

# Getting Started With the ULE Starter Kit

DSP Group Inc.

Revision 0.7

# **Table of Contents**

1         cronyms and Abbreviations       1         ad Bibliography.       1	1.1. Purpose.       1         1.2. Definitions, Acronyms and Abbreviations       1         1.3. References and Bibliography.       1         2. Known Issues.       3         3. DECT-ULE System Overview       4         3.1. DECT-ULE Characteristics.       5         3.2. Identities.       5         3.3. Registration       5         3.4. Security.       6         3.5. ULE Paging       6         3.6. ULE Paging       6         3.6. ULE Paging       6         3.6. ULE Paging       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4.1. DECT-ULE Expansion Board (DU-EB)       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3.2. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15	1.	Introduction		1
tronyms and Abbreviations       1         ad Bibliography.       1	1.2. Definitions, Acronyms and Abbreviations       1         1.3. References and Bibliography.       1         2. Known Issues.       3         3. DECT-ULE System Overview       4         3.1. DECT-ULE Characteristics.       5         3.2. Identities.       5         3.3. Registration       5         3.4. Security.       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register.       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		1.1. Purpose		1
and Bibliography.       1         3       Overview         4       4         aracteristics       5         5       5         6       6         1       6         1       6         1       6         1       6         1       6         1       6         1       7         2       7         2       7         2       7         2       8         10       7         10       10         11       10         12       14         13       19         13       19	1.3. References and Bibliography.       1         2. Known Issues.       3         3. DECT-ULE System Overview       4         3.1. DECT-ULE Characteristics.       5         3.2. Identities.       5         3.3. Registration       5         3.4. Security.       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		1.2. Definitions, Acronyms and Abbreviations		1
3         Overview       4         aracteristics       5         5       5         6       6         isfer       6         6       6         9       6         9       6         9       6         9       6         9       6         9       6         9       6         9       6         9       7         9       7         9       7         9       10         10       10         11       10         11       11         12       14       12         14       13       19         15       15       15	2. Known Issues       3         3. DECT-ULE System Overview       4         3.1. DECT-ULE Characteristics       5         3.2. Identities       5         3.3. Registration       5         3.4. Security       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		1.3. References and Bibliography.		1
Overview       4         aracteristics       5         5       5         6       6         isfer       6         6       6         7       6         6       7         ency Band       7         Kit Overview       8         ansion Board (DU-EB)       8         10       10         vironment       10         10       11         call       11         11       11         API       12         th a scope       13         10       15	3. DECT-ULE System Overview       4         3.1. DECT-ULE Characteristics       5         3.2. Identities       5         3.3. Registration       5         3.4. Security       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15	2.	Known Issues.		3
aracteristics	3.1. DECT-ULE Characteristics.       5         3.2. Identities.       5         3.3. Registration       5         3.4. Security.       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4.1. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register.       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15	3.	DECT-ULE System Overview		4
5 5 6 6 6 6 7 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1	3.2. Identities.       5         3.3. Registration       5         3.4. Security.       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive.       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         3.10. Radio Frequency Band       7         4.1. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register.       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.1. DECT-ULE Characteristics		5
5 6 6 6 6 6 7 6 7 6 7 6 7 6 7 6 7 6 7 6 7 7 8 8 ansion Board (DU-EB) 8 7 Kit Overview 8 ansion Board (DU-EB) 10 10 10 11 11 11 2 API 11 12 th a scope 13 19 15	3.3. Registration       5         3.4. Security       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.2. Identities		5
6 isfer	3.4. Security       6         3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.3. Registration		5
isfer       6         isfer       6         6       6         6       7         ency Band       7         Kit Overview       8         ansion Board (DU-EB)       8         10       10         vironment       10         10       11         call       11         D API       12         th a scope       13         10       15	3.5. ULE Paging       6         3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.4. Security		6
isfer       6         6       6         7       6         7       7         ency Band       7         Kit Overview       8         ansion Board (DU-EB)       8 <i>v</i> ironment       10         10       10 <i>v</i> ironment       10         11       11         call       11         12       14         13       19	3.6. ULE Data Transfer       6         3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.5. ULE Paging		6
6 ency Band Xit Overview ansion Board (DU-EB) <i>v</i> ironment 10 <i>v</i> ironment 10 11 11 2 API API 12 h a scope 13 19 15	3.7. Keep Alive       6         3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.6. ULE Data Transfer		6
6 Pency Band 7 Kit Overview 8 ansion Board (DU-EB) 8 10 vironment 10 10 11 call 11 0 API 12 th a scope 13 19	3.8. Hibernation       6         3.9. HAN-FUN       7         3.10. Radio Frequency Band       7         4. DECT-ULE Starter Kit Overview       8         4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		3.7. Keep Alive		6
7ency BandKit Overviewansion Board (DU-EB)810vironment1011call0 API12:h a scope1319	3.9. HAN-FUN73.10. Radio Frequency Band74. DECT-ULE Starter Kit Overview84.1. DECT-ULE Expansion Board (DU-EB)85. Hands On105.1. Set up the environment105.2. Register105.3. Send an alert115.4. Make a voice call115.5. Use the CMND API125.6. RF sensing with a scope135. Development Setup15		3.8. Hibernation		6
ency Band	3.10. Radio Frequency Band74. DECT-ULE Starter Kit Overview84.1. DECT-ULE Expansion Board (DU-EB)85. Hands On105.1. Set up the environment105.2. Register105.3. Send an alert115.4. Make a voice call115.5. Use the CMND API125.6. RF sensing with a scope135. Development Setup15		3.9. HAN-FUN		7
Kit Overview       8         Jansion Board (DU-EB)       8         10       10         vironment       10         10       10         11       10         12       11         12       13         10       15	4. DECT-ULE Starter Kit Overview84.1. DECT-ULE Expansion Board (DU-EB)85. Hands On105.1. Set up the environment105.2. Register105.3. Send an alert115.4. Make a voice call115.5. Use the CMND API125.6. RF sensing with a scope135. Development Setup15		3.10. Radio Frequency Band		7
ansion Board (DU-EB)	4.1. DECT-ULE Expansion Board (DU-EB)       8         5. Hands On       10         5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       10         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15	4.	DECT-ULE Starter Kit Overview		8
