TTCAN explained

Second edition

by
Lars-Berno Fredriksson
KVASER AB

010430

1 Introduction

The standard, ISO/WD 11898-4 *"Road vehicles - Controller area network (CAN) - Part 4: Time triggered communication,"* can be hard to read and comprehend as it is a mix of a standard on a specific type of scheduling CAN messages, a local clock and a global time, common to all nodes in one or more CAN networks. I will try to explain the standard in a simple way by starting with the clocks, then moving to the time triggers and at last discuss the guiding rules for the scheduling. For those familiar with CanKingdom, the most important task is to understand the clock as this is the basic feature to be used in CanKingdom systems. My description of TTCAN may give the impression that it describes how it will be implemented in silicon. However, this may not be the case as the complex standard can be described in different ways leading to different architectures, all producing the same result. I hope that my comments will give silicon designers some ideas on making implementations covering a broader market than TTCAN.

2 Time

The concept of time in TTCAN is a linear and finite time. A Time Master (TM) transmits a synchronization message, either periodically or at an event, and all nodes use this message to calculate their time slots for transmission and reception of their messages. The common time starts at the transmission of the synchronization message and ends with the last time slot. There are two levels of TTCAN:

1. Level one where the clock in each node runs independently, clocked by the bit time on the bus.
2. Level two, where the clock in each node is synchronized to the TM clock and the time unit is a programmable fraction of a second. Here the common time is called "Global Time"

Although the time is linear and finite according to the standard, any chip supporting the TTCAN level 2 will be capable of keeping a circular time.

The time in TTCAN is measured in NTUs (Network Time Units). The nominal value of the NTU is programmable but the true value is kept by the Time Master. The Time Master transmits the value of his clock in a time message, called Reference Message in the standard,

and is to some extent equivalent with the Time Herald in CanKingdom. The value of the Time Master's clock is captured at the sampling point of the Start Of Frame (SOF) bit of the time message.

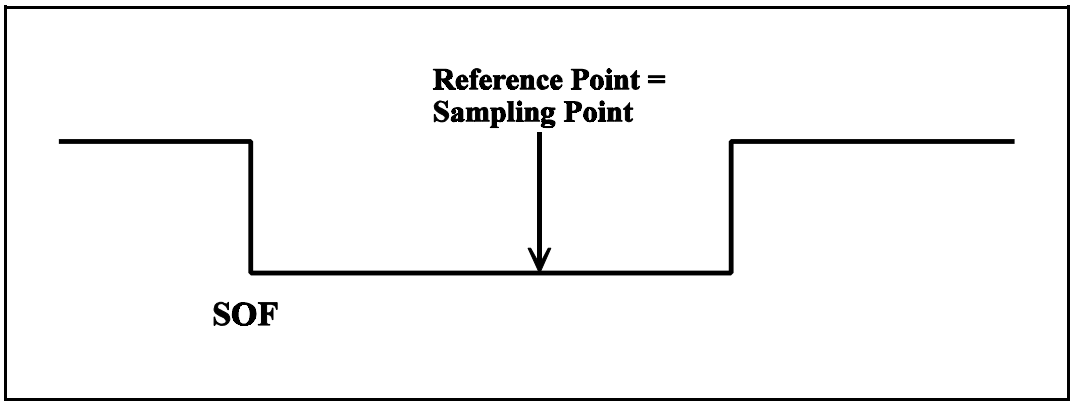

Figure 1 The value of the Time Masters's clock is captured at the sampling point of the time message

Each node capture the local time of the reference message at their sampling point of the SOF bit. It should be noted that their capture is done at a later time than the master depending on the propagation delay and their bit-timing register settings. However, this delay can be regarded static and taken care of by the application. The TTCAN standard disregards from this delay.

3 The TTCAN clock

3.1 Overview

The TTCAN standard is written with a specific type of schedule and a specific implementation in mind. To allow for a more general use of the concept, it might be better to look upon TTCAN with the clock in the center and supporting hardware and software around it. Figure 2 shows the basic features needed for a clock to support the standard.

