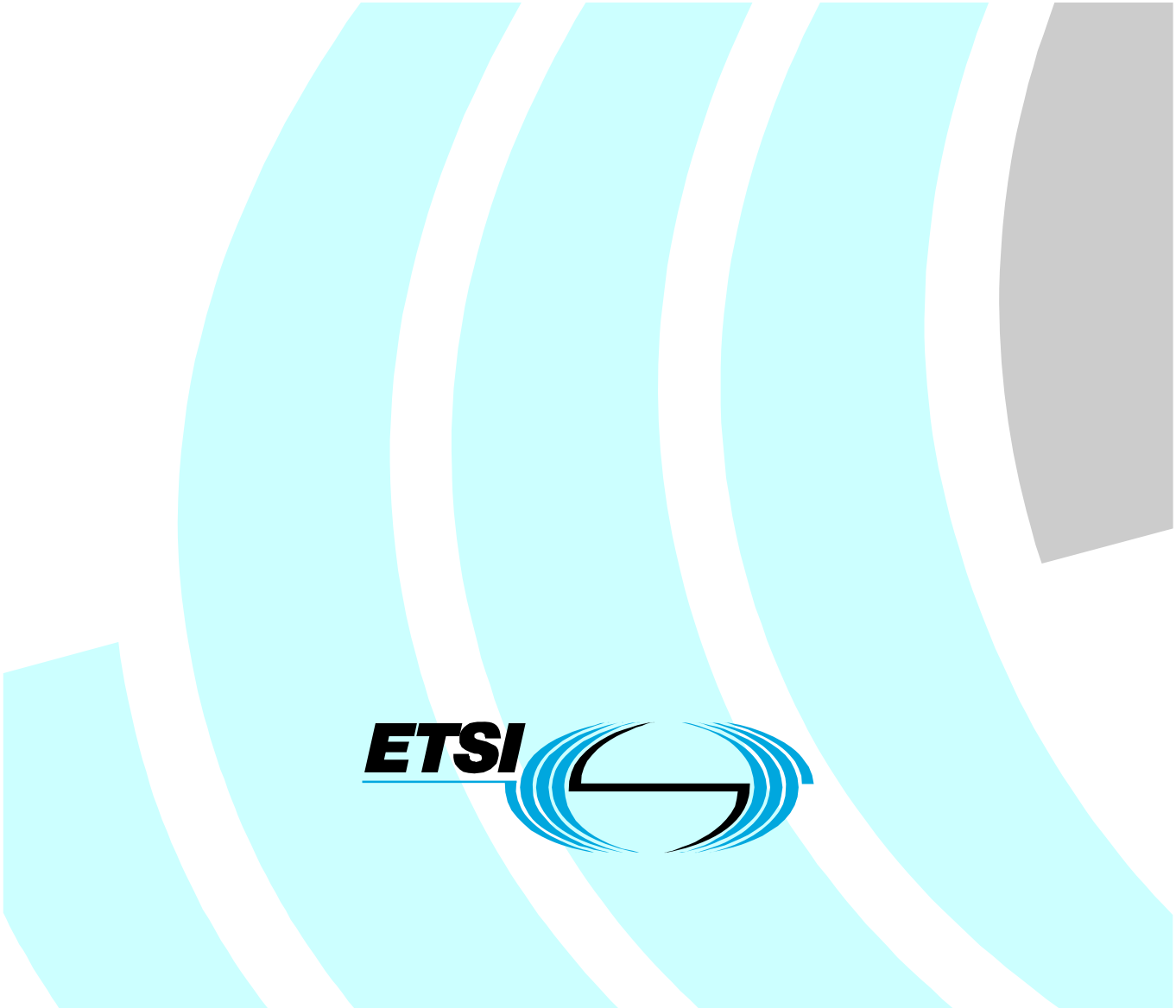
ETSI TR 102 944 V1.1.1 (2011-07)

Technical Report

Reconfigurable Radio Systems (RRS); Use Cases for Baseband Interfaces for Unified Radio Applications of Mobile Device



Reference
DTR/RRS-02005

Keywords
CRS, SDR

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Reconfigurable Radio Systems (RRS).

1 Scope

The objective of the present document is to collect Use Cases that are needed for standardizing the BaseBand Interface (BBI) of Mobile Device (MD) that enables the MD to be configured into various radio applications. The Use Cases to be defined in the present document are related to the internal interface of Unified Radio Application (URA) which has been defined in [i.1]. In order to support the flexible configuration of MD into various radio applications, a standard set of baseband interfaces should be adopted in the URA of MD. Therefore, the present document suggests variety of Use Cases related to the configuration method using the standard interface to be defined later in ETSI TC RRS. The present document will suggest component level Use Cases not system level Use Cases, which particularly means that the Use Cases to be specified in the present document are for supporting interoperations among baseband signal processing modules inside MD which are needed for multiradio application configuration of MD which adopts the standard baseband interface.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

Not applicable.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 102 680 (V1.1.1): "Reconfigurable Radio Systems (RRS); SDR Reference Architecture for Mobile Device".

[i.2] ETSI TR 102 839 (V1.1.1): "Reconfigurable Radio Systems (RRS); Multiradio Interface for Software Defined Radio (SDR) Mobile Device Architecture and Services".

[i.3] ETSI TR 103 062 (V1.1.1): "Reconfigurable Radio Systems (RRS) Use Cases and Scenarios for Software Defined Radio (SDR) Reference Architecture for Mobile Device".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

radio application package: package containing Radio Controller (RC) code, user defined function block code, and metadata needed for setting up and running radio application(s)

NOTE: RC code is downloaded into application processor while the user defined function block is downloaded into baseband processor in accordance with the contents of the metadata. Metadata indicate which function blocks are to be combined in what order for implementing given radio application(s) in the baseband processor.

