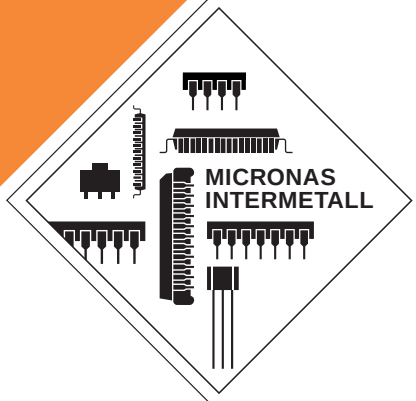

CAN Manual

Edition Aug. 7, 1996
6251-406-1DS

MICRONAS
INTERMETALL

Contents

Contents, continued

CAN Manual

This manual describes the user interface of the CAN module. For further information about the CAN bus, please refer to the CAN specification 2.0B from Bosch.

1. Features

– Bus controller according to CAN Licence Specification 1992

– Supports standard and extended telegrams

– FullCAN: at least 16 Rx and Tx telegrams

– Variable number of receive buffers

– Programmable acceptance filter
Single, group or all telegrams received.

– Time stamp for each telegram

– Overwrite mode programmable for each telegram

– Programmable baud rate. Max. 1 MBd @ 8 MHz

– Sleep mode

– CAN clock frequency 8 MHz, CPU 8 MHz

– Temperature –40...+85°C

– Totally static design

The CAN interface is a VLSI module which enables coupling to a serial bus in compliance with CAN specification 2.0B. It controls the receiving and sending of telegrams, searches for Tx telegrams and interrupts and carries out acceptance filtering. It supports the transmission of telegrams with standard (11 bit) and extended (29 bit) addresses.

The CAN interface can be configured as BasicCAN or FullCAN. It enables several active receive and transmit telegrams and supports the remote transmission request. The number of telegrams which can be handled depends mainly on the size of the communication RAM (16 byte per Tg), the system clock and the transmission speed. A maximum of 254 telegrams can be handled.

A mask register makes it possible to receive different groups of telegram addresses with different receive telegrams. The transmitting or receiving of a telegram as well as the occurrence of an error can trigger an interrupt.

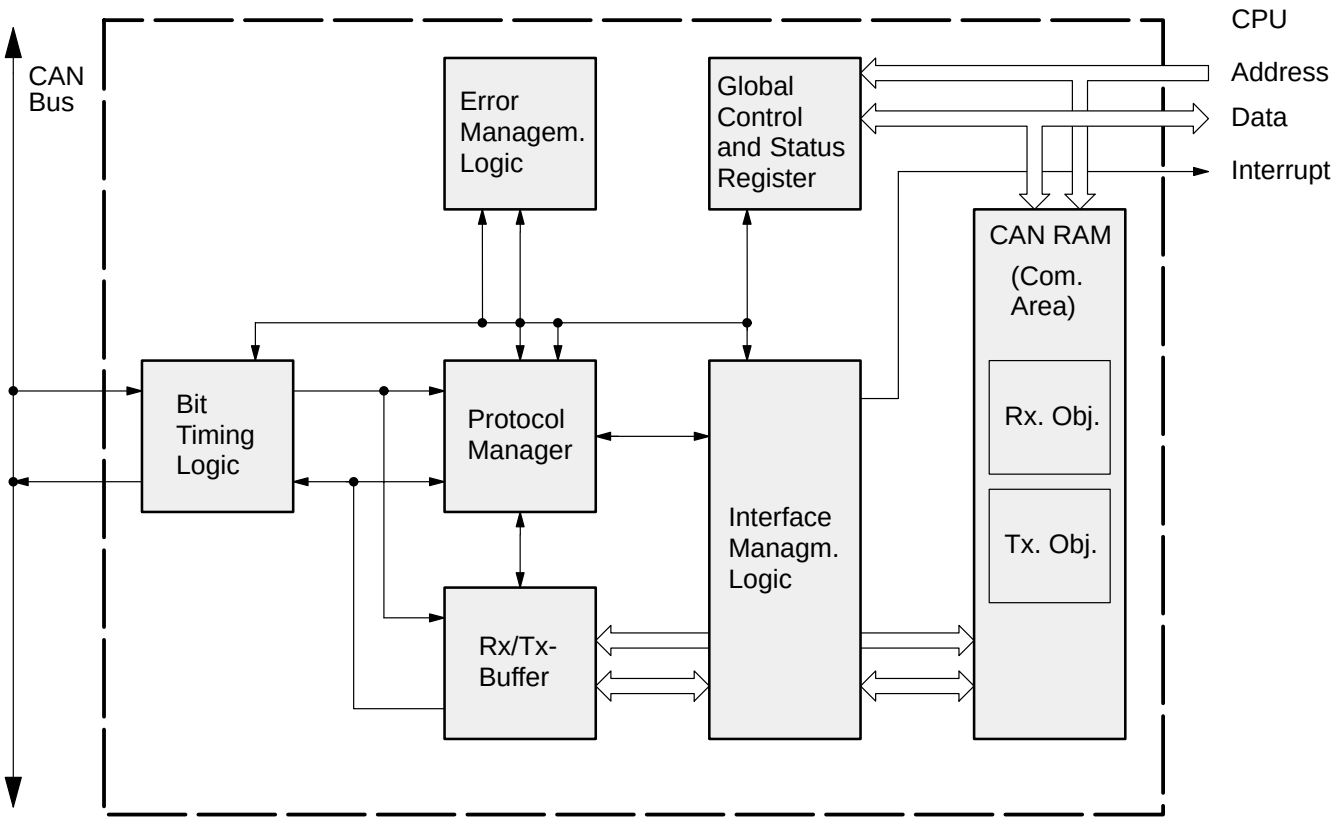

Fig. 1–1: Block diagram of the CAN bus interface

2. Abbreviations

BI	CAN Bus Interface
BTL	Bit Timing Logic
CAN	Controller Area Network
CA	Communication Area
CO	Communication Object
CM	Communication Mode
CRC	Cyclic Redundancy Code
DLC	Data Length Code
EoCA	End of CA
Ext. ID	Extended Identifier
Ext. Tg	Extended Telegram
GCS	Global Control and Status Register
ID	Identifier
IML	Interface Management Logic
Rx. Obj.	Receive Object
RxTg	Receive Telegram
Std. ID	Standard Identifier
Std. Tg	Standard Telegram
TD	Telegram Descriptor
TQ	Time Quantum
Tx. Obj.	Transmit Object
TxTg	Transmit Telegram

3. Functional Description

3.1. HW Description

The CAN bus interface (BI short for bus interface) consists of the following components:

Bit Timing Logic: Scans the bus and synchronizes the CAN bus controller to the bus signal.

Protocol Manager: The PM monitors or generates the composition of a telegram and performs the arbitration, the CRC and the bit stuffing. It controls the data flow between Rx/Tx buffers and CAN bus. It also drives the Error Management Logic.

Error Management Logic: Adds up the error messages received from the protocol manager and generates error messages when particular values are exceeded. Guarantees the error limitation as per the CAN Spec. V2.0B.

Interface Management Logic: The IML scans the Communication Area (CA) in the CAN-RAM for transmit telegrams. As soon as it finds one, it enters it into the Rx/Tx buffer and reports it to the protocol manager as ready for transmission. If a telegram is received, the IML carries out the acceptance filtering, i.e. scans the CA, taking into account the Identifier Mask Register in the GCS, for a Tg with the appropriate address. After correct reception, it copies the Tg from the Rx/Tx buffer to the CA. The IML also reports to the CPU the valid transfer of a telegram or given errors per interrupt.

Rx/Tx buffer: This is used to buffer a full telegram (ID, DLC, data) during sending and receiving.

Global Control and Status Register: The GCS contains registers for the configuration of the BI. It also contains error and status flags and an identifier mask. The Error Counter and the Caption Timer can also be read from the GCS.

Receive Object: Telegrams received can be entered into an Rx-Object and can be retrieved from the application.

Transmit Object: The application enters data into the Tx-Object and reports it ready for transmission. The BI sends the telegram as soon as the bus traffic allows.

