CAN for time-triggered systems

Lars-Berno Fredriksson, Kvaser AB

Communication protocols have traditionally been classified as time-triggered or event-triggered. A lot of efforts have been made to develop new protocols for control systems, e.g., LIN, TTP, TTCAN and FlexRay. The focus has rather been on the communication than on the system leading to systems designed for the communication. A system needs different types of communication in different situations. The communication has to handle time scheduled as well as sequence scheduled and event triggered messages simultaneously in an efficient controller system. CAN is the almost perfect protocol for this task. The Start of Frame (SOF) is well suited for synchronizing applications. Synchronized applications produce synchronized messages. By moving the timing from the communication layer to the application layer (where it belongs), CAN turns into the preferable protocol for time-triggered systems as it provides not only a robust communication during normal conditions but also during emergency situations. Unscheduable events as well as global clock failures can be handled in a predictable way.

1 Introduction

It is often said that a time triggered communication protocol is to be preferred for safety critical systems. It is argued that a time triggered communication is predictable but an event triggered is not. Hence, the former type should be chosen as a base for the system. I think this argument is too hastily swallowed. It has to be accepted that a message can be corrupted now and then during normal conditions. How much of the bandwidth has to be allocated for handling such disturbances? Emergency messages cannot be time scheduled and is often very time critical. How much of the bandwidth have to be allocated to handle such events? The nature of these problems is unpredictable events and the remedy is a higher bandwidth. An even more intriguing question is: ?What happens if the time synchronization between the different nodes is lost?" This situation has to be avoided or the communication is lost in a time triggered communication.

Time triggered systems are simpler to make predictable and to analyze than event triggered ones which make them the preferred choice for safety critical applications. But does a time triggered system need a time triggered communication protocol? The answer is no. As will be shown in this article, an event triggered protocol as CAN is perfect for time triggered systems providing not only robust communication but means for time and sequence synchronization, system supervision as well as failure handling.

2 Time-triggered and event triggered communication protocols

The main difference between a Time-Triggered and an Event-Triggered communication protocol is that time is a part of the specification in the former but not in the latter. At lower levels, both are event triggered but a time triggered protocol has an additional level for the timing. This means that we have a specific notion of time for the communication. In a time triggered communication, every node has to have synchronized clocks with the same accuracy. If a message is corrupted, the receiver has to wait for the next slot reserved for the message. Alarm messages, rarely sent but requiring low latency time, have to have frequent but unused slots. FlexRay has solved the problem by dividing fixed time slots in two parts, one for scheduable messages and one for event triggered messages. Still, every node has to be accurately synchronized to a global time before any communication takes place. The communication forces every node to keep its time constantly synchronized to a global time within close limits, even if the local application does not need it.

In control systems, we have to synchronize the applications in different nodes one way or another. Some may require very

accurate synchronization, others may not. Well-synchronized applications will produce well-synchronized transmission events. An event triggered communication will work as a time triggered one. The well-synchronized messages can be used as triggers for simpler nodes, which then can be synchronized in a cheap way. Further, by observing the communication, the quality of the synchronization of the applications in different nodes can be monitored. A time triggered protocol would hide this. Thus, an event triggered communication protocol is to be preferred for time triggered systems as it can be made simpler, cheaper and more robust than time triggered ones. It will also work if the time synchronization is lost and provide possibilities for a graceful degradation of the system.

3 Basic assumptions for time triggered systems with an event triggered communication

Some basic assumptions have to be made for the subsequent discussion:

- There is a system designer responsible for the system and one or more module designers, each responsible for one or more modules.
- There are three types of events that have to be handled in parallel:
  - Time scheduable events
  - Sequently scheduable events
  - Non scheduable events
- The notion of time is a key element in time triggered systems and is different in different parts of a system.
- The time triggered systems discussed are control systems.

