DWM1001 Firmware User Guide

Based on DWM1001-DEV board

Version 1.4

This document is subject to change without notice

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

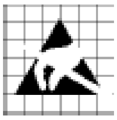
DOCUMENT INFORMATION

Disclaimer

Note: For the purpose of this document, “**DWM1001**” can also refer to “**DWM1001C**”. The only difference between both modules is that **DWM1001C** is certified and its OTP memory is calibrated to comply with FCC/ETSI regulations.

LIFE SUPPORT POLICY

Decawave products are not authorized for use in safety-critical applications (such as life support) where a failure of the Decawave product would reasonably be expected to cause severe personal injury or death. Decawave customers using or selling Decawave products in such a manner do so entirely at their own risk and agree to fully indemnify Decawave and its representatives against any damages arising out of the use of Decawave products in such safety-critical applications.



Caution! ESD sensitive device.

Precaution should be used when handling the device in order to prevent permanent damage

DISCLAIMER

(1) This Disclaimer applies to the software provided by Decawave Ltd. ("Decawave") in support of its DWM1001 module product ("Module") all as set out at clause 3 herein ("Decawave Software").

(2) Decawave Software is provided in two ways as follows: -
(a) pre-loaded onto the Module at time of manufacture by Decawave ("Firmware");
(b) supplied separately by Decawave ("Software Bundle").

(3) Decawave Software consists of the following components (a) to (d) inclusive:

(a) The **Decawave Positioning and Networking Stack** ("PANS"), available as a library accompanied by source code that allows a level of user customisation. The PANS software is pre-installed and runs on the Module as supplied, and enables mobile "tags", fixed "anchors" and "gateways" that together deliver the DWM1001 Two-Way-Ranging Real Time Location System ("DRTLS") Network.

(b) The **Decawave DRTLS Manager** which is an Android™ application for configuration of DRTLS nodes (nodes based on the Module) over Bluetooth™.

(c) The **Decawave DRTLS Gateway Application** which supplies a gateway function (on a Raspberry Pi ®) routing DRTLS location and sensor data traffic onto an IP based network (e.g. LAN), and consists of the following components:
- DRTLS Gateway Linux Kernel Module
- DRTLS Gateway Daemon
- DRTLS Gateway Proxy
- DRTLS Gateway MQTT Broker
- DRTLS Gateway Web Manager

(d) **Example Host API functions**, also designed to run on a Raspberry Pi, which show how to drive the Module from an external host microprocessor.

(4) The following third party components are used by Decawave Software and are incorporated in the Firmware or included in the Software Bundle as the case may be: -

(a) The PANS software incorporates the Nordic SoftDevice S132-SD-v3 version 3.0.0 (production) which is included in the Firmware and is also included in the Software Bundle;

(b) The PANS software uses the eCos RTOS which is included in the Software Bundle. The eCos RTOS is provided under the terms of an open source licence which may be found at: http://ecos.sourceware.org/license-overview.html;

(c) The PANS software uses an open source CRC-32 function from FreeBSD which is included in the Software Bundle. This CRC-32 function is provided under the terms of the BSD licence which may be found at: https://github.com/freebsd/freebsd/blob/386ddae58459341ec567604707805814a2128a57/COPYRIGHT;

(d) The Decawave DRTLS Manager application uses open source software which is provided as source code in the Software Bundle. This open source software is provided under the terms of the Apache Licence v2.0 which may be found at http://www.apache.org/licenses/LICENSE-2.0;

(e) The Decawave DRTLS Gateway Application uses the following third party components: -

(i) The Linux Kernel which is provided as source code in the Software Bundle. The Linux Kernel is provided under the terms of the GPLv2 licence which may be found at: https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html and as such the DWM1001 driver component of the DRTLS Gateway Application is provided under the same license terms;

(ii) The three.js JavaScript library, the downloadable version of which is available here https://threejs.org/, is provided under the terms of the MIT Licence which may be found at https://opensource.org/licenses/MIT.

Items (a), (b), (c), (d) and (e) in this section 4 are collectively referred to as the "Third Party Software"

(5) Decawave Software incorporates source code licensed to Decawave by Leaps s.r.o., a supplier to Decawave, which is included in the Firmware and the Software Bundle in binary and/or source code forms as the case may be, under the terms of a license agreement entered into between Decawave and Leaps s.r.o.

(6) Decawave hereby grants you a free, non-exclusive, non-transferable, worldwide license without the right to sub-license to design, make, have made, market, sell, have sold or otherwise dispose of products incorporating Decawave Software, to modify Decawave Software or incorporate Decawave Software in other software and to design, make, have made, market, sell, have sold or otherwise dispose of products incorporating such modified or incorporated software PROVIDED ALWAYS that the use by you of Third Party Software as supplied by Decawave is subject to the terms and conditions of the respective license agreements as set out at clause 4 herein AND PROVIDED ALWAYS that Decawave Software is used only in systems and products based on Decawave semiconductor products. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER DECAWAVE INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Decawave semiconductor products or Decawave Software are used.

(7) Downloading, accepting delivery of or using Decawave Software indicates your agreement to the terms of (i) the license grated at clause 6 herein, (ii) the terms of this Disclaimer and (iii) the terms attaching to the Third Party Software. If you do not agree with all of these terms do not download, accept delivery of or use Decawave Software.

(8) Decawave Software is solely intended to assist you in developing systems that incorporate Decawave semiconductor products. You understand and agree that you

remain responsible for using your independent analysis, evaluation and judgment in designing your systems and products. THE DECISION TO USE DECAWAVE SOFTWARE IN WHOLE OR IN PART IN YOUR SYSTEMS AND PRODUCTS RESTS ENTIRELY WITH YOU AND DECAWAVE ACCEPTS NO LIABILTY WHATSOEVER FOR SUCH DECISION.

(9) DECAWAVE SOFTWARE IS PROVIDED "AS IS". DECAWAVE MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO DECAWAVE SOFTWARE OR USE OF DECAWAVE SOFTWARE, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. DECAWAVE DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO DECAWAVE SOFTWARE OR THE USE THEREOF.

(10) DECAWAVE SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON DECAWAVE SOFTWARE OR THE USE OF DECAWAVE SOFTWARE. IN NO EVENT SHALL DECAWAVE BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, INCLUDING WITHOUT LIMITATION TO THE GENERALITY OF THE FOREGOING, LOSS OF ANTICIPATED PROFITS, GOODWILL, REPUTATION, BUSINESS RECEIPTS OR CONTRACTS, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION), LOSSES OR EXPENSES RESULTING FROM THIRD PARTY CLAIMS. THESE LIMITATIONS WILL APPLY REGARDLESS OF THE FORM OF ACTION, WHETHER UNDER STATUTE, IN CONTRACT OR TORT INCLUDING NEGLIGENCE OR ANY OTHER FORM OF ACTION AND WHETHER OR NOT DECAWAVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF DECAWAVE SOFTWARE OR THE USE OF DECAWAVE SOFTWARE.

(11) You acknowledge and agree that you are solely responsible for compliance with all legal, regulatory and safety-related requirements concerning your products, and any use of Decawave Software in your applications, notwithstanding any applications-related information or support that may be provided by Decawave.

(12) Decawave reserves the right to make corrections, enhancements, improvements and other changes to its software, including Decawave Software, at any time.

Mailing address: Decawave Ltd.,
Adelaide Chambers,
Peter Street,
Dublin D08 T6YA
IRELAND.

1 INTRODUCTION

1.1 Overview

The development of real-time positioning network using UWB technology is not a trivial project. The intention of DWM1001 is to simplify the development of UWB RTLS by providing a complete solution in a single module.

The DWM1001 module comes pre-loaded with embedded firmware which provides two-way ranging (TWR) real time location system (RTLS) functionality and networking. The module can be configured and controlled via its API, which can be accessed through a number of different interfaces allowing flexibility to the product designer. The details of the API are described in document [2]. Additionally, Decawave also provides the module firmware in the form of libraries and source code along with a build environment so that user can customise the operation and/or add their own functions.