3.2. Memory Map

From the CAN bus interface the user sees two storage areas in the user RAM area. The BI is configured with the Global Control and Status Registers (GCS). It also indicates the status here. The communication area (CA) contains the Rx and Tx telegrams.

The communication area lies in the CAN-RAM. The end of the Com. Area is fixed by the first check byte of an object whose 3 MSBs contain only ones (Communication Mode = 7 = EoCA). The area after this is available to the user.

The CA comprises communication objects (COs). A CO consists of 6 bytes telegram descriptor (TD), 8 data bytes and the Time Stamp which is 2 bytes long. The TD contains the address (ID) and the length of a telegram (DLC) as well as control bits which are needed for access to the CO and for the transmission of a telegram.

In the BasicCAN and the FullCAN versions, all the communication objects have the same, maximum size of 16 byte. Unassigned storage locations in the data area of a CO can be freely used.

The maximum number of COs is limited by the time which the CAN interface has to search for an identifier in the Com. Area.

3.3. Global Control and Status Registers (GCS)

The GCS registers can be used to determine the behavior of the CAN interface. As well as flags for the interrupts, halt and sleep modes, they also contain interrupt index, ID mask, bus timing, error status, output control registers, baud rate prescalers, Tx and Rx error counters as well as the capture timer.

Access modes:

r: read
w: write
i: init (BI halted)
w0: clear
w1: set

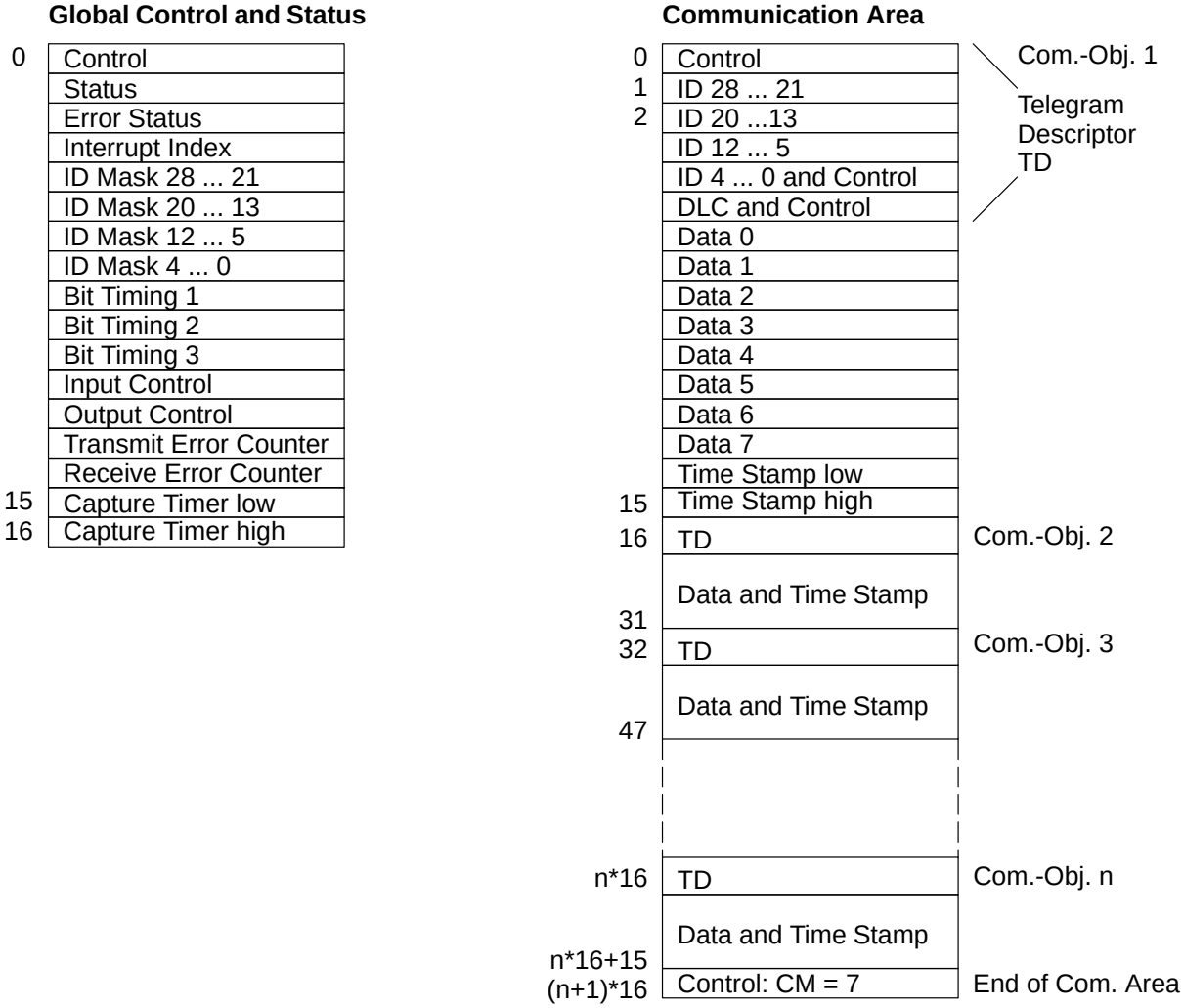

Fig. 3–1: Memory allocation

3.3.1. CTR Control Register:

7	6	5	4	3	2	1	0
HLT	**SLP**	**GRSC**	**EIE**	**GRIE**	**GTIE**	**rsvd**	**rsvd**

HLT **Halt**:
CPU: r,w BI: r

Puts the CAN interface in the halt mode. Transmissions which have been started are brought to an end. The halt acknowledge is indicated in the status register (HACK). Re-initialization can be carried out in the halt mode (set to HACK). After this, the halt flag must be deleted again. After a reset, HLT is set.

If HLT is set during a Tx-Tg and this has to be repeated (error or no acknowledge), the BI still stops. The corresponding TxCO is still reserved, however, and can no longer be operated from BI. Therefore, when HLT is set, the CA should always be re-initialized if the last Tx-Tg has not been correctly transmitted (Status Transfer Flag is still deleted).

SLP **Sleep:**
CPU: r,w BI: r,w0

The BI goes into the sleep mode when the sleep flag is set and a started Tg is terminated. The sleep mode is exited as soon as a dominant bus level is detected, or the sleep flag is deleted.

GRSC **Global Rescan:**
CPU: r,w1 BI: r,w

The microprocessor can set this flag in order to initiate a transmit telegram search at the beginning of the Com. Area. The BI resets the bit. The BI also sets the GRSC flag if the flag RSC has been set in a telegram descriptor of a Tx-Tg just operated, and thereby initiates a rescan. If the microprocessor writes a zero, nothing happens.

EIE **Error-Interrupt-Enable:**
CPU: r,w BI: r

Masks the error interrupt.

GRIE **Global Rx-Interrupt-Enable:**
CPU: r,w BI: r

Masks the receive interrupt.

GTIE **Global Tx-Interrupt-Enable:**
CPU: r,w BI: r

Masks the transmit interrupt.

3.3.2. STR Status Register:

7	6	5	4	3	2	1	0
HACK	**BOFF**	**EPAS**	**ERS**	**rsvd**	**rsvd**	**rsvd**	**rsvd**

HACK **Halt-Acknowledge:**
CPU: r BI: r,w

Is set by the BI when it enters the halt mode. It is deleted again when the halt mode is exited.

BOFF **Bus-Off:**
CPU: r BI: r,w

With this flag the BI indicates whether the node is still actively participating in the bus. If the transmit error counter reaches a value of > 255 (overflow), the node is separated from the bus and the flag is set.

EPAS **Error-Passive:**
CPU: r BI: r,w

With this flag the BI indicates whether the node is still participating in the bus with active Error Frames. If an error counter has reached a value > 127, the node still transmits passive error frames and the flag is set.

ERS **Error-Status:**
CPU: r BI: r,w

This flag is set when the BI detects an error. It is set even if an error counter is greater than 96. It means that a bit has been set in the error status register. As soon as all the flags in the error status register are deleted, ERS is also deleted.

