



# Quick Start Guide

ISIS-IOBC-MAN-0001

Version: 1.0

CI Number: N/A

## Release Information

	Name	Function	Signature	Date
Prepared by:	R. Fernandez	Documentation Manager		
Reviewed by:	A. H. Sauge	Software Engineer		
Approved by:	H. Y. Oei	COO		
Authorized by:	H. Y. Oei	COO		



# Quick Start Guide

Doc ID: ISIS-IOBC-MAN-0001  
Issue: 1.0  
Date: 2023-06-06  
Page: 2 of 30

## Distribution List

Name	Organization	Description
N/A	ISISPACE	Innovative Solutions In Space B.V.

## Disclaimer

The contents of this document are subject to the relevant provisions in the contract concluded between the parties. ISISPACE Group ("ISISPACE") shall not be liable, in full or in part, for any damage arising out from the application or use of any product or circuit described herein, in case such application or use are carried out in a manner not in line with the instructions and warranties provided in the User Manual, Safety Manual, product information sheets or any other document provided by ISISPACE upon the delivery of the product ("Documents"). Further, ISISPACE shall not be liable for any damage caused by any use which exceeds the function(s) of the product, or does not conform to such function(s) as described in the Documents. ISISPACE shall not be liable for any damage arising from a use which is not carried out in a manner conforming to acceptable practices in the aerospace industry.

ISISPACE warrants that the product is supplied after relevant tests had shown the product is in good order and functioning, as far as these tests may indicate and predict product functionality.



# Quick Start Guide

Doc ID: ISIS-IOBC-MAN-0001  
Issue: 1.0  
Date: 2023-06-06  
Page: 3 of 30

## Change Log

Version	Date	Affects	Description
1.0	2023-06-06	All	First Version. Document created from ISIS-OBC Quickstart Guide version 2.2 and ISIS.ISIS-OBC.DS.01 version 1.3.



# Quick Start Guide

Doc ID: ISIS-IOBC-MAN-0001  
Issue: 1.0  
Date: 2023-06-06  
Page: 4 of 30

## Table of Contents

1	Introduction.....	7
1.1	Applicable Documents.....	7
1.2	Reference Documents.....	7
1.3	Specification.....	7
1.4	Embedded Systems Software Design.....	8
1.5	Deliverable hardware.....	9
2	Handling and Storage.....	10
2.1	Handling.....	10
2.1.1	Electrostatic discharge ESD.....	10
2.1.2	Exposed voltages.....	10
2.1.3	Current Limit Protection.....	10
2.1.4	Operation Conditions.....	10
2.2	Storage.....	11
2.3	Disposal.....	11
3	Detailed Description of Operations.....	12
3.1	Setting up the Integrated Development Environment.....	12
3.1.1	Install Programmer Hardware Driver: J-Link from Segger.....	12
3.1.2	Install USB-to-Serial Hardware Driver: FTDI.....	12
3.1.3	Install iOBC SDK.....	12
3.1.4	Extract iOBC First-project.....	13
3.1.5	Extract iOBC Libraries.....	13
3.2	Setting-up the Hardware.....	14
3.3	Configuring Eclipse IDE.....	15
3.3.1	Import the 'isis-obc-first-project' into Eclipse.....	16
3.3.2	Optional: Configure the J-Link GDB Server Configuration.....	18
3.3.3	Check/Add the Debug Launch Configuration.....	20
3.4	Code Development and Debug/Release Builds.....	21
3.4.1	Compiling Code.....	21
3.4.2	iOBC First-project as Template.....	21
3.4.3	Eclipse Build Configuration.....	24
3.5	Running your Program.....	25
3.5.1	Debugging your Code.....	25
3.6	Flashing Your Code.....	28
3.6.1	iOBC Bootloader.....	29
3.6.2	Backing-Up and Restoring Flashed Code.....	29
4	Support.....	30

© 2023. All rights reserved. Disclosure to third parties of this document or any part thereof, or the use of any information contained therein for purposes other than provided for by this document, is not permitted except with express written permission of ISISPACE Group.

**COMMERCIAL IN CONFIDENCE**

## Figures

Figure 1 – Non-optional IOBC deliverables (daughterboard is optional).....	9
Figure 2 - SEGGER SAM-ICE Programmer/Debugger Atmel OEM version .....	12
Figure 3 - The IOBC SDK installer.....	13
Figure 4 - Final file structure of the C:\ISIS\ISIS-OBC\ main project folder. Keeping the zip files is optional.....	14
Figure 5 - IOBC embedded software development hardware configuration .....	15
Figure 6 - Eclipse IDE welcome screen (left) and project screen (right) shown after closing the welcome screen .....	15
Figure 7 - Select Eclipse Project Explorer import function.....	16
Figure 8 - (import ...) Select General > 'Existing Projects into Workspace' .....	17
Figure 9 - isis-obc-first-project and any library demo projects are selected for import. IMPORTANT: 'Copy projects into workspace' should be deselected, to make sure the project source files are not copied into the workspace.....	17
Figure 10 - Verify/Add the GDB launch configuration .....	19
Figure 11 - Opening Eclipse external tools configuration window .....	19
Figure 12 - Update/Verify that the correct path to the GDB server executable is in place for the 'JLink-samg20-GDB' launch configuration.....	20
Figure 13: Adding the debug launch configuration .....	20
Figure 14 - isis-obc-first-project opened in the Eclipse IDE, with the red arrow indicating the Build icon .....	21
Figure 15 - Relative file locations. The library directory should be in the same directory as the project.....	22
Figure 16 - Satellite-subsystems library added to the directory. ....	22
Figure 17 - Adding the library include path .....	23
Figure 18 - Add the library to the linker settings .....	23
Figure 19 - Setting the build configuration .....	24
Figure 20 - Putty configuration window.....	26
Figure 21 - Retrieve the COM port assigned to the USB-to-Serial from the device manager .....	26
Figure 22 - Start Debug session .....	27
Figure 23 - SAM-BA program.....	28

## Tables

Table 1 - Applicable Documents .....	7
Table 2 - Reference Documents .....	7
Table 3 - General Specifications .....	7
Table 4 – Non-optional IOBC deliverables .....	9

## Acronyms

Name	Description
1/2/3U	1-Unit, 2-Unit, 3-Unit; commonly referring to the singles and multiples of the commercially available CubeSat sizes
ANTS	ISIS Deployable Antenna System
ANTS-EM	Electrical model of the ISIS Deployable Antenna System without RF or mechanical function
AIV	Assembly, integration, verification
COTS	Commercial off the shelf
COM	Centre of Mass
EGSE	Electrical Ground Support Equipment
EMC	Electro-Magnetic Compatibility



## Quick Start Guide

Doc ID: ISIS-IOBC-MAN-0001

Issue: 1.0

Date: 2023-06-06

Page: 6 of 30

Name	Description
EMI	Electro-Magnetic Interference
EPS	Electronic power system
ESD	Electrostatic discharge
I <sup>2</sup> C	Inter integrated circuit communication bus used to control the Antenna system.
ISIS	Innovative Solutions In Space BV.
MGSE	Mechanical Ground Support Equipment
RF	Radio Frequency
RH	Relative Humidity
LED	Light emitting diode.
N/A	Not Applicable
TBC	To Be Confirmed
TBD	To Be Determined
TBW	To Be Written
UM	User Manual
VNA	Vector network analyser
UHF	Ultra High frequency (400-470MHz within this context)
VHF	Very High frequency (136-170MHz within this context)

## 1 Introduction

The ISISPACE On Board Computer (iOBC) is a general-purpose computing board that adheres to the CubeSat standard. It provides powerful processing capabilities and comes with an extensive set of software tailored specifically for space applications. The basic interfaces are standardized improving robustness, while a high level of customization can be provided through the daughterboard concept without the need to sacrifice additional volume.

This document provides aid for getting started with embedded software development for the iOBC hardware Revision C or higher. Chapter 1.4 provides a general discussion on embedded systems that provides a primer for people unfamiliar with embedded systems and embedded software development. Chapter 3.1 dives into the details of installing drivers and the iOBC Software Development Kit (SDK). Chapter 3.2 describes how to connect the iOBC hardware to the development machine. Eclipse configuration in Chapter 3.3 concludes the setup of the IDE. Code development and compiling/running/debugging application code is discussed in Chapters 3.4 and 3.5. Chapter 3.6 provides information on flashing code to the non-volatile memory. Chapter 4 describes how to get support.

### 1.1 Applicable Documents

The table below contains documents which applicability is required. The contents of the present document follow the standards, guidelines and requirements here mentioned.

Table 1 - Applicable Documents

Reference	Name	Version
---	---	---

### 1.2 Reference Documents

The table below contains documents that are not fully applicable and will provide supplementary information relevant for the present document.

