White Paper

AFDX/ARINC 664
Concept, Design, Implementation and Beyond

By Detlev Schaadt, CTO, SYSGO AG
**Whitepaper AFDX/ARINC 664 – Concept, Design, Implementation and Beyond**

**Introduction**

“Avionic Full-Duplex Switched Ethernet” (AFDX), designated ARINC 664, is a specification for a deterministic aircraft data network bus for aeronautical, railway and military systems. The network is based on standard IEEE 802.3 Ethernet technology. The benefits from using commercial-off-the-shelf (COTS) Ethernet components include reduced overall costs, faster system development and less-costly maintenance for the system network. Hardware components, cables and test equipment for Ethernet are field proven and much more affordable than “built-to-spec” avionics solutions. Standard commercial grade Ethernet won’t meet avionics network requirements. Therefore, AFDX extends the Ethernet standard by adding Quality of Service (QoS) and deterministic behavior with a guaranteed dedicated bandwidth. AFDX™ is currently used in the Airbus A380 and A400M as well as in the Boeing 787 Dreamliner.

**History of Avionic Data Busses**

Aircraft control systems generally consist of a number of sensors to read environmental or inertial data, an avionic system performing a certain flight relevant control function and outputs, for instance to control actuators to perform rudder or flap movements. There always has been a need to interconnect between these components and traditionally, a set of sensors and actuators have been connected to one avionics function. The main data busses used for these purposes where ARINC-429 and MIL-STD 1553. The following sections will give a brief overview of these technologies.

**ARINC-429**

ARINC-429 is a two-wire data bus that is application-specific for commercial and transport aircraft. The connection wires are twisted pairs. Words are 32 bits in length and most messages consist of a single data word. The specification defines the electrical and data characteristics and protocols. ARINC 429 uses a unidirectional data bus standard (Tx and Rx are on separate ports) known as the Mark 33 Digital Information Transfer System (DITS). Messages are transmitted at either 12.5 or 100 Kbit/s to other system elements that are monitoring the bus messages. The transmitter is always transmitting either 32-bit data words or the NULL state. No more than 20 receivers can be connected to a single bus (wire pair).

**MIL-STD-1553**

MIL-STD-1553 is military standard published by the United States Department of Defense (DoD) that defines the mechanical, electrical and functional characteristics of a serial data bus. It was originally designed for use with military avionics, but has also become commonly used in spacecraft on-board data handling (OB DH) subsystems, both military and civil. It features a dual redundant balanced line physical layer, a (differential) network interface, time division multiplexing, half-duplex command/response protocol and up to 31 remote terminals (devices). The communication speed rated up to 1 Mbit per second. All communication on the bus is initiated by the bus controller, which polls the remote terminals to send or receive data. It was first published as a U.S. Air Force standard in 1973, and was first used on the F-16 Fighting Falcon fighter aircraft. It is widely used now by all branches of the U.S. military and has been adopted by NATO as STANAG 3838 AVS.

**IMA requires new communication capabilities**

A number of innovations and changes are delivering new capabilities to aircraft operations. Modern aircrafts are equipped with numerous electronic components. Some of them – like flight control and guidance systems – provide flight critical functions, while others may provide assistance services that are not critical to maintain airworthiness, but reduce the workload of the crew. The amount of capabilities is increasing, and so is the amount of information that needs to be processed and displayed.

The industry more and more requires a flexible and scalable standard architecture to support the broad spectrum of capabilities and performance with standardized interfaces to fit a wide range of aircrafts.

Simultaneously a need exists to improve factors like system dispatchability, reliability and maintainability while – at the same time – reducing the physical dimensions of size, weight, number of connectors, etc. To reduce lifecycle and construction costs, the usage of common system building blocks with high levels of reuse is aimed at.

