Integrated Project - EUWB

Contract No 215669

Deliverable

D2.7.1

Architecture definition of CR experimental test-bed

Contractual date:	**M18**
Actual date:	**M20**
Authors:	**Haim Kupershmidt, Udi Ashkenazi, Dalia Aisenberg, Amir Krause, Marcin Filo and Abdur Rahim**
Participants:	**WIS, CNET**
Work package:	**WP2**
Security:	**PU**
Nature:	**Report**
Version:	**1.0**
Total number of pages:	**48**

Abstract

This deliverable defines the Architecture of the cognitive UWB radio experimental test-beds: a DAA test-bed and a WUSB platform by WISAIR based on the WISAIR 601 chip and the HWA.

The first platform is aimed at examination of spectrum scanning and detection tasks, while the second platform is aimed at testing and validating higher layer functionality and behaviour of UWB cognitive radio.

Keywords

Cognitive Radio, Ultra-WideBand, MB-OFDM, PHY-MAC layer Requirements, DAA, Higher Layer

Table of Contents

List of Figures

List of Tables

Abbreviations

BP	Beacon Period
CR	Cognitive Radio
DAA	Detect And Avoid
DRP	Distributed Reservation Protocol
EUWB	CoExisting Short Range Radio by Advanced Ultra-WideBand Radio Technology
FW	Firmware
GUI	Graphical User Interface
HW	Hardware
HWA	Host Wired Adapter
MAS	Medium Access Slot
MSD	Mass Storage Device
OS	Operating System
SW	Software
UWB	Ultra Wide Band
USB	Universal Serial Bus
WUSB	Wireless USB

1 Executive Summary

This deliverable describes the architecture of the CR (Cognitive Radio) experimental test-beds, to be implemented in WP2 of the EUWB project. Two (2) test-beds are intended, and the functionalities covered by both test-beds together encompass the whole CR concept, from detection and related policies, through higher layer behaviour and indications in a realistic environment and finally to adaptation of the cognitive elements of the radios in the system:

- Transmit Power
- UWB Channel

The DAA test-bed is aimed mostly at spectrum scanning, detection of signals and their spectral based classification and identification as victims with higher priority for using the channel or as non important interference signals. From the layers perspective, the DAA platform core is in the PHY and MAC layers.

The Higher-Layer is intended to focus on the algorithms aimed at operating at the higher layers of the CR and show the possibilities for realistic CR systems to function, providing WUSB UWB system as an example. Consequently, The system architecture is aimed at solving the following challenge:

- Provide a realistic testing UWB environment for CR behaviour experiments.

The concept selected for the architecture takes advantage of the existing services of an existing ("of the shelf" / commercial) UWB system, following the WUSB standard, to provide the environment. Yet it enables control over the cognitive elements of the system:

- Transmit Power
- UWB Channel

The Higher-Layer test-bed architecture also supports feedbacks on the UWB link state to the algorithms.

The Control Protocol connecting the Host Controller Server Application and the Host Controller Client Application is part of an inter UWB project collaboration and extends a control protocol created in the WALTER project for testing to the purpose of inter-communicating of instructions to the UWB WUSB device. That interface is also utilized by the UCELLS project for a similar purpose.

The DAA test-bed functionality as also using previous work done in the PULSERS project, the PULSERS II project and the WALTER project.

2 Introduction

This deliverable describes the architecture of the CR (Cognitive Radio) experimental test-beds, to be implemented in WP2 of the EUWB project. Two (2) separate test-beds are designed: DAA (Detect And Avoid) testing and Higher-Layer test Bed. Both test-beds share the same basic HW – a WISAIR dongle, but have separate FW and SW, according to the functionality of each element. These test-beds are expected to congregate at some point in time, as future cognitive products should have both capabilities, but the current separated approach provides both functionalities with the ability of being examined separately.

The DAA test-bed is aimed mostly at spectrum scanning, detection of signals and their spectral based classification and identification as victims with higher priority for using the channel or as non important interference signals. From the layers perspective, the DAA platform core is in the PHY and MAC layers.

The Higher-Layer test-bed is intended to focus on the algorithms aimed for operating at the higher layers of the CR and show the possibilities for realistic CR systems to function, providing WUSB UWB system as an example. Consequently, the system architecture is aimed at solving the following challenge:

- Provide a realistic testing UWB environment for CR behaviour experiments.

The combined functionalities covered by both test-beds together encompass the whole CR concept, from detection and related policies, through higher layer behaviour and indications in a realistic environment and finally to adaptation of the cognitive elements of the radios in the system:

- Transmit Power
- UWB Channel

The deliverable has been structured into several sections:

Section 3 describes the DAA test-bed: Section 3.1 describes system structure and the main elements. Section 3.2 describes the design consideration for this test-bed. Section 3.3 describes the known limitations of the system currently. And section 3.4 describes the recommended preliminary tests, for which this system was optimized. Additionally, a description of the DAA GUI is provided in appendix B of the deliverable.

Section 4 describes the Higher-Layer test-bed: Section 4.1 describes system structure and the main elements. Section 4.2 describes the design consideration for this test-bed. Section 4.3 describes the inherently supported features and the suggested architecture obtains from the commercial device. Section 4.4 describes the control interface which defines the available inputs and instructions for the CR algorithms. Section 4.5 describes the possible algorithms for implementation in utilizing this architecture. At the end, section 5 concludes the deliverable. Additionally, a full description of the control protocol is provided in appendix A of the deliverable.

3 DAA Test Bed Structure Overview

The DAA test-bed is aimed mostly at spectrum scanning, detection of signals and their spectral based classification and identification as victims with higher priority for using the channel or as non important interference signals. From the layers perspective, this functionality of the DAA platform core is in the PHY and MAC layers. The control mechanisms over the detection were implemented in higher layers. Appendix B provides an additional perspective on the system – describing the user interface.

Note the DAA test-bed platform presented is not a communications platform at this stage.

3.1 Block Structure

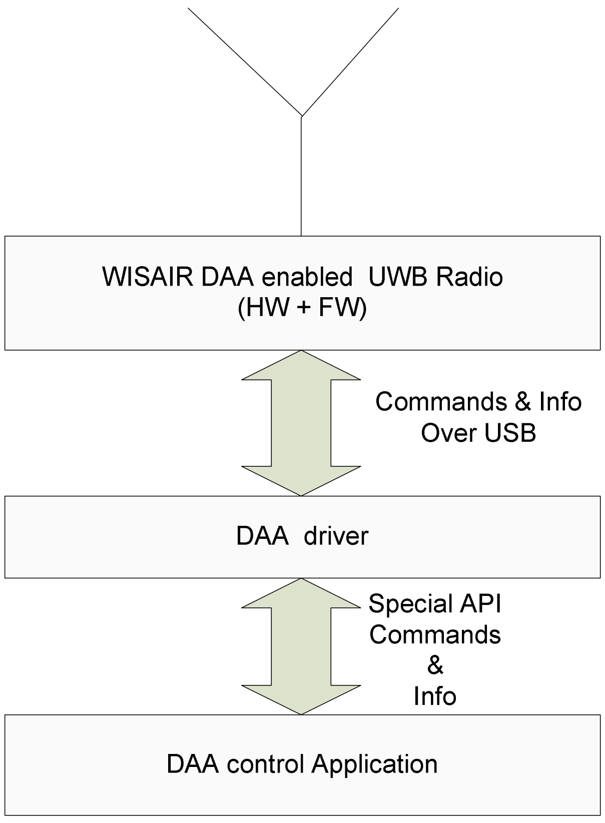

WISAIR DAA enabled UWB Radio (HW + FW)
Commands & Info Over USB
DAA driver
Special API Commands & Info
DAA control Application

Figure 3-1: DAA Experimental test-bed logical structure

Figure 4-1 presents the logical block diagram of the DAA test-bed, during tests. The architecture consists of the following parts:

3.1.1 WISAIR DAA enabled UWB radio

This is the WISAIR UWB dongle with special FW intended to enable and support the DAA test bed functionality. In the test bed configuration, the UWB dongle does not support UWB communications, though it can emulate the existence of communication in its timing regime.

