

CC750

(SPI-CAN)

Target Specification

Revision 1.2

16.08.00

spec_cover_inter.fm

K8/EIS - Klasse-2989

061.0/2.5 - 26.11.97

Robert Bosch GmbH
Automotive Equipment Division 8
Development of Integrated Circuits (MOS)

Copyright Notice and Proprietary Information

Copyright © 2000 Robert Bosch GmbH. All rights reserved. This specification is owned by Robert Bosch GmbH. The specification may be reproduced or copied. No part of this specification may be modified or translated in any form or by any means without prior written permission of Robert Bosch GmbH.

Disclaimer

ROBERT BOSCH GMBH, MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

ROBERT BOSCH GMBH, RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO THE PRODUCTS DESCRIBED HEREIN. ROBERT BOSCH GMBH DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT DESCRIBED HEREIN.

1. Introduction	5
1.1 General Data	5
1.2 Features	5
1.3 Functional Overview	8
1.4 CAN Controller	9
1.5 Intelligent Memory	9
1.6 CPU Interface Logic	9
2. Package Diagram	10
3. Product Description	11
3.1 Pin Description	11
3.2 Hardware Reset	12
3.2.1 Reset values of CC750 registers	12
3.2.2 Reset values of CC750 output pins	13
3.3 Software Initialisation	13
3.4 Configuration of Bit Timing	13
3.5 Silent Mode	14
3.6 Low Current Modes	14
4. Functional Description	15
4.1 CC750 Address Map	15
4.2 Control Register (00H)	16
4.3 Status Register (01H)	18
4.3.1 Status Interrupts	20
4.4 CPU Interface Register (02H)	21
4.5 Clocking Description	22
4.6 Global Mask - Standard Register (06-07H)	23
4.7 Global Mask - Extended Register (08-0BH)	23
4.8 Acceptance Filtering Implications	25
4.9 Message 15 Mask Register (0C-0FH)	26
4.10 Receive Error Counter (1FH)	27
4.11 Transmit Error Counter (2FH)	28
4.12 Bit Timing Registers	28
4.12.1 Bit Timing Overview	28
4.12.2 CC750 Bit Timing Definitions	29
4.12.3 CC750 Bit Time Segments	30
4.12.4 Calculation of the Bit Time	30
4.12.5 Example for Bit Timing at high Baudrate	31

4.12.6 Bit Timing Registers 0 + 1 (3FH + 4FH)	31
4.13 Interrupt Register (5FH)	33
4.14 Serial Reset Address (FFH)	34
4.15 CC750 Message Objects (MO)	34
4.15.1 Message Object Structure	34
4.15.2 Control 0 + 1 Registers	35
4.15.3 Handling of Message Objects	39
4.15.4 Arbitration 0, 1, 2, 3 Registers	41
4.15.5 Configuration Register	42
4.15.6 Data Bytes	43
4.16 Special Treatment of Message Object 15	43
5. FLOW DIAGRAMS	44
5.1 CC750 handling of Message Objects 1-14 (Transmit)	44
5.2 CC750 handling of Message Objects 1-14 (Receive)	45
5.3 CPU Handling of Message Objects 1-14 (Transmit)	46
5.4 CPU Handling of Message Objects 1-14 (Receive)	47
5.5 CPU Handling of Message Object 15 (Receive)	48
6. CPU Interface Logic	49
6.1 Serial Interface Techniques	49
6.2 Serial Interface Protocol	50
6.3 Serial Control Byte	51
7. Electrical Specification	53
7.1 Handling Instructions	53
7.2 Absolute Maximum Ratings	53
7.3 DC-Characteristics	53
7.4 A.C. Characteristics	54
7.5 Waveforms for testing	55
8. Appendix	58
8.1 Documentation of Changes	58
8.1.1 Changes on Revisions	58
8.1.1.1 Revision 1.1	58
8.1.1.2 Revision 1.2	58
8.1.2 Others	58

1. Introduction

1.1 General Data

Device Name	:	CC750
Package	:	SOIC16-W
Device Number	:	0 272 230 455
1 st Application	:	EDC16C2 Common Rail Controller

1.2 Features

- Supports CAN Protocol Version 2.0 A, B
 - Standard Data and Remote Frames
 - Extended Data and Remote Frames
- Programmable Global Mask
 - Standard Message Identifier
 - Extended Message Identifier
- 15 Message Objects of 8-byte Data Length
 - 14 Tx/Rx Buffers
 - 1 Rx Buffer with Shadow Buffer and Programmable Mask
- Programmable Bit Rate
- Flexible Interrupt Structure
- Flexible Status Interface
- Serial Interface
- SOIC16-W Package

The serial communications controller is a highly integrated device that performs serial communication according to the CAN Protocol Version 2.0 A, B. The CAN protocol uses a multi-master (contention based) bus configuration for the transfer of “communication objects” between nodes of the network. This multi-master bus is also referred to as CSMA/CR or Carrier Sense, Multiple Access, with Collision Resolution.

The CC750 performs all serial communication functions such as transmission and reception of messages, message filtering, transmit search, and interrupt search with minimal interaction from the host microcontroller, or CPU. The CC750 supports the standard and extended message frames in CAN Specification 2.0 part B. It has the capability to transmit, receive, and perform message filtering on extended message frames with a 29-bit message identifier. Due to the backward compatible nature of CAN Specification 2.0, the CC750 also fully supports the standard message frames in CAN Specification 2.0 part A.

A communication object consists of an identifier along with control data segments. The control segment contains all the information needed to transfer the message. The data segment contains from 0 to 8 bytes in a single message. All communication objects are stored in the Memory of the corresponding CAN chip for each node. A transmitting node broadcasts its message to all other nodes on the network. An acceptance filter at each node decides whether to receive that message. A message is accepted only if a communication object with a matching message identifier has been set up in the CAN Memory for that node.

CAN not only manages the transmission and reception of messages but also the error handling, without any burden on the CPU.

CAN features several error detection mechanisms. These include Cyclical Redundancy Check (CRC) and bit coding rules (“bit stuffing/destuffing”). The polynomial of the CRC has been optimized for control applications with short messages. If a message was corrupted by noise during transmission, it is not accepted at the receiving nodes. Current transmission status is monitored in the control segment of the appropriate communication object within the transmitting node, automatically initiating a repeated transmission in the case of lost arbitration or errors. CAN also has built-in mechanisms to locate error sources and to distinguish permanent hardware failures from occasional soft errors. Defective nodes are switched off the bus, implementing a fail-safe behaviour (thus, hardware errors will not let defective nodes control the bus indefinitely).

The message storage is implemented in an intelligent memory, which can be addressed by the CAN controller and the CPU. The CPU controls the CAN controller by selectively modifying the various registers and bit fields in the Memory. The content of the various bit fields are used to perform the functions of acceptance filtering, transmit search, interrupt search and transfer completion.

In order to initiate a transfer, the transmission request bit has to be written to the message object. The entire transmission procedure and eventual error handling is then done without any CPU involvement. If a communication object has been configured to receive messages, the CPU easily reads its data registers using CPU read instructions. The message object may be configured to interrupt the CPU after every successful message transmission or reception.

The CC750 features a serial interface to connect the CPU with less as possible leads. This minimizes the needed area and facilitate the board layout.

The CC750 provides storage for 15 message objects of 8-byte data length. Each message object can be configured as either transmit or receive, except for the last message object.

The last message object is a receive only double buffer with a special acceptance mask designed to allow select groups of different message identifiers to be received.

The CC750 also implements a global acceptance masking feature for message filtering. This feature allows the user to globally mask any identifier bits of the incoming message. There are different programmable global mask registers for standard and extended messages.

The CC750 provides an improved set of network management and diagnostic functions including fault confinement and a built-in monitoring tool. The built-in monitoring tool alerts the CPU when a global status change occurs. Global status changes include message transmission and reception, error frames, or sleep mode wake-up. In addition, each message object offers full flexibility in detecting when a data or remote frame has been sent or received.

The CC750 offers function compatibility with the CC770, Intel 82527 and 82526, with the following exceptions:

Differences to CC770:

- only one SPI mode
- no parallel ports
- no Clockout pin
- only one Tx and one Rx pin
- no parallel port and Clockout related registers/flags
- no Enable Additional Functions switch (always active)
- Receive Error Counter moved from address 06FH to 01FH
- Transmit Error Counter moved from address 07FH to 02FH

Differences to Intel 82526/7:

- only one SPI mode
- no parallel ports
- no Clockout pin
- only one Tx and one Rx pin
- no analog input comparator and $V_{CC}/2$ source
- no parallel port and Clockout related registers/flags
- readability of Receive and Transmit Error Counters added
- maskable Xtd and Dir bit in the Message 15 Mask register

The CC750 is fabricated in Bosch's reliable HC65-ST/0.65 technology and is available in a SOIC16-W Package for the automotive temperature range (-40 °C to +125 °C ambient).

1.3 Functional Overview

The CC750 CAN controller consists of three functional blocks. The CPU Interface logic manages the interface between the CPU (host microcontroller) and the CC750 using an Serial Peripheral Interface (SPI). The CAN controller interfaces to the CAN bus and implements the protocol rules of the CAN protocol for the transmission and reception of messages. The Memory is the interface layer between the CPU and the CAN bus.

The CC750 Memory provides storage for 15 message objects of 8-byte data length. Each message object has a unique identifier and can be configured to either transmit or receive, except for the last message object. The last message object is a receive only buffer with a special mask design to allow select groups of different message identifiers to be received.

Each message object contains control and status bits. A message object with the direction receive will send a remote frame by requesting a message transmission. A message object with the direction set as transmit will be configured to automatically send a data frame whenever a remote frame with a matching identifier is received over the CAN bus. All message objects have separate transmit and receive interrupts and status bits, allowing the CPU full flexibility in detecting when a remote or data frame has been sent or received.

The CC750 also implements a global masking feature for acceptance filtering. This feature allows the user to globally mask, or "don't care", any identifier bits of the incoming message. This mask is programmable to allow the user to design an application-specific message identification strategy. There are separate global masks for standard and extended frames.

The incoming message first passes through the global mask and is matched to the identifiers in message objects 1-14. If there is no identifier match then the message passes through the local mask in message object 15. The local mask allows a large number of infrequent messages to be received by the CC750. Message object 15 is also buffered to allow the CPU time to service a message received.

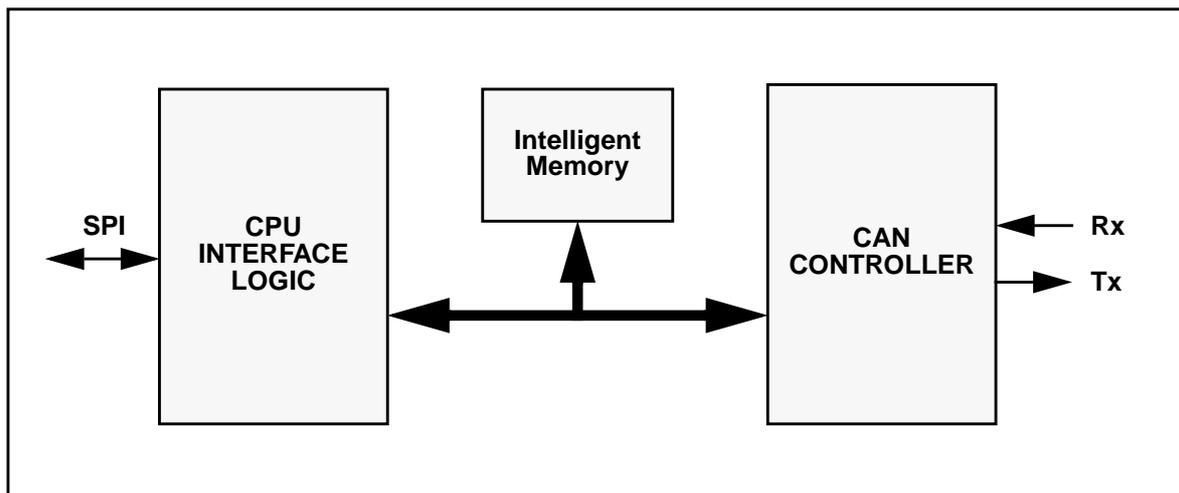


Figure 1: Block Diagram of CC750.

spec_general_features.fm

K0/EIS - Klose-2989

061/2/2.3 - 15.08.97

1.4 CAN Controller

The CAN controller controls the data stream between the Memory (parallel data) and the CAN busline (serial data). The CAN controller also handles the error management logic and the message objects.

1.5 Intelligent Memory

The Memory is content addressable (CAM) for the Can Controller which does the acceptance filtering in one clock cycle, whereas the CPU Interface Logic accesses the Memory via register address (RAM). The advantage of this access is the speed and the minimized area.

The access to the CAM is timeshared between the CPU Interface Logic and the CAN bus (through the CAN controller).

The Memory is addressed from 00H to FFH.

1.6 CPU Interface Logic

The CPU Interface Logic controls the data stream between the SPI (serial data) and the Memory (parallel data).

