TRANSCEIVER FACILITY IMPLEMENTATION FOR A WIMAX-LIKE WAVEFORM

Alejandro Sanchez (Thales Communications SA, Colombes, France; alejandro.sanchez@fr.thalesgroup.com); Eric Nicollet (Thales Communications SA, Colombes, France; eric.nicollet@fr.thalesgroup.com)

ABSTRACT

This paper aims at providing feedback on the implementation of the Transceiver Facility Specification on a THALES proprietary high-performance SDR platform for a WiMAX-like waveform. The paper gives an overview of the Transceiver concept, its history and the issues that it addresses. The framework of the work of this paper, the EULER project, is presented and the Transceiver role within, highlighted. A short introduction of the EULER waveform is provided followed by the design of the Transceiver for the THALES SDR platform. Then some of the salient features of the Transceiver programming interface are covered in detail to enable their analysis in front of the constraints set by the WiMAX-like waveform. The result of this analysis is a design that tries to remedy to some of the identified drawbacks of the current specification version. A list of modifications and enhancements towards a new version are compiled at the end.

1. INTRODUCTION

In the SDR arena, the Transceiver Facility Specification [1] (referred to as the "Facility" in this paper) is seen as a key enabler to enhance portability between waveforms and platforms by providing a common, intermediate level, interface for radio transceivers programming and control. The Facility has been selected as a foundation for the project EULER. EULER is a European cooperative project devoted to SDR and that aims, as an important goal among others, to showcase the portability of a waveform for different platforms provided by several project consortium members. The project is organized in such a way that waveform and platforms developers work independently and then bring together the different components for waveform and platform integration. The Facility thus plays an important role as enabler and enhancer of this portability goal.

This paper is organized in the following way:
Chapter 2 summarizes the Facility rationale, the EULER project goals and the role of the Transceiver concept in the project. Chapter 3 takes a glance on the EULER proposed waveform architecture. Chapter 4 describes the Transceiver implementation architecture on THALES platform, i.e the Transceiver decomposition and features mapping on the platform processing resources. Chapter 5 delves into the details of the current version of the Facility document in order to enable chapter 6 to deal with the usage of the Transceiver application programming interfaces (APIs), in accordance to the waveform needs. The work is done by keeping in mind that a new version of the Facility is ongoing and that it will address these issues in a more efficient and elegant way. In this scope solutions and new approaches to deal with most of the exposed problems or drawbacks are also proposed. Furthermore an overview of the upcoming overall features of the new version will be provided in chapter 7.

The goal of this paper is to highlight the capabilities of the current version of the Facility for real-time transmitting receiving control and baseband data exchange for a time division duplex (TDD) waveform based on the WiMAX wireless technology. To that end, use cases of the Transceiver are depicted, focusing on the interface methods usage and associated parameters values. The presentation also intends to identify and point out the main shortcomings of the current version. The issues analyzed lay basically in the time management features domain, data exchange functionalities or generic interfaces definition.

The work presented in this paper has been done thanks to the European Commission funding for the EULER project of the Framework Programme Seven, Cooperation, Securities theme, Grant Agreement FP7-SEC-218133.

2. THE TRANSCEIVER FACILITY AND THE EULER PROJECT

One of the key principles of the whole SDR approach [2] [3] is the fundamental separation between waveforms and platforms. Many benefits stem from the definition of these two separated parts. Typically, Portability and Reconfigurability are usually highlighted as the main benefits but equally important are the new business models enabled by the existence of different suppliers acting as either platform or waveform providers.

However, the application of this key concept becomes difficult when moving close to the RF hardware of radio equipment. The classical approach is to set the waveforms and platforms boundary (from the radio chain perspective) between the baseband modem (seen as the lowest end of the waveform or radio access technology) and the radio frequency Front-End (in the form of an RFIC or an RF module). Aiming at that, several commercial standards have been defined such as OBSAI [4], CPRI [5] or DiGRF [6]. Those standards are low-level specifications too close to hardware to allow the full benefits from the waveform and platform separation approach in terms of Portability and Reconfigurability. Therefore the need for an intermediate level interface to enforce the approach is widely acknowledged. The Transceiver Facility Specification tries to fill this gap [7].

