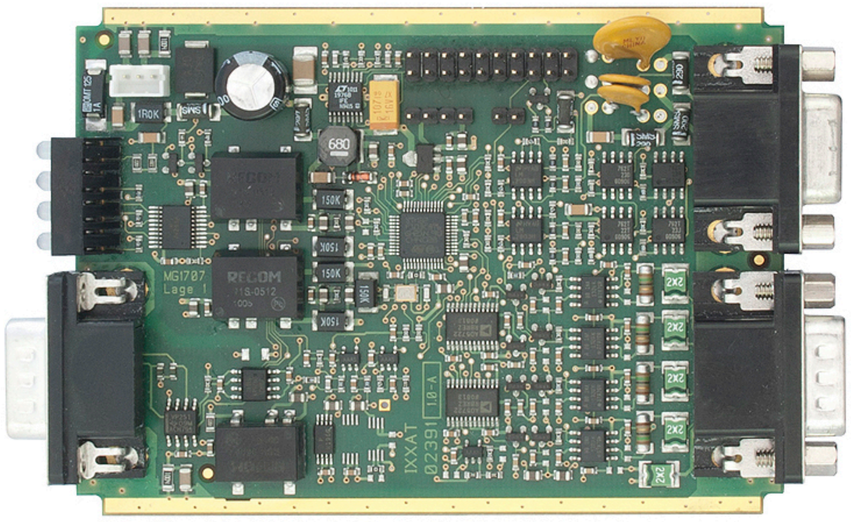
Manual

CANio ADK

Application Development Kit Manual



IXXAT

IXXAT Automation GmbH
Leibnizstr. 15
88250 Weingarten
Germany

Tel.: +49 751 56146-0
Fax: +49 751 56146-29
Internet: www.ixxat.com
E-Mail: info@ixxat.com

Support

In case of unsolvable problems with this product or other IXXAT products please contact IXXAT in written form by:

Fax: +49 751 56146-29
E-Mail: support@ixxat.de

Further international support contacts can be found on our webpage www.ixxat.com

Copyright

Document number: 4.01.0098.20002
Version: 1.2

1 Introduction

This manual for the CANio application development kit contains supplementary information for the delivered software and hardware. The manual also includes target-system-specific notes, a description how to start the demo application on a CANio hardware module and a description how to update the CANio firmware.

1.1 References

[1] Keil - Evaluation Software (https://www.keil.com/demo/eval/arm.htm)

[2] Keil - uVision (http://www.keil.com/uvision/)

[3] Keil - ARM Development Tools (http://www.keil.com/arm/realview.asp)

[4] Keil - ULINK (http://www.keil.com/ulink/)

[5] STMicroelectronics (http://www.st.com)

[6] ST - STM32 reference manual for application developers STM32F103 (http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/REFERENCE_MANUAL/CD00171190.pdf)

[7] IXXAT - CANio Config Tool (http://www.ixxat.com/can_canio500_analog_digital_can_modul_en.html)

[8] IXXAT – CANio 500 HW manual (http://www.ixxat.com/can_canio500_analog_digital_can_modul_en.html)

[9] IXXAT – CANio 250 HW manual (http://www.ixxat.com/can_canio250_digital_can_io_module_en.html)

1.2 Definitions, acronyms, abbreviations

CAN	Controller Area Network
API	Application Layer
I/O	Input/Output
ADK	Application Development Kit
SW	Software
HW	Hardware
LED	Light Emitting Diode
JTAG	Joint Test Action Group
USB	Universal Serial Bus

1.3 CANio hardware modules

All CANio hardware modules are equipped with a STM32F103 Micro controller from STMicroelectronics and provide a CAN controller, LEDs, digital and/or analog I/Os as well as a JTAG Debug connector. In the following the main differences of the CANio hardware modules are described:

- The **CANio 500** module provides both digital and analog I/Os:
  - 4 digital inputs and 4 digital outputs
  - 4 analog inputs and 4 analog outputs
- The **CANio 250** module provides a high number of digital I/Os:
  - 16 digital in- or outputs (configurable)
    - 8 digital inputs and 8 digital outputs.
    - 16 digital inputs
    - 16 digital outputs
- The **CANio 250 PlugIn** module is competly generic and can be used exactly in the way it is required by the user. It provides:
  - Up to 25 digital in- or outpus (configurable).
  - Up to 10 analog inputs (configurable).
  - Up to 2 SPIs (configurable).

1.4 About the CANio ADK

The CANio ADK is a universal application development kit for developing software on every CANio hardware module. When using the CANio ADK, the user can set the focus on the development of the application software. The hardware and the hardware drivers, a boot-manager and tools for updating the software already exist.

ADK highlights:

- SW library with hardware drivers for the CANio hardware modules (CAN, LED and digital/analog I/O drivers)
  - Standard user interface for the CANio250 PlugIn module
  - Extended user interface for the CNAio500/250 modules
- Bootloader for verifying, updating and starting the application software
- Up to 4 configurable SW timers for executing cyclic application tasks
- High resolution timer for measuring times
- Keil µVision4 demo application project with suitable development settings for the CANio 500/250 modules.
- Tool for updating the CANio application software via CAN

2 Contents of delivery

The CANio ADK consists of following components:

- Hardware
  - CANio hardware module
  - Power Supply cable
- Software
  - CANio ADK (ADK library, ADK demo application, ADK bootloader)
  - CANio Config Tool
- Documentation:
  - CANio ADK manual
  - Printed CANio hardware manual

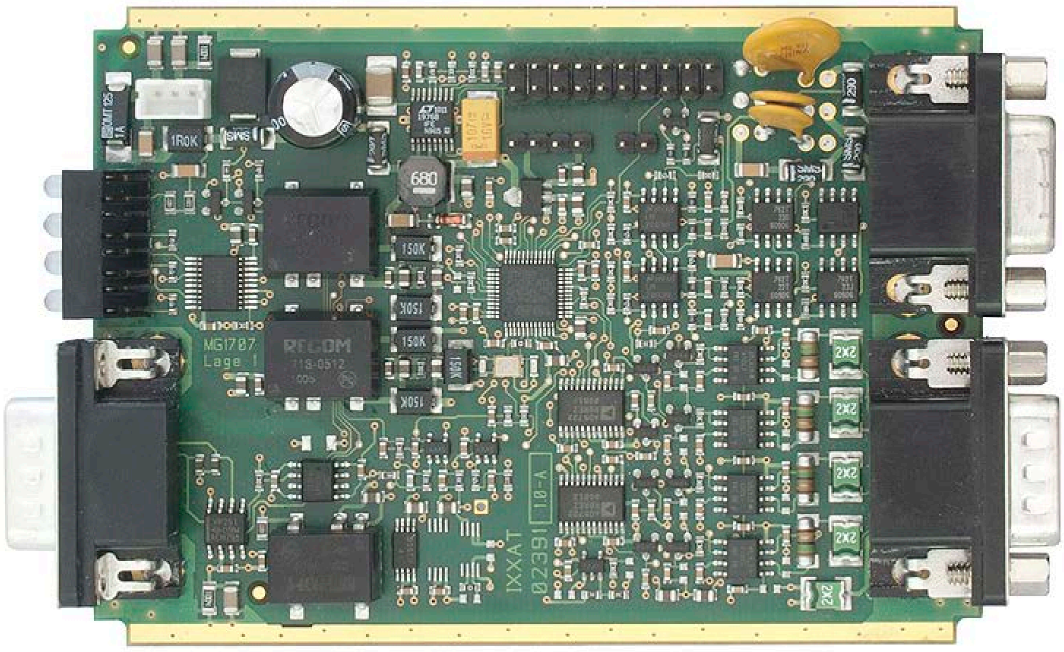


Fig. 2-1: CANio hardware module (the picture shows a CANio 500 module)

3 Software installation

3.1 Keil µVision4 development environment

For installing the Keil µVision4 development environment, download the current Keil µVision4 software from the Keil homepage [1] and run the setup program.