2. Package Diagram

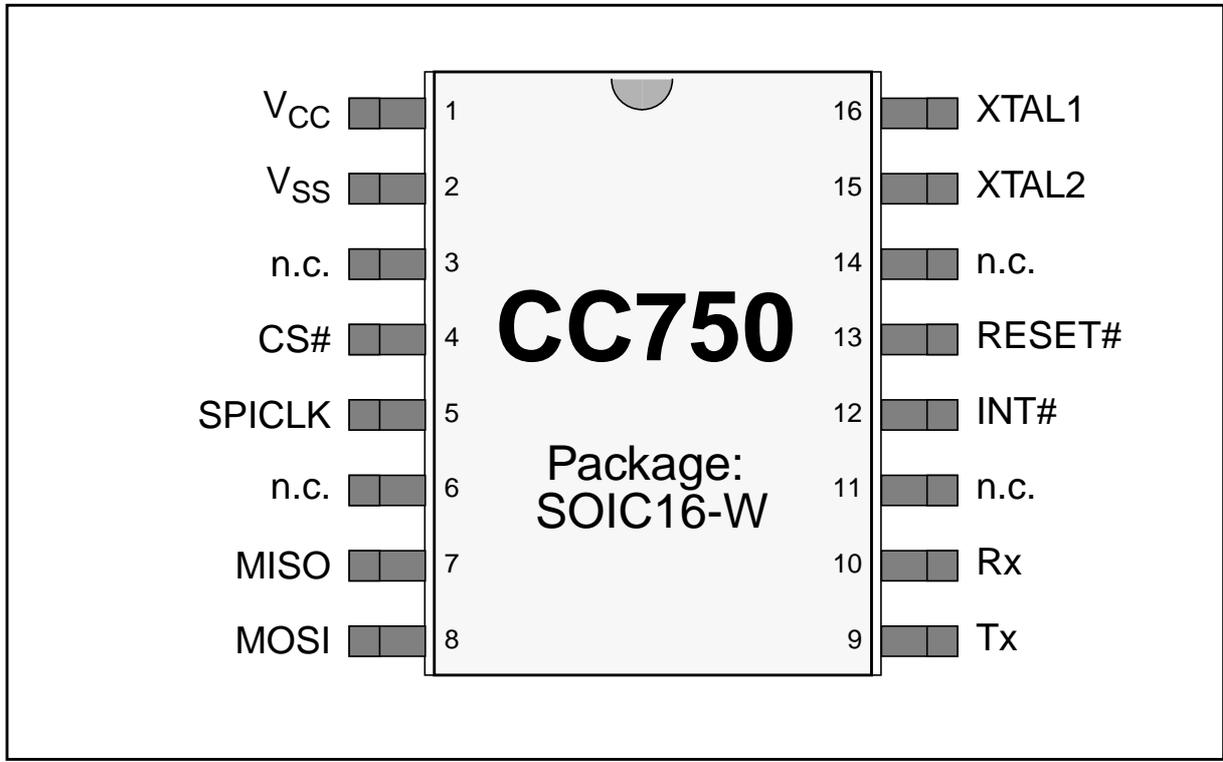


Figure 2: Package Diagram of CC750

3. Product Description

3.1 Pin Description

Symbol	Pin	Function
V_{CC}	1	Power connection must be shorted externally to +5V DC to provide power to the entire chip.
V_{SS}	2	Ground (0V) connection must be shorted externally to a V_{SS} board plane to provide digital ground.
n.c.	3	Should be connected to a V_{SS} board plane to increase EMV.
CS#	4	A low level on this pin enables the CPU to access the CC750.
SPICLK	5	Clock input of Serial Peripheral Interface.
n.c.	6	Should be connected to a V_{SS} board plane to increase EMV.
MISO	7	Master In Slave Out is the data output from the CC750 serial interface.
MOSI	8	Master Out Slave In is the data input from the CC750 serial interface.
Tx	9	Serial push-pull data output to the CAN bus transceiver. During a recessive bit Tx is high, during a dominant bit Tx is low.
Rx	10	Input from CAN bus transceiver. During a recessive bit Rx is high, during a dominant bit Rx is low.
n.c.	11	Should be connected to a V_{SS} board plane to increase EMV.
INT#	12	The interrupt pin is an open drain output (requires external pull up resistor) to the CPU.
RESET#	13	Warm Reset: (V_{CC} remains valid while RESET# is asserted), RESET# must be driven to a low level for 1 ms minimum. Cold Reset: (V_{CC} is driven to a valid level while RESET# is asserted), RESET# must be driven low for 1 ms minimum measured from a valid V_{CC} level. No falling edge on the Reset pin is required during a cold reset event.
n.c.	14	Should be connected to a V_{SS} board plane to increase EMV.
XTAL2	15	Push-pull output from the internal oscillator. XTAL2 and XTAL1 are the crystal connections to an internal oscillator. If an external oscillator is used, XTAL2 may not be connected. XTAL2 may not be used as a clock output to drive other CPU's.
XTAL1	16	Input for an external clock.

Table 1: Pin description

3.2 Hardware Reset

3.2.1 Reset values of CC750 registers

During power up, the RESET pin must be driven to a valid low level (V_{IL}) for 1 ms measured from a valid V_{CC} level to ensure the oscillator is stable. The registers of the CC750 have the following values after warm reset:

Register	Address	Reset Value
Control Register	00H	01H
Status Register	01H	00H
CPU Interface Register	02H	20H
Global Mask - Standard	06+07H	unchanged
Global Mask - Extended	08-0BH	unchanged
Message 15 Mask	0C-0FH	unchanged
Receive Error Counter	1FH	00H
Transmit Error Counter	2FH	00H
Bit Timing Register 0	3FH	00H
Bit Timing Register 1	4FH	00H
Interrupt Register	5FH	00H
SPI Reset Address	FFH	not readable
Message Objects 1-15		unchanged

Table 2: Reset values of CC750 registers

The error management counters and the Bus Off state are reset by a hardware reset.

If a hardware reset occurs at power on, registers defined as unchanged should be interpreted as undefined.

3.2.2 Reset values of CC750 output pins

The CC750 output pins have the following states after reset:

Pin	Reset state
Tx	1 (recessive state)
INT#	Z
MISO	Z

Table 3: Reset states of CC750 output pins

3.3 Software Initialisation

Software initialization is started by setting the Init bit in the Control Register, either by software, hardware reset, or by going Bus Off. While Init is set, all message transfers to and from the CC750 are stopped and the Tx output is high (recessive). The error counters are unchanged. Initialization is used to configure the CC750 Memory without interference to or by CAN bus.

Resetting Init completes initialization and the CC750 synchronizes itself to the CAN bus by waiting for 11 consecutive recessive bits (called bus idle) before it will take part in bus activities.

Note:

The Bus Off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting Init. If the device goes Bus Off, it will set Init of its own accord, stopping all bus activities. Once Init has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the Bus Off recovery sequence, the Error Management Counters will be reset.

During the waiting time after the resetting of Init, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the Bus Off recovery sequence.

Software initialization does not change configuration register values.

3.4 Configuration of Bit Timing

After setting bits Init and CCE in the CAN Control Register, the bit timing can be configured by writing to the Bit Timing Registers.

While Bit Timing Register 0 controls the (Re)Synchronisation Jump Width and the Baud Rate Prescaler, Bit Timing Register 1 is used to define the position of the Sample Point inside a Bit Time. For a detailed description how to program the bit timing see chapter 4.12.

3.5 Silent Mode

The CAN Controller can be set in Silent Mode by programming the Control Register bit Silent to one.

In Silent Mode, the CC750 is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus may remain in recessive state.

The Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames).

3.6 Low Current Modes

Power Down and Sleep Modes are activated by the PwD and Sleep bits in the CPU Interface Register (02H) under the control of the programmer.

During Power Down and Sleep Mode the CPU Interface Register is the only accessible register. In this mode the oscillator is not active and no access to the message objects is possible.

The CC750 exits from Power Down by either a hardware reset or by resetting the PwD bit to "0". The CPU must read the hardware reset bit (bit 7, register 02H) to ensure the CC750 has exited Power Down.

The CC750 enters Sleep Mode after the Sleep bit in the CPU Interface Register (bit 3, register 02H) is set and a possibly pending transmission on the CAN Bus has finished.

Sleep mode is exited by resetting the Sleep bit or when there is activity on the CAN bus. The CC750 requires a minimum of 10 ms to come out of Sleep Mode after bus activity occurs.

Power Down and Sleep Mode should not be entered directly after reset. The user program must perform a minimum Memory configuration at any time (preferably during the initialization) prior to entering these modes.

Programming the following registers satisfies the minimum configuration requirement:

- Control Register (00H) (set CCE bit to "1")
- CPU Interface Register (02H) (DMC bit application specific)
- Bit Timing Register 0 (3FH) (application specific)
- Bit Timing Register 1 (4FH) (application specific)
- All MO Control 0 Registers (reset MsgVal bit to "0")
- Control Register (00H) (reset Init and CCE bits to "0")

4. Functional Description

This section discusses the functional operation of the CC750 by describing the registers used to configure the chip and Message Objects.

4.1 CC750 Address Map

The CC750 allocates an address space of 256 bytes and is selected by activating signal CS#. All registers are organized as 8-bit registers.

Address	Register
00H	Control
01H	Status
02H	CPU Interface
03H	reserved
04H	reserved
05H	reserved
06+07H	Global Mask - Standard
08-0BH	Global Mask - Extended
0C-0FH	Message 15 Mask
10-1EH	Message 1
1FH	Receive Error Counter
20-2EH	Message 2
2FH	Transmit Error Counter
30-3EH	Message 3
3FH	Bit Timing 0 *
40-4EH	Message 4
4FH	Bit Timing 1 *
50-5EH	Message 5
5FH	Interrupt
60-6EH	Message 6
6FH	reserved
70-7EH	Message 7

Table 4: CC750 address map

Address	Register
7FH	reserved
80-8EH	Message 8
8FH	reserved
90-9EH	Message 9
9FH	reserved
A0-AEH	Message 10
AFH	reserved
B0-BEH	Message 11
BFH	reserved
C0-CEH	Message 12
CFH	reserved
D0-DEH	Message 13
DFH	reserved
E0-EEH	Message 14
EFH	reserved
F0-FEH	Message 15
FFH	Serial Reset Address

Table 4: CC750 address map

NOTE:

* The CPU may write to the Bit Timing Registers only if the CCE bit is "1" (Control Register).

4.2 Control Register (00H)

7	6	5	4	3	2	1	0
res	CCE	res	Silent	EIE	SIE	IE	Init
rw	rw	r	rw	rw	rw	rw	rw

The default value of the Control Register after a hardware reset is 01H.
 Reserved bits read as "0" and must be written as "0".

spec_functional_description.fm

K0/EIS - Klose-2989

061.2/2.3 - 15.08.97

CCE Change Configuration Enable

- one The CPU has write access to the Bit Timing Registers. Additionally the Init bit should be set in order to prevent CAN bus errors.
- zero The CPU has no write access to the Bit Timing Registers.

This bit is reset by the CPU to provide protection against unintentional rewriting of critical registers by the CPU following the initialization sequence.

Silent Silent Mode

- one Silent Mode is active
- zero Normal operation

In Silent Mode, the CC750 is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN Bus and it cannot start a transmission. For additional Information, see chapter 3.5.

EIE Error Interrupt Enable

- one Error interrupts enabled. A change in the error status of the CC750 will cause an interrupt to be generated.
- zero Error interrupts disabled. No error interrupt will be generated.

Error interrupts are BOff and Warn in the Status Register. Error Interrupt Enable is set by the CPU to allow the CC750 to interrupt CPU when an abnormal number of CAN bus errors have been detected.

It is recommended to enable this interrupt during normal operation.

SIE Status Change Interrupt Enable

- one Status Change Interrupt enabled. An interrupt will be generated when a CAN bus error is detected in the Status Register or a transfer (reception or transmission) is successfully completed, independent of the interrupt enable bits in any Message Object.
- zero Status Change Interrupt disabled. No status interrupt will be generated.

Status Change Interrupts are WakeUp, RxOK, TxOK, and LEC0-2 in the Status Register.

RxOK occurs upon every successful message transmission on the CAN bus, regardless of whether the message is stored by the CC750.

The LEC bits are very helpful to indicate whether Bit or Form Errors are occurring. In normal operation it is not advised to enable this interrupt for LEC since the CAN protocol was designed to handle these error conditions in hardware by error frames and the automatic retransmission of messages. When cumulative LEC occur, the warning and Bus Off flags will be set. The Error Interrupt should be enabled to detect these conditions.

In most applications, the SIE bit should not be set. Since this interrupt will occur for every message, the CPU will be unnecessarily burdened. Instead, interrupts should be implemented on a Message Object basis so interrupts occur only for messages that are used by the CPU.

The SIE bit is set by the CPU.

NOTE:

If the Status Change Interrupts and Message Object receive/transmit interrupts are enabled, there will be two interrupts for each message successfully received or transmitted by a Message Object.

IE Interrupt Output Enable

- one Interrupt Pin enabled.
- zero Interrupt Pin disabled. The CC750 will generate no interrupts although the Interrupt Register (5FH) will still be updated.

Applies to EIE, SIE, and Message Object Tx/Rx interrupts. For example the Interrupt Register (5FH) contains a value other than zero, indicating the interrupt source (Message Object or Status), and the IE bit is set to one, an interrupt will be generated. No interrupt will be lost because of periodic setting or resetting of this bit.

The Interrupt Output Enable bit is set by the CPU.

INIT Initialization

- one Software initialization is enabled.
- zero Software initialization is disabled.

Following a hardware reset, this bit will be set.

The Init bit is written by the CPU and is set by the CC750 when it goes Bus Off. Initialization is a state which allows the user to configure the CC750 Memory without the chip participating in any CAN bus transmissions. While Init equals one, all message transfers to and from the CAN bus are stopped, and the status of the CAN bus output Tx is recessive. Initialization will most often be used the first time after power-up and when the CC750 has removed itself from the CAN bus after going Bus Off.

Init should not be used in normal operation when the CPU is modifying transmit data; the CPUUpd bit in the Control 1 register from each Message Object is used in this case.

4.3 Status Register (01H)

7	6	5	4	3	2	1	0
BOff	Warn	WakeUp	RxOK	TxOK	LEC		
r	r	r	rw	rw	rw		

The default value of the Status Register after a hardware reset is 00H.

BOff Bus Off Status

- one There is an abnormal rate of occurrences of errors on the CAN bus.
- zero The CC750 is not Bus Off.

The Bus Off condition occurs when the Transmit Error Counter in the CC750 has reached the limit of 256. In consequence, the CC750 going Bus Off. During Bus Off, no messages

spec_functional_description.fm

0612/2.3 - 15.08.97 K3/EIS - Klose-2989

can be received or transmitted.

The only way to exit this state is by resetting the Init bit in the Control register (location 00H). When this bit is reset, the Bus Off recovery sequence begins. The Bus Off recovery sequence resets the Transmit and Receive Error Counters. After the CC750 counts 128 packets of 11 consecutive recessive bits on the CAN bus, the Bus Off state is exited.