4 Time

Time can be used for computing physical and chemical processes and for bringing events in a sequently order. It is essential to keep these two different uses in mind: In the first case we need a universal concept of time, in the second case not. In time triggered systems, we often have to deal with these two concepts of time in parallel. A common mistake is trying to force the two concepts into one as this only leads to unnecessary complex tasks. For mathematical models of physical and chemical processes, we need time according the definition of a second. For any other purpose, there are usually other definitions that better fits the problem to solve.

Time is hard to define but simple to measure. Luckily, in time triggered systems we only have to measure time. For this, we need event genera-tors that generate repeating events, e.g., an oscillator or a rotating body. By counting the events, we get the time. Each event generator can generate a Time Domain (TD) if we can identify a specific event as the starting point. Time Event Generators (TEGs) can be of two types:

- Constant Frequency
- Variable Frequency and those can drive TDs of two types:
- Linear Time
- Circular Time
- which in turn can be of two types:
- Continuous Time
- Discontinuous Time

Different TDs can be related to each other by Reference Events (REs). A RE is an event that can be observed and time stamped in two or more TDs, simultaneously or with a known difference in time. For each problem, there is a TD where the simplest solution exists. This solution may then be transformed into another TD by a time operator.

5 Clocks

Time is measured by clocks. Clocks can be of two types:

- Real Clocks
- Virtual Clocks

A Real Clock is a counter counting Time Events and visualizes time according to the rules of a TD.

A Virtual Clock is a clock showing time by a mathematical operator transforming time in one TD into time in another TD.

6 Synchronization

The basis for control systems is synchronization of events in different parts of the system. These events are detected or generated by micro controllers, each

driven by a local oscillator. The local oscillator is well suited as the TEG for a local TD as the response time and computing time is directly influenced by the oscillator.

The traditional way of synchronization is to synchronize the clocks to a global clock showing global time. To do so, we need at least two Reference Events, each associated with a specific time in the global TD. By the first RE, we can compute the time offset and by the second we can compute the frequency deviation between the local clock and the global clock. Then we can adjust the local clock to show the global time by either adjusting the local TEG frequency or inserting or deleting Time Events or a combination of both methods.

A better approach is to keep the local time and convert it to and from the time of another TD when needed. The accuracy of the local clock is kept and it can be optimized to the local need. Simple modules can be kept simple and indirectly rely on other clocks in the system for the communication. Time conversions do not need much computing power. To do it every time when exchanging information with other nodes is simpler and more reliable than manipulating the local clock.

