DSP GROUP®

Getting Started

With the ULE Starter Kit

DSP Group Inc.

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 its 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
HF	HAN-FUN
HAN-FUN	Home Area Network FUNtional
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
[2]	CMBS HAN SERVER PROTOCOL	1.15	26 June, 2018	/doc/CMBS HAN SERVER PROTOCOL.pdf
[3]	DECT-ULE Expansion Board User Manual	3.0	October 17, 2018	/doc/DECT-ULE Expansion Board User Manual.pdf
[4]	DECT-ULE Expansion Board Schematics	REV 2	August 09, 2017	/doc/DECT-ULE Expansion Board Schematics.pdf
[5]	fwtool manual	0.1	-	/fwtool/README.pdf

Chapter 2. Known Issues

2.1. DU-EB Expansion Board issues

- Board will not power up if run from a USB wall plug, please connect to a Laptop/PC/USB Hub
- `MIC` jack is a line-in (no microphone power provided)
  - `MIC` jack only records the right channel
- `SPK` jack is a differential output (designed to drive small 4/8ohm speakers)
  - Normal line-out headsets/speaker may produce hum

2.2. DU-EB Expansion Board Firmware issues

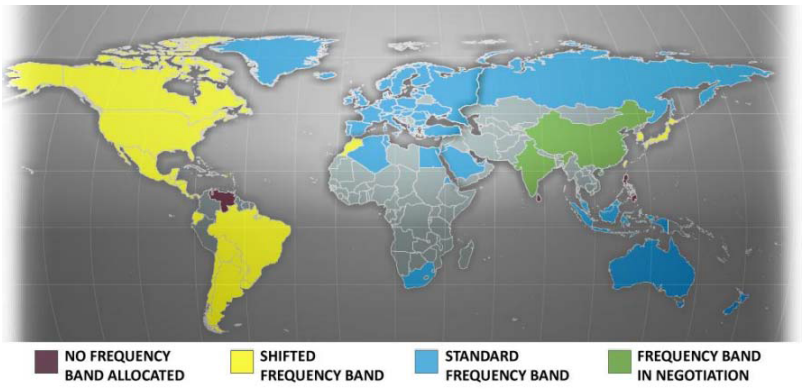
- Firmware will drop CMND messages just after sending `CMND_MSG_PARAM_SET_*` messages
  - Wait 1000ms after receiving the corresponding `CMND_MSG_PARAM_SET_*RES` response before sending the next `CMND` message

Chapter 3. Support

For further information, questions or comments please do not hesitate to contact us via our developer portal at https://www.dspg.com/developers.

Chapter 4. DECT-ULE System Overview

This chapter provides a quick overview of 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 uses a 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. Devices can either be battery or line powered.

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

DECT is a synchronous system, the base transmits a beacon and devices align their timing to it. The beacon carries system relevant information such as identity and base capabilities.

In DECT, only devices can establish data links between base and device. If the base wants to send information and is not currently in a link with a device, it has to contact the device and ask it to establish a link. 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 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.

4.1. 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

4.2. 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). All the devices and bases in one area must have unique identities.

4.3. Registration

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

- The base and device share a secret authentication key
- The base and device negotiate ULE capabilities
- A logical device number is assigned to each device, this allows easy addressing of devices within a given DECT system
    - Logical device numbers are unique within a base, unlike the RFPI and IPEI they do not have to be geographically unique
    - If the same device is registered to another DECT base it will probably 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 a device to register to a base, the base must be set to accept new devices. This is known as opening the base for registration
- For security reasons, the base will only remain open for a certain time (e.g. 2 minutes). If the device did not manage to register in time, the base needs to be opened again
- The base will close for registration once it starts to register a device so if a second device is to be registered the base needs to be opened 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 device registration from the base

e.g. to use this device with another base.

- In ULE deregistration is triggered by the base
- Deregistration can only proceed when the device is in contact with the base
- If deregistration is started at the base side while the device is asleep, the base records this device's state as blacklisted
- The next time a blacklisted device contacts the base deregistration starts automatically
- It is also possible to unilaterally deregister a device at the base without the device making contact, this method can be used if the device is lost or non-functional

4.4. Security

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

4.5. 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)
- Respond to an incoming voice call

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 only have to listen to the beacon 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 have not negotiated paging details cannot be paged.

4.6. ULE Data Transfer

Only the ULE device can directly initiate a data transfer. Therefore, only a ULE device can send data to the base at any time (e.g. on certain sensor conditions). If the base needs to push data to the device it needs the device initiate contact and this always results in 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 4.7)
- The base can wait for any other asynchronous data transfer started by the device, to push its own data to the device

4.7. 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 a burglar alarm sensor has been damaged or run out of battery then an alarm should be raised. To address this requirement most ULE sensors will periodically report their health status to the base using an extremely short data transfer. This method is called 'keep alive'.

4.8. HAN-FUN

DECT-ULE uses an application layer protocol called HAN-FUN (Home Area Network FUNctional

protocol), the HAN-FUN protocol uses the lower level protocols described by the ETSI DECT-ULE standards. The intention is to ensure interoperability of ULE devices.

HAN-FUN was developed by the ULE Alliance, which is an industry consortium of companies with the goal of promoting DECT-ULE. Further details about the HAN-FUN standard and the ULE Alliance can be found in Chapter 8 or under www.ulealliance.org.

4.9. Radio Frequency Band

DECT-ULE can be deployed in many countries worldwide. Country specific regulations often require different radio settings from those used in Europe, this can change the frequencies to be used, the transmit power, or the applied channel selection rules.

In order to comply with the law, make sure to use the appropriate setting for the country where the ULE Starter Kit is used.

Chapter 5. DECT-ULE Starter Kit Overview

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

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

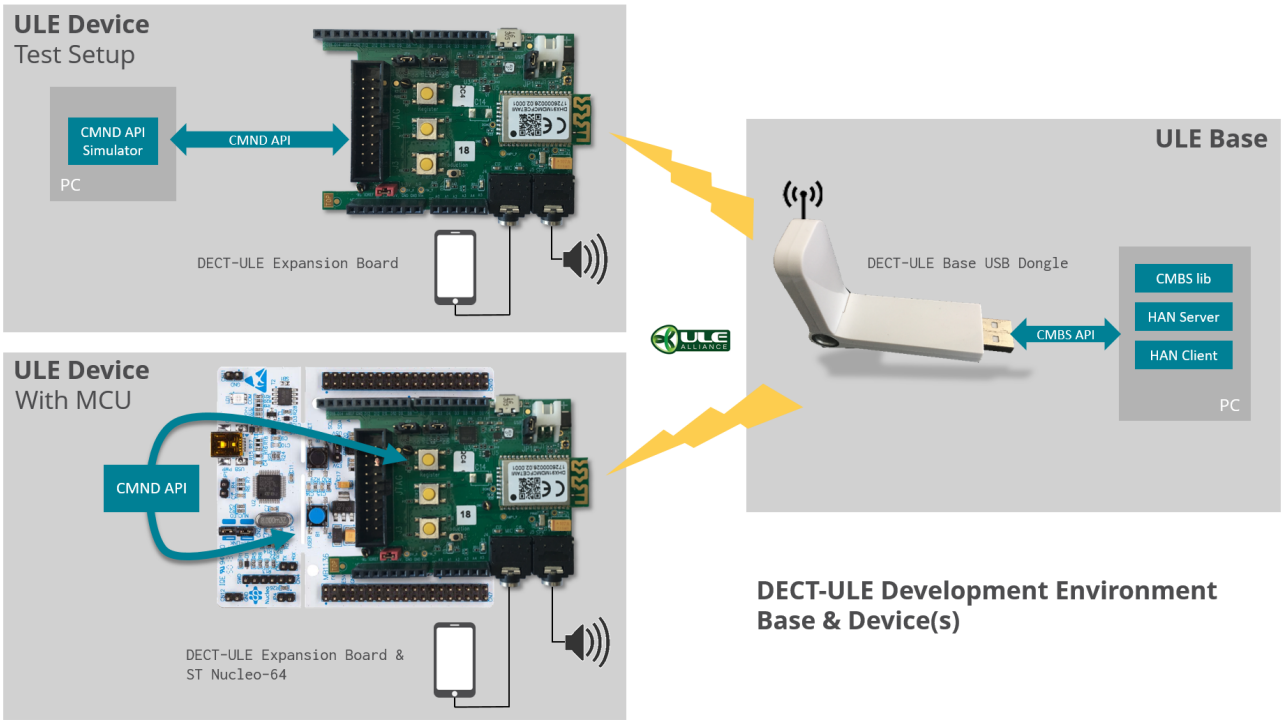


Figure 1. Hardware and software elements of the ULE Starter Kit

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

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

The DECT-ULE Expansion Board features the DSP Group DHAN-J Module which comes loaded with dual-mode (data and audio) ULE device firmware. The firmware incorporates physical, MAC and transport layers, enabling it to function as a DECT-ULE device.

The DECT-ULE Expansion Board can be stacked on top of an Arduino R3 compatible board.

The DECT-ULE Expansion Board can be configured via jumpers to support the following power sources and UART connectivity:

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

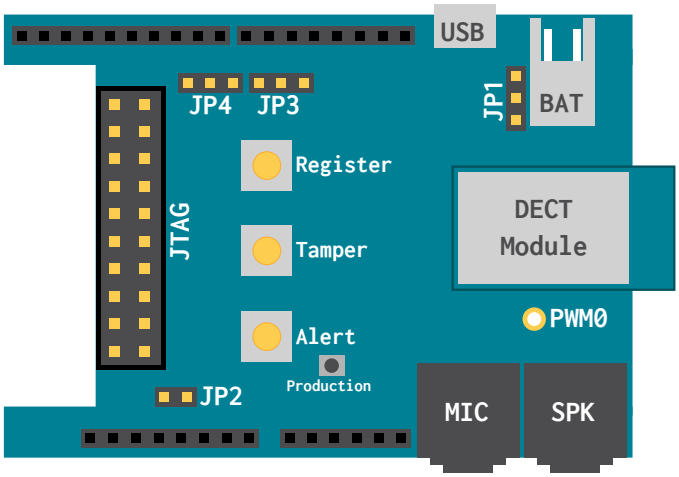


Figure 2. DECT-ULE Expansion Board Overview

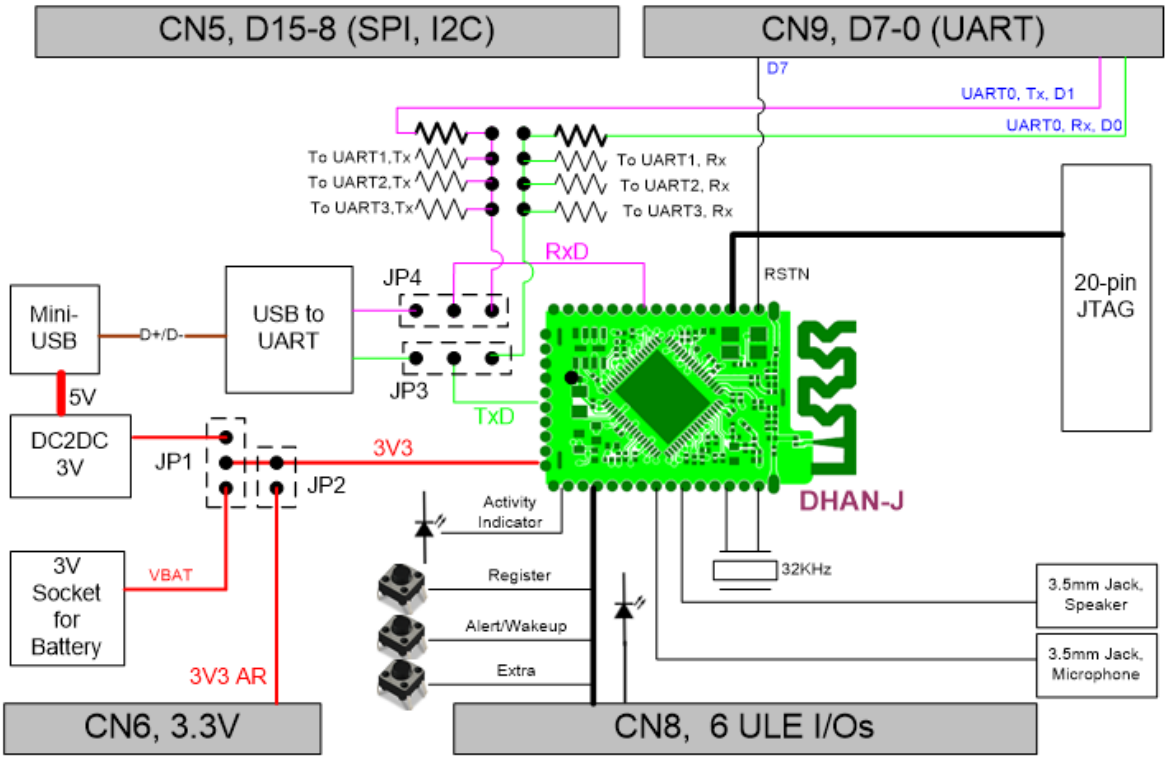


Figure 3. DECT-ULE Expansion Board Schematics

5.2. DECT-ULE Base USB Dongle

The DECT-ULE Base USB Dongle comes loaded with dual-mode (data and audio) ULE base firmware which enables it to function as a DECT-ULE base. It is powered from the PC's USB port and requires no physical configuration.

Chapter 6. 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.