Table 2 - Reference Documents

Reference	Name	Version
---	---	---

### 1.3 Specification

Table 3 - General Specifications

Parameter	Typical Values
<b>Electrical Characteristics</b>	
Processor	400MHz, 32-bit ARM9 (AT91SAM9G20)
RAM	64MB SDRAM
Non-volatile data storage	2x2GB SD-Cards with SafeFAT file system 512kB FRAM (high endurance and fast read/writes)
Code storage	1MB NOR-Flash
Timing	1xRTC, 1xRTT
Watchdog	External on-board watchdog and power supervisor
On-board sensing	Temperature, current and voltage measurements with over-current protection

Parameter	Typical Values
Hardware interfaces	1x I2C (master or slave, Fast-mode, $\leq 400\text{ kbit/s}$ ) 1x SPI: Up to 8 slaves ( $\leq 10\text{ Mbit/s}$ ) 2x UARTs ( $\leq 10\text{ Mbit/s}$ , depending on configuration): 1x LVCMOS or RS232 levels (hardware configuration) 1x RS232 or RS422/485 levels (software configuration) 1x ADC: 8 input channels, 8 or 10-bit modes PWM: 6 output channels GPIO: 27 pins USB: 1x Host and 1x Device ( $\leq 12\text{ Mbit/s}$ ) 1x Image Sensor Interface for directly interfacing with CMOS image sensors (shared with GPIOs)
Programming and debug capabilities	JTAG for programming and debugging, Additional debug UART for console user-interface, 4x LEDs
Average power consumption	380mW, typical usage @ 3.3V supply
<b>Environmental Characteristics</b>	
Qualified operating temperature range	-25°C to +65°C
Storage temperature range	-40°C to +80°C (RH < 60%)
<b>Physical Characteristics</b>	
Dimensions	96 x 90 x 12.4mm (including FM daughterboard)
Mass	94g (excluding daughterboard)

## 1.4 Embedded Systems Software Design

This section provides a primer to embedded systems design. It can be skipped without loss of information.

An embedded system is a generic term that refers to any computer system that is located along with the hardware it controls. The main control unit of the system is referred to as the controller. It has historically been carefully selected to supply just enough processing power to perform their controller function while keeping cost and power consumption to a minimum. These carefully selected controllers are referred to as microcontrollers. With processing power becoming less expensive in terms of power consumption and cost, increasing amounts of embedded systems employ a more generic and much more powerful controller unit referred to as a central processing unit (CPU). The iOBC uses a processing unit based on the ARM architecture.

A computer running Microsoft Windows is used to develop and deploy embedded software to the iOBC controller hardware. This machine will be referred to as the *development machine*. The development machine requires drivers to be installed to allow interfacing with attached hardware, as well as programs that use the drivers to perform various tasks. ISISPACE provides a Software Development Kit (SDK) that along with this document provide all required programs.

To interact with the iOBC hardware, two types of electronic support equipment (EGSE) are used:

1. A programmer to deploy and debug embedded programs running on the iOBC processor.
2. An USB-to-Serial converter which provides a separate full-duplex data channel between the development machine and the iOBC controller.

Hardware interfaces to the serial data connection lines are usually provided for in EGSE, and provide a mechanism for relaying debug messages, commanding of embedded programs during ground testing and setting final configurations before flight.



Programs that control the hardware attached to the development machine by interacting with the installed drivers are contained within the IOBC Software Development Kit (SDK) installer package, along with applications that provide translation of program code to machine instructions (the compiler/linker toolchain) and a programming environment that aids an embedded developer in efficiently producing embeddable software. The collection of these programs is usually brought together and controlled from within a single application, which is referred to as the Integrated Development Environment (IDE).

Once coding of an embeddable program has been completed, the IDE is instructed to perform a build using the toolchain programs. A successful build produces a file that contains machine instructions specific to the embedded processor architecture. The IDE can then instruct the programmer hardware to transfer the machine code to the embedded processor and execute the program. The embedded software developer has included code within the program that emits debug trace and/or error messages, which are relayed over the USB-to-Serial interface to the development machine. The debug messages are received and displayed in a terminal application, which allows the developer or e.g., test conductor to verify the program's behaviour. The terminal application also allows sending data over the USB-to-Serial interface to the embedded processor, at which point code included by the embedded software developer can take appropriate action.

Once correct program behaviour has been verified by using the programming hardware, the program is 'burned' into embedded memory using some type of non-volatile memory. From this point onwards the hardware can perform its functions without the need for an attached development machine.

## 1.5 Deliverable hardware

The non-optional IOBC deliverables are depicted in Figure 1 and detailed in

Table 4 – Non-optional IOBC deliverables

Item	Quantity	Comments
IOBC Motherboard	1	Revision C or higher
SD Cards	2	May be delivered assembled into the motherboard
Power break-out board	1	--
J-TAG programmer	1	--
J-TAG programmer adapter	1	Including harnessing



Figure 1 – Non-optional IOBC deliverables (daughterboard is optional)

## 2 Handling and Storage

### 2.1 Handling



**Important Note:** This product is sensitive to Electrostatic Discharge (ESD).

#### 2.1.1 Electrostatic discharge ESD

**CAUTION**



The printed circuit board can be damaged by electrostatic discharge. Do not touch any of the boards unless it is absolutely necessary. If you must handle them, wear a grounded wrist strap and take other antistatic precautions. Wear a grounded wrist strap any time you must handle the board.

#### 2.1.2 Exposed voltages

**WARNING**



Handling the board with an active power supply connection is not recommended. The board itself could be damaged and there is a possibility of electric shock hazard.

In the event of Emergency, disconnect the power supply and proceed, if required, with first aid activities.

#### 2.1.3 Current Limit Protection

**CAUTION**



Ensure that over-current protection to a level of 1A is present when connecting to external power supplies.

#### 2.1.4 Operation Conditions

**CAUTION**



Limit the number of connector mating cycles to less than 50 (about 10 cycles are used during functional testing)

## CAUTION



The IOBC is supplied with final functional test results. Operating it outside its prescribed operating conditions may impede functionality.

Ensure that the system is always operated within its qualification temperature range.

## CAUTION



This system does not have a protective housing and is therefore not intended for outdoor use on ground as the board electronics might be damaged.

## 2.2 Storage

### CAUTION



Store the IOBC inside a sealed ESD bag and in an environment controlled area.

The absolute maximum ratings for storage temperature are from -40 to +80°C with a Relative Humidity <60%

## 2.3 Disposal

### WARNING



This product contains materials that can be harmful for the Environment and as such it should not be disposed of with conventional waste but treated according to WEEE regulations (UE Directives 2002/96/EC and further amendments) and brought to an appropriate recycling facility.

## 3 Detailed Description of Operations

### 3.1 Setting up the Integrated Development Environment

To set up the integrated development environment on the development machine several software items must be installed and configured. The installation is comprised of the following steps:

1. Install J-Link driver of the SAM-ICE programmer/debugger.
2. Install driver of the FTDI USB-to-Serial device.
3. Install the iOBC SDK containing required IDE application software.
4. Extract the iOBC libraries.
5. Extract the iOBC first-project.

Each of the installation steps will be described in the following sections.

#### 3.1.1 Install Programmer Hardware Driver: J-Link from Segger

Download and install the J-Link drivers for the programmer/debugger, shown in Figure 2. The newest driver version can be installed from:

<https://www.segger.com/jlink-software.html>

**Note:** It is highly recommended to install J-Link 7.50 or newer. Additionally, for users of 64bit Windows (most people), it's highly recommended to install the 64bit version of J-Link. In the event an older version is used, you may need to read Section 3.3.2 to update the project to match the installed environment.

The serial number can be found on the bottom of the SAM-ICE debugger.



*Figure 2 - SEGGER SAM-ICE Programmer/Debugger Atmel OEM version*

#### 3.1.2 Install USB-to-Serial Hardware Driver: FTDI

Download and install drivers for the FTDI USB-to-Serial adapter from:

<http://www.ftdichip.com/Drivers/VCP.htm>.

Locate the setup executable that matches the Windows architecture used on the development machine.

#### 3.1.3 Install iOBC SDK

Install the iOBC SDK (Software Development Kit) provided by ISISPACE. The installer 'ISIS-OBC SDK.exe' can be found with the board's software package.

The installer allows optional install of the following applications:

- Eclipse IDE
- Atmel SAM-BA for interfacing with ARM processor/FLASH memory hardware
- PuTTY Terminal Client
- MinGW GNU compiler/linkers for Windows
- ARM Toolchain for compiling code for the ARM processor architecture

**Note:** All software is installed into a fixed directory with path: C:\ISIS\

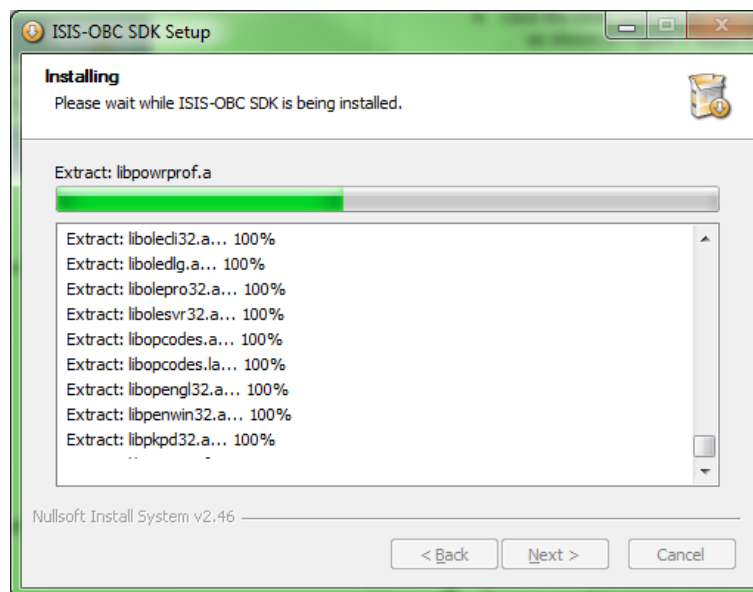


Figure 3 - The iOBC SDK installer

### 3.1.4 Extract iOBC First-project

The iOBC first-project is an Eclipse project that serves as a template for new iOBC code development projects. Each new project can be started by making a copy of the template project after which search-and-replace is performed to update project name and file linkage to that of the new project.

