

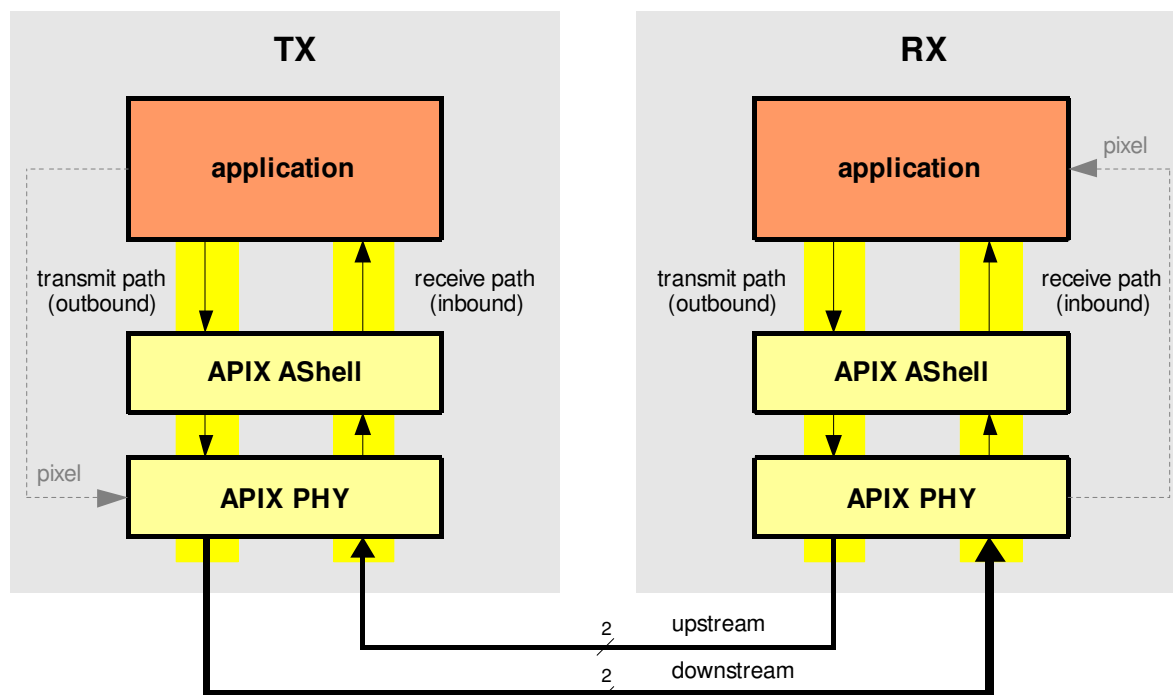
# Product Brief

## AShell – A Communication Protocol for Inova Semiconductors APIX Product line

The APIX Automotive Shell called the *AShell* is an abstraction layer for the APIX Automotive PIXel Link Interface from Inova Semiconductors.

It can be implemented as part of a system allowing a secure and error free data exchange on the bidirectional full duplex sideband channels of the standardized APIX link.

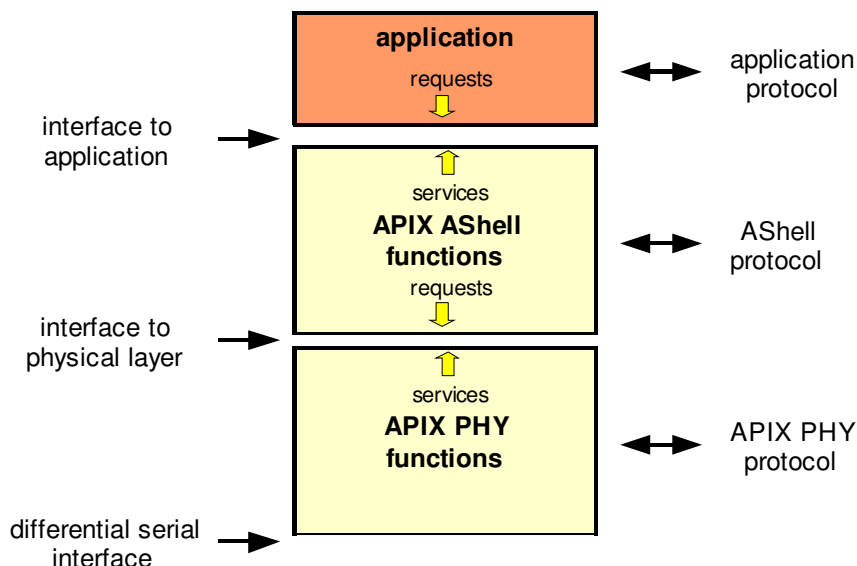
This can be done in Hardware or Software and is part of the APIX Interface standard, which is supported by a number of Chip suppliers such as Fujitsu, Toshiba and Xilinx and of course Inova Semiconductors APIX product family INAP125xxx.



Picture 1: APIX based communication system

## Overview

An application, AShell and the APIX Physical Layer form a stacked bidirectional wire-based communication system (picture 1). As part of this communication stack, AShell provides two interfaces for the exchange of messages from local to a remote island or vice versa. AShell hides implementation details and specific features of APIX Phy and provides an appropriate interface to the embedding application of the APIX serial link. This encapsulation provides for higher levels of abstraction and supports a functional layered architecture of the communication system. A pixel (video) data stream of an application to be transmitted from TX to RX device is bypassed and thus not affected by the AShell layer. As with most layered communication architectures, a collection of related **functions** is referred to as **layer** that provides **services** to the layer above and receives services from the layer below (picture 2). Each layer has **interfaces** to the layers above and below. In general, each layer communicates by the means of a **protocol** with its counterpart on the remote island.



**Picture 2:**

## AShell Services

Apart from the error control functions added to the APIX communication stack, the AShell is more or less a wrapper layer.

The AShell provides the following services:

- transmission of application data ensuring data integrity
- reception of application data ensuring data integrity
- supply of information about transmission link status as well as simple error statistics

## AShell Functions

To provide the aforementioned services, AShell implements the following functions:

### 1. Transaction framing and de-framing

Payload data sinked to or sourced by an application is compiled to or extracted from Protocol Data Units (PDU) transactions exchanged as protocol entities between local and remote AShell.

### 2. Data exchange through APIX Phy

AShell handles various interface signals from APIX Phy such that data is sent and received via APIX Phy in any of the operation modes provided by the embedded or external APIX Phy.

### 3. Data integrity control

AShell implements a CRC-24 polynomial to detect most transmission errors. Only valid transactions are offered to the application. The transmit path of the AShell generates and the receive path checks the CRC sum that is part of the Protocol Data Unit (PDU) exchanged between the AShells of both communicating islands.

### 4. Error control (optional)

AShell implements a window based ACK protocol to manage bit errors occurring during serial transmission over the wire-based link. As long as the bit errors do not affect synchronization at different levels, communication is kept alive without any intervention by the application. This function can be disabled to support different requirements of the application or software-based implementations of AShell services.

### 5. Status report

Information about the status of APIX Phy and the serial transmission link as well as simple error statistics are collected and presented to the application at the interface.

## 6. Convenient wrapping of the APIX Phy interface

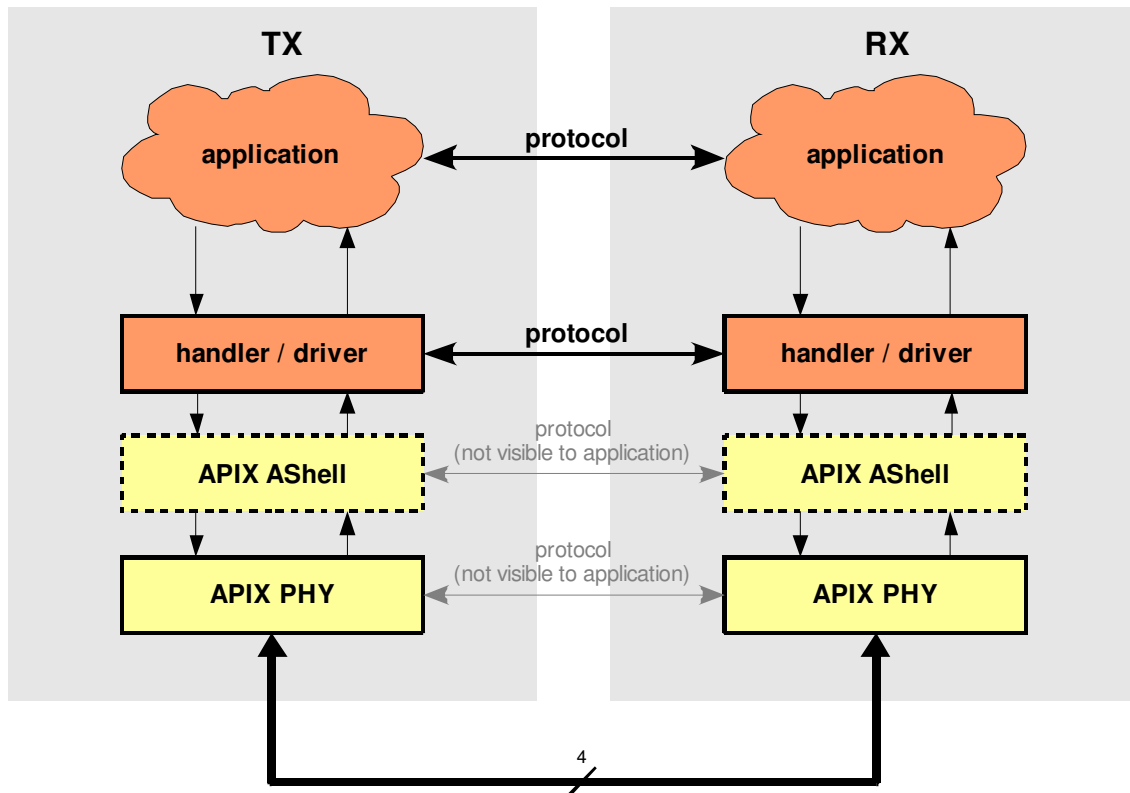
AShell provides an appropriate broker between APIX Physical Link Layer and the respective application operating in different clock domains. AShell thus hides implementation details and special features of APIX Phy.

## 7. Configuration of APIX Phy and AShell (optional)

If AShell is part of an embedded APIX communication stack, i.e. AShell and Phy are integrated in one device, the complete and consistent configuration interface, both for AShell and for APIX Phy, is exposed to the application at the AShell interface.

## Application Interface

An APIX compliant serial transmission system does not necessarily require or include the AShell layer. If functions provided by AShell are not needed, APIX Phy compatible implementations of the serial transmission system on both the local and remote island are the only requirement. To enhance interoperability, however, abstraction based on the AShell specification simplifies the interfacing of applications to APIX Phy.



**Picture 3: Application centric view of APIX communication stack**

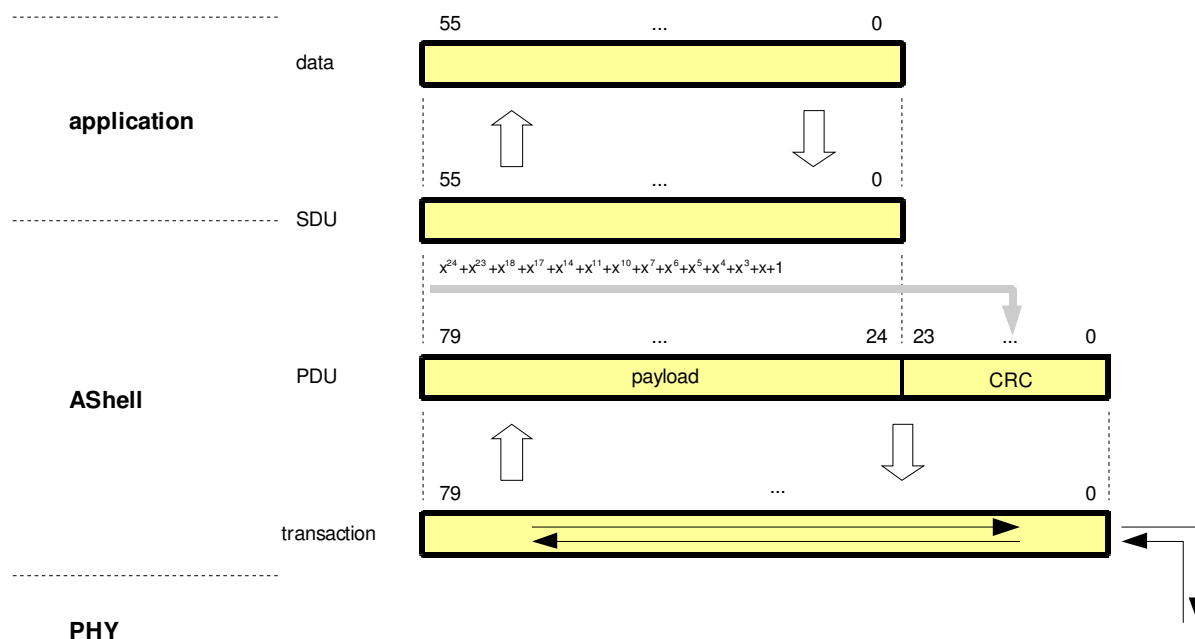
For example, an **application handler** wraps bus transactions of a local Microcontroller to the hardware protocol required by AShell and vice versa. The AShell resources could thus be mapped into the address space of the surrounding hardware where AShell is embedded. The timing and dependencies of existing interface signals are translated as required.

As an option, a handler or driver as part of the application can implement locally mapped (mirrored) resources of the remote island (e.g. flags, interrupt request lines, register...). Furthermore, prioritized scheduling of concurrent pending requests for data transmission over the APIX link can be implemented by this agent if required.

This handler, usually implemented as hardware, provides for maximum flexibility in mapping the requirements of the application to the generic AShell interface. For example, if an application has to read addresses from the remote I/O space every time a specific remote interrupt is received, an appropriate implementation of the handler might autonomously collect the data to be read and send the remote interrupt notifications right along with this data. This reduces the resulting communication overhead.

## Principle of Operation

If an application indicates valid payload data (AShell SDU) at the transmitting path (outbound direction), this data is captured crossing clock domains and a transaction (AShell PDU) is compiled. The 24 bit wide CRC is computed over the 56 bit wide payload provided by the application. The resulting CRC sum is appended to the Service Data Unit (SDU) forming the 80 bit wide Protocol Data Unit which forms the AShell transaction (picture 4).



**Picture 4: Communication stack data flow**

At the receiving path (inbound direction), the CRC sums of incoming transactions are checked to identify corrupted transactions. Only the payload data of valid transactions is delivered as Service Data Unit to the inbound data port of the application.

All transactions are transmitted in the order of their compilation. A transaction is transmitted only after a preceding transaction was sent (delivered to APIX Phy). *Protocol control transactions* (exchange of control data between local and remote AShell) with highest priority are inserted between completely transmitted *payload*

*transactions*. When a transaction is compiled and handed over to the Phy for serialization and transmission and no new data is provided by the application at its interface to AShell, a so-called *empty control transaction* is inserted in the transaction stream.

Control transactions (*empty or protocol control transactions*) are processed by the receiving path of AShell and are not handed over to the application.

Serialization and transmission of transaction data start at the least significant bit (LSB) position and progress in the direction of the most significant bit position (MSB). The rightmost bit is referred to as LSB, and the leftmost bit marks the MSB.

Service Data Units (SDU) accepted by AShell are transmitted until successful reception is acknowledged by the remote counterpart. The acknowledge mechanism is part of the protocol defined by the AShell layer. In case of transmission errors detected through unmatched CRC sums, all transactions of a certain sliding window including the affected transaction(s) are resent without any intervention by the application. In any case, the order of transactions accepted for transmission is maintained.

From a layered communication stack architecture point of view, there are several levels of synchronization which have to be established and maintained for proper communication (Table 3).

Layer/module	Synchronization at level	Purpose
AShell	Transaction	Alignment to transaction boundary
Phy	Serial frame	Alignment to frame boundary
	Serial bit	Data/bit recovery

**Table 3: Level of synchronization**

If synchronization at one of the levels is lost or if a programmable number of consecutive corrupted transactions is exceeded, automatic repetition of transactions is terminated and the AShell indicates this error condition. Every time AShell is unable to comply with its offered service to deliver accepted transactions to the remote site without bit errors, the violation of this service is indicated. The application can enforce a restart of the AShell protocol engine, which results in an abort of all transmissions and the repetition of transactions, enforcing resynchronization at transaction level.

Special *control transactions* are used to perform synchronization at transaction level. The alignment is based on special magic bit patterns and the transaction frame with a matching CRC sum. During synchronization, both sites generate a special synchronization protocol.

AShell can operate in several operation modes which are not mutually exclusive:

- **Compatibility mode**

The interface between AShell and Phy is configured such that interworking with the chip devices INAP125xxx from Inova Semiconductors is possible.

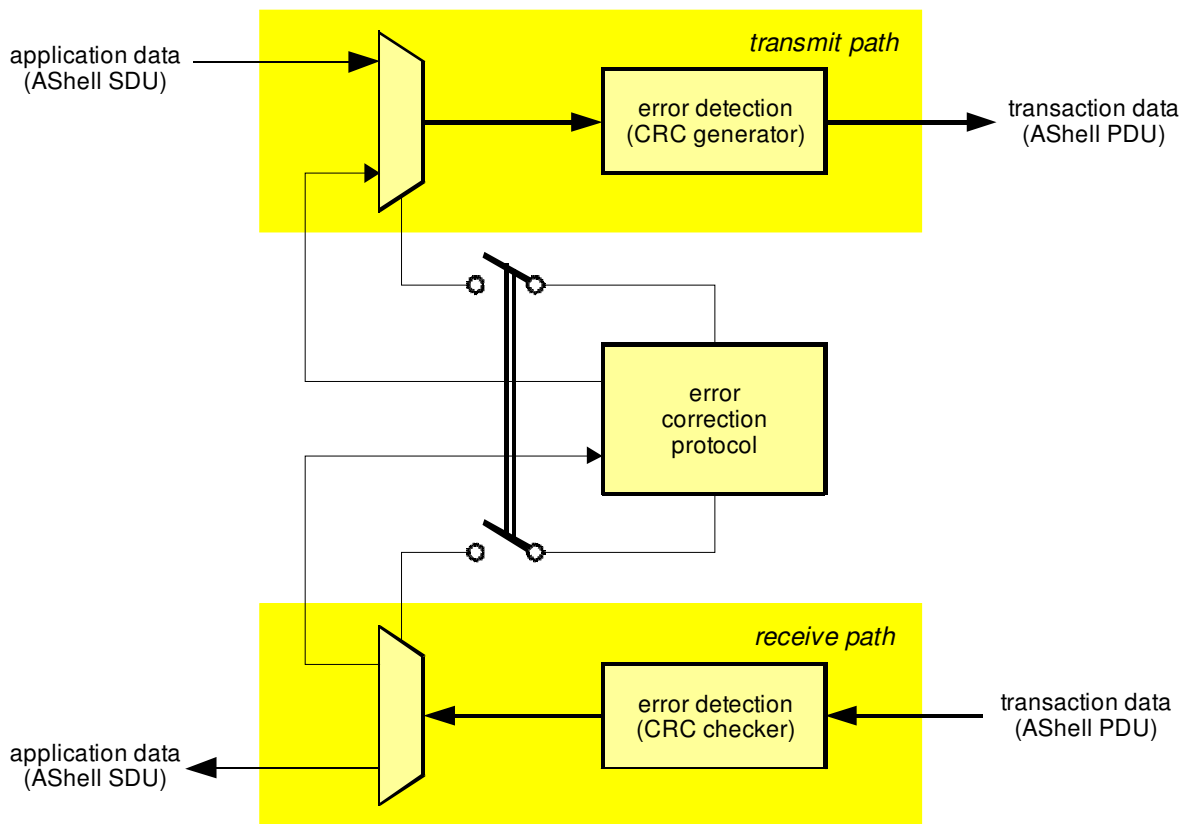
With the same configuration, AShell can cooperate with any external (embedded) APIX Phy connected via I/O pad cells.

- **Integrity only mode**

The error control function of AShell can be limited to control data integrity by CRC generation and check. In this case, the windowed positive acknowledge protocol controlling the repetition of transactions is disabled (picture 5).

- **Payload only mode**

For special implementations with AShell functions implemented by software, the transmission of *empty control transactions* can be suppressed to reduce the resulting processor load.



**Picture 5: AShell data integrity and error control functions**

**For more information contact**

[info@inova-semiconductors.de](mailto:info@inova-semiconductors.de)

[www.inova-semiconductors.de](http://www.inova-semiconductors.de)