10         vironment       10         10       10         10       10         11       11         call       11         0 API       12         :h a scope       13         ip       15	5. Hands On       10         5.1. Set up the environment       10         5.2. Register.       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		4.1. DECT-ULE Expansion Board (DU-EB)		8
vironment	5.1. Set up the environment       10         5.2. Register       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15	5.	Hands On	. 1	0
10 11 call	5.2. Register.       10         5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		5.1. Set up the environment	. 1	0
11 call	5.3. Send an alert       11         5.4. Make a voice call       11         5.5. Use the CMND API       12         5.6. RF sensing with a scope       13         5. Development Setup       15		5.2. Register	. 1	0
call	5.4. Make a voice call115.5. Use the CMND API125.6. RF sensing with a scope135. Development Setup15		5.3. Send an alert	. 1	11
D API	5.5. Use the CMND API125.6. RF sensing with a scope135. Development Setup15		5.4. Make a voice call	. 1	11
h a scope	5.6. RF sensing with a scope135. Development Setup15		5.5. Use the CMND API	. 1	12
ıp	5. Development Setup		5.6. RF sensing with a scope	. 1	13
		6.	Development Setup	. 1	15
ench Installation	6.1. System Workbench Installation		6.1. System Workbench Installation	. 1	15
ation			6.2. Python Installation	. 1	16
Installation 17	6.2. Python Installation		6.3. STLink Driver Installation	. 1	17
	6.2. Python Installation166.3. STLink Driver Installation17	7.	Device Development	. 1	8
ent	6.2. Python Installation		7.1. Import, Build and Run Examples	. 1	8
ent	6.2. Python Installation       16         6.3. STLink Driver Installation       17         7. Device Development       18         7.1. Import, Build and Run Examples       18		7.2. Project Structure	. 2	21
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure21		7.3. Example Behavior	. 2	21
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior21		7.4. Included examples	. 2	22
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples22		7.5. Creating your own project	. 2	24
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project24	8.	Base Development	. 2	25
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development25		8.1. Running the demo ule-hub	. 2	25
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub25		8.2. Using the han_client module.	. 2	28
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28	A	opendix A: DSPG Test Application	. 2	29
ent18and Run Examples18ure21nvior21nples22own project24t25emo ule-hub25_client module28est Application29	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29		A.1. Starting the DSPG Test Application	. 2	<u>29</u>
Instantion18and Run Examples18ure21ivior21nples22own project24t25emo ule-hub25_client module28est Application29SPG Test Application29	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application29		A.2. Open registration	. 3	30
InstantionInstant InstantionInstant Run ExamplesInstant Run Examples <td>6.2. Pytnon Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application29A.2. Open registration30</td> <td></td> <td>A.3. Log information during registration</td> <td>. 3</td> <td>31</td>	6.2. Pytnon Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application29A.2. Open registration30		A.3. Log information during registration	. 3	31
Image: Sector of the sector	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30A.3. Log information during registration31		A.4. Event counters	. 3	32
Image: Sent sent sent sent sent sent sent sent s	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30A.3. Log information during registration31A.4. Event counters32		A.5. Voice call handling	. 3	33
ench Installation	6.1. System Workbench Installation	7.	<ul> <li>6.1. System Workbench Installation</li> <li>6.2. Python Installation</li> <li>6.3. STLink Driver Installation</li> <li>Device Development</li> <li>7.1. Import, Build and Run Examples</li> <li>7.2. Project Structure</li> <li>7.3. Example Behavior</li> <li>7.4. Included examples</li> </ul>		1 1 1 1 2 2
ation 16			6.2 Python Installation	ر . 1	16
ation			6.2. Python Installation	. 1	6
Installation 17	6 / PV/fnon installation		6.3 STLink Driver Installation	· · 1	17
Installation 17	6.2. Python Installation		6.3. STLink Driver Installation	. 1	17
	6.2. Python Installation166.3. STLink Driver Installation17	7.	Device Development	. 1	18
nt	6.2. Python Installation 16 6.3. STLink Driver Installation 17 7. Device Development 18		7.1. Import. Build and Run Examples	. 1	18
ent	6.2. Python Installation       16         6.3. STLink Driver Installation       17         7. Device Development       18         7.1 Import Build and Run Examples       18		7.2 Project Structure		21
ent	6.2. Python Installation       16         6.3. STLink Driver Installation       17         7. Device Development       18         7.1. Import, Build and Run Examples       18         7.2. Project Structure       21		7.2. Project Structure	. 4	21
and Run Examples 18 ure 21	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure21		7.3. Example Behavior	. 2	21
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior21		7.4. Included examples	. 2	22
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples22		7.5. Creating your own project	. 2	24
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project24	8.	Base Development	. 2	25
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243 Base Development25	0.	8.1. Running the demo ule-hub		25
ent	6.2. Python Installation       16         6.3. STLink Driver Installation       17         7. Device Development       18         7.1. Import, Build and Run Examples       18         7.2. Project Structure       21         7.3. Example Behavior       21         7.4. Included examples       22         7.5. Creating your own project       24         8. Base Development       25         8.1. Running the demo ule-hub       25		8.2 Using the ban client module	. 2	20
ent	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the ban client module28	۸ı	onendix A: DSPG Test Application		-0 29
ent18and Run Examples18ure21ivior21nples22own project24t25emo ule-hub25_client module28est Application29	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29	.1	A.1. Starting the DSPG Test Application	. 7	29
Instantion18and Run Examples18ure21ivior21nples22own project24t25emo ule-hub25_client module28est Application29SPG Test Application29	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application29		A.2. Open registration	. 3	30
and Run Examples18and Run Examples18ure21avior21aples22own project24t25emo ule-hub25_client module28est Application29SPG Test Application29tion30	6.2. Pytnon Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30		A.3. Log information during registration	. 3	31
InstantionInstant InstantionInstant InstantionInstant InstantionInstant InstantionInstant InstantionIn	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30A.3. Log information during registration31		A.4. Event counters	. 3	32
Image: Section of the section of th	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project243. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30A.3. Log information during registration31A.4. Event counters32		A.5. Voice call handling	. 3	33
and Run Examples18and Run Examples18ure21nvior21nples22own project24t25emo ule-hub25_client module28est Application29SPG Test Application29tion30on during registration31's32dling33	6.2. Python Installation166.3. STLink Driver Installation177. Device Development187.1. Import, Build and Run Examples187.2. Project Structure217.3. Example Behavior217.4. Included examples227.5. Creating your own project248. Base Development258.1. Running the demo ule-hub258.2. Using the han_client module.28Appendix A: DSPG Test Application29A.1. Starting the DSPG Test Application30A.3. Log information during registration31A.4. Event counters32A.5. Voice call handling33				

