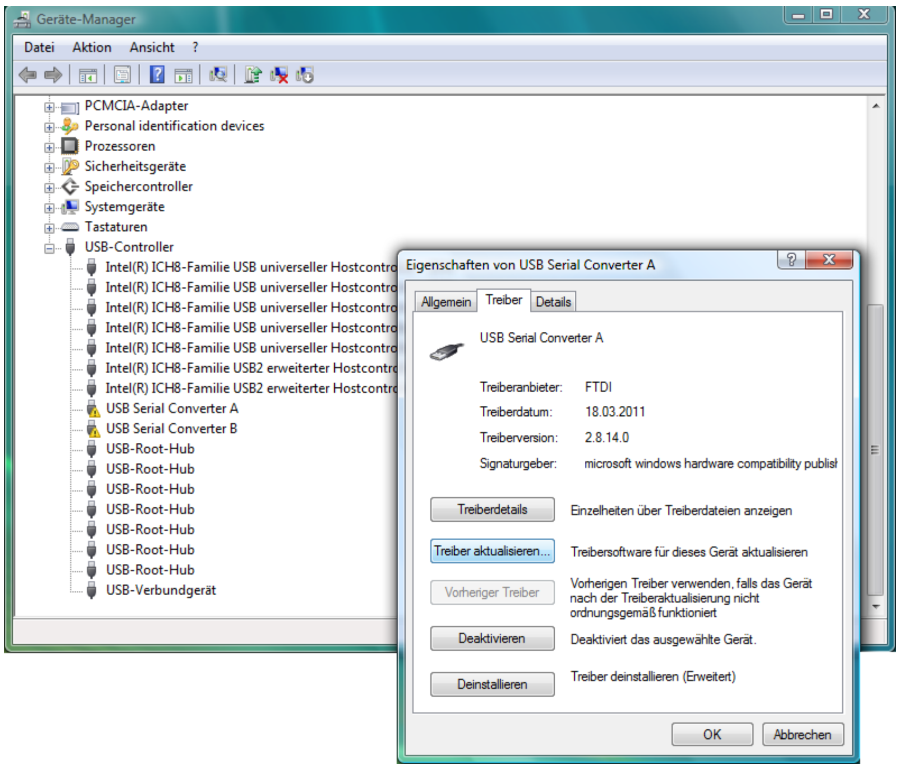
To check whether the drivers are correctly installed, open the device manager.



```
PCMCIA-Adapter
Personal identification devices
Prozessoren
Sicherheitsgeräte
Speichercontroller
Systemgeräte
Tastaturen
USB-Controller
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2830
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2831
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2832
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2834
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2835
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontroller - 2836
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontroller - 283A
    USB Serial Converter A
    USB Serial Converter B
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Verbundgerät
```

Figure 28.

Yellow exclamation marks on “USB Serial Converter A” or “... B” or “Microfoot Debug Adapter V1.00” indicate that the driver has not been correctly installed. Right-click on the device with the exclamation mark, and choose Reinstall driver.



```
PCMCIA-Adapter
Personal identification devices
Prozessoren
Sicherheitsgeräte
Speichercontroller
Systemgeräte
Tastaturen
USB-Controller
    Intel(R) ICH8-Familie USB universeller Hostcontro
    Intel(R) ICH8-Familie USB universeller Hostcontro
    Intel(R) ICH8-Familie USB universeller Hostcontro
    Intel(R) ICH8-Familie USB universeller Hostcontro
    Intel(R) ICH8-Familie USB universeller Hostcontro
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontro
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontro
    USB Serial Converter A
    USB Serial Converter B
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Verbundgerät
```