Most of the DAA related processing is performed already at the dongle, and only minimal digested information is sent to the driver.

The DAA enabled UWB radio is connected via USB to the PC which runs the specialized driver.

3.1.2 DAA Driver

This is a special Driver, running the DAA functionality in the UWB radio and communicating with it: providing timing related instructions and receiving resulting information. The driver receives the digested information from the dongle and finishes the processing of the measurements. The DAA driver runs on PC (Windows OS). Similarly to the UWB radio dongle; The DAA driver can only support the DAA application and does not support communications.

The DAA driver is connected via USB to the WISAIR DAA enabled UWB radio. The driver is connected to the DAA Application via a special API, designed to support the control over the DAA required by the test-bed and indicate measurement test results.

3.1.3 DAA Control Application & GUI

The purpose of the DAA Control Application & GUI is primarily the user interface of the DAA test bed system. Appendix B provides a description of the possible features.

The DAA Control Application & GUI is connected to the DAA driver via the API.

3.2 Design Considerations

The architecture of the DAA platform is mostly derived from the combinations of the WISAIR dongle DAA capabilities, in addition to the desire to enable testing of flexible DAA (mostly detection) capabilities. The dongle ability to send information to the PC is limited, hence the necessity of performing most of the information processing in the dongle and sending only the digested information to the driver (PC) for additional processing.

The communication part was disabled in order to allow maximum flexibility in testing the DAA, since the target of the test bed is the study of DAA performance.

3.3 Limitations and known issues

The DAA sample and its' SW has some operating limitations and aimed to be used as reference unit for initial tests. The system was partially tested in Wisair and might has more limitations or SW bugs that Wisair doesn't know of.

Signal pattern detection

It shall detect signal patterns according the ETSI DAA test spec. (TR102763_v14).

Signal level

It shall detect signal level according the ETSI DAA test spec. (TR102763_v14). However, the samples might be more sensitive than required.

Detection frequencies

3.1-3.6 GHz - the DAA radio support a wider range but the current system is configured to detect signals at 3.1 – 3.6.

Stopping the detection

At the moment there is no "Stop detection" button. In case of no signal detection, the scanning will stop after about 100 seconds. If the setup operator wants to stop scanning before 100 sec. we recommend one of 2 options : a. Feeding the setup with a constant strong signal to create detection, b. Closing the application, unplugging the device and then restarting the setup again.

Known issue – switching between detection modes

On some systems, when switching from one detection mode to another (for example : "Full DAA" to "Partial 18.75%") the system will "remember" the previous mode at its' first run. To avoid mistakes, run each mode at least twice or more and ignore the first run after the mode switch.

3.4 Recommended tests

We recommend to start performing tests with both radar and BWA signals according to the ETSI DAA test spec. (TR102763_v14).The main goal should be introduction to the detection capabilities of UWB devices. Therefore, you should focus on the detection statistics on a limited time.

4 Higher-Layer Test Bed Structure Overview

The System Architecture is aimed at solving the following challenge:

- Provide a realistic testing UWB environment for CR behaviour experiments.

The concept selected takes advantage of the existing services of an existing (almost “of the shelf” / commercial) UWB system to provide the environment. Yet it enables control over the cognitive elements of the system:

- Transmit Power[1]
- UWB Channel

The system architecture supports feedbacks on the UWB state.

4.1 Block Structure

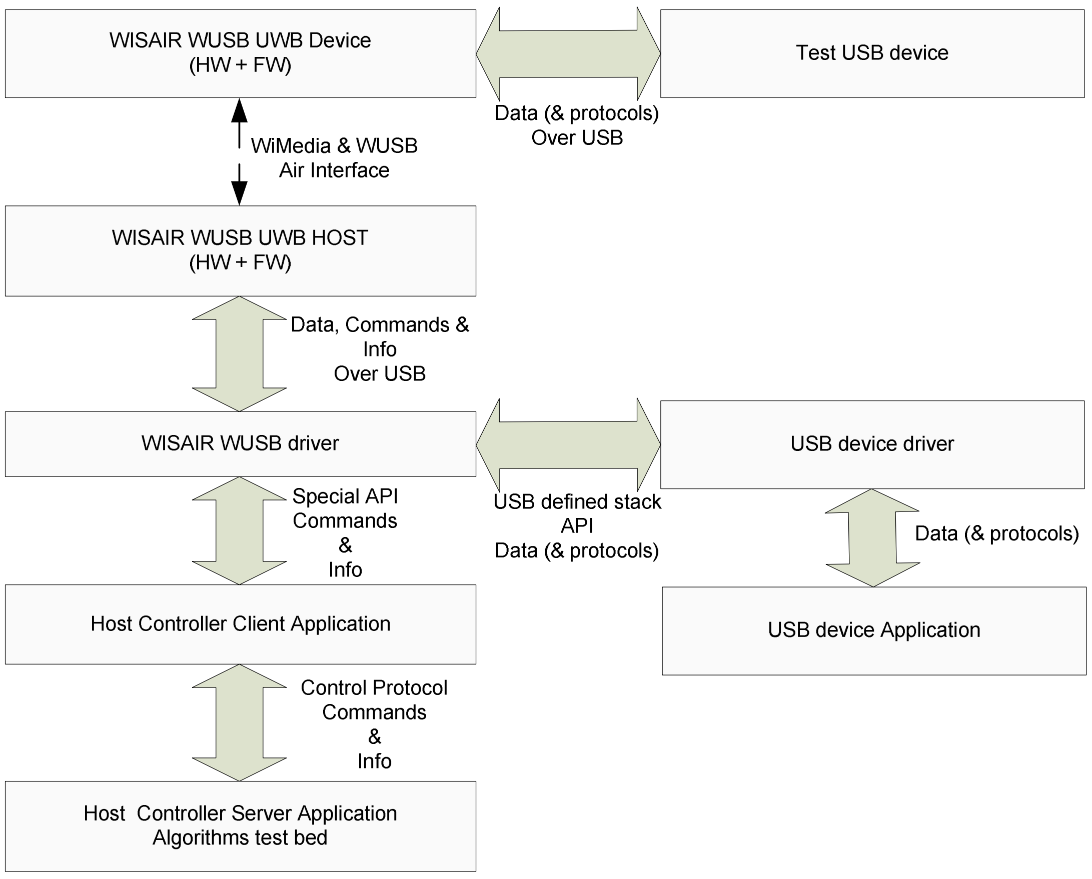

WISAIR WUSB UWB Device (HW + FW)
Test USB device
Data (& protocols) Over USB
WiMedia & WUSB Air Interface
WISAIR WUSB UWB HOST (HW + FW)
Data, Commands & Info Over USB
WISAIR WUSB driver
USB device driver
USB defined stack API Data (& protocols)
Special API Commands & Info
Data (& protocols)
Host Controller Client Application
USB device Application
Control Protocol Commands & Info
Host Controller Server Application Algorithms test bed

Figure 4-1: Higher Layer Experimental test-bed logical structure

[1] The transmit power is only controlled for data transmissions as shell be explained later on.

Figure 4-1 presents the logical block diagram of the Higher Layer test-bed, during tests. The architecture consists of the following parts:

4.1.1 TEST USB device

This is a USB device connected to the system. Any standard commercial USB2.0 device may be connected at this point.

