Application Note, V1.2, Aug. 2008

AP29006

TC1796

TTCAN Getting Started on TC1796-Board & Toolkits

Microcontrollers

Never stop thinking

Edition 2008-08

Published by
Infineon Technologies AG
81726 München, Germany

v

<Device1>

Revision History: V1.2, 2008-08

Previous Version(s):
V1.0 training file, 2006
V1.2 AppNote, 2008

Page	Subjects (major changes since last revision)

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

Table of Contents **Page**

1 Introduction

Time Triggered Communication on CAN (**TTCAN**) is specified in **ISO 11989-4** as a higher level protocol extension to the CAN protocol. The **TTCAN** protocol is fully compatible with the existing CAN protocol. The time-triggered functionality will be able to operate RTOS based on static cyclic scheduling of all tasks. The new features allow for a deterministic behaviour of a CAN network and the synchronization of networks.

In the 32-bit TriCore-based **AUTO_NG** Microcontroller (**TC1796**) implemented **MultiCAN** module supports the **TTCAN** operating.

This documentation is aimed to guide users to a quick start to **TTCAN** bus system using **TC1796** TriBoard. It contains three main parts. As first the **CAN** and **TTCAN** protocol will be described in brief. The second part is abstract about **MultiCAN** implementation in **TC1976**. A **TTCAN** configuration example will be given here afterwards. Thought this **TTCAN** application example user can have a overview of **TTCAN** software project creating by **DAvE**.

In this note, we will be using the following hardware and tools:

- code Generator: DAvE v2.1
- compiler: Altium Tasking Tool for TriCore V2.3 (C Compiler, Assembler & Linker)
- debugger & downloader: Memtool or PLS (optional) for code download
- TC1796 Board: 2x
- CAN analysis tool: vector-informatic CANalyzer (optional)

Documentation references in this note:

- ISO 11898
- TC1796 User's Manual (v2.0, July 2005)
- TriBoard TC1796 User's Manual (v3.1, January 2005)

2 CAN Protocol

CAN History:

- developed by BOSCH, standardized in ISO 11898, first deployed in 1986
- CAN (Controller Area Network) data link layer: ISO 11898-1
- CAN physical layers: ISO 11898-2 (high-speed transceiver) or in ISO 11898-3 (fault-tolerant low-speed transceiver)
- used in automotive application and industrial automation
- CAN standards:
  - CAN version 1.0
  - Standard CAN (version 2.0A)
  - Extended CAN (version 2.0B)
  - Timer-Triggered CAN

CAN bit rate and tolerant:

- 40m – 1M bps
- 100m – 500k bps
- 100 – 250k bps

CAN Bus:

- connected in wired-ANN
- non-return to zero with bit stuffing
- dominant/recessive bit 0/1
- bit wise arbitration

Message types:

- data frame: length of message (8 Data bytes)
  max. 111bits (without stuff bits) to 136bits (with stuff bits)
- remote frame
- error frame
- overload frame

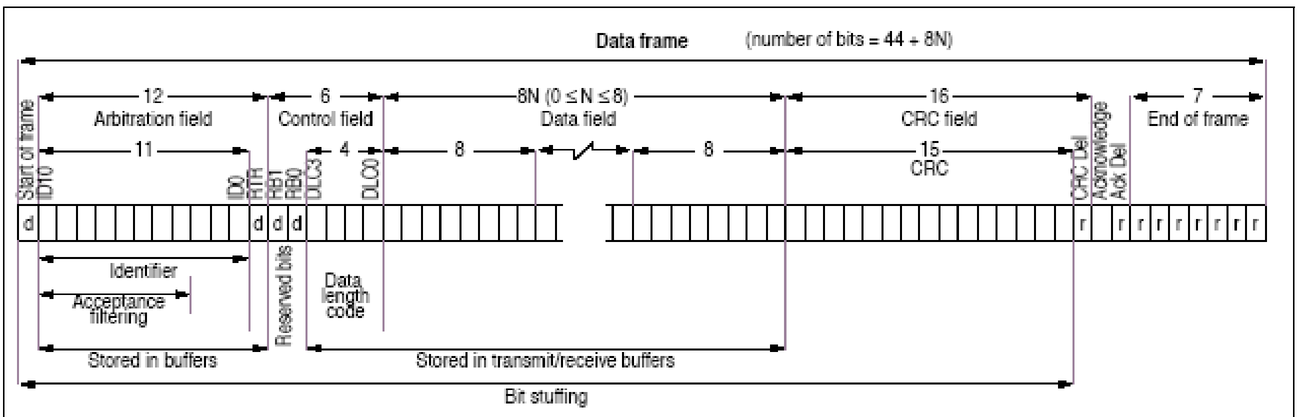


Figure 1 CAN Data Frame

Bit Timing:

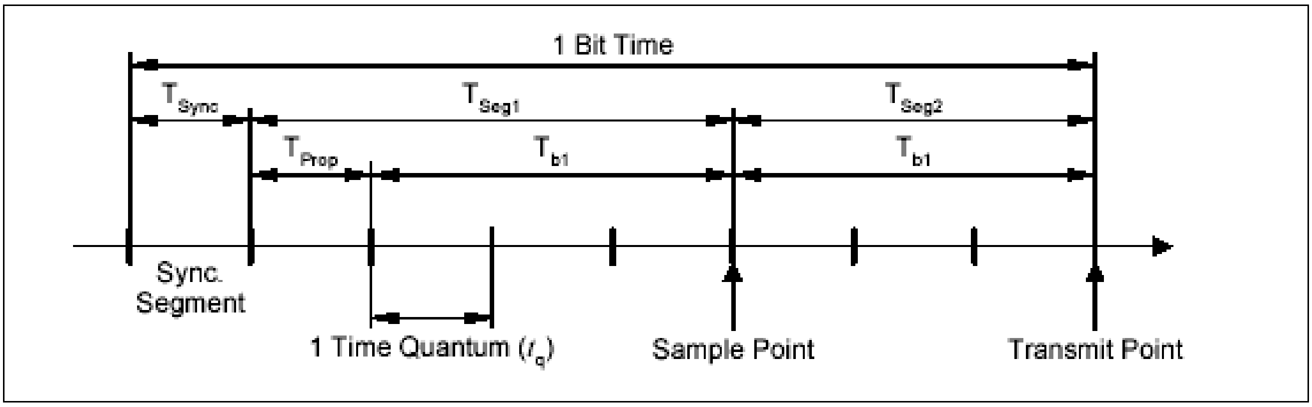


Figure 2 CAN Bit Timing

Detection mechanisms:

- bit monitoring Remote frame
- bit stuffing
- frame check
- acknowledgement check
- cyclic redundancy check

Fault confinement:

- 4 states
  - normal
  - error active
  - error passive
  - bus off
- registers
  - transmit error count
  - receive error count

3 Time Triggered CAN (TTCAN)

Motivation of TTCAN:

- Advantages time triggered systems
  Under the bit wise arbitration of CAN, the access may be delayed, if some other message is already in the process of transmission or if another message with higher priority also competes for the bus. Even the message with the highest priority may experience a small latency. The lower the priority of a message is, the higher the latency jitter for the media access.
- Predictability
  a deterministic communication pattern on the bus is guaranteed under high load.
- Features: same protocol and physical layer as the well now CAN system.
  development experience still usable.
- Features: time master can be distributed (8 potential time master).

TTCAN mechanisms: How do the **TTCAN** messages become triggered?

- each **TTCAN** controller is equipped with a local time counter (**local time**).
- the l**ocal time** is incremented each **NTU** (network time unit).
- synchronization of each CAN nodes is achieved by a periodic transmission of a **reference message** (transmitted by **time master**).
- the cycle based sending/receiving of the messages is controlled by predefined **system matrix** with **cycle time**.
- a **global Time** is required for distributed system design and for support of OSEK time operation systems. the time master establishes its **local time** as **global time** by transmitting **reference message**.

Two level extensions: **TTCAN** will be implemented in two levels.

- **Level 1**: is restricted to the cycle message transfer only. Time triggered operation is provided by **cycle time**.
  - guarantees the time triggered operation of CAN based on the **reference message** of a **time master**.
  - fault tolerance of that functionality is established by **potential time masters**.
- **Level 2**: in addition supports a **global time**. It additionally provides increased synchronization quality and external clock synchronization.
  - a continuous drift correction among the CAN controllers is realized (with help of **Time Unit Ratio**).

TTCAN implementation:

The **TTCAN** is expanded by two functional blocks, the **trigger memory** and the **frame synchronization entity** (**FSE**).

The **trigger memory** stores the **time marks** of the **system matrix** that are linked to the messages in the message RAM.

The **frame synchronization entity** is the state machine that controls the time triggered communication. It synchronizes itself to the **reference messages** on the **CAN** bus, controls the **cycle time**, and generates **time triggers** to provide the time triggered communication schedule.

It is divided into six blocks:

- **Time base builder** (generation of **NTU** and automatic adjust)
- **Time schedule organizer** (trigger memory)
- **Cycle time controller unit** (time trigger control and configuration)
- **Master state administrator** (current and backup master, strictly time-triggered behaviour)
- **Application operation monitor** (status register)
- **Failure handling** (error reports)

4 MultiCAN Module in TC1796

4.1 MultiCAN Feature

- It contains 4 nodes with Full-CAN functionality, up to 1MBaud
- List structure (more flexible)
- Analyzer Mode.
  - Listening to the bus, without taking part of the protocol
  - Auto-baud rate detection (No error frames on the bus)
- Up to 128 Message Object
- Programmable acceptance mask filtering for each message object
- V2.0 B active for each CAN node
- Baud rate programmable for each node
- FIFO and Gateway functionality

Especially for TTCAN:

- CAN node 0 supports TTCAN functionality (level 1 and level 2) with event-driven or time triggered mode
- Scheduler and timing synchronization unit
- Timing-related interrupt functionality
- Alternate message feature. It is possible to send out another message inside an exclusive window to indicate that the assigned message is not ready.
- Arbitration windows supported in time triggered mode
- Timing related Interrupt functionality
- Usable as time master and global time information is available

4.2 Module Overview

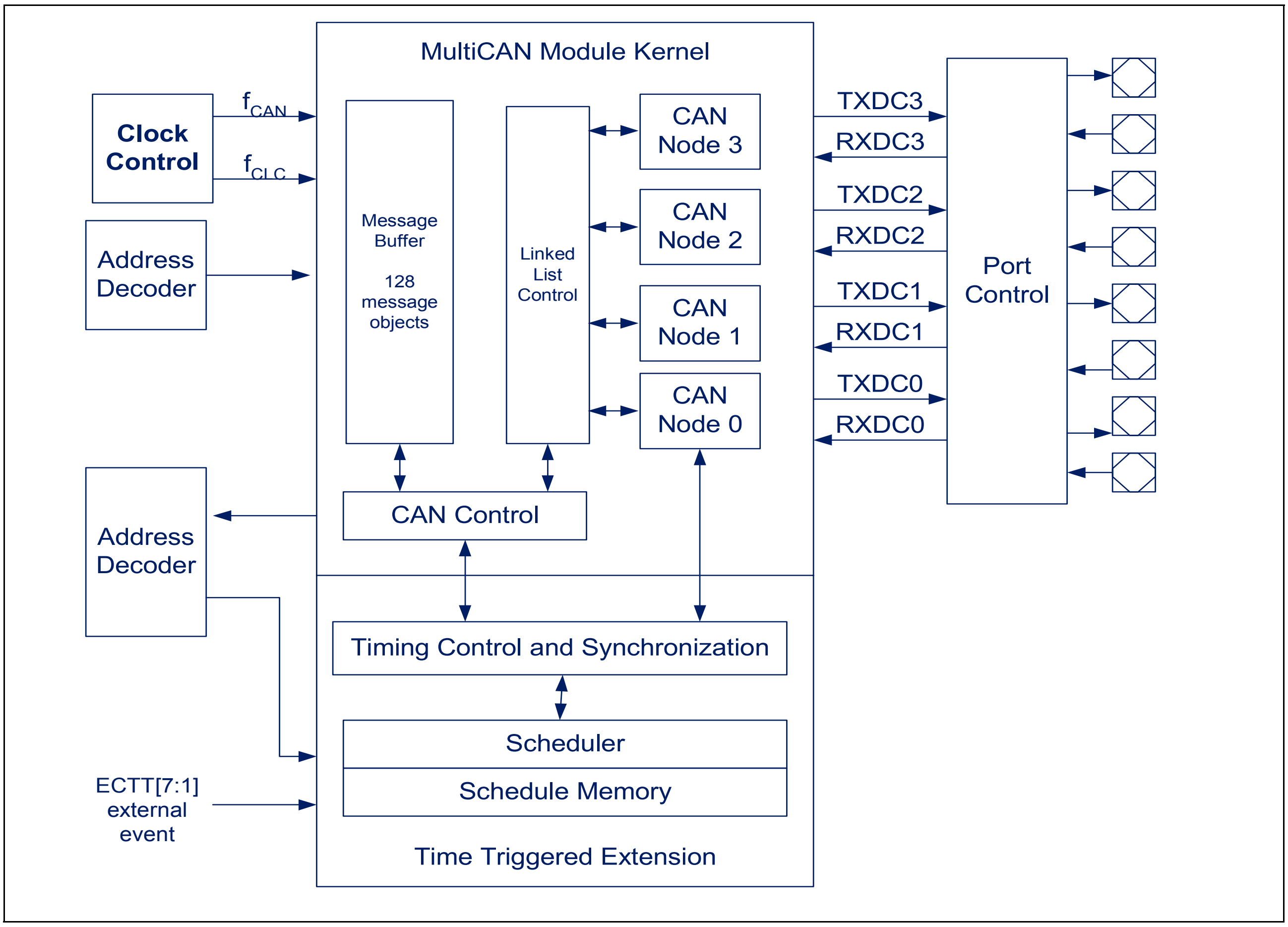


Figure 3 Overview of the MultiCAN Module

4.3 Network Time Unit (NTU) and Local Time Generation (LTG)

4.3.1 ISO Terms and Definition

Network time unit (**NTU**): it is the unit in which all times are measured.

- in **Level 1**: it is the **nominal bit time**.
- in **Level 2**: it is a fraction of a physical second.

Local time: it is measured in **NTU**.