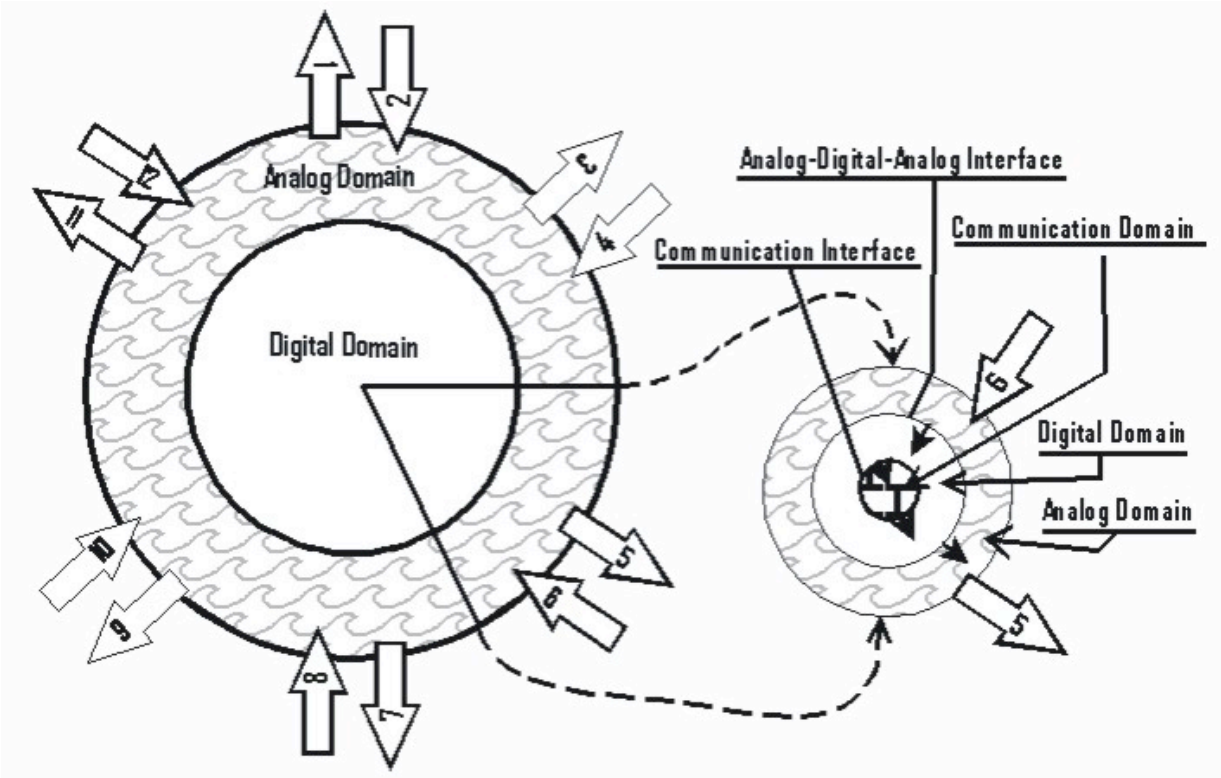

Figure 1: The centralized system is sliced into smaller parts, each forming an ECU, and a communication layer is added.

7 Systems design

The type of communication protocol has to be considered at an early stage of the system design. A good approach is to look at the whole system as a centralized one to start with. The outer boundary of the system is analog and composed of input and output events. Inside, there is a corresponding analog-to-digital and digital-toanalog boundary. The first step is to identify all sensors and actuators needed and their timing requirements to make the system work.

The second step is to slice the system into suitable parts. Each part will be the specification for an Electronic Controller Unit (ECU) and can be seen as a subsystem with boundaries like the central system. These subsystems have to exchange information, so we have to add a new boundary, a communication boundary, with timing requirements and restrictions (fig. 1).

The requirements are depending on how we sliced the system in the first place and the restrictions on the communication protocol chosen in the second place. This

is a typical hen and egg problem. Here, we choose CAN as the egg and we will see what kind of hen we will get.

The third step is to schedule the inputs and outputs as well as the communication.

8 Scheduling

This process is very similar to project planning and PERT/CPM or GANTT methods and tools can to advantage be used here. In a first step, the system designer creates a System Schedule. He schedules all foreseeable major tasks and events required for the system performance according to a virtual time line, showed by a virtual system clock, i.e., we have a Virtual System Time Domain (VSTD), one and the same for every node in the system. A purely theoretical world. If one node needs an input value measured by another module, it has to be received within a certain time to keep the system stable. (This time frame can be extended if the time of capture is known). As we now are working in a virtual time domain, we can calculate the maximum time tolerances allowed for every event to keep the system within specified performance as well as the tolerances for keeping it safe. It is important to identify the communication interface at each node and determine the earliest time a transmit message can be ready for delivery and the latest time each receiving node has to receive it.

The communication is scheduled in a traditional way with time slots slightly longer than the longest message. This will ensure we have an adequate bandwidth. Only scheduable messages should be scheduled. No time slot should be reserved for unscheduable messages as alarms, retransmissions, etc. When we have managed to schedule every message, we know he system would work if we had perfect clocks in the system. We also know that we have more band-width than needed during normal conditions. As the length of every message is known, the actual slack at each slot is also known.

Sometimes it is difficult to find a proper schedule for a message. The slots do not even up with the repetition rate of this message and other messages. After a certain time, it will hit a slot already occupied. One solution might be to assign slots att a higher rate than needed. This is a waist of bandwidth. As we will see later, another solution is to use the same slot for two or more messages now and then. In a CAN system, it can be advantageously scheduling messages to deliberately collide and take advantage of the collision resolution mechanism in CAN.

In the next step, the system designer breaks down the System Schedule into Module Schedules according to the VSTD. Each module will have its own local schedule of every scheduable event related to the module. As each node has to rely upon one or more local TDs (fig.2), Reference Events have to be identified and scheduled for the coordination of the different TDs. For any module designer, one thing is certain; his module will be attached to the communication and form a node in a system. Then it is rather obvious to look for suitable REs on the communication. In CAN, reception of a message is a good choice. SOF of a message might be even better. A time converter converting time in one TD to another, will require time stamps of the REs in each domain, so it is natural for a module designer to offer the service of transmitting the time stamps on request.

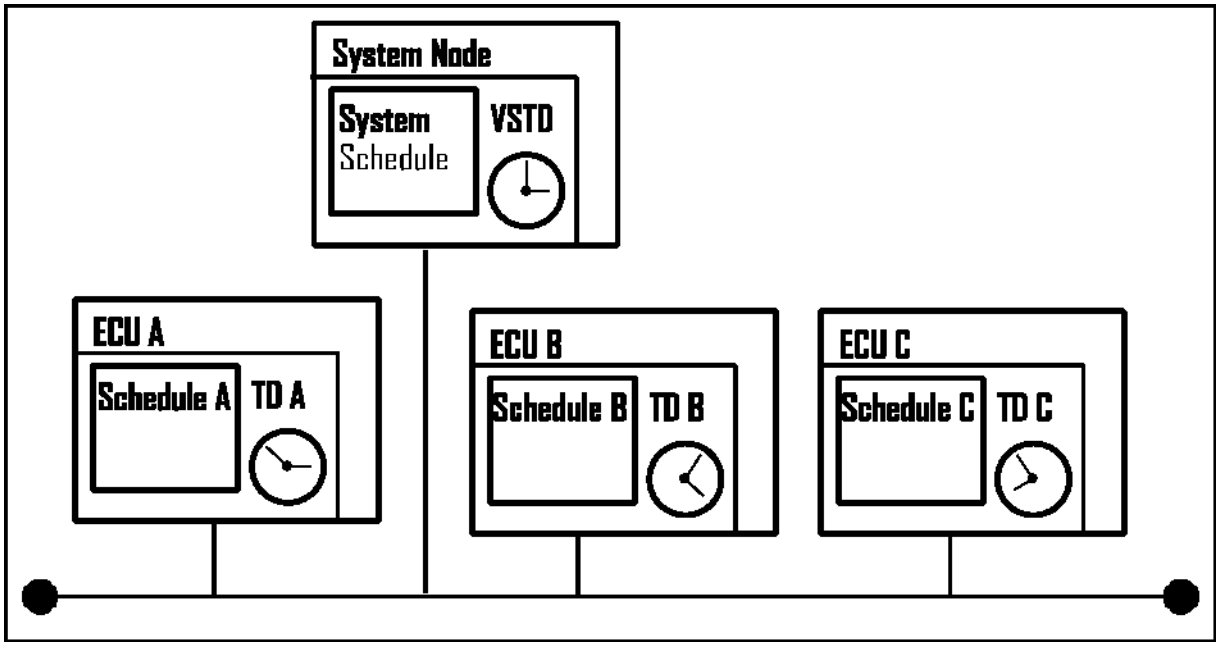

Figure 2: The system schedule is broken down into a local schedule adjusted to the local TD at each node

9 The system schedule in runtime

The schedule in runtime is depending on how well the different TDs can be synchronized to the VSTD. Some advanced nodes may require several local TDs with different properties, e.g., one TD with a continuous circular time measured by a high resolution, high accuracy clock for closed loop control tasks, one TD with a discontinuous linear time and medium resolution clock for the communication and one continuous linear time with a low resolution clock for calendar time. A simple node, e.g., measuring temperature at command, may do well with a TD with a very short discontinuous linear time and low resolution clock, just enough for capturing the temper-

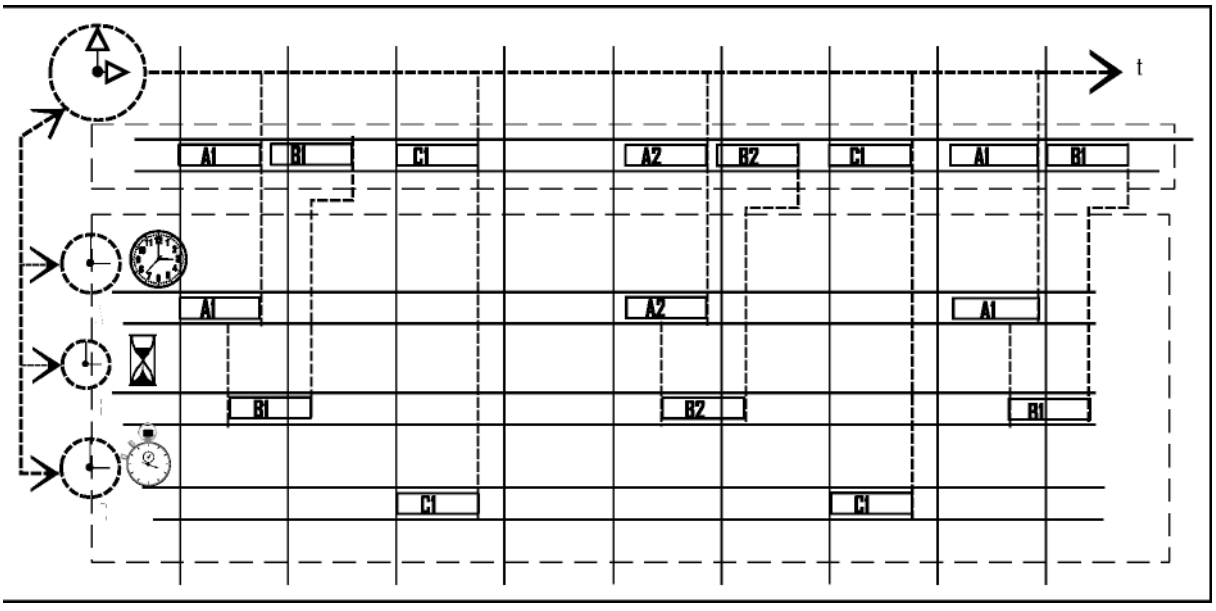

Figure 3: If a schedule jitter is acceptable, messages from a node (B) with a less accurate clock are scheduled to deliberately collide with the previous message. The message will appear in a timely order on the bus due to the CAN collision resolution.

ature and transmitting the value within a few milliseconds from the reception of the trigger message. In between, it does not need any notion of time even in a time triggered system.

10 Coordination of local time domains

The basis for coordination of TDs is REs. The only correlation between the different TDs needed for the communication to work is that the clocks keep their frequency within the tolerance required by the CAN specification. This is continuously checked by the CAN controllers. As long as messages run on the bus, the oscillators are within allowed tolerance. CAN messages are therefore ideally suited for bringing REs. The SOF is the best but only some CAN controllers make this available for the applications. Some other CAN controllers offer time stamping of received messages, including their own transmit messages. This is also a good RE but not as well-defined as the SOF. Finally, the ACK bit transmission is always available and possible to use as a RE. As we are not manipulating the local clock, just capturing the time, we can wait to accept the RE until the message is correctly received. Then, CAN has a highly reliable, built in RE mechanism free of charge. Every node in the system has captured the same event and every TD is related to the same event.

The appropriate quality of the local TDs is mainly a problem for the module designer to solve. If he does not foresee any message ex-change with other nodes in a system, he may offer a specific TD with a linear and discontinuous time dedicated to message transmissions and to coordination of events with other nodes. The time starts when a message with matching CAN identifier is received and lasts for a given period of time. Any local event can be scheduled ac-cording to the local clock associated with this TD, e.g., capturing of measured values and transmission of messages. If the SOF of a message is used as RE between two or more nodes, their relative time can be kept within a fraction of a micro second.

The system designer can have a node of his own with a Virtual Clock and the system schedule. The relation between the different TDs can be calculated, either directly by time stamped REs from the nodes, or indirectly by time stamping received messages from the nodes and comparing the reception time with the system schedule. New and better local schedules can be calculated and downloaded.

11 The communication schedule in runtime

Nodes with well-synchronized clocks will transmit at correct time. As long as a later message transmission from another node does not hit the SOF when colliding, it will not disturb the timeliness of the schedule. A node can then be scheduled to deliberately collide with the message in the previous slot. CAN will see to it that the transmission of the message will be started three bit times after the previous one. If the node in the following time slot is scheduled to hit in the middle of the previous one, the clock can deviate half a message length from the ideal time and the time slots will still be kept (fig.3). This makes it possible to allow nodes with less accurate clocks to participate in the communication where other nodes need high precision time slots.

In any system, sometimes a message gets corrupted by a disturbance. In a reasonably well designed system, this occurs less than once in ten thousand messages. When a message is corrupted, CAN will automatically retransmit it. If the system is designed to admit some jitter of the message schedule, the automatic retransmission is a great feature for real time systems. The retransmitted message may collide with the next one, but the collision is resolved by the CAN rules. If the schedule is tight, there might some further collisions along the road, but as the slots are longer than the messages, the slacks will sooner or later add up to a full message and the schedule is back in shape. As few messages are involved in the collisions, the maximum jitter and the time for the recovery is computable. Depending on the CAN identifier of the respective messages, they might change order, but in a predictable way.

Alarms and similar unscheduable messages are handled in the same way and need not to have any predefined slots.

12 Additioal advantages with CAN

CAN will also work in a real time system where the TDs are fed with variable

frequency TEGs as long as the relative frequency accuracy between the different nodes is kept within the CAN specification. The base frequency can be generated by the crankshaft of an engine. The system time will then vary with the engine speed. Time runs slow at low revs and fast at high revs. Such a time base facilitates the coordination of events that are linked to the position of the shaft, i.e., the replacement of mechanical controls by electronic ones. The bit rate on the communication will then be variable if expressed as bits per second.

If there are three or mor bit times between a message and adjacent messages, we know that no arbitration has taken place. Then, the length of the message is a foot print of the transmitter's oscillator. By measuring the time from SOF to ACK, we get the relative difference between oscillator of transmitter and our own. It is then possible to continuously check that the relative difference of the oscillator frequencies are with in the limits for a safe communication.

13 Summary

By definition, real time systems have to be coordinated in time. However, the definition of time is ambiguous and time is always measured by clocks. The design of real time systems is simplified by allocating one or more Time Do-mains at each node where time is measured by local clocks counting Time Events. The TDs have to be related to each other at an accuracy given by the respective applications. These relations can be calculated by time stamped Reference Events. The communication itself does not have to be time triggered as time synchronized applications will generate time synchronized transmission events. The CAN protocol is very well suited for the communication in time triggered systems. It inherently provides suitable reference events. The quality of the time synchronization between different nodes can be monitored by the appearance of messages on the bus. The communication will work even if the synchronization is lost. Simple modules can have cheap and simple clocks and still participate in systems where other nodes need high performance and expensive clocks.

References

[1] WO 2004/055682 Schematizing of messages in distributed and supervision system

[2] WO 2005/107174 System and device for a fixed and/or mobile system, in particular for vehicles, for example cars

[3] EP 1 624 361 Device in a modularized system for effecting time-stamping of events/reference events