Simply extract the 'isis-obc-first-project.zip' contents into e.g., a subfolder of the ISISPACE root folder located at C:\ISIS\. For our purposes we'll use the folder C:\ISIS\ISIS-IOBC\ to place new iOBC development projects in. However, any other location will do.

### 3.1.5 Extract iOBC Libraries

The iOBC libraries provide means for rapid application development by acting as a layer between the low-level hardware and the developed application.

There are three different modules available at the time of writing:

- Hardware Abstraction Layer (HAL): provides low level generic drivers for the SAM9G20 processor peripherals.
- Satellite Subsystems (SIL): provides control functions for all ISISPACE subsystems as well as some select 3rd party subsystems. It uses the HAL library for performing lower-level operations.
- Mission Support (MIS): provides additional building blocks needed for robust and efficient OBC flight software. It uses the HAL library for performing lower-level operations.

Depending on the purchased package, either only the HAL, HAL+SIL or the HAL+SIL+MIS libraries have been provided as versioned zip files. The content of these zip files should be extracted to the same folder as where the 'isis-obc-first-project' resides.

The default location that is used within this document is `C:\ISIS\ISIS-IOBC\` (see Section 3.1.4). The resulting directory structure for the HAL+SIL option looks as is shown in Figure 4.







 hal	17-8-2015 12:28	File folder
 isis-obc-first-project	17-8-2015 11:52	File folder
 satellite-subsystems	17-8-2015 12:28	File folder
 isis-iobc-hal-0.1.8.zip	2-3-2015 14:10	zip Archive
 isis-iobc-satellite-subsystems-0.1.0.zip	13-3-2015 16:16	zip Archive
 isis-obc-first-project.zip	17-8-2015 12:31	zip Archive

Figure 4 - Final file structure of the C:\ISIS\ISIS-IOBC\ main project folder. Keeping the zip files is optional.

## 3.2 Setting-up the Hardware

This section will provide instructions on how to set up the iOBC hardware and related tooling.

**Important Notice:** Before performing the hardware related steps described in this section, please read through the iOBC Datasheet for information on safety instructions related to the handling of the iOBC hardware.

The following steps outline the required connections to setting up the iOBC hardware.

- Connect the programming adapter to the iOBC.
- Connect the SAM-ICE programmer/debugger to the programming cable.
- Connect the delivered Supply Voltage Break-out board to the iOBC CSKB connector.
- Connect the Supply Voltage Break-out board to a bench power supply configured to deliver 3.3V with a safety current limitation of 150mA.
- Connect the USB cable to the SAM-ICE and to your computer.
- Connect the Debug port (USB-to-Serial) cable to your computer.
- Power the iOBC

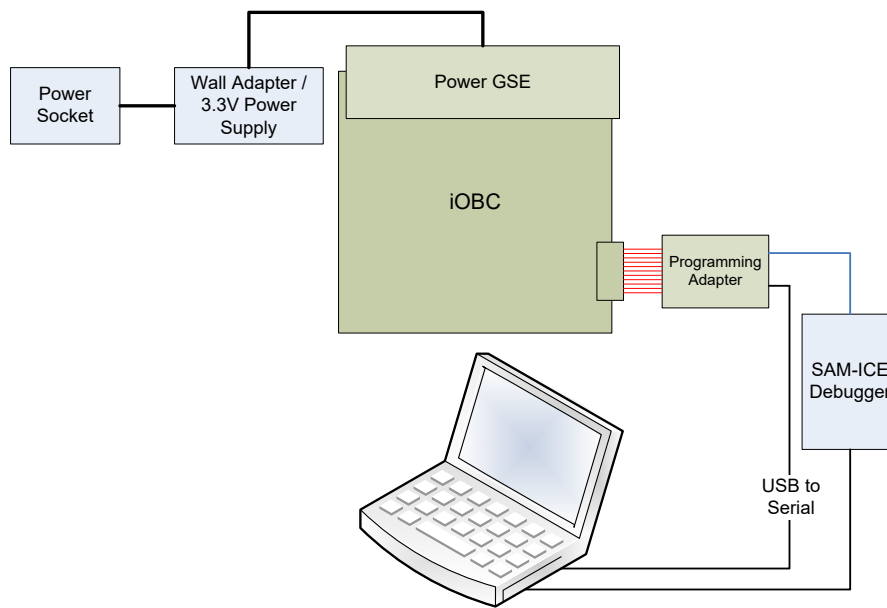


Figure 5 - iOBC embedded software development hardware configuration

## 3.3 Configuring Eclipse IDE

The Eclipse IDE application can now be opened using the start menu links.

To do this navigate to and open:

Start > All Programs > ISIS-OBC SDK > Eclipse IDE

An empty Eclipse IDE window is shown similar to that shown in Figure 6 (left). The initial welcome screen can be closed straight away with the x symbol on the 'Welcome' tab. The default project screen will appear as shown in Figure 6 (right).

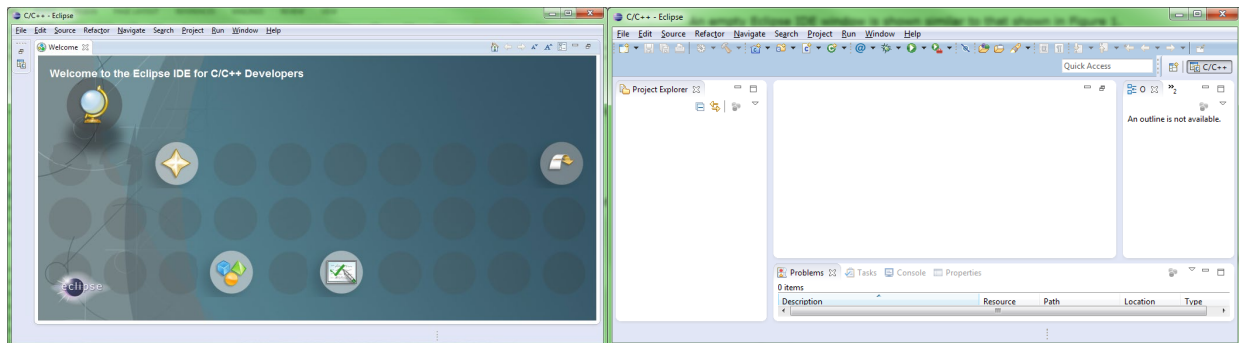


Figure 6 - Eclipse IDE welcome screen (left) and project screen (right) shown after closing the welcome screen

The following first-time configuration steps need to be performed to allow development within the Eclipse environment:

1. Import "isis-obc-first-project" into Eclipse.
2. Add the J-Link GDB server launch configuration to Eclipse.
3. Configure path to J-Link GDB server executable.
4. Add the Debug/Release launch configurations to Eclipse.

These steps will be described in the following sections.

## 3.3.1 Import the 'isis-obc-first-project' into Eclipse

To import the 'isis-obc-first-project' into Eclipse, which was copied as described in section 3.1.5, the following steps need to be performed:

1. Right click in Eclipse Project Explorer and click 'Import ...'

The Eclipse Project Explorer is found on the left of the Eclipse application window as shown in Figure 7.

2. Select 'General' > 'Existing Projects into Workspace' and click Next. See Figure 8.
3. Select 'Choose root directory' and select the location of the 'isis-obc-first-project' folder. Default used in this document: `C:\ISIS\ISIS-OB\`.
4. The 'isis-obc-first-project' should be selected automatically. See Figure 9. Please make sure 'Copy projects into workspace' is not selected.
5. Click 'Finish' to complete the import.