- in **Level 1**: a 16 bit integer value without any fractional part. It is incremented each **NTU** = one nominal **CAN bit time**.
- in **Level 2**: 16 bits of the non fractional part plus at least 3 fractional bits. **MultiCAN** uses 10 bits as fractional part, **local time** is incremented 2^{10} times, each increment is the local equivalent of **NTU**/1024.
- The value of **local time** is captured as **sync mark** (at the pulse of frame synchronization (given by **SOF** of each send message), the current value of the **local**

time is saved as **sync mark**) at the SOF bit of each message. When a valid **reference message** is received, this message's **sync mark** becomes the new **ref mark**.

- The capturing of **local time** into **sync mark** at each **SOF** must be done in hardware **TUR** (Time Unit Ration, **level 2** only) and automatic **TUR** adjust.
  All **FSE** (Frame Synchronization Entity) do have there own oscillator, which provides clock ticks. **TUR** is used for clock synchronization. It takes care for the correct relation between the **system clock** generated by oscillator and **NTU**. **TUR** is specific for each **FSE**. The **NTU** is generated locally, based on the local system clock period t_{sys} and **TUR** (**NTU=TUR*** t_{sys}). CAN node 0 can automatically calculate the new value that is written to **TURADJ** for adjusting the correct value for the **local time**.

4.3.2 MultiCAN Implementation

The following **Figure 4** shows the overview of the time block builder in MultiCAN.

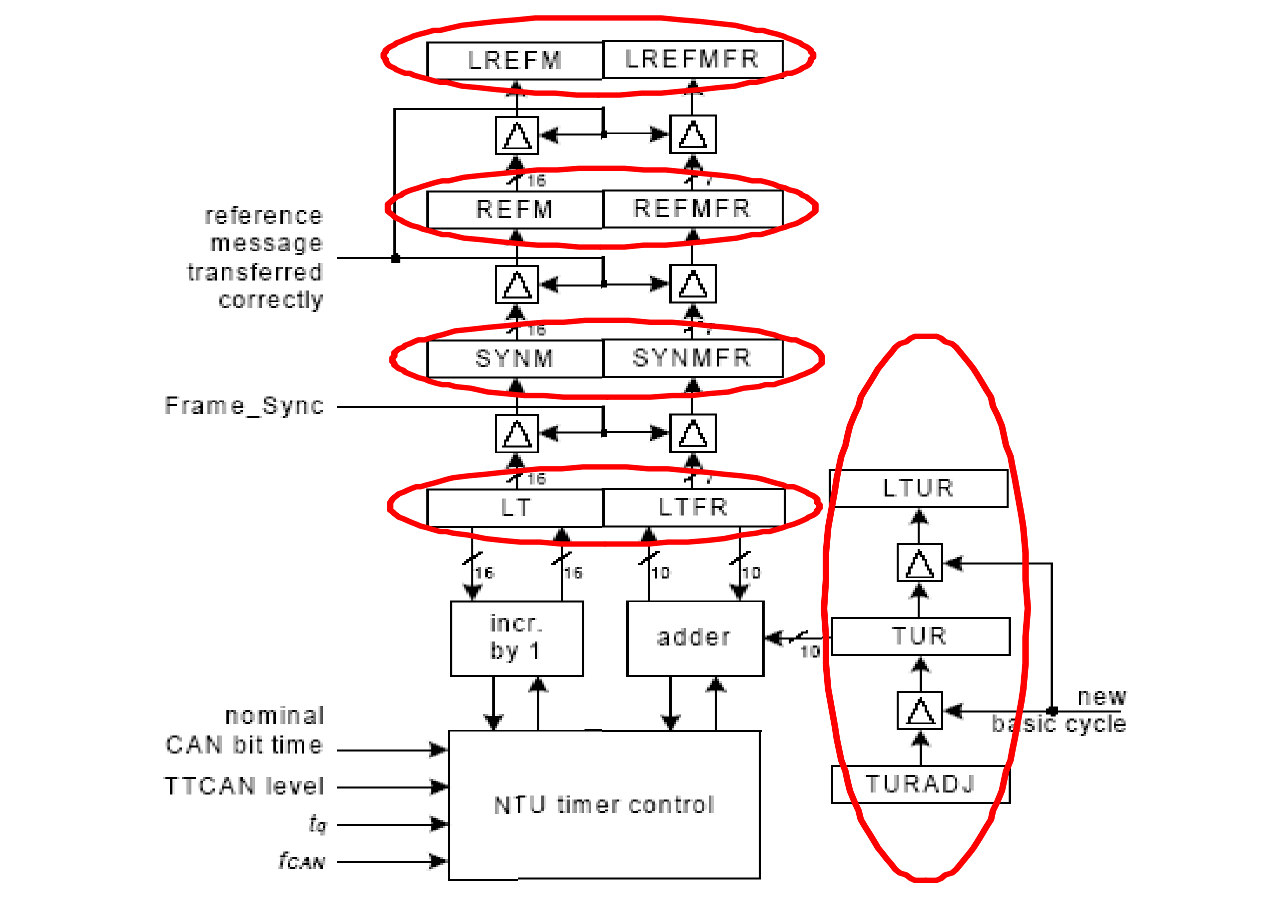


Figure 4 Generation of the Local Time

Register **SYNMR** (synchronization mark register): SYNM + SYNMFR

Register **REFMR** (reference mark register): REFM + REFMFR

Register **LREFMR** (last reference mark register): LRFEF + LRFEFFR

Register **GMR** (global mark register): GM + GMFR

Register **LGMR** (last global mark register): LG + LGFR

– all above registers are status register with 16 bits integer and 7 bits fraction parts

Register **LTR** (local time register):

– LT(16 bits, integer part of **NTU**) + LTFR (10 bits, fractional part of **NTU**)

Register **TURR** (time unit ratio register):

– bits TURADJ: **TUR** adjust
– bit ADJEN: adjust enable
– bit VAL: Valid
– bit LTCS: clock source selection for the local time. 0b:=t_q 1b:=f_{CAN}
– bits LTDIV: local time divider (if f_{CAN} is selected for local time generation)
– bits TUR: actual time unit ratio value

- Each time a **new reference** message is correctly received, the difference between the time values in the reference message (**GM**) and one before (**LGM**) is calculated
- The value of **TUR** is updated with the **beginning** of each **basic cycle**:
  TURADJ = **LTUR** * (**GM** - **LGM**) / (**REFM** - **LREFM**)

4.4 Time Schedule Organizer (Trigger Memory) and System Matrix

4.4.1 ISO Terms and Definition for Trigger Memory

The Trigger Memory stores the **time marks** of the **system matrix** that are linked to the messages in the message RAM; the data is provided to the Frame Synchronization Entity (**FSE**).

Basic cycle: Its elements are several consecutive **time windows**. Each **basic cycle** is starting with the **time window** for the **reference message**. Different **basic cycles** are distinguished by the **cycle count**, a **cycle count** is incremented each cycle up to the maximum value after which it is restarted again (cycle count: number of **basic cycles**).

*Note: in **TTCAN** not all **basic cycles** necessarily have to be the same.*

*Note: the number of **basic cycles** within a matrix cycle (**cycle count max** + 1) shall be an integer power 2.*

Cycle time: it is the actual difference between **local time** and **local ref mark**, restarting at the beginning of each **basic cycle** when **ref mark** is reloaded. It represents the time elapsed in the current **basic cycle**, starting from 0.

Repeat factor: it specifies the repetition rate of a message. It determines the number of **basic cycles** between two successive transmissions/receptions of the message.

Time mark: it specifies an instant of **cycle time** at which a certain action is expected or planned. Furthermore, it consists of the **base mark** and the **repeat factor** information.

Base mark: it determines the number of the first **basic cycle** after the beginning of the **matrix cycle** in which the message must be sent/received.

*Note: the first **time mark** must not be lower than max. length of the reference message + 5 bit times.*

4.4.2 ISO Terms and Definition for System Matrix

In a time triggered system, all messages of all nodes in the network are organized as components of a **system matrix**. The system matrix itself consists of **time windows**, organized in **basic cycles** (rows) and transmission columns (columns). It specifies the sequence of messages transmitted in each **basic cycle**. It represents the communication overview of a **TTCAN** network.

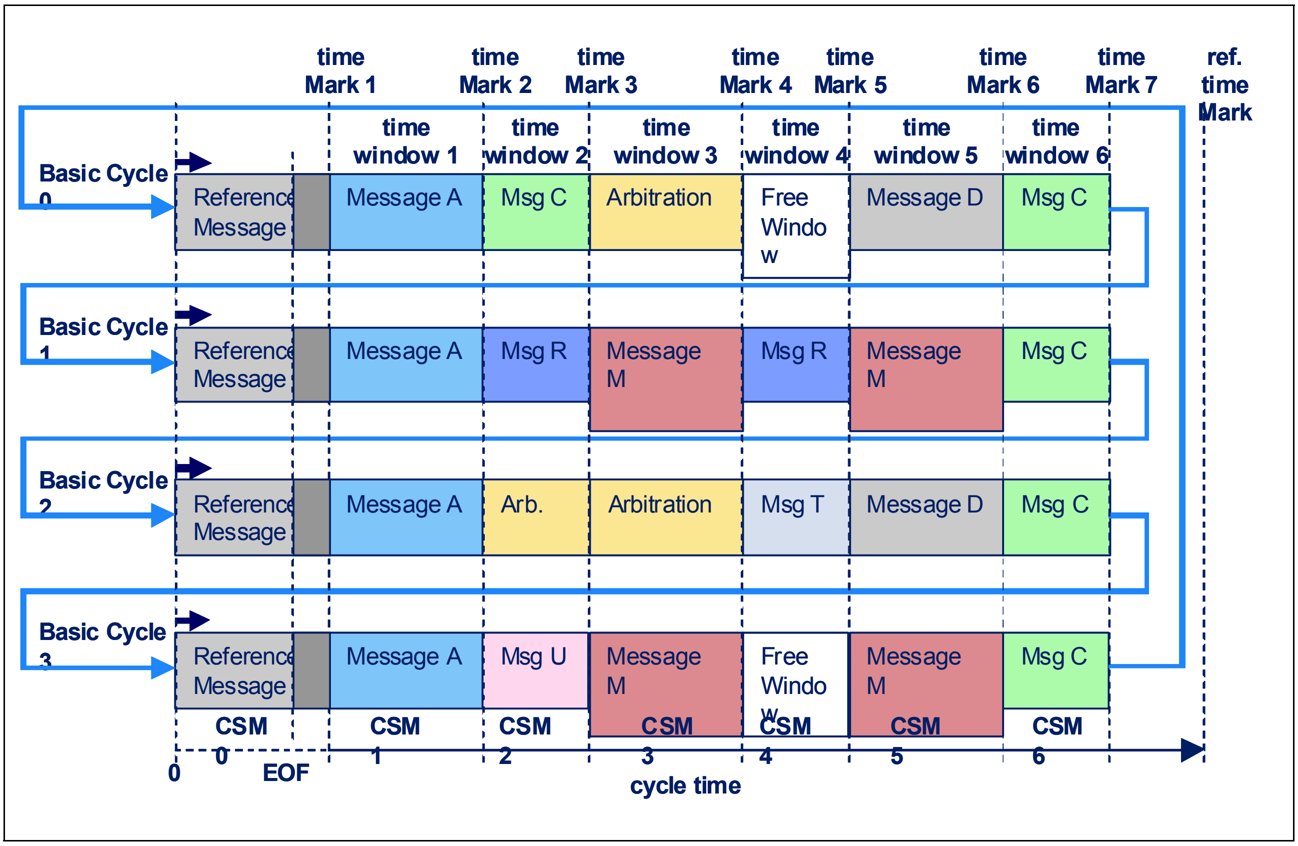


Figure 5 System Matrix

Three different types of **time windows**:

Exclusive time window: is assigned to periodic messages.

- exclusively reserved for one message without competition for **CAN** network access.
- the automatic retransmission of messages that could not be transmitted successfully is disabled, guaranteeing that messages in **exclusive time windows** are not delayed.
- Only one node in network may start a transmission in an **exclusive window**.

- The periodic transmission will be started by **scheduler entries**.

Arbitrating time window: is assigned to spontaneous messages.

- several CAN nodes in the network may start a transmission within **Tx Enable window** of an **arbitrating time window**. messages can compete for the bus by the bit wise arbitrating mechanism of CAN.
- Two types: **long merged arbitration time window/short single arbitration time window**.
- The automatic retransmission is allowed within **merged arbitration time window**.

Free time window: time windows are free if no messages are scheduled in the **system matrix** for those window. It is reserved for further extensions of the network.

4.4.3 MultiCAN Implementation

4.4.3.1 Overview of the Time Scheduler in MultiCAN

The scheduler stores the **time marks** of the **system matrix** and the instructions based on the **cycle time**.

The message transmission and reception of **TTCAN** is controlled by a scheduler mechanism. This scheduler is based on the **cycle time** and delivers the **time marks**. The time marks are defined by the time mark entries **TMEx**. Whenever a **time mark** is reached, programmable actions (defined by instructions entries **INSTRxy**) can take place.