With this Keil evaluation software (the application software code area is limited to 32 Kbytes), it is possible, to compile, link and debug the delivered CANio500/250 ADK demo application.

Note, that the Keil µVision4 software is not freeware! For more information about the Keil evaluation software, please contact the Keil homepage [1].

3.2 IXXAT Software + Tools

3.2.1 CANio ADK

For installing the IXXAT CANio ADK, run the corresponding setup program "CANio ADK_V1.01.00.exe".

3.2.2 CANio Config Tool

For installing the IXXAT CANio Config Tool, run the corresponding setup program. The current version of the CANio Config Tool can be downloaded from the IXXAT homepage [7].

4 Development environment

The delivered software was implemented and tested with the Keil µVision4 development environment [2] together with the RealView MDK-ARM toolchain [3]. As debug and download interface a Keil Ulink2 USB-JTAG adapter [4] has been used.

The development of application software should be done on an IXXAT Hardware Module, since debugging is only supported on those platforms.

4.1 Software environment

For editing, compiling and linking Keil µVision4 and the RealView MDK-ARM toolchain, version 4.14 was used.

4.2 Flash memory layout

On the CANio the following Flash memory layout is used:

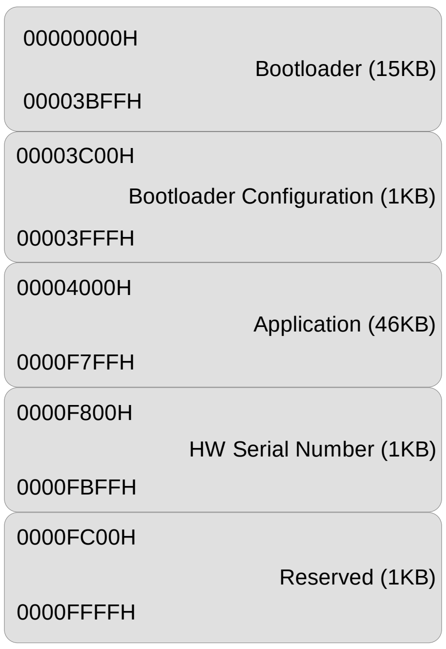


Fig. 4-1: CANio memory layout

4.2.1 Bootloader area

This Flash area contains the CANio bootloader and should not be erased. If the bootloader has been erased by mistake, the bootloader must be recovered (see chapter 7.1).

The bootloader is responsible for the following tasks:

- Verifying the application software via a stored checksum.
- Starting the application software.
- Updating the application software with the CANio Config Tool [7] via CAN.

4.2.2 Bootloader configuration area

This Flash area contains the application software checksum and the number of used Flash pages. The bootloader configuration sector is programmed automatically after a successful software update with the CANio Config Tool [7] via CAN.

4.2.3 Application software area

This Flash area contains the application software and the application configuration data. The Flash memory can be used individually by the user.

4.2.4 HW serial number area

This Flash area contains the hardware serial number and should not be erased.

4.2.5 Reserved

Reserved for further use.

4.3 CANio evaluation kit board layout

The different CANio modules are based on individual PCB layouts. For details on the external connector pin assignment, please refer to the corresponding hardware manual. Within the scope of this document, only the relevant connectors for the debugging with the Keil Ulink2 USB-JTAG adapter and the power supply are described in detail.

4.3.1 CANio500 evaluation kit

Fig. 4-2 shows the CANio500 evaluation board layout. For the IXXAT CANio500 Evaluation Kit, the CANio500 - Version with 0 to 10 V Analog Inputs (order number 1.01.0098.0000) is used.

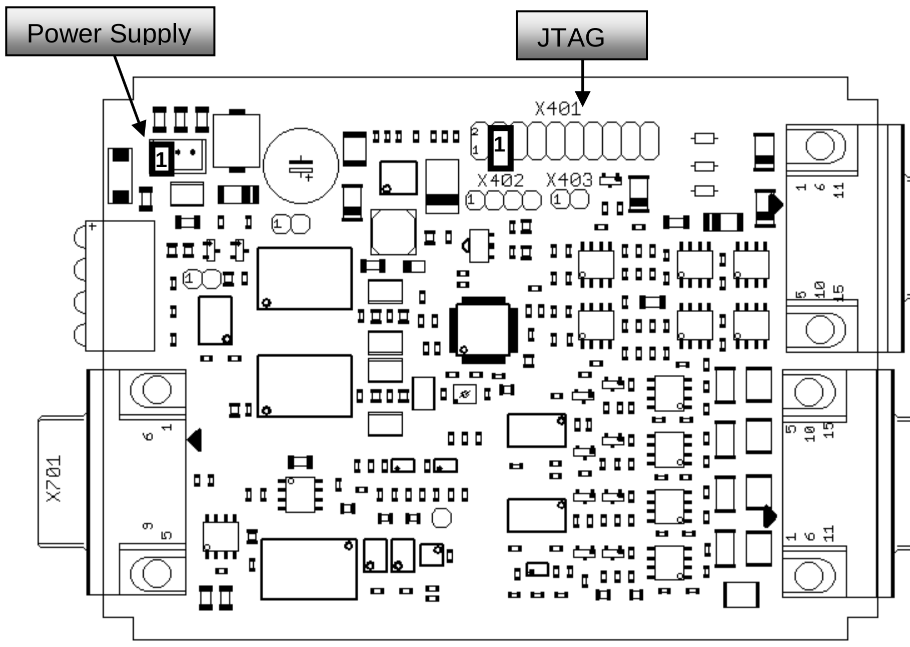


Fig. 4-2: CANio500 Evaluation Kit Board Layout

4.3.2 CANio250 evaluation kit

Fig. 4-3 shows the CANio250 evaluation board layout.

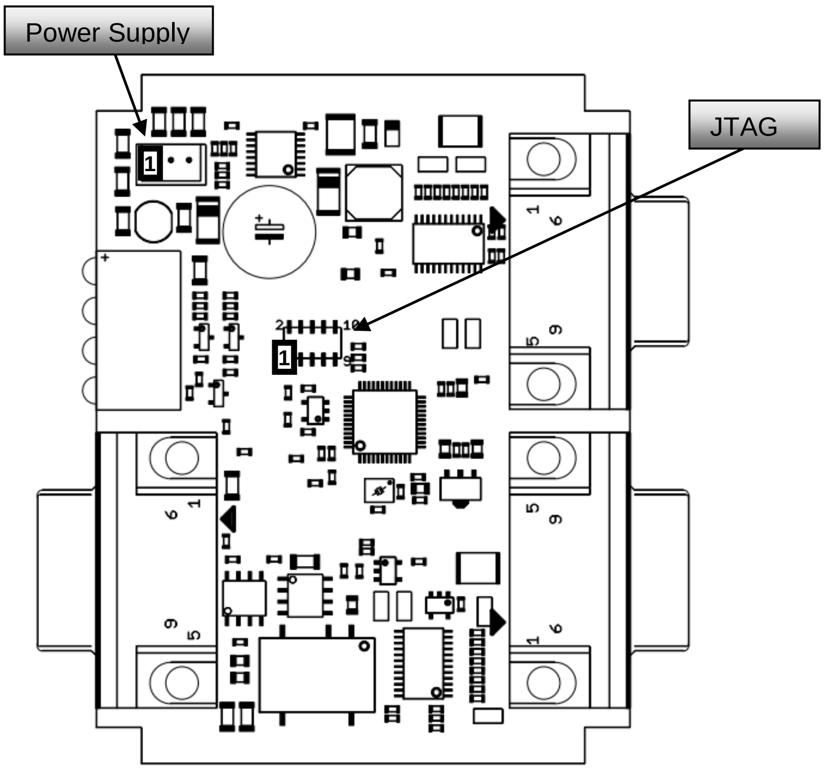


Fig. 4-3: CANio250 Evaluation Kit Board Layout

4.3.3 CANio250 Plug-In evaluation kit

Fig. 4-4 shows the CANio250 evaluation board layout.