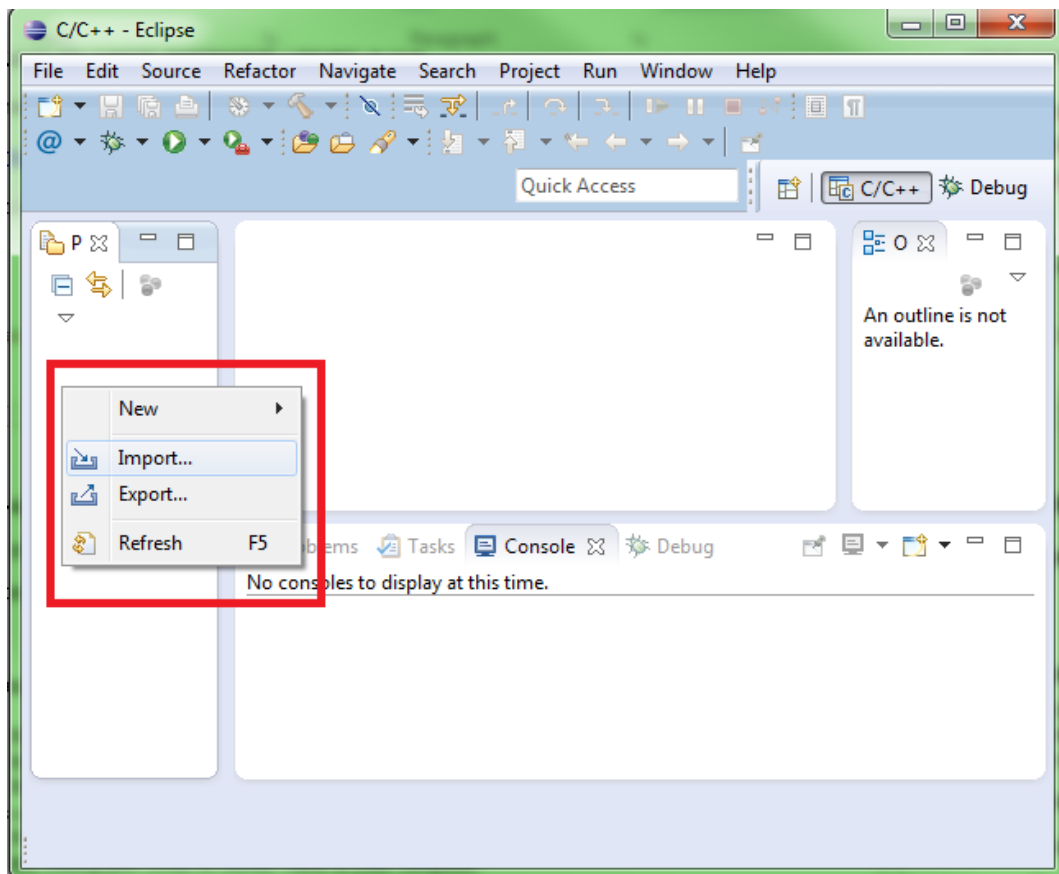


Figure 7 - Select Eclipse Project Explorer import function



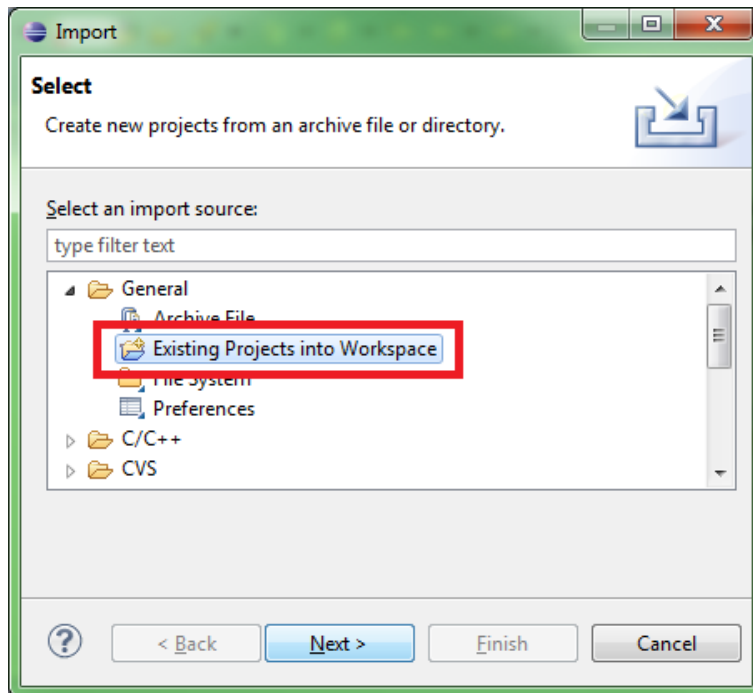


Figure 8 - (import ...) Select General > 'Existing Projects into Workspace'

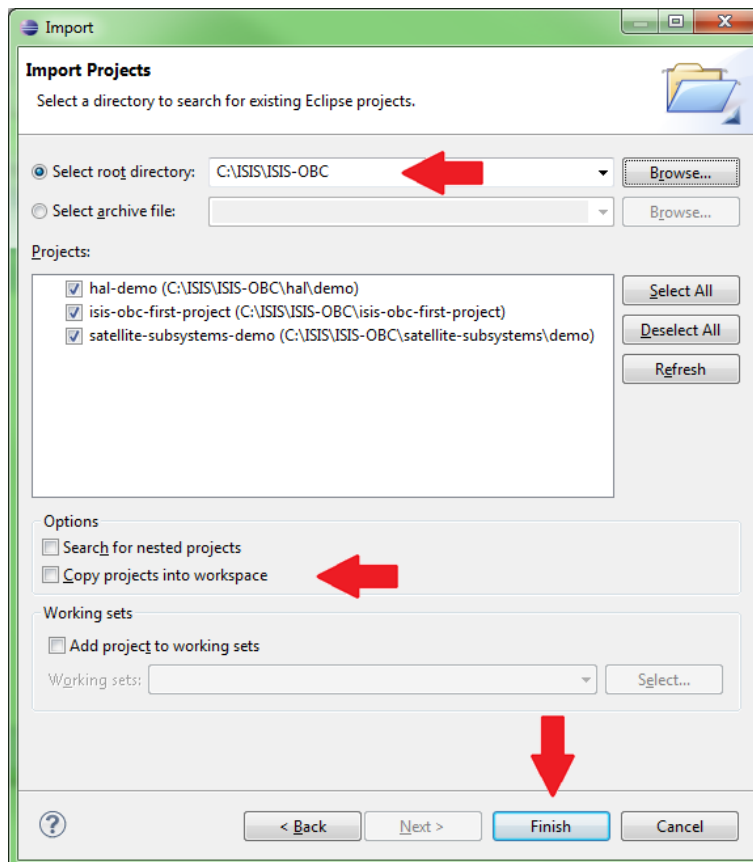



Figure 9 - isis-obc-first-project and any library demo projects are selected for import. **IMPORTANT:** 'Copy projects into workspace' should be deselected, to make sure the project source files are not copied into the workspace.

## 3.3.2 Optional: Configure the J-Link GDB Server Configuration

**Note:** This section is only needed when using an older version of J-Link. For setups using J-Link 7.50 and newer, this should not be necessary.


Debugging services are provided for using a GDB server application, for which an executable is installed by the J-Link installer. The following steps should be completed to make sure Eclipse can find the GDB server executable that should have been installed as described in Section 3.1.1.

To add the launch configuration, do the following:

1. Select the Jlink-samg20-GDB.launch file in the Eclipse Project Explorer window.
2. Click the small down arrow next to the debug-tools  icon, expand 'Run As' and click Jlink-samg20-GDB.

The J-Link GDB debug launch configuration is now added to Eclipse.

The steps to update the path to the J-Link GDB server application are as follows:

1. Locate the J-Link installation directory. The default is  
(on win32) `C:\Program Files\SEGGER`  
(on win64) `C:\Program Files (x86)\SEGGER`
2. Make a note of the folder name within the installation folder. It has a format such as JLinkARM\_Vxxxx (e.g., JLinkARM\_V478c).
3. Click the small down arrow next to the debug-tools  icon and choose 'External Tools Configurations ...' as shown in Figure 11.
4. Select the 'Jlink-samg20-GDB'.
5. Note: This is the only configuration that needs to be updated when this configuration is the only one that will be used during debugging.
6. Make sure the 'Location' string contains the correct path using the J-Link version installed on the system. Copy/paste the path to an explorer for quick verification. See Figure 12.

E.g., the location string with JLinkARM\_V478c on Windows 32-bit using the default install path becomes:

```
C:\Program Files\SEGGER\JLinkARM_V478c\JLinkGDBServerCL.exe
```

The Windows 64-bit has a different 32-bit program files folder name and becomes:

```
C:\Program Files (x86)\SEGGER\JLinkARM_V478c\JLinkGDBServerCL.exe
```