The Bus Off Status bit is written by the CC750.

Warn Warning Status

- one There is an abnormal rate of occurrences of errors on the CAN bus.
- zero There is no abnormal occurrence of errors.

The Warning condition occurs when an error counter in the CC750 has reached the limit of 96. When this bit is set, an interrupt will occur if the EIE and IE bits of the Control Register (00H) are set.

The Warning Status bit is written by the CC750.

WakeUp Wake Up Status

- one The CC750 has left Power Down or Sleep mode.
- zero No wake up.

Setting the Sleep bit in CPU Interface register (02H) to "1" will place the CC750 into Sleep mode. While in Sleep mode, the WakeUp bit is "0". The WakeUp bit will become "1" when bus activity is detected or when the CPU writes the Sleep bit to "0". The WakeUp bit will also be set to "1" after the CC750 comes out of Power Down mode.

The WakeUp interrupt is reset by reading the Status Register.

This bit is written by the CC750.

RxOK Receive Message Successfully

- one Since this bit was last reset to zero by the CPU, a message has been successfully received.
- zero Since this bit was last reset by the CPU, no message has been successfully received.

This bit is never reset by the CC750. A successfully received message may be any CAN bus transmission that is error-free, regardless of whether the CC750 has configured a Message Object to receive that particular message identifier.

The CC750 will set this bit, the CPU may clear it.

TxOK Transmit Message Successfully

- one Since this bit was last reset to zero by the CPU, a message has been successfully transmitted (error free and acknowledged by at least one other node).
- zero Since this bit was last reset by the CPU, no message has been successfully transmitted. This bit may be cleared by the CPU.

The CC750 will set this bit, but will not clear it.

LEC Last Error Code**0 No error****1 Stuff Error**

More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.

2 Form Error

The fixed format part of a received frame has the wrong format.

3 Acknowledgment Error (AckError)

The message transmitted by this device was not acknowledged by another node.

4 Bit 1 Error

During the transmission of a message (with the exception of the arbitration field), the CC750 wanted to send a recessive level (bit of logical value 1), but the monitored CAN bus value was dominant.

5 Bit 0 Error

During the transmission of a message (with the exception of the arbitration field), the CC750 wanted to send a dominant level (bit of logical value 0), but the monitored CAN bus value was recessive. During Bus Off recovery, this status is set each time a recessive bit is received (indicating the CAN bus is not stuck dominant).

6 CRC Error

The CRC checksum was incorrect in the message received. The CRC received for an incoming message does not match with the CRC value calculated by this device for the received data.

7 Unused

This field contains a code which indicates the type of the first error to occur in a frame on the CAN bus. If a message has been transferred (reception or transmission) without error, this field will be cleared to '0'.

The code '7' is unused and may be written by the CPU to check for updates.

4.3.1 Status Interrupts

If the SIE bit in the Control Register (00H) is set and the CC750 has updated the Status Register, the Interrupt Register (5FH) will contain a "1". The Status Register must be read if a Status Change Interrupt occurs. Reading the Status Register will clear the Status Change Interrupt.

A Status Change Interrupt will occur on every successful reception or transmission, regardless of the state of the RxOK and TxOK bits. Therefore, if TxOK is set and a subsequent transmission occurs, an interrupt will occur (if enabled) even though TxOK was previously equal to one.

There are two ways to implement receive and transmit interrupts. The difference between these two methods is one relies on the hardwired priority of the Message Objects and the other is suitable for polling.

The first and preferred method uses the TxIE and RxIE bits in the Control 0 register for each corresponding Message Object. Whenever a message is transmitted or received by this Message Object, the corresponding interrupt is serviced in accordance with its priority (if the IE bit of register 00H is set). This method uses the hardwired priority scheme of the CC750 which requires minimal CPU intervention.

The second method sets the SIE bit of the Control Register (00H) to "1" which will force an interrupt whenever successful message transmissions or receptions occur. The TxOK bit will be set when any of the Message Objects transmits a message. The RxOK bit will be set on any successfully received message. This may be any CAN bus transmission that is error-free, regardless of whether the CC750 has configured a Message Object to receive that particular message identifier. This method allows the user to more easily define the interrupt priority of each Message Object by polling the Message Objects following an SIE interrupt.

4.4 CPU Interface Register (02H)

7	6	5	4	3	2	1	0
OscSt	res	DMC	PwD	Sleep	res	res	res
r	r	rw	rw	rw	r	r	r

The default value of the CPU Interface Register after hardware reset is 20H. Reserved bits read as "0".

OscSt Oscillator Stopped

- one Oscillator has stopped.
- zero Oscillator is running.

If the CPU sets the PwD or Sleep Bit while a transmission or reception on the CAN Bus is pending, this data transfer will be resumed before the Oscillator stops working.

The CC750 indicates Power Down or Sleep Mode has been entered by setting this bit.

DMC Divide Memory Clock

- one The memory clock, MCLK, is equal to SCLK/2.
- zero The memory clock, MCLK, is equal to SCLK.

This bit is written by the CPU.

spec_functional_description.fm

K8/EIS - Klose-2989

0612/2.3 - 15.08.97

PwD Power Down Mode enable and **Sleep** Mode enable

PwD	Sleep	Function
0	0	Both Power Down and Sleep Modes are not active.
0	1	Sleep Mode is active. These bits are written by the CPU.
1	X	Power Down Mode is active.

Table 5: Function of Power Down and Sleep bits

Information about low Current Modes see chapter 3.6.

The Sleep bit or the Power Down bit is reset by the CPU.

4.5 Clocking Description

The clocking of the CC750 depends on the oscillator (XTAL), the system clock (SCLK) and the memory clock (MCLK). The SCLK and MCLK frequencies are determined by the external oscillator (XTAL) and the DMC bit in the CPU Interface Register (02H).

The SCLK is equal to XTAL/2 and controls the processing functions of the CC750 such as bit timing control and transceiver control logic or in other words, the CAN bus. So SCLK is used to calculate t_q which is referenced in the description of the Bit Timing registers, see chapter 4.12.

The MCLK may be equal to SCLK/2 or SCLK depending upon the value of the DMC bit in the CPU Interface Register (02H).

The maximum MCLK frequency for various oscillator frequencies is shown below:

f_{XTAL}	SCLK	MCLK	DMC bit
4 MHz	2 MHz	2 MHz	0
8 MHz	4 MHz	4 MHz	0
10 MHz	5 MHz	5 MHz	0
12 MHz	6 MHz	6 MHz	0
16 MHz	8 MHz	8 MHz	0
20 MHz	10 MHz	5 MHz	1

Table 6: Maximum MCLK frequency for various oscillator frequencies

Frequency of SCLK = $f_{XTAL}/2$

Frequency of MCLK = $f_{SCLK}/(1 + \text{DMC bit}) = f_{XTAL}/[2*(1 + \text{DMC bit})]$

4.6 Global Mask - Standard Register (06-07H)

Global Mask (06H)

7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
rw							

Global Mask (07H)

7	6	5	4	3	2	1	0
ID20	ID19	ID18	res				
rw	rw	rw	r				

The default value of the Global Mask standard after a hardware reset is unchanged. Reserved bits read as "1".

IDx Identifier bit at position X

- one must-match (incoming bit value must match to the corresponding bit in the Arbitration Register from a Message Object)
- zero don't care (accept a "0" or "1" for that bit position)

The Global Mask Standard register applies only to messages using the standard CAN Identifier and thereby to Message Objects with the Xtd bit set to "0". This feature, also called message acceptance filtering, allows the user to Globally Mask, or "don't care" any identifier bits of the incoming message. This mask is programmable to allow the user to develop an application specific masking strategy.

Note:

When a remote frame is sent, an CC750 receiver node will use the Global Mask Registers to determine whether the remote frame matches to any of its Message Objects. If the CC750 is programmed to transmit a message in response to a remote frame message identifier, the CC750 will transmit a message with the message identifier of the CC750 Message Object. The result is the remote message and the responding CC750 transmit message may have different message identifiers because some CC750 Global Mask Register bits are "don't care".

4.7 Global Mask - Extended Register (08-0BH)

Global Mask Extended (08H)

7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
rw							

spec_functional_description.fm

K3/EIS - Klasse-2989
061.2/2.3 - 15.08.97

Global Mask Extended (09H)

7	6	5	4	3	2	1	0
ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
rw							

Global Mask Extended (0AH)

7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
rw	rw	rw	rw	rw	rw	rw	rw

Global Mask Extended (0BH)

7	6	5	4	3	2	1	0
ID4	ID3	ID2	ID1	ID0	res		
rw	rw	rw	rw	rw	r		

The default value of the Global Mask extended after a hardware reset is unchanged.
Reserved bits read as "0".

IDx Identifier bit at position x

- one must-match (incoming bit value must match to the corresponding bit in the Arbitration Register from a Message Object)
- zero don't care (accept a "0" or "1" for that bit position)

The Global Mask extended register applies only to messages using the extended CAN identifier and thereby to Message Objects with the Xtd bit set to "1". This feature allows the user to Globally Mask, or "don't care", any identifier bits of the incoming message. This mask is programmable to allow the user the develop an application specific masking strategy.

Note:

When a remote frame is sent, an CC750 receiver node will use its Global Mask Registers to determine whether the remote frame matches to any of its Message Objects. If the CC750 is programmed to transmit a message in response to a remote frame message identifier, the CC750 will transmit a message with the message identifier of the CC750 Message Object. The result is the remote message and the responding CC750 transmit message may have different message identifiers because some CC750 Global Mask Register bits are "don't care".

4.8 Acceptance Filtering Implications

The CC750 implements two acceptance masks which allow Message Objects to receive messages with a range of message identifiers (IDs) instead of just a single message ID. This provides the application the flexibility to receive a wide assortment of messages from the bus.

The CC750 observes all messages on the CAN bus and stores any message that matches a message's ID programmed into an "active" Message Object. It is possible to define which message ID bits must identically match those programmed in the Message Objects to store the message. Therefore, ID bits of incoming messages are either "must-match" or "don't-care". By defining bits to be "don't-care", Message Objects will receive multiple message IDs.

4.9 Message 15 Mask Register (0C-0FH)

Message 15 Mask Register (0CH)

7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
rw							

Message 15 Mask Register (0DH)

7	6	5	4	3	2	1	0
ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
rw							

Message 15 Mask Register (0EH)

7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
rw	rw	rw	rw	rw	rw	rw	rw

Message 15 Mask Register (0FH)

7	6	5	4	3	2	1	0
ID4	ID3	ID2	ID1	ID0	MDir	MXtd	res
rw	rw	rw	rw	rw	rw	rw	r

The default value of the Message 15 Mask Register after a hardware reset is unchanged. Reserved bit read as "0".

IDx Identifier bit at position X

- one must-match (incoming bit value must match to the corresponding bit in the Arbitration Register from the Message Object 15)
- zero don't care (accept a "0" or "1" for that bit position)

spec_functional_description.fm

K8/EIS - Klose-2989

061.2/2.3 - 15.08.97

MDir Mask Direction Bit

- one must-match (incoming Direction bit value must match to the corresponding bit in the Arbitration Register from the Message Object 15)
- zero don't care (Message Object 15 accepts Remote and Data Frames)

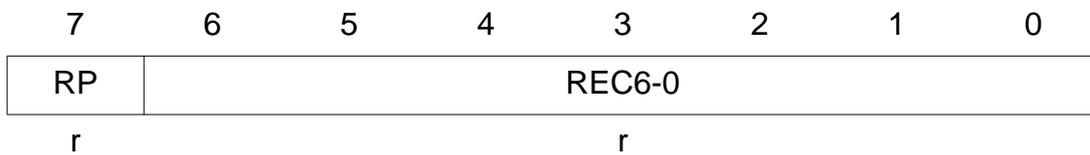
MXtd Mask Extended Bit

- one must-match (incoming Xtd bit value must match to the corresponding bit in the Arbitration Register from the Message Object 15)
- zero don't care (Message Object 15 accepts Standard and Extended Identifier Frames)

The Message 15 Mask Register is a programmable local mask. This feature allows the user to locally mask, or “don't care”, any identifier bits of the incoming message for Message Object 15. Incoming messages are first checked for an acceptance match in Message Objects 1- 14 before passing through to Message Object 15. Consequently, the Global Mask and the Local Mask apply to messages received in Message Object 15 in that way, that Message 15 Mask is “ANDed” with the Global Mask. This means that any bit defined as “don't-care” in the Global Mask will automatically be a “don't care” bit for message 15.

For the receive-only Message Object 15, it is also possible to mask the bits Dir and Xtd, allows the reception of Standard and Extended as well as Data and Remote Frames in this Message Object.

4.10 Receive Error Counter (1FH)



The default value of the Receive Error Counter after a hardware reset is 00H.

RP Receive Error Passive

- one The Receive Error Counter has reached the *error passive* level as defined in the CAN Specification.
- zero The Receive Error Counter is below the *error passive* level.

REC6-0 Receive Error Counter

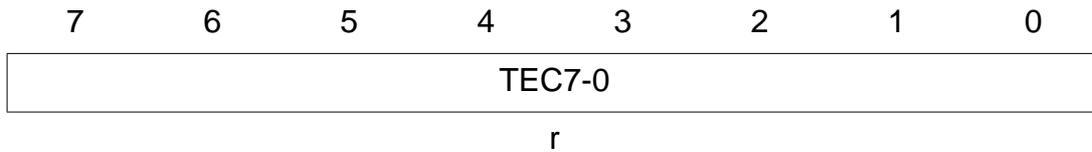
Actual state of the Receive Error Counter. Values between 0 and 127.

spec_functional_description.fm

K0/EIS - Klasse-2989

0612/2.3 - 15.08.97

4.11 Transmit Error Counter (2FH)



The default value of the Transmit Error Counter after a hardware reset is 00H.

TEC7-0 Transmit Error Counter

Actual state of the Transmit Error Counter. Values between 0 and 255.