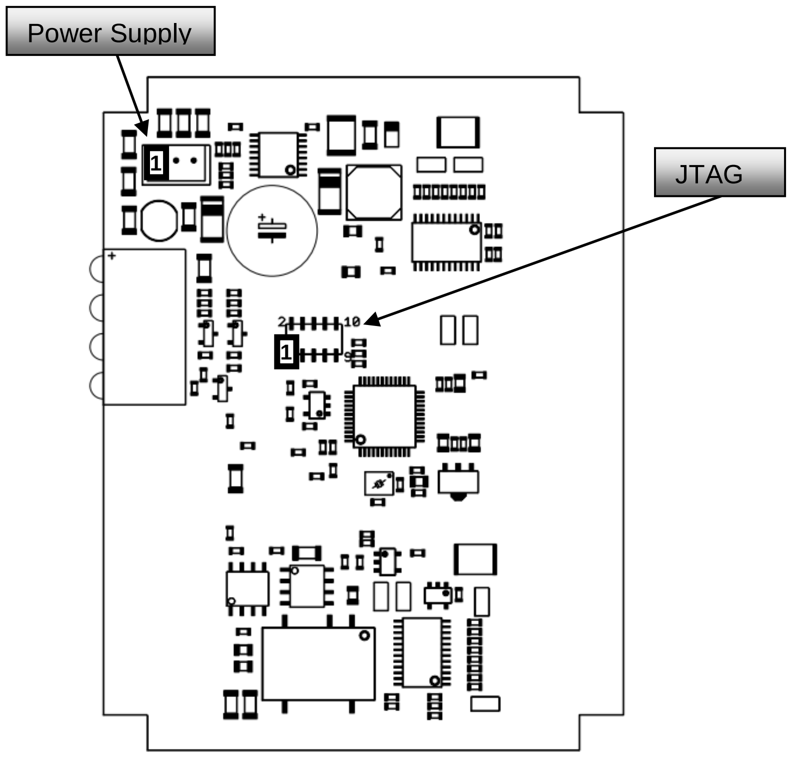


Fig. 4-4: CANio250 Plug-In Evaluation Kit Board Layout

4.3.4 Board installation

Connect the Power Suppy cable and the Keil Ulink2 USB-JTAG debugger to the CANio Evaluation Kit as shown in Fig. 4-5 (using the CANio500 Eval Kit as example)

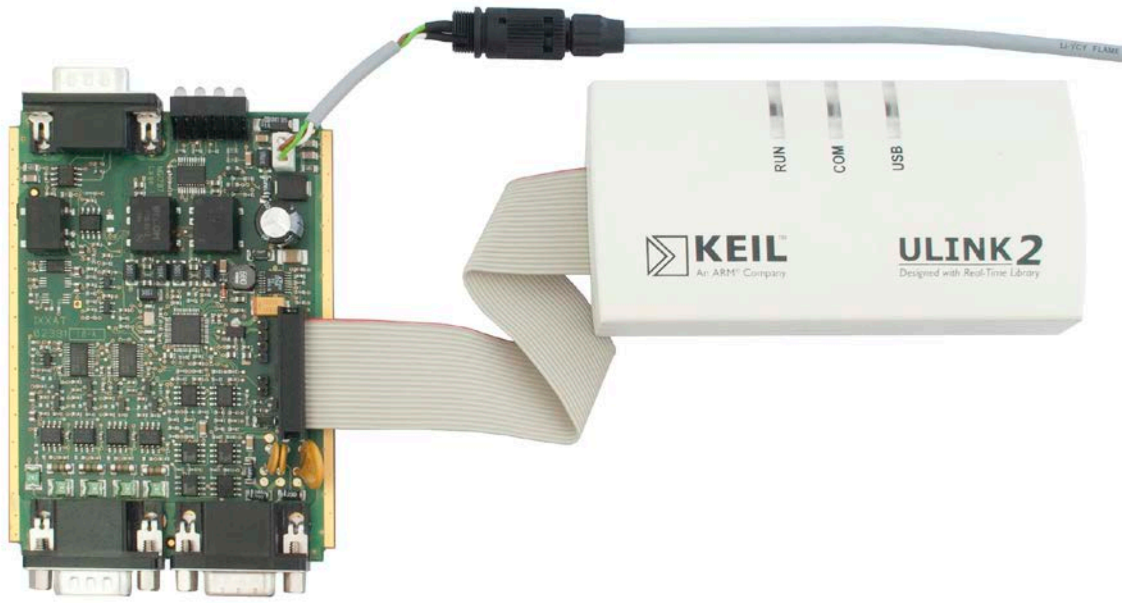


Fig. 4-5: CANio500 Evaluation Kit board installation example

5 Demo application

The ADK software is delivered with a demo application, which is located in the default sub-directory “IXXAT\CANio_ADK\ADK_DEMO” in the Windows program folder. The demo application is suitable for the CANio250/500 modules. Therefore the corresponding hardware has to be selected in ADKDemo.c.

```
#define CANIO500          0U    /* CANio 500 hardware */
#define CANIO250          1U    /* CANio 250 hardware */

/* select the used hardware (CANio 250/500) */
#define CANIO_HW_TYPE     CANIO250
```

A detailed description of the demo application is given in the ADKDemo.c source code.

5.1 Demo application - short description

The ADK demo application demonstrates the handling of the main functions of the ADK software.

5.1.1 Functions

- **Unit CAN**
  The demo application initializes the CAN controller and receives and transmits CAN messages. This CAN demo application is only working, if the CANio500/250 hardware module is connected to a CAN network with at least one additional CAN node. The CAN baud rate is set to 500 kBit/s.
- **Unit DIO**
  The demo application initializes the digital I/Os, reads values from the digital inputs and writes values to the digital outputs. If the input values change, a callback function is executed.
- **Unit AIO**
  The demo application initializes the analog I/Os, reads values from the analog inputs and writes values to the analog outputs.
- **Unit LED**
  The demo application initializes the LEDs and switches the LED LED_IDX_2 off, green or red.
- **Unit TIM**
  The demo application initializes the timer unit and configures up to 4 software timers.

5.1.2 Important defines

- **CANIO_HW_TYPE**
  Set the define CANIO_HW_TYPE to CANIO250 or to CANIO500 depending on the used CANio hardware module.
- **DEBUG_VERSION**
  Set the define DEBUG_VERSION to TRUE, when a debug version should be generated. If the define DEBUG_VERSION is set to TRUE, the bootloader configuration sector is programmed with default values. With these settings it is possible to start the application via the bootloader without verifying the application checksum. You can use this mode for developing and debugging this program. In the debug mode, the duo LED LED_TYPE_USR2 can only set to red or green (dependent on the JTAG signal level).
- **USE_ASSERT**
  Set the define USE_ASSERT to TRUE, when additional debug information is required. If the define is set to TRUE, all ADK return values are evaluated. If a unexpected return value is detected, the function HandleAssert() is called with the following parameters:
    - Source code filename
    - Source code line number
    - Text of the expression that was evaluated