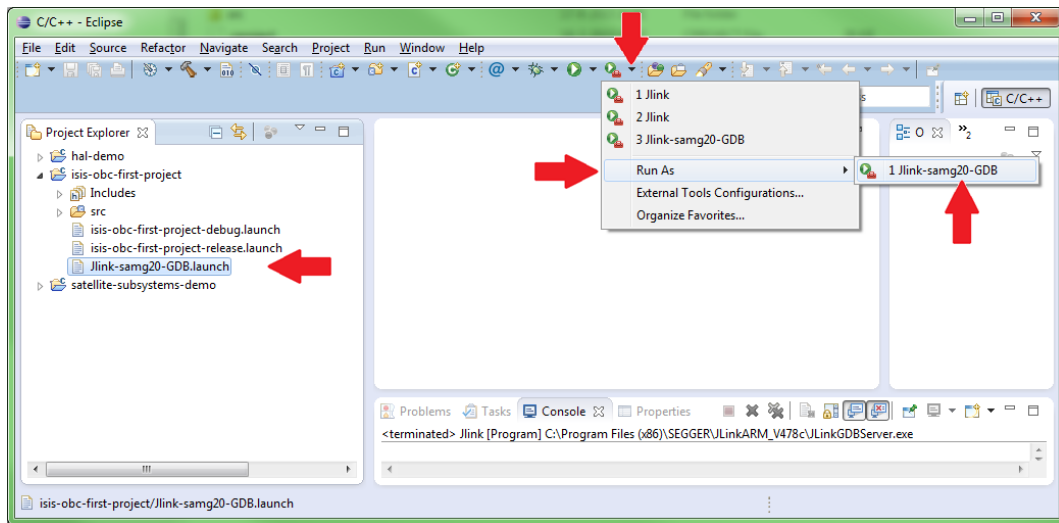


Figure 10 - Verify/Add the GDB launch configuration

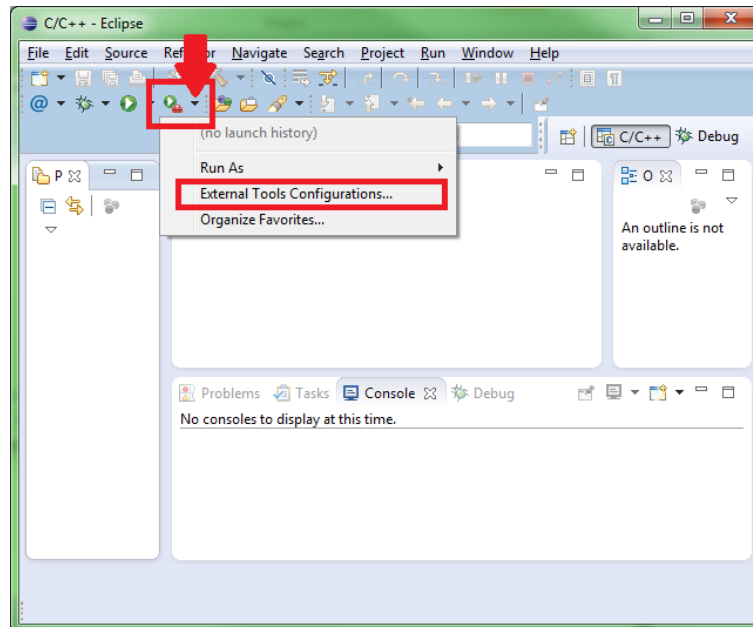


Figure 11 - Opening Eclipse external tools configuration window

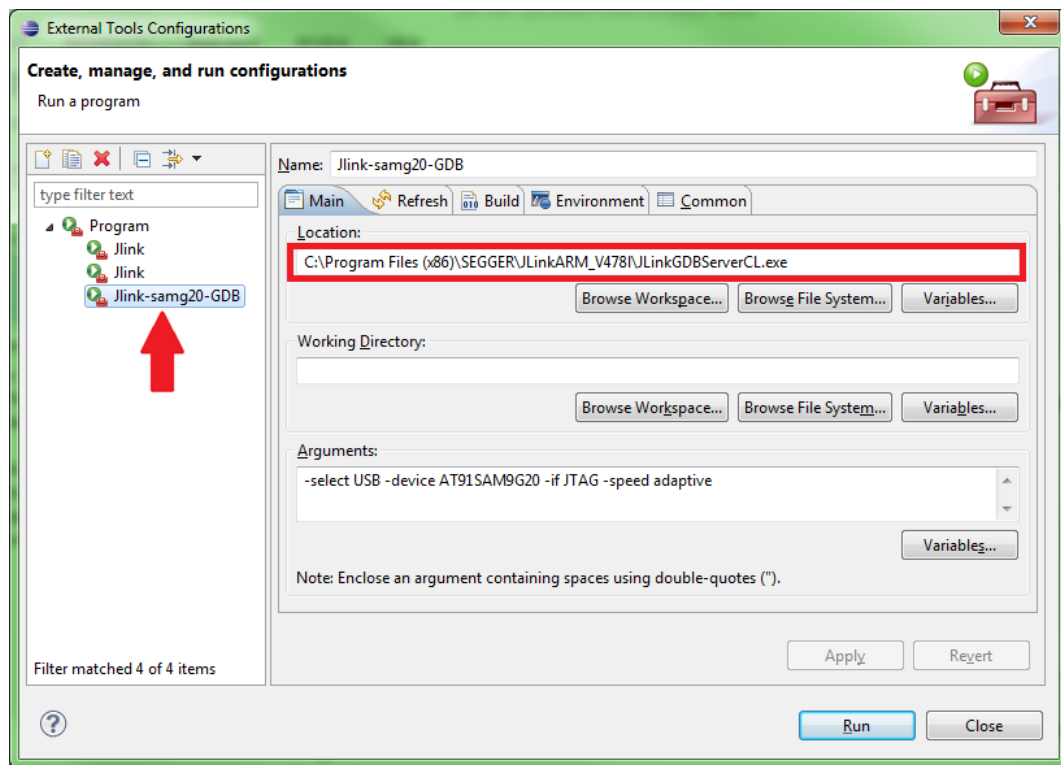



Figure 12 - Update/Verify that the correct path to the GBD server executable is in place for the 'JLink-samg20-GDB' launch configuration.

### 3.3.3 Check/Add the Debug Launch Configuration

The debug launch configuration should be added to the debug list by performing the following steps.

1. Select isis-obc-first-project-debug.launch in the project explorer.
2. Click the small down arrow next to the debug  icon.
3. Expand 'Debug As'.
4. Click isis-obc-first-project-debug.

This adds the debug launch configuration to the list. The same can be performed for the release config. See Figure 13.

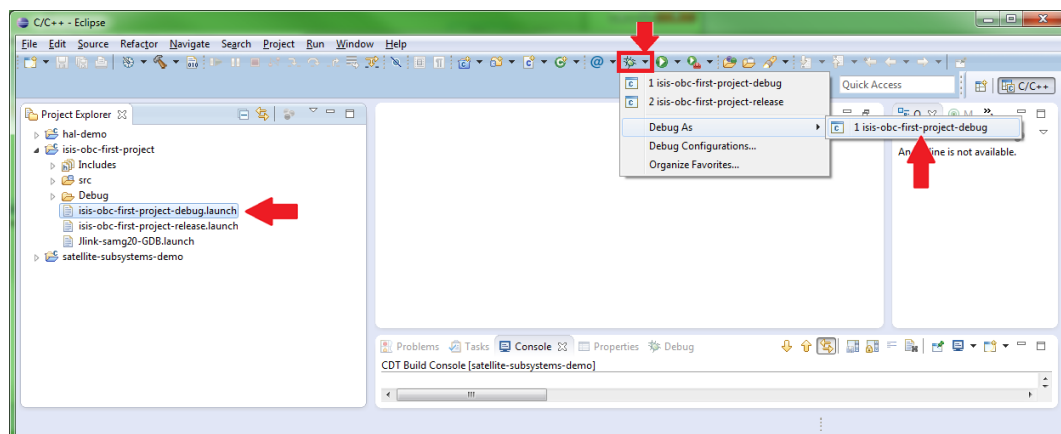



Figure 13: Adding the debug launch configuration

## 3.4 Code Development and Debug/Release Builds

With the IDE set-up as described in the previous sections, the code provided in the isis-obc-first-project can be build and used for checking correct IDE behaviour. The main.c starts up a main task that enters an infinite loop and blinks the iOBC LEDs in specific patterns.

### 3.4.1 Compiling Code

To compile the project, do the following:

- Make sure the project is selected by clicking on its name in the Project Explorer window, which is located on the left-hand side of the window.
- Click the build  icon.

This is shown in Figure 14.

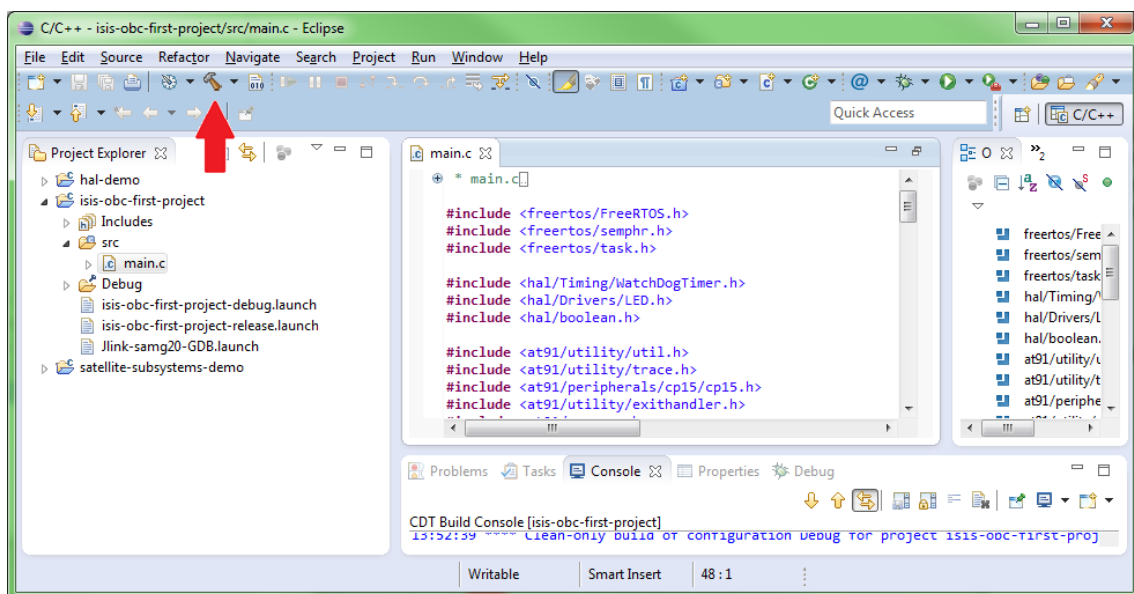


Figure 14 - isis-obc-first-project opened in the Eclipse IDE, with the red arrow indicating the Build icon

### 3.4.2 iOBC First-project as Template

The iOBC first-project can be used as a template for new projects by a simple copy/paste operation. This can be directly performed from the Project Explorer. The Project Explorer also allows seamless copy and paste operations to/from the Windows (file) explorer.

#### 3.4.2.1 Creating a New Project Using the Template

To create a new project using the template the following steps should be performed:

1. Create a copy-in-place of the isis-obc-first-project directory.
2. Open each of the Eclipse files located in the new project directory (.cproject, .project and \*.launch) using a text editor (e.g., notepad) and perform a rename of all occurrences of *isis-obc-first-project* to the new project name.

#### 3.4.2.2 Library Linkage

The linkage to the libraries within a project makes use of relative paths and expect the library folder in the same folder as the project root directory. This means the library root directories must be in the same directory as the project root directory.