This section uses GUIs that only run on Windows, if you do not have access to a Windows machine see section 6.8

Please make sure your board is configured with the right jumper settings to use power from the USB connection and to route the UART via the USB:

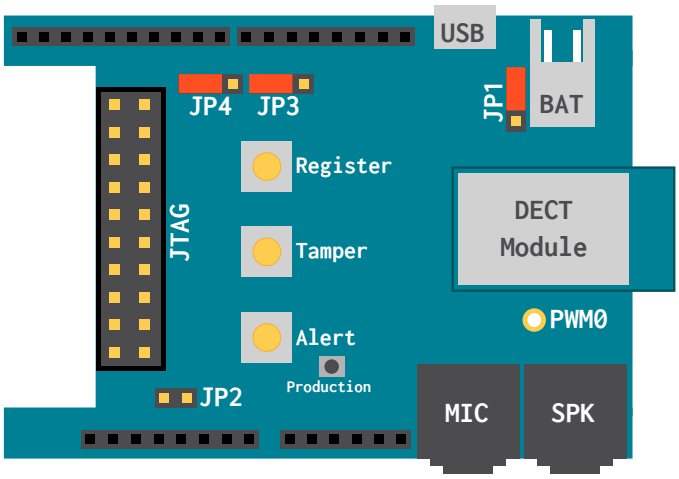
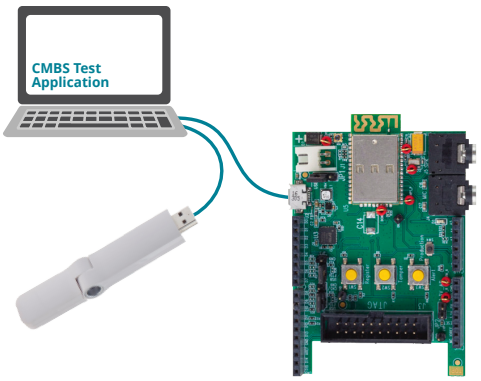


Figure 4. DU-EB: Powered from USB, UART to USB

Powering the expansion board from a wall plug will not work! See Chapter 2.

6.1. Set up the environment

- Connect a Windows PC to the USB dongle and start the DSPG Test Application (see appendix A.1)
- Connect a Windows PC to the Expansion Board using a USB cable



6.2. Regional settings

Depending on your location you may need to configure all your hardware devices for a specific earth region in order to ensure operation within the boundaries of your local wireless regulation authority. By default, all hardware devices are configured for the `eu` region (Europe).

If you operate outside of Europe, please use the `fwtool` shipped as part of this package in order to select a different region configuration. The `fwtool` is a console application, therefore open a terminal window and navigate to the `fwtool/bin` directory of the release package.

List all available devices:

```
> fwtool list
COM??:cmbs
COM??:cmnd
...
```

Then update the region settings on all the devices, one by one. For example for the `us`:

```
> fwtool --dev COM??:cmbs region us
...
> fwtool --dev COM??:cmnd region us
...
```

Please refer to the `fwtool` manual for further information on the supported regions.

6.3. 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 2 minutes have elapsed.

6.4. 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)

6.5. Make a voice call

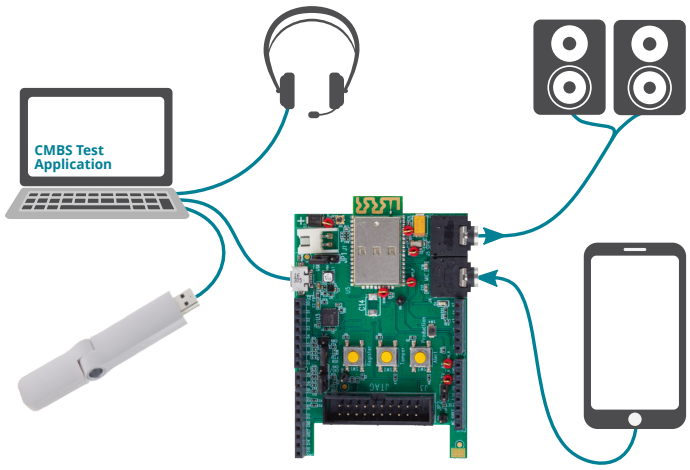
A major advantage of ULE is that it uses DECT to provide a clear and protected voice link.

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 see Appendix C



- 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

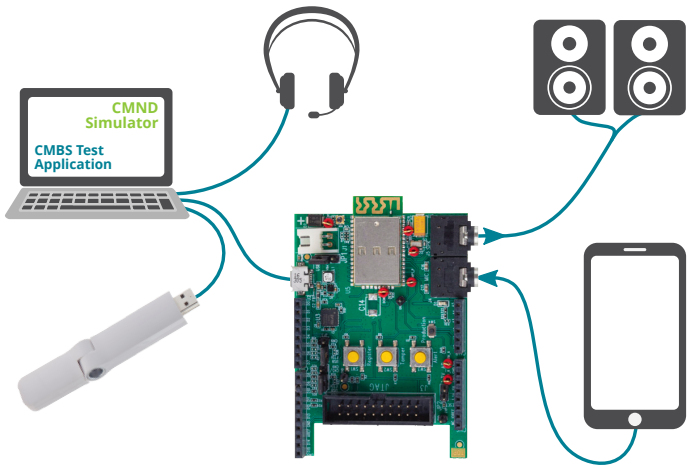
6.6. Use the CMND API

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

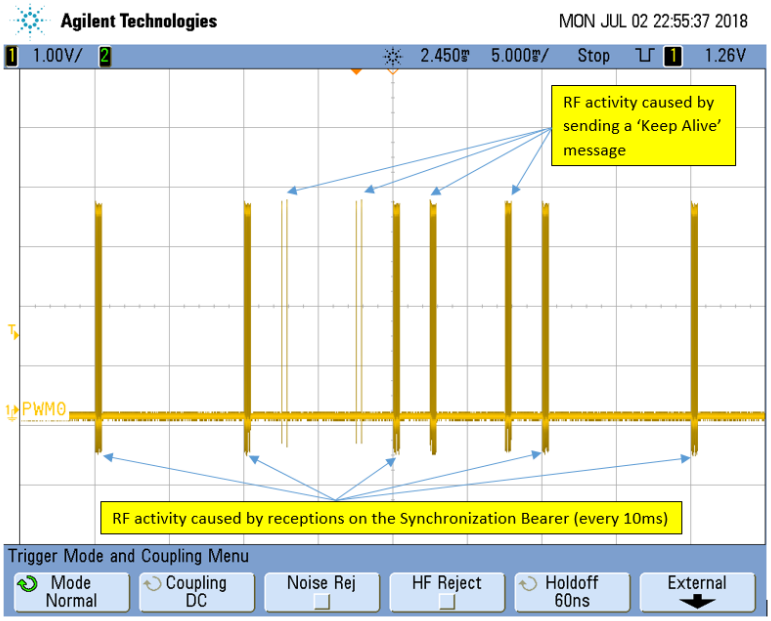
See appendix B.2

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

6.7. 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



6.8. Hands on with Linux

The DSPG Test Application and the CMND Simulator used in the sections above only run with Windows but it is possible to do some of the exercises described there using a Linux PC rather than a Windows PC. The base can be controlled using the development program described in Chapter 10 and the DU-EB can be controlled using the physical buttons.

Step through section 10.1 until the han_app is running. Then the first thing to do is register the device to the base.

Register

This is described in section 10.1.4.

Send an alert

Press the Alert button (SW3) on the DU-EB, the ule-hub demo window will show the received alert message.

Make a voice call

See Appendix D for advice about audio routing on the Linux PC.

Follow the instructions in section 6.5.

Chapter 7. Development Setup

Chapter 9 and Chapter 10 contain example software to control the device and base. Several tools are required in order to work with these examples:

- 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

This chapter describes how to install tools.

Only 64 bit operating systems are supported. 32 bit operating systems may work to some extend, but there are known issues.

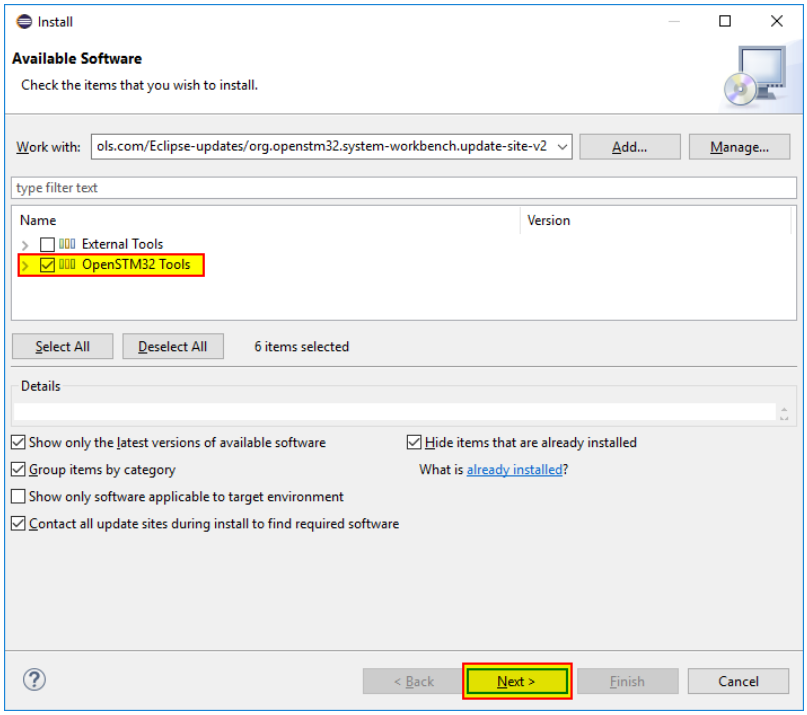
7.1. System Workbench Installation

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" if warned about unsigned packages
- Restart Eclipse when asked to so do

7.2. 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.

During the Python 2.7.x installation the option "Add python.exe to Path" must be enabled to allow typing `python` into a command prompt without needing the full path.