5.2 Start the development environment

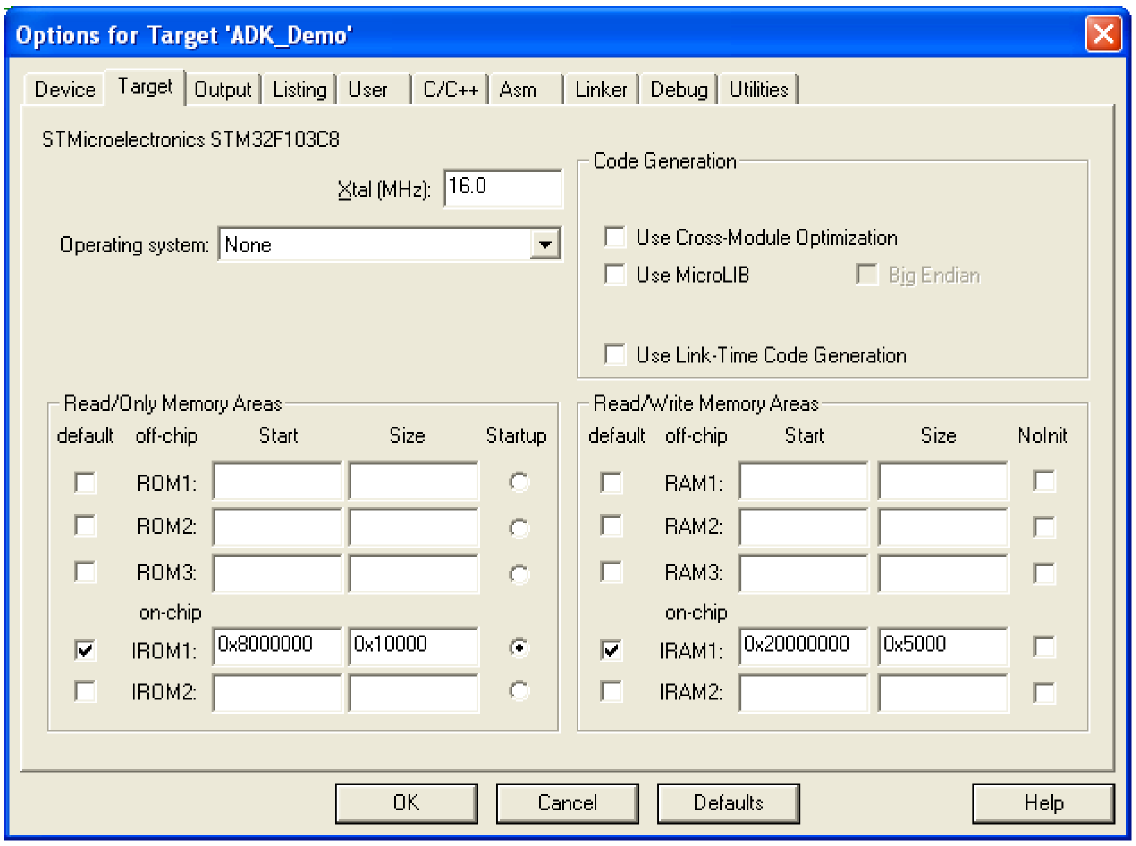
To compile, link and debug the demo application, the µVision IDE is used. You can launch the µVision IDE by clicking on the Keil µVision4 desktop icon or by selecting Keil µVision4 from the Start Menu.

When the µVision IDE is started, the demo application project file „ADK_Demo\Proj\ADK_Demo.uvproj“ can be opened in the menu “Project\Open Project“.

Additional information about the Keil µVision IDE can be found at http://www.keil.com/uvision [2].

5.3 Project settings

The target specific compiler options can be set in the menu "Project\Options for Target" in the Target tab (see Fig. 5-1). In the delivered project file ADK_Demo.uvproj these options are already set to the values, required by the IXXAT CANio500/250 hardware module.



STMicroelectronics STM32F103C8

Xtal (MHz): 16.0

Operating system: None

Code Generation

- [ ] Use Cross-Module Optimization
- [ ] Use MicroLIB
- [ ] Big Endian
- [ ] Use Link-Time Code Generation

Read/Only Memory Areas

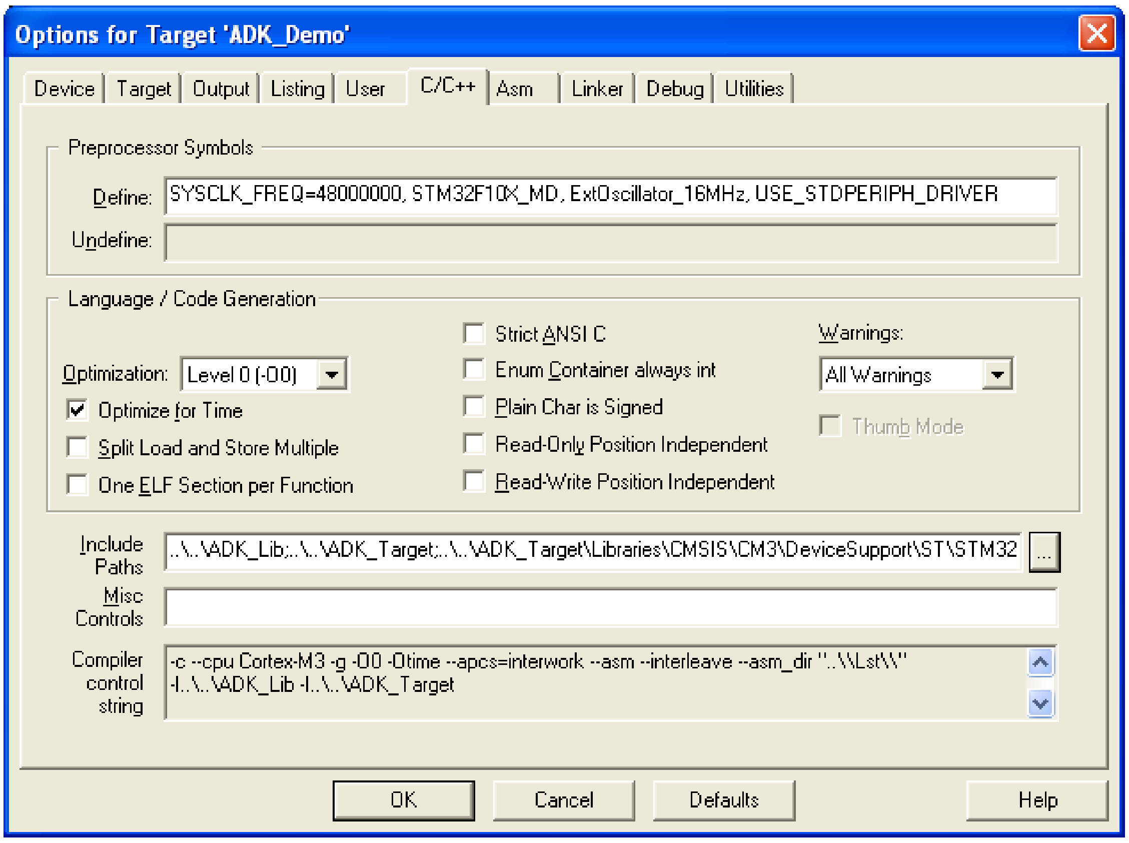
default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x8000000	0x10000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x20000000	0x5000	☐
☐	IRAM2:			☐

Fig. 5-1: Target settings of the demo application project

The C/C++ specific compiler options can be set in the menu "Project\Options for Target" in the C/C++ tab (see Fig. 5-2).



Preprocessor Symbols

Define: SYSCLK_FREQ=48000000, STM32F10X_MD, ExtOscillator_16MHz, USE_STDPERIPH_DRIVER

Undefine:

Language / Code Generation

Optimization: Level 0 (-O0)

[✓] Optimize for Time

[ ] Split Load and Store Multiple

[ ] One ELF Section per Function

[ ] Strict ANSI C

[ ] Enum Container always int

[ ] Plain Char is Signed

[ ] Read-Only Position Independent

[ ] Read-Write Position Independent

Warnings: All Warnings

[ ] Thumb Mode

Include Paths: ..\..\ADK_Lib;..\..\ADK_Target;..\..\ADK_Target\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32

Misc Controls:

Compiler control string:
```
-c --cpu Cortex-M3 -g -O0 -Otime --apcs=interwork --asm --interleave --asm_dir "..\\Lst\\"
-I..\..\ADK_Lib -I..\..\ADK_Target
```

Fig. 5-2: C/C++ settings of the demo project

Following target specific prepropressor defines exist:

- SYSCLK_FREQ=48000000:
  Micro controller system clock frequency
- STM32F10X_MD:
  Micro controller type
- ExtOscillator_16MHz
  External oscillator with 16 Mhz clock frequency is used
- USE_STDPERIPH_DRIVER:
  The STMicroelectronics peripherals driver is used