An example of a directory structure is shown in Figure 15. The example shows the isis-obc-hal-0.1.8 zip and the 'hal' directory that was extracted from this zip. The first-project can now find the 'hal' library and will successfully build. Note that the zip file is not used and could optionally be removed.

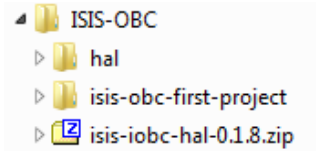


Figure 15 - Relative file locations. The library directory should be in the same directory as the project.

Library updates are simple: just replace the existing library directory with the new version of the library. The projects will automatically use the new libraries during the next build.

### 3.4.2.3 Linking an Additional Library

Several libraries are available through ISISPACE. To use an additional library the path to the library directory should be set up properly. This involves setting include paths and adding the library name in Eclipse project properties. In this example the satellite-subsystems library will be added.

1. First the new library is extracted to the directory that also includes the project. The directory is shown in Figure 16.

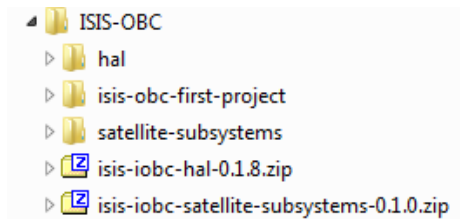


Figure 16 - Satellite-subsystems library added to the directory.

2. Add the include path to the library: Open Eclipse, right click the first project and open project properties. Navigate to **C/C++ Build > Settings**. Under Tool Settings choose **Cross ARM C Compiler > Includes**. Add the path to the *include* folder of the library. For satellite subsystems this is "\${external-dependencies}/satellite-subsystems/satellite-subsystems/include". See Figure 17.

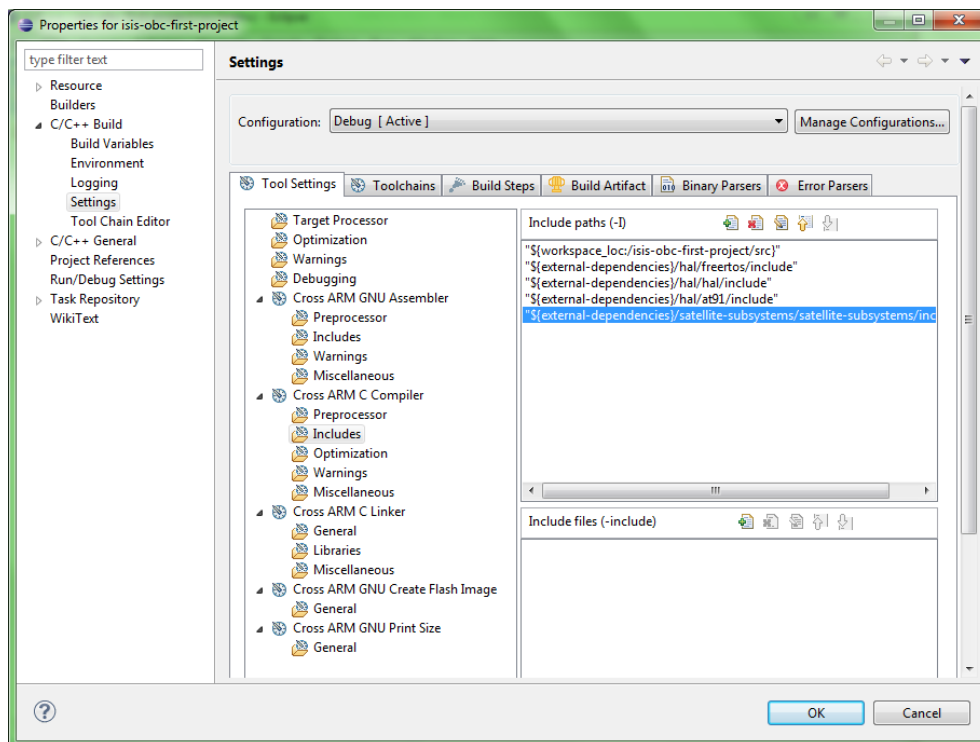


Figure 17 - Adding the library include path

3. Add the library to the linker settings: Select **Cross ARM C Linker > Libraries**. Add the library name with a capital D at the end to the **Libraries** list. Add the path to the **lib** directory to the **Library search path** list. See Figure 18.

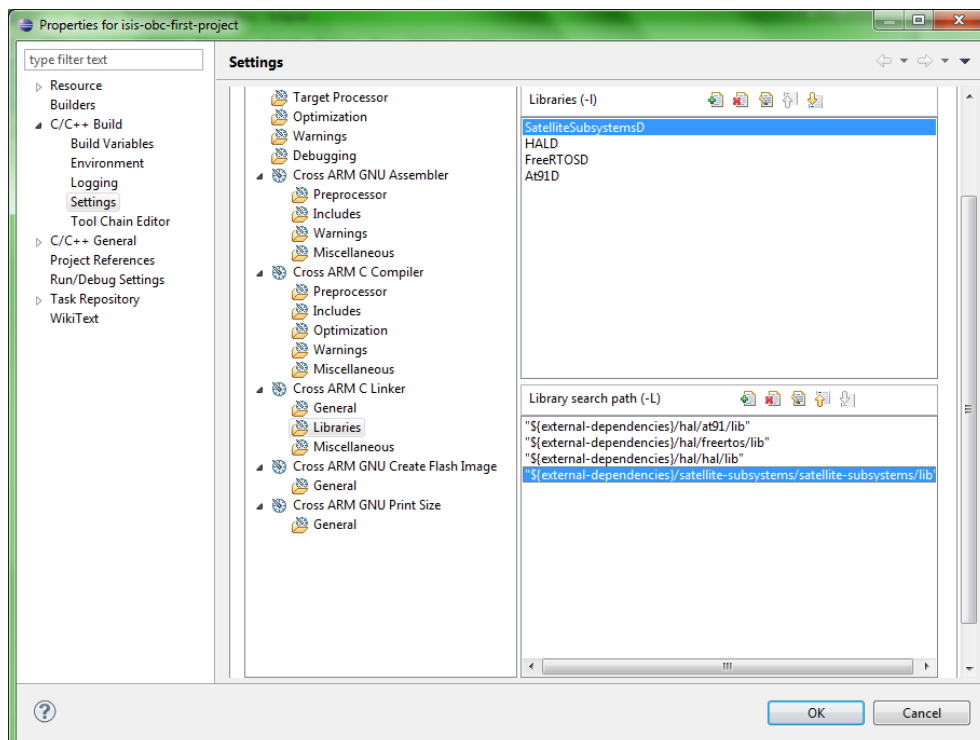


Figure 18 - Add the library to the linker settings

4. Click ok. The library can now be used from within the project.

## 3.4.3 Eclipse Build Configuration

Eclipse provides so called 'Build Configurations'. The active build configuration is used when code is built. The standard configurations provided are Debug and Release.

### 3.4.3.1 Debug Build

A Debug Build is a non-optimal compilation of the embedded software to facilitate efficient debugging. The IDE instructs the compiler to not use code optimization and to add debug symbols within the compiled program. This allows the debugging service to perform its duty of providing the developer with program state information. The IOBC code also adds additional code that allows debug messages to be emitted. The Debug Configuration is therefore larger and slower than necessary for unattended execution of the program.

### 3.4.3.2 Release Build

A Release Build is an optimized compilation of the embedded software targeted at unattended/autonomous execution of the program. This configuration should be used once the code performs to satisfaction using the Debug Build, and the program is to be officially put through verification test procedures.

**Note:** Due to code optimizations applied by the compiler and excluding debug related functions from the build, the resulting machine code can be quite different. The functional aspect should remain the same, however in some particular cases differences might still occur. It is therefore very important to use the release build for verification purposes, to be able to catch any problems arising from release build optimizations.

### 3.4.3.3 Changing the Active Build Configuration

The current Build Configuration used by Build can be set through the context menu by right clicking the project and choosing 'Build Configurations' as shown in Figure 19.