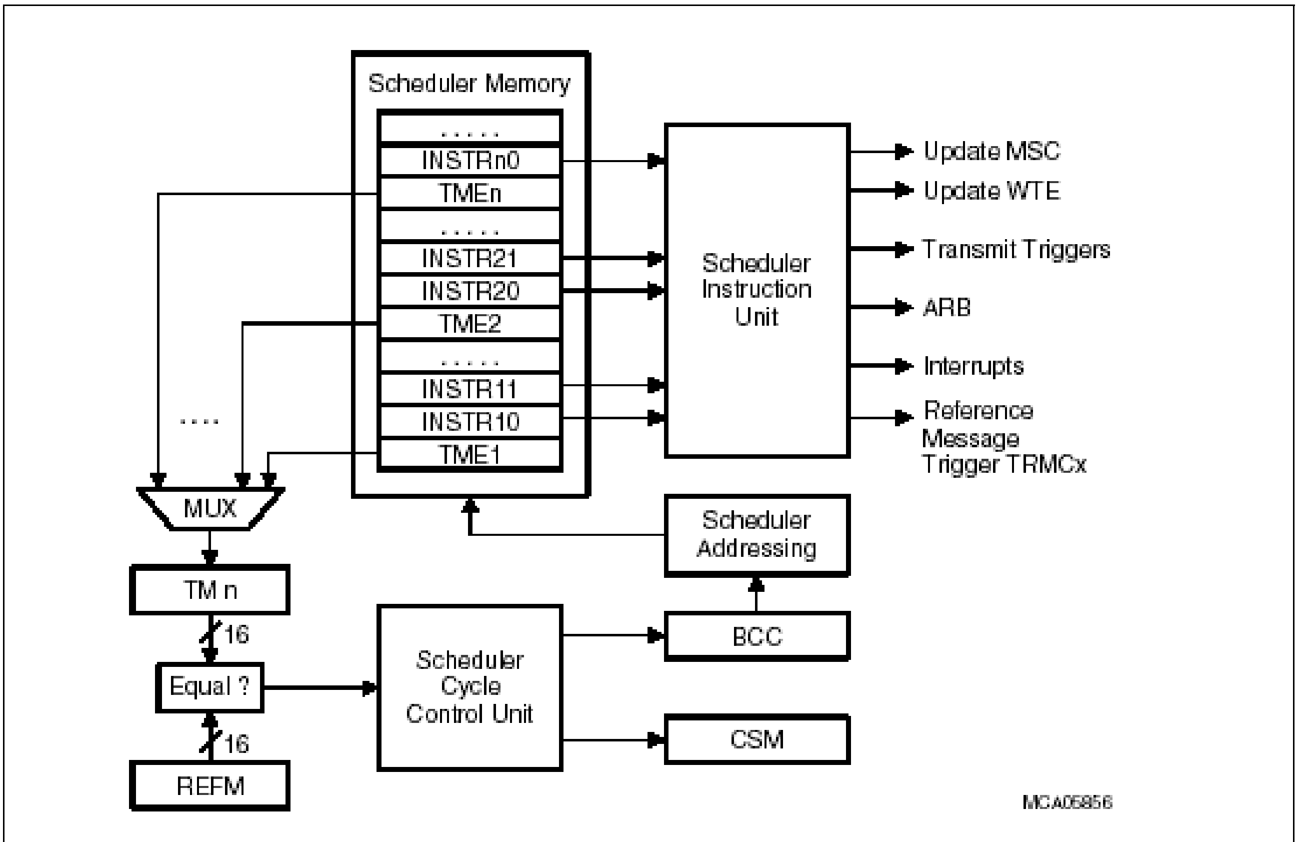


Figure 6 Scheduler Overview

Scheduler memory contains the time mark entries (**TMEx**) and the scheduler instruction entries (**INSTRns**).

- **TME** determines the behavior of the TTCAN when the next time mark is reached.
- **INSTR** contains the action and the transfer behavior.

MultiCAN has a total number of 8 entry types, it will be described more details in **Section 4.4.3.2**.

In MultiCAN the scheduler memory has a size of 128 words and the start address is 0xF000,5600.

The last word (32-bits) address of the scheduler memory is reserved for the start pointer STPTR0.

The value written at this address determines the start location of the first entry for the

TTCAN node. STPTR0 indicates how many entries below STPTR0 the first time mark entry (TME1) is located.

Reading by the scheduler is started with the basic cycle end entry (**BCE**), **TME1**, **INSTR10**, **INSTR11**, ..., **TME2**, **INSTR20**, **INSTR21**...

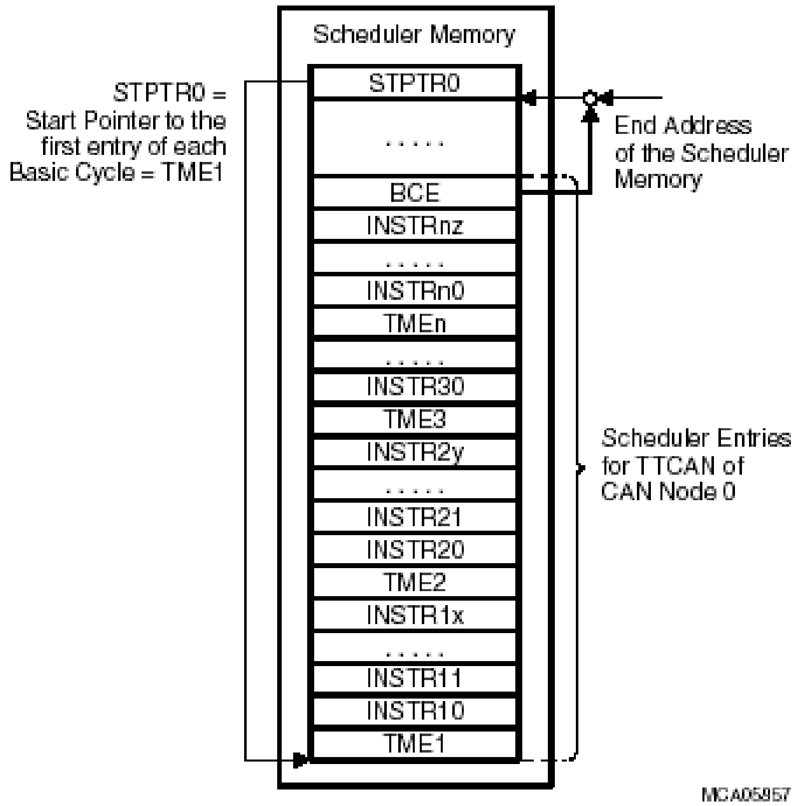


Figure 7 Scheduler Memory

4.4.3.2 Scheduler Entry Types

MultiCAN has 8 entry types (one time entry type and 7 instruction entry types). Each entry uses 4-bit wide code (EC) that indicates the type of the entry and other bits fields define the time or the activities. Each entry is 32-bit wide.

time mark entry (EC=0001b): defines the behaviour of the **TTCAN** device when the next **time mark** is reached. Each time mark entry can be followed by a number of **scheduler instructions**.

```
32-bit instruction code:
EC(0001b) | 4-bit interrupt node pointer | 16-bit time mark value
```

Note: The number of scheduler entries following a time mark shouldn't exceed the number 10.

interrupt control entry(EC=0010b): generate an interrupt when the **time mark** n ins reached

```
32-bit instruction code:
EC(0010b) | interrupt line | cycle offset | repeat factor
```

arbitration entry (EC=0011b): defines the behaviour of the **time window** starting with the time mark n

```
32-bit instruction code:
EC(0011b) | Arbitration mode) | cycle offset | repeat factor
```

transmit control entry (EC=0100b): defines a **MSG** to be transmitted after the **time mark** n.

*Note: the transmission start ins possible while the **transmit enable window***

```
32-bit instruction code:
EC(0100b) | MSG | alternative MSG | tx enable bit | cycle offset | repeat factor
```

receive control entry (EC=0101b: controls the **receive message** information in the **time window** between the **time mark** n-1 and n

```
32-bit instruction code:
EC(0101b) | MSG | check enable bit | cycle offset | repeat factor
```

reference message entry (EC=0110b)

```
32-bit instruction code:
EC(0110b) | time mark value | Gap Mode
```

basic cycle end entry (EC=0111b): ins the watch trigger that ins used to generate a **watch trigger event** when the **cycle time** reaches this defined **time window**

```
32-bit instruction code:
EC(0111b) | time mark value | Gap Mode
```

end of scheduler memory entry (EC=other combinations): immediately stops the reading of entries in the scheduler memory and sets the **TTCAN** node in **configuration mode**.

```
32-bit instruction code:
| EC(1xxxb) |
```

4.4.3.3 Setup of the Schedules Entries in MultiCAN

- The entries in the scheduler memory can only be set up while bit **NCR0**.**CCE** of the **TTCAN nodes 0** the scheduler memory is set at the same time.
- The **scheduler instruction entries** are always 32 bit wide.
- The total amount of **time mark entries** and **scheduler instruction entries** is limited by the size of the scheduler memory (128 instruction).
- The value of **STPTR0** indicates how many entries (counted in 32 bit words) below **SPTR0** the first **time mark entry** (**TME1**) can be found.

To configure the scheduler memory the register **STPTR0** (scheduler start pointer node 0 register) should be configured first. If the maximum value 127 is initialized in the bits filed **STPTR0**.**STPTR**, the first 32-bit entry (**TME1**) should be written at address 0xF0005600.

4.5 Cycle Time Controller Unit (Time Trigger Control)

4.5.1 ISO Terms and Definition

Cycle time unit controls trigger Information and sending/receiving handling.

In **TTCAN** all messages in the network are organized as components of a **system matrix**.

The timing of **TTCAN** (**time trigger control**) is based on the matching of **time marks** with the current **cycle time**. The **time marks** are essential parts of the **TTCAN** triggers.

Rx trigger: It is specified, when the reception of a message shall be verified.

The necessary information for it is:

- **Time mark** (after which the reception of this message is expected to be completed
- **Cycle offset** (the position within the transmission column in respect to the first reception)
- repeat factor
- only in **exclusive window**

***Tx trigger**:* it specifies the beginning of the message's time window

- **time mark**
- **cycle offset** (the within transmission column in respect to first sending
- **repeat factor**
- corresponding message object
- in e**xclusive window** or **arbitrating time windows.**

Tx ref trigger: it triggers the **reference message**, to start the next **basic cycle**

Tx enable window: it is a window which restricts the starting event to send a message to a specified time period.

- It is opened at the beginning of the message's time window (with **Tx trigger**) and it is closed after a specified number of nominal CAN bit times (1..16) specified by the system configuration.
- Within a **time window** the transmission of a message may be started during the **Tx Enable window** (bit **SOF** of the message shall be within this window).
- It may vary for the different **TTCAN** controllers in the network.
- In arbitration window the **Tx enable window** starts at the beginning of the first **time window** and ends at the end of the **Tx enable window** of the last **merged window**.

Note: If the bus is not idle during this initial phase of the time window, then the message will fail to be transmitted. This requirement is necessary to ensure that messages may not be released so late in a time window as to excessively delay the release of the subsequent message in the next time window.

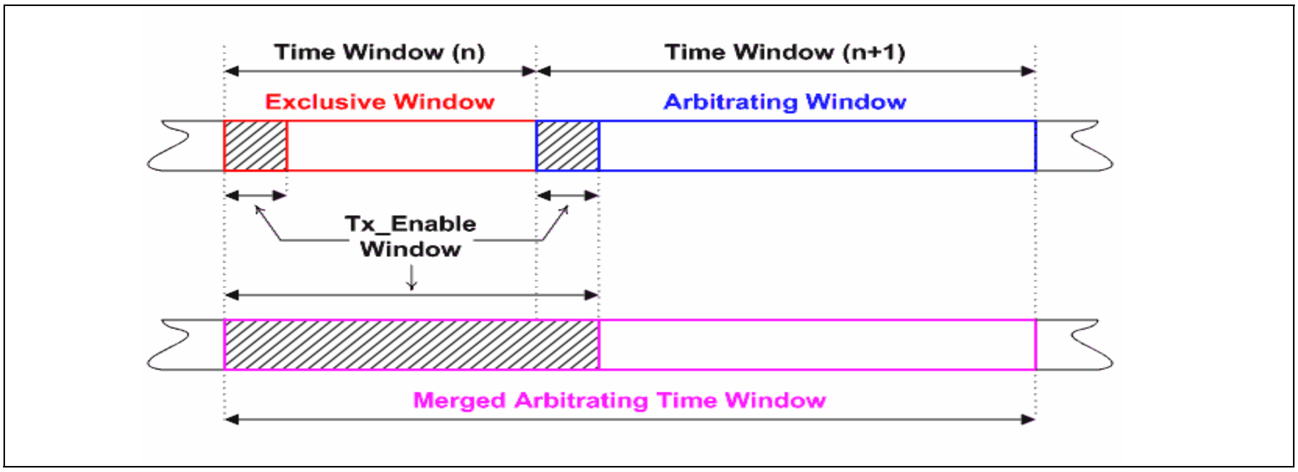


Figure 8 Time Window

reference message: Time-triggered and periodic communications clocked by a **time *master's reference message:*** each valid **reference message** starts a new **basic cycle** and causes a reset of each node's **cycle time**.

– in **Level 1**: the **reference message** contains at least one data byte.
– in **Level 2**: consists of at least four data bytes.

Byte 1: Next_Is_Gap bit and cycle count

7	6	5	4	3	2	1	0
Next_Is_ Gap	Reserved	Cycle Count (5)	Cycle Count (4)	Cycle Count (3)	Cycle Count (2)	Cycle Count (1)	Cycle Count (0)

Byte 2: Fractional part NTU_Res of Master_Ref_Mark and discontinuity bit

7	6	5	4	3	2	1	0
NTU_Res (6)	NTU_Res (5)	NTU_Res (4)	NTU_Res (3)	NTU_Res (2)	NTU_Res (1)	NTU_Res (0)	Disc_Bit

Byte 3: Low byte of Master_Ref_Mark

7	6	5	4	3	2	1	0
MRM (7)	MRM (6)	MRM (5)	MRM (4)	MRM (3)	MRM (2)	MRM (1)	MRM (0)

Byte 4: High byte of Master_Ref_Mark

7	6	5	4	3	2	1	0
MRM (15)	MRM (14)	MRM (13)	MRM (12)	MRM (11)	MRM (10)	MRM (9)	MRM (8)

Figure 9 Reference Message in Level 2

– **MRM**:= **local time** of master
– **Disc_bit**: instead of a **reference message** a **gap** will delay operation
– **NTU_Res**: the counter is at least 19 bit where all but the 16 **MSB** give fractional parts of an **NTU** (the counter counts in units of **NTU**/2^n if **NTU_Res** coves n)
– **Cycle_count**

– **Next_ins_gap**

trigger for the reference message:

– **time mark** (Reference message entry) has reached: time triggered communication:
– **external event trigger**: An edge at an external input / Software trigger (special synchronized start of a basic cycle)

time gap: in a **Level 2** system, which not completely time triggered, a **time gap** can be used. When **next is gap** is set inside a **reference message**, at the end of this **basic cycle** a **time gap** will take place. the cycle message transfer ins discontinued. The system will wake up again on an event.

In some applications it can be advantageous to trigger the transmission of a **reference message** by an event external to the bus. In this case the application has to signal this to all other bus members in the reference message *preceding* this event synchronization by setting one bit in this previous **reference message**, the **Next_Is_Gap** bit.