3.3.3. ESTR Error Status Register:

As long as a bit is set in the ESTR, the ERS bit is also set in the status register. If EIE has been set in the control register, an interrupt is also triggered; i.e. the value 254 is entered in the register IDX as soon as it is free, and a one is transmitted on the interrupt line.

To erase a bit in the ESTR the user must write a one at the appropriate place. Places at which he writes a zero will not be changed. Because it makes sense to erase only those bits which have previously been read, only the byte which has been read has to be re-written.

7	6	5	4	3	2	1	0
GDM	**CTOV**	**ECNT**	**BIT**	**STF**	**CRC**	**FRM**	**ACK**

GDM **Good Morning:**
CPU: r,w BI: w1

Is set by the BI when it is aroused from the sleep mode by a dominant bus level. The user must delete it.

CTOV **Capture Time Overflow:**
CPU: r,w BI: w1

Is set by the BI when the capture timer (CTIM) overflows. The user must delete it.

ECNT **Error Counter Level:**
CPU: r,w BI: w1

Is set by the BI as soon as the transmit error counter or the receive error counter exceeds a limit value. The user must delete it.

BIT **Bit Error:**
CPU: r,w BI: w1

Is set by the BI when a transmitted bit is not the same as the bit received. The user must delete the flag.

STF **Stuff Error:**
CPU: r,w BI: w1

Is set by the BI when 6 identical bits are received successively in one Tg. The user must delete it.

CRC **CRC Error:**
CPU: r,w BI: w1

Is set by the BI when the CRC received does not coincide with the CRC calculated. The user must delete it.

FRM **Form Error:**
CPU: r,w BI: w1

Is set by the BI when an incorrect bit is received in a field with specified bit location (start of frame, end of frame, ...). The user must delete it.

ACK **Acknowledge Error:**
CPU: r,w BI: w1

Is set by the BI when there is no acknowledge for a transmitted Tg. The user must delete it.

3.3.4. Interrupt Index Register:

IDX
CPU: r,w BI: r,w

The interrupt index indicates the source of the interrupt. If a transmission has been the cause of an interrupt, the interrupt index points to the corresponding telegram descriptor (IDX = 0..253). If an error has been responsible for the interrupt, the interrupt index designates the error status register (IDX = 254). After dealing with the interrupt, the user must eliminate the cause of the interrupt and set the interrupt index to minus one (255 = EMPTY). As soon as the IDX is empty, the BI can enter a new index and initiate an interrupt. An interrupt can only be initiated when IDX contains the value 255.

3.3.5. Identifier Mask Register:

IDM
CPU: r,i BI: r

The identifier mask register is 29 bits long; the MSB is in the MSB position in the lowest byte address. The IDM defines a mask for the acceptance of address groups. Only the permitted bits are used for comparison with a received identifier. Whether the mask is used can be determined individually for each receive object.

0: Don't care.
1: Compare.

3.3.6. Bit-Timing Register 1:

7	6	5	4	3	2	1	0
MSAM	**SYN**	**BPR**	**BPR**	**BPR**	**BPR**	**BPR**	**BPR**

MSAM **Multi Sample:**
CPU: r,i BI: r

0: The bus level is determined only once per bit.
1: The bus level is determined three times per bit.

SYN **Sync On:**
CPU: r,i BI: r

SYN determines the bit synchronization mode.
0: Synchronization is carried out only with negative edges.
1: Synchronization is also carried out with rising edges.

BPR **Baud Rate Prescaler:**
CPU: r,i BI: r

The baud rate prescaler sets the length of a time quantum for the bit timing logic.

$t_Q = t_{Clk} \times (BPR + 1)$.

With the 6-bit counter it is possible to extend t_Q by a factor of 1...64. Values from 0 to 63 are allowed.

0: No extension
1: $t_Q = t_{Clk} \times 2$
2: $t_Q = t_{Clk} \times 3$
3: $t_Q = t_{Clk} \times 4$
etc.

3.3.7. Bit Timing Register 2:

7	6	5	4	3	2	1	0
rsvd	**TSEG2**	**TSEG2**	**TSEG2**	**TSEG1**	**TSEG1**	**TSEG1**	**TSEG1**

TSEG2 **Time Segment 2:**
CPU: r,i BI: r

TSEG2 determines the number of time quanta after the sample point. Permitted entries: 1...7 ($ 2..8 TQ).

TSEG1 **Time Segment 1:**
CPU: r,i BI: r

TSEG1 determines the number of time quanta before the sample point. Permitted entries: 2..15 ($ 3..16 TQ).

3.3.8. Bit Timing Register 3:

7	6	5	4	3	2	1	0
rsvd	**rsvd**	**rsvd**	**rsvd**	**rsvd**	**SJW**	**SJW**	**SJW**

SJW **Synchronization Jump Width:**
CPU: r,i BI: r

SJW defines by how many TQs a bit may be lengthened or shortened because of resynchronization. Permitted entries: 1..4 ($1..4 TQ). Values greater than 4 must not be used.

3.3.9. Input Control Register:

7	6	5	4	3	2	1	0
rsvd	**rsvd**	**rsvd**	**rsvd**	**rsvd**	**XREF**	**REF1**	**REF0**

XREF **External Reference:**
CPU: r,i BI: r

0: The internal reference is used.
1: The external reference is used where available.

REF1 **Use Reference for RxD1:**
CPU: r,i BI: r

0: RxD1 is used as inverted input signal.
1: The supply voltage is used as inverted input signal.

REF0 **Use Reference for RxD0:**
CPU: r,i BI: r

0: RxD0 is used as input signal.
1: The supply voltage is used as input signal.

3.3.10. Output Control Register:

Reserved for control of transmit transistors.

7	6	5	4	3	2	1	0
rsvd	**rsvd**	**rsvd**	**rsvd**	**rsvd**	**rsvd**	**rsvd**	**rsvd**

3.3.11. Transmit Error Counter:

TEC **Tx-Error Counter:**
CPU: r BI: r,w

8-bit counter. Contains the number of transmit errors.

3.3.12. Receive Error Counter:

REC **Rx-Error Counter:**
CPU: r BI: r,w

7-bit counter. Contains the number of receive errors.

3.3.13. Capture Timer:

CTIM **Capture Timer:**
CPU: r BI: r,w

16-bit counter. Is incremented with a clock pulse derived from the CAN bus. Because it can only be read byte-wise, the low byte (addr. 15) must be read first. The corresponding high byte (addr. 16) is latched isochronously therefore. When the CTIM overflows, the flag CTOV in the error status register is set.

3.4. Communication Area (CA)

The CA is in the CAN-RAM. It consists of com. objects each of which is 16 bytes long. The CA begins at address 0 of the CAN-RAM with the first byte of a CO. It ends with the first byte of a C0 which consists once in its 3 MSBs (communication mode = 7 = EoCA). The following bytes can be used by the application. If the CAN-RAM contains 16 COs, there is no place and no need to mark the end of CA.

Every telegram which this node is to receive or transmit is represented by a CO. As well as the data and the time stamp, this also contains a header, the telegram descriptor (TD), in which the attributes of the communication object are stored.

The COs are entered in order of priority into the CA. This starts with the highest priority (the lowest identifier). The identifier defines the priority of a Tg. If the first eleven bits of an ext. Tg are the same as the identifier of a std. Tg, the Tg with standard identifier has the higher priority.

3.4.1. Telegram Descriptor (TD)

The telegram descriptor is 6 bytes long and forms the beginning of a CO. Telegrams with std. and ext. identifiers have different TDs. They differ only in the length of the identifiers. 18 bits therefore are not allocated in the TD of a std. Tg. They cannot be used by the application because they are overwritten by the reception of a Tg.

Forms of access:
r: read
w: write
i: init (BI halted or CM = inactive)
w0: clear
w1: set

CM **Communication Mode:**
CPU: r,i BI: r

CM defines the type of telegram.
0: Inactive Inactive. No participation in the bus traffic.
1: Send Send data.
2: Receive Receive data.
3: Fetch Fetch data via remote frame.
4: Provide Have data fetched via remote frame.
5: Rx-All Receive everything.
6: rsvd Don't use (provis. EoCA)
7: EoCA End of Communication Area.