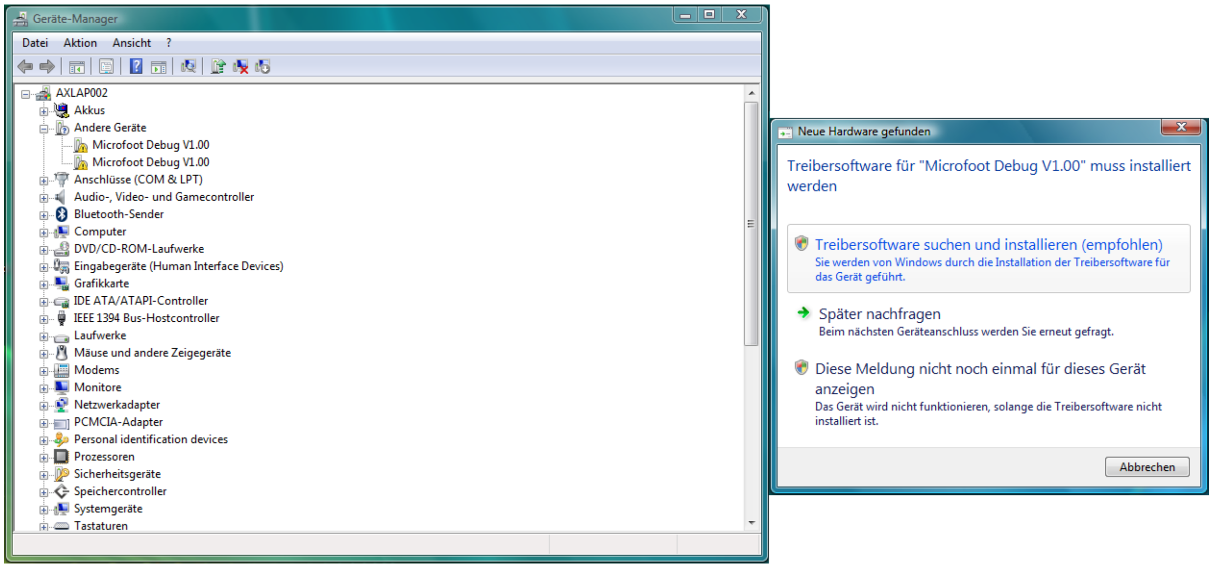
USB Serial Converter A

Treiberanbieter:	FTDI
Treiberdatum:	18.03.2011
Treiberversion:	2.8.14.0
Signaturgeber:	microsoft windows hardware compatibility publis

Treiberdetails	Einzelheiten über Treiberdateien anzeigen
Treiber aktualisieren...	Treibersoftware für dieses Gerät aktualisieren
Vorheriger Treiber	Vorherigen Treiber verwenden, falls das Gerät nach der Treiberaktualisierung nicht ordnungsgemäß funktioniert
Deaktivieren	Deaktiviert das ausgewählte Gerät.
Deinstallieren	Treiber deinstallieren (Erweitert)

Figure 29.

The first time a Debug Adapter is installed, Windows may prompt for the installation of a driver; choose to install a driver now.



AXLAP002
Akkus
Andere Geräte
Microfoot Debug V1.00
Microfoot Debug V1.00
Anschlüsse (COM & LPT)
Audio-, Video- und Gamecontroller
Bluetooth-Sender
Computer
DVD/CD-ROM-Laufwerke
Eingabegeräte (Human Interface Devices)
Grafikkarte
IDE ATA/ATAPI-Controller
IEEE 1394 Bus-Hostcontroller
Laufwerke
Mäuse und andere Zeigegeräte
Modems
Monitore
Netzwerkadapter
PCMCIA-Adapter
Personal identification devices
Prozessoren
Sicherheitsgeräte
Speichercontroller
Systemgeräte
Tastaturen