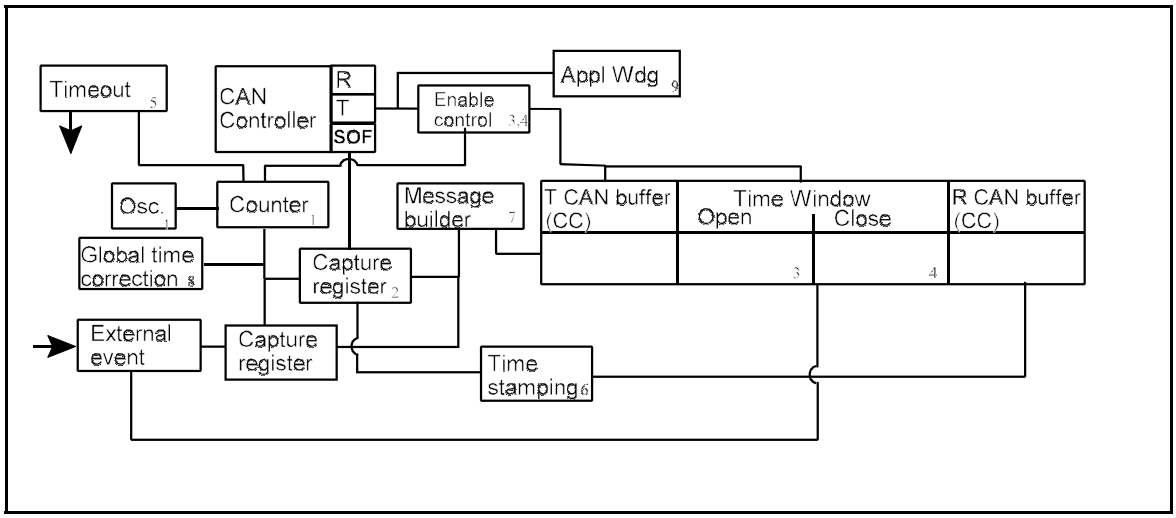

Figure 2 Clock overview

The heart is an oscillator and counter that can be adjusted to the global time. There are two ways to implement a synchronization of the local time to a global time: Either the local time source is adjusted to the the phase and value of the global time or the local time is kept and any reference to the global time is recalculated before it is used. The “Global time correction” in figure 2 represents both methods.

Associated with the time counter is a capture register. This capture the counter value at the sampling point of each SOF and makes the captured value available for a time stamping unit. This unit adds the reception time as additional bytes to each received message. Even messages transmitted are received as well by the transmitter. It also captures the counter value at external events and make the values available to applications.

The message building unit adds the captured time value of selected transmit messages as the first bytes in the data field of the very same messages.

Associated to each transmit or receive CAN Id is a time window. For transmit messages, an transmit enable unit controls the transmit function. When the time counter reaches the Open value, it enables transmission of the message and when the counter reaches the Close value, it disables the transmission.

3.2 Level 1

The TTCAN level 1 clock is, as minimum, a 16 bit counter counting NTUs. The time is linear and the first epoch starts at zero with the Reference Message Cycle 0. A new epoch starts at zero the next Reference Message. The length of an epoch is controlled by the time master by transmitting the Reference Message starting the next epoch. The standard allows for 1, 2, 4, 8, 16, 32, or 64 epochs to be set before restarting the time on epoch1. Thus time needs two counters, one epoch counter and one “time in epoch” counter. An epoch is called a “Basic Cycle” in the standard.

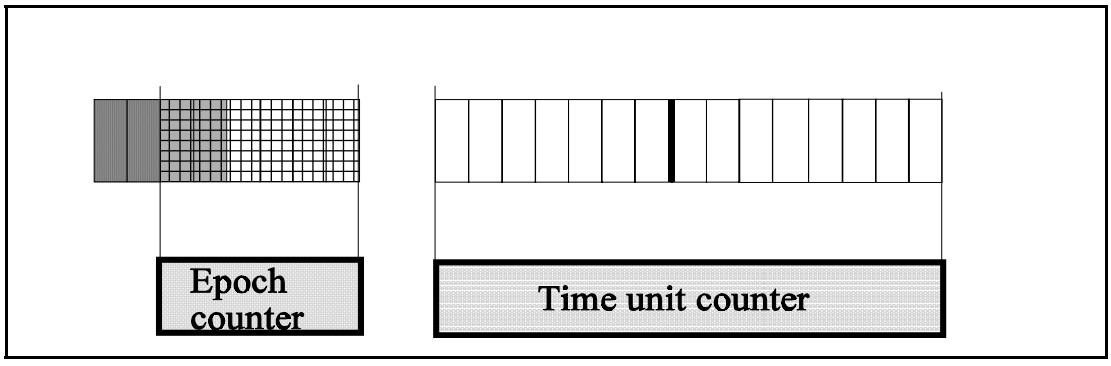

Figure 3 Time counters level 1

3.3 Level 2

The TTCAN level 2 clock is, as minimum, a 19 bit counter where the three least significant bits represents a fraction of the NTU. The standard allows for additionally six most significant bits and another four least significant bits, ending up in a 29 bit counter. The optional six most significant bits are programmable to set the wraparound of the counter.

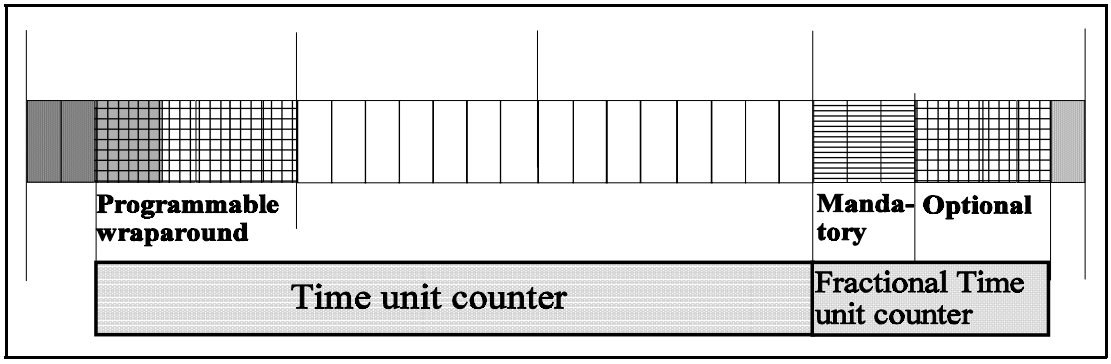

Figure 4 Time counter level 2

3.4 Scheduling primitives

Associated to each scheduled Envelope (CAN Id) are two mask filters working on the clock. They can be programmed bitwise as 0, 1 or don't care. These masks only include the NTU counting part of the clock. Combined, they form a time window when a message is allowed to be transmitted or expected to be received.

For transmit messages, one is triggering the transmit enable bit in the CC and the other one disabling the same bit. For a receive window, the associated CAN Id can be programmed as don't care, accepting any message. The time window for transmit messages is related to the appearance of Start Of Frames. The transmission of a message is allowed to start during the time window, i.e., it is regarded correct if the SOF appears at or after the Open time but before the Close time. Reception of a message is regarded as correct if the SOF appears at or after the Open time and the completion of the reception, including local acceptance filtering, is made before the Close time. Thus, the transmit and receive time windows are different for the very same message.

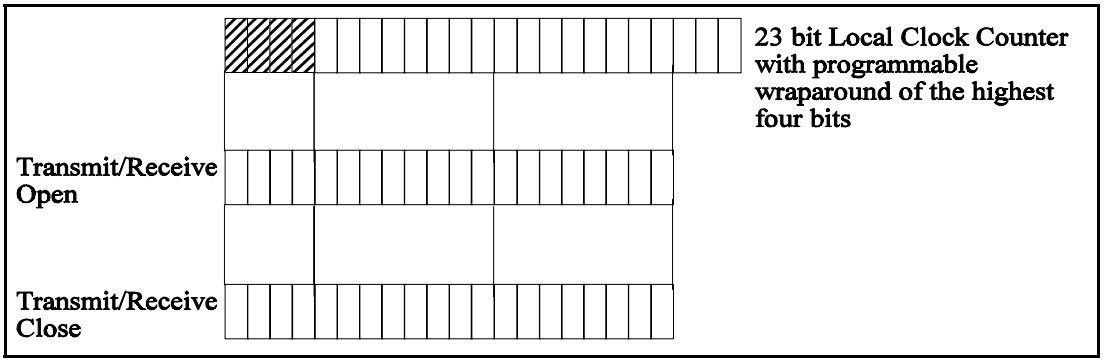

Figure 5 The clock and associated masks

The External event unit captures the time at an external event. Beside time stamping events, this feature can be used to set an offset to a message schedule by adding the captured time to the Time window.

3.5 Clock synchronization message

TTCAN has the following rules for the time message

1 Time messages are sent form Time Masters (TM)
2 There are up to eight hierarchical TM in a TTCAN system
3 A specific CAN Id, with the three lowest bits as 000, is used for time messages from the highest priority TM and the lower priority TMs use the following CAN Ids in order of their priority

Thus the TM synchronization message (Reference Message) is linked to a specific CAN identifier. The next seven CAN Ids are reserved for potential Time Masters. Any TTCAN chip receiving one of those seven Ids will treat the message as a valid Reference Message.

The format of the Reference Message looks as below:

Document name:	Time Message	
Document List:	x	
Document Number:	x	
Document type:	Transmit (Time herald) Receive (City)	
Page description.		
Number of Lines:	At least 4	
Data description:	Time message.	
Line description.		
Line 0:	Lrwwwwww	Last continous time message 1/0 = yes/no r reserved w optional highest time bits
Line 1:	tttttttt	Highest mandatory time byte. (Level 2 only)
Line 2:	tttttttt	Low mandatory time byte. (Level 2 only)
Line 3:	mmmooooD	m mandatory fraction of the NTU (Level 2 only) o optional fracton of NTU D global time discontinuity.1/0 = yes/no
Line 4:	uuuuuuuu	u Undefined and optional
Line 5:	uuuuuuuu	
Line 6:	uuuuuuuu	
Line 7:	uuuuuuuu	

If a Reference Message is disturbed during transmission, it shall immediately be updated and retransmitted. (Note: This procedure is definitely unacceptable in most time scheduled systems!)

The Last continuos time message bit L indicates that this is the last time message within a linear global time and that the next time message will be the start of a new epoch. To achieve a circular time, L is never set. L is called Next_is_Gap in the TTCAN standard.

The global time discontinuity bit D is set if the TM has made an offset change in the global time but not made any phase adjustment, i.e., not made any correction of the length of the NTU. According to the standard, there shall not be two consecutive time messages with the D bit set. However, no failure procedure is defined for this function and I doubt the value of this mechanism.

3.6 Clock summary

We have now gone through the clock attached to a CAN Controller and it offers the following features:

1. A counter, minimum 16 bits with optionally additional programmable wraparound significant bits and optionally additional less significant bits, connected to an oscillator.
2. A capture register, triggered by SOF sampling point as well as external and time events.
3. A time mask, setting the Transmit Enable bit active on the CC.
4. A time mask, setting the Transmit Enable bit passive on the CC.
5. A time mask, generating an event when the clock reaches a matching value.
6. A device storing the captured time of a received message
7. A device adding the captured time to the first bytes in the data field of selected transmit messages.
8. A device adjusting the local time to a global time by messages received from a time master.
9. An Application Watchdog

4 Scheduling assistance

We have now a clock with some basic features and we need a protocol engine to fulfill the TTCAN specification.

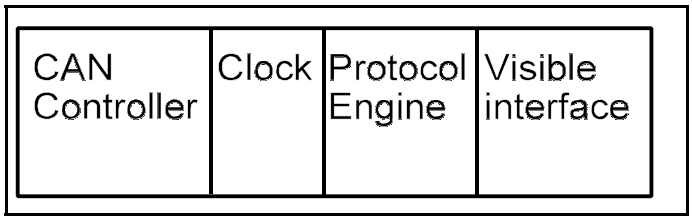

Figure 6 General TTCAN block diagram

The protocol engine has some visible interfaces. The TTCAN standard has specified a set of mandatory interfaces to suit the TTCAN schedule. Some of them are pretty much generic, some other highly specific to the matrix schedule but there are also some missing that would be nice to see.

4.1 (Almost) generic visible interfaces

The standard specifies a set of visible interfaces. Some of them are almost generic and some other ones are specific to the matrix type of schedule that is a part of the specification. Below are the part that can be regarded as generic ones. Text taken from the standard is written in bold font followed by my comments.

4.1.1 TUR_Config (9.1.1.1)

There are 2 register sets for TUR. The first one (TUR_Config) is written by the CPU on initialisation and for the external clock synchronisation process. The second one (TUR_Actual, which is influenced by the drift correction and which is read only for the CPU) is used by the controller internally, it is loaded from TUR_Config on startup and on synchronisation command. Both values are readable by the CPU. The format of these two registers is implementation specific.

The TUR (Time Unit Ratio) is the ratio between the NTU of the Time Master and the time tick used in the module. There are different ways to implement this feature, e.g., by a true recalculation of local time to global time or directly adjusting the time counter. This can be done by adding an extra bit to a pre-scaler counter, alt. stopping it for one tick, at certain values of the time counter. As the format of these registers are implementation specific, the standard only requires that the application can store actual calibration information and download startup values.

4.1.2 TUR_Actual (9.2.1.8)

Read only value containing the current value of TUR.

4.1.3 Operating_Mode (9.1.1.2)

Possible modes are: Configuration mode, standard CAN communication, strictly time triggered communication or event synchronised time triggered communication.
NOTE – Operating modes may only change via configuration mode.
It should be noted that "*strictly time triggered communication*" is misleading. For this mode, the TTCAN standard requires that a synchronization time message (Reference Message) is sent at each wraparound of the 16-bit counter. However, the only requirement to achieve *a strictly time triggered communication* is that the global time is kept within a defined accuracy within each node. This can be done in the event synchronized time triggered mode as well. Thus, any strictly time triggered communication where the time messages does not appear at each wraparound of the 16-bit counter has to be set up in event synchronized time triggered mode.

4.1.4 TTCAN level (9.1.1.3)

One bit to distinguish between level 1 and level 2.
Level 1 does not support the Global time correction block.

4.1.5 Master/Slave (9.1.1.4)

One bit to distinguish between (backup) time masters and time slaves.

According to the standard, only the time masters have the capability of putting the time stamp of the message into the data field. This is a feature that would be nice to have in slaves as well, in the first hand for diagnostics at the system level.

4.1.6 External_Clock_Synchronization (9.1.1.5)

One bit to configure whether external clock synchronisation will be allowed during runtime or not (only level 2)
How the synchronization to an external clock should be done is not specified, so this interface will be very chip specific. In my opinion, the most important thing is to be able to adjust the length of the NTU to an external reference, e.g., the GPS second tick.

4.1.7 Appl_Watchdog_Limit (9.1.1.7)

This 8-bit value specifies the period for the application watchdog in Appl_Watchdog_Limit times 256 NTUs.
The watchdog will be reset periodically by the application to ensure it is alive. According to the standard, the CC changes into silent mode when the limit is reached. I am not sure that this behavior always is the best. It would be good to have the possibility to set up the CC to transmit a (scheduled) error message. If the CC also offers to set an I/O at reception of a specific message, such a feature could be used to make a hard reset of the CPU by a message from the system level.

4.1.8 Trigger (9.1.2.1)

Each trigger's configuration data consists of different parts: First a reference to a message for which it is valid, second the time mark when it may be activated, third at which position in the transmission column it is to be activated first, and fourth the Repeat_Factor. In addition the trigger must contain the information whether it refers to an arbitrating or an exclusive window.
In the part "..., *second the time mark when it may be activated, third at which position in the transmission column it is to be activated first*" the second and third parts actually is the same as one time mark. The column stems from the scheduling matrix and, as shown earlier, is the same as the highest bits in the level 2 clock.

The Repeat_Factor can be seen as constant that shall be added to the time trigger value when the trigger event has passed. The message will then be transmitted and received periodically. In the TTCAN, this addition can only be made at the bits above the 16 bit time representation, but this restriction cannot be accepted for most other schedules. As the interface is not specified in detail, I hope that most implementations will allow for the whole time format.

The sentence "*In addition the trigger must contain the information whether it refers to an arbitrating or an exclusive window*" refers to the error handling: If a transmit message fails to be sent or received in an arbitrating window, this is not regarded as a failure but it is regarded as a failure if it is placed in an exclusive window.

4.1.9 Watch_Trigger (9.1.2.3)

The Watch_Trigger for a node consists of a time mark and of the coded definition of the operating mode for which the Watch_Trigger is active.

When the watch trigger is reached, the CC change to silent mode.

4.1.10 Cycle_Count_Max (9.1.2.7)

This value specifies the number of the last basic cycle in the system matrix. Usually the range is 0 to 15. Optionally (see 4.2.1) a controller may extend the upper limit to 63.
As commented, the Cycle_Count can be regarded as the higher bits of the level 2 global time. The next issue of the draft standard will allow an optional range of 0 - 6 bits. Then this is a programmable wraparound for a circular time.

4.1.11 Cycle Time (9.2.1.4)

As 16-bit value counting NTUs

4.1.12 Cycle_Count (9.2.1.5)

This 4-bit value gives the number of the current basic cycle (0 to 15).
The cycle count can be up to a 6-bit value in the current issue of the standard. At the last meeting, it was decided that it can be from 0 up to 6 bits. In level 2, the cycle count is the same as the high bits of the global time. Thus, [Cycle_Count, Cycle_Time] expresses the global time in NTUs. The NTU fractional part ought to be added as well.

4.1.13 Message Objects (9.1.3)

There has to be one message object for each Tx_Trigger, Rx_Trigger and Tx_Ref_Trigger. The message objects are configured during initialisation and may be updated during time triggered operation.

Message Objects for Tx_Trigger and Rx_Trigger provide storage for one LLC frame (as in ISO 11898-1) together with control and status information and 3-bit message status count (MSC).

Message Objects for Tx_Ref_Triggers provide storage for the reference message identifier and for the data length code. Parts of the reference message's data is provided by the Frame Synchronisation Entity (FSE).

The last three bits of the identifier of the reference message specify the time master priority of this node.
This interface is almost generic. It says that we have to distinguish between three kinds of message objects: Transmit objects, receive objects and time messages. Associated to each message is a time window. The standard requires some specific rules for the Xx_Triggers. Only the Tx_Ref_Triggers (the Time Master) requires the ability to put the time stamp at transmission into the data field, a feature that ought to be a selectable feature for any transmit object.

Funny enough, I have not found any where how the time slaves recognize the Reference Messages. There ought to be a bit in the receive message object identifying whether it is a time message or not. Actually, the transmitting CC does not need to know that it is a time master! Only the receiving nodes need to know if a message with a time stamped data field should be considered a time message from the Time Master or not.

The MSC is a counter, very specific to the matrix schedule suggested in the standard.

The FSE is not referenced anywhere else in the standard, as paragraph 10 will be deleted in the final edition of the standard.

4.1.14 Master State (9.2.1.2)

The master state is a 6 bit vector which combines the controller states in the field of error, synchronisation and master-slave relation, i.e. a tripel (error level, sync_mode, master-slave_mode).
The vector as such is generic but some of the rules behind it can be questioned. Some of the errors have already been discussed. According to the standard, a node shall be regarded as synchronized "*After reset a node regards itself synchronised to the network after it successfully transmitted a reference message or after the occurrence of the second consecutive reference message (if the last reference message it received did not contain a set Disc_Bit).*"
This is not adequate. Please see special comment on this issue below.

4.1.15 Global Time (9.2.1.3)

As monotic continuous 16-bit value counting NTUs.
This is a row in the schedule matrix. For other types of schedules, it ought to be the complete time, i.e., including the Cycle_Count and the NTU fractional part.

4.1.16 Host_Alive_Sign (9.2.3.1)

This interface is used to serve the application watchdog regularly.
This interface interacts with the Appl_Watchdog_Limit. If the limit is reached, the CC turns into silent mode.

4.2 Matrix schedule

The matrix schedule is the most controversial part of the standard and limits its use. To make chip implementations for a broader market, they must be able to support other types of schedules as well. This is surely possible as the matrix is only a model for simplifying the scheduling task. During run time it is unfolded to a time line anyhow.

The fundamental idea behind TTCAN is a linear time of up to 16 bits lenght, i.e., 65535 NTUs. This is also the maximum lenght of the rows in a schedule matrix (figure 7). The start of each row is indicated by a time message (Reference Message). In TTCAN level 1, the time master only sends the higher bits of the time counter, above the 16 bits, and the lower bits are reset to zero. The higher bits can thus be seen as a row counter in a schedule matrix where the first column contains time messages. The total number of rows in a matrix shall be a power of 2, i.e., minimum1 but then only 2, 4, 8, 16, 32 or 64 lines are allowed. If the length of a matrix row is FF_h , this is indicated by the continuation bit L (Next_is_Gap) set to 0 in the time message.

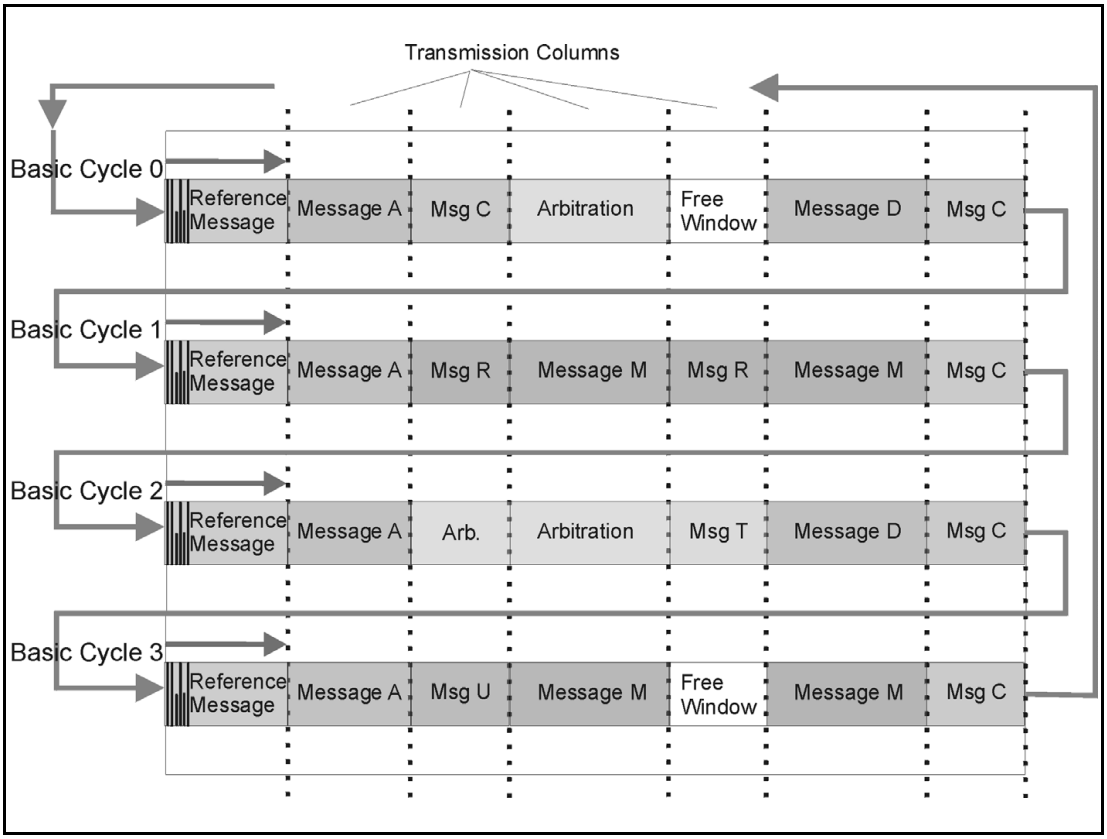

Figure 7 TTCAN schedule matrix

A protocol engine for TTCAN has to fulfil the following specific requirements:

Exclusive window
Arbitrating window
Event triggered schedule start

4.2.1 Exclusive window

For an exclusive window, the transmit disable mask shall be programmed to a 1,2,3 or 4 NTUs higher value than the transmit enable mask.

4.2.2 Arbitrating window

If a time window includes more than one message, the transmit disable mask shall be programmed to include the time for the number of messages allowed in the window but the last one plus 1,2,3 or 4 time units. The messages get access to the bus by arbitration.

4.2.3 Event triggered schedule start

In TTCAN an external event can start a limited time schedule of transmission and reception of messages. The sequence is started with a time message with the bit L set to 0 and continues until a time message with the bit L set to 1 is received. Then only pending messages are transmitted.

This functionality can generically be achieved in the following way: A time stamped message is transmitted at the event. Each receiving node adds the time in this message to the respective time windows. CanKingdom Ver. 4 offers a great variety of time triggered schedules.

4.3 TTCAN specific visible interfaces

Some of the specified visible interfaces in the TTCAN standard reflects the matrix scheduling method or a specific implementation of the global time.

4.3.1 Interrupt_Enable Register (9.1.1.6)

This register contains one enable bit for each interrupt source specified in the application interface (Interrupt_Status_Vector).
In case an error has been detected, the application is notified by updating the Interrupt_Status_Vector (see 8.1 in the TTCAN standard).

4.3.2 System Matrix (9.1.2)

The system matrix configuration may be read and written by the CPU during initialisation, it is locked during time triggered communication. The node view of the system matrix consists of the different Tx_Triggers, Rx_Triggers and Tx_Ref_Triggers.
Thus the node does not see any matrix!

4.3.3 Merging (9.1.2.2)

For each Tx_Trigger it has to be defined whether the time window belonging to that trigger shall be merged with the following window.
This is due to the awkward way time windows are specified in the standard. A straight forward implementation with a time for opening a window and a time for closing the window (alt. an open period) ought to be considered as a correct alternative.

4.3.4 Initial_Ref_Offset (9.1.2.4)

This is an 8-bit value for the initialisation of Ref_Trigger_Offset.
This refers to the specific way potential time masters compete for bus access in the matrix schedule (see 7.3 in the TTCAN standard.) There are reasons for doubts about this procedure and I hope implementors will make it possible to inactivate this behavior.

4.3.5 Expected_Tx_Trigger (9.1.2.5)

This 8-bit value limits the number of messages the node may try to transmit in one matrix cycle.
CAN enemies have managed to make the "Babbling Idiot" problem a major issue although few, if anyone, experienced CAN system designers have ever seen it in reality in any system. The Expected_Tx_Trigger related procedure looks as an extension of the "Babbling Idiot" quasi problem. There are reasons for doubts whether this procedure will enhance or diminish safety. Further, it is related to the matrix scheduling idea and I hope implementors will make it possible to deactivate this behavior.

4.3.6 Length of Tx_Enable Window (9.1.2.6)

This 4-bit value specifies the length of the time period (1-16 NTUs) in which a transmission may be started.
The value ought to be extended to at least two bytes. The first note of 6.2.2 in the TTCAN standard reads "*The Tx_Enable window of merged arbitrating windows starts at the beginning of the first time window and ends at the end of the Tx_Enable window of the last merged time window. So the length of the last of the time windows to be merged specifies the maximum length of the messages that may be transmitted in the resulting merged arbitrating time window*." By lengthening the TX_Enabling Window the need for arbitrating windows would disappear. See also 4.3.3 above.

4.3.7 Time Master priority (9.2.1.1)

This 3-bit value shows the priority of the current time master (the last three bits of the identifier of the reference message).
The idea of having a limited number of potential time masters in a system with consecutive CAN Ids on their time messages might be a good compromise between flexibility and simple implementation. However, some nodes might have to support more than one clock, e.g., gateways, that ought to be considered.

4.3.8 Wait_for_Event (9.2.1.6)

This bit is set when a Next_is_Gap bit was received in the last reference message. It is reset by the start of the next basic cycle.
This refers to the bit L of Line 0 in the time message. TTCAN assumes that a time message is sent at each wraparound of the 16 bit time counter. If not, the Next_is_Gap bit has to be set. In CanKingdom, any event triggered behavior is set up by the Action/Reaction command (King's Page 5). CK ver.4 includes the possibility to assign schedules to events.

4.3.9 Ref_Trigger_Offset (9.2.1.7)

Read only 8-bit value containing the current value of the Ref_Trigger_Offset.
This refers to the specific way potential time masters compete for bus access in the matrix schedule. (See 6.4.3 in the TTCAN standard.) There are reasons for doubts about this procedure and I hope implementors will make it possible to inactivate this behavior.

4.3.10 MSC (9.2.1.9)

Message status count, only for periodic messages.
This is very specific for the matrix schedule.

4.3.11 Interrupt_Status_Vector (9.2.2)

The Interrupt_Status_Vector is very TTCAN matrix specific and may cause severe problems for other kinds of schedules.

4.3.11.1 Operational Interrupt sources (9.2.2.1)

- **Start_of_Basic_Cycle.**
- **Start_of_System_Matrix.**

- **Synch_to_Event: A Next_is_Gap bit was received in the last reference message.**
- **Disc_Flag: This is a status flag that is set to show that a discontinuity in the global time has happened..**
- **Change_of_Masterstate: The local controller changed its master state.**
- **Global_Time_Wrap: The global time had an overflow.**

In the Operational Interrupt sources, only Change_of_Masterstate can be regarded generic. The Global_Time_Wrap refers to the 16-bit time and the other sources are matrix based.

4.3.11.2 Error detection interrupt sources (9.2.2.2)

For a detailed description see section 8.3.

- **Application_Watchdog**
- **Tx_Overflow**
- **Tx_Underflow**
- **Scheduling_Error_1**
- **Scheduling_Error_2**
- **Watch_Trigger_Reached**
- **CAN_Bus_Off**

Of the error sources, only the Application_Watchdog, Watch_Trigger_Reached and CAN_Bus_Off can be regarded generic. The Application_Watchdog has been discussed earlier. The Watch_Trigger_Reached also result in a silent mode which can be questioned. In my opinion, it should result in an out_of_sync interrupt.

5 Further comments

In my opinion, the TTCAN standard is unnecessarily complex, inflexible and expensive due to the mixing of a very specific way of scheduling and the integration of this scheduling method and the global clock. I think it would be simpler to treat the problem in two levels starting with establishing a global clock and then using this clock to establish schedules. However, during the standardization work, it was not possible to get acceptance for this idea. Accepting this, we have to do the best of the situation and try to get the standard as good as possible. The global time is essential for the future of CAN and for the foreseeable time, this is the only way to get CAN Controllers supporting a global time. Below I have made some comments to point out some of the weaknesses that must be corrected before the standard is accepted. I also point out some features that can be implemented without violating the proposed standard which will increase the market for the TTCAN chips.

5.1 Retransmission of reference messages

According to section 4.2 of the TTCAN standard, a corrupted reference message shall be immediately retransmitted. (“*If the reference message is disturbed by an error it is retransmitted immediately*.*”)* This behavior cannot be accepted in many safety critical systems and it has to be possible to deactivate the automatic retransmisson.

5.2 When a node is considered synchronized

Due to the secondary role of the global time in the TTCAN standard, it has not regarded the needed attention. This has to be compensated in chip implementations by additional interfaces.

5.2.1 Basic synchronization

The last paragraph of section 7.2 in the TTCAN standard reads *"After reset a node regards itself synchronised to the network after it successfully transmitted a reference message or after the occurrence of the second consecutive reference message (if the last reference message it received did not contain a set Disc_Bit)."*

This is not correct. A node has to receive at least three time messages from the time master before it can regard itself synchronized. It will need a first message for correcting the time offset, a second message to adjust to the reference NTU and a third message to validate that the adjustments are made correctly. As the offset will drift during the time needed for checking and adjusting the NTU length (phase error), it would be simpler to firstly adjust the NTU and secondly adjust the offset, but this would require a fourth time message (if validation is a requirement.)

5.2.2 Master Time requirements

The time master's time has to be stable during the synchronization procedure. This is only safeguarded by the Disc_Bit. The standard assumes that the time master only adjust the offset, not the NTU. This might not always be true. Further, the standard does not say anything about how potential time masters shall be synchronized and if different time masters should be accepted by a node during its synchronization procedure. This is a weak point in the standard. There should be an interface telling whether or not more than one time master is allowed during synchronization.

Different nodes in one and the same system might need time updates at different rates. The longer time between time messages, the more accurately the phase error can be corrected. Therefore the time master may have to send different Reference Messages at different rates. A time master must then have the capability to transmit "Reference Messages" with different CAN Ids. A time master does not have to know that he is a time master per se, but a node has to know by which CAN message(s) the time master information is received. The TTCAN standard has provided one solution to the multi-master problem (where the TM has to know that he is a time master), but there are several other solutions, with better or worse quality, requiring that a node can transmit time stamped messages that can be interpreted by a receiver as a Reference Message.

5.2.3 Node time requirements

Not all nodes in a system will require the same accuracy of the global time. Some might need to be synchronized within a microsecond and other within tenth of milliseconds. The TTCAN standard does not provide any possibility to set or check the accuracy. There should be an interface to program the maximum number of bits that the local global time can deviate form a reference message before an Out_of_Synch flag is set. Such a flag is also missing in the standard.

5.2.4 The Discontinue Bit

As argued above, the Disc_Bit does not solve any problems with readjusted time masters. Further, the well-known problem with last bit error in EOF may now cause a new problem. A receiver monitoring a dominant bit at the last bit of End of Frame will nevertheless accept the message as correct but a transmitter, monitoring a dominant bit at the same place, will retransmit the message. This has until now resulted in a duplicate message transmission, quite in accordance with the CAN specification. But for a Reference Message, it will not be a duplicate as the time will be updated, and, if the Disc_Bit is set, result in two consecutive Reference Messages with the Disc_Bit set. Thus a CAN error of no importance would create a nonexisting schedule error.

The main need for external global time calibration is to get the NTU within specified limits. Compared with other global clock synchronization problems, handling of offsets at the system level is the easiest one and it is not essentially time critical.

I cannot see that the Disc_Bit solves any problem but creates several ones.

5.3 Time stamping of transmit messages

The TTCAN standard does not include the possibility to time stamp transmit messages (other than by the time masters). This is a needed feature that ought to be implemented. As the first bytes in control messages often are used for encoding purposes in higher layer protocols, e.g., in CanKingdom and DeviceNet, the time stamp should be added as the last bytes of the data field. The time stamping function could then be nicely integrated in the higher layer protocols. As the TTCAN core clock most probably will be implemented as a four byte counter, the full value could be transmitted. I think a fixed DLC of eight byte would be acceptable for time stamped messages.

5.4 Wraparound values

CAN networks are often connected to other (CAN or non-CAN) networks via gateways. It would be useful to synchronize the global times connected networks and this might require programmable wraparound values of the clock. As we have seen, the highest byte of the TTCAN clock has to have a programmable wrap around. Most probably it will be necessary to have a programmable wraparound also for the fractional NTU counter. Then it might be a good idea to use four cascaded 8-bit counters, each with programmable wraparound.

5.5 Event generation

According to the TTCAN standard, time messages should be sent on external events. This feature should not be restricted to time messages. It would be a nice feature if an external event (I/O) could be generated at reception of (a) specific message(s).

5.6 Scheduling by action reaction (additional info only)

By King's Page 5 in CanKingdom, a node can be setup to react on reception of specific CAN Ids. This feature can be used in several ways in a Time Triggered CAN system. Nodes without any clock support can be integrated in a time scheduled system by piggybacking on time scheduled messages. When a time scheduled message appears on the bus, one or more nodes

can be set up to transmit messages as a reaction. Nodes with clock support can be set up to send scheduled messages, using the action message to activate the schedule. For more information on this topic, please have a look at CanKingdom, chapter 6, Bus Management. The CK spec can be downloaded from http://www.kvaser.com/archive/ck/