1

Reminders about the CAN Protocol

As an introduction to this chapter, we will remind you of some general points about all the architectures of embedded systems, and starting from now we will take a very surprising turn by passing judgement on the properties of the well-known controller area network (CAN) protocol, presenting its principal limitations and finally imagining solutions which open up new horizons for decades to come.

1.1 The Limitations of CAN

Firstly, the concept of CAN, which was designed almost 30 years ago, is perfect for current applications, and will remain perfect for very many applications. However, time passes, and some of the inherent limitations of CAN, which have been known since its genesis, are now clearly visible. They are:

- **Limitations of bit rate** – Since it began, the maximum gross bit rate of CAN has been limited to 1 Mbit/s, and forthcoming and future application fields of embedded multiplexed networks require higher gross bit rates, of the order of 5–10 Mbit/s, either for purely functional reasons of timing or because of saturation of communication capacity. Everything must therefore be rebuilt. The word 'everything' in the previous sentence probably surprises you, but it's true! In fact, everything must be rethought and rebuilt, because 1 Mbit/s, the maximum bit rate value for CAN, corresponds in practice to the limit of a technical philosophy in which it was still possible to avoid talking too much about the phenomena and/or effect of line propagation, reflection coefficient, stubs, Smith's abacus, and so on. Beyond this value, when designing protocols and their physical layers, it is impossible to avoid considering and taking account of these physical parameters and their effects.
- **Limitations of distance and topological flexibility** – It should also be noted that the 1 Mbit/s maximum value of CAN is related to the structure of the acknowledgement bit of the protocol. In fact, so that the protocol functions correctly, it is necessary to be certain that the sum of the outgoing and incoming times of the signal allows

FlexRay and its Applications: Real Time Multiplexed Network, First Edition. Dominique Paret.

the acknowledgement signal to fall within the duration of the bit. This special feature of the protocol imposes limits on the propagation time, and therefore a maximum distance, between nodes which are present on the network, but it also excludes some topological possibilities and solutions involving propagation asymmetries according to which branches of networks are used.

- **Limitations of the possibility of topological redundancy** – This point is linked to the two previous ones (distance and acknowledgement). Creating a network which makes it possible to provide redundancy of communication at the level of physical layers according to a CAN architecture/topology is difficult, not to say impossible. Consequently, it seems futile to hope to implement systems which are entirely controlled using links which function according to the famous 'X-by-Wire' (everything by wire) concept, which we will describe in detail in a later chapter.
- **Limitations of access to the medium in real time** – As we will show and/or remind you later, CAN has a strong 'event-oriented' orientation. The phases of communication on the network are mainly initiated by sporadic, random, probable, and so on events. Also, CAN lacks a 'real time' orientation, or in other words a philosophy with a 'time-oriented' orientation; that is, one in which the communication phases are initiated as a function of a clock, a date, a fixed instant. To get round that while preserving the structure of CAN, one of the first responses was the creation, by the R. Bosch company, of a higher-level application layer called 'TTCAN' or time-triggered communication on CAN, which is initiated by events in time, to refresh CAN a little (see Chapter 2).

It should be noted that all these points have been covered by the appearance of the FlexRay concept, which we will describe in detail later.

1.2 'Event-Triggered' and 'Time-Triggered' Aspects

1.2.1 The Probabilistic Side of CAN

By its design and structure, the CAN protocol encourages transmission of communication frames when events occur at a node of the network. This is what is called an 'event-triggered' system. In fact, often a participant sends a message following an action, a reaction or a request for information as a function of the requirements of the intended application and/or of its own task.

As we explained in numerous previous works, CAN messages are prioritised (offline) by the system designer, using values which the designer has chosen to assign to the identifiers of the communication frames. On this principle, at a given instant, no node can be certain that its message is transmitted immediately, because of the conflict management and arbitration resulting from the values of the competing identifiers at this precise moment of access to the network. This type of concept and the management of it give transmission of messages on the network in CAN a strong 'probabilistic' emphasis, because it is subject to the arbitration procedure. The latter is a function of the respective values of the competing message identifiers at the time of the attempt to access, and then seize, the bus or medium, which makes the timing of this transmission – and the associated latency time – very dependent on the probability of the appearance of the respective values of the identifiers.

The only true CAN message which is truly 'deterministic' is the message with the identifier hex 0000, since, for this identifier value only, the latency time is strictly known,

and its value is '*one CAN frame minus one bit plus the inter-frame time (3 bits) . . .*', since, to within a bit, this (highest priority) message could not access the network last time round.

For other messages (identifiers other than hex 0000), that depends on big ideas of scheduling, obscure calculations of probability applied to the respective values of the activity model of the network, and to the appearance of the respective values of the competing message identifiers. Also, the probability of this arbitration phase taking place is excessively high, since each time the medium is occupied – as is very often the case – all the other nodes which have been unable to access it wait for the propitious moment to try to get it back, and all starting at the same moment, just after the inter-frame phase required by the CAN protocol, are all immediately subjected to the arbitration procedure.[1]

The problem then occurs when what is wanted is to communicate – transmit or receive – definitely, at a precise, predetermined instant, so that the timing is deterministic. In principle, nothing in CAN permits this. Consequently, it is necessary to create new systems, certain actions of which are triggered spontaneously at precise instants. These are usually called 'time-triggered' (TT) systems; that is, in our case three principal concepts, TTCAN, TTP/C (or Time Triggered Protocol Class C) and FlexRay, which we will explain in detail below.

1.2.2 The Deterministic Side of Applications

In very many applications, it is or becomes necessary to trigger certain actions spontaneously at precise instants. Such systems are called 'time-triggered' or systems functioning in so-called 'real time' mode. When systems must function in 'real time' (which in principle does not exist and is merely an abuse of language[2]), the big problem occurs when what is wanted is to be sure of communicating – transmitting or receiving – at a predetermined instant, or in specific time slots, thus adding a 'deterministic' aspect to communication.

As already mentioned, in principle nothing in CAN makes it possible to guarantee this. In these cases, it is therefore necessary to set up a mini real time 'operating system' of TT type, for example on the higher layers of the OSI (Open Systems Interconnection) model (at layer 5, 'session' or layer 7, 'application') or to integrate or encapsulate this type of function into a definition of the protocol which is capable of solving all or part of this problem.

To do this, customarily, so that information can circulate on the network, specific, well-defined time windows are used. How these time windows are implemented is, in principle, completely free and non-limiting. The only specific point consists of ensuring that all the participants are perfectly synchronised, so that each can talk or respond in its turn without overlapping into the time windows of its neighbours. To do this, it is generally necessary either to transmit a 'reference clock' cyclically to the whole network so that each participant resets its clock or to synchronise the clocks of all participants.

[1] We refer anyone who is interested in this subject to the numerous publications written by Mr Laurent George.

[2] To remove any doubt, the term 'real time' is customarily understood as actually implying 'time with known latency', that is it means that one is certain that at a precise instant the thing which is supposedly being done actually is. Also, very often the terms 'real time' and ideas of 'deterministic' systems are confused. How, in certain deterministic conditions, the whole functioning can be assimilated to an idea of 'functioning in quasi real time' – that is, with deterministic access to the communication medium and known latency times – will be explained below.