VOICE IS ALL AROUND

Getting Started

With the ULE Starter Kit

DSP Group Inc.

Revision 0.7

Table of Contents

Chapter 1. Introduction

1.1. Purpose

The purpose of this document is to:

- provide you with the opportunity to familiarize yourself with the DECT-ULE system and capabilities
- introduce you to the DSP Group hardware and software solutions for IoT applications
- give you a brief exposure to the development tools DSP Group provides for creating application software at both Base and Device sides of the system

1.2. Definitions, Acronyms and Abbreviations

NAME	DESCRIPTION
Base	Base Station
C → N	Message sent from CMND to Node Host
CMND	Cordless module node
DECT	Digital Enhanced Cordless Telecommunications
DECT frame	DECT time base of 10ms
DU-EB	DECT-ULE - Expansion Board
GPIO	General Purpose Input Output pin
HS	Handset
HAN	Home Area Network
Hub	Another term for base
Hybrid	A hybrid base or device supports legacy (voice) DECT and DECT-ULE at the same time. Note: DSP Group bases are always hybrid bases.
HW	Hardware
MCU	Microcontroller Unit
N → C	Message sent from Node Host to CMND
Node	Another term for (ULE) device
Node Host	This is the microcontroller hardware provided by the User. It communicates with the CMND via a protocol over UART.
RSSI	Received Signal Strength Indicator
SW	Software
ULE	Ultra-Low Energy

1.3. References and Bibliography

#	DOCUMENT NAME	VERSION	DATE	LOCATION
[1]	DECT HAN CMND API Specification	M1.0	June 2018	/doc/HAN ULE Device - CMND API SPEC M.pdf

#	DOCUMENT NAME	VERSION	DATE	LOCATION
[2]	CMBS HAN SERVER PROTOCOL	1.15	26 June, 2018	/doc/CMBS HAN SERVER PROTOCOL.pdf
[3]	DECT-ULE Expansion Board (DU-EB) User's Manual	2.3	June 18, 2018	*TBD*
[4]	DECT-ULE Expansion Board Schematics	REV 2	August 09, 2017	*TBD*

Chapter 2. Known Issues

Following are the currently known issues:

DU-EB Expansion Board

- Board will not power up if run from a USB wall plug, please connect to a Laptop/PC/Hub
- MIC jack is a line-in (no microphone power provided)
  - MIC jack only records the right channel
- SPK jack is using differential output
  - Connect passive speakers (minimum 4ohm) between left and right channel, no ground needed
  - Active stereo speakers will produce hum at mid-to-high gain levels
  - Stereo headsets produce hum

DU-EB Expansion Board Firmware

- Firmware will drop CMND messages just after sending `CMND_MSG_PARAM_SET_*` messages
  - Wait 1000ms after receiving the corresponding `CMND_MSG_PARAM_SET_*RES` response before sending the next `CMND` message

Chapter 3. DECT-ULE System Overview

This chapter provides a quick overview over the DECT-ULE system. The DECT-ULE standard was developed by ETSI for Europe, but has been adopted by many countries worldwide.

The DECT system is using the star topology and typically consists of one base and one or more handsets and devices. Handsets are regular voice handsets, whereas devices are ULE capable and can either be battery or line powered.

This guide focuses on ULE, and therefore on base and devices only.

Base and devices align their timing, making DECT a synchronous system. For that purpose, the base transmits a beacon and devices receive it. The exchanged beacon carries system relevant information such as identity and base capabilities.

In DECT, only devices are establishing data links between base and device. The base has to contact the device and ask it to establish a link if it wants to send information. This process of contacting the device is called paging. Paging information is part of the information carried by the beacon.

ULE device types

- Battery powered: these devices are in deep sleep states most of the time. The beacon is received only infrequently (e.g. every minute) which also determines the response time for base to device communication.
- Line powered: theses device can permanently receive the beacon. This can be used to achieve fast data transfer times in either direction.

3.1. DECT-ULE Characteristics

- Range: outdoor ~600m; indoor ~60m
- Battery lifetime of up to 10yrs can be achieved (dependent on use case)
- The number of events (data transfers) a device can support in a given timeframe depends on device type, the amount of data to be transferred, and also on the RF environment. Typical values are <100 bytes/min for battery operated devices and 1kbytes/sec for line powered devices (numbers depend on use case).
- Latency for line powered devices can be as low as 20ms, battery operated devices typically achieve 30-50ms latency for device to base data transfer
- Hybrid devices can also support voice calls or higher data rates of up to 500kBit/s

3.2. Identities

DECT-ULE bases and devices have a unique 40 bit identity called RFPI for the base and IPEI for the device (similar to an Ethernet MAC address). Devices or bases in one area must have unique identities.

3.3. Registration

In order for base and device to form a system, both need to be paired. A DECT-ULE base can pair with many ULE devices. In DECT, the process of pairing is called registration. Without registration, the device can’t do anything. As part of the over-the-air registration the following is done:

- A secret authentication key gets known to device and base
- ULE capabilities are negotiated between base and device
- A logical device number is assigned to each device that allows for easy addressing devices within a given DECT system
    - If the same device is registered to another DECT base, it will likely get another device number
    - If the same device is re-registered to the same base it will normally get the same device number

Registration in practice

- In order to allow devices to register to a base, the base must be prepared to accept new devices ('open the base')
- For security reasons, the base will only allow new devices for a certain time (e.g. 2 minutes). If the device did not manage to register in time, the base needs to be prepared again
- Registration is typically a one time process. However if device or base capabilities change e.g. due to SW update, it may be required to re-register the device

    ```
    *Deregistration*
    ```

Deregistration is the process where the user deliberately removes a registration from a base e.g. to use this device with another base.

- In ULE deregistration is triggered by the base
- Due to this, deregistration can only proceed when the device is in contact with the base

* When deregistration is started at the base side while the device is in sleep, the base puts this device to state `blacklisted`
* Next time the device makes contact with the base, deregistration starts automatically

3.4. Security

DECT-ULE uses authentication and encryption. ULE data transfer is encrypted using AES-128. The encryption key is created during the authentication procedure.

3.5. ULE Paging

ULE Paging is the process where the base can request the device to do certain things. For example:

* Raise a data bearer (so base can push data to the device)
* Announce broadcast from base to device (not supported in this version of ULE Starter Kit)

ULE Paging can only be sent to specific ULE devices in certain DECT frames. This allows battery operated devices to sleep most of the time, because they have to listen to the beacon only in those frames where potentially they can be paged. Line powered devices may be paged in every DECT frame (10ms) to reduce latency. The negotiation about when paging can happen is done between base and device during registration. Devices which did not undergo this negotiation cannot be paged.

3.6. ULE Data Transfer

Only the ULE device can directly initiate a data transfer. Hence only an ULE device can send data to the base at any time (e.g. on certain sensor conditions). However if the base needs to push data to the device, this needs the device to contact the base and hence always comes with a certain delay:

* The base can page the device when the paging interval is reached ("resume paging")
* The base can wait for the device to access the base e.g. due to a keep alive event (see 3.7)
* The base can wait for any other asynchronous data transfer started by the device, to push its own data to the device

3.7. Keep Alive

For many applications targeted by ULE (security, automation, healthcare) it is important to know whether a given sensor is still operational. For example if some burglar alarm has been damaged, or run out of battery, then this alone gives rise for an alarm. To address this requirement, most ULE sensor will periodically report their health status to the base using an extremely short data transfer. This method is called 'keep alive'.

3.8. Hibernation

Hibernation is a term for an extreme deep sleep state of the DHAN-J. The time to sleep is programmable, for example to allow periodic wakeup every 10 minutes, or to wake for a specific DECT frame where a paging message may arrive (not supported in current ULE Starter Kit). Another option to wake from this hibernation state is a GPIO event (e.g. button pressed). During hibernation phase critical calibration and programming parameters are stored in non-volatile memory.

3.9. HAN-FUN

The ULE Alliance is an industry consortium of companies with the goal to push DECT-ULE to markets. The application layer protocol HAN-FUN (Home Area Network FUNctional protocol) is using the DECT-ULE standard, and has been released by the ULE Alliance in November 2013 to ensure interoperability of ULE devices.

Although the ULE Starter Kit package described in this document uses HAN-FUN protocol to some extent, this is largely hidden from the user and hence not further detailed here. If one is interested in HAN-FUN details, please refer to www.ulealliance.org.

3.10. Radio Frequency Band

DECT-ULE can be deployed in many countries worldwide. The country specific regulations often require some different radio settings compared to Europe which will change the frequencies to be used, the transmit power, or the applied channel selection rules. In order not to break the law when the ULE Starter Kit is used, make sure to use the appropriate setting for the country it is used in!

Chapter 4. DECT-ULE Starter Kit Overview

The DECT-ULE Starter Kit consists of one DECT-ULE Base USB Dongle and one or more DECT-ULE Expansion Boards. Both base and device use the UART for communicating with their host. The protocol spoken on the base side is called CMBS (Cordless Module Base), the procotol on the device side is called CMND (Cordless Module Node).

In a typical development environment, the Base USB Dongle is controlled by a PC. The Expansion board can also be controlled by a PC for a quick test drive, but will typically be connected to another microcontroller board for development purposes.

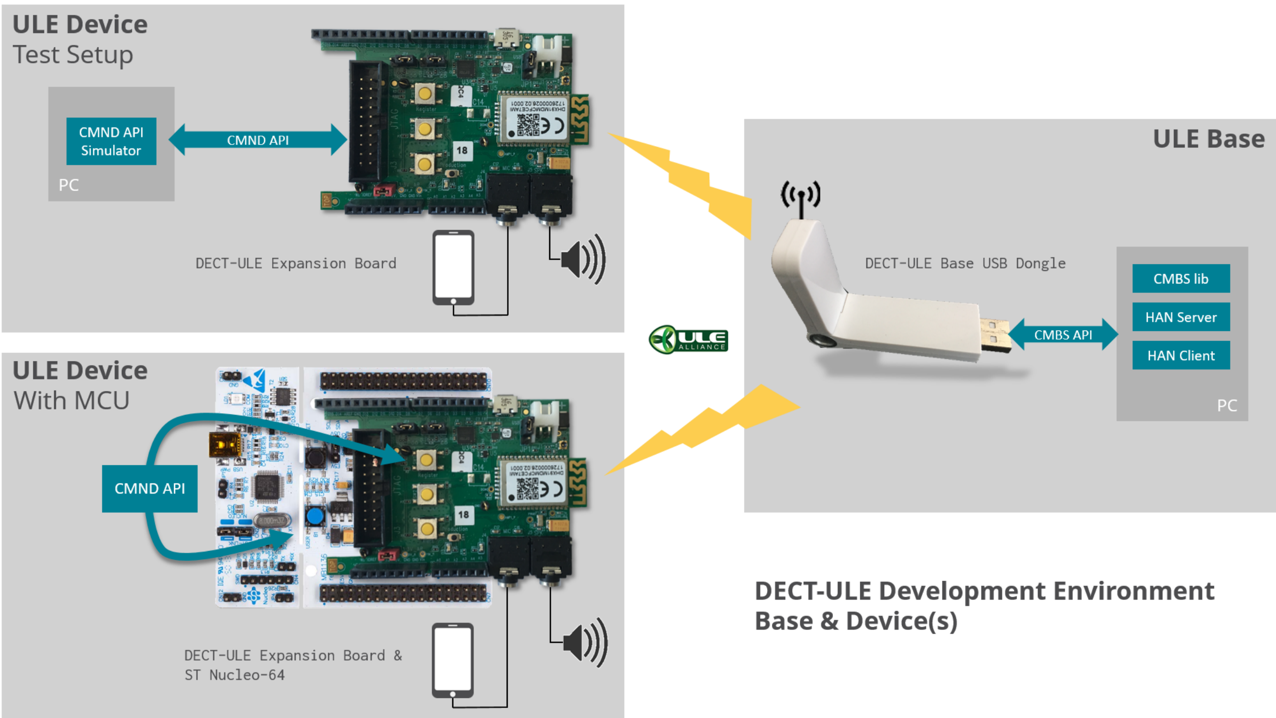


Figure 1. Hardware and software elements of the ULE Starter Kit

Element(s)	Part number to order
DECT-ULE Expansion Board	HOMEA-DHX913-EXTDHNJ-NN-IL.BRD
DECT-ULE Base USB Dongle	XCEDR-DCX813-ULEDNGL-BN-HK.BRD
DECT-ULE System Evaluation Tool (1 Dongle + 1 Expansion Board)	HOMEA-DEVTOOL-BN-IL.SET

4.1. DECT-ULE Expansion Board (DU-EB)

The DECT-ULE Expansion board can be stacked on top of Arduino R3 compatible board. It features the DSP Group DHAN-J Module which comes loaded with a dual-mode (data and audio) ULE device firmware which incorporates physical, MAC and transport layers, enabling it to function as a DECT-ULE device. It can be configured via jumpers to support the following use-cases:

- Powered via micro USB (connect to Laptop/PC/Hub)
- Powered via 3V socket for battery (2 x AA)
- Powered from board it is stacked on top of

Similar to the power schemes, the UART connectivity can be configured via jumpers, too:

- UART routed to micro USB
- UART routed to board it is stacked on top of

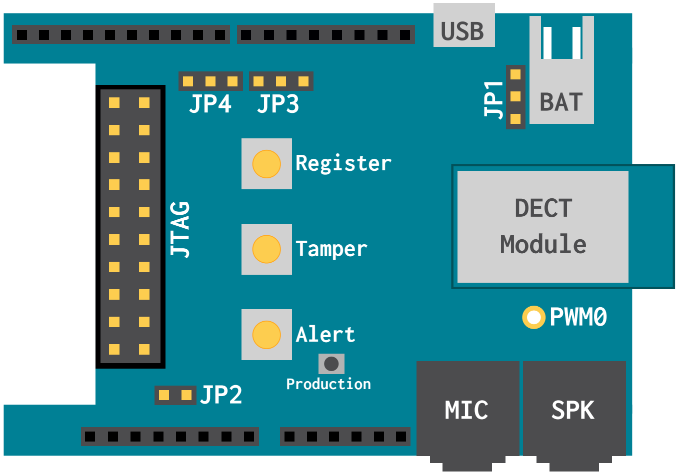


Figure 2. DECT-ULE Expansion Board Overview

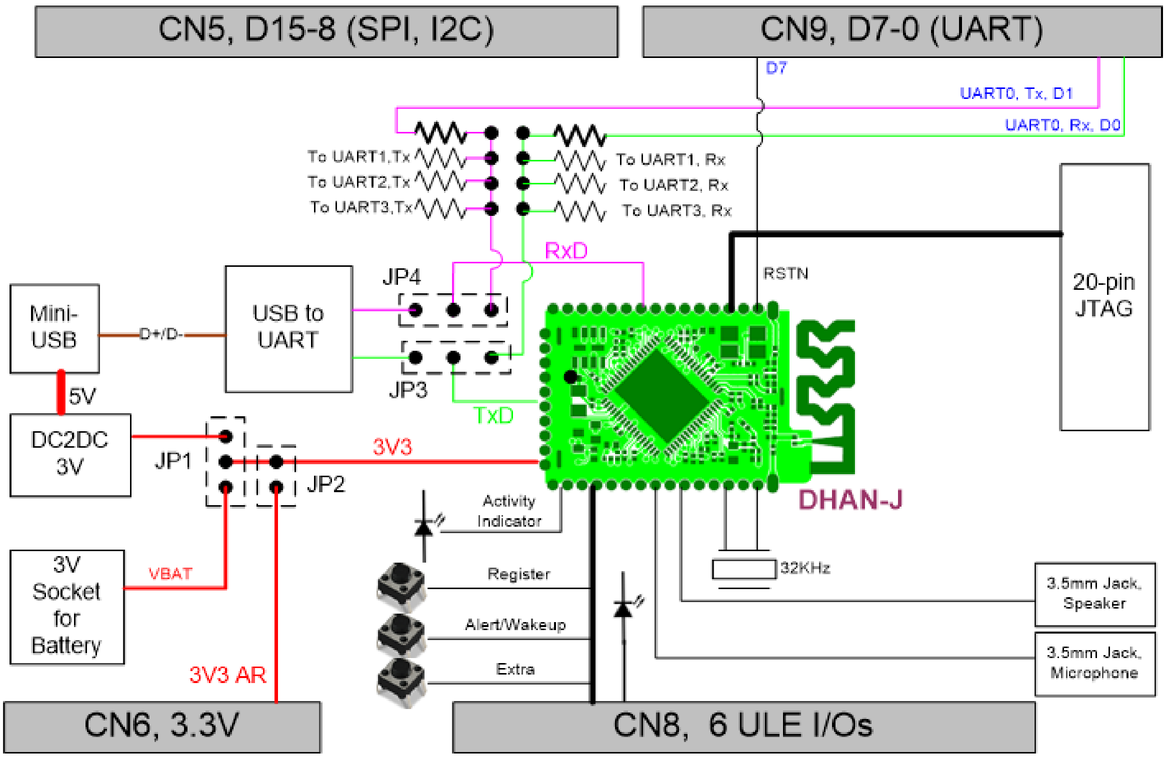


Figure 3. DECT-ULE Expansion Board Schematics

Chapter 5. Hands On

After unpacking your ULE Starter Kit (with USB dongle and DECT-ULE Expansion board), you can quickly get a first impression of its features with minimal effort. Please make sure your board is configured with the right jumper setting:

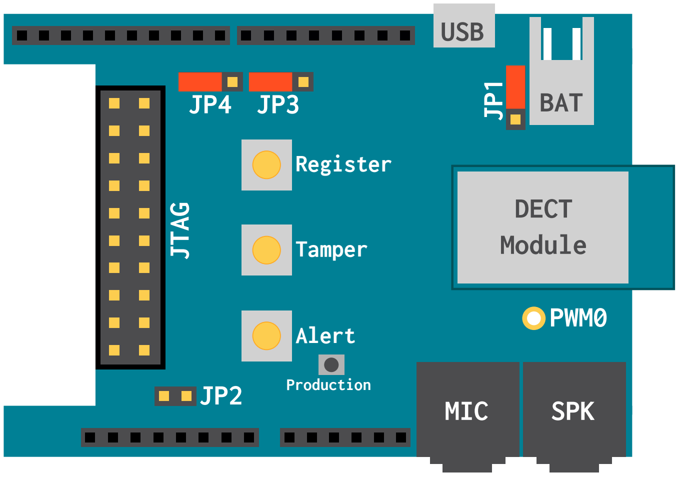
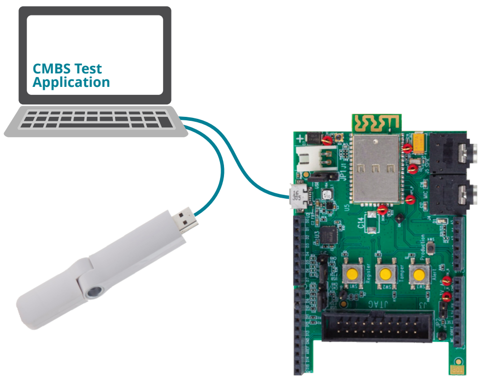


Figure 4. DU-EB: Powered from USB, UART to USB

5.1. Set up the environment

- Connect a Windows PC to the USB dongle and start the DSPG Test Application (see appendix A.1)
- Power up the DU-EB via USB (default jumper setting) or batteries (see appendix E.1)



5.2. Register

The registration procedure must be performed as first step to pair the ULE device (DU-EB) and the base (USB dongle). The device and the base save the registration details in non-volatile memory.

- Open the base for registration with the DSPG Test Application (see appendix A.2)

- Within 2 minutes press the Register button (SW1) on DU-EB and watch the registration log information in the DSPG Test Application (see appendix A.3)
- The base will automatically close for registration when a DU-EB registers or after the 2 minutes has elapsed.

5.3. Send an alert

- Press Alert button (SW3) on DU-EB and watch the Alert counter being incremented in the HAN window of the DSPG Test Application (see appendix A.4)

5.4. Make a voice call

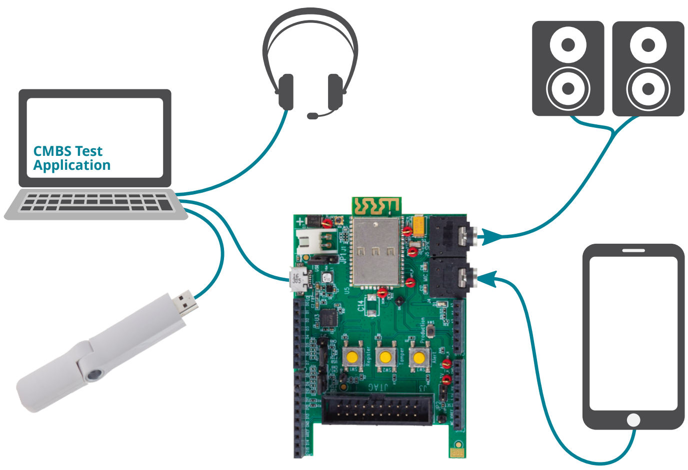
A major advantage of ULE is that it uses DECT to provide a clear and protected voice link.

You only have to connect audio equipment to the test setup.

- Connect a smartphone or audio player to the 3.5 mm line-in (MIC) jack of the DU-EB, start playing your music

> The MIC jack requires a LINE-IN input signal, adjust your player volume to control audio saturation and clipping.

- Connect speakers or earphones to the 3.5mm line-out (SPK) jack of the DU-EB
- Connect a headset (or earphones and microphone) to the PC
- Configure audio routing on the PC
  - Windows: see Appendix C
  - Linux: see Appendix D



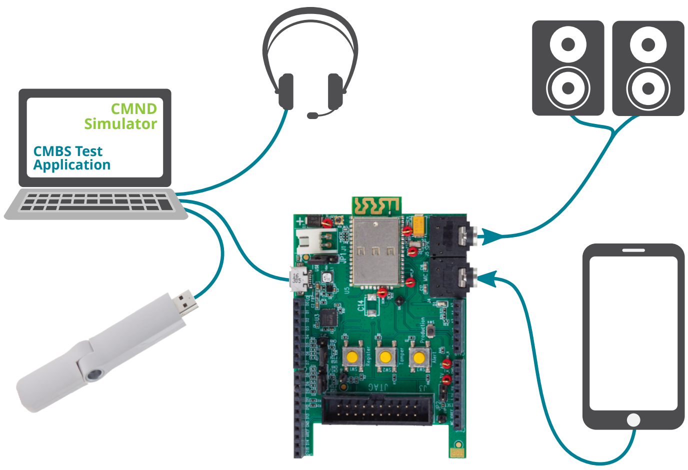
- Press the Tamper button (SW2) on DU-EB to start an outgoing voice call
- You will hear your music playing on the PC earphones and hear your voice on the earphones connected to the DU-EB line-out
- Press the Tamper button (SW2) on DU-EB to end the voice call

5.5. Use the CMND API

All the operations accessed by pressing buttons on the DU-EB can also be performed using the CMND Simulator interface to the CMND API. And there are additional operations which are not available by button press.

The DU-EB must be connected to the Windows PC using a USB connection. If you have been running the DU-EB on batteries see appendix E.1 before connecting the USB cable.

Start the DSPG Test Application and the CMND Simulator (see appendix A.1 and B.1)



Registration

- Open the base for registration with the DSPG Test Application (see appendix A.2)
- Within 2 minutes send 'Register' message via the CMND Simulator (see appendix B.2) and watch the registration log information in the DSPG Test Application (see appendix A.3)

Raise an alarm

- Send 'Alert on' message via the CMND Simulator (see APPENDIX B.2) and watch the alert counter being incremented in the HAN window of the DSPG Test Application (see appendix A.4)

Start an outgoing voice call

- Configure the audio sources as for voice calling using the buttons

See appendix A.5 Voice call handling

- Select the Voice Call window of the CMND Simulator, untick 'Auto Handling', and press 'Start Call'
- The CMND Simulator indicates the new Call Status 'Voice Call Connected'
- You hear your music playing on the PC earphones and hear your voice on the earphones connected to the DU-EB line-out
- In the Voice Call window of the CMND Simulator press 'End Call'
- The CMND Simulator indicates the new Call Status 'Voice Call Released'

Start an incoming voice call

See appendix A.5 and B.3 Voice call handling

- Select the Voice Call window of the CMND Simulator, untick 'Auto Handling'
- Select the Calls window of the DSPG Test Application
- In the Calls window of the DSPG Test Application, set device 'In Link' by:
  - Pressing 'TxRequest' and waiting until the 'Link Status' changes to 'In Link'
  - Sending a 'Keep Alive' message via CMND Simulator may shorten the time to get 'In Link'
- In the Calls window of the DSPG Test Application press 'Request Call'
- In the Voice Call window of the CMND Simulator press 'Call Request Response'
- The DSPG Test Application indicates the new DCM state 'Connected'
- You hear your music playing on the PC earphones and hear your voice on the earphones connected to the DU-EB line-out
- In the Calls window of the DSPG Test Application select the call instance and press 'Release'
- The DSPG Test Application indicates the new DCM state 'Idle'

Keep Alive

- Send 'Keep Alive' message via CMND Simulator (see appendix B.2) and watch the Keep Alive counter being incremented in HAN window of the DSPG Test Application (see appendix A.4)

Miscellaneous messages from CMND Simulator

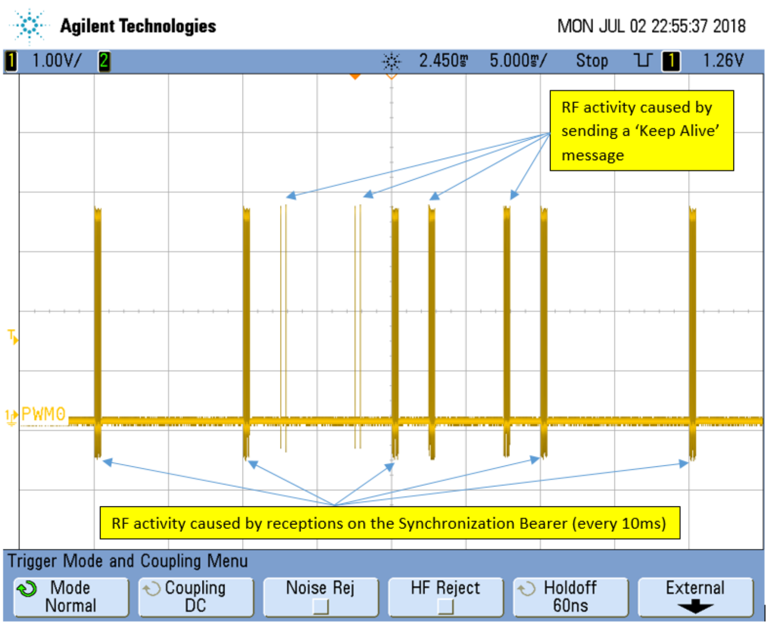
See appendix B.2

- Send 'Get Version'
- Send 'Reset Req'
- Send 'RSSI Request'
- ...

5.6. RF sensing with a scope

If you are interested in seeing the RF signals.

- Apply a scope to `PWM0` PIN of the DU-EB
- Use CMND Simulator to set the DECT EEPROM byte at start address `35A` (hex) to `40` (hex) (see appendix B.4)
- Send 'Reset Req' message via CMND Simulator to activate the new EEPROM setting (see appendix B.2)
- Observe radio activity on scope


Agilent Technologies
MON JUL 02 22:55:37 2018
1.00V/
2.450s
5.000s/
Stop
1.26V
RF activity caused by sending a 'Keep Alive' message
PWM0
RF activity caused by receptions on the Synchronization Bearer (every 10ms)
Trigger Mode and Coupling Menu
Mode Normal
Coupling DC
Noise Rej
HF Reject
Holdoff 60ns
External

Chapter 6. Development Setup

In order to go through the next sections of this document to try the provided software examples for device and base, the following tools are needed:

- System Workbench for STM32 (based on Eclipse)
- Python (3.x is recommended but the example code will work with 2.7.x)
- STLink driver for STM32 chips

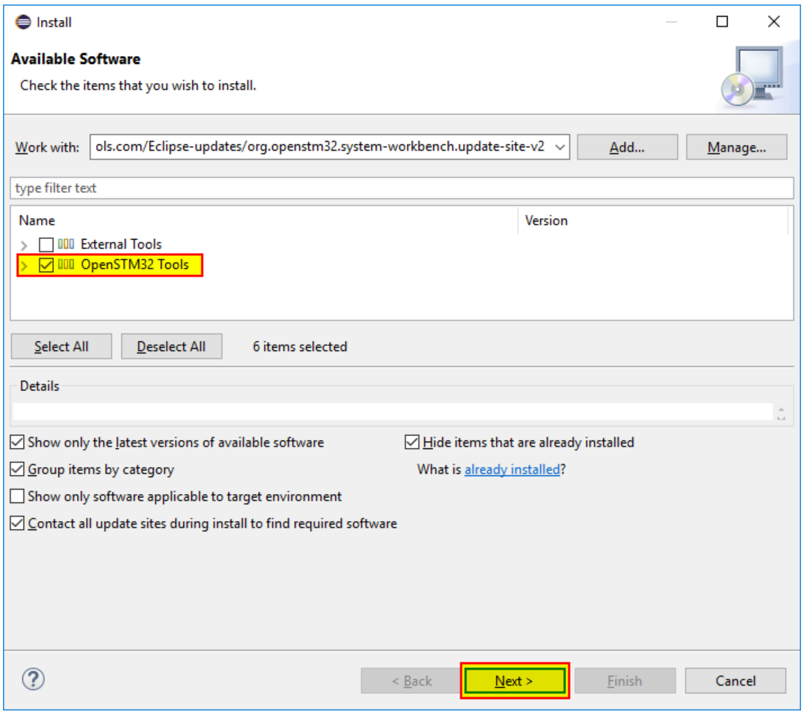
6.1. System Workbench Installation

System Workbench is based on Eclipse which requires Java to run. This document assumes that Java JRE 10 is installed on your system, if not it can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/jre10-downloads-4417026.html. For help on some common Java installation issues, please see Java Troubleshooting Appendix G.