The Specification document became public and was published on January 28, 2009 [1] under the umbrella of the former SDR Forum now the Wireless Innovation Forum. The Transceiver Subsystem Interface Task Force is the group within the Forum undertaking these activities. While significant research was initially performed within the scope of the French Department of Defence, it is important to note that the definition of the specification was completed in the framework of EU funded research [8]. FP6-FP7 projects such as E2R-II [9] and E3 [10] were the sponsors of the final specification publication.

Since then a lot of work has been carried out not only to promote and disseminate this first version of the specification but also to implement and validate it through real waveforms and platforms. Similarly to the definition, some of the implementation activities are being done in the FP7 EU funded framework, as the EULER project illustrates.

As an FP7 Security project, EULER - EUropean Software Defined radio for wireLess in joint sEcuRity operations - [11] focuses on the potential advantages brought by the SDR in international crises, disaster situations. In these scenarios fast deployment, high data rate for data/video services and interoperability among different first responders organizations equipment arise as immediate requirements. In that scope two of the project main goals are the enforcement of the SDR business model of separated waveform and platform suppliers and the porting of the waveform on different platforms. For that purpose the waveform is decomposed in three different modules following an SCA resources [12] approach (PHY, MAC layers and SEC sub-layer). The development of each module is allocated to a different EULER consortium member. Specification of interfaces is thereby an important task done in a cooperative way. Then the independent development of the components by each partner is done followed by a host simulation (in a dedicated simulation environment) and the later integration as a whole waveform in the platform. Two SDR platforms are targeted for the portability exercise. On one hand, this working strategy offered an excellent environment for the application of the Transceiver concept of the Facility since it could meet the project requirement of setting a common interface between waveform and platforms. On the other hand it was expected that the current version of the specification could benefit from the feedback that the implementation of a high data rate real waveform on real hardware platforms brings.

3. THE EULER WAVEFORM ARCHITECTURE AND THE TRANSCEIVER FACILITY ROLE

The EULER waveform (EWF) architecture was designed from the very beginning as a set of separated independent modules. In accordance to partner's know-how, three modules were identified as fundamental building blocks of the EWF: The Media Access Control (MAC) layer block, the physical layer block (dubbed SWiMM) and a security sub-layer block (SEC). Three modules, modelled into three SCA resources. Consequently, a significant effort was dedicated to define the interfaces between these three entities within the waveform, thus MAC-PHY and MAC-SEC interfaces were defined. For the other interfaces dedicated to the platform, SCA architecture and its related interfaces were selected. Two interfaces regarding the data flow were added: the data interface between PHY and the functional radio part of the platform and the interface between the upper MAC and platform provided internet protocol (IP) service (out of scope of this paper). For the first, the Facility interfaces were selected.

4. THE TRANSCEIVER MAPPING ON THE THALES PLATFORM

The hardware composing the platform on which the implementation presented in this paper was done is integrated by four processing resources: A GPP, a DSP and two FPGAs. The mapping of the EWF modules in the platform on which the Consortium members agreed upon, was quite typical: MAC and Security processing hosted by the GPP and physical layer signal processing performed by the DSP.

It is important to underline that the project did not intend to provide a very efficient, optimized implementation but rather a proof-of-concept of the approach. Therefore the decision was made to not exploit the processing capacities of the FPGAs for the EWF signal processing. Nevertheless, they were identified as key elements of the Transceiver implementation.

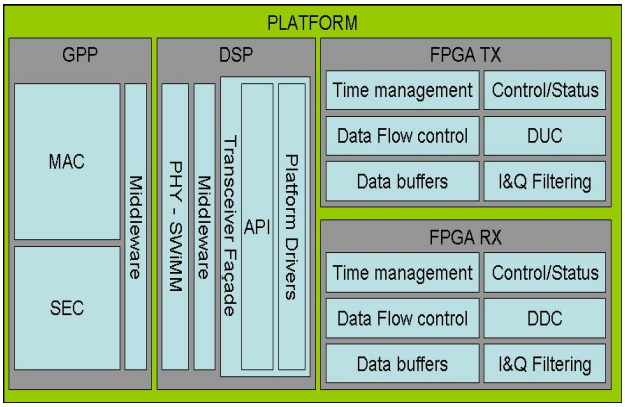

Figure 1: Transceiver implementation and mapping on the processing resources of the THALES SDR Platform

The transceiver channelization, filtering and sampling rate conversion functionalities were located in the FPGAs. The DSP part of the Transceiver was dedicated to host a façade. The Figure 1 depicts the Transceiver implementation on the platform processing resources.

The Transceiver façade hosted by the DSP could be considered as the key element of this proof-of-concept implementation from the perspective of the current version of the Facility since it contains the entire API.

5. QUICK OVERVIEW OF THE TRANSCEIVER API

The Transceiver Application Programming Interface (API) was designed to be able to support as many waveforms as possible with a reduced set of operations and parameters so complexity is minimized. Table 1 and Table 2 summarize all API operations and parameters divided into two categories *control* and *data exchange.*

Control operations are dedicated to programming the transceiver for transmitting and receiving at target times for a given duration determined by the stop and start times couple (the *Timing Profile*) and with a predefined channelization, carrier frequency or power (the *Tuning Profile*). *Data exchange* operations enable data samples exchanges (send/receive) between modem and Transceiver.

One of the main features of the Transceiver API is the time management mechanisms. Two are available: *Absolute Time* and *Event Based Time*.

- *Absolute Time* is intended for systems sharing a common time reference source, well known by both sides of the API, for the instance the baseband modem and the Transceiver. For example in the platform described in chapter 4, this means that DSP and FPGAs have access to the same time reference.
- *Event Based Time* is conversely aiming at platform systems in which only one side of the API is mastering the time, typically the Transceiver rather than the modem, since it is very likely that it will integrate the RF local oscillator.

The code below reproduces the C++ reference implementation of the time management mechanisms provided by the Facility document.

```
// Domain type AbsoluteTime
typedef struct AbsoluteTimeStruct
{
    ULong      secondCount;
    ULong      nanosecondCount;
}AbsoluteTime;

// Domain type EventBasedTime
typedef                          struct
EventBasedTimeStruct
{
    UShort     eventSourceId;
    enum{ Beginning, Previous, Next
} eventCountOrigin;
    ULong              eventCount;
    Latency    timeShift;
}EventBasedTime;
```

The Facility introduces as well the concept of *Undefined* time to be used (for transmit and receive stop times) to request for an immediate stop. In the early phases of the design, when interpreting this concept, the definition appeared as misleading. The document does not clarify whether this *Undefined* setting should apply to the parameters of table 1 operations, for example to the `requestedReceiveStopTime` or to the `requestedTransmitStopTime` or whether it is intended as a new time management mechanism at the same level of the aforementioned two modes. Furthermore in case we take the former assumption and we considerer *Undefined* as a "key" value rather than a type, meaning immediate stop, what value would mean immediate start? *Undefined* too? And what should be used to request for a true undefined activation window for which the stop time is unknown at the activation starting time? The current specification version tells nothing in these last cases.

The conclusion to these questions was that the document is ambiguous and lacks of completeness here. Moreover, two different functionalities seem to mix up in the term *Undefined*, the true undefined, unknown value, and the immediate.

The design carried out decided to define two new time management mechanisms that could be easily mapped to two new data types at the same level of those clearly exposed in

Control Operations Signature (pseudo-code)	Used by	Realized by	Description
createTransmitCycleProfile(**Time requestedTransmitStartTime,** **Time requestedTransmitStopTime,** **UShort requestedPresetId,** **Frequency requestedCarrierFrequency,** **AnaloguePower requestedNominalRFPower)**	Waveform Application	Transceiver Subsystem	Creation of a Transmit Cycle Profile.
configureTransmitCycle(**Ulong targetCycleId,** **Time requestedTransmitStartTime,** **Time requestedTransmitStopTime,** **Frequency requestedCarrierFrequency,** **AnaloguePower requestedNominalRFPower)**	Waveform Application	Transceiver Subsystem	Configuration of an existing Transmit Cycle Profile.
setTransmitStopTime(**Ulong targetCycleId,** **Time requestedTransmitStopTime)**	Waveform Application	Transceiver Subsystem	Specification of the end time of a Transmit Cycle.
createReceiveCycleProfile (**Time requestedReceiveStartTime,** **Time requestedReceiveStopTime,** **Ulong requestedPacketSize,** **UShort requestedPresetId,** **Frequency requestedCarrierFrequency);**	Waveform Application	Transceiver Sub-system	Creation and configuration of a Receive Cycle.
configureReceiveCycle (**Ulong targetCycleId,** **Time requestedReceiveStartTime,** **Time requestedReceiveStopTime,** **Ulong RequestedPacketSize,** **Frequency requestedCarrierFrequency);**	Waveform Application	Transceiver Sub-system	Configuration of an existing Receive Cycle.
setReceiveStopTime (**Ulong targetCycleId,** **Time requestedReceiveStopTime);**	Waveform Application	Transceiver Sub-system	Configuration or reconfiguration of the end of a Receive Cycle.

Table 1: Programming API Control Operations

Data Operations signature (in pseudo-code)	Used by	Realized by	Description
pushBBSamplesTx(**BBPacket thePushedPacket,** **Boolean endOfBurst)**	Waveform Application	Transceiver Subsystem	Notifies availability of a baseband samples packet.
setReceiveStopTime (**Ulong targetCycleId,** **Time requestedReceiveStopTime);**	Waveform Application	Transceiver Sub-system	Configuration or reconfiguration of the end of a Receive Cycle.

Table 2: Programming API Data Operations

the document: *Immediate* and *Undefined*. The modifications proposed in the chapter 7 address this point.

6. THE SWIMM REAL-TIME CONSTRAINTS

As introduced in chapter 3 the EWF is composed by three modules. The rest of this paper focuses on the real-time constraints stemming from the physical layer module or SWiMM. SWiMM stands for "SDR WiMAX Modem". It is actually a simplified profile or sub-set of the IEEE802.16.e standard, with a reduced number of supported features and fixed parameters. It is a TDD waveform, with a fundamental frame alternating transmitting and receiving slots and guard intervals in between. Slots duration and occurrences in the time will depend on the modem role, either taking the role of a Base Station (BS) or a Mobile Station (MS). Frame, transmitting and receiving slots and intervals start, stop times and durations are the critical constraints to be

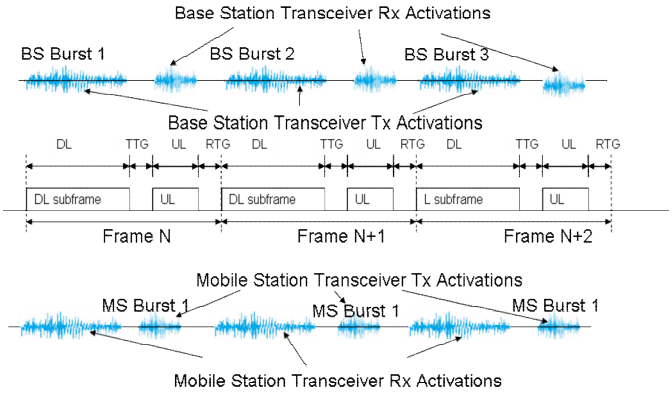

Figure 2: EWF SWiMM frame

satisfied. The Figure 2 depicts in its simplest form the EWF frame.

The basic behaviours that the modem of such a waveform will request from any transceiver are easy to foresee: “Start transmitting at a well-defined time and for a well-defined duration, then stop”; “Start receiving at a well-defined time and for a well-defined duration, then stop”; “Guarantee that the time intervals and the whole cyclic timing are respected”.

Additionally, and only in the case of a MS, one more supported behaviour is necessary for starting a synchronization procedure: “Start receiving as soon as possible and for an undefined duration in order to get synchronized”.

If we focus on the different combinations of start/stop times, durations and order of the activations we come to an exhaustive list of use cases. A further detailed analysis yields up to nine use cases covering both the Base Station and Mobile Station situations:

- Use case 1: BS first transmission
- Use case 2: BS first receive
- Use case 3: BS second and further transmissions
- Use case 4: BS second and further receive
- Use case 5: MS first receive for synchronization procedure.
- Use case 6: MS stop receive for stop synchronization procedure
- Use case 7: MS first receive after synchronization
- Use case 8: MS transmissions when synchronized
- Use case 9: MS second and further receive when synchronized

The platform of the EULER project does not provide a common time reference source for the DSP hosting the SWiMM modem layer of the EWF and the FPGAs. When Implementing the Transceiver, the *Event Based Time* mechanism was the choice of this design. This time management mode deserves some complementary description. In order to support as many waveforms as possible, *Event Based Time* is quite rich in functionalities. It is defined by:

- An event source to identify events as time references, known and shared by both sides of the API.
- An event origin -*beginning, previous, next*- to refer respectively to the very first, the last or next event occurrences.
- An event counter, in order to wait for several occurrences of the reference event, prior to initiate any activation action.
- A time shift, used to convey the amount of time to wait after the event occurrence, prior to the activation.

From the perspective of the SWiMM modem only a subset of these functionalities is needed. All the use cases could be supported by an event source and a time shift. No real need to use the different identification choices offered by the event origin or even the event counter. However, the critical issue is the definition and identification of these event sources and the way to accurately signal the event occurrences, only in that way the communication of timing information between modem and Transceiver could be guaranteed.

The Specification proposes four event sources: *TransmitStartTime*, *TransmitStopTime*, *ReceiveStartTime*, *ReceiveStopTime*. For example if we look at the *TransmitStartTime* event source an event will occur at any time transmission activation starts. That event occurrence is well know by the Transceiver (since it directly controls the radio access), the modem can therefore use this as a reference for typically requesting a further receive activation based on this event source by indicating a time shift corresponding to transmission duration plus the guard interval. In the design exposed within this paper only two event sources were needed to cover completely all the use cases: *TransmitStartTime* and *ReceiveStartTime*.

However the critical aspect was the accurate identification of the event occurrences, in other words, how does the SWiMM know, when an event occurred so it can use it as reference for new activations? This was the main obstacle the design stumbled upon and it is one of main shortcomings of the current version of the specification: there is no mechanism, functionality or interface enabling the synchronization between modem and Transceiver.

To solve this fundamental problem the design took advantage of the TDD characteristics of the EWF waveform. The basic idea is simple: use as much as possible the sole event source whose events occurrences may be inferred by the Waveform: the *ReceiveStartTime* event source. When a reception takes place, SWiMM will receive samples (`pushBBSamplesRx()` operation

invoked by the Transceiver), then it is aware of the *ReceiveStartTime* occurrence. The previous principle, along with assumption that no new *receiveStartTime* will happen until the next frame (5 ms later) gives the waveform enough time to program new transmitting/receiving activations (actually two) before the next occurrence. The procedure is iterated.

The previously described strategy is valid provided that a reception took place. However a number of use cases do not fulfill that and other solutions are implemented for those (see Figures 3, 6 and 7).

The approach does not necessitate the complex features of the *Event Based Time*. Moreover, it cannot support the different options of the event count origin since it is able to refer to the event only when it knows that it happened: *next* is then no useful (*beginning* could be, but it is discarded to avoid tracking time from the very beginning). The accurate occurrence time of the event is not known by SWiMM. This works basically because the EULER waveform does not need it (only frame accuracy is required).

Figures 3, 4, 5, 6 display four of the uses cases.

7. CONCLUSIONS AND ENHANCEMENTS PROPOSED FOR A NEW SPECIFICATION VERSION

The outcome of this design and implementation has provided very interesting insights on this Facility. This section summarizes the main conclusions and proposes modifications and enhancements towards a new version.

The `configureTransmitCycle()` and `configureReceiveCycle()` operations appear of very little utility. Other waveforms might take advantage of these operations potentially, but that it is still unclear. Originally, they were intended for radio access technologies where activations (also known as cycles) may be requested well in advance, "bufferized" by the Transceiver, and modified afterwards. This approach would be particularly concerned by the issue exposed in chapter 6 on the reference sources events identification. On the other hand, if *Absolute Time* mechanism with a shared time reference is used, the operations might provide interesting alternatives. In any case, the most likely parameter to be modified after activations requests is duration of the activation, the *Timing profile*, rather than the, carrier frequency, channelization or any other settings of the *Tuning profile*. Therefore, and in order to get rid of the ambiguities and unnecessary complexities regarding *Undefined* or immediate activations exposed in chapter 5, a very simple an elegant solution might be to replace the stop time parameter from the operation for a duration parameter. As a bonus this will not only avoid the confusions but also might simplify the programming and reduce the number of error cases of incoherent stop and start time values (e.g stop before start). These two previous remarks, lead to the proposal of new operations for conveying in essence the same type of information but organized in a different way. For example we could envision independent operations for *Event Based Time* and *Absolute Time*, together with a dedicated one enabling the waveform to provide activation durations at any given time (especially after the start time and during the activation, a very interesting option for synchronization or push-to-talk PTT modems). An additional complementary operation would convey all the *Tuning profile* information.

In platforms with separated baseband modem and Transceiver timing domains and in if the *Event Based Time* is wanted, the definition of a synchronization interface or set of operations devoted to synchronization issues appears as a mandatory upgrade. The solution adopted in this paper for the EWF could be considered as a workaround for a very basic problem. This new interface should allow both sides of the API to know the event occurrences. The resolution and accuracy of such an interface deserve an in-depth analysis and it will certainly also depend on the underlying communication mechanisms between modem and Transceiver (simple call if hosted by the same processing resource, transport layer such as CORBA for SCA architectures).

8. REFERENCES

[1] E. Nicollet "Transceiver Facility Specification" SDRF-08-S-0008-V1.0.0, January 28, 2009.

[2] Dr. J. Mitola "The software radio architecture", Mitre Corp. IEEE Communications magazine, May 1995.

[3] Dr. Taj A. Sturman "An Evaluation of Software Defined Radio" – Main Document, QinetiQ report for Ofcom, March 15, 2006.

[4] http://www.obsai.org/

[5] http://www.cpri.info/spec.html

[6] http://www.mipi.org/specifications/digrfsm-specifications

[7] E.Nicollet, L.Pucker "Standardizing a transceiver API for software-defined and cognitive radio systems" RF Design, February 1, 2008. http://preview.rfdesign.com/software_radio/signal_processing

[8] E.Nicollet "The Transceiver Facility Platform Independent Model (PIM) Definition, Content and Usage" SDR Forum Technical Conference '06, Orlando, November 2006.

[9] "E2R: End to End Reconfigurability" ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/cnt/e2rii_en.pdf

[10] "E3: End to End Efficiency" https://ict-e3.eu/

[11] http://www.euler-project.eu/

[12] JPEO JTRS, "Software Communications Architecture Specification", *version 2.2.2 / FINAL*, 15 May 2006.

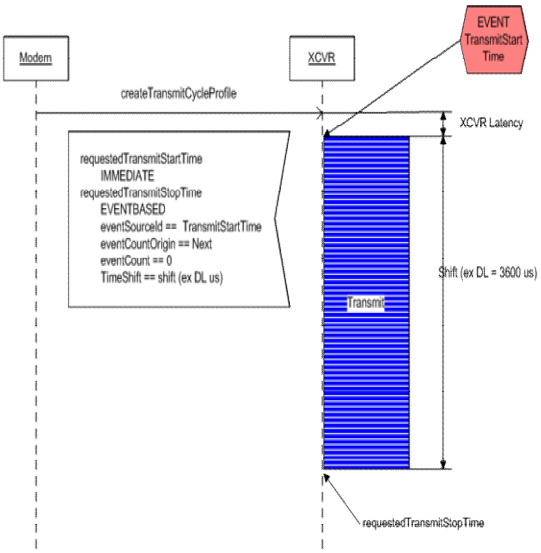

Figure 3: Use case 1 – BS first transmission. Transmission start time is *Immediate* and the stop time is requested using the *Event Based Time* with the `TransmitStartTime` as reference source and a time shift of DL for the duration (`EventCountOrigin` parameter value is not used actually).

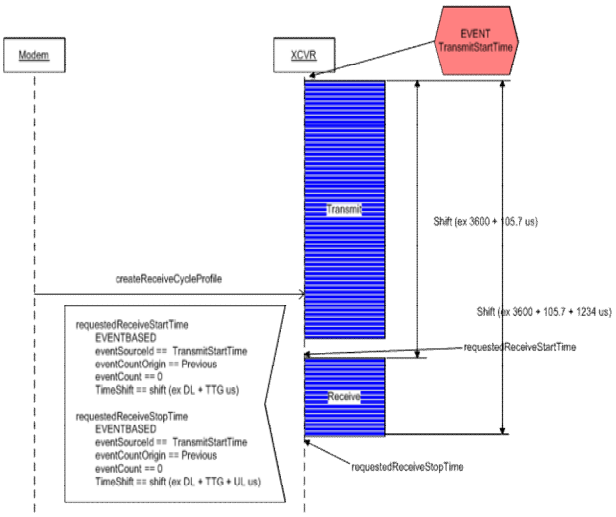

Figure 6: Use case 2 – BS first receive. The receiving activation start and stop times are indicated using the *Event Based Time* and the `TransmitStartTime` as reference event. For the very first reception since no previous reception was done the `TransmitStartTime` is used. (`EventCountOrigin` parameter value is not used actually).

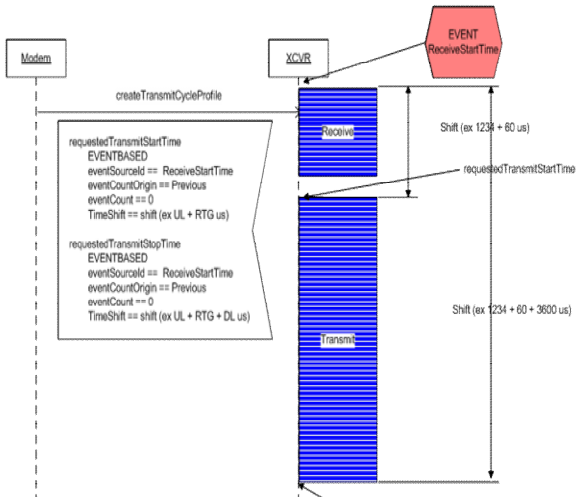

Figure 5: Use case 3 – BS second and further transmissions. The transmitting activation start and stop times are indicated using the *Event Based Time* and the `ReceiveStartTime` as reference event. (`EventCountOrigin` parameter value is not used actually).

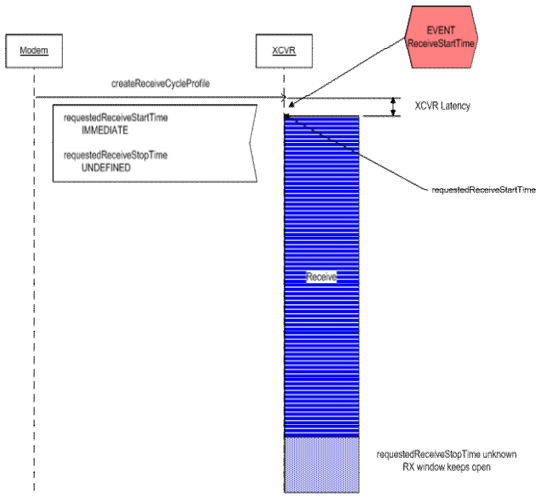

Figure 7: Use case 5 – First receive for synchronization procedure. Transmission start time is *Immediate* and the stop time is *Undefined* since the synchronisation acquisition time is yet unknown.