Consequently, the aircraft manufacturers as well as the main function suppliers for avionic subsystems have been thinking about a new architecture for avionic systems, providing a standardized hosting platform for avionic applications. The idea of “Integrated Modular Avionics” (IMA) was born and it is quite simple:

- A general purpose computing platform provides the computational power
- The interfaces to sensors and actuators are standardized and provided on separate I/O boards
- The computing platform provides a real-time operating system (RTOS) with a standard interface (API) to avionic applications

The IMA approach introduces a number of advantages over the traditional solutions, as resources can now be shared, computational power can be added to the system, if growth concepts requires additional performance. Standardized platform software provides standard functionality common to all subsystems, like maintenance functions, data loading functions, BITE, etc. The IMA concept goes as far as allowing multiple avionic applications to execute on one computing module in so called partitions. To retain fault containment within partition boundaries, the underlying operating system is required to provide strict partitioning of CPU-time and other resources, like memory, common I/O channels, or communication paths.

While most of the avionic software applications still anticipate data communication and distribution over ARINC-429, a new network infrastructure was needed to reflect the new flexibility in aircraft system design and cope with the mere number of interconnections between avionic subsystems.
AFDX, the next generation avionic data network

The solution to the new challenge is to use commercially proven hardware base technology and apply a protocol to it, capable of delivering the reliability in packet transport and timing. Asynchronous Full Duplex Switched Ethernet (AFDX) is based on standard IEEE 802.3 10/100 Mbit Ethernet hardware. This is a well known standard and it is widely used. The packets are enveloped as IP and UDP packets, a software protocol known from the internet. However, the base requirements for a network for avionics applications are not covered by standard technology:

- reliable packet transport
- bounded transport latency

Special extensions to the network protocol and a fixed topology are needed to meet these requirements. The AFDX protocol has been standardized by an ARINC working group. ARINC-664 describes aircraft data networks (AND) and part 7 specifically deals with AFDX.

Flexible Network Topology

Normal Ethernet technology uses a topology where each node is treated equally. If a peer wants to transmit a packet on the network and the media is occupied, a collision occurs, the peer backs up and tries again, until the transmission is successful, or a predefined amount of time has elapsed. This behavior introduces variable length latency, which is not tolerable in safety critical applications. In order to avoid collisions on the network, a switched full duplex topology is introduced.

Here, two separate Ethernet controllers receive the frame and a redundancy management algorithm in the protocol layer forwards at most one of the two identical packets appropriately to upper layers.

Physical redundancy can be configured on a per connection basis, allowing less data links with fewer requirements for availability to utilize the additional bandwidth offered by the second data link.

Sub Systems, AFDX End Systems and Switches

An AFDX network consists of so called End Systems and switches. An End System is a component connected to the AFDX network and capable of handling all AFDX related protocol operations. Usually, an End System is part of an avionic or aircraft subsystem, which needs to send or receive data over the AFDX network. One or more switches, depending on the network hierarchy, are located on the data path between two End Systems.

In order to decouple transmit operation from data reception all data paths use separate data buffers, thus creating true full duplex data links between End Systems. While this enables data communication using full wire speed between End System end AFDX switch, it becomes obvious that the bottleneck appears at the connection between switches. Here, all traffic appears that has to leave the switch and the sum of the individual transmission rates may easily be larger than the capacity of the inter switch data link.

It has been mentioned previously that AFDX – at the application level – is intended to replace ARINC-429 connections. With ARINC-429 representing point-to-point or point-to-multipoint connections, it is not surprising, that AFDX has the same characteristics. The point-to-point and point-to-multipoint connections are represented by virtual links (VL). A single VL may connect exactly two End Systems, in which case it represents a point-to-point connection. It may also connect one sending End System with multiply reading End Systems, in which case it represents a point-to-multipoint (multicast) connection. The advantages of this architecture lie in the fact that AFDX presents itself compatible at the application level and saves a large amount of cable runs by multiplexing many individual VLs onto a single wire connection, utilizing the increased bandwidth of a 100Mbit Ethernet connection. The VL-bundle is demultiplexed at the destination switch and forwarded to the appropriate End Systems.

Many protocol features that have been introduced to IP or even TCP layers have to do with the fact that a peer to peer connection via
Internet is inherently unsafe. Often, the route between peers is not known up front, it may even change during the session. Large packets are fragmented and re-assembled on their way from sender to recipient and sometimes, packets are received out of order. An avionics real-time capable network like AFDX is not exposed to many of these scenarios. In principle, all network parameters are known and constant and the resulting IP layer can be a lean implementation, free of fallback and retry algorithms. Key parameters are the fact that packet fragmentation is indeed possible to allow for packets larger than the MTU at application level. However, the network guarantees all packets to be ordinal integer.

Host Communication API

In section 1 it was mentioned that the AFDX technology has to be viewed in the context of IMA technology. The industry has also standardized on an API for IMA Systems; the name of this standard is ARINC-653 APEX. This API defines so called Ports as only means for an application to exchange data with entities outside their own context. This entity may be a local device driver accessing local hardware, or an endpoint of a data channel to a remote computer. The advantages of such an approach are obvious:

- The application uses a unified API to exchange data.
- The Channel is configured independently from the Ports at the module integration level. Thus, a change in channel configuration is transparent to the application using the Port.

The ARINC-653 APEX defines Queuing Ports and Sampling Ports.

Queuing Ports

A Queuing Port is message oriented; a queue with a preconfigured depth accepts messages until all space is occupied. Further attempts to place a message in the queue will result in the application to wait until there is room for data, if the Port operates in blocking mode. A single source Queuing Port can only connect to exactly one destination port (1:1 relation).

Sampling Ports

In principle, a Sampling Port is a Queuing Port with a queue depth of one message. Consequently, a new message overwrites an old message, if the message has not been consumed yet. The configuration allows specifying a so called refresh rate. The age of a message is compared against the refresh rate and depending on the result a read service call returns the data’s validity. It is possible to connect one source port to multiple destination ports (1:n relation). The port API supports functions to open and write to source ports and functions to open and read from destination ports. Service calls operating on sampling ports never block the caller.

Service Access Point (SAP) Ports

While ARINC-653 defines two different port types, a third type exists for AFDX, the Service Access Point (SAP) port. In principle, this port type is defined to be a Queuing Port which makes additional parameters available to the API. The requirement for such a SAP port is the ability to communicate with a legacy TCP/IP based network (Compliant Network). Such communication is used for file transfers to file servers or to update an avionic system with new software. In order to communicate with a Compliant Network, a transmitting ES has the ability to specify the destination address: IP address and port number. For this purpose, these addresses are made available when the ES receives a request from the compliant network.

The exact API for the SAP port is not specified in the standard; it is however appropriate to use a subset of the well known Berkley UNIX Socket API.
One of the most desirable features of an ARINC-429 connection was the fact that it represented a private line between sender and recipient(s) of data. The physical bandwidth of the connection was available at all times, and no concurrent transmit requests occurred. This degree of separation is mandatory when interconnecting avionic systems with different levels of criticality. The AFDX technology achieves this separation by introducing the concept of Virtual Links (VL). Each logical connection is represented by a Virtual Link, thus providing the same properties as a legacy ARINC-429 connection: a unidirectional private line with bounded latency and guaranteed bandwidth. The identification of a given VL is accomplished by using 16 Bit of the 48 Bit MAC address as VL id. Since the IEEE 802.3 standard allows multicast packet transmission on the Ethernet it is possible to use the standard mechanism to allow a packet to be routed to multiple destination nodes. One of the properties of AFDX however is the fact that a VL may have one and only one End System sending into it.

The bandwidth control is achieved by the timing constraints associated to each VL. The mapping of the VL into the global switched network realizes a partitioning of the topology; each VL has an associated time fragmentation called BAG (Bandwidth Allocation Gap) and a maximum frame size. The BAG associated to each VL is the precise time unit between two consecutive IP frames sent by the End System. A VL may not use all the bandwidth, i.e. the time between two consecutive packets may be larger than BAG. However, it may not be smaller than BAG and it is the responsibility of the End System to enforce this. The ARINC-664 specification allows for BAG values in the range of 1…128ms in units of 2^n (BAG = 2^n, n = {0…7}).

When outbound packet streams from multiple VLs are merged for transmission, it is possible that more than one VL has a packet ready and eligible for transmission, due to equal or multiple BAG values. This is a legal case and the packet scheduler needs to serialize the packets, thus introducing a queuing delay (jitter) to the BAG.

The following graphic illustrates the effects of the traffic regulation for a single virtual link. No frame will be transmitted while a VL is eligible but has no data for transmission.

When outbound packet streams from multiple VLs are merged for transmission, it is possible that more than one VL has a packet ready and eligible for transmission, due to equal or multiple BAG values. This is a legal case and the packet scheduler needs to serialize the packets, thus introducing a queuing delay (jitter) to the BAG.

The bandwidth control is achieved by the timing constraints associated to each VL. The mapping of the VL into the global switched network realizes a partitioning of the topology; each VL has an associated time fragmentation called BAG (Bandwidth Allocation Gap) and a maximum frame size. The BAG associated to each VL is the precise time unit between two consecutive IP frames sent by the End System. A VL may not use all the bandwidth, i.e. the time between two consecutive packets may be larger than BAG. However, it may not be smaller than BAG and it is the responsibility of the End System to enforce this. The ARINC-664 specification allows for BAG values in the range of 1…128ms in units of 2^n (BAG = 2^n, n = {0…7}).

When outbound packet streams from multiple VLs are merged for transmission, it is possible that more than one VL has a packet ready and eligible for transmission, due to equal or multiple BAG values. This is a legal case and the packet scheduler needs to serialize the packets, thus introducing a queuing delay (jitter) to the BAG.

The following graphic illustrates the effects of the traffic regulation for a single virtual link. No frame will be transmitted while a VL is eligible but has no data for transmission.

When outbound packet streams from multiple VLs are merged for transmission, it is possible that more than one VL has a packet ready and eligible for transmission, due to equal or multiple BAG values. This is a legal case and the packet scheduler needs to serialize the packets, thus introducing a queuing delay (jitter) to the BAG.

The bandwidth control is achieved by the timing constraints associated to each VL. The mapping of the VL into the global switched network realizes a partitioning of the topology; each VL has an associated time fragmentation called BAG (Bandwidth Allocation Gap) and a maximum frame size. The BAG associated to each VL is the precise time unit between two consecutive IP frames sent by the End System. A VL may not use all the bandwidth, i.e. the time between two consecutive packets may be larger than BAG. However, it may not be smaller than BAG and it is the responsibility of the End System to enforce this. The ARINC-664 specification allows for BAG values in the range of 1…128ms in units of 2^n (BAG = 2^n, n = {0…7}).

When outbound packet streams from multiple VLs are merged for transmission, it is possible that more than one VL has a packet ready and eligible for transmission, due to equal or multiple BAG values. This is a legal case and the packet scheduler needs to serialize the packets, thus introducing a queuing delay (jitter) to the BAG.
quickly. The Simple Network Management Protocol (SNMP) has been developed as a common network standard to manage large networks and its components.

SNMP is based on the manager/agent model consisting of a manager, an agent, a database of management information, managed objects and the network protocol. The manager provides the interface between the human network manager and the management system. The agent provides the interface between the manager and the physical device(s) being managed.

**Figure 12: SNMP is based on the manager/agent model of a network management architecture.**

The manager and agent use a Management Information Base (MIB) and a relatively small set of commands to exchange information. The MIB is organized in a tree structure with individual variables, such as point status or description, being represented as leaves on the branches. A long numeric tag or object identifier (OID) is used to distinguish each variable uniquely in the MIB and in SNMP messages.

SNMP uses five basic messages (GET, GET-NEXT, GET-RESPONSE, SET, and TRAP) to communicate between the manager and the agent. The GET and GET-NEXT messages allow the manager to request information for a specific variable. The agent, upon receiving a GET or GET-NEXT message, will issue a GET-RESPONSE message to the manager with either the information requested or an error indication as to why the request cannot be processed. A SET message allows the manager to request a change be made to the value of a specific variable in the case of an alarm remote that will operate a relay. The agent will then respond with a GET-RESPONSE message indicating the change has been made or an error indication as to why the change cannot be made. The TRAP message allows the agent to spontaneously inform the manager of an important event.

The small number of commands used is only one of the reasons SNMP is “simple.” The other simplifying factor is its reliance on an unsupervised or connectionless communication link. This simplicity has led directly to its widespread use, specifically in the Internet Network Management Framework. Within this framework, it is considered robust because of the independence of the managers from the agents, e.g. if an agent fails, the manager will continue to function, or vice versa.

**Implementing AFDX**

The AFDX technology has successfully been deployed in the Airbus A380 program; the fact that newer programs like A400M or Boeings 787 Dreamliner are following this path shows that this technology will emerge into subsequent aircraft development programs. In principle there are two ways to construct a general purpose End System for AFDX:

On the A380 all End Systems are built around ASIC technology, which means that the AFDX protocol has been implemented as hardware, using programmable logic devices. Such an approach inherently has higher performance capabilities and can be easier to certify. On the other hand, the two main technology integrators – Airbus and Boeing – have already created significant deviations from the ARINC-664 standard and certain flexibility is required to accommodate to these differences.

An alternate method to build an AFDX End System is to use an embedded CPU and design the necessary hardware around it, i.e. memory, two Ethernet controllers, communication interface to avionic. In this case the complete AFDX protocol stack runs as firmware on the embedded CPU. In general, it is necessary to have the AFDX protocol implemented on a separate hardware to offload the main CPU from the timing and performance requirements of the AFDX protocol. The host CPU has real-time requirements allocated to it, and therefore the effects of network traffic must be decoupled from it. However, there may be LRU designs which have only little AFDX performance requirements. For such systems, it is feasible to implement the software stack and the application software on a single CPU.

Such design freedom already shows the main advantage of a software implementation of the AFDX protocol: the flexibility in design is outstanding compared to a hardware implementation. It has been mentioned that the standard already faces deviations introduced from real-world implementations and further performance requirements and/or technology changes will certainly lead to changes (or enhancements) in the protocol. The desire to use Gigabit Ethernet for instance is a good example. It is inherently easier to make changes to software than to change a hardware design.

A software implementation on a separate CPU includes the agents for ICMP and SNMP; a clear advantage over a hardware implementation where these handlers would be in responsibility of the host CPU (and use a certain percentage of the CPU time budget).
Glossary

AFDX  | Avionics Full Duplex Switched Ethernet.
APEX  | Application Executive, as specified in [ARINC653].
ARINC | Aeronautical Radio Incorporation.
End System | An active AFDX client, connected to an AFDX Network to communicate with other clients respecting AFDX rules is called an End System. It is the subsystem part which must be embedded in each avionics equipment, connected to the AFDX network to communicate with the other AFDX clients.
BAG   | Bandwidth Allocation Gap: minimal gap between sending time of two consecutive frames for a VL.
Frame | The AFDX protocol is based on the IEEE 802.3 frame principle: this is the minimum information unit transferred in an AFDX link, including protocol encapsulation information.
IMA   | Integrated Modular Avionics.
MAC   | Medium Access Control.
SAP   | Service Access Point: this is a particular AFDX Port for low-level applications, for frame transmission instead of highest level messages.
Skew  | Time difference in the arrival of redundant frame copies for both networks.
SN    | Sequence Number: this a byte added to the AFDX frame payload to manage the redundancy.
SNMP  | Simple Network Management Protocol
ICMP  | Internet Control Message Protocol. One of the core protocols of the Internet protocol suite. It is mainly used by networked computers’ operating systems to send error messages —indicating, for instance, that a requested service is not available or that a host or router could not be reached.
VL    | Virtual Link: the communication channels used to transfer user’s data from an End-System to one or more End Systems, across the switched AFDX network.