For Python 3.x you should see something similar to 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 something similar to 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.
>>>
```

7.3. STLink Driver Installation

Windows

Navigate to the `device\Utilities\STLink` directory of the ULE Starter Kit, unzip the zip file, and doubleclick `dpinst_x86.exe` for 32 bit Windows or `dpinst_amd64.exe` for 64 bit Windows. 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 8. HAN-FUN Standard Overview

This chapter provides a quick overview of the HAN-FUN (HF) standard. It is useful background for the code examples in Chapter 9 and Chapter 10 but it could be skipped just now and consulted as reference material when studying the examples in detail.

The HAN-FUN standard comprises several documents, each handling special aspects like protocols, services, interfaces, profiles, interworking, etc. It was developed by the ULE Alliance, an industry consortium which promotes DECT-ULE. Further details about the ULE Alliance and the HAN-FUN standard can be found at https://www.ulealliance.org.

The HAN-FUN standard is designed to support applications from areas like home automation, security, smart energy and health within a Home Area Networks (HAN). Although it can be used with several different transport layers, it is optimized for operating over a DECT-ULE transport layer without hampering other transport layers such as TCP/IP.

8.1. HF Network Topology

HAN-FUN operates as a star network with Devices linked to a central Hub. This is the same topology as DECT systems. Each entity in the network has a unique address.

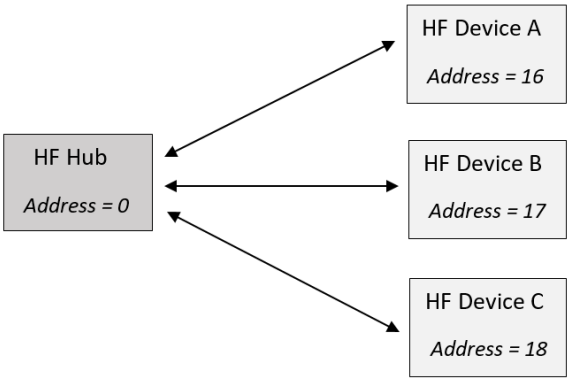


Figure 5. HAN-FUN network topology

8.2. HF Network Entities

8.2.1. HF Hub

The HF Hub is the center of the star network. The Hub runs the HF protocol stack on top of a transport layer, and some, or all, of the HF network services described in 'HF-Service' document. An HF Hub may include all the processing necessary for the network to operate or it may delegate some, or all, of that processing to a third party, e.g. a cloud service.

There is only one hub in a HF network and it is always assigned HF address 0x0000.

In the ULE Starter Kit the DECT-ULE USB dongle is the HF Hub.

8.2.2. HF Device

All entities other than the HF Hub that implement the HF protocol are called HF devices. From a DECT point of view, each of these HF Devices is a DECT Portable Part (PP).

In order to join the HF network each HF device must be registered using the procedures of the transport layer, DECT-ULE in this case, and then registered to the HF network. During the HF registration process the HF Hub allocates the HF Device an address in the range 0x0001 to 0x7FFE. HF Device address 0x7FFF is a reserved address used for 'unknown' devices.

In the ULE Starter Kit the HF Device comprises the DHAN-J DECT-ULE radio (mounted on the DU-EB) and an STM32 MCU host.

8.2.3. HF Unit

An HF Unit is a conceptual entity that instantiates the functionality of a specific HF Profile inside an HF Device. HF Profiles include smoke detectors, switches, etc. They are described in the 'HF-Profile' document. HF messages are always exchanged between HF Units.

An HF Device can host up to 254 HF Units either instantiating the same HF Profile or a variety of HF Profiles. Each HF Unit has a fixed pre-defined identifier (ID). The HF Unit ID combined with the HF Device address produces a fully qualified HF network address that allows HF Units to communicate with each other.

Two special HF Unit IDs exist.

- Unit ID 0x00 is reserved for implementation of the 'Device Management' service and other HF core services. These are described in the 'HF-Service' document.
- Unit ID 0xFF is a reserved ID required for the correct operation of some HF core services.

As an example, the following figure shows a simple HF network of one HF Hub and one HF device. The HF Hub 'Management' profile is on HF Unit 0. The HF Device 'Management' profile is on HF Unit 0 and an there are two 'AC Outlet' profile instances (HF Unit 1 and 2).

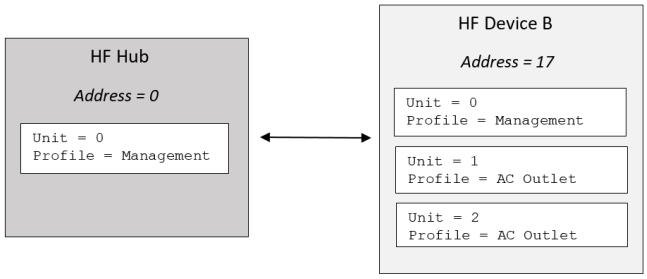


Figure 6. HAN-FUN Units and Profiles

The HF Hub management profile can be accessed at 'Address 0 Unit 0'.

The HF Device management profile can be accessed at 'Address 17 Unit 0'.

One AC Outlet profile can be accessed at 'Address 17 Unit 1'.

The other AC Outlet profile can be accessed at 'Address 17 Unit 2'.

8.2.4. HF Interface

An HF Interface is a conceptual entity inside an HF Unit that defines a collection of commands and attributes which allow HF Units in the system to communicate with one another. An HF Unit can have mandatory and optional HF Interfaces. Each HF Interface must be assigned a client or server role. An HF Interface implementing the client side communicates with an HF Interface that implements the server side. For more details on HF Interfaces consult the 'HF-Interface' document.

The DHAN-J on the DU-EB accommodates up to eight optional HF Interfaces.

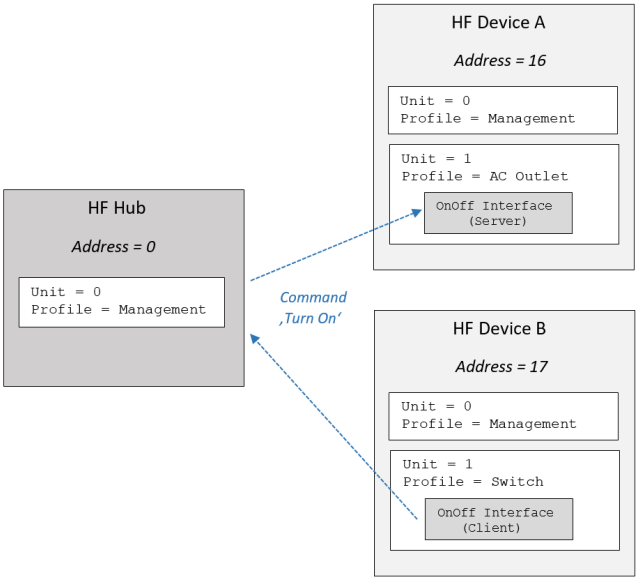


Figure 7. HAN-FUN Interfaces

8.3. HF Services

There are a set of mandatory HF Services which are required for the correct operation of an HF network and a set of optional services that provide advanced network features which may be useful on certain applications. Each HF Service has an associated HF Interface.

The HAN-FUN standard defines the mandatory and optional services for the HF Hub and the HF Devices.

The HF Service 'Device Management' which is responsible for the HF Device registration and discovery is a mandatory service for the HF Hub.

For more details on HF Services consult the 'HF-Service' document.

8.4. HF Profiles

An HF Profile is a collection of HF Interfaces which define a specific functionality for an application. HF Profiles achieve this by describing which HF interfaces and HF services an application must implement. In addition, when applicable, an HF Profile defines how information must be stored and interpreted on the relevant HF Interfaces.

For example, the HF Profile 'Simple Detector' details all HF Interfaces and HF Services that allow an HF Unit to sense and send an alert.

For more details on HF Profiles consult the 'HF-Profile' document.

8.5. HF Messages

HF Messages are composed of a 'Network Layer' part and an 'Application Layer' part.

The 'Network Layer' part of the HF Messages is responsible for identifying the HF Device or HF Devices involved in a communication. It carries the information required to deliver the message (HF destination address, address type and HF destination Unit ID) and to enable a receiver to identify the

sender (HF source address and HF source Unit ID).

The 'Application Layer' part of the HF Messages is responsible for the handling and processing of application data. It carries HF specific protocol information (about the HF Interface to communicate with) in addition to the application data itself. Requests are general messages that usually operate over an HF Interface's attributes (either individually or as a pack) and may or may not involve a response. Responses can be either general or interface-specific.

For more details on HF Messages consult the 'HF-Protocol' document.

8.6. HAN-FUN usage in code examples

8.6.1. Base

The following figure shows the HF Units and HF Interfaces configured in the USB Dongle. In addition to the 'Management' Unit 0 there is one HF Unit for routing the received messages to an external Host Application (e.g. the DSPG Test Application). Mandatory HF Interfaces are in bold.

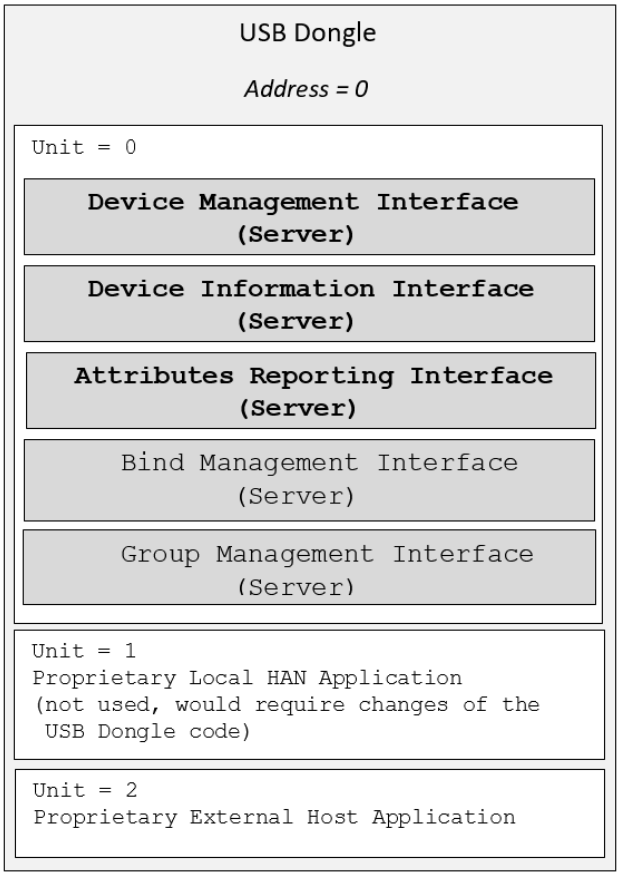


Figure 8. HAN-FUN Units and Interfaces of the USB Dongle

Message routing within the USB dongle is based on two general binding rules which are set during start up of the USB dongle.

- Any message for an unknown HF Device (0x7FFF) and an unknown HF Unit (0xFF) will be routed to device address 0 and unit 2, i.e. to the External Host Application.
- Any message (independent of the specified HF Interface) for HF Device 0 will be routed to device address 0 and unit 2, i.e. to the External Host Application.

The CMND API on the device side does not allow specifying a destination HF Device and destination

HF Unit for most of the messages, e.g. Alert On. The target code on the DU-EB sets the destination HF device to 0x7FFF and the destination HF Unit to 0xFF for these messages. In the base these messages are automatically routed to the External Host Application.

8.6.2. Device

The following figure shows the HF units and HF interfaces configured in the DHAN-J on the DU-EB. In addition to the 'Management' Unit 0 for the HF Device registration there are 3 HF Units for covering the scenarios from the example code. Mandatory HF Interfaces are in bold.

The 'Voice Call' and 'Raw Data' scenarios are realized by proprietary HF profiles.

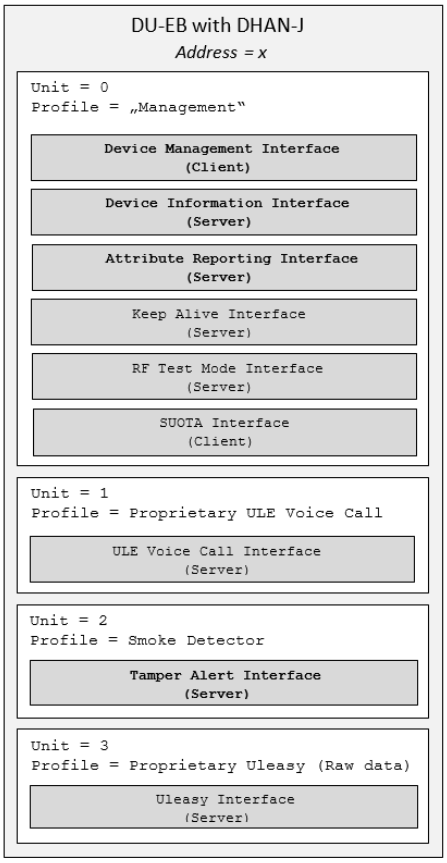


Figure 9. HAN-FUN Units and Interfaces of DU-EB

The proper source HF Device, source HF Unit and the HF Interface must be selected when sending a CMND command, e.g. to send raw data to the base.

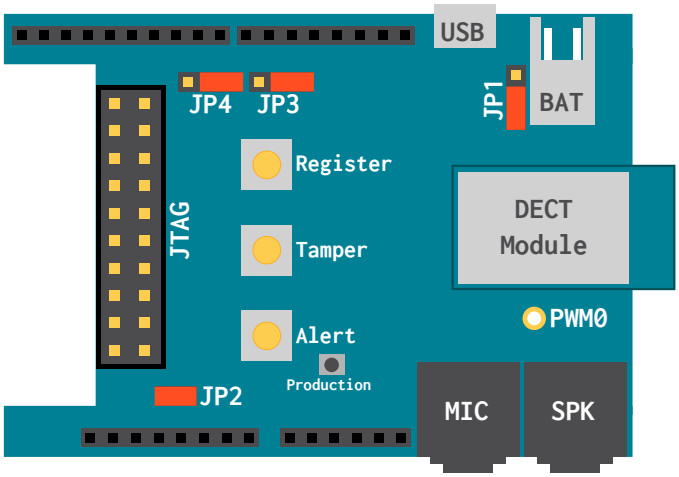
```
// prepare FUN IE structure
st_IeFun.u16_SrcDeviceId = u16_DeviceId; // from our ID
st_IeFun.u8_SrcUnitId = DSPG_RAW_DATA_UNIT_NUMBER;         // Unit = 3
st_IeFun.u16_DstDeviceId = 0; // to Base
st_IeFun.u8_DstUnitId = 2;
st_IeFun.u16_InterfaceId = DSPG_RAW_DATA_INTERFACE_ID;     // Proprietary Interface = 0x7F16
st_IeFun.u8_InterfaceType = 1;
st_IeFun.u8_InterfaceMember = 1;
st_IeFun.u8_AddressType = 0;
st_IeFun.u16_DataLen = u16_DataSize;
st_IeFun.u8_MessageType = CMND_FUN_MSG_TYPE_COMMAND;
// copy the data into FUN structure (with limit of memory available to hold the buffer)
memcpy( st_IeFun.pu8_Data, pu8_Data, st_IeFun.u16_DataLen );
// send to base
p_Fun_SendReq( &st_Packet, &st_IeFun );
```

According to the message routing rules in the base, value 0x7FFF for the destination HF Device and value 0xFF for the destination HF Unit could also be used.

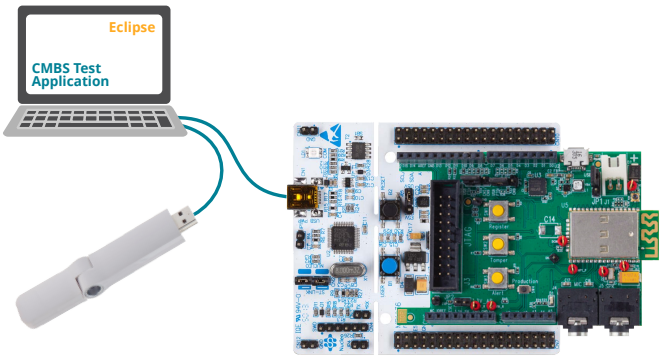
Chapter 9. Device Development

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 used 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 the jumper configuration needs to be set so that the expansion board is powered from the Nucleo and the UART is routed to the connectors. In contrast to the jumper configuration for the "Hands On" scenarios in Chapter 6, jumper JP2 (Power) has to be connected and jumpers JP1 (Power), JP3 (UART) and JP4 (UART) have to be changed to the positions as shown in the figure below.



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



9.1. Import, Build and Run Examples

All the examples are in the `device/Examples` directory of the ULE Starter Kit package. All the example projects are structured and work in the same way. As a demonstration of how to use the Eclipse environment you will now import, build and run the "Registration" example.

- 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"

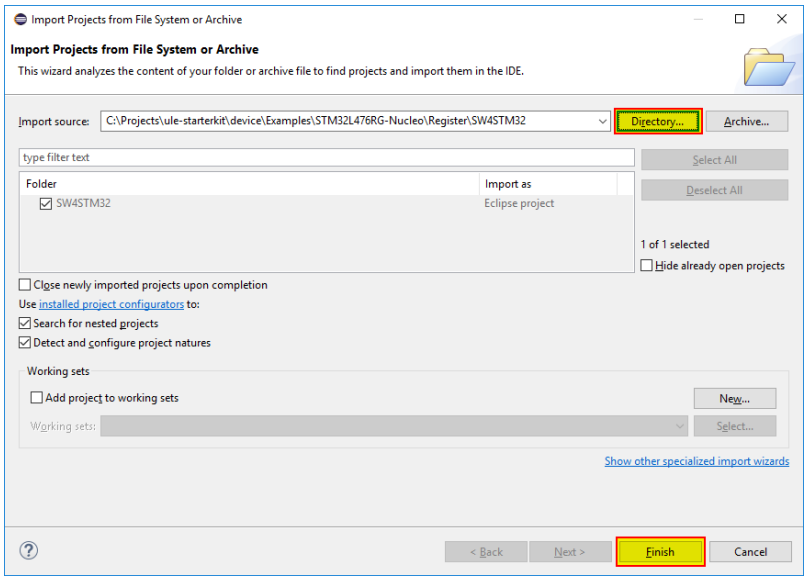


Figure 10. Importing a project

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.

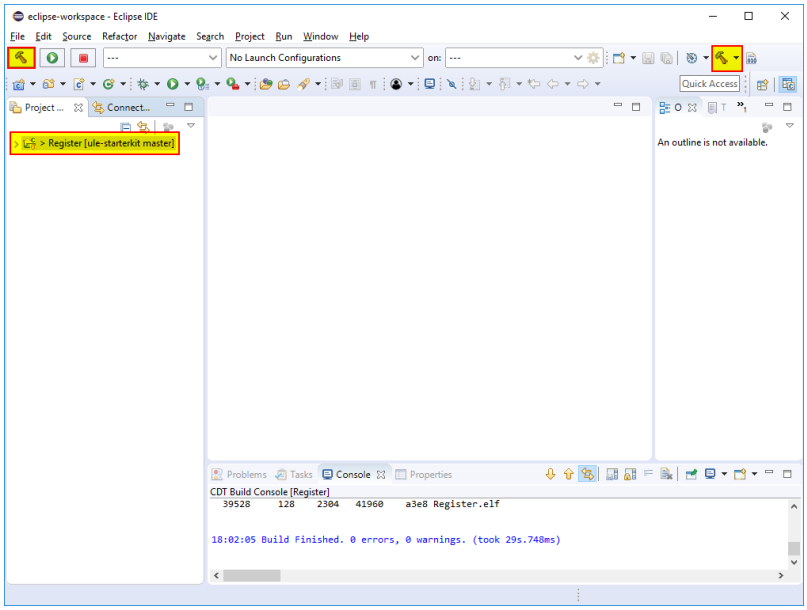


Figure 11. Building a project

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

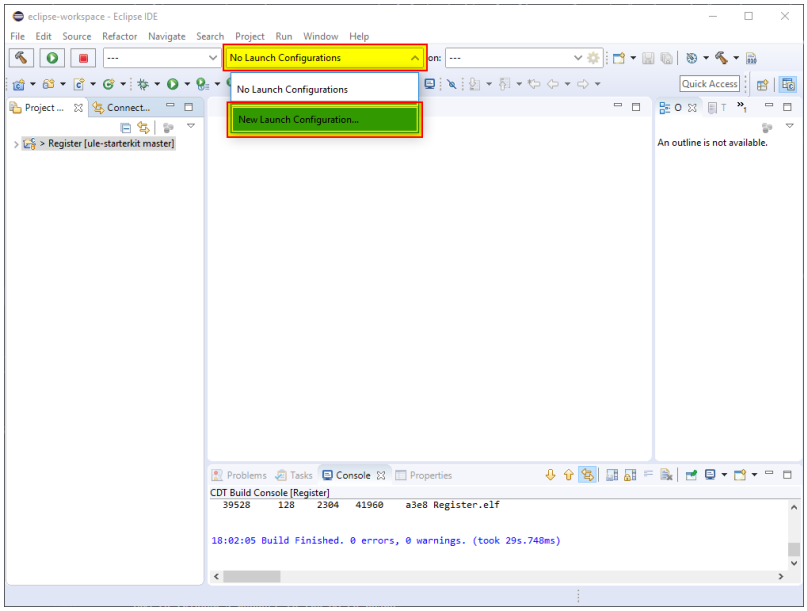


Figure 12. Creating a Launch Configuration

- 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

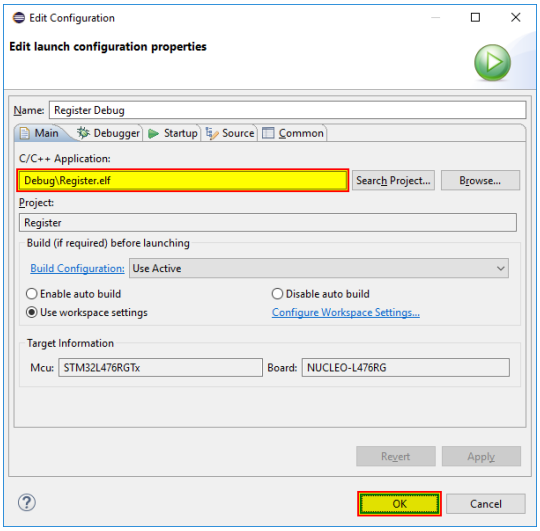


Figure 13. Reviewing the Launch Configuration

- 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.

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

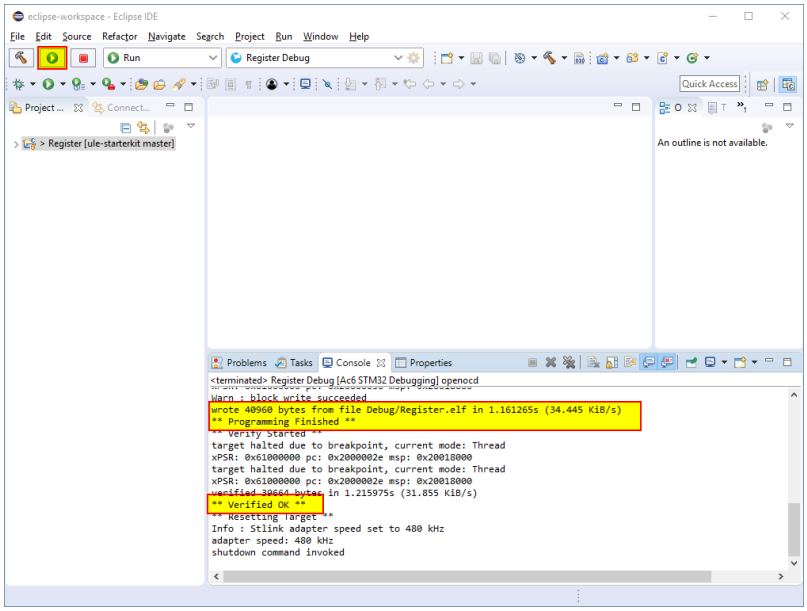


Figure 14. Running examples

9.2. 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

9.3. Example Behaviour

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

9.3.1. 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`, baud rate 115200.

This is the part of the log produced by the RawData example:

```
[*] 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
```

9.3.2. 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.

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

The specific action for the example is triggered when the blue button is pressed. The example is in an infinite loop waiting for the button.

For all examples except registration, the device must be registered to the base for the example to work.

9.3.4. 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)

9.4. Included examples

9.4.1. Register

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

- Load the Register example code into Eclipse and run the code as shown in chapter 9.1.
- 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).

9.4.2. 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.

- Load the Alert example code into Eclipse and run the code as shown in chapter 9.1.
- Pressing the blue button on ST-Nucleo triggers the alert.

9.4.3. Raw Data

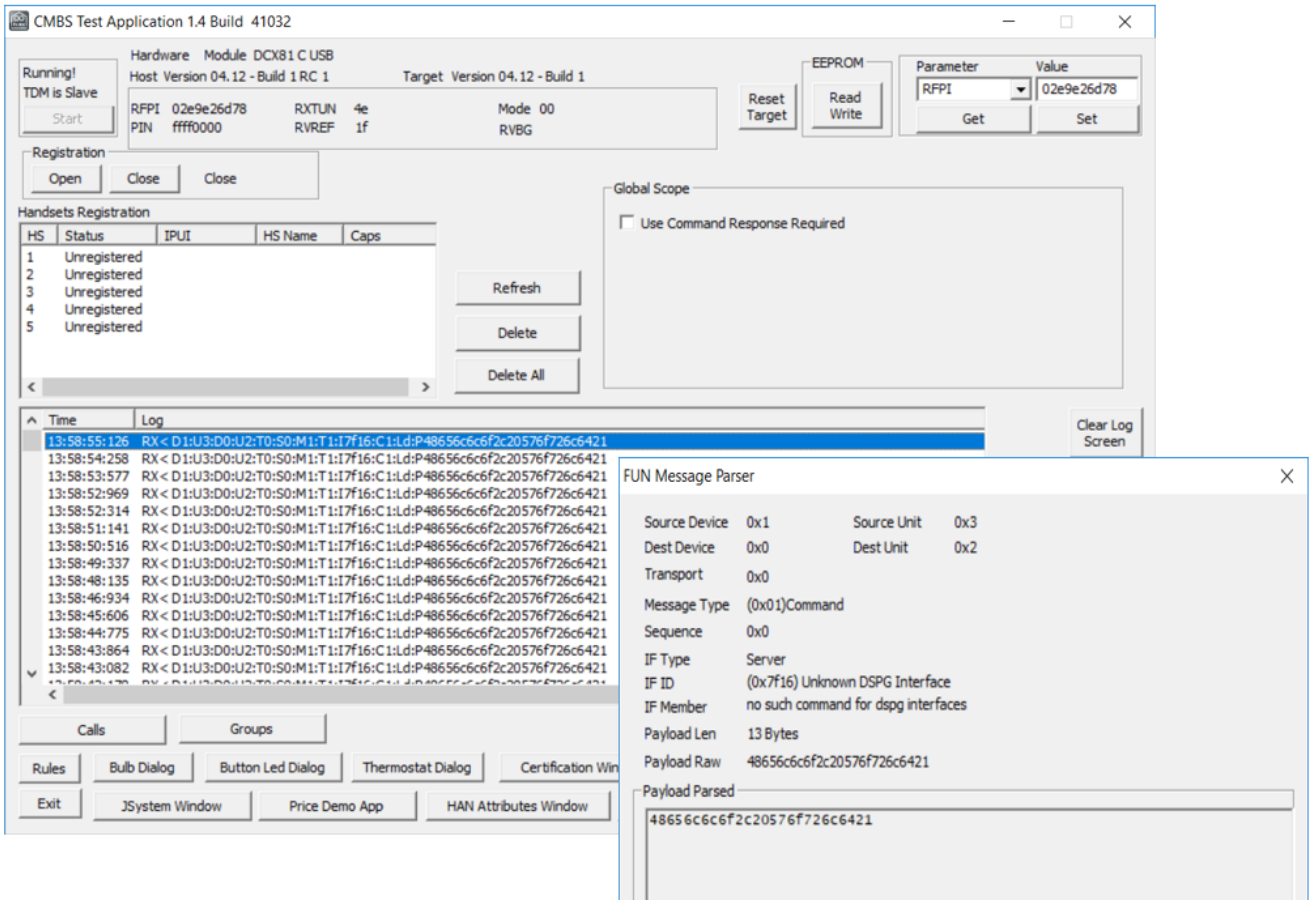
This example shows how to command the EU-DB to send a raw data FUN message to the hub.

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

- Load the Raw Data example code into Eclipse and run the code as shown in chapter 9.1.
- Pressing the blue button on ST-Nucleo triggers sending the data.

The simple implementation of the example blocks for 500ms when toggling the Nucleo LED to indicate success so allow time for this if repeatedly pressing the blue button.

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.



9.4.4. Voice Call

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

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

- Load the Voice Call example code into Eclipse and run the code as shown in chapter 9.1.
- 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.

9.5. Creating your own project

When creating your own project, start by creating a project for your hardware platform, then copy/integrate the CmndLib into your project structure. You can find the CmndLib at `device/uleasy/CmndLib` in the ULE Starter Kit.

At this stage the project will not build. This is because it is missing some glue code, so 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, ... );
```

These functions have already been implemented for the STM32L476RG-Nucleo, so if you are using this 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 10. 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 this:

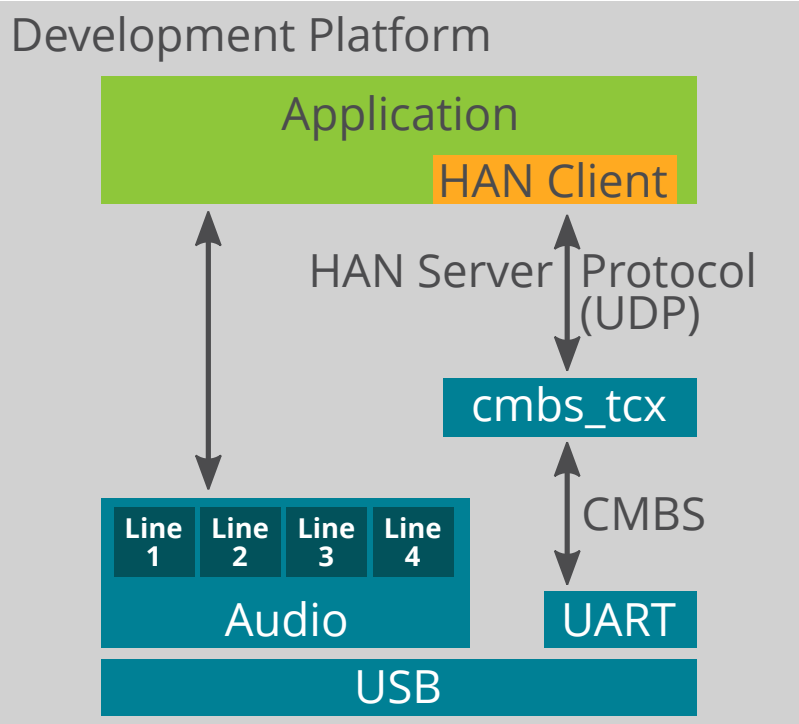


Figure 15. Base Development Stack

The blue boxes are supported and delivered by either the operating system vendor or DSP Group. The HAN Server Protocol is specified in the CMBS HAN SERVER PROTOCOL document [2] (/doc/CMBS HAN SERVER PROTOCOL.pdf). 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.

10.1. Running the demo ULE hub

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

10.1.1. 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:

```
c:>cd \path\to\base\ule-hub
...\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
```

10.1.2. 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>cmbs_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

This package provides several Linux binaries, each for a different architecture. Substitute mentions of `cmbs_tcx` below with the actual binary name:

`cmbs_tcx.linux-amd64`	Linux, 64 bit
`cmbs_tcx.linux-arm`	Linux, ARMv6

Open a terminal, then:

```
$ cd /path/to/base/tools
$ ./cmbs_tcx -usb -com 0 -han ①
DSP Group Demo Software Version:0412 Build:517
...
```

① this will look for `/dev/ttyACM0`, use `-com 1` for `/dev/ttyACM1`

10.1.3. 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 on a specific command.

Messages from the HAN Client to the demo application are prefixed by a time stamp.

10.1.4. 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 been closed automatically.

10.1.5. 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'
```

10.1.6. 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
```

10.2. 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.

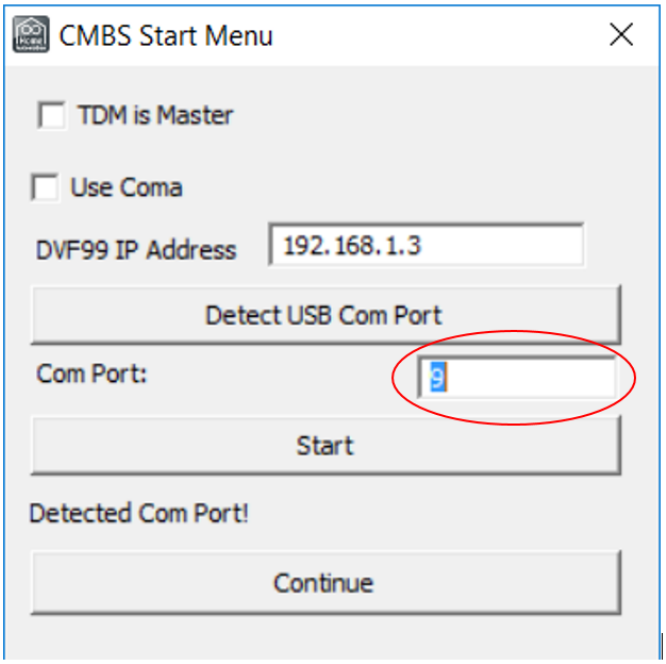


Figure 16. Starting the DSPG Test Application

A.2. Open registration

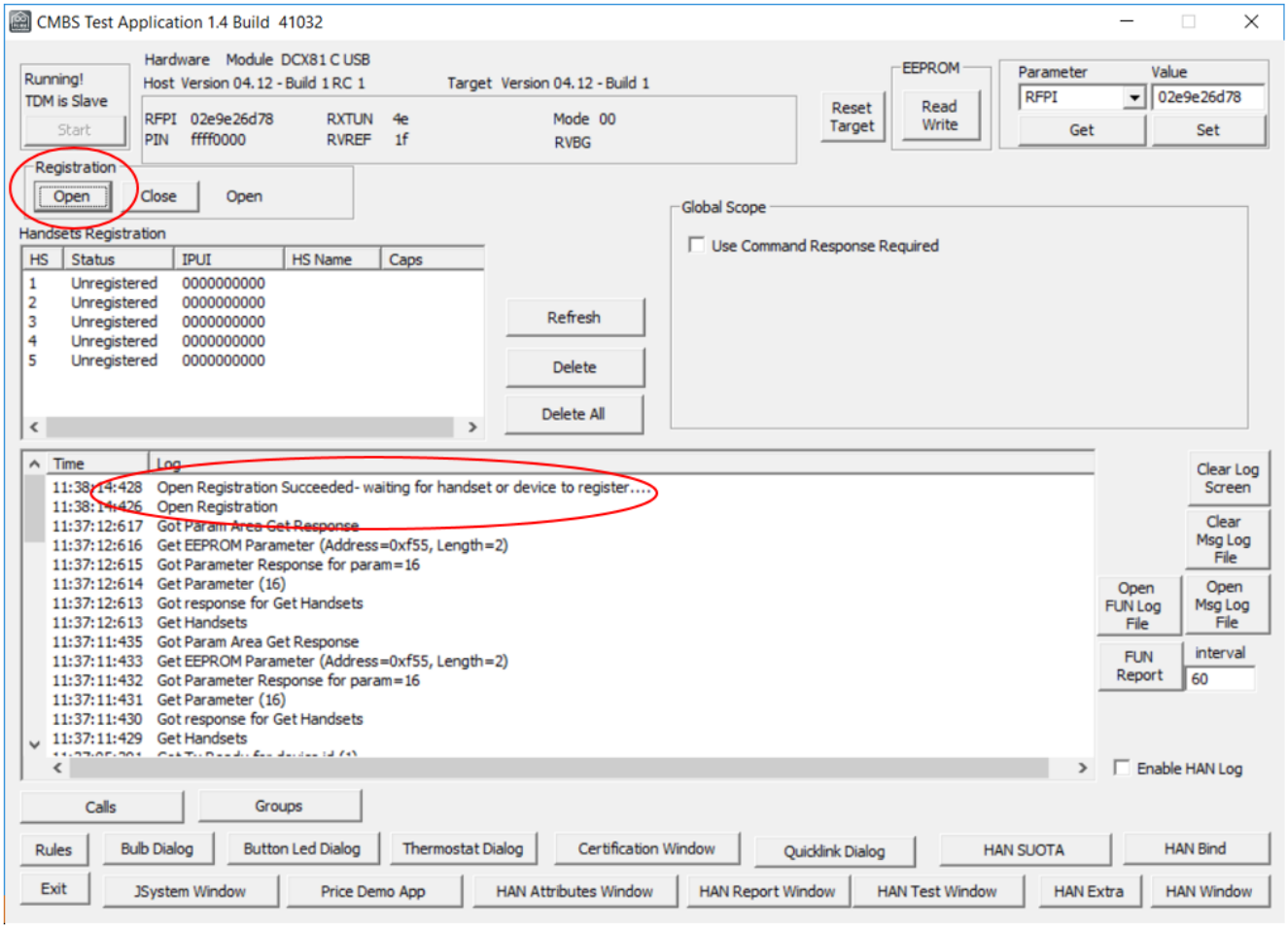


Figure 17. Open base for registration

A.3. Log information during registration

The Error indication can be ignored.

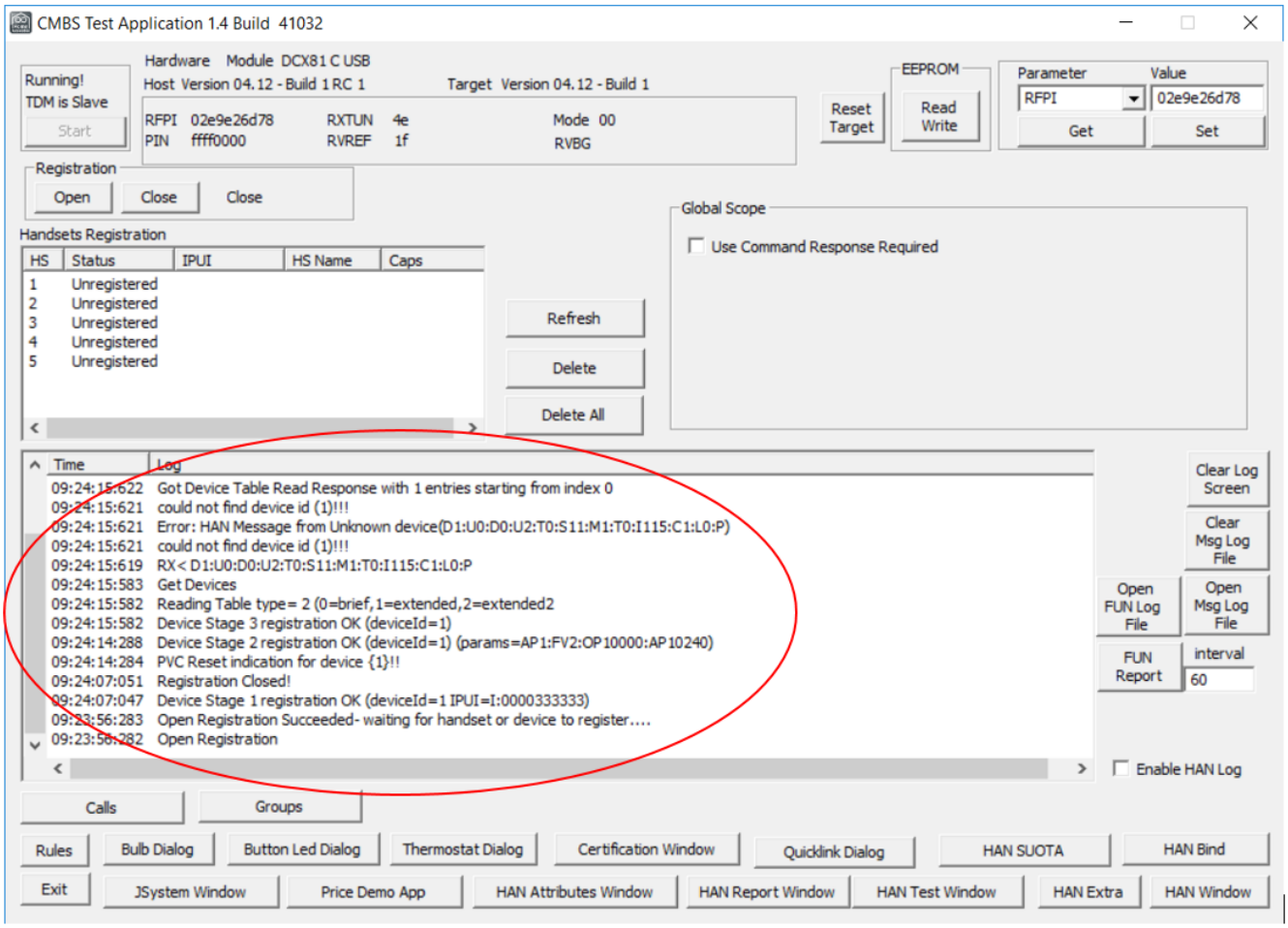


Figure 18. Log information during successful registration

A.4. Event counters

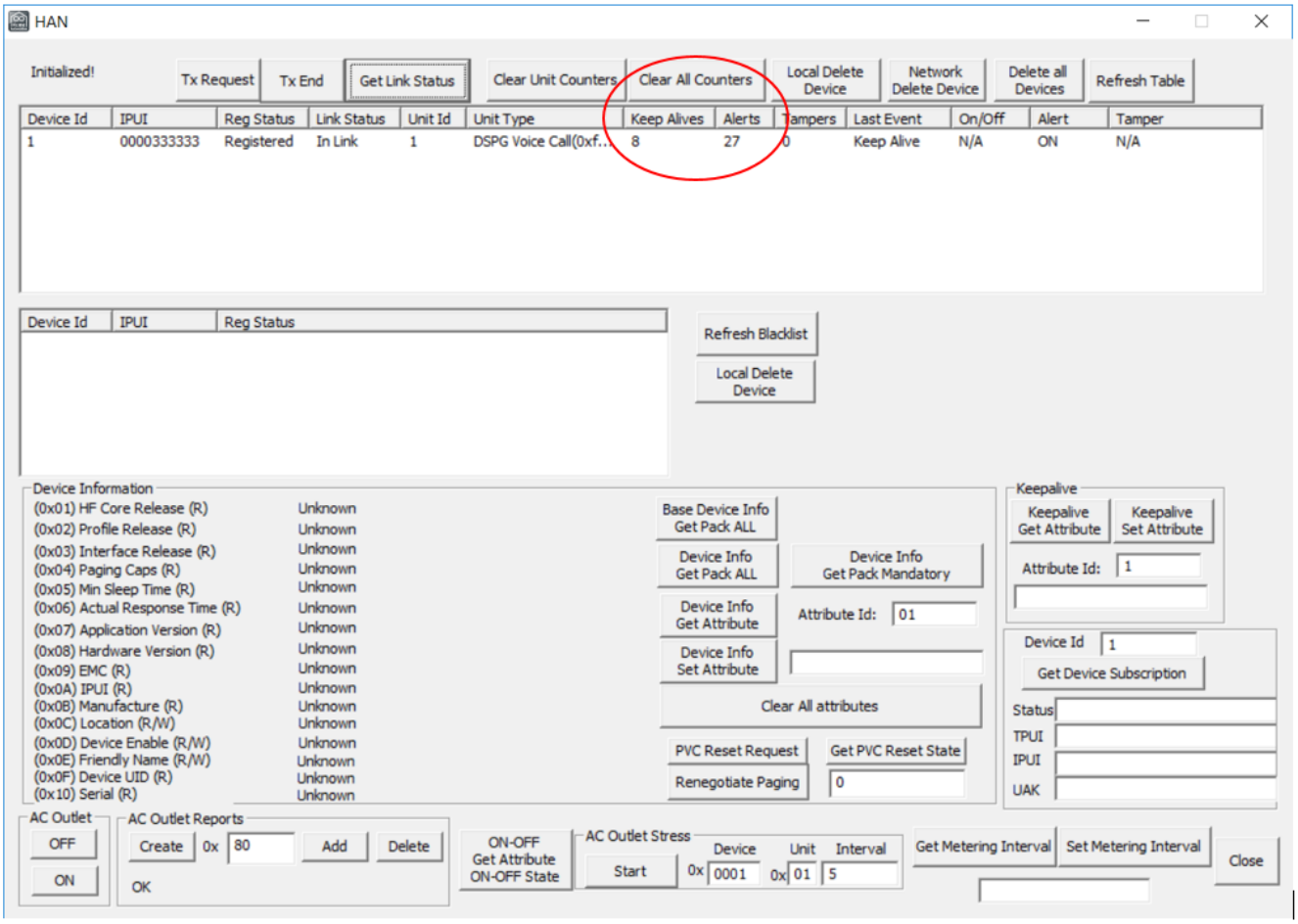


Figure 19. Event Counters (Keep Alive and Alert) in HAN window

A.5. Voice call handling

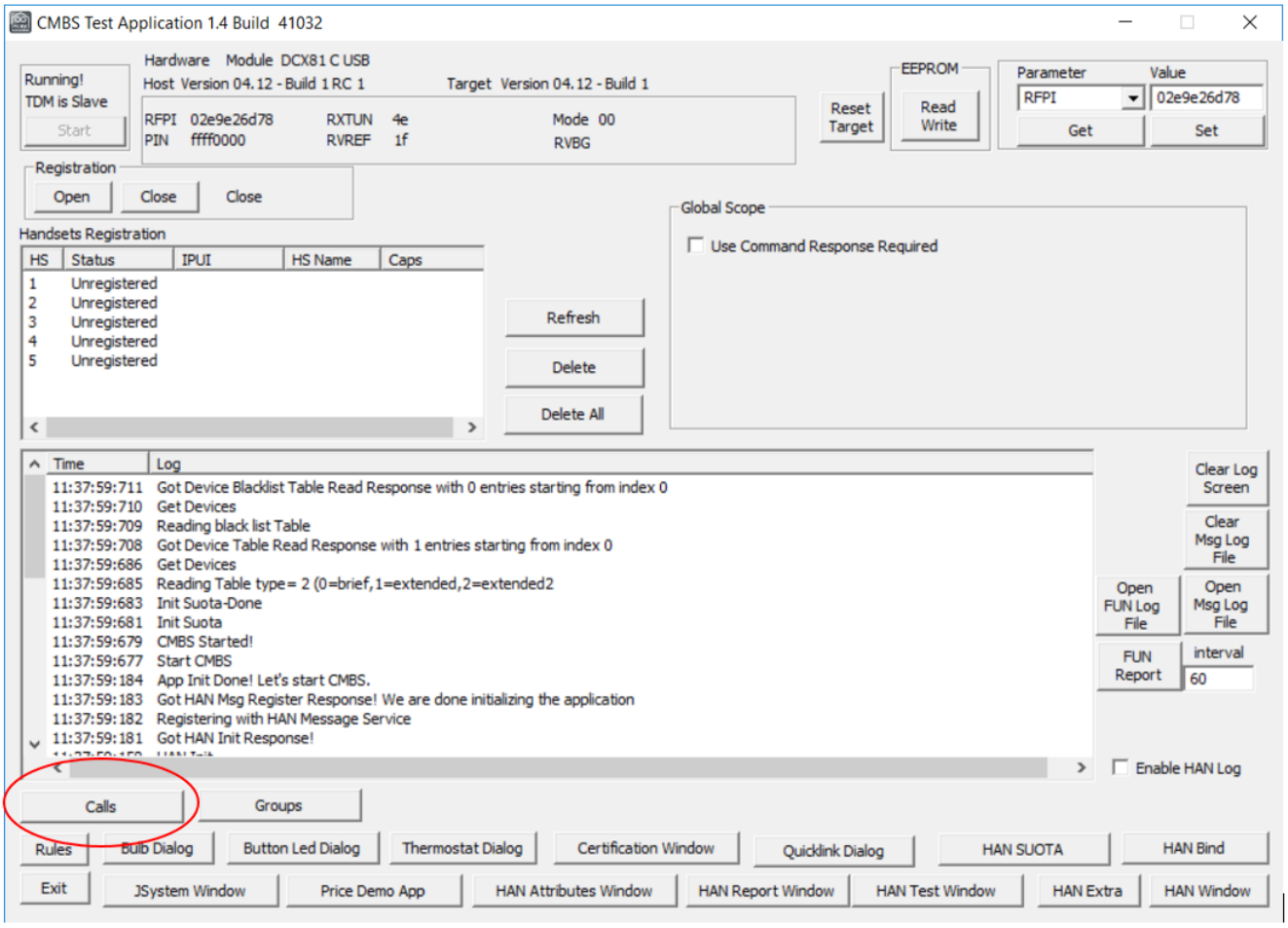


Figure 20. Selecting Calls window

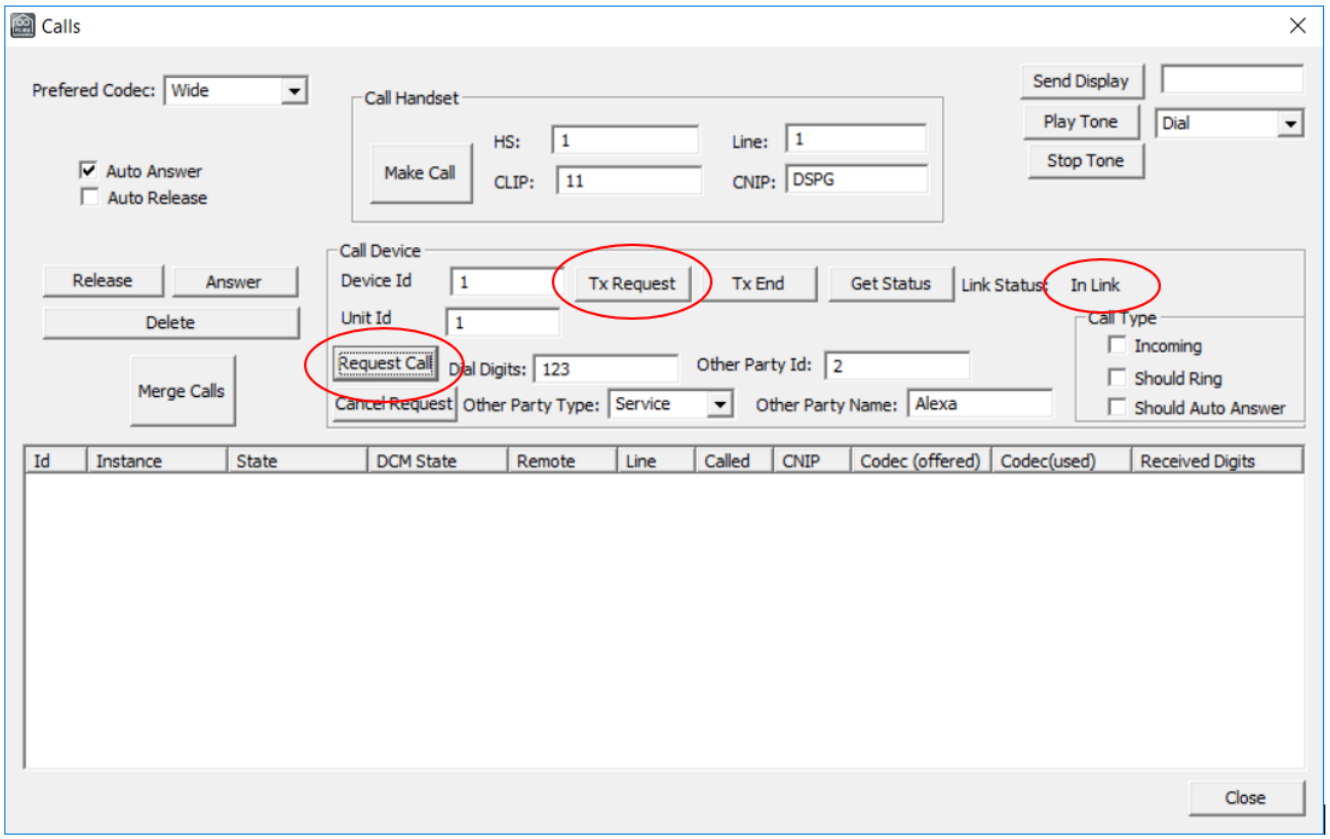


Figure 21. Calls window setup call

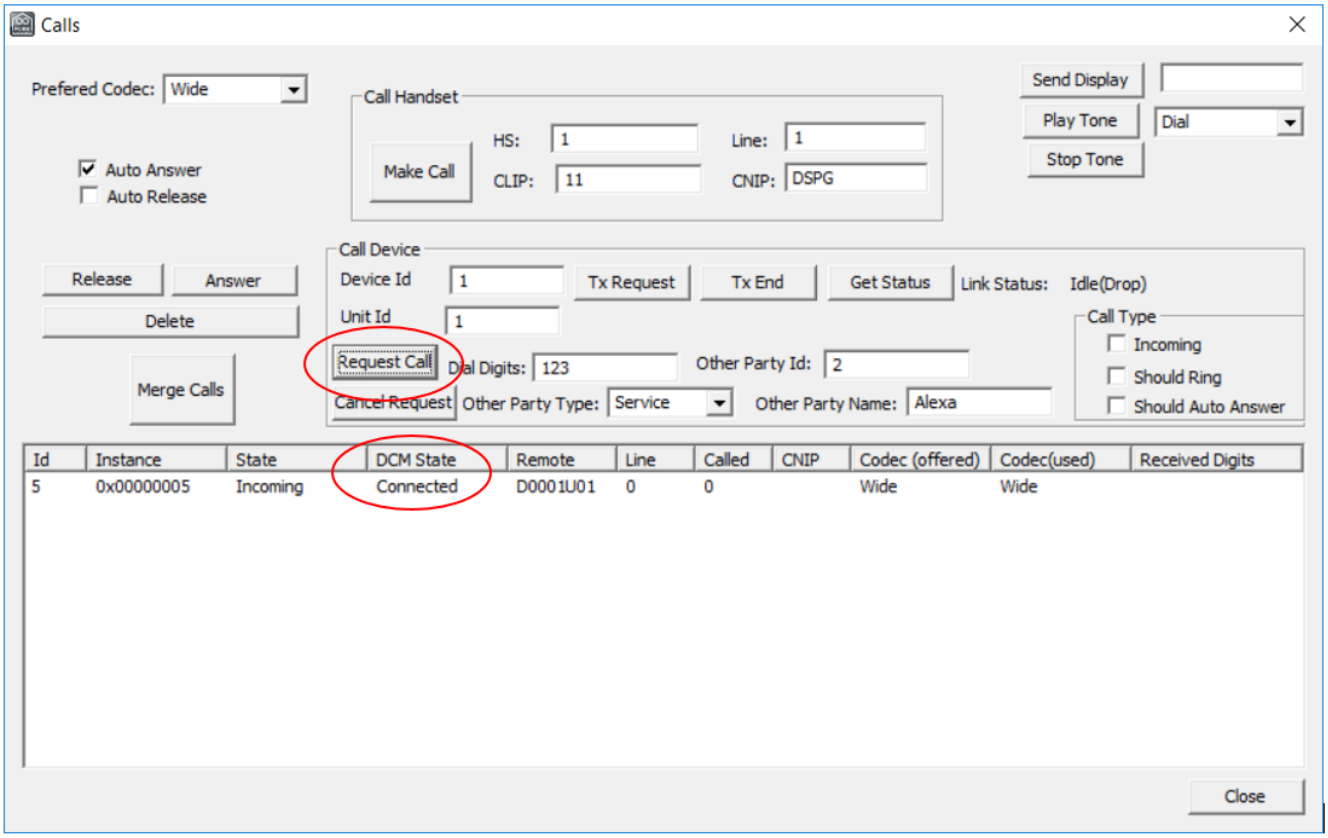


Figure 22. Calls window call connected

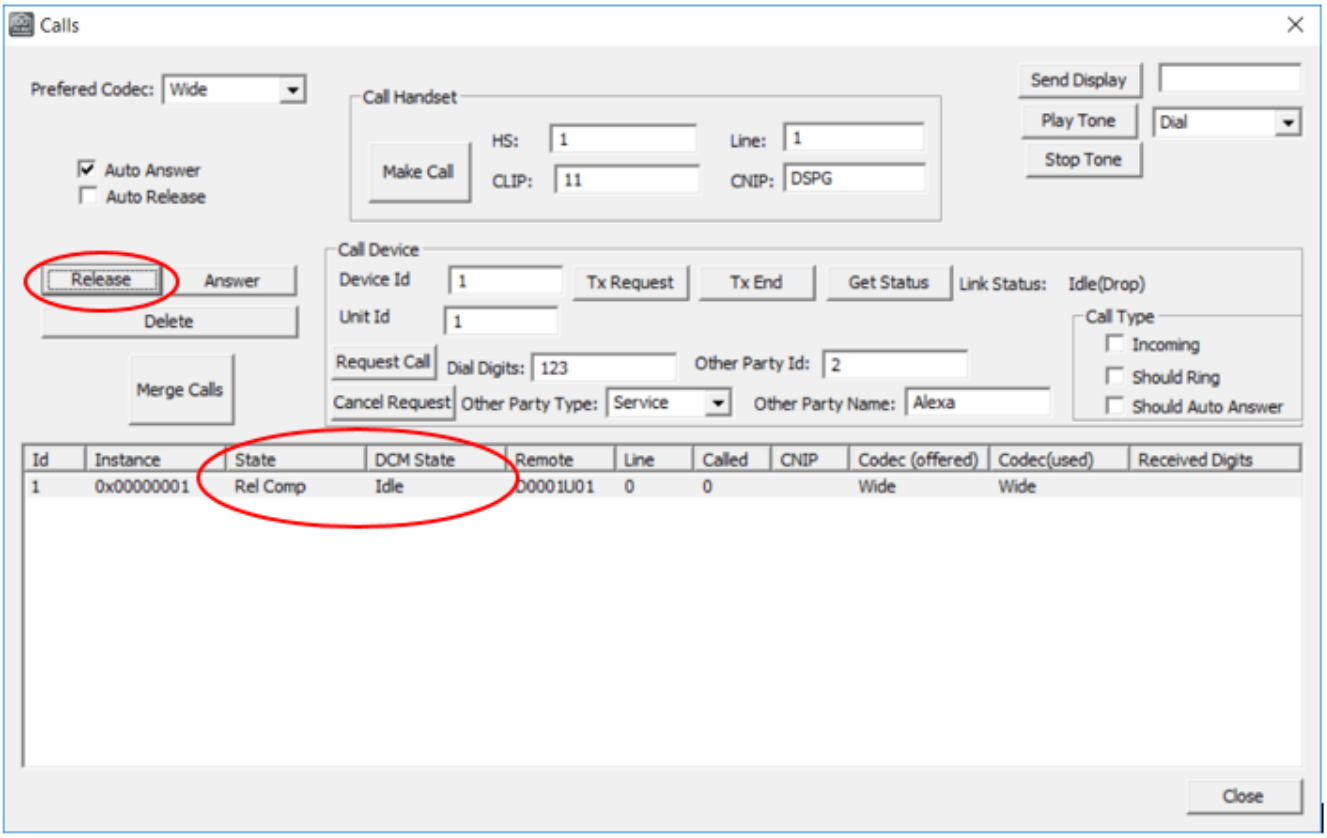


Figure 23. Calls window call release

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'.

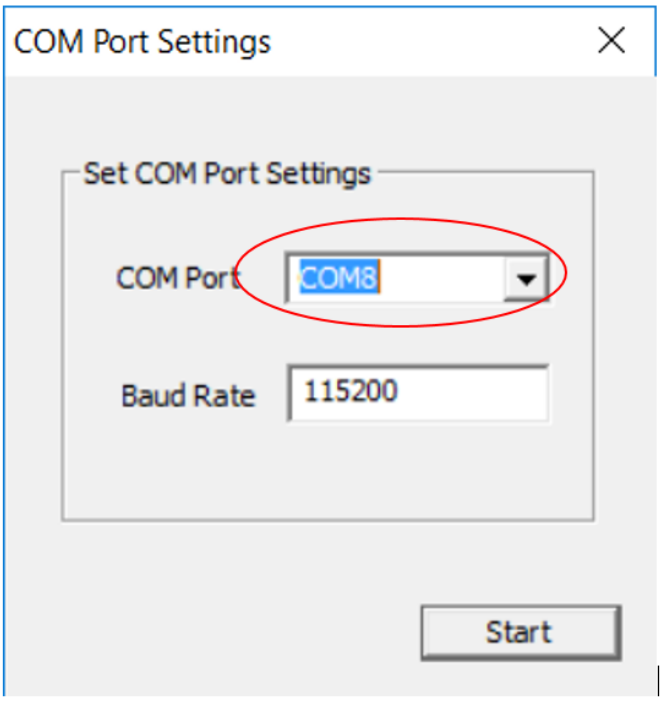


Figure 24. Starting the CMND Simulator

B.2. Sending CMND messages

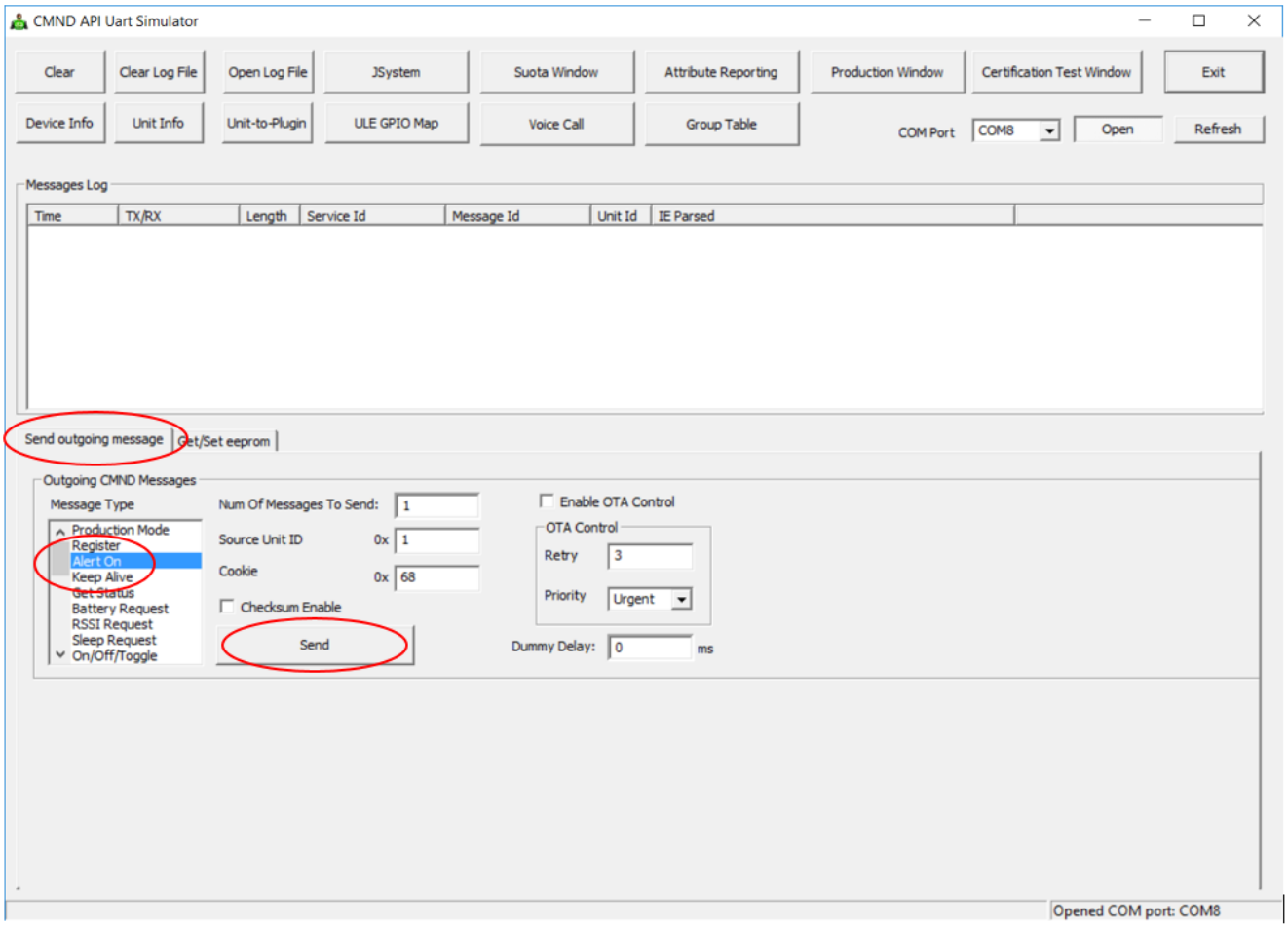


Figure 25. Sending CMND messages

B.3. Voice call handling

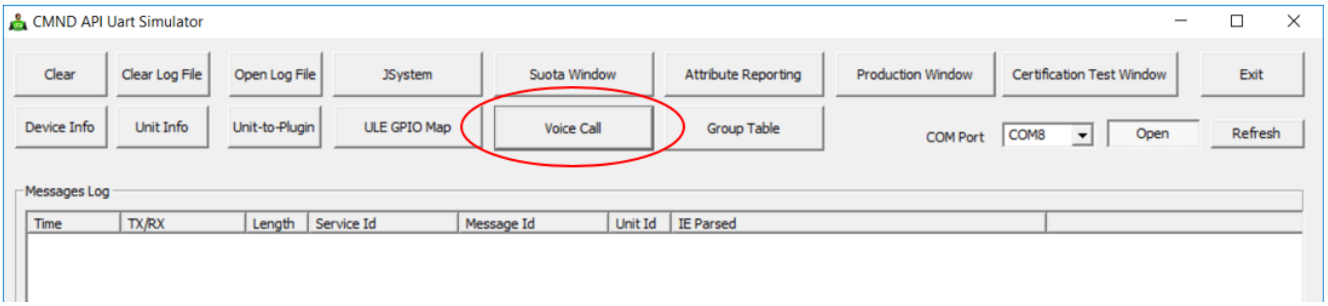


Figure 26. Selecting Voice Call window

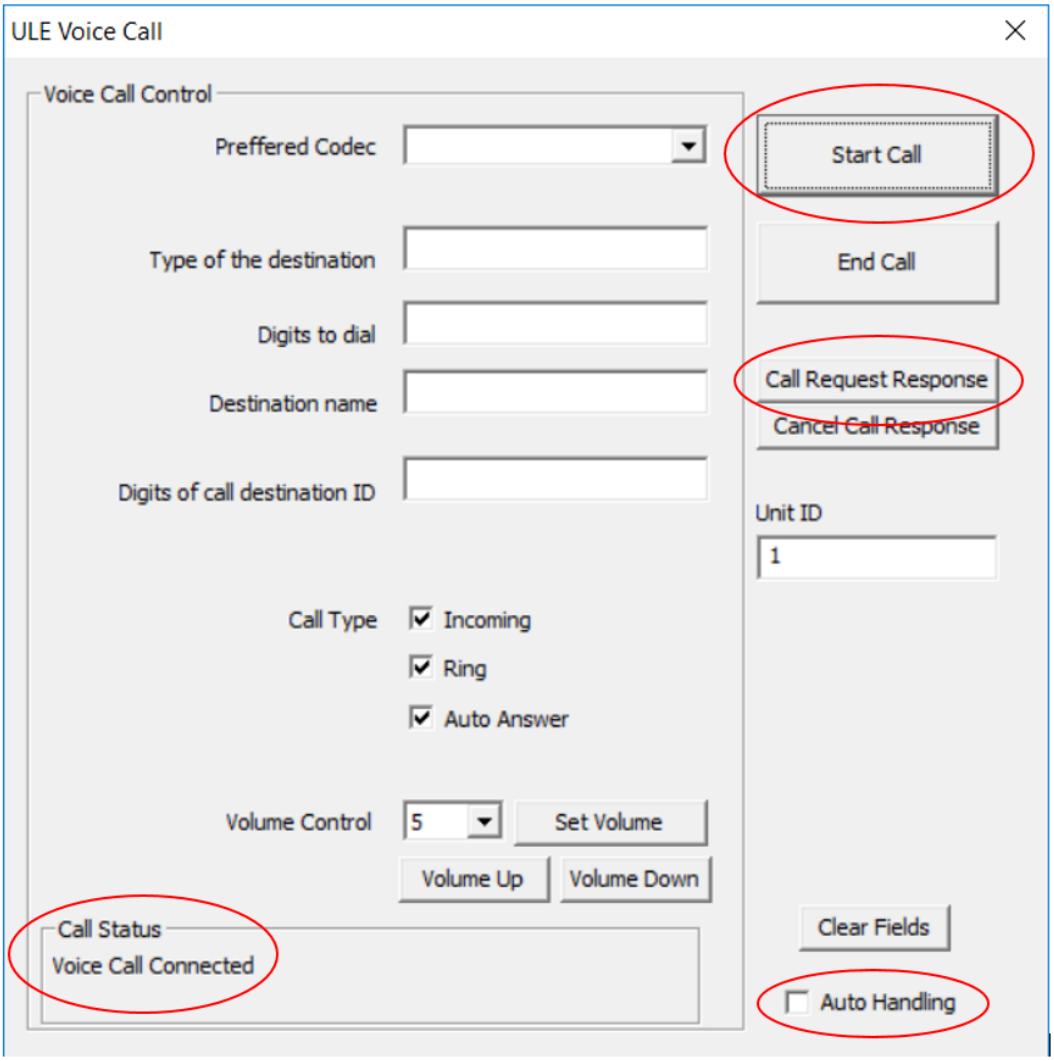


Figure 27. Voice Call window

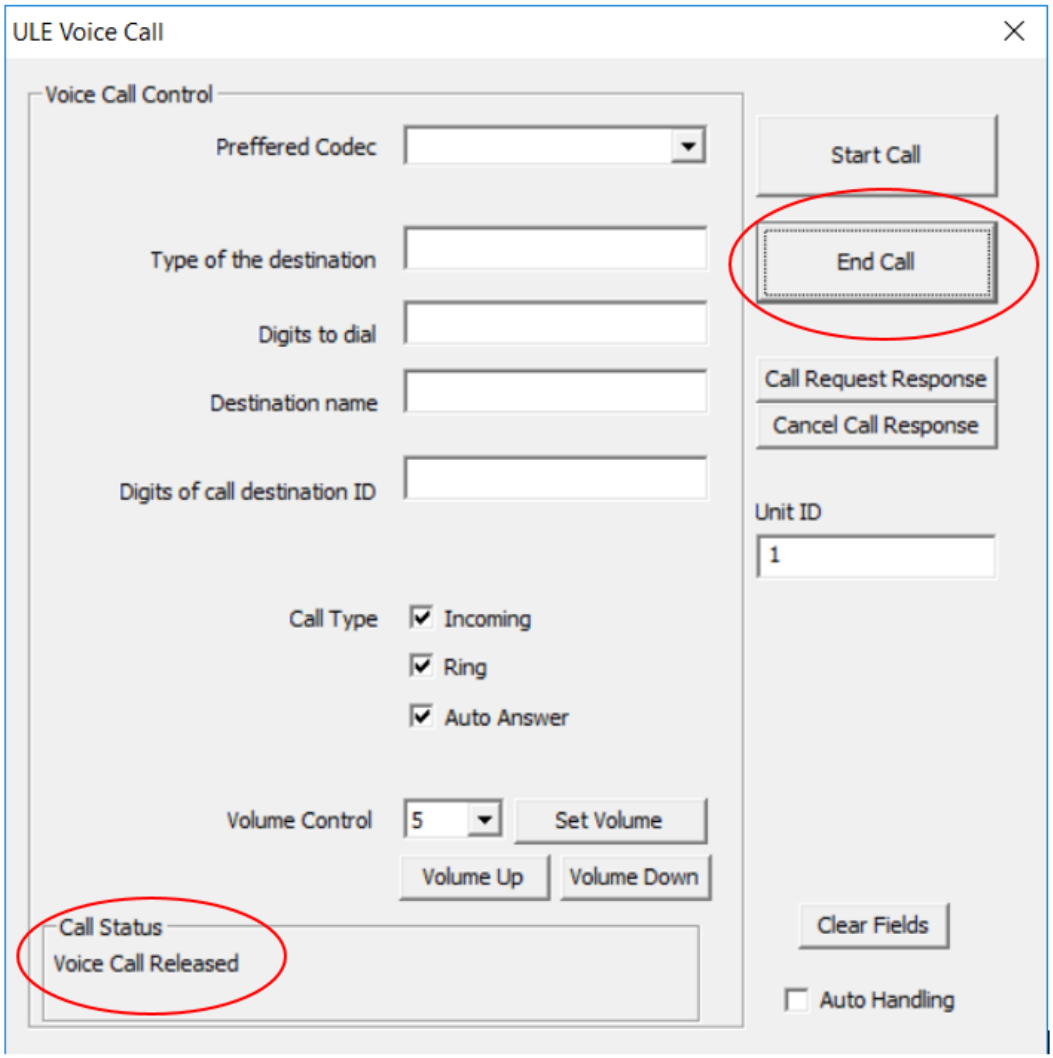


Figure 28. Voice Call window release call

B.4. Changing an EEPROM byte

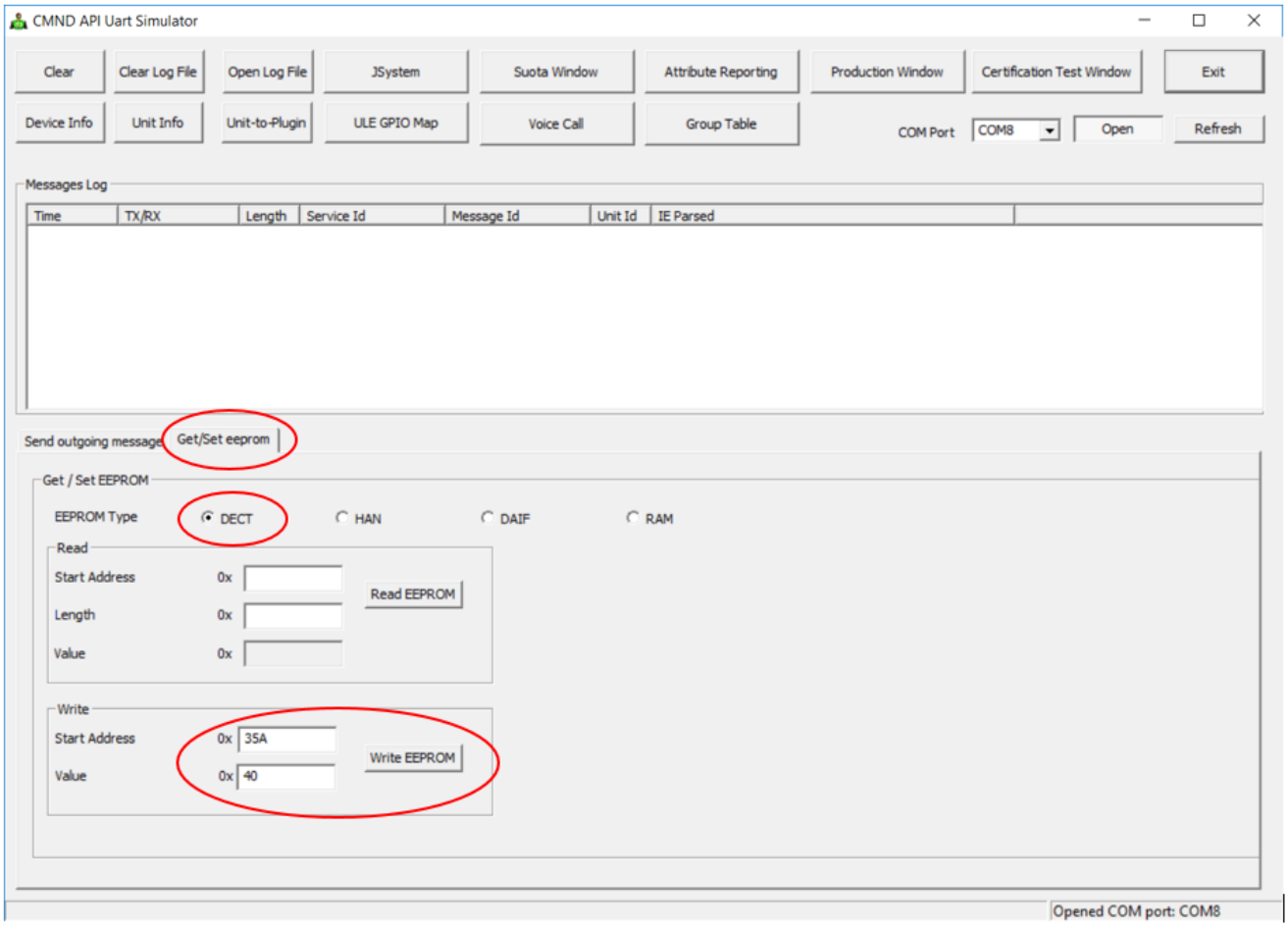


Figure 29. Changing an EEPROM byte

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'

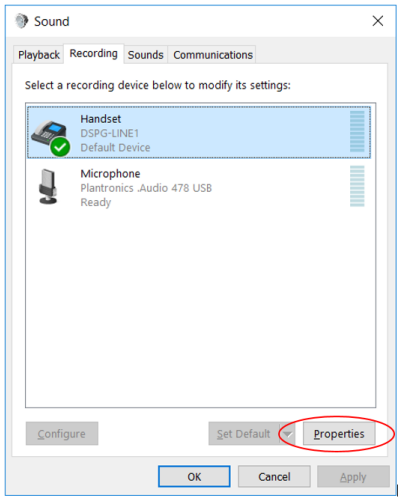


Figure 30. Selecting DSPG dongle as recording device

- Choose the 'Listen' tab and tick the 'Listen to this device' box
- Select your PC audio destination as the 'Playback through this device'

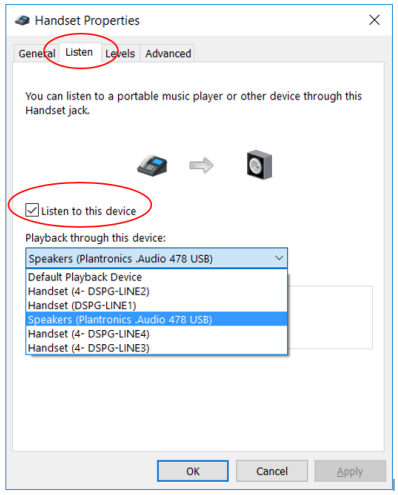


Figure 31. Selecting your PC audio destination as playback device

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

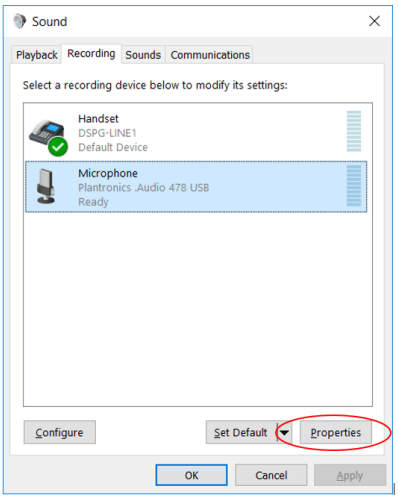


Figure 32. Selecting your PC audio source as recording device

- Choose the 'Listen' tab and tick the 'Listen to this device' box
- Select 'Handset(DSPG-LINE1)' as the 'Playback through this device'

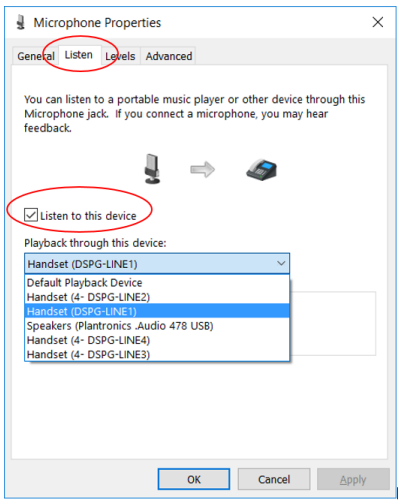


Figure 33. Selecting the DSPG dongle as playback device

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

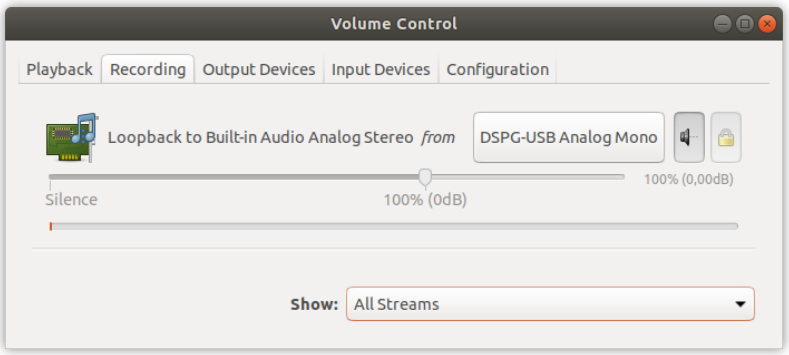
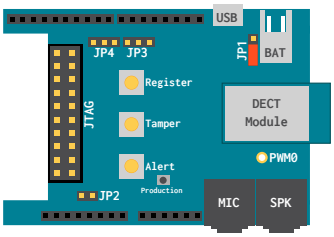


Figure 34. Volume control dialog box

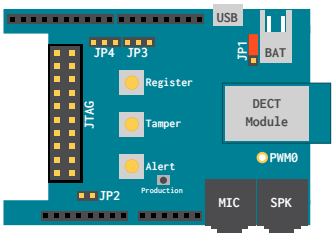
Appendix E: DU-EB jumper settings

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

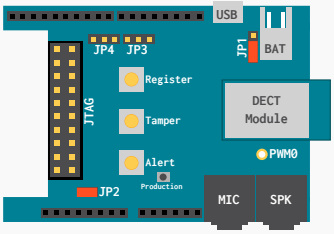
Power supply configurations are controlled via jumpers JP1 and JP2.



Power via batteries or power supply



Power via USB



Power from host board

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

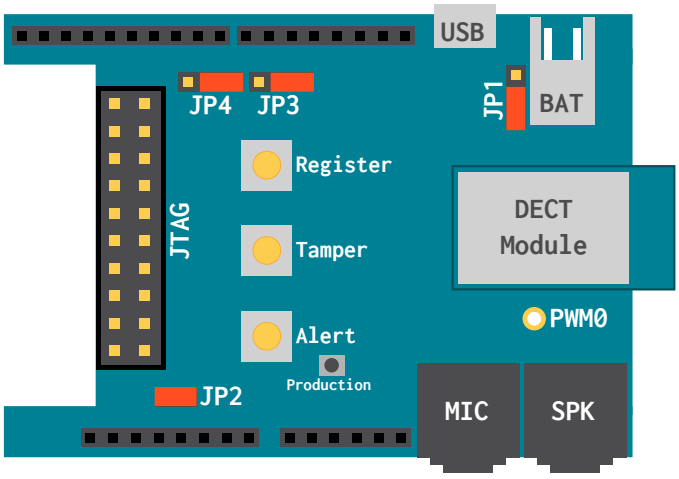


Figure 35. DU-EB jumper settings for ST-Nucleo

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

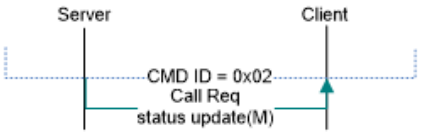


Figure 20 – DSPG ULE Voice Call Interface: Call Request Status Update

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

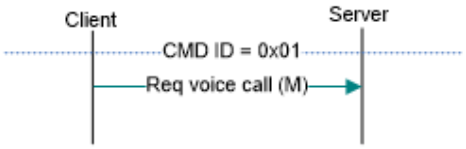


Figure 20 – DSPG ULE Voice Call Interface: Request Call Setup

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

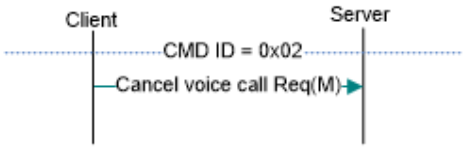


Figure 20 – DSPG ULE Voice Call Interface: Cancel Call Request

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"

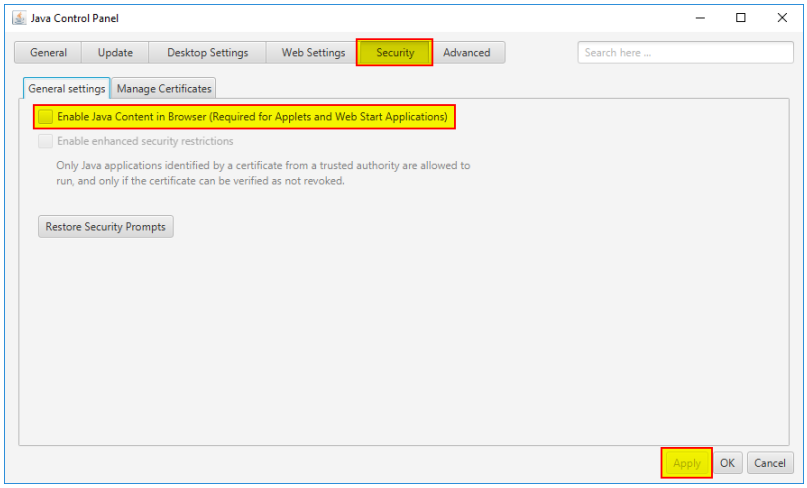


Figure 36. Java Control Panel

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.