This document describes what is included in the DWM1001 firmware and how these various elements cooperate together, and explains how users can add their own customisations. The aim of this user guide is to help the users with their development based on the DWM1001-DEV board. After reading this guide developers should be able to compile, build and run the DWM1001 firmware, including custom modifications.

1.2 What's included in the firmware release

Each DWM1001 firmware release includes:

1) The pre-compiled DWM1001 Firmware, including the firmware Positioning and Networking stack (PANS) library for on-board development, is provided in the DWM1001 on-board package. The firmware and library architecture and its basic partitioning are described in Section 2.
2) Firmware tool chain, described in Section 3. The detailed steps to prepare the necessary tool chain, download the firmware user application package, and compile/build/flash their own application are introduced.
3) Simple examples demonstrating how to use the DWM1001 APIs. These are described in Section 4.

2 Firmware Overview

2.1 High-Level Architecture

The firmware embedded in the DWM1001 module basically provides two types of functions: the PANS API and the PANS library which provides lower level functions. The PANS API, includes the Generic API, (these include different API sets for different interfaces and the corresponding parser, which acts as the translator between the user APIs (C, UART, SPI and BLE) and the PANS library). Figure 1 shows the architecture and components of the DWM1001 firmware.

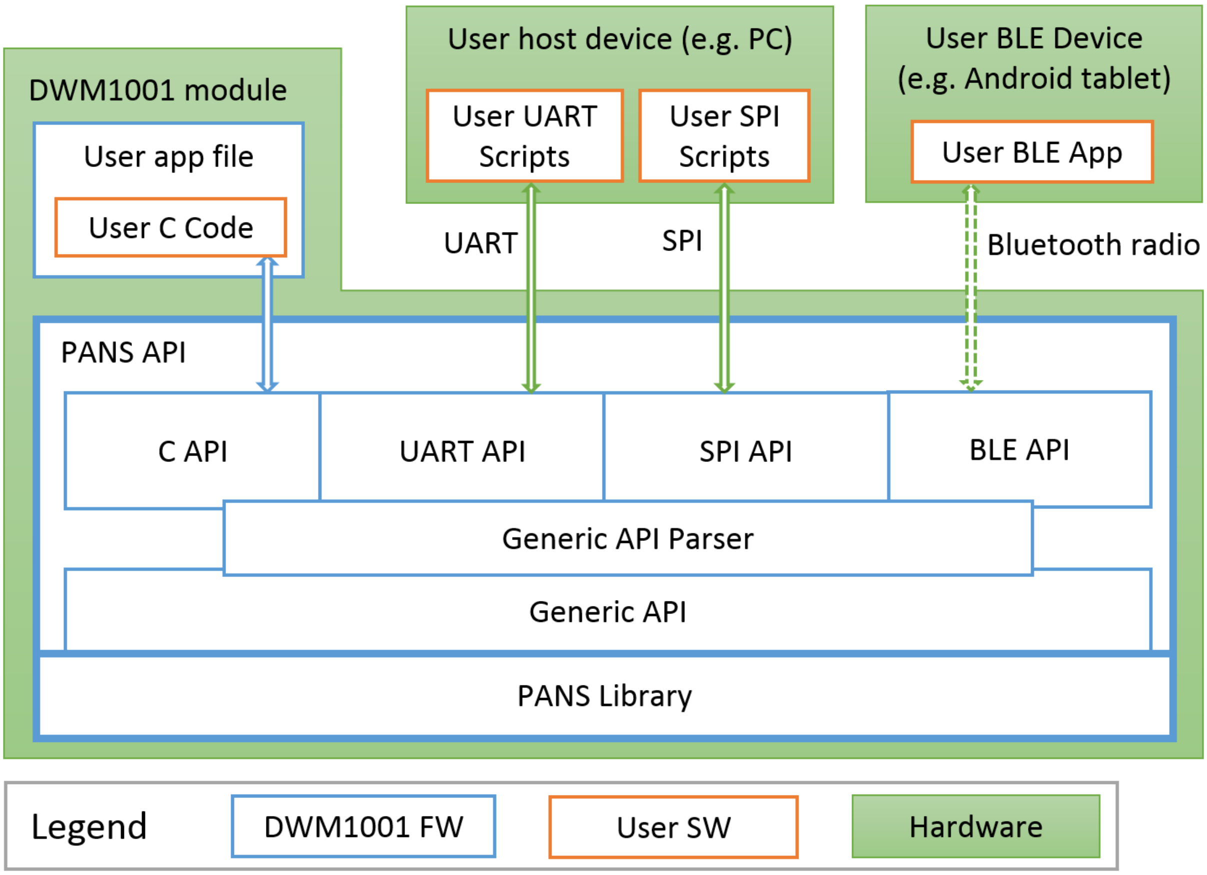


Figure 1 High-Level Architecture of DWM1001 Firmware vs. User Software

As can be seen in Figure 1, apart from using the DWM1001 module itself, the DWM1001 module can be physically connected to external controlling hardware, either wired or wirelessly over Bluetooth radio. The PANS API provides the users with four sets of API to call the PANS library functions through different interfaces:

- **User C code:** an on-board user space, allowing to include an application-specific code in the user application file provided in the DWM1001 firmware, using the firmware development tool chain provided by Decawave, see Section 4.2.

- **SPI:** using a host device (e.g. PC) to communicate with the DWM1001 module using TLV (Type-Length-Value format, detailed in [2]) format requests and responses through SPI bus, for configuration and data transmission.

- **UART:** using a host device (e.g. PC) to communicate the DWM1001 module through UART bus, for configuration and data transmission. Two modes of communications are provided over the UART interface: UART Generic mode using TLV format requests and responses; and UART Shell mode using terminal prompt commands.

- **BLE:** using a Bluetooth Low Energy (BLE) device (e.g. Android tablet) to control and configure through Bluetooth radio.

All these API sets provide the same set of generic functions calls, namely the Generic API. The Generic API Parser acts as the translator between the four API sets and the Generic API. When an API command is called from any of the above API Interfaces, the command goes through the Generic API Parser which translates the API command into Generic API function calls. If the API command needs a return message, the DWM1001 responds through the same interface.

The use of C code, UART and SPI APIs are detailed with simple examples in Section 4. More detailed information is provided in the API document [2].

Note1: The external interfaces, including the UART, the SPI and the BLE, are used by the external APIs in the PANS library for Host connection. The on-board user application through C code API cannot make use of the external interfaces due to compatibility reasons.

Note2: Decawave does not provide the library source code, or support any use of the PANS library except through the PANS API which is described in the API Document [2]

2.2 Overview of the PANS Library

Figure 2 illustrates the architecture of the PANS Library in detail. From bottom up the main components are the SoftDevice and the BLE Protocol stack from Nordic Semiconductor, eCos RTOS system with embedded drivers of the components, IoT layer protocols and the applications layer. Section 2.3 gives a brief introduction to each of the components in the library and the operations on the application layer.

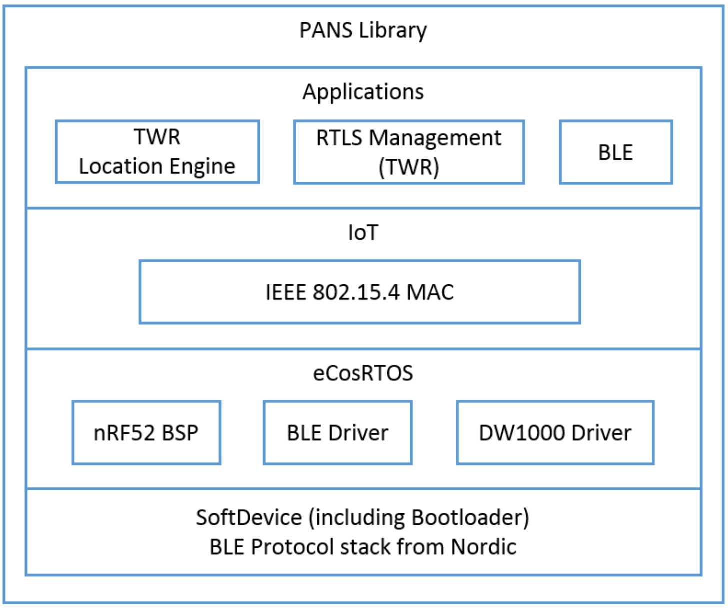


Figure 2 DWM1001 Firmware LibraryComponents

2.3 *Components and operations of the PANS Library*

The PANS library includes a few firmware components to drive the DWM1001 module. Some operations are implemented based on these components to perform the positioning and networking function.