4.12 Bit Timing Registers

4.12.1 Bit Timing Overview

A CAN message consists of a series of bits that are transmitted in consecutive bit times. A bit time accounts for propagation delay of the bit, CAN chip input and output delay, and synchronization tolerances. This section describes components of a bit time from the perspective of the CAN Specification and the CC750.

According to the CAN Specification, the nominal bit time is composed of four time segments. These time segments are separate and non-overlapping as shown below:

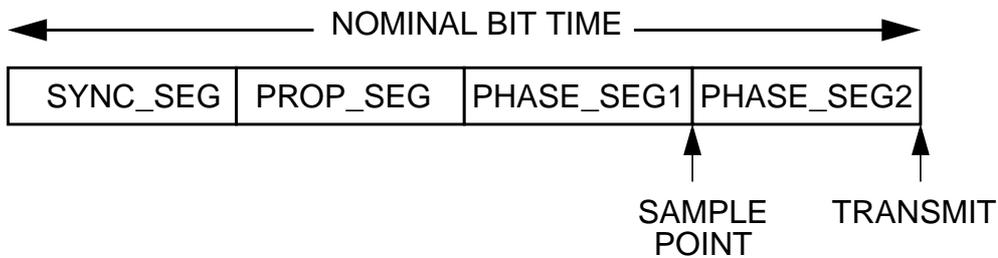


Figure 3: Time Segments of Bit Time

SYNC_SEG Synchronisation Segment

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment.

PROP_SEG Propagation Time Segment

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay and the output driver delay.

spec_functional_description.fm

K01/EIS - Klose-2989
0612/2.3 - 15.08.97

NOTE:

The factor of two accounts arbitration which requires nodes consecutively to synchronize to different transmitters.

PHASE_SEG1, PHASE_SEG2: Phase Buffer Segment1,2

These segments are used to compensate for edge phase errors and can be lengthened or shortened by resynchronization.

SAMPLE POINT:

The sample point is the point of time at which the bus level is read and interpreted as the value of that respective bit. Its location is at the end of PHASE_SEG1.

4.12.2 CC750 Bit Timing Definitions

In this application, the Synchronisation Segment is represented by t_{Sync} , the Phase Buffer Segment2 is represented by t_{TSeg2} , while t_{TSeg1} is the summation of the Propagation Time Segment and the Phase Buffer Segment1.

The preceding figure represents a bit time from the perspective of the CC750. A bit time is subdivided into time quanta. One time quantum is derived from the System Clock (SCLK) and the Baud Rate Prescaler (BRP). Each segment is a multiple of the Time Quantum t_q . The length of these segments is programmable, with the exception of the Synchronisation Segment, which is always 1 t_q long.

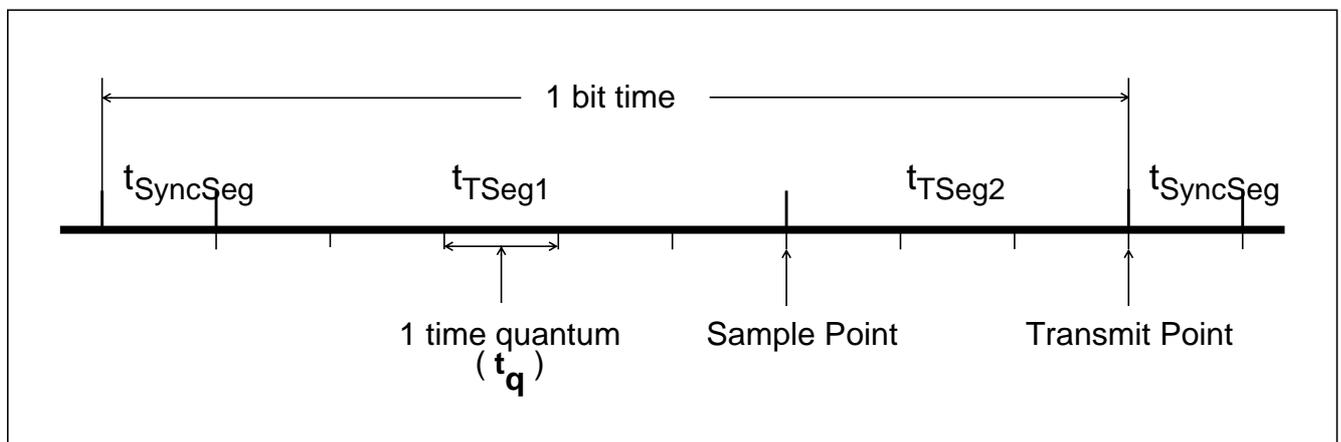


Figure 4: Bit Timing

spec_functional_description.fm

K0/EIS - Klose-2989

061.2/2.3 - 15.08.97

4.12.3 CC750 Bit Time Segments

The following are relationships of the CC750 bit timing:

$$\text{bit time} = t_{\text{SyncSeg}} + t_{\text{TSeg1}} + t_{\text{TSeg2}} \quad (\text{see preceding figure})$$

$$t_{\text{SyncSeg}} = 1 \cdot t_q$$

$$t_{\text{TSeg1}} = (\mathbf{TSeg1} + 1) \cdot t_q$$

$$t_{\text{TSeg2}} = (\mathbf{TSeg2} + 1) \cdot t_q$$

$$t_q = (\mathbf{BRP} + 1) \cdot t_{\text{SCLK}}$$

$$f_{\text{Bit}} = f_{\text{XTAL}} / [2 \cdot (\mathbf{BRP} + 1) \cdot (\mathbf{TSeg1} + \mathbf{TSeg2} + 3)]$$

The variables **TSeg1**, **TSeg2**, and Baud Rate Prescaler **BRP** are the programmed numerical values from the Bit Timing Registers. The actual interpretation by the hardware of these values is such that **one more** than the values programmed here is used, as shown in the brackets of the equations.

4.12.4 Calculation of the Bit Time

The programming of the bit time has to regard the CAN Specification Rev. 2.0 and depends on the desired baudrate, the CC750 oscillator frequency f_{XTAL} and on the external physical delay times of the bus driver, of the bus line and of the input comparator. The delay times are summarised in the Propagation Time Segment, its actual value t_{Prop} is:

t_{Prop} is two times the maximum of the sum of physical bus delay, the input comparator delay, and the output driver delay rounded up to the nearest multiple of t_q .

To fulfil the requirements of the CAN specification, the following conditions must be met :

$$t_{\text{TSeg2}} \geq 1 \cdot t_q \quad = \text{Information Processing Time}$$

$$t_{\text{TSeg2}} \geq t_{\text{SJW}}$$

$$t_{\text{TSeg1}} \geq 2 \cdot t_q$$

$$t_{\text{TSeg1}} \geq t_{\text{SJW}} + t_{\text{Prop}}$$

$$t_{\text{TSeg1}} \geq t_{\text{SJW}} + t_{\text{Prop}} + 2t_q \quad \text{for 3 Sample Mode (bit Spl="1" in register 4FH)}$$

Note:

In order to achieve correct operation according to the CAN protocol the total bit time should be at least $8 t_q$, i.e. $\mathbf{TSeg1} + \mathbf{TSeg2} \geq 5$ (as programmed in the Bit Timing Register 1).

To operate with a baudrate of 1 MBit/s, the frequency of **SCLK** has to be at least 8 MHz, in consequence f_{XTAL} has to be at least 16 MHz.

The maximum tolerance **df** for **XTAL** depends on the Phase Buffer Segment1 (PB1), the Phase Buffer Segment2 (PB2), and the Resynchronisation Jump Width (SJW):

$$df \leq \frac{\min(PB1, PB2)}{2 \times (13 \times \text{bit time} - PB2)}$$

AND

$$df \leq \frac{SJW}{20 \times \text{bit time}}$$

$$(PB1 = t_{TSeg1} - t_{Prop} ; PB2 = t_{TSeg2})$$

4.12.5 Example for Bit Timing at high Baudrate

In this example, the frequency of **XTAL** is 20 MHz, **BRP** is 0, the bitrate is 1 MBit/s.

t_q	100 ns	=	$t_{SCLK} = t_{XTAL} \cdot 2$
delay of bus driver	50 ns		
delay of receiver circuit	30 ns		
delay of bus line (40m)	220 ns	=	520 ns, round up
t_{Prop}	600 ns	=	$6 \cdot t_q$
t_{SJW}	100 ns	=	$1 \cdot t_q$
t_{TSeg1}	700 ns	=	$t_{Prop} + t_{SJW}$
t_{TSeg2}	200 ns	=	Information Processing Time + $1 \cdot t_q$
$t_{Sync-Seg}$	100 ns	=	$1 \cdot t_q$ (fix)
bit time	1000 ns	=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
maximal oscillator tolerance 0.39 %		=	$\frac{\min(PB1, PB2)}{2 \times (13 \times \text{bit time} - PB2)}$
		=	$\frac{0.1 \mu s}{2 \times (13 \times 1 \mu s - 0.2 \mu s)}$

In this example, the Bit Timing Registers must be programmed with the following values:

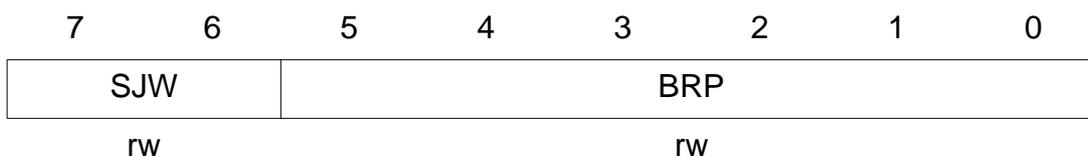
Bit Timing Register 0 (3FH): 00H

Bit Timing Register 1 (4FH): 16H

4.12.6 Bit Timing Registers 0 + 1 (3FH + 4FH)

Bit Timing Registers are used to define the CAN bus frequency, the sample point within a bit time, and the mode of synchronization.

Bit Timing Register 0 (3FH)



The default value of the Bit Timing Register 0 after a hardware reset is 00H.

spec_functional_description.fm K8/EIS - Klasse-2989 0612/2.3 - 15.08.97

SJW (Re) Synchronization Jump Width

The valid programmed values are 0-3. The **SJW** defines the maximum number of time quanta a bit time may be shortened or lengthened by one resynchronization.

The actual interpretation of this value by the hardware is to use one more than the programmed value.

BRP Baud Rate Prescaler

The valid programmed values are 0-63. The baud rate prescaler programs the length of one time quantum as follows: $t_q = t_{SCLK} \cdot (BRP + 1)$ where t_{SCLK} is the period of the system clock (SCLK).

Bit Timing Register 1 (4FH)

7	6	5	4	3	2	1	0
Spl	TSeg2			TSeg1			
rw	rw			rw			

The default value of the Bit Timing Register 1 after a hardware reset is 00H.

Spl Sampling Mode

- one The CAN bus is sampled three times per bit time for determining the valid bit value using majority logic.
- zero Bus is sampled once, may result in faster bit transmissions rates.

Sampling mode = "0" may result in faster bit transmissions rates, while sampling mode = "1" is more immune to noise spikes on the CAN bus.

TSeg2 Time Segment 2

The valid programmed values are 0-7. TSeg2 is the time segment after the sample point.

The actual interpretation of this value by the hardware is one more than the value programmed by the user.

TSeg1 Time Segment 1

The valid programmed values are 1-15. TSeg1 is the time segment before the sample point.

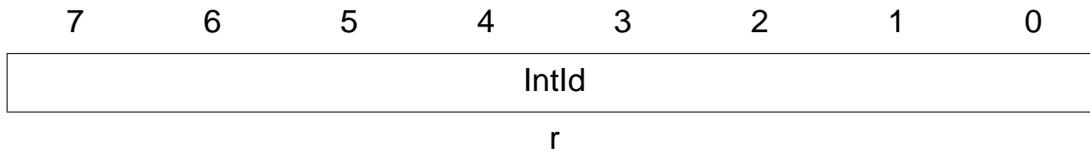
The actual interpretation of this value by the hardware is one more than the value programmed by the user.

spec_functional_description.fm

K8/EIS - Klose-2989

0612/2.3 - 15.08.97

4.13 Interrupt Register (5FH)



The default value of the Interrupt Register after a hardware reset is 00H.

IntId Interrupt Identifier

The Interrupt Register is a read-only register. The value in this register indicates the source of the interrupt. When no interrupt is pending, this register holds the value "0". If the SIE bit in the Control Register (00H) is set and the CC750 has updated the Status Register, the Interrupt Register will contain a "1". This indicates an interrupt is pending due to a change in the Status Register. The value 2 + Message Object Number indicates the IntPnd bit in the corresponding Message Object is set. There is an exception in that Message Object 15 will have the value 2, giving Message Object 15 the highest priority of all Message Objects.

Interrupt Source	Value
none	00H
Status Register	01H
Message Object 15	02H
Message Object 1	03H
Message Object 2	04H
Message Object 3	05H
Message Object 4	06H
Message Object 5	07H
Message Object 6	08H
Message Object 7	09H
Message Object 8	0AH
Message Object 9	0BH
Message Object 10	0CH
Message Object 11	0DH
Message Object 12	0EH
Message Object 13	0FH
Message Object 14	10H

Table 7: Interrupt Register values with corresponding Interrupt Sources

spec_functional_description.fm K0/EIS - Klose-2989 061.2/2.3 - 15.08.97

For example, a message is received by Message Object 13 with the IE (Control Register) and RxIE (Message Object 13 Control 0 Register) bits set. The interrupt pin will be pulled low and the value 15 (0FH) will be placed in the Interrupt Register.

If the value of the Interrupt Register equals "1", then the Status Register at location 01H must be read to update this Interrupt Register. The Status Change Interrupt has a higher priority than interrupts from the Message Objects. Register 5FH is automatically set to "0" or to the lowest value corresponding to a Message Object with IntPnd set. When the value of this register is two or more, the IntPnd bit of the corresponding Message Object Control Register is set.

The CC750 will respond to each status change event independently and will not bundle interrupt events in a single interrupt signal. However, if two status change events occur before the first is acknowledged by the CPU, the next event will not generate a separate interrupt output. Therefore, when servicing Status Change Interrupts, the user code should check all useful status bits upon each Status Change Interrupt.