Bit **Next_Is_Gap** configuration:

– If the **reference message** is triggered by a **time mark**: reset bit **Next_Is_Gap**
– If the **reference message** is triggered by **external event**: set bit **Next_Is_Gap** to 1

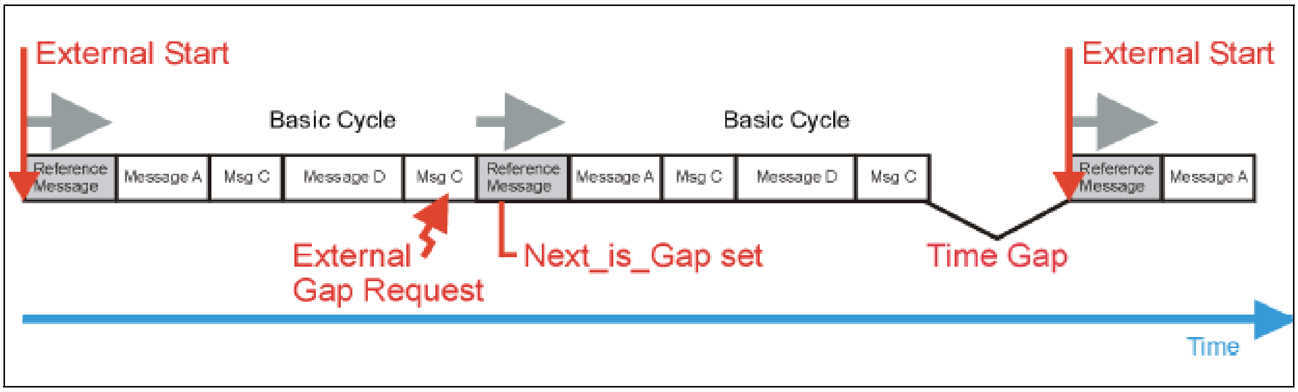


Figure 10 Gap

4.5.2 Cycle Time Controller Unit in MultiCAN

Cycle control unit delivers the values for the **basic cycle** and **matrix cycle** handling, including the possibility to generate interrupt if a new **basic cycle** has started or a new **matrix cycle** has started. If the **TTCAN** node receives a **reference message**, the value for **BCC** is taken from the **reference message**.

Each time a **reference message** is received correctly, the cycle control unit starts again comparing the **cycle time** to the first **time mark** (TM1).

The cycle control unit also elaborates the number of the current **basic cycle** that ins indicated by the basic cycle count **BCC**. The value of **BCC** together with the value of **CSM** clearly identify each **time window** in the **matrix cycle**.

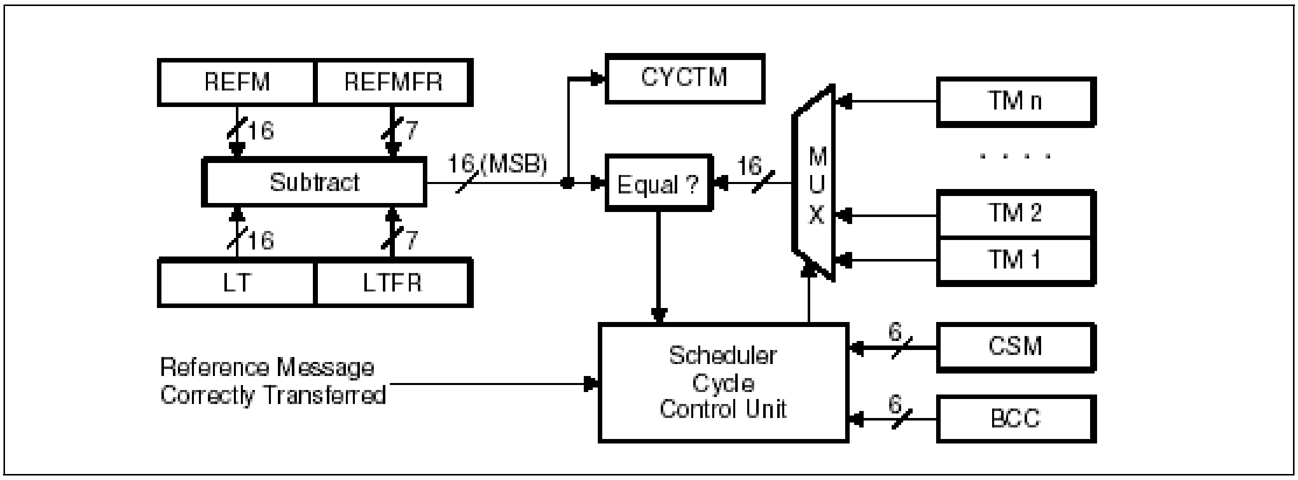


Figure 11 Generation of the Cycle Time

Register **TTCR** (Time Trigger Control Register):

Bits	31–28	27–24	23–22	21–16
Field	RMDLC	TENW	0	CYCLE
Access	rw	rw	r	rw

Bits	15	14–12	11–9	8	7	6–4	3–2	1–0
Field	0	TMPRIO	0	TTLVL	ETM	ETSSEL	ETESEL	TTM
Access	r	rw	r	rw	rw	rw	rw	rw

bits **RMDLC**: reference message DLC (Level 1: DLC=1..8, Level 2: DLC=4..8)

bits **TENW**: Tx Enable Window. how many CAN bit times can elapse before a pending Tx trigger is discarded (1..16 CAN bit times

bits **CYCLE**: Basic Cycle Number. The number of the last basic cycle in the matrix cycle (0..127)

bits **TMPRIO**: time master priority, used for the ID bits 2..0 in the reference message

bit **TTLVL**: TTCAN Level:

bit **TEM**: external Trigger Mode:

0b: HW trigger event is active if it occurs after
1b: SW (stored ETREV) trigger event in is active

bits **ETSSEL**: external trigger source selection

bits **ETESEL**: external trigger event selection

bits **TTM**: time trigger mode

00b: no TTCAN functionality
01b: time slave
10b: actual or potential time Master

Register **TTFMR** (Time Trigger Flag Modification Register):

31–16
0
r

15–6	5	4	3–2	1–0
0	ETR EVR	STE	NIGSR	CFGMEL
r	w	w	w	w

bits **CFGMEL**: configuration mode enter/leave

bits **NIGSR**: next is gap set/reset

bit **STE**: software trigger event

bit **ETREVR**: reset external trigger event

Register **TTCFGR** (Time Trigger Configuration Register): Configuration Register

31–24	23–16
TTCNT	RTO
rh	rh

15–8	7	6–0
EXPTT	0	IRO
rw	r	rw

bits **TTCNT**: transmit trigger counter

bits **RTO**: reference trigger offset (indicates the actual reference trigger offset)

bits **EXPTT**: expected transmit triggers (how may tx requests are expected in a matrix cycle)

bits **IRO**: initial reference offset. 0..127

4.6 Master State Administrator

Master State Administrator (current and backup master):

Up to 8 **FSEs** of a **TTCAN** network may be **potential time masters**, only one of them becomes the current time master once the normal time triggered bus communication is established.

At system start up, after the hardware reset, all **potential time masters** will perform the function of **time master** and will try to send - according to a defined priority (3 **LSBs** of **ID**) and waiting time (**ref trigger offset**) - a **reference message.**

The **FSE** with the highest priority uses the highest **CAN ID** and starts its transmission the shortest time after its hardware reset.

The reference message priorities of different potential time masters may only differ in the three **LSBs**.

4.6.1 MultiCAN Implementation

Time trigger mode is configured in **TTCR** register (bits field **TTM**)

Register **TTSR** indicates the status information:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					ETR EV	ETR	NIG	0	MSCMAX			0	MSCMIN		
r					rh	rh	rh	r	rh			r	rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REC F	TRA F	TM PC	CFG M	ARB	REF TRG	EFF	EFI	0		SYNCS		MSR		ERRS	
rh	rh	rh	rh	rh	rh	rh	rh	r		rh		rh		rh	

bits **MSR**: master-slave relation

bit **NIG**: next is Gap

bits **SYNCS**: synchronization state

00b: Send-off, no syn. activity in progress

01b: Synchronizing, FSE is in syn. process

10b: In_Gap, FSE is synchronized, gap expected

11b: In_Schedule, FSE is synchronized, no gap

The **configuration mode** is entered automatically after reset or by SW (write **TTFMR**.**CFGMEL**=01).

- During the configuration: **local time**, **global time**, and **cycle time** are reset.
- **CAN** node and **MSGs** should be configured
- For **TTCAN** the Schedule memory and TUR has to be set up completely

It can be left by only by SW (write **TTFMR**.**CFGMEL**=10).

- **local time** starts after leaving the configuration mode
- The **synchronization phase** is entered automatically
- For **time masters**, the transmission of the **reference messages** are scheduled like in a **gap** while the **TTCAN** node is in the state ’**synchronizing**’.

The system must be synchronized after finishing the **configuration phase**.

’Synchronizing’ is restarted by the reception of a **reference message** or (for **potential time masters**) by entering and leaving the **configuration mode**. ’Synchronizing’ is completed when two successive reference messages have been observed,

- an **FSE** regards itself synchronized to the network after the occurrence of the second consecutive **reference message**
- During synchronization, nodes will not transmit any messages and **global time** is regarded as invalid.
- The **init watch trigger** is taken into account until the first message is correctly received or transmitted. An **init watch trigger event** is detected when the **cycle time** reaches the value of 2^{16}-1.

4.7 Operation Monitor and Failure Handling

ISO terms and definition:

Tx count: it resets at start of each **matrix cycle**. It increments on each active trigger: **Tx count** $_{max}$**= expected Tx trigger**.

In case the maximum number of **Tx triggers** has been reached, no further transmission in **exclusive** time windows takes place for this device.

Tx overflow: in case **Tx trigger** reaches the **expected Tx trigger** value, the **Tx overflo**w flag will be set

Tx underflow: in case not all **Tx triggers** became active within a **matrix cycle**

Expected Tx trigger: threshold value of maximum amount of **Tx triggers** in one **matrix cycle**

MultiCAN Implementation:

- 8-bit field EXPTT (**expected Tx triggers**) in the register **TTCFGR** (time trigger configuration register)
- 4-bit field **MSC(0…7)** in the register **MOFCRn** (message object function register) indicates message status count. It is to detect scheduling errors for **exclusive time windows**. A scheduling error is detected, when **MSC** is greater than 7 or the difference between the highest **MSC** and lowest **MSC** of all messages on this CAN node is larger than 2 within one **system matrix**.

4.7.1 MultiCAN Implementation

Four levels of error severity:

- **S0:** no error, normal operation
- **S1:** warning/only notification, interrupt flag is set
  - $\mathbf{MSC_{MAX}}$ – $\mathbf{MSC_{MIN}}$ > 2 at the end of a **matrix cycle**
  - A receive message object has reached its **MSC** of 7
  - **Tx underflow**: Not all transmit triggers were active.
- **S2:** error, interrupt flag is set. if **TTCFGR**.**RTO**=127 all transmission are disabled (except reference messages).
  - A transmit-message object reached an **MSC** of 7
  - **Tx overflow**, more than the number of specified **Tx triggers** has been taken

- **S3:** severe error. Init bit is set, all CAN bus operation is stopped.
  - Application Watchdog, the application failed to service the watchdog.
  - bus off
  - Configuration error: a merged arbitration window is not properly closed or a **Tx trigger** occurs in reference message **time window**
  - Watch trigger event: this trigger occurs if the **reference message** is missing (the time master stopped sending reference messages)

Three interrupt groups:

Error		**interrupt line**	**Flag**
TENWER	Tx Enable Window Error	ERRINP	TENWER
TTER	transmit trigger error	ERRINP	TTOF/TTUF
WTE	watch trigger event	ERRINP	IWTE/WTE
AWD	application watchdog	ERRINP	AWDERR
SE	scheduler error	ERRINP	SERR1/SERR2
Notification		**interrupt line**	**Flag**
NBC	new basic cycle	NBCINP	NBC
ERRSC	error state change	NOTIFINP	ERRS
MSRC	master-slave relation change	NOTIFINP	MSR
SYNCSC	synchronization state change	NOTIFINP	SYNCS
NOTIF	notification	NOTIFINP	WFE/DISC

TTCAN interrupt structure:

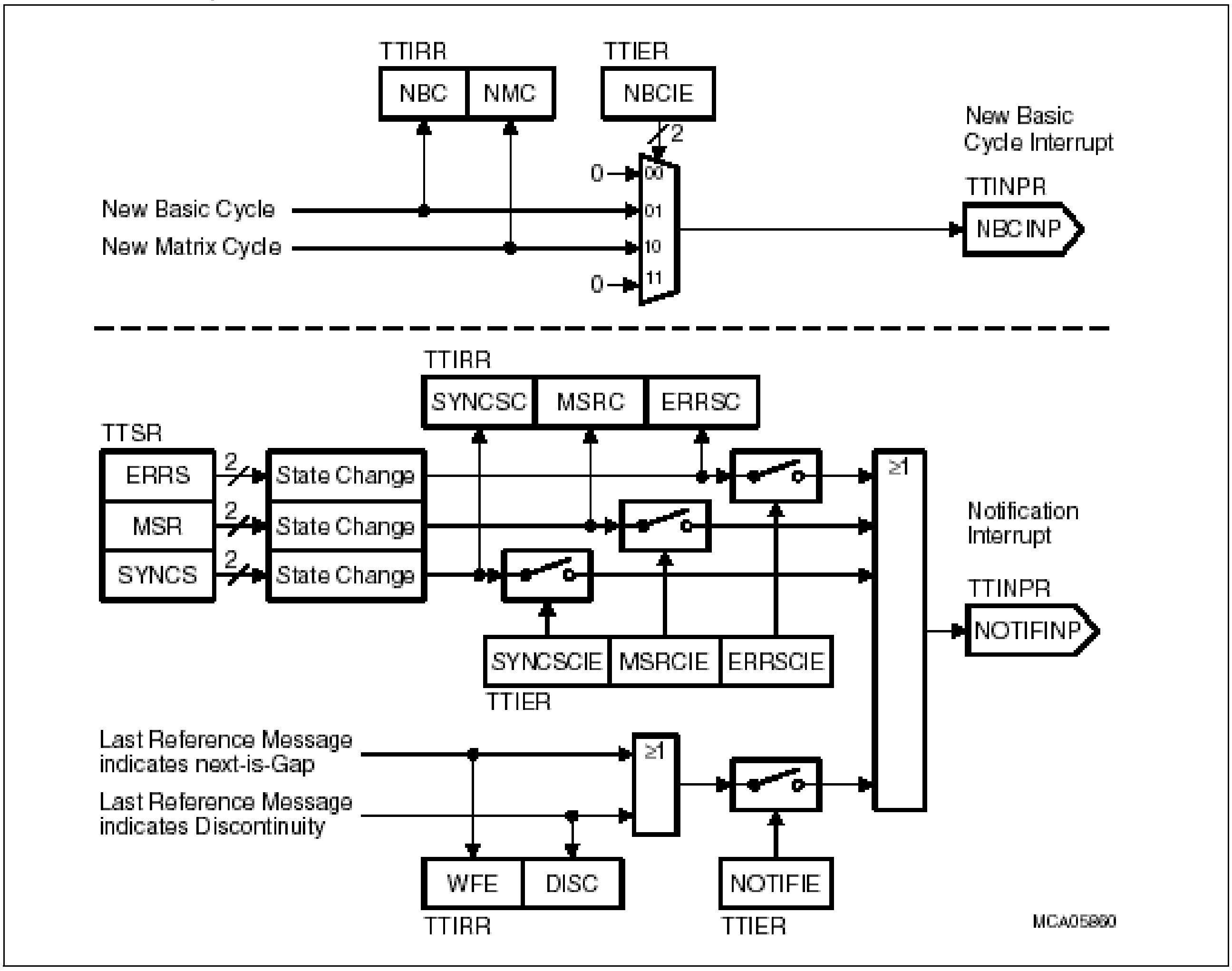


Figure 12 TTCAN Interrupt Structure

Register **TIER** (Time Trigger Interrupt Enable Register):