Treibersoftware für "Microfoot Debug V1.00" muss installiert werden

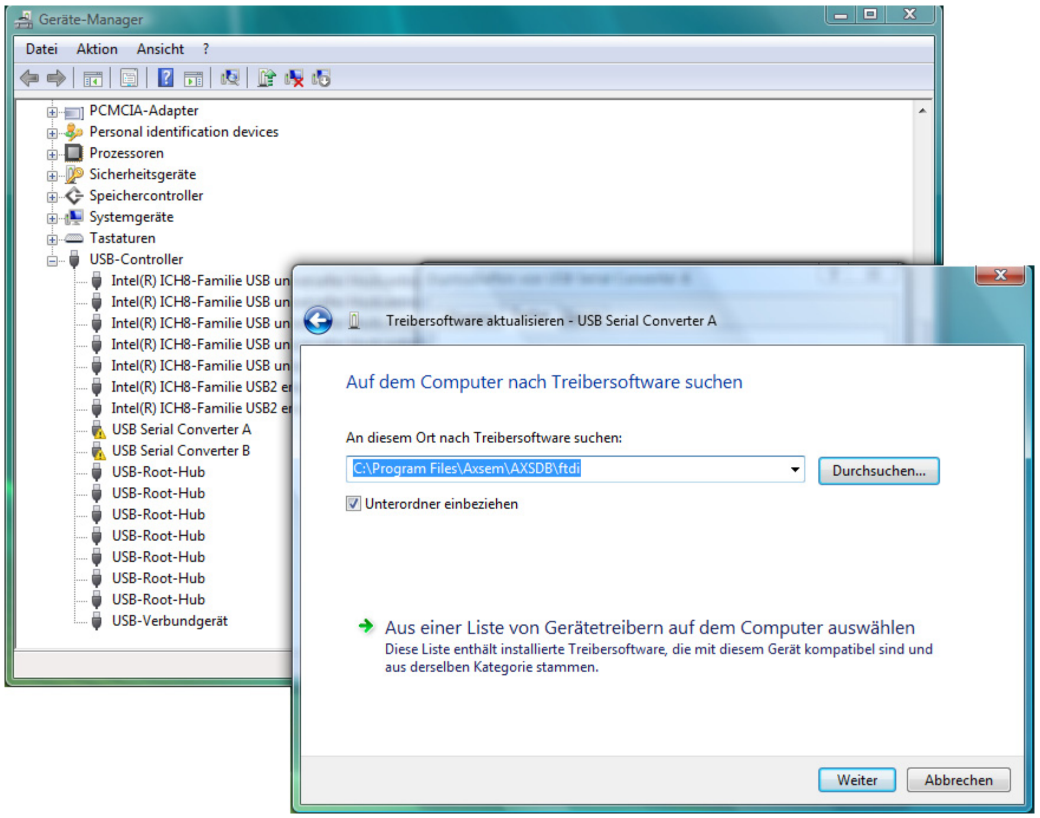
Treibersoftware suchen und installieren (empfohlen)
Sie werden von Windows durch die Installation der Treibersoftware für das Gerät geführt.

Später nachfragen
Beim nächsten Geräteanschluss werden Sie erneut gefragt.

Diese Meldung nicht noch einmal für dieses Gerät anzeigen
Das Gerät wird nicht funktionieren, solange die Treibersoftware nicht installiert ist.

Figure 30.

Newer Versions of Windows (starting with XP), if connected to the internet, offer the option to search Windows Update for a suitable driver. You can choose this option; it is the easiest option, though may take some time. As an alternative, the installer also puts suitable drivers into C:\Program Files\Axsem\AXSDB\ftdi. You can choose to install from this directory and its subdirectories as well.



