Space Plug-and-Play Avionics over CAN Bus*

Christoffer Holmstedt
christoffer.holmstedt@gmail.com

ABSTRACT

Abstract. What is the state of the art? Maximum 250 words.

1. INTRODUCTION

1.1 Rationale

1.1.1 Space Plug-and-Play Avionics (SPA)

The main goal with Space Plug-and-Play Avionics (SPA) is to make the assembly process of a spacecraft go as fast as possible. Development of separate components can take years but when it's time to connect the component to a system, it should be done in a matter of hours or days.

For this SPA is being defined in a set of standard documents ranging from how the communication protocol should work to how different types of subnets comply with the general logical interface standard [3, 4, 6, 5, 2].

Some key concepts with the Space Plug-and-Play technology

- Assembly of components to a complete system should be done in hours or days.
- SPA defines a set of technologies that can used for communication; this includes, but is not limited to SPA-u for "USB subnets" and SPA-s for SpaceWire subnets.

1.1.2 CAN Bus

The CAN Bus was created by Bosch for the automotive industry [1], it's designed to reduce wiring and have a high-level of safety built-in. There is no routing system in a CAN Bus so all connected nodes can read all messages and send as soon as the bus is free. Instead all types of messages have a Message ID linked to it and all nodes that want to read that kind of message filter them out.

The Message IDs also defines the priority in a CAN Bus. If two nodes starts to send a message at the same time, arbitration will be done by comparing all bits in the message id. As soon as a node detects that another message is being sent at the same time with a lower Message ID it will stop transmitting.

Some key concepts with the CAN Bus

- All nodes can read all messages.
- Multiple nodes are not allowed to send messages with the same message ID.
- Message IDs are used for priority.

1.2 SPA over CAN Bus

This report will go through the problems with adapting SPA for the CAN Bus as well as possible solutions. This work is done as a part of a project at Mälardalens University where an Autonomous Underwater Vehicle is built. The main motiviation for "SPA over CAN Bus" are the requirements from the customer for this project. Throughout the report basic knowledge about the CAN Bus as well as the general concepts of SPA is assumed.

The report is structured as follows. It starts with terms and definitions, followed by problem statement, solutions are given in the "SPA CAN Bus Adaptation" chapter. The conclusions from this research is given in the end as well as if suggested solutions are worth going forward with in the Autonomous Underwater Vehicle project.

2. TERMS AND DEFINITIONS

Terms and definitions are copied from documents related to these topics. These are the SPA Standard Drafts [3, 4, 6, 5, 2] and the CAN Bus 2.0 Specification [1].

2.1 General

- **UUID:** Universally Unique Identifier
- **URN:** Uniform Resource Name
- **ASIM:** Appliqué Sensor Interface Module
- **UUID** Universally Unique Identifier

*This report was written in October 2013 in the advanced level project course DVA425 at Mälardalen University, Sweden.

2.2 Space Plug-and-Play Avionics

- **SM-x:** SPA Subnet Manager, where x represents a given technology protocol.
- **CUUID:** Component Universally Unique Identifier.
- **xTEDS:** Extensible Transducer Electronic Data Sheet.
- **XUUID:** xTEDS Universally Unique Identifier.
- **CAS:** The Central Addressing Service (CAS) is responsible for providing logical address blocks to be assigned to each hardware or software component. The CAS stores the logical address block and logical address for each SPA Manager in the SPA Network.
- **Plug-and-Play (PnP):** Ability to connect a device to the larger system and have the two work together with little or no set-up required (e.g., automated system recognition and data exchange).
- **SPA Lookup Service:** The SPA Lookup Service is responsible for accepting component registration and providing data source route information for components requesting a particular type of service (or returning a nack if the service is not available).
- **SPA Manager:** The SPA manager is responsible for performing discovery for a particular subnet. It maps incoming packets to the correct SPA endpoint on the subnet, encapsulating the SPA packet with the correct protocol header. In the reverse direction it removes the protocol header and possibly adds a new header conforming to the subnet the packet is about to enter. It is also responsible for topology discovery and reporting within the subnet.
- **SPA-L / SPA-Local:** The SPA Local Subnet Adaptation specifies the means by which SPA components interoperate on a local processing node. It specifies the inter-process communication method used in order to establish a SPA local network and the messages, protocols, and interactions required in order to facilitate PnP functionality.

2.3 CAN Bus

- **CAN FD:** CAN with flexible data-rate. Supported from Linux kernel 3.6, can be tested through virtual CAN interfaces.

3. PROBLEM STATEMENT

3.1 Bootstraping

When starting a new system the first thing that occurs is the "Network Topology Discovery" [4]. This process assigns a logical address to each component in the network. This contradicts the CAN Bus specification where no component is identified by a specific address.