Bits	31–16
Field	0
Type	r

Bits	15–13	12	11	10	9	8	7	6	5–4	3	2	1–0
Field	0	NOTIFIE	SEIE	SYNCSCIE	MSRCIE	ERRSCIE	AWDIE	WTEIE	0	TTERIE	TENWERIE	NBCIE
Type	r	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Register **TTINPR** (Time Trigger Interrupt Node Pointer Register):

31–16
0
r

15–12	11–8	7–4	3–0
0	NOTIFINP	NBCINP	ERRINP
r	rw	rw	rw

Register **TTIRR** (Time Trigger Interrupt Request Register):

31–18	17	16
0	TURERR	CFGERR
r	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SERR2	SERR1	DISC	WFE	EOS	SYNCSC	MSRC	ERRSC	AWDERR	IWTE	WTE	TTOF	TTUF	TENWER	NBC	NMC
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

5 TTCAN Application Example

5.1 Application Example Description

There is a configuration example for TTCAN system. It consists two TTCAN node operating in level 2 at a bit rate of 500KBit/s. the two nodes have a system clock frequency of 40 MHz, the network time unit NTU is 2us. One node is a time master and another node is operating as a time slave.

the system matrix consists of four basic cycles 0…3, each basic cycle has five transmission columns at cycle time Time 0x0190(400), 0x0258(600), 0x03E8(1000), 0x0640(1600) and 0x7D0. The length of the Basic Cycle is 0x07D0 * NTUs=2000 * 2 us=4 ms.

Master transmits M_MSG2 and M_MSG3 in exclusive time window. Slave transmits S_MSG8 and S_MSG9 in exclusive time window. Master checks whether S_MSG8 and S_MSG9 are received on time. Slave checks whether M_MSG2 and M_MSG3 are received on time.

- Schedule setting and memory configuration for TTCAN master and slave node:

	0x0190	0x258	0x3E8	0x0640	0x07D0	0x2710
Basic cycle 0	M_MSG2	S_MSG8	-		Ref. MSG	BCE
Basic cycle 1	S_MSG9	S_MSG8	M_MSG3	-		
Basic cycle 2	M_MSG2	S_MSG8	-			
Basic cycle 3	S_MSG9	S_MSG8	M_MSG3	-		

- Master schedule entry

TM1	TM2	TM3	TM4	TM5
TME: 0x0190	TME: 0x0258	TME: 0x03E8	RME: 0x07D0	BCE: 0x2710
TCE (MSG2) (offset=0,Repeat=2)	RCE (MSG19) (offset=1,Repeat=2)	RCE (MSG18) (offset=0,Repeat=4)		
		TCE (MSG3) (offset=1,Repeat=2)		

- Slave schedule entry

TM1	TM2	TM3	TM4
TME: 0x0190	TME: 0x0258	TME: 0x0640	BCE: 0x2710
TCE (MSG9) (offset=1,Repeat=2)	TCE (MSG8) (offset=0,Repeat=4)		
	RCE (MSG12) (offset=0,Repeat=2)		

- Message configuration:

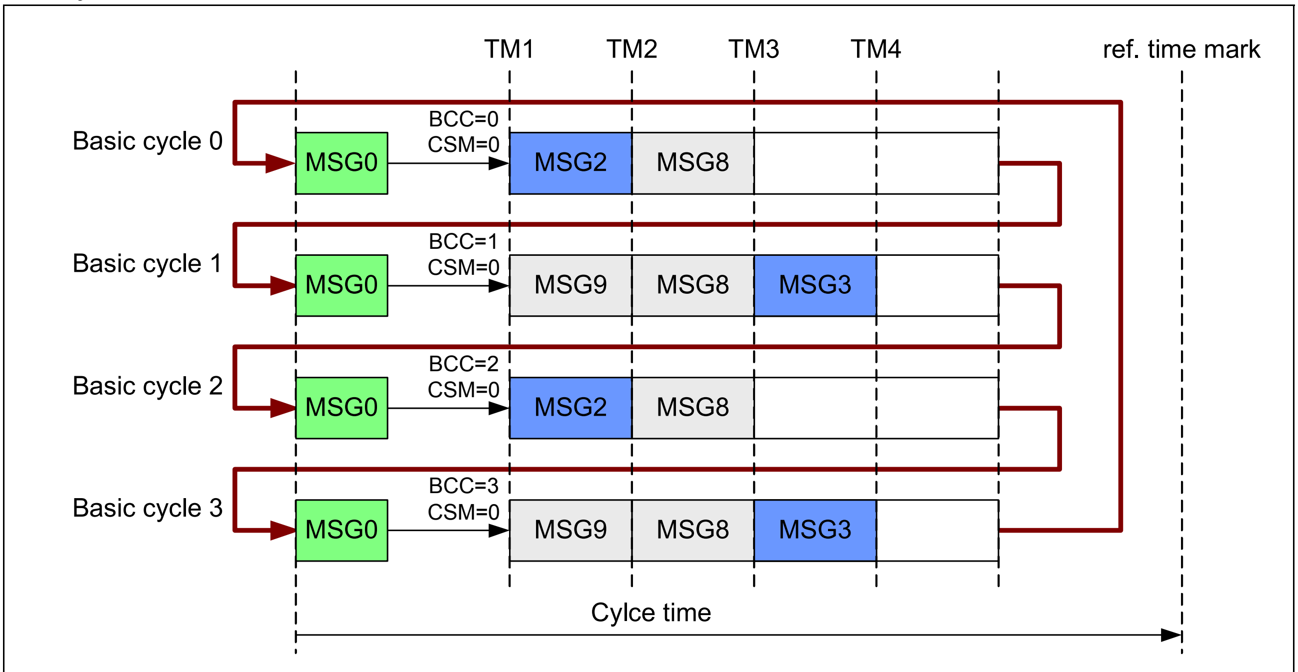
MSG number	Identifier	DLC	MSG	Interrupt	comments
MSG0 (Ref. MSG)	0x118	4	Rx		
MSG2	0x248	8	Tx	Tx_ok ISR (SRN0)	
MSG3	0x258	8	Tx	Tx_ok ISR (SRN0)	
MSG8	0x448	8	Tx	Tx_ok ISR (SRN0)	
MSG9	0x458	8	Tx	Tx_ok ISR (SRN0)	
MSG12	0x248		Rx	Rx_ok ISR (SRN1)	
MSG13	0x258		Rx	Rx_ok ISR (SRN1)	
MSG18	0x448		Rx	Rx_ok ISR (SRN1)	
MSG19	0x458		Rx	Rx_ok ISR (SRN1)	

- Message Transmit ok interrupt service routine:

Set the transmit request (TxRQ) in this message object control register (MSGCTR) for the next time trigger event

No arbitrating time window is defined in this application. Message transmission is only allowed in the execution time window.

- System matrix:



- Interrupt service routine for the service request nodes:
  - SRN0: message transmit ok interrupt for Tx MSG8, MSG9, MSG2 and MSG3
  - SRN1: message receive ok interrupt for Rx MSG18, MSG18, MSG12 and MSG13

- SRN2: TTCAN new matrix cycle
- SRN3: TTCAN Notification interrupt
- SRN4: TTCAN error interrupt
- SRN5: CAN node 0 error (BOFF, EWARN, LLE, LOE, INIT and LEC)

5.2 System Overview

Two TC1796 boards are used:

- CAN node 0 Tx and Rx pins should be connected
- external oscillator = 20 MHz
- DIP switch S301: internal memory access
  S301:1..8=on-off-on-on-on-on-off-off
- DIP switch S401.5 = OFF (use internal flash)
- DIP switch S402.1 = OFF for master board
  DIP switch S402.1 = ON for slave board

Note: S401 and S402 are on the back side

- The internal flash is used (0xA000,0000)
- RS323 connect to PC COM for program download

For program execution:

- S301:1..8=on-off-on-on-on-on-off-off

For program download:

- S301:1..8=on-on-on-on-on-on-off-off

For detailed information about the T1796 board, please refer to www.infineon.com

CANalyzer (optional) for CAN message monitoring:

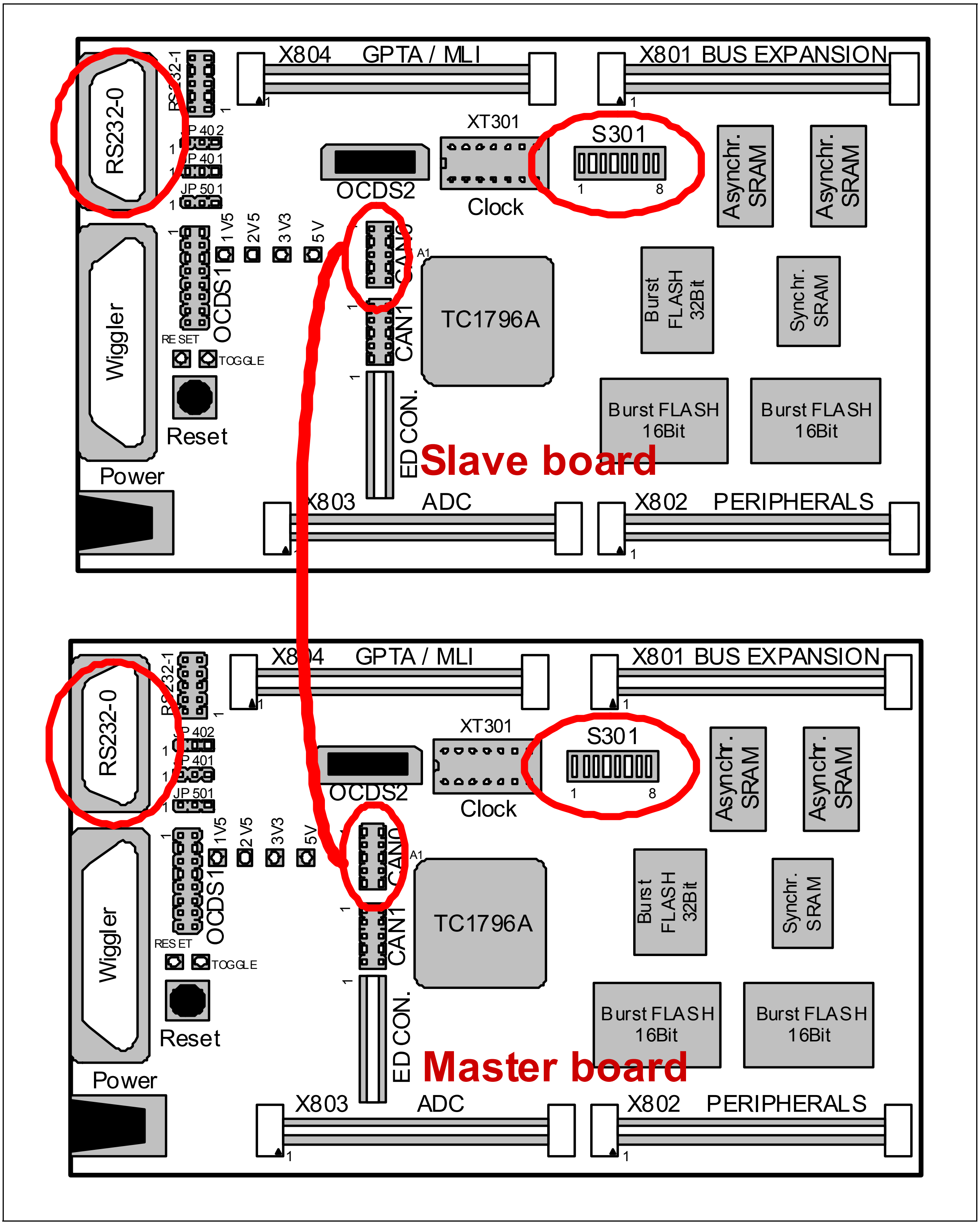


Figure 13 Master and Slave Board Connection

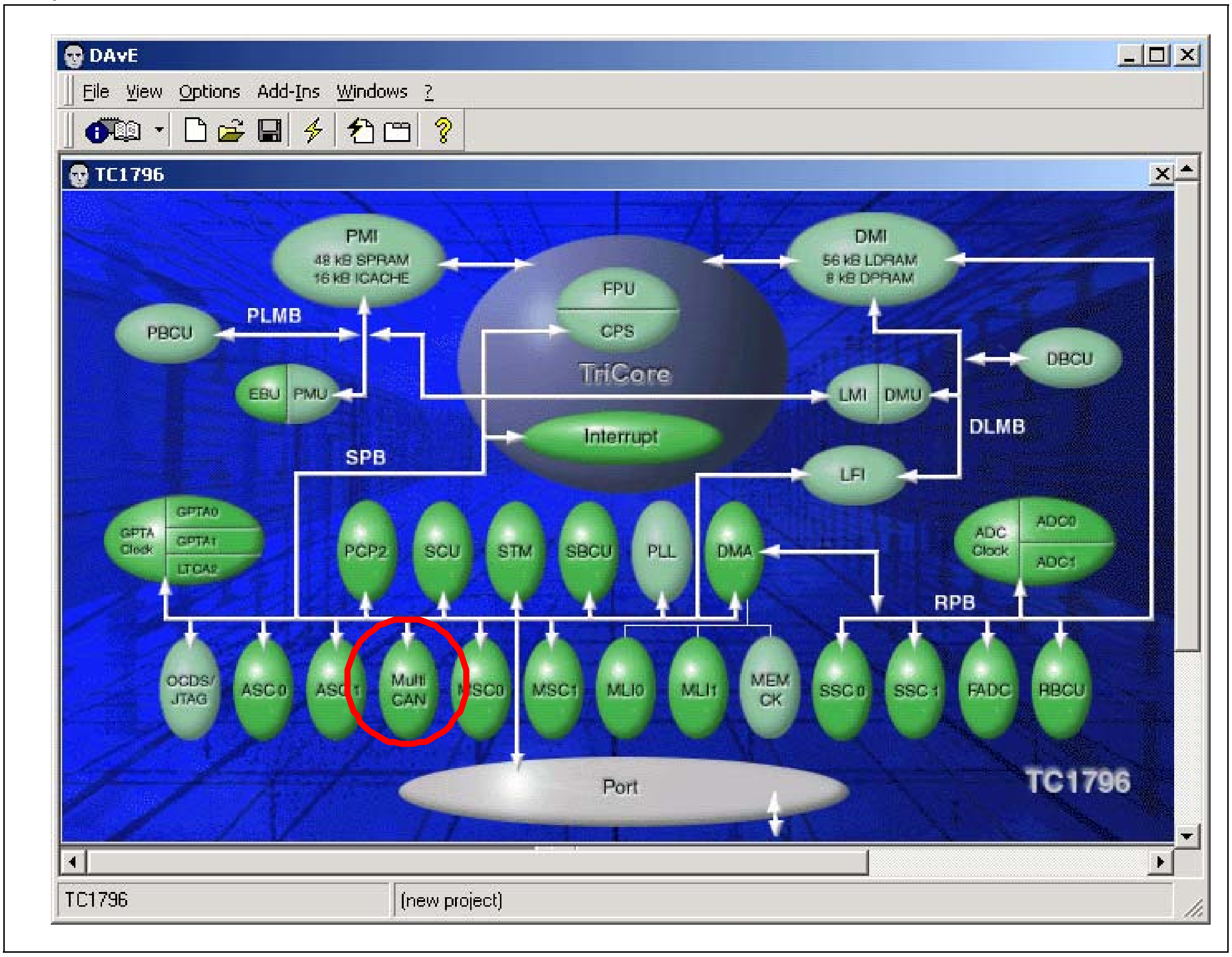
5.3 Using DAvE to Configure SW Project

DAvE Install: download and install DAvE, please refer to **www.infineon.com**

DAvE setup and code generation:

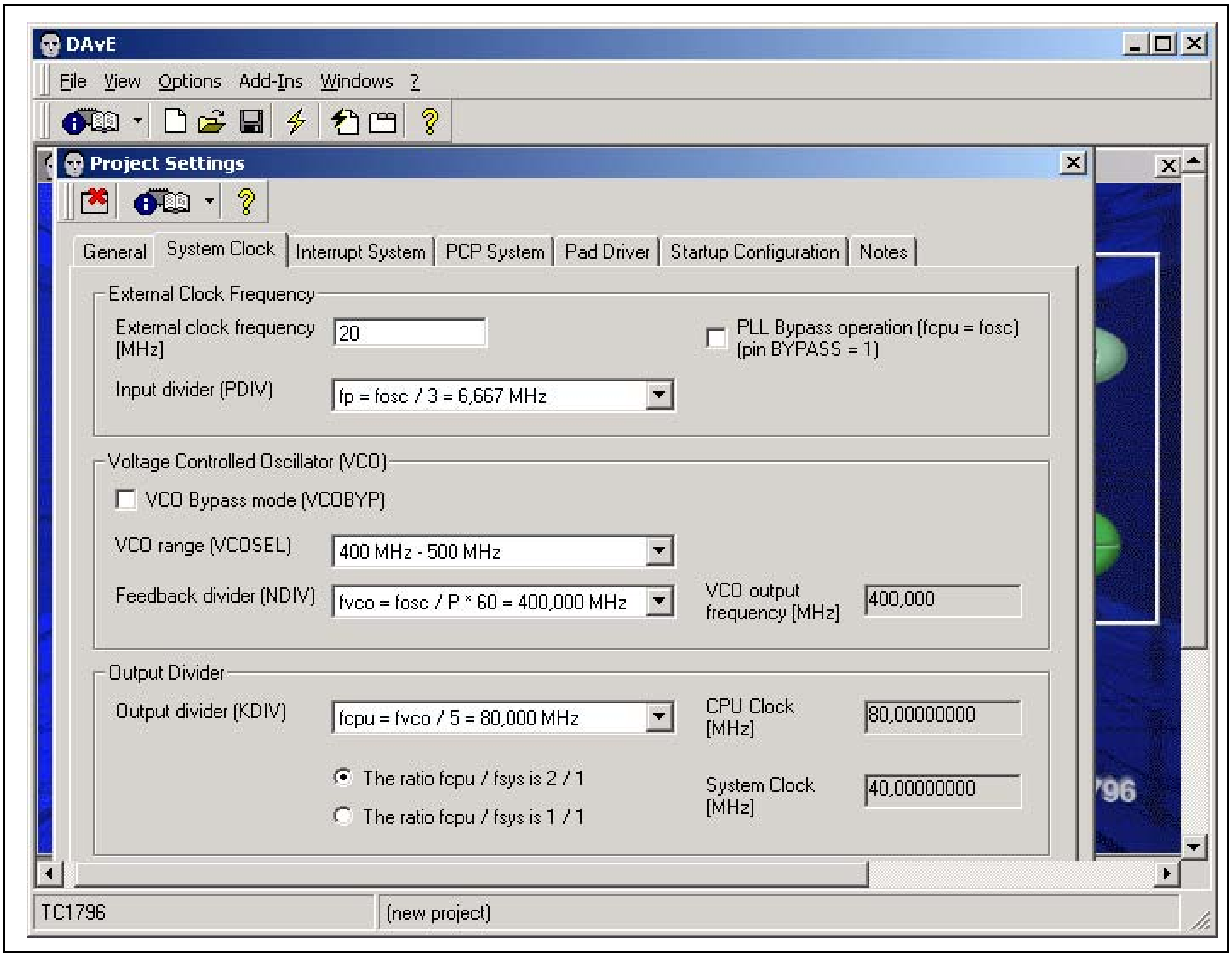
Step 1: create new project TTCAN_DAVE.dav

Step 2: select device TC1796



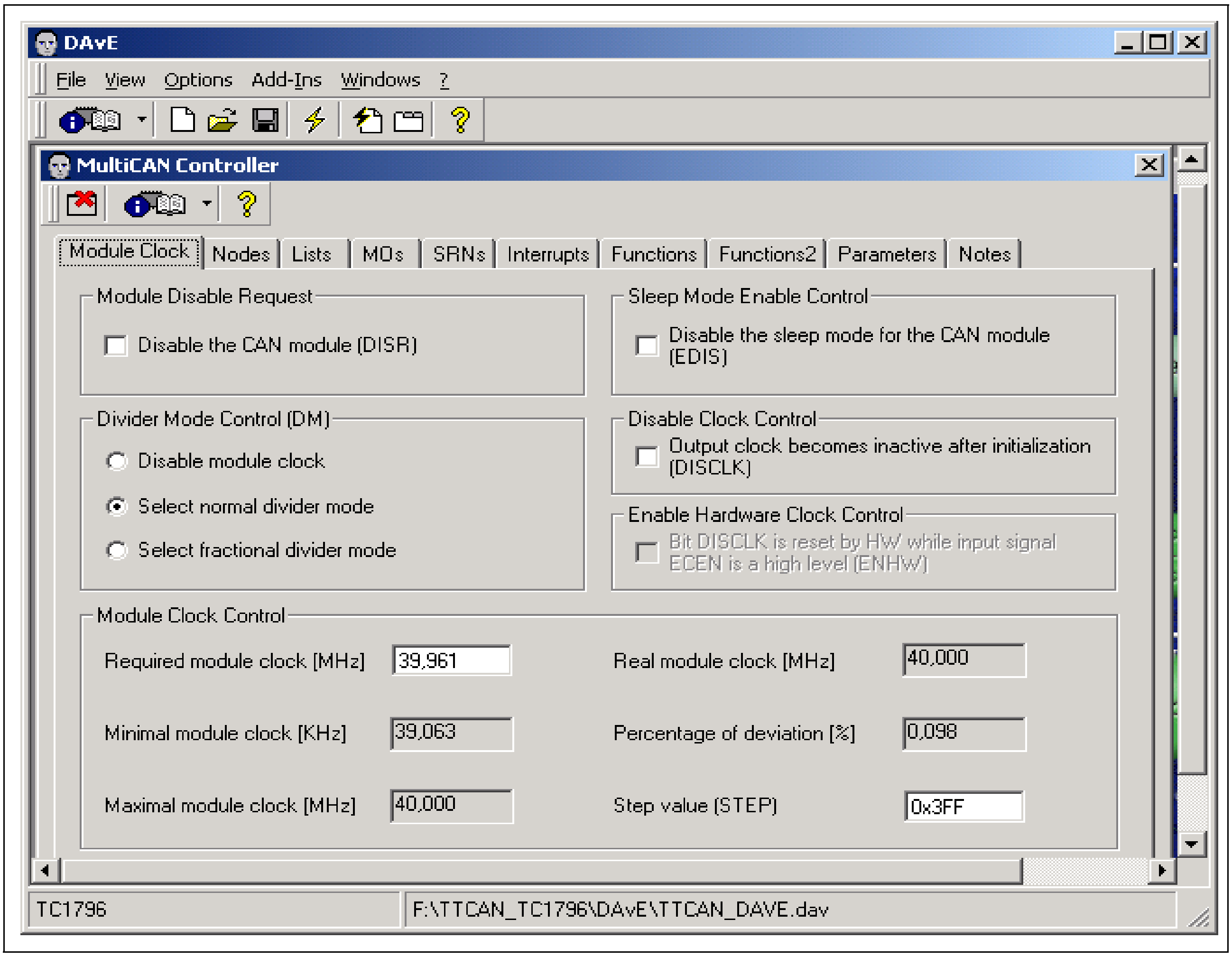
Step 3: general settings:

- PLL set up: fsys=40MHz, fosc=20MHz
- global interrupt enable

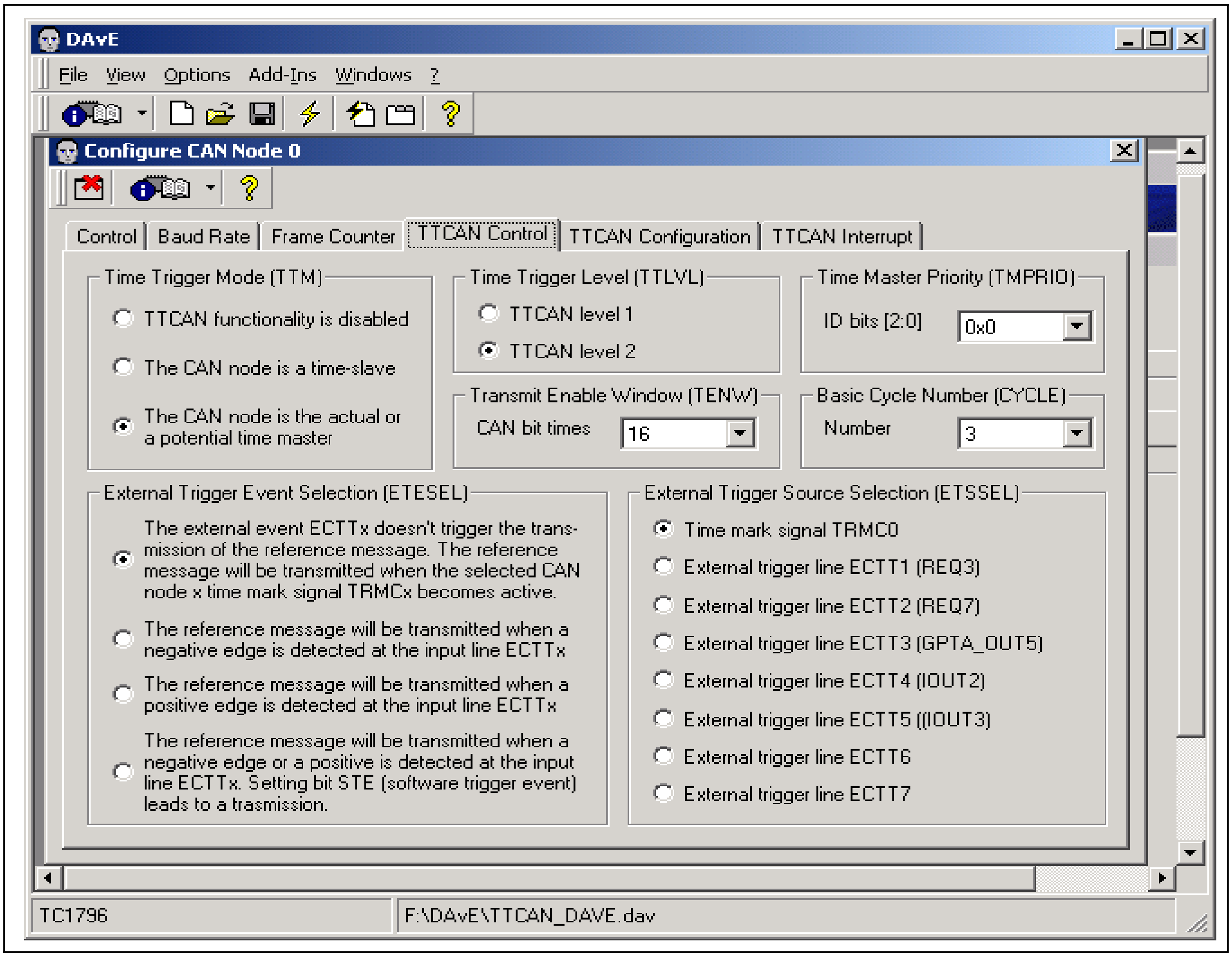


Step 4: MultiCAN configuration

- module clock: normal divider mode



- Nodes: only CAN node 0 initialization
  – Control: Pins setting: Rx pin=P6.8, Tx pin=P6.9, enable ALIE and LECIE using URN5
  – Baud Rate: 500KBaud, 1 bit timing = 20 Tq
  – TTCAN control: time master, TTLVL=2, TENW=16 CAN bit time(1..16),CYCLE=3(0..63)
  – TTCAN configuration: EXPTT=4(0..255), IRO=0(0..127), Adjust enable
  – TTCAN interrupt: NBC interrupt=SRN2, Notification interrupt=SRN3, error interrupt=SRN4

