Advantages with a global clock in CAN systems

by
Lars-Berno Fredriksson
KVASER AB
010923

1 Introduction

The communication in most realtime control systems is time scheduled. One reason for this is that a message stream with a known latency and a small jitter is easier to handle than a message transmission based on events or periodic transmissions relying on uncoordinated local clocks. A true time scheduled communication relies on a common clock. Each message has a specific time slot where it is transmitted or received. The time slots can be source/destination oriented or message oriented. In the first case, each node gets a time slot where it should transmit or receive messages. The source or destination is given by the time slot but the message content has to be recognized by an identifier. In the second case, the message is identified by the time slot itself and no overhead is needed. The common clock can be either a real, physical one or a virtual one. A physical clock is often implemented as a source transmitting a pulse train on separate line, read by every node. In theory, it is a simple solution, but in practice sensitive to disturbances and the extra line adds to cost. An alternative is to use the GPS time, either directly or indirectly by so-called pseudolites, as the common clock. In case of a virtual clock, the global time does not exist in any node. Each node adjusts or relates its local clock to certain events that can be detected and are known to occur at specific times.

CAN has often wrongly been regarded as an inherently event-triggered protocol and therefore up-front been disregarded for time scheduled systems. It is not true that CAN is event-triggered. The standard only specifies how message collisions on the bus are resolved in a fully predictable and efficient way. But collisions are not required to happen in a CAN system. The very soft-spots of traditional time controlled systems are how to handle a loss of the time master or a corrupted time schedule and the often extensive bandwidth that has to be reserved for emergency messages. These problems are vanquished by CAN combined with a global clock. According to the CAN spec., any CAN controller should be capable producing a time quantum of 40 ns. SOF is one time quantum and any CC is hard synchronized to an SOF appearing on the bus. Only a few digital components have to be added to have a high resolution, high accuracy clock integrated in a CAN Controller. **Thus, CAN is ideally suited as a base for high performance and very robust time controlled systems.**

2 A virtual global clock for CAN

CAN is well suited for the implementation of a virtual clock. SOF is used for hard synchronization of the bit -timing clock in each node in a system and thus a suitable event for a virtual clock as well. The virtual clock is setup by selecting one node as time master at the system startup. It transmits a specific time-message. The time master reads its own local clock at the SOF of the time-message and puts the value as data in the very same message. The receivers read their local clocks at the SOF and compare their time with the value in the message. This is done three times: The offset is calculated at the first message, the phase error at the second and the local clocks are adjusted

accordingly. The synchronization is verified with the third message. After this setup, the time master is not needed any longer. Any time-stamped message from a selected group of nodes, or all nodes, can be used for the resynchronization of the local clocks to the, now virtual, global time.

3 Reduced bandwidth requirement with CAN

A time scheduled communication is the most efficient solution during normal conditions in well predictable systems. However, emergency situations have to be handled, and emergency messages have to have a short latency time. Several time slots have then to be reserved for emergency messages. A great deal of the available bandwidth is then lost during normal conditions. This bandwidth can be recovered by using CAN. As any message has a unique priority compared with any other message in CAN, emergency messages can be sent directly upon emergency events. Such a message will only destroy one time slot. From a scheduling point of view this is acceptable. The emergency message can be seen as a bus disturbance, and such are foreseen during the design and taken care of at normal conditions anyhow.

4 Schedule failures

A weakness of time schedules systems is that schedule errors are difficult to handle. The three main schedule failure types are:

Phase error of a local schedule

Corrupted local schedule

Global schedule breakdown

All three types are easily detected and handled by applying CAN.

4.1 Phase error of a local schedule

If a node does not meet the time window of its slot for transmission, this is usually due to a more than anticipated drift in the local clock. The local clock accuracy compared with the global clock is not within limits. Sometimes this problem is solved by a bus guard in the node, relying on a separate clock. If one of the clocks is out of bounds, the transmission is prohibited. This solution can be applied in a CAN network but CAN offers another possibility: If a node should slide into the next time slot, the CAN Controller at the node owning that slot will just wait until the previous message is completed and then automatically transmit the pending message. No message will be destroyed and any other node knowing the schedule has the possibility to detect the failure. There are several strategies to choose from for a system to recover from this failure type. As no messages are destroyed when using CAN, a schedule phase error problem is not as critical and easier to solve in a CAN system than in other systems.