The assumption is made that it's more important to implement SPA over CAN Bus than it is to follow the intentions of the CAN Bus specification. The first thing that needs to be defined is how should different components be addressed in a local CAN Bus subnet?

3.1.1 Random Message ID

Temp text

3.2 Number of Message IDs used

3.3 Fragmentation

Traversing the SPA CAN Network.

4. SPA CAN BUS ADAPTATION

Each problem from chapter 3 is addressed in this chapter with corresponding subsection headlines.

4.1 Bootstraping

According to SPA each subnet should have its own subnet manager, for the CAN Bus subnet that component is called SM-c (SPA Manager CAN Bus). When the SM-c has been assigned a logical address block from the Central Addressing Service (CAS) component the SM-c must assign a logical address to each component on its subnet. It's the SM-c responsibility to keep a routing table between SPA logical addresses and CAN Bus subnet message IDs.

A straightforward solution is to assign one CAN Bus Message ID for each component that it should listen on. The problem with this is that if more than one component tries to contact another component, at the same time, both will start their respective message with the same CAN Message ID. This will cause a bit error on one of the components that may cause it switch to "bus off" state. To reactivate the components that have gone into "bus off" state, special messages must be sent over the CAN Bus. To prevent the message id collision, each pair of components should be assigned a unique pair of CAN Bus Message IDs that they can communicate over.

With the mapping of SPA logical addresses to local CAN Bus subnet addresses out of the way, it's time to look at the actual assignment. During "Network Topology Discovery" within a SPA-Local interconnect (within one processing unit) all components can contact the SM-l component through a Well-known port of 3500 [2]. Applying a similiar approach on a CAN Bus where all connected components contact the SM-c with a well-known CAN Bus Message ID would cause similiar problems as described earlier, message ID collisions.

For a SpaceWire subnet the process is turned around, it's the "SPA Manager SpaceWire" (SM-s) that is responsible to initiate contact. It first try to identify all others SM-s components on its subnet and then other SPA Components. An example on how this is done is given in the SPA SpaceWire Adaptation Standard Annex A [6]. The difference with CAN Bus is the already available addresses. The SM-s sends a specific message to all possible addresses in the network, which is not possible on a CAN Bus. A SM-c could instead send a specific message with a well-known message id that all other SM-c:s on the network could listen to. The new problem that arise concerns the response, which message ID should the response have?

If a well-known message ID is defined for the response, message ID collision could occur if multiple SM-c:s exist on the network. This is not a viable solution. To make SPA work

over CAN Bus minor configuration must be made before adding new components and during initial assembly of a CAN Bus subnet.

Earlier in chapter 4 it says that each pair of components must be assigned a unique pair of CAN Bus Message IDs. With only 10 components in a CAN Subnet that would mean 90 message IDs have to be defined manually. This clearly shows that defining all Message IDs during the configuration phase is not a desired solution. To simplify this process the SM-c is given the responsiblitiy to map all message IDs between different SPA components. The assignment that is done manually is the message IDs used between the SM-c and each component on the network. With 10 components on the network only 18 message IDs would have to be assigned instead of earlier mentioned 90 (with increase amount of components the difference will be even higher).

To recap one possible solution for Network Topology Discovery within a CAN Bus is as follows:

1. Configure each component with its CUUID
2. Configure each component with its xTEDS file and XUUID
3. Assign a pair of CAN Bus Message IDs for each component. One to listen on and one to transmit on.
4. Configure the responsible SPA Manager CAN Bus to listen and send on the earlier defined CAN Bus Message IDs for each component.

5. CONCLUSIONS

5.1 Future work

Bootstraping. Instead of assuming that you have to step away from the CAN Bus specification to make SPA work...that is each component must have its own address...maybe it's possible for a SM-c to handle many-to-many relations between SPA components.

All components have a unique SPA logical address but within each CAN Bus subnet messages can be read by multiple components (that is listening to the same can bus msg id).

6. REFERENCES

[1] Can bus 2.0 specification. `http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf`. Accessed 2013-10-03.

[2] Spa local subnet adaptation standard draft. Not available for public review.

[3] Spa logical interface standard draft. `http://aiaa.kavi.com/public/pub_rev/SPA_S-133-3-201X_PR_Final.pdf`. Accessed 2013-10-03.

[4] Spa networking standard draft. `http://aiaa.kavi.com/public/pub_rev/SPA_S-133-2-201X_PR_Final.pdf`. Accessed 2013-10-03.

[5] Spa physical interface standard draft. `public/pub_rev/SPA_S-133-2-201X_PR_Final.pdf`. Accessed 2013-10-03.

[6] Spa spacewire subnet adaptation standard draft. `http://aiaa.kavi.com/public/pub_rev/SPA_S-133-9-201X_PR.pdf`. Accessed 2013-10-03.