2.3.1 SoftDevice and BLE driver

SoftDevice is a Nordic Semiconductor feature-rich library for BLE. The SoftDevice employed on the DWM1001 is the S132, a concurrent multi-link SoftDevice for Central, Peripheral, Broadcaster and Observer roles in BLE applications. The BLE driver / library is included in the S132 SoftDevice, providing the DWM001 with BLE features to create complex network topologies, communication and firmware update over-the-air [1].

2.3.2 eCos RTOS and BSP

eCos RTOS is a free open source real-time operating system. The eCos system provides very good run-time performance and the board support package (BSP) for the DWM1001 hardware platform. It includes all the necessary drivers for the module components (i.e. accelerometer, BLE & DW1000).

2.3.3 DW1000 driver

Decawave's DW1000 API driver. For details please see DW1000 Software API Guide [2].

2.3.4 Stationary indication

An accelerometer component LIS2DH12TR of slave address 0x19 as an I2C peripheral device is implemented on the DWM1001 module, i.e. the read and write address are 0x33 and 0x32 respectively. This accelerometer provides a simple "stationary" indication function. The DWM1001-based tag can be configured to use *Responsive* or *Low-Power* mode. It will be in *Low-Power* mode when stationary, and will enter *Responsive* mode when moving. The real-time accelerometer data is accessible through provided Shell API command and I2C functions in the C code API, see [2] for detailed information.

2.3.5 PANS Network operation

The PANS networking stack, allows discovery, joining and leaving. The UWB frames are sent according to 802.15.4 standard frame formats. The MAC layer functionality implemented on the DWM1001 module, as described in DWM1001 System Overview document [3], controls the mechanism for joining, leaving, installation, commissioning of nodes and associated two-way ranging protocol and data transfer. A number of different use cases e.g. follow-me, large-scale asset-tracking, navigation, home network are supported.

2.3.6 Commissioning

DRTLS can be commissioned with Decawave RTLS Manager Android application.
After power on, new nodes will advertise over Bluetooth. Decawave RTLS Manager application is used to connect to the node and configure it (e.g. its role as anchor or tag; its x,y,z coordinates if it is an anchor and other attributes).

2.3.7 RTLS Management

The RTLS Management application supports configuration of DWM1001 modules as tags and anchors which will participate in TWR. The tags will range to nearby anchors and use internal location engine to calculate position. The full implementation is detailed in the DWM1001 System Overview document [3].

2.3.8 TWR solver / Location Engine

The TWR solver / Location Engine in the tags calculates tag's x, y and z coordinates is implemented on the DWM1001 module. TWR results between the tag and the relative anchors are sent to the solver for the calculation of tag's position. The TWR results are accessible through the APIs. The internal location engine can be disable and user customised location calculation performed instead, e.g. adding extra filtering of change LE/solver algorithms.

3 FIRMWARE TOOL CHAIN

3.1 Tool Chain Overview

The Virtual Box image is now deprecated and the recommended methodology for firmware development is to use Segger Embedded Studio.

The Decawave firmware tool chain includes the following parts:

- Segger Embedded Studio and the GNU ARM Embedded Toolchain 5.4 2016q3
- DWM1001 on-board package.

Segger Embedded Studio is ideal for DWM1001 development as it offers a free commercial use license for the nrf51-52 Nordic Semiconductors MCU.

https://www.segger.com/news/segger-embedded-studio-ide-now-free-for-nordic-sdk-users/

The tool chain can be used for developing a new application with added functionalities which will reside in the DWM1001 module and will run on top of the PANS library and the main module functionality.

3.2 Content in the tool chain

The figure 3 shows the tool chain used in the DWM1001 firmware development.

The GNU ARM Embedded Toolchain (arm-none-eabi-gcc version 5.4.1 for cross-compatibility) and Segger Embedded Studio (SES) must be installed by user.

The distributed DWM1001 on-board package contains the user app source files and the libraries needed to compile and build the DWM1001 user firmware. A SES project is also provided for each example.

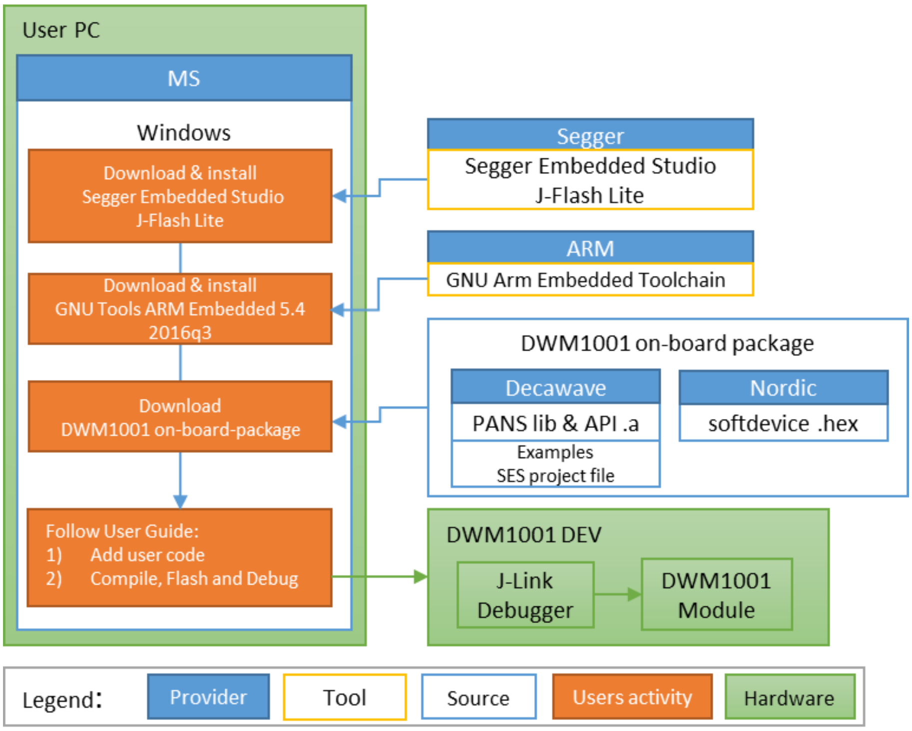


Figure 3: Tool chain and source components in DWM1001 firmware development

3.2.1 Hardware part of toolchain

As illustrated in green color in Figure 3, a PC with Microsoft Windows OS and a DWM1001-DEV board are required as the hardware. The DWM1001-DEV board provides the DWM1001 module as the target and a J-Link debugger.

3.2.2 Software part of toolchain

In order to perform user application development for dwm1001, the following software must be downloaded and installed on the windows computer. We recommend to use the default installation path.