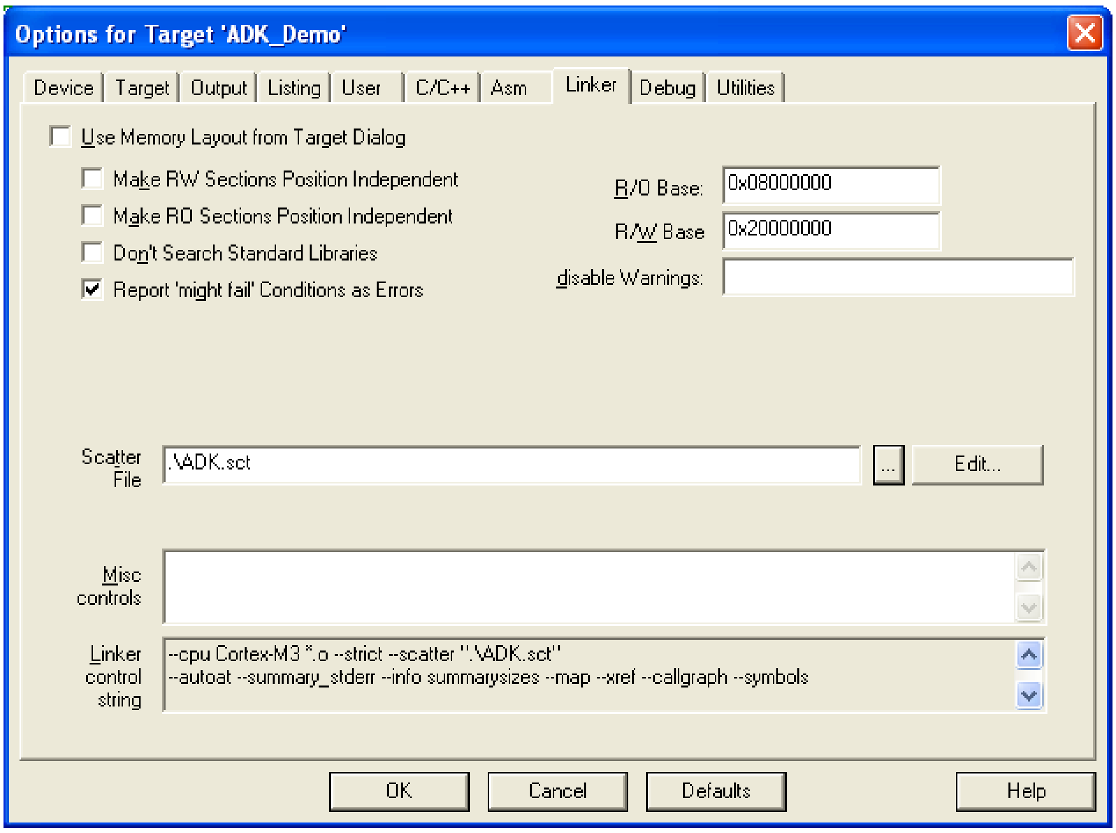
Following Include paths exist:

- ..\..\ADK_Lib
- ..\..\ADK_Target
- ..\..\ADK_Target\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x
- ..\..\ADK_Target\Libraries\STM32F10x_StdPeriph_Driver\inc

If a different directory structure is used for the software development, the above listed “include paths” to the folders ADK_LIB and ADK_Target have to be adapted. Additionally, the path to the ADK library file ADK_LIB.LIB in the project window has to be adapted. Click with the right mouse button on the file ADK_LIB.LIB and update the file path.

The project specific Linker settings can be set in the menu “Project\Options for Target” in the Linker tab (see Fig. 5-3). For the CANio ADK projects, the specific scatter file “ADK.sct” is used.

If a different directory structure is used for the software development, the scatter file “ADK.sct” has to be copied to the new software development location and the new path to the scatter file has to be updated in the Linker settings.



- [ ] Use Memory Layout from Target Dialog
- [ ] Make RW Sections Position Independent
- [ ] Make RO Sections Position Independent
- [ ] Don't Search Standard Libraries
- [✓] Report 'might fail' Conditions as Errors

R/O Base: 0x08000000
R/W Base: 0x20000000
disable Warnings:

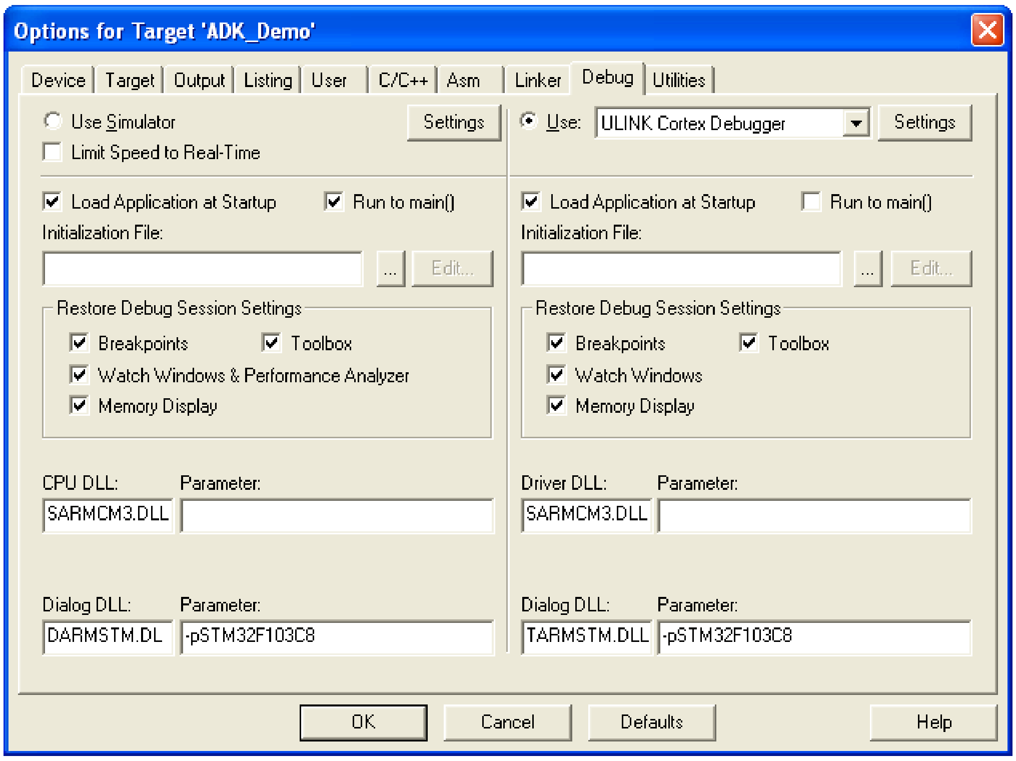
Scatter File: .\ADK.sct

Misc controls:

Linker control string:
```
--cpu Cortex-M3 *.o --strict --scatter ".\ADK.sct"
--autoat --summary_stderr --info summarysizes --map --xref --callgraph --symbols
```

Fig. 5-3: Linker settings of the demo project

To use the Ulink2 USB-JTAG adapter, select the Ulink ARM Debugger in the Debug tab (see Fig. 5-4).


Options for Target 'ADK_Demo'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

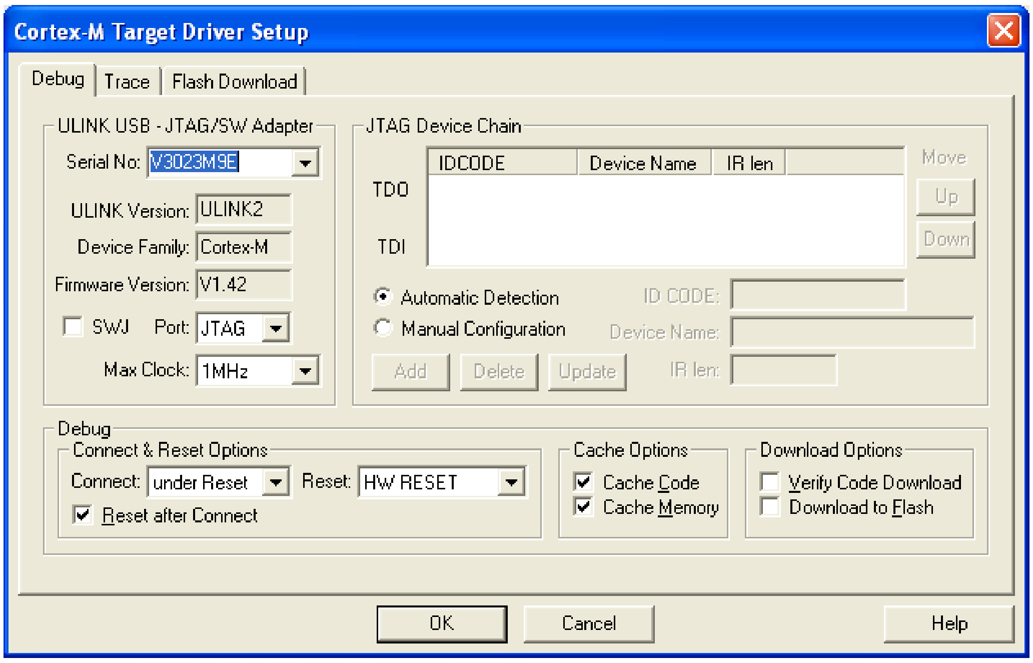
Use Simulator [ ] — Settings
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display
CPU DLL: SARMCM3.DLL Parameter:
Dialog DLL: DARMSTM.DL Parameter: -pSTM32F103C8

[✓ Use:] ULINK Cortex Debugger — Settings
[✓] Load Application at Startup [ ] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display
Driver DLL: SARMCM3.DLL Parameter:
Dialog DLL: TARMSTM.DLL Parameter: -pSTM32F103C8

OK | Cancel | Defaults | Help

Fig. 5-4: Debug settings of the demo project

By pressing the button “Settings”, the ULINK USB/-JTAG/SW Adapter” can be configured.


Cortex-M Target Driver Setup

Debug | Trace | Flash Download

ULINK USB - JTAG/SW Adapter
Serial No: V3023M9E
ULINK Version: ULINK2
Device Family: Cortex-M
Firmware Version: V1.42
[ ] SWJ Port: JTAG
Max Clock: 1MHz

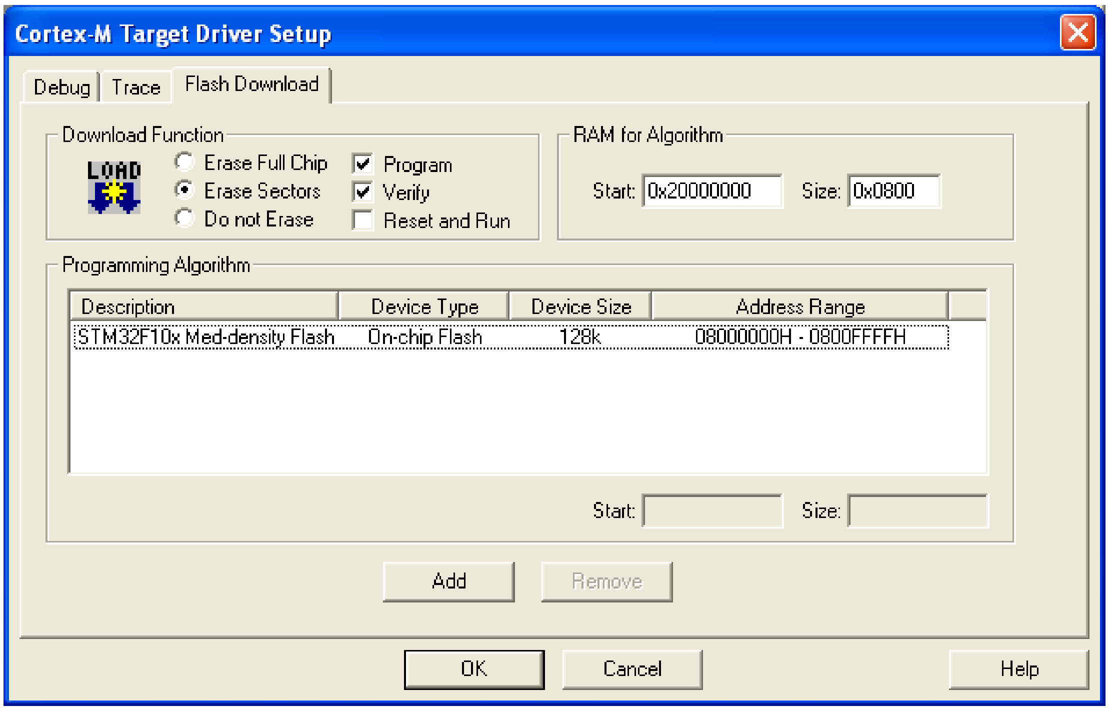
JTAG Device Chain
IDCODE | Device Name | IR len
TDO
TDI
Move Up Down
[✓ Automatic Detection] ID CODE:
Manual Configuration Device Name:
Add Delete Update IR len:

Debug
Connect & Reset Options
Connect: under Reset Reset: HW RESET
[✓] Reset after Connect
Cache Options
[✓] Cache Code
[✓] Cache Memory
Download Options
[ ] Verify Code Download
[ ] Download to Flash

OK | Cancel | Help

Fig. 5-5: Cortex-M Target Driver Setup – Debug settings of the demo project

For programming the STM32 Flash memory check if in the menu “Project/Options for Target/Debug/Settings/Flash Download” the “STM32F10x Med-density Flash” device is listed (see Fig. 5-6). If the Flash is not listed, add the “STM32F10x Med-density Flash” device.



Download Function: ( ) Erase Full Chip, (●) Erase Sectors, ( ) Do not Erase; [✓] Program, [✓] Verify, [ ] Reset and Run

RAM for Algorithm: Start: 0x20000000 Size: 0x0800

Programming Algorithm

Description	Device Type	Device Size	Address Range
STM32F10x Med-density Flash	On-chip Flash	128k	08000000H - 0800FFFFH

Start: Size:

Fig. 5-6: Cortex-M Target Driver Setup – Flash settings of the demo project

5.4 Build and start the demo application

To build and start the demo application the following directory structure is presumed:

```
+-ADK_Demo              // demo application directory
  +--PROJ               // demo application project file
  +--SRC                // demo application source files
  +--LST                // demo application list files
  +--OBJ                // demo application object files
  +--HEX                // demo application HEX file
+-ADK_LIB               // ADK library and header files
+-ADK_TARGET            // header file with IXXAT data types and
  |                     // modified STMicroelectronics files
  +--Libraries          // STMicroelectronics library files
```

The demo application project file for the ADK is located in the sub-directory “IXXAT\CANio_ADK\ADK_DEMO\PROJ”.

The demo application can be built in the menu “Project\Build target” or “Project\Rebuild all target files”. After the demo application has been successfully built, the application can be downloaded to the STM32 Flash memory via the Ulink2 USB-JTAG adapter in the menu “Flash\Download”. The connection of the Keil Ulink2 JTAG debugger on the CANio hardware module is described in chapter 1.3.

For easier usage of the microcontroller, STMicroelectronics provides a free library for accessing the CPU registers and peripherals. The IXXAT ADK supplies only these files of the library which are required to set up the ADK. You can find these files in the directory “TARGET\Libraries”. In case you require other files or complete documentation for your specific project, please verify the libraries provided with the compiler, respectively consult the STMicroelectronics homepage [5] and search for the keywords “STM32 Software Library”.

5.5 Debugging

The Debugger can be started in the menu “Debug/Start/Stop Debug Session”. If the debugger is active, the demo application can be started/stopped in the menu “Debug” by selecting “Run” or “Stop”.

When resetting the STM32 micro controller, the STM32 program counter is set to the IXXAT bootloader address. The bootloader then normally verifies the stored checksum with the calculated checksum of the application software and will start the application only, if the checksum is valid.

To disable the checksum calculation, the user has to set the define **DEBUG_VERSION** to the value **TRUE** (For further information, see the description in the source file “ADKDemo.c” of the delivered demo application project).

In this case, the application software checksum and the number of the used Flash pages are set to 0 and the application is always started without checksum calculation.

In the delivered demo application, the define **DEBUG_VERSION** is also used to initialize the LED unit. When the debug mode is active and the JTAG debugger is used, the 2nd user LED can not be controlled completely by the user (multiplexed JTAG/LED port).

5.6 Create a new Keil µVision4 project

For creating a new Keil µVision4 Project, the following steps have to be done:

- Create a new project in the menu “Project\New µVision Project” with the desired project name and project directory.
- In the next dialog window “Device for Target”, choose the CPU type STM32F103C8 from STMicroelectronics.
- Select “No” in the next message box. The startup code is already included in the ADK library file.
- Add the ADK library file to the project by selecting the Item “Target” in the project window and click the right mouse button. From the context menu, add a new group with the menu item “Add Group” and add the ADK library “ADK_LIB.lib” with the menu item “Add Files to Group”.
- Add all new source files to the project by selecting the “Source Group” in the project window and click the right mouse button. From the context menu add the source files with the menu item “Add Files to Group”.
- Adjust the following project settings, which are described in the chapter 5.3:
  - C/C++ specific compiler settings (Defines and Include paths).
  - Linker settings
  - Debug settings
  - Flash settings

6 System startup and bootloader activation

The system startup is shown in the system startup diagram (see Fig. 6-1). After power-up or reset, the bootloader checks if a valid application is available. This valid application will then be started, if a valid reset source in the STM32 status control register is found (Power Up or Reset) and the level of the digital input DigitInResPin is HIGH.

If the application is running, the bootloader can be activated by calling the ADK function JumpToBootLoader(). To prevent, that the application is started again, the STM32 status control register is cleared (Power Up and Reset).

DigitInResPin	: Reserved Pin (see also the CANio hardware manual [8])
Reset Source	: Reset information in the "STM32 Control/Status register"

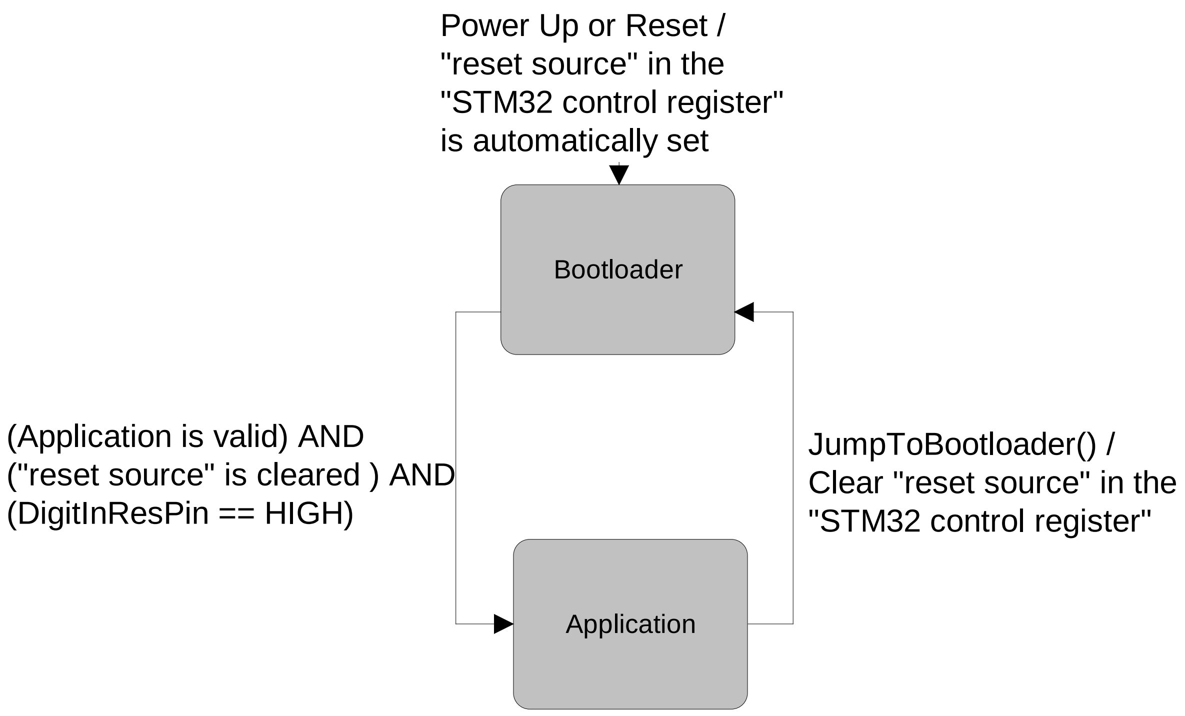


Fig. 6-1: System startup diagram

7 SW update mechanism

There are several mechanisms for programming the application software to the STM32 Flash memory.

7.1 Keil µVision4 development environment

Within the Keil µVision4 environment, the application software can be downloaded to the STM32 Flash memory via the Ulink2 USB-JTAG debugger. You can start the programming in the menu “Flash\Download”.

With this mechanism, there is no checksum generated. Therefore it is recommended, to set the define **DEBUG_VERSION** in the application software to the value **TRUE** for disabling the checksum calculation of the application software by the bootloader.

The CANio bootloader can also be updated by opening the Keil µVision4 project “ADK_Bootloader\Proj\ADK_Bootloader.uvproj”. You can start the programming of the bootloader in the menu “Flash\Download”.

7.2 IXXAT CANio Config Tool

With the IXXAT CANio Config Tool [8], an application can be downloaded to the STM32 Flash memory via CAN. The software update is only possible, if the CANio bootloader is running. Therefore a possibly active application software must call the function ADK_JumpToBootloader() (see demo application), or the DigitInResPin (see CANio hardware manual) must be set to ground before powering up the CANio module.

With this mechanism, an application checksum is generated and stored in the bootloader configuration Flash page. Therefore it is recommended, to set the define **DEBUG_VERSION** in the application software to the value **FALSE** for enabling the checksum calculation of the application software by the bootloader.

For further information the description of the IXXAT CANio Config Tool [8] is recommended.

8 ADK SW

8.1 Overview

The ADK supplies a library with several functions for developing software on a CANio hardware module. Fig. 8-1 shows all supplied ADK software units.