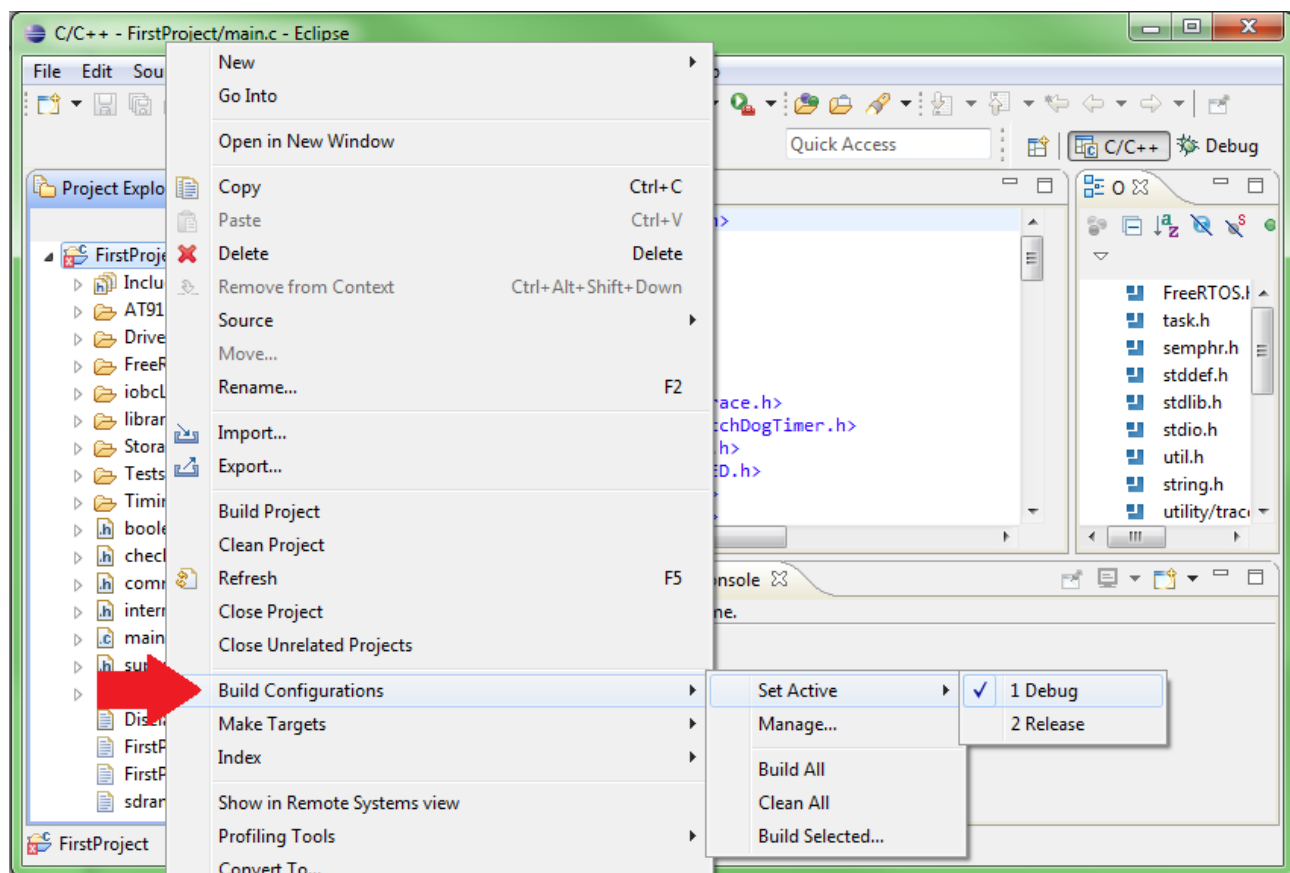


Figure 19 - Setting the build configuration



## 3.5 Running your Program

After a program has been successfully build the code can be transferred to the embedded processor and executed on the embedded hardware. Setup the hardware as described in Chapter 3.2.

For debugging purposes, the programmer/debugger hardware and software takes care of setting up the embedded processor so it will run the code build within the IDE. It provides functionality to pause code execution, using a so-called breakpoint, and read-out memory on the embedded processor. The code itself is directly copied to the volatile memory of the processor, such that the write-cycles of the non-volatile flash-based program memory remain intact.


For release purposes the build code will be stored in the non-volatile flash-based program memory using a dedicated program called sam-ba. The iOBC controller will then be able to execute the stored program autonomously and independent from the development machine.

In both cases the USB-to-Serial data interface is available to transfer commands and data between the iOBC processor and a computer which has at least the FTDI driver installed.

### 3.5.1 Debugging your Code

To perform code debugging at least the programmer/debugger must be connected. Generally, also the USB-to-serial connection is used in parallel that is used to emit debug and program state messages. The debugging services are provided by the widely used GNU DeBugger (GDB).

#### 3.5.1.1 Start the GDB Server

To start the GDB server, click on the small down pointing arrow next to the External Tools  icon and then on Jlink-samg20-GDB. This launch item has been configured in chapter 3.3.

#### 3.5.1.2 Start PuTTY for the USB-to-Serial Interconnect

For usage of the serial interface the PuTTY application is provided.

- Start the putty executable found via Start > All Programs > ISIS-OBC SDK > PuTTY. A window similar to Figure 20 is shown. Select Serial and set the speed to 115200 (115.2kbaud).
- Set the COM port that has been assigned to the FTDI USB-to-Serial device as COM##, where the ## is the com port number used. This can be found in the windows device manager. Go to start and right click Computer. Select properties. Select device manager and expand to the Ports section. The device is USB Serial Port (COM##). See Figure 21.
- Optionally provide a name in the 'saved sessions' textbox and click 'Save' to save the configuration. This can be loaded later by clicking the name and then clicking 'Load'.
- Click Open.

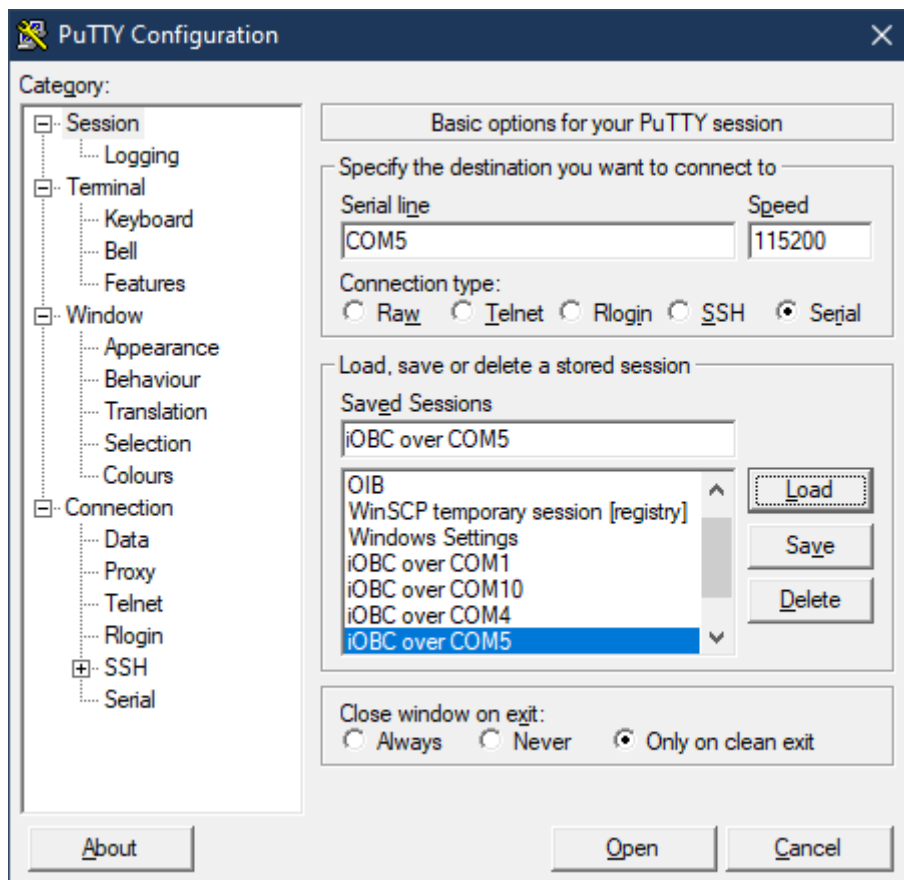


Figure 20 - Putty configuration window

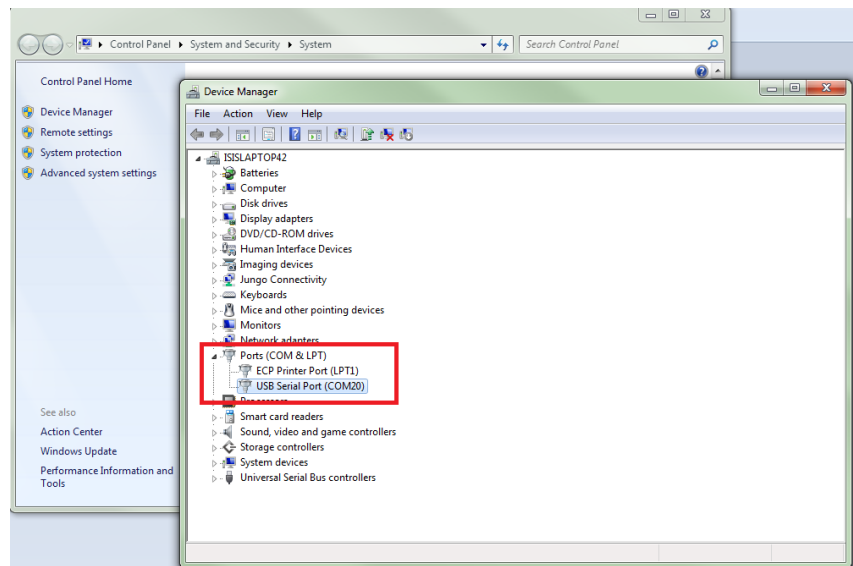
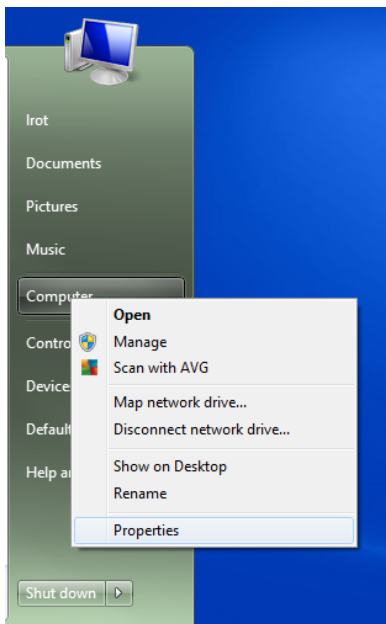



Figure 21 - Retrieve the COM port assigned to the USB-to-Serial from the device manager

### 3.5.1.3 Starting a Debug Session

Launch the Debug of your project by clicking on the small down pointing arrow next to the debug  icon. Then click the name of the debug configuration. See Figure 22.