- Segger Embedded Studio v4.12
  https://www.segger.com/downloads/embedded-studio

  Default installation path:
  `C:\Program Files\SEGGER\SEGGER Embedded Studio for ARM 4.12`

- Segger J-Flash Lite (J-Link software suite)
  https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack

  Default installation path:
  `C:\Program Files (x86)\SEGGER\JLink_V622g`

- GNU ARM Embedded Toolchain 5.4 2016q3
  https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads

  Default installation path:
  `C:\Program Files (x86)\GNU Tools ARM Embedded\5.4 2016q3\bin`

For cross-compatibility with the compiled PANS library, please ensure the version of the GNU ARM Embedded Toolchain corresponds to 5.4 2016q3 - Figure 4

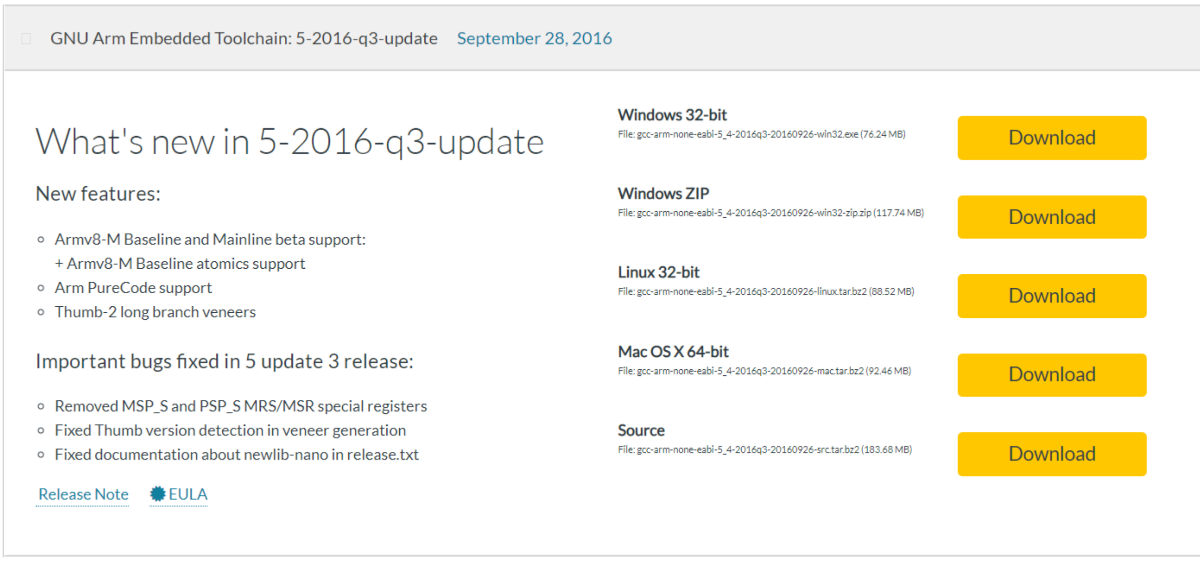


Figure 4: Version of GNU ARM Embedded Toolchain to download

3.2.3 Example application package for DWM1001

```
├── examples                                   // Examples directory
│   ├── dwm-simple-example                     // Simple Example
│   │   ├── dwm-simple.c                       // C source file
│   │   ├── dwm-simple-example.emProject       // SES project file
│   │   └── Output                             // Output directory
│   │       ├── compiler                       // Compiler output: .o
│   │       └── linker                         // Linker output: .elf
│   ├── dwm-range-iot
│   ├── dwm-timer
│   └── dwm-uart
├── include                                    // PANS Header file
├── lib                                        // PANS compiled library
├── nordicsemi                                 // Nordic SDK 12.1.0
├── recovery                                   // PANS reference binaries
└── utilities                                  // Firmware complementary images
    └── generate_example_hex                   // Softdevice driver for BLE
```

Figure 5: DWM1001 on-board-package structure

The DWM1001 on-board package is provided for download on Decawave website [5].

Refer to the Figure 5 for the detailed content of the DWM1001 on-board package.
Its main components are the DWM1001 PANS library, the firmware binaries such as NRF softdevice and bootloader, and the user application examples.

In DWM1001 firmware, the eCos library and other third-party software constitute the PANS library as introduced in Section 2.2. These source files are not provided in the on-board package.

The nordic semiconductor SDK for nrf52832 is also provided as users may need it for application development. Note that the sdk version used within PANS is: SDK v 12.1.0

3.3 Guides to flash the DWM1001 with factory image

DWM1001 comes with pre-flashed factory image of firmware. This image is provided in the DWM1001 on-board package: /dwm/recovery/DWM1001_PANS_R2.1.hex. The necessary steps to flash the factory image on the DWM1001-DEV board are described below.

The J-Flash Light tool can be used to flash the factory image through the DWM1001-DEV over a few different platforms.

1) Connect the module with a micro USB data cable, shown in Figure 6.

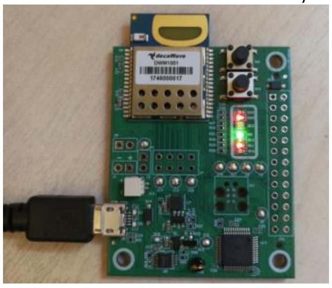
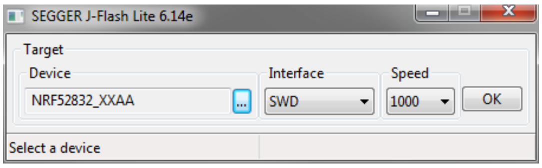
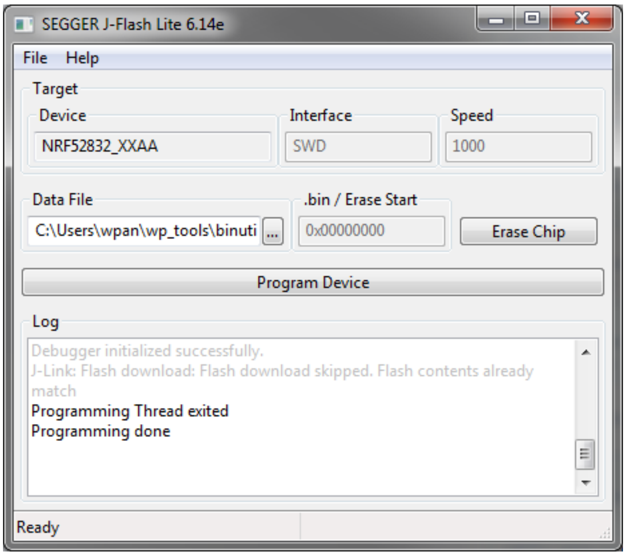


Figure 6 DWM1001 DEV board – micro USB connection

2) Flash the image DWM1001 module
   a. Open J-Flash Lite
   b. Choose nrf52832_XXAA as Device and SWD as interface, use default speed 1000. Click "OK".



   c. Click "Erase Chip" to do a full chip erase.
   d. In Data File, click ... and browse to the hex file provided in the DWM1001 on-board package (/dwm/recovery/DWM1001_PANS_R2.1.hex) to flash, click "Program Device".



The LEDs on the boards should be active once the flash update completes.

4 USER APPLICATION EXAMPLES

4.1 Overview

As illustrated in Figure 1, DWM1001 provides many ways to use its API functions. Examples that show the use of the APIs are listed in this section. In C code, UART Shell, UART Generic, and SPI, examples of getting the location of the node through the API are presented. The API document [2] provides more detailed information.

A tool to open serial port between host device and the DWM1001 module over UART is needed in the firmware development. In Windows, PuTTY can be used; in Linux with a Raspberry Pi for example, minicom can be used. The UART baud rate on the DWM1001 module is 115200 bps and the configuration is 8N1.

4.2 *"C code API" user application example*

"C code API" example is an application, which is running as part of the on-board firmware, utilizing a system resources of built-in to the module Cortex M4F microcontroller. The application is running as a thread application in multi-thread environment, driving by included to the PANS library eCos real-time operation system.

4.2.1 Firmware image partitioning

The flash size of the MCU (nRF52832) used on the DWM1001 module is 512KB from address 0 to 0x80000. The partitioning of the flash is illustrated as in Figure 7.