After resetting the IntPnd bit in the Control 0 Register of individual Message Objects, the minimum delay of the CC750 resetting the interrupt pin and updating the Interrupt Register (5FH) is 3 MCLK cycles and a maximum of 14 MCLK cycles (after the CPU write operation to this register is finished). When a Status Change Interrupt occurs, reading the Status Register (01H) will reset the interrupt pin in a maximum of 4 MCLK cycles + 145 ns. Clearing the IntPnd bit of the Message Object will deactivate the INT# pin.

4.14 Serial Reset Address (FFH)



The serial reset address is used to synchronize accesses between the CC750 and the CPU when the CPU cannot provide a chip select. The CPU must write a string of 16 "FFH" bytes to achieve synchronization.

4.15 CC750 Message Objects (MO)

4.15.1 Message Object Structure

The Message Object is the means of communication between the host microcontoller and the CAN controller in the CC750. Message Objects are configured to transmit or receive messages.

There are 15 Message Objects located at fixed addresses in the CC750. Each Message Object starts at a base address that is a multiple of 16 bytes and uses 15 consecutive bytes. For example, Message Object 1 starts at address 10H and ends at address 1EH. The remaining byte in the 16 byte field is used for other CC750 functions. In the above example the byte at address 1FH is used for the Receive Error Counter.

Message Object 15 is a receive-only Message Object that uses a local mask called the

spec_functional_description.fm K3/EIS - Klose-2989 061.2/2.3 - 15.08.97

Message 15 Mask Register. This mask allows a large number of infrequent messages to be received by the CC750. In addition, Message Object 15 is buffered to allow the CPU more time to receive messages.

Address	Function
Base Address +0	Control 0
+1	Control 1
+2	Arbitration 0
+3	Arbitration 1
+4	Arbitration 2
+5	Arbitration 3
+6	Configuration
+7	Data 0
+8	Data 1
+9	Data 2
+10	Data 3
+11	Data 4
+12	Data 5
+13	Data 6
+14	Data 7

Table 8: Message Object Structure

4.15.2 Control 0 + 1 Registers

Control 0 Register (Base Address + 0)

7	6	5	4	3	2	1	0
MsgVal		TxIE		RxIE		IntPnd	
rw		rw		rw		rw	

spec_functional_description.fm

K8/EIS - Klasse-2989

061.2/2.3 - 15.08.97

Control 1 Register (Base Address + 1)

7	6	5	4	3	2	1	0
RmtPnd		TxRqst		MsgLst/CPUUpd		NewDat	
rw		rw		rw		rw	

The default values of the Control 0 and Control 1 registers after a hardware reset are unchanged.

Each bit in the Control 0 and Control 1 bytes occurs twice; once in true form and once in complement form. This bit representation makes testing and setting these bits as efficient as possible. The advantage of this bit representation is to allow write access to single bits of the byte, leaving the other bits unchanged without the need to perform a read/modify/write cycle.

For example, a HC08 CPU would set the TxRqst bit of the Control 1 byte with the following instructions:

```
LDA    #$EF           ;load 11101111 into accumulator register
STA    CTRL1         ;write 11101111 to Control 1,
                    ;setting TxRqst
```

The representation of these two bits is described below:

Direction	MSB	LSB	Meaning
Write	0	0	not allowed (indeterminate)
	0	1	reset
	1	0	set
	1	1	unchanged
Read	0	1	reset
	1	0	set

Table 9: Representation of bit pairs in Control Registers

MsgVal Message Valid

- one The Message Object is valid.
- zero The Message Object is invalid.

The MsgVal bit is an individual halt bit for each Message Object. While this bit is reset the CC750 will not access this Message Object for any reason. This bit may be reset at any time if the message is no longer required, or if the identifier is being changed. If a message identifier is changed, the Message Object must be made invalid first, and it is not necessary

spec_functional_description.fm

0612/2.3 - 15.08.97 K8/EIS - Klose-2989

to reset the chip following this modification.

The CPU must reset the MsgVal bit of all unused messages during initialization of the CC750 before the Init bit of the Control Register (00H) is reset. The contents of Message Objects may be reconfigured dynamically during operation and the MsgVal bit assists reconfiguration in many cases.

The MsgVal bit must be set to indicate the Message Object is configured and is ready for communication transactions.

This bit is written by the CPU.

IMPORTANT NOTE:

Two or more Message Objects must not have the same message identifier and also be valid at the same time!

If more than one CC750 transmit Message Object has the same message ID, a successful transmission of the higher numbered Message Objects will not be recognized by the CC750. The lower numbered Message Object will be falsely identified as the transmit Message Object and its transmit request bit will be reset. The actual transmit Message Object will re-transmit without end because its transmit request bit will not be reset.

This could result in a catastrophic condition since the higher numbered Message Object may dominate the CAN bus by resending its message without end.

To avoid this condition, applications should require all transmit Message Objects to use message IDs that are unique. If this is not possible, the application should disable lower numbered Message Objects with similar message IDs until the higher numbered Message Object has transmitted successfully.

TxIE Transmit Interrupt Enable

one An interrupt will be generated after a successful transmission of a frame.

zero No interrupt will be generated after a successful transmission of a frame.

The Transmit Interrupt Enable bit enables the CC750 to initiate an interrupt after the successful transmission by the corresponding Message Object.

This bit is written by the CPU.

RxIE Receive Interrupt Enable

one An interrupt will be generated after a successful reception of a frame.

zero No interrupt will be generated after a successful reception of a frame.

This bit enables the CC750 to initiate an interrupt after the successful reception by the corresponding Message Object.

This bit is written by the CPU.

NOTE:

In order for TxIE or RxIE to generate an interrupt, IE in the Control Register must be set.

IntPnd Interrupt Pending

- one This Message Object has generated an interrupt.
- zero No interrupt was generated by this Message Object since the last time the CPU cleared this bit.

This bit is set by the CC750 following a successful transmission or reception as controlled by the RxIE and TxIE bits.

The CPU must clear this bit when servicing the interrupt.

RmtPnd Remote Request Pending

- one The transmission of this Message Object has been requested by a remote node and is not yet done.
- zero There is no waiting remote request for the Message Object.

This bit is only used by Message Objects with direction = transmit. This bit is set by the CC750 after receiving a remote frame which matches its message identifier, taking into account the Global Mask Register. The corresponding Message Object will respond by transmitting a message, if the CPUUpd bit = zero. Following this transmission, the CC750 will clear the RmtPnd bit. In other words, when this bit is set it indicates a remote node has requested data and this request is still pending because the data has not yet been transmitted.

NOTE:

Setting RmtPnd will not cause a remote frame to be transmitted. The TxRqst bit is used to send a remote frame from a receive Message Object.

TxRqst Transmit Request

- one The transmission of this Message Object has been requested and has not been completed.
- zero This Message Object is not waiting to be transmitted.

This bit is set by the CPU to indicate the Message Object data should be transmitted. Setting TxRqst will send a data frame for a transmit Message Object and a remote frame for a receive Message Object.

If direction = receive a remote frame is sent to request a remote node to send the corresponding data.

TxRqst is also set by the CC750 (at the same time as RmtPnd in Message Objects whose direction = transmit) when it receives a remote frame from another node requesting this data. This bit is cleared by the CC750 along with RmtPnd when the message has been successfully transmitted, if the NewDat bit has not been set.

MsgLst Message Lost

Only valid for Message Objects with direction = receive.

- one The CC750 has stored a new message in this Message Object when NewDat was still set.
- zero No message was lost since the last time this bit was reset by the CPU.

This bit is used to signal that the CC750 stored a new message into this Message Object when the NewDat bit was still set. Therefore, this bit is set if the CPU did not process the contents of this Message Object since the last time the CC750 set the NewDat bit; this indicates the last message received by this Message Object overwrote the previous message which was not read and is lost.

This definition is only valid for Message Objects with direction = receive. For Message Objects with direction = transmit, the definition is replaced by CPUUpd.

CPUUpd CPU Updating

Only valid for Message Objects with direction = transmit.

- one This Message Object may not be transmitted.
- zero This Message Object may be transmitted, if direction = transmit.

The CPU sets this bit to indicate it is updating the data contents of the Message Object and the message should not be transmitted until this bit has been reset. The CPU indicates message updating has been completed by resetting this bit (it is not necessary to use the MsgVal bit to update the Message Object's data contents).

The purpose of this bit is to prevent a remote frame from triggering a transmission of invalid data.

NewDat New Data

- one The CC750 or CPU has written new data into the data section of this Message Object.
- zero No new data has been written into the data section of this Message Object since the last time this bit was cleared by the CPU.

This bit has different meanings for receive and transmit Message Objects.

4.15.3 Handling of Message Objects

For Message Objects with **direction = receive**, the CC750 sets the NewDat bit whenever new data has been written into the Message Object.

When the received data is written into Message Objects 1-14, the unused data bytes will be overwritten with non-specified values.

The CPU should clear the NewDat bit before reading the received data and then check if the bit remained cleared when all bytes have been read. If the NewDat bit is set, the CPU should re-read the received data to prevent working with a combination of old and new data. See flow diagram in chapter 5.4.

When the received Data is matched to Message Object 15, new data is written into the shadow register. The foreground register is not over-written with new data. For Message

Object 15 messages, the data should be read first, the IntPnd reset, and then the NewDat and RmtPnd bits are reset. Resetting the NewDat and RmtPnd bits before resetting the IntPnd bit will result the interrupt line remaining active. See flow diagram in chapter 5.5.

For Message Objects with **direction = transmit**, the CPU should set the NewDat bit to indicate it has updated the message contents. This is done at the same time the CPU clears the CPUUpd bit. This will ensure that if the message is actually being transmitted during the time the message was being updated by the CPU, the CC750 will not reset the TxRqst bit. In this way, the TxRqst bit is reset only after the actual data has been transferred. See flow diagram in chapter 5.3.

Each bit in the Control 0 and Control 1 registers may be set and reset by the CPU as required.

Conditions required to trigger a transmission:

Flags	Register	Remote Frame	Data Frame
Init	Control	0	
MsgVal	MO-Control 0	1	
TxRqst	MO-Control 1	1	
MsgLst/CPUUpd	MO-Control 1	0	
NewDat	MO-Control 1	don't care	should be set
Dir	MO-Configuration	0	1

Table 10: Bit combinations to start transmissions

NOTES:

To program a transfer request, the Control Register 1 of the Message Object should have the TxRqst and NewDat bits set to "1". Therefore, this register may be written with the value 066H to initiate a transmission.

A remote frame may be received, an interrupt flag set, and no data frame transmitted in response by configuring a Message Object in the following manner. Set the CPUUpd and RxIE bits in the Message Object Control Register to "1". Set the Dir bit in the Message Configuration Register to "1". A remote frame will be received by this Message Object, the IntPnd bit will be set to "1" and no transmit message will be sent.

Message Object Priority

If multiple Message Objects are waiting to transmit, the CC750 will first transmit the message from the lowest numbered Message Object, regardless of message identifier priority.

If two Message Objects are capable of receiving the same message (possibly due to message filtering strategies), the message will be received by the lowest numbered Message Object. For example, if all acceptance mask bits were set as "don't care", Message Object 1 will receive all messages.

spec_functional_description.fm

K8/EIS - Klose-2989

0612/2.3 - 15.08.97

4.15.4 Arbitration 0, 1, 2, 3 Registers

Arbitration 0 (Base Address + 2):

7	6	5	4	3	2	1	0
ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
rw							

Arbitration 1 (Base Address + 3):

7	6	5	4	3	2	1	0
ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
rw							

Arbitration 2 (Base Address + 4):

7	6	5	4	3	2	1	0
ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5
rw	rw	rw	rw	rw	rw	rw	rw

Arbitration 3 (Base Address + 5):

7	6	5	4	3	2	1	0
ID4	ID3	ID2	ID1	ID0	res		
rw	rw	rw	rw	rw	r		

The default value of the Arbitration Register after a hardware reset is unchanged. Reserved bits read as "0".

ID0-ID28 Message Identifier

ID0-ID28 is the identifier for an extended frame.

ID18-ID28 is the identifier for a standard frame.

NOTE:

When the CC750 receives a message, the entire message identifier, the Data Length Code (DLC), the Direction bit (Dir) and the Extended Identifier bit (Xtd) are stored (additionally to the data section) into the corresponding Message Object.

spec_functional_description.fm

K8/EIS - Klasse-2989

061.2/2.3 - 15.08.97

4.15.5 Configuration Register

Configuration (Base Address + 6):

7	6	5	4	3	2	1	0
DLC				Dir	Xtd	res	
rw				rw	rw	r	

The default value of the Message Configuration Register after a hardware reset is unchanged.

Reserved bits read as "0".

DLC Data Length Code

The valid programmed values are 0-8. The Data Length Code of a Message Object is written with the value corresponding to the data length.

Dir Direction

- one Direction = transmit. When TxRqst is set, the Message Object will be transmitted.
- zero Direction = receive. When TxRqst is set, a remote frame will be transmitted. When a message is received with a matching identifier, the message will be stored in the Message Object.

Xtd Extended Identifier

- one This Message Object will use an extended 29 bit message identifier.
- zero This Message Object will use a standard 11 bit message identifier.

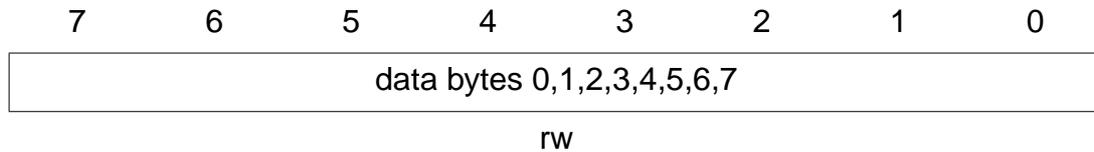
If the Message Configuration Register Xtd bit is "0" to specify a standard frame, the CC750 will reset the extended bits in the Arbitration Registers (arbitration bits 0-17) to "0" whenever a data frame is stored in this Message Object.

An extended receive Message Object (Xtd = "1") will not receive standard messages (except if this bit is masked out, which is possible for Message Object 15).

If a Message Object receives a data frame from the CAN bus, the entire message identifier, the Data Length Code (DLC), the Direction bit (Dir) and the Extended Identifier bit (Xtd) are stored (additionally to the data section) into this Message Object. Therefore, if acceptance filtering (masking registers) is used, the masked-off "don't care" bits will be rewritten corresponding to the message ID of the incoming message.

4.15.6 Data Bytes

Data Bytes (Base Address + 7 ... Base Address + 14):



The default value of the data bytes 0-7 after a hardware reset is unchanged.

When the CC750 writes new data into the message buffer, only the data bytes defined by the DLC are valid. Unused data bytes will be overwritten by non-specified values.

4.16 Special Treatment of Message Object 15

Message Object 15 is a receive-only Message Object with a programmable local mask called the Message 15 Mask Register. Since this Message Object is a receive only Message Object, the TxRqst bit and the TxIE have been hardwired inactive and the CPUUpd bit has no meaning.

The incoming messages for Message Object 15 will be written into a two-message alternating buffers to avoid the loss of a message if a second message is received before the CPU has read the first message. Once Message Object 15 is read, it is necessary to reset the NewDat and the RmtPnd bits to allow the CPU to read the shadow message buffer which will receive the next message or which may already contain a new message.

If two messages have been received by Message Object 15, the first will be accessible to the CPU. The alternate buffer will be overwritten if a subsequent (third receive) message is received. Once again, after reading message 15, the user program should reset the IntPnd bit followed by a reset of the NewDat and RmtPnd bits in the Message Object Control Registers.

The Xtd bit in the Message Configuration Register determines whether a standard or an extended frame will be received by this Message Object. This bit could be masked out, see chapter 4.9.

5. FLOW DIAGRAMS

The following flowcharts describe the operation of the CC750 and suggested flows for the host-CPU.

5.1 CC750 handling of Message Objects 1-14 (Transmit)

These are the operations the CC750 executes to transmit Message Objects. This diagram is useful to identify when the CC750 sets bits in the Control Registers.

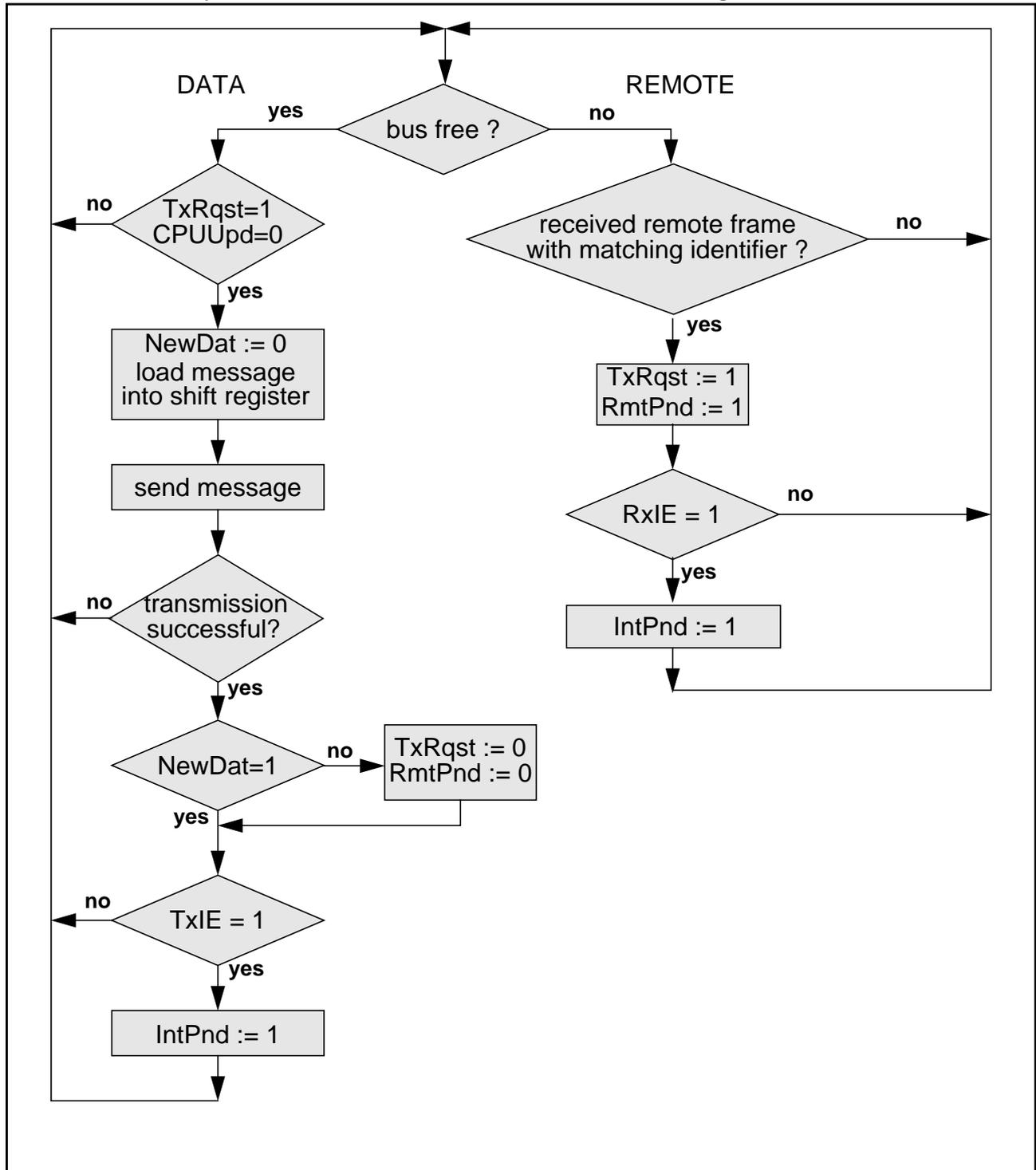


Figure 5: CC750 handling of Message Objects 1-14 (Transmit)

spec_flow_diagrams.fm

K8/EIS - Klose-2989

061/2/2.3 - 15.08.97

5.2 CC750 handling of Message Objects 1-14 (Receive)

These are the operations the CC750 executes to receive Message Objects. This diagram is useful to identify when the CC750 sets bits in the Control Registers.

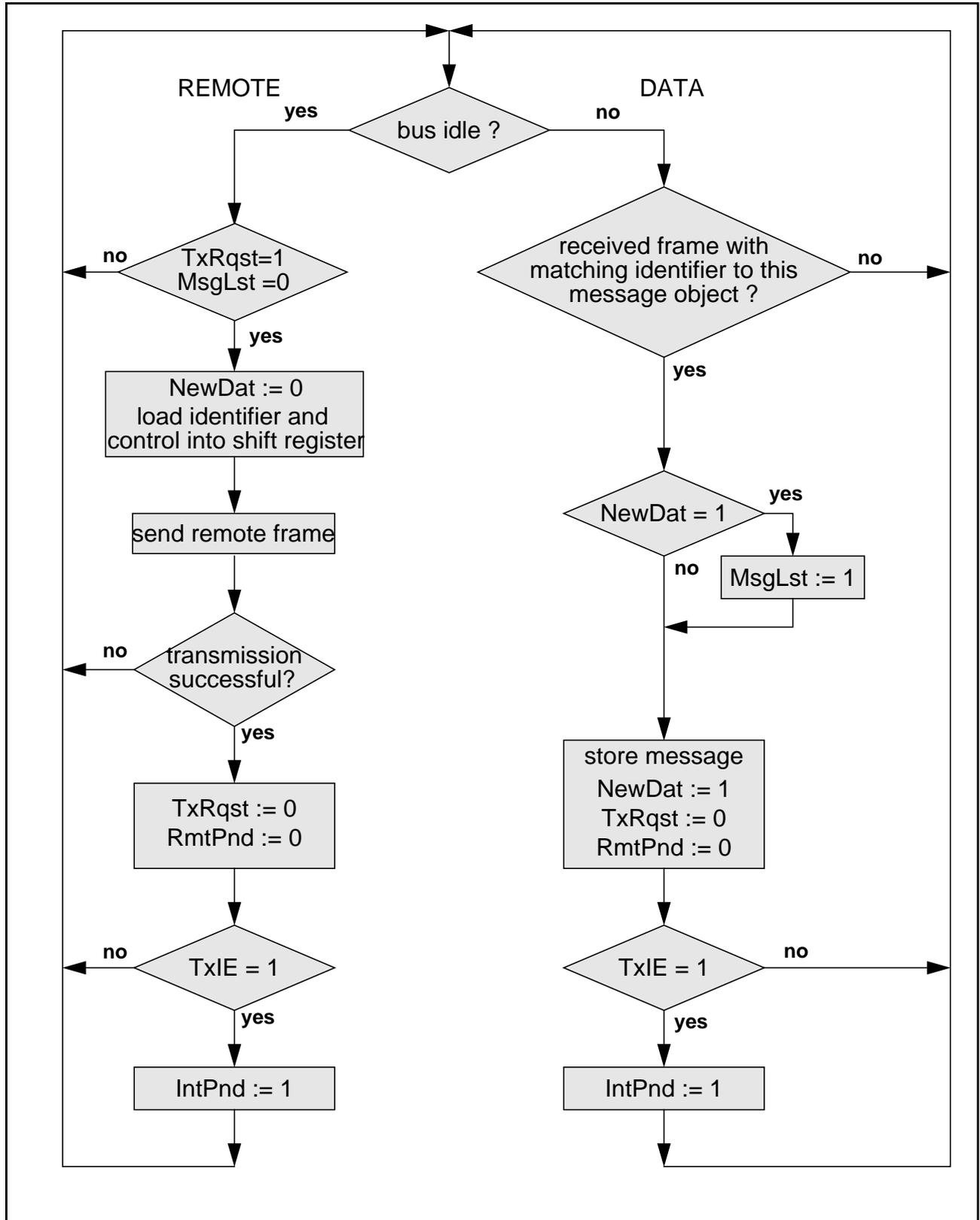


Figure 6: CC750 handling of Message Objects 1-14 (Direction = Receive)

spec_flow_diagrams.fm
 K8/EIS - Klose-2989
 061/2/2.3 - 15.08.97

5.3 CPU Handling of Message Objects 1-14 (Transmit)

These are the operations the host-CPU executes to transmit Message Objects 1-14.

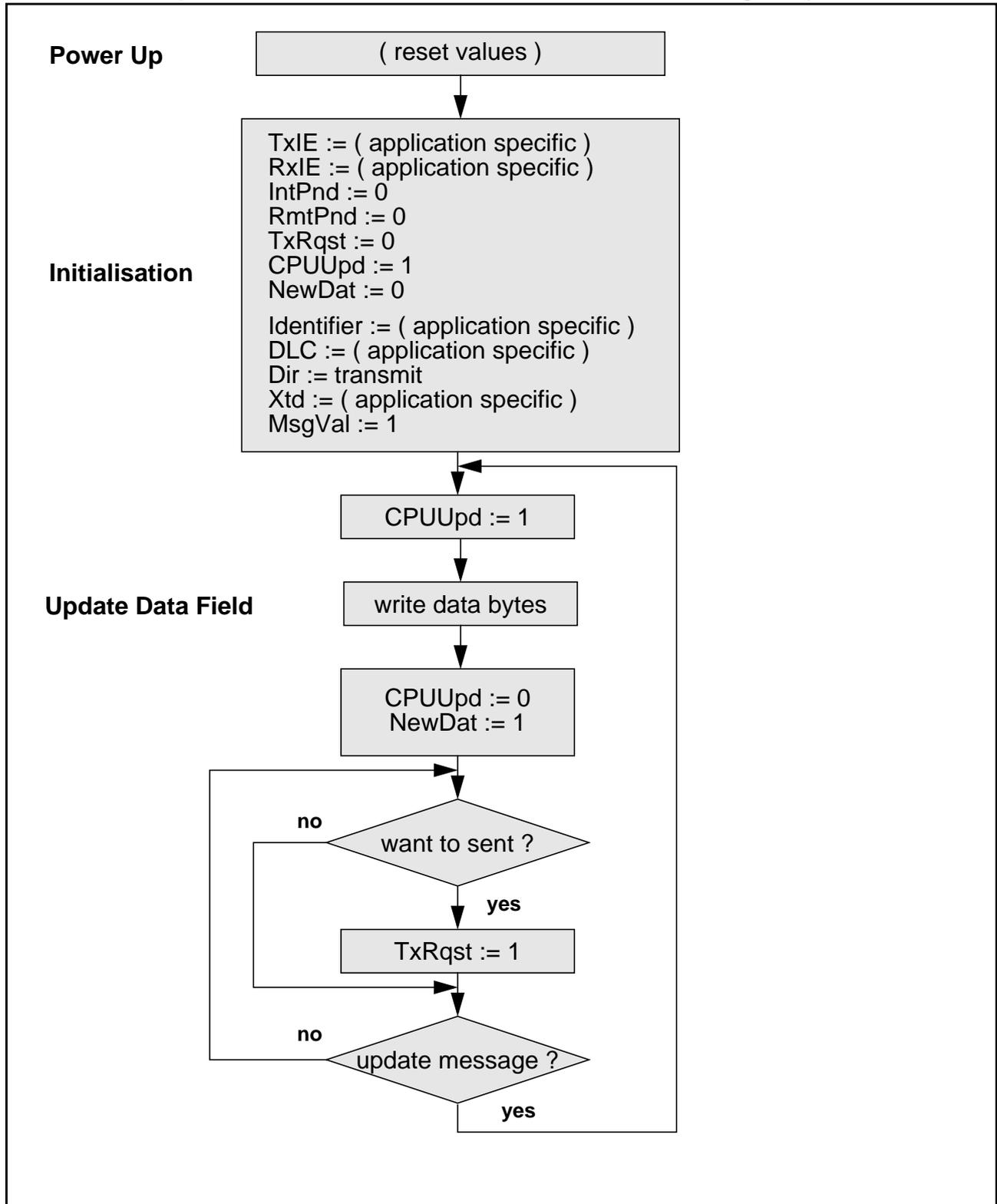


Table 11: CPU Handling of Message Objects 1-14 (Transmit)

spec_flow_diagrams.fm

K8/EIS - Klose-2989

061.2/2.3 - 15.08.97

5.4 CPU Handling of Message Objects 1-14 (Receive)

These are the operations the host-CPU executes to receive Message Objects 1-14.