It is recommended to work with devices of the Mass Storage Device (MSD) category, such as Disk on Key and portable Hard Disk (HD). Since this device category supports both traffic directions: In (from the device to the host\PC) and Out (from the Host\PC to the device).

It is possible to change the connected devices and to connect multiple devices in order of aggregating the communications.

4.1.2 WISAIR WUSB device based on UWB radio

This is the WISAIR WUSB commercial device, UWB radio (HW & FW), WiMedia and WUSB standards compliant.

The WUSB device is connected via the USB table adapter (providing the power) to the TEST USB device, and connected wirelessly to the WUSB host utilizing a UWB radio, operating in MB-OFDM compliant to the WiMedia standard and following the WUSB protocol.

4.1.3 WISAIR WUSB Host based on UWB radio

This is the WISAIR WUSB commercial Host, UWB radio, WiMedia and WUSB standards compliant. The Host is implemented in HW and FW, and is considered as a black box for the test-bed purpose.

The WUSB Host is connected via USB to the PC which runs the relevant driver, and connected wirelessly to the WUSB device utilizing a UWB radio, operating in MB-OFDM compliant to the WiMedia standard and following the WUSB protocol.

4.1.4 WISAIR WUSB Host Driver

This is the WISAIR WUSB Host Driver, running on the PC (Windows OS).

The driver for the CR experimental test bed is especially modified to support a Special API which provides specific controls over the driver operation to the Host Client Application.

The driver is based on the standard WISAIR WUSB Host Driver, whose functionality supports the standard USB 2.0 communications operation over the system and multiple features which either should be controlled in a shorter response time or are tightly coupled to specific implantation limitations.

The WUSB Host driver is connected via USB to the WISAIR WUSB Host. The driver is connected to the USB application driver via driver stack as defined in the USB standard, for the purpose of providing communications. The driver is connected to the Host Client application via a special API, designed to support the features required by the test-bed.

The driver shall support the standard communications operations of the UWB even while responding to the instructions given via the API.

4.1.5 Host Controller Client Application

The purpose of the host Controller Client application is to translate instructions received in the Control Protocol to driver API instructions and to relay driver API responses back via the Control Protocol.

This application enables the distancing of the application for implementing algorithms to be tested from the driver, in order to simplify the algorithms and provide greater flexibility and portability, since the control protocol enables the Host Server application to run on a remote computer.

The Host Controller Client Application is connected to the WUSB driver via the API and to the Host Controller Server Application & Algorithmic test-bed via the Control Protocol which runs on a TCP Socket (and may communicate application on the same PC or on separate PCs).

4.1.6 Host Controller Server Application & Algorithmic test-bed

The purpose of this block is to serve as the test bed for implementing and testing algorithms. This block will make the cognitive decisions and instructions. The instructions shall be sent via the Control protocol to the Host Controller Client Application which translates the instructions to the driver via the API.

This application may run on another PC and not on the PC running the WUSB driver and the client application, so long as these PCs are connected via LAN or WLAN. Further, the Host Controller Server Application and the Control Protocol are adjusted to support instructions to multiple Host Controller Clients – making it possible for a single application to control multiple Hosts.

4.1.7 The USB device driver Block and the USB application block

These blocks are the driver and application corresponding to the specific USB device (see 4.1.1). These are standard blocks: The driver is the same as the one required for any USB device (and not unique for WUSB), and the application may be any services running on the system and accessing the USB device (e.g. file transfer, movie player, text editor opening file).

These elements provide the "load" as a logical part of the UWB device for the test bed. They are the source (and\or sink) of the test-bed system own data communications transfers.

4.2 Design Considerations

This architecture was selected for the Higher-Layer CR experimental test-bed for the following reasons:

4.2.1 Working with WUSB, based on existing solution

The WUSB provides the CR with a realistic radio communications environment. WUSB is currently the main service using the UWB radio currently. Relaying on an existing solution allows only to build the additional controls without significantly adding to the complexity of the test-bed HW, FW, & SW and keeping the test-bed within scope.

Section 4.3 details a list some of the important inherently supported features the existing solution provides to this architecture.

4.2.2 Separating the data-path from the CR control algorithms

This architecture, like many other radio architectures, creates a total separation between the radio management algorithms, including the CR test algorithms, and the data path. This separation exists already as part of the WiMedia and WUSB protocol stack architecture and is being followed in this architecture for the very same reason – letting each segment of the stack work on a well defined task and ensuring inter-block effects to be only in predefined and well specified parameters.

This way the existing protocol stack will work as it should in normal situations and create realistic scenarios for the CR algorithms being tested.

4.2.3 Distancing the algorithmic implementation test-bed from the driver

The test-bed implementation of the CR algorithms is intentionally distanced and separated from the actual driver and this provides multiple advantages to the system:

The driver

- Simpler and more standard interface

The Control Application interface is simpler to implement since it takes a more standard form.

- Easy to implement multiple versions of test-bed applications

The test bed is desired to examine multiple algorithms, distancing the driver from the test-bed application architecture simplifies the creation of separate applications to examine each of these algorithms and their effect.

- Flawless driver operation

It is desirable to limit the exposure of the driver to SW misbehaviours and time-outs which may occur on experimental SW. If these problems would be reflected to the driver then a few of them might create OS crush.

4.2.4 Asynchronous control interface

The WUSB architecture practically requires the host controller (HW & FW) to work asynchronously to the WUSB driver in the PC. The host controller has to perform multiple decisions at a very strict and limited timing durations and delays. The WUSB architecture is designed taking this into account and defining the driver interface to the controller in an Asynchronous manner. This architecture is only more evident in this type of host controller – HWA (Host Wired Adapter) which is connected to the PC via a second modem (USB2.0) which has similar throughput magnitude to the WUSB throughput for both the data and the controls.

Given this architecture, there was no point in trying to synchronize the driver and the control application (server or client).

4.2.5 Control Protocol over TCP network socket

The TCP network socket enables communications between the controller server and the control application, which may run on the same PC/Platform or on different PCs/Platforms connected via LAN or WLAN.

This architecture enables the algorithm application of the test-bed to be implemented separately, using different SW language, a different computer and\or a different OS, and reducing the system wide limitations.

An example of the possibilities the network socket enables is running a simulation including multiple radio systems in some scenario over a different computer and see the effect on a UWB node by running an Application server module as part of that simulation.

4.2.6 Compliance to Multi-Project \ Multi usage architecture

Another consideration supporting this architecture is it being well adjusted to multiple additional EC projects working on UWB, all introducing similar requirements for a remote control feature over the UWB radios:

- The requirements of a demonstrator in EUWB WP6 are very similar
- Almost identical requirements stem from the FP7 UCELLS project.
- Further, the WALTER project defined a testing protocol which contained a large sub-set of the required features.

The suggested architecture is envisioned to serve all of these requirements. Further, the architecture is planned in such a way that it expends the control protocol containing a sub-set of the required features, which had been defined by CSR for the WALTER project.

4.3 Inherently supported features \ capabilities

The architecture of the Higher-Layer CR experimental test bed inherits multiple essential features from the standard WiMedia and WUSB sub system elements, which should operate without intervention and can provide useful information for the algorithms. These features support the actual standard communication and let the CR algorithms being tested focus on the CR algorithmic aspects and not on the implementation side of these features. It should be noted that in order to work properly in the original system, these features & capabilities only provide limited abilities for intervention.

In this chapter, subsection 4.3.1 details the main features related to the WiMedia standard, subsection 4.3.2 details the main features related to the WUSB standard. And subsection 4.3.3 details the services and capabilities the inherited system provides for radio management.

4.3.1 WiMedia features \ capabilities

4.3.1.1 WiMedia PHY

The system architecture follows the WiMedia PHY specifications for a HDR UWB radio, including the MB-OFDM modulation, the preambles, the packet structures, etc.