Install Eclipse by downloading the installer for Eclipse Photon 64 bit from https://www.eclipse.org/downloads. When running the Eclipse installer, make sure to select "Eclipse IDE for C/C++ Developers" for installation.

Following the successful installation of Eclipse, the System Workbench Addon can be added:

- Start Eclipse
- Select "Help" > "Install New Software ..."
- Create new update site:
- Click "Add ..."
- Name: "System Workbench for STM32 - Bare Machine edition"
- Location: http://ac6-tools.com/Eclipse-updates/org.openstm32.system-workbench.update-site-v2
- Click "OK" to create site
- Select "OpenSTM32 Tools"



- Click "Next >"
- Click "Next >" again to confirm "Install Details"
- Accept license and start installation
- Select "Install Anyway" when warned about unsigned packages
- Restart Eclipse when asked to so do
- Wait for ARM GCC toolchain installation to finish (observe bottom right corner)

6.2. Python Installation

Windows

Download the Python Installer from https://www.python.org and run it. After logging out and in again, launch a command prompt (`cmd.exe`) and try running `py`. For Python 2.7.x, try `python` instead.

For Python 3.x you should see the following:

```
C:\>py
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Linux

The `python3` and `python3-pip` packages are needed, please install them via your package manager. After installation, open a terminal and run `python3`. You should see the following:

```
$ python3
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

6.3. STLink Driver Installation

Windows

Navigate to the `device\Utilities\STLink` directory of the ULE Starter Kit, unzip the zip file, and run the `stlink_winusb_install.bat` batch file using administrator rights (Right Click > "Run as administrator"). Follow the instructions of the driver installer.

Linux

There is an open source implementation of the STLink-v2 driver available at https://github.com/texane/stlink. Follow the installation instructions at https://github.com/texane/stlink#installation.

For Ubuntu, these are the steps needed:

```
$ sudo apt install build-essential cmake libusb-1.0-0-dev
$ git clone https://github.com/texane/stlink
$ cd stlink
$ make release
$ cd build/Release
$ sudo make install
```

Now setup permissions:

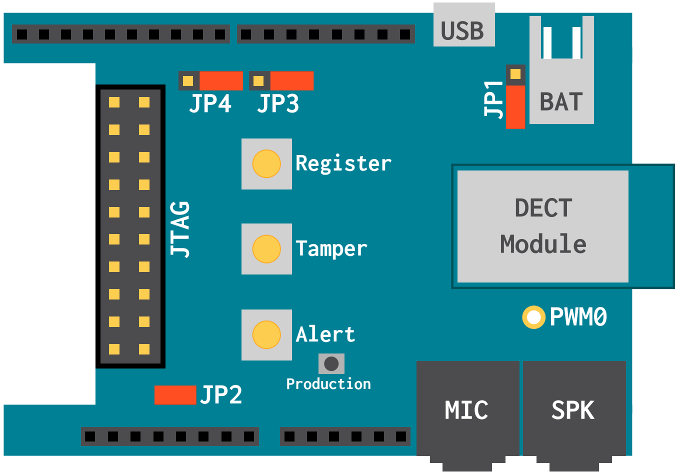
```
$ sudo addgroup --system stlink
$ sudo adduser <your-user> stlink
$ sudo adduser <your-user> dialout
```

Finally, reboot your system to make sure the updated permissions take effect.

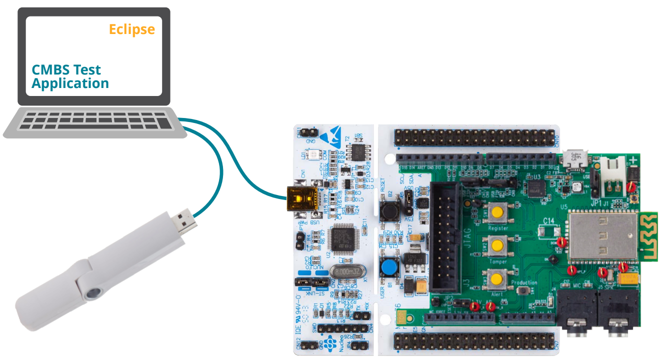
Chapter 7. Device Development

This package ships with several example projects to illustrate development of device applications. The DECT-ULE expansion board is stacked on top of a ST Nucleo-64 board, the `STM32L476RG`. The Nucleo board is programmed with the supplied example applications and controls the expansion board via GPIO and UART lines. The protocol spoken on the UART line is still `CMND` (as in the previous chapters), and the example applications use the `CmndLib` library to facilitate building and parsing `CMND` messages.

When stacking the DECT-ULE expansion board on top the Nucleo board, it's jumper configuration needs to be adapted so the expansion board is now powered from the Nucleo and the UART is routed to the connectors.



The jumpered expansion board and Nucleo board can then be connected to your development PC via the Nucleo's USB Mini-B port so that the complete development setup now looks like this:



7.1. Import, Build and Run Examples

All examples are located in the `device/Examples` directory of the ULE Starter Kit package. All example projects are structured and work in the same way. For your reference, you will now import, build and run the "Registration" example using Eclipse.

- In Eclipse, select "File > Open Projects from File System ..."

In the next step, make sure you select the `SW4STM32` subdirectory inside of the example you want to load.

- Navigate to the "Registration" example, make sure you select the `SW4STM32` subdirectory in the example folder. This subdirectory contains all the Eclipse project files and settings.
- Click "Finish"

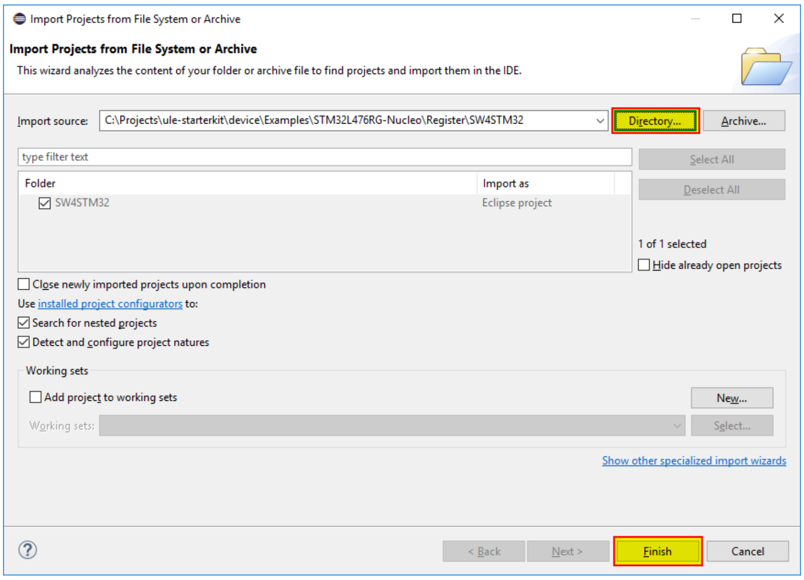


Figure 5. Importing a project

You should now see the project structure on the left hand side of Eclipse. Select the "Registration" project, and click the "Hammer" icon in order to build it.

The "Hammer" icon on the left side will build *all projects* in your workspace, with their currently active configuration. In order to build only the selected project, or choose another build configuration (Debug/Release), use the right "Hammer" icon.

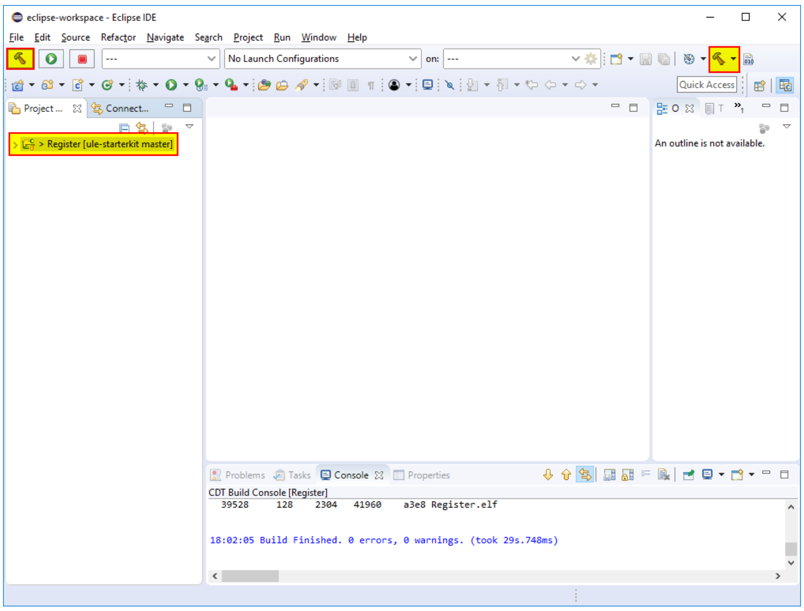


Figure 6. Building a project

Once the project is built, you can now run or debug it on the Nucleo. In order to be able to do this, a launch configuration has to be created first. This will tell eclipse which binary to load to the Nucleo.

- Select the "Register" project
- Select "New Launch configuration ..." from the launch configuration dropdown menu
- Select Debug or Run (Debug will halt in `main()`, allowing you to single step)
- Select "Ac6 STM32 C/C++ Application", Next ...
- Ensure the `Debug/Register.elf` ELF file was selected
- Click OK

> Eclipse versions prior to the Photon release have a bug where the debugger/programmer will not launch using the default launch configuration. In these versions, uncheck the "Reset and Delay" and "Halt" checkboxes in the "Startup" tab of the launch configuration properties.

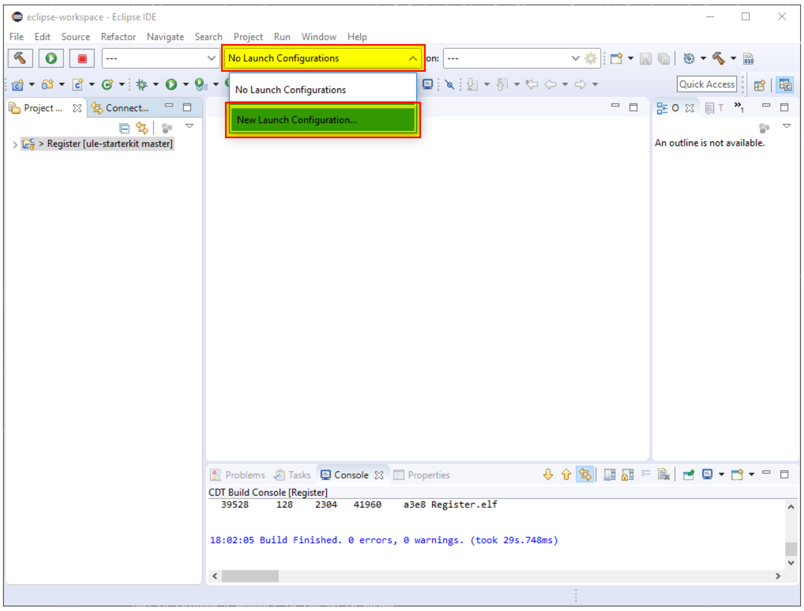


Figure 7. Creating a Launch Configuration

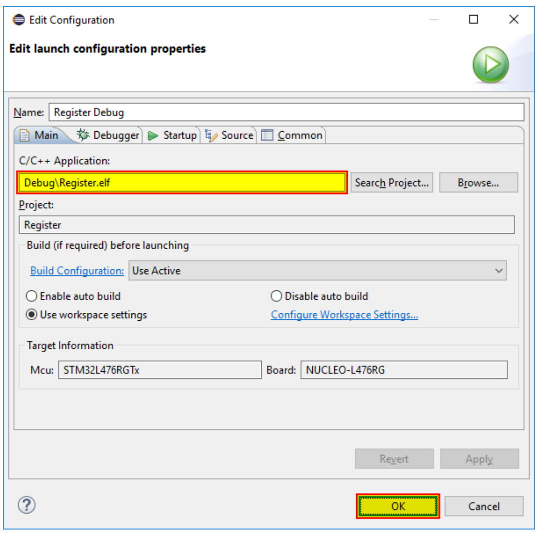


Figure 8. Reviewing the Launch Configuration

Finally, click the "Play" icon next to the left "Hammer" icon. Eclipse will program the example application to the Nucleo and will run it.

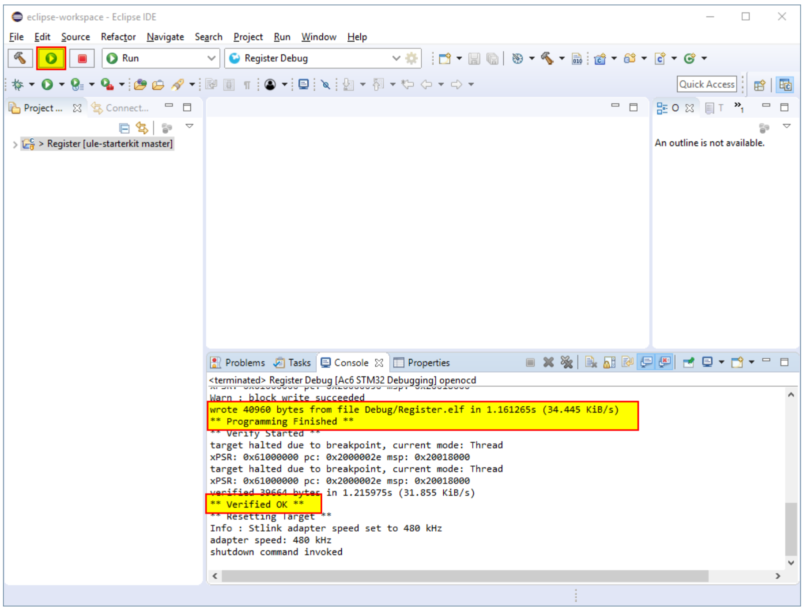


Figure 9. Running examples

7.2. Project Structure

All included examples are structured in the same way:

Table 1. Project structure

Project Path	Package Path	Description
`Drivers/`	`device/Drivers`	Nucleo/STM32 hardware drivers
`Example/{Inc,Src}`	`device/Examples/STM32L476RG-Nucleo/<Example>`	Example specific code
`Example/Support`	`device/Examples/STM32L476RG-Nucleo/Support`	Glue code common for all examples
`uleasy/CmndLib`	`device/uleasy/CmndLib`	Library for parsing and building `CMND` messages

7.3. Example Behavior

All included examples follow the same setup and procedure. Here is an overview.

7.3.1. Log Output

The example applications all output log information on the Nucleo USB virtual COM port. In order to view it, connect a terminal emulator to the `STMicroelectronics STLink Virtual COM Port`, baudrate 115200.