Optionally the demo project for a library (e.g., hal-demo) can be run in debug or release mode. To do this first select the project and then select the debug/release launch. This will launch demo/test applications for that particular library.

**IMPORTANT NOTICE:** the demo/test code serves as simple examples of how to use the library functionality but is by no means guaranteed to have high quality/flight quality nor uses all available functionality. The demo/test code should only be used as a first directional hint when starting to use a new part of the library. Always refer to the library documentation for a full explanation and overview of all available functions, and carefully structure the code according to your preferred coding style. It is highly discouraged to directly copy/paste blocks of code from the demo/test software during development.

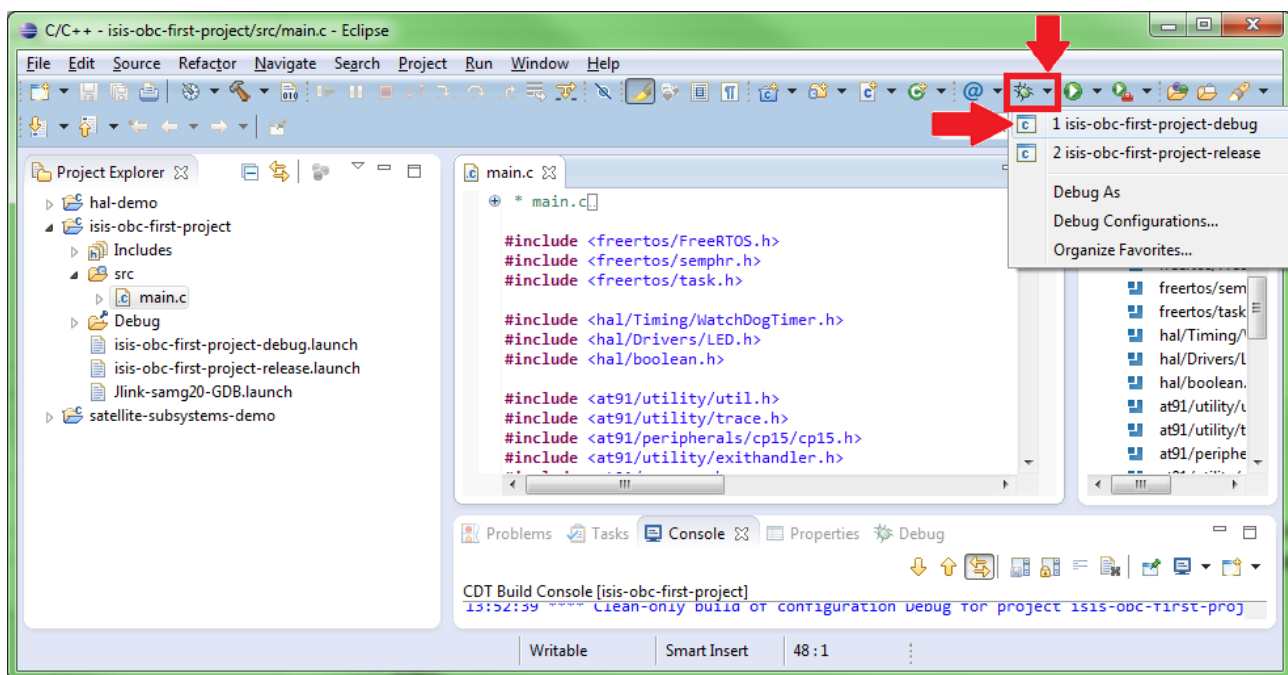


Figure 22 - Start Debug session

### 3.5.1.4 Debugging the Application

Now you can control the execution of your code by setting breakpoints; clicking play and pause. You can also inspect local variables and add watchpoints for specific variables. You can also see the debug output of your program (such as printf's) using PuTTY.

## 3.6 Flashing Your Code

Once you have debugged your code, you can save the code into the Flash memory of the OBC. This is done using the SAM-BA program that is started through Start > All Programs > ISIS-OBC SDK > Atmel SAM-BA.

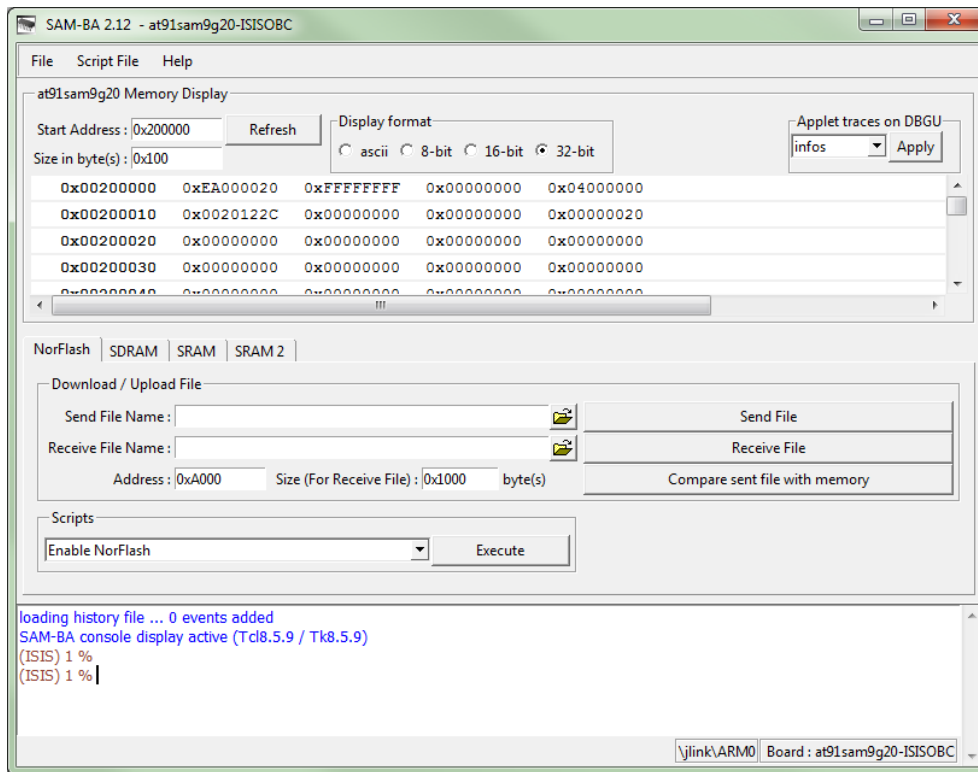


Figure 23 - SAM-BA program

- Make sure you have set up the hardware as indicated earlier.
- Make sure you have closed any running debug session within Eclipse.
- Launch SAM-BA.
- Select the *at91sam9g20-ISISOBC* board under boards and click "Connect".
- Under the NOR-Flash tab of SAM-BA, click on "Execute" to enable NOR-Flash.
- Browse for a file using the folder icon next to the "Send File" button.
- The Address field is set to 0xA000 which is the default value where the application code should be flashed to. The bootloader occupies address range 0x0000 through 0xA000 and is write-protected when using the *at91sam9g20-ISISOBC* board selection. See Section 3.6.1 if the bootloader memory needs to be updated.
- Click "Send File" and wait for your program to upload.
- Click "Compare sent file with memory", SAM-BA will read back the contents of the flash and verify that the firmware is flashed correctly. If the process was successful, it will display a message indicating that the sent file and memory match exactly.
- Close SAM-BA, remove power from the OBC and power it up again.
- Your code will now be running on the OBC. You can see the output of your program (such as printf's) using PuTTY.

**Note:** It is advised to use "Compare sent file with memory" to verify that your firmware was uploaded correctly.

## 3.6.1 iOBC Bootloader

The NOR-Flash of an iOBC contains a bootloader memory space that starts at address 0 and extends until address 0xA000. The bootloader prepares the board, so programs stored from 0xA000 onwards are loaded into SDRAM and started. You should not overwrite the bootloader memory area under regular circumstances. However, the bootloader memory can be written to by selecting the *at91sam9g20-ISISOBC-UpdateBootloader* board when starting SAM-BA.

To update the bootloader memory area:

- Launch SAM-BA.
- Select the *at91sam9g20-ISISOBC-UpdateBootloader* board under boards and click "Connect".
- Follow the steps for flashing code to NOR-Flash.

**Note:** A backup of the standard bootloader binary is provided along with the SDK installer and can be used to update the board when the original bootloader flash memory area is accidentally overwritten.

## 3.6.2 Backing-Up and Restoring Flash Code

It is possible to backup code flashed to the iOBC hardware. Make sure you have set up the hardware as indicated earlier.

- Launch SAM-BA and click "Connect".
- Under the NOR-Flash tab of SAM-BA, click on "Execute" to enable NOR-Flash.
- (Option 1) Backup of program excluding bootloader:
  - Enter "0xA000" next to "Address".
  - Enter "0xF6000" next to "Size (For Receive File)".
- (Option 2) Backup of program including bootloader:
  - Enter "0x0" next to "Address".
  - Enter "0x100000" next to "Size (For Receive File)".
- Browse to the location where you want to save the backup using the folder icon next to the "Receive File" button.
- Click "Receive File" and wait for the operation to finish.

The file you specified now contains a backup of the contents of the NOR-Flash of the OBC.

To restore backups follow the procedure of flashing code described in Section 3.6.1 for backups taken from start address 0xA000 (option 1), and Section 3.6.2 for backups taken from start address 0x0 (option 2).

## 4 Support

In case of issues, please contact ISISPACE ([support@isispace.nl](mailto:support@isispace.nl)).