4.2 Corrupted local schedule

This error is when a node transmits a wrong signal in its own time slot or uses a time slot that belongs to another node. Theoretically, no message identifiers are needed in a time scheduled system. The identification lies inherently in the schedule. But then, it is (almost) impossible to detect

a slot error. Every CAN message has a unique identifier. Thus, it is easy to detect if a node transmits a wrong signal in its own time slot. It is also easy to detect if a node transmits in another node's time slot. In this case either both messages will be transmitted back to back or the one that first appears on the bus, depending on the slot length and the slot protection mechanism selected for the system. In any case, the failure is easily detected as well as its source and the failure is not critical.

4.3 Global schedule breakdown

The most probable cause for a global schedule breakdown is a corrupted global clock and the second most is failure when modifying the schedule. A global schedule failure is the real Achilles heel of most time scheduled systems as it can lead to a total break down of the system. But here CAN offers a real advantage. As the bandwidth is adequate, all messages will be transmitted, even if the schedule is corrupted, and no information is lost. As the system during normal conditions would survive at least two consecutively corrupted messages from any node, there is time for a reconfiguration to a fall back mode. A fallback mode can rely upon a chain of related messages. One node transmits its messages periodically according to its local clock, the next node uses this message to trigger a delay of its own transmit message. This message will in turn trigger the next node and so forth. If the expected trigger does not show up, the message is sent anyhow at a timeout. Even if collisions will take place, these are resolved by the CAN arbitration and no data is lost. Implementation of a virtual global clock would reduce the risk for a global clock failure.

5 Handling of corrupted nodes

A node can be corrupted in many ways leading to a breakdown of the bus communication. The "Babbling Idiot" case is often used against CAN. A Babbling Idiot is a node continuously transmitting a high priority message which then blocks the bus access for any message with lower priority. To cure this problem and other local schedule failures, it is often recommended to have a separate bus guard by which the bus access for transmission is controlled. However, it is difficult to make a bus guard completely CPU or CAN Controller independent and there are other failures by which a node can destroy the communication, e.g., a shortcut in the bus driver, a faulty bit rate, etc. An alternative to a bus guard would be to use the "RED-CAN"[1] concept and to distribute the local schedules of the immediate neighbors at the left- and right-hand side to each node. The RED-CAN concept is that each CAN driver has the possibility to break up and terminate the bus at each side of the node. The bus is physically connected in a ring. At startup, two predefined adjacent nodes disconnect the section in between and terminate the bus. The cable between the two nodes is then redundant. If any other section is broken or shortcut during run time, this section is sealed off and the redundant part is integrated into the bus. If a node is found faulty, it can be disconnected from the bus by its neighbors. A great advantage with RED-CAN is not only that it saves cable and connectors compared with other bus redundant concepts, it also solves the problem with a complete node failure. Traditionally, all buses are connected to each node and the loss of one node can then kill all the buses. In RED-CAN, the bus sections on both sides of the node are physically disconnected at distance, leaving the remaining part of the system intact.

---

[1] See patent US6151298: Electronic bus system

By distributing the local schedules of a node to its immediate neighbors, the error detection capability is enhanced. If a node transmits at an unscheduled time, its neighbors will disconnect it from the bus. If a node erroneously finds that its neighbors violate the schedule because its own clock is not properly synchronized to the global clock, it may disconnect itself from the bus. In this case, it might be better that the node turns into a listen only mode until it has resynchronized its clock.

6 Other use of the clock

A global clock is not only useful for message scheduling. It can be used for synchronizing applications or tasks between different nodes. A communication will always have a jitter in time, regardless of time triggered or not. Even in time triggered systems, most often the global clock is used only for communication tasks and applications are in turn synchronized by the messages. Using the global clock directly to synchronize the applications decreases the needed of communication bandwidth and increases the robustness of the system. By time stamping the capture of measured values, the actual jitter is known in a feedback loop. By taking that in account, the quality of a feedback loop can be considerably enhanced. Set-point values can be distributed asynchronously and executed in concert by several nodes at a given time. Different types of errors can be detected by time stamping events and messages.

7 Conclusion

The combination of the CAN protocol and a global clock offer great advantages over current time scheduled systems. Some weak points of time scheduled systems, as the detection and handling of schedule or clock failures and the waste of bandwidth for emergency messages, are overcome in an elegant way. The bit synchronization mechanism in CAN, with its hard synchronization at SOF, is well suited as a base for an accurate high resolution global clock. A CAN Controller with an integrated clock, that can be synchronized to a common global time in the system, will be an important component for making CAN the ideal protocol not only for safety critical systems but also for advanced control systems.