See reference [3] for all the definitions of terms in this section.

4.3.1.2 Beaconing according to WiMedia MAC

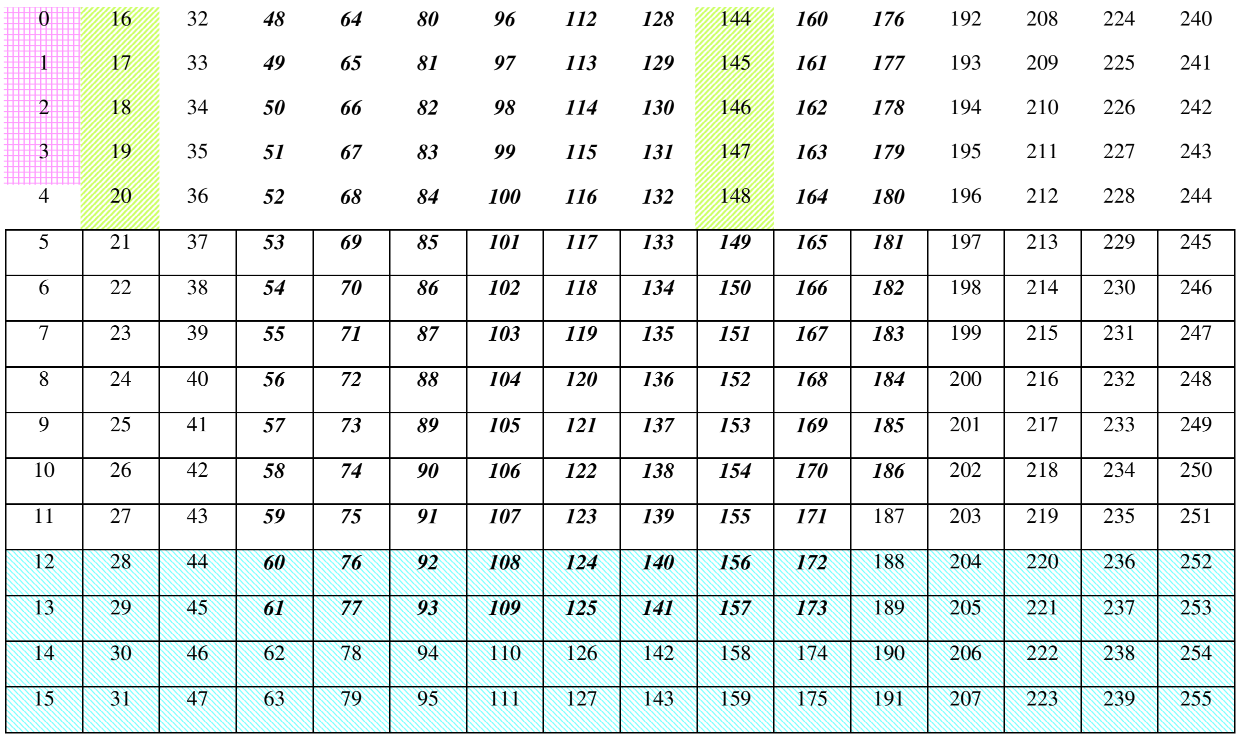
All the UWB devices (WUSB Host and WUSB devices) send beacons according to the WiMedia MAC. The beaconing provides the system with relevant system information provides the WUSB devices with the basic information for connection and provides synchronization and time-base for all the UWB WiMedia devices in the channel.

4.3.1.3 WiMedia DRP reservations

The system reserves MAS according to the MAS availability in that channel: either reserving all the MAS in the super frame (if no other UWB WiMedia device reserved any MAS) or sharing several Hosts in the same channel.

The MAS reservations are part of the information sent in the beacons (see 4.3.1.2). As the WiMedia MAC standard species, MAS reservations may be either safe (owner does not have to release on request) or unsafe (owner has to release on request from other device) .The maximal number of safe MAS for reservation is 122, but a device may reserve the whole super frame, combining unsafe reservations.

The MAS reservations are done using the line reservation concept, Figure 3-1 shows the WiMedia MAC reservation types, in the figure MAS numbers: 12-15, 28-31,44-47, … 252-255 are all in the same 4 lines reservation.



0	16	32	***48***	***64***	***80***	***96***	***112***	***128***	144	***160***	***176***	192	208	224	240
1	17	33	***49***	***65***	***81***	***97***	***113***	***129***	145	***161***	***177***	193	209	225	241
2	18	34	***50***	***66***	***82***	***98***	***114***	***130***	146	***162***	***178***	194	210	226	242
3	19	35	***51***	***67***	***83***	***99***	***115***	***131***	147	***163***	***179***	195	211	227	243
4	20	36	***52***	***68***	***84***	***100***	***116***	***132***	148	***164***	***180***	196	212	228	244
5	21	37	***53***	***69***	***85***	***101***	***117***	***133***	***149***	***165***	***181***	197	213	229	245
6	22	38	***54***	***70***	***86***	***102***	***118***	***134***	***150***	***166***	***182***	198	214	230	246
7	23	39	***55***	***71***	***87***	***103***	***119***	***135***	***151***	***167***	***183***	199	215	231	247
8	24	40	***56***	***72***	***88***	***104***	***120***	***136***	***152***	***168***	***184***	200	216	232	248
9	25	41	***57***	***73***	***89***	***105***	***121***	***137***	***153***	***169***	***185***	201	217	233	249
10	26	42	***58***	***74***	***90***	***106***	***122***	***138***	***154***	***170***	***186***	202	218	234	250
11	27	43	***59***	***75***	***91***	***107***	***123***	***139***	***155***	***171***	187	203	219	235	251
12	28	44	***60***	***76***	***92***	***108***	***124***	***140***	***156***	***172***	188	204	220	236	252
13	29	45	***61***	***77***	***93***	***109***	***125***	***141***	***157***	***173***	189	205	221	237	253
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

Figure 3-1: WiMedia MAS reservations types

4.3.2 WUSB features \ capabilities

In the following, the important WUSB features will be described and explained. All the definitions of terms can be found in reference [2].

4.3.2.1 Topology

The WUSB has a star topology, as shown in Figure 3-2, the host is located at the hub of the network, and the devices are at the nodes. Data is only sent to the host or from it, and not directly between devices. A group of physical-devices containing a host and its devices are called a cluster. A cluster shares the same WiMedia channel and the reservations for which the Host is responsible.

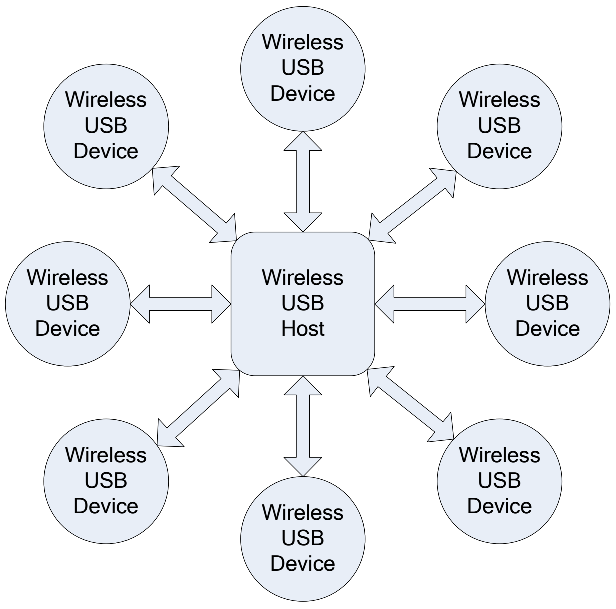


Figure 3-2: WUSB star topology

4.3.2.2 Data Access – TG, MMC, Bursts and Burst Acks

The basic channel access in WUSB is a TG (Transaction group). The structure of a TG is shown in Figure 3-3: .

