TRANSCEIVER FACILITY IMPLEMENTATION FOR A WIMAX-LIKE WAVEFORM

Eric Nicollet & Alejandro Sanchez
Thales Communications SA, Colombes, France

OUTLINE

- Introduction
- The EULER project and the EULER waveform
- The Transceiver Facility in EULER
- Transceiver Implementation architecture
- API Overview
- EULER waveform constraints
- Conclusions

INTRODUCTION

Waveform and Platform separation is a key SDR concept.

- Enhances portability and reconfigurability
- But… it is more difficult to apply to Radio signal chain RF Hardware

Several standards dealing with this are available

- Interfaces proposed by OBSAI, CPRI, DiGRF.
- Those are low level interfaces

Need for an alternative intermediate level interface

- The Transceiver Facility Specification tries to fill this gap
  - Original specification published on January 28, 2009 within the SDR Forum
  - The document creation was funded by French MoD and EU FP6-7 (E2R-II,E3)

The Transceiver Facility implementation is being carried out with EU funding as well: the EULER project

EULER project FP7 Securities theme project

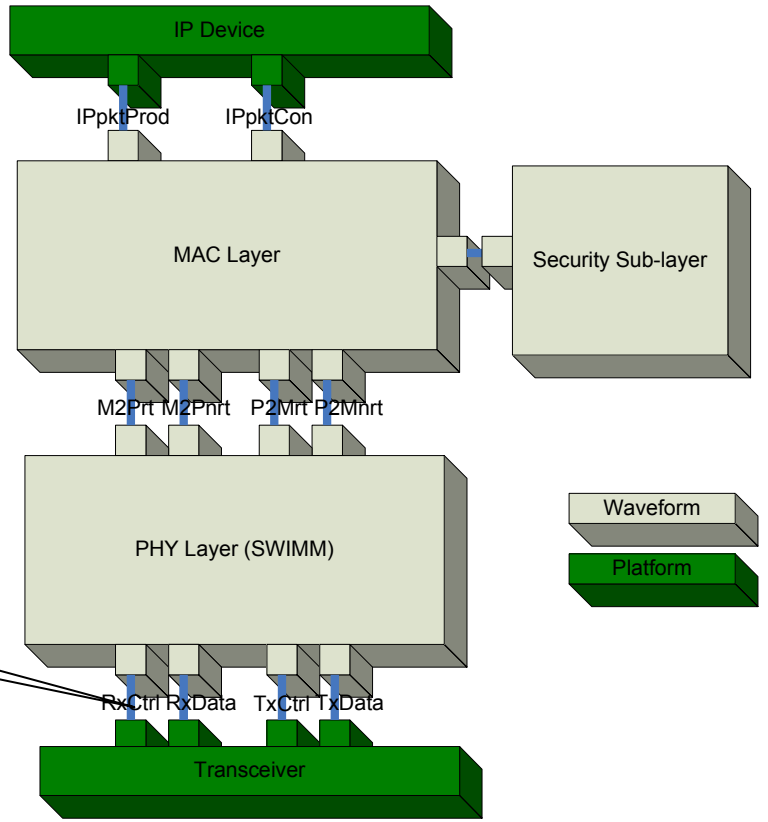
- focuses on the potential advantages brought by the SDR in international crises, disaster situations
- Requirement: fast deployment, high data rate for data/video services and interoperability among different first responders organizations belonging to different countries in border crisis scenarios
- One goal is the enforcement of the SDR business model of separated waveform and platform suppliers and the porting of the waveform on different platforms

EULER WAVEFORM (EWF)

EULER waveform is decomposed in three modules (SCA resources)

- PHY, MAC layers and SEC sub-layer
- each module is developed by an EULER consortium member
- interfaces definition is thereby an important task!

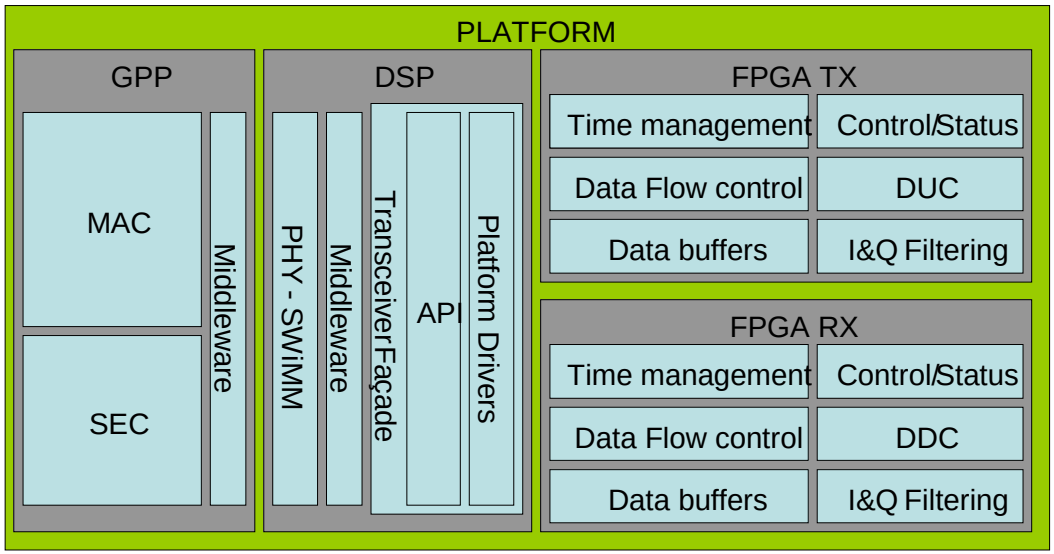
  - Transceiver Facility chosen for PHY<>Radio transceiver interface



TRANSCEIVER IMPLEMENTATION ARCHITECTURE

Implementation platform integrated by four processing resources: A GPP, a DSP and two FPGAs

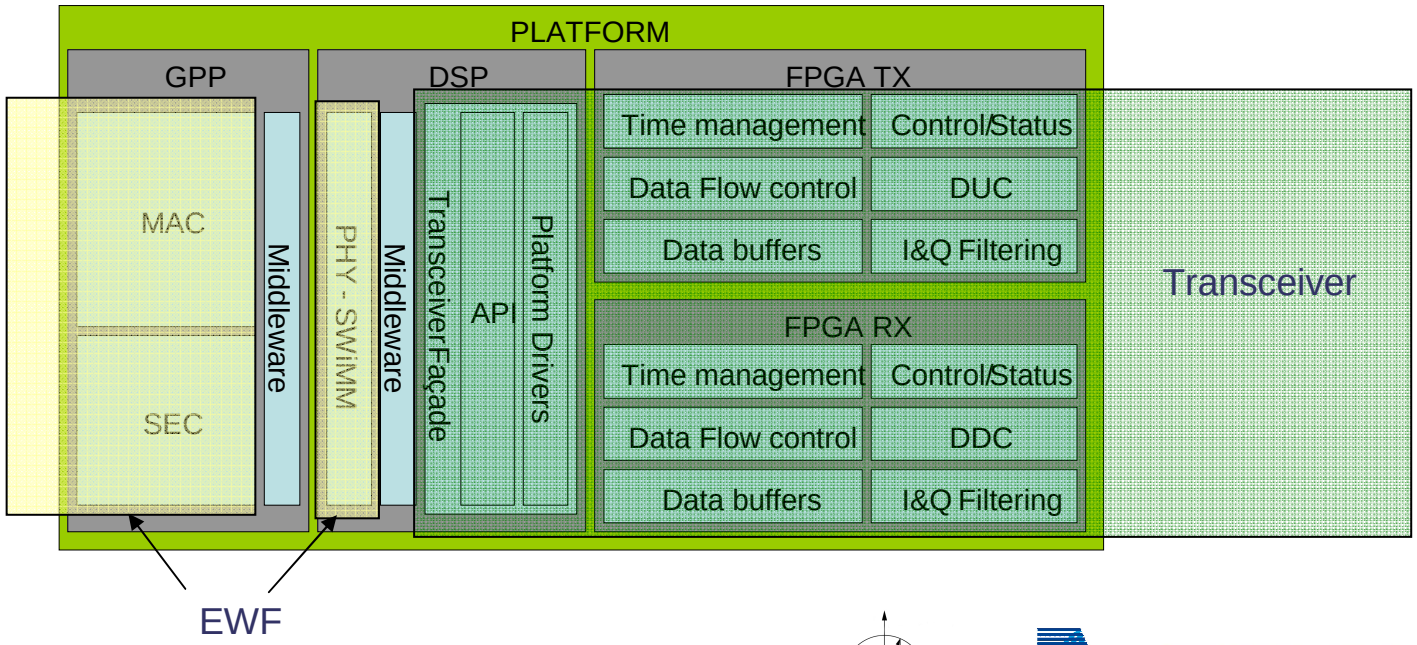
- EWF modules mapping was quite typical: MAC and Security processing hosted by the GPP and physical layer signal processing performed by the DSP.
- FPGAs key for transceiver implementation



TRANSCEIVER IMPLEMENTATION ARCHITECTURE

Implementation platform integrated by four processing resources: A GPP, a DSP and two FPGAs

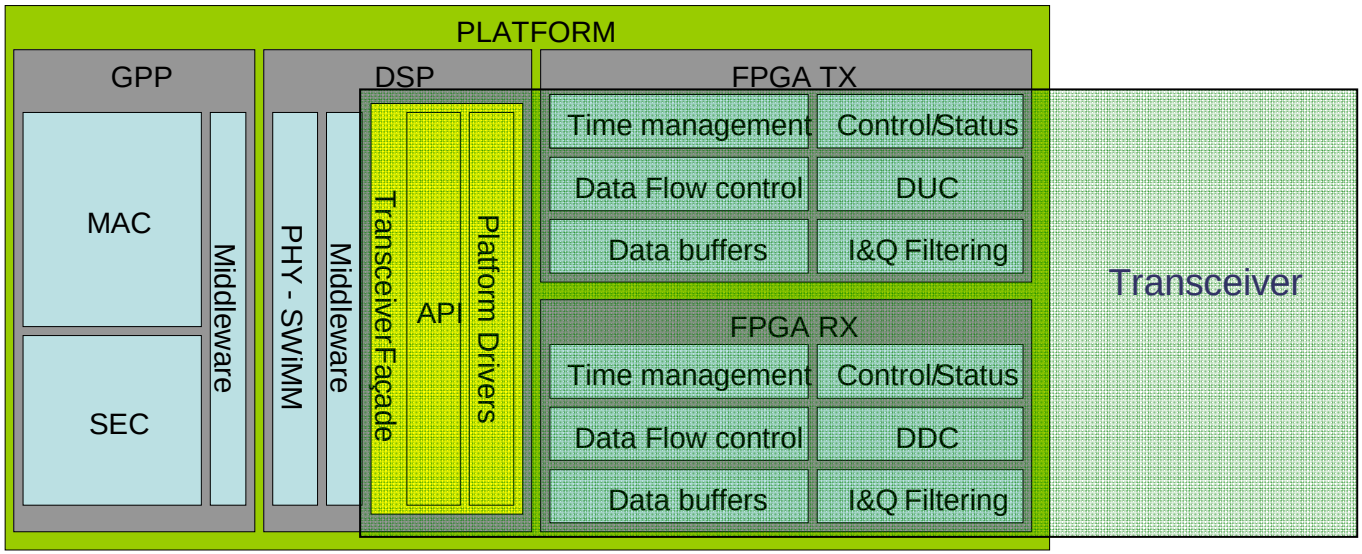
- EWF modules mapping was quite typical: MAC and Security processing hosted by the GPP and physical layer signal processing performed by the DSP.
- FPGAs key for transceiver implementation



TRANSCEIVER IMPLEMENTATION ARCHITECTURE

Implementation platform integrated by four processing resources: A GPP, a DSP and two FPGAs

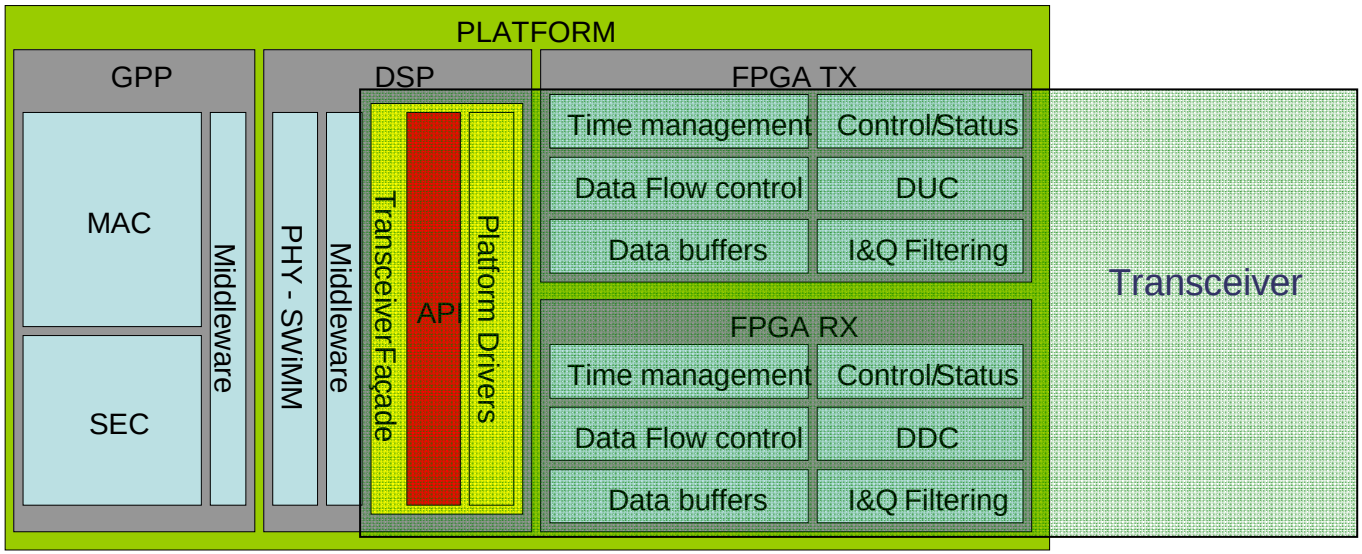
- EWF modules mapping was quite typical: MAC and Security processing hosted by the GPP and physical layer signal processing performed by the DSP.
- FPGAs key for transceiver implementation



TRANSCEIVER IMPLEMENTATION ARCHITECTURE

Implementation platform integrated by four processing resources: A GPP, a DSP and two FPGAs

- EWF modules mapping was quite typical: MAC and Security processing hosted by the GPP and physical layer signal processing performed by the DSP.
- FPGAs key for transceiver implementation



TRANSCEIVER FACILITY OVERVIEW

Transceiver API designed to be able to support as many waveforms as possible with a reduced set of operations and parameters:

- *Control* operations are dedicated to programming the transceiver for TXing and RXing with a *Timing Profile* and *Tuning Profile*
- *Data exchange* operations enable data samples exchanges (send/receive) between Modem and Transceiver

Time management mechanisms

- *Absolute Time* is intended for systems sharing a common time reference source
- *Event Based Time* is conversely aiming at platform systems in which only one side of the API is mastering the time

TRANSCEIVER FACILITY OVERVIEW

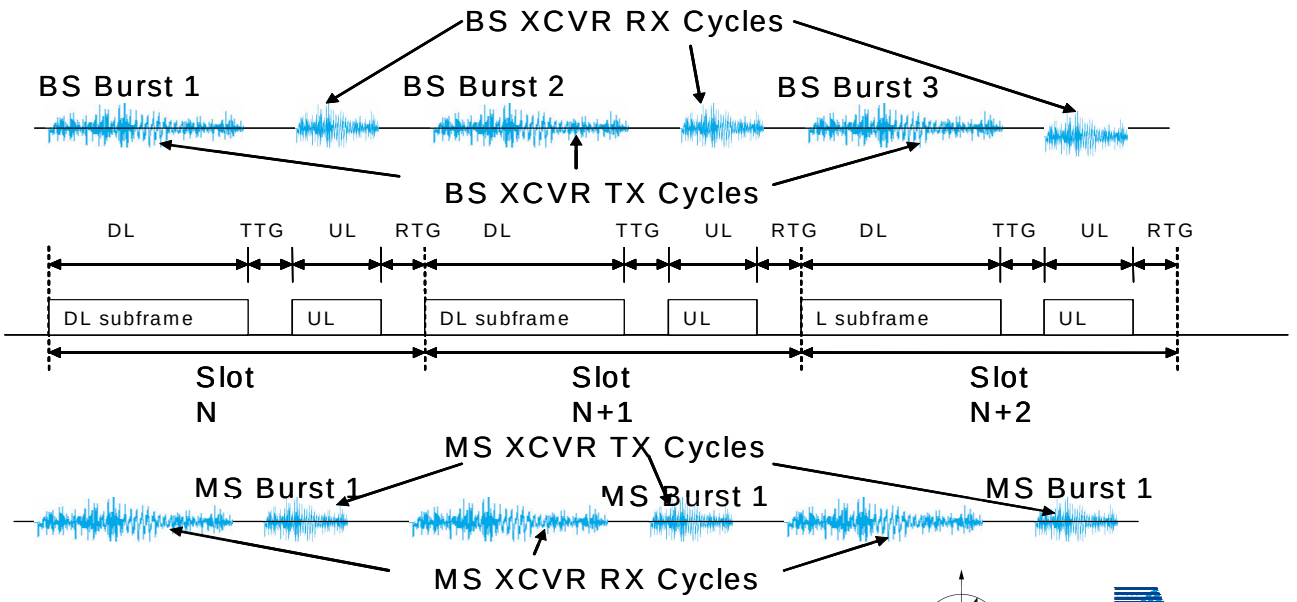
Undefined Time issue:

- to be used for TX and RX stop times to request for an immediate stop
- the definition appeared as misleading
  - apply to the parameters of the operations as a value?
  - new time management mechanism at the same level of the aforementioned two modes?
- 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 the document: *Immediate* and *Undefined*

SWIMM REAL-TIME CONSTRAINTS

SWiMM stands for SDR WiMAX Modem

- **simplified profile or sub-set of the IEEE802.16.e standard**
  - TDD waveform: fundamental frame alternating TXing and RXing slots and guard intervals in between
  - SWiMM 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**



SWIMM REAL-TIME CONSTRAINTS

High-level requirements:

- "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"
- MS case only: "Start receiving as soon as possible and for an undefined duration in order to get synchronized"

Further detailed analysis yields up to 9 use cases (both BS & MS):

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

DESIGN AND IMPLEMENTATION CHOICES

EULER Platform does not provide a common time reference source for the DSP hosting the SWiMM modem layer of the EWF and the FPGAs.

- *Event Based Time* mechanism was the choice of this design

***Event Based Time* is quite rich in functionalities:**

- 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.

DESIGN AND IMPLEMENTATION CHOICES

All the use cases could be supported by an event source and a time shift

In this design only two event sources were needed to cover completely all the use cases: *TransmitStartTime* and *ReceiveStartTime* (known by the Tranceiver not PHY)

The critical issue is the identification of these event sources and the way to accurately signal the event occurrences to the PHY SWiMM

- **main shortcoming of the current version of the specification: there is no mechanism, functionality or interface enabling the synchronization between a modem and Transceiver.**

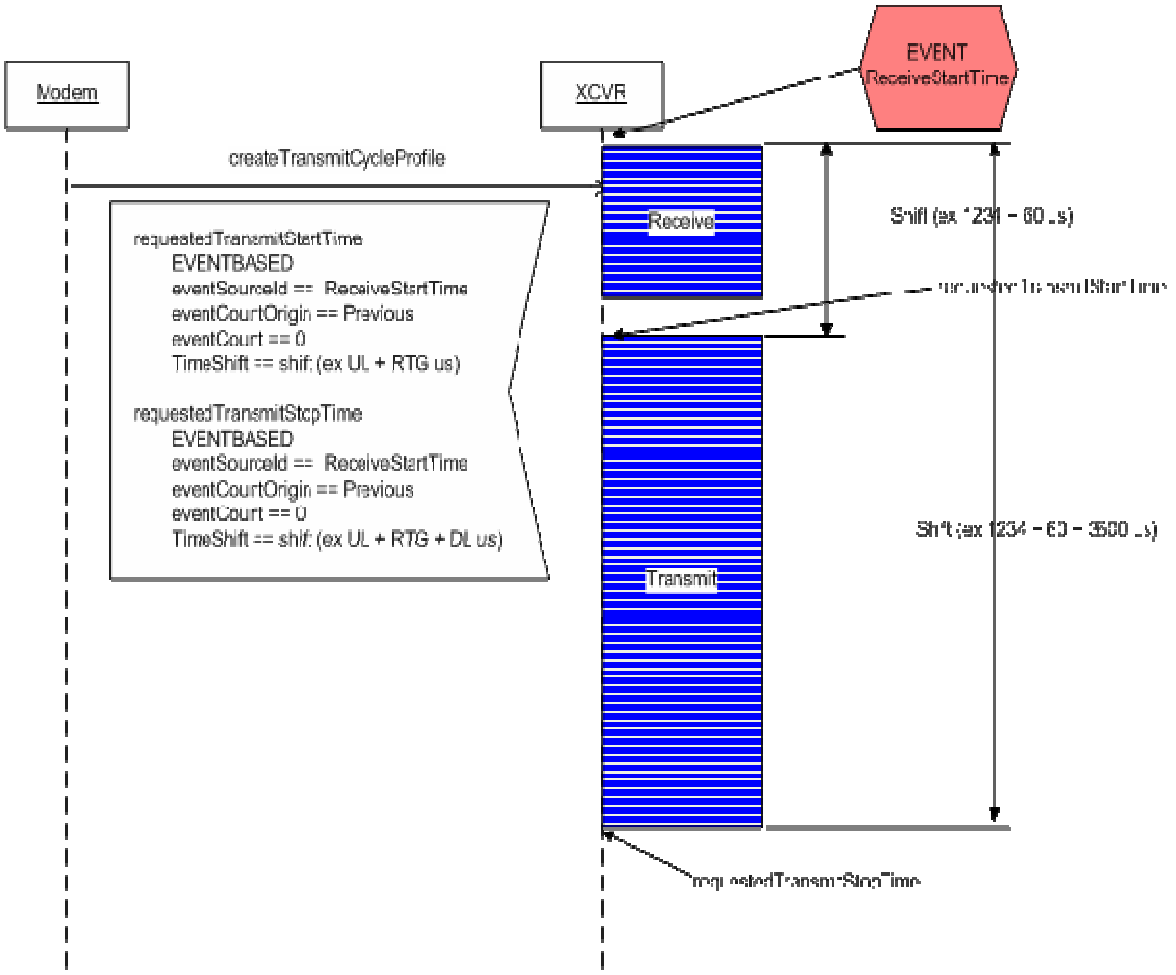
DESIGN AND IMPLEMENTATION CHOICES

Design took advantage of the TDD characteristics of the EWF waveform: use 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 then it is aware of the *ReceiveStartTime* occurrence
- This 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 before the next occurrence
- 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)

DESIGN AND IMPLEMENTATION CHOICES

Example



CONCLUSIONS

Following the implementation main enhancements for a new version could be:

- Even a more reduced set of operations (and related parameters is possible)
- in order to get rid of the ambiguities and unnecessary complexities regarding *Undefined* or immediate activations an elegant solution might be to replace the stop time parameter with a *duration* parameter
- 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
- Additional synchronization interface should allow both sides of the API to know the event occurrences