Appendix B: CMND Simulator	õ
B.1. Starting the CMND Simulator	5
B.2. Sending CMND messages	7
B.3. Voice call handling	3
B.4. Changing an EEPROM byte	)
Appendix C: Windows audio routing	1
C.1. Routing audio from DU-EB microphone to PC	1
C.2. Routing audio from PC to the DU-EB speaker	1
Appendix D: Linux audio routing	3
Appendix E: DU-EB jumper settings	4
E.1. DU-EB jumper setting for power supply 44	4
E.2. DU-EB jumper settings for operation with ST-Nucleo	4
Appendix F: ULE Voice Call Interface	ō
Appendix G: Java Troubleshooting	9
G.1. Error 1603	9
G.2. Error 1607	9



# **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
$C \to N$	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
$N \rightarrow C$	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
  - $\circ\,$  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







# **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"



🖨 Install	_		×
Available Software			
Check the items that you wish to install.		0	-
Work with: ols.com/Eclipse-updates/org.openstm32.system-workbench.update-site-v2 v	<u>A</u> dd	<u>M</u> anage	
type filter text			
Name Version			
> 000 External Tools > 000 OpenSTM32 Tools			
Select All Deselect All 6 items selected			
Details			
			al. La
Show only the latest versions of available software	stalled		
Group items by category What is <u>already installed</u> ?			
Show only software applicable to target environment			
Contact all update sites during install to find required software			
(?) < Back Next >	<u>F</u> inish	Cance	

- 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"

Import Project	s from File System or Archive		– 🗆 X
Import Projects	from File System or Archive		
This wizard analy	zes the content of your folder or archive file to find projects and impo	rt them in the IDE.	
Import source:	C:\Projects\ule-starterkit\device\Examples\STM32L476RG-Nucleo\R	egister\SW4STM32	✓ Directory <u>A</u> rchive
type filter text			Select All
Folder		Import as	Deselect All
SW4STM	32	Eclipse project	Derective
Close newly in Use installed pro Search for nes Detect and co Working sets	nported projects upon completion ect configurators to: ted projects nfigure project natures to working sets		1 of 1 selected Hide already open projects
14/	•		
w <u>o</u> rking sets:			✓ S <u>e</u> lect
			Show other specialized import wizards
?		< <u>B</u> ack <u>N</u> ext >	<u>Einish</u> Cancel

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.

eclipse-workspace - Eclipse IDE Eile Edit Severe Defeater Navinate Company	anda Desirat Dun Window Hele	- 🗆 X
	V No Launch Configurations V on: V ☆ I □ V	💿 🖲 🕶 😽 🕶 📾
📸 ▼ 🚳 ▼ 💽 ▼ 🮯 ▼ 🕸 ▼ 🔿 ▼ 🤮	בי 💁 ד 😕 😂 🖋 ד 🗟 🗉 ח בער בי בער בי בער בער בער בער בי בער בער בער בי ב	Quick Access 🔡 📴
🍋 Project 🙁 😫 Connect 📟 🗖		🗄 O 🛛 🗏 T 🦄 🗖 🗖
E S Register [ule_staterkit marter]		an outline is not available
> 27 > Register [ule-statterkit master]		An outline is not available.
	🖹 Problems 🧔 Tasks 📮 Console 🛛 🔲 Properties 🕹 🗘 😭 🚮 🚍	🗟 📑 📮 🕶 📑 🗖
	CDT Build Console [Register] 39528 128 2304 41960 a3e8 Register.elf	^
	18:02:05 Build Finished. 0 errors, 0 warnings. (took 29s.748ms)	
	<	~
·		

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.