Radio Controller (RC): software component performing the following functions:

1) transferring context information from corresponding function block(s) in baseband processor to monitor;

2) transferring receive user data packet from Medium Access Control (MAC) buffer to networking stack; and

3) transferring transmit source data packet from networking stack to MAC buffer.

NOTE: An RC performs also upper layer processing of radio application that operates in non real-time. The monitor, to which the context information is transferred, denotes an application that uses the context information in non real-time such as Mobility Policy Manager (MPM). An RC, which operates in application processor in non real-time, can access function block, which operates in baseband processor in real-time, through driver which is prepared in application processor.

BaseBand Interface (BBI): interface consisting of a Radio Application Interface (RAI) and a Context Information Interface (CII)

NOTE: RAI is for baseband signal processing and CII is for transferring the context information to monitor. BBI includes:

1) function block definition of radio application;

2) interface among the function blocks;

3) interface between RC and each of corresponding function block(s).

function block: each modem function needed for real-time implementation of radio application(s)

NOTE: A function block includes not only the modem functions in Layer1 (L1), L2, and L3 but also all the control functions that should be processed in real-time for implementing given radio application(s). Function block is categorized into *standard function block* and *user defined function block*. In more details:

1) *Standard function block* can be shared by many radio applications. For example, Forward Error Correction (FEC), Fast Fourier Transform (FFT)/Inverse Fast Fourier Transform (IFFT), (de)interleaver, Turbo coding, Viterbi coding, Multiple Input Multiple Output (MIMO), Beamforming, etc are the typical category of standard function block.

2) *User defined function block* includes those function blocks that are dependent upon a specific radio application. It is used to support special function(s) required in a specific radio application or to support a special algorithm used for performance improvement. In addition, the user defined function block can be used as baseband controller function block which is to control the function blocks operating in baseband processor in real-time and to control some context information that are to be processed in real-time such as Channel State Information (CSI).

driver: set of software components that includes installer, loader, back-end compiler or full compiler (if necessary), standard function block pool (if necessary), and any other components needed for setting up and running radio application(s)

NOTE: A driver provides the following functions as well:

1) enabling RC, which operates in application processor mostly in non real-time, to access each of corresponding function block(s) operating in baseband processor in real-time;

2) back-end compiler for translating platform-independent IR into vendor assembly in the case of using platform independent IR for user defined function block;

3) full compiler for compiling source code into vendor assembly in the case of using source code for user defined function block;

4) installer for storing radio application package to storage device such as flash memory;

5) loader for loading RC code to application processor; and

6) loader for loading function block(s) code to baseband processor.

A driver which is provided by a modem chip manufacturer is prepared in application processor and includes standard function blocks needed for the configuration of various radio applications. During the configuration of radio application, RC is loaded in application processor and standard function blocks and user defined function blocks are loaded in baseband processor in accordance with the contents of metadata. It particularly means that Driver includes two loaders: one is to load RC in application processor and the other is to load the function blocks in baseband processor. Although there are varieties of modem chip vendors each of which has its own architecture and functioning in baseband processor, the RC which operates in application processor can access each of corresponding function block(s) in baseband processor using the driver which is provided in compliance with BBI by the modem chip vendor.

Intermediate Representation (IR): code obtained as a result of compiling high level code with front-end compiler

NOTE: IR is a non-executable code and independent of baseband processor. It is a structural and behavioural representation of radio application code. Since a user defined function block should be used in every kind of modem chip, user defined function block in radio application package is provided in IR in mid- or long-term scenario. The reason why user defined function block is to be provided in IR instead of executable code is to resolve the portability problem existing in executable code. When user defined function block is provided in IR (platform-independent), the user defined function block is translated into vendor assembly which is executable in a specific baseband processor using back-end compiler that is provided by modem chip manufacturer in the driver of application processor.

vendor assembly: code obtained as a result of compiling the IR with back-end compiler

NOTE: Vendor assembly is executable code and, thus, applicable only to a specific baseband processor. The back-end compiler is provided by modem chip provider because vendor assembly has to be prepared for each of baseband processors. In short, back-end compiler translates IR into vendor assembly, which can be ported on a specific baseband processor for which the back-end compiler translates IR.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

BB	BaseBand
BBA	BB Accelerator
BBI	BB Interface
BPA	Baseband Parameter Aggregation
CII	Context Information Interface
CSI	Channel State Information
FEC	Forward Error Correction
FFT	Fast Fourier Transform
GSM	Global System for Mobile communications
IFFT	Inverse Fast Fourier Transform

IR	Intermediate Representation
LTE	Long Term Evolution
MAC	Medium Access Control
MD	Mobile Device
MIMO	Multiple Input Multiple Output
MPM	Mobility Policy Manager
MURI	MUltiRadio Interface
RAI	Radio Application Interface
RC	Radio Controller
RLC	Radio Link Control
RRC	Radio Resource Control
RRS	Reconfigurable Radio System
RSSI	Received Signal Strength Indication
SDR	Software Defined Radio
TR	Technical Report
URA	Unified Radio Application
URAI	Unified Radio Application Interface
WiMAX	Worldwide Interoperability for Microwave Access

4 Information

4.1 Background

This clause describes the background and scope of Use Cases of BBI. Particularly, we are interested in how the Use Cases to be discussed in the present document are related to the SDR architecture, Unified Radio Application Interface (URAI), and Multiradio Interface (MURI), which are the main topics of [i.1] and [i.2], respectively.

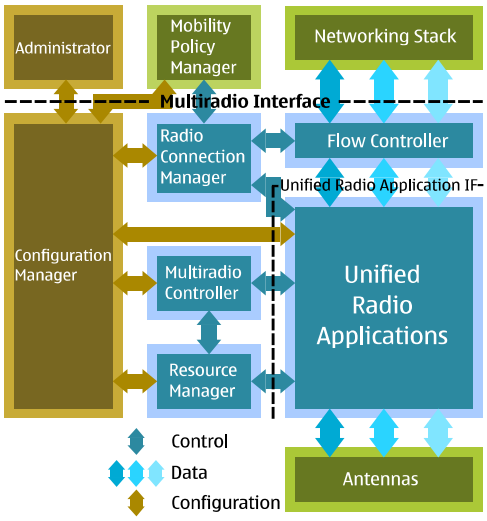


Figure 1: Functional architecture of SDR equipment

In [i.1] and [i.2], the functional architecture of multiradio computer device has been shown as in Figure 1 that is based on the multiradio computer concept. The BBIs are related to the interfaces mainly in the Unified Radio Application (URA) shown in Figure 1 except the ones that are to be defined for transferring the context information.

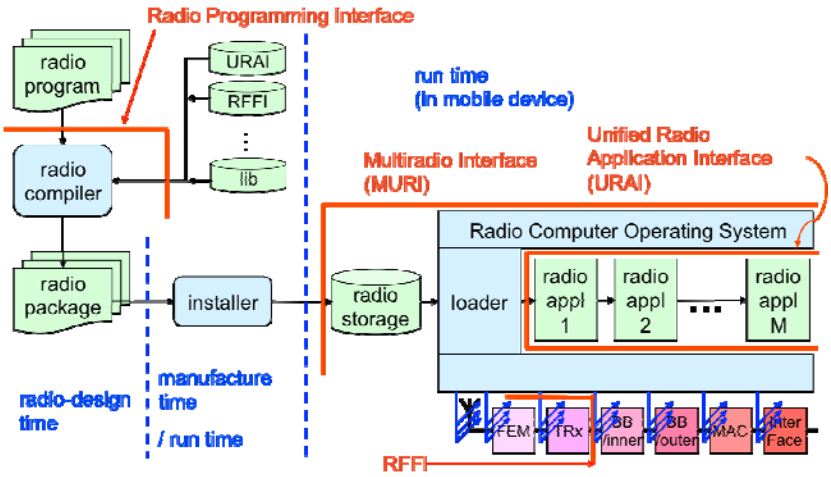


Figure 2: Compile-time and run-time functions of Radio Computer

Figure 2 illustrates compile-time and run-time functions of radio computer [i.2]. According to the scenario shown in Figure 2, radio program is built into radio package through radio compiler. Radio package contains not only the binary code of the radio program components but also metadata about the radio system. The loader component of the radio operating system will install and load radio packages into the execution environment of the radio computer. Use Cases described in the present document follows the scenario shown in Figure 2. Particularly, the present document shows Use Cases of the standard BB interface needed for the configuration of various radio applications focusing on baseband signal processing modules of URA.

4.2 Hardware/Software Framework

This clause introduces hardware and software framework of baseband processor and application processor in MD that are basis of BBI Use Cases to be provided in the present document.

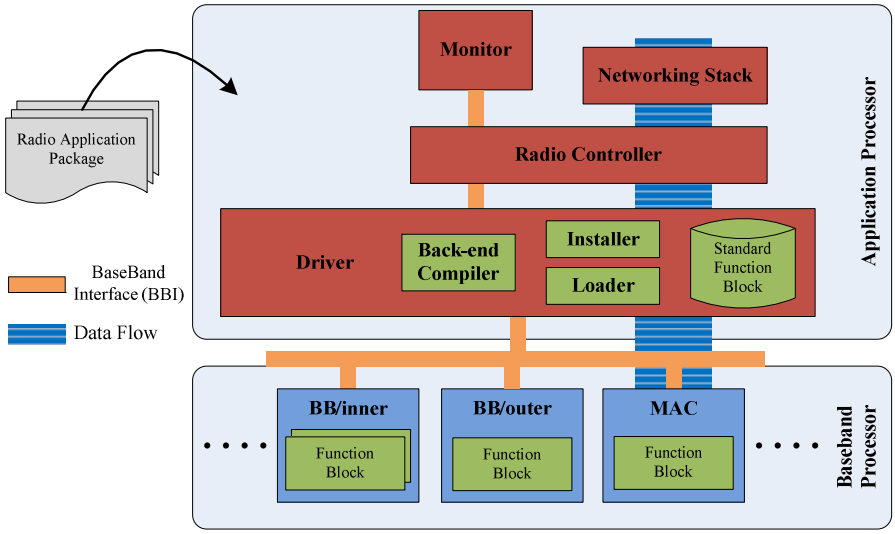


Figure 3: Hardware and software structure of radio computer according to radio application configuration

Figure 3 illustrates hardware and software structure of radio computer set up as a result of radio application configuration. It can be observed that the radio application configuration shown in Figure 3 is in accordance with the radio computer concept shown in [i.2]. Radio application package contains RC code, user defined function block(s) code and metadata. Radio application package that is prepared in compliance with the BBI is installed in a storage and loaded by installer and loader, respectively. RC is loaded in application processor which operates in most cases in non real-time. RC controls transmitting and receiving user data packet. It also transfers context information to monitor for MPM to refer to it. Function blocks are loaded in baseband processor which operates in real-time. Standard function blocks which are installed in driver are loaded in baseband processor together with user defined function block which was included in radio application package. Note that user defined function block should be translated into vendor assembly before loading in baseband processor because radio application package provides it in platform-independent IR. Loader refers to the contents of metadata when it loads the function blocks in baseband processor.

Depending on the radio computer deployment scenarios as detailed in clause 5.4 of [i.2], the translation of function blocks into vendor assembly may be performed off-line by the vendor delivering executable, platform-specific code to the mobile devices or by translating platform-independent code into platform-specific code by the back-end compiler on the platform as illustrated in Figure 3. Following the indications in clause 5.4 of [i.2], the off-line generation of vendor assembly corresponds to a short- to mid-term scenario while in the mid- to long-term the delivery of platform-independent code is targeted.

Note that the baseband signal processing is executed through the BBI in the function blocks of BB/inner and BB/outer for (de)modulation and (de)coding, respectively, in the same concept that have been described as physical radio computing platform in the 7 stages of radio computer concept [i.1] and [i.2]. Besides the function blocks for modem functions, function blocks needed for real-time upper layer operations such as MAC is also supported by the BBI as shown in Figure 3.

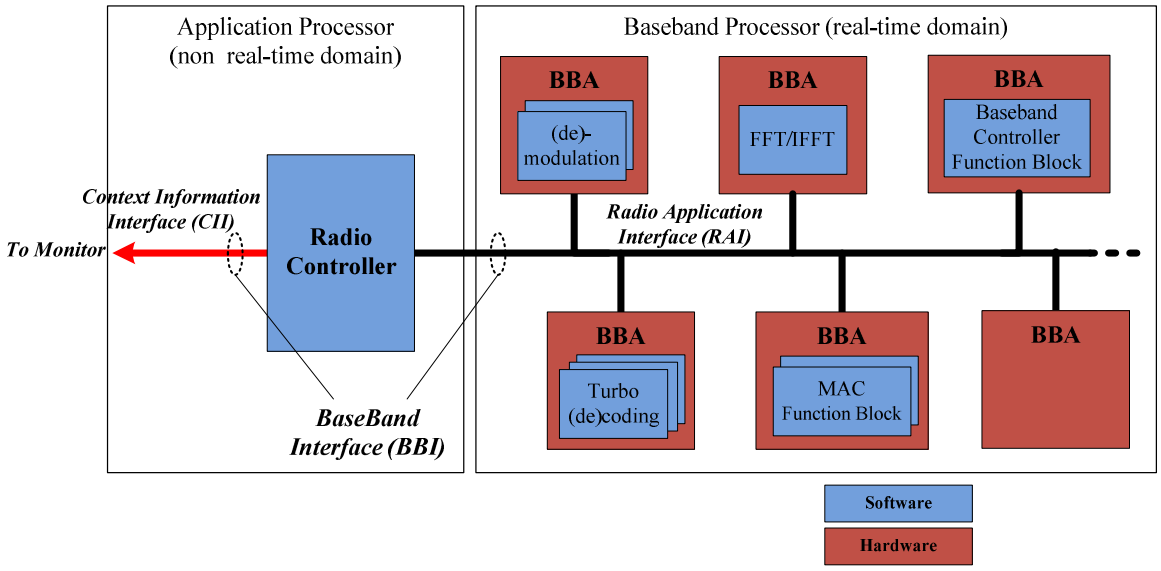


Figure 4: Internal structure of application processor and baseband processor

Figure 4 illustrates internal structure of application processor and baseband processor of MD needed for the configuration of multiradio application. As shown in Figure 4, RC and function blocks are loaded in application processor and baseband processor operating in non real-time domain and real-time domain, respectively. Note that the present document does not put any restriction on application processor that it should operate in non real-time. It particularly means that the application processor may operate either in non real-time or real-time depending upon processor vendor's choice. In most practical cases, however, the application processor usually operates in non real-time. The baseband processor consists of multiple number of BaseBand Accelerators (BBAs). Each BBA is a hardware framework for implementing the function block(s). BBAs, on each of which corresponding function block(s) is ported, interface with one another through the BBI. In addition, it is through the BBI that the RC transfers the context information to the monitor.

As shown in Figure 4, the BBI can be categorized into two parts. The main part of BBI is RAI needed for baseband signal processing such as BBA operation itself, data exchanges among BBAs, and interface between RC and each of BBAs. The other part of BBI is the CII needed for transferring the context information to the monitor. Context information that is to be processed in non real-time is transferred to monitor while context information to be processed in real-time such as real-time CSIs is processed in baseband controller function block in baseband processor. Though the transmission of context information is not directly related to MD configuration, it can be used for other purposes in RRS, for example, adaptive modulation, network-driven link selection, etc. Note that the baseband controller function block is provided as a part of the user defined function blocks from the radio application package. The baseband controller function block performs all the control functions necessary for the baseband processor to execute given radio application in real-time.

The key philosophy of generating Use Cases in the present document is to maximize the configuration flexibility of MD radio application. In order to maximize the flexibility, the architecture of baseband signal processor has been determined in such a way that variety of baseband signal processing algorithms are modularized into function blocks to perform desired radio application using the function blocks while the RC controls the operations such as transferring context information to monitor through the BBI as shown in Figure 4.

5 Use Cases

5.1 Use Case 1: Fixed Pipeline

Figure 5 illustrates the configuration of baseband processor and application processor for Use Case of fixed pipeline. In the Use Case of fixed pipeline, each function block implemented on BBA in MD baseband processor is fixed with a hardware. Each of the function blocks required for implementing variety of radio applications such as Long Term Evolution (LTE), Worldwide Interoperability for Microwave Access (WiMAX), Global System for Mobile communications (GSM), etc is provided as a fixed hardware. Since the function block as well as BBA adopts its own architecture and functioning depending upon the chip manufacturer, the chip manufacturer should provide a driver which supports the BBI for RC to be able to access each BBA as shown in Figure 5. Note that, although all the function blocks in baseband processor are provided in fixed hardware, the RC is provided from the radio application package to be loaded on the operating system of application processor during the configuration time.

The pipeline of baseband processor is determined by the contents of metadata which is provided in the radio application package. Referring to the contents of metadata, the loader, which is provided in the driver, sets up the pipeline in the baseband processor by combining the function blocks needed for implementing a given radio application. Note that the baseband controller function block, which is for controlling the BBAs operating in real-time, is also provided in fixed hardware.

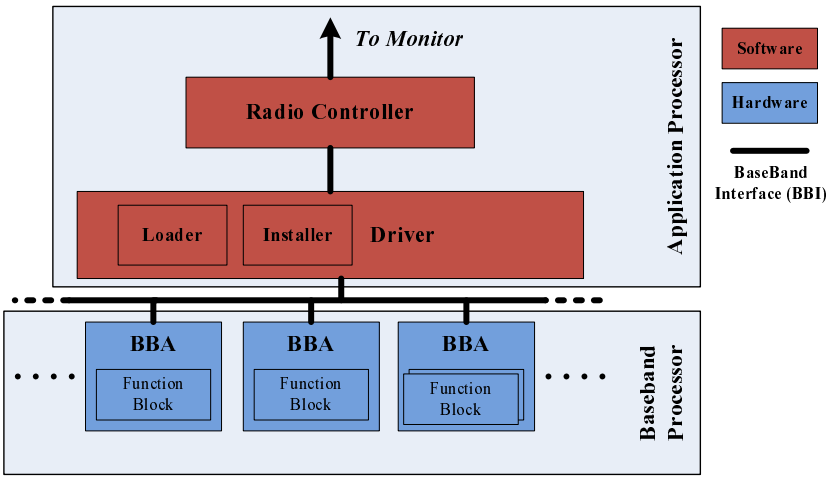


Figure 5: Use Case of fixed pipeline

5.2 Use Case 2: Programmable Pipeline

In the Use Case of programmable pipeline, each of function blocks is implemented using software. We categorize the programmable pipeline into two cases, one is without the capability of user defined function block and the other is with the capability of user defined function block. All the software function blocks are prepared in standard function block pool in the driver except the ones provided as user defined function block in the radio application package. As mentioned in clause 4, the RC and function blocks are loaded in application processor and BBA(s) of baseband processor, respectively, during the configuration time. Function blocks are loaded into BBA(s) in accordance with the contents of metadata that is contained in radio application package. Metadata specify what function blocks are needed in what order for implementing the desired radio application. Although there are varieties of BBA vendors each of which has its own architecture and functioning, the RC can access each of BBAs through the BBI using the driver which is provided by the BBA vendor when the RC needs to fetch the context information.

5.2.1 Without the capability of User Defined Function Block

Figure 6 illustrates the configuration of baseband processor and application processor for Use Case of programmable pipeline without the capability of user defined function block. As the user defined function block is not supported, every function block required for the desired radio application is prepared in the standard function block pool. Standard function block, which should be coded in compliance with the BBI, is prepared in vendor assembly which is executable in a specific baseband processor. Function blocks needed for configuring the BBAs according to the desired radio application are loaded from the standard function block pool by the loader according to the contents of metadata. The baseband controller function block is also prepared in vendor assembly. Note that the baseband controller function block is to control the function blocks operating in baseband processor in real-time and to control some context information that are to be processed in real-time such as CSI.

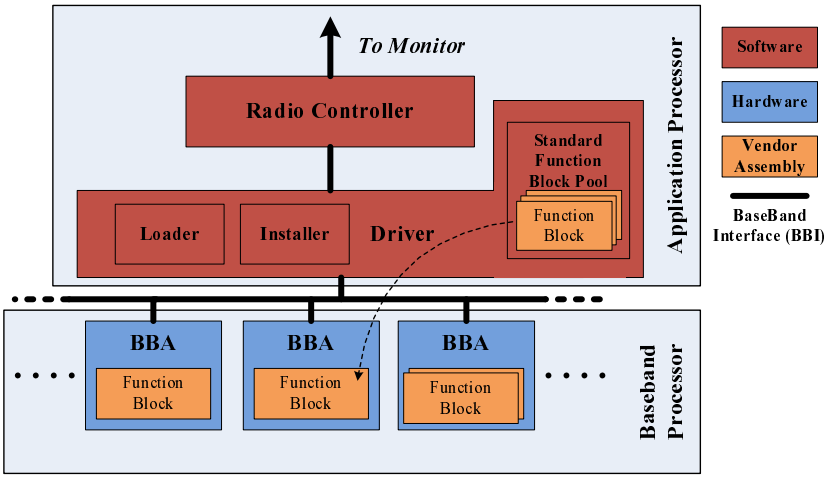


Figure 6: Use Case of programmable pipeline without the capability of user defined function block

5.2.2 With the capability of User Defined Function Block

Figure 7 illustrates the configuration of baseband processor and application processor for Use Case of programmable pipeline with the capability of user defined function block. Since the user defined function blocks should be portable on every kind of BBA provided by the variety of BBA vendors, the user defined function block should be coded in IR which is platform-independent. As mentioned in clause 4 and depending on the radio computer deployment scenarios as detailed in clause 5.4 of [i.2], the technology is expected to evolve into a direction allowing that platform-independent IR is obtained as a result of compiling high level code with front-end compiler on the platform. In a near- to mid-term scenario, however, the compilation is expected to be performed off-line with platform-specific code being provided by the vendors. Note that IR has been introduced in the present document in order to resolve the problem of portability which is an intrinsic problem with the vendor assembly. It particularly means that, in mid- or long-term scenario, user defined function block will be provided in platform-independent IR such that the user defined function block becomes portable on any kind of baseband processors after back-end compilation while the back-end compiler itself is prepared in the driver of application processor as shown in Figure 3 of clause 4.2 of the present document.

As defined in clause 3.1, IR is not an executable code and it is not dependent upon a specific baseband processor. The user defined function blocks are loaded into BBA(s) together with the other function blocks prepared in vendor assembly in the standard function block pool. The loading is performed by the loader provided in the driver. For guaranteeing the full portability of user defined function block in mid- or long-term scenario, the user defined function block is prepared in platform-independent IR when it is provided from the radio application package. Since it is loaded into BBA(s) of which the architecture and functioning is dependent upon its own vendor, the back-end compiler prepared in the driver translates the IR of the user defined function block into vendor assembly in order for the user defined function block to become compatible with the BBA which can only support the vendor assembly.

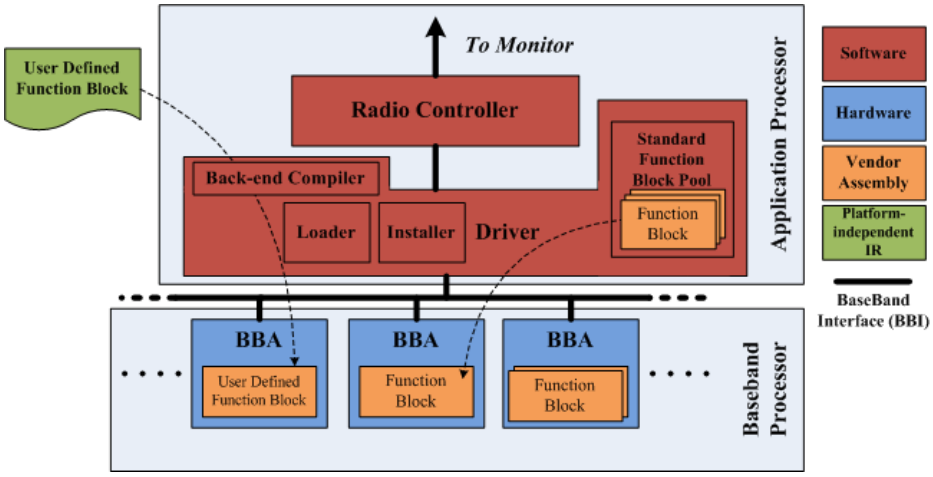


Figure 7: Use Case of programmable pipeline with the capability of user defined function block

5.2.3 With additional capability for BBA to support the IR

Figure 8 illustrates the configuration of baseband processor and application processor for Use Case of programmable pipeline with the capability of user defined function block. Another functionality added in this Use Case compared to the one shown in Figure 7 is that the user defined function block coded in platform-independent IR (mid- to long-term scenario) is now supported by the BBA without the back-end compiler prepared in the driver. In this Use Case, every function block including the ones in the standard function block pool as well as the user defined function block can be coded in platform-independent IR because each BBA supports the platform-independent IR in this Use Case as shown in Figure 8.

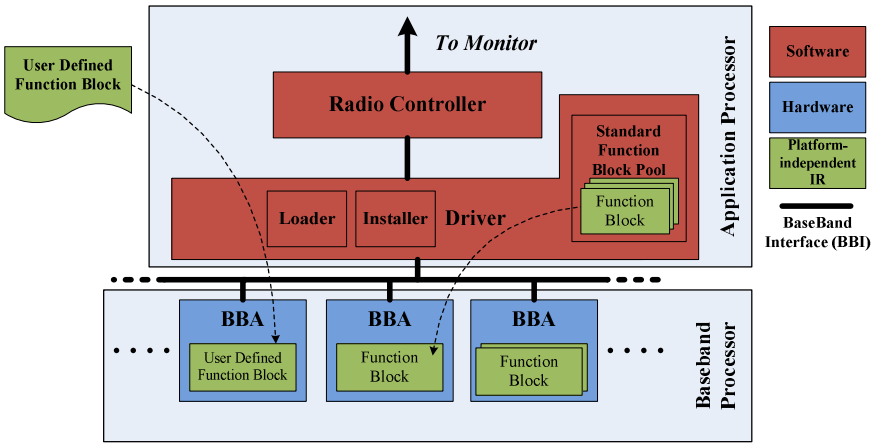


Figure 8: Use Case of programmable pipeline with the additional capability for BBA to support the platform-independent IR

5.3 Use Case 3: Hybrid Pipeline

Figure 9 and Figure 10 illustrate the configuration of baseband processor and application processor for Use Case of hybrid pipeline. In these Use Cases, some function blocks are implemented using fixed hardware on BBA(s) while the other function blocks are implemented using the programmable pipeline. Some operations with extraordinarily heavy complexity could be supported by the fixed hardware function block(s). Function blocks required by many radio applications in common could be also implemented in the fixed hardware function block(s). The hybrid pipeline is mainly for high-speed computation and/or efficient power consumption on BBA. As mentioned earlier in the other Use Cases, the RC and function blocks are loaded from the radio application package into the application processor and baseband processor, respectively, during the configuration time. Loader sets up the function block pipeline in accordance with the contents of metadata. Note that, as in the Use Cases of fixed pipeline and that of programmable pipeline, the RC can access each of the BBAs for fetching the context information through the BBI using the driver, although there is a variety of BBA vendors each of which has its own architecture and functioning.

The function blocks in the standard function block pool can be coded in either platform-specific vendor assembly or platform-independent IR depending upon the translation capability of BBA. Figure 9 shows that all the function blocks in the standard function block pool are coded in vendor assembly because BBA does not support the platform-independent IR in this Use Case. In this case, the driver should include the back-end compiler for the user defined function block to be supported by the BBA(s) unless the user defined function block is compiled in off-line. As shown in Figure 10, however, all the function blocks which are coded in platform-independent IR can be loaded into BBA(s) without the back-end compiler because the BBA supports the platform-independent IR in this Use Case.

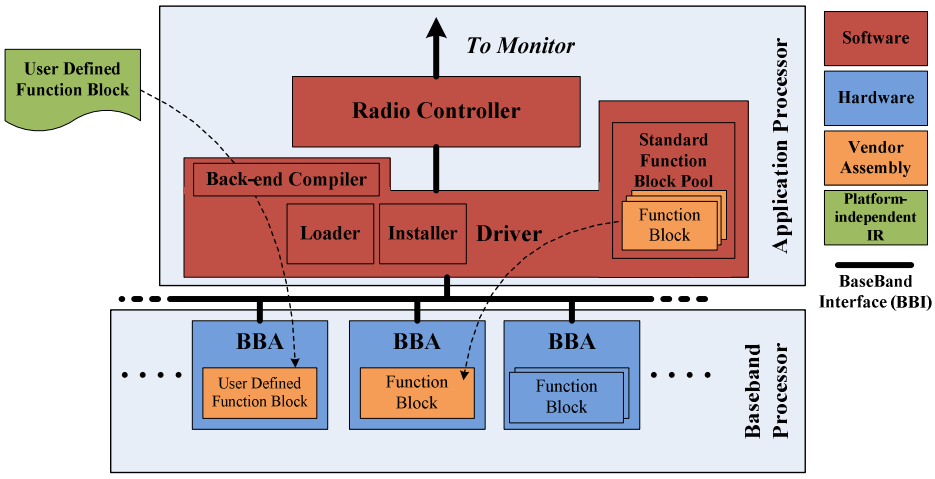


Figure 9: Use Case of hybrid pipeline with interpreter for converting platform-independent IR into vendor assembly