PCMCIA-Adapter
Personal identification devices
Prozessoren
Sicherheitsgeräte
Speichercontroller
Systemgeräte
Tastaturen
USB-Controller
Intel(R) ICH8-Familie USB un
Intel(R) ICH8-Familie USB un
Intel(R) ICH8-Familie USB un
Intel(R) ICH8-Familie USB un
Intel(R) ICH8-Familie USB un
Intel(R) ICH8-Familie USB2 e
Intel(R) ICH8-Familie USB2 e
USB Serial Converter A
USB Serial Converter B
USB-Root-Hub
USB-Root-Hub
USB-Root-Hub
USB-Root-Hub
USB-Root-Hub
USB-Root-Hub
USB-Root-Hub
USB-Verbundgerät

Treibersoftware aktualisieren - USB Serial Converter A

Auf dem Computer nach Treibersoftware suchen

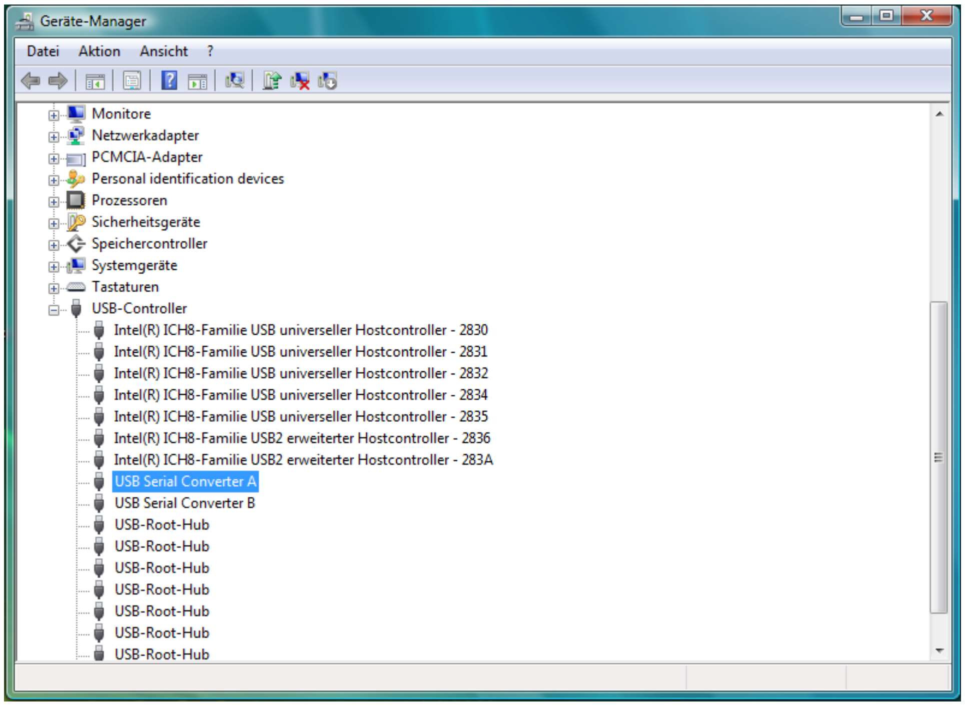
An diesem Ort nach Treibersoftware suchen:
C:\Program Files\Axsem\AXSDB\ftdi

☑ Unterordner einbeziehen

Aus einer Liste von Gerätetreibern auf dem Computer auswählen
Diese Liste enthält installierte Treibersoftware, die mit diesem Gerät kompatibel sind und aus derselben Kategorie stammen.

Figure 31.

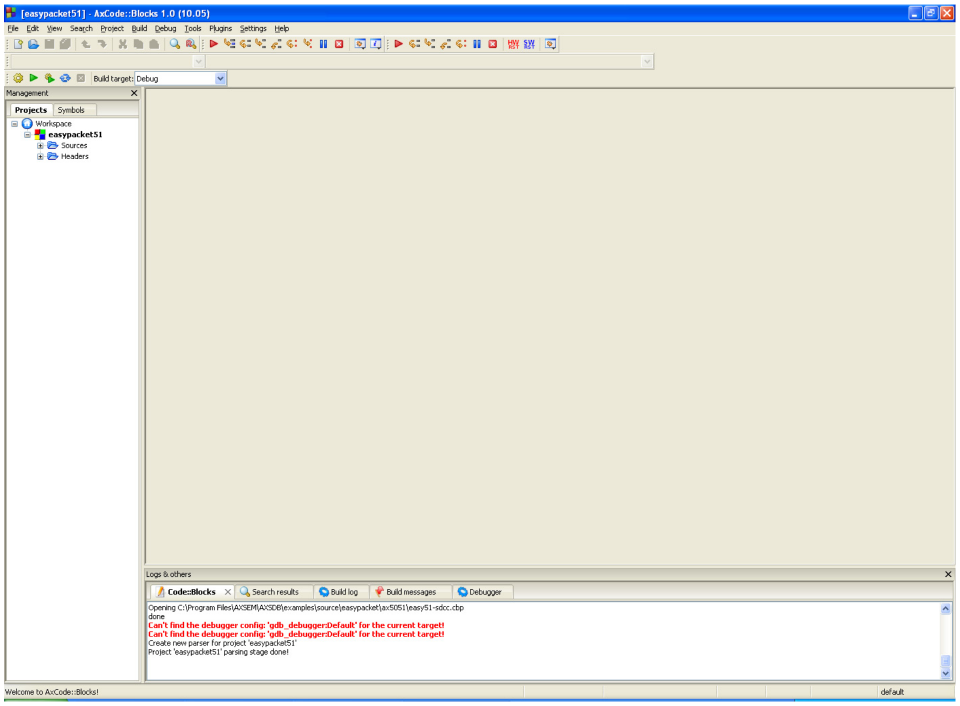
After successfully re-installing the drivers, the device manager should look as follows. After installing the drivers, you should reboot Windows.



```
Monitore
Netzwerkadapter
PCMCIA-Adapter
Personal identification devices
Prozessoren
Sicherheitsgeräte
Speichercontroller
Systemgeräte
Tastaturen
USB-Controller
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2830
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2831
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2832
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2834
    Intel(R) ICH8-Familie USB universeller Hostcontroller - 2835
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontroller - 2836
    Intel(R) ICH8-Familie USB2 erweiterter Hostcontroller - 283A
    USB Serial Converter A
    USB Serial Converter B
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
    USB-Root-Hub
```

Figure 32.

The following error means that the debugger plugin is not correctly configured. See below for how to correct this error.

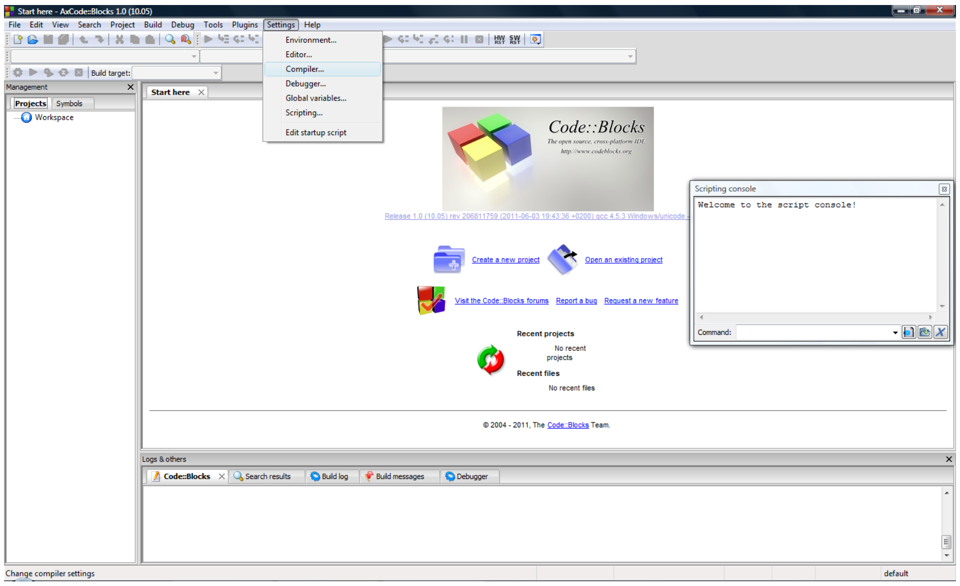


```
Opening C:\Program Files\AXSEM\AXSDB\examples\source\easypacket\ax5051\easy51-sdcc.cbp
done
Can't find the debugger config: 'gdb_debugger:Default' for the current target!
Can't find the debugger config: 'gdb_debugger:Default' for the current target!
Create new parser for project 'easypacket51'
Project 'easypacket51' parsing stage done!
```