RawData example output

```
[*] 00018022 Send raw FUN request
MCU->CMND: FUN<0108> SEND_REQ<01> [IE_FUN [DstDeviceId: 0x0000, SrcDeviceId: 0x0001,
AddressType: 0, DstUnitId: 0x0002, MessageSequence: 0, MessageType: 1, InterfaceType: 1,
InterfaceId: 0x7f16, InterfaceMember: 0x01, RawData[13]: 48 65 6c 6c 6f 2c 20 57 6f 72 6c 64
21]]
CMND->MCU: GENERAL<0000> LINK_CFM<07> [IE_RESPONSE [OK<00>]]
[*] 00018102 Got LinkCfm response, result = 0x0
```

7.3.2. Startup

When the example applications are starting, they will first initialize all the needed hardware peripherals of the Nucleo and then jump to `ExampleMain()`. Here the expansion board reset line is released (connected to `GPIOA8` of the Nucleo) and the example code enters into an endless loop polling for events.

The first event received will be the `hello` indication from the expansion board telling the example application whether it is registered to a base and if so, which device id it was assigned.

The end of the startup phase is indicated with a long blink of the green LED on the ST-Nucleo.

7.3.3. How to trigger a request to the DU-EB board

An infinite loop waits for the press of the blue button. If the device is registered, the specific action for the example is triggered.

7.3.4. Indications

A green LED on the ST-Nucleo board is used to indicate the result of the request:

GREEN LED	RESULT
1 long blink	The request was successful.
1 short blink	The request was not sent because the device is not registered.
2 short blinks	The request was not sent because of UART problems.
3 short blinks	The request was not successful because it was not accepted by the DU-EB or the base (e.g. when the base is powered down)

7.4. Included examples

7.4.1. Register

This example shows how to command the DU-EB to send a registration request to the hub.

Pressing the blue button on ST-Nucleo triggers the registration request.

The hub must be open for registration for the registration request to succeed (see appendix A.2).

7.4.2. Alert

This example shows how to command the DU-EB to send an alert to the hub.

The device must have been registered to the hub for this example to succeed.

Pressing the blue button on ST-Nucleo triggers the alert.

7.4.3. Raw Data

This example shows how command the EU-DB to send a raw data FUN message to the hub.

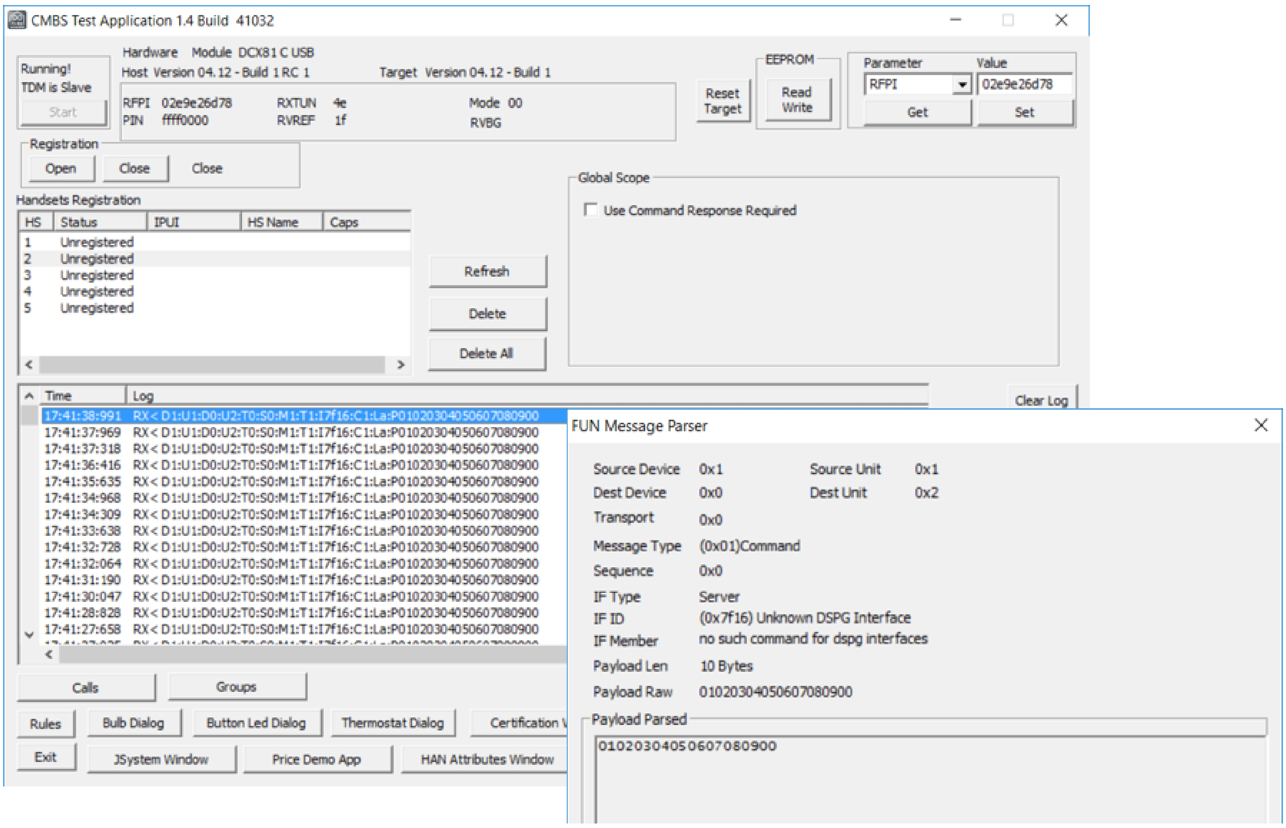
When starting the example code a *set CMND parameter* message is sent to the DU-EB to reduce the default minimum sleep time (paging interval) from 5 seconds to 2 seconds.

The device must have been registered to the hub for this example to succeed.

Pressing the blue button on ST-Nucleo triggers sending sending the data.

> Pressing the blue button repeatedly will only work slowly. This is because of the basic implementation of the example which will block for 500ms when toggling the Nucleo LED for the success indication.

The reception of the raw data FUN message is indicated in the CMBS Test Application. Right clicking the message shows the details in the 'FUN Message Parser' window.



7.4.4. Voice Call

This example shows how to use the EU-DB to control a voice call.

When starting the example code a *set CMND parameter* message is sent to the DU-EB to increase the default minimum sleep time (paging interval) from 5 seconds to 10 seconds.

The device must have been registered to the hub for this example to succeed.

Pressing the blue button on ST-Nucleo triggers the following action depending on the current voice call state:

Voice call state	Action
Voice call is not active	A start voice call request message is sent to the DU-EB
Voice call is active	An end voice call request message is sent to the DU-EB.
Voice call setup request from base is detected	An answer incoming voice call request message is sent to the DU-EB

The green LED will blink rapidly to indicate that a call setup from the base has been detected.

7.5. Creating your own project

When creating your own project, simply start as usual by creating a project for your hardware platform. Once that step is done, simply copy/integrate the CmndLib into your project structure. You can find at `device/uleasy/CmndLib` in the ULE Starter Kit.

When building your project now, you will notice that it won't build anymore because of some missing glue code. You will need to implement some platform specific functions:

CmndLib/CmndLib_UserImpl.h

```
u64 p_CmndLib_UserImpl_GetTickCountMs( void );
```

CmndLib/CmndLib_UserImpl_StringUtil.h

```
int p_CmndLib_UserImpl_strnlen( const char* str, size_t maxlen );
void p_CmndLib_UserImpl_strncat( char* dst, size_t maxlen, const char* src, size_t count );
int p_CmndLib_UserImpl_snprintf( char* dst, size_t maxlen, const char* format, ... );
```

For the STM32L476RG-Nucleo these functions have already been implemented, so if you are using the same platform you can just integrate the following files from the ULE Starter Kit:

- `device/Examples/STM32L476RG-Nucleo/Support/CmndLib_UserImpl.c`
- `device/Examples/STM32L476RG-Nucleo/Support/CmndLib_UserImpl_StringUtil.c`

Chapter 8. Base Development

This package ships with the tooling needed to develop an application in the base. The typical development stack on the base looks like the following:

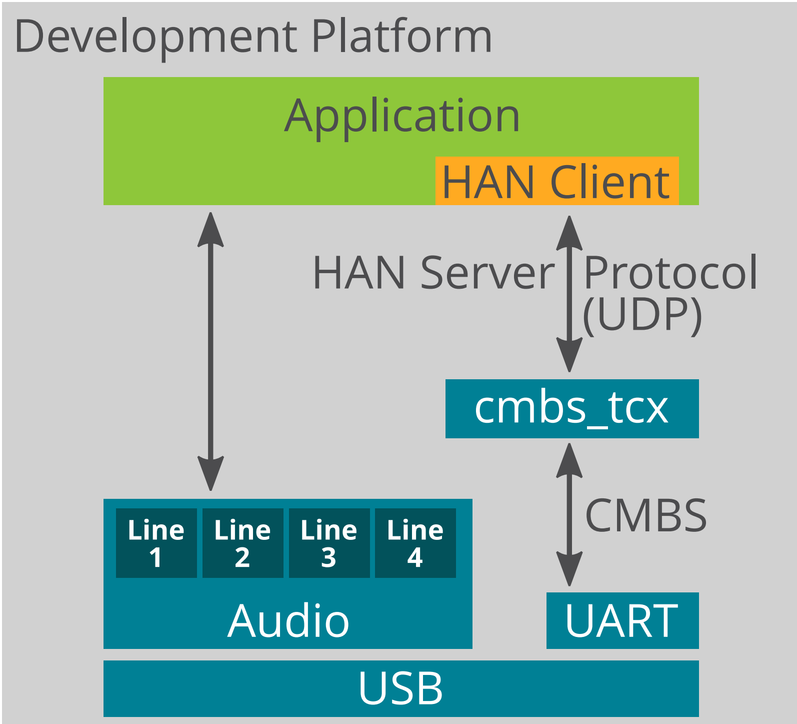


Figure 10. Base Development Stack

The blue boxes are supported and delivered by either the operating system vendor or DSP Group. The HAN Server Protocol is specified in [2]. The HAN Client exists as a reference python implementation, to be picked up and integrated into your application.

The ULE Starter Kit also ships with a demo application which illustrates the use of the HAN Client reference implementation.

Component	Package Path	Description
`cmbs_tcx`	`base/tools/cmbs_tcx`	Binary which proxies HAN Server Protocol requests to the USB dongle and back. It implements the server part of the HAN Server Protocol [2].
`han_client`	`base/ule-hub/han_client.py`	Python module which (partially) implements the HAN Server Protocol client side and wraps it in an easy to use API.
Demo	`base/ule-hub/han_app.py`	Python application which demonstrates the use of the `han_client` module.

8.1. Running the demo ule-hub

The software package includes demo software for the ULE hub. This software is written in Python and demonstrates how to use the HAN client API to replace the DSPG Test Application.

8.1.1. Install the Python dependencies

Before running the demo software for the first time, there are python dependencies which need to be installed. This only needs to be done once.

Windows

Open a command prompt, then:

```
...\base\ule-hub>py -m pip install -r requirements.txt
```

Linux

Open a terminal, then:

```
$ cd /path/to/base/ule-hub
$ pip3 install -r requirements.txt
```

8.1.2. Start the HAN server

The demo depends on the HAN server being available, start it from the .../base/tools directory:

Windows

Open a command prompt, then:

```
...\base\tools>cmbx_tcx -han
DSP Group Demo Software Version:0412 Build:772
|-----------------------------------|
|  Available COM ports in system:  |
|-----------------------------------|
| COM4: High-Speed USB Serial Port
| COM5: High-Speed USB Serial Port
| COM6: High-Speed USB Serial Port
| COM7: High-Speed USB Serial Port
| COM12: USB Serial Device
Auto detected USB Dongle on COM12
...
```

Linux

Open a terminal, then:

```
$ cd /path/to/base/tools
$ ./cmbs_tcx -usb -com 0 -han (will look for /dev/ttyACM0)
DSP Group Demo Software Version:0412 Build:517
...
```

8.1.3. Run the demo

Start the demo software from the .../base/ule-hub directory:

Windows

Open a command prompt, then:

```
...\base\ule-hub>py han_app.py
```

Linux

Open a terminal, then:

```
$ cd /path/to/base/ule-hub
$ ./han_app.py
```

Type `help` to get a list of available commands. Type `help <command>` for more information.

8.1.4. Register a device

First, open the registration window in the base, in the ule-hub demo:

```
> open_reg
16:24:24.822 Registration window open
```

Then, subscribe a device, for example by pressing the "Register" button (it may take half a minute).

```
16:25:03.157 Device 1: registered (or registration updated)
16:25:03.157 Registration window closed (reason: DEV_REGISTERED)
```

The device is now paired with the base and the registration window in the base has automatically been closed.

8.1.5. Sending Raw Data

Run the Nucleo board with the "RawData" example. When pressing the blue button on the Nucleo, the following message will be shown by the ule-hub application:

```
16:26:34.214 Device 1: message from raw data unit: 'Hello, World!'
```

Also try sending a message to the device:

```
> send 1 "hello from the base"
16:27:03.836 Device 1: message has been queued for delivery ...
16:27:04.294 Device 1: message delivered.
```

In the terminal emulator connected to the Nucleo virtual serial port, you will see the following:

```
CMND->MCU: FUN<0108> CMND_MSG_FUN_RECV_IND<02> [IE_FUN [DstDeviceId: 0x0001, SrcDeviceId:
0x0000, AddressType: 0, DstUnitId: 0x0003, MessageSequence: 2, MessageType: 1, InterfaceType:
0, InterfaceId: 0x7f16, InterfaceMember: 0x01, RawData[15]: 68 65 6c 6c 6f 20 66 72 6f 6d 20 62
61 73 65]]
Got Raw FUN message: 'hello from base'
```

8.1.6. Make a voice call

- Run the Nucleo board with the "VoiceCall" example
- In the ule-hub demo:

```
> devices
Device(id=1, ipui=02e9e5b579)
> call 1
16:28:12.059 Device 1: message has been queued for delivery ...
16:28:17.306 Device 1: message delivered.
```

The green LED on the Nucleo should now flash, press the blue button to accept the call.

```
16:28:36.862 Call 0: established with Device 1
```

The call with id 0 has now been established. You should be able to hear and transmit audio on Line-1 of the USB Dongle. To release the call, either press the blue button again, or do it via the ule-hub:

```
> release 0
16:29:31.381 Call 0: device 1 released
16:29:31.479 Call 0: released
```

8.2. Using the han_client module

In order to use the `han_client` python module, simply copy `han_client.py` to your project or configure the python path to look for modules in `base/ule-hub`.

Appendix A: DSPG Test Application

The executable of the DSPG Test Application is located under /base/tools/DSPG Test Application.exe.

A.1. Starting the DSPG Test Application

The Com Port of the USB dongle should be automatically detected. If not, press 'Detect USB Com Port' or set the Com Port which your PC has assigned when the USB dongle was connected. Press 'Start'. All other settings can be ignored.

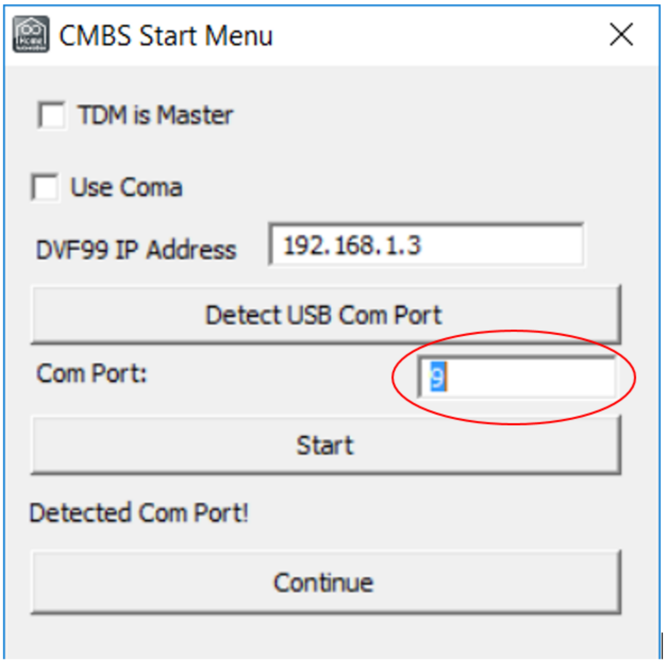


Figure 11. Starting the DSPG Test Application

A.2. Open registration

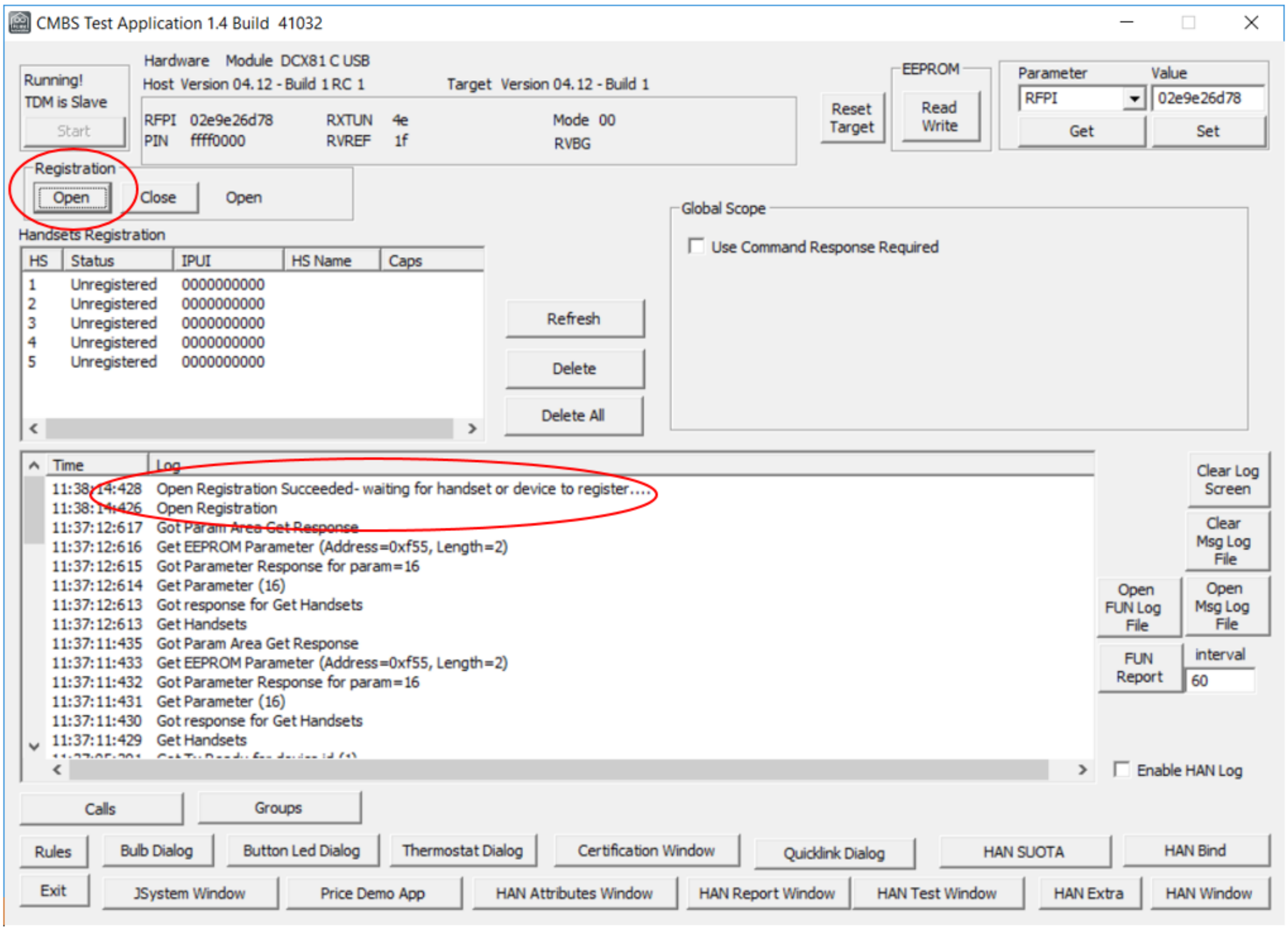


Figure 12. Open base for registration

A.3. Log information during registration

The Error indication can be ignored.

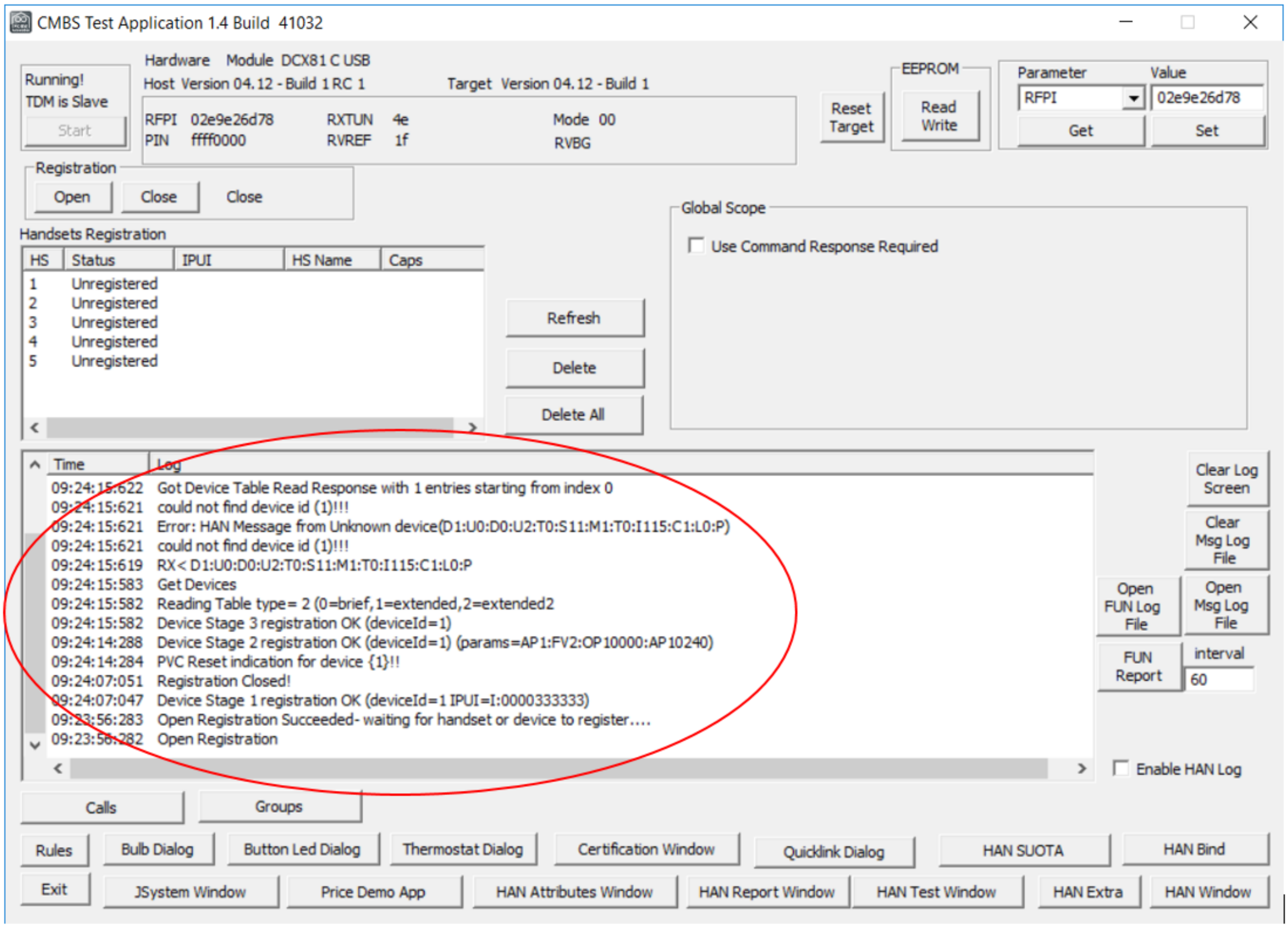


Figure 13. Log information during successful registration

A.4. Event counters

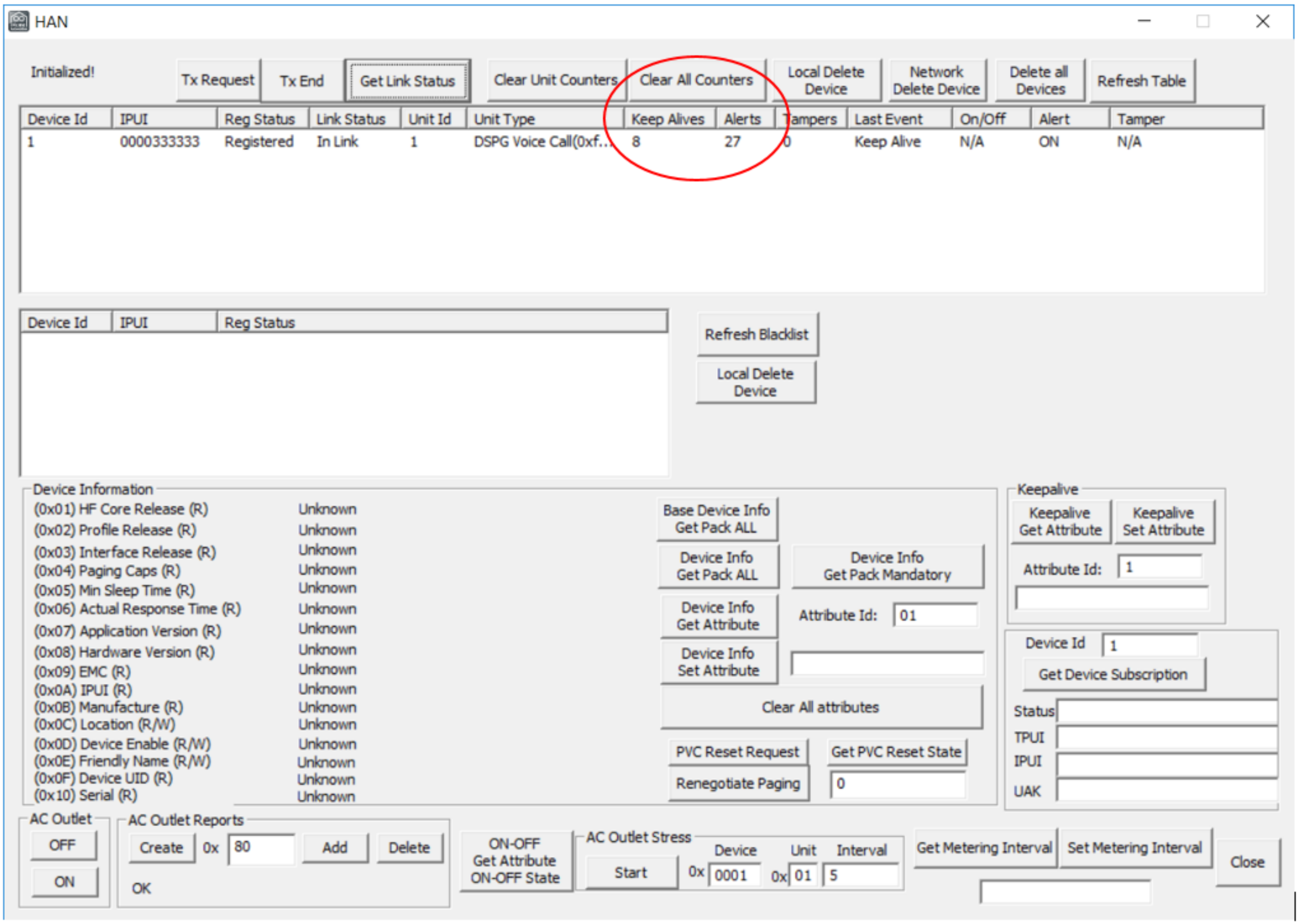


Figure 14. Event Counters (Keep Alive and Alert) in HAN window

A.5. Voice call handling

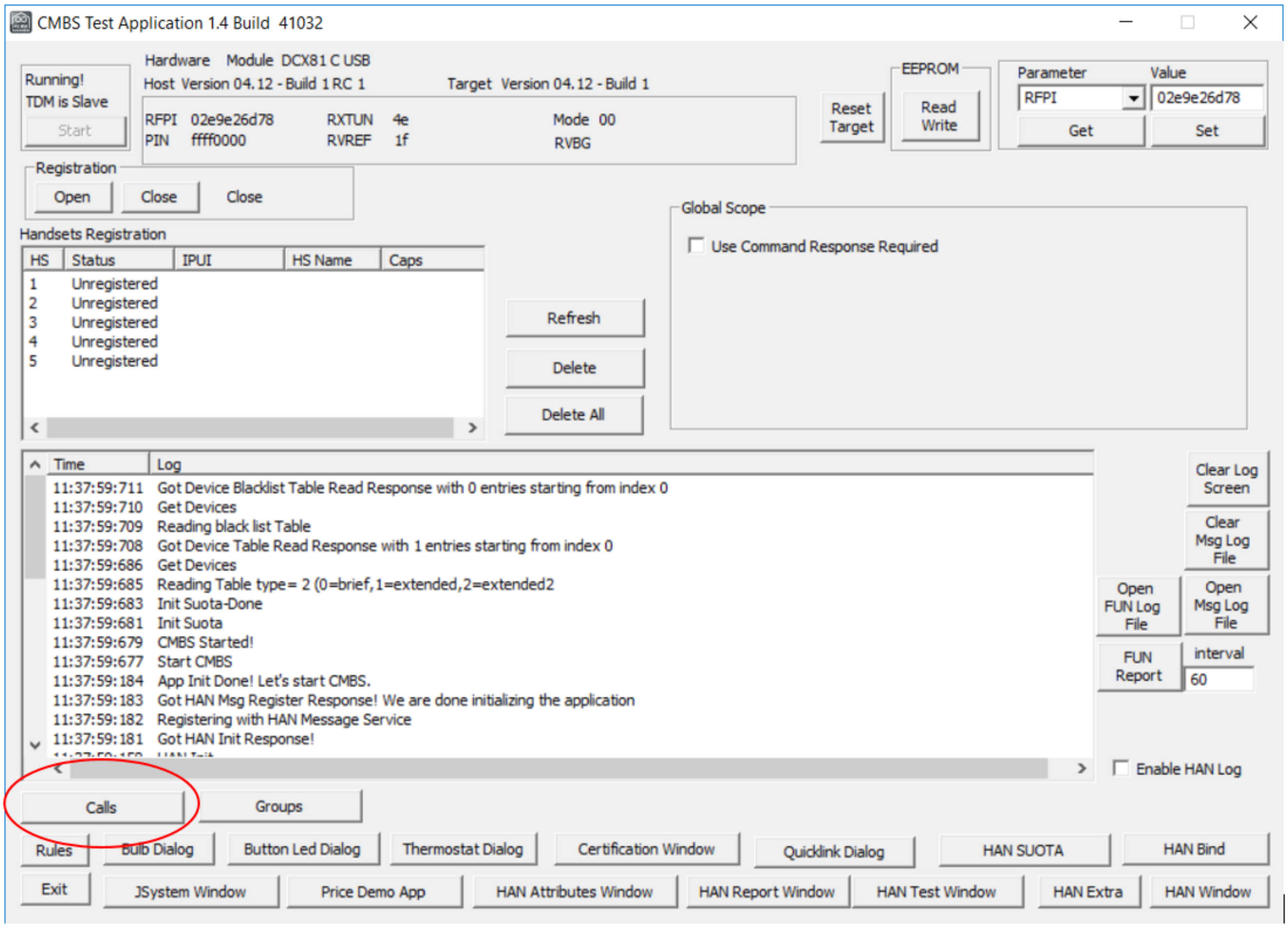


Figure 15. Selecting Calls window

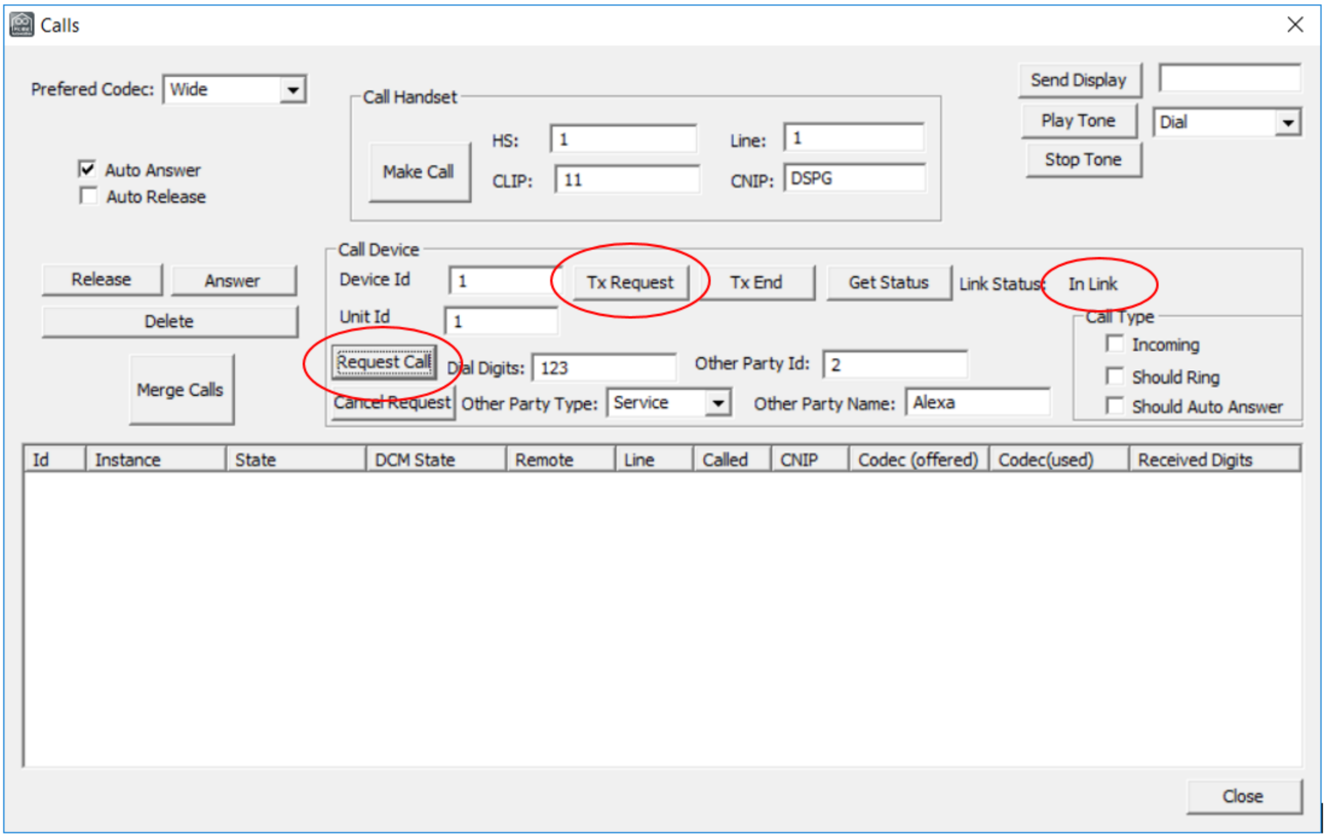


Figure 16. Calls window setup call

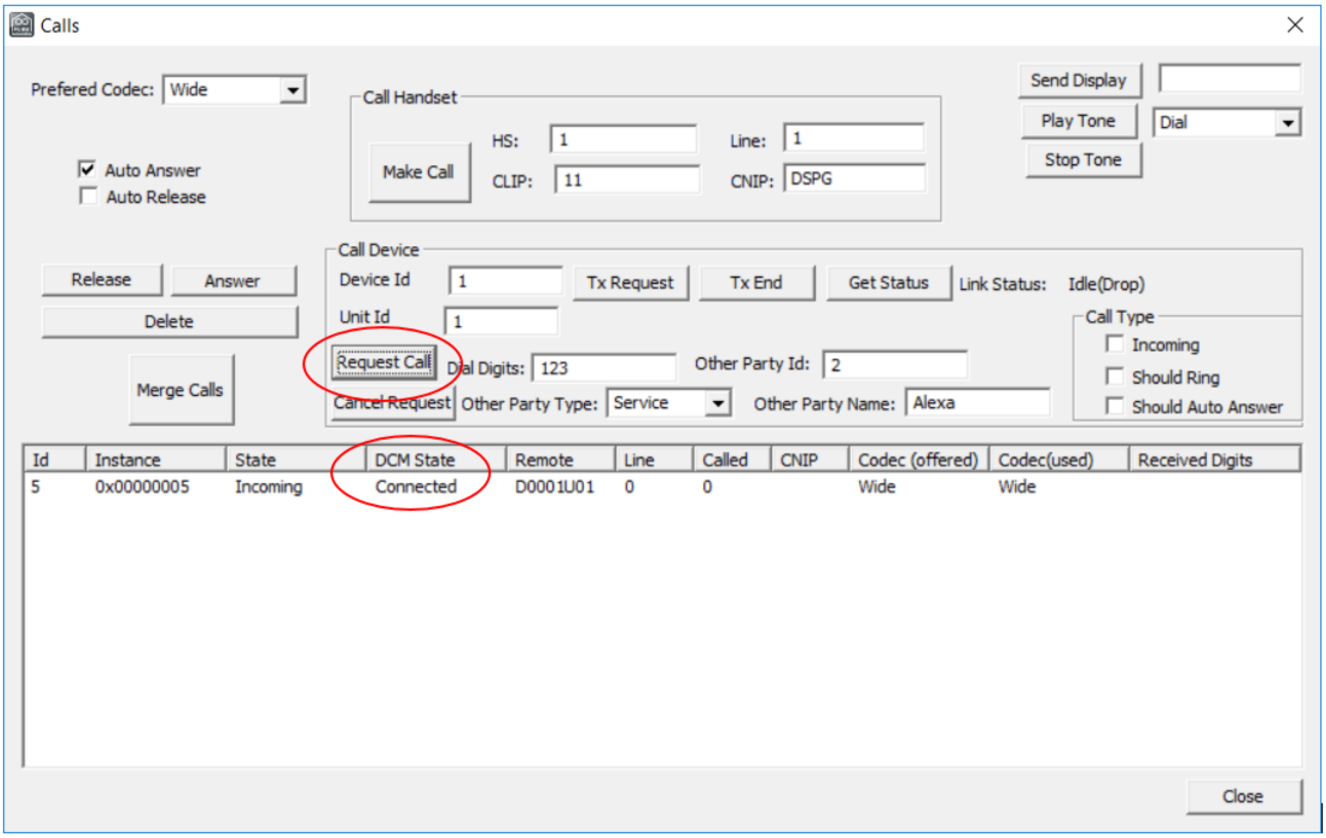


Figure 17. Calls window call connected

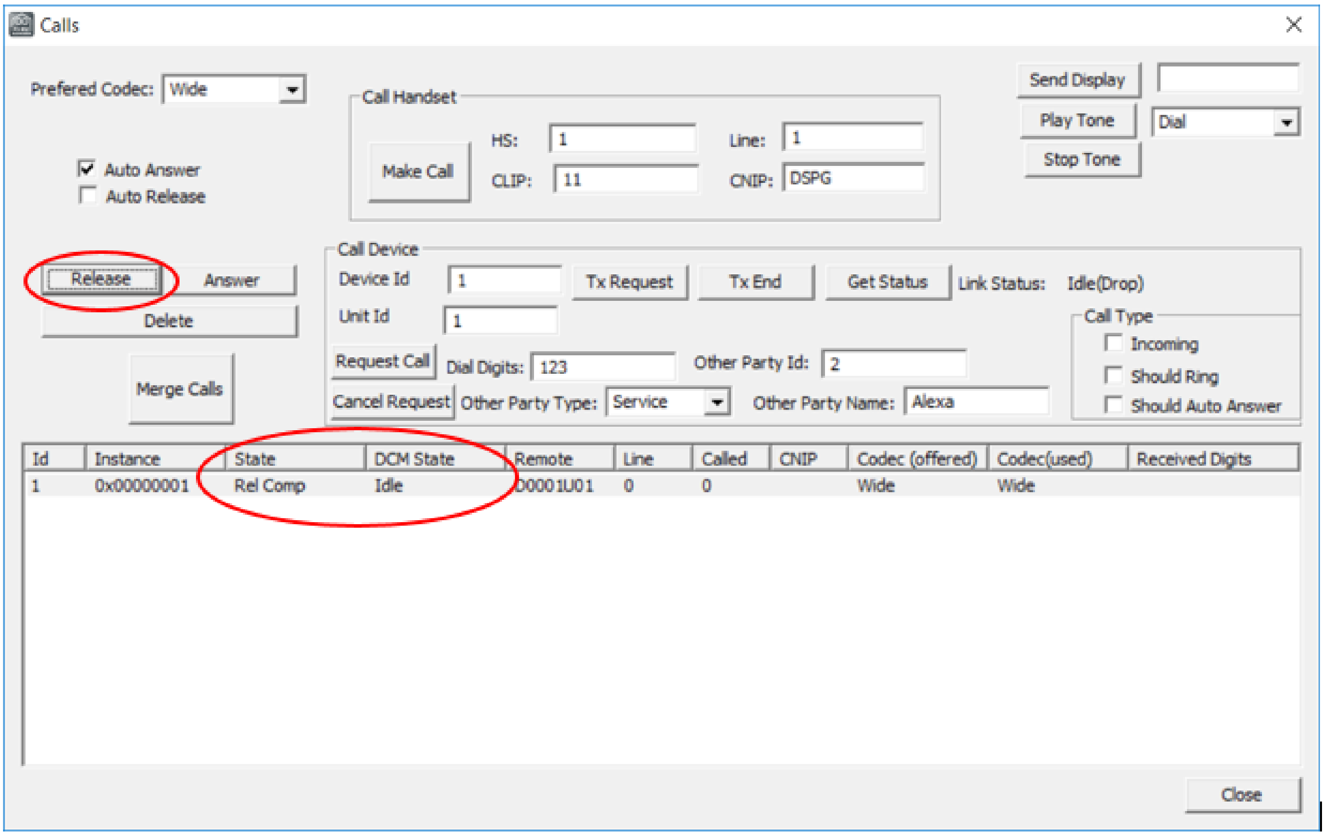


Figure 18. Calls window call release

Appendix B: CMND Simulator

The executable of the CMND Simulator is located under /device/Utilities/CmndApiUartSimulator.exe.

B.1. Starting the CMND Simulator

Select the COM port that your PC has assigned when the DU-EB was connected via USB cable and press 'Start'.

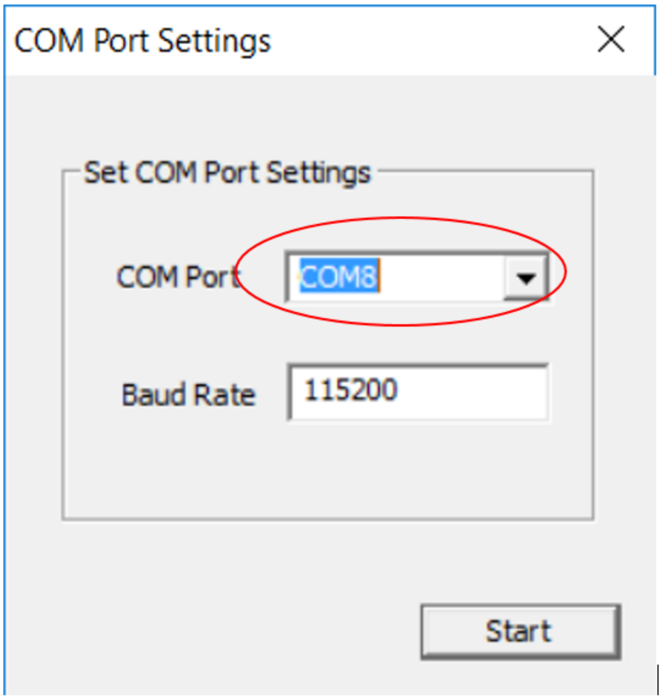


Figure 19. Starting the CMND Simulator

B.2. Sending CMND messages

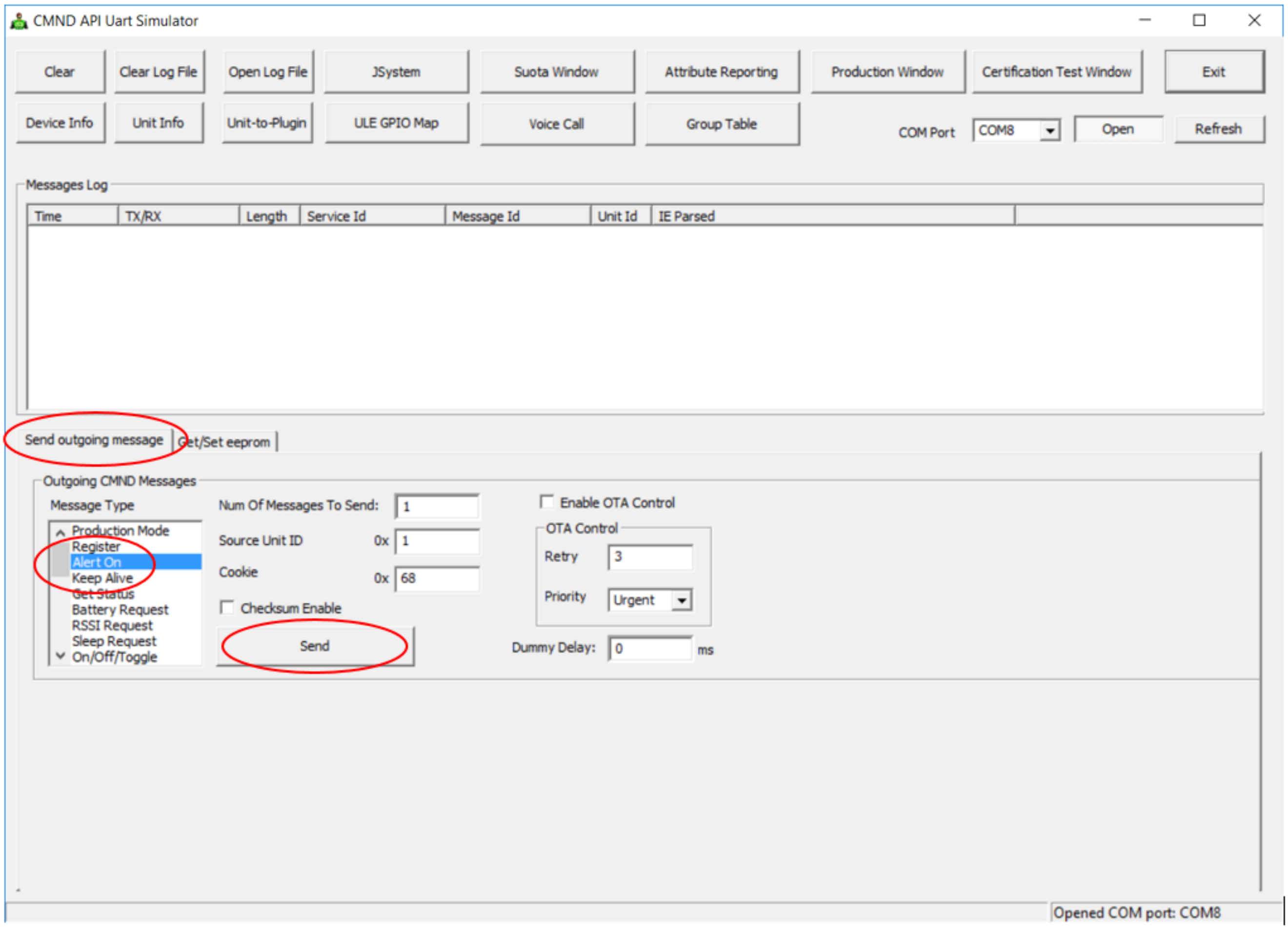


Figure 20. Sending CMND messages

B.3. Voice call handling

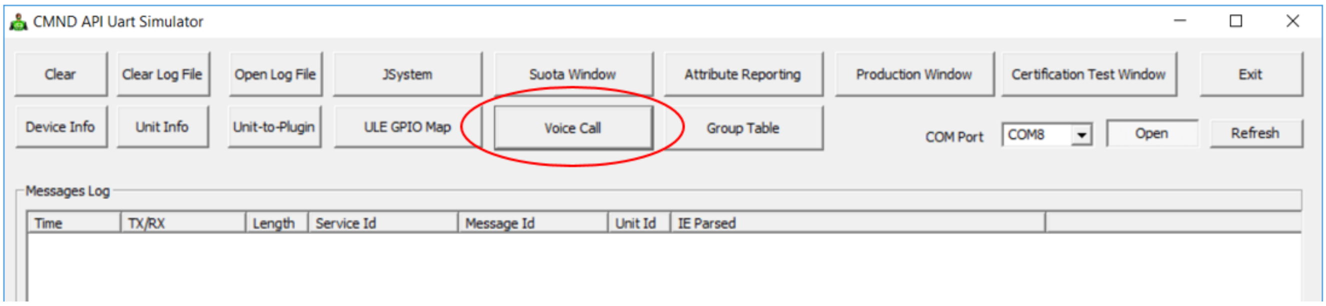


Figure 21. Selecting Voice Call window

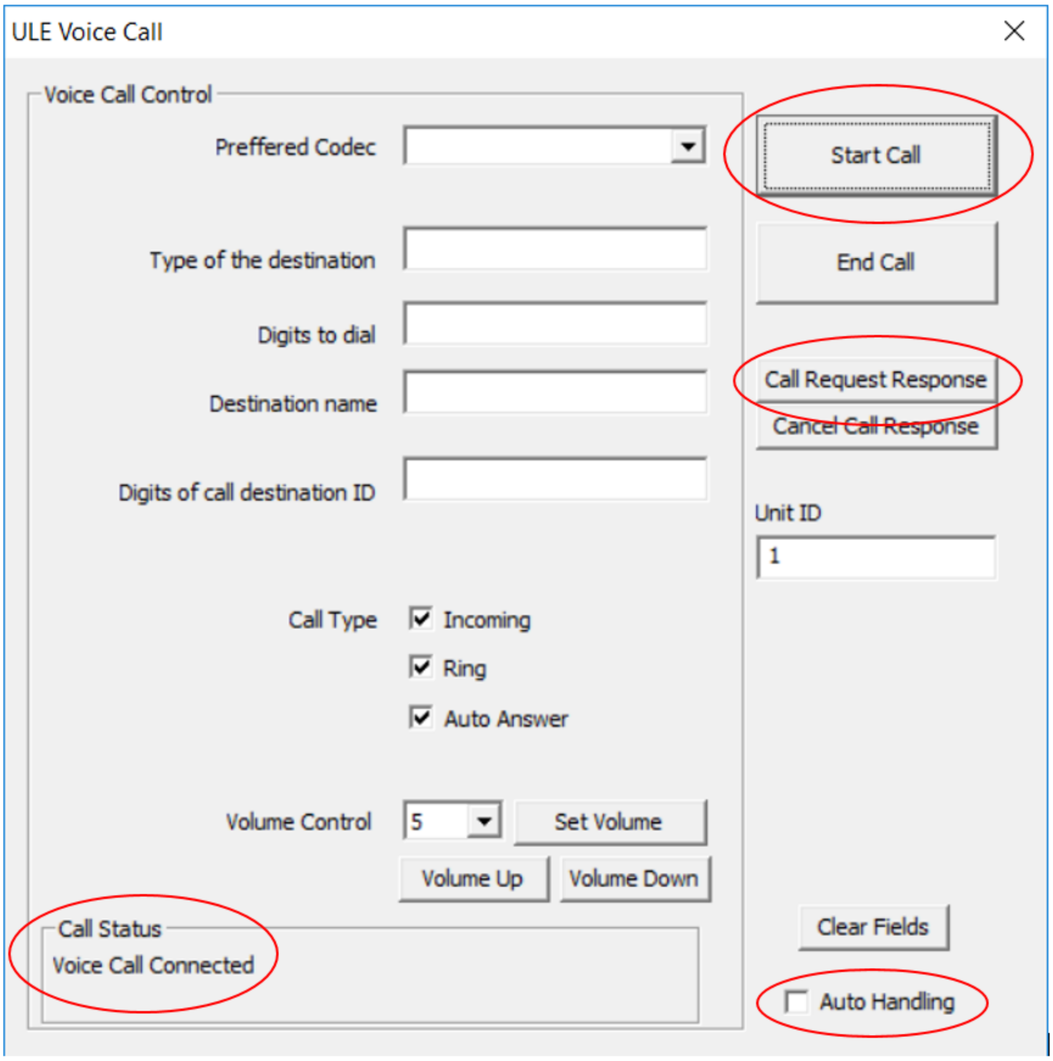


Figure 22. Voice Call window

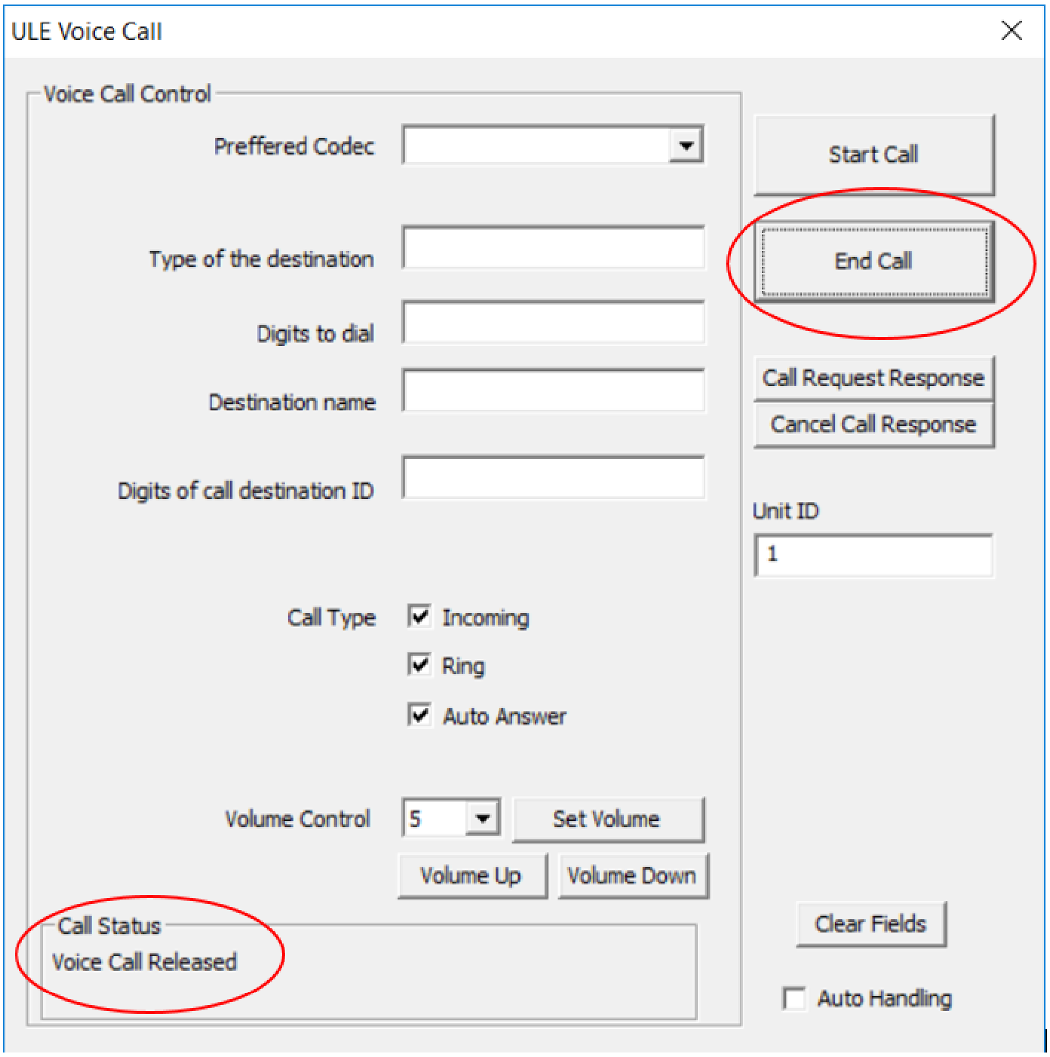


Figure 23. Voice Call window release call

B.4. Changing an EEPROM byte

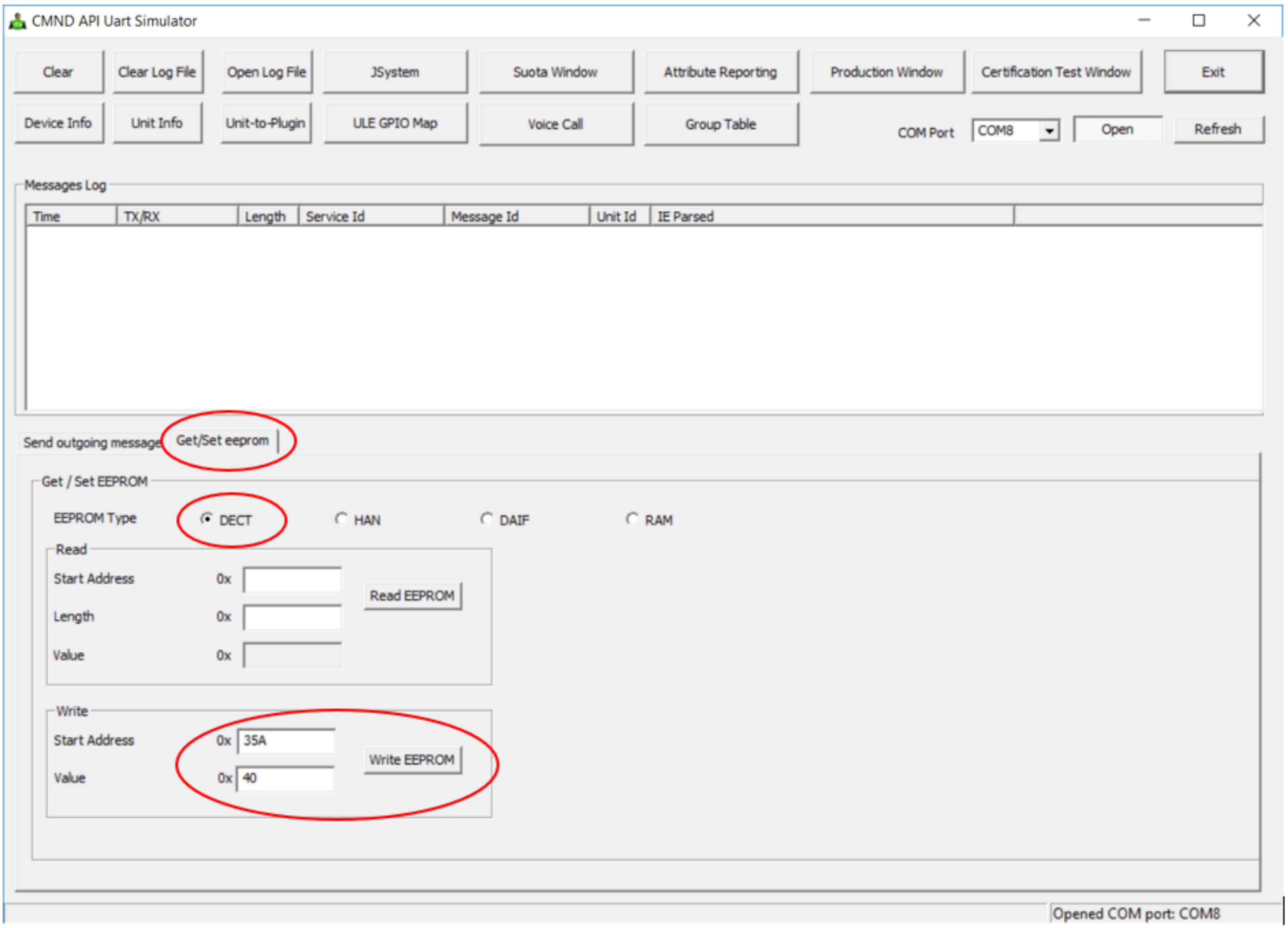


Figure 24. Changing an EEPROM byte

Appendix C: Windows audio routing

C.1. Routing audio from DU-EB microphone to PC

- Connect audio source (e.g. MP3-Player) to the 3.5 mm MIC jack on the DU-EB
- Select 'Handset DSPG-LINE1' as the recording device
- Select 'Properties' of 'Handset DSPG-LINE1', choose the 'Listen' tab and tick the 'Listen to this device' box
- Select your PC audio destination as the 'Playback through this device'

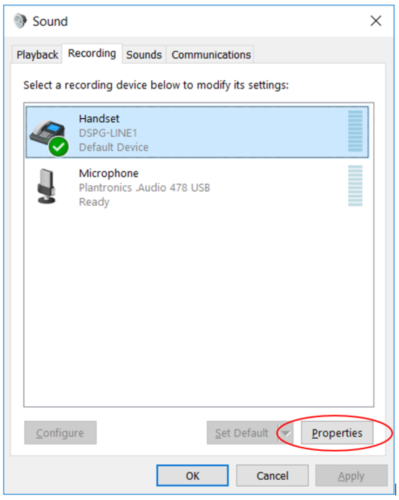


Figure 25. Selecting DSPG dongle as recording device

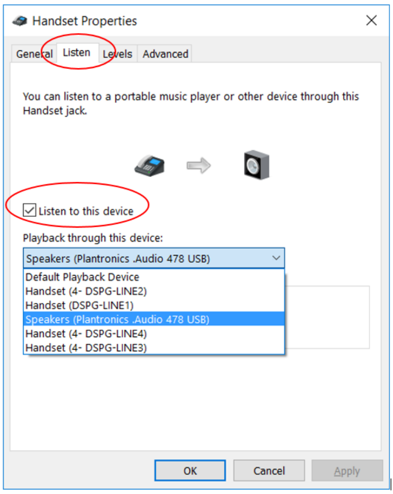


Figure 26. Selecting your PC audio destination as playback device

C.2. Routing audio from PC to the DU-EB speaker

- Connect a speaker/headset/earphones to the 3.5 mm SPK jack on the DU-EB
- Select your PC audio source as the recording device
- Select 'Properties' of your PC audio source, choose the 'Listen' tab and tick the 'Listen to this device' box
- Select 'Handset(DSPG-LINE1)' as the 'Playback through this device'

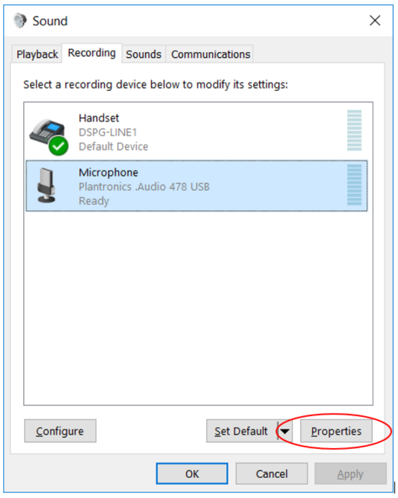


Figure 27. Selecting your PC audio source as recording device

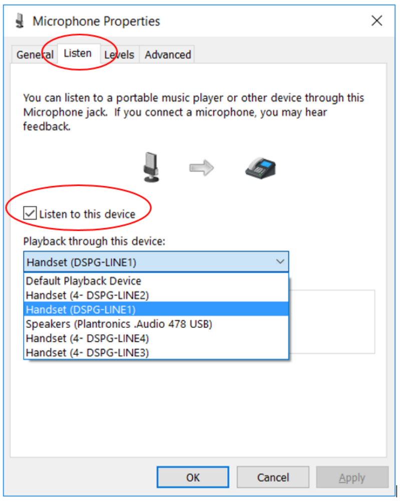


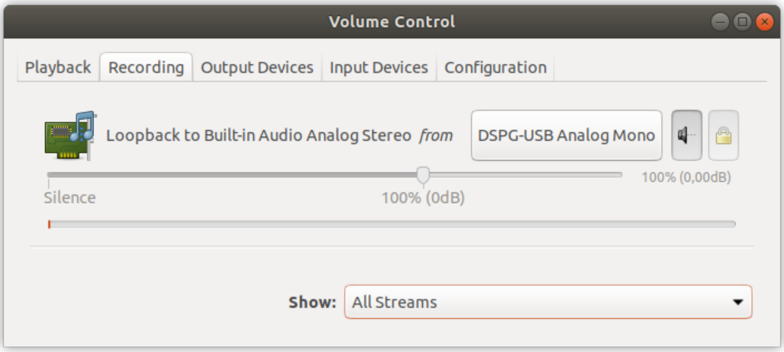
Figure 28. Selecting the DSPG dongle as playback device

Appendix D: Linux audio routing

- Install "PulseAudio Volume Control" from "Ubuntu Software"
- Open a terminal, load pulseaudio loopback module

```
$ pactl load-module module-loopback
```

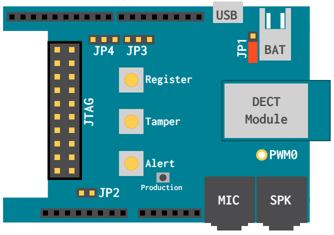
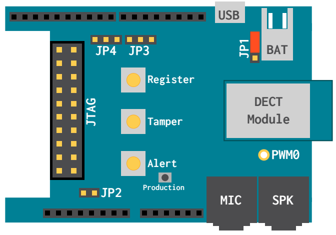
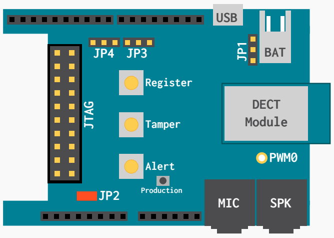
- Start "PulseAudio Volume Control"
- In "Recording" tab, change "Show" to "All Streams"
- Select "DSPG-USB Analog Mono" for Loopback



Appendix E: DU-EB jumper settings

E.1. DU-EB jumper setting for power supply

Power supply configurations are controlled via jumpers JP1 and JP2.

 Power via batteries or power supply	 Power via USB
 Power from host board	

E.2. DU-EB jumper settings for operation with ST-Nucleo

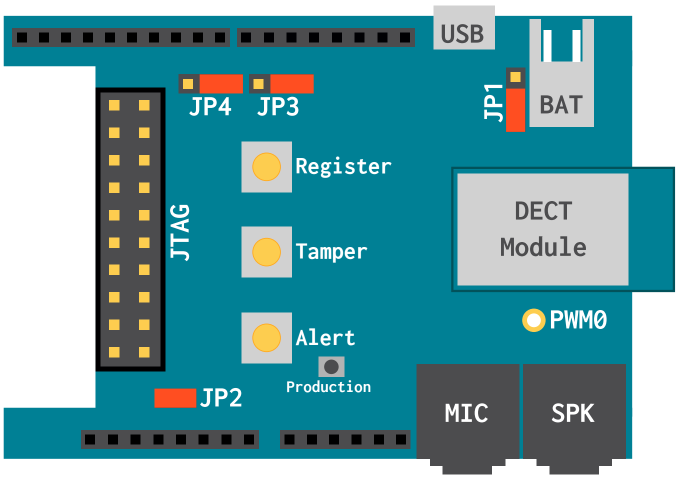


Figure 29. DU-EB jumper settings for ST-Nucleo

Appendix F: ULE Voice Call Interface

This chapter describes the DSPG proprietary ULE Voice Call interface which extends the HAN FUN standard to enable voice calls between device and the base.

For the DECT-ULE Expansion Board, this interface is available at Unit 1.

22. DSPG ULE Voice Call interface (0x7F11)

This interface enables requesting a device to establish a call to the BS.

22.1 Server Attributes

None.

22.2 Client Attributes

None.

22.3 Server to Client Commands

22.3.1 Voice Call Request Status Update

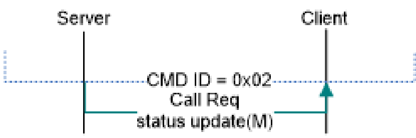


Figure 20 – DSPG ULE Voice Call Interface: Call Request Status Update

This command is sent to a client implementation of the DSPG ULE voice call interface, and tell it the status of the call.

Call Request status:

In Progress – device received the request and processing it

Rejected – device rejects the request

Example, if call request is sent to a device, device can immediately send “In Progress” to indicate the request has been received by the device.

If device is ringing, and after some timeout, device can send status Reject to tell the base it cannot make the call.

Table 50 – Data in Payload of the Call Request Status Update command

Field Name	Field Description	Type	Value	M/O
Status	Status of the call request	U8	0x01- In Progress 0x02 - Rejected	M

- Data Ordering of Payload of call request status update Command

8	7	6	5	4	3	2	1	Octet
Status								**1**

22.4 Client to Server Commands

22.4.1 Voice Call Request

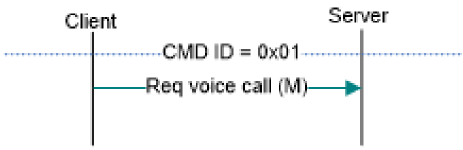


Figure 20 – DSPG ULE Voice Call Interface: Request Call Setup

This command is sent to a server implementation of the DSPG ULE voice call interface, and requests it to establish a voice call.

Table 51- Payload of the commands

Field Name	Field Description	Type		Value	M/O
Preferred Codec Field ID	Field ID	U8		0x1	O
Preferred Codec	Codec preferred by the device	U8		0x00 - NB 0x01 - WB	
Digits Field ID	Field ID			0x2	O
Digits	Digits which should be dialed by the device	U8	Len	0x00 - 0x20	
		U8	Char string	0x00 - 0xFF (each U8)	
Other Party Type Field ID	Field ID	U8		0x3	O
Other Party Type	Other party type	U8		0x00-HS 0x01-Device 0x02-Number 0x03-Service	
Other Party Name Field ID	Field ID	U8		0x4	O
Other Party Name	Other party name	U8	Len	0x00 - 0x20	
		U8	Char string	0x00 - 0xFF (each U8)	
Other Party Id Field ID	Field ID	U8		0x5	O
Other Party Id	Other party id	U8	Len	0x00 - 0x20	
		U8	Char	0x00 - 0xFF	

			string	(each U8)	
Call Type Field ID	`Field ID`	U8		0x6	O
Call Type	`Call type Mask`	U8		Bit 0=Call Direction (1=incoming) (0=outgoing) Bit 1=Should Ring (0=no,1=yes) Bit 2=Auto Answer (0=no,1=yes)	

Table 52- Data Ordering of Payload of call request Command

8	7	6	5	4	3	2	1	Octet
preferred codec filed id								**1**
preferred codec								**2**
Digits field id								**3**
Digits length (N)								**4**
Digit (char 1)								**5**
…								**…**
Digit (char N)								**4+N**
Other party type field id								**5+N**
Other party type								**6+N**
Other party name filed id								**7+N**
Other party name length (M)								**8+N**
Other party name (char 1)								**9+N**
…								**…**
Other party name (char M)								**8+N+M**
Other party id field id								**9+N+M**
Other party id len								**10+N+M**
Other party id (char 1)								**11+N+M**
Other party id (char K)								**10+N+M+K**
Call Type field								**11+N+M+K**
Call Type								**12+N+M+K**

22.4.1.1 Other Party Type

0x00-HS- Indicates the other party is a Handset
0x01-Device- Indicates the other party is a ULE Device
0x02-Number – Indicates the other party is Phone number (Voip or PSTN)
0x03-Service- Indicates the other party is a Service (examples: alexa, voice mail, voice notification service)

22.4.1.2 Call type

Device Behavior table

Direction Bit (Bit LSB 0)	Ring Bit (Bit LSB 1)	Auto Answer Bit (Bit LSB 2)	Device Behavior
0 (outgoing)	0	0 or 1	Makes an outgoing call
0 (outgoing)	1	0	Ring till answer(callback Feature)
0 (outgoing)	1	1	Ring once and make an outgoing call
1 (Incoming)	0	0	INVALID
1 (Incoming)	0	1	auto answer
1 (Incoming)	1	0	Ring till answer
1 (Incoming)	1	1	Ring once and answer the call

22.4.2 Cancel Voice Call Request

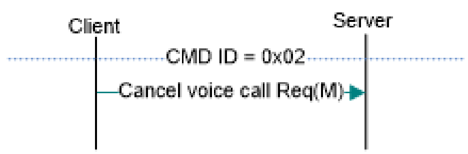


Figure 20 – DSPG ULE Voice Call Interface: Cancel Call Request

This command is sent to a server implementation of the DSPG ULE voice call interface, and requests it to cancel a previously sent voice call request.

This command has no payload

Appendix G: Java Troubleshooting

G.1. Error 1603

This error can be a result of your virus scanner blocking the installation of browser or shell extensions. Disable your virus scanner and retry.

G.2. Error 1607

This error can be resolved by temporarily deactivating "Java Content in Browser":

- Press Windows key, type "Configure Java", press Enter
- Alternatively, find the Java Control Panel in the system preferences
- Select the "Security" tab
- Uncheck "Enable Java Content in Browser" check box, click "Apply"

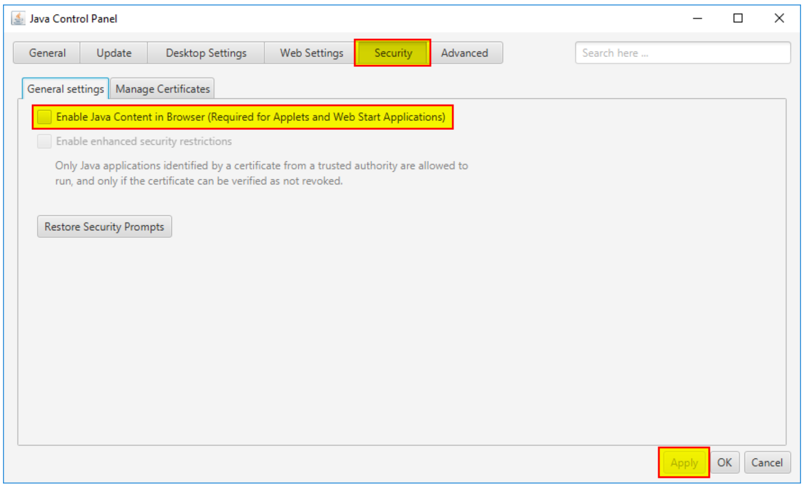


Figure 30. Java Control Panel

To make sure all settings are applied, reboot your PC. Once the PC has been rebooted, proceed with running the Java installer again. It should finish successfully now. Once it finished, optionally enable "Java Content in Browser" again.