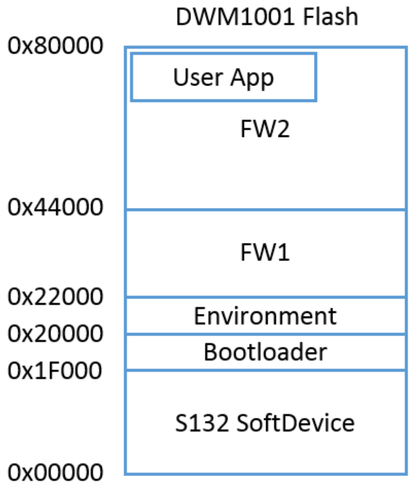


Figure 7 DWM1001 Flash Address Map

The DWM1001 firmware includes the following parts: Nordic S132 Softdevice, Bootloader, Environment, FW1 and FW2:

- **Softdevice**, of size 124KB starting from address 0, provides Bluetooth low energy central and peripheral protocol stack solution.
- **Bootloader**, of size 4KB start from address 0x1F000, is a firmware manager controlling the choice of FW1 and FW2 during booting/reseting.
- **Environment**, of size 8KB start from address 0x20000, is a flash section reserved for the firmware to store user configuration information. Doing power off/on, reset or firmware re-flash will not clear the Environment section in the flash. To clear the

environment section, a full-erase operation introduced in Section 3.3 is needed. Or alternatively, the “frst” shell command introduced in the PANS API [2] can be used.

- **FW1**, of size 136KB starting from address 0x22000, is a piece of firmware for the over-the-air (OTA) firmware update.

- **FW2**, size of up to 240KB starting from address 0x44000, is the firmware image that includes the full PANS library and the c code user application, where the user application is acting as an independent thread in the firmware and can occupy up to 3KB RAM and 60KB Flash. When rebuilding/reflashing the user application firmware, the whole FW2 is being operated.

4.2.2 Compiling/debugging user application in the firmware

To add user customized features, the user customized code needs to be added to the application files and the whole project needs to be re-built. A simple example of making a C code user application is given in examples/dwm-simple/dwm-simple.c. The dwm-simple example can be edited, compiled and debugged with Segger Embedded Studio.

4.2.2.1 Segger Embedded Studio tool chain path setup

Open the “examples/dwm-simple/dwm-simple.emProject” file with Segger Embedded Studio. The project should be loaded as in

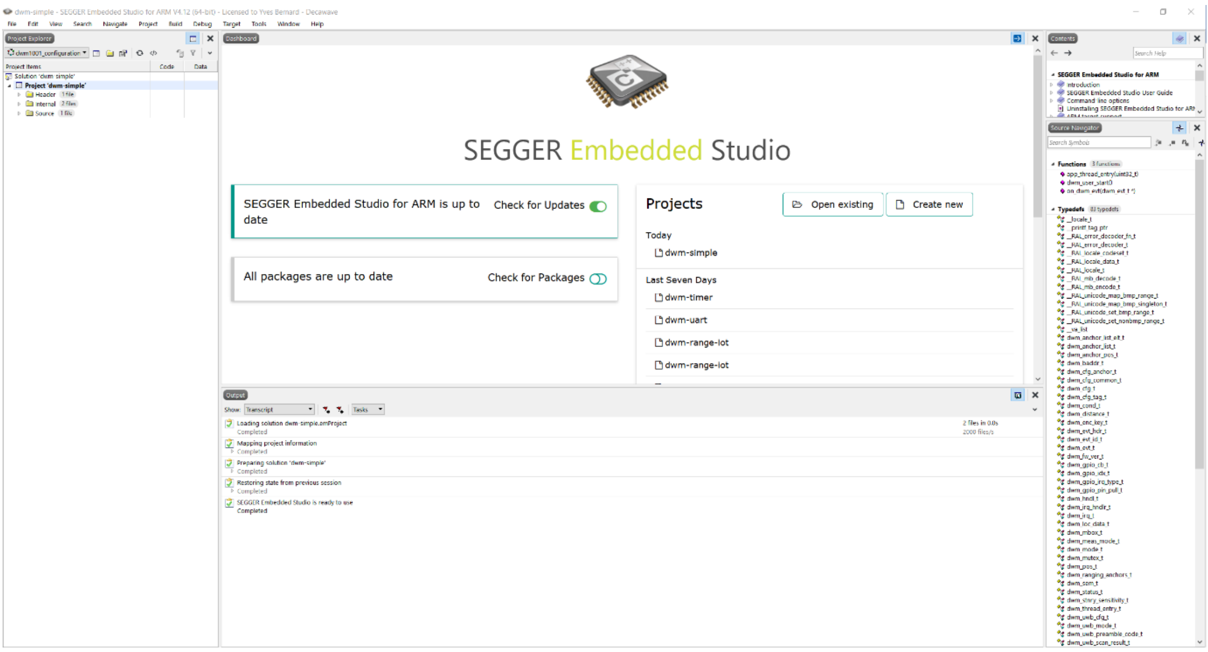


Figure 8: Segger Embedded Studio on project opening

PANS was initially developed and compiled GNU ARM Embedded Toolchain 5.4 2016q3. It is required to use the exact same version to avoid any retro-compatibility issues.

The first step consists in setting the compiler tool chain to the external GCC compiler installed previously as per section 3.2.2.

1. Right click on the solution name in the project item section, on the left of the main SES window. The solution name is at the top of the project structure: dwm-simple. See Figure 9.

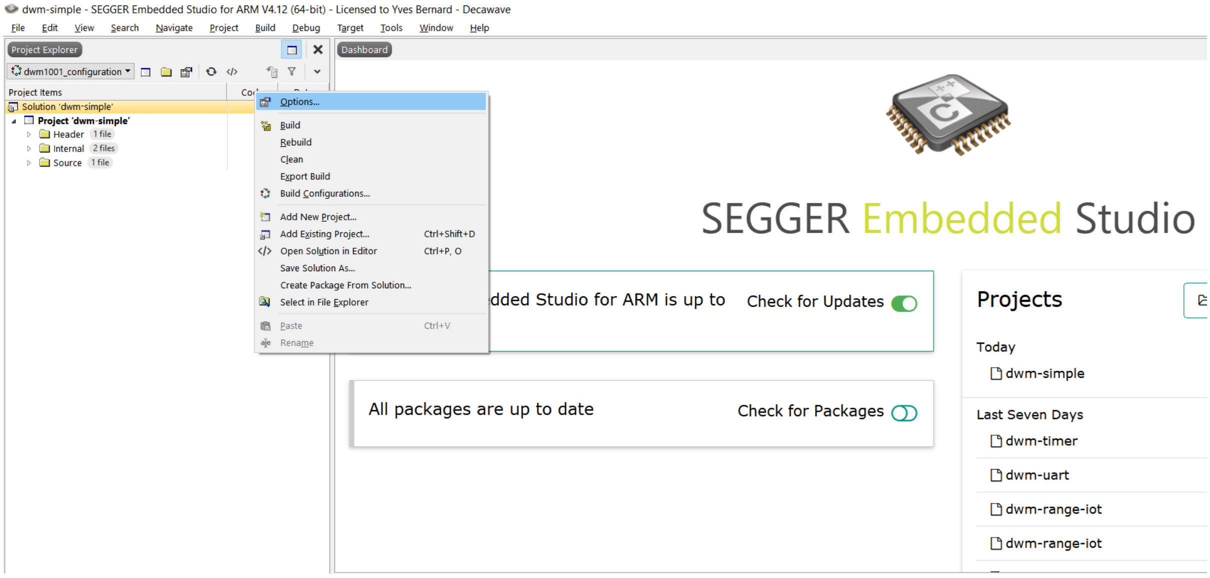


Figure 9: SES - Opening solution configuration menu

2. Click on "option" within the opening menu

3. Set the variable "Tool Chain Directory" to the path of the bin directory of the GNU Arm Embedded Toolchain previously installed. The Figure 10 shows the setting for the default install path.

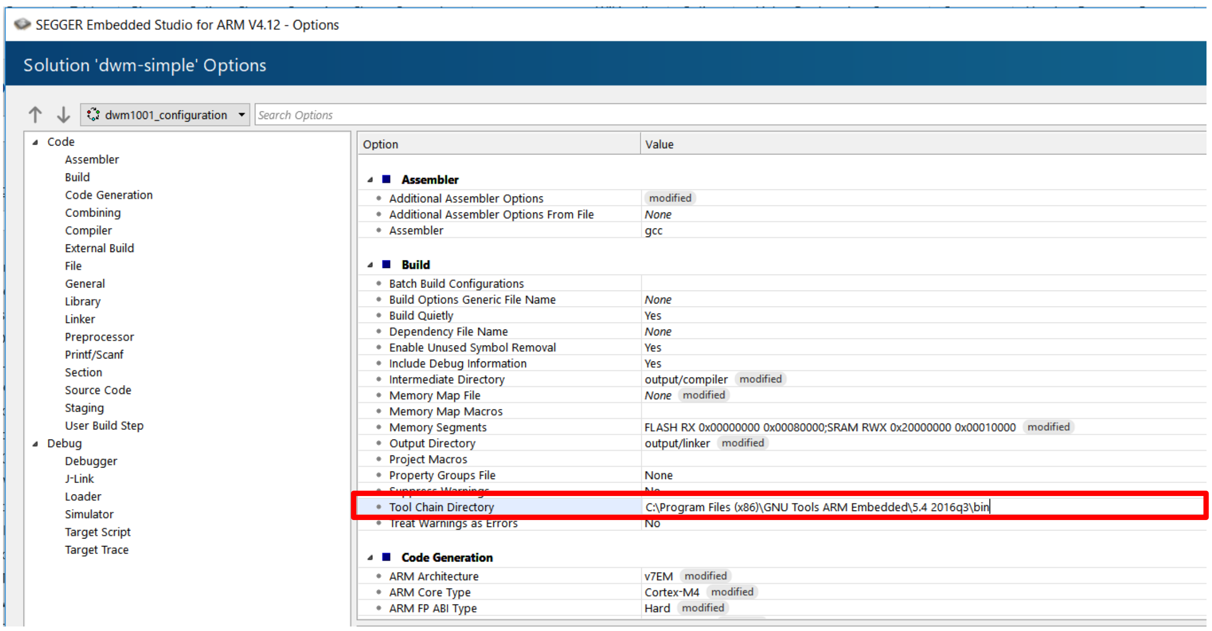


Figure 10: SES - Configuration of GNU ARM Embedded Toolchain install path

4. installation path. Click on ok to save the new value.

4.2.2.2 Segger Embedded Studio: Build and flash the user application

In order to build the project, open the “build” menu from the SES menu bar and click on “build dwm-simple” or press “F7”. The log for the build is displayed in the output windows. If the build is successful, the target mcu memory map is also displayed as information, with the occupied and remaining amount of flash/RAM memory. See Figure 11.

The “dwm-simple” example can easily be customized and recompiled by the user. Follow the instructions below to perform an initial PANS customization.

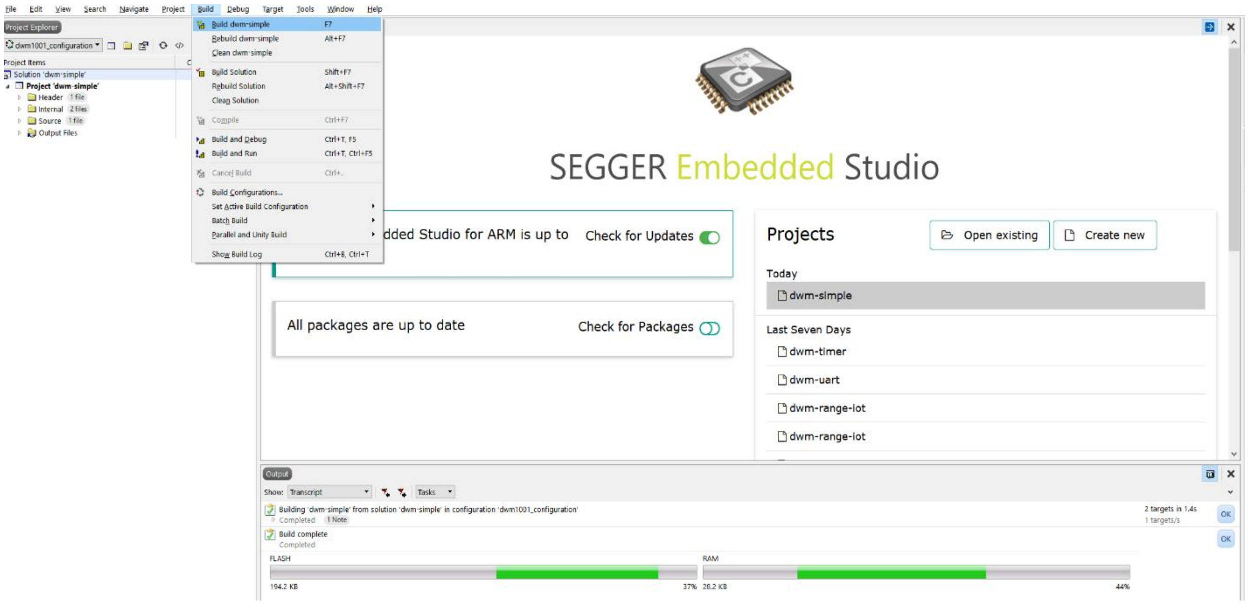


Figure 11: SES - Compiling the project

1. Within the dwm-simple.c file, find the function app_thread_entry

Within the app-thread-entry function:

2. Add the following to local variables:
```
dwm_pos_t pos;
```
3. Find while(1) and add in the brace:
```
dwm_pos_get(&pos);
printf("x=%ld, y=%ld, z=%ld, qf=%u \n", pos.x, pos.y, pos.z,
pos.qf);
printf("\t\t    time=%lu \n", dwm_systime_us_get());
```

Note: printf will send the message to through UART interface when Shell mode is enabled.
Save.
Build the project and eliminate errors.

This change intends to read the position information in the device and print the message on the terminal. A system timer is added in the end of the message to indicate the message time.

The source file dwm/examples/dwm-simple/dwm-simple.c can be customized to add/modify functionalities. All available C code API functions are listed in header file dwm/include/dwm.h. More detailed information of the C code API is introduced in the DWM1001 API Guide [2].

The compiled example can be flashed on the target device directly from SES. Plug the DWM1001-Dev device over USB to the computer, and click on "target" from the menu bar. The option "Download dwm-simple" can be used to load the example.

The compilation output can be found in the examplesd/dwm-simple/Output directory.
If the compilation is successful, then SES will create a compiled image:
/dwm-simple/Output/linker/dwm-simple_fw2.bin

Please note that SES is only compiling and producing an image corresponding to FW2. If the user wishes to flash the full PANS library on a blank DWM1001 device, then the softdevice, bootloader and FW1 will have to be flashed before FW2 as per described in section 4.1.

In order to simplify this task and generate a unique hex file, a batch script is provided in the /utilities directory.

Open the generate_example_hex.bat file with a text editor and alter the following fields depending on the project you are currently modifying and compiling.

```
SET example_name="dwm-simple"
SET fw2_path="..\examples\dwm-simple\Output\linker\dwm-
simple_fw2.bin"
```

Save the file and execute the generate_example_hex file by double clicking on it.

A new image file combining the softdevice, bootloader, FW1 and customized FW2 will be created:
`/dwm-simple/Output/dwm1001_dwm-simple.hex.`

This new file can be flashed using J-Lite as explained in Section 3.3.

4.2.2.3 Segger Embedded Studio: Debug the user application

SES also supports debugging. In order to start the debugging mode, click on "Debug" from the menu bar or press F5.

Note that a DWM1001-Deb must be connected to the computer to start the debugging mode successfully. The device is automatically flashed with the latest compiled project when starting debugging.

As presented by Figure 12, SES has the default options and features usually offered by a debugger. With the center widget, the user can setup break points and run the application through the code. The equivalent assembly program is displayed on left widget.
On the top right widget, note that the default breakpoints list contains the ARM V7M

exceptions.

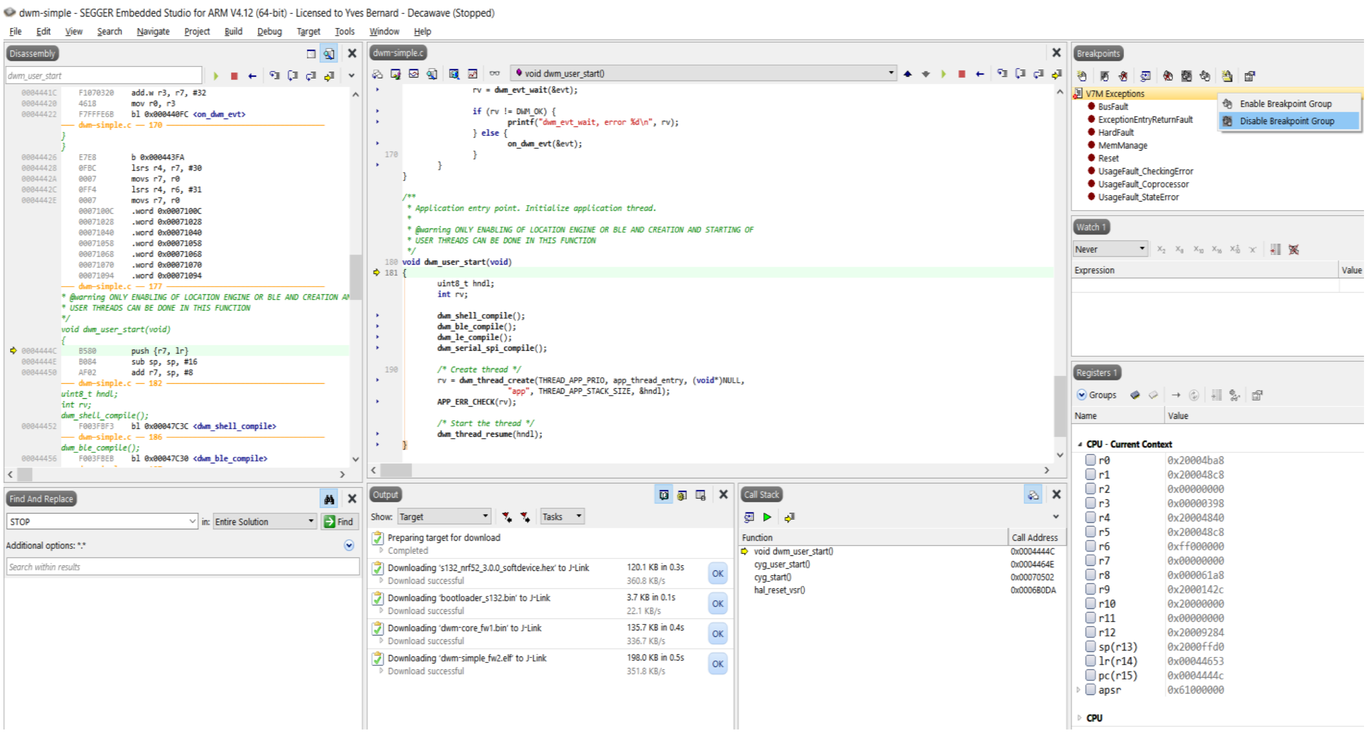


Figure 12: SES - Debugging window

We recommend to disable the V7M exceptions as they will be automatically triggered within the compiled library and will interferer with the user application debugging. In order to disable them, right click on "V7M exception" as shown on Figure 12.

Note1: A few compilation options are provided in the dwm_user_start() function, with function names of dwm_XXXX_compile(), where XXXX is the component name. Disabling these functions can disable the corresponding components in the DWM1001 firmware.

Note2: It is possible to use breakpoints to debug the firmware. However, the user has to be very carefull because the nRF52 softdevice interrupts are of highest priority, and if there is BLE activity its interrupts will conflict with user interrupt, thus the BLE should either be disabled or its interrupts masked. Disabling function dwm_ble_compile() can disable the BLE compilation so as to void BLE operations during debugging. The function can be re-enabled after debugging.

Note3: SWO debug printf is not supported on the DWM1001 DEV board.

4.3 UART applications example

The PANS library provides API functions through UART interface. The connection and simple examples are introduced here.

4.3.1 UART connection

The DWM1001 DEV board provides UART access through both the USB connector and the pins on the external connector. Both accesses are introduced here.

4.3.1.1 UART connection through COM port over USB

The UART connection can be setup simply through a USB data cable as shown in Figure 6. To find the device name of the DWM1001 DEV board in the Windows system:

1) Open Device and Printers,
2) Find the device J-Link:

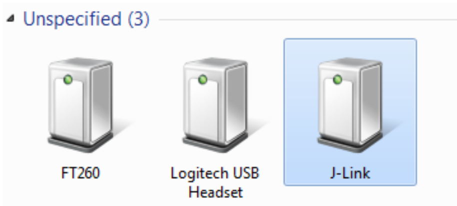


Figure 13 J-Link Device in Windows

3) Double click the J-Link icon, go to Hardware tab and find the COM port with number as the device name (e.g. in Figure 14 below J-Link is device name is COM28).

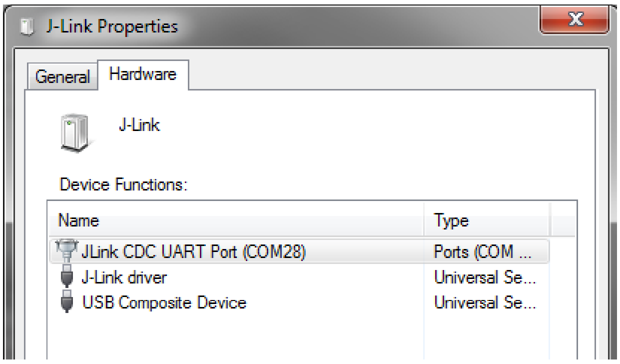


Figure 14: DWM1001 Device COM port over J-Link

In different Linux systems, the UART devices may show different names.

4.3.1.2 UART connection through external connector Tx and Rx pins

Other than using USB cable to connect, the external header pins provided on J10 of the DWM1001 DEV board also provides the UART interface. Table 1 shows the pins needed in the UART connection.

Table 1 UART pin connections

Pin to use in DWM1001 DEV		Pin to be connected to
Connector J10	Function	
Pin 6	GND	GND
Pin 8	RXD	TXD
Pin 10	TXD	RXD

For Raspberry Pi 3 using header pins connection, the device name is /dev/serial0.

The connection between the DWM1001 DEV board and a Raspberry Pi 3 (model B) is shown in Figure 15Figure 15.

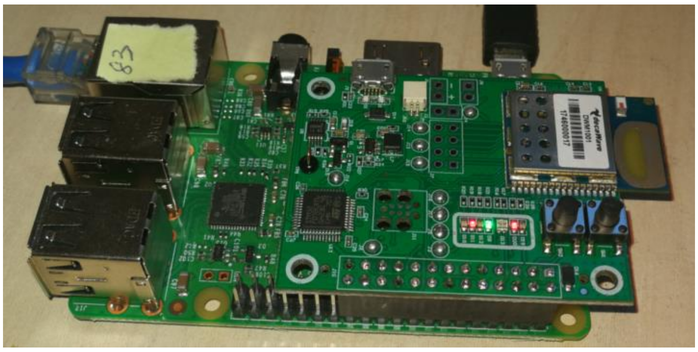


Figure 15: Connecting DWM1001 to Raspberry Pi 3 over header pins

Note: Pins on J10 of the DWM1001 DEV board are compatible with Raspberry Pi 3 connector J8 header pins 1-26.

4.3.2 UART examples

UART interface is accessible through two mode, the Shell mode and the Generic mode. Both Generic mode and Shell mode can be used to communicate with the DWM1001 module through the UART connection. The default mode of the DWM1001 UART is Generic mode. The two modes are transferrable:

Generic mode to Shell mode: presses “Enter” twice within one second, or input two bytes [0x0D, 0x0D] within one second.

Shell mode to Generic mode: input “quit” command.

For more information of the two UART API modes, please refer to DWM1001 API Guide [2].

4.3.2.1 UART Shell mode example

On a raspberry pi, in order to connect to the device, perform the following:

- minicom -D /dev/serial0

When seeing “Device or resource busy”, try multiple times until it works.

If the connection over UART is successful, “Welcome to minicom” message will show on the terminal. Now hit “Enter” key twice within one second to enter UART shell mode. “dwm>” should present in the terminal when this is all done.

To run the UART Shell command for dwm_pos_get, type “apg” followed by “Enter” key. The position of the module in the whole DRTLS will be printed out, see Figure 16. Type “?” followed by “Enter” key to get help information in UART Shell mode. More information of the UART Shell commands is introduce in the DWM1001 API Guide [2].

```
Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Apr 22 2017, 09:14:19.
Port /dev/serial0, 15:17:54

Press CTRL-A Z for help on special keys


 DWM1001 TWR Real Time Location System

 Copyright :  2016-2019 LEAPS and Decawave
 License   :  Please visit https://decawave.com/dwm1001_license
 Compiled  :  Feb 11 2019 04:21:32

 Help      :  ? or help

dwm> si
[361018.670 INF] sys: fw2 fw_ver=x01030001 cfg_ver=x00010700
[361018.670 INF] uwb0: panid=x2607 addr=xDECA50F56E101AAC
[361018.680 INF] mode: bn (act,-)
[361018.690 INF] uwbmac: connected
[361018.690 INF] enc: off
[361018.690 INF] ble: addr=F3:15:F3:45:09:E8
dwm>

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7 | VT102 | Offline | serial0
```

Figure 16: Connect to DWM1001 device through UART shell

4.3.2.2 UART Generic mode example

A simple example to make use of the UART Generic API is given in the DWM1001 Host Api package [4]. The example runs on the Raspberry Pi platform:

1) Download the dwm1001_host_api package onto the Raspberry Pi 3 device. Navigate to folder examples/ex1_TWR_2Hosts/tag/
2) Use nano editor to edit Makefile:
   - ➢ nano Makefile
3) Change the configuration parameter USE_INTERFACE to use UART interface:
   USE_INTERFACE = 0
4) Press “Ctrl + o” and “Enter” to save. Press “Ctrl + x” to exit nano editor.
5) Use “make” command to build the example:
   - ➢ make
6) Run the executable:
   - ➢ ./tag_cfg
7) Check log.txt file for the detail of UART data transmission.

```
pi@rpi-83:~/dwm1001_host_api/examples/ex1_TWR_2Hosts/tag $ ./tag_cfg
dwm_init(): dev0
Opening log file log.txt
        LMH_UARTRX_Init()...
        UART: Init start.
        UART: Init done.
Setting to tag: dev0.
dwm_cfg_tag_set(&cfg_tag): dev0.
Wait 2s for node to reset.
Comparing set vs. get: dev0.

Configuration succeeded.

Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
```

Figure 17: Run simple UART example on Raspberry Pi 3

The simple UART Generic mode example project is designed specifically for Raspberry Pi platform. The source file examples/ex1_TWR_2Hosts/tag/tag_cfg.c can be changed to add/modify functionalities. All available UART API functions are listed in header file include\dwm_api.h. More detailed information of the UART Generic mode is introduced in the DWM1001 API Guide [2].

The code/makefile needs to be changed, to suit other platforms than Raspberry PI.

4.4 SPI applications example

The DWM1001 module provides APIs over the SPI interface. The connection and a simple example are introduced here.

4.4.1 SPI connection

To connect to the DWM1001 module over SPI, the SPI pins on external connector J10 on the DWM1001 DEV board can be used. Table 2 shows the pins needed in the SPI connection.

Table 2 SPI pin connections

Pin to use		Pin to be connected to
Pin number	Function	
Pin 19	MOSI	MOSI
Pin 21	MISO	MISO
Pin 23	SCLK	SCLK
Pin 25	GND	GND
Pin 24	CSN	CSN

The connection with Raspberry Pi 3 (model B) is shown in Figure 15.

Note: the connector J10 on the DWM1001 DEV board is compatible with Raspberry Pi 3 connector J8 header pins 1-26. Pin 4 from J10 provides 5V power from Raspberry Pi to the DWM1001 DEV board.

4.4.2 SPI example

A simple example to make use of the SPI API is given in the DWM1001 Host API package [4]. The example can run on the Raspberry Pi platform:

1) Download the dwm1001_host_api package onto the Raspberry Pi 3 device. Navigate to folder examples/ex1_TWR_2Hosts/tag/
2) Use nano editor to edit Makefile:
   - nano Makefile
3) Change the configuration parameter USE_INTERFACE to use SPI interface:
   USE_INTERFACE = 1
4) Press "Ctrl + o" and "Enter" to save. Press "Ctrl + x" to exit nano editor.
5) Use "make" command to build the example:
   - make
6) Run the executable:
   - ./tag_cfg
7) Check log.txt file for the detail of SPI data transmission.

```
pi@rpi-83:~/dwm1001_host_api/examples/ex1_TWR_2Hosts/tag $ ./tag_cfg
dwm_init(): dev0
Opening log file log.txt
        LMH_SPIRX_Init(): SPI dev0...
        SPI0: spi mode: 0
        SPI0: bits per word: 8
        SPI0: max speed: 8000000 Hz (8000 KHz)
        SPI0: Reseting DWM1001 to SPI:IDLE
        LMH: LMH_SPIRX_Init for SPI dev0 done.
Setting to tag: dev0.
dwm_cfg_tag_set(&cfg_tag): dev0.
Wait 2s for node to reset.
Comparing set vs. get: dev0.

Configuration succeeded.

Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
Wait 1000 ms...
dwm_loc_get(&loc):
        [121,50,251,100]
```

Figure 18: Run simple SPI example on Raspberry Pi 3

The simple SPI example project is designed specifically for Raspberry Pi platform. The source file examples/ex1_TWR_2Hosts/tag/tag_cfg.c can be changed to add/modify functionalities. All available SPI API functions are listed in header file include\dwm_api.h. More detailed information of the SPI API is introduced in the DWM1001 API Guide [2].

The code/makefile needs to be changed to suit other platforms than Raspberry PI.

5 REFERENCES

This document refers to the documents listed below. Note that References' format can be as the author chooses.

1. Nordic nRF52 series Softdevice introduction, available from www.nordicsemi.com
2. DW1000 Software API Guide, available from www.decawave.com
3. DWM1001 System Overview, available from www.decawave.com
4. DWM1001 Host API source code package available from www.decawave.com
5. DWM1001 on-board source code package, available from www.decawave.com
6. DWM1001 Android application source package, available from www.decawave.com

6 DOCUMENT HISTORY

6.1 Revision History

Table 3: Document History

Revision	Date	Description
1.4	28-Sep-2020	Update for R2.1
1.3	28-Mar-2019	Update for R2
1.2	07-Aug-2018	Logo Change
1.1	11-Jan-2018	Editorial changes
1.0	18-Dec-2017	Initial version

7 Change Log

Revision 1.1

Page	Change Description
27	Remove unreleased documents from references
27	Repair url links for 2 references
10	Remove paragraph related to fig 2
10	Remove of renumber reference as references are removed or renumbered

Revision 1.2

Page	Change Description
ALL	Update with new logo

8 FURTHER INFORMATION

Decawave develops semiconductors solutions, software, modules, reference designs - that enable real-time, ultra-accurate, ultra-reliable local area micro-location services. Decawave's technology enables an entirely new class of easy to implement, highly secure, intelligent location functionality and services for IoT and smart consumer products and applications.

For further information on this or any other Decawave product, please refer to our website www.decawave.com.