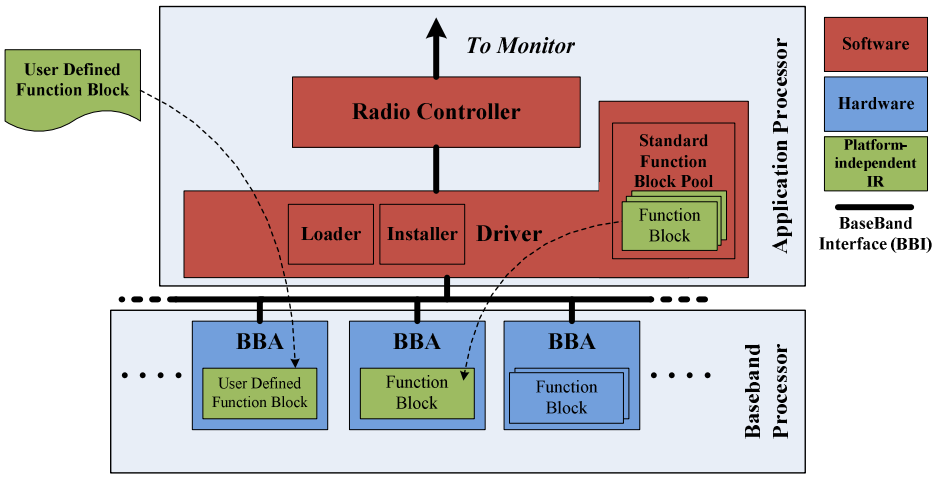


Figure 10: Use Case of hybrid pipeline with platform-independent IR only

5.4 Use Case 4: Context information processing

This Use Case is concerned with the efficient provision of BB parameters to monitor. Figure 11 illustrates the SDR Reference Architecture for Mobile Device with modifications. With respect to the architecture described in [i.1], the monitor and RC are added and it shows more detailed architecture focused on BBI. As shown in Figure 11, the Baseband parameters are to be sent from the RC to the monitor. Monitor denotes an application using non real-time context information like MPM.

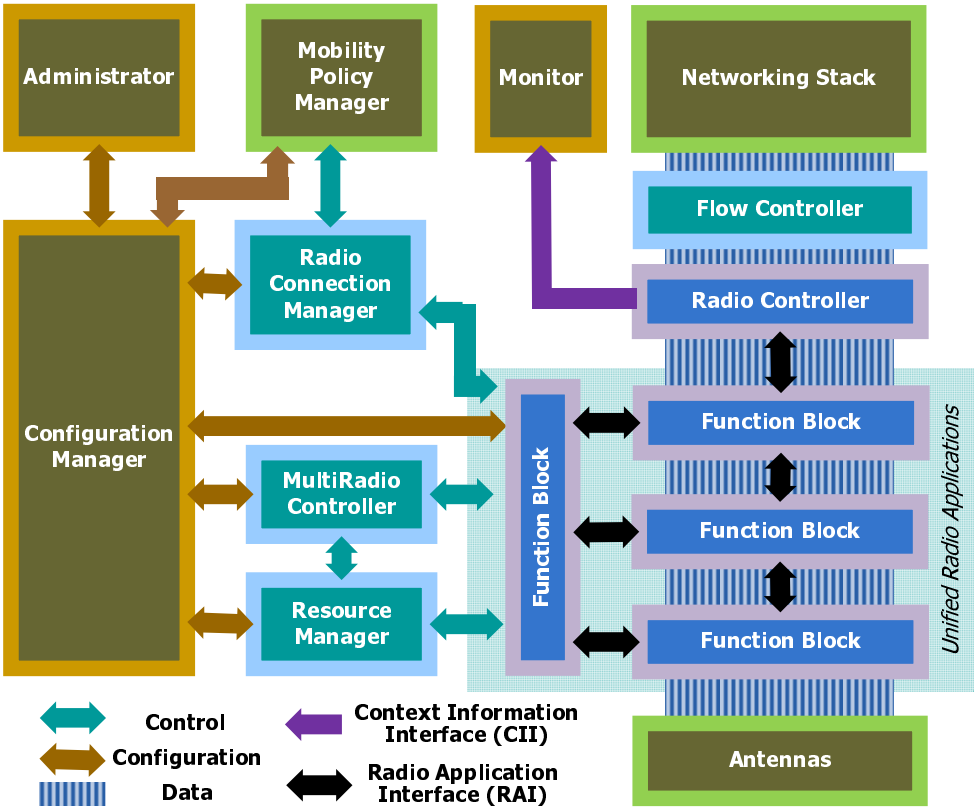


Figure 11: Reconfiguration architecture for Mobile Device with Modifications

The BB parameters to be sent to the monitor are referred to as "context information". The context information is transferred to the monitor through the CII. Figure 12 illustrates how the context information is processed. As shown in Figure 12, the context information provided to RC from the function blocks through the RAI is transferred to the monitor from the RC. It is noteworthy that additional bandwidth is inevitable in the procedure of transferring the context information to the monitor. In order to minimize the additional bandwidth, Baseband Parameter Aggregation (BPA) unit has been introduced. The BPA unit collects all the context information to be transferred to the monitor such as Received Signal Strength Indication (RSSI) measurement, CSI, etc, and processes the context information in such a way that the transferring of the context information occupies as little data bandwidth as possible.

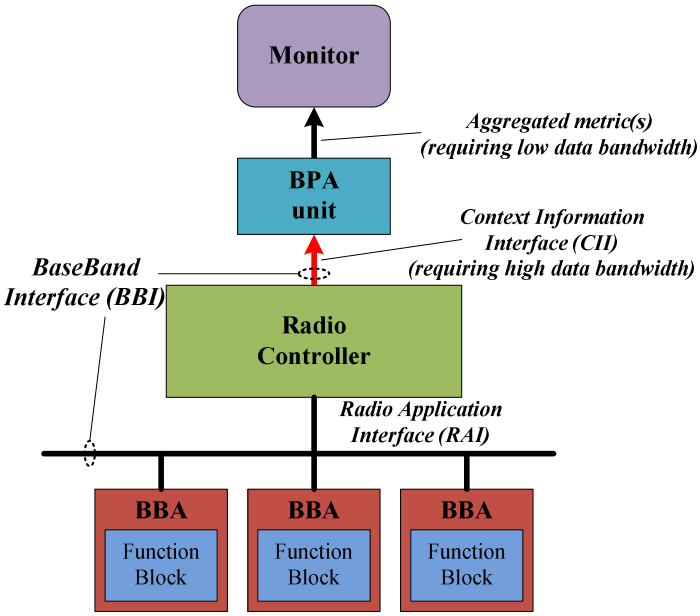


Figure 12: Use Case of context information processing using BPA unit

Figure 13 illustrates Input/Output configuration of the BPA unit. As shown in Figure 13, the inputs of BPA unit consist of context information which are baseband parameters to be transferred to monitor from the RC. The BPA unit converts the context information which are inputs of RC into a single or a few metric(s) such that a minimum bandwidth is consumed during the procedure of transferring the context information to the monitor.

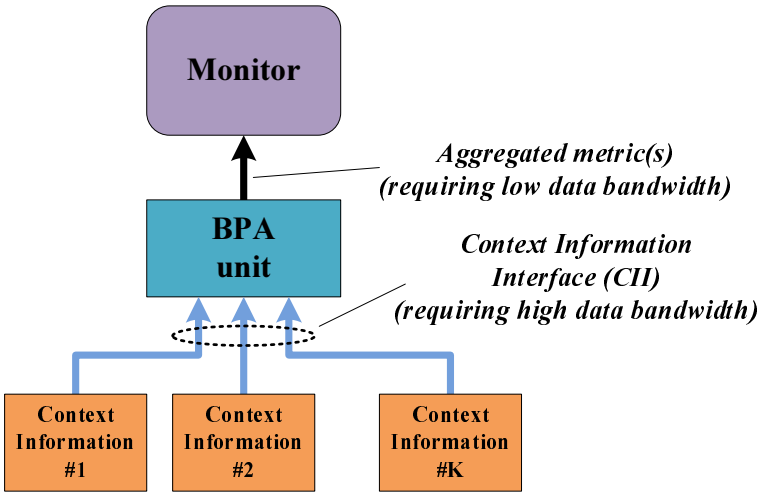


Figure 13: Input/Output of BPA unit

BPA introduced in this Use Case could be used, for instance, either in the Use Case of "Terminal-Centric Configuration in a Heterogeneous Radio Context" or "Network Driven Configuration in a Heterogeneous Radio Context" shown in [i.3].

6 Challenges

TR 102 680 [i.1] and TR 102 839 [i.2] suggest physical radio computing platform consisting of 7 stages. The 7 stages are as follows.

1) antennas;
2) front-end modules (filters, power amplifiers, etc.);
3) RF transceiver;
4) baseband processors for (de)modulation;
5) baseband processors for (de)coding;
6) control processors for protocol stacks;
7) application interface units.

The function blocks that are prepared in compliance with the standard BB interface mentioned in the present document are related to 4) baseband processors for (de)modulation and 5) baseband processors for (de)coding. As shown in Figure 3, however, the BBI is necessary not only for the function blocks needed for 4) and 5) but also for the upper layer interface operating in real-time processing such as MAC. Therefore, further work seems to be essential for specifying and categorizing each protocol layer of each of radio applications in such a way that real-time and non real-time domain are specified for each of protocol layer for each of radio applications. For example, in the case of 3GPP LTE, we need to specify which of physical layer, MAC layer, Radio Link Control (RLC) layer, Radio Resource Control (RRC) layer, etc are to be processed in the application processor or baseband processor for non real-time or real-time processing, correspondingly.

It is also noteworthy that, in all use cases shown in clauses 5.1 to 5.3 of the present document, it is possible to combine the application processor and baseband processor. It particularly means that a single processor could provide all the functions required to both application processor and baseband processor.

7 Conclusion

In the present document, we suggested Use Cases of BBI needed for configuration of MD. We also introduced a basic architecture of hardware and software platform required for the BBI.

- Hardware/Software platform:
  - A detailed substance is given in clause 4.2.
- Fixed pipeline use case:
  - A detailed substance is given in clause 5.1.
- Programmable pipeline without the capability of user defined function block use case:
  - A detailed substance is given in clause 5.2.1.
- Programmable pipeline with the capability of user defined function block use case:
  - A detailed substance is given in clause 5.2.2.
- Programmable pipeline with additional capability for BBA to support the IR use case:
  - A detailed substance is given in clause 5.2.3.
- Hybrid pipeline use case:
  - A detailed substance is given in clause 5.3.

- Context information processing use case:
  - A detailed substance is given in clause 5.4.

Based on Use Cases provided in the present document, ETSI Technical Specifications will be generated as follows:

- Normative System Requirements and System Architecture Specification (TS).
- Normative Protocol and Interfaces Specification (TS).

It is noteworthy that, through the Use Cases provided in the present document, configuration scenario mentioned in [i.1], [i.2] and [i.3] can operate smoothly and efficiently. The ultimate goal of the present document is to provide a milestone for generating BBI of reconfigurable multiradio MD.

History

Document history		
V1.1.1	July 2011	Publication