Figure 33. AxCode::Blocks 1.0

Check that AXSDB now recognizes the target.

Start AxCode::Blocks. Verify that the correct debugger plugin is selected. To do this, open the compiler settings window.



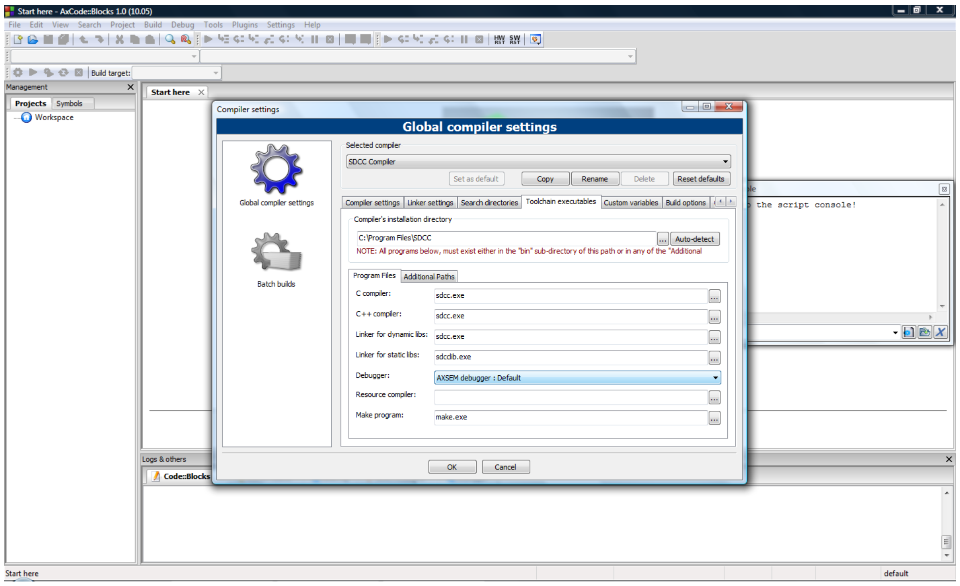
```
Welcome to the script console!
```

Release 1.0 (10.05) rev 206811759 (2011-06-03 19:43:36 +0200) gcc 4.5.3 Windows/unicode

© 2004 - 2011, The Code::Blocks Team.

Figure 34. AxCode::Blocks 1.0

Check that the debugger plugin named “AXSEM debugger : Default” is selected.



Selected compiler: SDCC Compiler

Compiler's installation directory: C:\Program Files\SDCC

NOTE: All programs below, must exist either in the "bin" sub-directory of this path or in any of the "Additional

C compiler:	sdcc.exe
C++ compiler:	sdcc.exe
Linker for dynamic libs:	sdcc.exe
Linker for static libs:	sdcclib.exe
Debugger:	AXSEM debugger : Default
Resource compiler:	
Make program:	make.exe

Figure 35. AxCode:Blocks 1.0 – Global Compiler Settings

Now check that the correct axsdb binary is configured in the plugin configuration. To do this, open the debugger settings.