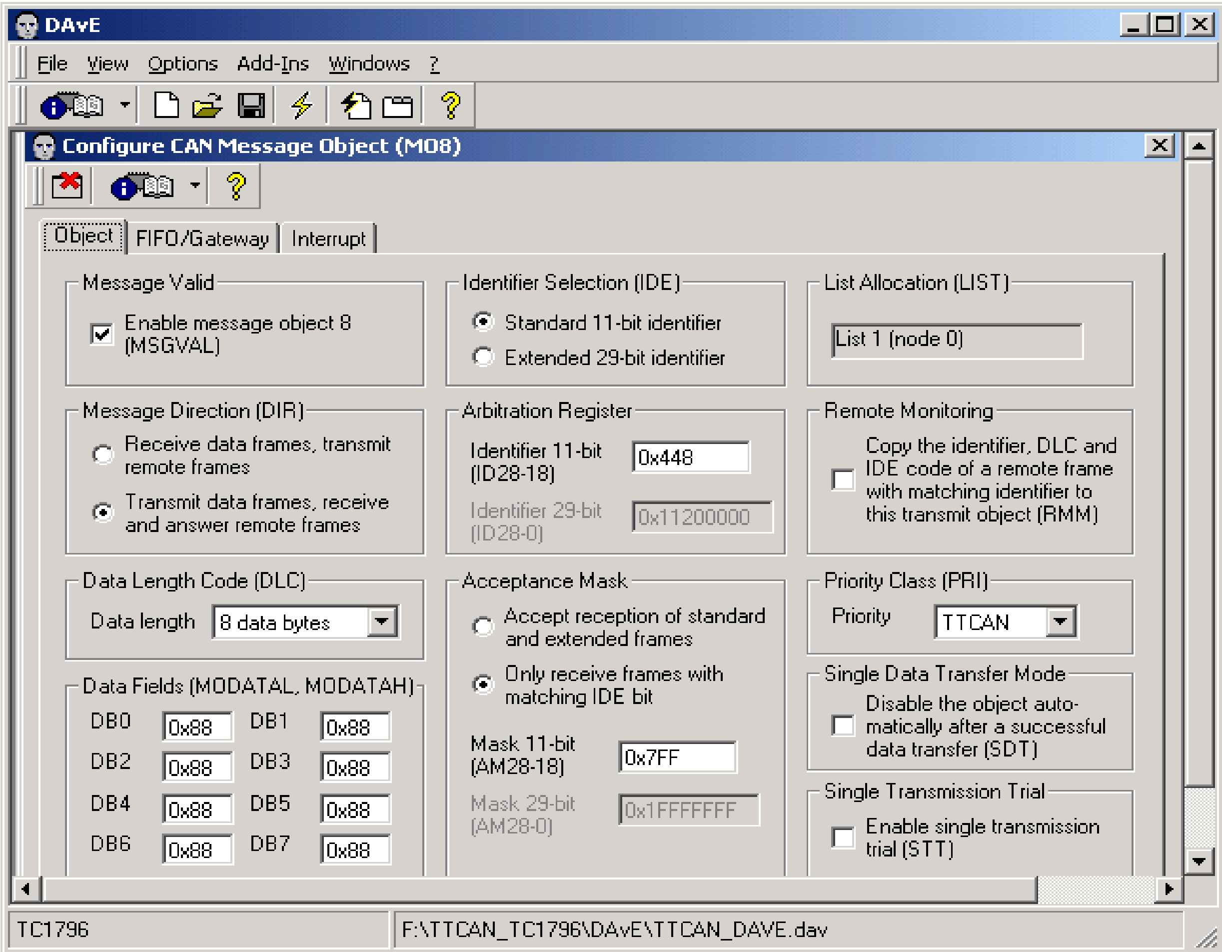


– CAN Lists: assign MO0, MO8,MO9, MO2,MO3,MO18,MO19,MO12,MO13 to CAN node 0

Note: reference message MO0 should be the first MSG an node 0

– CAN messages:
– configure the predefined messages according to table on page 33
– SRNs: enable the CAN service request nodes SRE0...SRE5
– Interrupts: interrupts level configuration
– Function: Select CAN_vInit(void) and CAN_vTransmit(void) function

Step 6: generating code and save the DAvE file



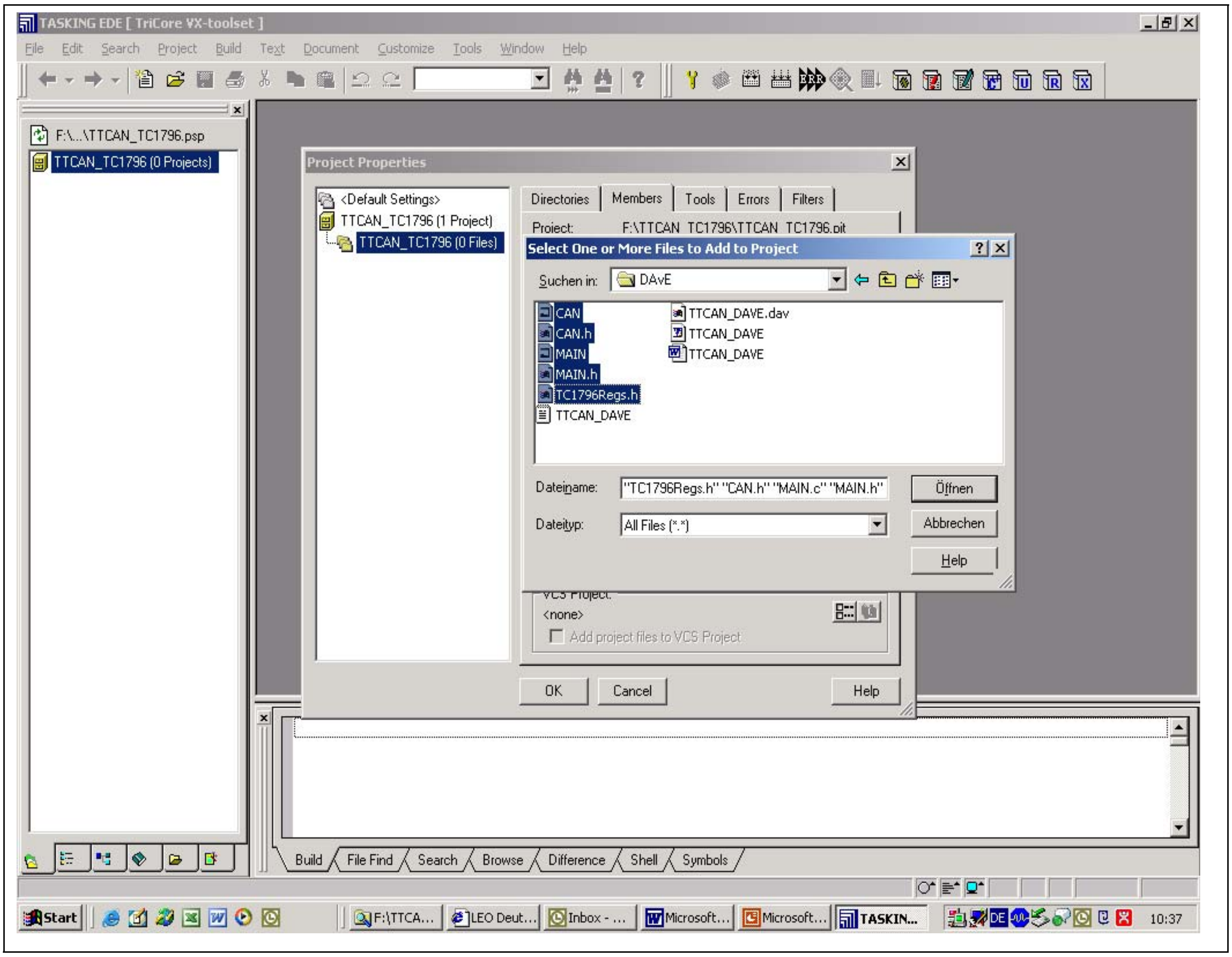
5.4 Using Tasking to Generate Code

Start Tasking Complier TriCoreV2.3r1

Step 1: open tool Tasking TriCore v2.0r1 EDE and create new project space (TTCAN_TC1796.psp)

Step 2: create new project (TTCAN_Tc1796.pjt)

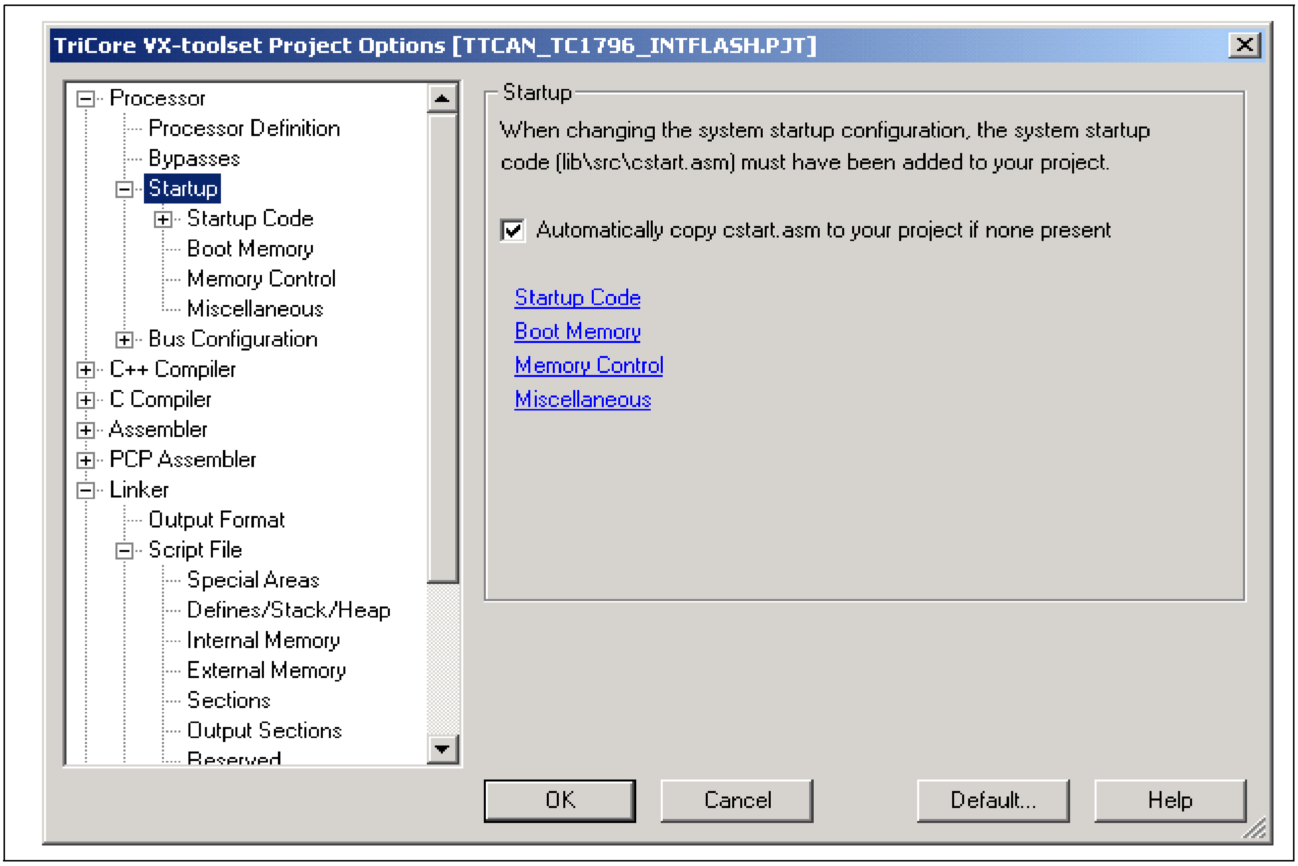
Step 3: Import DAvE file: CAN.c, CAN.h, MAIN.c, MAIN.h and TC1796Regs.h



Step 4: modify the project file property and option

- processor definition: TC1796
- deactivate the initialization of bus configuration in start up code
- enable DAvE file under menu project/option/C Compiler
- define reset start address(0xA0000000)/interrupt table(0xA0004000)
- define section the internal ROM ‚int_c' at 0xA0000000

Note: you can restore the option file (ttcan_tc1796_v2_3_IntFlash.opt) with menu project/load options directly



It is ready to compile and link successfully.

5.5 Code Modification

Step 1: modify DAvE files for master and slave board for TTCAN network communication

the code configured by DAvE file is for a master CAN operating. Pin0.8 is used for master and slave selection. If Pin0.8 is latched a low level after a HW reset, a master CAN mode is selected on the board. Otherwise, a high level after a HW reset on Pin0.8 selects a slave CAN mode on the board. Therefore, one SW code can be used for the master and the slave operation.