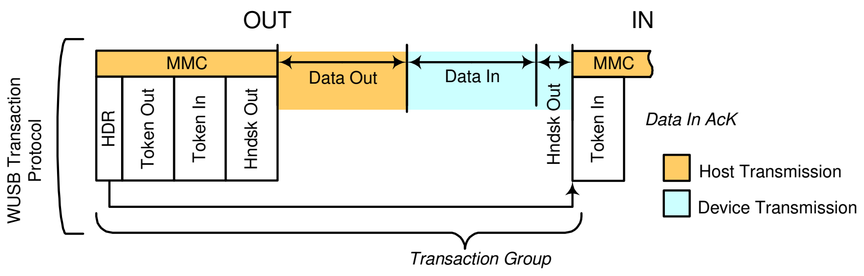


Figure 3-3: WUSB TG(Transaction group)

A TG must have an MMC – Micro Management Command, in which the Host defines all the relevant parameters mentioned in 4.3.2.2. The MMC duration varies based on the data content of the MMC, a typical maximal value would be about 25µSec. Note that the MMC also contain a section for Acknowledgment of all incoming data packets. Finally, in each MMC, the timing of the next TG – the start of the next MMC is defined.

After the MMC, the Host transmits the data it should send to all other devices, after that, the devices send their data and then the devices send their acknowledgments for the data sent previously. This arrangement is intended to minimize the flow direction reversals (changes from Tx to Rx and vice versa) of both Host and devices.

The Acks format is always a Burst Ack, with variable memory size of up to 32 packets. The Acks may also indicate a flow control situation for which the WUSB protocol has a built in mechanism to resolve.

These services are not in the scope of the CR, and it is therefore very useful of the WUSB existing elements to handle them seamlessly.

4.3.2.3 WUSB Management – Host responsibilities

While the WUSB devices do create their own beacons, their operation is actually controlled by the host. The host decides on all the communication characteristics, for example:

- The **Channel** (frequency band group, and the Frequencies set and time interleaving),
- The DRP reservations,
- The internal cluster timings
  - When each data packet shall be sent.
  - When Acknowledgment shall be sent.
- The number of packets in a burst.
- The size of the data packets per each packet (but only if supported by the device).
- The data rate per each packet burst.
- The **transmit power** per each packet burst.

Some of the information sent in the MMC (channels and transmit power) should be decided upon using the CR algorithms, while other information is generated at multiple other elements of the stack.

Other host decisions are based on negotiations with the device \ device driver, and in some cases – the host may have only one choice to decide upon.

As explained above, part of the Host responsibilities in the MMC, is to decide on the number of packets to send to each EP (End-point – logical entity in a device with unique attributes and purpose), the Host has to divide the available throughput in a fair manner considering also the buffer availability of the devices and their service types.

The Data flow is handled by the "standard" sections of the WUSB protocol stack without intervention of the control applications.

This feature of the system is very important for the CR, since most of the instructions generated by the CR algorithms in the applications are translated to commands of a host to a device in the MMC.

4.3.2.4 Data – memory allocation, Buffering, Packetizing and Fragmenting

The WUSB architecture is responsible for all the data related operations, in a seamless manner. The data related functionality includes multiple steps, from memory allocations, to data buffering, Packetizing and Fragmenting.

These services are not in the scope of the CR, and it is therefore very useful of the WUSB existing elements to handle them seamlessly.

4.3.2.5 Higher layer – USB 2.0 Transports

The WUSB supports any USB2.0 traffic over it. An example of a typical application is the MSD (Mass Storage Device) which includes it's own higher layer protocol.

These services are the core of creating "a user" which loads the system with realistic data and creates realistic usage.

4.3.2.6 WUSB "Maintenance": Association & Connection (plug & play)

The WUSB driver handles multiple "maintenance procedures", most notable among those are the Association and Connection features, setting which devices are associated with a host and thus allowed to connect to it (in order to protect the data privacy) and then arranging the devices according to the UWB requirements for plug & play devices.

Both of these services are not in the scope of the CR, and it is therefore very useful of the WUSB existing elements to handle them seamlessly.

4.3.3 Radio management services

The following services are provided by the system architecture, and appear to be the main interest of the CR algorithms to take advantage of.

4.3.3.1 Power changing

A CR should control the transmission power. In the case of the WUSB architecture, it is the responsibility of the host to control the transmission power of the various devices. The suggested architecture supports an efficient and diversified power control, letting the host control the transmission power of each device separately.

According to the WUSB protocol the host may change the transmission power level of data packets over a range of approximately 15dB, in steps of approximately 2dB. The host controls the power transmission level per the logical end-points for each CTA thus setting the power of each data burst transmission separately. The power control is done as part of the information the host must specify for each CTA.

In the test bed architecture, the decision to change the transmission power level of some device should be made at the Host Controller Server Application & Algorithmic test-bed (see 2.1.6), which should send an appropriate instruction to the Host Controller Client Application (see 2.1.5), the Client application uses the WUSB Host driver API to indicate the power change, and the WUSB Host driver (see 2.1.4) parses the instruction to EP level and instructs the Host (see 2.1.3), depending on the transmitter the decision is related to, the host shall either thereafter change the power level fields in the following MMC commands (described in 3.2.3) to that device, or change its own transmit power level.

Power changes may be done asynchronously, and the expected delay for execution is rather short – several mSecs, mostly at the PC, processing the decision. Actual measurement is problematic since a transmission power level change will only be seen at the next opportunity of that device to transmit, which depends on the data flow state of that device, and the overall system load.

The selected architecture, controlling the power via the WUSB, has the limitation of controlling only the data transmission power and not the transmission power of control packets. This limitation is considered of low importance as the system throughput requires each system to minimize the control packets air-time, and the duration of control packets is typically much shorter then the duration of data packets.

4.3.3.2 Channel Changing

A CR should control the logical channel (frequency band group, and the Frequencies set and time interleaving) it operates. In the case of the WUSB architecture, it is the responsibility of the host and the host driver to set the logical channel of the cluster, according to the channel decision (which may be received at some other element, in the test bed – at the Host Controller Client Application).

Unlike the transmission power level, the host controls the channel by informing the devices only of channel changes and not in a continuous manner. The host is expected to notify the devices of a power change at a sufficient time before the actual change, and multiple times. Further, the host is required to scan the intended channel prior to actually starting to operate at the new channel, locate the BP (Beacon Period) and join it. Aside from this issue, the flow of a logical channel change instruction is very similar to the flow of the power change (see 3.3.1) but only until the host involvement.

The notification procedure of a logical channel change and the required channel scans require significant time for the execution of this command, to the level of over 100mSecs (and possibly a few hundred). The UWB radio RF capability to hop between the bands at ~3MHz, is not relevant for this issue. This is the expected delay for executing this command, as explained only a negligent part of it is related to the PC execution.

Channel changing should not be taken lightly, since in some scenarios, changing the channel may cause the communications to break. This is particularly true when joining congested channels and if a device misses the channel changing notification due to communication problems in the existing logical channel.

The selected architecture can change the logical channels in Band Group 1 of the WiMedia specifications (3.1 to 4.8GHz).

4.3.3.3 Dynamic Rate Adaptation

In order to support the data communications given the dynamic environment changes and its effect on the link-budget, the WUSB has to dynamically adapt the rate. The existing solution part of the architecture inherently adapts the data rate of each device transmitting according to the current conditions in order to achieve desirable throughput and delay.

The WUSB standard supports 8 channel rates, defined by 8 modes (all or part may be supported by each WUSB Device). The 8 modes enable the trade off selection between high throughput and robustness. The existing host and host driver automatically set the mode, depending on the channel quality and the device capabilities and requirements. It should be noted that while the WUSB standard includes some hooks for measuring channel quality and to set the channel mode, it does not dictate the methods to use these hooks.

WISAIR implementation of the rate adaptation rate changes is a process extremely similar to the power changes (rates are also part of each MMC), except this process and related decisions are being handled at the driver and not at the Host Controller Client Application. The decision to keep the rate adaptation algorithm location was because this functionality seems to be more closely related to data-path maintenance then to CR considerations, further and WiMedia / WUSB device capabilities and drivers negotiations. Yet the decisions made by the rate adaptation algorithm, and related statistics, may (and should) provide inputs to the CR algorithms.

Note the rate adaptation algorithm intentionally does not change the rate of all the End-Points of a device, only the main data End-Points are affected.

4.4 Interface for algorithms

The architecture of the test-bed places the implementation of the CR algorithms at the Host Controller Server Application & Algorithmic test-bed (see 2.1.6). The control interface of the application defines the possible abilities and limitations for the algorithms. For this reason, this section aims to describe the interface.

A full description of this protocol is given in Appendix A. Below a short summery of the main characteristics.

4.4.1 Protocol scope

The host control protocol is designed to provide an external application control over the transmit behaviour of multiple WUSB hosts and the devices in their clusters. The external application may run on the same computer or on another computer connected to the host via LAN or WLAN.

This protocol was integrated to support the control requirements from multiple FP7 projects working on UWB: UCELLS, EUWB (this deliverable and also WP6). The protocol extends the MAC automation protocol defined by CSR in the WALTER projects, and adds features to it.

4.4.2 Protocol principles

The host control protocol is defined over a socket, allowing the actual physical connection definition to be specified per specific connection.

The control protocol should be robust enough to cover not only Ethernet, but also Wireless LAN \ WI-FI connections at maximal range. Further, this protocol closes a control loop, and therefore simplicity is important.

The following principles were found as desirable:

1 Retransmits shall use the Ethernet \ Wireless LAN \ WI-FI TCP-IP protocol to guarantee messages delivery.

2 A single virtual message pipeline from the host server controller to each host client controller. The next message to a host controller shall not be sent until the current message has been finalized, but commands for different Hosts may be sent.

3 The Host client controller shall acknowledge message reception. The command Ack provides feedback for commands which require long execution time of the WUSB host (such as channel change).

4.4.3 Supported main Instructions \ Messages

The features defined by the control protocol are listed in Table 4.4.3-1, additional description of the purpose of the important features for the CR experimental test bed is given below in addition to the general description in the Appendix:

Table 4.4.3-1: Main Relevant Control Interface Instructions \ Messages List

Feature	Description
Set Channel	Set the UWB channel
Get Channel	Get the current UWB channel
Get Devices Num	Get the number of connected devices
Set Power Level	Set the power level of specific device
Get Power Level	Get the power level of specific device
Get Device Communications Status	Get the communication status (Rate, Total Packets and Error Packets) for a device and direction

4.4.3.1 Set Channel

Action:

Set the UWB channel to desired UWB channel.

Purpose:

This feature sets the UWB logical channel, and enables the CR algorithm to set the operational band and TFI.

This ability lets the CR prevent interference at a desirable frequency, by using a desired FFI band. The ability also lets the controller to increase the capacity at the cost of range and transmitted power (by spreading the hosts over several FFI channels, which are allowed lower transmitted power) or to improve the range at the cost of capacity (changing to TFI which allows increased transmission power).

4.4.3.2 Get Channel

Action:

Return the current UWB channel.

Purpose:

This feature could help the controller to verify the host channel in cases of control communication failures. It complements the Set Channel feature.

4.4.3.3 Get Device Number

Returns the number of currently connected UWB devices.

Action:

Get the number of WUSB UWB devices currently connected to the host.

Purpose:

Inform the central controller of the host topology.

4.4.3.4 Set Power Feature

Action:

Set the transmit power for data, of a specific WUSB device or the host.

Purpose:

This feature is essential to the CR since controlling the Transmit power facilitates control of the level of interference that radio generates.

4.4.3.5 Get Power Feature

Action:

Get the transmit power for data, of a specific WUSB device or the host.

Purpose:

This feature could help the controller to verify the host power in cases of control communication failures. It complements the Set Power feature.

4.4.3.6 Get Device Communications Status Feature

Action:

Get the updated communications status of a specific device, The status is provided per end-point of each device, and includes:

- The EP direction, To the Host or from the Host (note, control EPs are bidirectional but are of little interest due to their low level of communications).
- The data rate, set to that EP by the rate adaptation algorithm.
- The total number of packets attempting transmission from this EP (including errors).
  - Note, this value is being actively saturated with limited memory each time it grows beyond an internal parameter.
- The number of packet errors sent or received in this EP.
  - Note, this value is being actively saturated with limited memory each time it grows beyond an internal parameter.

Purpose:

This feature is the main source of cluster communications status information for the CR algorithms.

This status provides the CR algorithms with information on the current active links, thus for example enables the controller to refrain from reducing power if the connection is bad, or to identify a possible interference as explained in section 4.5.

4.5 Possible algorithms for Implementation

This architecture is aimed at implementation of CR algorithms and CR issues of the following types:

- Implementation and examination of General MAC & Higher Layers sniffing mechanisms, (see a short description below).
- Examination of the CR behaviour and decisions given complex environment and information dissemination.

4.5.1 General MAC & Higher Layers sniffing mechanisms

The following concepts of CR algorithms may be used to indicate the existence of interference, as explained in [1]:

- Low Quality Link

If a link suffers from a lot of errors, this might indicate the link is suffering from interference or from adverse propagation channel conditions or being at extreme range.

- Asymmetrical Link

A strong indication for interference would be if the link quality is asymmetrical, meaning high quality in one direction and low quality in the other.

- Link Quality Fluctuations

Most Interference sources do not operate in a constant manner, since the interfering communication is also divided between several devices, and has its own packet durations and communication patterns, which would often create observable fluctuations when examined via the grid of the UWB time domain.

4.5.2 Cognitive Pilot Channel algorithm

The cognitive pilot channel concepts will be used to design a logarithm where the WUSB host will receive information about the nature and characteristics of the interference signal form the existing licensed systems. Based on the interference knowledge, the host will be able its parameter is optimal manner. CPC will also be used by Host to distribute its decision about the power changing, channel changing to the client.

There are two methods could be used for CPC:

1) Out-band CPC- information from exiting licensed radio to WUSB host
2) In-band CPC – host information will be distribute via in-band CPC

5 Conclusions

This document describes the architecture of the CR experimental test-beds. Two (2) test-beds are intended, and the functionalities covered by both test-beds together encompass the whole CR concept, from detection and related policies, through higher layer behaviour and indications in a realistic environment and finally to adaptation of the cognitive elements of the radios in the system:

- Transmit Power
- UWB Channel

The Control Protocol connecting the Host Controller Server Application and the Host Controller Client Application is part of an inter UWB project collaboration and extends a control protocol created in the WALTER project for testing to the purpose of inter-communicating of instructions to the UWB WUSB device. That interface is also utilized by the UCELLS project for a similar purpose.

The DAA test-bed functionality as also using previous work done in the PULSERS project, the PULSERS II project and the WALTER project.

6 References

[1] Zeisberg, S., Schreiber, V.: “EUWB - Coexisting Short Range Radio by Advanced Ultra-Wideband RadioTechnology”, ICT Mobile and Wireless Communications Summit, Stockholm, June 2008, accepted for publication
[2] URL of EUWB consortium: http://www.euwb.eu
[3] FP7-ICT-215669: EUWB Project Deliverable/Report D2.1.2. "MB-OFDM-based “sniffer” function"
[4] Wireless Universal Serial Bus Specification, Revision 1.0, May 12, 2005
[5] ECMA 368 v1.0
[6] ETSI DAA test spec. (TR102763_v14)

Acknowledgement

The EUWB consortium would like to acknowledge the support of the European Commission partly funding the EUWB project under Grant Agreement FP7-ICT-215669 [1],[2].

Appendix A – Control Protocol

This Appendix provides a full description of the control protocol, including the features not required for the CR test-bed.

1. Generic message format

Messages are comprised entirely of Ascii strings, any decimal or hexadecimal values are passed as Ascii characters. Message formats are presented in Table A-1and Table A-2.

Table A-1 : Message Format Server Originator

	Bit 7 Bit 0
Octet 0	Message Id Most significant character
Octet 1	Message Id Least significant character
Octet 2	First character of Parameters
.	.
.	.
Octet n	Final Character of parameters

Table A-2 : Message Format Client Originator

	Bit 7 Bit 0
Octet 0	Response Id Most significant character
Octet 1	Response Id Least significant character
Octet 2	First character of Status
Octet 3	Second character of status
.	.
.	.
Octet n	Final Character of Status

The response may be a status or result; both are defined below.

2. DEV_CONFIG_SET_REQ

a. Purpose

Configure the specified device prior to operation.

b. Message Direction

Message Originated by Server Originator.

c. Message Content

Msg id	id
Parameter 1	Channel
Parameter 2	Device Address

d. Message Format

Octet	**Value**
Octet 0	“2”
Octet 1	“6”
Octet 2	Most significant character of channel
Octet 3	Least significant character of channel
Octet 4	Dev Addr 1
Octet 5	Dev Addr 2
Octet 6	Dev Addr 3
Octet 7	Dev Addr 4
Octet 8	Dev Addr 5
Octet 9	Dev Addr 6

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

Channel is the Channel Number from 9 to 15.

3. DEV_CONFIG_SET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by client.

c. Message Content

Msg id	id
parameter	status

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“7”
Octet 2	Status octet 1
Octet 3	Status octet 2

4. DEV_START_REQ

a. Purpose

Start the specified device

b. Message Direction

Message Originated by Server

c. Message Content

Msg id	id

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“0”

5. DEV_START_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by client.

c. Message Content

Msg id	id
parameter	status

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“1”
Octet 2	Status octet 1
Octet 3	Status octet 2

6. DEV_STOP_REQ

a. Purpose

Stop the specified device irrespective of its current status.

b. Message Direction

Message Originated by server.

c. Message Content

Msg id	id

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“2”

7. DEV_STOP_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by client.

c. Message Content

Msg id	id
parameter	status

d. Message Format

Octet	**Value**
Octet 0	“2”
Octet 1	“3”
Octet 2	Status octet 1
Octet 3	Status octet 2

8. DEV_RESET_REQ

a. Purpose

Request that the specified device is reset to a known state (this may invoke a complete power reset or a S/W controlled H/W reset – this will be specific to the device control S/W in use).

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id

d. Message Format

Octet	**Value**
Octet 0	“2”
Octet 1	“4”

9. DEV_RESET_CFM

a. Purpose

Confirms or otherwise successful reset of device.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
parameter	status

d. Message Format

Octet	**Value**
Octet 0	“2”
Octet 1	“5”
Octet 2	Status octet 1
Octet 3	Status octet 2

10.DEV_CHANNEL_GET_REQ

a. Purpose

Get current PHY channel.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Device Address

d. Message Format

Octet	**Value**
Octet 0	"2"
Octet 1	"A"
Octet 2	Dev Addr 1
	Dev Addr 2
	Dev Addr 3
	Dev Addr 4
	Dev Addr 5
Octet 7	Dev Addr 6

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

11.DEV_CHANNEL_GET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter 1	status
Parameter 2	Current PHY channel

d. Message Format

Octet	Value
Octet 0	"2"
Octet 1	"B"
Octet 2	Status octet 1
Octet 3	Status octet 2
Octet 4	Most significant character of channel
Octet 5	Least significant character of channel

Channel is the Channel Number from 9 to 15.

12. DEV_POWER_SET_REQ

a. Purpose

Configure the specified device prior to test.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Power level
Parameter 2	Device Address

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“C”
Octet 2	Most significant character of power
Octet 3	Least significant character of power
Octet 4	Dev Addr 1
Octet 5	Dev Addr 2
Octet 6	Dev Addr 3
Octet 7	Dev Addr 4
Octet 8	Dev Addr 5
Octet 9	Dev Addr 6

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

13.DEV_POWER_SET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter	status

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“D”
Octet 2	Status octet 1
Octet 3	Status octet 2

14.DEV_POWER_GET_REQ

a. Purpose

Get current TX Power level.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Device Address

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“E”
Octet 2	Dev Addr 1
Octet 3	Dev Addr 2
Octet 4	Dev Addr 3
Octet 5	Dev Addr 4
Octet 6	Dev Addr 5
Octet 7	Dev Addr 6

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

15.DEV_POWER_GET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter 1	status
Parameter 2	Current TX power level

d. Message Format

Octet	Value
Octet 0	"2"
Octet 1	"F"
Octet 2	Status octet 1
Octet 3	Status octet 2
Octet 4	Most significant character of power
Octet 5	Least significant character of power

16.DEV_DEVICE_NUM_GET_REQ

a. Purpose

Get the number of devices currently connected to the host.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Device Address

d. Message Format

Octet	Value
Octet 0	"2"
Octet 1	"G"
Octet 2	Dev Addr 1
Octet 3	Dev Addr 2
Octet 4	Dev Addr 3
Octet 5	Dev Addr 4
Octet 6	Dev Addr 5
Octet 7	Dev Addr 6

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

17.DEV_DEVICE_NUM_GET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter 1	status
Parameter 2	Number of devices

d. Message Format

Octet	Value
Octet 0	"2"
Octet 1	"H"
Octet 2	Status octet 1
Octet 3	Status octet 2
Octet 4	Most significant character of number of devices
Octet 5	Least significant character of number of devices

18.DEV_DEVICE_COMM_STATUS_GET_REQ

a. Purpose

Get the current communication status of a specific device: number of transmitted packets, number of failed packets and current data rate.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Host Address
Parameter 2	Device Address

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“I”
Octet 2	Dev Addr 1
	Dev Addr 2
	Dev Addr 3
	Dev Addr 4
	Dev Addr 5
Octet 7	Dev Addr 6
Octet 8	CDID 1
	CDID 2
	CDID 3
	CDID 4
	CDID 5
	CDID 6
	CDID 7
	CDID 8
	CDID 9
	CDID 10
	CDID 11
	CDID 12
	CDID 13
	CDID 14
	CDID 15
Octet 23	CDID 16

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

CDID is a unique device identifier.

19.DEV_DEVICE_NUM_GET_CFM

a. Purpose

Response to the above indicating success or failure.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter 1	status
Parameter 2	In direction Rate adaptation Info
Parameter 3	Out direction rate Adaptation Info

where parameter 2 and parameter 3 has the structure:

Field id	id
Field 1 – 2 bytes	Direction
Field 2 – 4 bytes	Packets number
Field 3 – 4 bytes	Errors number
Field 4 – 2 bytes	Current PHY Rate

"In" is direction from Device to Host and parameter Direction must be equal 1, "Out" – from Host to Device and Direction = 0. Current PHY Rate is number from 0 (53 Mb) to 7 (480 Mb).

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“J”
Octet 2	Status octet 1
Octet 3	Status octet 2
Octet 4	In direction Rate adaptation Info 1st byte
Octet 1 5	In direction Rate adaptation Info 12th byte
Octet 16	Out direction Rate adaptation Info 1st byte
Octet 27	Out direction Rate adaptation Info 12th byte

20.DEV_DEVICE_BLOCK_REQ

a. Purpose

Disconnect device from the host and disallow it to reconnect.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Host Address
Parameter 2	Device Address

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“K”
Octet 2	Dev Addr 1
Octet 3	Dev Addr 2
Octet 4	Dev Addr 3
Octet 5	Dev Addr 4
Octet 6	Dev Addr 5
Octet 7	Dev Addr 6
Octet 8	CDID 1
Octet 9	CDID 2
Octet 10	CDID 3
Octet 11	CDID 4
Octet 12	CDID 5
Octet 13	CDID 6
Octet 14	CDID 7
Octet 15	CDID 8
Octet 16	CDID 9
Octet 17	CDID 10
Octet 18	CDID 11
Octet 19	CDID 12
Octet 20	CDID 13
Octet 21	CDID 14
Octet 22	CDID 15
Octet 23	CDID 16

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

CDID is unique device identifier.

21.DEV_DEVICE_BLOCK_CFM

a. Purpose

Confirms or otherwise successful reset of device.

b. Message Direction

Message Originated by Client.

c. Message Content

Msg id	id
Parameter	status

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“L”
Octet 2	Status octet 1
Octet 3	Status octet 2

22.DEV_DEVICE_UNBLOCK_REQ

a. Purpose

Allow the device to connect to the host.

b. Message Direction

Message Originated by Server.

c. Message Content

Msg id	id
Parameter 1	Host Address
Parameter 2	Device Address

d. Message Format

Octet	Value
Octet 0	“2”
Octet 1	“M”
Octet 2	Dev Addr 1
Octet 3	Dev Addr 2
Octet 4	Dev Addr 3
Octet 5	Dev Addr 4
Octet 6	Dev Addr 5
Octet 7	Dev Addr 6
Octet 8	CDID 1
Octet 9	CDID 2
Octet 10	CDID 3
Octet 11	CDID 4
Octet 12	CDID 5
Octet 13	CDID 6
Octet 14	CDID 7
Octet 15	CDID 8
Octet 16	CDID 9
Octet 17	CDID 10
Octet 18	CDID 11
Octet 19	CDID 12
Octet 20	CDID 13
Octet 21	CDID 14
Octet 22	CDID 15
Octet 23	CDID 16

Where a Device Address is the EUI-48 address of 12 Hexadecimal Digits.

CDID is unique device identifier.

23.DEV_DEVICE_UNBLOCK_CFM

a. Purpose

Confirms or otherwise successful reset of device.

b. Message Direction

Message Originated by Client

c. Message Content

Msg id	id
parameter	status

d. Message Format

Octet	Value
Octet 0	"2"
Octet 1	"N"
Octet 2	Status octet 1
Octet 3	Status octet 2

Appendix B – DAA Operation Manual

1. Introduction

Wisair has decided to send the DAA samples following the industry's need for reference units. The main goal is presenting detection capabilities according to ETSI DAA test spec. (TR102763_v14) and enabling the laboratories trying out their setups. The samples do not implement avoidance mechanism. The application provided scans for signals for 2 sec. (except for "Full DAA") and shows the detection results in case of detection. If no detection was achieved, the application will continue 2 more seconds.

The samples and the SW were partially tested at Wisair and still have some limitations and issues presented at this document. Please make sure that you read the entire document before starting the test session.

2. DAA setup installation

This section will guide you through the installation process. We recommend using a "clean" system without previous UWB installations.

1. Copy directory "Wisair_DAA_v1_2" to C: drive (has to be C: and no other drive)
2. If the files are zipped make sure that you unzip them and have the main directory ""C:\Wisair_DAA_v1_2"
3. Install ATEasy6 on your system from "C:\Wisair_DAA_v1_2\ATEasy_installation" This is the application environment.
4. Copy all files from "C:\Wisair_DAA_v1_2\AX_DLLs" to "C:\WINDOWS\system32"
5. From the same folder ("C:\WINDOWS\system32") run install.bat . CMD window will appear. Follow the instructions.
6. Install Wisman application from
7. "C:\Wisair_DAA_v1_2\Wisman_installation\Wisman_2.0.24.0"
8. Plug the dongle to one of the PC USB ports (cable extender is recommended). Make sure that you get "Wisair Wisman Device" under one of the USB ports at the device manager view.
9. Connect RF cable between the test emulator (generating signals according to ETSI DAA test spec.) and the dongle's RF connector.
10. Run the DAA application " WisairDAA_v1_2.exe" which is located at
11. "C:\Wisair_DAA_v1_2"

3. Operation manual

The current section will give you some operating instructions.

a. GUI

The GUI has a few sections which allows configuration and detection indication.

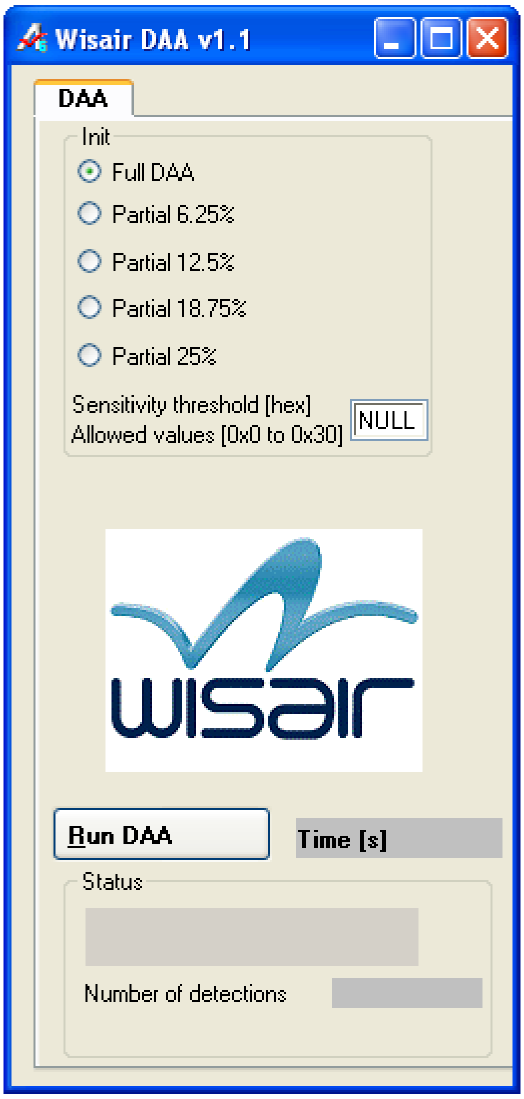


Figure B-1– Wisair DAA v1.1 GUI

- Init section
  5 detection options

○ Full DAA – Uses 100% of the time for detection (1 sec.)
○ Partial 6.25% - Uses 6.25% of the time for detection (2 sec.)
○ Partial 12.5% - Uses 12.5% of the time for detection (2 sec.)
○ Partial 18.75% - Uses 18.75% of the time for detection (2 sec.)
○ Partial 25% - Uses 25% of the time for detection (2 sec.)

Sensitivity threshold

This section gives control of the sensitivity threshold for detection.

Pay attention :Higher values will result in better sensitivity.

Valid values are 0 to 30 [hex]. No need to add “0x”.

- <u>“Run DAA” button</u>

Initiate detection according to the init section configuration.

- <u>Time</u>

Shows the scanning time in seconds from 2 sec. after the “Run DAA” button was pressed until the detection is done. The 2 sec. delay is for the start time of the EUT.

- <u>Status</u>

It shows the process status – “scanning” or “signal detected”.

- <u>Number of detections</u>

It shows the number of detections within the scanning time (according to “Time”) in case that detection was achieved.