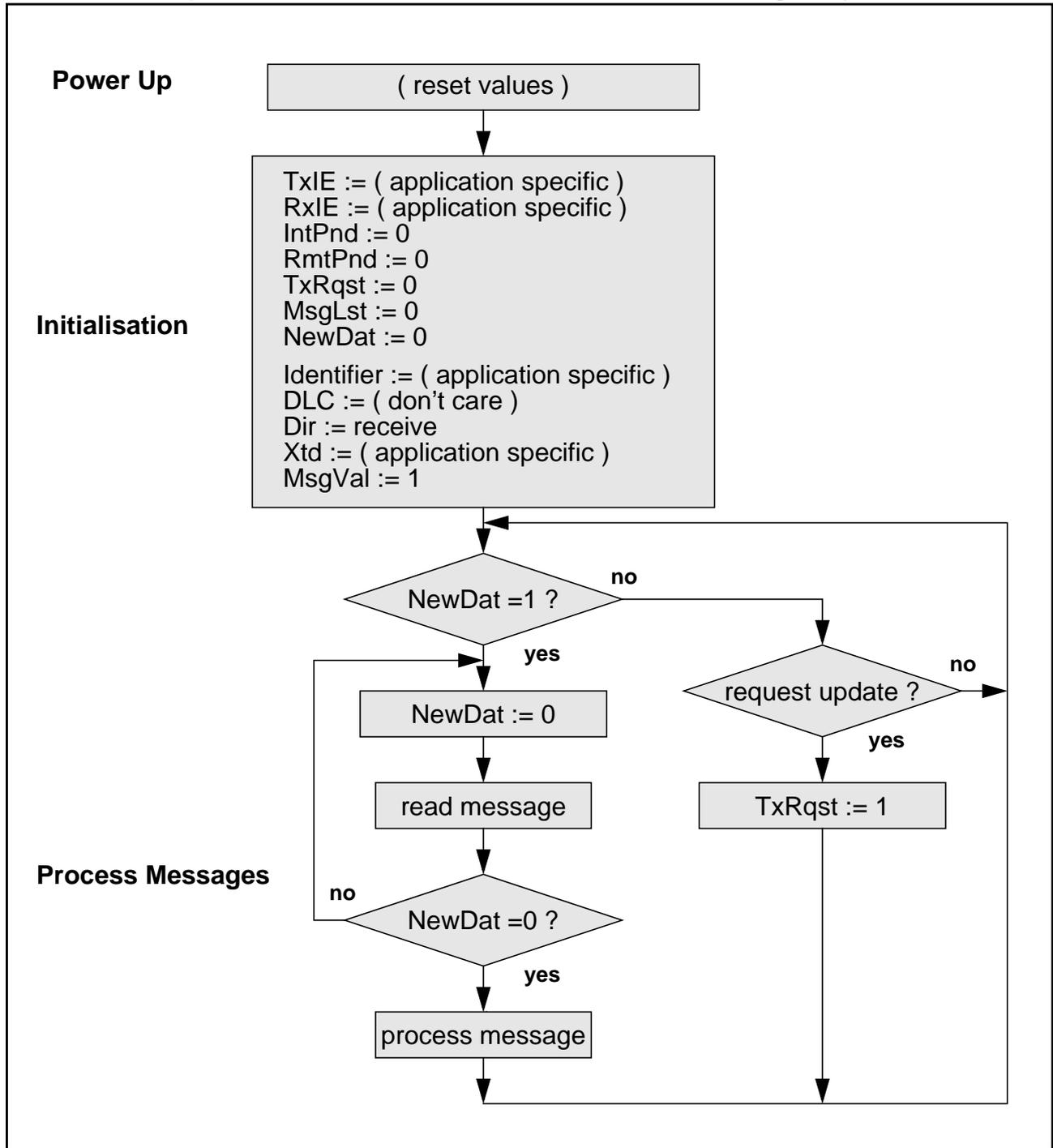


Table 12: CPU Handling of Message Objects 1-14 (Receive)

spec_flow_diagrams.fm

K8/EIS - Klose-2989

061.2/2.3 - 15.08.97

5.5 CPU Handling of Message Object 15 (Receive)

These are the operations the host-CPU executes to receive Message Object 15.

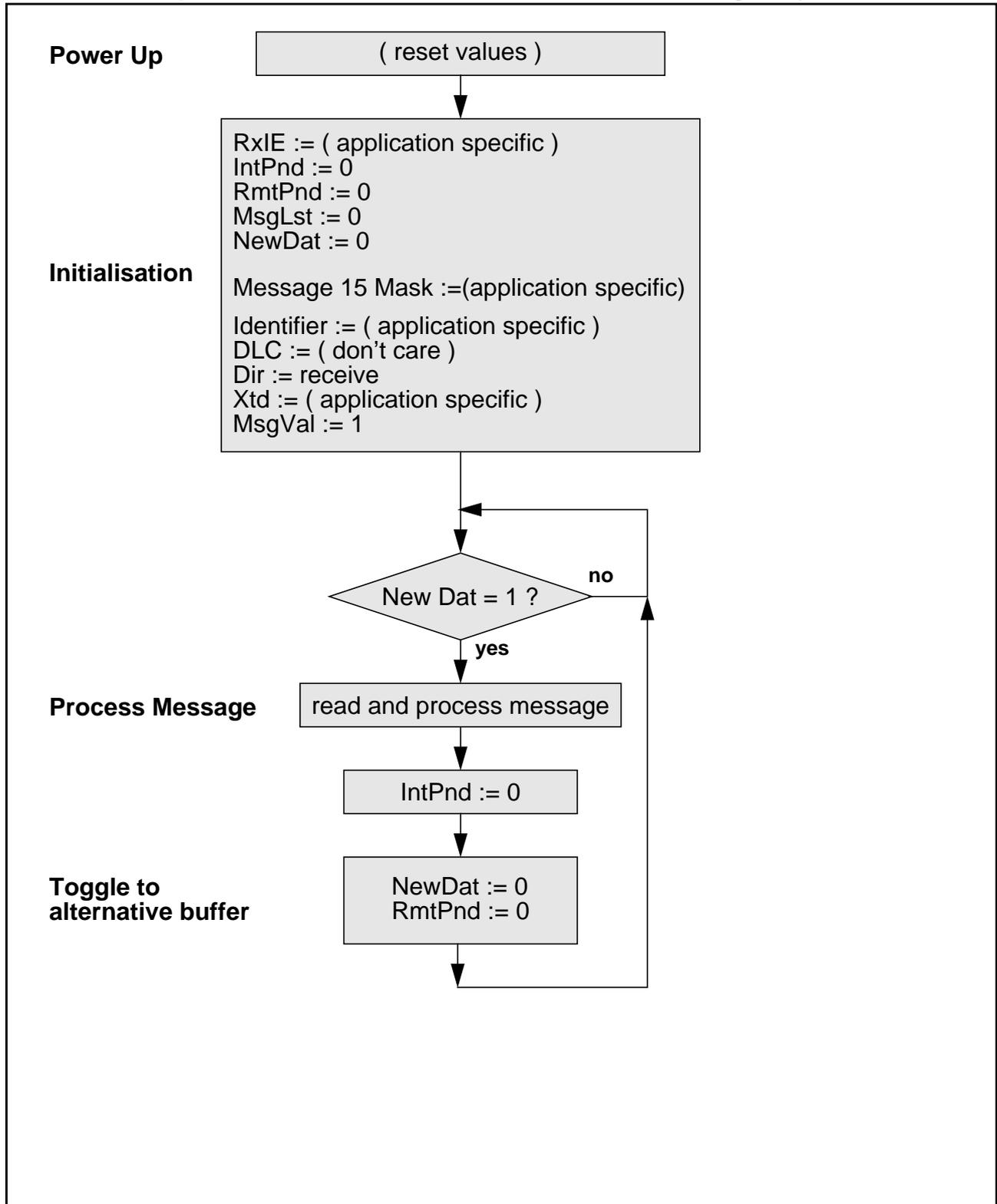


Figure 7: CPU Handling of Message Object 15 (Receive)

spec_flow_diagrams.fm

K8/EIS - Klose-2989

0612/2.3 - 15.08.97

6. CPU Interface Logic

The CPU Interface Logic (CIL) controls the data stream between the SPI (serial data) and the RAM (parallel data). Therefore the CIL converts serial address/control/data signals from the CPU to parallel read and write accesses to the internal memory bus.

The CIL allows a direct serial interface connection to the CC750 for most commonly used CPUs.

The internal memory bus is a non-multiplexed parallel bus that is used by both the CIL and the CAN Controller to read and write to the Message Memory.

6.1 Serial Interface Techniques

The serial interface on the CC750 is fully compatible to the SPI protocol of Motorola and will interface to most commonly used serial interfaces. The serial interface is implemented in slave mode only, and responds to the master using the specially designed serial interface protocol. This serial interface allows an interconnection of several CPU's and peripherals on the same circuit board.

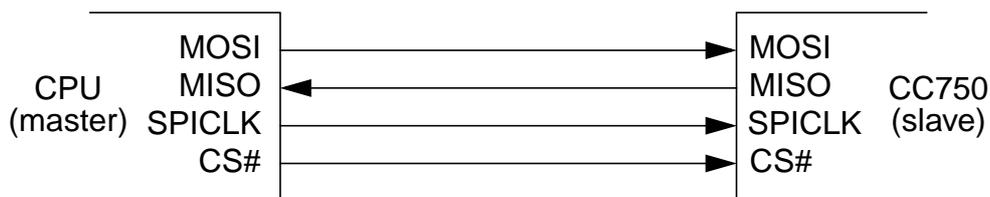


Figure 8: Interconnection for serial communication

MOSI: Master Out Slave In

The MOSI pin is the data output of the master device (CPU) and the data input of the slave device (CC750). Data is transferred serially from the master to the slave on the signal line, with the most significant bit first and least significant bit last.

MISO: Master In Slave Out

The MISO pin is the data output of the slave device (CC750) and the input of the master device (CPU). Data is transferred serially from the slave to the master on the signal line, with the most significant bit first and least significant bit last.

CS#: Chip Select (used as Slave Select for the SPI interface)

A low level on the slave select input (CS#) enables the CC750 to accept data on the MOSI pin. The CS# must not toggle during data transfer.

The CC750 will only drive data to the serial data register if CS# is "0".

SPICLK: Serial Clock

The master device provides the serial clock for the slave device. Data is transferred synchronously to this clock in both directions. The master and the slave devices exchange a data byte during a sequence of eight clock pulses.

6.2 Serial Interface Protocol

The general format of the data exchange from the CC750 to the master is a bit-for-bit exchange on each SPICLK clock pulse. Data is read on the rising edge of the SPICLK, and is changed on the falling edge of SPICLK.

Bit exchanges in multiples of 8 bits and up to 15 bytes of data are allowed. A maximum of 17 bytes can be send to the CC750-SPI, including one address byte, one SPI Control Byte, and 15 data bytes.

At the beginning of a transmission over the serial interface, the first byte will be the address of the CC750 special function register or the CC750 RAM to be accessed. The next byte transmitted is a Control Byte, which contains the number of bytes to be transmitted and whether this is to be a read or write access to the CC750. These first two bytes are followed by the data bytes (1 to 15).

To ensure the CC750 device is not out of synchronization, the CC750 will transmit the values "AAH" and then "55H" through the MISO pin while the master transmits the Address and Control Byte. The master may check for the reception of these bytes.

If the master did not receive the first synchronization byte ("AAH"), the Data transmitted since the last reception of the synchronization bytes are invalid. The CPU should abort the actual transmission. The CC750 can be re-synchronised by transmitting an FFH byte (SPI Reset Address).

If the master receive the first synchronization byte ("AAH") but not the second one ("55H"), the transmission of the address was disturbed. The CPU should abort the actual transmission in order to prevent data loss through transmission or reception of data from wrong addresses. The CC750 can be re-synchronised by transmitting an FFH byte (SPI Reset Address).

When the SPI receives an Address or Control byte with the value FFH, the SPI interface will be reset. In this case, the SPI will assume the next byte is an address.

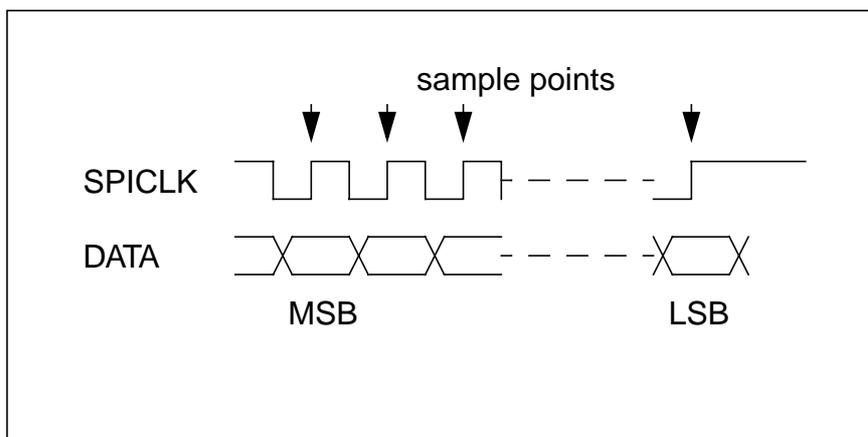


Figure 9: Serial data communication

spec_cpu_interface_logic.fm

K3/EIS - Klose-2989

061.2/2.3 - 15.08.97

6.3 Serial Control Byte

The Serial Control Byte is transmitted by the CPU to the CC750 as follows:

7	6	5	4	3	2	1	0
Dir	Sync			SDLC			

Dir Serial transmission direction

- zero The data bytes will be read, so the CC750 will transfer information to the CPU.
- one The data bytes will be sent from the CPU to the CC750.

Sync Synchronisation

These three bits must always be sent as "000".

SDLC Serial Data Length Code

These four bits contains the number of bytes to be transmitted. Valid programmed values are 1-15.

The first data byte (third byte of the SPI protocol) will be written to or read from the CC750 address (first byte of the SPI protocol). After this, the address is incremented by the SPI logic and the next data byte is written or read from this address. In one data stream, a maximum of 15 data bytes can be transferred. A DLC of zero is not allowed. After a DLC of zero is received, the SPI must be resynchronized. The SPI must also be resynchronized if one of the Synchronisation Bits was received as "1".

When the CPU conducts a READ, the CPU sends an address byte and a Serial Control Byte. While the CC750 responds back with data, it ignores the MOSI pin (transmission from the CPU).

The CPU may transmit the next address and Serial Control Byte after CS# is de-activated and then re-activated. This means the chip select should be activated and de-activated for each read or write transmission.

Synchronization bytes must be monitored carefully. For example, if the CC750 does not transmit the AAH and 55H synchronization bytes correctly, then the previous transmission may be incorrect too. The MISO pin is tri-stated if CS# is inactive.

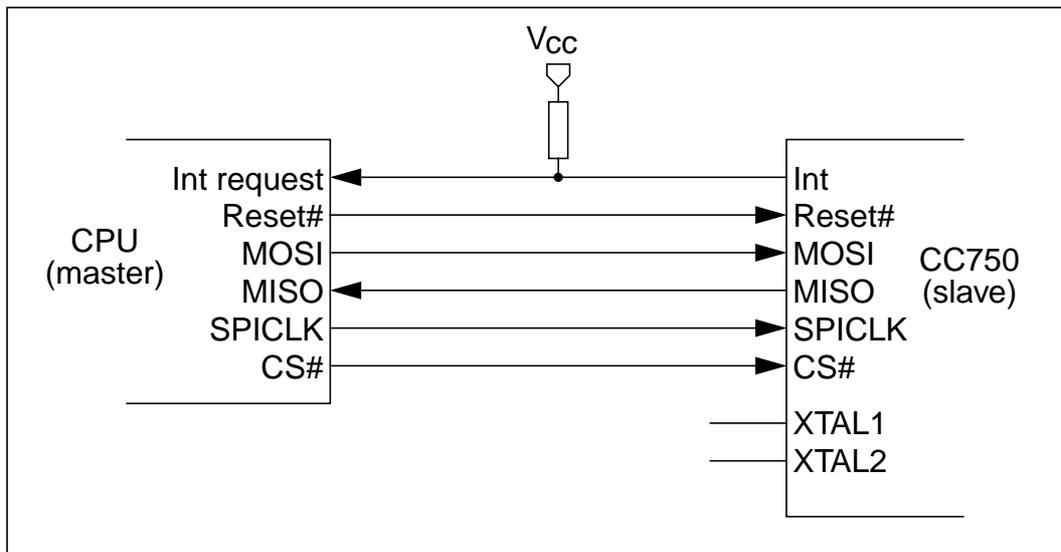


Figure 10: CC750 SPI Interface Schematic

7. Electrical Specification

7.1 Handling Instructions

Handle with extreme care. Pins should not be touched. Follow ESD (Electrostatic Discharge) protection procedure.

7.2 Absolute Maximum Ratings

Functional operation under any of these conditions is not implied. Operation beyond the limits in this table may impair the lifetime of the device.

Parameter	Value	Cat*
Maximum rise-time of Supply-Voltage	1 V/ μ s	C
Maximum Supply-Voltage (V_{CC} - V_{SS})	- 0.5 V to +7.0 V	C
Maximum current at all inputs/outputs	\pm 25 mA	C
Protection of inputs/outputs against ESD (HBM)	\pm 1,5kV (1.5 k Ω , 100 pF)	C
Storage temperature	-40°C to +150°C	C

Table 13: Absolute Maximum Ratings

Note: For the conditions listed above the IC is protected against latch up effects.

7.3 DC-Characteristics

Conditions: $V_{CC} = 5V \pm 10\%$, $T_A = -40^\circ\text{C}$ to $+125^\circ\text{C}$

	Parameter	Min	Max	Unit	Test Conditions	Cat*
V_{IL}	Input Low Voltage	-0.5	0.8	V		B
V_{IH}	Input High Voltage (All inputs except RESET#)	3.0	$V_{CC}+0.5$	V		B
V_{IH1}	Input High Voltage RESET# Hysteresis on RESET#	3.0 100	$V_{CC}+0.5$	V mV		B A
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 1.6$ mA	A
V_{OH}	Output High Voltage	$V_{CC}-0.8$		V	$I_{OH} = -1.6$ mA	A
I_{LC}	Input Leakage Current		± 1	μ A	$V_{SS} < V_{IN} < V_{CC}$	A
C_{IN}	Pin Capacitance ⁽¹⁾		10	pF	$f_{XTAL} = 1$ kHz	C
I_{CC}	Supply Current ⁽²⁾		50	mA		A

Table 14: DC-Characteristics

	Parameter	Min	Max	Unit	Test Conditions	Cat*
I_{SLEEP}	Sleep Current ^(2,3)		15	μA	XTAL1 clocked	A
I_{PD}	Power down Current ^(2,3)		15	μA	XTAL1 clocked	A

Table 14: DC-Characteristics

(1) Typical value based on characterization data.

(2) $f_{\text{XTAL}} = 16 \text{ MHz}$, $f_{\text{MCLK}} = 8 \text{ MHz}$

(3) All pins are driven to V_{SS} or V_{CC} , $V_{\text{CC}} = 5\text{V}$

7.4 A.C. Characteristics

Conditions: $V_{\text{CC}} = 5\text{V} \pm 10\%$, $V_{\text{SS}} = 0\text{V}$, $T_{\text{A}} = -40^\circ\text{C}$ to $+125^\circ\text{C}$, $C_{\text{L}} = 100 \text{ pF}$

Symbol	Parameter	Min	Max	Unit	Cat*
f_{XTAL}	Oscillator Frequency	8	20	MHz	B
f_{SCLK}	System Clock Frequency	4	10	MHz	B
f_{MCLK}	Memory Clock Frequency	2	8	MHz	B
f_{SPICLK}	SPI Clock Frequency	0.5	f_{MCLK}	MHz	B
t_{CYC}	$1/f_{\text{SPICLK}}$	125	2000	ns	B
t_{SKHI}	Clock High Time	45		ns	B
t_{SKLO}	Clock Low Time	45		ns	B
t_{LEAD}	Enable Lead Time	70		ns	B
t_{LAG}	Enable Lag Time	70		ns	B
t_{ACC}	Access Time		60	ns	B
t_{PDO}	Data Out Delay Time		30	ns	B
t_{HO}	Data Out Hold Time	0		ns	B
t_{DIS}	Data Out Disable Time		125	ns	B
t_{SETUP}	Data Setup Time	25		ns	B
t_{HOLD}	Data Hold Time	25		ns	B
t_{RISE}	Input Rise Time		45	ns	C
t_{FALL}	Input Fall Time		45	ns	C
t_{CS}	Chip Select High Time	125		ns	B

Table 15: A.C. Characteristics

***Category:**

- A Parameter measured as analog value in series testprogram.
- B Parameter tested as go/nogo test in series testprogram.
- C Characterized only or guaranteed by design.

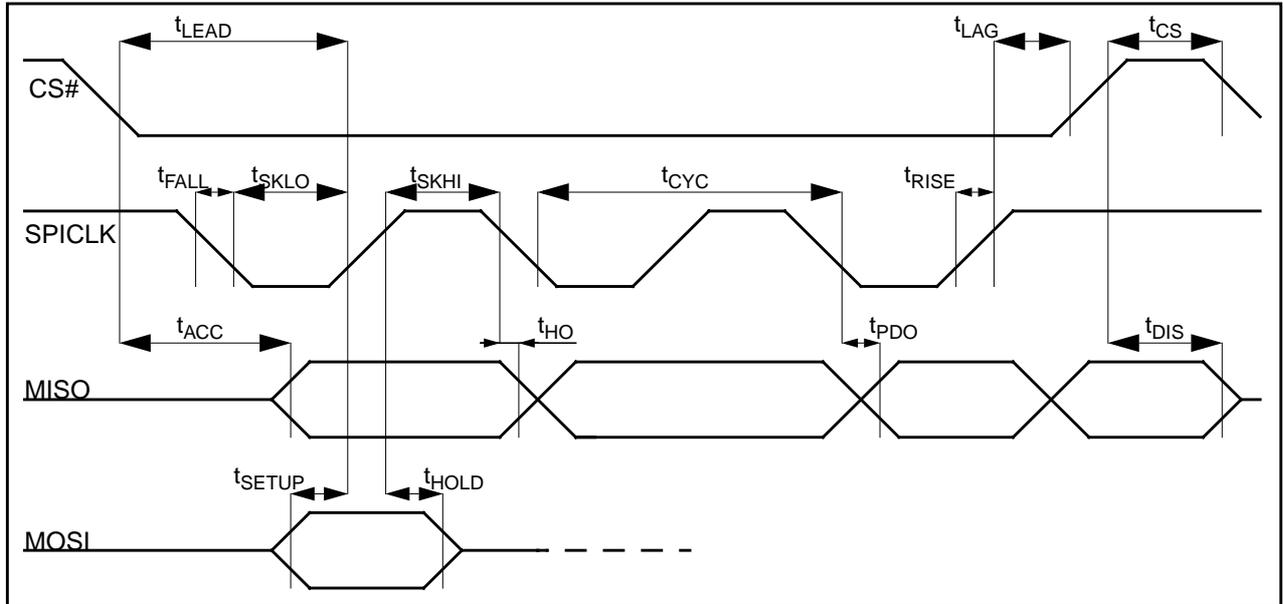


Figure 11: A.C. Characteristics for Serial Interface

7.5 Waveforms for testing

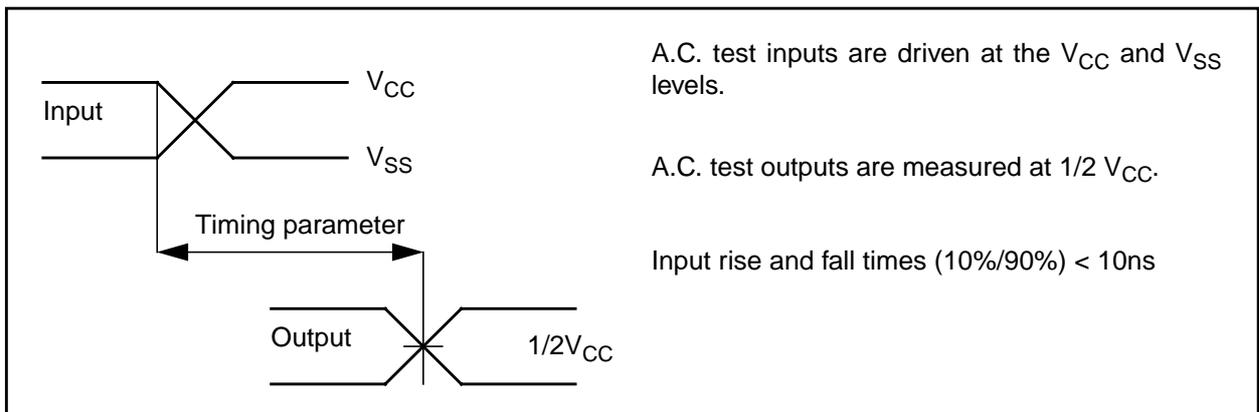


Figure 12: Input and output reference waveforms

spec_e_characteristics.fm

K8/EIS - Klose-2989

061712.3 - 15.08.97

List of Figures

Figure 1: Block Diagram of CC750.....	8
Figure 2: Package Diagram of CC750	10
Figure 3: Time Segments of Bit Time	28
Figure 4: Bit Timing	29
Figure 5: CC750 handling of Message Objects 1-14 (Transmit).....	44
Figure 6: CC750 handling of Message Objects 1-14 (Direction = Receive).....	45
Figure 7: CPU Handling of Message Object 15 (Receive).....	48
Figure 8: Interconnection for serial communication	49
Figure 9: Serial data communication	50
Figure 10: CC750 SPI Interface Schematic.....	52
Figure 11: A.C. Characteristics for Serial Interface.....	55
Figure 12: Input and output reference waveforms.....	55

List of Tables

Table 1: Pin description	11
Table 2: Reset values of CC750 registers	12
Table 3: Reset states of CC750 output pins	13
Table 4: CC750 address map	15
Table 5: Function of Power Down and Sleep bits	22
Table 6: Maximum MCLK frequency for various oscillator frequencies	22
Table 7: Interrupt Register values with corresponding Interrupt Sources	33
Table 8: Message Object Structure	35
Table 9: Representation of bit pairs in Control Registers	36
Table 10: Bit combinations to start transmissions	40
Table 11: CPU Handling of Message Objects 1-14 (Transmit)	46
Table 12: CPU Handling of Message Objects 1-14 (Receive)	47
Table 13: Absolute Maximum Ratings	53
Table 14: DC-Characteristics	53
Table 15: A.C. Characteristics	54

8. Appendix

8.1 Documentation of Changes

8.1.1 Changes on Revisions

8.1.1.1 Revision 1.1

This is a fully revised Target Specification.

8.1.1.2 Revision 1.2

Differences in function compatibility to CC770 and Intel 82526/7 are described more detailed. See chapter 1.2.

Bit Timing Registers are writeable without setting the Init bit.

Init set to one does break a transmission or reception of a message in process.

Change flow diagrams 7.2 and 7.4.

Add test categories to the electrical characteristics.

Change lower storage temperature from -65°C to -40°C.

Change Burn In circuit and parameters.

Protection against ESD is adjusted to 1,5kV.

8.1.2 Others

EOF