As long as the CO is inactive (CM = 0) or locked (LCK = TRUE), the BI accesses the first byte of the CO only by reading. All other bytes are neither read nor written. The inactive mode is suitable therefore for reconfiguration of a CO online; i.e. while the node is taking part in the bus traffic.

RSC **Rescan:**
CPU: r,w BI: r

If the rescan bit has been set in a transmit object just processed, the search for active Tx objects is started at the beginning of the communication area. Otherwise, the search continues at this transmit object until the end of the CA is reached. From there, the system jumps back to the beginning of the CA.

MID **Mask Identifier:**
CPU: r,w BI: r

If MID has been deleted, the identifier received is compared bit-by-bit with the identifier from the telegram descriptor, i.e. the entire identifier must be the same so that the telegram received is transferred into this CO. If MID has been set, only bits which are allowed in the ID mask register of the GCS are used for the comparison.

Extended Addr. Format (EXF is set)

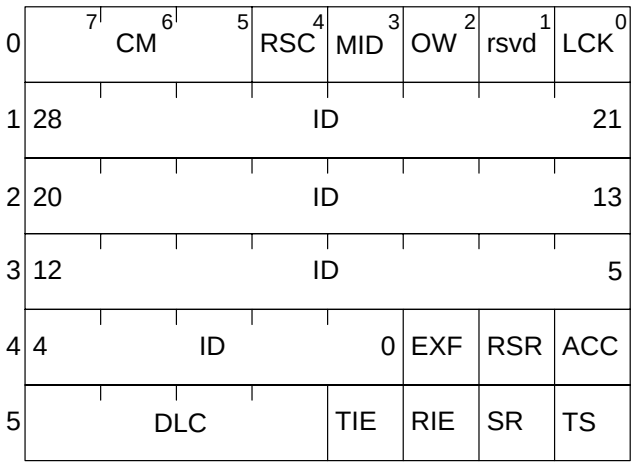

Fig. 3–2: Extended TD Map

Standard Addr. Format (EXF is deleted)

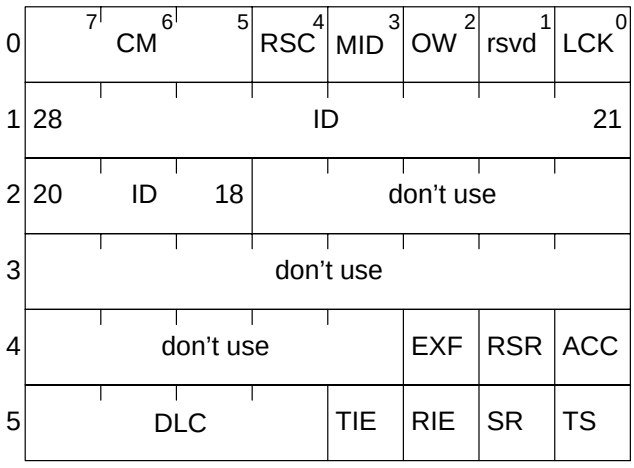

Fig. 3–3: Standard TD Map

OW **Overwrite:**
CPU: r,w BI: r

When OW is set, the com. object may be overwritten even if the application has not yet fetched the contents (TS set). The BI must of course obtain right of access (LCK deleted).

LCK **Lock:**
CPU: r,w BI: r

Lock determines the right of access for the BI.
TRUE: BI does not have right of access.
FALSE: BI has right of access.

ID **Identifier:**
CPU: r,i BI: r,w

The ID contains the address of the telegram. 11 bits in the standard mode or 29 bits in the extended mode.

ACC **Access:**
CPU: r BI: r,w

Access determines the right of access for the CPU.
TRUE: CPU has right of access.
FALSE: CPU does not have right of access.

RSR **Remote Send Request:**
CPU: r,i BI: r,w

In the provide mode, RSR signals a send request from outside; in the fetch mode it means that a remote Tg is being sent. It is set by the BI if a remote telegram has been received. It is deleted as soon as the corresponding data telegram has been transmitted.

EXF **Extended Format:**
CPU: r,i BI: r,w

In order to send/receive telegrams with extended address format, this flag must be switched on. For standard telegrams it is deleted.

DLC **Data Length Code:**

The DLC fixes the number of data bytes transmitted. Only telegrams with 0 to max. 8 data bytes are transmitted. If the DLC of a TxTg contains a value >8, the entered DLC and exactly 8 bytes will be transmitted. In the case of RxTgs the received DLC, and therefore also values > 8 will be entered by BI.

TIE **Tx Interrupt Enable:**
CPU: r,w BI: r

Masks the Tx interrupt for this com. object.

RIE **Rx Interrupt Enable:**
CPU: r,w BI: r

Masks the Rx interrupt for this com. object.

SR **Send Request:**
CPU: r,w1 BI: r,w0

With SR, the microprocessor issues a send request. Both the microprocessor and the BI write the SR flag. If the microprocessor writes a one, the telegram is sent. The BI deletes the SR flag after successful transmission.

TS **Transfer Status:**
CPU: r,w BI: r,w1

The TS flag is set by BI after a successful transfer and is deleted by the microprocessor after a com. object has been processed.

3.4.2. Time Stamp

The last two bytes in the CO are used for the time stamp.

At each SoF (Start of Frame) the free-running 16-bit counter CTIM is loaded into a register. When the Tg has been correctly transmitted, this register is copied to the two time stamp bytes of the corresponding CO.

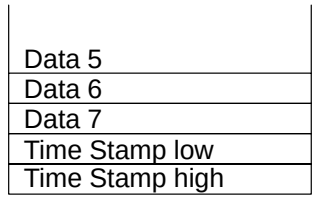

Fig. 3–4: Time stamp

4. Application Notes

4.1. Initialization

In the initialization phase, a configuration of the CAN node takes place. The mode of operation of the BTL and the bus coupling is set. The communication area is created in the CAN-RAM. The different telegrams are specified in it.

The-CAN node must be halted (HACK = TRUE) to carry out the initialization. After a reset, the flags HLT and HACK are set and initialization can take place. If initialization is required online, the flag HLT must be set. However, the BI must terminate any current transmission before it comes to a halt. For the user this means that he must wait until HACK has been set. If HLT is deleted after initialization, then BI begins to participate in the bus traffic and to scan the CA for tasks.

During the initialization, the error status register (ESR) and the interrupt index (IDX) should be deleted, otherwise no interrupts can be initiated.

If telegrams with different identifiers are to be received in a single CO, the identifier mask register must be initialized. This defines which bit of the ID received must be the same as the ID in the CO.

The bit timing registers 1, 2 and 3 and the output control registers 1 and 2 must be initialized in all cases.

The CA must be created in the CAN-RAM. The different COs are created one after the other starting at the address 0. It is important at this point that the three MSBs have been set in the first byte after the last CO, i.e. at an address divisible by 16 (CM = End of CA). This is not necessary if the CAN-RAM is completely filled with COs.

The communication mode (CM), the identifier, the data length code, the extended format flag (EXF) and the remote send request flag must be initialized inside each CO. The lock flag (LCK) must be deleted and the access flag (ACC) must be set in order that the BI may also view this CO. The transfer status flag (TS) must be deleted so that interrupts are not initiated erroneously.

4.2. Handling the COs

4.2.1. Principles

If the user wishes to access a CO, then he must lock out the BI from access to it. Also the BI reserves access for itself to one CO. In this case the user may not have access. When scanning the CA, the BI ignores inactive or locked COs; i.e. it reads only the first byte and then jumps to the next CO.

Reservations Procedure
If the user would like to access a com. object, then he must first set LCK. Then he must interrogate ACC. If it is TRUE, he has right of access. After the operation he must delete LCK.

```
LCK = TRUE;
if (ACC == TRUE)
    {
    /* CPU has right of access */
    }
LCK = FALSE;
   _________________ or ___________________
LCK = TRUE;
while (ACC == FALSE)
    {
    */ wait until BI is ready */
    }
/* CPU has right of access */
LCK = FALSE;
```

Fig. 4–1: Access to a CO by the user

When the BI is accessing a com. object, it first deletes ACC and then reads LCK. If LCK is FALSE, it has right of access.

```
ACC = FALSE;
if (LCK == FALSE)
    {
    /* BI has right of access */
    }
ACC = TRUE;
```

Fig. 4–2: Access to a CO by the BI

The BI does not wait at a CO until it becomes free.

The BI scans the CA from beginning to end. After a TxTg has been transmitted, the next TxTg entered is reported ready to send.

It makes sense to enter the COs in the CA in order of their priority. The priority is determined by the ID. The lowest ID has the highest priority. If the first bits of an extended ID are identical with a standard ID, the standard ID has higher priority. The CO with the highest priority is at the beginning of the CA. This ensures that Tx-Tgs with high priority are transmitted first when a rescan is initiated.

4.2.2. Configuration

A CO may be configured only in the inactive and/or locked mode or when HACK has been set. Otherwise it can lead to access conflicts between the user and BI.

The communication mode (CM) is determined in the configuration phase. The identifiers are also entered. The flag EXF must not be overlooked. The flag RSR and DLC determine whether and how many data bytes will be transmitted in the telegram. The interrupts can be permitted. In the case of a receive telegram it is necessary under certain circumstances to set the flags MID and OW. In the case of a transmit telegram, the flag RSC must be adjusted.

4.2.3. Transmit Telegram

CM = Send

A transmit telegram is used to send data. How many data bytes will be sent is fixed in the DLC. The data is entered directly after the TD. Unused data bytes can be freely used by the user. If after the transmission of this telegram the user would like the next Tx-Tg in the CA to be sent, he deletes the RSC flag. If he sets the RSC, then the transmit search starts again at the beginning of the CA. The RSR flag has been deleted.

The set SR flag tells BI that this telegram is to be sent; the SR can be likened to a postage stamp. The TS flag must be deleted before the CO is released with the deletion of LCK.

If the BI finds a CO whose SR flag has been set, it reserves this (ACC = FALSE) and reports it ”ready to send”. It will be transmitted as soon as no higher-priority telegrams occupy the bus. After successful transmission, it deletes the flag SR and sets TS. The setting of ACC re-releases the CO. Whether an interrupt will be triggered depends on whether IDX in the GCS contains the value minus one (255) and transmit interrupts are permitted.

The user should now reserve the CO, reset the flag TS and delete IDX so that other interrupts can also be reported. Should he wish to send further data, he can now enter this.

4.2.4. Receive Telegram

CM = Receive

With a receive telegram, data is received. If the EXF flag and the unmasked bits of the identifier of a received telegram are the same as those of a receive CO, the telegram will be copied to the CO. At the same time, the entire ID, the DLC and the data bytes are overwritten by the received ID, DLC and data. Only as many data bytes as the received DLC specify will be overwritten (max. 8). The DLC actually received will be entered. A permitted receive CO is only used when TS has been deleted or OW has been set.

Once a telegram has been received and copied to a CO, the flag TS is set. An interrupt will also be initiated if receive interrupts are permitted and IDX contains the value minus one (255).

If the user detects the receipt of a telegram (TS set), he must reserve the CO. Then he can read the data and, before releasing the CO again, delete TS.

4.2.5. Receive All Telegrams

CM = Rx-All

If, while searching for an RX-CO, the BI comes across a free Rx-All-CO, the received telegram will be entered here without regard to ID and EXF.

Rx-All-COs should be applied at the end of the CA.

4.2.6. Fetch Telegram

CM-Fetch

A fetch CO is used to request data from another node. This is done by sending a telegram with the identifier of the desired data. The remote transmission request flag is set in this Tg. No data is therefore sent with it. If another node has the desired data available, this is transmitted with the same ID as soon as bus traffic allows.

In this mode, only the reception of the data telegram can trigger an interrupt.

The sequence of a fetch cycle is represented for the user in pseudo-code.

```
if (TS == FALSE && SR == FALSE)    /* CO is empty */
  {
  LCK = TRUE;                 /* claim CO */
  /* wait until BI released this CO */
  while (ACC == FALSE) {/* do anything else */}
  SR = TRUE;                /* send this Tg */
  TS = FALSE;
  LCK = FALSE;              /* release CO */
  }
```

The BI now transmits the telegram with the RTR flag set. The other node receives the Tg, provides the data and returns the telegram with RTR flag deleted. After the reply telegram has been received, the BI sets the flag TS. The user waits for the data.

```
/* wait for answer */
while (TS == FALSE) {/* do anything else */}
LCK = TRUE;          /* claim CO */
/* wait until BI released this CO */
while (ACC == FALSE) {/* do anything else */}
/* copy data */
TS = FALSE;
LCK = FALSE;         /* release CO */
```

Instead of waiting for the answer, it is also possible for notification to be given by a receive interrupt.

4.2.7. Provide Telegram

CM = Provide

A provide CO is used to prepare data for fetching. It is the counterpart of a fetch CO. In a provide CO the RSR flag is cleared. It will be set and deleted by the BI. The data can be prepared in two ways:

In the first case, the user does not become active until a remote frame has been received (Rx interrupt or polling from RSR). After the CO has then been reserved, the data is written, the SR flag is set and the CO is released. The BI ensures then that the data is transferred back.

In the second case, the data has already been entered, SR has been set and TS deleted before the request. When the remote frame is received, the user does not need to become active. Also, no Rx interrupt will be initiated. The data is simply fetched. In this case the requesting RTR telegram must contain the correct DLC because, with an RTR telegram too, a received DLC overwrites the local DLC.

In both cases a Tx interrupt can occur after the data telegram has been transmitted.

4.2.8. Data Length Code

The data length code is 4 bits long. It can therefore contain values between 0 and 15. In principle, no more than 8 bytes can be transmitted. Empty data telegrams (DLC = 0) are also possible.

If a telegram with a DLC greater than 8 is received, this value will be written into the DLC of the CO, but exactly 8 bytes of data will be copied.

If the DLC of a Tx-CO contains a value greater than 8, this DLC will be transmitted, but only 8 bytes of data.

4.2.9. Overwrite Mode

The BI normally processes a CO only when the transfer status TS has been deleted; i.e. the user has processed the CO since the last transmission. In the case of COs with which telegrams are received, the TS flag can be by-passed. If overwrite (OW) is permitted, the BI may overwrite a previously received telegram. When accessing data therefore, the user always receives the most up-to-date data.

4.3. Interrupts

All interrupts are enabled or disabled by the global interrupt enable flags, GTIE for Tx interrupts, GRIE for Rx interrupts and EIE for error interrupts in the GCS register. This is the only location for the error interrupts. A Tx interrupt can be enabled in the corresponding CO with the Tx interrupt enable flag TIE. An Rx interrupt can be enabled in the corresponding CO with the Rx interrupt enable flag RIE.

An interrupt can only be initiated when the interrupt index IDX is empty (minus one). To initiate it, the BI enters the number (0...253) of the appropriate CO in the IDX. When an error interrupt is involved, the number 254 is entered.

The BI attempts to initiate an interrupt immediately after successful transfer. If this does not work (IDX not empty), the interrupt is pending (also error interrupt).

The BI permanently scans the CA. If, while doing so, it finds a CO whose interrupt condition is satisfied (e.g. TIE and TS are set), it reports this. This means that interrupts not yet reported will not be reported in the sequence of their occurrence, but in the sequence in which they are discovered later.

The interrupt service routine of the user must read the IDX. The interrupt source is stored here. If IDX points to a CO (0...253), the user must reserve this. After this, he must first delete TS so that this CO does not initiate an interrupt again. Only then may he release IDX (IDX = 255) so that the BI can enter further interrupts.

4.4. Rescan

The normal transmit strategy searches for the next transmit CO in the CA. If all the transmit COs are ready to send, they are processed one after the other. This is a democratic strategy.

If higher-priority TxTgs are reported in the meantime, these are not processed until the complete list has been finished. With rescan, the search for Tx telegrams is started again at the beginning of the CA. By this means the user can force the normal strategy to be interrupted and a search to be made first of all for higher-priority

TxTgs. A transmit CO already reported will of course be transmitted first.

The rescan requirement can be achieved dynamically, when a transmit CO is reported, by setting the global rescan flag GRSC.

It is also possible to configure a rescan strategy statically. Each Tx-CO has the rescan flag RSC. If it is set, the system starts from the front with the transmit search after this CO has been processed. It is possible, for instance, to set RSC in the low-priority Tx-COs. Each time a low-priority Tx-CO has been handled, the search continues for higher-priority objects.

The user must ensure that each Tx-CO is processed.

4.5. Time Stamp

The time stamp of a CO shows the user how much time has elapsed since the transmission of the object. For this purpose, he compares the time stamp with the capture timer CTIM. Because the time stamp contains the value of the CTIM at the time of the start of transmission, the difference is proportional to the time which has elapsed.

The time stamp mechanism also enables network-wide synchronization. A master transmits a Tg. All note the transmission time (local time). Then the master transmits its (global) transmission time. The difference between local and global time shows by how much one's own clock (timer) is wrong.

4.6. Errors

In the error status register (ESTR) error messages and status data are collected which can generate an error interrupt. As long as a flag is set in the ESTR, the flag ERS is also set in the status register. This means that the value 254 is written in IDX and an interrupt is generated when EIE has been set.

An error interrupt is deleted by first deleting ESTR and then releasing IDX.

The 5 flags BIT, STF, CRC, FRM and ACK originate from the protocol manager. The flag GDM (Good Morning) is not an error flag. GDM is set when the BI is aroused from the sleep mode by a dominant bus level.

The flag ECNT (error counter level) indicates that an error counter has exceeded a limit value. It is set when the transmit error counter exceeds the values 95, 127 and 255 or the receive error counter exceeds the values 95 and 127.

When the BI is in the Bus-Off mode, it no longer actively participates in the bus traffic. Nor does it receive telegrams, but continues to observe the bus. As soon as the BI has detected 128 x 11 successive recessive bits, it reverts from the Bus-Off mode to the error-active mode. At the same time the error counters are cleared.

4.7. Layout of the CA

The CA contains all COs beginning with the lowest identifier. The three MSBs must be set in the byte after the last CO (End of CA).

If the BI has received an identifier complete, it starts at the beginning of the CA with the search for an appropriate Rx-CO. If a rescan is initiated, the BI also starts from the beginning with the transmit search.

4.7.1. Buffers

Several successive receive COs may be allocated with the same identifier. The BI stores a received Tg in the first free Rx-CO. Using this mechanism it is possible to construct a receive buffer. If RIE is set in the last CO, the CPU is not informed until the buffer is full resistance.

4.7.2. Basic/Full CAN

For a Basic CAN application, a single Tx-CO will be used. All outgoing telegrams will be transmitted with this. The user must receive all Rx-Tgs and must himself decide whether he needs it (acceptance filtering). For this case it is possible to use an Rx-All-CO. But it is necessary to ensure that this can be processed before the next Tg arrives.

For this reason, it is a good idea to employ 2 or 3 Rx-All-COs as buffers after the Tx-CO.

In the case of a FullCAN application, one uses the built-in acceptance filtering and sets up a CO specifically for each desired Rx-Tg and Tx-Tg.

If the CAN-RAM is not adequate, mixed strategies are also possible. The acceptance filtering, of course, burdens the CPU with communication tasks.

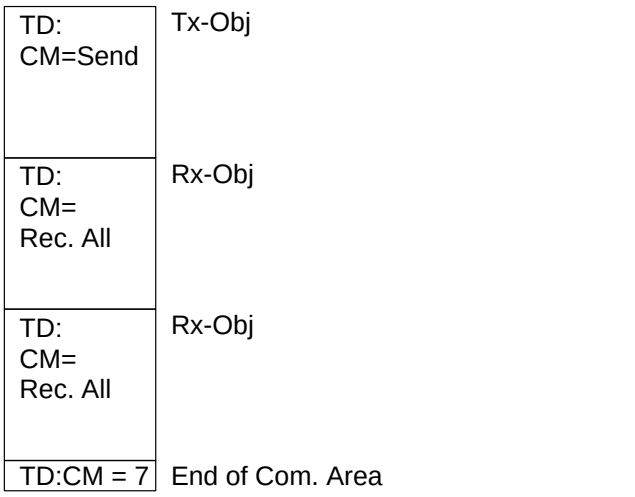

Fig. 4–3: Example: CA of a BasicCAN with 2 Rx-buffers

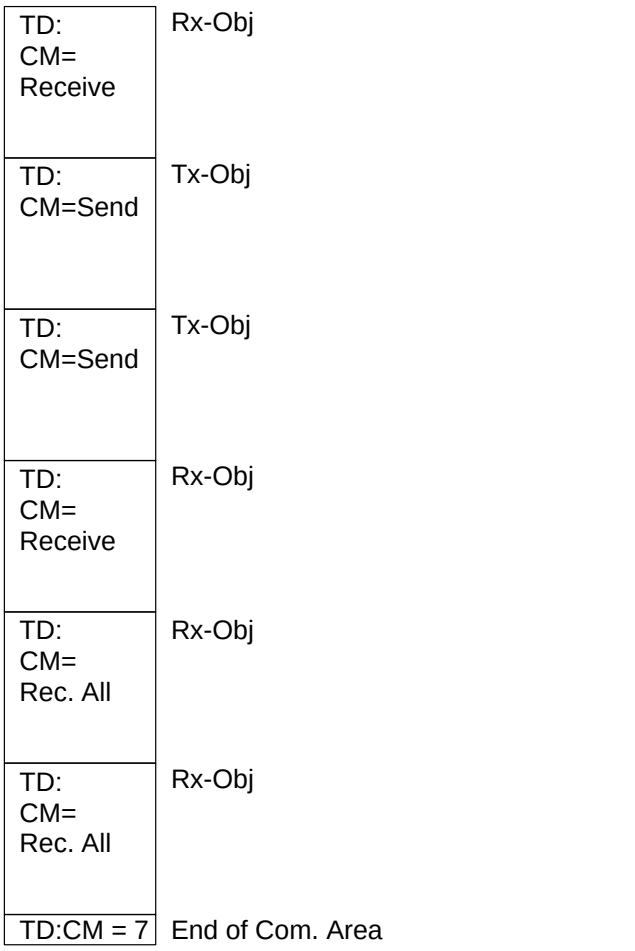

Fig. 4–4: Example: CA of a FullCAN with 2 Rx-objects, 2 Tx-objects, and 2 Rx-buffers

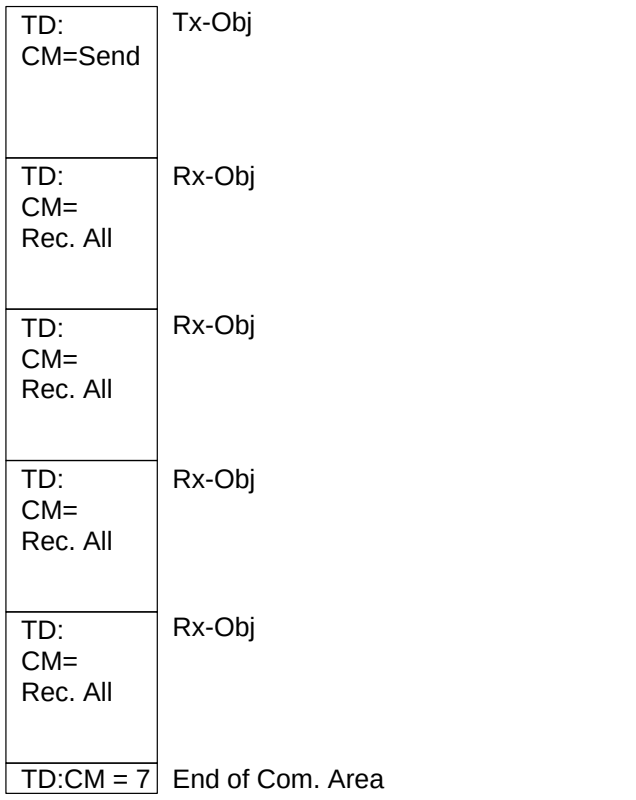

Fig. 4–5: Example: CA of a BasicCAN with 4 Rx-buffers

4.7.3. Bus Monitor

With some Rx-All-COs it is possible to construct a user-friendly bus monitor. The CPU has merely to observe whether anything has been received. The contents of the CO must be stored. The transmission time can be calculated from the time stamp.

4.7.4. Maximum number of COs

The maximum number of COs depends on the size of the CAN-RAM, the band-rate, the system clock, the BI and the CPU accesses to the CAN-RAM.

The BI can handle a maximum of 254 objects. The limiting factor is the 8-bit register IDX in the GCS. IDX can contain 256 different values. The values 255 (empty) and 254 (error) are reserved. The remaining values 0...253 can indicate 254 objects.

The maximum number of COs is, of course, limited to a greater extent by the size of the CAN-RAM. The BI can only access the CAN-RAM. Therefore the CA can only be applied there.

16 bytes are reserved for each CO. One extra byte for coding EoCA after the last CO must not be forgotten. The CAN-RAM area after the EoCA is freely available to the user. No EoCA is necessary if the CAN-RAM is filled completely with COs.

$$Max.\ Number\ CO = \frac{Ram\ Size}{16}$$

There are a maximum number of 16 COs possible in a CAN-RAM of 256 bytes.

The next limiting factor can be calculated from the baud rate and system clock. After the BI has received an identifier, it must be possible for it to scan the entire CA before the telegram comes to an end.

$$Max.\ Number\ CO = \frac{t_{CA\ Scan}}{t_{CO\ Scan}}$$

$$t_{CO\ Scan} = 9\ t_{Clk}$$

$$t_{CA\ Scan} = 28\ t_{Bit}$$

$$t_{Bit} = (TSEG1 + TSEG2 + 3)\ t_Q$$

$$t_Q = (BPR + 1)\ t_{Clk}$$

$t_{CA\ Scan}$ is the time required to scan the CA. $t_{CO\ Scan}$ is the time needed to process an object.

With an oscillator frequency of 8 MHz and a baud rate ($1/t_{bit}$) of 1 MBd, the BI could handle 24 COs. Naturally, this value needs to be rounded off.

The value thus calculated is further limited, however, by the CPU accesses to the CAN-RAM. Each cycle required by the CPU to write or read data in the CAN-RAM is missing from the BI. The BI is halted by CPU accesses. This reduces the time which the BI has to scan the CA. Where there is a reduced CPU clock time, in particular, the user should have only limited access to the CAN-RAM.

$$K_{BI} = \frac{Z_{BI}}{Z_G}$$

$$Max.\ Number\ CO = \frac{K_{BI}\ t_{CA\ Scan}}{t_{CO\ Scan}}$$

Z_{BI} is the number of BI cycles in the total cycles (Z_G), over a relatively long period (mean value). K_{BI} therefore represents a correction factor.

Example for an 8-bit CPU:

The Load and Store-Accu commands require 4 cycles.
OP code Adr.L Adr.H DB
Of the 4 cycles, only the last occupies the CAN-RAM.
If a block move without loop is programmed,
LDA 600;
STA 680;
LDA 601;
STA 681;
etc.
then only 3 of 4 cycles remain for the BI, i.e. 75%. K_{BI} would then be 0.75. The maximum number of COs is then 18. This applies only when source and destination lie in the CAN-RAM. If one of the two lies outside, then K_{BI} is 0.875.

If, of course, a loop is programmed,

LDA 600,X	Z_{BI} = 3 of 4
STA 680,X	Z_{BI} = 3 of 4
DEX	Z_{BI} = 2 of 2
BNE NXT	Z_{BI} = 2 of 2

10 of 12 cycles are available to the BI. This gives a K_{BI} of 0.833. The BI can then handle 20 COs. If the source and destination are not in the CAN-RAM, there are as many as 22.

Example for an 16-bit CPU:

The Load and Store-Accu commands require 5 cycles.
OP code Adr.L Adr.H DBL DBH

Of the 5 cycles, the last two occupy the CAN-RAM. In the worst case, 3 of 5 cycles remain for the BI, i.e. 60%. K_{BI} would then be 0.6. The maximum number of COs is then 14.

5. Bit Timing Logic

In the bit timing logic the transmission speed (baud rate) and the sample point within one bit will be configured. By shifting the sample point it is possible to take account of the signal propagation delay in different buses. Furthermore, the nature of the sampling and the bit synchronization can also be defined.

5.1. Baud Rate Prescaler

The baud rate prescaler is a 6-bit counter. It divides the system clock down by the factor 1...64. The output is the clock for the bit timing logic. This clock TQ_{CLK} defines the time quantum (t_Q). The time quantum is the smallest time unit into which a bit is subdivided.

5.2. Bit Timing

A bit duration consists of a programmable number of TQ_{CLK} cycles. The cycles are split up into the segments SYNCSEG, TSEG1 and TSEG2.

5.2.1. Bit Timing Definition

Sync.Seg.
It is expected that a bit will begin in the synchronization segment. If the bit level changes, the resynchronization ensures that the edge lies inside this segment. The sync.seg is always one time quantum long.

Prop.Seg.
This part of a bit is necessary to compensate for delay times of the network. It is twice the sum of the signal propagation delay on the bus plus input comparator delay plus output driver delay.

Phase Seg.
Phase segments 1 and 2 are necessary to compensate phase differences. They can be lengthened or shortened by resynchronization.

Sample Point
The bus level is read at this point and interpreted as a received bit.

TSEG1
The CAN implementation combines propagation delay segment and phase segment 1 to form time segment TSEG1.

TSEG2
TSEG2 corresponds to phase segment 2.

SJW
The synchronization jump width gives the maximum number of time quanta by which a bit may be lengthened or shortened by resynchronization.

$$t_{Bit} = t_{SYNCSEG} + t_{TSEG1} + t_{TSEG2}$$

$$t_Q = t_{Clk}\ (BPR + 1)$$

$$t_{SYNCSEG} = 1\ t_Q$$

$$t_{TSEG1} = (TSEG1 + 1)\ t_Q$$

$$t_{TSEG2} = (TSEG2 + 1)\ t_Q$$

$$t_{SJW} = SJW\ t_Q$$

The baud rate is then calculated as follows:

$$BR = \frac{1}{t_{Bit}}$$

$$t_{Bit} = t_{Clk}\ (BPR + 1)\ (3 + TSEG1 + TSEG2)$$

$$BR = \frac{f_{OSC}}{(BPR + 1)\ (3 + TSEG1 + TSEG2)}$$

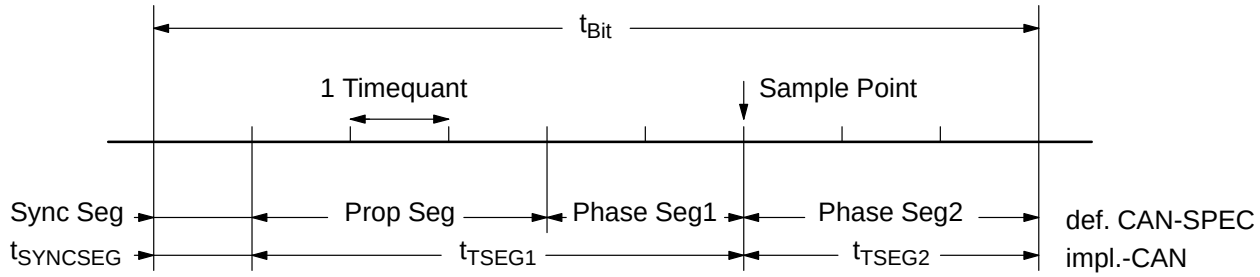

Fig. 5–1: Bit Timing Definition

5.2.2. Bit Timing Configuration

Certain boundary conditions need to be observed when programming the bit timing registers. The correct location of the sample point is especially important with maximum bus length and at high baud rate.

$t_{TSEG2} \geq 2\ t_Q$ = Information Processing Time

$t_{TSEG2} \geq t_{SJW}$

$t_{TSEG1} \geq 3\ t_Q$

$t_{TSEG1} \geq t_{TSEG2}$

$t_{TSEG1} \geq t_{Prop} + t_{SJW}$

The information processing time is the internal processing time. After the receipt of a bit (sample point) this time is needed to calculate the next bit for transmission.

With a baud rate of 1 MBd a bit should be at least 8 t_Q long.

In the case of a triple sample mode (MSAM = 1), the following boundary condition must also be observed:

$$t_{TSEG1} \geq t_{Prop} + t_{SJW} + 2\ t_Q$$

The triple sample mode offers better immunity to interference signals. In the simple sample mode a higher transmission speed is possible.

For high baud rates and maximum bus length, neither SYN nor MSAM may be switched on. Bosch advises against both adjustment facilities. When an input filter matched to the baud rate or a bus driver is used, the triple sample mode is not necessary. If SYN is set, synchronization will also be made with the soft edge (dominant to recessive) and this will mean higher demands being imposed on the clock tolerances.

5.2.3. Synchronization

The BTL carries out synchronization at an edge (change of the bus level) in order to compensate for phase shifts between the oscillators of the different CAN nodes.

5.2.3.1. Hard Synchronization

Hard synchronization is carried out at the start of a telegram. The BTL ensures that the first negative edge is in the sync. seg.

5.2.3.2. Resynchronization

Resynchronization takes place during the transmission of a telegram. If the BTL detects an edge outside the sync. seg., it can lengthen or shorten the bit. If it detects the edge during TSEG1, t_{TSEG1} is lengthened. If it detects the edge during TSEG2, t_{TSEG2} is shortened. In this way, it ensures that the edges lie in the sync. seg. T_{SJW} is the maximum time a bit can be lengthened or shortened.

Two forms of resynchronization are possible. In normal operation, synchronization is carried out only with the negative edge (dominant $ recessive). At low transmission speeds, synchronization can also be carried out with the rising edge (SYN = 1).

6. Bus Coupling

The bus coupling describes the connection of the internal signals rx (receive line) and tx (transmit line) to the pins to the CAN bus.

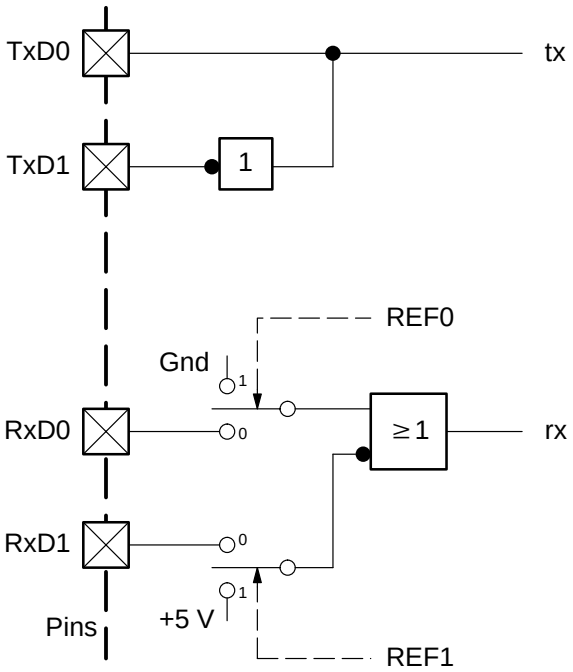

Fig. 6–1: Bus coupling

Dominant: tx = 0; TxD1 > TxD0; TxD0 = 0; TxD1 = 1
rx = 0; RxD1 > RxD0; RxD0 = 0; RxD1 = 1
Recessive: tx = 1; TxD1 < TxD0; TxD0 = 1; TxD1 = 0
rx = 1; RxD1 > RxD0; RxD0 = 1; RxD1 = 0

The output pins are push/pull drivers for TLL levels. The input pins are also designed for TTL levels.

Integrated transceivers (Siliconix Si9200, Philips 82C250 etc.) are available for physical coupling in the high-speed range in compliance with ISO/DIS 11898.

For a laboratory system a “minimum bus” can be constructed by means of a wire-Or circuit.

To utilize the advantages of differential signal transmission, an analogue comparator is necessary.

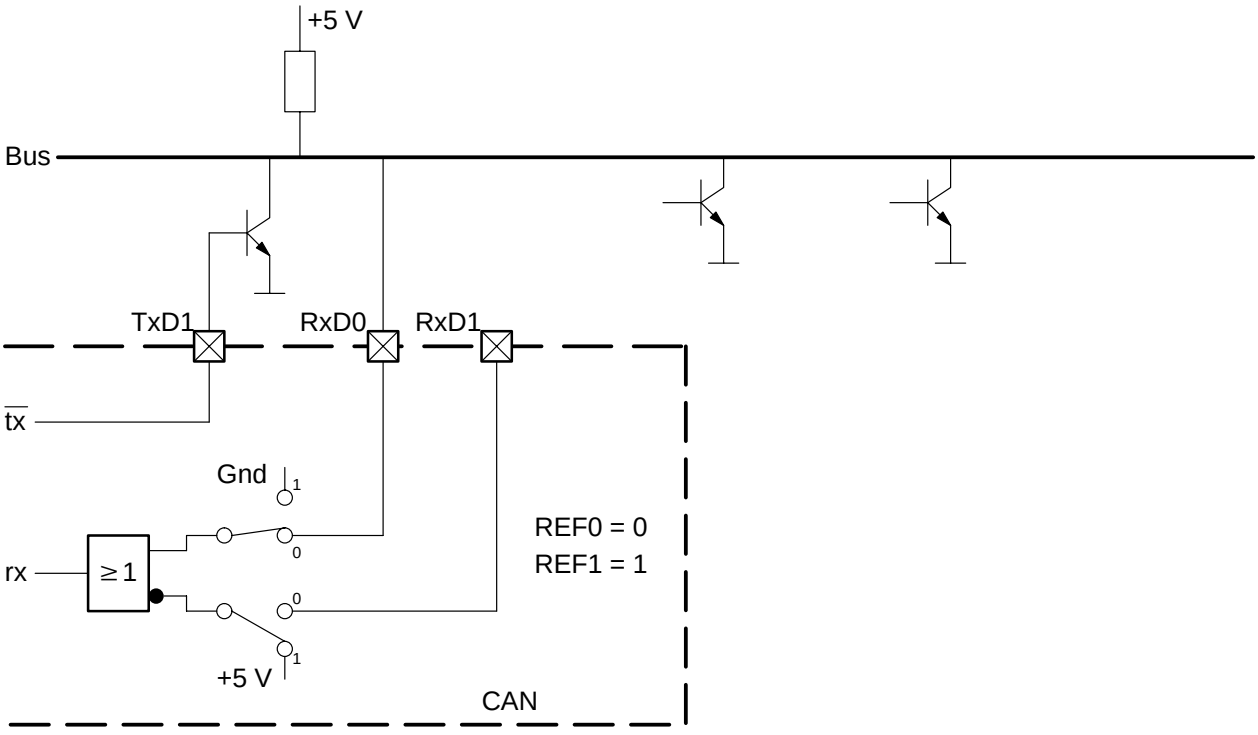

Fig. 6–2: Minimum bus

7. CAN Manual Documentation History

1. Advance Information: “CAN Manual”, Dec. 16, 1994, 6251-406-1AI. First release of the advance information. Valid for LCAN 0001.

2. Advance Information: “CAN Manual”, Sept. 26, 1995, 6251-406-2AI. Second release of the advance information. Valid for LCAN 0003.

3. Data Sheet: “CAN Manual”, Aug. 7, 1996, 6251-406-1DS. First release of the data sheet. Valid for LCAN 0005.

MICRONAS INTERMETALL GmbH
Hans-Bunte-Strasse 19
D-79108 Freiburg (Germany)
P.O. Box 840
D-79008 Freiburg (Germany)
Tel. +49-761-517-0
Fax +49-761-517-2174
E-mail: docservice@intermetall.de
Internet: http://www.intermetall.de

Printed in Germany
by Simon Druck GmbH & Co., Freiburg (08/96)
Order No. 6251-406-1DS