- Define unsigned int uwUser_Switch and insert the following instruction in main(void):

```
...
uwUser_Switch = (~((P0_IN) & 0xff00)) >> 8;//read dip-switch P0.8...P0.15
...
```

- In function CAN_vInit(void) modify registers TTCR, TTCFGR and TURR for master and slave mode (TTCR.TTM: Master or slave mode. TTCFG.EXPTT: = 4 for master mode; = 6 for slave mode).

```
...
 if ((uwUser_Switch & 0x01) == 0) //for Master TTCAN node: basic cycle number (0...3)
 {
   CAN_TTCR        =  0x4F030102;  // load  node 0 time trigger control register
```

```
  CAN_TTCFGR      =  0x00000400;  // load  node 0 time trigger configuration
  CAN_TURR        =  0x00000661;  // load  node 0 time unit ratio register
}
else
{
  CAN_TTCR        =  0x4F030101;  // load  node 0 time trigger control register
  CAN_TTCFGR      =  0x00000600;  // load  node 0 time trigger configuration
  CAN_TURR        =  0x00000661;  // load  node 0 time unit ratio register
}
...
```

Step 2: Configure the scheduler memory

- Add schedule/time entry (for master and slave TTCAN node) in CAN_vConfigureSchedulerEntries(void) according to table on page 32

Step 3: Modify the interrupt service routine

- Include while (1), for error or debug using
- In message transmit ok interrupt service interrupt routine set TxRQ for the next time trigger event and modify data bytes in the message objects

Step 4: Link and locate code in the internal flash at address 0xA0000000

5.6 Using Memtool for Downloading

Download code with Memtool:

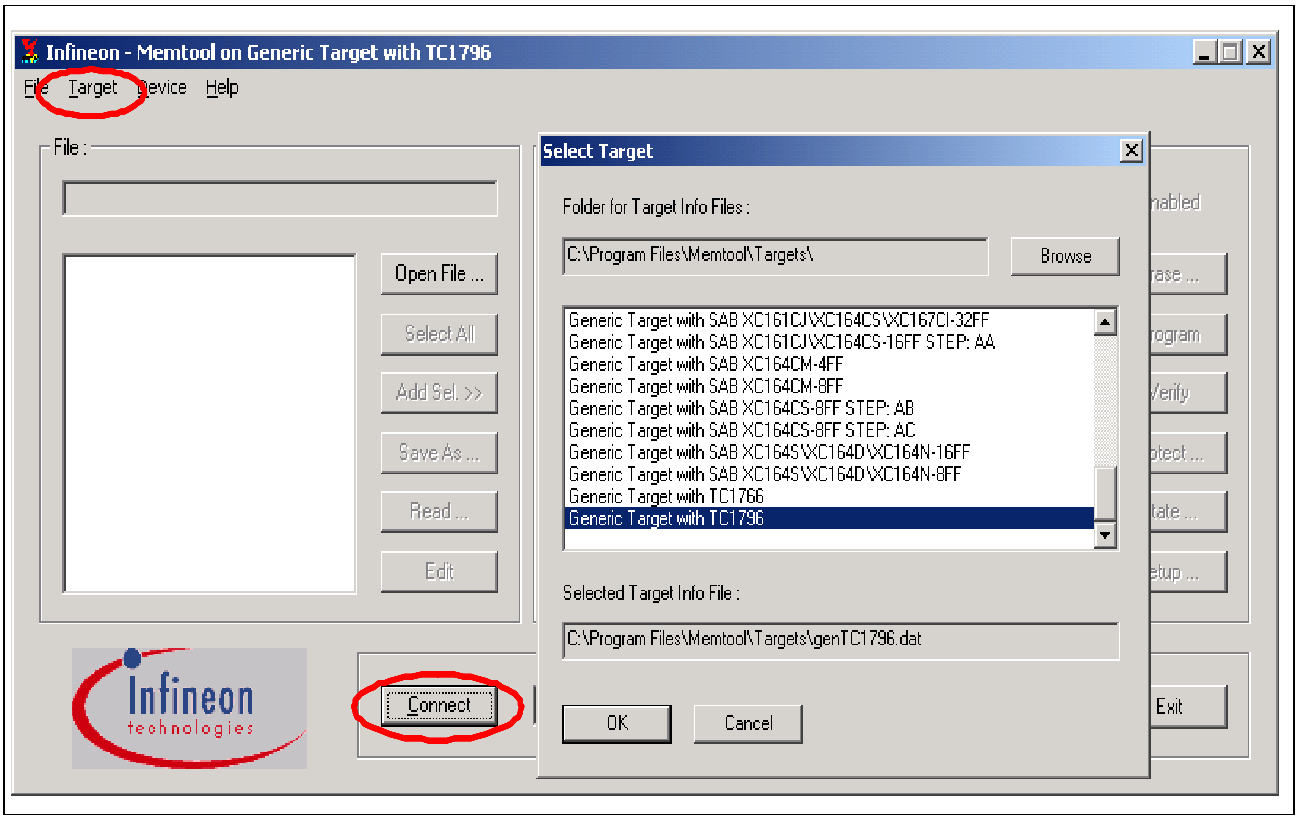
Step 1: download and install the freeware Memtool, please refer to **www.infineon.com**

Step 2: Hardware connection

- connect the RS232(ASC interface) to your PC
- set ASC boot mode on the Board. DIP switch S301:1..8=on-**on**-on-on-on-on-off-off

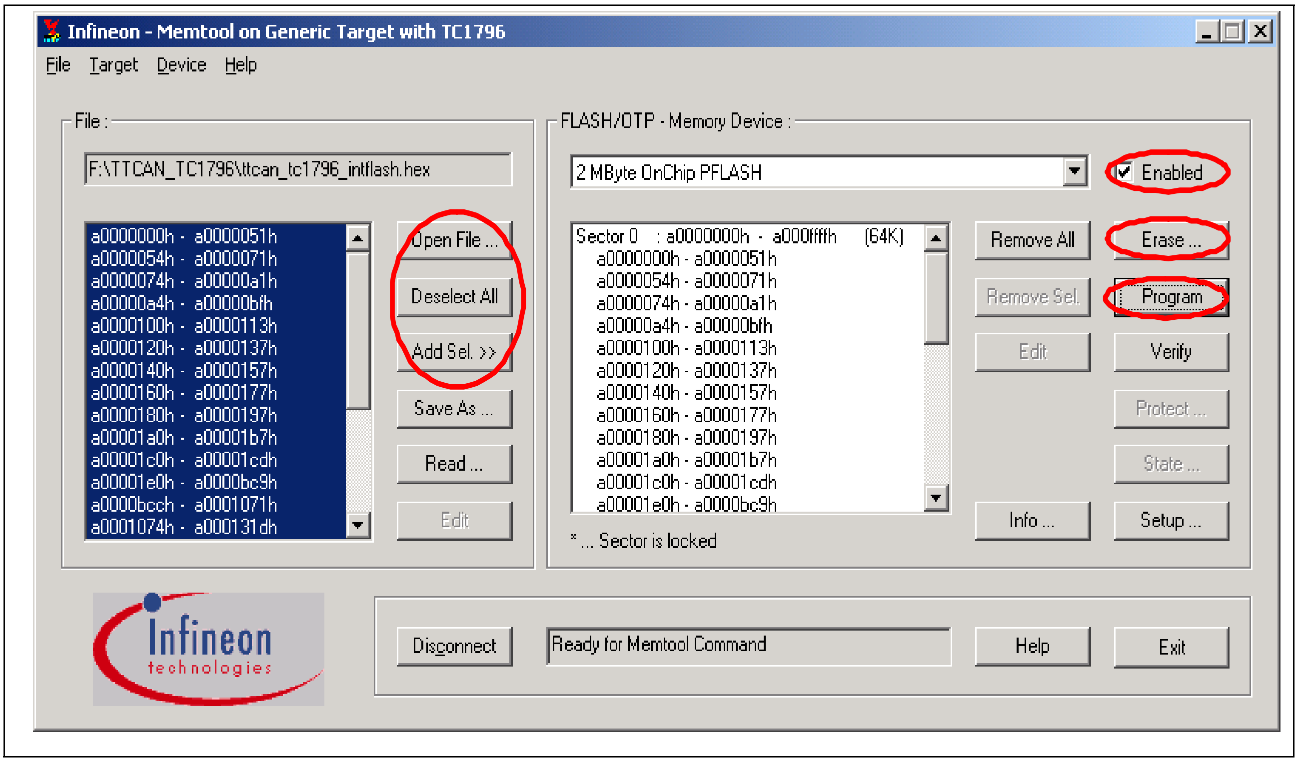
Step 3: start Memtool and download the code to the device

- select target TC1796 and connect



Step 4: start Memtool and download the code to the device

- enable 2Mbytes internal flash
- open the hex file
- select and add selection
- erase sections and program

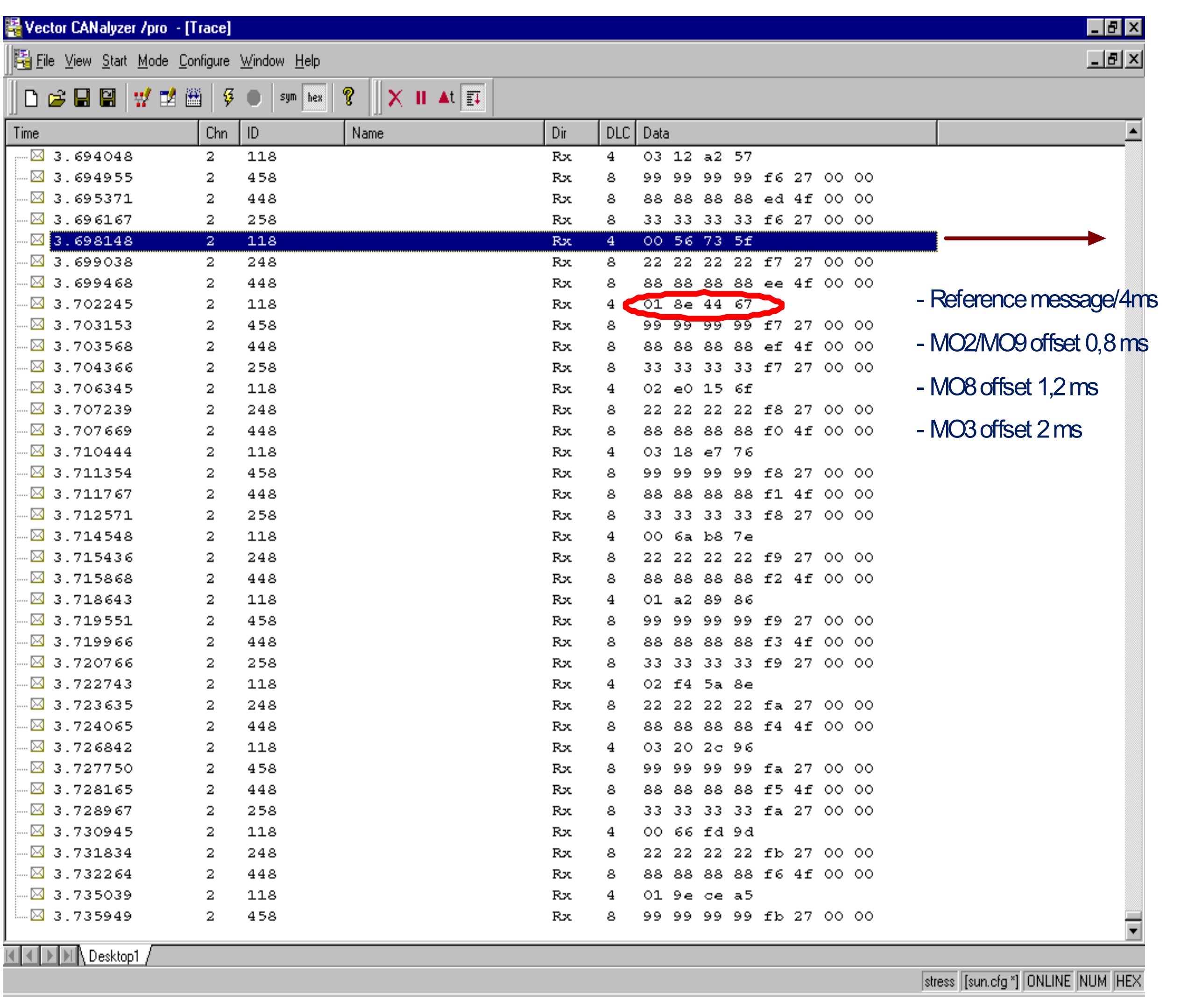


5.7 Code Execution from Internal Flash after HW Reset

- set DIP switch S301 (**master** and **slave** board) for internal memory access S301:1..8=on-off-on-on-on-on-off-off
- set DIP switch S402.1 = OFF for **master board**
- set DIP switch S402.1 = ON for **slave board**

6 Appendix (Trace Window of CANalyzer)

Trace window of CANalyzer: message information on CAN bus



Time	Chn	ID	Name	Dir	DLC	Data
3.694048	2	118		Rx	4	03 12 a2 57
3.694955	2	458		Rx	8	99 99 99 99 f6 27 00 00
3.695371	2	448		Rx	8	88 88 88 88 ed 4f 00 00
3.696167	2	258		Rx	8	33 33 33 33 f6 27 00 00
3.698148	2	118		Rx	4	00 56 73 5f
3.699038	2	248		Rx	8	22 22 22 22 f7 27 00 00
3.699468	2	448		Rx	8	88 88 88 88 ee 4f 00 00
3.702245	2	118		Rx	4	01 8e 44 67
3.703153	2	458		Rx	8	99 99 99 99 f7 27 00 00
3.703568	2	448		Rx	8	88 88 88 88 ef 4f 00 00
3.704366	2	258		Rx	8	33 33 33 33 f7 27 00 00
3.706345	2	118		Rx	4	02 e0 15 6f
3.707239	2	248		Rx	8	22 22 22 22 f8 27 00 00
3.707669	2	448		Rx	8	88 88 88 88 f0 4f 00 00
3.710444	2	118		Rx	4	03 18 e7 76
3.711354	2	458		Rx	8	99 99 99 99 f8 27 00 00
3.711767	2	448		Rx	8	88 88 88 88 f1 4f 00 00
3.712571	2	258		Rx	8	33 33 33 33 f8 27 00 00
3.714548	2	118		Rx	4	00 6a b8 7e
3.715436	2	248		Rx	8	22 22 22 22 f9 27 00 00
3.715868	2	448		Rx	8	88 88 88 88 f2 4f 00 00
3.718643	2	118		Rx	4	01 a2 89 86
3.719551	2	458		Rx	8	99 99 99 99 f9 27 00 00
3.719966	2	448		Rx	8	88 88 88 88 f3 4f 00 00
3.720766	2	258		Rx	8	33 33 33 33 f9 27 00 00
3.722743	2	118		Rx	4	02 f4 5a 8e
3.723635	2	248		Rx	8	22 22 22 22 fa 27 00 00
3.724065	2	448		Rx	8	88 88 88 88 f4 4f 00 00
3.726842	2	118		Rx	4	03 20 2c 96
3.727750	2	458		Rx	8	99 99 99 99 fa 27 00 00
3.728165	2	448		Rx	8	88 88 88 88 f5 4f 00 00
3.728967	2	258		Rx	8	33 33 33 33 fa 27 00 00
3.730945	2	118		Rx	4	00 66 fd 9d
3.731834	2	248		Rx	8	22 22 22 22 fb 27 00 00
3.732264	2	448		Rx	8	88 88 88 88 f6 4f 00 00
3.735039	2	118		Rx	4	01 9e ce a5
3.735949	2	458		Rx	8	99 99 99 99 fb 27 00 00

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG