CC770

(Stand Alone CAN Controller)

Target Specification

Revision 1.3

17.11.03

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

Copyright Notice and Proprietary Information

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

1.1 General Information

This specification describes the functionality of the CC770 with design step D and E.

1.2 General Data

Device Name	:	CC770
Packages	:	PLCC44, MQFP 44*, Chip
Device Number	:	CC770E PLCC package : 0 272 230 535 CC770D PLCC package : 0 272 230 480 CC770D chip on wafer :1 279 993 117
1st Application	:	Replacement of AN82527

* Not in production.

1.3 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 CPU Interface
  - 8-bit Multiplexed
  - 16-bit Multiplexed
  - 8-bit Synchronous Non-Multiplexed
  - 8-bit Asynchronous Non-Multiplexed
  - Serial Interface
- Two 8-bit Bidirectional I/O Ports
- Flexible Interrupt Structure
- Flexible Status Interface
- Programmable Clock Output
- Compatibility with the AN82527
- PLCC44, MQFP 44* Package, Chip

* Not in production.

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 CC770 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 CC770 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 CC770 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 CC770 features a powerful CPU interface that offers flexibility to directly interface to many different CPUs. It can be configured to interface with CPUs using an 8-bit multiplexed, 16-bit multiplexed, or 8-bit non-multiplexed address/data bus for Intel and Motorola architectures. A flexible serial interface is also available when a parallel CPU interface is not required.

The CC770 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 dedicated acceptance mask designed to allow select groups of different message identifiers to be received.

The CC770 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 CC770 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 CC770 offers hardware, or pinout and software compatibility with the AN82527. Since some reserved bits of the AN82527 are used for additional functions in the CC770, these bits must be programmed as described in the Intel Specification to achieve software compatibility.

The CC770 is fabricated in Bosch's reliable HC65-ST/65 technology and is available in a 44-lead PLCC for the automotive temperature range (-40 °C to +125 °C ambient).

1.4 Functional Overview

The CC770 CAN controller consists of six functional blocks. The CPU Interface logic manages the interface between the CPU (host microcontroller) and the CC770 using an address/data bus or the 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 RAM is the interface layer between the CPU and the CAN bus. The two port blocks provide 8-bit low speed I/O capability. The clockout block allows the CC770 to drive other chips, such as the host-CPU.

The CC770 RAM 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 set as 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 CC770 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 CC770. Message object 15 is also buffered to allow the CPU time to service a message received.

1.5 CC770 Block Diagram

The CC770 consists of the following functional blocks:

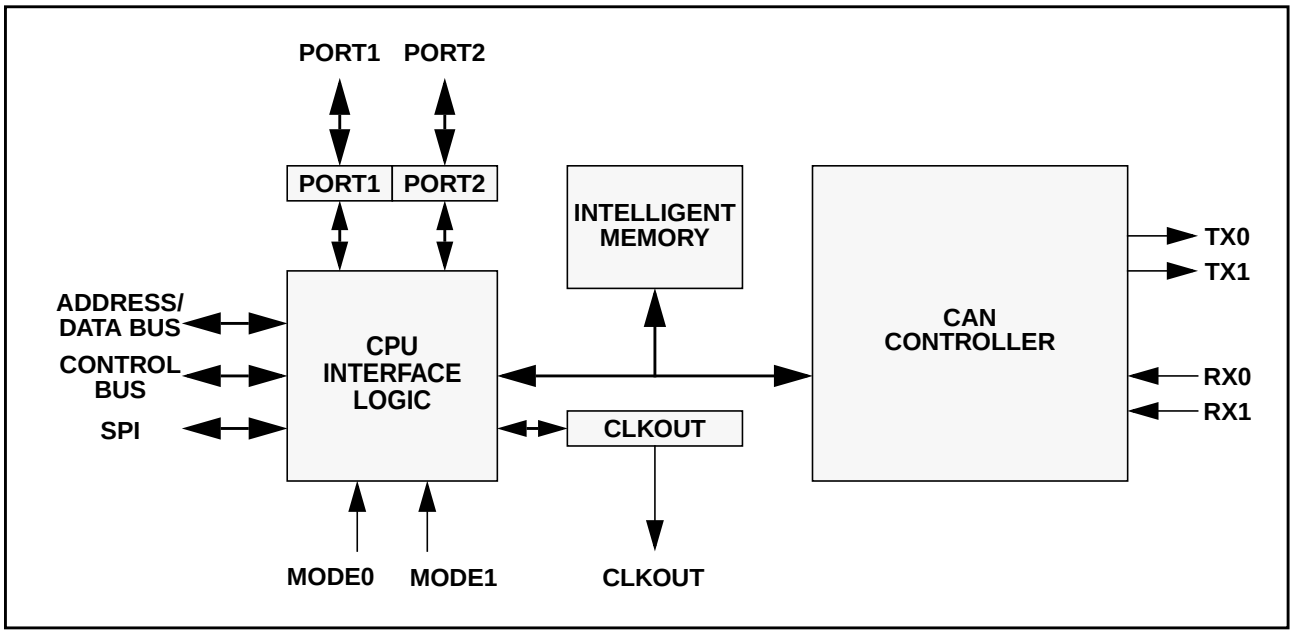


Figure 1: Block Diagram of CC770.

1.5.1 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.

1.5.2 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.

1.5.3 CPU Interface Logic

The CC770 provides a flexible CPU interface capable of interfacing to many commonly used microcontrollers. The following five modes could be selected using two CPU interface mode pins and the RD# and WR# pins:

Mode 0 (1) is an 8-bit Intel multiplexed address data bus.

If the RD# and WR# pins are tied low at reset in Mode 0, the serial interface (SPI) mode is entered.

Mode 1 (2) selects 16-bit Intel multiplexed address data bus.

Mode 2 (3) selects an 8-bit Motorola multiplexed address data bus.

Mode 3 (4) selects an 8-bit non-multiplexed address data bus for either synchronous or asynchronous communication.

1.5.4 Clockout

The on-chip clock generator consists of an oscillator, clock divider register and a driver circuit. The Clockout output range is XTAL (external crystal frequency) to XTAL/15.

The Clockout output driving strength (slew rate) is programmable in four steps.

1.5.5 Two 8-Bit Ports

Two 8-bit low speed input/output (I/O) ports are available on-chip. Depending on the CPU interface selected, at least 7 and up to 16 of these I/O pins are available for system use.

1. Mode 1 pin = 0, Mode 0 pin = 0
2. Mode 1 pin = 0, Mode 0 pin = 1
3. Mode 1 pin = 1, Mode 0 pin = 0
4. Mode 1 pin = 1, Mode 0 pin = 1

2. Package Diagram

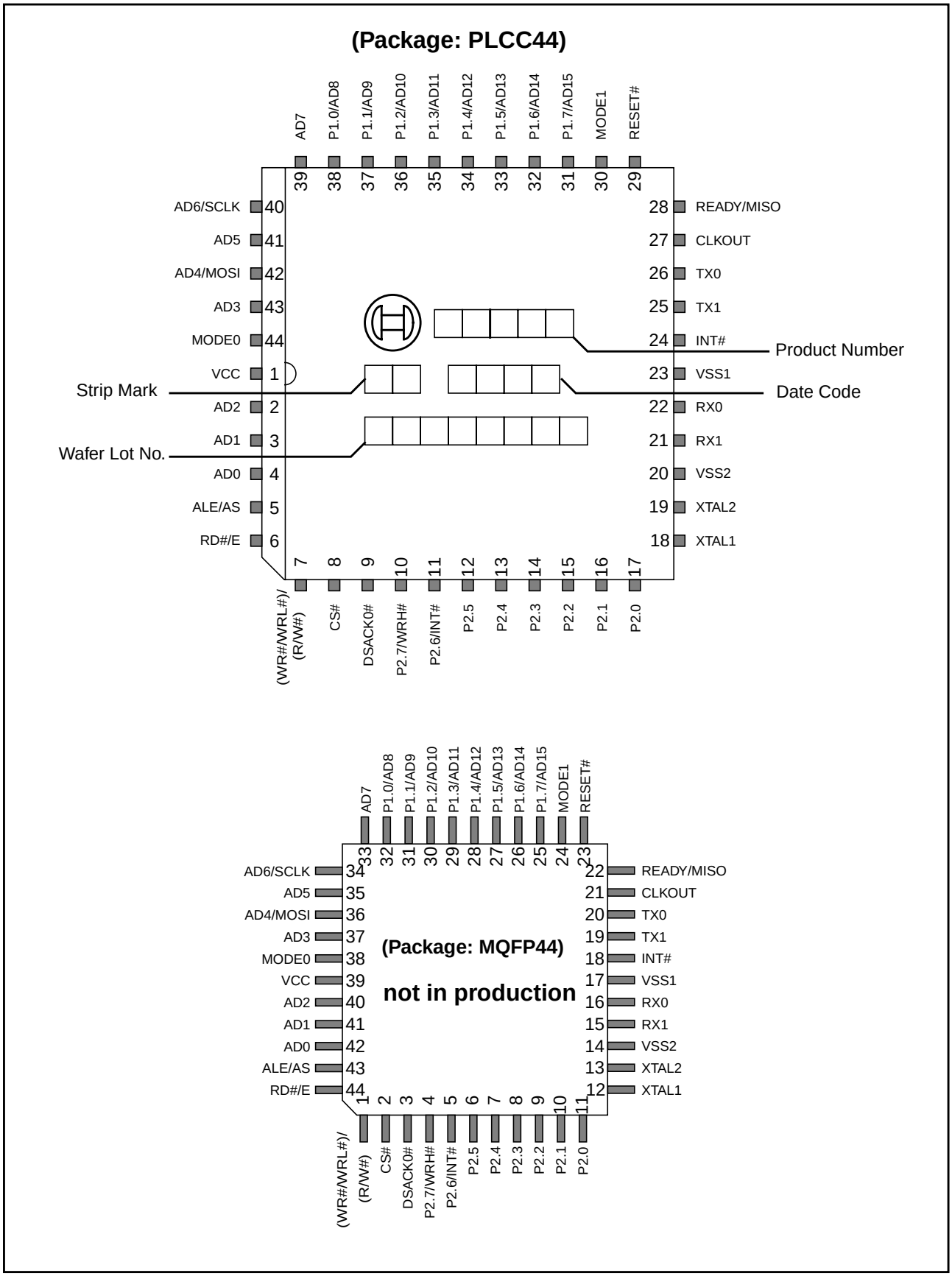


Figure 2: Package Diagrams of CC770

3. Product Description

3.1 Pin Description (PLCC 44)

Symbol	Pin	Function
V_{CC}	1	Power connection must be shorted externally to +5V DC to provide power to the entire chip.
AD2-0	2 - 4	Multi Function Pin, see Table 2.
ALE/ AS	5	ALE used for CPU Interface Modes 0 and 1. AS used for Motorola modes. (In Mode 3, pin must be tied high.)
RD#/ E	6	RD# used for CPU Interface Modes 0 and 1. E used for Motorola modes. (In Mode 3 Asynchronous, pin must be tied high.)
WR#/ WRL#/ R/W#	7	For Intel CPU Interface Modes 0 and 1. For Mode 1, 16-bit multiplexed mode, function is: write low byte For CPU Interface Mode 3.
CS#	8	A low level on this pin enables the CPU to access the CC770.
DSACK0#	9	DSACK0# is an open-drain output(1) to sychronize accesses from the CPU to the CC770 for CPU Interface Mode 3.
P2.7/ WRH#	10	Port 2, Bit 7(2) For Mode 1, 16-bit multiplexed mode, function is: write high byte
P2.6/ INT#	11	Port 2, Bit 6(2) Interrupt output (INT#) when MUX bit = 1 in CPU Interface Register
P2.5	12	Port 2, Bit 5(2)
P2.4	13	Port 2, Bit 4(2)
P2.3	14	Port 2, Bit 3(2)
P2.2	15	Port 2, Bit 2(2)
P2.1	16	Port 2, Bit 1(2)
P2.0	17	Port 2, Bit 0(2)
XTAL1	18	Input for an external clock.
XTAL2	19	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 CPUs.
V_{SS2}	20	Ground (0V) connection must be shorted externally to a V_{SS} board plane to provide analog ground for PLL.

Table 1: Pin description

Symbol	Pin	Function
RX1	21	Inverting input from CAN bus transceiver if DcR0 bit in the Bus Configuration Register (Address 2FH) is set.
RX0	22	Input from CAN bus transceiver if DcR0 bit in the Bus Configuration Register (Address 2FH) is zero.
V_{SS1}	23	Ground (0V) connection must be shorted externally to a V_{SS} board plane to provide digital ground.
INT#	24	The interrupt pin is an open drain output (requires external pull up resistor) to the CPU. The function of this pin is determined by the MUX bit in the CPU Interface Register (Address 02H), see chapter 4.4.
TX1	25	Inverted serial push-pull data output to the CAN bus transceiver. During a recessive bit TX1 is low, during a dominant bit TX1 is high.
TX0	26	Serial push-pull data output to the CAN bus transceiver. During a recessive bit TX0 is high, during a dominant bit TX0 is low.
CLKOUT	27	Programmable clock output. This push-pull output may be used to drive the clock input of the CPU.
READY/ MISO	28	READY is an open-drain output to synchronize accesses from the CPU to the CC770 for CPU Interface Modes 0 and 1. MISO is the serial push-pull data output in the SPI mode.
RESET#	29	Warm Reset: (V_{CC} remains valid while RESET# is asserted), RESET# must be driven to a low level for 1 µs 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) to ensure the oscillator and the PLL is stabile. No falling edge on the Reset pin is required during a cold reset event.
Mode1	30	Select, together with Pin 44, one of the four parallel interface modes.(3)
P1.7-0	31-38	Port 1 with multi Function Pins, see Table 2.(2)
AD7-3	39-43	Multi Function Pins, see Table 2.
Mode0	44	Select, together with Pin 30, one of the four parallel interface modes. See chapter 7.1.(3)

Table 1: Pin description

Symbol	Pin	**Mode 0** 8-Bit Intel Multiplexed	**Mode 1** 16-Bit Intel Multiplexed	**Mode 2** 8-Bit Motorola Multiplexed	**Mode 3** 8-Bit Non-Multiplexed	**SPI-Mode** Serial Interface
AD0	4	AD0	AD0	AD0	A0	ICP
AD1	3	AD1	AD1	AD1	A1	CP
AD2	2	AD2	AD2	AD2	A2	CSAS
AD3	43	AD3	AD3	AD3	A3	STE
AD4/MOSI	42	AD4	AD4	AD4	A4	MOSI
AD5	41	AD5	AD5	AD5	A5	Unused
AD6/SCLK	40	AD6	AD6	AD6	A6	SCLK
AD7	39	AD7	AD7	AD7	A7	Unused
P1.0/AD8	38	P1.0	AD8	P1.0	D0	P1.0
P1.1/AD9	37	P1.1	AD9	P1.1	D1	P1.1
P1.2/AD10	36	P1.2	AD10	P1.2	D2	P1.2
P1.3/AD11	35	P1.3	AD11	P1.3	D3	P1.3
P1.4/AD12	34	P1.4	AD12	P1.4	D4	P1.4
P1.5/AD13	33	P1.5	AD13	P1.5	D5	P1.5
P1.6/AD14	32	P1.6	AD14	P1.6	D6	P1.6
P1.7/AD15	31	P1.7	AD15	P1.7	D7	P1.7

Table 2: Multi Function Pins

Notes to pin descriptions:

(1) DSACK0# is often used as a pulldown output with a 3.3 kΩ pullup resistor and a 100 pF load capacitance. An open-drain output is used because several peripherals may be connected to the DSACK0# line. The CC770 specifies a TCHKH timing (CS# high to DSACK0# high) equal to 55 ns, however a 3.3 kΩ resistor will not sufficiently charge the line when DSACK0# is floated by the CC770. To meet this timing, the CC770 has an active pullup that drives the DSACK0# output until it is high, and then the pullup is turned off. Therefore, the pullup is active for a short time only.

(2) Port1 and Port2 pins are weakly held high until the Port Configuration Registers have been written (locations 9FH and AFH respectively).

(3) Mode0 and Mode1 pins are internally connected to weak pulldowns. These pins will be pulled low during reset if unconnected. Following reset, these pins float.

3.2 Hardware Reset

3.2.1 Reset values of CC770 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 and the PLL is stable. On warm reset (V_{CC} remains valid while RESET# is asserted), RESET# must be driven to a low level for 1 µs minimum.

The registers of the CC770 have the following values after warm reset:

Register	Address	Reset Value
Control Register	00H	01H
Status Register	01H	undefined
CPU Interface Register	02H	61H
High Speed Register	04+05H	unchanged
Global Mask - Standard	06+07H	unchanged
Global Mask - Extended	08-0BH	unchanged
Message 15 Mask	0C-0FH	unchanged
Clockout Register	1FH	00H or 01H depending on CPU Interface Mode
Bus Configuration	2FH	00H
Bit Timing Register 0	3FH	00H
Bit Timing Register 1	4FH	00H
Interrupt Register	5FH	00H
P1 Configuration Register	9FH	00H
P2 Configuration Register	AFH	00H
P1 In	BFH	FFH
P2 In	CFH	FFH
PI Out	DFH	00H
P2 Out	EFH	00H
SPI Reset Address	FFH	not readable
Message Objects 1-15		unchanged

Table 3: Reset values of CC770 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 CC770 output pins

The behaviour of CC770 output pins in reset procedure:

Pin	output state while reset is active	output state direct after reset
Mode 0/1	weakly pulled low	high impedance
Port 1/2	weakly pulled high	high impedance input
Clockout	active	
TX0	1 (recessive)	
TX1	0 (recessive)	
INT#	float	
DSACK0#	float	

Table 4: Reset states of CC770 output pins

3.3 Software Initialization

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 CC770 are stopped and the TX0 and TX1 outputs are recessive. The error counters are unchanged. Initialization is used to configure the CC770 memory without interference to or by CAN bus.

Resetting Init bit completes initialization and the CC770 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 4.14.

3.5 Silent Mode

The CAN Controller can be set in Silent Mode by programming the Bus Configuration Register bits DcR1 and DcR0 both to one.

In Silent Mode, the CC770 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 CC770 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 CC770 has exited Power Down.

The CC770 enters Sleep Mode after the Sleep bit in the CPU Interface Register (bit 3, register 02H) is set and a possibly ongoing 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 CC770 requires a minimum of 10 ms to come out of Sleep Mode after bus activity occurs to ensure the oscillator and the PLL is stabile.

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 explains the functional operation of the CC770 by describing the registers used to configure the chip and Message Objects.

4.1 CC770 Address Map

The CC770 allocates an address space of 256 bytes. All registers are organized as 8-bit registers.

Address	Register
00H	Control
01H	Status
02H	CPU Interface
03H	reserved
04+05H	High Speed Read
06+07H	Global Mask - Standard
08-0BH	Global Mask - Extended
0C-0FH	Message 15 Mask
10-1EH	Message 1
1FH	ClkOut *
20-2EH	Message 2
2FH	Bus Configuration *
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	Receive Error Counter
70-7EH	Message 7
7FH	Transmit Error Counter

Table 5: CC770 address map

Address	Register
80-8EH	Message 8
8FH	reserved
90-9EH	Message 9
9FH	P1CONF *
A0-AEH	Message 10
AFH	P2CONF *
B0-BEH	Message 11
BFH	P1IN
C0-CEH	Message 12
CFH	P2IN
D0-DEH	Message 13
DFH	P1OUT
E0-EEH	Message 14
EFH	P2OUT
F0-FEH	Message 15
FFH	Serial Reset Address

Table 5: CC770 address map

NOTE:

* The CPU may write to the Configuration 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	EAF	res	EIE	SIE	IE	Init
rw	rw	rw	r	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".

CCE Change Configuration Enable

one The CPU has write access to the Configuration Registers (ClkOut, Bus Configuration,...). Init bit should be set in order to stop CAN bus activities, if the values in the Bit Timing Registers must be changed.

zero The CPU has no write access to the Configuration Registers.

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

EAF Enable Additional Functions

one The Receive Error Counter (address 6FH) and the Transmit Error Counter (address 7FH) can be read. The additional mask bits MDir and MXtd in the Message 15 Mask are enabled.

zero The addresses 6FH and 7FH are reserved. The additional mask bits MDir and MXtd in the Message 15 Mask are disabled and will be read as 00, independent of the last value written to those bits while EAF was "1".

The CC770 provides additional functions giving the programmer the possibility to read the Error Counters and to mask Xtd and Dir bits in the Message 15 Mask, allowing the reception of all possible messages not received by other Message Objects.

To enable this functions the programmer has to set the EAF bit. In the AN82527 this bit is reserved and must not be set to 1.

EIE Error Interrupt Enable

one Error interrupts enabled. A change in the error status of the CC770 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 CC770 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 CC770.

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 BusOff 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 CC770 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 CC770 when it goes busoff. Initialization is a state which allows the user to configure the CC770 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 CC770 has removed itself from the CAN bus after going busoff.

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 undefined.

BOff Bus Off Status

one There is an abnormal rate of occurrences of errors on the CAN bus.

zero The CC770 is not busoff.

The Bus Off condition occurs when the Transmit Error Counter in the CC770 has reached the limit of 256. In consequence, the CC770 going busoff. During busoff, no messages 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 busoff recovery sequence begins. The busoff recovery sequence resets the Transmit and Receive Error Counters. After the CC770 counts 128 packets of 11 consecutive recessive bits on the CAN bus, the busoff state is exited.

The Bus Off Status bit is written by the CC770.

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 CC770 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 CC770.

WakeUp Wake Up Status

one The CC770 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 CC770 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 CC770 comes out of Power Down mode.

The WakeUp bit and WakeUp interrupt is reset by reading the Status Register.

This bit is written by the CC770.

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 CC770. A successfully received message may be any CAN bus transmission that is error-free, regardless of whether the CC770 has configured a Message Object to receive that particular message identifier.

The CC770 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 is never reset by the CC770.

The CC770 will set this bit, the CPU may 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 CC770 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, the CC770 wanted to send a dominant level (bit of logical value 0), but the monitored CAN bus value was recessive. During busoff 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 CC770 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 CC770 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 CC770 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
RstST	DSC	DMC	PwD	Sleep	MUX	StEn	CEn
r	rw	rw	rw	rw	rw	rw	rw

The value of the CPU Interface Register during the hardware reset is E1H, after the end of the hardware reset the default value is 41H.

RstSt Hardware Reset Status

- one The hardware reset of the CC770 is active (RESET is low). While reset is active, no access to the CC770 is possible, except read access on the CPU interface register.
- zero Normal operation.

The CPU must ensure this bit is zero before further access to the CC770 after reset.

This bit is written by the CC770.

DSC Divide System Clock (SCLK).

- one The system clock is equal to XTAL/2.
- zero The system clock is equal to XTAL.

This bit is written by the CPU.

DMC Divide Memory Clock (MCLK)

one The memory clock is equal to SCLK/2.

zero The memory clock is equal to SCLK.

This bit is written by the CPU.

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 6: Function of Power Down and Sleep bits

Information about low Current Modes see chapter 3.6.

The Sleep bit and the Power Down bit may be set and reset by the CPU. The Sleep bit will also be reset by CAN bus activity during Sleep Mode.

MUX (see text below)

one Pin 24 = float, pin 11 = INT#.

zero Normal operation: Pin 24 = INT#, Pin 11 = P2.6.

This bit is written by the CPU.

Bit 2 (MUX) controls whether the interrupt output is available at pin 11 (MUX = 1) or pin 24 (MUX = 0). The AN82527 provides an output voltage of VCC/2 at pin 24 if MUX = 1 to support an "ISO low speed physical layer". The CC770 lets pin 24 float if MUX = 1, assuming an appropriate CAN bus driver IC is utilized.

StEn Clockout Stretch Enable

one The Clockout for the CPU is stretched to lengthen the read and write cycles of the CC770 instead of generating wait states.
Example: CPU read data from CC770
When the CPU writes the address byte to the CC770 the Clockout is disabled, until the data of the selected address is available for read.

zero The Clockout is not streched.

CEn Clockout enable

one Clockout signal is enabled, (default after reset).

zero Clockout signal is disabled.

Accesses to the CPU Interface Register are asynchronous, so it is possible to read and write this register even if there is no clock input or during Power Down Mode and Sleep Mode.

4.4.1 Clocking Description

There are two analogue clocks and four digital clocks in the CC770. The analogue clocks are the crystal oscillator clock (XTAL) and the PLL clock (PLLCLK). The digital clocks are the system clock (SCLK), the memory clock (MCLK), the Clockout pin signal, and the SPI clock (SPICLK). While the SPI clock is directly controlled by the SPI master, the other digital clocks are derived from the analogue clocks.

The PLLCLK (two times the frequency of XTAL) is provided as clock source for SCLK in order to allow a sufficient system clock frequency SCLK for high CAN bit rates at a moderate XTAL frequency. The balanced two-phase clock SCLK is always the result of a divide-by-2; CPU Interface Register's bit DSC decides whether the PLL (DSC="0", SCLK=XTAL) or XTAL (DSC="1", SCLK=XTAL/2) is the source of SCLK. SCLK is the clock of the CAN's bit timing and other CAN protocol functions.

The memory clock MCLK is derived from SCLK, it is either the same as SCLK (DMC="0") or it is SCLK/2 (DMC="1"), compensating for the limited clocking range of the CC770's Content Addressable Memory (CAM).

The Clockout pin signal is always derived from PLLCLK, its actual frequency is controlled by the CDv value in the ClkOut register (1FH).

If the Clockout pin is not used and the SCLK frequency is sufficient at XTAL/2, then the PLL should be disabled in order to reduce the CC770's electromagnetic interference (EMI). This is done by setting DSC="1", StEn="1", and CEn="0".

When the PLL is enabled, the minimum XTAL frequency is 8 MHz. Since the CC770's design is fully static, the XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled.

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

Table 7: Maximum MCLK frequency for various oscillator frequencies

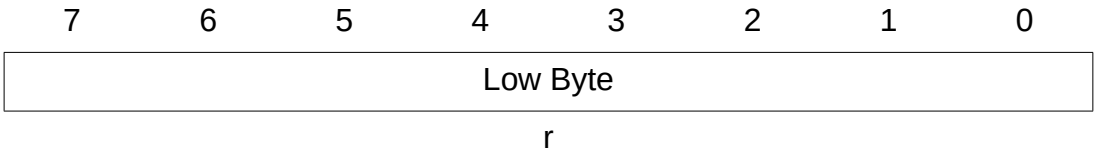
Notes:

Frequency of SCLK = $f_{XTAL}/(1 + DSC\ bit)$

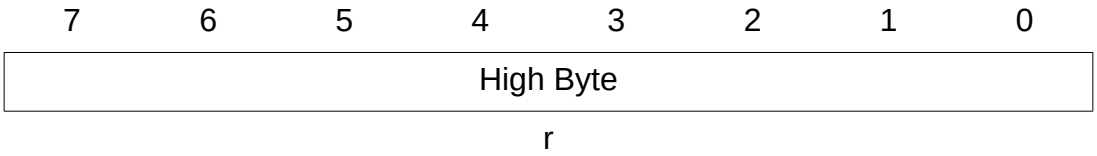
Frequency of MCLK = $f_{SCLK}/(1 + DMC\ bit) = f_{XTAL}/[(1 + DSC\ bit) * (1 + DMC\ bit)]$

4.5 High Speed Read Register (04+05H)

High Speed Read Register (04H)



High Speed Read Register (05H)



The value of the High Speed Read register is not affected by a hardware reset.

The High Speed Read register is a read only register and is the output buffer for the CPU Interface Logic. This register is part of the CPU Interface Logic and is not located in the RAM.

During a read to the RAM (low speed registers) this register is loaded with the value of the low speed register being accessed.

The High Speed Read register is available to provide a method to read the CC770 when the CPU (host microcontroller) is unable to satisfy read cycle timings for low speed CC770 registers. In other words, if the read access time of the CC770 is too slow for the CPU and the CPU cannot extend the read bus cycle, the following double read method should be used.

4.5.1 Double Read Operation

The CPU can execute double reads where the first read addresses the low speed register and the second read addresses the High Speed Read register. The first read is a dummy read for the CPU, however the low speed register value is stored in the High Speed Read register. The second read to the High Speed Read register will provide the data from the desired low speed register.

The advantage of double reads is both read operations have fast access times. The first read of the low speed register requires 40 ns (verify in current data sheet) to load the High Speed Read Register (the data on the address/data pins is not valid). The second read of the High Speed Read register requires 45 ns (verify in current data sheet) and the data on the address/data bus is valid.

Therefore, if the access time of a low speed register is too long for the CPU then a second read to the High Speed Register will produce the correct data. Please note Low and High Speed registers have different access timing specifications in the CC770 data sheet.

During a 16-bit read access the low and high byte will contain the 16-bit value from the read access. For an 8-bit read access the low byte will contain the value from the read access.

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

Global Mask (06H)

7	6	5	4	3	2	1	0
Msk28	Msk27	Msk26	Msk25	Msk24	Msk23	Msk22	Msk21
rw	rw	rw	rw	rw	rw	rw	rw

Global Mask (07H)

7	6	5	4	3	2	1	0
Msk20	Msk19	Msk18	res				
rw	rw	rw	r				

The value of the Global Mask Standard is not affected by a hardware reset.

Reserved bits read as "1".

Mskx Mask 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 CC770 receiver node will use the Global Mask Registers to determine whether the remote frame matches to any of its Message Objects. If the CC770 is programmed to transmit a message in response to a remote frame message identifier, the CC770 will transmit a message with the message identifier of the CC770 Message Object. The result is the remote message and the responding CC770 transmit message may have different message identifiers because some CC770 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
Msk28	Msk27	Msk26	Msk25	Msk24	Msk23	Msk22	Msk21
rw	rw	rw	rw	rw	rw	rw	rw

Global Mask Extended (09H)

7	6	5	4	3	2	1	0
Msk20	Msk19	Msk18	Msk17	Msk16	Msk15	Msk14	Msk13
rw	rw	rw	rw	rw	rw	rw	rw

Global Mask Extended (0AH)

7	6	5	4	3	2	1	0
Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5
rw	rw	rw	rw	rw	rw	rw	rw

Global Mask Extended (0BH)

7	6	5	4	3	2	1	0
Msk4	Msk3	Msk2	Msk1	Msk0	res		
rw	rw	rw	rw	rw	r		

The value of the Global Mask Extended is not affected by a hardware reset.

Reserved bits read as "0".

Mskx Mask 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 CC770 receiver node will use its Global Mask Registers to determine whether the remote frame matches to any of its Message Objects. If the CC770 is programmed to transmit a message in response to a remote frame message identifier, the CC770 will transmit a message with the message identifier of the CC770 Message Object. The result is the remote message and the responding CC770 transmit message may have different message identifiers because some CC770 Global Mask Register bits are "don’t care".

4.8 Acceptance Filtering Implications

The CC770 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 CC770 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
Msk28	Msk27	Msk26	Msk25	Msk24	Msk23	Msk22	Msk21
rw	rw	rw	rw	rw	rw	rw	rw

Message 15 Mask Register (0DH)

7	6	5	4	3	2	1	0
Msk20	Msk19	Msk18	Msk17	Msk16	Msk15	Msk14	Msk13
rw	rw	rw	rw	rw	rw	rw	rw

Message 15 Mask Register (0EH)

7	6	5	4	3	2	1	0
Msk12	Msk11	Msk10	Msk9	Msk8	Msk7	Msk6	Msk5
rw	rw	rw	rw	rw	rw	rw	rw

Message 15 Mask Register (0FH)

7	6	5	4	3	2	1	0
Msk4	Msk3	Msk2	Msk1	Msk0	MDir	MXtd	res
rw	rw	rw	rw	rw	rw	rw	r

The value of the Message 15 Mask is not affected by a hardware reset.

Reserved bit read as "0".

Mskx Mask 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)

MDir Mask Direction Bit (EAF must be set, see note)

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 (EAF must be set, see note)

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)

Notes:

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.

To enable the MDir and MXtd bits the EAF bit has to be set in the Control Register 00H.

If EAF="0", the additional mask bits MDir and MXtd in the Message 15 Mask Register are disabled and the bits will be read as "00", independent of the last value written to those bits while EAF was set. The internal interpretation is "11", so the bits Dir and Xtd must match for acceptance filtering.

4.10 ClkOut Register (1FH)

7	6	5	4	3	2	1	0
0	0	SL1	SL0	CDv			
r	r	rw	rw	rw			

The default value of the ClkOut Register after a hardware reset is 00H (Modes 0 and 1 and serial mode) or 01H (Modes 2 and 3).

The ClkOut register controls the frequency of the ClkOut signal as well as the slew rate. The default frequency of ClkOut depends on the CPU interface mode. For Modes 0, 1 and

serial mode the default frequency is XTAL. For Modes 2 and 3 the default frequency is XTAL/2. The following tables lists the programmable ClkOut frequencies and the slew rates:

CDv	ClkOut Frequency
0000	XTAL
0001	XTAL/2
0010	XTAL/3
0011	XTAL/4
0100	XTAL/5
0101	XTAL/6
0110	XTAL/7
0111	XTAL/8
1000	XTAL/9
1001	XTAL/10
1010	XTAL/11
1011	XTAL/12
1100	XTAL/13
1101	XTAL/14
1110	XTAL/15
1111	Reserved

Table 8: Programming ClkOut

SL1	SL0	slew rate
0	0	fast
0	1	medium to fast
1	0	medium to slow
1	1	slow

Table 9: Programming ClkOut slew rates

Note:

The SL0/1 bits adjusts the driving current of the CLKOUT pin. So the resulting slew rate also depends on the external capacitance of the CLKOUT circuit. Therefore the optimum configuration of the SL0/1 bits is application specific, EMI requirements have to be regarded.

4.11 Bus Configuration Register (2FH)

7	6	5	4	3	2	1	0
res	res	Pol	res	DcT1	res	DcR1	DcR0
rw	rw	rw	rw	rw	r	rw	rw

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

Reserved bit read as "0".

Pol Polarity

one A logical one is interpreted as dominant and a logical zero is recessive on the Rx0 input.

zero A logical one is interpreted as recessive and a logical zero is dominant bit on the Rx0 input.

DcT1 Disconnect TX1 output

one TX1 output driver disabled, recommended for applications with bus driver ICs.

zero TX1 output driver enabled.

DcR1 Silent Mode

one Silent Mode is active, if DcR0 bit is set.

zero Normal operation

In Silent Mode, the CC770 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.

DcR0 Select Rx input

one Rx1 is enabled and used as inverted CAN input.

zero Rx0 is enabled and used as non inverted CAN input.

Notes:

To be compatible to AN82527, the obsolete bits 7, 6, and 4 remain writeable.

Since the CC770 has no analog input comparator, its function is the same as that of the AN82527 with CoBy (bit 6) set. In the CC770, CoBy remains writable. Pol (bit 5) can invert the input value. DcR1 remains writable but without DcR0, it has no function. If DcR0 is set, Rx1 is used as (inverting) CAN input.

In applications with a bus driver IC (CF150 or other), the function of the CC770 is the same as that of AN82527.

4.12 Receive Error Counter (6FH)

7	6	5	4	3	2	1	0
RP	REC6-0						
r	r						

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.

Note:

The Receive Error Counter is only readable, if the EAF bit is set in the Control Register. Otherwise RP and REC6-0 are reserved bits.

4.13 Transmit Error Counter (7FH)

7	6	5	4	3	2	1	0
TEC7-0							
r							

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.

Note:

The Transmit Error Counter is only readable, if the EAF bit is set in the Control Register. Otherwise TEC7-0 are reserved bits.

4.14 Bit Timing Registers

4.14.1 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 CC770.

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:

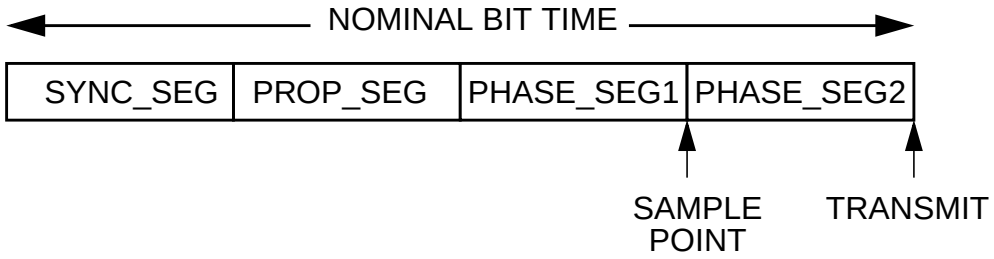


Figure 3: Time Segments of Bit Time

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.

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.

4.14.2 CC770 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 CC770. 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.

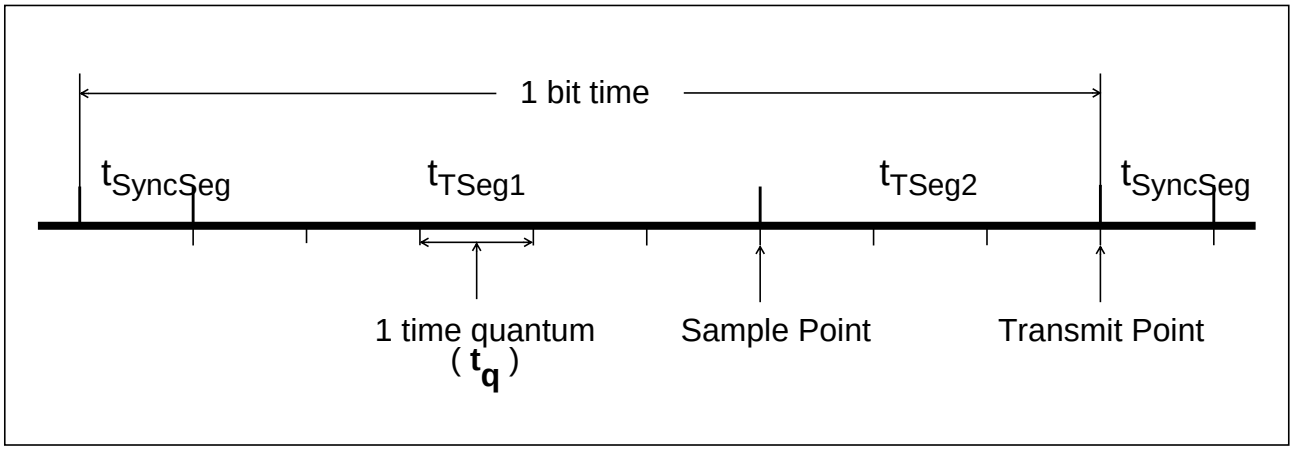


Figure 4: Bit Timing

4.14.3 CC770 Bit Time Segments

The following are relationships of the CC770 bit timing:

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

$t_{SyncSeg} = 1 \bullet t_q$

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

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

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

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

The **DSC** bit is programmed in the CPU Interface Register, 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 from the equations.

4.14.4 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 CC770 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 $\mathbf{t_q}$.

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

$t_{TSeg2} \geq 1 \bullet t_q$ = Information Processing Time
$t_{TSeg2} \geq t_{SJW}$
$t_{TSeg1} \geq 2 \bullet t_q$
$t_{TSeg1} \geq t_{SJW} + t_{Prop}$
$t_{TSeg1} \geq t_{SJW} + t_{Prop} + 2t_q$ 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 $\mathbf{t_q}$, i.e. **TSeg1** + **TSeg2** ≥ 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 $\mathbf{f_{XTAL}}$ 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})

4.14.5 Example for Bit Timing at high Baudrate

Configuration:

f_{XTAL} = 20 MHz, f_{SCLK} = 10MHz (**DSC**=1), **BRP** = 0, bitrate should be 1 MBit/s.

t_q	100	ns	= $t_{SCLK} = t_{XTAL} \bullet 2$
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	= $6 \bullet t_q$
t_{SJW}	100	ns	= $1 \bullet t_q$
t_{TSeg1}	700	ns	= $t_{Prop} + t_{SJW}$
t_{TSeg2}	200	ns	= Information Processing Time + $1 \bullet t_q$
$t_{Sync\text{-}Seg}$	100	ns	= $1 \bullet t_q$ (fix)
bit time	1000	ns	= $t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$

maximal oscillator tolerance 0.39 %

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

$$= \frac{0.1\mu s}{2 \times (13 \times 1\mu s - 0.2\mu s)}$$

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

4.14.6 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)

7	6	5	4	3	2	1	0
SJW		BRP					
rw		rw					

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

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: $t_q = t_{SCLK} \bullet (\mathbf{BRP} + 1)$ 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.

4.15 Interrupt Register (5FH)

7	6	5	4	3	2	1	0
IntId							
r							

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 CC770 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 10: Interrupt Register values with corresponding Interrupt Sources

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 CC770 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 CC770 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.

4.16 Serial Reset Address (FFH)

7	6	5	4	3	2	1	0
Serial Reset Address							
w							

The serial reset address is used to synchronize accesses between the CC770 and the CPU.

For example the CPU cannot provide a chip select signal, it is possible to write as many ones into the CC770 SPI until you get the value "AAH" from the MISO pin. Then the CC770 is synchronized and data transfer could be started. For further informations see chapter 7.4.

4.17 CC770 Message Objects (MO)

4.17.1 Message Object Structure

The Message Object is the means of communication between the host microcontoller and the CAN controller in the CC770. Message Objects are configured to transmit or receive messages.

There are 15 Message Objects located at fixed addresses in the CC770. 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 CC770 functions. In the above example the byte at address 1FH is used for the ClkOut register.

Message Object 15 is a receive-only Message Object that uses a local mask called the Message 15 Mask Register. This mask allows a large number of infrequent messages to be received by the CC770. 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 11: Message Object Structure

4.17.2 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	

Control 1 Register (Base Address + 1)

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

The values of the Control 0 and Control 1 registers are not affected by a hardware reset.

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 CPU would set the TxRqst bit of the Control 1 byte with the following instructions:

```
...
MO_CTRL1.R   EQU   #0011   ;Message Object 1 Control 1 register
...
LDA          #$EF          ;set TxRqst of Message Object 1
STA          MO_CTRL1.R    ;
```

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 12: Representation of bit pairs in Control Registers

MsgVal Message Valid

set The Message Object is valid.

reset The Message Object is invalid.

The MsgVal flag is an individual halt flag for each Message Object. While this flag is reset the CC770 will not access this Message Object for any reason. This flag 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 to reset the chip following this modification.

The CPU must reset the MsgVal flag of all unused messages during initialization of the CC770 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 flag assists reconfiguration in many cases.

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

This flag 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 CC770 transmit Message Object has the same message ID, a successful transmission of the higher numbered Message Objects will not be recognized by the CC770. The lower numbered Message Object will be falsely identified as the transmit Message Object and its transmit request flag will be reset. The actual transmit Message Object will re-transmit without end because its transmit request flag 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

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

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

The Transmit Interrupt Enable flag enables the CC770 to initiate an interrupt after the successful transmission by the corresponding Message Object.

This flag is written by the CPU.

RxIE Receive Interrupt Enable

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

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

This flag enables the CC770 to initiate an interrupt after the successful reception by the corresponding Message Object.

This flag 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

set This Message Object has generated an interrupt.

reset No interrupt was generated by this Message Object since the last time the CPU cleared this flag.

This flag is set by the CC770 following a successful transmission or reception as controlled by the RxIE and TxIE flags.

The CPU must clear this flag when servicing the interrupt.

RmtPnd Remote Request Pending

set The transmission of this Message Object has been requested by a remote node and is not yet done.

reset There is no waiting remote request for the Message Object.

This flag is only used by Message Objects with direction = transmit. This flag is set by the CC770 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 flag is reset. Following this transmission, the CC770 will clear the RmtPnd flag. In other words, when this flag 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 flag is used to send a remote frame from a receive Message Object.

TxRqst Transmit Request

set The transmission of this Message Object has been requested and has not been completed.

reset This Message Object is not waiting to be transmitted.

This flag 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 CC770 (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 flag is cleared by the CC770 along with RmtPnd when the message has been successfully transmitted, if the NewDat flag has not been set.

MsgLst Message Lost

Only valid for Message Objects with direction = receive.

set The CC770 has stored a new message in this Message Object when NewDat was still set.

reset No message was lost since the last time this flag was reset by the CPU.

This flag is used to signal that the CC770 stored a new message into this Message Object when the NewDat flag was still set. Therefore, this flag is set if the CPU did not process the contents of this Message Object since the last time the CC770 set the NewDat flag; 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.

set This Message Object may not be transmitted.

reset This Message Object may be transmitted, if direction = transmit.

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

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

NewDat New Data

set The CC770 or CPU has written new data into the data section of this Message Object.

reset No new data has been written into the data section of this Message Object since the last time this flag was cleared by the CPU.

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

4.17.3 Handling of Message Objects

For Message Objects with **direction = receive**, the CC770 sets the NewDat flag 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 flag before reading the received data and then check if the flag remained cleared when all bytes have been read. If the NewDat flag 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 6.4.

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 flags are reset. Resetting the NewDat and RmtPnd flags before resetting the IntPnd flag will result the interrupt line remaining active. See flow diagram in chapter 6.5.

For Message Objects with **direction = transmit**, the CPU should set the NewDat flag to indicate it has updated the message contents. This is done at the same time the CPU clears the CPUUpd flag. This will ensure that if the message is actually being transmitted during the time the message was being updated by the CPU, the CC770 will not reset the TxRqst flag. In this way, the TxRqst flag is reset only after the actual data has been transferred. See flow diagram in chapter 6.3.

Each flag 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	set	
TxRqst	MO-Control 1	set	
MsgLst/CPUUpd	MO-Control 1	reset	
NewDat	MO-Control 1	don’t care	should be set
Dir	MO-Configuration	0	1

Table 13: Bit combinations to start transmissions

NOTES:

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

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 flags 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 flag will be set to "1" and no data frame will be sent in response.

Message Object Priority

If multiple Message Objects are waiting to transmit, the CC770 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.

4.17.4 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	rw	rw	rw	rw	rw	rw	rw

Arbitration 1 (Base Address + 3):

7	6	5	4	3	2	1	0
ID20	ID19	ID18	ID17	ID16	ID15	ID14	ID13
rw	rw	rw	rw	rw	rw	rw	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 values of the Arbitration registers are not affected by a hardware reset.

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 CC770 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.

4.17.5 Configuration Register

Configuration (Base Address + 6):

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

The value of the Configuration register is not affected by a hardware reset.

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.

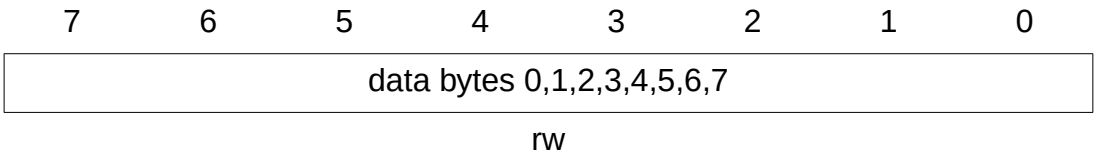
If the Message Configuration Register bit Xtd is "0" to specify a standard frame, the CC770 will reset the extended bits in the Arbitration Registers (arbitration bits 0-17) to "0" whenever a data frame is stored in this message object.

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 only).

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.

4.17.6 Data Bytes

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



The values of the data byte 0-7 registers are not affected by a hardware reset.

When the CC770 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.

4.18 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 and the TxIE flags have been hardwired inactive and the CPUUpd flag 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 flags 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 flag followed by a reset of the NewDat and RmtPnd flags 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. Port Registers

5.1 Port 1 Registers

P1CONF (9FH)

7	6	5	4	3	2	1	0
P1CONF 0-7							
rw							

The default value of the P1CONF register after a hardware reset is 00H.

P1CONF 0-7 Port 1 Input/Output Configuration bits

one Port pin configured as a push-pull output.

zero Port pin configured as a high-impedance input.

P1IN (BFH)

7	6	5	4	3	2	1	0
P1IN 0-7							
rw							

The default value of the P1IN register after a hardware reset is FFH.

P1IN 0-7 Port 1 Data In

one A one (high voltage) is read from the pin.

zero A zero (low voltage) is read from the pin.

P1OUT (DFH)

7	6	5	4	3	2	1	0
P1OUT 0-7							
rw							

The default value of the P1OUT register after a hardware reset is 00H.

P1OUT 0-7 Port 1 Data Out

one A logical one (high voltage) is written to the pin.

zero A logical zero (low voltage) is written to the pin.

5.2 Port 2 Registers

P2CONF (AFH)

7	6	5	4	3	2	1	0
P2CONF 0-7							
rw							

The default value of the P2CONF register after a hardware reset is 00H.

P2CONF 0-7 Port 2 Input/Output Configuration bits

one Port pin configured as a push-pull output.

zero Port pin configured as a high-impedance input.

P2IN (CFH)

7	6	5	4	3	2	1	0
P2IN 0-7							
rw							

The default value of the P2IN register after a hardware reset is FFH.

P2IN 0-7 Port 2 Data In

one A one (high voltage) is read from the pin.

zero A zero (low voltage) is read from the pin.

P2OUT (EFH)

7	6	5	4	3	2	1	0
P2OUT 0-7							
rw							

The default value of the P2OUT register after a hardware reset is 00H.

P2OUT 0-7 Port 2 Data Out

one A logical one (high voltage) is written to the pin.

zero A logical zero (low voltage) is written to the pin.

6. FLOW DIAGRAMS

The following flowcharts describe the operation of the CC770 and suggested flows for the host-CPU.

6.1 CC770 handling of Message Objects 1-14 (Transmit)

These are the operations the CC770 executes to transmit Message Objects. This diagram is useful to identify when the CC770 sets bits in the Control Registers.

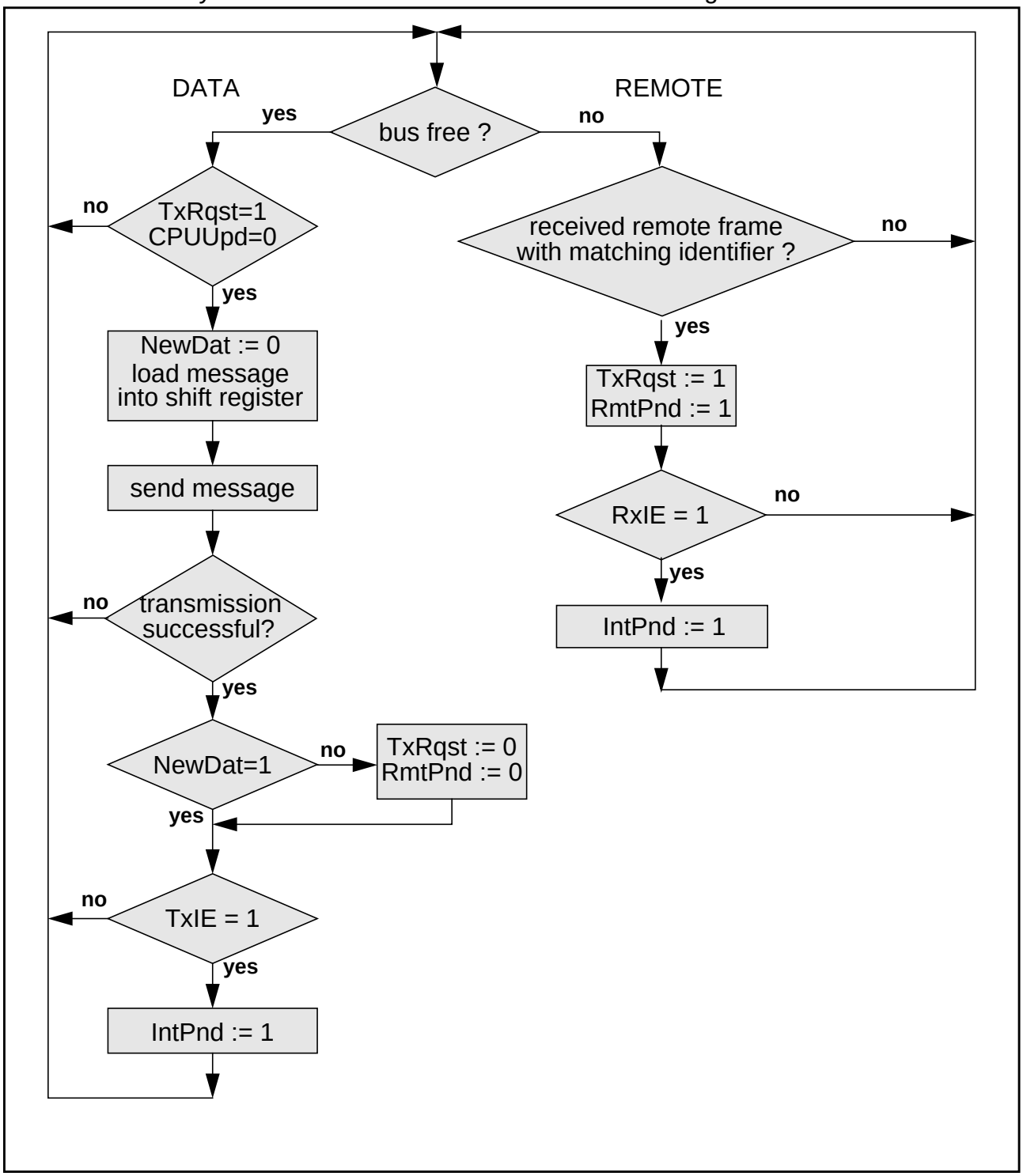


Figure 5: CC770 handling of Message Objects 1-14 (Transmit)

spec_flow_diagrams.fm K8/EIS - Klose-2969 061.2/2.3 - 15.08.97

6.2 CC770 handling of Message Objects 1-14 (Receive)

These are the operations the CC770 executes to receive Message Objects. This diagram is useful to identify when the CC770 sets bits in the Control Registers.

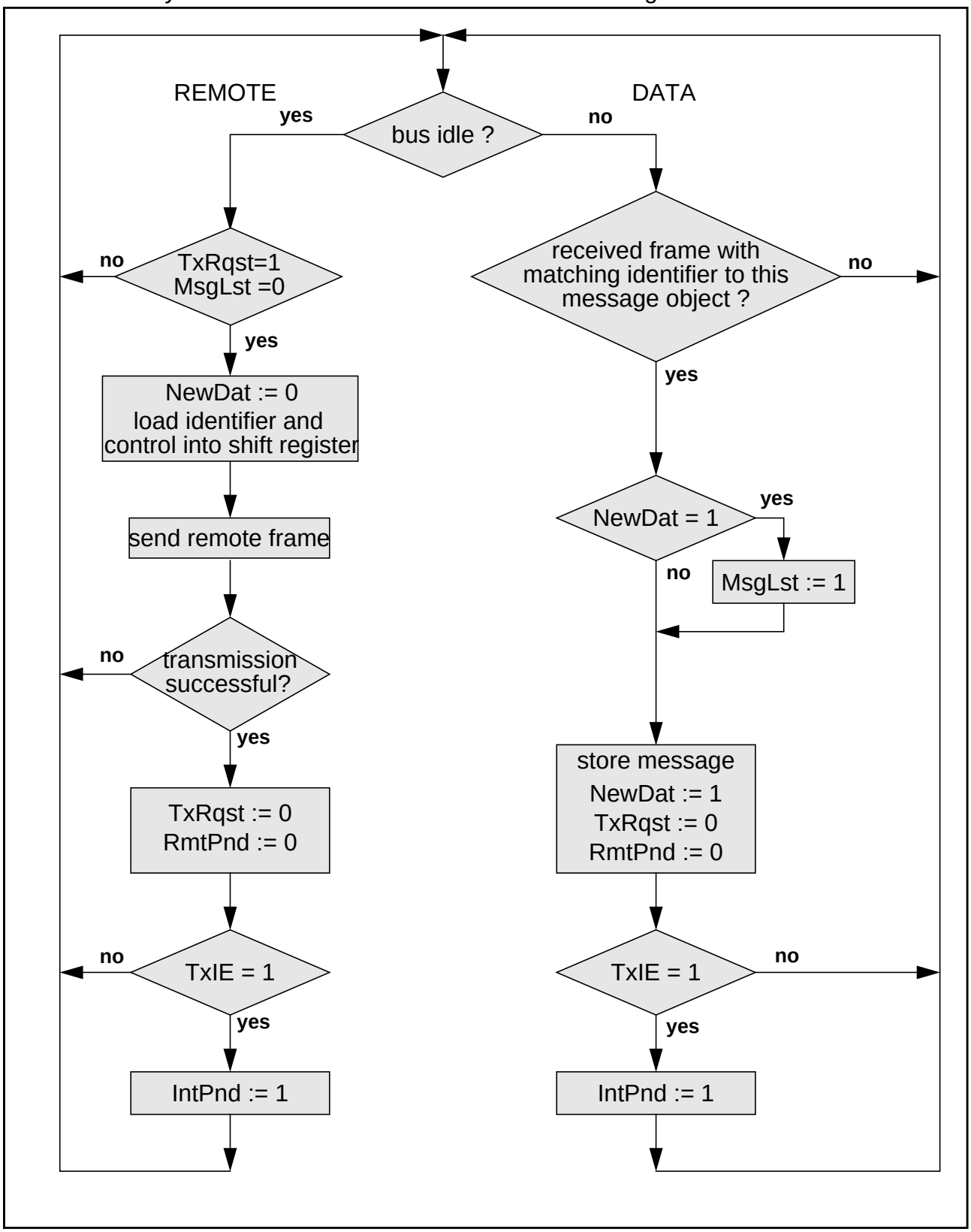


Figure 6: CC770 handling of Message Objects 1-14 (Direction = Receive)

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

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

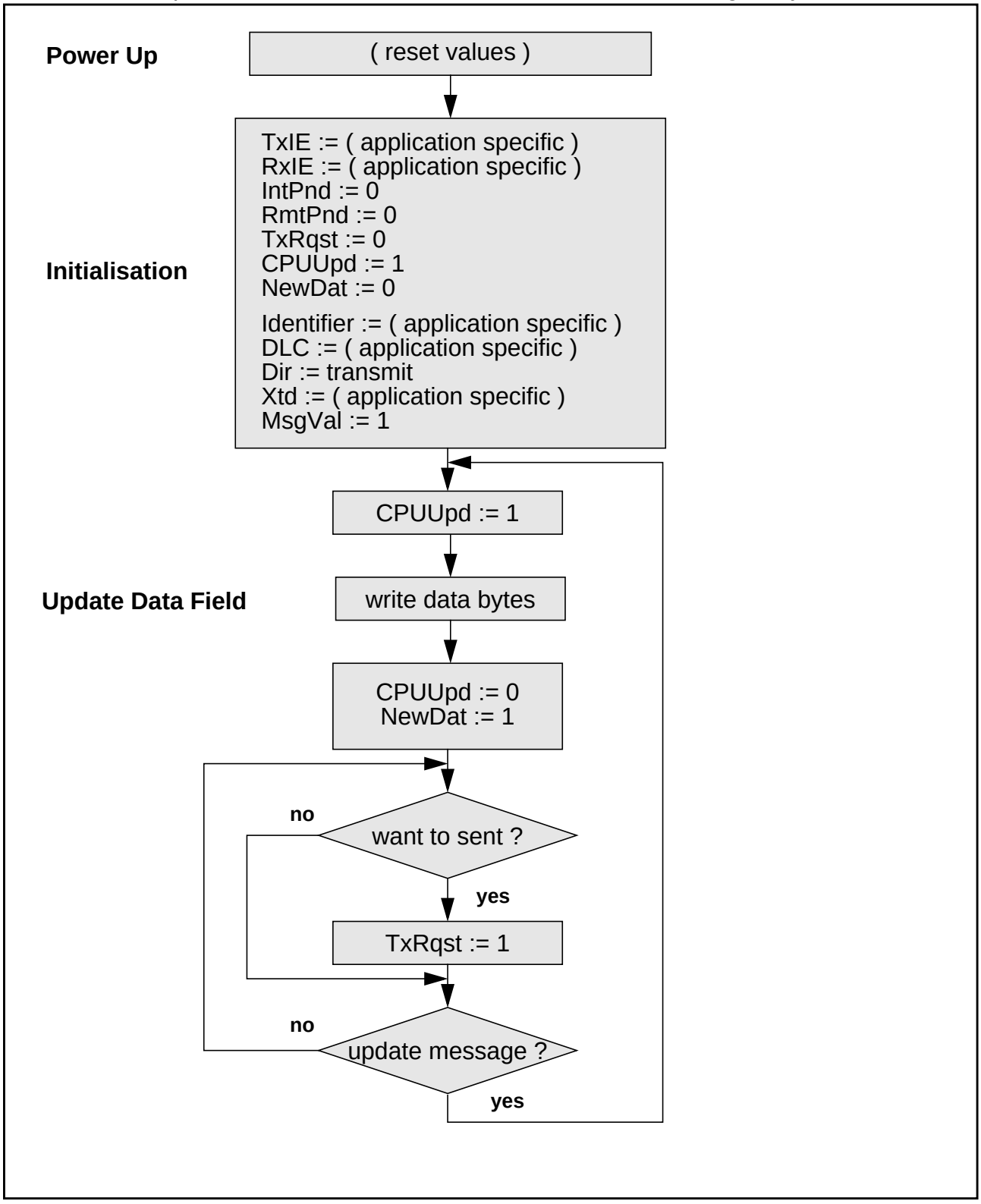


Table 14: CPU Handling of Message Objects 1-14 (Transmit)

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

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

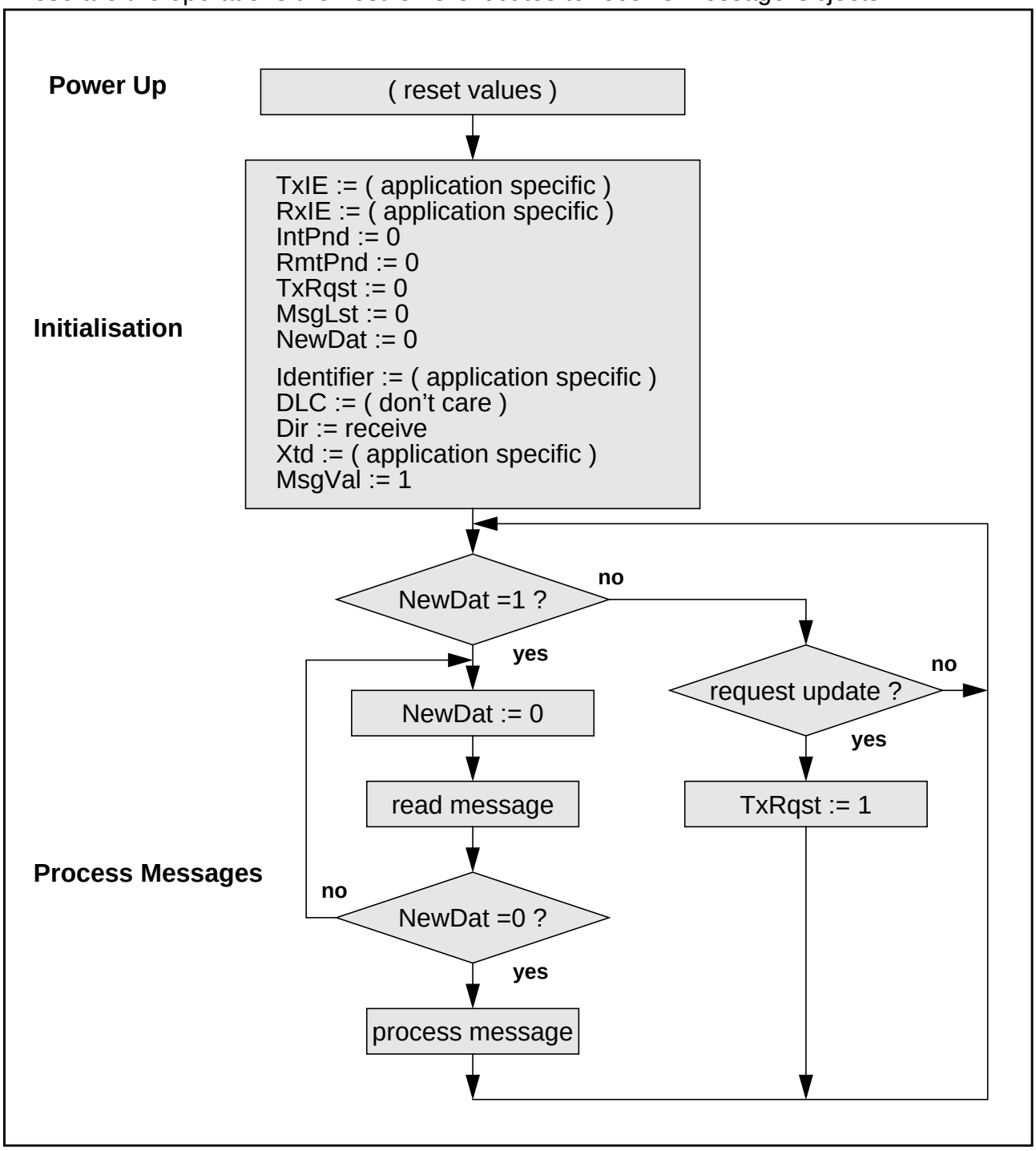


Table 15: CPU Handling of Message Objects 1-14 (Receive)

6.5 CPU Handling of Message Object 15 (Receive)

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

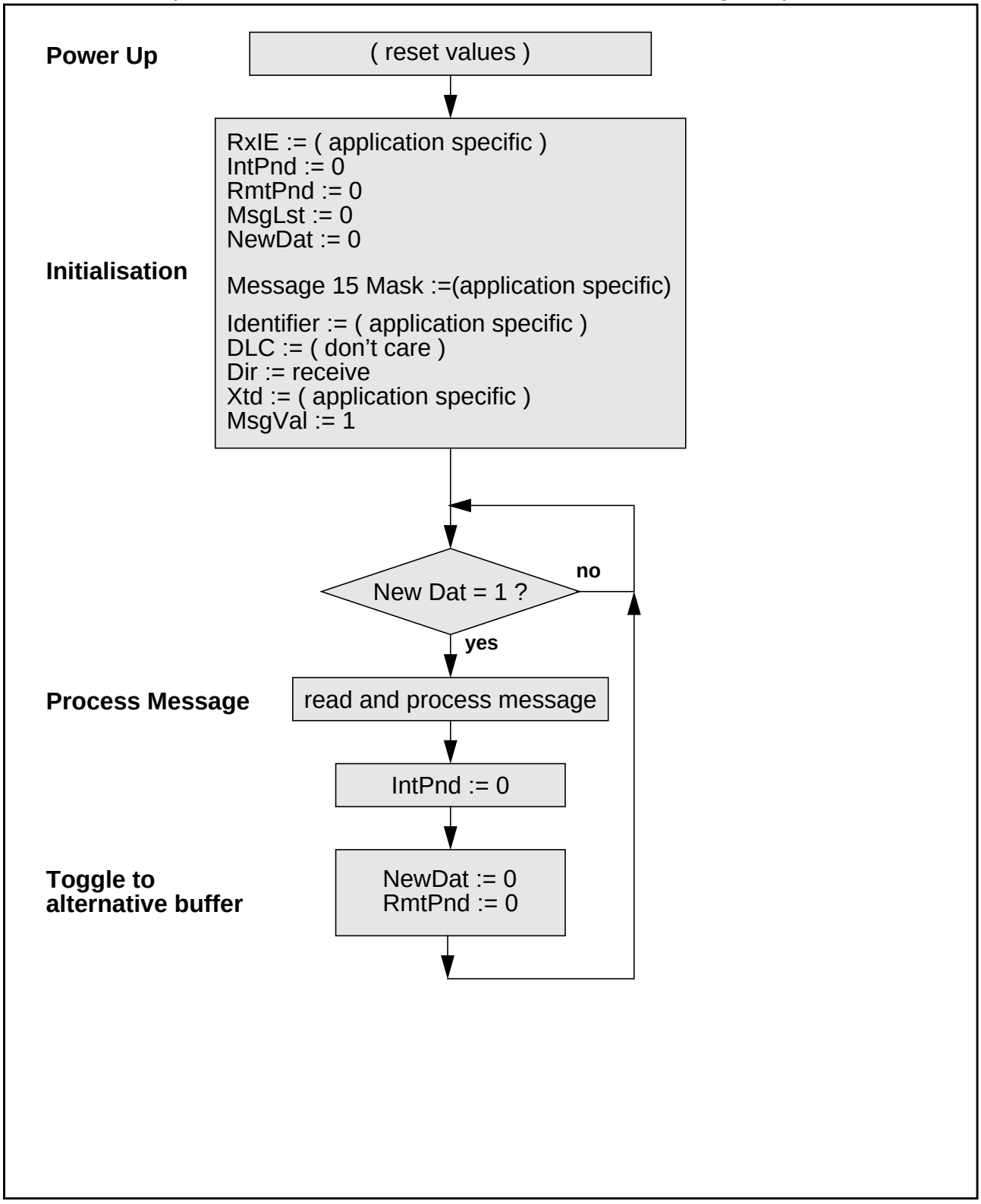


Figure 7: CPU Handling of Message Object 15 (Receive)

7. CPU Interface Logic

The CIL (CPU Interface Logic) is a flexible interface between the CPU and the CC770 RAM. The CIL allows a direct serial interface or parallel interface connection to the CC770 for most commonly used CPUs. Therefore it converts serial or parallel address/control/data signals from the CPU into parallel read and write accesses to the internal memory bus.

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.

7.1 CPU Interface Description

There are five CPU interface modes used to interface a CPU to the CC770. These include four parallel interface modes (mode 0 to mode 3) and one serial interface mode (SPI).

While RESET# is active, the two mode pins (MODE0, MODE1) select one of the following interface modes:

Mode1	Mode0	Interface Mode
0	0	Mode 0: 8-bit multiplexed Intel architecture If both WR# and RD# are driven low during reset, the SPI mode is selected instead of Mode 0.
0	1	Mode 1: 16-bit multiplexed Intel architecture
1	0	Mode 2: 8-bit multiplexed Motorola architecture
1	1	Mode 3: 8-bit non-multiplexed Motorola architecture

Table 16: CPU interface modes

Note:

The MODE0 and MODE1 inputs are weakly pulled low while RESET# is active, but they have high impedance after reset. Therefore they must be driven to a valid logical value after reset.

7.2 Parallel Interfacing Techniques

Mode 0

Mode 0 is intended to interface to Intel architectures (ALE, RD#, WR#) using an 8-bit multiplexed address/data bus. A READY output is provided to force wait states in the CPU.

Mode 1

Mode 1 is intended to interface to Intel architectures (ALE, RD#, WR#) using a 16-bit multiplexed address/data bus. A READY output is provided to force wait states in the CPU.

Mode 2

Mode 2 is intended to interface to Motorola architectures (AS, E, R/W#) using an 8-bit multiplexed address/data bus.

Mode 3

Mode 3 is intended to interface to Motorola architectures using an 8-bit non-multiplexed address/data bus. The asynchronous mode uses R/W#, CS#, and DSACK0# (E=1). The synchronous mode uses R/W#, CS#, and E. Mode 3 uses the address/data bus as the address bus and Port 1 as the data bus.

For CPUs which do not provide a READY or DSACK0# input and do not meet the address/ data bus timing restrictions, a double read mechanism must be used. When writing to the CC770 the programmer must ensure that the time between two consecutive write accesses is not less than two memory clock (MCLK) cycles. When reading the CC770, a double read is programmed. The first read will be to the message object memory address and the second read will be to the High Speed Read register (04H, 05H). After the first CPU read access, the CC770 stores the data contents to the High Speed Read register for the second read.

7.3 Serial Interface Techniques

The serial interface on the CC770 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.

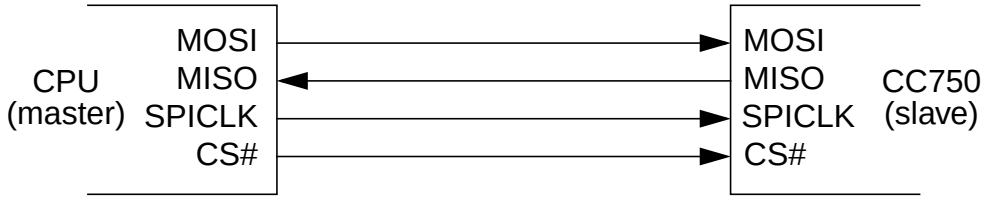


Figure 8: Interconnection for serial communication

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 (CC770). 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 (CC770) 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)

An asserted state on the slave select input (CS#) enables the CC770 to accept data on the MOSI pin. The CS# must not toggle during data transfer.

The CC770 will only drive data to the serial data register if CS# is at asserted state.

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.

7.4 Serial Interface Protocol

The general format of the data exchange from the CC770 to the master is a bit-for-bit exchange on each SPICLK clock pulse. 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 CC770-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 CC770 special function register or the CC770 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 CC770. These first two bytes are followed by the data bytes (1 to 15).

To ensure the CC770 device is not out of synchronization, the CC770 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 CC770 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.

AD0 (ICP) Idle Clock Polarity

one SCLK is idle high.

zero SCLK is idle low.

AD1 (CP) Clock Phase

one Data sampled on the falling edge of SCLK (ICP = 0) or data is sampled on the rising edge of SCLK (ICP = 1).

zero Data is sampled on the rising edge of SCLK (ICP = 0) or data is sampled on the falling edge of SCLK (ICP = 1).

AD2 (CSAS) Chip select active state

one Asserted state of CS# is logic high.

zero Asserted state of CS# is logic low.

AD3 (STE) Synchronization Transmission Enable

one The first two bytes which will be sent to the CPU after CS# is asserted are AAH and 55H.

zero The first two bytes which will be sent to the CPU after CS# is asserted are 00H and 00H.

Enables the transmission of the synchronization bytes while the Address and Control Bytes are transferred.

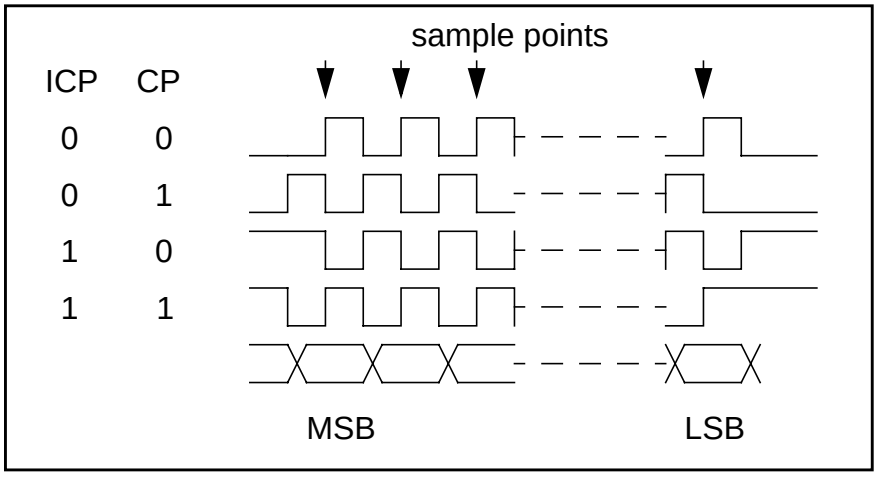


Figure 9: Serial data communication

7.5 Serial Control Byte

The Serial Control Byte is transmitted by the CPU to the CC770 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 CC770 will transfer information to the CPU.

one The data bytes will be sent from the CPU to the CC770.

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 CC770 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 CC770 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 CC770 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.

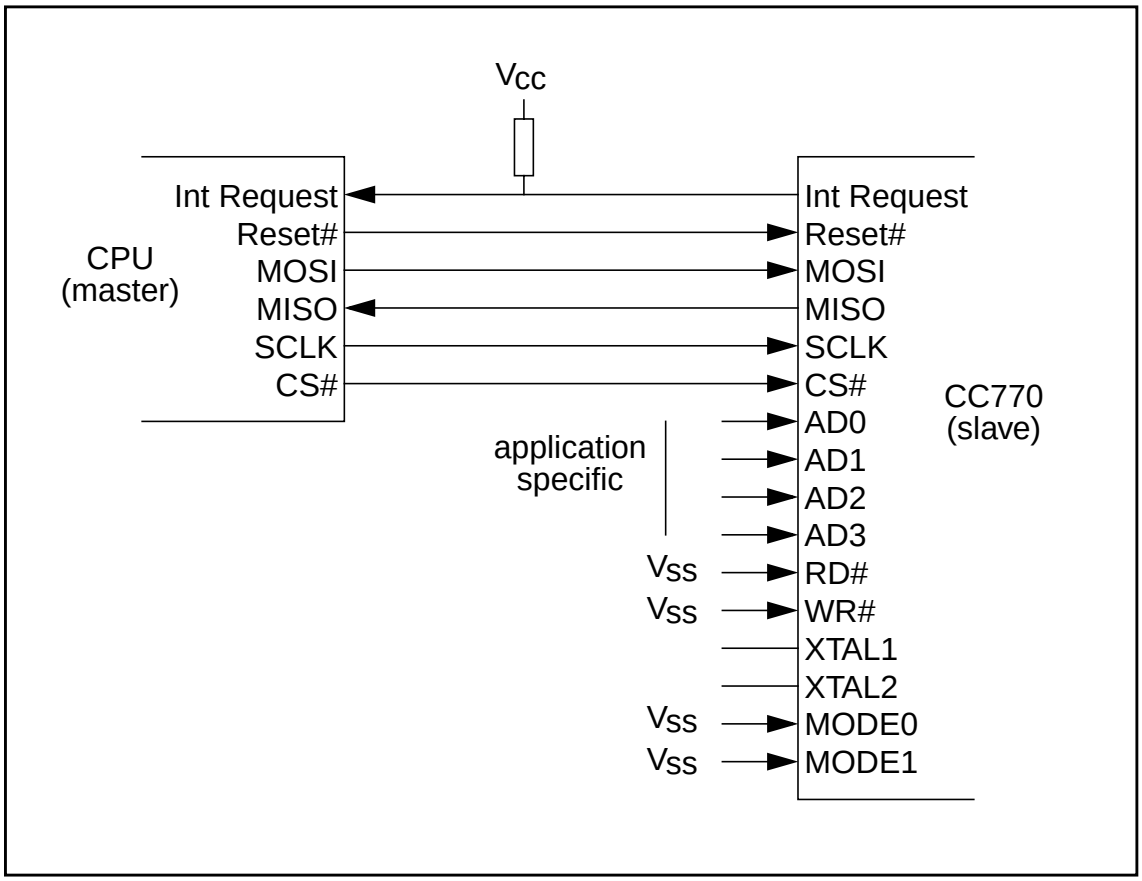


Figure 10: CC770 SPI Interface Schematic

The MODE0 and MODE1 pins may be left open since they are weakly pulled low during reset.

8. Electrical Specification

8.1 Handling Instructions

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

8.2 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/µs	C
Maximum Supply-Voltage (V_{CC}-V_{SS})	- 0.5 V to +7.0 V	C
Maximum current at all inputs/outputs	± 25 mA	C
Protection of inputs/outputs against ESD (HBM)	± 800V (1.5 kΩ,100 pF)	C
Storage temperature	-40°C to +150°C	C

Table 17: Absolute Maximum Ratings

Note:

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

***Category:**

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

8.3 DC-Characteristics

Conditions: V_{CC} = 5V ±10%, T_A = -40°C to +125°C

	Parameter	Min	Max	Unit	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#	3.0 200	V_{CC}+0.5	V mV		B A
V_{OL}	Output Low Voltage		0.45	V	I_{OL} = 1.6 mA	A

Table 18: DC-Characteristics

	Parameter	Min	Max	Unit	Conditions	Cat
V_{OH}	Output High Voltage	V_{CC}-0.8		V	I_{OH} = -200 µA	A
I_{LK}	Input Leakage Current		±1	µA	$V_{SS}<V_{IN}<V_{CC}$	A
C_{IN}	Pin Capacitance(1)		10	pF	f_{XTAL} = 1 kHz	C
I_{CC}	Supply Current(2)		50	mA		A
I_{SLEEP}	Sleep Current, no Load(2)		100	µA		A
I_{PD}	Power down Current(2)		25	µA	XTAL1 clocked	A

Table 18: DC-Characteristics

(1) Typical value based on characterization data.

(2) All pins are driven to V_{SS} or V_{CC}, V_{CC} = 5V, f_{XTAL} = 16 MHz, f_{MCLK} = 8 MHz

8.4 CLOCKOUT Specification

Parameter	Min	Max	Conditions	Cat
CLOCKOUT Frequency	XTAL/15	XTAL	load = 50 pF	B

Table 19: CLOCKOUT Specification

8.5 A.C. Characteristics

8.5.1 AC-Characteristics for 8/16-Bit Multiplexed Intel Modes (Modes 0, 1)

Conditions: V_{CC} = 5V ±10%, V_{SS} = 0V, T_A = -40 to +125°C, CL = 100 pF

Symbol	Parameter	Min	Max	Conditions	Cat
$1/t_{XTAL}$	Oscillator Frequency (1)	8 MHz	20 MHz		B
$1/t_{SCLK}$	System Clock Frequency	4 MHz	10 MHz		B
$1/t_{MCLK}$	Memory Clock Frequency	2 MHz	8 MHz		B
t_{AVLL}	Address Valid to ALE Low	7.5 ns			B
t_{LLAX}	Address Hold after ALE Low	10 ns			B
t_{LHLL}	ALE High Time	30 ns			B
t_{LLRL}	ALE Low to RD# Low	20 ns			B
t_{CLLL}	CS# Low to ALE Low	10 ns			B
t_{QVWH}	Data Setup to WR# High	27 ns			B

Table 20: A.C. Characteristics for 8/16-Bit Multiplexed Intel Modes (Modes 0, 1)

Symbol	Parameter	Min	Max	Conditions	Cat
t_{WHQX}	Input Data Hold after WR# high	10 ns			B
t_{WLWH}	WR# Pulse Width	30 ns			B
t_{WHLH}	WR# High to next ALE High	8 ns			B
t_{WHCH}	WR# High to CS# High	0 ns			B
t_{RLRH}	RD# Pulse Width This time is long enough to initiate a double read cycle by loading the High Speed Registers (04H, 05H), but is too short to READ from 04H and 05H (See t_{RLDV})	40 ns			B
t_{RLDV}	RD# Low to Data Valid (only for registers 02H, 04H, 05H)	0 ns	55 ns		B
t_{RLDV1}	RD# Low Data to Data Valid (for registers except 02H, 04H, 05H) Read Cycle without previous write (2) Read Cycle with previous write (2)		 $1.5t_{MCLK}$+100ns $3.5t_{MCLK}$+100ns		B
t_{RHDZ}	Data Float after RD# High	0 ns	45 ns		B
t_{CLYV}	CS# Low to READY Setup Condition: Load Cap. on the READY output: 50 pF		32 ns 40 ns	V_{OL}=1.0V V_{OL}=0.45V	B
t_{WLYZ}	WR# Low to READY Float for a write cycle if no previous write is pending (3)		145 ns		B
t_{WHYZ}	End of Last Write to READY Float for a write cycle if a previous write cycle is active (3)		$2t_{MCLK}$ +100ns		B
t_{RLYZ}	RD# Low to READY Float (for registers except 02H, 04H, 05H) read cycle without previous write (2) read cycle with a previous write (2)		 $2t_{MCLK}$+100ns $4t_{MCLK}$+100ns		B
t_{WHDV}	WR# High to Output Data Valid on port 1/2	t_{MCLK}	$2t_{MCLK}$+500ns		B
t_{COPD}	CLKOUT Period	$(CD_V + 1) * t_{OSC}$ (4)			C
t_{CHCL}	CLKOUT High Period	$(CD_V+1)*0.5*t_{OSC}$-10ns	$(CD_V+1)*0.5*t_{OSC}$+15ns		C

Table 20: A.C. Characteristics for 8/16-Bit Multiplexed Intel Modes (Modes 0, 1)

NOTES:

References to WR# also pertain to WRH# (write high byte) .

1. The XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled, see chapter 4.4.

2. Definition of “read cycle without a previous write”:

   The time between the rising edge of WR#/WRH# (for the previous write cycle) and the falling edge of RD# (for the current read cycle) is greater than 2 t_{MCLK} .

3. Definition of “write cycle with a previous write”:

   The time between the rising edge of WR#/WRH# (for the previous write cycle) and the rising edge of WR#/WRH# (for the current write cycle) is less than 2 t_{MCLK} .

4. Definition of CD_V is the value loaded in the CLKOUT register representing the CLKOUT divisor.

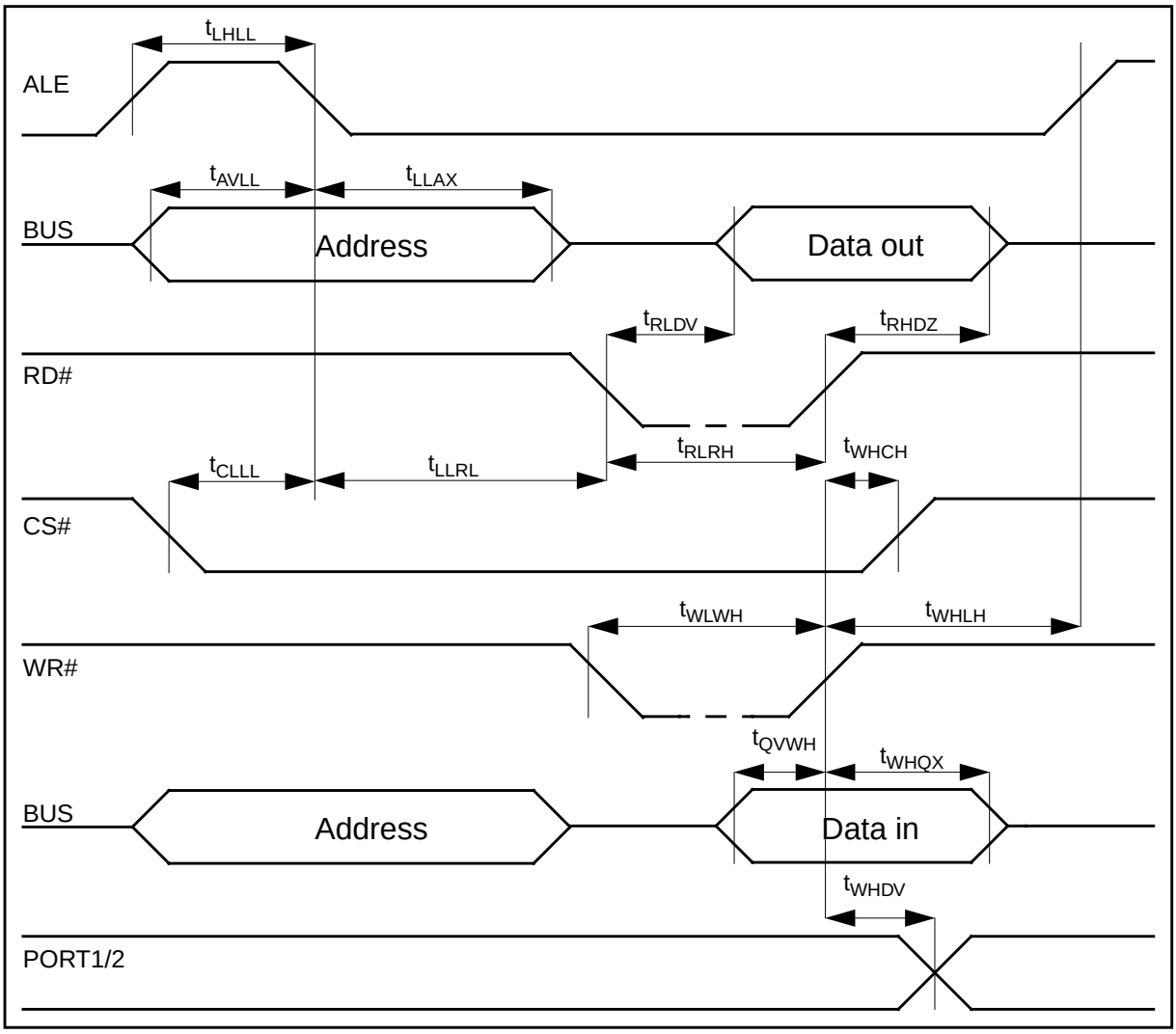


Figure 11: Timing for 8/16-Bit Multiplexed Intel Modes (Modes 0, 1)

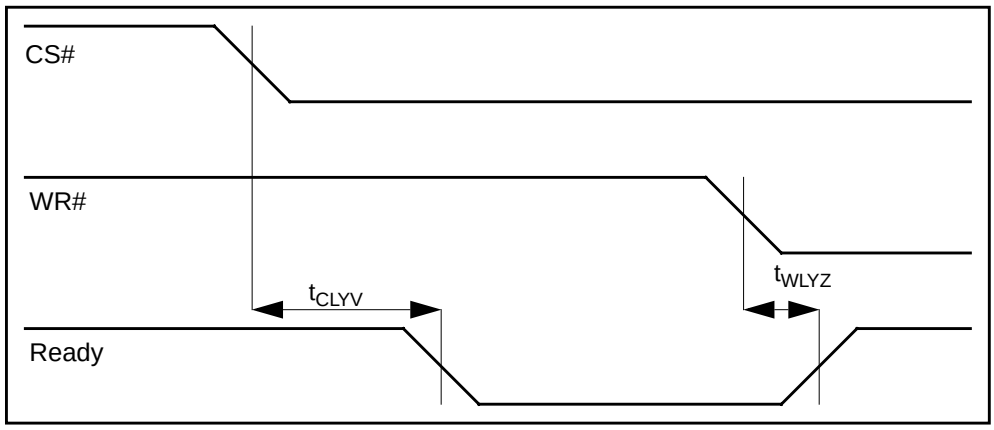


Figure 12: Ready Output Timing (write cycle, no previous write is pending)

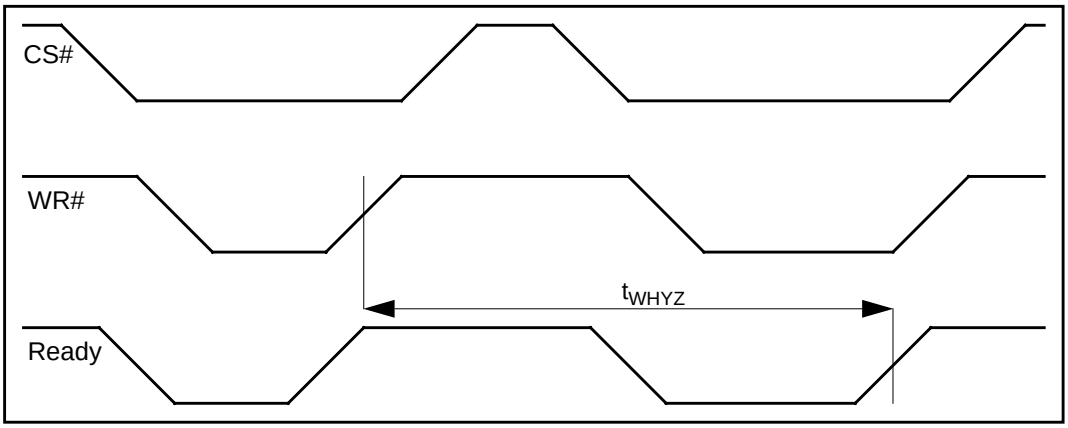


Figure 13: Ready Output Timing (write cycle, previous write cycle is active)

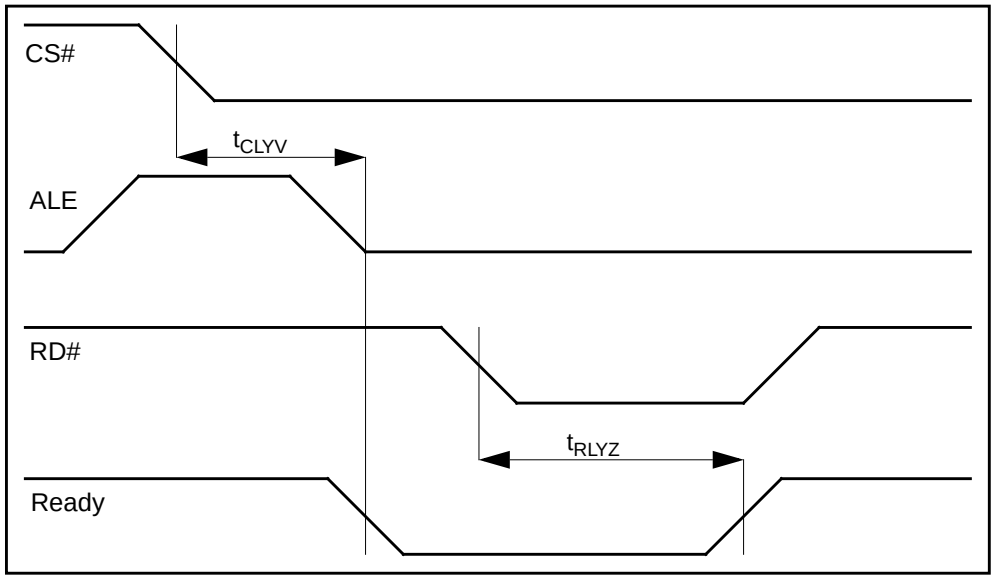


Figure 14: Ready Output Timing for a Read Cycle

8.5.2 A.C. Characteristics for 8-Bit Multiplexed Motorola Mode (Mode 2)

Conditions: V_{CC} = 5V ±10%, V_{SS} = 0V, T_A = -40 C to +125 °C, C_L = 100 pF

Symbol	Parameter	Min	Max	Cat
$1/t_{XTAL}$	Oscillator Frequency (1)	8 MHz	20 MHz	B
$1/t_{SCLK}$	System Clock Frequency	4 MHz	10 MHz	B
$1/t_{MCLK}$	Memory Clock Frequency	2 MHz	8 MHz	B
t_{AVSL}	Address Valid to AS Low	7.5 ns		B
t_{SLAX}	Address Hold after AS Low	10 ns		B
t_{ELDZ}	Data Float after E Low	0 ns	45 ns	B
t_{EHDV}	E High to Data Valid for registers 02H, 04H, 05H	0 ns	45 ns	B
	read cycle without a previous write (2) read cycle with a previous write (registers except for 02H, 04H, 05H)		$1.5t_{MCLK}$+100ns 3 t_{MCLK}+100ns	B
t_{QVEL}	Data Setup to E Low	30 ns		B
t_{ELQX}	Input Data Hold after E Low	20 ns		B
t_{ELDV}	E Low to Output Data Valid port 1/2	t_{MCLK}	$2t_{MCLK}$+500ns	B
t_{EHEL}	E High Time	45 ns		B
t_{ELEL}	End of previous write (Last E Low) to E Low for a write cycle	2 t_{MCLK}		B
t_{SHSL}	AS High Time	30 ns		B
t_{RSEH}	Setup Time of R/W# to E High	30 ns		B
t_{SLEH}	AS Low to E High	20 ns		B
t_{CLSL}	CS# Low to AS Low	20 ns		B
t_{ELCH}	E Low to CS# High	0 ns		B
t_{COPD}	CLKOUT Period	$(CD_V + 1) * t_{OSC}$ (3)		C
t_{CHCL}	CLKOUT High Period	$(CD_V + 1)*$ $0.5*t_{OSC}$-10ns	$(CD_V + 1)*$ $0.5*t_{OSC}$+15ns	C

Table 21: A.C. Characteristics for 8-Bit Multiplexed Motorola Mode (Mode 2)

NOTES:

1. The XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled, see chapter 4.4.
2. Definition of “Read Cycle without a Previous Write”:

   The time between the falling edge of E (for the previous write cycle) and the rising edge of E (for the current read cycle) is greater than 2 t_{MCLK} .

3. Definition of CD_V is the value loaded in the CLKOUT register representing the CLKOUT divisor.

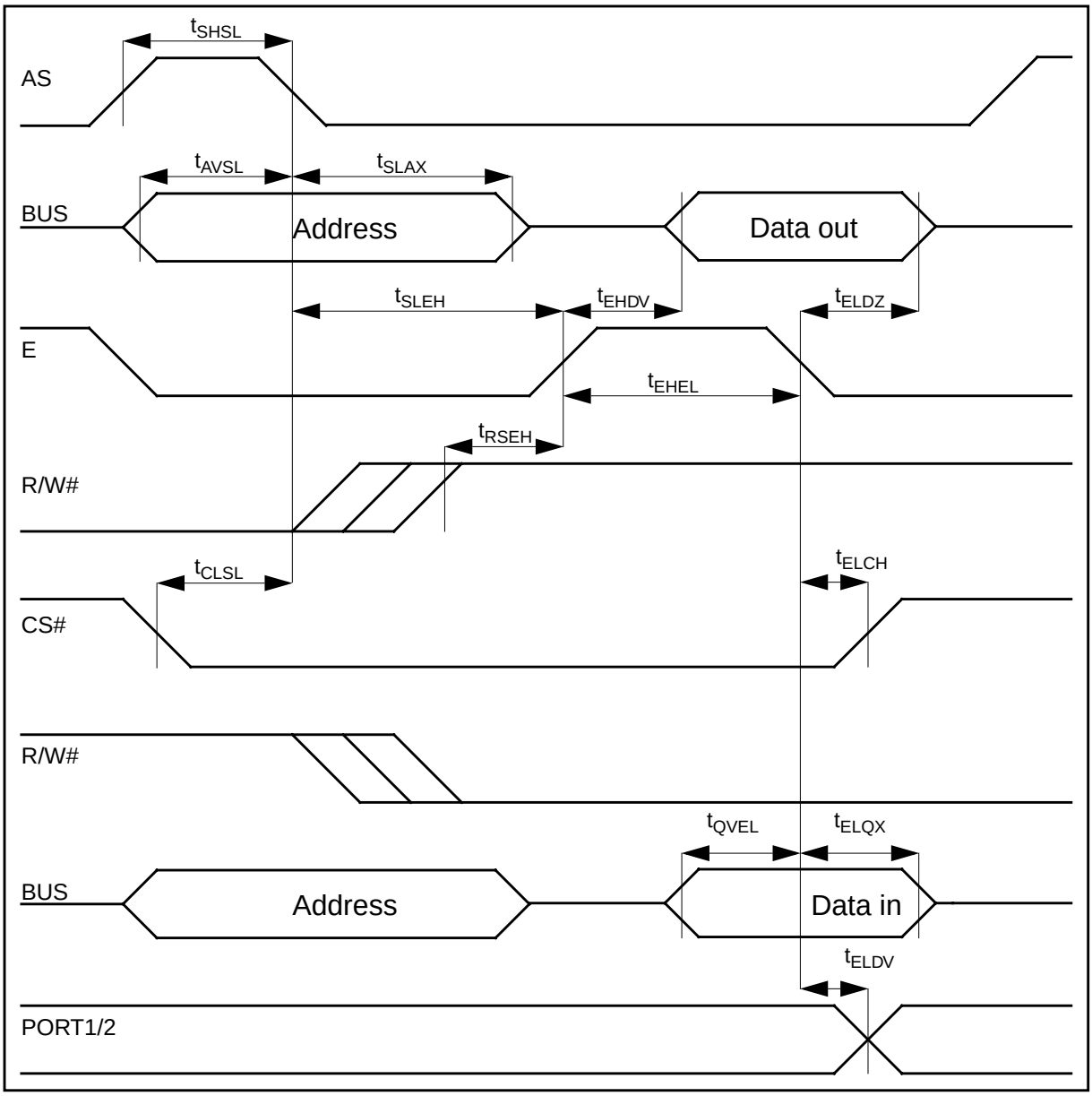


Figure 15: Timing for 8-Bit Multiplexed Motorola Mode (Mode 2)

8.5.3 A.C. Characteristics for 8-Bit Non-Multiplexed Asynchronous (Mode 3)

Conditions: V_{CC} = 5V ±10%, V_{SS} = 0V, T_A = -40 C to +125 °C, C_L = 100 pF

Symbol	Parameter	Min	Max	Cat
$1/t_{XTAL}$	Oscillator Frequency (1)	8 MHz	20 MHz	B
$1/t_{SCLK}$	System Clock Frequency	4 MHz	10 MHz	B
$1/t_{MCLK}$	Memory Clock Frequency	2 MHz	8 MHz	B
t_{AVCL}	Address or R/W# Valid to CS# Low Setup	3ns		B
t_{CLDV}	CS# Low to Data Valid for High Speed Registers (02H, 04H, 05H)	0 ns	55 ns	B
	For Low Speed Registers (Read Cycle without Previous Write) (2)	0 ns	1.5 t_{MCLK} +100 ns	B
	For Low Speed Registers (Read Cycle with Previous Write) (2)	0 ns	3.5 t_{MCLK} +100 ns	B
t_{KLDV}	DSACK0# Low to Output Data Valid for High Speed Read Register		23 ns	B
	For Low Speed Read Register	< 0 ns		B
t_{CHDV}	CC770 Input Data Hold after CS# High	15 ns		B
t_{CHDH}	CC770 Output Data Hold after CS# High	0 ns		B
t_{CHDZ}	CS# High to Output Data Float		35 ns	B
t_{CHKH1}	CS# High to DSACK0# = 2.4V (3)	0 ns	55 ns	B
t_{CHKH2}	CS# High to DSACK0# = 2.8V		150 ns	B
t_{CHKZ}	CS# High to DSACK0# Float	0 ns	100 ns	B
t_{CHCL}	CS# Width between Successive Cycles	25 ns		B
t_{CHAI}	CS# High to Address Invalid	7 ns		B
t_{CHRI}	CS# High to R/W# Invalid	5 ns		B
t_{CLCH}	CS# Width Low	65 ns		B
t_{DVCH}	CPU Write Data Valid to CS# High	20 ns		B
t_{CLKL}	CS# Low to DSACK0# Low for High Speed Registers and Low Speed Registers Write Access without Previous Write (4)	0 ns	67 ns	B

Table 22: A.C. Characteristics for 8-Bit Non-Mux Asynchronous (Mode 3)

Symbol	Parameter	Min	Max	Cat
t_{CHKL}	End of previous write (CS# High) to DSACK0# Low for a write cycle with a previous write (4)	0 ns	2 t_{MCLK} + 145 ns	B
t_{COPD}	CLKOUT Period	(CD_V + 1) * t_{OSC} (5)		C
t_{CHCL}	CLKOUT High Period	(CD_V + 1)* 0.5* t_{OSC}-10ns	(CD_V + 1) * 0.5*t_{OSC}+15ns	C

Table 22: A.C. Characteristics for 8-Bit Non-Mux Asynchronous (Mode 3)

NOTES:

E and AS must be tied high in this mode.

1. The XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled, see chapter 4.4.
2. Definition of “Read Cycle without a Previous Write”:

   The time between the rising edge of CS# (for the previous write cycle) and the falling edge of CS# (for the current read cycle) is greater than 2 t_{MCLK} .
3. An on-chip pullup will drive DSACK0# to approximately 2.4V. An external pullup is required to drive this signal to a higher voltage.
4. Definition of “Write Cycle without a Previous Write”:

   The time between the rising edge of CS# (for the previous write cycle) and the rising edge of CS# (for the current write cycle) is greater than 2 t_{MCLK} .
5. Definition of CD_V is the value loaded in the CLKOUT register representing the CLKOUT divisor.

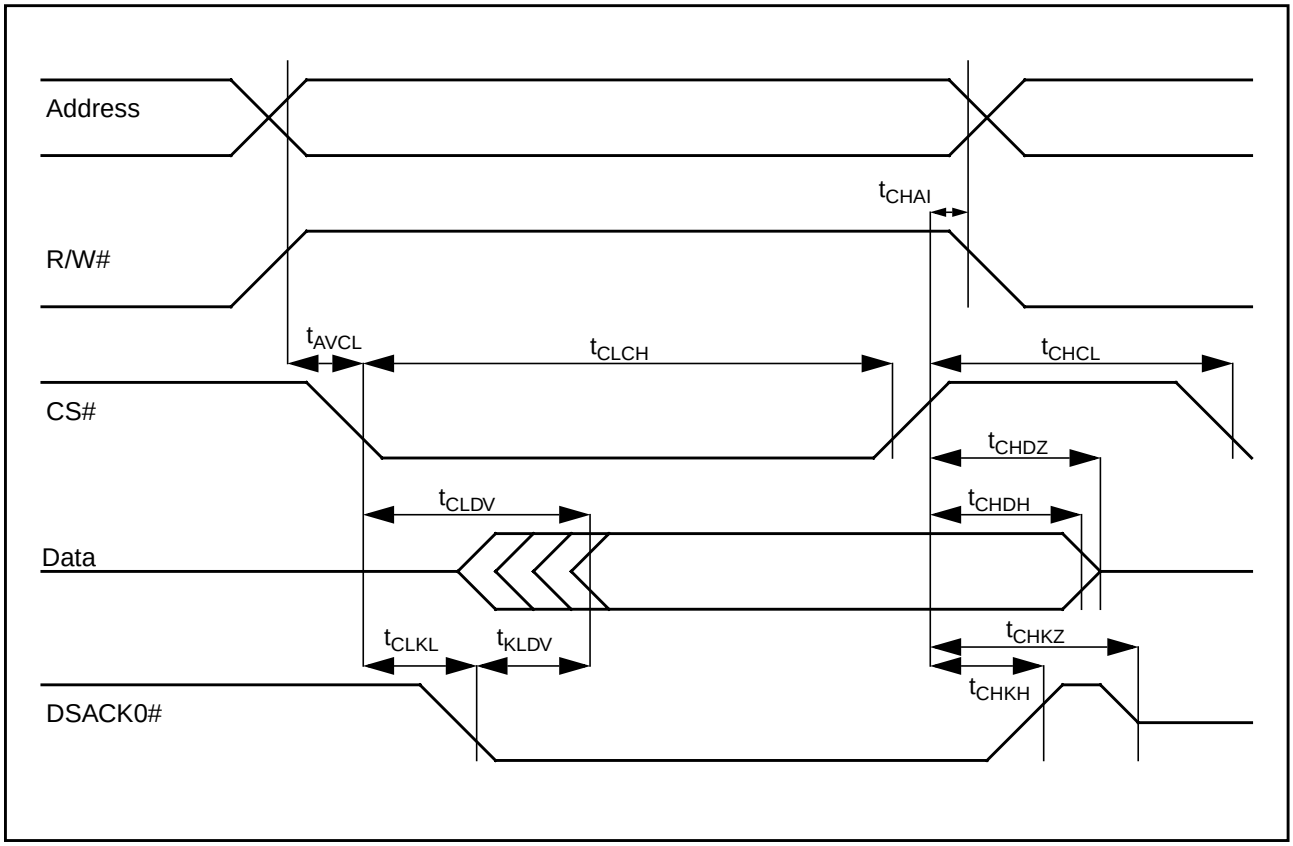


Figure 16: Timing for 8-Bit Non-Mux Asynchronous Mode (Mode 3), read cycle.

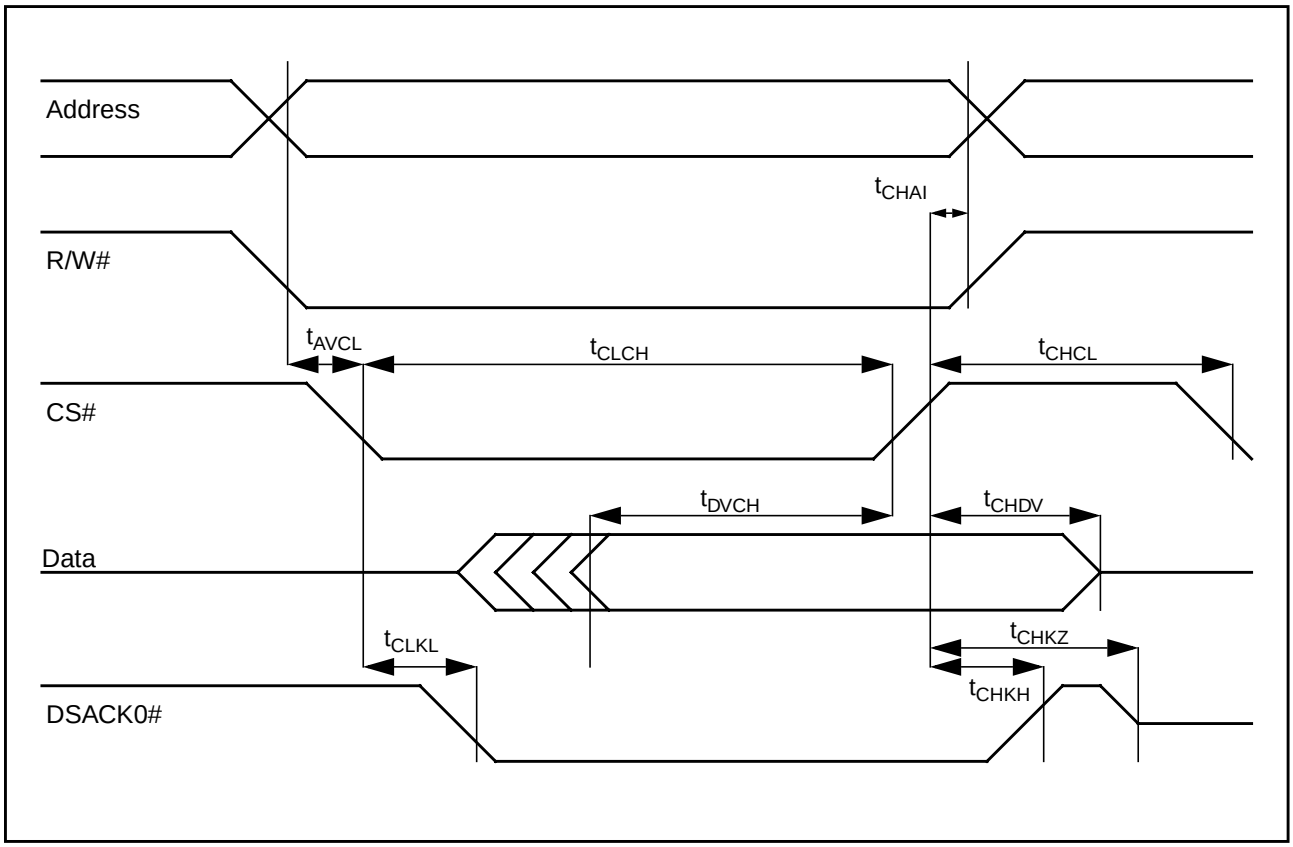


Figure 17: Timing for 8-Bit Non-Mux-Async Mode (Mode 3), write cycle.

8.5.4 A.C. Characteristics for 8-Bit Non-Multiplexed Synchronous (Mode 3)

Conditions: V_{CC} = 5V ±10%, V_{SS} = 0V, T_A = -40 C to +125 °C, C_L = 100 pF

Symbol	Parameter	Min	Max	Cat
$1/t_{XTAL}$	Oscillator Frequency (1)	8 MHz	20 MHz	B
$1/t_{SCLK}$	System Clock Frequency	4 MHz	10 MHz	B
$1/t_{MCLK}$	Memory Clock Frequency	2 MHz	8 MHz	B
t_{EHDV}	E High to Data Valid out of High Speed Register (02H, 04H, 05H)		55 ns	B
	Read Cycle without previous write for Low Speed Registers (2)		1.5 t_{MCLK} +100 ns	B
	Read Cycle with previous write for Low Speed Registers (2)		3.5 t_{MCLK} +100 ns	B
t_{ELDH}	Data Hold after E Low for a read cycle	5 ns		B
t_{ELDZ}	Data Float after E Low		35 ns	B
t_{ELDV}	Data Hold after E Low for a write cycle	15 ns		B
t_{AVEH}	Address and R/W# to E setup	25 ns		B
t_{ELAV}	Address and R/W# Valid after E falls	15 ns		B
t_{CVEH}	CS# Valid to E High	0 ns		B
t_{ELCV}	CS# Valid after E Low	0 ns		B
t_{DVEL}	Data Setup to E Low	55 ns		B
t_{EHEL}	E Active width	100 ns		B
t_{AVAV}	Start of a write cycle after a previous Write Access	2 t_{MCLK}		B
t_{AVCL}	Address or R/W# to CS# Low Setup	3 ns		B
t_{CHAI}	CS# High to Address Invalid	7 ns		B
t_{COPD}	CLKOUT Period	$(CD_V + 1) * t_{OSC}$ (3)		C
t_{CHCL}	CLKOUT High Period	$(CD_V + 1)*0.5*t_{OSC}$-10ns	$(CD_V + 1)*0.5*t_{OSC}$+15ns	C

Table 23: A.C. Characteristics for 8-Bit Non-Mux Asynchronous (Mode 3)

NOTES:

1. The XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled, see chapter 4.4.

2. Definition of “Read Cycle without a Previous Write”:

   The time between the falling edge of E (for the previous write cycle) and the rising edge of E (for the current read cycle) is greater than 2 t_{MCLK} .

3. Definition of CD_V is the value loaded in the CLKOUT register representing the CLKOUT divisor.

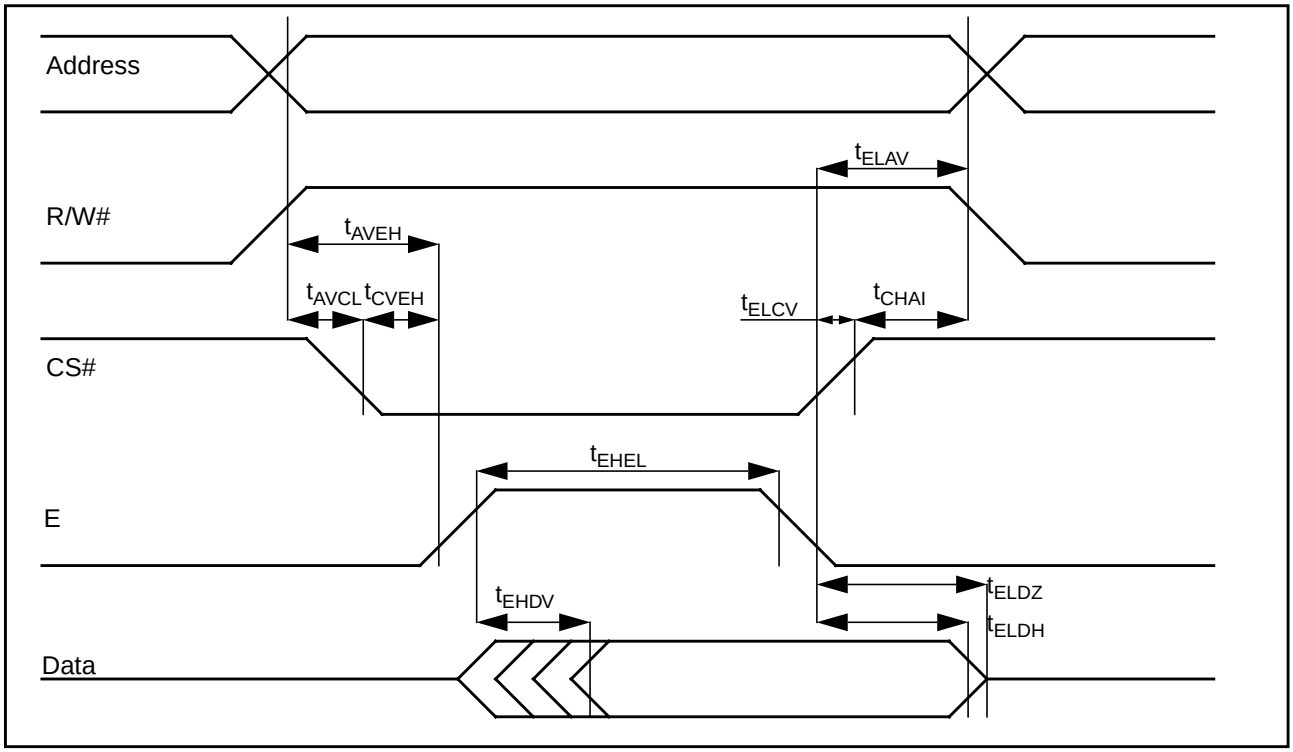


Figure 18: Timing for 8-Bit Non-Mux Synchronous Mode (Mode 3), read cycle.

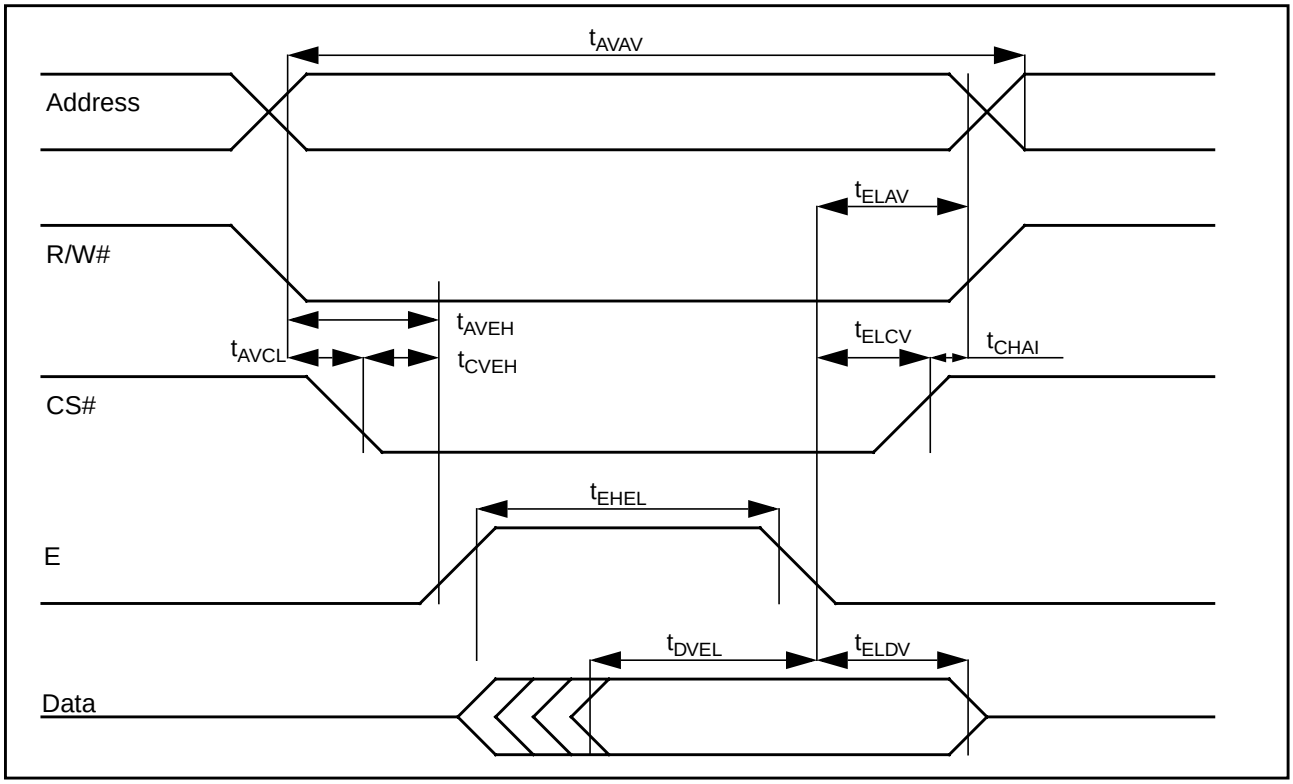


Figure 19: Timing for 8-Bit Non-Mux Synchronous Mode (Mode 3), write cycle.

8.5.5 A.C. Characteristics for Serial Interface Mode

Conditions: V_{CC} = 5V ±10%, V_{SS} = 0V, T_A = -40°C to +125°C, C_L = 100 pF

Symbol	Parameter	Min	Max	Unit	Cat
f_{XTAL}	Oscillator Frequency (1)	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}	$1/f_{SPICLK}$	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
t_{COPD}	CLKOUT Period	$(CD_V+ 1) * t_{OSC}$ (2)		ns	C
t_{CHCL}	CLKOUT High Period	$(CD_V+ 1) * 0.5*t_{OSC}-10$	$(CD_V+ 1) * 0.5*t_{OSC}+15$	ns	C

Table 24: A.C. Characteristics for Serial Interface Mode

NOTE:

1. The XTAL frequency may be lower than 8 MHz when the crystal at the XTAL pins is replaced by a clock generator and the PLL is disabled, see chapter 4.4.
2. Definition of CD_V is the value loaded in the CLKOUT register representing the CLKOUT divisor.

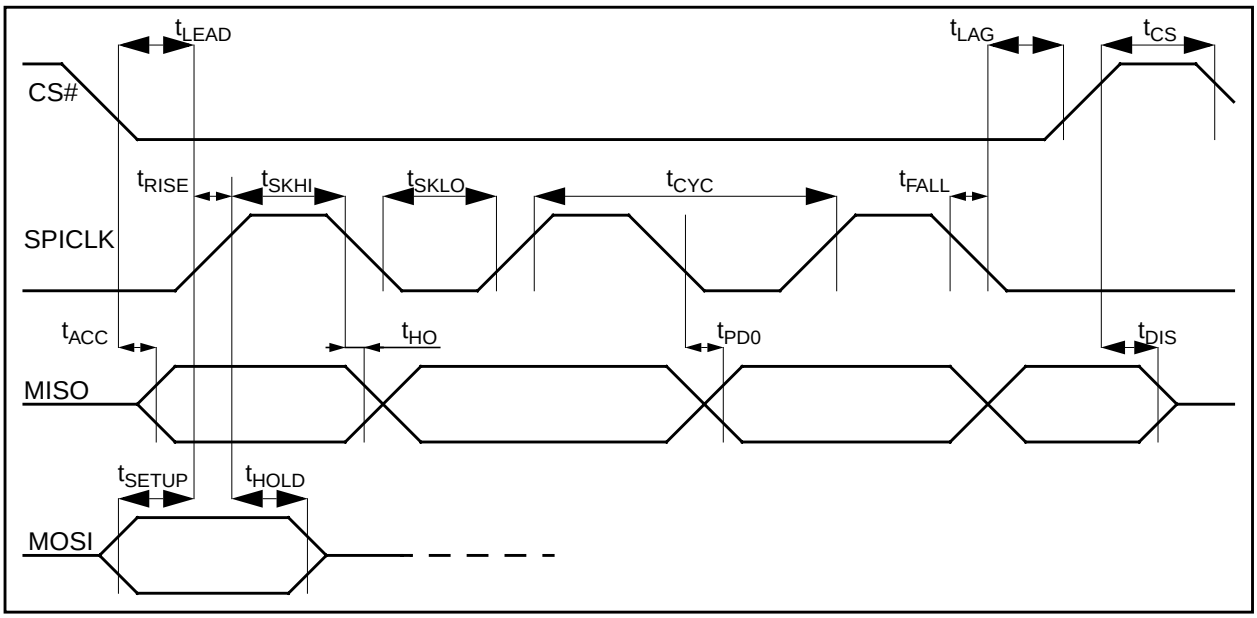


Figure 20: Timing for Serial Interface Mode (Polarity=0,Phase=0)

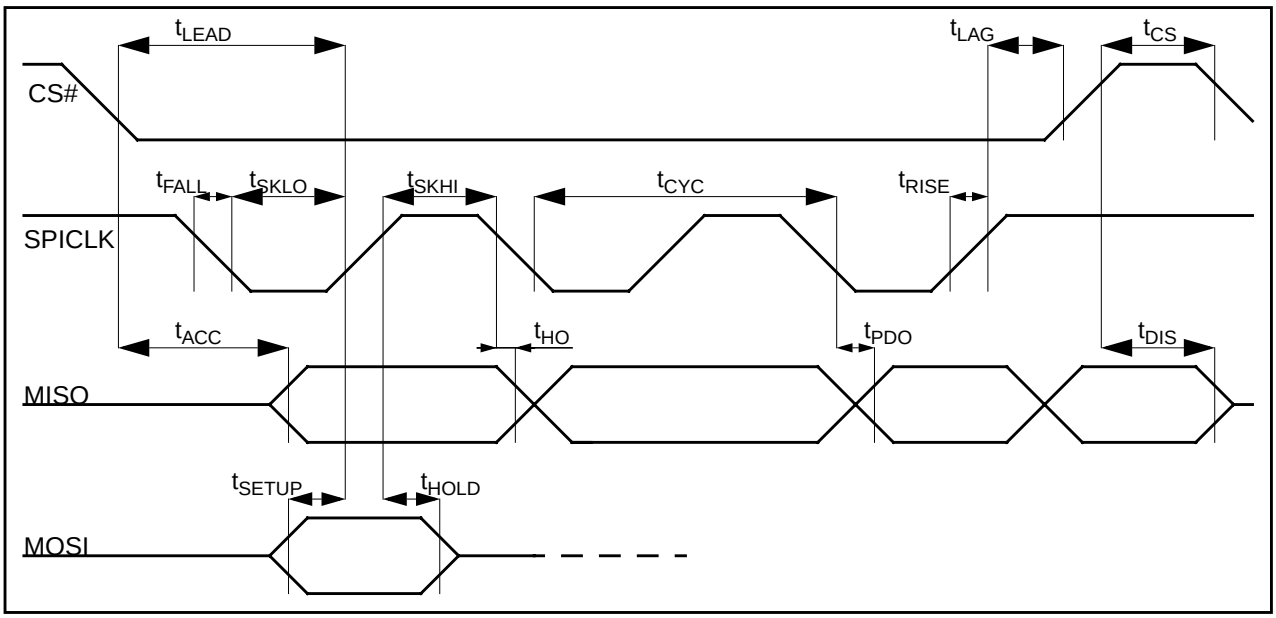


Figure 21: Timing for Serial Interface Mode (Polarity=1,Phase=1)

8.5.6 Waveforms for testing

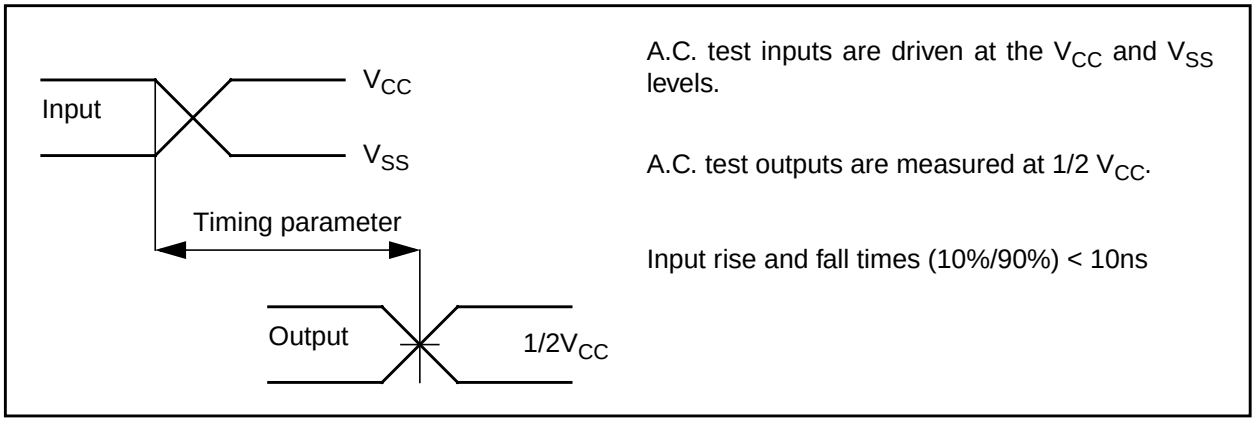


Figure 22: Input and output reference waveforms

9. Stepping specific errata

The subject of this paragraph is to describe functional and electrical divergences of the CC770D (this means CC770 with stepping D) to this Target Specification.

9.1 Identification of affected devices

9.1.0.1 PLCC44 package

The first line of the top-side marking, right beside the Bosch anchor, denotes the product number.

All devices with the product number 30480 are affected by the errata described in this chapter.

9.1.1 Chip on wafer

In the edge, located by the AD6 and AD7 pads, the product designator is readable in metal 3 layer.

All devices with the designator CC770D are affected by the errata described in this chapter.

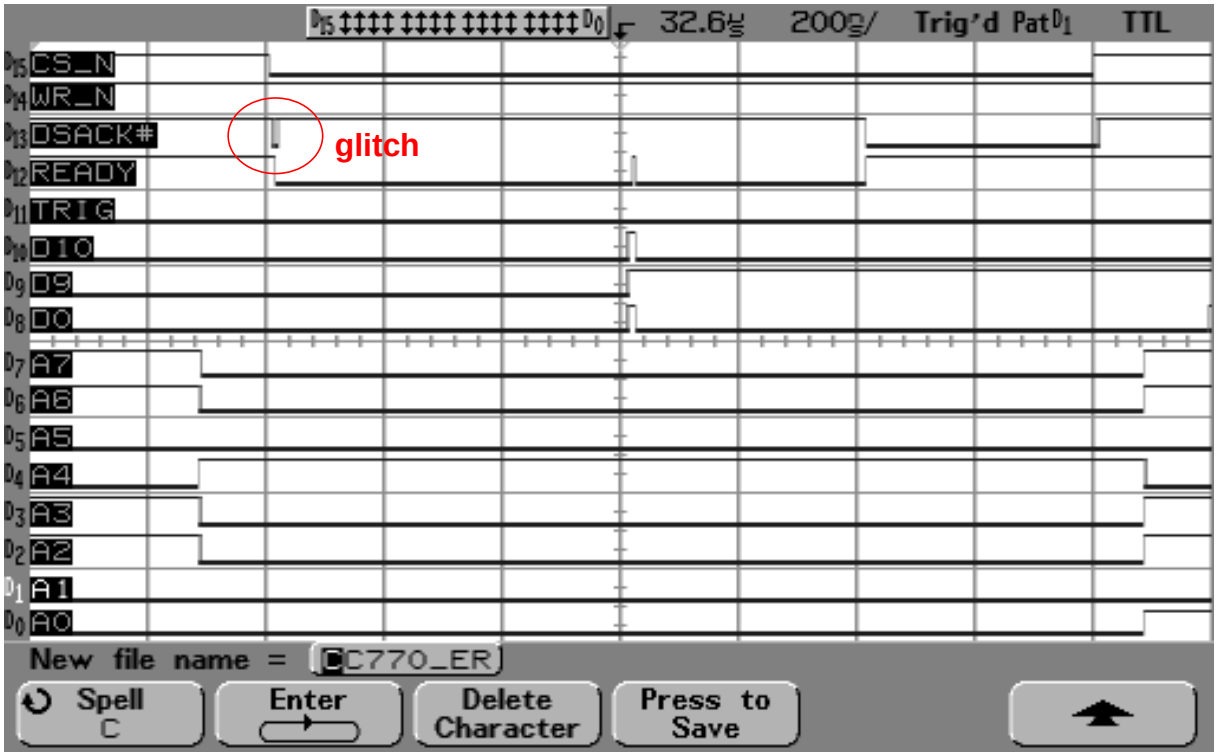
9.2 Errata

9.2.1 Glitches on DSACK0# pin

9.2.1.1 Description

This erratum is only relevant for applications which uses the 8-bit non multiplexed address data bus interface (interface mode 3) of the CC770 to connect a micro controller.

In this interface mode, DSACK0# signal marks valid data for read accesses to the CC770. When accessing the high speed read register the behaviour of this signal is correct. When accessing the low speed registers, glitches can occur on the DSACK0# pin before data gets valid on the data bus. The plot of the oscilloscope shows such a glitch of the DSACK0# signal.



9.2.1.2 Work-around

It is not necessary to use the DSACK0# signal if the data lines are evaluated after tCLDV.

If this data access timing is not supported by the micro controller, the glitch can be suppressed by a capacitor between the DSACK0# pin and digital ground.

9.2.2 Data corruption

9.2.2.1 Description

This erratum is only relevant for applications which uses the 8-bit asynchronous non multiplexed address data bus interface (interface mode 3) of the CC770 to connect a micro controller.

Invalid data may be read, depending on the actual phase between the rising edge of XTAL1 and the falling edge of CS#.

Note:

When CC770D and micro controller are supplied by different clock sources, their clock phases are not fixed, even if the different clock sources have the same frequency.

9.2.2.2 Work-around

Use other interface mode of CC770.

10. Appendix

10.1 Documentation of Changes

10.1.1 Changes on Revisions

10.1.1.1 Revision 1.0

Initial version.

10.1.1.2 Revision 1.1

Specification left preliminary state.

Protection against ESD reduced to 800 V.

Timing parameters t_{COPD} and t_{CHCL} are now categorized to C.

10.1.1.3 Revision 1.2

Package Diagram now more detailed.

Clockout pin is active while reset is active, see chapter 3.2.2.

Flare clocking limitations, see chapter 4.4.1.

Behaviour of Serial Reset address, see chapter 4.16.

Mode0/1 Pins must be connected in any configuration, see note in chapter 7.1

f_{SPICLK} must not be higher than f_{MCLK}, see chapter 8.5.5.

Notes to the design steps B and C are not supported anymore.

10.1.1.4 Revision 1.3

Add errata description for CC770 with stepping D.

Product number of package diagram modified.

EOF