🖨 eclipse-workspace - Eclipse IDE			- 🗆 X
File Edit Source Refactor Navigate S	earch Project Run Window Help		
<ul><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li><li></li></ul>	V No Launch Configurations	on: 🗸 🏟 🖬 🕇	• 🗟 🔞 • 🔦 • 📸
🖆 • 🚳 • 🖻 • 🞯 • 🚸 • O • 9	No Launch Configurations	▣◣▯◙▾▨▾やや▾⇔▾	Quick Access 🔡 📴
🔁 Project 🙁 😫 Connect 😑 🗖	New Launch Configuration	•	
E SA   Se ♥			An outline is not available.
	🖹 Problems 🧔 Tasks 📃 Console 🔀 🔲 Pr	operties 🕹 🗘 🔝 📓	= 🛼 🛃 📮 🕶 🗖 🖛 🗖
	CDT Build Console [Register]		
	39528 128 2304 41960 a3e	8 Kegister.eit	^
	18:02:05 Build Finished. 0 errors, 0 v	varnings. (took 29s.748ms)	
			~
·	<		>

Figure 7. Creating a Launch Configuration

it launch configuration proper	rties	
		J
ame: Register Debug		_
Main 🔅 Debugger 🕨 Start	up 🤤 Source 🔲 <u>C</u> ommon	
C/C++ Application:		
Debug\Register.elf	Search Project Browse	
roject:		
Register		
Build (if required) before launching	9	
		$\sim$
Build Configuration: Use Active		_
Enable auto build	O Disable auto build	
Enable auto build     Use workspace settings	O Disable auto build Configure Workspace Settings	
Build Configuration:         Use Active           O Enable auto build         •           • Use workspace settings         Target Information	O Disable auto build Configure Workspace Settings	
Build Configuration: Use Active Denable auto build Use workspace settings Target Information Mcu: STM32L476RGTx	O Disable auto build Configure Workspace Settings Board: NUCLEO-1476RG	
Build Configuration: Use Active Enable auto build Use workspace settings Target Information Mcu: STM32L476RGTx	O Disable auto build Configure Workspace Settings Board: NUCLEO-L476RG	
Build Configuration;         Use Active           O Enable auto build         Image: Configuration           Image: Configuration         State of the configuration           Mcur:         STM32L476RGTx	O Disable auto build Configure Workspace Settings Board: NUCLEO-L476RG	
Build Configuration: Use Active Cable auto build Use workspace settings Target Information Mcue STM32L476RGTx	O Disable auto build Configure Workspace Settings Board: NUCLEO-L476RG	
Build Configuration:         Use Active           Enable auto build         @           @ Use workspace settings         Target Information           Mcuc STIM32L476RGTx	O Disable auto build Configure Workspace Settings Board: NUCLEO-L476RG	

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.

👄 eclipse-workspace - Eclipse IDE		- 0	$\times$
<u>Eile E</u> dit <u>S</u> ource Refac <u>t</u> or <u>N</u> avigate S	arch <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp		
🐔 🚺 🔳 🔘 Run	✓ Segister Debug ✓ 🔅 1 🐨 - 🔛 🔞 - 🍕 - 📸 1 🖆 - 😂 -	🔁 🕶 🕑 🕶	
* • O • 8 • 4 • 🖄 🗁 🛷 •	◎ = = = = = = = = = = = = = = = = = = =	Quick Access	80
🚡 Project 🙁 🔄 Connect 😑 🗖		1 0 % E T "1 =	
ר <mark>אין וישיי די רובי ארציין אין אייר די רובי ארציין אייר די רובי ארציין אייר די רובי ארציין אייר די רובי ארציין ארציין אייר די רובי ארציין ארציין אייר די רובי ארציין אר</mark>		An outline is not available.	
	Problems Tasks Console 22 Properties  Stemminated Register Debug (Ac6 STM32 Debugging) openeed  Warn : block write succeeded  wrote 40960 bytes from file Debug/Register.elf in 1.161265s (34.445 KiB/s)  ** Verstry Started  ** Verstry Started  ** Verstry Started  *** Verstry Started	- <b>6</b> - <b>6</b> - <b>6</b> - <b>6</b>	^
	<pre>Xr&gt;m: rowsumedowe pc: wx.veweedoze msp: wx.200120000 target halted due to breakpoint, current walked: Thread target halted due to breakpoint, current walked: Thread target halted due to breakpoint in 1.215975s (31.855 K18/s) ** Verified over ** ** Verified over ** **********************************</pre>		*

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
<pre>Example/{Inc,Src}</pre>	<pre>device/Examples/STM32L476RG-Nucleo/<example></example></pre>	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

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)

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

### 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.

CMBS Test Application 1.4 Build 41032	- 🗆 X
Running!         Hardware         Module         DCX81 C US8           TDM is Slave         Host Version 04.12 - Build 1 RC 1         Target Version 04.12 - Build 1           Start         RFPI         02e9e26d78         RXTUN         4e         Mode 00           PIN         ffff0000         RVREF         1f         RVBG	Reset     Parameter     Value       Read     RFP1     02e9e26d78       Write     Get     Set
Registration       Open     Close       Handsets Registration       HS     Status       1     Unregistered       2     Unregistered       3     Unregistered       4     Unregistered       5     Unregistered	Global Scope
C Delete Al	
Inte         Log           17/41:37:951         RX < D1:U1:D0:U2:T0:50:M1:T1:17/16:C1:La:P01020304050607080900	FUN Message Parser X
17:41:35:635 RX < D1:U1:D0:U2:T0:S0:M1:T1:I7/16:C1:La:P01020304050607080900 17:41:34:968 RX < D1:U1:D0:U2:T0:S0:M1:T1:I7/16:C1:La:P01020304050607080900 17:41:34:309 RX < D1:U1:D0:U2:T0:S0:M1:T1:I7/16:C1:La:P01020304050607080900 17:41:33:538 RX < D1:U1:D0:U2:T0:S0:M1:T1:I7/16:C1:La:P01020304050607080900	Source Device         0x1         Source Unit         0x1           Dest Device         0x0         Dest Unit         0x2           Transport         0x0         0x0         0x1
1741132728 RX < D1U1000U217050M1111171661114370102004050607080900 1741132728 RX < D1U1000U217050M1171117f161143701020304050607080900 17411321064 RX < D1U1000U217050M117117f161C1143701020304050607080900 17411311190 RX < D1U1000U217050M117117f161C1143701020304050607080900 174113119047 RX < D1U1000U217055M117117f161C1143701020304050607080900	Message Type         (0x01)Command           Sequence         0x0           TE Type         Server
17:41:28:828 RX < D1:U1:D0:U2:T0:S0:M1:T1:17/16:C1:La:P01020304050607080900 ↓ 17:41:27:658 RX < D1:U1:D0:U2:T0:S0:M1:T1:17/16:C1:La:P01020304050607080900 <	IF ID (0x7F16) Unknown DSPG Interface IF Member no such command for dspg interfaces
Calls Groups	Payload Len 10 bytes Payload Raw 01020304050607080900
Rules         Bulb Dialog         Button Led Dialog         Thermostat Dialog         Certification V           Exit         JSystem Window         Price Demo App         HAN Attributes Window	Payload Parsed 01020304050607080900

### 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.

📾 CMBS Start Menu	Х
TDM is Master	
Use Coma	
DVF99 IP Address 192.168.1.3	]
Detect USB Com Port	
Com Port:	$\supset$
Start	
Detected Com Port!	
Continue	

Figure 11. Starting the DSPG Test Application



# A.2. Open registration

CMBS Test Application 1.4 Build 41032	- 🗆 X
Hardware Module DCX81 C USB Running! TDM is Slave Start PIN ffff0000 RVREF 1f RVBG EEPROM Reset Read Write Parameter Read Write Read Write Read Write Read Read Read Read Read Read Read Rea	Value Value 02e9e26d78 t Set
Registration     Open       Handsets Registration     Global Scope       HS     Status     IPUI       HS     Variable       1     Unregistered	
2         Unregistered         000000000           3         Unregistered         000000000           4         Unregistered         000000000           5         Unregistered         000000000           C         Delete	
<ul> <li>Time Log</li> <li>11:38:14:426 Open Registration Succeeded- waiting for handset or device to register</li> <li>11:38:14:426 Open Registration</li> <li>11:37:12:617 Got Parametres Get Response</li> <li>11:37:12:616 Get EEPROM Parameter (Address=0xf55, Length=2)</li> <li>11:37:12:616 Get Parameter (16)</li> <li>11:37:12:613 Got Parameter (16)</li> <li>11:37:12:613 Got Parameter (16)</li> <li>11:37:11:435 Got Parameter (Address=0xf55, Length=2)</li> <li>11:37:11:435 Got Parameter (Address=0xf55, Length=2)</li> <li>11:37:11:432 Got Parameter (16)</li> <li>11:37:11:432 Got Parameter (16)</li> <li>11:37:11:430 Got response for Get Handsets</li> </ul>	Clear Log Screen Clear Msg Log File FUN Log File FUN Report 60 Enable HAN Log
Calls Groups	1
Rules         Build Dialog         Build Dialog         Thermostat Dialog         Certification Window         Quicklink Dialog         HAN SUOTA           Exit         JSystem Window         Price Demo App         HAN Attributes Window         HAN Report Window         HAN Test Window         HAN Test Window	Extra HAN Window

Figure 12. Open base for registration



# A.3. Log information during registration



The Error indication can be ignored.

CMBS Test Ar	pplication 1.4 Build 41032						- c	×
		~~						
Running!	Hardware Module DCX81CU Host Version 04, 12 - Build 1 RC	1 Taroet	Version 04, 12 - Build 1		EEPROM	Parameter	Value	
TDM is Slave		i inget			Reset Read	RFPI	▼ 02e9	e26d78
Start	RFPI 02e9e26d78 RXT PIN ffff0000 RVF	UN 4e EF 1f	Node 00 RVBG		Target Write	Get		Set
Registration								
Open	Close Close			Global Scope				
Handsets Registra	ation	·			d Decoopee Dequired			
HS Status	IPUI HS Name	Caps		i Ose Comman	lu Response Requireu			
1 Unregiste	red red							
3 Unregiste	red		Refresh					
4 Unregiste	red red							
5 On egiste			Delete					
			Delete All					
<		>						
∧ Time	Log							Clear Log
09:24:15:62	<ol> <li>Got Device Table Read Response</li> <li>could not find device id (1)</li> </ol>	nse with 1 entries sta	rting from index 0					Screen
09:24:15:62	1 Error: HAN Message from Un	nown device(D1:U0:D	0:U2:T0:S11:M1:T0:I11	5:C1:L0:P)				Clear
09:24:15:62	<ol> <li>could not find device id (1)!!!</li> <li>RX &lt; D1110000112:T0:S11:M</li> </ol>	1-T0-T115-C1-L0-P		$\sim$				File
09:24:15:58	3 Get Devices					c	Open	Open
09:24:15:58	<ol> <li>Reading Table type = 2 (0=br</li> <li>Device Stage 3 registration 0</li> </ol>	ief,1=extended,2=ex	tended2			FU	NLog	Msg Log
09:24:14:28	8 Device Stage 2 registration O	K (deviceId=1) (parar	ns=AP1:FV2:OP10000:AF	10240)				interval
09:24:14:28	4 PVC Reset indication for device 1 Registration Closed	te {1}!!				R	FUN Report	60
9:24:07:04	7 Device Stage 1 registration O	K (deviceId=1 IPUI=I	:0000333333)					00
09:23:56:28	3 Open Registration Succeeded	<ul> <li>waiting for handset</li> </ul>	or device to register					
V 09:23:50.20	2 Open Registration							
< Comparison of the second sec						>	Enable F	HAN LOG
Calls	Groups							
Rules Bu	Button Led Dialo	g Thermostat Di	alog Certification	Window	uicklink Dialog HA	N SUOTA	HAN	N Bind
Exit	JSystem Window Price	Demo App	HAN Attributes Window	HAN Report W	/indow HAN Test Window	HAN Extra	HAN	N Window

Figure 13. Log information during successful registration



## A.4. Event counters

AN HAN						-	×
Initialized! Tx Request	Tx End Get Link Status	Clear Unit Counters Clea	ar All Counters	Local Delete Device Delete Dev	c Delete all Devices	Refresh Table	
Device Id IPUI Reg Stat 1 0000333333 Register	<u>us   Link Status   Unit Id   Uni</u> ed In Link 1 DSi	It Type Keep PG Voice Call(0xf., 8	Alives Alerts 27	ampers Last Event	<u>On/Off</u> Alert N/A ON	N/A	
Device Id IPUI Reg Stat	lus		Refresh Blac Local Dele Device	klist te			
Device Information (0x01) HF Core Release (R) (0x02) Profile Release (R)	Unknown Unknown		Base Device Info Get Pack ALL		Keepalive Keepaliv Get Attrib	e Keepalive set Attribute	
(0x03) Interface Release (R) (0x04) Paging Caps (R) (0x04) Paging Caps (R)	Unknown Unknown Unknown		Device Info Get Pack ALL	Device Info Get Pack Mandatory	Attribute	Id: 1	
(0x05) Min sleep Time (R) (0x06) Actual Response Time (R) (0x07) Application Version (R)	Unknown Unknown		Device Info Get Attribute	Attribute Id: 01			
(0x08) Hardware Version (R) (0x09) EMC (R) (0x08) IPUT (R)	Unknown Unknown Unknown		Device Info Set Attribute		Get De	vice Subscription	
(0x0B) Manufacture (R) (0x0C) Location (R/W)	Unknown Unknown		Cle	ar All attributes	Status		_
(0x0D) Device Enable (R/W) (0x0E) Friendly Name (R/W) (0x0F) Device UID (R) (0x10) Serial (R)	Unknown Unknown Unknown Unknown		PVC Reset Reque	est Get PVC Reset State	IPUI UAK		_
AC Outlet AC Outlet Reports OFF ON OK	Add Delete Get ON-	ON-OFF Attribute OFF State	Device 0x 0001 0	Unit Interval Get Me	tering Interval Se	t Metering Interval	Close

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



# A.5. Voice call handling

CMBS Test A	pplication 1.4 Build 410	032				-	X
Running! TDM is Slave Start	Hardware Module DC Host Version 04.12 - Bu RFPI 02e9e26d78	X81CUSB ild 1RC 1 Targe RXTUN 4e	t Version 04.12 - Build 1 Mode 00		Reset Read Target Write	Parameter RFPI Get	Value 02e9e26d78 Set
Registration - Open Handsets Regist	Close Close ration	S Name Caps	RVDG	Global Scope	sponse Required		
1 Unregist 2 Unregist 3 Unregist 5 Unregist	ered ered ered ered ered		Refresh Delete Delete All				
Time 11:37:59:7 11:37:59:7 11:37:59:7 11:37:59:6 11:37:59:6 11:37:59:6 11:37:59:6 11:37:59:6 11:37:59:1 11:37:59:1 11:37:59:1 11:37:59:1 11:37:59:1	Log 11 Got Device Blacklist Ta 10 Get Devices 10 Get Devices 10 Get Devices 10 Get Devices 10 Get Devices 11 Get Devices 12 Get Devices 13 Init Suota-Done 11 Init Suota 14 App Init Donel Let's s 13 Got HAN Msg Register 14 Registering with HAN 15 GOt HAN Init Respons 14 App Init Donel Let's s 15 Got HAN Init Respons 15 CHAN INIT RESPONSE 15 CHAN INIT RESPONSE 1	able Read Response with 0 e le d Response with 1 entries st 2 (0=brief, 1=extended, 2=e tart CMBS. : Response! We are done ini Message Service e!	ntries starting from index 0 arting from index 0 ixtended2 tializing the application			Op FUN Fi Fu Rep	Clear Log Screen Clear Msg Log File Open Msg Log File Nsort 60 Enable HAN Log
Calls Rules 5	Groups UIb Dialog Button L	ed Dialog Thermostat [	Dialog Certification V	Vindow Quicklir	nk Dialog HAN	SUOTA	HAN Bind
Exit	JSystem Window	Price Demo App	HAN Attributes Window	HAN Report Window	W HAN Test Window	HAN Extra	HAN Window

Figure 15. Selecting Calls window



Calls	×
Prefered Codec:     Wide <ul> <li>Call Handset</li> <li>HS:</li> <li>I</li> <li>Line:</li> <li>I</li> <li>Play Tone</li> <li>Dia</li> <li>Stop Tone</li> <li>Stop Tone</li></ul>	
Release       Answer         Delete       Device Id         Unit Id       1         Request Call       Del Digits:         123       Other Party Id:         2       Should         Cancel Request       Other Party Type:         Service       Other Party Name:	ng Ring Auto Answer
Id         Instance         State         DCM State         Remote         Line         Called         CNIP         Codec (offered)         Codec(used)         Receive	ed Digits
	Close

Figure 16. Calls window setup call

🗎 Calls		×
Prefered Codec: Wide	Call Handset HS: 1 Line: 1 Make Call CLIP: 11 CNIP: DSPG	Send Display Play Tone Stop Tone
Release     Answer       Delete     Merge Calls	Call Device Device Id 1 Tx Request Tx End Get Status Link Status Unit Id 1 Request Call Dal Digits: 123 Other Party Id: 2 Cancel Request Other Party Type: Service Vother Party Name: Alexa	atus: Idle(Drop) Call Type Incoming Should Ring Should Auto Answer
Id     Instance     State       5     0x00000005     Incoming	DCM State         Remote         Line         Called         CNIP         Codec (offered)         Codec (offered)         Codec           Connected         D0001U01         0         0         Wide         W	odec(used) Received Digits
		Close

Figure 17. Calls window call connected



🚰 Calls		×
Prefered Codec: Wide	Call Handset	Send Display       Play Tone       Dial
Auto Answer	Make Call CLIP: 11 CNIP: DSPG	Stop Tone
Release Answer Delete Merge Calls	Call Device Device Id 1 Tx Request Tx End Get Status I Unit Id 1 Request Call Dial Digits: 123 Other Party Id: 2 Cancel Request Other Party Type: Service V Other Party Name: Alexa	Link Status: Idle(Drop) Call Type Incoming Should Ring Should Auto Answer
Id Instance State 1 0x00000001 Rel Comp	DCM State Remote Line Called CNIP Codec (offere Idle 00001U01 0 0 Wide	d) Codec(used) Received Digits Wide
		Close

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'.

CO	M Port Settings			×
[	-Set COM Port S	Settings —		]
	COM Port	COM8	•	
	Baud Rate	115200		
			Start	

Figure 19. Starting the CMND Simulator



# **B.2. Sending CMND messages**

🐣 CMND API I	Uart Simulator								_	ПΧ
				1	1			1	1	
Clear	Clear Log File	Open Log File	JSystem	Suota Wind	low Attrib	ute Reporting	Production Window	Certification	Test Window	Exit
Device Info	Unit Info	Unit-to-Plugin	ULE GPIO Map	Voice Cal	G	oup Table	COM Port	COM8 -	Open	Refresh
Messages Log										
Time	TX/RX	Length Serv	ice Id	Message Id	Unit Id IE Pars	ed				
Send outgoing C Outgoing C Message T Aproduce Alert C	message Get/Si MND Messages Type ter on	et eeprom   Num Of Messages T Source Unit ID Conkie	o Send: 1 0x 1	T Ena OTA C Retry	able OTA Control Control					
Reep 7 Get 3t Batter RSSI R Sleep I ✓ On/Of	Allve atus y Request Request Request ff/Toggle	Checksum Enabl	e	Priority Dummy Dela	y Urgent ▼ ay: 0	ms				
								0	pened COM po	rt: COM8

Figure 20. Sending CMND messages



## B.3. Voice call handling

Å	CMND API	Jart Simulator								-		×
	Clear	Clear Log File	Open Log Fil	e JSystem	Suota Windo	w	Attribute Reporting	Production Window	Certification Test Windo	w	Exit	
	Device Info	Unit Info	Unit-to-Plugi	ULE GPIO Map	Voice Call		Group Table	COM Port	COM8 • Op	en	Refresh	1
Γ	Messages Log											
	Time	TX/RX	Length	Service Id	Message Id	Unit Id	IE Parsed					





Figure 22. Voice Call window



ULE Voice Call		×
Voice Call Control		٦
Preffered Codec	-	Start Call
Type of the destination		End Call
Digits to dial		
Destination name		Call Request Response Cancel Call Response
Digits of call destination ID		Unit ID
Call Type	<ul> <li>✓ Incoming</li> <li>✓ Ring</li> <li>✓ Auto Answer</li> </ul>	
Volume Control	5 Set Volume Volume Up Volume Down	
Call Status Voice Call Released		Clear Fields

Figure 23. Voice Call window release call



# **B.4. Changing an EEPROM byte**

Clear	Clear Log File	Open Log File	ISystem	Sunta I	Mindow	Attribute Reporting	Production Window	Certificati	on Test Window	Evit	+
Cical	clear coy rile	opencognie	Joystein						on rest window		
vice Info	Unit Info	Unit-to-Plugin	ULE GPIO Map	Voice	e Call	Group Table	COM Port	COM8	Open	Refre	es
essages Log											
îme	TX/RX	Length Ser	rvice Id	Message Id	Unit Id	IE Parsed					_
	C-1/C	- •									
d outgoing	message Get/Si	et eeprom									
d outgoing	message Get/Si	et eeprom									
d outgoing Get / Set E	eprom	et eeprom									
d outgoing Get / Set E	eprom	et eeprom									
d outgoing Get / Set E EEPROM	EPROM	et eeprom	C HAN	C DAIF	ć	° RAM					
d outgoing Get / Set E EEPROM	p message Get/Si EPROM Type (•	DECT	C HAN	C DAIF	ć	RAM					
d outgoing Get / Set E EEPROM Read	g message Get/Si EPROM Type (*	DECT	C HAN	C DAIF	¢	` RAM					
d outgoing Get / Set E EEPROM Read Start Add	g message Get/Si EPROM Type (* dress	DECT	C HAN	C DAIF	¢	° RAM					
d outgoing Get / Set E EEPROM Read Start Add	g message Get/Si EPROM Type (• dress	DECT	C HAN	C DAIF	Ċ	° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length	g message Get/S EPROM Type (• dress	DECT	C HAN		¢	° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length	g message Get/Si EPROM Type (* dress	DECT	C HAN		¢	° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length	g message Get/Si EPROM Type (• dress	DECT	C HAN		¢	° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length Value	g message Get/S EPROM Type (* dress	DECT	C HAN		¢	° RAM					
d outgoing Set / Set E EEPROM Read Start Add Length Value	g message Get/S EPROM Type (*	DECT	C HAN			° RAM					
Get / Set E EEPROM Read Start Add Length Value	g message Get/S EPROM Type (* dress	DECT	C HAN	C DAIF	¢	` RAM					
d outgoing Set / Set E EEPROM Read Start Add Length Value	ressage Get/Si EPROM Type (*	DECT	C HAN		¢	° RAM					
d outgoing Set / Set E EEPROM Read Start Add Length Value Write Start Add	g message Get/S EPROM Type (* dress	DECT	C HAN		(	° RAM					
Get / Set E EEPROM Read Start Add Length Value Write Start Add	g message Get/S EPROM Type (* dress	DECT	C HAN Read EEPR		(	` Ram					
Get / Set E EEPROM Read Start Add Length Value Write Start Add Value	g message Get/S EPROM Type (* dress	DECT	C HAN Read EEPR			° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length Value Start Add Value	g message Get/S EPROM Type (° dress	et eeprom           DECT           0x           0x           0x           0x           0x           0x           0x           0x	C HAN Read EEPR			` RAM					
d outgoing Get / Set E EEPROM Read Start Add Length Value Write Start Add Value	g message Get/Si EPROM Type (* dress	DECT	C HAN Read EEPR		(	` RAM					
d outgoing Get / Set E EEPROM Read Start Add Length Value Write Start Add Value	g message Get/S EPROM Type (* dress	et eeprom DECT 0x 0x 0x 0x 0x 0x 0x 0x 40	C HAN Read EEPR			° RAM					
d outgoing Get / Set E EEPROM Read Start Add Length Value Write Start Add Value	r message Get/S EPROM Type (° dress	et eeprom           DECT           0x           0x           0x           0x           0x           0x           0x	C HAN Read EEPR		(	` RAM					
d outgoing Set / Set E EEPROM Read Start Add Length Value Start Add Value	ressage Get/Si EPROM Type (• dress	et eeprom           DECT           0x           0x           0x           0x           0x           0x           0x	C HAN			` RAM					

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'

Sound	i	×
Playback	Recording Sounds Communications	
Select a	recording device below to modify its settings:	
<i>~</i>	Handset DSPG-LINE1 Default Device	
3	Mccophone Plantronics Audio 478 USB Ready	
<u>C</u> onfi <u>c</u>	gure Set Default	<u>P</u> roperties
	OK Cancel	Apply

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'



Sound			×
layback	Recording	Sounds Communications	
Select a i	ecording o	levice below to modify its settings:	
400	Handset DSPG-LII Default [	NE1 Device	
3	Microph Plantron Ready	one ics .Audio 478 USB	
	Ready		
<u>C</u> onfig	ure	Set Default	<u>Properties</u>
		OK Cancel	Apply



Microphone Properties	х
General Listen Levels Advanced	
You can listen to a portable music player or other device through this Microphone jack. If you connect a microphone, you may hear feedback.	
l 🚽 🥔	
Listen to this device	
Playback through this device:	
Handset (DSPG-LINE1)	
Default Playback Device	
Handset (4- DSPG-LINE2) Handset (DSPG-LINE1)	
Speakers (Plantronics .Audio 478 USB)	
Handset (4- DSPG-LINE4)	
Hanuset (4* DSPG*EINES)	
OK Cancel Apply	





# **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

	Volume Control	- • •
Playback	Recording Output Devices Input Devices Configuration	
	Loopback to Built-in Audio Analog Stereo from DSPG-USB Analog Mono	(B)
Silence	100% (0dB)	
	show: All Streams	•



# **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.



## 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.

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

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

8	7	б	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.

Field Name	Field Description	Ту	pe	Value	M/O	
Preferred Codec Field ID	Field ID	U8		U8 0x1		
Preferred Codec	Codec preferred by the device	τ	18	0x00 - NB 0x01 - WB	0	
Digits Field ID	Field ID			0x2		
Digits	Digits which should be	U8	Len	0x00 - 0x20	0	
Digits	dialed by the device		Char string	0x00 - 0xFF (each U8)		
Other Party Type Field ID	Field ID	U8		0x3		
Other Party Type	Other party type	τ	18	0x00-HS 0x01-Device 0x02-Number 0x03-Service	0	
Other Party Name Field ID	Field ID	τ	J8	0x4		
		U8	Len	0x00 - 0x20	о	
Other Party Name	Other party name	U8	Char string	0x00 - 0xFF (each U8)		
Other Party Id Field ID	Field ID	τ	18	0x5		
Other Party Id	Other party id	U8	Len	0x00 - 0x20	0	
		U8	Char	0x00 - 0xFF		

Table 51-	Payload of	the commands
Table 31-	a ayioau oi	the commanus



			string	(each U8)		
Call Type Field ID	Field ID	U	8	0x6		
Call Type	Call type Mask	υ	8	Bit 0=Call Direction (1=incoming) (0=outgoing) Bit 1=Should Ring (0=no,1=yes) Bit 2=Auto Answer (0=no,1=yes)	0	

#### Table 52- Data Ordering of Payload of call request Command

8	7	б	5	4	3	2	1	Octet		
preferred codec filed id										
preferred codec								2		
Digits field id								3		
Digits length (N)								4		
			Dig	it (char 1)				5		
			Digi	it (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 part	y name (cł	nar M)			8+N+M		
			Other pa	arty id fiel	d id			9+N+M		
	Other party id len							10+N+M		
Other party id (char 1)							11+N+M			
			Other pa	rty id (cha	r K)			10+N+M+K		
	Call Type field									
	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"

📓 Java Control Panel		-		×
General Update Desktop Settings Web Settings Security Advanced Search here				
General settings Manage Certificates				
Enable Java Content in Browser (Required for Applets and Web Start Applications)				
Enable enhanced security restrictions				
Only Java applications identified by a certificate from a trusted authority are allowed to run, and only if the certificate can be verified as not revoked.				
Restore Security Prompts				
	Appl	y O	K Ca	ncel

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.