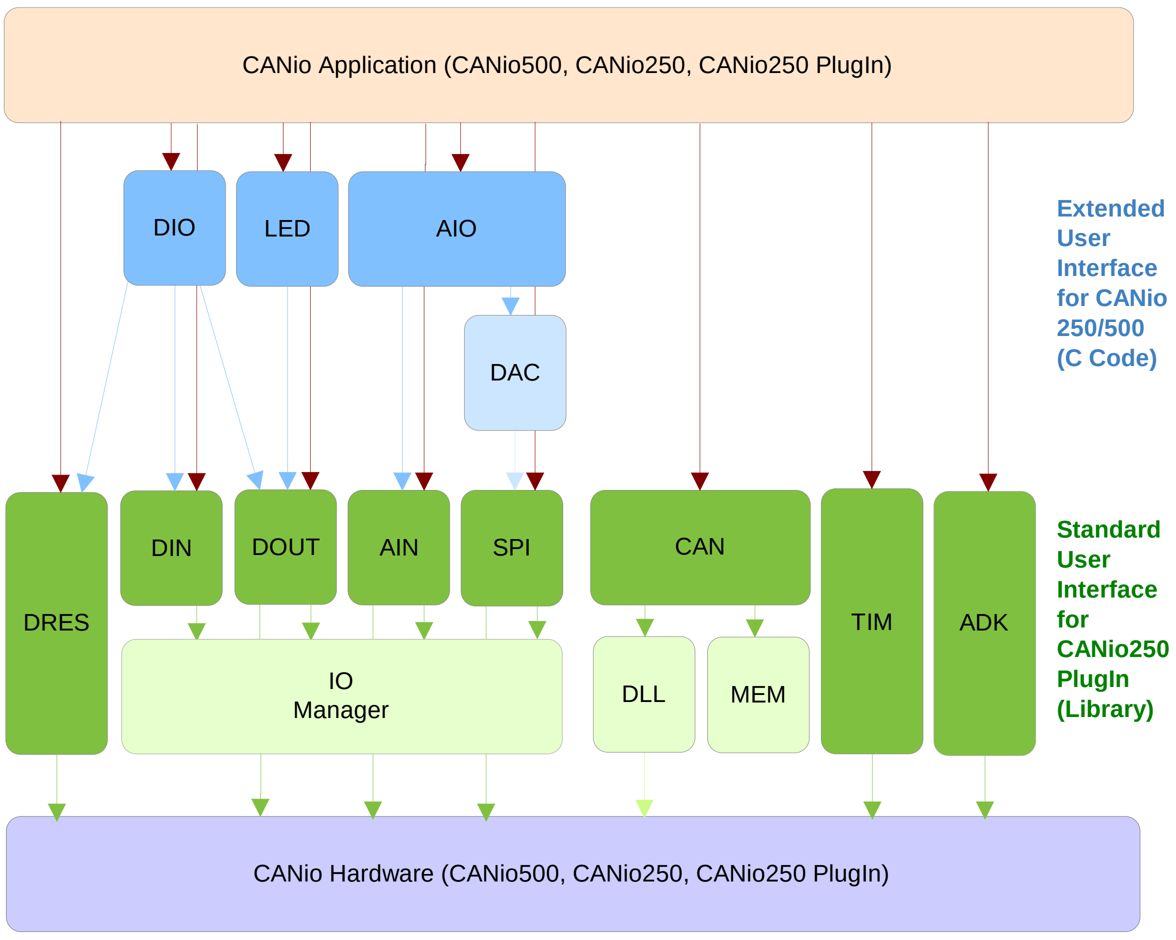

CANio Application (CANio500, CANio250, CANio250 PlugIn)
DIO
LED
AIO
DAC
Extended User Interface for CANio 250/500 (C Code)
DRES
DIN
DOUT
AIN
SPI
CAN
TIM
ADK
Standard User Interface for CANio250 PlugIn (Library)
IO Manager
DLL
MEM
CANio Hardware (CANio500, CANio250, CANio250 PlugIn)

Fig. 8-1: ADK SW overview

The ADK provides a standard user interface (as C library) for the CANio250 PlugIn module:

- Generic analog input driver (analog outputs can be realized via the SPI)
- Generic digital input/output driver (some digital Pins are reserved).
- Digital reserved driver (PB5/PB6 are reserved for the hardware type and I/O direction, PB7 is reserved for the test mode)
- SPI driver
- CAN driver
- Timer and common ADK functions

The ADK also provides an extended user interface (as C code) for the CANio250/500 modules:

- Platform specific analog input/output functions (based on the SPI driver)
- Platform specific digital input/output functions (based on the digital input/output and reserved drivers)
- LED functions (based on the digital output driver)

The standard and extended user interface is described in detail in the corresponding HTML description, which can be started in the Windows menu under "IXXAT\Canio_ADK\Doc\API".

Note that the standard user interface can be used on every CANio hardware module, but it is recommended to use the extended user interface on the CANio250/500 module. Therewith the development effort of CNAio250/500 applications can be strongly reduced, because of the platform specific I/Os functions, which are provided by the extended user interface. The in the delivery included demo application uses also the extended user interface. The demo application demonstrates also how the standard user interface can be used.

8.2 Callback functions

The ADK supplies two kinds of callback functions.

8.2.1 Timer callback functions

The user can configure several callback functions and software timers with a cycle time resolution of 1 milli second. The callback functions are called automatically by the function TIM_Main(), when a SW timer exceeds.

To keep the timer callback functions running, the user should call cyclically the function TIM_Main().

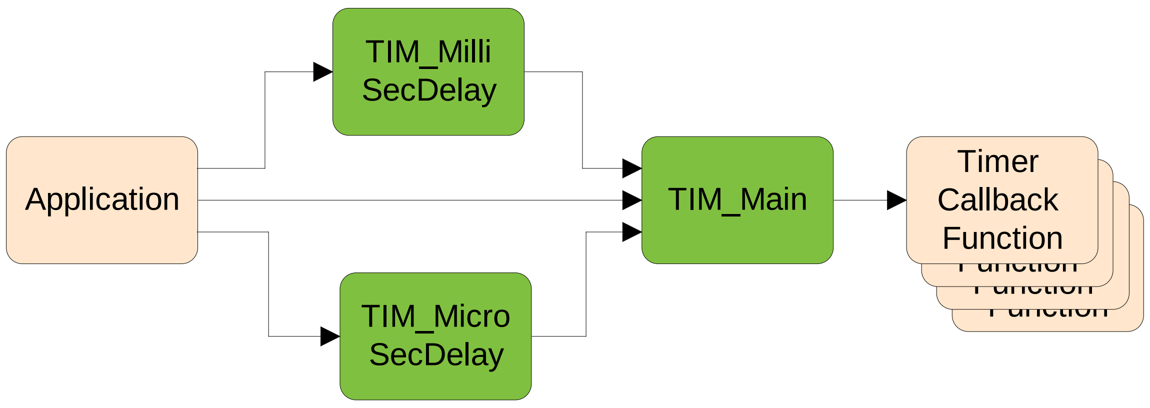


Fig. 8-2: Timer Callback Function data flow diagram

8.2.2 Digital input callback functions

The user can configure several callback functions for the digital inputs. The callback functions are called automatically by the ADK digital input interrupt handler, when the level of a digital input changes.

Since the callback function is running in the context of the digital input interrupt handler, the execution time of the function should be as short as possible.

In addition, it should be avoided to call the non reentrant functions of the units CAN and AIO. Instead of the reentrant functions of the units DIO, TIM and LED should be called.

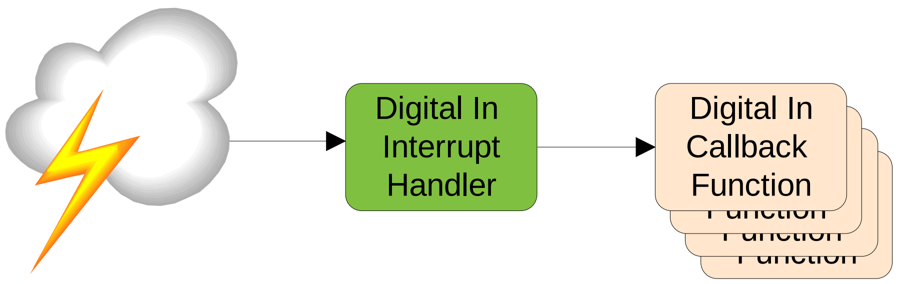


Fig. 8-3: Digital Input Callback Function data flow diagram

9 CANio ADK API

IXXAT will further extend and improve the CANio ADK API based on the feedback of the users. Thus the related detailed documentation would also been a subject for often changes. To prevent from providing a not actual manual, the detailed ADK API documentation is automatically created out the the underlying C-Code. IXXAT delivers this HTML based documentation as part of the ADK installation CD.

In case that a printed version is need, IXXAT can provide this on demand.

9.1 CANio standard ADK API

The standard CANio ADK API can be found in:

Start→Programs→IXXAT→Canio_ADK→ADK_Manual→STANDARD.

9.2 CANio extended ADK API

The extended CANio ADK API can be found in:

Start→Programs→IXXAT→Canio_ADK→Doc→ADK_Ext_Interface