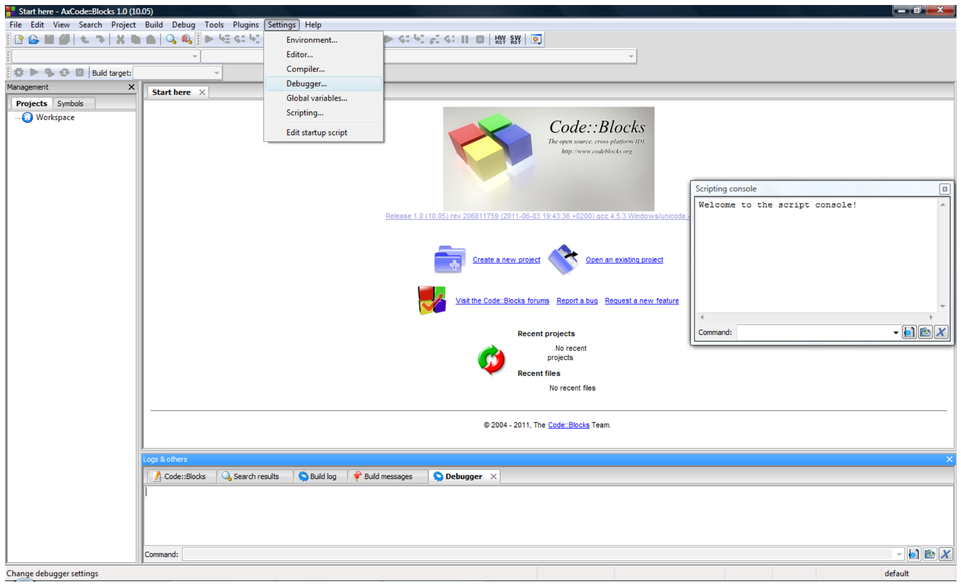
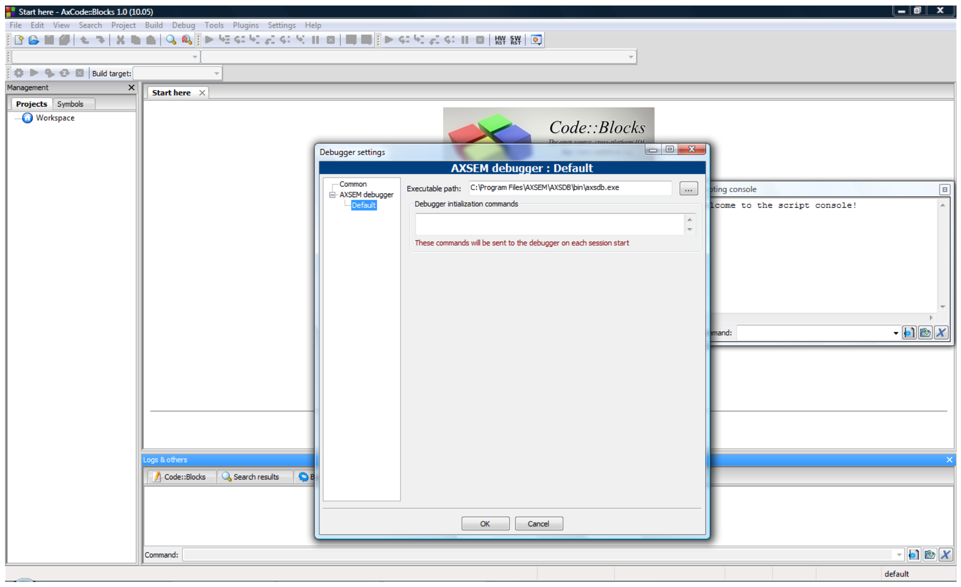


Figure 36. AxCode:Blocks 1.0 – Settings

Select AXSEM debugger – default. Check that the Executable path is correct.



Executable path: C:\Program Files\AXSEM\AXSDB\bin\axsdb.exe

These commands will be sent to the debugger on each session start

Figure 37. AxCode:Blocks 1.0 – Axsem Debugger : Default

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
19521 E. 32nd Pkwy, Aurora, Colorado 80011 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81−3−5817−1050

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative